ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΩΝ ΔΕΔΟΜΕΝΩΝ ΜΕ APACHE SPARK"

ΧΟΥΣΤΟΥΛΑΚΗΣ ΜΙΧΑΗΛ
Α.Μ.: 2260

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: ΤΑΜΠΑΚΑΣ ΒΑΣΙΛΕΙΟΣ

ΠΑΤΡΑ 2023

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΧΑΡΙΣΤΙΕΣ

Η εργασία αυτή είναι αφιερωμένη σε κάποια κοντινά μου πρόσωπα τα οποία σε μια δύσκολη περίοδο της ζωής μου στάθηκαν δίπλα μου, δίνοντας μου την δύναμη να

εκπληρώσω τον στόχο που είχα θέσει στον εαυτό μου πριν χρόνια. Όσος χρόνος κι αν περάσει, στο τέλος της ημέρας αυτό που έχει σημασία είναι η απόφαση.

ABSTRACT

In today's digital era, social media platforms serve as a rich source of valuable information, offering researchers new opportunities to analyze user sentiments and opinions. This thesis references the data that overwhelm modern daily life, as well as the techniques developed to manage it, giving rise to a distinct branch of computer science solely dedicated to this field.

Subsequently, an extensive analysis of the Apache Spark software follows, including its architecture and functionality, step-by-step installation, and the utilization of simple algorithms on small-scale data.
Furthermore, the capabilities of Apache Spark are presented in a more complex scenario through the method known as Sentiment Analysis on a large-scale dataset collected from the popular social networking platform Twitter.
The aim of this study is to leverage the parallel processing capabilities of Apache Spark to address the inherent challenges of processing and analyzing a large volume of data.

Keywords: Data, Big Data, Apache Hadoop, Apache Spark, Sentiment Analysis.

ΠΕΡΙΛΗΨΗ

Στη σημερινή ψηφιακή εποχή, οι πλατφόρμες κοινωνικών μέσων αποτελούν μια πλούσια πηγή πολύτιμων πληροφοριών, παρέχοντας στους ερευνητές νέες ευκαιρίες για την ανάλυση των συναισθημάτων και των απόψεων των χρηστών.

Στην παρούσα διπλωματική εργασία γίνεται αναφορά στα δεδομένα, τα οποία κατακλύζουν την σύγχρονη καθημερινότητα καθώς και οι τεχνικές που εφευρέθηκαν για την διαχείρισή τους δίνοντας το έναυσμα για την δημιουργία ενός ξεχωριστού κλάδου της επιστήμη της πληροφορικής που ασχολείται αποκλειστικά με αυτό το κομμάτι.

Έπειτα, ακολουθεί εκτενής ανάλυση του λογισμικού Apache Spark, της αρχιτεκτονικής του και του τρόπου λειτουργίας του, εγκαθιστώντας το βήμα βήμα και χρησιμοποιώντας απλούς αλγόριθμους σε δεδομένα μικρού όγκου.

Τέλος, γίνεται παρουσίαση των δυνατοτήτων του Apache Spark σε ένα πιο περίπλοκο σενάριο μέσω της μεθόδου που ονομάζεται Ανάλυση Συναισθήματος (Sentiment Analysis), σε ένα σύνολο δεδομένων μεγάλου όγκου που έχουν συλλεχθεί από την δημοφιλή πλατφόρμα κοινωνικής δικτύωσης Twitter.

Ο στόχος αυτής της μελέτης είναι να αξιοποιήσει τις δυνατότητες παράλληλης επεξεργασίας του Apache Spark για να χειριστεί τις εγγενείς προκλήσεις της επεξεργασίας και της ανάλυσης ενός μεγάλου όγκου δεδομένων.

Λέξεις Κλειδιά: Δεδομένα, Μεγάλα Δεδομένα, Apache Hadoop, Apache Spark, Ανάλυση Συναισθήματος

ΚΕΦΑΛΑΙΟ 1

1.1 ΕΙΣΑΓΩΓΗ ΣΤΑ ΔΕΔΟΜΕΝΑ

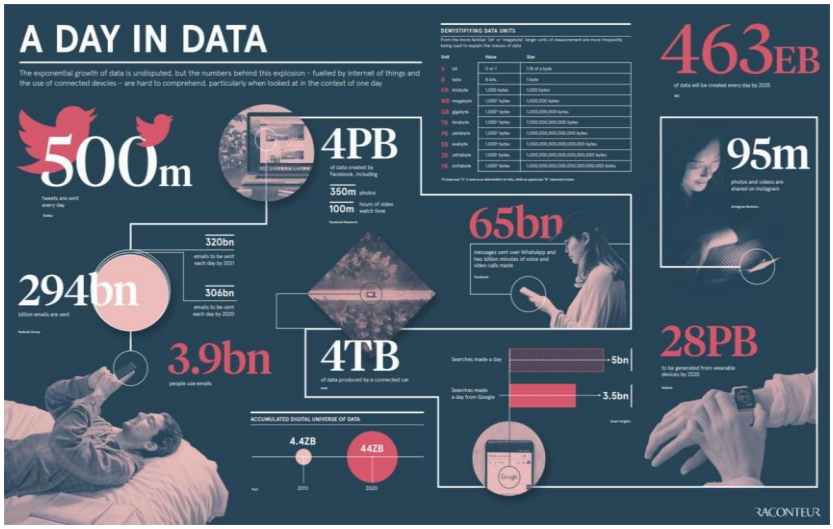
Ο σύγχρονος κόσμος αλλάζει με ιλιγγιώδεις ρυθμούς. Η Παγκοσμιοποίηση, η ραγδαία τεχνολογική εξέλιξη και ακολούθως η ψηφιοποίηση δημιουργούν νέες ανάγκες και ευκαιρίες. Το διαδίκτυο δεν αποτελεί απλά ένα δημοφιλές μέσο αλλά επηρεάζει σε μεγάλο βαθμό τις ζωές και την καθημερινότητα όλων. Η γνώση δεν περιορίζεται στη γραμμική σκέψη αλλά απαιτεί πλέον συνδυαστική σκέψη και σύνθεση των πληροφοριών. Ο 21ος αιώνας αναμφισβήτητα αντιπροσωπεύει την προσαρμογή σε μια νέα πραγματικότητα στην οποία κυριαρχεί το φαινόμενο της ψηφιοποίησης, λαμβάνοντας υπόψη την ολοένα συχνότερη επαφή του ανθρώπου με τις νέες ψηφιακές τεχνολογίες στην καθημερινή ζωή, από το χώρο εργασίας μέχρι τις δραστηριότητες ψυχαγωγίας που σχετίζονται με τον ψηφιακό κόσμο.

Τι είναι όμως τα δεδομένα;

Τα δεδομένα είναι πληροφορία που προέρχεται από διάφορες πηγές και διακινείται διαρκώς σε ένα ταχύρρυθμα αναπτυσσόμενο ψηφιακό σύμπαν. Το διαδίκτυο των πραγμάτων (Internet of Things), η χρήση μαζικών δεδομένων (big data), η τρισδιάστατη εκτύπωση (3D-printing), τα κρυπτονομίσματα, οι υπηρεσίες ροής (Streaming Services) αποτελούν μόνο μερικά από τα παραδείγματα

ψηφιοποίησης του κόσμου και διακίνησης μεγάλου όγκου δεδομένων. Σύμφωνα με τον Jeff Desjardins, σε άρθρο του στο World Economic Forum, μέχρι το 2025, έχει εκτιμηθεί πως καθημερινά θα δημιουργούνται 463 exabytes δεδομένων, αριθμός αντίστοιχος με 212.765.957 DVDs. ([1])

Ένας πιο εμπεριστατωμένος ορισμός για τα δεδομένα είναι ότι πρόκειται για μη οργανωμένες πληροφορίες που συναντώνται σε διάφορες μορφές και υποβάλλονται σε επεξεργασία για να προκύψει νόημα. Γενικά, τα δεδομένα αποτελούνται από γεγονότα, παρατηρήσεις, αντιλήψεις, αριθμούς, χαρακτήρες, σύμβολα και εικόνες που μπορούν να ερμηνευθούν έτσι ώστε το νόημα που θα εξαχθεί να μπορεί να χρησιμοποιηθεί για διαφορετικούς σκοπούς. Καθημερινά παράγεται ένας ακαθόριστος αριθμός δεδομένων ανά δευτερόλεπτο. Αυτό μεταφράζεται σε πολλά tweets, emails, blog posts και κάθε είδους ψηφιακή πληροφορία που μπορούμε να σκεφτούμε.



Εικόνα 1.1: Ο όγκος των δεδομένων που διακινούνται καθημερινά.

1.2 Η ΕΠΙΣΤΗΜΗ ΤΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΔΕΔΟΜΕΝΩΝ

Η επιστήμη των δεδομένων αναφέρεται στην κατασκευή συστημάτων που επιτρέπουν τη συλλογή και χρήση των δεδομένων και στη συνέχεια τη διαχείριση και ανάλυσή τους. Η χρηστικότητα των δεδομένων συνήθως απαιτεί μεγάλους υπολογισμούς και χώρο αποθήκευσης, καθώς επίσης επεξεργασία και καθαρισμό των δεδομένων.

Περίπου μεταξύ των δεκαετιών 1970 και 1980 ο όρος μεθοδολογία της διαχείρισης της πληροφορίας (Information Engineering Methodology - IEM) επινοήθηκε για να περιγράψει τον σχεδιασμό της βάσης δεδομένων και τη χρήση προγραμμάτων για την ανάλυση και επεξεργασία δεδομένων. Αυτές οι τεχνικές σκόπευαν να χρησιμοποιηθούν από διαχειριστές βάσεων δεδομένων (Database Administrators – DBAs) και από αναλυτές συστημάτων που στηρίζονταν στην κατανόηση των επεξεργαστικών αναγκών των οργανισμών για τη δεκαετία του 1980. Στις αρχές της δεκαετίας του 2000, τα δεδομένα και η τα εργαλεία της επεξεργασίας τους τα διαχειριζόταν το τμήμα IT (Information Technology) των περισσότερων εταιρειών. Ακολούθησαν άλλες ομάδες των επιχειρήσεων που χρησιμοποίησαν τα δεδομένα ως κομμάτι της δουλειάς τους (π.χ. δημιουργία αναφορών).

Στη δεκαετία του 2010, με την άνοδο του διαδικτύου, η μαζική αύξηση στον όγκο των δεδομένων, στην ταχύτητα και την ποικιλομορφία τους οδήγησε στον όρο μεγάλα δεδομένα. Εξαιτίας της επέκτασης των δεδομένων, μεγάλες εταιρείες όπως η Google, Facebook, Apple και Microsoft ξεκίνησαν να ασχολούνται με τη διαχείριση δεδομένων, μία τεχνολογία λογισμικού η οποία εστιάζει στα δεδομένα και πιο

συγκεκριμένα στην υποδομή, στην αποθήκευση, στην προστασία, στην εξόρυξη, στην μοντελοποίηση, στην επεξεργασία και την διαχείριση μεταδεδομένων. Τα δεδομένα άρχισαν να χρησιμοποιούνται από πολλά περισσότερα τμήματα μιας επιχείρησης όπως οι πωλήσεις και το Marketing και όχι μόνο το τμήμα IT, όπως συνέβαινε παλαιότερα.

1.2.1 ΕΙΔΗ ΔΕΔΟΜΕΝΩΝ

Δομημένα Δεδομένα – Structured Data

Μπορούν να κατηγοριοποιηθούν με βάση τη δομή τους ως εξής: Δομημένα, Ημι-δομημένα και Μη δομημένα. Τα Δομημένα δεδομένα (Structured) είναι τα δεδομένα που έχουν μία καθορισμένη δομή, είναι λιγότερο περίπλοκα στη διαχείρισή τους και την επεξεργασία. Σε πολλές περιπτώσεις μπορούν να αναπαρασταθούν σε έναν πίνακα με σειρές και στήλες. Τα δομημένα δεδομένα μπορούν να συλλεχθούν, να εξαχθούν, να αποθηκευτούν και να οργανωθούν σε τυπικές βάσεις δεδομένων. Παραδείγματα τέτοιων δεδομένων είναι εκείνα των σχεσιακών βάσεων SQL και των OLTP (Online Transaction Processing) συστημάτων. Ωστόσο, αποτελούν ένα μικρό σύνολο του όγκου των δεδομένων που είναι διαθέσιμα και έχουν περιορισμένη χρησιμότητα λόγω της προκαθορισμένης δομής τους. ([2]), ([3]), ([4])

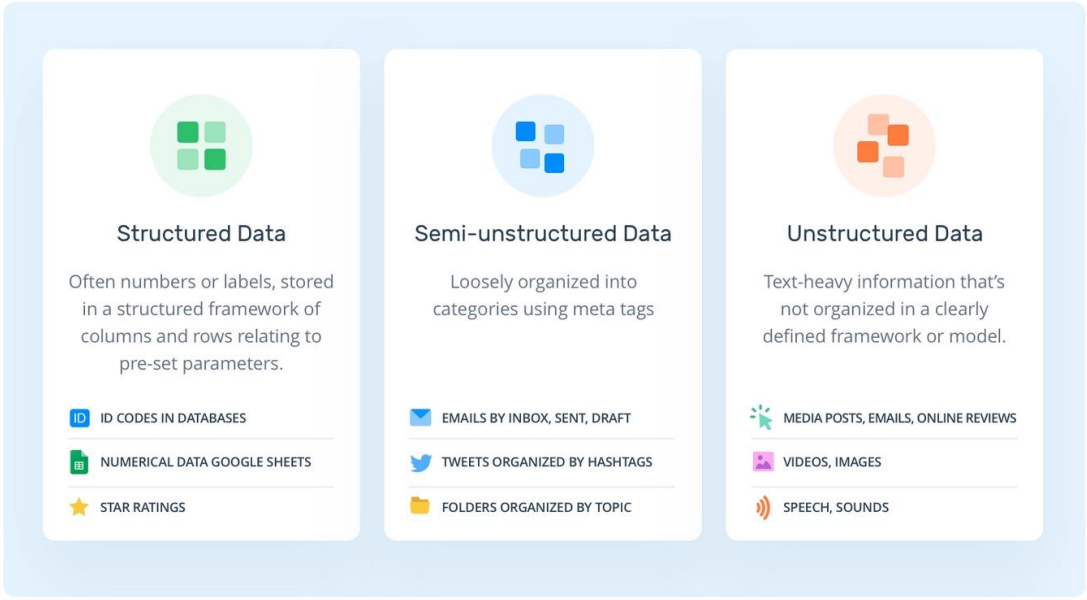
Ημι-δομημένα Δεδομένα – Semi-structured Data

Τα Ημι-δομημένα δεδομένα (Semi-structured) είναι δεδομένα που έχουν κάποιες οργανωτικές ιδιότητες, αλλά στερούνται σταθερού σχήματος. Περιέχουν ετικέτες και στοιχεία ή μεταδεδομένα, που χρησιμοποιούνται για την ομαδοποίηση των δεδομένων και την οργάνωσή τους σε μια ιεραρχία. Μερικά παραδείγματα των Ημι-δομημένων δεδομένων περιλαμβάνουν τα Email, τη γλώσσα σήμανσης XML και τα αρχεία JSON. Βασικό τους μειονέκτημα αποτελεί το γεγονός ότι δεν μπορούν να αποθηκευτούν σε μορφή γραμμών και στηλών όπως στις βάσεις δεδομένων. ([2]), ([7]), ([8])

Μη Δομημένα Δεδομένα – Unstructured Data

Μη δομημένα δεδομένα είναι εκείνα που δεν έχουν μια εύκολα αναγνωρίσιμη δομή και, ως εκ τούτου, δεν μπορούν να οργανωθούν σε μια σχεσιακή βάση δεδομένων με τη μορφή γραμμών και στηλών. Δεν ακολουθεί κάποια συγκεκριμένη μορφή, ακολουθία ή κανόνες. Σε αντίθεση με τα δομημένα δεδομένα αποτελούνται από πολλές διαφορετικές πηγές δεδομένων και διαθέτουν μια ποικιλία εφαρμογών επιχειρηματικής ανάλυσης. Ορισμένες από τις πηγές μη δομημένων δεδομένων θα μπορούσαν περιλαμβάνουν: ιστοσελίδες, τροφοδοσίες μέσων κοινωνικής δικτύωσης εικόνες σε διάφορες μορφές, αρχεία βίντεο και ήχου, διάφοροι τύποι εγγράφων (PDF, PowerPoint) και έρευνες. Τα μειονεκτήματα των μη δομημένων δεδομένων χρειάζονται ειδικές βάσεις δεδομένων NoSQL με τα δικά τους εργαλεία ανάλυσης, ενώ παράλληλα έχουν μία δυσκολία στην ταξινόμηση καθώς δεν υπάρχει προκαθορισμένη δομή. ([2]), ([5]), ([6])

-



Εικόνα 1.2: Δομημένα, ημι-δομημένα και αδόμητα δεδομένα.

1.2.2 ΕΙΔΗ ΑΠΟΘΗΚΕΥΣΗΣ ΔΕΔΟΜΕΝΩΝ

Ως προς τον τρόπο αποθήκευσης των δεδομένων υπάρχουν τέσσερα κύρια είδη που χρησιμοποιούνται ως επί τω πλείστων, το καθένα εκπληρώνοντας ένα διαφορετικό σκοπό για την αντιμετώπιση μιας συγκεκριμένης ανάγκης: Οι Λειτουργικές Βάσεις Δεδομένων (Operational / OLTP Databases) οι Λίμνες δεδομένων (Data Lakes), οι Αποθήκες δεδομένων (Data Warehouses) και τα Πρατήρια δεδομένων (Data Mart).

Λειτουργική Βάση Δεδομένων – Operational / OLTP Database

Οι λειτουργικές ή αλλιώς OLTP (Online Transaction Processing) βάσεις δεδομένων χρησιμεύουν ως κεντρικός χώρος αποθήκευσης για τα δεδομένα που παράγονται από διάφορες επιχειρηματικές διαδικασίες και οι λειτουργίες τους βασίζονται σε "συναλλαγές" που πραγματοποιούν οι χρήστες με ερωτήματα (queries). Ο όρος "συναλλαγή" αναφέρεται σε μια μεμονωμένη εργασία ή μια ακολουθία εργασιών που εκτελούνται σε μια βάση δεδομένων. Οι συναλλαγές σε μια λειτουργική βάση δεδομένων συνήθως περιλαμβάνουν λειτουργίες όπως:

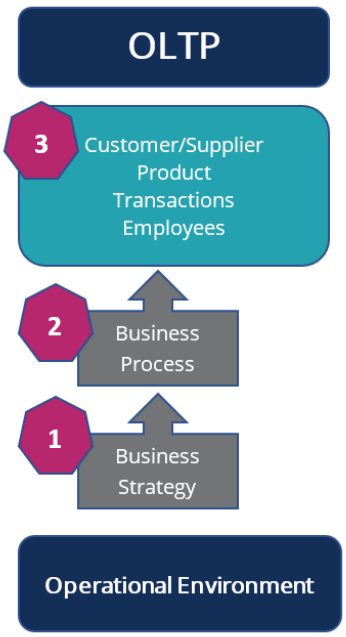
- Εισαγωγή δεδομένων: Προσθήκη νέων εγγραφών ή στοιχείων στη βάση δεδομένων.
- Ενημέρωση δεδομένων: Τροποποίηση υπαρχόντων στοιχείων ή εγγραφών στη βάση δεδομένων.
- Διαγραφή δεδομένων: Αφαίρεση στοιχείων ή εγγραφών από τη βάση δεδομένων.

Οι βάσεις δεδομένων αυτές έχουν σχεδιαστεί για να παρέχουν ταυτόχρονη πρόσβαση σε πολλούς χρήστες ή εφαρμογές, διασφαλίζοντας παράλληλα την αξιοπιστία των δεδομένων. Βασίζονται συνήθως στο μοντέλο σχεσιακών δεδομένων, όπου τα δεδομένα οργανώνονται σε πίνακες, σειρές και στήλες με την επιβολή περιορισμών που αποσκοπούν στην διατήρηση της μοναδικότητας, όπως η χρήση πρωτεύοντων κλειδιών (primary keys). Χρησιμοποιούν ένα κανονικοποιημένο μοντέλο για την αποφυγή πλεονασμού των δεδομένων και τηρούν τις ιδιότητες ACID για την ακεραιότητα τους. Ο όρος "ACID" είναι μια συντομογραφία των λέξεων Atomicity (Ατομικότητα), Consistency (Συνέπεια), Isolation (Απομόνωση) και Durability (Ανθεκτικότητα).

- Η ατομικότητα εξασφαλίζει ότι μια συναλλαγή αντιμετωπίζεται ως μια ενιαία μονάδα εργασίας. Σημαίνει ότι είτε όλες οι λειτουργίες μιας συναλλαγής έχουν ολοκληρωθεί επιτυχώς, είτε καμία από αυτές δεν εφαρμόζεται καθόλου. Εάν οποιοδήποτε μέρος μιας συναλλαγής αποτύχει, ολόκληρη η συναλλαγή επαναφέρεται και η βάση δεδομένων παραμένει στην αρχική της κατάσταση.
- Η συνέπεια εγγυάται ότι μια συναλλαγή μεταφέρει τη βάση δεδομένων από μια έγκυρη κατάσταση σε μια άλλη. Διασφαλίζει ότι τα δεδομένα ικανοποιούν όλους τους καθορισμένους περιορισμούς και κανόνες ακεραιότητας. Με άλλα λόγια, η βάση δεδομένων παραμένει συνεπείς πριν και μετά την εκτέλεση μιας συναλλαγής.
- Η απομόνωση αποσκοπεί στην ανεξάρτητη λειτουργία των ταυτόχρονων συναλλαγών χωρίς να παρεμβαίνουν η μία στα δεδομένα της άλλης.

Επιπλέον βασικά χαρακτηριστικά της λειτουργικής / OLTP αποθήκευσης είναι οι ενημερώσεις σε πραγματικό χρόνο και η εφαρμογή μέτρων ασφαλείας στις συναλλαγές για την προστασία ευαίσθητων δεδομένων.

([114]), ([115]), ([116]), ([117]), ([118]), ([119])



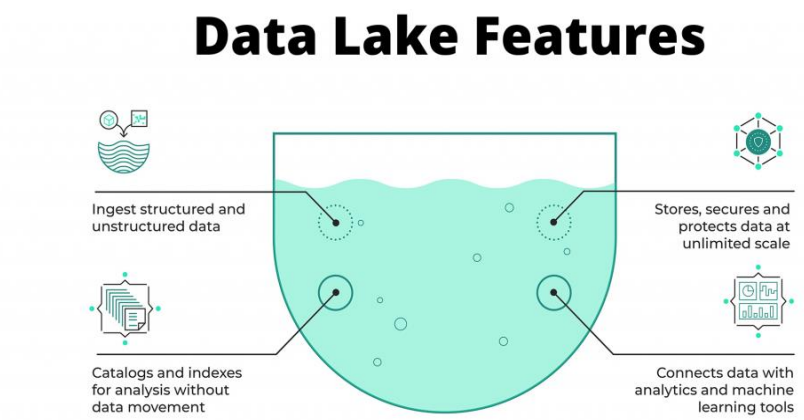
Εικόνα 1.2.1: Διάγραμμα ροής της OLTP βάσης δεδομένων.

Λίμνη Δεδομένων – Data Lake

Μια λίμνη δεδομένων είναι ένας μεγάλος αποθηκευτικός χώρος ανεπεξέργαστων δεδομένων, είτε ημι-δομημένων είτε μη δομημένων. Αυτά τα δεδομένα συγκεντρώνονται από διάφορες πηγές και αποθηκεύονται σε ακατέργαστη κατάσταση, χωρίς προκαθορισμένη δομή ή σχήμα. Το σχήμα εφαρμόζεται κατά την ανάλυση δεδομένων ή όταν γίνεται πρόσβαση στα δεδομένα, ακολουθώντας μια προσέγγιση σχήματος κατά την ανάγνωση. Αυτή η ευελιξία επιτρέπει την εύκολη εξερεύνηση και ανάλυση διαφορετικών τύπων δεδομένων.

Οι λίμνες δεδομένων είναι εξαιρετικά επεκτάσιμες και ικανές να χειρίζονται τεράστιους όγκους δεδομένων. Μπορούν να επεκτείνουν τη χωρητικότητα αποθήκευσης και να αξιοποιήσουν πόρους υπολογιστικού νέφους με βάση τη ζήτηση. Οι λίμνες δεδομένων υποστηρίζουν την εξερεύνηση δεδομένων, την ανάλυση και τη μηχανική μάθηση. Οι χρήστες μπορούν να επεξεργαστούν, να μετασχηματίσουν και να δομήσουν δεδομένα μέσα στη λίμνη ή να τα εξαγάγουν για περαιτέρω ανάλυση σε εξειδικευμένα περιβάλλοντα.

Παρόλο που οι λίμνες δεδομένων αποτελούν μία ισχυρή αρχιτεκτονική προσέγγιση λόγω του της αυξανόμενης ποικιλίας και του όγκου των δεδομένων, η προετοιμασία και η αναδιαμόρφωση αυτών των δεδομένων για ανάλυση είναι χρονοβόρα. ([9]), ([102]), ([103])



Εικόνα 1.2.2: Χαρακτηριστικά της λίμνης δεδομένων.

Αποθήκη Δεδομένων – Data Warehouse

Μια αποθήκη δεδομένων είναι ένα σύνολο δεδομένων από πολλές πηγές σε ένα ενιαίο, κεντρικό αποθετήριο που ενοποιεί τα χαρακτηριστικά και τη μορφή τους, καθιστώντας τα έτοιμα προς χρήση. Μια τυπική αποθήκη δεδομένων περιλαμβάνει συχνά μια σχεσιακή βάση δεδομένων για την αποθήκευση και διαχείριση δεδομένων, μια διαδικασία ELT (Extraction, Loading and Transformation) για την προετοιμασία των προς ανάλυση δεδομένων και δυνατότητες στατιστικής ανάλυσης, αναφοράς και εξόρυξης δεδομένων με σκοπό την παρουσίασης σε χρήστες επιχειρήσεων.

Σύμφωνα με τον W.H.Inmon, κορυφαίο αρχιτέκτονα στην κατασκευή συστημάτων αποθήκης δεδομένων, “μια αποθήκη δεδομένων είναι μια θεματοστρεφής (subject-oriented), ενσωματωμένη (integrated), χρονικά μεταβλητή (time-variant) και μη ασταθής (nonvolatile) συλλογή δεδομένων για την υποστήριξη των διαδικασιών λήψης αποφάσεων.”

- Θεματοστρεφής: Τα δεδομένα οργανώνονται λογικά γύρω από κύρια θέματα του οργανισμού, π.χ. γύρω από πελάτες, πωλήσεις ή προϊόντα που παράγονται.
- Ενσωματωμένη: Όλα τα σχετικά με το θέμα δεδομένα συνδυάζονται και μπορούν να αναλυθούν μαζί.
- Χρονικά μεταβλητή: Τα ιστορικά δεδομένα διατηρούνται με λεπτομερειακή μορφή, τα οποία αναφέρονται στη συλλογή δεδομένων από προηγούμενες χρονικές περιόδους.
- Μη ασταθής: Τα δεδομένα είναι μόνο για ανάγνωση, δεν ενημερώνονται ή αλλάζουν από τους χρήστες. Η κύρια μέθοδος αποθήκευσης σε αποθήκες δεδομένων είναι η OLAP (Online Analytical Processing).

Τα συστήματα OLAP συχνά αποθηκεύουν ιστορικά δεδομένα για ανάλυση τάσεων. Η OLAP εστιάζει σε σύνθετα αναλυτικά ερωτήματα (queries) και ανάλυση “πολυδιάστατων” (multidimensional) δεδομένων. Με τον όρο “πολυδιάστατα δεδομένα” αναφερόμαστε σε δεδομένα που είναι οργανωμένα και αντιπροσωπεύονται σε πολλαπλές διαστάσεις ή άξονες. Στο πλαίσιο της αποθήκευσης και ανάλυσης δεδομένων, τα πολυδιάστατα δεδομένα αντιπροσωπεύουν πληροφορίες που μπορούν να αναλυθούν και να διερευνηθούν από διάφορες οπτικές γωνίες.

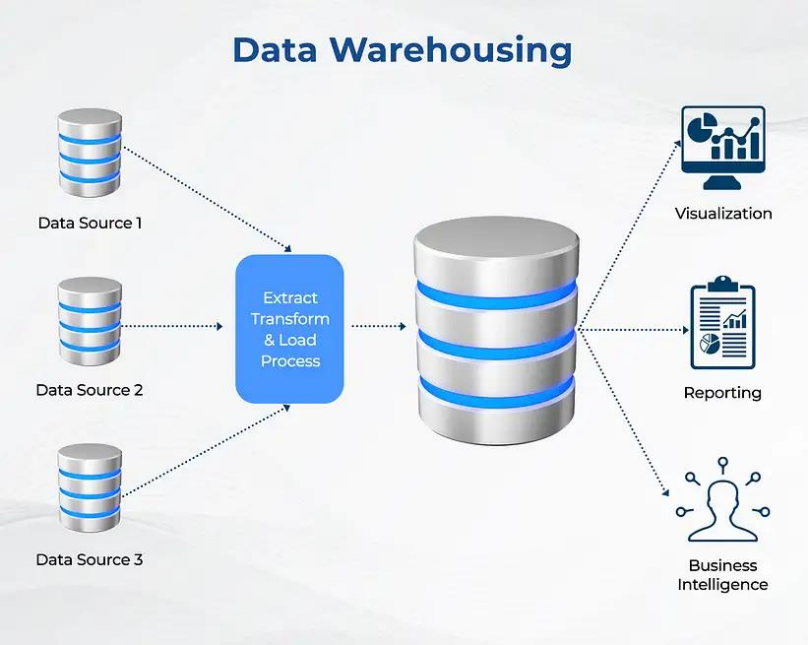
Ας εξετάσουμε ένα παράδειγμα: Έστω ότι έχουμε ένα σύνολο δεδομένων που περιλαμβάνει πληροφορίες σχετικά με τις συναλλαγές πωλήσεων. Κάθε εγγραφή

συναλλαγής μπορεί να περιέχει χαρακτηριστικά όπως ημερομηνία, προϊόν, κατάστημα, ποσότητα και έσοδα. Σε μια παραδοσιακή δισδιάστατη αναπαράσταση, οι σειρές αντιπροσωπεύουν μεμονωμένες συναλλαγές και οι στήλες αντιπροσωπεύουν τα χαρακτηριστικά (ημερομηνία, προϊόν, κατάστημα κ.λπ.). Αυτή η μορφή πίνακα εξυπηρετεί την απλή αναζήτηση και επεξεργασία συναλλαγών.
Ωστόσο, μια πολυδιάστατη αναπαράσταση, υποστηρίζει την προσθήκη πρόσθετων διαστάσεων στα δεδομένα, επιτρέποντας πιο ολοκληρωμένη ανάλυση. Για παράδειγμα, μπορούμε να εισάγουμε τη διάσταση του χρόνου, επιτρέποντάς την ανάλυση των πωλήσεων σε διαφορετικές περιόδους (ημερήσια, μηνιαία, ετήσια). Μπορούμε επίσης να προσθέσουμε επιπλέον ιδιότητες όπως κατηγορία προϊόντων ή τμήμα πελατών. Οι διαστάσεις αυτές δημιουργούν περισσότερες προοπτικές για την ανάλυση των δεδομένων.

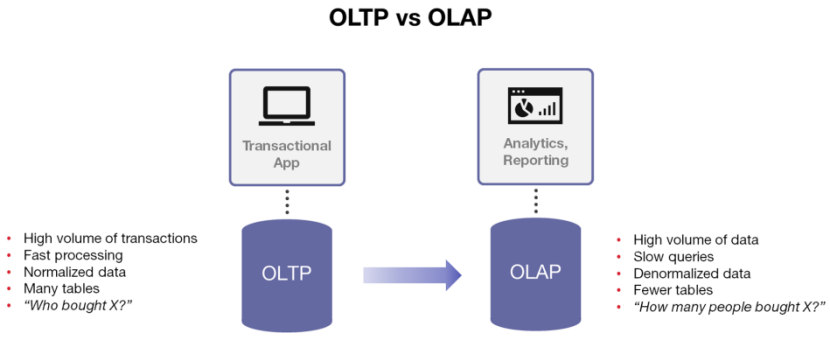
Ένα βασικό χαρακτηριστικού των OLAP βάσεων ονομάζεται κύβος δεδομένων (data cube) το οποίο είναι μια πολυδιάστατη δομή. Κάθε κελί στον κύβο αντιπροσωπεύει έναν συνδυασμό τιμών από διαφορετικές διαστάσεις και περιέχει συγκεντρωτικά μετρήσεις όπως έσοδα, ποσότητα πωλήσεων ή μέση τιμή.
Δύο δημοφιλείς τεχνικές μοντελοποίησης πολυδιάστατων μοντέλων δεδομένων που χρησιμοποιούνται στις βάσεις δεδομένων OLAP είναι το σχήμα αστεριού (star schema) και το σχήμα χιονονιφάδας (snowflake schema).

- Το σχήμα νιφάδας χιονιού είναι ένας τύπος πολυδιάστατου σχήματος που αναπαριστά δεδομένα με ιεραρχικό τρόπο. Ονομάζεται "νιφάδα χιονιού" επειδή η δομή του σχήματος μοιάζει με το σχήμα μιας νιφάδας χιονιού με κλαδιά που αναπτύσσονται έξω από έναν κεντρικό πίνακα δεδομένων. Σε ένα σχήμα νιφάδας χιονιού, οι διαστάσεις κανονικοποιούνται σε πολλούς σχετικούς πίνακες.
- Το σχήμα αστεριού είναι μια απλούστερη τεχνική μοντελοποίησης δεδομένων που οργανώνει τα δεδομένα σε έναν κεντρικό πίνακα γεγονότων που συνδέεται με πίνακες πολλαπλών διαστάσεων. Ονομάζεται "αστέρι" επειδή η δομή μοιάζει με αστέρι με τον πίνακα γεγονότων στο κέντρο και τους πίνακες διαστάσεων να ακτινοβολούν προς τα έξω. Το σχήμα αστεριού αποκανονικοποιεί τις διαστάσεις σε έναν ενιαίο πίνακα για κάθε διάσταση.

([12]), ([99]), ([100]), ([101]), ([102]), ([114])



Εικόνα 1.2.3: Σχεδιάγραμμα της αποθήκης δεδομένων.



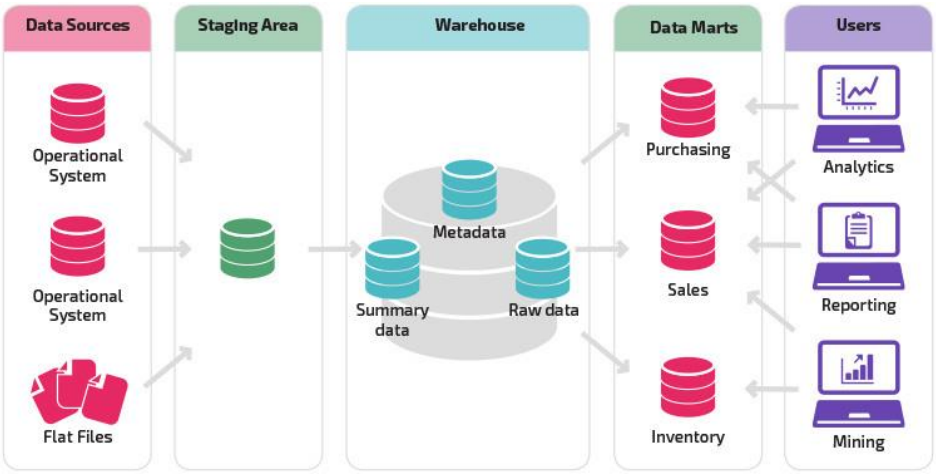
Εικόνα 1.2.4: Διαφορές OLTP και OLAP αποθήκευσης.

Πρατήριο Δεδομένων - Data Mart

Το πρατήριο δεδομένων (data mart) είναι παρόμοιος αποθηκευτικός χώρος με την αποθήκη δεδομένων με την διαφορά εδώ να βρίσκεται στο γεγονός ότι η έκθεση των δεδομένων στους χρήστες ή στα τμήματα μιας επιχείρησης είναι περιορισμένη. Για παράδειγμα, μπορεί να δημιουργηθεί ένα πρατήριο δεδομένων για την παροχή

αναφορών και αναλύσεων για το τμήμα marketing. Περιορίζοντας τα δεδομένα στο συγκεκριμένο επιχειρησιακό τμήμα, η επιχείρηση δεν χρειάζεται να προσπελάσει άσχετα δεδομένα.

Η ιδιαιτερότητα αυτή των πρατηρίων δεδομένων μεταφράζεται σε δύο κύρια πλεονεκτήματα. Το πρώτο από αυτά είναι η ασφάλεια. Περιορίζοντας την έκθεση των μη χρήσιμων στοιχείων στο εκάστοτε τμήμα, διασφαλίζεται η ορθή χρήση αυτών. Το δεύτερο είναι η ταχύτητα, εφόσον τα δεδομένα προς διαχείριση είναι λιγότερα, ο φόρτος επεξεργασίας μειώνεται και οι κώδικες (queries) τρέχουν γρηγορότερα. Αποτελεί εντούτοις μία πιο κοστοβόρα λύση σε σύγκριση με τις αποθήκες δεδομένων, ενώ δεν μπορεί και να αποθηκεύσει μεγάλο όγκο δεδομένων που προέρχονται από κάθε ένα από τα τμήματα ενός οργανισμού. ([10]), ([11]), ([12]), ([13])



Εικόνα 1.2.5: Το πλήρες σχήμα μιας αποθήκης δεδομένων και η θέση των data marts.

1.2.3 ΑΓΩΓΟΙ ΔΕΔΟΜΕΝΩΝ - DATA PIPELINES

Ένας αγωγός δεδομένων (data pipeline) είναι ένα μέσο μεταφοράς δεδομένων από ένα μέρος (την πηγή) σε έναν προορισμό (όπως μια αποθήκη δεδομένων). Στην πορεία, τα δεδομένα μετασχηματίζονται και βελτιστοποιούνται, φτάνοντας σε μια

κατάσταση που μπορούν να αναλυθούν και να χρησιμοποιηθούν για την ανάπτυξη επιχειρηματικών πληροφοριών.

Ένας αγωγός δεδομένων είναι ουσιαστικά ένα από τα βήματα που εμπλέκονται στη συγκέντρωση, οργάνωση και μετακίνηση δεδομένων. Οι σύγχρονοι αγωγοί δεδομένων αυτοματοποιούν πολλά από τα βήματα που εμπλέκονται στον μετασχηματισμό και τη βελτιστοποίηση των συνεχών ροών δεδομένων. Συνήθως, αυτό περιλαμβάνει τη φόρτωση ανεπεξέργαστων δεδομένων σε έναν πίνακα για ενδιάμεση αποθήκευση και, στη συνέχεια, την αλλαγή τους πριν την τελική εισαγωγή τους στους πίνακες αναφορών προορισμού.

Τα δεδομένα μπορούν να προέρχονται από μια μεγάλη ποικιλία πηγών (API, βάσεις δεδομένων SQL και NoSQL, αρχεία κλπ.) όμως αυτά τα δεδομένα συνήθως δεν είναι έτοιμα για άμεση χρήση. Τις εργασίες προετοιμασίας δεδομένων συνήθως αναλαμβάνουν οι επιστήμονες δεδομένων ή οι μηχανικών δεδομένων, οι οποίοι δομούν τα δεδομένα ώστε να ανταποκρίνονται στις ανάγκες της εκάστοτε επιχειρηματικής χρήσης. Μόλις τα δεδομένα φιλτραριστούν, συγχωνευθούν και συνοψιστούν κατάλληλα, μπορούν στη συνέχεια να αποθηκευτούν και να εμφανιστούν για χρήση. Οι καλά οργανωμένοι αγωγοί δεδομένων παρέχουν τη βάση για μια σειρά έργων δεδομένων. Αυτό μπορεί να περιλαμβάνει διερευνητικές αναλύσεις δεδομένων και εργασίες μηχανικής μάθησης.

Υπάρχουν δύο κύριοι τύποι αγωγών δεδομένων:

1) Επεξεργασία Παρτίδας

Όπως υποδηλώνει το όνομα, η μαζική επεξεργασία φορτώνει «παρτίδες» δεδομένων σε ένα χώρο αποθήκευσης κατά τη διάρκεια καθορισμένων χρονικών διαστημάτων, τα οποία συνήθως προγραμματίζονται σε εργάσιμες ώρες εκτός αιχμής. Με αυτόν τον τρόπο, άλλοι φόρτοι εργασίας δεν επηρεάζονται, καθώς οι εργασίες μαζικής επεξεργασίας τείνουν να λειτουργούν με μεγάλο όγκο δεδομένων, γεγονός που μπορεί να επιβαρύνει το σύστημα. Η ομαδική επεξεργασία είναι συνήθως η βέλτιστη διοχέτευση δεδομένων όταν δεν υπάρχει άμεση ανάγκη ανάλυσης ενός συγκεκριμένου συνόλου δεδομένων (π.χ. μηνιαία λογιστική).

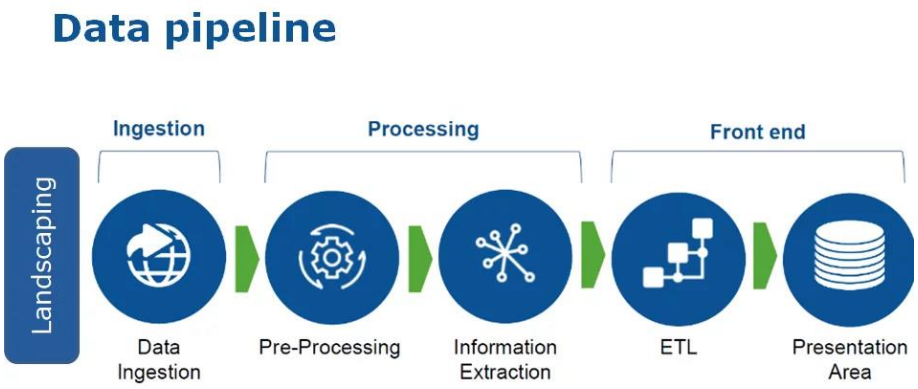
Οι εργασίες ομαδικής επεξεργασίας σχηματίζουν μια ροή εργασίας διαδοχικών εντολών, όπου η έξοδος μιας εντολής γίνεται η είσοδος της επόμενης εντολής. Για παράδειγμα, μια εντολή μπορεί να ξεκινήσει την απορρόφηση δεδομένων, η επόμενη εντολή μπορεί να ενεργοποιήσει το φιλτράρισμα

συγκεκριμένων στηλών και η επόμενη εντολή μπορεί να χειριστεί τη συγκέντρωση. Αυτή η σειρά εντολών θα συνεχιστεί έως ότου τα δεδομένα μετασχηματιστούν πλήρως και εγγραφούν σε αποθήκη δεδομένων.

2) Ροή Δεδομένων

Σε αντίθεση με την ομαδική επεξεργασία, τα δεδομένα ροής αξιοποιούνται όταν απαιτείται για τη συνεχή ενημέρωση των δεδομένων. Για παράδειγμα, οι εφαρμογές ή τα συστήματα σημείων πώλησης χρειάζονται δεδομένα σε πραγματικό χρόνο για να ενημερώσουν το απόθεμα και το ιστορικό πωλήσεων των προϊόντων τους. Με αυτόν τον τρόπο, οι πωλητές μπορούν να ενημερώνουν τους καταναλωτές εάν ένα προϊόν είναι σε απόθεμα ή όχι. Μια μεμονωμένη ενέργεια, όπως μια πώληση προϊόντος, θεωρείται "συμβάν" και τα σχετικά συμβάντα, όπως η προσθήκη ενός στοιχείου στο ταμείο, συνήθως ομαδοποιούνται ως "θέμα" ή "ροή". Αυτά τα συμβάντα στη συνέχεια μεταφέρονται μέσω συστημάτων ανταλλαγής μηνυμάτων όπως το λογισμικό ανοιχτού κώδικα, Apache Kafka.

Δεδομένου ότι τα συμβάντα δεδομένων υποβάλλονται σε επεξεργασία λίγο μετά την πραγματοποίησή τους, τα συστήματα επεξεργασίας ροής έχουν χαμηλότερο λανθάνοντα χρόνο από τα συστήματα παρτίδας, αλλά δεν θεωρούνται τόσο αξιόπιστα όσο τα συστήματα επεξεργασίας παρτίδων, καθώς τα μηνύματα μπορεί να απορριφθούν ακούσια ή να περάσουν μεγάλο χρονικό διάστημα στην ουρά.
([14]), ([15])



Εικόνα 1.2.6: Βήματα λειτουργίας ενός Αγωγού Δεδομένων.

1.2.4 ETL / ELT

Οι διαδικασίες ETL (extract, load, transform) και ELT (extract, transform, load) είναι και οι δύο διαδικασίες ενσωμάτωσης δεδομένων οι οποίες μετακινούν ανεπεξέργαστα δεδομένα από ένα σύστημα χρησιμοποιώντας μετατροπές που περιλαμβάνουν υπολογισμούς, συνδέσεις και διαγραφή διπλοτύπων, φορτώνοντας τα τελικά σε κάποιο είδος αποθήκης δεδομένων.

Η διαδικασία ETL χρησιμοποιείται όταν τα δεδομένα πρέπει να μετατραπούν, ούτως ώστε να ενσωματωθούν στην βάση προορισμού. Τα εξαγόμενα δεδομένα δρομολογούνται σε έναν server επεξεργασίας και στη συνέχεια τα ανεπεξέργαστα δεδομένα μετατρέπονται σε δεδομένα που βασίζονται σε SQL. Η μέθοδος αυτή εμφανίστηκε την δεκαετία ‘70 όταν οι βάσεις δεδομένων διέθεταν πεπερασμένη ισχύ επεξεργασίας. Η μετατροπή των δεδομένων πριν φτάσουν στην αποθήκη επέτρεπε στους οργανισμούς να περιορίσουν την χρήση του πολύτιμου χώρου, επεξεργαστικής ισχύς και εύρους ζώνης μεταξύ της ροής.

Σήμερα, οι σύγχρονες βασισμένες στο νέφος αποθήκες δεδομένων με τις υψηλές ταχύτητες που διαθέτουν μπορούν να αποθηκεύσουν εικονικά απεριόριστη ποσότητα δεδομένων και να προσφέρουν κλιμακωτή επεξεργασία. Αυτή η τεχνολογική εξέλιξη ανέδειξε μια νέα αρχιτεκτονική ενοποίησης δεδομένων που ονομάζεται ELT, στην οποία τα δεδομένα φορτώνονται αμέσως στην αποθήκη δεδομένων ή στη λίμνη δεδομένων μετά την εξαγωγή τους και μετασχηματίζονται μόνο όταν οι χρήστες είναι έτοιμοι να χρησιμοποιήσουν τα δεδομένα για ανάλυση. Τόσο η ETL όσο και η ELT έχουν τα δυνατά και τα αδύνατα σημεία τους, καθένα από τα οποία παρέχει μεγαλύτερη αξία για ορισμένες περιπτώσεις χρήσης.

Η διαδικασία ETL χρησιμοποιείται καλύτερα για τον συγχρονισμό πολλών περιβαλλόντων χρήσης δεδομένων. Τα πλεονεκτήματα της είναι:

1) Μεγαλύτερη συμμόρφωση στους κανονισμούς απορρήτου.

Οι εταιρείες που υπόκεινται σε κανονισμούς απορρήτου δεδομένων, όπως ο GDPR, η HIPAA ή ο νόμος περί απορρήτου των καταναλωτών της Καλιφόρνια (CCPA), πρέπει να αφαιρέσουν, να συγκαλύψουν ή να κρυπτογραφήσουν

συγκεκριμένα πεδία δεδομένων για να προστατεύσουν το απόρρητο των πελατών τους. Το ETL παρέχει μεγαλύτερη ασφάλεια δεδομένων επειδή εκτελεί μετασχηματισμούς που προστατεύουν τα προσωπικά δεδομένα πριν τα τοποθετήσει στην αποθήκη δεδομένων. Αυτή η ασφάλεια δεδομένων εμποδίζει τους διαχειριστές του συστήματος να έχουν πρόσβαση στις ευαίσθητες πληροφορίες μέσω αρχείων καταγραφής στην αποθήκη δεδομένων.

2) Μειωμένο κόστος αποθήκευσης

Επειδή η ETL μεταφέρει μόνο δεδομένα που έχουν μετατραπεί σε αποθήκη δεδομένων, ο οργανισμός μπορεί να εξοικονομήσει κόστος αποθήκευσης αποθηκεύοντας μόνο τα δεδομένα που χρειάζονται. Αντίθετα, η ELT φορτώνει όλα τα δεδομένα σας στην αποθήκη δεδομένων.

Επειδή οι αγωγοί δεδομένων εκτελούν εξελιγμένους μετασχηματισμούς προσαρμοσμένους στις ανάγκες ανάλυσης των χρηστών, απαιτούν μια ειδική ομάδα μηχανικών για τη δημιουργία και τη διατήρηση προσαρμοσμένου κώδικα. Η επιπλέον ανάπτυξη απαιτεί χρόνο, καθιστά δύσκολη την προσθήκη πηγών δεδομένων και περιορίζει την επεκτασιμότητα.

Η διαδικασία ELT χρησιμοποιείται καλύτερα σε σύνολα δεδομένων μεγάλου όγκου ή κατά την χρήση δεδομένων σε πραγματικό χρόνο. Τα πλεονεκτήματα της είναι:

1) Μεγαλύτερη ευελιξία

Σε αντίθεση με την ETL, η ELT δεν απαιτεί την ανάπτυξη πολύπλοκων αγωγών πριν από την συγχώνευση δεδομένων. Τα δεδομένα απλώς αποθηκεύονται στην αποθήκη δεδομένων χωρίς να χρειάζεται να μετασχηματιστούν και να δομηθούν πρώτα, αποκτώντας άμεση πρόσβαση σε όλες τις πληροφορίες.

2) Απλότητα

Οι βάσεις δεδομένων SQL προσφέρουν πολλές ενσωματωμένες δυνατότητες για αναζήτηση και διαχείριση δεδομένων. Οι σύγχρονες λύσεις ELT μπορούν να αξιοποιήσουν αυτές τις δυνατότητες για να μετατρέψουν τα δεδομένα μετά τη φόρτωσή τους στην αποθήκη. Αυτό διευκολύνει τις ομάδες IT επιχειρήσεων να διαχειρίζονται μετασχηματισμούς δεδομένων χρησιμοποιώντας τις ενσωματωμένες διαδικασίες SQL μέσα σε βάσεις δεδομένων όπως ο SQL Server.

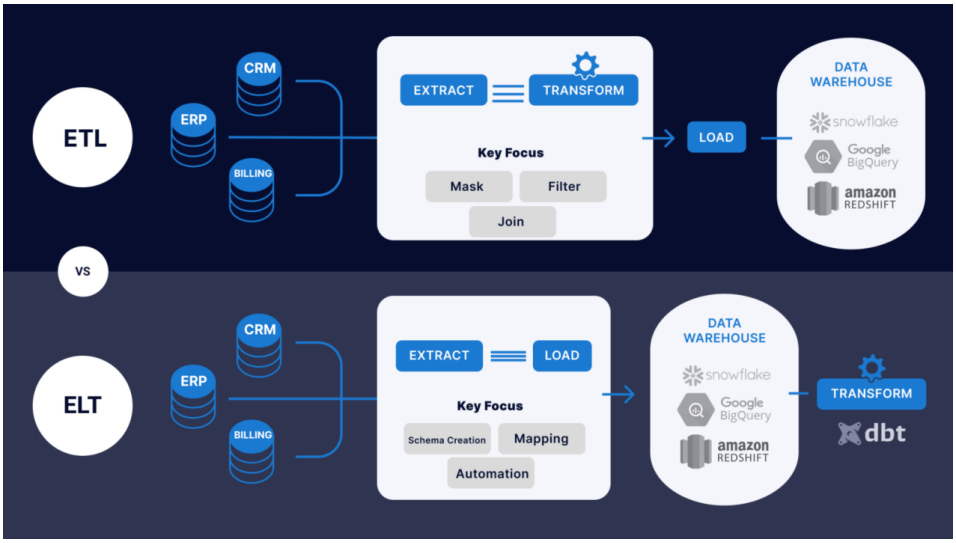
3) Γρήγορη απορρόφηση δεδομένων

Επειδή δεν απαιτείται η μετατροπή δεδομένων σε ειδική μορφή πριν αποθηκευτούν στην αποθήκη δεδομένων ή στη λίμνη δεδομένων, η ELT μπορεί να απορροφήσει άμεσα τα δεδομένα. Οι χρήστες δεν χρειάζεται να περιμένουν για τον καθαρισμό ή την τροποποίηση τους.

4) Μετατροπή μόνο των απαραίτητων δεδομένων

Κατά την ELT, οι χρήστες χρειάζονται μετασχηματισμό μόνο των δεδομένων που απαιτούνται για μια συγκεκριμένη ανάλυση και μπορούν να μετασχηματίσουν ευέλικτα τα δεδομένα με διάφορους τρόπους για να παράγουν συγκεκριμένες μετρήσεις, προβλέψεις και αναφορές. Αντίθετα, η ETL απαιτεί την τροποποίηση ολόκληρου του αγωγού εάν η δομή που αποφασίστηκε προηγουμένως δεν επιτρέπει νέους τύπους ανάλυσης.

Ενώ η ELT είναι εξαιρετική για οργανισμούς που πρέπει να διαχειρίζονται μεγάλες ποσότητες μη δομημένων δεδομένων, αυτές οι λύσεις είναι λιγότερο συμβατές και αξιόπιστες σε σύγκριση με τις αντίστοιχες ETL. Επειδή η ELT απαιτεί το ανέβασμα ευαίσθητων δεδομένων πριν την μετατροπή τους, η διαδικασία εκθέτει ιδιωτικά δεδομένα σε αρχεία καταγραφής που είναι προσβάσιμα στους διαχειριστές του συστήματός. Τα εργαλεία και τα συστήματα της ELT εξακολουθούν να εξελίσσονται, πράγμα που σημαίνει ότι δεν είναι τόσο αξιόπιστα όσο της ETL. Επιπλέον, ενώ το ETL χρειάζεται περισσότερη προσπάθεια για τη ρύθμιση, η δομή δεδομένων του παρέχει πιο ακριβείς πληροφορίες από την ELT. ([16])



Εικόνα 1.2.7: Σχεδιάγραμμα διαδικασιών ETL/ELT.

1.2.5 ΜΕΤΑΔΕΔΟΜΕΝΑ - METADATA

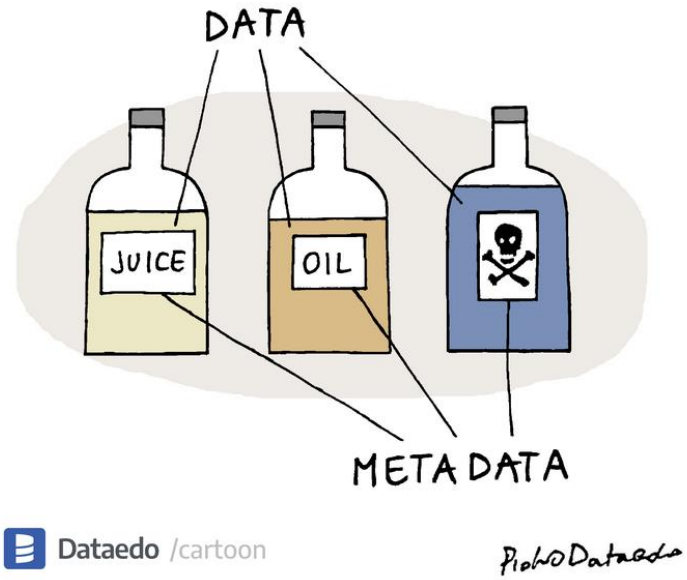
Η διαχείριση των Μεταδεδομένων είναι το μέσο για να περιγραφούν άλλα δεδομένα. Περιλαμβάνει την εγκατάσταση πολιτικών χρήσης και διαδικασιών που διασφαλίζουν ότι η πληροφορία μπορεί να ενσωματωθεί, να αξιολογηθεί, να διαμοιραστεί και να προσπελαστεί ώστε να παρέχεται ένα καλύτερο αποτέλεσμα για τον οργανισμό.

Τα Μεταδεδομένα παράγονται όποτε τα δεδομένα δημιουργούνται, εξάγονται, προστίθενται, αφαιρούνται ή ανανεώνονται. Παραδείγματα μεταδεδομένων περιλαμβάνουν το μέγεθος αρχείων, τις ημερομηνίες δημιουργίας αρχείων και τα ονόματα συγγραφέων. Περισσότερα δεδομένα που μπορούν να προστεθούν είναι ο τίτλος, οι επισυνάψεις ή/και τα σχόλια.

Ο στόχος των μεταδεδομένων είναι να διευκολύνουν ένα άτομο ή πρόγραμμα στον εντοπισμό συγκεκριμένων συνόλων δεδομένων. Αυτό απαιτεί τον σχεδιασμό μιας αποθήκης μεταδεδομένων, την εφαρμογή της και την διευκόλυνση της χρήσης πληροφοριών εντός της αποθήκης. Στα πλεονεκτήματα των μεταδεδομένων περιλαμβάνονται: α) Συνοχή στις έννοιες των μεταδεδομένων ώστε η ποικιλία στις ορολογίες να μην προκαλεί προβλήματα στην ανάκτηση δεδομένων, β) Λιγότερη μείωση στην προσπάθεια και μεγαλύτερη συνοχή στις διάφορες περιπτώσεις δεδομένων χάρη στην δυνατότητα των δεδομένων να επαναχρησιμοποιηθούν

κατάλληλα, γ) διατήρηση των πληροφοριών για το σύνολο του οργανισμού που να μην εξαρτάται από τις γνώσεις ενός συγκεκριμένου υπαλλήλου και δ) μεγαλύτερη αποτελεσματικότητα που οδηγεί σε ταχύτερη ολοκλήρωση των διάφορων projects.

Όταν ένας οργανισμός εφαρμόζει πολιτικές χρήσης για να διαχειρίζεται μεταδεδομένα, είναι σημαντικό για τους managers να συμφωνούν όλοι σε ένα κοινό λεξιλόγιο και ταξινόμηση των δεδομένων. Οι προμηθευτές εργαλείων ETL προσφέρουν εφαρμογές διαχείρισης μεταδεδομένων για ταξινόμηση και διαχείριση των μεταδεδομένων ETL όπως επίσης και μεταδεδομένα που σχετίζονται με τις εφαρμογές πηγής και προορισμού. ([17]), ([18])



Εικόνα 1.2.8: Οπτικός ορισμός της έννοιας των μεταδεδομένων.

1.3 ΜΕΓΑΛΑ ΔΕΔΟΜΕΝΑ – BIG DATA

Ο όρος "Μεγάλα Δεδομένα" ή αλλιώς "Big Data" αναφέρεται ολοένα και συχνότερα τα τελευταία χρόνια στον κλάδο της πληροφορικής και όχι μόνο. Αυτό είναι κάτι λογικό εάν σκεφτούμε πως ο όγκος των δεδομένων που διακινούνται πλέον στο διαδίκτυο είναι τεράστιος. Επομένως από ένα σημείο και έπειτα η αναζήτηση τρόπων για την διαχείριση της υπέρογκης αυτής ροής ήταν αναπόφευκτη.

Παραθέτοντας την επεξήγηση του όρου σε μια φράση θα μπορούσαμε να πούμε πως "τα Μεγάλα Δεδομένα αναφέρονται σε έναν μεγάλο όγκο δεδομένων, τα

οποία δεν δύναται να αποθηκευτούν χρησιμοποιώντας την παραδοσιακή προσέγγιση εντός ενός χρονικού ορίου". Η έννοια των μεγάλων δεδομένων απέκτησε δημοφιλία στις αρχές 2000, όταν ο αναλυτής Doug Laney διατύπωσε την πλέον διαδεδομένη επεξήγηση του όρου ως τα τρία "V" (Volume – Όγκος, Velocity – Ταχύτητα, Variety – Ποικιλομορφία):

1) Όγκος

Οι οργανισμοί συλλέγουν δεδομένα από μια πληθώρα πηγών, συμπεριλαμβανομένων των συναλλαγών, έξυπνες συσκευές (IoT), βίντεο, εικόνες, ήχος, κοινωνικά δίκτυα και άλλα. Κατά το παρελθόν, η αποθήκευση όλων αυτών των δεδομένων θα ήταν κοστοβόρα. Χρησιμοποιώντας όμως φθηνότερους τρόπους αποθήκευσης όπως λίμνες δεδομένων και το νέφος, η κατάσταση έχει διευκολυνθεί.

2) Ταχύτητα

Η Ταχύτητα ουσιαστικά μετράει πόσο γρήγορα έρχονται τα δεδομένα. Κάποια από αυτά λαμβάνονται σε πραγματικό χρόνο, ενώ σε άλλες περιπτώσεις ανά παρτίδα. Με την ανάπτυξη του IoT (Internet of Things), τα δεδομένα μεταφέρονται σε μια επιχείρηση με ασύλληπτες ταχύτητες τα οποία και πρέπει να διευθετηθούν σε ένα εύλογο χρονικό περιθώριο.

3) Ποικιλομορφία

Τα δεδομένα συναντώνται σε πολλούς τύπους και μορφές. Δομημένα, αριθμητικά, έγγραφα, emails, συναλλαγές κ.ο.κ. Παλαιότερα, τα δεδομένα που συλλέγονταν, παραδίδονταν σε μορφή αρχείου βάσης δεδομένων (excel, csv). Πλέον παρουσιάζονται σε μορφές όπως βίντεο, κείμενο, pdf, γραφικά κλπ. κάτι το οποίο προϋποθέτει εκτενέστερη επεξεργασία.

Έκτοτε, κι άλλοι επιστήμονες έχουν αποδώσει επιπλέον ιδιότητες στον όρο, συμπεριλαμβανομένων:

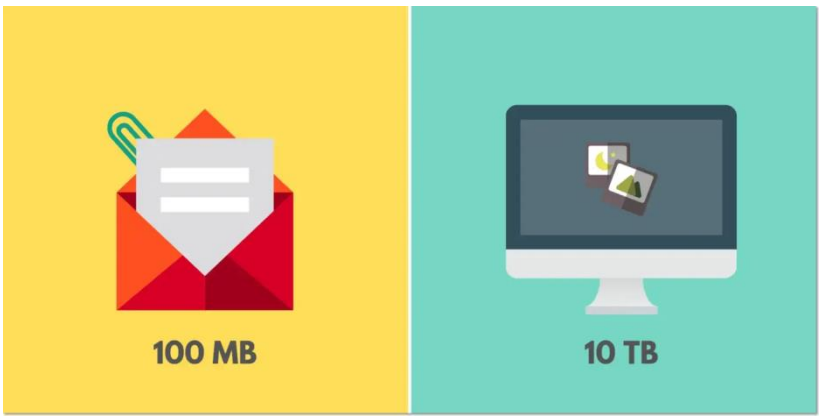
4) Εξαντλητικότητα (Exhaustivity): Καταγραφή ενός ολόκληρου συστήματος αντί για δειγματοληψία (Mayer-Schonberger και Cukier, 2013).

5) Λεπτή δομή (Fine-grained) (όσον αφορά την ανάλυση) και μοναδικού ευρετηρίου (όσον αφορά την ταυτοποίηση (Dodge and Kitchin, 2005).

6) Σχετικότητα (Relationality): Περιέχονται κοινά πεδία τα οποία επιτρέπουν την συνένωση διαφορετικών συνόλων δεδομένων (Boyd και Crawford, 2012).

7) Επεκτασιμότητα (Extensionality): Μπορούν να προστεθούν / αλλάξουν εύκολα νέα πεδία και να επεκταθούν γρήγορα σε μέγεθος (Marz και Warren, 2012).
8) Ακρίβεια (Veracity): Τα δεδομένα μπορεί να είναι ακατάστατα, θορυβώδη και να περιέχουν ανακριβεια και σφάλμα (Marr, 2014).
9) Τιμή (value): Μπορούν να εξαχθούν πολλές πληροφορίες και τα δεδομένα να επαναχρησιμοποιηθούν (Marr, 2014). ([113])

Πόσο ογκώδη όμως πρέπει να είναι τα δεδομένα για να κατηγοριοποιηθούν με τον όρο "Μεγάλα Δεδομένα";
Συνήθως τα δεδομένα τα οποία καταλαμβάνουν gigabytes και πάνω, θεωρούνται μεγάλα δεδομένα.



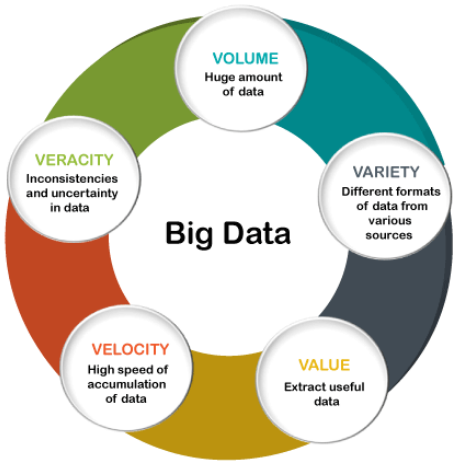
Εικόνα 1.3: "Μικρά" δεδομένα και "Μεγάλα" δεδομένα.

Υπάρχουν όμως και φορές, όπου ακόμη και μικρότερου όγκου δεδομένα μπορούν να χαρακτηριστούν ως μεγάλα δεδομένα. Για παράδειγμα, αν προσπαθήσουμε να επισυνάψουμε ένα έγγραφο μεγαλύτερο των 100 megabytes σε ένα email δεν θα μπορέσουμε να το καταφέρουμε. Κι αυτό διότι τα συστήματα email δεν υποστηρίζουν αρχεία μεγαλύτερα του ορίου αυτού. Άρα λοιπόν σύμφωνα με τους περιορισμούς του email, ο όγκος αυτός μπορεί να θεωρηθεί ως "Μεγάλα Δεδομένα.

Σε ένα ακόμη παράδειγμα, ας υποθέσουμε πως πρέπει να επεξεργαστούμε 10 terabytes αρχείων εικόνας, αλλάζοντας το μέγεθός τους και βελτιώνοντάς τις μέσα σε ένα εύλογο χρονικό διάστημα. Χρησιμοποιώντας ένα desktop σύστημα Η/Υ ενδεχομένως να μην μπορούσαμε να το καταφέρουμε αυτό εντός λογικού χρονικού ορίου, οπότε αυτό θα απαιτούσε την χρήση ενός δυνατότερου μηχανήματος, όπως

ένας server με δυνατές επιδόσεις. Ο όγκος αυτός λοιπόν των αρχείων εικόνας μπορεί να χαρακτηριστεί αντιστοίχως ως “Μεγάλα Δεδομένα”.

Με την εμφάνιση του όρου “Μεγάλα Δεδομένα” ακολούθησαν δύο μεγάλες προκλήσεις που συνεπάγονται στον όρο αυτό. Η πρώτη πρόκληση έχει να κάνει με το πώς αποθηκεύουμε και διαχειριζόμαστε τόσο μεγάλο όγκο δεδομένων επαρκώς. Η δεύτερη με το πώς επεξεργαζόμαστε και συλλέγουμε πληροφορίες από έναν τόσο μεγάλο όγκο δεδομένων μέσα σε συγκεκριμένο χρονικό διάστημα. ([19]), ([20]), ([21]), ([22])



Εικόνα 1.3.1: Χαρακτηριστικά των Μεγάλων Δεδομένων.

1.4 ΤΟ ΛΟΓΙΣΜΙΚΟ APACHE SPARK

Οι προκλήσεις που έθεσε στο τραπέζι ο όρος “Μεγάλα Δεδομένα”, οδήγησαν στην έρευνα και ανάπτυξη μεθόδων για τον έλεγχο και τη διαχείριση της τεράστιας ποσότητας δεδομένων που διακινούνται στον αχανή πλέον κόσμο του διαδικτύου. Γυρίζοντας τον χρόνο πίσω στο 2009, στο εργαστήριο AMPlab του Πανεπιστημίου της Καλιφόρνια στο Μπέρκλεϋ, έλαβε χώρα ένα project, δημιουργός του οποίου ήταν αρχικά ο φοιτητής διδακτορικού Matei Zaharia και νυν συνιδρυτής της αμερικάνικης εταιρείας επιχειρηματικού λογισμικού Databricks. Επακόλουθο του project αυτού ήταν η γέννηση ενός λογισμικού ανοιχτού κώδικα που θα έφερνε ριζικές αλλαγές στον τρόπο επεξεργασίας τεράστιου όγκου δεδομένων.

Το Spark μετά την δημιουργία του εξελίχθηκε σε μία ευρεία κοινότητα προγραμματιστών. Η αρχική ομάδα του εργαστηρίου AMPlab ξεκίνησε επίσης μια start-up εταιρεία ονόματι Databricks για την θωράκιση του project, το οποίο μεταφέρθηκε στο Ίδρυμα λογισμικού Apache (Apache Software Foundation) το 2013. Σήμερα, το project αυτό αναπτύσσεται συνεργατικά από μια κοινότητα εκατοντάδων προγραμματιστών από εκατοντάδες οργανισμούς.

Η έκδοση 1.0 του Spark κυκλοφόρησε το 2014. Μέχρι και την στιγμή της συγγραφής της παρούσας διπλωματικής εργασίας είναι η πιο ενεργά αναπτυσσόμενη μηχανή ανοιχτού κώδικα για επεξεργασία μεγάλων δεδομένων.

Τι είναι όμως το πλέον διαδεδομένο λογισμικό ως Apache Spark;

Το λογισμικό Apache Spark (ή Spark για συντομία) είναι μία ενοποιημένη υπολογιστική μηχανή με βιβλιοθήκες για παράλληλη επεξεργασία σε υπολογιστικές συστάδες, η οποία χρησιμοποιείται για την διαχείριση δεδομένων, την επιστήμη δεδομένων και την μηχανική μάθηση. Έχει σχεδιαστεί για να παρέχει την απαραίτητη υπολογιστική ισχύ, ταχύτητα, επεκτασιμότητα και δυνατότητα προγραμματισμού που απαιτείται για τα μεγάλα δεδομένα.

Το Spark είναι σχεδιασμένο για να υποστηρίζει μια ευρεία γκάμα από εργασίες ανάλυσης δεδομένων, όλες χρησιμοποιώντας την ίδια υπολογιστική μηχανή, η οποία επιτρέπει στις εφαρμογές να τρέχουν γρηγορότερα κάνοντας χρήση της μνήμης εντός μιας συστάδας υπολογιστών. Συστάδα είναι μια συλλογή κόμβων που επικοινωνούν μεταξύ τους και διαμοιράζουν δεδομένα. Κατά τη λειτουργία του, επεκτείνεται με το να διαμοιράζει φόρτο προς επεξεργασία ανάμεσα στους κόμβους με ενσωματωμένη δυνατότητα παραλληλισμού και ανοχής σε σφάλματα.

Επίσης, περιλαμβάνει APIs (Application Programming Interfaces) για δημοφιλείς ως προς τους αναλυτές δεδομένων γλώσσες προγραμματισμού. Οι βιβλιοθήκες του, υποστηρίζουν μια πληθώρα εφαρμογών από SQL έως ροές δεδομένων (streaming) και αλγορίθμους μηχανικής μάθησης (machine learning) ενώ ταυτόχρονα μπορεί να τρέξει οπουδήποτε, από laptop μέχρι σε συστάδα χιλιάδων servers. ([24]), ([23])

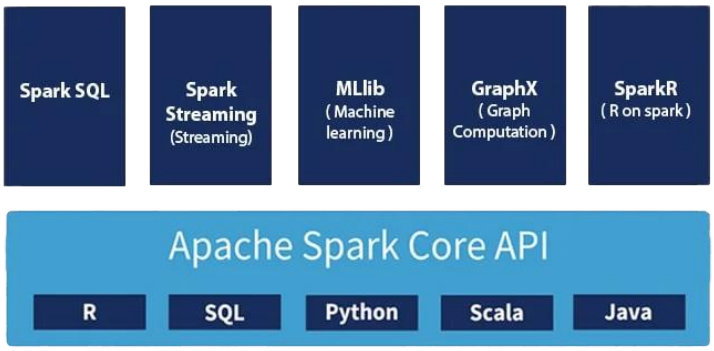
Εικόνα 1.4: Πλατφόρμες με της οποίες συνεργάζεται το λογισμικό Apache Spark.

1.4.1 Χαρακτηριστικά του Apache Spark

- **Ταχύτητα:** Το Spark αποδίδει έως και 100 φορές ταχύτερα σε σχέση με προηγούμενα μοντέλα, όπως το MapReduce στην επεξεργασία τεράστιων όγκων δεδομένων. Διαθέτει επίσης την ικανότητα διαίρεσης των δεδομένων σε κομμάτια με ελεγχόμενο τρόπο.
- **Ισχυρό Caching:** Προσφέρεται δυναμική χρήση της κρυφής μνήμης και δυνατότητες επιμονής των δεδομένων στο δίσκο μέσω απλού προγραμματισμού.
- **Ποικιλία τρόπων ανοίγματος:** Υπάρχει η δυνατότητα χρήσης διαφόρων διαχειριστών συστάδας όπως Mesos, Yarn και Spark Cluster Manager για την εκκίνηση του Spark.
- **Υπολογισμοί σε πραγματικό χρόνο:** Εξαιτίας της επεξεργασίας στη μνήμη, προσφέρει υπολογισμούς σε πραγματικό χρόνο και χαμηλή καθυστέρηση.
- **Πολυγλωσσικό:** Υποστηρίζει τις γλώσσες Java, Scala, Python και R, καθώς επίσης και διεπαφή κονσόλας σε Scala και Python.

  ([26])

1.4.2 Τα δομικά στοιχεία του Apache Spark



Εικόνα 1.4.1: Τα κύρια στοιχεία που δομούν το Apache Spark.

Το Spark διαθέτει 6 δομικά στοιχεία στο οικοσύστημά του. Τον Πυρήνα Spark και τα Spark SQL, Spark Streaming, Mlib και GraphX που ενεργούν πάνω από τον πυρήνα. ([25]), ([26]), ([27]), ([28]), ([29])

1) Πυρήνας Spark

Όπως παραπέμπει και η ονομασία, πρόκειται για την κύρια μονάδα της λειτουργίας του Spark. Αναλαμβάνει τον προγραμματισμό των εργασιών, την ανάκτηση από τυχόν λάθη, την διαχείριση μνήμης και τις λειτουργίες εισόδου – εξόδου. Μπορούμε να παρομοιάσουμε την λειτουργία του με εκείνη που εκτελεί ο επεξεργαστής (CPU) σε έναν υπολογιστή. Παρέχει APIs για της υποστηριζόμενες γλώσσες προγραμματισμού, ενώ όλα τα υπόλοιπα στοιχεία του Spark έχουν τα δικά τους APIs που είναι χτισμένα πάνω στον Πυρήνα.

Στον Πυρήνα Spark, γίνεται χρήση κάποιων δομημένων APIs, που αποτελούνται από τα RDDs, τα DataFrames και τα DataSets. Τα DataFrames και τα DataSets είναι σχεδιασμένα και βελτιστοποιημένα για να λειτουργούν με δομημένα δεδομένα, ενώ τα RDDs μπορούν να χειριστούν τόσο δομημένα όσο και μη δομημένα δεδομένα.

- RDD

Το RDD είναι μια αμετάβλητη, ανεκτική σε σφάλματα, συλλογή δεδομένων που κατανέμεται σε πολλούς υπολογιστικούς κόμβους που μπορούν να χρησιμοποιηθούν παράλληλα με μια χαμηλού επιπέδου προγραμματιστική διεπαφή (API) που προσφέρει μετασχηματισμούς (transformations) και δράσεις (actions). Ανεκτική σε σφάλματα (fault tolerant) σημαίνει ότι μπορεί να ανακάμψει αυτόματα από αποτυχίες, ενώ αμετάβλητη (immutable) σημαίνει ότι όταν δημιουργηθεί ένα RDD, δεν μπορούμε να το αλλάξουμε. Μετά την εφαρμογή ενός μετασχηματισμού σε ένα RDD προκύπτει ένα νέο RDD. Η κοινή χρήση δεδομένων στη μνήμη κάνει τα RDD 10-100 φορές ταχύτερα από την κοινή χρήση δίσκων και δικτύων. Είναι μια δομή δεδομένων "χωρίς σχήμα" (schema-less).

- Directed Acyclic Graph (DAG)

Το Spark δημιουργεί ένα Κατευθυνόμενο Ακυκλικό Γράφημα (Directed Acyclic Graph - DAG) για να προγραμματίσει εργασίες και να οργανώσει τους κόμβους εργάτες σε όλη τη συστάδα. Το DAG είναι ένα πεπερασμένο κατευθυνόμενο γράφημα χωρίς κατευθυνόμενους κύκλους. Υπάρχουν πολλές κορυφές και ακμές, όπου κάθε ακμή κατευθύνεται από την μία κορυφή έως την άλλη. Περιέχει μια συνοχή από κορυφές έτσι ώστε κάθε ακμή κατευθύνεται από μια προηγούμενη θέση στην επόμενη εντός της συνοχής.

Η παρακολούθηση αυτή των εργασιών στους κόμβους εργάτες κάνει εφικτή την ανοχή σε σφάλματα καθώς επανεφαρμόζει τις καταγεγραμμένες ενέργειες στα δεδομένα από προηγούμενη κατάσταση.

Ο Διοργανωτής (Scheduler) χωρίζει τα RDDs σε στάδια βασισμένα σε διάφορες διαμορφώσεις που έχουν υποστεί. Στην επισκόπηση σταδίου, οι λεπτομέρειες από όλα τα RDDs που ανήκουν στο συγκεκριμένο στάδιο, επεκτείνονται κι έτσι ο χρήστης έχει τη δυνατότητα να εμβαθύνει σε αυτές. Κάθε στάδιο αποτελείται από εργασίες βασισμένες στις διχοτομήσεις (partitions) των RDDs που θα εκτελέσουν τον ίδιο υπολογισμό παράλληλα. Η λέξη “γράφημα” αναφέρεται στην πλοήγηση ενώ οι λέξεις “κατευθυνόμενο” και “ακυκλικό” αναφέρονται στον τρόπο με τον οποίο γίνεται.

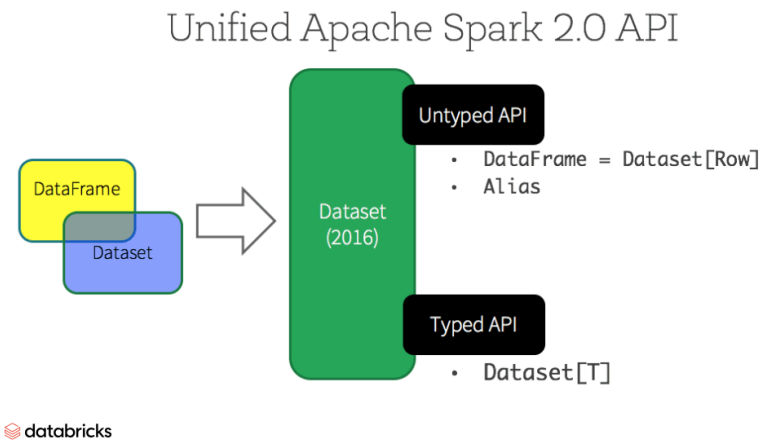
- DataFrame

Το DataFrame εισήχθη με την έκδοση 1.3 του Spark και είναι μια προγραμματιστική διεπαφή (API) υψηλότερου επιπέδου σε που είναι δομημένη πάνω από το RDD. Προσφέρει έναν πιο αποτελεσματικό τρόπο εργασίας με δομημένα και ημιδομημένα σύνολα δεδομένων όπου κάθε εγγραφή είναι μια σειρά αποτελούμενη από ένα σύνολο στηλών, και κάθε στήλη έχει έναν καθορισμένο τύπο δεδομένων. Η έννοια του DataFrame είναι γνωστή από τις γλώσσες Python και R όπου και αναφέρεται σε δεδομένα πίνακα με προγραμματιστικές μεθόδους για φιλτράρισμα, υπολογισμό νέων στηλών και συνάθροιση. Στο Spark τα δεδομένα αυτά είναι κατανεμημένα και οι υπολογισμοί σε αυτά γίνονται μέσω της μηχανής Spark SQL. Τα DataFrames είναι επίσης γνωστά για την ικανότητά τους να εκτελούν κατανεμημένη επεξεργασία σε μία συστάδα μηχανών, επιτρέποντας επεκτάσιμη επεξεργασία δεδομένων.

- DataSet

Το DataSet αφορά μόνο τις γλώσσες Java και Scala. Είναι μια κατανεμημένη συλλογή αντικειμένων JVM που παρέχει ασφάλεια τύπων (type safe) σε αντίθεση με το DataFrame. Σε αντίθεση με τα RDD που χρησιμοποιούν Java σειριοποίηση για σειριοποίηση αντικειμένων τα DataSet χρησιμοποιούν τον μηχανισμό σειριοποίησης του Encoder που επιτρέπει στο Spark να εκτελεί διάφορες πράξεις όπως φιλτράρισμα, ταξινόμηση, hashing χωρίς αποσειριοποίηση όπως στους κανονικούς μηχανισμούς σειριοποίησης. Προσφέρει τη δυνατότητα πράξεων πάνω σε κλάσεις οριζόμενες από τον χρήστη (user defined). Έτσι για παράδειγμα αφού ορίσει ο χρήστης του Spark τον δικό του τύπο δεδομένων και εκτελέσει τους χειρισμούς του, το Spark μπορεί αυτόματα να το μετατρέψει σε DataFrame και να το χειριστεί ο χρήστης περαιτέρω χρησιμοποιώντας τις εκατοντάδες συναρτήσεις που περιλαμβάνει το Spark. Τα δυο APIs έχουν ενοποιηθεί στην έκδοση 2.0 του Spark στη Scala και στη Java και το DataFrame θεωρείται ψευδώνυμο του Dataset [Row] στη Scala.

([28]), ([29])



Εικόνα 1.4.2: Η ένωση (merge) των DataFrame APIs με τα DataSets APIs που εφαρμόστηκε στην έκδοση 2.0 και στις μετέπειτα εκδόσεις.

2) Spark SQL

Το Spark SQL περιλαμβάνει υποστήριξη για SQL στο Spark και απλοποιεί τη διαδικασία αναζήτησης δεδομένων που είναι αποθηκευμένα τόσο σε RDD (κατανεμημένα σύνολα δεδομένων του Spark) όσο και σε εξωτερικές πηγές. Ασχολείται κυρίως με την διερεύνηση, ανάλυση και επεξεργασία δομημένων δεδομένων. Ενοποιεί τις γραμμές μεταξύ δομημένων APIs του Πυρήνα Spark και σχεσιακών πινάκων. Η ενοποίηση αυτών διευκολύνει τους προγραμματιστές να αναμειγνύουν εντολές SQL με πολύπλοκα αναλυτικά στοιχεία, όλα μέσα σε μία μόνο εφαρμογή.

Ο χρήστης του Spark, μπορεί να εκτελέσει queries (ερωτήματα) σε δομημένα δεδομένα χρησιμοποιώντας είτε τη δομημένη γλώσσα ερωτημάτων SQL είτε το DataFrame/RDD/DataSet API. Ενεργώντας ως κατανεμημένη μηχανή ερωτημάτων, επιτρέπει τα ερωτήματα σε πίνακες Hive και αρχεία parquet να τρέχουν έως και 100 φορές γρηγορότερα καθώς και την εκτέλεση ερωτημάτων SQL πάνω σε εισαγόμενα δεδομένα και RDDs χωρίς καμία τροποποίηση.

Τέλος, ο χρήστης μπορεί να γράψει και να διαβάσει δεδομένα σε διάφορα μορφότυπα (formats) και συστήματα αποθήκευσης (json, csv, parquet, orc, Avro, Hive, διάφορες πηγές RDBMS, κ.λπ.). ([30]), ([31])

3) Spark Streaming

Το Spark Streaming είναι μια επέκταση της προγραμματιστικής διεπαφής εφαρμογών (API) του Πυρήνα Spark η οποία επιτρέπει την υψηλής απόδοσης και ανεκτική σε σφάλματα επεξεργασία ροής δεδομένων πραγματικού χρόνου. Τα δεδομένα μπορούν να συγχωνευτούν από διάφορες πηγές όπως η πλατφόρμα επεξεργασίας ροών δεδομένων Apache Kafka, το λογισμικό συλλογής και μετακίνησης μεγάλων ποσοτήτων δεδομένων καταγραφής Apache Flume, η υπηρεσία ροών δεδομένων πραγματικού χρόνου Kinesis της Amazon, οι θύρες TCP και επεξεργάζονται με περίπλοκους αλγορίθμους με υψηλού επιπέδου λειτουργίες διαχείρισης δεδομένων, όπως οι map, reduce, join και window.

Τέλος, τα επεξεργασμένα δεδομένα μπορούν να αποσυρθούν σε συστήματα αρχείων, βάσεις δεδομένων και ζωντανών πινάκων διαχείρισης. Το σημαντικό γεγονός έγκειται στη δυνατότητα ενσωμάτωσης των αλγορίθμων μηχανικής μάθησης και γραφημάτων του Spark σε ροές δεδομένων.

Εικόνα 1.4.3: Χρήση του στοιχείου Spark Streaming.

Εσωτερικά, το Spark Streaming δέχεται ζωντανά δεδομένα εισόδου και τα διαχωρίζει σε παρτίδες, οι οποίες έπειτα επεξεργάζονται από την μηχανή του Spark ούτως ώστε να παραχθεί η τελική ροή των αποτελεσμάτων σε παρτίδες. Αυτή η υψηλού επιπέδου μέθοδος φιλτραρίσματος της σημαντικής πληροφορίας ονομάζεται διακριτοποιημένη ροή (discretized stream) ή αλλιώς DStream. Με τον όρο “διακριτοποίηση” εννοούμε την διαδικασία μετατροπής ενός συνεχούς εύρους τιμών σε διακριτές τιμές. Αυτή η διαδικασία συνήθως πραγματοποιείται ως ένα πρώτο βήμα για να γίνουν κατάλληλες για αριθμητική αξιολόγηση και εφαρμογή σε ηλεκτρονικούς υπολογιστές.

Το DStream είναι χτισμένο πάνω στην λογική του RDD και αυτό εξασφαλίζει στο Spark Streaming την ομαλή λειτουργία με τα υπόλοιπα στοιχεία του Spark. Μερικά παραδείγματα ροών δεδομένων που μπορούμε συχνά να συναντήσουμε στην καθημερινότητα είναι οι υπηρεσίες Netflix, Pinterest και Uber. Είναι πολύ πιθανό επίσης να έχουμε κάνει και χρήση μιας τουλάχιστον από τις προαναφερθείσες υπηρεσίες. Το Spark Streaming μπορεί να ενσωματωθεί με το Kafka που είναι μια πλατφόρμα αυτόνομης προσωρινής αποθήκευσης για ροές εισόδου. Το Kafka ενεργεί ως το κεντρικό σημείο για τις ροές πραγματικού χρόνου που έχουν υποστεί επεξεργασία με αλγορίθμους στο Spark Streaming. ([32])

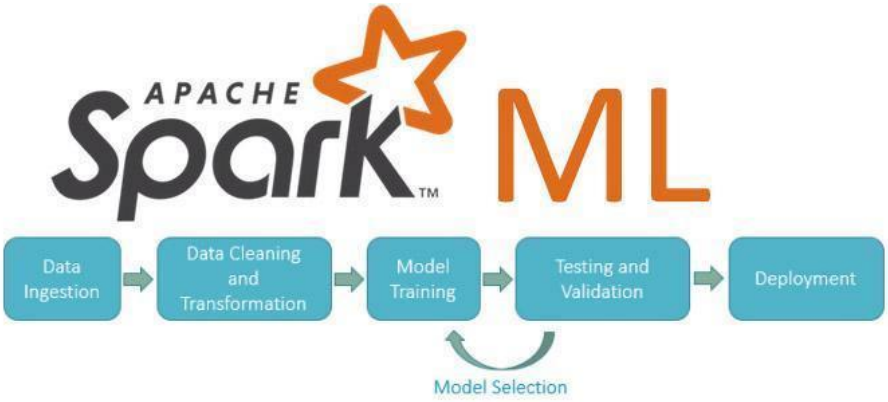
Εικόνα 1.4.4: Λειτουργία του στοιχείου Spark Streaming.

4) Spark MLlib

Η Βιβλιοθήκη Μηχανικής Μάθησης του Spark (MLlib ή Machine Learning library) έχει σχεδιαστεί για απλότητα, επεκτασιμότητα και ευκολία στην ενσωμάτωση με άλλα εργαλεία.

Χτισμένη πάνω από το Spark, Η Βιβλιοθήκη Μηχανικής Μάθησης περιλαμβάνει κοινούς αλγορίθμους μάθησης και εργαλεία, συμπεριλαμβανομένων της κατηγοριοποίησης (classification), της παλινδρόμησης (regression), συσταδοποίησης (clustering), συνεργατικού φιλτραρίσματος (collaborative filtering), μείωσης διαστασιμότητας (dimensionality reduction) αποτελώντας την βάση για την βελτιστοποίηση των επιπέδων των νευρωνικών δικτύων. Η Βιβλιοθήκη Μηχανικής Μάθησης του Spark ενσωματώνεται απρόσκοπτα με τα υπόλοιπα στοιχεία του Spark, όπως το Spark SQL, το Spark Streaming και τα DataFrames. Η βιβλιοθήκη είναι χρησιμοποιήσιμη σε γλώσσα Java, Scala και Python ως μέρους των εφαρμογών Spark ούτως ώστε ο χρήστης να μπορεί να την συμπεριλάβει σε μία ολοκληρωμένη ροή εργασίας.

Η Βιβλιοθήκη Μηχανικής Μάθησης επιτρέπει την προεπεξεργασία, την μετατροπή και την εκπαίδευση των μοντέλων και πραγματοποιώντας προβλέψεις σε δομημένη ροή. Το Spark παρέχει μια εξεζητημένη διεπαφή προγραμματισμού εφαρμογών (API) για την εκτέλεση μιας πληθώρας από εργασίες μηχανικής μάθησης, από κατηγοριοποίηση σε παλινδρόμηση, συσταδοποίηση σε βαθιά μάθηση. ([33])



Εικόνα 1.4.5: Βήματα λειτουργίας του στοιχείου Spark ML.

5) Spark GraphX

Το GraphX είναι ένα διαμοιρασμένο πλαίσιο επεξεργασίας γραφημάτων. Είναι βασισμένο στην πλατφόρμα Spark και παρέχει μία απλή, εύχρηστη και πλούσια διεπαφή για υπολογισμούς και εξορύξεις γραφημάτων, η οποία διευκολύνει την ανάγκη για διαμοιρασμένη επεξεργασία γραφημάτων. Υπάρχουν πολλές σχεσιακές αλυσίδες μεταξύ ανθρώπων και οντοτήτων στα κοινωνικά δίκτυα, ειδικά στις μεγάλες εταιρείες κοινωνικών δικτύων. Οι σχέσεις αυτές απαιτούν γραφικούς υπολογισμούς.

Το GraphX είναι μια νέα διεπαφή προγραμματισμού εφαρμογών (API) του Spark και χρησιμοποιείται για τον υπολογισμό γραφημάτων και διαμοιρασμένων γραφημάτων (παραλληλισμός γραφημάτων). Το GraphX επεκτείνει την δομή δεδομένων RDD με την εισαγωγή του Γραφήματος Ανθεκτικής Κατανεμημένης Ιδιοκτησίας (RDPG) το οποίο είναι ένα ελεγχόμενο πολλαπλό γράφημα με κορυφές και άκρες που κατέχουν ιδιότητες.

Για την υποστήριξη των γραφηματικών υπολογισμών το GraphX έχει αναπτύξει ένα βασικό σετ από λειτουργικές διεργασίες και ένα βελτιστοποιημένη διεπαφή προγραμματισμού εφαρμογών Pregel (API). Επιπλέον, το GraphX

περιλαμβάνει μια ταχεία αναπτυσσόμενη συλλογή από γραφηματικούς αλγορίθμους (graph algorithms) και γραφηματικούς χτίστες (graph builders) για την απλοποίηση της ανάλυσης γραφημάτων. ([34])

6) SparkR

Η R είναι η ευρέως χρησιμοποιούμενη γλώσσα για στατιστικές αναλύσεις, αποτελούμενη από περισσότερα από 10000 πακέτα για διαφορετικούς σκοπούς. Χρησιμοποιεί την προγραμματιστική διεπαφή των DataFrames η οποία παρέχει οπτικοποίηση των δεδομένων έτσι ώστε να διευκολύνεται η δουλειά των αναλυτών. Ωστόσο, η R δεν υποστηρίζει παράλληλη επεξεργασία και περιορίζεται στην διαθέσιμη ποσότητα μνήμης μιας μηχανής. Για τέτοιες περιπτώσεις δημιουργήθηκε το SparkR.

Το SparkR είναι ένα εργαλείο εκτέλεσης της γλώσσας R στο Spark. Για τη χρήση του SparkR το μόνο που απαιτείται είναι η εισαγωγή του στο περιβάλλον πριν την εκτέλεση του κώδικα. Ο τρόπος χρήσης είναι παρόμοιος με την προγραμματιστική διεπαφή της γλώσσας Python. Ως εκ τούτου, το μεγαλύτερο μέρος των εντολών που είναι διαθέσιμες στην Python είναι και στο SparkR. ([27])

1.4.3 Η Αρχιτεκτονική του Apache Spark

Σε αυτό το υποκεφάλαιο θα επικεντρωθούμε στην αρχιτεκτονική του Spark. Όπως έχει ήδη ειπωθεί το Spark είναι μια κατανεμημένη υπολογιστική μηχανή που βασίζεται στην μνήμη.

Το Spark δεν διαθέτει το δικό του κατανεμημένο αποθηκευτικό σύστημα, εξαρτάται από διάφορα αποθηκευτικά συστήματα για κατανεμημένους υπολογισμούς. Μερικά από τα οποία είναι:

- Τοπικό Σύστημα Αρχείων (Local File System): Αυτό είναι ένα προεγκατεστημένο σύστημα αρχείων που χρησιμοποιείται από τον Spark όταν τρέχει σε τοπική λειτουργία. Είναι κατάλληλο για δοκιμές και εφαρμογή, αλλά όχι για παραγωγική χρήση.
- Amazon Simple Storage Service (S3): Για μη επείγουσες μαζικές εργασίες. Το S3 ταιριάζει σε πολύ συγκεκριμένες περιπτώσεις, όπου η τοπική θέση των δεδομένων δεν είναι αναγκαία.

- Cassandra: Ιδανικό για ανάλυση σε ροές δεδομένων αλλά ιδιαίτερα επιβαρυντικό για μαζικές εργασίες.
- HDFS (Hadoop Distributed File System): Η πλατφόρμα Apache Hadoop αποτελεί ένα πλαίσιο για κατανεμημένη αποθήκευση. Ταιριάζει πολύ σε περιπτώσεις μαζικών εργασιών.

Το Spark μπορεί να τρέξει με διάφορους τρόπους, ανάλογα με την εκάστοτε περίπτωση χρήσης. Παρακάτω παρουσιάζονται ορισμένοι από τους πιο συνήθεις:

- Τοπική λειτουργία: Το Spark τρέχει σε μία συγκεκριμένη μηχανή ή κόμβο, χρησιμοποιώντας το τοπικό σύστημα αρχείων ως πηγή δεδομένων. Αυτή η λειτουργία είναι χρήσιμη για ανάπτυξη και δοκιμή, καθώς δεν απαιτεί διανεμημένη συστάδα.
- Αυτοτελής λειτουργία: Σε αυτή την περίπτωση, το Spark τρέχει σε μία αφοσιωμένη συστάδα μηχανών, με έναν αρχηγό κόμβο και μερικούς κόμβους εργάτες. Αυτή η λειτουργία είναι κατάλληλη για μικρές έως μεσαίου μεγέθους συστάδες και μπορεί εύκολα να εγκατασταθεί και να ρυθμιστεί.
- Apache Mesos: Το Mesos είναι ένας διαχειριστής συστάδας που χρησιμοποιείται για να τρέξει το Spark και άλλες διανεμημένες εφαρμογές. Ο Mesos παρέχει απομόνωση και διανομή των πόρων, επιτρέποντας σε πολλαπλά πλαίσια να τρέξουν στην ίδια συστάδα.
- Hadoop YARN: Το YARN αποτελεί τον διαχειριστή των πόρων στο Hadoop και το Spark μπορεί να τρέξει σαν μία εφαρμογή YARN, επιτρέποντας του την αξιοποίηση των πόρων σε μία συστάδα Hadoop.
- Kubernetes: Είναι μία πλατφόρμα ανοιχτού κώδικα, η οποία οργανώνει τα κιβώτια (containers) και μπορεί να χρησιμοποιηθεί με σκοπό να τρέξει το Spark σε ένα κιβωτιοποιημένο περιβάλλον. Αυτή η λειτουργία επιτρέπει στο Spark να εκμεταλλεύεται την δυνατότητα επέκτασης και ελαστικότητας των κιβωτιοποιημένων αναπτύξεων (deployments).
  ([35]), ([36]), ([59]), ([61]), ([98])

Προτού προχωρήσουμε, θα είναι καλό να αναλυθούν κάποιες γενικές βασικές έννοιες και κάποιες σχετικές με το λογισμικό Hadoop του Apache Software Foundation:

Ορισμός	**Επεξήγηση**
Κόμβος	Ονομάζεται ένα μηχάνημα που διαθέτει κάποιος φυσικούς πόρους όπως επεξεργαστική ισχύς και χωρητικότητα μνήμης.
Συστάδα	Σύνολο από κόμβους οι οποίοι επικοινωνούν μεταξύ τους.
Κύριος Κόμβος ή Κόμβος Αρχηγός	Ο κύριος κόμβος είναι ο κόμβος που συνήθως ενορχηστρώνει/συντονίζει/επιβλέπει τις διαδικασίες και τους ρόλους στους άλλους κόμβους.
Κόμβος Εργάτη	Ο κόμβος εργάτη αναφέρεται στον κόμβο που εκτελεί τον κώδικα εφαρμογής. Ο κόμβος εργάτη επεξεργάζεται τα δεδομένα που είναι αποθηκευμένα στον κόμβο και αναφέρει τους πόρους στον κύριο κόμβο. Βάσει της διαθεσιμότητας πόρων, ο κύριος κόμβος προγραμματίζει τις εργασίες.
Ενέργειες Spark (Actions)	Με τον όρο ενέργειες Spark εννοούμε τις εργασίες που οδηγούν στην εκτέλεση μετασχηματισμών δεδομένων στα RDDs ή DataFrames, επιστρέφοντας μία τιμή. Όταν μία ενέργεια εκτελεστεί, το Spark επαληθεύει τους μετασχηματισμούς που έχουν εφαρμοστεί στα δεδομένα και επιστρέφουν αποτέλεσμα. Οι ενέργειες Spark χρησιμοποιούνται για να γραφούν δεδομένα στο δίσκο, για να παρουσιαστούν δεδομένα στην κονσόλα, για να συλλεχθούν δεδομένα στο πρόγραμμα οδηγού ή να εκτελεστούν προσθέσεις. Μερικές από τις ενέργειες στο Spark είναι: count(), top(), και collect().
Εικονική Μηχανή Java (Java Virtual Machine - JVM)	Στο Hadoop, η JVM χρησιμοποιείται για την εκτέλεση διαφόρων στοιχείων του οικοσυστήματος Hadoop,

	συμπεριλαμβανομένων του NameNode, DataNode, ResourceManager, NodeManager και των MapReduce διεργασιών. Η JVM παρέχει ένα επίπεδο αφαίρεσης μεταξύ του λειτουργικού συστήματος και του λογισμικού Hadoop, επιτρέποντας στο Hadoop να είναι ανεξάρτητο από την πλατφόρμα. Η JVM διαχειρίζεται την εκτέλεση κώδικα Java, συμπεριλαμβανομένης της διαχείρισης μνήμης και της διαχείρισης νημάτων.
SparkContext	Το SparkContext είναι το σημείο εισόδου σε οποιαδήποτε λειτουργία Spark. Είναι η κύρια διεπαφή μεταξύ μιας συστάδας Spark και της εφαρμογής Spark. Είναι υπεύθυνο για τη ρύθμιση εσωτερικών υπηρεσιών και τη δημιουργία σύνδεσης με τον Διαχειριστή Συστάδας (Cluster Manager). Χρησιμοποιείται για την δημιουργία RDD (Resilient Distributed Datasets), συσσωρευτές και μεταβλητές εκπομπής.
SparkSession	Το SparkSession, είναι ένα API υψηλού επιπέδου που παρέχει μια πιο φιλική προς το χρήστη διεπαφή για αλληλεπίδραση με το Spark. Παρουσιάζεται στο Spark 2.0 και προορίζεται να αντικαταστήσει το προηγούμενο σημείο εισόδου, το SQLContext. Το SparkSession παρέχει πρόσβαση στη λειτουργικότητα του Spark, συμπεριλαμβανομένων των DataFrames, του Spark SQL, των Datasets και των API μηχανικής εκμάθησης. Περιλαμβάνει επίσης το SparkContext ως ένα από τα στοιχεία του.

Εργασία Spark (Spark Job) Στάδιο Spark (Spark Stage) Διεργασία Spark (Spark Task)	Στο Spark, μια εργασία αναφέρεται σε ένα σύνολο διεργασιών που υποβάλλονται για εκτέλεση σε μια συστάδα Spark. Μια εργασία ξεκινά από μια ενέργεια στο πρόγραμμα οδήγησης, όπως count(), collect() ή save(), η οποία ενεργοποιεί ένα σύνολο μετασχηματισμών που ορίζονται στο πρόγραμμα και που πρόκειται να εκτελεστούν στο στοιχείο RDD (Resilient Distributed Dataset). Κάθε ενέργεια καταλήγει σε ένα ή περισσότερα στάδια, τα οποία αποτελούνται από μία ή περισσότερες διεργασίες που εκτελούνται παράλληλα στους κόμβους εργάτες.

Παρακάτω θα δούμε τις κύριες λειτουργίες που τρέχουν στον κόμβο Αρχηγό και στους κόμβους Εργατών μια συστάδας με εγκατεστημένο το Spark μαζί με Hadoop.

1.4.4 Διεργασίες στον Κόμβο Αρχηγό (Master Node)

Σε επίπεδο Hadoop(HDFS):

NameNode (Κόμβος Ονομάτων): Στο Hadoop, το NameNode είναι ένα πρόγραμμα που τρέχει σε μια συγκεκριμένη μηχανή στην συστάδα HDFS (Hadoop Distributed File System) και είναι ένα κρίσιμο στοιχείο του Κατανεμημένου Συστήματος Αρχείων (HDFS). Είναι το κεντρικό κομμάτι της αρχιτεκτονικής HDFS και εξυπηρετεί ως κεντρική αποθήκη για την αποθήκευση μεταδεδομένων και κατάλογο πληροφοριών για τα αρχεία που είναι αποθηκευμένα στη συστάδα HDFS. Ο NameNode διατηρεί την ιεραρχία του ονομαστικού χώρου και ανιχνεύει την τοποθεσία κάθε συνόλου δεδομένων που αποθηκεύονται στη συστάδα HDFS. Επιπλέον, καταγράφει τον παράγοντα αναπαραγωγής (replication factor) για κάθε σύνολο και την ταυτότητα των DataNodes που αποθηκεύουν κάθε μπλοκ. Κάθε φορά που μια εφαρμογή πελάτης στέλνει μια αίτηση στον NameNode, εκείνος απαντά με

τις τοποθεσίες των μπλοκ που αποτελούν το αρχείο. Για να εξασφαλιστεί μεγάλη διαθεσιμότητα και ανοχή σε σφάλματα, το Hadoop παρέχει έναν μηχανισμό για την αντιγραφή των μεταδεδομένων του NameNode σε άλλη μηχανή, οι οποία ονομάζεται Δευτερεύον Κόμβος Ονομάτων (Secondary Namenode). Στην περίπτωση αποτυχίας του βασικού NameNode, ο δευτερεύον NameNode μπορεί να εξασφαλίσει τη συνεχή διαθεσιμότητα της συστάδας HDFS.

([42]), ([43]), ([44]), ([45]), ([46])

Σε επίπεδο Spark:

Spark Driver: Ο Οδηγός Spark (Spark Driver) είναι ένα πρόγραμμα ή διεργασία που τρέχει στον Κόμβο Αρχηγό και κατά την εκτέλεση μιας εφαρμογής Spark, δημιουργεί το Περιεχόμενο Spark (SparkContext). Το SparkContext είναι υπεύθυνο για τη δημιουργία και τη διαχείριση των κατανεμημένων υπολογιστικών πόρων, όπως οι κόμβοι εκτελεστών, που απαιτούνται για την εκτέλεση των εργασιών της εφαρμογής. Επίσης υποδηλώνει τους μετασχηματισμούς και τις ενέργειες στα RDDs δεδομένων και υποβάλει κάθε αίτημα στον Κόμβο Αρχηγό.

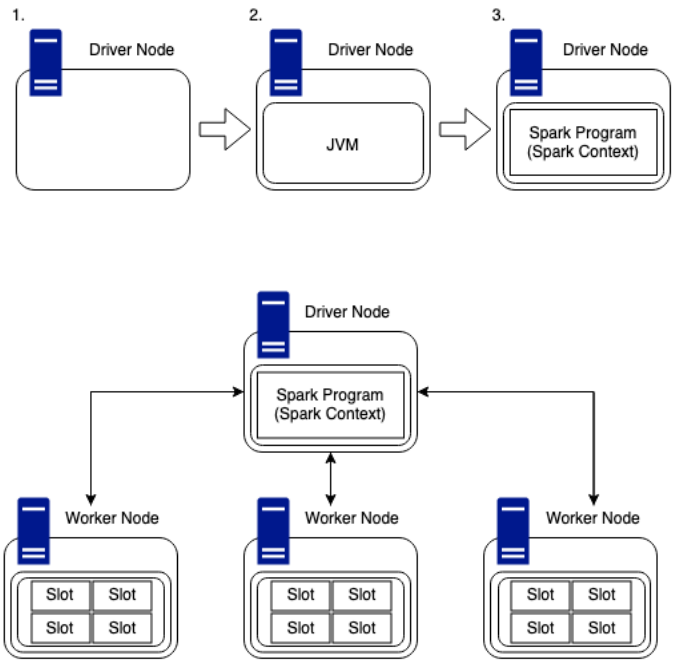
Ο Οδηγός Spark ακολουθεί μία ιεραρχική αρχιτεκτονική Αρχηγού-Εργάτη και ελέγχει τον Διαχειριστή Συστάδας (Cluster Manager) (Αυτόνομος Διαχειριστής Συστάδας Spark, Hadoop Yarn, Kubernetes ή Mesos), ο οποίος με τη σειρά του διαχειρίζεται τους κόμβους εργάτες παραδίδοντας τα αποτελέσματα των δεδομένων στην εφαρμογή πελάτη. Ο Οδηγός Spark χρησιμοποιείται για να διευθύνει ολόκληρη τη συστάδα Spark. Αυτό σημαίνει ότι διαχειρίζεται όλη τη δουλειά που έχει κατανεμηθεί σε όλη τη συστάδα καθώς επίσης και τις μηχανές που είναι διαθέσιμες.

Όταν ο χρήστης υποβάλει μια εφαρμογή ως εργασία Spark, το πρόγραμμα οδηγού ξεκινά μαζί με τις εκάστοτε ρυθμίσεις του. Ο οδηγός τρέχει τη μέθοδο main () της εφαρμογής και δημιουργεί το Περιεχόμενο Spark. Με βάση τον κώδικα της εφαρμογής, χρησιμοποιώντας το Περιεχόμενο Spark δημιουργούνται μετασχηματισμοί και ενέργειες. Μέχρι να γίνει κλήση μιας ενέργειας, όλοι οι μετασχηματισμοί πηγαίνουν στο Περιεχόμενο Spark με τη μορφή Κατευθυνόμενου Ακυκλικού Γράφου (DAG), όπου δημιουργείται ένα πλάνο εκτέλεσης (Λογικό Πλάνο και Φυσικό Πλάνο). Όταν γίνει κλήση της ενέργειας, δημιουργείται μια εργασία Spark με πολλαπλές διεργασίες. Ανάλογα με τις διεργασίες που θα δημιουργηθούν, ο

οδηγός δίνει εντολή στον Διαχειριστή Συστάδας (Cluster Manager) να δεσμεύσει τους εκτελεστές Spark (δηλαδή τους κόμβους Εργάτες) για την επεξεργασία των διεργασιών.

Όταν οι πόροι κατανεμηθούν, οι διεργασίες ξεκινούν από τον Διαχειριστή Συστάδας στους κόμβους εργάτες μαζί με τις ρυθμίσεις της εφαρμογής. Ο Οδηγός διαθέτει τα μεταδεδομένα των υποεργασιών που έχουν διαμοιραστεί με τους εκτελεστές. Όταν οι υποεργασίες ολοκληρωθούν απ' τους εκτελεστές, τα αποτελέσματα διαμοιράζονται πίσω με τον οδηγό. Ο Οδηγός Spark μπορεί να ρυθμιστεί ώστε να διαθέτει συγκεκριμένο αριθμό πυρήνων CPU. Από προεπιλογή, στον οδηγό εκχωρείται 1 πυρήνας αλλά αυτές οι τιμές μπορούν να αλλάξουν.

([37]), ([38])



Εικόνα 1.4.6 (πάνω) και 1.4.7 (κάτω): Η αρχιτεκτονική του κόμβου Οδηγού και των κόμβων Εργατών (Από τον Data Scientist Luke Thorp)

1.4.5 Διεργασίες στον Κόμβο Εργάτη (Worker Node)

Σε επίπεδο Hadoop (HDFS):

DataNode (Κόμβος Δεδομένων): Στο Hadoop, το DataNode είναι ένα ακόμη κρίσιμο στοιχείο του Κατανεμημένου Συστήματος Αρχείων (Hadoop Distributed File System - HDFS). Είναι ένα πρόγραμμα που τρέχει σε ξεχωριστά μηχανήματα σε μια συστάδα HDFS και είναι υπεύθυνο για την αποθήκευση και την προώθηση των δεδομένων σε μια συστάδα. Ένα σύστημα αρχείων Hadoop μπορεί να έχει πολλούς κόμβους δεδομένων αλλά μόνο έναν NameNode. Κάθε DataNode διαχειρίζεται την αποθήκευση των δεδομένων στο τοπικό του σύστημα αρχείων και επικοινωνεί με τον NameNode για να αναφέρει τον διαθέσιμο αποθηκευτικό χώρο, τα μπλοκ δεδομένων και τις τοποθεσίες αυτών. Όταν μια εφαρμογή - πελάτης θέλει να διαβάσει ή να γράψει σε ένα αρχείο, επικοινωνεί με τον NameNode για να αποκτήσει τα μεταδεδομένα και στη συνέχεια απευθύνεται στους DataNodes για να αποκτήσει πρόσβαση στα μπλοκ δεδομένων. Το DataNode έχει σχεδιαστεί για να λειτουργεί με αποκεντρωμένο τρόπο και η συστάδα μπορεί να κλιμακωθεί προς τα πάνω ή προς τα κάτω προσθέτοντας ή αφαιρώντας DataNodes. Το Hadoop έχει σχεδιαστεί για να αναπαράγει δεδομένα σε πολλούς DataNodes για να διασφαλίζει τη διαθεσιμότητα δεδομένων, την ανοχή σφαλμάτων και την ανθεκτικότητα των δεδομένων. Σε περίπτωση αποτυχίας του DataNode, το Namenode μπορεί να ανακατευθύνει αυτόματα αιτήματα ανάγνωσης και εγγραφής σε άλλους διαθέσιμους DataNodes στη συστάδα.

([42]), ([44]), ([45]), ([46])

Σε επίπεδο Spark:

Spark Executor: Οι Εκτελεστές Spark είναι οι κόμβοι Εργάτες από το διανεμημένο μέρος της συστάδας. Είναι υπεύθυνοι για την εκτέλεση των εργασιών που έχουν ανατεθεί από το πρόγραμμα Οδηγού Spark. Όταν υποβάλλεται μια εφαρμογή Spark, το πρόγραμμα οδηγού ζητά πόρους από τον Διαχειριστή Συστάδας, όπως το YARN ή το Mesos, για την εκκίνηση των εκτελεστών σε κόμβους εργατών. Κάθε εκτελεστής έχει τη δική του διαδικασία JVM και εκτελείται σε ξεχωριστό κόμβο στη συστάδα.

Οι εκτελεστές είναι υπεύθυνοι για την εκτέλεση των εργασιών που τους έχουν ανατεθεί από το πρόγραμμα οδήγησης. Κάθε εκτελεστής εκτελεί πολλές εργασίες ταυτόχρονα και αποθηκεύει τα δεδομένα στη μνήμη ή στο δίσκο ανάλογα με τις ρυθμίσεις διαμόρφωσης.

Το Spark επιτρέπει τη δυναμική κατανομή των εκτελεστών, πράγμα που σημαίνει ότι μπορούν να εκκινηθούν πρόσθετοι εκτελεστές κατά την εκτέλεση της εφαρμογής, εάν υπάρχουν περισσότερες εργασίες από αυτές που μπορούν να χειριστούν οι διαθέσιμοι εκτελεστές. Αυτή η δυνατότητα μπορεί να βοηθήσει στη βελτιστοποίηση της χρήσης πόρων και στη βελτίωση της απόδοσης της εφαρμογής. Ο κόμβος οδηγός μπορεί να χρησιμοποιήσει τις δυνατότητες διαχείρισης της συστάδας για να διαπιστώσουμε ποιο hardware είναι διαθέσιμο στον εργάτη.

Υπάρχει μία ποσότητα διαθέσιμης μνήμης η οποία μπορεί να διαιρεθεί σε δύο μέρη, στη μνήμη αποθήκευσης και την μνήμη εργασιών. Η προκαθορισμένη αναλογία τους είναι 50/50, αλλά αυτό μπορεί να αλλάξει στις ρυθμίσεις του Spark.
Κάθε εργάτης μπορεί να έχει συνδεδεμένους πολλούς δίσκους. Παρόλο που το Spark δουλεύει στη μνήμη (όχι σε δίσκο), εξακολουθεί να χρειάζεται χώρο στον δίσκο καθώς τα δεδομένα μπορούν να περάσουν στο δίσκο εάν δεν υπάρχει αρκετή διαθέσιμη μνήμη ή από επιλογή του χρήστη όπως στην περίπτωση της επιμονής στην κρυφή μνήμη (cache persistence).

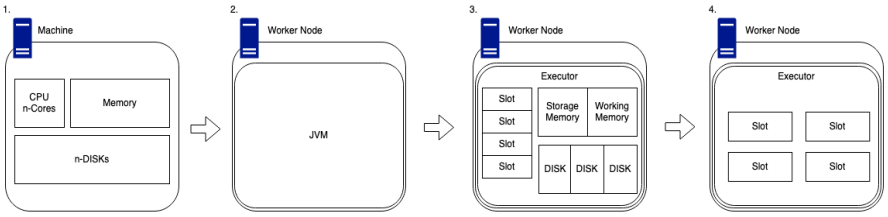
Η βασική συστάδα θα καταγράψει τον αριθμό των θέσεων, οι οποίες είναι στην ουσία ο αριθμός των πυρήνων που είναι διαθέσιμοι στη συσκευή. Αυτές οι θέσεις ταξινομούνται ως διαθέσιμα υπολογιστικά τεμάχια και μπορεί ο κόμβος οδηγός να τους διαθέτει εργασίες για να ολοκληρώσουν.

Οι εκτελεστές Spark ή οι εργάτες διανέμονται σε όλη τη συστάδα. Κάθε εκτελεστής έχει ένα φάσμα γνωστό ως πυρήνα επεξεργασίας δεδομένων. Βασισμένο στο διαθέσιμο μέγεθος του πυρήνα σε έναν εκτελεστή, συλλέγουν εργασίες από τον οδηγό για να συνεχίσουν τη λογική του πυρήνα στα δεδομένα και διατηρούν τα δεδομένα στη μνήμη ή στην αποθήκη του δίσκου. Μπορούν να διαβάσουν δεδομένα τόσο από εσωτερικά όσο και από εξωτερικά συστήματα αποθήκευσης.

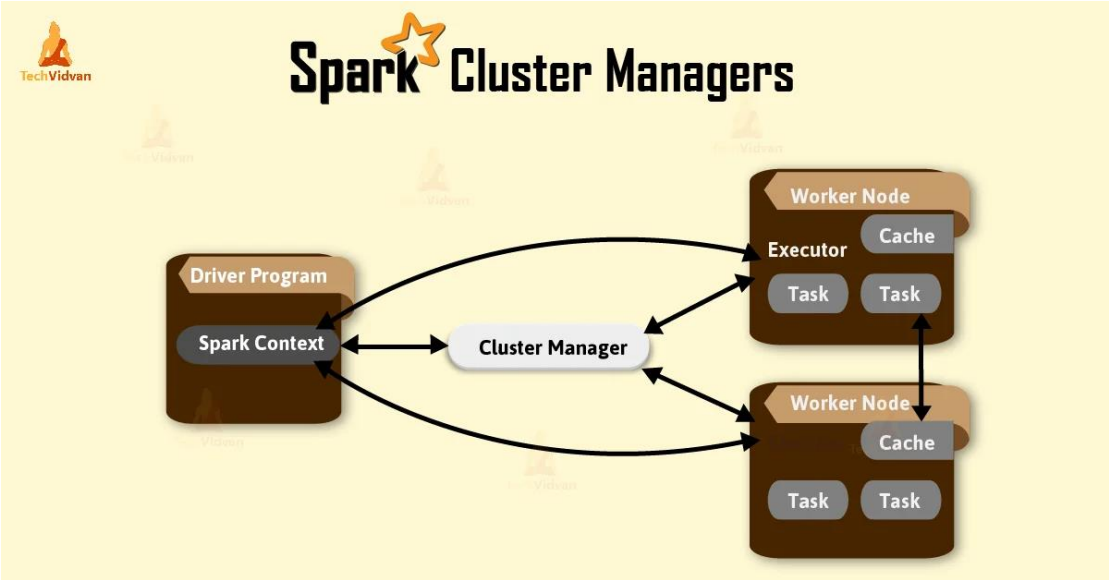
Ο αριθμός των εκτελεστών που εκκινούνται σε μια εφαρμογή Spark εξαρτάται από τους διαθέσιμους πόρους στη συστάδα και τις ρυθμίσεις διαμόρφωσης. Οι εκτελεστές μπορούν να ρυθμιστούν ώστε να διαθέτουν συγκεκριμένο αριθμό

πυρήνων CPU και μνήμης. Από προεπιλογή, σε κάθε εκτελεστή εκχωρείται 1 πυρήνας και 1 GB μνήμης, αλλά αυτές οι τιμές μπορούν να αλλάξουν.

([38]), ([39]), ([40]), ([41])



Εικόνα 1.4.8: Η αρχιτεκτονική των Εκτελεστών Spark (Από τον Data Scientist Luke Thorp)



Εικόνα 1.4.9: Η αρχιτεκτονική του Διαχειριστή Συστάδας (Spark Cluster)

1.4.6 Διαχειριστές Συστάδας (Cluster Managers)

Η έννοια του Διαχειριστή Συστάδας (Cluster Manager) δεν περιορίζεται στο Spark ή στο Hadoop, αλλά χρησιμοποιείται γενικά στον κατανεμημένο προγραμματισμό. Τόσο το Spark όσο και το Hadoop έχουν τους δικούς τους

διαχειριστές συστάδας, όπως ο Mesos, ο YARN για το Hadoop και ο αυτόνομος διαχειριστής συστάδας του Spark. Η θέση του είναι ανεξάρτητη από τους κόμβους αρχηγού και εργατών. Το μόνο απαιτούμενο είναι ότι πρέπει να βρίσκεται σε ένα δίκτυο αναγνωρισμένο από τους κόμβους εργάτες του Spark. Το κύριο καθήκον του Cluster Manager είναι να εξασφαλίσει πόρους για όλες τις εφαρμογές. Είναι δηλαδή μια εξωτερική υπηρεσία για την απόκτηση απαιτούμενων πόρων μέσα στη συστάδα.

([49])

Υπάρχουν αρκετοί Διαχειριστές Συστάδας διαθέσιμοι για χρήση με το Spark, όπως:

1. Αυτόνομος Διαχειριστής Συστάδας (Standalone Cluster Manager)

Είναι μέρος της διανομής Spark και είναι διαθέσιμος ως ένας απλός διαχειριστής συστάδας. Ο αυτόνομος διαχειριστής συστάδας είναι ανθεκτικός και μπορεί να χειριστεί τις αποτυχίες εργασιών και παρέχει δυνατότητες διαχείρισης πόρων σύμφωνα με τις απαιτήσεις των εφαρμογών.

Ο Αυτόνομος Διαχειριστής Συστάδας Spark είναι ένας απλός και εύχρηστος τρόπος εκτέλεσης του Spark σε μια συστάδα, ωστόσο, δεν είναι τόσο ισχυρός ή επεκτάσιμος όσο άλλοι διαχειριστές συστάδας όπως το YARN ή το Mesos, ειδικά για ανάπτυξη εφαρμογών μεγάλης κλίμακας.

Τα στοιχεία του Αυτόνομου Διαχειριστή Συστάδας Spark:

Spark Master: Η διαδικασία Spark Master (Αρχηγός Spark) είναι υπεύθυνη για τον συντονισμό της κατανομής των εργασιών σε όλο τη συστάδα. Όταν εκκινείται η διαδικασία Spark Master, ακούει σε μια συγκεκριμένη διεύθυνση δικτύου για εισερχόμενες συνδέσεις από τους εργάτες Spark. Μόλις ένας εργάτης συνδεθεί με την διαδικασία Master, εκείνη του εκχωρεί ένα μοναδικό αναγνωριστικό και αρχίζει να παρακολουθεί τους πόρους του (CPU, μνήμη κ.λπ.). Όταν ένας χρήστης υποβάλλει μια εργασία Spark, η εργασία υποβάλλεται πρώτα στην διαδικασία Master, η οποία στη συνέχεια προγραμματίζει την εργασία στους διαθέσιμους Workers. Η διαδικασία Master λαμβάνει υπόψη τους διαθέσιμους πόρους κάθε Worker κατά τον προγραμματισμό εργασιών για να διασφαλίσει ότι κάθε εργασία εκτελείται όσο το δυνατόν πιο αποτελεσματικά. Η διαδικασία Master παρακολουθεί επίσης την κατάσταση των Workers στη συστάδα. Εάν ένας Worker βγει εκτός λειτουργίας ή δεν

ανταποκρίνεται, η διαδικασία Master ειδοποιείται και μπορεί να λάβει τα κατάλληλα μέτρα, όπως να αναθέσει εκ νέου τα καθήκοντα του κόμβου εργάτη σε άλλους διαθέσιμους.

Spark Worker: Η διαδικασία Spark Worker (Εργάτης Spark) είναι υπεύθυνη για την εκτέλεση εργασιών σε έναν κόμβο στη συστάδα Spark. Μια συστάδα Spark μπορεί να έχει πολλαπλούς Workers, με κάθε Worker να τρέχει σε ξεχωριστό κόμβο. Κάθε Spark Worker μπορεί να εκτελέσει έναν ή περισσότερους Spark Executors, οι οποίοι είναι υπεύθυνοι για την εκτέλεση εργασιών που έχουν ανατεθεί από την διαδικασία Master.

Λεπτομερέστερα:

1. Όταν το Spark ξεκινά σε αυτόνομη λειτουργία, ο χρήστης εκκινεί τη διαδικασία Spark Master χρησιμοποιώντας την αντίστοιχη εντολή στον κόμβο αρχηγό της συστάδας.
2. Μόλις εκτελεστεί η διαδικασία Spark Master, ο χρήστης μπορεί στη συνέχεια να ξεκινήσει την διαδικασία Spark Worker σε καθέναν από τους κόμβους εργατών.
3. Όταν εκκινείται η διαδικασία Spark Worker, καταχωρείται στο Spark Master και αναφέρει τον όγκο των πόρων (CPU, μνήμη, κ.λπ.) που είναι διαθέσιμοι σε αυτόν τον κόμβο.
4. Μόλις το Spark Master έχει μια λίστα με όλους τους διαθέσιμους κόμβους εργάτες και τους πόρους τους, μπορεί να ξεκινήσει να προγραμματίζει εργασίες και διεργασίες στην συστάδα. Όταν ένας χρήστης υποβάλλει μια εργασία Spark στη συστάδα, το Spark Master δημιουργεί ένα DAG (Directed Acyclic Graph) με στάδια και εργασίες που πρέπει να εκτελεστούν.
5. Το Spark Master αναθέτει διεργασίες σε κάθε κόμβο εργάτη με βάση τους διαθέσιμους πόρους και τον φόρτο εργασίας στον κόμβο. Στη συνέχεια, οι Spark Workers ξεκινούν να εκτελούν τις διεργασίες που τους έχουν ανατεθεί.
6. Καθώς οι Spark Workers εκτελούν διεργασίες, στέλνουν ενημερώσεις πίσω στον Spark Master σχετικά με την πρόοδο των διεργασιών. Στη συνέχεια, ο Spark Master ενημερώνει την κατάσταση των διεργασιών στις εσωτερικές δομές δεδομένων του και αναφέρει την κατάσταση στον χρήστη.
7. Μόλις ολοκληρωθούν όλες οι διεργασίες για μια εργασία, το Spark Master αναφέρει την τελική κατάσταση της εργασίας στον χρήστη.

([47]), ([48])

2. Hadoop YARN (Yet Another Resource Manager)

Αυτός ο διαχειριστής συστάδας λειτουργεί ως κατανεμημένο υπολογιστικό πλαίσιο. Το YARN παρέχει δυνατότητες διαχείρισης πόρων, επιτρέποντας σε πολλές εφαρμογές να εκτελούνται χωρίς διενέξεις. Κατανέμει πόρους σε εφαρμογές με βάση τις απαιτήσεις και τη διαθεσιμότητά τους, διασφαλίζοντας ότι οι χρησιμοποιούνται αποτελεσματικά.

Το Spark αλληλεπιδρά με τον διαχειριστή πόρων του YARN για να ζητήσει και να εκχωρήσει πόρους (μνήμη, CPU, κλπ.) για την εκτέλεση εφαρμογών Spark. Στη συνέχεια, ο προγραμματιστής YARN προγραμματίζει τις διεργασίες και τα στάδια του Spark στους διαθέσιμους πόρους της συστάδας με βάση τις ρυθμίσεις του χρήστη. Έτσι, σε αυτήν την περίπτωση, ο προγραμματισμός διεργασιών και σταδίων Spark διαχειρίζεται από τον προγραμματιστή YARN.

Τα στοιχεία του Διαχειριστή Συστάδας YARN:

- *NodeManager*: Στο Hadoop YARN, ένας Διαχειριστής Κόμβου (NodeManager) είναι μια διαδικασία που εκτελείται σε κάθε κόμβο εργάτη της συστάδας και διαχειρίζεται τους διαθέσιμους πόρους στον κόμβο. Ο NodeManager είναι υπεύθυνος για την εκκίνηση και την παρακολούθηση κοντέινερ, τα οποία είναι απομονωμένα περιβάλλοντα που εκτελούν εφαρμογές ή διεργασίες που ζητούνται από τον Διαχειριστή Πόρων (ResourceManager). Ο NodeManager παρακολουθεί επίσης τη χρήση των πόρων στον κόμβο, συμπεριλαμβανομένης της χρήσης της CPU, της μνήμης και του δίσκου, και αναφέρει τις πληροφορίες αυτές στον ResourceManager. Αυτό επιτρέπει στον ResourceManager να λαμβάνει τεκμηριωμένες αποφάσεις σχετικά με τον τρόπο κατανομής πόρων σε όλη την συστάδα.
- *ResourceManager*: Ο ResourceManager είναι η κεντρική αρχή συντονισμού στην αρχιτεκτονική Hadoop YARN. Είναι υπεύθυνος για τη διαχείριση των πόρων της συστάδας και παρέχει έναν προγραμματιστή που εκχωρεί πόρους στις εφαρμογές που εκτελούνται. Ο ResourceManager εκτελείται στον κόμβο αρχηγό μιας συστάδας YARN και αλληλεπιδρά με τους NodeManagers που εκτελούνται στους κόμβους εργάτες για την εκτέλεση εργασιών. Επίσης,

παρακολουθεί τους NodeManagers και τους επανεκκινεί σε περίπτωση αποτυχίας. Ο ResourceManager διαδραματίζει βασικό ρόλο στην αποτελεσματική κοινή χρήση των πόρων μεταξύ διαφορετικών εφαρμογών που εκτελούνται σε μια συστάδα.

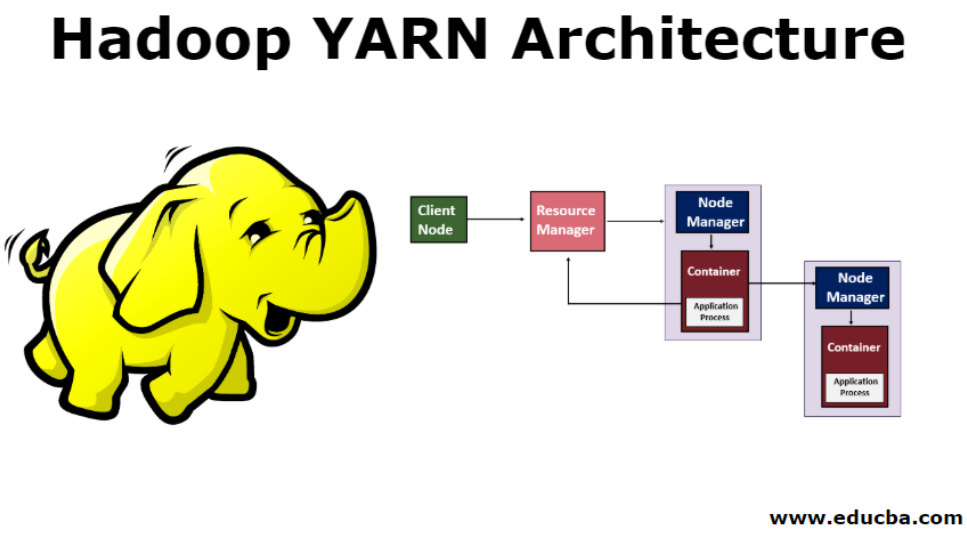
- *ApplicationMaster*: Ο ApplicationMaster είναι υπεύθυνος για τη διαχείριση και την παρακολούθηση της εκτέλεσης μιας συγκεκριμένης εφαρμογής που εκτελείται σε μια συστάδα YARN. Όταν ένας πελάτης υποβάλλει μια εφαρμογή για εκτέλεση στην συστάδα, ο ResourceManager εκχωρεί έναν ApplicationMaster στην εφαρμογή. Μόλις εκχωρηθεί ο ApplicationMaster, είναι υπεύθυνος για τη διαπραγμάτευση πόρων με τον ResourceManager, την αίτηση κοντέινερ από τους NodeManagers και την παρακολούθηση της προόδου της εφαρμογής. Συντονίζει επίσης εργασίες εντός της εφαρμογής, διαχειρίζεται τις αποτυχίες και αναφέρει την τελική κατάσταση της εφαρμογής στον πελάτη.

Λεπτομερέστερα:

1. Ο Διαχειριστής Πόρων (ResourceManager - RM) του YARN λαμβάνει την υποβολή εργασίας και διαπραγματεύεται πόρους για την εργασία, όπως μνήμη και πυρήνες επεξεργαστή στους κόμβους της συστάδας.
2. Στη συνέχεια, ο ResourceManager ξεκινά ένα κοντέινερ σε έναν από τους διαθέσιμους NodeManagers (NMs) στη συστάδα.
3. Το κοντέινερ εκκινεί τη διαδικασία ApplicationMaster (AM), η οποία είναι υπεύθυνη για τον συντονισμό της εργασίας και την αίτηση πρόσθετων κοντέινερ, όπως απαιτείται.
4. Στη συνέχεια, ο ApplicationMaster διαπραγματεύεται με τον ResourceManager για την διάθεση πόρων που θα χρησιμοποιηθούν για την εκκίνηση πρόσθετων κοντέινερ και την εκτέλεση των διεργασιών της εργασίας.
5. Ο ResourceManager εκκινεί κοντέινερ σε άλλους NodeManagers για την εκτέλεση των διεργασιών της εργασίας, με κάθε κοντέινερ να φιλοξενεί έναν ή περισσότερους εκτελεστές Spark.
6. Οι εκτελεστές Spark εκτελούν τις διεργασίες που περιλαμβάνει η εργασία, επικοινωνώντας με το πρόγραμμα οδηγού και ανταλλάσσοντας δεδομένα.

7. Καθ' όλη τη διάρκεια του κύκλου ζωής της εργασίας, ο ResourceManager και οι NodeManagers παρακολουθούν την κατάσταση των κοντέινερ και χειρίζονται τυχόν αστοχίες που ενδέχεται να προκύψουν. Ο ApplicationMaster επικοινωνεί επίσης με το πρόγραμμα Οδηγού για να αναφέρει την πρόοδο και να λάβει οδηγίες για το πώς να προχωρήσει στην εργασία.

([47]), ([48]), ([52])



Εικόνα 1.4.10: Η αρχιτεκτονική του Hadoop YARN.

3. Apache Mesos

Ο Apache Mesos είναι ένας διαχειριστής συστάδας που παρέχει αποτελεσματική απομόνωση και κοινή χρήση πόρων σε κατανεμημένες εφαρμογές ή πλαίσια. Επιτρέπει στις εφαρμογές να εκτελούνται σε μια κοινόχρηστη δεξαμενή πόρων, συμπεριλαμβανομένων της CPU, της μνήμης και της αποθήκευσης στον δίσκο, ενώ διασφαλίζει υψηλή διαθεσιμότητα και ανοχή σφαλμάτων. Το Mesos χρησιμοποιεί μια αρχιτεκτονική δύο επιπέδων, με τον Mesos Master να συντονίζει την κατανομή πόρων και τον Mesos Agent να διαχειρίζεται τις προσφορές πόρων και να εκτελεί εργασίες σε έναν κόμβο εργάτη. Ο Mesos Master διατηρεί μια συνολική εικόνα των διαθέσιμων πόρων της συστάδας και αποφασίζει πώς να κατανείμει αυτούς τους

πόρους στα διαφορετικά πλαίσια που εκτελούνται σε ανώτερα επίπεδα. Ο Mesos Agent, από την άλλη πλευρά, επικοινωνεί με τον Mesos Master για να προσφέρει πόρους και να λαμβάνει εργασίες για εκτέλεση από τα πλαίσια. Ο Mesos μπορεί να υποστηρίξει διάφορες κατανεμημένες εφαρμογές, συμπεριλαμβανομένων των Hadoop, Spark και άλλων πλαισίων επεξεργασίας μεγάλων δεδομένων, καθώς και συστήματα ενορχήστρωσης κοντέινερ όπως το Docker και το Kubernetes.

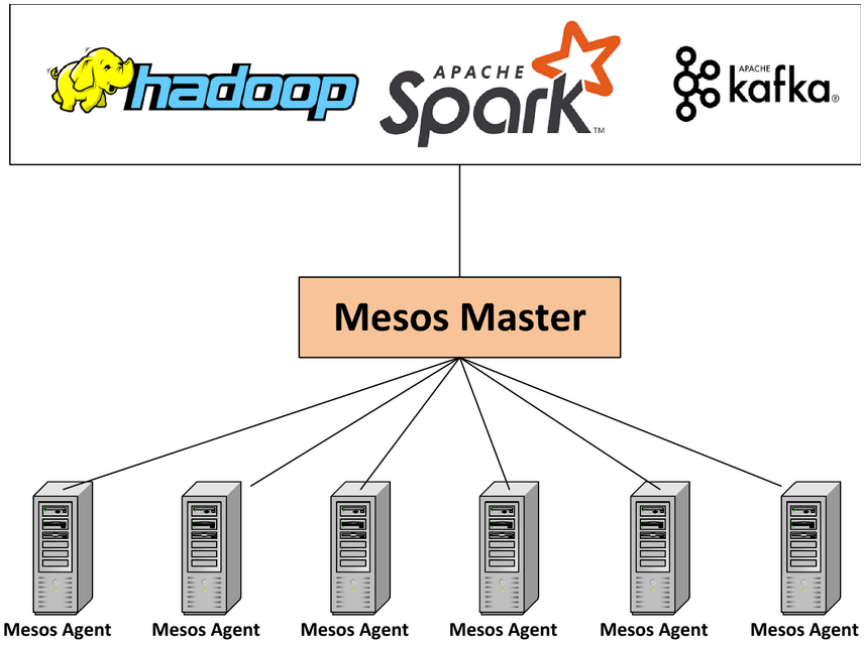
Τα στοιχεία του Διαχειριστή Συστάδας Mesos:

- *Mesos Master* (Αρχηγός Mesos): Ο Διαχειριστής Συστάδας Mesos αποτελείται από μία διαδικασία Mesos Master, η οποία είναι υπεύθυνη για τη διαχείριση των πόρων της συστάδας. Ο Mesos Master είναι ένας κεντρικός προγραμματιστής που λαμβάνει προσφορές πόρων από τους Mesos Agents και προγραμματίζει εργασίες σε όλη τη συστάδα.
- *Mesos Agent* (Πράκτορας Mesos): Η διαδικασία Mesos Agent εκτελείται σε κάθε κόμβο εργάτη στη συστάδα και είναι υπεύθυνη για την προσφορά πόρων στον Mesos Master. Κάθε Mesos Agent εγγράφεται στον Mesos Master και προσφέρει τους διαθέσιμους πόρους σε αυτόν τον κόμβο.
- *Mesos Framework* (Πλαίσιο Mesos): Το Mesos Framework είναι μια κατανεμημένη εφαρμογή που εκτελείται στη συστάδα Mesos. Το Mesos Framework αποτελείται από δύο μέρη: τον Mesos Scheduler και τον Mesos Executor.
- *Mesos Scheduler* (Προγραμματιστής Mesos): Ο Mesos Scheduler είναι ένα στοιχείο του Mesos Framework που αλληλεπιδρά με τον Mesos Master για να αποκτήσει πόρους για την εκτέλεση εργασιών. Ο Mesos Scheduler λαμβάνει προσφορές πόρων από τον Mesos Master και επιλέγει τους κατάλληλους πόρους για την εκτέλεση εργασιών.
- *Mesos Executor* (Εκτελεστής Mesos): Ο Mesos Executor είναι ένα στοιχείο του Mesos Framework που εκτελείται σε κάθε κόμβο εργασίας στην συστάδα. Ο Mesos Executor είναι υπεύθυνος για την εκκίνηση και τη διαχείριση εργασιών σε έναν κόμβο εργάτη.

Λεπτομερέστερα:

1. Μια κύρια διεργασία Mesos εκτελείται σε έναν συγκεκριμένο κόμβο, ο οποίος είναι υπεύθυνος για τη διαχείριση των πόρων της συστάδας και τον προγραμματισμό διεργασιών σε Πράκτορες Mesos.
2. Σε κάθε κόμβο της συστάδας, είναι εγκατεστημένοι πολλαπλοί πράκτορες Mesos, οι οποίοι είναι υπεύθυνοι για την προσφορά πόρων (CPU, μνήμη, δίσκος και θύρες) στον Αρχηγό Mesos.
3. Τα πλαίσια Mesos αναπτύσσονται πάνω από τον Αρχηγό Mesos, τα οποία αποτελούνται από έναν προγραμματιστή (Scheduler) και έναν εκτελεστή (Executor). Ο προγραμματιστής (Scheduler) είναι υπεύθυνος να ζητά πόρους από τον Mesos Master για την εκκίνηση διεργασιών, ενώ ο εκτελεστής είναι υπεύθυνος για την εκτέλεση των διεργασιών στους πράκτορες Mesos.
4. Όταν ένα πλαίσιο Mesos υποβάλλει μια εργασία στον Αρχηγό Mesos, ο δεύτερος επιλέγει ένα σύνολο πρακτόρων που πληρούν τις απαιτήσεις της εργασίας όσον αφορά τους πόρους και τους αναθέτει διεργασίες.
5. Οι πράκτορες Mesos λαμβάνουν τις διεργασίες και τις μεταφέρουν στο δικό τους περιβάλλον κοντέινερ.
6. Ο Mesos Master παρακολουθεί συνεχώς τους πράκτορες και τη χρήση των πόρων τους για να διασφαλίσει ότι οι εργασίες εκτελούνται αποτελεσματικά.
7. Μόλις ολοκληρωθεί μια εργασία, ο εκτελεστής ειδοποιεί τον Αρχηγό Mesos, ο οποίος ενημερώνει την κατάσταση της εργασίας και αποδεσμεύει τους πόρους που διατέθηκαν.

([47]), ([48]), ([53])



Εικόνα 1.4.11: Η αρχιτεκτονική του Apache Mesos.

4) Kubernetes

Το Kubernetes είναι μια πλατφόρμα ενορχήστρωσης κοντέινερ ανοιχτού κώδικα που αυτοματοποιεί την ανάπτυξη, την κλιμάκωση και τη διαχείριση εφαρμογών με κοντέινερ. Παρέχει μια πλατφόρμα για την ανάπτυξη και τη διαχείριση εφαρμογών με κιβώτια, με δυνατότητες όπως η αυτόματη κλιμάκωση, η ανάρρωση και οι κυλιόμενες ενημερώσεις. Χρησιμοποιείται ευρέως για τη διαχείριση μεγάλων κιβωτίων εφαρμογών σε περιβάλλοντα παραγωγής.

Το Kubernetes μπορεί επίσης να χρησιμοποιηθεί ως διαχειριστής συμπλέγματος για την εκτέλεση κατανεμημένων πλαισίων επεξεργασίας δεδομένων όπως το Apache Spark. Σε αυτό το πλαίσιο, ο Kubernetes διαχειρίζεται την κατανομή των πόρων και τον προγραμματισμό εργασιών σε όλη την συστάδα, επιτρέποντας στο Spark να λειτουργεί ως κατανεμημένο σύστημα πάνω από μια υποδομή με κιβώτια. Αυτό επιτρέπει την αποτελεσματική χρήση πόρων και τη δυναμική κλιμάκωση των συστάδων Spark για την κάλυψη του μεταβαλλόμενου φόρτου εργασίας.

Τα στοιχεία του Διαχειριστή Συστάδας Kubernetes:

- *kube-apiserver*: Ο διακομιστής API που παρέχει το Kubernetes API χρησιμοποιώντας endpoints (τελικά σημεία επικοινωνίας) για τη διαχείριση πόρων στην συστάδα.
- *kube-controller-manager*: Ο διαχειριστής ελεγκτών είναι υπεύθυνος για τη διαχείριση διαφορετικών ελεγκτών που χειρίζονται την αναπαραγωγή, τα endpoints, τους λογαριασμούς υπηρεσιών και άλλες λειτουργίες.
- *kube-scheduler*: Ο προγραμματιστής εκχωρεί φόρτους εργασίας σε κόμβους με βάση τη διαθεσιμότητα πόρων και τις απαιτήσεις φόρτου εργασίας.
- *pod*: Ένα pod είναι η μικρότερη μονάδα ανάπτυξης στο Kubernetes. Ένα pod είναι ένας κόμβος αρχηγός για ένα ή περισσότερα κοντέινερ και μπορεί να θεωρηθεί ως ένα instance μιας ομάδας κοντέινερ. Κάθε pod έχει τη δική του διεύθυνση IP και μπορεί να επικοινωνεί με άλλα pods στην ίδια συστάδα.
- *kubelet*: Το kubelet είναι ένας πράκτορας που εκτελείται σε κάθε κόμβο και επικοινωνεί με τον διακομιστή API για να διασφαλίσει ότι τα κοντέινερ εκτελούνται σωστά.

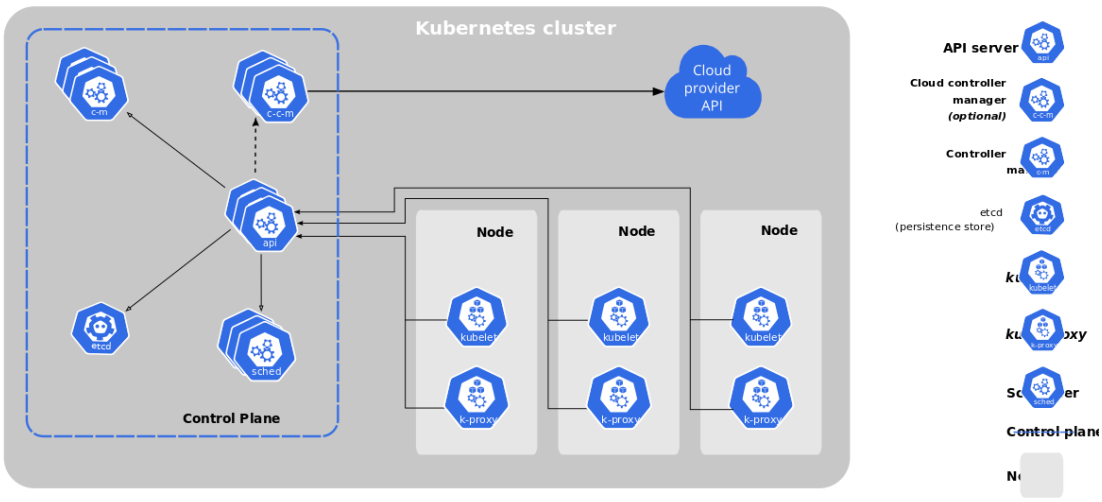
Λεπτομερέστερα:

1. Ο διακομιστής API του Kubernetes λαμβάνει το αντικείμενο ανάπτυξης και δημιουργεί ένα σύνολο αντικειμένων pod με βάση τις προδιαγραφές ανάπτυξης.

   Ο διαχειριστής συστάδας λαμβάνει την υποβολή (ssubmit) μιας εφαρμογής Spark η οποία περιλαμβάνει τον κωδικό εφαρμογής, τις εξαρτήσεις (dependencies) και τις παραμέτρους διαμόρφωσης.
2. Ο διαχειριστής συστάδας αλληλεπιδρά με το Kubernetes API για να ζητήσει τους απαραίτητους πόρους (CPU, μνήμη κ.λπ.) για την εκτέλεση της εφαρμογής Spark. Καθορίζει την απαιτούμενη κατανομή πόρων με βάση τη διαμόρφωση της εφαρμογής και τη διαθέσιμη χωρητικότητα στη συστάδα.
3. Ο διαχειριστής συστάδας συσκευάζει τον κώδικα εφαρμογής Spark, τις εξαρτήσεις και τη διαμόρφωση σε εικόνες κοντέινερ (container images). Χρησιμοποιούνται εργαλεία όπως το Docker για τη δημιουργία των εικόνων, διασφαλίζοντας ότι περιλαμβάνονται όλα τα απαραίτητα στοιχεία για την εκτέλεση της εφαρμογής Spark.

4. Δημιουργούνται Kubernetes Pods, όπου κάθε Pod αντιπροσωπεύει έναν εκτελεστή Spark. Ο αριθμός των Pods που δημιουργούνται εξαρτάται από τον επιθυμητό παραλληλισμό και τις απαιτήσεις πόρων της εφαρμογής Spark.
5. Ο διαχειριστής ελεγκτών Kubernetes (*kube-controller-manager*) λαμβάνει τα αντικείμενα pod και τα προγραμματίζει να εκτελεστούν σε έναν κόμβο της συστάδας με βάση τους διαθέσιμους πόρους.
6. Ο προγραμματιστής Kubernetes (*kube-scheduler)* εκχωρεί το pod σε έναν κόμβο με βάση την διαθεσιμότητα του κόμβου, την χρήση πόρων και τις προτιμήσεις που καθορίζονται από τον χρήστη.
7. Ο πράκτορας kubelet του Kubernetes εκτελείται σε κάθε κόμβο και είναι υπεύθυνος για την εκκίνηση και τη διακοπή κοντέινερ, την παρακολούθηση της υγείας του κοντέινερ και την αναφορά κατάστασης κοντέινερ στον διακομιστή API του Kubernetes.
8. Η Διεπαφή του Χρόνου Εκτέλεσης Κοντέινερ (Container Runtime Interface - CRI) στον Kubernetes, είναι υπεύθυνη για το τράβηγμα (pulling) εικόνων κοντέινερ, τη δημιουργία κοντέινερ και τη διαχείριση των κύκλων ζωής του κοντέινερ.
9. Εάν ο φόρτος εργασίας ή οι απαιτήσεις πόρων αλλάξουν, ο διαχειριστής συστάδας μπορεί να κλιμακώσει δυναμικά τη συστάδα Spark προσθέτοντας ή αφαιρώντας Pods. Προσαρμόζει τον αριθμό των εκτελεστών με βάση τις μεταβαλλόμενες απαιτήσεις.
10. Ολοκλήρωση εφαρμογής: Όταν η εφαρμογή Spark ολοκληρώσει την εκτέλεσή της, ο διαχειριστής συστάδας εκτελεί εργασίες εκκαθάρισης. Τερματίζει τα Pods που σχετίζονται με την εφαρμογή, απελευθερώνει τους εκχωρημένους πόρους και διασφαλίζει ότι η συστάδα Spark επιστρέφει στην αρχική της κατάσταση.

([50]), ([51])



Εικόνα 1.4.12: Η αρχιτεκτονική του Διαχειριστή Συστάδας Kubernetes.

1.4.7 Τρόποι Εκτέλεσης

Κατά την εκτέλεση μιας εφαρμογής, δημιουργείται μια εργασία Spark η οποία υποβάλλεται είτε τοπικά είτε στην συστάδα με την εντολή “spark-submit” (όπως θα αναφερθεί και παρακάτω στο επόμενο κεφάλαιο). Η συμπεριφορά της εργασίας Spark εξαρτάται εξ ολοκλήρου από μία παράμετρο που αποτελεί το κρίσιμο σημείο έναρξης. Αυτή η παράμετρος είναι το στοιχείο Οδηγός που αναφέρθηκε παραπάνω. . Το σημείο που παραμένει ο Οδηγός, καθορίζει και τη συμπεριφορά της εργασίας Spark.

Στο Spark, υπάρχουν τρεις τρόποι εκτέλεσης:

Τοπική λειτουργία (Local Mode): Αυτή η λειτουργία χρησιμοποιείται για την εκτέλεση εφαρμογών Spark σε ένα μόνο μηχάνημα. Στην τοπική λειτουργία, το Spark εκτελείται σε μία διαδικασία JVM (Java Virtual Machine) και τα δεδομένα υποβάλλονται σε επεξεργασία στη μνήμη. Χρησιμοποιείται κυρίως για δοκιμαστικούς σκοπούς καθότι απαιτείται η χρήση λιγότερων πόρων στο σύστημα. ([55]), ([56]), ([58])

Λειτουργία Πελάτη (Client Mode): Η λειτουργία πελάτη είναι μια λειτουργία εκτέλεσης όπου η διαδικασία Οδηγού εκτελείται στο μηχάνημα πελάτη, (το

μηχάνημα πελάτη είναι συνήθως το τοπικό μηχάνημα ενός χρήστη ή ένα καθορισμένο μηχάνημα από το οποίο υποβάλλεται η εφαρμογή Spark στη συστάδα). παρέχοντας τη δυνατότητα αλληλεπίδρασης με την εφαρμογή κατά τη διάρκεια του χρόνου εκτέλεσης. Επιτρέπει τη συνεχή παρακολούθηση της εφαρμογής Spark, προσφέρει ευελιξία στη διαχείριση του κύκλου ζωής της εφαρμογής και παρέχει πληροφορίες σε πραγματικό χρόνο για τη συμπεριφορά της εφαρμογής.
([56]), ([57])

Λειτουργία Συστάδας (Cluster Mode): Αυτή η λειτουργία χρησιμοποιείται για την εκτέλεση εφαρμογών Spark σε μια συστάδα την οποία διαχειρίζεται ένας Διαχειριστής Συστάδας (Cluster Manager) εκτός του αντίστοιχου Spark, όπως το Apache Mesos ή το Hadoop YARN. Στη λειτουργία συστάδας, το πρόγραμμα οδήγησης Spark επικοινωνεί με τους κόμβους εργάτες και συντονίζει την εκτέλεση εργασιών.
([54]), ([55]), ([56])

Ακολουθούν ορισμένα πλεονεκτήματα και μειονεκτήματα κάθε λειτουργίας εκτέλεσης στο Spark:

Τοπική λειτουργία:

Πλεονεκτήματα:

- Εύκολη εγκατάσταση και εκτέλεση, ιδανική για ανάπτυξη και δοκιμή, χωρίς ειδική διαχείρισης συστάδας.

Μειονεκτήματα:

- Περιορισμένη επεκτασιμότητα, δεν μπορεί να αξιοποιήσει κατανεμημένους υπολογιστικούς πόρους.

Λειτουργία Πελάτη:

Πλεονεκτήματα:

- Απλή και εύκολη στη ρύθμιση, κατάλληλη για συστάδες μικρού έως μεσαίου μεγέθους, μπορεί να εκτελέσει λειτουργίες ειδικά για το Spark, όπως προγραμματισμό, παρακολούθηση και ανοχή σφαλμάτων

Μειονεκτήματα:

- Περιορισμένη επεκτασιμότητα, απαιτεί χειροκίνητη διαμόρφωση και διαχείριση

Λειτουργία Συστάδας:

Πλεονεκτήματα:

- Υψηλότερο επίπεδο ανοχής σφαλμάτων: Στη λειτουργία συστάδας, το Spark μπορεί να ανακάμψει από βλάβες ευκολότερα. Εάν ένας κόμβος εργάτη αποτύχει, το Spark μπορεί να εκτελέσει ξανά τις εργασίες σε έναν άλλο κόμβο εργάτη.
- Καλύτερη χρήση πόρων: Εφόσον το πρόγραμμα οδήγησης εκτελείται σε ξεχωριστό κόμβο από τους κόμβους εργάτες, υπάρχει λιγότερες διενέξεις για πόρους στους κόμβους εργάτες. Αυτό μπορεί να οδηγήσει σε καλύτερη χρήση των πόρων και πιο αποτελεσματική εκτέλεση εργασιών.
- Βελτιωμένη επεκτασιμότητα: Η λειτουργία συστάδας επιτρέπει στο Spark να επεκταθεί σε μεγάλο αριθμό κόμβων εργατών, κάτι που μπορεί να είναι σημαντικό για την επεξεργασία μεγάλων ποσοτήτων δεδομένων ή την εκτέλεση πολύπλοκων εργασιών.

Μειονεκτήματα:

- Μεγαλύτερος χρόνος εκκίνησης: Στη λειτουργία συστάδας, υπάρχει επιπλέον επιβάρυνση για την εκκίνηση του προγράμματος οδηγού και την επικοινωνία με τους κόμβους εργατών. Αυτό μπορεί να οδηγήσει σε μεγαλύτερους χρόνους εκκίνησης εργασιών σε σύγκριση με την αυτόνομη λειτουργία.
- Πιο δύσκολος εντοπισμός σφαλμάτων: Δεδομένου ότι το πρόγραμμα οδηγού μπορεί να εκτελείται σε ξεχωριστό κόμβο από τον υπολογιστή

πελάτη, μπορεί να είναι πιο δύσκολο να εντοπιστούν προβλήματα που προκύπτουν κατά την εκτέλεση της εργασίας.

- Απαίτηση αποκλειστικής συστάδας[1]: Η λειτουργία συστάδας απαιτεί μια αποκλειστική συστάδα για την εκτέλεση εργασιών Spark, το οποίο μπορεί να είναι πιο δαπανηρό από τη χρήση μιας διαμοιραζόμενης συστάδας[2] ή την εκτέλεση του Spark σε ένα μόνο μηχάνημα.

1.4.8 Αρχιτεκτονική Λειτουργίας Spark (Spark Runtime Architecture)

Η αρχιτεκτονική Λειτουργίας Spark runtime έγκειται στο τι συμβαίνει εντός της συστάδας, τη στιγμή της εκτέλεσης του κώδικα. Το Spark διαθέτει πρόθυμη (eager) και τεμπέλικη (lazy) αξιολόγηση. Οι ενέργειες Spark είναι πρόθυμες. Ωστόσο, οι μετασχηματισμοί είναι από τη φύση τους τεμπέληδες.

Αυτό ουσιαστικά σημαίνει ότι όταν καλούμε κάποια λειτουργία στα δεδομένα μας, δεν εκτελείται αμέσως, αλλά το Spark διατηρεί το αρχείο της λειτουργίας που καλείται. Αυτές οι λειτουργίες μπορεί να περιλαμβάνουν συνδέσεις και φίλτρα.

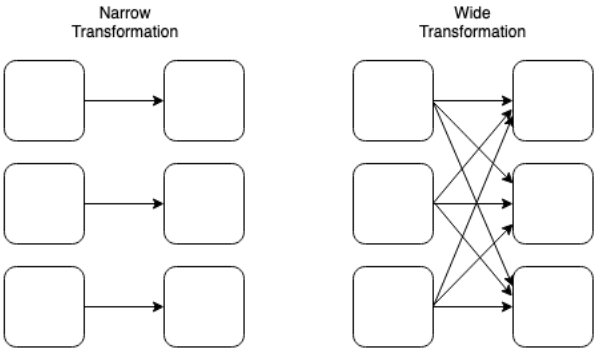
Οι ενέργειες είναι πρόθυμες, πράγμα που σημαίνει ότι όταν καλείται μια ενέργεια, η συστάδα θα αρχίσει να υπολογίζει τα αναμενόμενα αποτελέσματα μόλις εκτελεστεί η γραμμή κώδικα. Όταν εκτελούνται οι Ενέργειες, καταλήγουν σε ένα μη-RDD στοιχείο. ([60]), ([62])

Τύποι Μετασχηματισμών

[1] Σε μια αποκλειστική συστάδα, μια εφαρμογή (ή ένα σύνολο εφαρμογών) έχει αποκλειστική πρόσβαση σε όλους τους πόρους της συστάδας. Αυτό παρέχει περισσότερο έλεγχο στην κατανομή πόρων και καλύτερη απόδοση, καθώς η εφαρμογή μπορεί να χρησιμοποιήσει πλήρως τους διαθέσιμους πόρους χωρίς παρεμβολές από άλλες εφαρμογές. Ωστόσο, σημαίνει επίσης ότι η συστάδα δεν χρησιμοποιείται πλήρως όταν η εφαρμογή δεν εκτελείται ή λειτουργεί με χαμηλή χωρητικότητα.

[2] Σε μια κοινόχρηστη συστάδα, πολλές εφαρμογές μπορούν να εκτελούνται ταυτόχρονα και να μοιράζονται τους διαθέσιμους πόρους (CPU, μνήμη, δίσκος, κ.λπ.) της συστάδας. Η διαχείριση της κατανομής πόρων γίνεται συνήθως από έναν διαχειριστή συστάδας (π.χ. YARN, Mesos, Kubernetes), ο οποίος κατανέμει δυναμικά πόρους σε εφαρμογές που εκτελούνται με βάση τις απαιτήσεις και τις προτεραιότητές τους. Αυτό επιτρέπει την αποτελεσματική χρήση των πόρων, αλλά μπορεί επίσης να οδηγήσει σε διαμάχη πόρων και ζητήματα απόδοσης εάν οι εφαρμογές δεν διαχειρίζονται σωστά.

Υπάρχουν δύο τύποι μετασχηματισμών που είναι βασικοί για την κατανόηση του τρόπου με τον οποίο το Spark διαχειρίζεται δεδομένα και υπολογίζει με κατανεμημένο τρόπο: Οι στενοί (narrow) και οι ευρείες (wide). Οι μετασχηματισμοί δημιουργούν RDD ο ένας από τον άλλο.

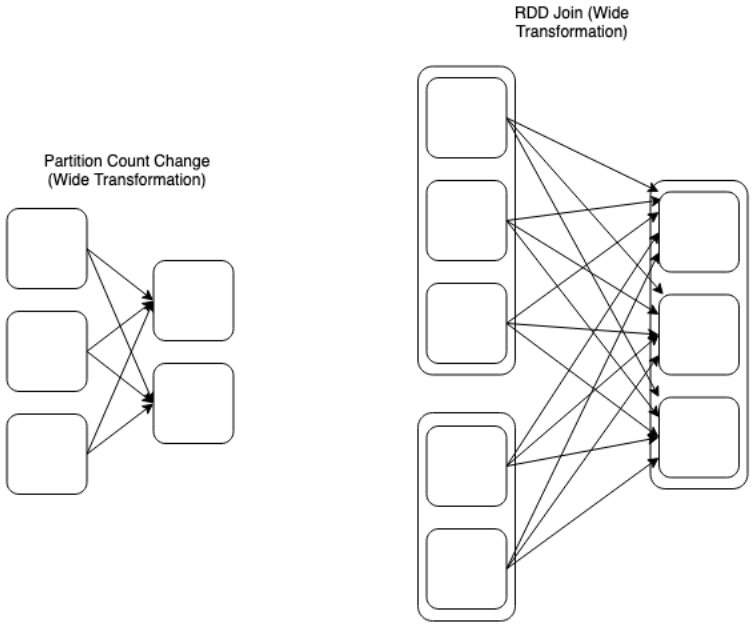


Εικόνα 1.4.13: Στενοί και Ευρείς μετασχηματισμοί (Από τον Data Scientist Luke Thorp).

Στενοί και Ευρείς Μετασχηματισμοί

- Οι στενοί (narrow) μετασχηματισμοί ορίζονται από ένα διαμέρισμα εισόδου που θα έχει ως αποτέλεσμα ένα διαμέρισμα εξόδου. Χαρακτηριστικό παράδειγμα αποτελεί ένα φίλτρο, καθώς μπορούμε να έχουμε ένα DataFrame δεδομένων και να το φιλτράρουμε σε ένα μικρότερο σύνολο δεδομένων χωρίς να χρειάζεται να κατανοήσουμε τυχόν δεδομένα που φυλάσσονται σε οποιονδήποτε άλλο κόμβο εργάτη.

- Οι ευρείς (wide) μετασχηματισμοί ορίζονται από το γεγονός ότι οι κόμβοι εργάτες χρειάζεται να μεταφέρουν δεδομένα σε όλο το δίκτυο για να ολοκληρώσουν την απαιτούμενη εργασία. Ένα παράδειγμα μιας τέτοιας περίπτωσης θα μπορούσε να είναι μια ένωση, καθώς για να ολοκληρωθεί πλήρως μια ένωση δύο συνόλων δεδομένων πολλές φορές απαιτείται η συλλογή δεδομένων από όλη την συστάδα.



Εικόνα 1.4.14: Επέκταση των Ευρείων μετασχηματισμών (Από τον Data Scientist Luke Thorp).

Οι ευρείς μετασχηματισμοί μπορούν να λάβουν πολλές μορφές, ενώ διάφοροι αριθμοί n τμημάτων εισόδου (input partitions) μπορεί να μην έχουν πάντα ως αποτέλεσμα τον ίδιο αριθμό τμημάτων εξόδου (output partitions). Όπως αναφέρθηκε παραπάνω, οι ενώσεις ταξινομούνται ως ευρείες μετασχηματισμοί. Έτσι, στην παραπάνω περίπτωση, έχουμε δύο RDD που έχουν διαφορετικό αριθμό τμημάτων που ενώνονται μεταξύ τους για να σχηματίσουν ένα νέο RDD.

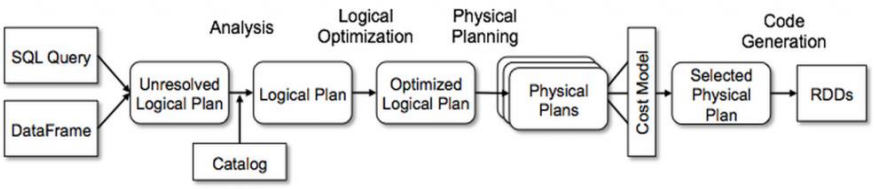
Λειτουργία (Runtime)

Όταν ο χρήστης υποβάλλει κώδικα στην διεργασία Οδηγού, το πρόγραμμα οδηγός μετατρέπει τον κώδικα που περιέχει μετασχηματισμούς (φίλτρα, συνδέσεις, groupby, ενώσεις, κλπ.) και ενέργειες (μετρήσεις, εγγραφές, κ.λπ.) σε ένα άλυτο λογικό σχέδιο.

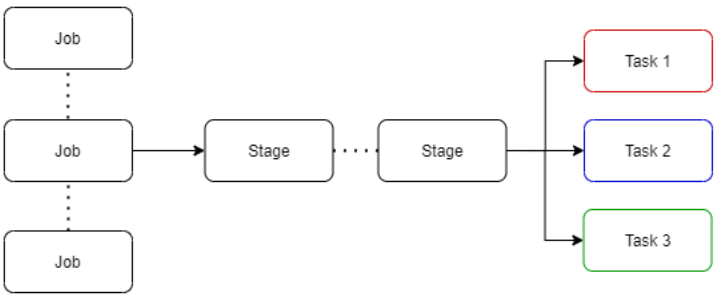
Στο στάδιο της υποβολής, ο οδηγός χρησιμοποιεί τον κατάλογο λογικού σχεδίου για να διασφαλίσει ότι ο κώδικας συμμορφώνεται με τις απαιτούμενες παραμέτρους. Αφού ολοκληρωθεί η ενέργεια αυτή, το άλυτο λογικό σχέδιο μετατρέπεται σε λογικό σχέδιο. Η διαδικασία έπειτα εκτελεί βελτιστοποιήσεις, όπως

ο σχεδιασμός μετασχηματισμών και ενεργειών. Η διαδικασία αυτή της βελτιστοποίησης είναι γνωστή ως βελτιστοποιητής καταλύτη (catalyst optimizer).

Μόλις ολοκληρωθούν οι βελτιστοποιήσεις, το λογικό σχέδιο μετατρέπεται σε πολλαπλά φυσικά σχέδια εκτέλεσης, τα οποία διοχετεύονται σε ένα μοντέλο κόστους που αναλύει κάθε σχέδιο και εφαρμόζει μια τιμή κόστους για να εκτελέσει το κάθε ένα από αυτά. Ως τελικό αποτέλεσμα, θα επιλεγεί το σχέδιο με το χαμηλότερο κόστος ολοκλήρωσης. Το επιλεγμένο αυτό σχέδιο εκτέλεσης θα περιέχει εργασίες με πολλαπλά στάδια. Ο βελτιστοποιητής καταλύτη αξιοποιεί προηγμένες δυνατότητες γλώσσας προγραμματισμού, όπως η αντιστοίχιση μοτίβων της γλώσσας Scala για τη δημιουργία ενός επεκτάσιμου βελτιστοποιητή ερωτημάτων (extensible query optimizer) και βασίζεται σε κατασκευές συναρτησιακού (functional) προγραμματισμού της γλώσσας Scala.



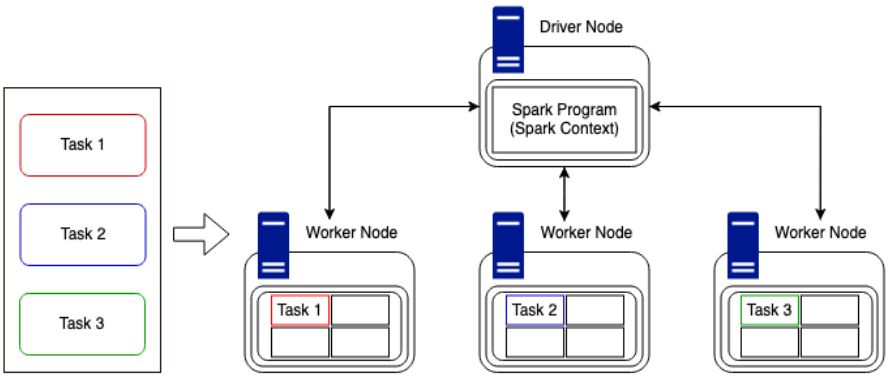
Εικόνα 1.4.15: Apache Spark Catalyst Optimizer



Εικόνα 1.4.16: Εκτέλεση του Φυσικού Πλάνου (Από τον Data Scientist Luke Thorp).

Φυσικό Σχέδιο Εκτέλεσης (Physical Execution Plan)

Τα στάδια αποτελούνται από μικρά φυσικά πλάνα εκτέλεσης που ονομάζονται διεργασίες (tasks), οι οποίες ομαδοποιούνται για αποστολή στη συστάδα Spark. Πριν διανεμηθούν, το πρόγραμμα οδηγού επικοινωνεί με τον διαχειριστή συστάδας για να διαπραγματευτεί τους πόρους.



Εικόνα 1.4.17: Ανάθεση διεργασιών στην συστάδα (Από τον Data Scientist Luke Thorp).

Ανάθεση Διεργασίας στη Συστάδα

Εφόσον έχουν κατανεμηθεί οι πόροι, οι διεργασίες διανέμονται στους κόμβους εργάτες που είναι διαθέσιμοι και το πρόγραμμα οδηγού παρακολουθεί την πρόοδο. Αυτό γίνεται με τον πιο αποτελεσματικό δυνατό τρόπο, λαμβάνοντας υπόψη τους τρέχοντες διαθέσιμους πόρους και τη δομή της συστάδας.

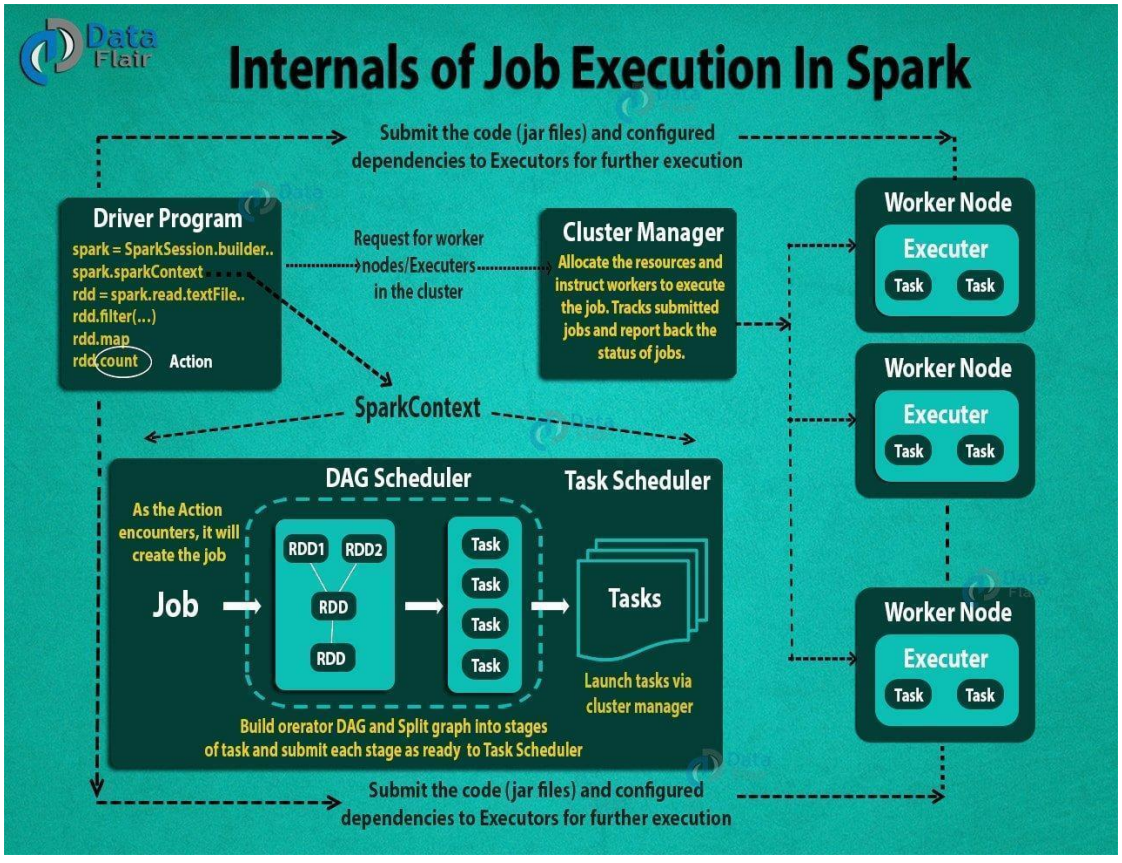
Στο παράδειγμα της παραπάνω εικόνας, έχουμε μία συστάδα που περιλαμβάνει έναν Οδηγό και τρεις κόμβους εργάτες. Ο κάθε κόμβος έχει διαθέσιμες τέσσερις υποδοχές υπολογισμού, άρα η συστάδα θα μπορεί να πραγματοποιήσει δώδεκα διεργασίες μαζί.

Αξίζει να σημειωθεί ότι η νέα μηχανή Photon του Spark είναι μια διανυσματική μηχανή αναζήτησης, η οποία έχει αναπτυχθεί για να ικανοποιεί και να εκμεταλλεύεται την πιο πρόσφατη αρχιτεκτονική της CPU. Αυτό επιτρέπει ταχύτερη

παράλληλη επεξεργασία δεδομένων, όπως επίσης και βελτιώσεις ζωντανής απόδοσης καθώς τα δεδομένα διαβάζονται κατά την εκτέλεση εργασιών.

Η Διεπαφή Χρήστη του Spark (Spark UI) επιτρέπει στον χρήστη να παρακολουθεί ολόκληρη τη διαδικασία φόρτου εργασίας και διαθέτει χρήσιμες καρτέλες σχετικές με Εργασίες και Στάδια. Μεταβαίνοντας σε ένα στάδιο, ο χρήστης μπορεί να εξετάσει μετρήσεις που σχετίζονται με συγκεκριμένες διεργασίες. Αυτή η μέθοδος μπορεί να χρησιμοποιηθεί για να εξεταστεί και να εντοπιστεί η αιτία που συγκεκριμένες εργασίες ίσως καθυστερούν περισσότερο από το αναμενόμενο.

1.4.9 Εκτέλεση Εργασιών Spark



Εικόνα 1.4.18: Σχεδιάγραμμα της εκτέλεσης των διεργασιών στο Apache Spark.

Η εκτέλεση μιας εργασίας Spark ξεκινά με την υποβολή της εφαρμογής χρησιμοποιώντας το εργαλείο εντολών “spark-submit”. Το spark-submit είναι διαθέσιμο σε όλες τις διανομές Spark και μπορεί να χρησιμοποιηθεί για την υποβολή εφαρμογών σε τοπική λειτουργία, λειτουργία συστάδας ή λειτουργία πελάτη. Το spark-submit δέχεται διάφορες επιλογές και παραμέτρους για τη διαμόρφωση της εφαρμογής Spark, όπως την κύρια διεύθυνση URL του Spark, το αρχείο jar της εφαρμογής, την κύρια κλάση της εφαρμογής και άλλες παραμέτρους για συγκεκριμένες εφαρμογές. Εκτός από τη βασική διαμόρφωση της εφαρμογής, το spark-submit επιτρέπει επίσης σύνθετες επιλογές διαμόρφωσης, όπως τον καθορισμό του αριθμού των πυρήνων του εκτελεστή, της ποσότητας μνήμης εκτελεστή και άλλων ιδιοτήτων διαμόρφωσης Spark.

Ακολουθεί ένα παράδειγμα εντολής για την υποβολή μιας εφαρμογής Spark χρησιμοποιώντας το spark-submit:

```
spark-submit \
--class com.example.MyApp \
--master yarn \
--deploy-mode cluster \
--num-executors 10 \
--executor-cores 2 \
--executor-memory 4g \
myapp.jar
```

Η παραπάνω εντολή υποβάλλει την εφαρμογή myapp.jar για εκτέλεση σε μια συστάδα YARN με 10 εκτελεστές, ο καθένας εκ των οποίων διαθέτει 2 πυρήνες CPU και 4 GB μνήμης. Η κλάση com.example.MyApp είναι το σημείο εισόδου για την εφαρμογή. ([63]), ([64]), ([65])

1.4.10 Τοπική Συστάδα και Συστάδα σε Νέφος

Η εκτέλεση του Spark σε μια τοπική συστάδα σημαίνει τον πλήρη έλεγχο της διαμόρφωσης υλικού και λογισμικού της συστάδας. Παρέχει βελτιστοποίηση της συστάδας για την κάθε εφαρμογή Spark και υψηλό βαθμό ευελιξίας όσον αφορά την προσαρμογή των ρυθμίσεων και την ρύθμιση της απόδοσης. Επιτρέπει επίσης την εργασία με ευαίσθητα ή εμπιστευτικά δεδομένα που δεν μπορούν να υποβληθούν σε επεξεργασία στο νέφος (cloud) για λόγους ασφαλείας ή συμμόρφωσης. Ωστόσο, η εκτέλεση του Spark σε τοπικό περιβάλλον έχει επίσης ορισμένους περιορισμούς. Οι πόροι της συστάδας περιορίζονται από το υλικό και τη χωρητικότητα των μηχανημάτων που είναι διαθέσιμα, επομένως η κλιμάκωση προς τα πάνω ή προς τα κάτω μπορεί να είναι δύσκολη και δαπανηρή.

Η εκτέλεση του Spark σε μια συστάδα που βασίζεται στο νέφος, παρέχει πολλά πλεονεκτήματα. Οι συστάδες αυτές είναι εξαιρετικά επεκτάσιμες και μπορούν εύκολα να κλιμακωθούν προς τα πάνω ή προς τα κάτω για να ταιριάζουν με το φόρτο εργασίας και τις απαιτήσεις δεδομένων της κάθε εφαρμογής Spark. Παρέχουν επίσης ενσωματωμένη ανοχή σφαλμάτων και αυτόματη διαχείριση πόρων, η οποία μπορεί να βοηθήσει στη βελτιστοποίηση της απόδοσης και στη μείωση του κόστους. Υπάρχουν όμως και ορισμένα πιθανά μειονεκτήματα. Μπορεί να υπάρχει μικρότερος έλεγχος στη διαμόρφωση υλικού και λογισμικού της συστάδας, γεγονός που θα μπορούσε να περιορίσει την απόδοση ή την δυνατότητα για προσαρμογή των ρυθμίσεις. Ενδέχεται επίσης να απαιτείται υψηλότερο επίπεδο ασφάλειας και συμμόρφωσης λόγω των πιθανών κινδύνων που σχετίζονται με την επεξεργασία ευαίσθητων ή εμπιστευτικών δεδομένων στο cloud.

Μερικές δημοφιλείς πλατφόρμες για την ανάπτυξη του Spark στο νέφος είναι οι Google Cloud Platform, Microsoft Azure και Databricks.

Στο τέλος της ημέρας, η επιλογή μεταξύ της εκτέλεσης του Spark τοπικά είτε στο cloud εξαρτάται από τις συγκεκριμένες ανάγκες και απαιτήσεις για την λειτουργία του Spark.

Εικόνα 1.4.19: Apache Spark και Apache Hadoop

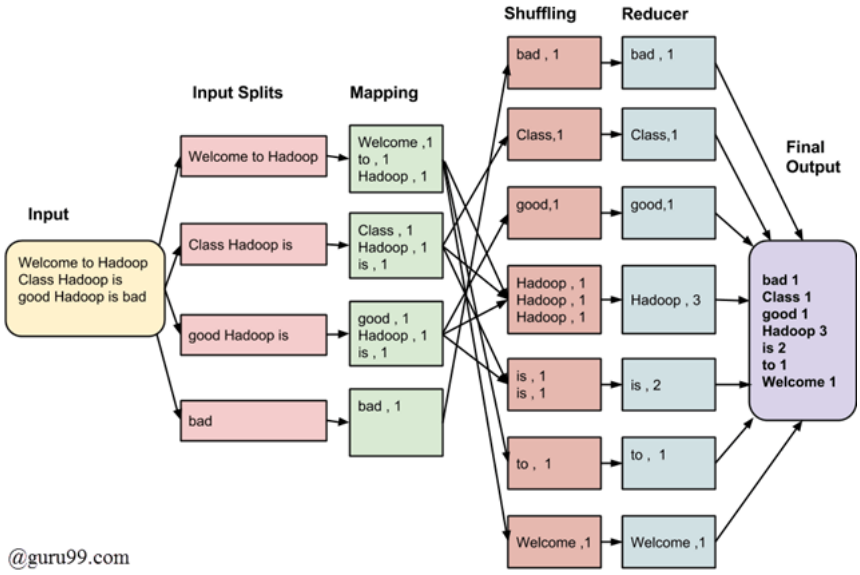
1.5 ΣΥΓΚΡΙΣΗ APACHE HADOOP ΚΑΙ APACHE SPARK

Το Apache Hadoop είναι μια συλλογή προγραμμάτων λογισμικού ανοιχτού κώδικα, βασισμένα σε γλώσσα Java που λειτουργεί με τη χρήση ενός δικτύου πολλών υπολογιστών για την επίλυση προβλημάτων τεράστιου όγκου δεδομένων. Παρέχει ένα πλαίσιο λογισμικού για κατανεμημένη αποθήκευση και επεξεργασία μεγάλων δεδομένων.

Ο πυρήνας του Apache Hadoop αποτελείται από ένα τμήμα αποθήκευσης, γνωστό ως Hadoop Distributed File System (HDFS), και ένα τμήμα επεξεργασίας που είναι ένα μοντέλο προγραμματισμού MapReduce.

Το MapReduce είναι ένα πρότυπο προγραμματισμού που επιτρέπει τεράστια επεκτασιμότητα σε εκατοντάδες ή χιλιάδες διακομιστές σε μια συστάδα Hadoop. Ο όρος MapReduce αναφέρεται σε δύο ξεχωριστές εργασίες που εκτελούνται. Η πρώτη είναι η εργασία χάρτη (map), η οποία παίρνει ένα σύνολο δεδομένων και το μετατρέπει σε ένα άλλο, όπου τα μεμονωμένα στοιχεία αναλύονται σε πλειάδες (ζεύγη κλειδιών/τιμών). Η εργασία μείωσης παίρνει την έξοδο από έναν χάρτη ως είσοδο και συνδυάζει αυτές τις πλειάδες δεδομένων σε ένα μικρότερο σύνολο

πλειάδων. Όπως υποδηλώνει η ακολουθία του ονόματος MapReduce, η εργασία μείωσης εκτελείται πάντα μετά την εργασία χάρτη.



Εικόνα 1.4.20: Βήματα εκτέλεσης της εργασίας MapReduce.

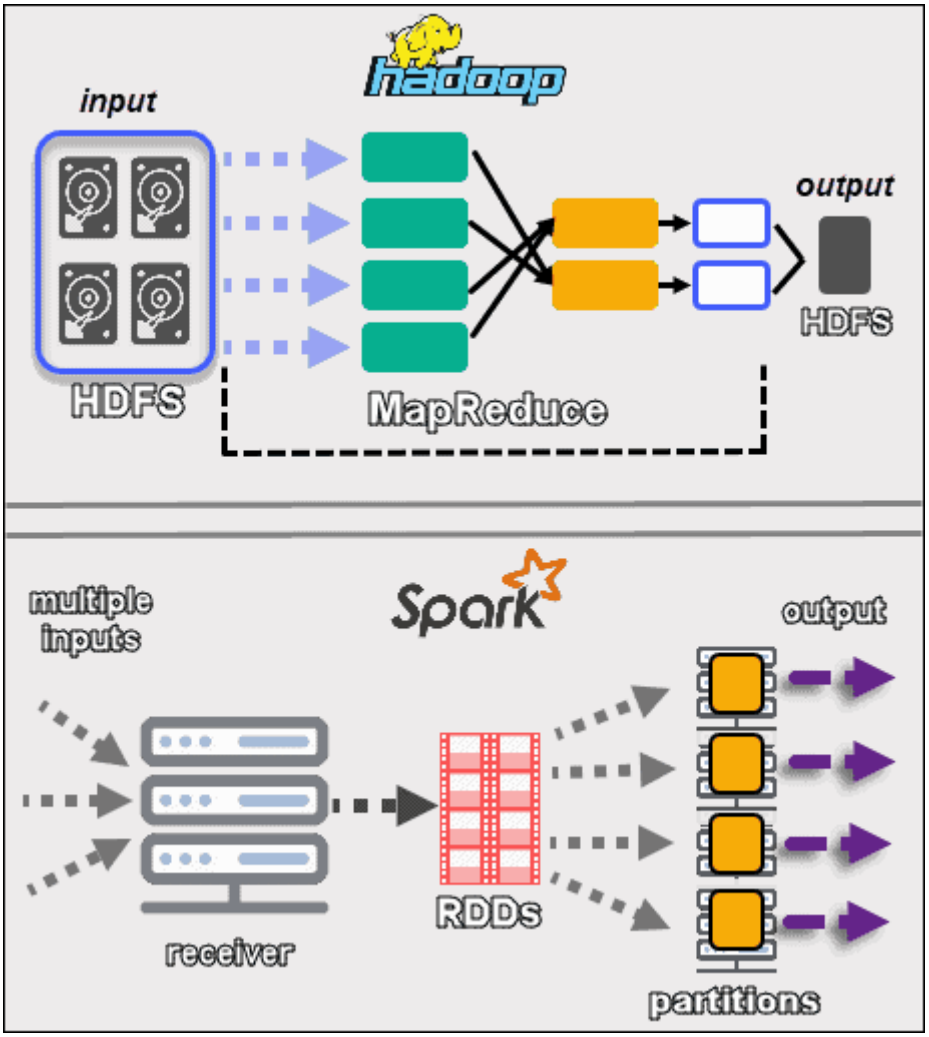
Το Hadoop χωρίζει και συσκευάζει τα αρχεία σε μεγάλα μπλοκ και τα διανέμει στους κόμβους μιας συστάδας. Στη συνέχεια, προωθεί τον συσκευασμένο κώδικα για παράλληλη επεξεργασία των δεδομένων. Αυτή η προσέγγιση εκμεταλλεύεται την τοπική θέση των δεδομένων, όπου οι κόμβοι χειρίζονται τα δεδομένα στα οποία έχουν πρόσβαση. Αυτό επιτρέπει την ταχύτερη και πιο αποτελεσματική επεξεργασία του συνόλου δεδομένων από ότι θα ήταν σε μια πιο συμβατική αρχιτεκτονική που βασίζεται σε ένα παράλληλο σύστημα αρχείων όπου ο υπολογισμός και τα δεδομένα διανέμονται μέσω δικτύων υψηλής ταχύτητας.

Το βασικό πλαίσιο Apache Hadoop αποτελείται από τα ακόλουθα στοιχεία:
Κοινά Στοιχεία Hadoop (Hadoop Common) – Περιέχει βιβλιοθήκες και βοηθητικά προγράμματα που χρειάζονται άλλες μονάδες Hadoop.
Hadoop Distributed File System (HDFS) – Ένα κατανεμημένο σύστημα αρχείων που αποθηκεύει δεδομένα σε τοπικά μηχανήματα, παρέχοντας υψηλό εύρος ζώνης σε όλη τη συστάδα.

Hadoop YARN – (παρουσιάστηκε το 2012) Μια πλατφόρμα υπεύθυνη για τη διαχείριση των υπολογιστικών πόρων σε συστάδες και την διάθεσή τους για τον προγραμματισμό εφαρμογών.
Hadoop MapReduce – Μια υλοποίηση του μοντέλου προγραμματισμού MapReduce για επεξεργασία δεδομένων μεγάλης κλίμακας.
Hadoop Ozone – (παρουσιάστηκε το 2020) Ένας κατανεμημένος επεκτάσιμος αποθηκευτικός χώρος για αναλύσεις και διαχείριση Μεγάλων Δεδομένων.

Οι πέντε βασικές διαφορές του Apache Spark έναντι του Hadoop MapReduce:

- Το Apache Spark είναι δυνητικά 100 φορές ταχύτερο από το Hadoop MapReduce.
- Το Apache Spark χρησιμοποιεί μνήμη RAM και δεν συνδέεται με το παράδειγμα δύο σταδίων του Hadoop.
- Το Apache Spark λειτουργεί καλά για μικρότερα σύνολα δεδομένων που μπορούν όλα να χωρέσουν στη μνήμη RAM ενός διακομιστή.
- Το Hadoop είναι πιο οικονομικό για την επεξεργασία μεγάλων συνόλων δεδομένων.
- Το Apache Spark είναι πλέον πιο δημοφιλές από το Hadoop MapReduce.



Εικόνα 1.4.21: Διαφορές Hadoop MapReduce και Spark.

Για χρόνια, το Hadoop ήταν ο αδιαμφισβήτητος πρωταθλητής των μεγάλων δεδομένων. Μέχρι τη στιγμή που εμφανίστηκε το Spark. Από την αρχική της κυκλοφορίας του το 2014, το Apache Spark έχει βάλει φωτιά στον κόσμο των μεγάλων δεδομένων. Με τα βολικά API του Spark και τις υποσχόμενες ταχύτητες έως και 100 φορές μεγαλύτερες από αυτές του Hadoop MapReduce, ορισμένοι αναλυτές πιστεύουν ότι το Spark σηματοδότησε την άφιξη μιας νέας εποχής στα μεγάλα δεδομένα.

Πώς μπορεί το Spark, ένα πλαίσιο επεξεργασίας δεδομένων ανοιχτού κώδικα, να επεξεργαστεί όλες αυτές τις πληροφορίες τόσο γρήγορα; Το μυστικό είναι ότι το Spark τρέχει με βάση τη μνήμη των κόμβων μιας συστάδας και δεν περιορίζεται με το παράδειγμα δύο σταδίων MapReduce του Hadoop. Αυτό κάνει την

επαναλαμβανόμενη πρόσβαση στα ίδια δεδομένα πολύ πιο γρήγορη. Δεκάδες μεγάλες εταιρείες τεχνολογίας όπως η Yahoo, η Intel, η Baidu, η Yelp και η Zillow χρησιμοποιούν ήδη το Spark ως μέρος των τεχνολογικών τους δομών.

Ενώ το Spark φαίνεται ότι πρόκειται να αντικαταστήσει το Hadoop MapReduce, εντούτοις δεν πρέπει να θεωρηθεί ακόμη ξοφλημένο.

Spark vs MapReduce: Απόδοση

Το Apache Spark επεξεργάζεται δεδομένα στη μνήμη τυχαίας πρόσβασης (RAM), ενώ το Hadoop MapReduce διατηρεί τα δεδομένα πίσω στο δίσκο μετά από μια ενέργεια χάρτη (map) ή μείωσης (reduce). Θεωρητικά, λοιπόν, το Spark θα πρέπει να ξεπερνά το Hadoop MapReduce. Παρόλα αυτά, το Spark χρειάζεται πολλή μνήμη. Όπως και οι τυπικές βάσεις δεδομένων, το Spark φορτώνει μια διεργασία στη μνήμη και τη διατηρεί μέχρι νεωτέρας για λόγους προσωρινής αποθήκευσης. Εάν το Spark εκτελείται στο Hadoop YARN με άλλες υπηρεσίες που απαιτούν πόρους ή εάν τα δεδομένα είναι πολύ μεγάλα για να χωρέσουν εξ ολοκλήρου στη μνήμη, τότε το Spark μπορεί να υποστεί σημαντικές υποβαθμίσεις στην απόδοση.

Το MapReduce τερματίζει τις διεργασίες μόλις ολοκληρωθεί μια εργασία, ώστε να μπορεί εύκολα να εκτελείται παράλληλα με άλλες υπηρεσίες.

Το Spark έχει το πάνω χέρι για επαναληπτικούς υπολογισμούς που πρέπει να προσπελάσουν τα ίδια δεδομένα πολλές φορές. Αλλά όταν πρόκειται για εργασίες ETL με ένα πέρασμα (π.χ. μετασχηματισμός δεδομένων ή ενοποίηση δεδομένων), τότε η καλύτερη επιλογή εδώ είναι το MapReduce.

Συμπέρασμα: Το Spark αποδίδει καλύτερα όταν όλα τα δεδομένα χωρούν στη μνήμη, ειδικά σε αποκλειστικές συστάδες. Το Hadoop MapReduce έχει σχεδιαστεί για δεδομένα που δεν χωρούν στη μνήμη και μπορούν να λειτουργήσουν καλά παράλληλα με άλλες υπηρεσίες.

Spark vs Hadoop MapReduce: Ευκολία χρήσης

Το Spark έχει προκατασκευασμένα APIs για Java, Scala και Python. Χάρη στα απλά δομικά στοιχεία του Spark, είναι εύκολο να δημιουργηθούν λειτουργίες που καθορίζονται από τον χρήστη. Το Spark περιλαμβάνει ακόμη και μια διαδραστική λειτουργία για την εκτέλεση εντολών με άμεση ανάδραση.

Το MapReduce είναι γραμμένο σε Java και είναι πολύ δύσκολο να προγραμματιστεί. Το Apache Pig διευκολύνει (αν και απαιτεί λίγο χρόνο για την εκμάθηση του), ενώ το Apache Hive προσθέτει άμεση συμβατότητα SQL. Ορισμένα εργαλεία Hadoop μπορούν επίσης να εκτελέσουν εργασίες MapReduce χωρίς προγραμματισμό. Για παράδειγμα, το Integrate.io είναι μια υπηρεσία ενοποίησης δεδομένων που είναι χτισμένη πάνω στο Hadoop και επίσης δεν απαιτεί προγραμματισμό ή ανάπτυξη.

Επιπλέον, το MapReduce δεν διαθέτει διαδραστική λειτουργία, αν και το Hive περιλαμβάνει μια διεπαφή γραμμής εντολών. Έργα όπως το Apache Impala και το Apache Tez θέλουν να ενσωματώσουν την δυνατότητα για πλήρη διαδραστικά ερωτήματα (full interactive querying) στο Hadoop.

Όσον αφορά την εγκατάσταση και τη συντήρηση, το Spark δεν δεσμεύεται από το Hadoop. Τόσο το Spark όσο και το Hadoop MapReduce περιλαμβάνονται στις διανομές των Hortonworks (HDP 3.1) και Cloudera (CDH 5.13).

Συμπέρασμα: Το Spark είναι πιο εύκολο να προγραμματιστεί και περιλαμβάνει διαδραστική λειτουργία. Το Hadoop MapReduce είναι πιο δύσκολο να προγραμματιστεί, αλλά υπάρχουν αρκετά εργαλεία που διευκολύνουν την διαδικασία αυτή.

Spark εναντίον Hadoop MapReduce: Κόστος

Το Spark και το MapReduce είναι λύσεις ανοιχτού κώδικα, αλλά απαιτούν χρήματα για την εξασφάλιση πόρων και προσωπικού. Τόσο το Spark όσο και το MapReduce μπορούν να εκτελούνται σε διακομιστές στο νέφος (cloud). Επιπλέον, και τα δύο εργαλεία έχουν παρόμοιες απαιτήσεις υλικού:

	Apache Spark ( )	Apache Hadoop balanced workload slaves ( )
Cores	8–16	4
Memory	8 GB to hundreds of gigabytes	24 GB
Disks	4–8	4–6 one-TB disks
Network	10 GB or more	1 GB Ethernet all-to-all

Το μέγεθος της μνήμης στη συστάδα Spark θα πρέπει να είναι τουλάχιστον όση και η ποσότητα των δεδομένων που πρέπει να επεξεργαστούν, καθώς τα δεδομένα πρέπει να χωρούν στη μνήμη για βέλτιστη απόδοση. Για επεξεργασία εξαιρετικά μεγάλων ποσοτήτων δεδομένων, το Hadoop αποτελεί την πιο οικονομική επιλογή, καθώς ο χώρος στον σκληρό δίσκο είναι λιγότερο ακριβός από τον χώρο στη μνήμη.

Από την άλλη πλευρά, λαμβάνοντας υπόψη την απόδοση του Spark και του MapReduce, το Spark θα πρέπει να είναι πιο οικονομικό. Το Spark απαιτεί λιγότερο υλικό για να εκτελεί τις ίδιες εργασίες ταχύτερα, ειδικά στο νέφος (cloud) όπου η υπολογιστική ισχύς πληρώνεται ανά χρήση.

Σχετικά με το ζήτημα της στελέχωσης, παρόλο που το Hadoop υπάρχει από το 2005, εξακολουθεί να παρατηρείται έλλειψη ειδικών του MapReduce στην αγορά. Σύμφωνα με μια έκθεση έρευνας της Gartner, το 57 τοις εκατό των οργανισμών που χρησιμοποιούν το Hadoop ισχυρίζονται ότι η απόκτηση των απαραίτητων δεξιοτήτων και ικανοτήτων είναι η μεγαλύτερη πρόκληση σχετικά με το Hadoop.

Τι σημαίνει λοιπόν αυτό για το Spark, το οποίο κυκλοφορεί μόλις από το 2014; Ενώ έχει ταχύτερη καμπύλη μάθησης, το Spark στερείται επίσης από ειδικευμένο προσωπικό. Εντούτοις, υπάρχει μια ευρεία γκάμα υπηρεσιών Hadoop-as-a-service και υπηρεσιών που βασίζονται στο Hadoop (όπως η ενοποίηση δεδομένων του Integrate.io), που συμβάλλουν στην κάλυψη αυτών των απαιτήσεων υλικού και προσωπικού. Οι επιλογές Spark-as-a-service είναι διαθέσιμες μέσω παρόχων όπως το Amazon Web Services.

Συμπέρασμα: Το Spark είναι πιο οικονομικά αποδοτικό σύμφωνα με τα κριτήρια αναφοράς, αν και η στελέχωση μπορεί να είναι πιο δαπανηρή. Το Hadoop MapReduce θα μπορούσε να είναι φθηνότερο επειδή υπάρχει περισσότερο διαθέσιμο προσωπικό και είναι πιθανότατα λιγότερο ακριβό για τεράστιους όγκους δεδομένων.

Spark vs Hadoop MapReduce: Συμβατότητα

Το Apache Spark μπορεί να εκτελεστεί ως αυτόνομη εφαρμογή, πάνω από το Hadoop YARN ή το Apache Mesos τοπικά ή στο cloud. Το Spark υποστηρίζει πηγές δεδομένων που υλοποιούν τη μορφή εισόδου Hadoop, ώστε να μπορεί να ενσωματωθεί με όλες τις πηγές δεδομένων και μορφές αρχείων που υποστηρίζει το Hadoop.

Συμπέρασμα: Η συμβατότητα του Spark με διάφορους τύπους δεδομένων και πηγές δεδομένων είναι ίδια με το Hadoop MapReduce.

Spark εναντίον Hadoop MapReduce: Επεξεργασία δεδομένων

Εκτός από την απλή επεξεργασία των δεδομένων, το Spark μπορεί επίσης να επεξεργάζεται γραφήματα και περιλαμβάνει τη βιβλιοθήκη μηχανικής εκμάθησης MLlib. Χάρη στην υψηλή του απόδοση, το Spark μπορεί να κάνει επεξεργασία σε πραγματικό χρόνο καθώς και επεξεργασία ανά παρτίδες. Το Spark προσφέρει μια πλατφόρμα που τα περιλαμβάνει όλα και που μπορεί να χρησιμοποιηθεί χωρίς διαχωρισμό των εργασιών σε διαφορετικές πλατφόρμες,

Το Hadoop MapReduce είναι εξαιρετικό για μαζική επεξεργασία. Για επεξεργασία σε πραγματικό χρόνο, προτείνεται η χρήση μιας άλλης πλατφόρμας, όπως το Impala ή το Apache Storm και για την επεξεργασία γραφημάτων, το Apache Giraph. Το MapReduce διέθετε το Apache Mahout για μηχανική εκμάθηση, αλλά έκτοτε καταργήθηκε με την έλευση του Spark.

Συμπέρασμα: Το Spark είναι ένα πολυεργαλείο επεξεργασίας δεδομένων, ενώ το Hadoop MapReduce κάνει τη διαφορά στην επεξεργασία ανά παρτίδες.

Spark εναντίον Hadoop MapReduce: Ανεκτικότητα Σφαλμάτων

Το Spark πραγματοποιεί επαναλήψεις ανά εργασία και υποθετική εκτέλεση, όπως και το MapReduce. Ωστόσο, το MapReduce έχει ένα μικρό πλεονέκτημα καθότι βασίζεται σε σκληρούς δίσκους και όχι στη μνήμη RAM. Εάν μια διαδικασία MapReduce διακοπεί στη μέση της εκτέλεσης, μπορεί να συνεχίσει από εκεί που σταμάτησε, ενώ το Spark θα πρέπει να ξεκινήσει την επεξεργασία από την αρχή.

Συμπέρασμα: Το Spark και το Hadoop MapReduce έχουν και τα δύο μεγάλη ανοχή σε σφάλματα, όμως το Hadoop MapReduce είναι ελαφρώς πιο ευέλικτο.

Spark εναντίον Hadoop MapReduce: Ασφάλεια

Όσον αφορά την ασφάλεια, το Spark είναι λιγότερο προηγμένο σε σύγκριση με το MapReduce καθώς ορίζεται ως "κλειστή" από προεπιλογή, γεγονός που μπορεί να δημιουργήσει ευάλωτο σημείο για επιθέσεις. Ο έλεγχος ταυτότητας στο Spark υποστηρίζεται για κανάλια RPC μέσω ενός κοινόχρηστου μυστικού κωδικού. Το Spark περιλαμβάνει την δυνατότητα καταγραφής συμβάντων και οι διεπαφές ιστού (Web UIs) μπορούν να προστατευθούν μέσω φίλτρων servlet javax. Επιπλέον, επειδή το Spark μπορεί να εκτελείται στο YARN και να χρησιμοποιεί HDFS, μπορεί να επωφεληθεί από έλεγχο ταυτότητας (authentication) Kerberos, δικαιώματα αρχείων HDFS και κρυπτογράφηση μεταξύ κόμβων.

Το Hadoop MapReduce μπορεί να επωφεληθεί από όλα τα οφέλη ασφάλειας του Hadoop και να ενσωματωθεί με έργα ασφάλειας, όπως το Knox Gateway και το Apache Sentry. Από την άλλη μεριά, οι προγραμματιστές του Spark θα πρέπει να βελτιώσουν οι ίδιοι την ασφάλεια του Spark.

Συμπέρασμα: Η ασφάλεια Spark εξακολουθεί να είναι λιγότερο ανεπτυγμένη σε σχέση με το MapReduce, το οποίο έχει περισσότερες δυνατότητες και έργα ασφαλείας.

Spark εναντίον Hadoop MapReduce: Τελικό Συμπέρασμα

- Το Apache Spark είναι δυνητικά 100 φορές ταχύτερο από το Hadoop MapReduce.
- Το Apache Spark χρησιμοποιεί μνήμη RAM και δεν συνδέεται με τη μέθοδο δύο σταδίων του Hadoop.
- Το Apache Spark λειτουργεί καλά για μικρότερα σύνολα δεδομένων που μπορούν όλα να χωρέσουν στη μνήμη RAM ενός διακομιστή.
- Το Hadoop είναι πιο οικονομικό για την επεξεργασία ογκωδών συνόλων δεδομένων.
- Το Apache Spark είναι πλέον δημοφιλέστερο από το Hadoop MapReduce.

Το Apache Spark αποτελεί το μέλλον της επεξεργασίας μεγάλων δεδομένων, αλλά εξακολουθούν να υπάρχουν περιπτώσεις χρήσης του Hadoop MapReduce.

Το Spark έχει εξαιρετική απόδοση και είναι ιδιαίτερα οικονομικό, χάρη στην επεξεργασία δεδομένων στη μνήμη. Είναι συμβατό με όλες τις πηγές δεδομένων και τις μορφές αρχείων του Hadoop, ενώ έχει επίσης ταχύτερη καμπύλη εκμάθησης, με φιλικά APIs διαθέσιμα για πολλές γλώσσες προγραμματισμού. Το Spark περιλαμβάνει ακόμη και δυνατότητες επεξεργασίας γραφημάτων και μηχανικής εκμάθησης.

Το Hadoop MapReduce είναι μια πιο ώριμη πλατφόρμα και δημιουργήθηκε ειδικά για την επεξεργασία σε παρτίδες. Το MapReduce μπορεί να είναι πιο οικονομικό από το Spark για εξαιρετικά μεγάλα σύνολα δεδομένων που δεν χωρούν στη μνήμη. Επιπλέον, το οικοσύστημα MapReduce είναι σήμερα μεγαλύτερο χάρη σε πολλά υποστηρικτικά έργα, εργαλεία και υπηρεσίες cloud.

Παρόλο που το Spark φαίνεται να έχει την πρωτιά, στα πιο συνήθη σενάρια δεν χρησιμοποιείται μόνο του. Εξακολουθεί να χρειάζεται το HDFS για την αποθήκευση των δεδομένων και ενίοτε ίσως υπάρξουν στιγμές που θα χρειαστούν και άλλες λειτουργίες του Hadoop, όπως το HBase, Hive, Pig και Impala. Ως άρτια λύση παραμένει το πλήρες πακέτο Hadoop με Spark για την επεξεργασία μεγάλων δεδομένων.

([66]), ([67])

ΚΕΦΑΛΑΙΟ 2

2.1 ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΔΟΚΙΜΗ ΤΟΥ APACHE SPARK

Αφού είδαμε κάποια πράγματα για το Spark, σε αυτό το κεφάλαιο θα εγκαταστήσουμε μια συστάδα από εικονικούς υπολογιστές με έναν κεντρικό κόμβο και δύο κόμβους "εργάτες".

Οι εικονικοί υπολογιστές θα χρησιμοποιούν λειτουργικό Linux.

Χρησιμοποιώντας το yarn ως Cluster Manager θα δοκιμάσουμε να τρέξουμε απλούς αλγόριθμους στο τερματικό του Spark, τόσο σε Scala, όσο και σε Python.

ΣΗΜΑΝΤΙΚΟ: Τα προγράμματα Hadoop και Spark είναι συμβατά και με το λειτουργικό σύστημα Windows της Microsoft. Για μεγαλύτερη ευελιξία όμως στην εγκατάσταση και τις ρυθμίσεις της συστάδας Spark, ο παρών οδηγός θα επικεντρωθεί σε περιβάλλον Linux.

Τα τεχνικά χαρακτηριστικά του υπολογιστή που χρησιμοποιήθηκε για τον οδηγό είναι τα εξής:

CPU	Memory	Disk
6 cores	16GB	1TB

Οι ελάχιστες απαιτήσεις για τον οδηγό αυτό είναι:

CPU	Memory	Disk
4 cores	8GB	250GB

ΣΗΜΑΝΤΙΚΟ: Εάν ο υπολογιστής του αναγνώστη έχει χαμηλότερες δυνατότητες από τις ελάχιστες απαιτήσεις, τότε θα πρέπει να ρυθμίσει το Hadoop διαφορετικά. Το ίδιο φυσικά ισχύει και για κάποιον που έχει αρκετούς διαθέσιμους πόρους και θέλει

να τους χρησιμοποιήσει. Παρακάτω, θα αναλυθούν κάποιες βασικές ρυθμίσεις του Hadoop.

2.2 ΠΡΟΕΤΟΙΜΑΣΙΑ – ΕΓΚΑΤΑΣΤΑΣΗ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ ΕΙΚΟΝΙΚΩΝ ΜΗΧΑΝΩΝ

Για να δημιουργήσουμε τους εικονικούς υπολογιστές θα πρέπει να κατεβάσουμε ένα λογισμικού εικονικοποίησης.

Περισσότερες πληροφορίες για τα εικονικά περιβάλλοντα μπορείτε να βρείτε στους παρακάτω συνδέσμους:

1) https://www.vmware.com/topics/glossary/content/virtual-machine.html

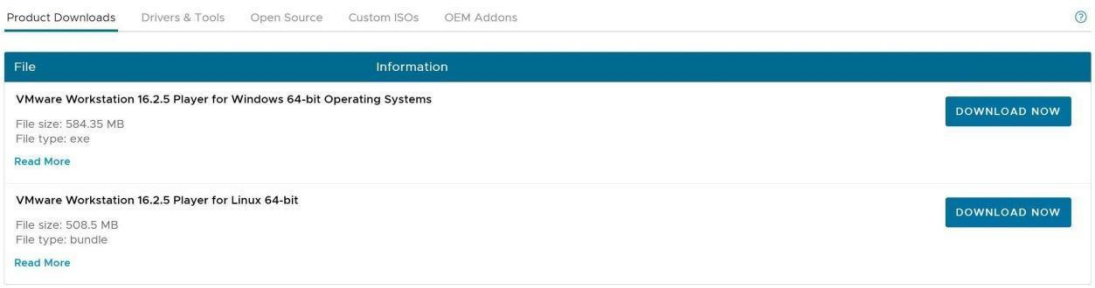
2) https://www.ibm.com/topics/virtual-machines

3) https://www.virtualbox.org/wiki/Virtualization

Οι πιο δημοφιλείς επιλογές είναι το VirtualBox της εταιρείας Oracle και το VMware Workstation Player της VMware.

Στον παρών οδηγό χρησιμοποιήθηκε το λογισμικό VMware Workstation Player το οποίο μπορούμε να το κατεβάσουμε από εδώ:

https://customerconnect.vmware.com/en/downloads/details?downloadGroup=WKST-PLAYER-1625&productId=1039&rPId=98562



Επιλέγουμε την αντίστοιχη έκδοση ανάλογα με το λειτουργικό του υπολογιστή μας.

ΓΙΑ WINDOWS: Η εγκατάσταση γίνεται με τις προτεινόμενες ρυθμίσεις του προγράμματος εγκατάστασης [VMware-player-full-xx.x.x-xxxxxxxx.exe].

ΓΙΑ LINUX:

1) Αφού κατεβάσουμε το αρχείο εγκατάστασης [VMware-Player-Full-xx.x.x-xxxxxxxx.x86_64.bundle] ανοίγουμε το τερματικό (terminal) στην τοποθεσία που βρίσκεται το αρχείο (π.χ. ~/home/$USER/Downloads).
2) Τρέχουμε τις παρακάτω εντολές:

```
chmod +x VMware-Player-Full-xx.x.x-xxxxxxxx.x86_64.bundle

sudo ./VMware-Player-Full-xx.x.x-xxxxxxxx.x86_64.bundle
```

3) Τέλος, εντοπίζουμε το πρόγραμμα στα εγκατεστημένα προγράμματα και αυτόματα εκτελείται το πρόγραμμα εγκατάστασης με εικονικό περιβάλλον όπως στο λειτουργικό Windows.

Έπειτα θα προχωρήσουμε στην λήψη ειδώλου δίσκου (ISO) για το λειτουργικό Linux που θα χρησιμοποιήσουμε.

Η διανομή (distribution ή distro) που θα χρησιμοποιήσουμε είναι η Debian. Στον παρακάτω σύνδεσμο υπάρχει η τελευταία έκδοση (version) της Debian από το επίσημο site:

https://www.debian.org/distrib/netinst

Ο παρών οδηγός έχει χρησιμοποιήσει την διανομή **Debian** και πιο συγκεκριμένα την **έκδοση 11** "**Bullseye**" που μπορεί να βρεθεί εδώ:

https://cdimage.debian.org/mirror/cdimage/archive/11.3.0/

Για να βρούμε την αρχιτεκτονική της κεντρικής μονάδας επεξεργασιας (CPU) του μηχανήματός μας μπορούμε να συμβουλευτούμε τους παρακάτω οδηγούς:

ΓΙΑ WINDOWS: https://pcguide101.com/cpu/what-is-my-processor-architecture/

ΓΙΑ LINUX: https://linuxopsys.com/topics/get-cpu-architecture-in-linux-system

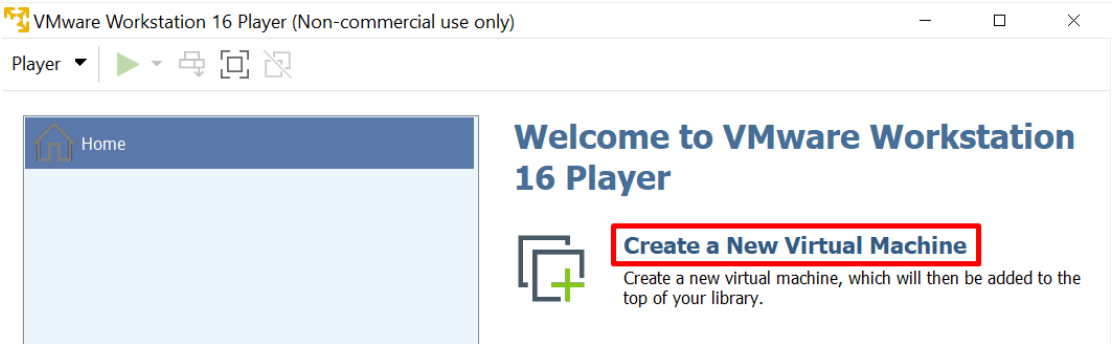
ΣΗΜΕΙΩΣΗ: . Ενώ τα λογισμικά Hadoop και Spark μπορούν να τρέξουν σε οποιαδήποτε πλατφόρμα Linux βασισμένη σε Ubuntu και Debian, ο αναγνώστης προτείνεται να εγκαταστήσει την διανομή με την οποία αισθάνεται πιο οικεία.

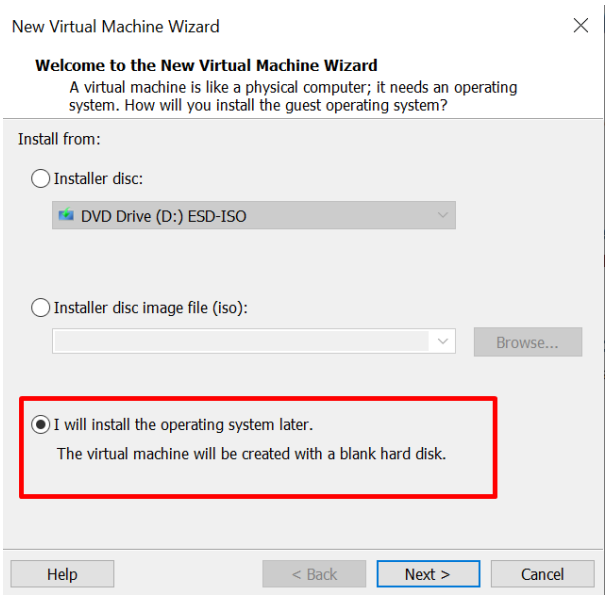
2.3 ΒΗΜΑ 1 - ΔΗΜΙΟΥΡΓΙΑ ΤΩΝ ΕΙΚΟΝΙΚΩΝ ΜΗΧΑΝΩΝ

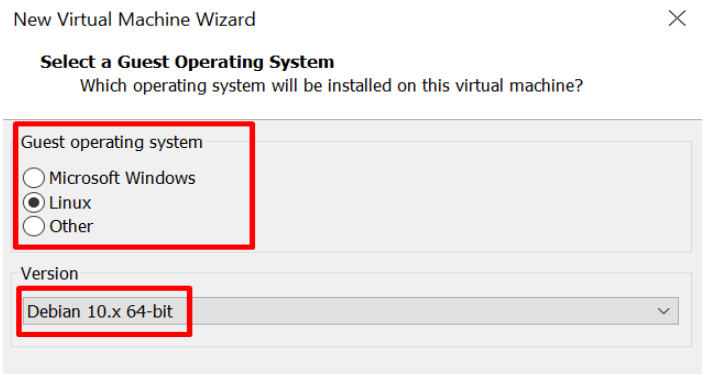
Αφού έχουμε εγκαταστήσει το λογισμικό VMware προχωράμε στην δημιουργία της πρώτης εικονικής μηχανής. Η μηχανή αυτή θα αποτελέσει την βάση από την οποία αργότερα θα δημιουργήσουμε μια συστάδα εικονικών μηχανών που θα συνδέονται μέσω του τοπικού δικτύου.

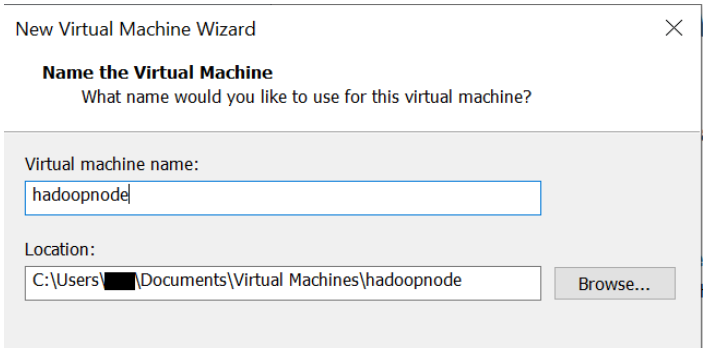
Τα χαρακτηριστικά της εικονικής μηχανής που θα δημιουργήσουμε είναι τα εξής:

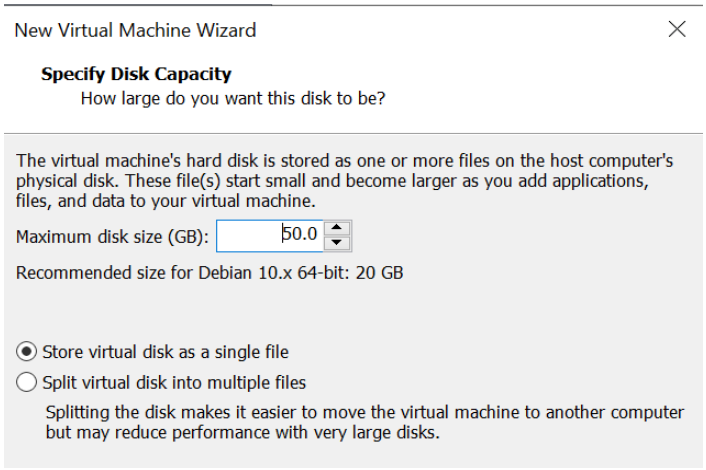
CPU	Memory	Disk	Network
2 cores	3GB	50 GB	Bridged

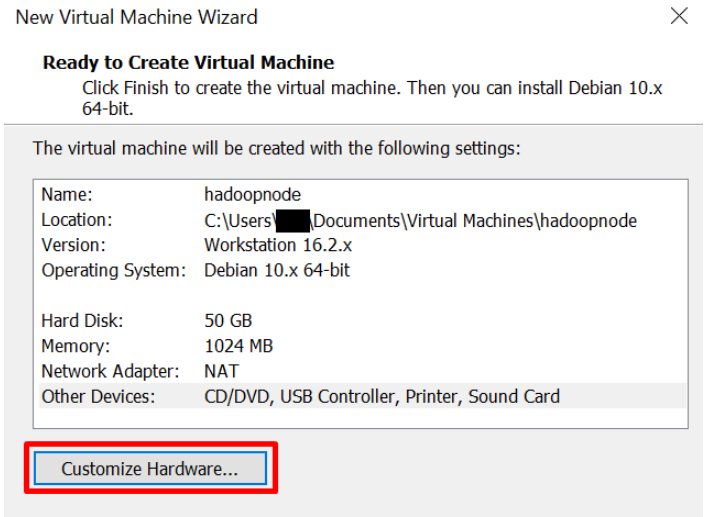


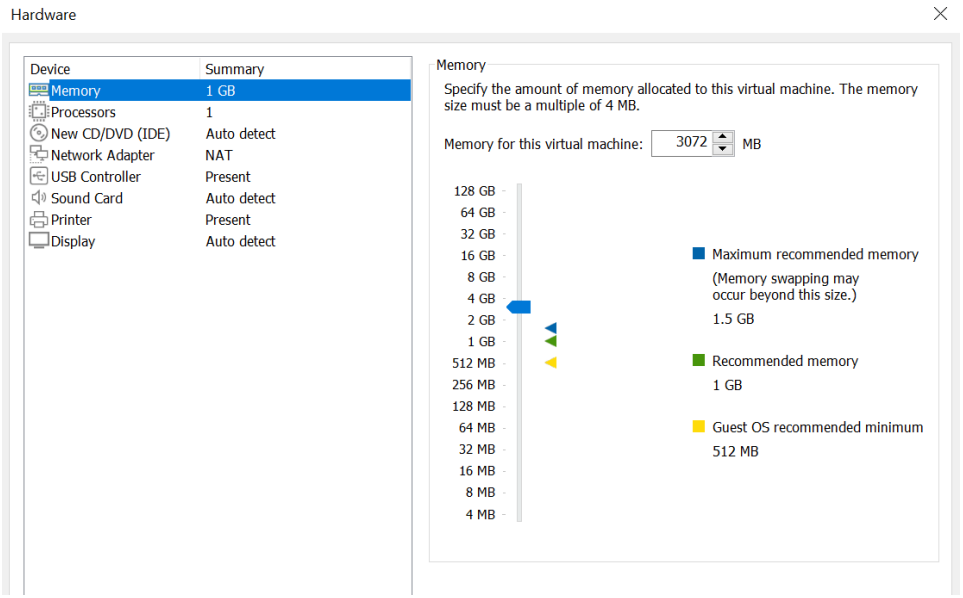


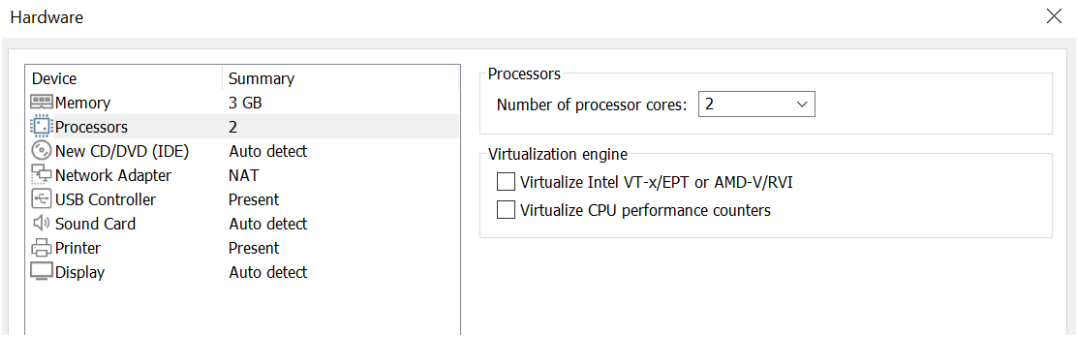


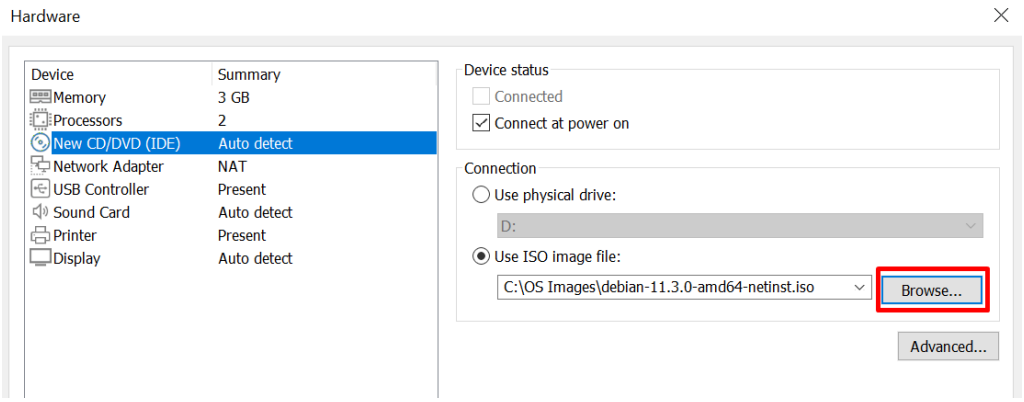

New Virtual Machine Wizard
Name the Virtual Machine
What name would you like to use for this virtual machine?
Virtual machine name:
hadoopnode
Location:
C:\Users\[redacted]\Documents\Virtual Machines\hadoopnode
Browse...

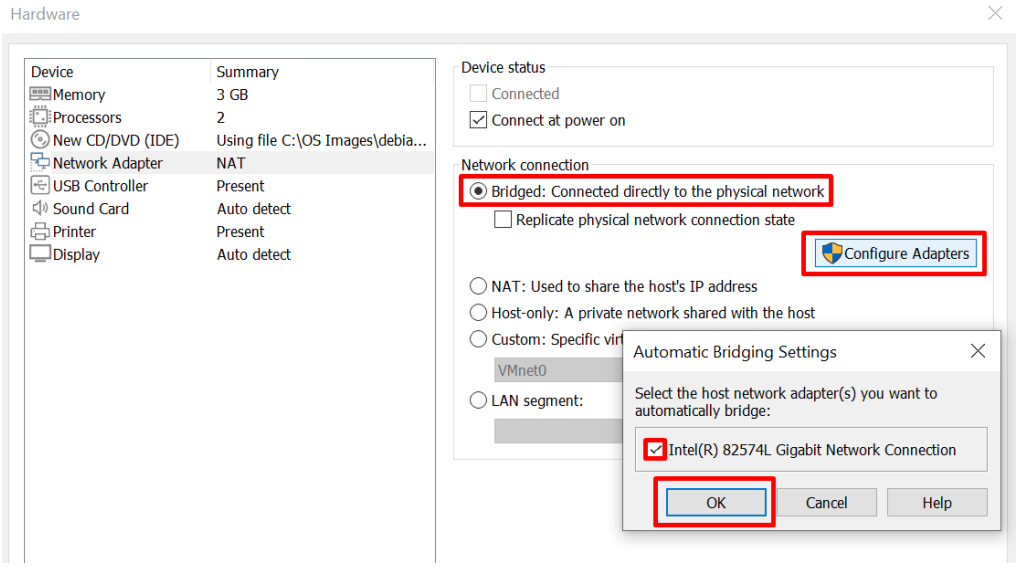

New Virtual Machine Wizard
Specify Disk Capacity
How large do you want this disk to be?
The virtual machine's hard disk is stored as one or more files on the host computer's physical disk. These file(s) start small and become larger as you add applications, files, and data to your virtual machine.
Maximum disk size (GB): 50.0
Recommended size for Debian 10.x 64-bit: 20 GB
Store virtual disk as a single file
Split virtual disk into multiple files
Splitting the disk makes it easier to move the virtual machine to another computer but may reduce performance with very large disks.

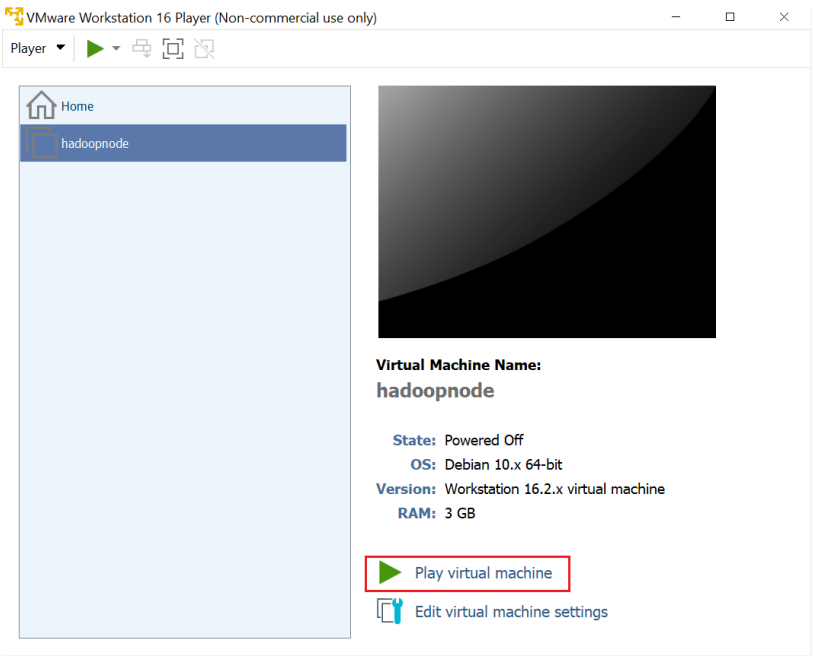

New Virtual Machine Wizard
Ready to Create Virtual Machine
Click Finish to create the virtual machine. Then you can install Debian 10.x 64-bit.
The virtual machine will be created with the following settings:
Name: hadoopnode
Location: C:\Users\[redacted]\Documents\Virtual Machines\hadoopnode
Version: Workstation 16.2.x
Operating System: Debian 10.x 64-bit
Hard Disk: 50 GB
Memory: 1024 MB
Network Adapter: NAT
Other Devices: CD/DVD, USB Controller, Printer, Sound Card
Customize Hardware...


Hardware
Device
Summary
Memory
1 GB
Processors
1
New CD/DVD (IDE)
Auto detect
Network Adapter
NAT
USB Controller
Present
Sound Card
Auto detect
Printer
Present
Display
Auto detect
Memory
Specify the amount of memory allocated to this virtual machine. The memory size must be a multiple of 4 MB.
Memory for this virtual machine:
3072
MB
128 GB
64 GB
32 GB
16 GB
8 GB
4 GB
2 GB
1 GB
512 MB
256 MB
128 MB
64 MB
32 MB
16 MB
8 MB
4 MB
Maximum recommended memory
(Memory swapping may occur beyond this size.)
1.5 GB
Recommended memory
1 GB
Guest OS recommended minimum
512 MB


Hardware
Device
Summary
Memory
3 GB
Processors
2
New CD/DVD (IDE)
Auto detect
Network Adapter
NAT
USB Controller
Present
Sound Card
Auto detect
Printer
Present
Display
Auto detect
Processors
Number of processor cores:
2
Virtualization engine
Virtualize Intel VT-x/EPT or AMD-V/RVI
Virtualize CPU performance counters


Hardware
Device
Summary
Memory
3 GB
Processors
2
New CD/DVD (IDE)
Auto detect
Network Adapter
NAT
USB Controller
Present
Sound Card
Auto detect
Printer
Present
Display
Auto detect
Device status
Connected
Connect at power on
Connection
Use physical drive:
D:
Use ISO image file:
C:\OS Images\debian-11.3.0-amd64-netinst.iso
Browse...
Advanced...


Hardware
Device
Summary
Memory
3 GB
Processors
2
New CD/DVD (IDE)
Using file C:\OS Images\debia...
Network Adapter
NAT
USB Controller
Present
Sound Card
Auto detect
Printer
Present
Display
Auto detect
Device status
Connected
Connect at power on
Network connection
Bridged: Connected directly to the physical network
Replicate physical network connection state
Configure Adapters
NAT: Used to share the host's IP address
Host-only: A private network shared with the host
Custom: Specific virt
VMnet0
LAN segment:
Automatic Bridging Settings
Select the host network adapter(s) you want to automatically bridge:
Intel(R) 82574L Gigabit Network Connection
OK
Cancel
Help


VMware Workstation 16 Player (Non-commercial use only)
Player
Home
hadoopnode
Virtual Machine Name:
hadoopnode
State: Powered Off
OS: Debian 10.x 64-bit
Version: Workstation 16.2.x virtual machine
RAM: 3 GB
Play virtual machine
Edit virtual machine settings

2.4 ΒΗΜΑ 2 - ΕΓΚΑΤΑΣΤΑΣΗ ΤΟΥ ΛΕΙΤΟΥΡΓΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΤΩΝ ΕΙΚΟΝΙΚΩΝ ΜΗΧΑΝΩΝ

Όταν εκκινήσουμε την εικονική μηχανή που δημιουργήσαμε παρατηρούμε πως η μηχανή διαβάζει το είδωλο δίσκου που τοποθετήσαμε στην εικονική είσοδο οπτικού μέσου (optical drive) καθώς δεν υπάρχει ακόμη εγκατεστημένο κάποιο λειτουργικό σύστημα.

Στο μενού εγκατάστασης που εμφανίζεται επιλέγουμε “Graphical Install” και ο γραφικός οδηγός εγκατάστασης ξεκινά ο οποίος μας κατευθύνει βήμα - βήμα.

Σε κάποιες ενότητες του οδηγού εγκατάστασης δεν έχει σημασία η επιλογή που θα κάνουμε, όπως η επιλογή της χώρας κατά την αρχή, έτσι ο οδηγός αυτός θα εστιάσει στα σημεία που πρέπει να δοθεί προσοχή, έτσι ώστε το αποτέλεσμα να συμβαδίζει με αυτό του οδηγού αυτού.

1. Στην ενότητα “Configure the keyboard” επιλέγουμε “American English”.
2. Στην ενότητα “Configure the network” πληκτρολογούμε “hadoopnode”.
3. Στην ενότητα “Domain name” αφήνουμε την ονομασία κενή.
4. Στην ενότητα “Set up users and passwords” ορίζουμε κωδικό για τον χρήστη root της εικονικής μηχανής. Επιλέγουμε έναν κωδικό που θα θυμόμαστε.

   Περισσότερες λεπτομέρειες για την έννοια του χρήστη root υπάρχουν εδώ:

   https://www.ibm.com/docs/en/aix/7.2?topic=passwords-root-account

5. Στην επόμενη σελίδα της ενότητας “Set up users and passwords” επιλέγουμε το όνομα χρήστη με τον οποίο θα δουλεύουμε (εκτός του χρήστη - διαχειριστή root). Πληκτρολογούμε “hadoop”.
6. Στην ενότητα “Partition disks” επιλέγουμε “Guided - use entire disk”.
7. Στο σημείο της ενότητας “Partition disks” σχετικά με το σχήμα των κατατμήσεων (partitions) του εικονικού δίσκου, επιλέγουμε “All files in one partition” και στην επόμενη σελίδα επιβεβαιώνουμε τις αλλαγές στον εικονικό δίσκο.
8. Στην ενότητα “Configure the package manager” επιλέγουμε “deb.debian.org”
9. Στην ενότητα “Software Selection” επιλέγουμε:
   a. Debian desktop environment
   b. Xfce
   c. SSH server

d. standard system utilities

10. Αφού πραγματοποιηθεί η εγκατάσταση του λειτουργικού, στην ενότητα "Install the GRUB boot loader" επιλέγουμε "Yes"

ΣΗΜΑΝΤΙΚΟ: Κατά την διάρκεια του οδηγού αυτού, ο αναγνώστης θα παρατηρήσει πως κάποιες εντολές ή ρυθμίσεις είναι γραμμένες με μωβ χρώμα. Η ένδειξη αυτή σημαίνει πως το σημείο εκείνο μπορεί να είναι διαφορετικό για κάποιον χρήστη που επιθυμεί να μην ακολουθήσει τον οδηγό κατά γράμμα.

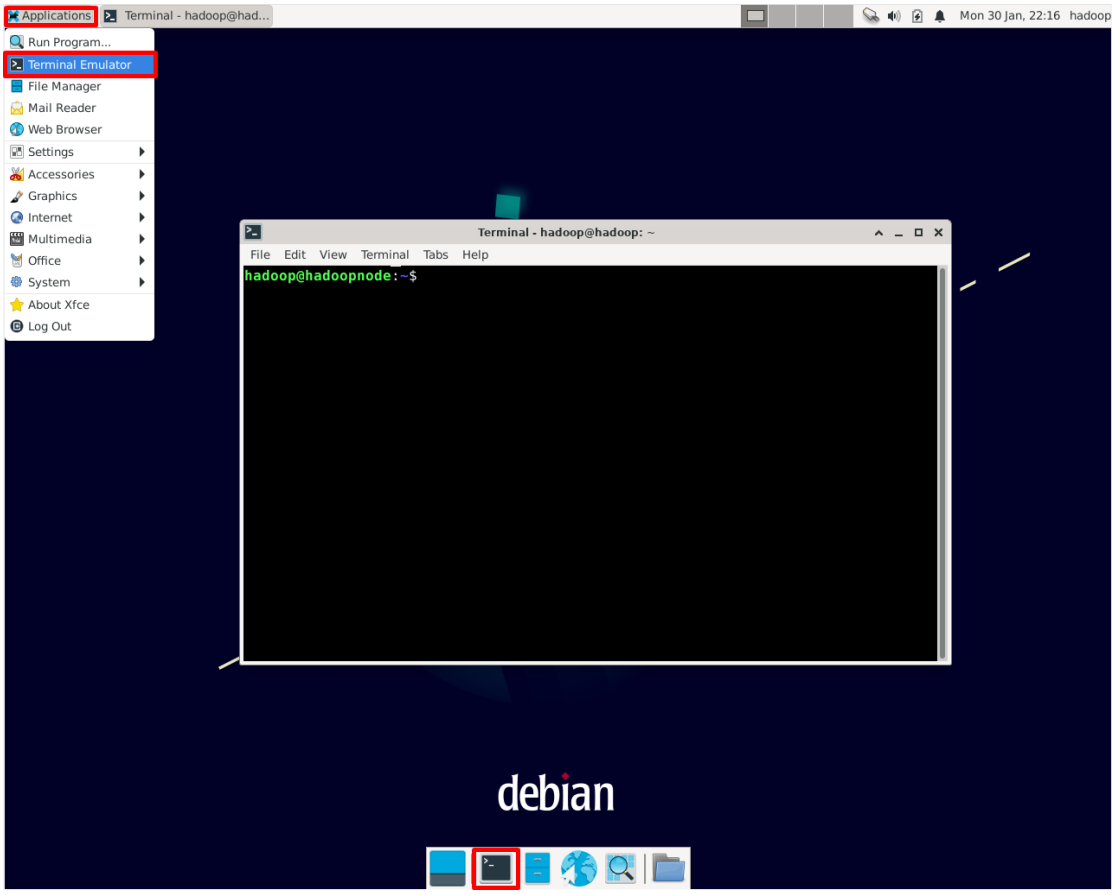
2.5 ΒΗΜΑ 3 - ΡΥΘΜΙΣΗ ΤΟΥ ΛΕΙΤΟΥΡΓΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΤΩΝ ΕΙΚΟΝΙΚΩΝ ΜΗΧΑΝΩΝ

Πλέον ο αρχικός μας κόμβος - εικονική μηχανή απέκτησε λειτουργικότητα. Το επόμενο βήμα είναι να κάνουμε κάποιες βασικές ρυθμίσεις στον κόμβο.

Τα βήματα που έχουν χαρακτηριστεί με αστερίσκο [*] θα πρέπει να επαναληφθούν αργότερα για κάθε κόμβο ξεχωριστά στη συστάδα.

Σε κάθε περιβάλλον Linux η πρώτη επαφή συνήθως που έχουμε είναι με την γραμμή εντολών (command line) την οποία και θα χρησιμοποιούμε στο μεγαλύτερο μέρος των ενεργειών που θα κάνουμε στην συνέχεια.

Για να αποκτήσουμε πρόσβαση στην γραμμή εντολών και στο κέλυφος (Shell) του Linux ανοίγουμε την εφαρμογή "Terminal".



Όπως βλέπουμε στην παραπάνω εικόνα, στο τερματικό (terminal) έχουμε το όνομα χρήστη (username) με το οποίο συνδεθήκαμε και το όνομα του οικοδεσπότη (hostname). Το δεύτερο αναφέρεται στον κόμβο - εικονική μηχανή.

Στην γραμμή εντολών λοιπόν πληκτρολογούμε τις εξής εντολές και πατάμε το πλήκτρο “Enter”:

➢ su

Σε αυτό το σημείο θα γίνει προτροπή για να κωδικό. Ο κωδικός αυτός είναι ο κωδικός root που επιλέξατε κατά την εγκατάσταση. Στο περιβάλλον Linux πριν την εκτέλεση μιας εντολής που χρειάζεται δικαιώματα διαχειριστή, την πρώτη φορά γίνεται προτροπή για κωδικό root. Συνήθως οι εντολές αυτές ξεκινούν με “sudo” το οποίο σημαίνει “superuser do”. Στην προκειμένη περίπτωση αυτό δεν αναγράφεται καθώς με την εντολή “su” πραγματοποιούμε είσοδο στο σύστημα απευθείας ως χρήστης root που βρίσκεται στην ομάδα διαχειριστών του συστήματος (sudoers).

Σκοπός αυτού είναι να εντάξουμε τον χρήστη hadoop στην ομάδα sudoers και να του εκχωρήσουμε δικαίωμα εκτέλεσης εντολών “sudo”.

➢ apt install open-vm-tools

Με την εντολή αυτή, το πακέτο open-vm-tools της VMware εγκαθίσταται ως μέρος του λειτουργικού συστήματος Linux.

Ο χρήστης root δεν χρειάζεται πάντα το “sudo” για την εκτέλεση μιας εντολής με δικαιώματα διαχειριστή όπως η εντολή “apt install”.

Λεπτομέρειες για το πακέτο open-vm-tools μπορείτε να βρείτε στην επίσημο site της VMware: https://docs.vmware.com/en/VMware-Tools/12.1.0/com.vmware.vsphere.vmwaretools.doc/GUID-8B6EA5B7-453B-48AA-92E5-DB7F061341D1.html

➢sudo adduser hadoop sudo

Ο χρήστης hadoop αποκτά δυνατότητα εκτέλεσης εντολών “sudo”.

➢ sudo reboot

Πραγματοποιείται επανεκκίνηση για να εφαρμοστεί η τοποθέτηση του χρήστη στην ομάδα sudo.

Για λόγους ασφαλείας δεν προτείνεται η εκτέλεση εντολών με τον χρήστη root καθώς έχει πρόσβαση σε όλα τα σημεία του συστήματος. Έτσι αν γίνει κάποιο σφάλμα μπορεί να καταστήσει άχρηστο όλο το λειτουργικό σύστημα.

➢ sudo apt update

Πραγματοποιεί αναβάθμιση στις πληροφορίες των πακέτων συστήματος Linux.

Σε αυτό το σημείο θα εγκαταστήσουμε την έκδοση 11 του προγραμματιστικού πακέτου Java Development Kit.

Το Java Development Kit ενσωματώνει τις Προδιαγραφές Γλώσσας Java (Java Language Specification - JLS) και τις Προδιαγραφές Εικονικών Μηχανών Java (Java Virtual Machine Specification - JVMS) και παρέχει την Συμβατική Έκδοση (Standard Edition - SE) της Προγραμματιστικής Διεπαφής Εφαρμογών Java (Java API).

Το Spark, όπως και το Hadoop δεν είναι συμβατά με όλες τις εκδόσεις της γλώσσας Java. Την χρονική περίοδο που γράφτηκε ο παρών οδηγός, οι συμβατές εκδόσεις είναι η 8 και η 11.

```
➤ sudo apt install openjdk-11-jdk
```

Μπορούμε να επαληθεύσουμε την εγκατάσταση με την εντολή:

```
➤ java -version
```

Εάν δεν εμφανιστεί κάποιο σφάλμα και μπορούμε να δούμε την έκδοση της Java τότε η εγκατάσταση έγινε επιτυχώς.

Για να γνωρίζει το λειτουργικό ανά πάσα στιγμή που βρίσκεται εγκατεστημένη η Java, θα πρέπει να κάνουμε export κάποιες περιβαλλοντικές μεταβλητές.

Το αρχείο του Linux που αναλαμβάνει αυτήν την δουλειά είναι ένα κρυφό αρχείο με το όνομα **bashrc.**

Για να επεξεργαστούμε το αρχείο bashrc τρέχουμε την εντολή

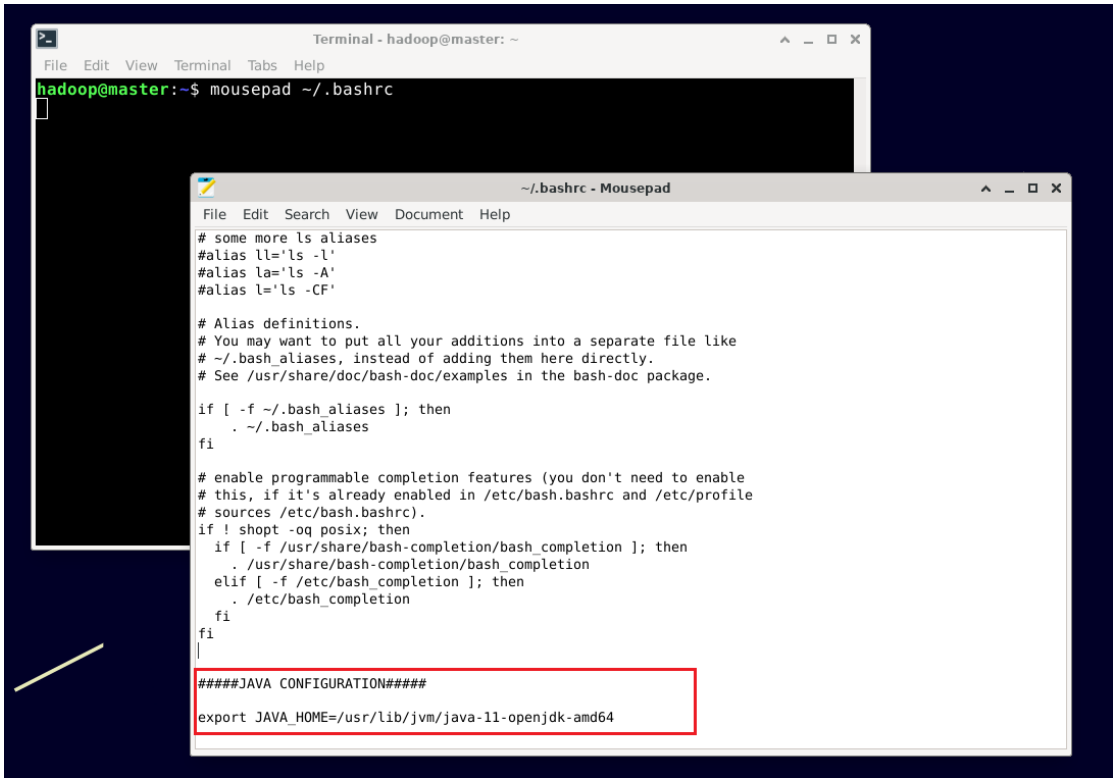
```
➤ mousepad ~/.bashrc
```

από οποιοδήποτε directory και χρησιμοποιώντας οποιαδήποτε εφαρμογή editing επιθυμούμε.

Στο τέλος του αρχείου bashrc προσθέτουμε τις εξής γραμμές:

```
#####JAVA CONFIGURATION#####

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```



Όταν προσθέσουμε εντολές ή μεταβλητές στο αρχείο bashrc, αφού αποθηκεύσουμε τις αλλαγές που κάνουμε, τρέχουμε την εντολή source ~/.bashrc για να θέσουμε σε άμεση εφαρμογή τις αλλαγές αυτές χωρίς να χρειάζεται επανεκκίνηση του συστήματος. Υπάρχουν βέβαια και περιπτώσεις όπου είναι προτιμότερο να κάνουμε επανεκκίνηση του συστήματος άσχετα με το αν έχουμε τρέξει την εντολή “source”.

➢ sudo apt-get install python3-pip

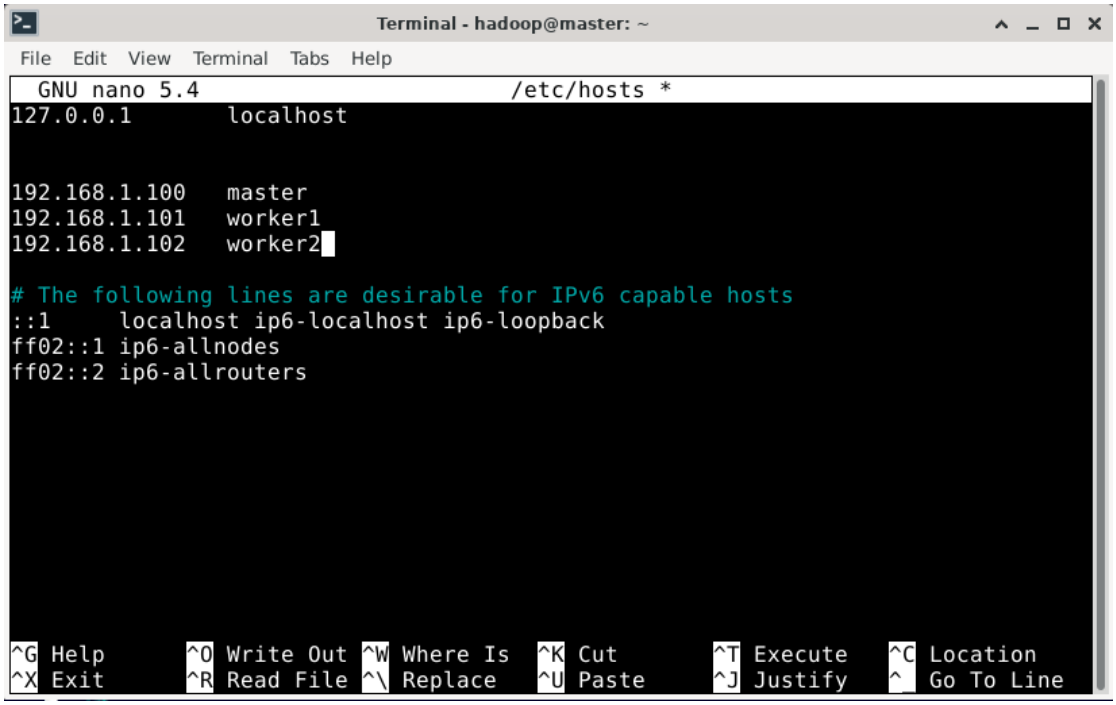
Το pip είναι ένα εργαλείο γραμμένο σε γλώσσα Python που χρησιμοποιείται για την εγκατάσταση και διαχείριση πακέτων. Θα χρειαστεί αργότερα για να εγκαταστήσουμε βιβλιοθήκες που βρίσκουμε και τις οποίες θέλουμε να κάνουμε εισαγωγή (import) στον κώδικα που έχουμε φτιάξει.

➢ sudo nano /etc/hosts

Εδώ ενημερώνουμε τον πίνακα διευθύνσεων της εικονικής μηχανής. Έτσι η εικονική μηχανή γνωρίζει ότι για να απευθυνθεί στον κόμβο worker1 θα πρέπει να χρησιμοποιήσει την διεύθυνση 192.168.1.101 .

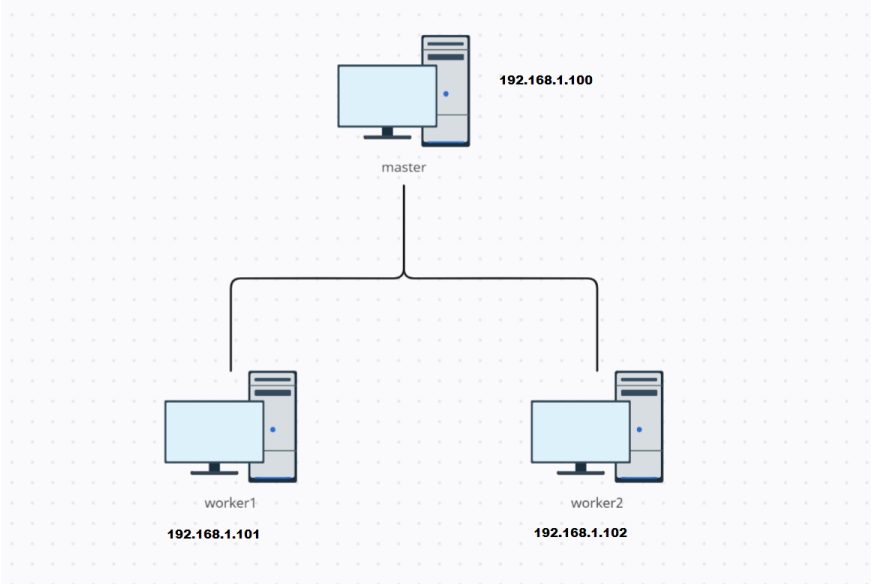
Σημειώνουμε προκαταβολικά τις διευθύνσεις που θέλουμε να χρησιμοποιήσουμε στην συστάδα.

Εάν κάνουμε το οποιοδήποτε λάθος ή θελήσουμε να αλλάξουμε κάτι μπορούμε να επαναλάβουμε το βήμα αυτό οποιαδήποτε στιγμή.



Πατάμε ctrl x και y για αποθήκευση.

Αυτό είναι το σημείο όπου οι διαδικασίες που έχουν επισημανθεί με αστερίσκο [*] θα πρέπει να επαναληφθούν για κάθε κόμβο ξεχωριστά ούτως ώστε στο τέλος να έχουμε το εξής δίκτυο από εικονικές μηχανές - κόμβους:



Αρχικά δημιουργούμε τον κόμβο master (κόμβος Αρχηγός).

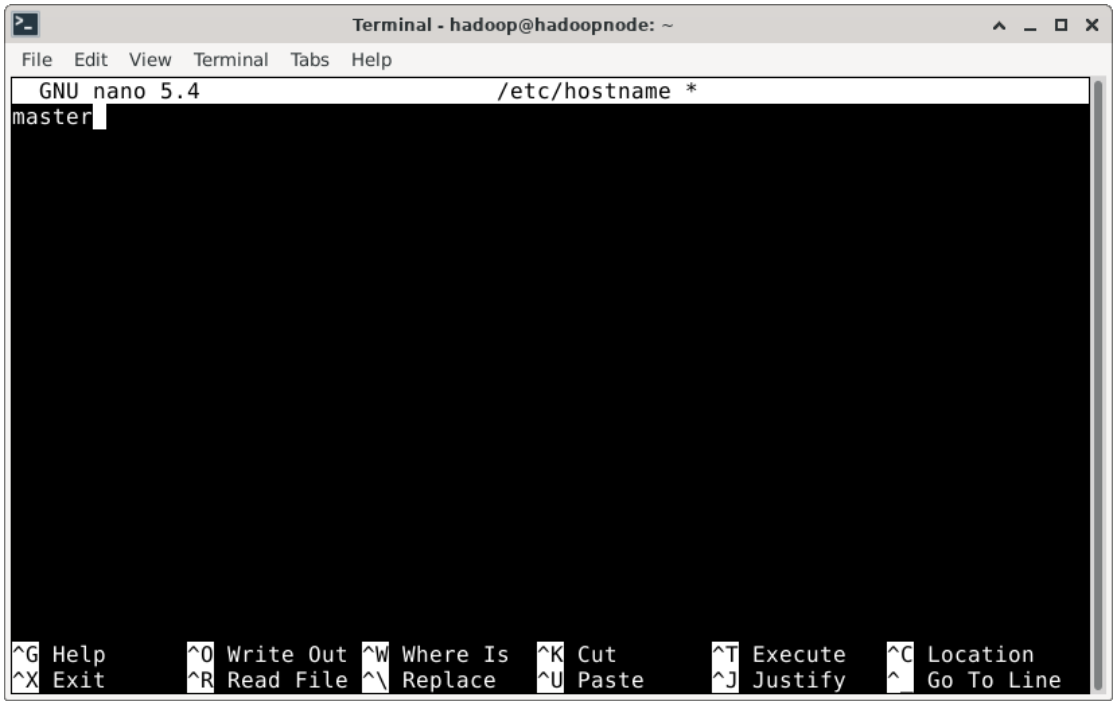
Όταν φτάσουμε στο σημείο δημιουργίας συστάδας αργότερα, τότε επαναλαμβάνουμε την διαδικασία για τους κόμβους worker1 και worker2 (κόμβοι Εργάτες).

Στη θέση του “master” βάζουμε “worker1” / “worker2”.

Στη θέση της διεύθυνσης “192.168.1.100” βάζουμε “192.168.1.101” και “192.168.1.102” αντίστοιχα.

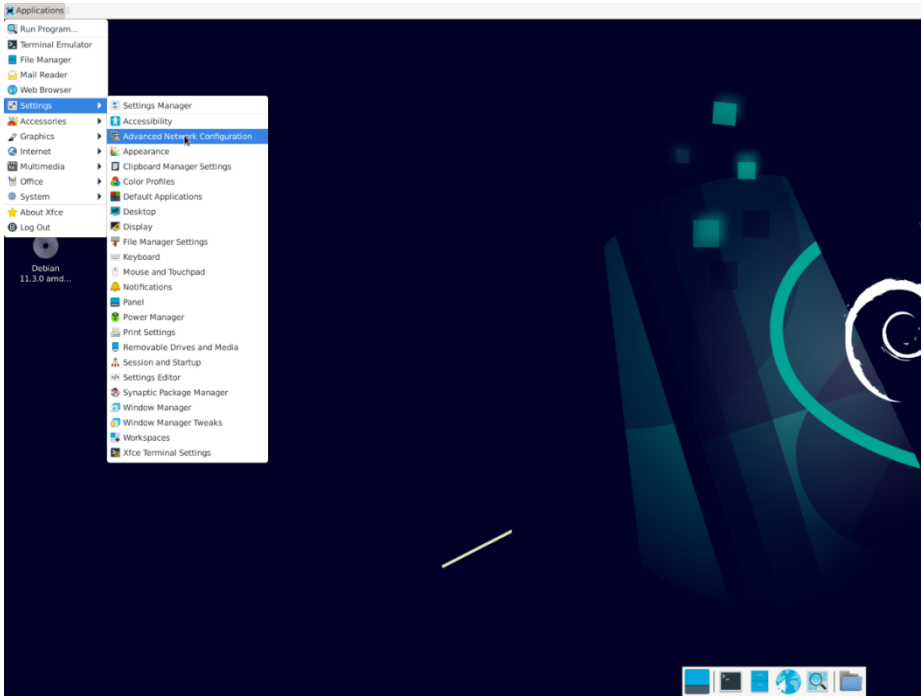
```
[*] ➤ sudo nano /etc/hostname
```

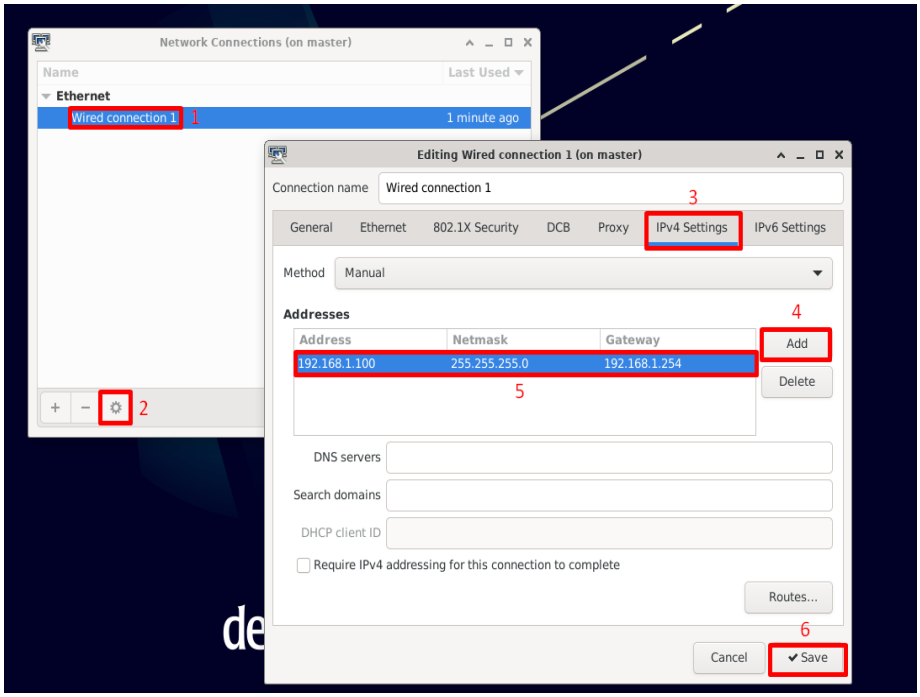
Γίνεται αλλαγή του ονόματος host (όνομα εικονικής μηχανής)



Πατάμε ctrl x και y για αποθήκευση.

[*] Αφού αλλάξαμε το όνομα της μηχανής, το επόμενο βήμα είναι να εκχωρήσουμε την στατική διεύθυνση ip που αντιστοιχεί στον κόμβο σύμφωνα με το σχέδιο:





ΣΗΜΑΝΤΙΚΟ: Η διεύθυνση στο πεδίο Gateway είναι η διεύθυνση του router που χρησιμοποιείτε στο τοπικό σας δίκτυο.

2.6 ΒΗΜΑ 4 - ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΡΥΘΜΙΣΗ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ APACHE HADOOP

Σε αυτό το σημείο έχουμε έναν κόμβο τον οποίο αργότερα θα χρησιμοποιήσουμε για να δημιουργήσουμε την συστάδα Spark. Ονομάσαμε τον κόμβο αυτό “master” και θα αποτελέσει τον κόμβο Αρχηγό.

Μέχρι στιγμής όμως ο κόμβος αυτός δεν διαθέτει κανένα εργαλείο για να διαχειριστεί τον οποιοδήποτε όγκο δεδομένων. Πριν όμως εγκαταστήσουμε το Spark που είναι το εργαλείο που θα μας βοηθήσει με αυτήν την δουλειά θα προετοιμάσουμε το κατανεμημένο σύστημα αποθήκευσης Hadoop και τον διαχειριστή πόρων YARN που θα μας επιτρέψουν να τρέχουμε τις εφαρμογές Spark παράλληλα εκμεταλλευόμενοι τους διαθέσιμους πόρους όλων των κόμβων της συστάδας.

Για να κατεβάσουμε το Apache Hadoop θα πρέπει να επισκεφθούμε τον παρακάτω σύνδεσμο:

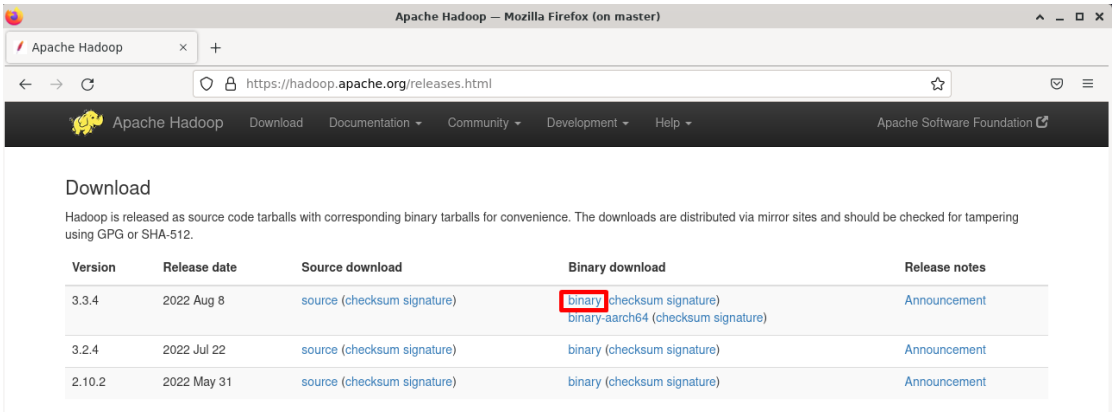
https://hadoop.apache.org/releases.html

ΣΗΜΕΙΩΣΗ: Εάν ο αναγνώστης επιθυμεί να κατεβάσει την έκδοση που χρησιμοποιήθηκε στον παρών οδηγό θα την βρει εδώ:

https://hadoop.apache.org/release/3.3.4.html

Και θα πρέπει να προχωρήσει εδώ

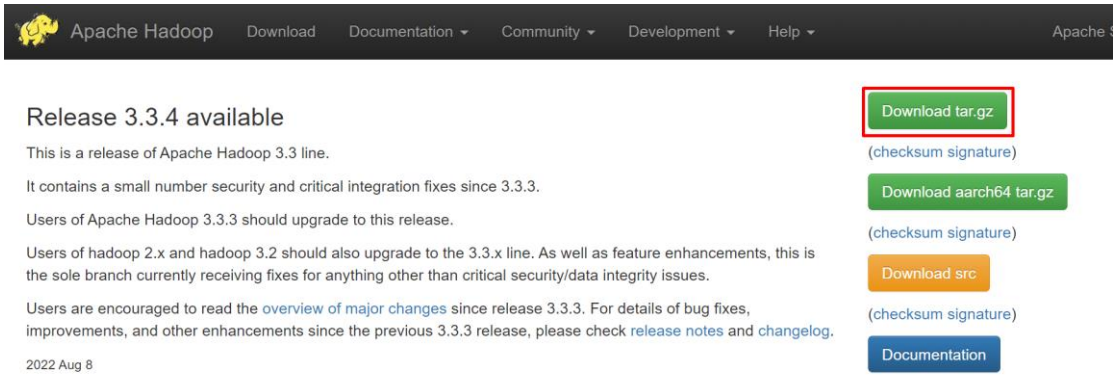
Στην ενότητα "Downloads" θα βρούμε μια σελίδα με τις τελευταίες εκδόσεις σε διαφορετικές μορφές αρχείων που μπορούμε να κατεβάσουμε.



Επιλέγουμε το αρχείο binary της τελευταίας έκδοσης.

Στην επόμενη σελίδα κάνουμε δεξί κλικ στον πρώτο σύνδεσμο ή στον σύνδεσμο κάτω από την ένδειξη "**HTTP**" και επιλέγουμε αντιγραφή συνδέσμου (Copy link address).

Για όσους επέλεξαν την έκδοση Hadoop του οδηγού, τότε στην σελίδα που οδηγεί ο σύνδεσμος κάνουμε δεξί κλικ στο κουμπί με την ένδειξη "**Download tar.gz**" και επιλέγουμε αντιγραφή συνδέσμου (Copy link address).



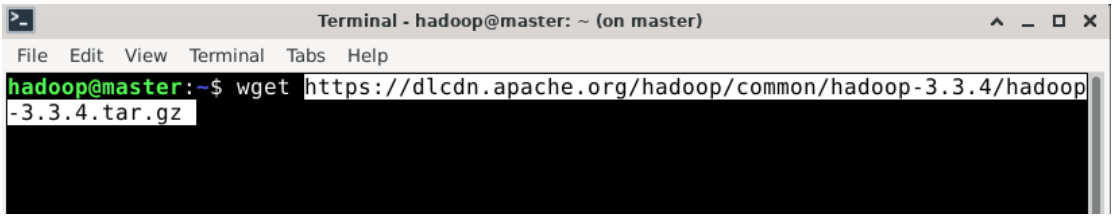
Έπειτα ανοίγουμε το terminal στο home directory (/home/hadoop/).

Για περισσότερες λεπτομέρειες σχετικά με τα μονοπάτια (paths) στο σύστημα αρχείων Linux μπορούμε να βρούμε εδώ:

https://linuxhandbook.com/linux-directory-structure/

Σημείωση: Από προεπιλογή όταν ανοίγουμε ένα instance του terminal το μονοπάτι οδηγεί στο home directory.

➤ wget [Σύνδεσμος για το hadoop που αντιγράψαμε πριν]

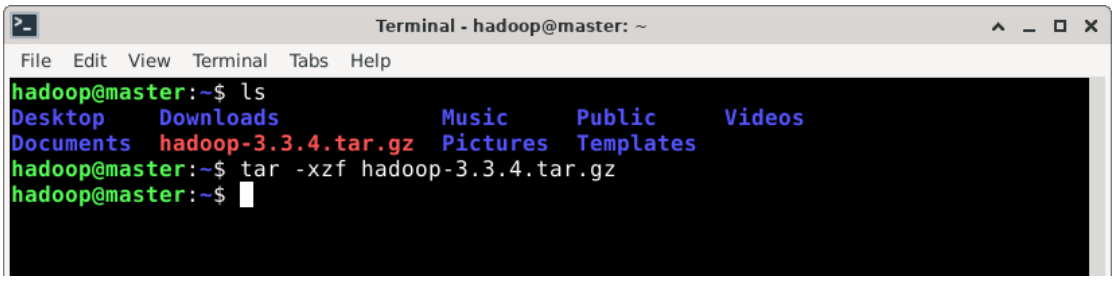


Περιμένουμε έως ότου κατέβει το συμπιεσμένο αρχείο που θα χρησιμοποιήσουμε για την εγκατάσταση. Για να βεβαιωθούμε ότι κατέβηκε επιτυχώς μπορούμε να χρησιμοποιήσουμε την εντολή:

➢ ls

Η εντολή ls εμφανίζει στο τερματικό όλα τα αρχεία και τους φακέλους που βρίσκονται στο τρέχων μονοπάτι. Εάν δούμε ένα αρχείο με την εξής μορφή **hadoop-x.x.x.tar.gz** σημαίνει πως όλα πήγαν καλά και μπορούμε να προχωρήσουμε στην εξαγωγή του φακέλου με τα αρχεία hadoop από το συμπιεσμένο αρχείο που κατεβάσαμε.

Στην παρακάτω εικόνα βλέπουμε τις εντολές που πρέπει να χρησιμοποιήσουμε:

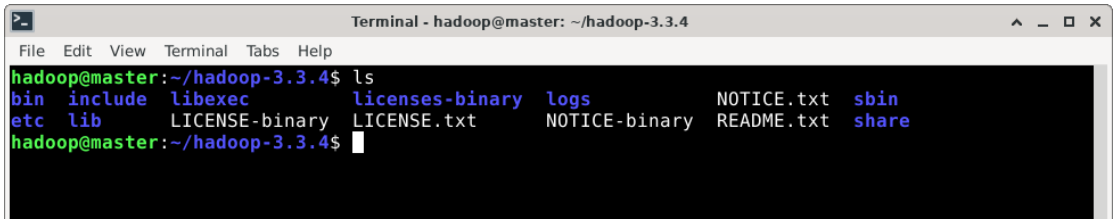


Αυτή ήταν η εγκατάσταση του Hadoop και αυτό ήταν το εύκολο μέρος. Πάμε τώρα στην ρύθμιση της συστάδας.

ΣΗΜΑΝΤΙΚΟ: Στις εντολές cd που χρησιμοποιούνται για αλλαγή path/directory ο αναγνώστης θα πρέπει να κάνει τις απαραίτητες αλλαγές ανάλογα με τυχόν διαφορές στην εγκατάσταση των προγραμμάτων ή στις εκδόσεις των προγραμμάτων.

Από το home directory τρέχουμε:

➢ cd hadoop-3.3.4



Στην παραπάνω εικόνα βλέπουμε την ρίζα (root) του καταλόγου αρχείων του συστήματος hadoop.

Οι φάκελοι που θα μας απασχολήσουν σε αυτόν τον οδηγό είναι οι τρεις φάκελοι: 1) **etc/hadoop** με τις βασικές ρυθμίσεις ,

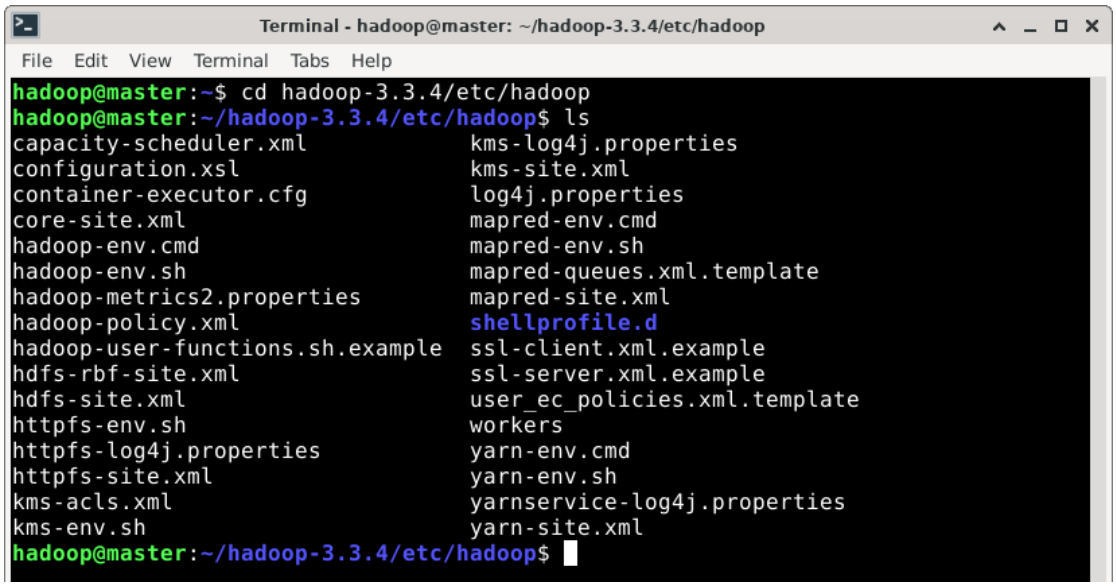
2) **bin** και

3) **sbin** με τα εκτελέσιμα binaries.

Σε αυτό το σημείο προτείνεται να πάρουμε λίγο χρόνο και να περιηγηθούμε στον κατάλογο αρχείων ούτως ώστε να αποκτήσουμε μια σχετική οικειότητα.

Από το root directory του Hadoop τρέχουμε:

➢ cd /etc/hadoop



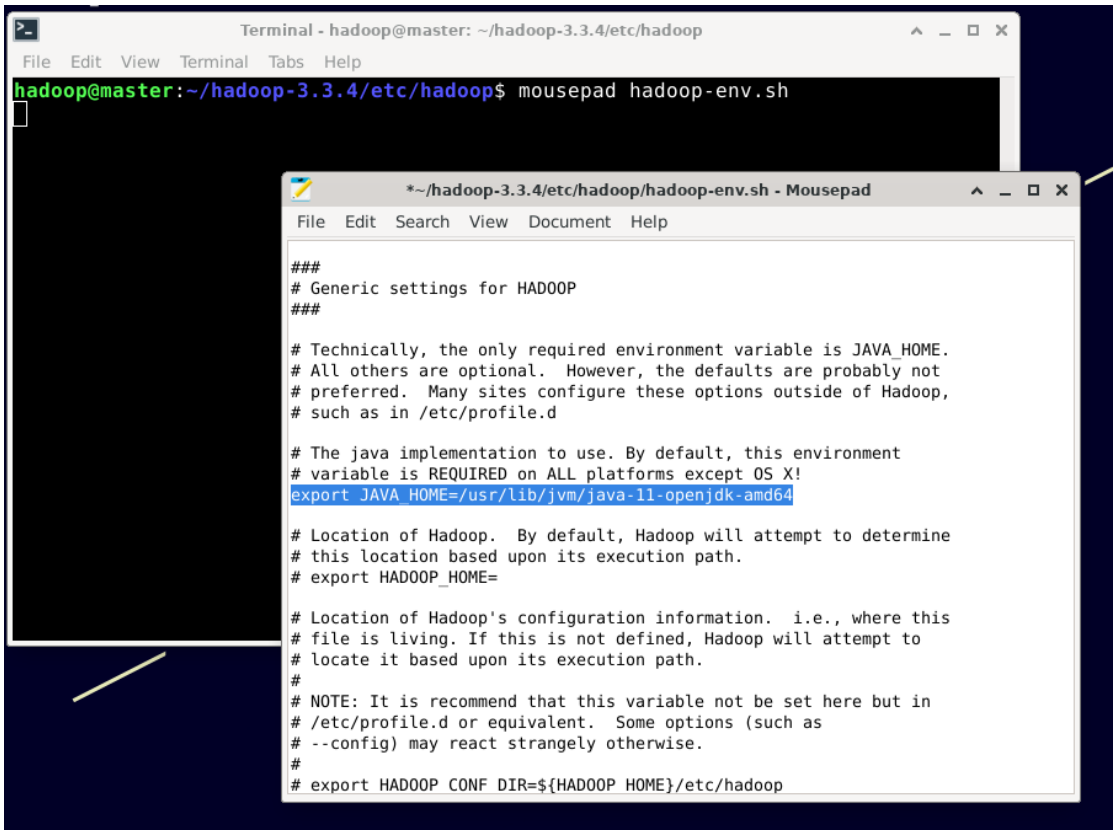
Από τον κατάλογο με τα αρχεία ρυθμίσεων της συστάδας hadoop θα ασχοληθούμε με τα αρχεία:

1) hadoop-env.sh — Οι περιβαλλοντικές ρυθμίσεις του Hadoop.
2) core-site.xml — Οι βασικές ρυθμίσεις του HDFS.
3) hdfs-site.xml — Οι ρυθμίσεις των κόμβων της συστάδας Hadoop.
4) yarn-site.xml — Οι ρυθμίσεις του YARN
5) workers — Καθορίζει τους κόμβους εργάτες.

Παρακάτω παρατίθενται οι ρυθμίσεις που χρησιμοποιήθηκαν για τον παρών οδηγό μαζί με μια σύντομη επεξήγηση για την καθεμία από αυτές.

ΣΗΜΑΝΤΙΚΟ: Οι ρυθμίσεις αυτές αποφασίστηκαν κατόπιν έρευνας και πειραματισμού για μια συστάδα εικονικών μηχανών που διαθέτουν συγκεκριμένα χαρακτηριστικά τα οποία αναφέρονται στην αρχή του κεφαλαίου αυτού. Για μηχανήματα με διαφορετικά χαρακτηριστικά θα πρέπει να εξεταστούν μια προς μια οι ρυθμίσεις και να αναθεωρηθούν κάποιες από τις τιμές που χρησιμοποιήθηκαν.

hadoop-env.sh



Στο αρχείο hadoop-env.sh όπως βλέπουμε στην εικόνα θα πρέπει να διαγραφεί το σύμβολο “#” και να προστεθεί το μονοπάτι της έκδοσης του πακέτου Java που εγκαταστήσαμε μετά την εντολή export.

Περισσότερες λεπτομέρειες για τις περιβαλλοντικές μεταβλητές στα συστήματα Linux θα αναφέρουμε παρακάτω. Το σύμβολο “#” ενημερώνει το σύστημα πως

επρόκειτο να ακολουθήσει κάποιο σχόλιο και όχι εντολή. Έτσι με την απομάκρυνση του ο υπολογιστής διαβάζει και τρέχει την εντολή export.

core-site.xml

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000</value>
    <description>
    Καθορίζει το προεπιλεγμένο URI του αρχείου συστήματος για το κατανεμημένο σύστημα αρχείων (HDFS) του Hadoop.
    Όταν ξεκινά μια συστάδα Hadoop, χρειάζεται να γνωρίζει την τοποθεσία του HDFS NameNode, ο οποίος είναι ο κύριος κόμβος
    που διαχειρίζεται τα μεταδεδομένα του συστήματος αρχείων και συντονίζει την πρόσβαση στα δεδομένα που αποθηκεύονται στο HDFS.
    </description>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/hadoop/hdfs/temp</value>
    <description>
```

Καθορίζει τον βασικό κατάλογο για τα προσωρινά αρχεία του Hadoop.

Αυτή η ιδιότητα χρησιμοποιείται για να καθορίσει τον κατάλογο όπου το Hadoop αποθηκεύει τα προσωρινά δεδομένα,

όπως τα ενδιάμεσα αποτελέσματα του MapReduce, τις τοπικές αντιγραφές των block δεδομένων και άλλα προσωρινά αρχεία.

</description>

</property>

</configuration>

hdfs-site.xml

<configuration>

<property>

<name>dfs.replication</name>

<value>2</value>

<description>

Αυτή η ιδιότητα καθορίζει τον παράγοντα αναπαραγωγής (replication factor) για τα αρχεία που αποθηκεύονται στο HDFS.

Στο HDFS, τα αρχεία χωρίζονται σε μπλοκ, τα οποία αναπαράγονται σε πολλούς κόμβους της συστάδας

για να εξασφαλιστεί η ανθεκτικότητα σε αποτυχίες και η διαθεσιμότητα των δεδομένων.

Η ιδιότητα dfs.replication καθορίζει τον αριθμό των αντιγράφων που δημιουργούνται για κάθε μπλοκ ενός αρχείου.

Αυτό διασφαλίζει ότι υπάρχουν πάντα τουλάχιστον τρία αντίγραφα κάθε μπλοκ διαθέσιμα στο σύστημα, ακόμη κι αν αποτύχουν ένας ή δύο κόμβοι.

</description>

</property>

<property>

<name>dfs.namenode.name.dir</name>

<value>/home/hadoop/hdfs/namenode/</value>

<description>

Το σημείο εγκατάστασης του NameNode.

</description>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>/home/hadoop/hdfs/datanode/</value>

<description>

Το σημείο εγκατάστασης του DataNode.

</description>

</property>

<property>

<name>dfs.permissions.enabled</name>

<value>false</value>

<description>

Αυτή η ιδιότητα καθορίζει εάν το HDFS θα επιβάλλει δικαιώματα ελέγχου πρόσβασης σε αρχεία και καταλόγους, το οποίο σημαίνει

ότι οι χρήστες και οι ομάδες έχουν δικαιώματα πρόσβασης ή απόρριψης πρόσβασης σε αρχεία βάσει των δικαιωμάτων τους.

Το HDFS υποστηρίζει τρεις τύπους δικαιωμάτων πρόσβασης: δικαιώματα ανάγνωσης / εγγραφής / εκτέλεσης για τον κάτοχο του αρχείου, την ομάδα και τους άλλους.

</description>

</property>

<property>

<name>dfs.permissions.superusergroup</name>

<value>hadoop</value>

<description>

Αυτή η ιδιότητα καθορίζει το όνομα της ομάδας του superuser, η οποία είναι μια ομάδα

που έχει απεριόριστη πρόσβαση σε όλα τα αρχεία και φακέλους στο HDFS,

ανεξάρτητα από τα δικαιώματα πρόσβασης τους.

</description>

</property>

</configuration>

yarn-site.xml

<configuration>

<property>

<name>yarn.resourcemanager.scheduler.class</name>

<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler</value>

<description>

Καθορίζει τον προγραμματιστή (scheduler) που χρησιμοποιείται από τον ResourceManager.

Σε αυτήν την περίπτωση η ιδιότητα έχει οριστεί σε CapacityScheduler,

που είναι ένας επεκτάσιμος προγραμματιστής με υποστήριξη πολλαπλών ουρών.

</description>

</property>

<property>

```
    <name>yarn.scheduler.capacity.root.queues</name>
    <value>prod,dev</value>
    <description>
    Καθορίζει τα ονόματα των ουρών που θα χρησιμοποιήσει ο CapacityScheduler.
    Σε αυτήν την περίπτωση, υπάρχουν δύο ουρές, η "prod" και η "dev".
    </description>
</property>
<property>
    <name>yarn.scheduler.capacity.prod.capacity</name>
    <value>0.5</value>
    <description>
    Καθορίζει το ποσοστό των πόρων της συστάδας που θα πρέπει να διατεθούν
    στην ουρά "prod". Σε αυτήν την περίπτωση έχει οριστεί σε 50%.
    </description>
</property>
<property>
    <name>yarn.scheduler.capacity.dev.capacity</name>
    <value>0.5</value>
    <description>
    Καθορίζει το ποσοστό των πόρων της συστάδας που θα πρέπει να διατεθούν
    στην ουρά "dev". Σε αυτήν την περίπτωση έχει οριστεί σε 50%.
    </description>
</property>
<property>
```

<name>yarn.scheduler.capacity.dev.maximum-capacity</name>

<value>0.5</value>

<description>

Καθορίζει το μέγιστο ποσοστό των πόρων συστάδας που θα μπορεί να χρησιμοποιήσει η ουρά "dev".

Σε αυτήν την περίπτωση, έχει οριστεί σε 50%.

</description>

</property>

<property>

<name>yarn.scheduler.capacity.prod.maximum-capacity</name>

<value>0.7</value>

<description>

Καθορίζει το μέγιστο ποσοστό των πόρων συστάδας που θα μπορεί να χρησιμοποιήσει η ουρά "prod".

Σε αυτήν την περίπτωση, έχει οριστεί σε 70%.

</description>

</property>

<property>

<name>yarn.acl.enable</name>

<value>0</value>

<description>

Ενεργοποιεί ή απενεργοποιεί τις λίστες ελέγχου πρόσβασης (access control lists - ACLs) του YARN.

</description>

</property>

<property>

<name>yarn.resourcemanager.hostname</name>

<value>master</value>

<description>

Καθορίζει το όνομα του κόμβου που θα φιλοξενεί τον ResourceManager.

</description>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-mb</name>

<value>512</value>

<description>

Καθορίζει την ελάχιστη ποσότητα μνήμης (σε megabytes) που μπορεί ο κάθε ένας container να αιτηθεί.

</description>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-mb</name>

<value>3072</value>

<description>

Καθορίζει την μέγιστη ποσότητα μνήμης (σε megabytes) που μπορεί ο κάθε ένας container να αιτηθεί.

</description>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

<description>

Καθορίζει τις υπηρεσίες βοηθητικών λειτουργιών που πρέπει να ξεκινήσουν στον NodeManager.

</description>

</property>

<property>

<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

<description>

Καθορίζει την κλάση που υλοποιεί την υπηρεσία MapReduce ShuffleHandler.

</description>

</property>

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>3072</value>

<description>

Καθορίζει την συνολική ποσότητα μνήμης (σε megabytes) που είναι διαθέσιμη σε κάθε NodeManager.

</description>

</property>

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

<description>

Ενεργοποιεί ή απενεργοποιεί την εικονική μνήμη (Virtual Memory - VM) που επιβλέπει για containers.

</description>

</property>

<property>

<name>yarn.nodemanager.vmem-pmem-ratio</name>

<value>2.1</value>

<description>

Καθορίζει την αναλογία της εικονικής μνήμης ως προς τη φυσική μνήμη που μπορεί να χρησιμοποιηθεί από έναν container.

</description>

</property>

<property>

<name>yarn.nodemanager.resource.cpu-vcores</name>

<value>2</value>

<description>

Καθορίζει τον αριθμό των πυρήνων του επεξεργαστή που θα είναι διαθέσιμοι σε κάθε NodeManager.

</description>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-vcores</name>

<value>1</value>

<description>

Καθορίζει τον ελάχιστο αριθμό των πυρήνων του επεξεργαστή που μπορεί να αιτηθεί το κάθε container.

</description>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-vcores</name>

<value>2</value>

<description>

Καθορίζει τον μέγιστο αριθμό των πυρήνων του επεξεργαστή που μπορεί να αιτηθεί το κάθε container.

</description>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>master:8088</value>

<description>

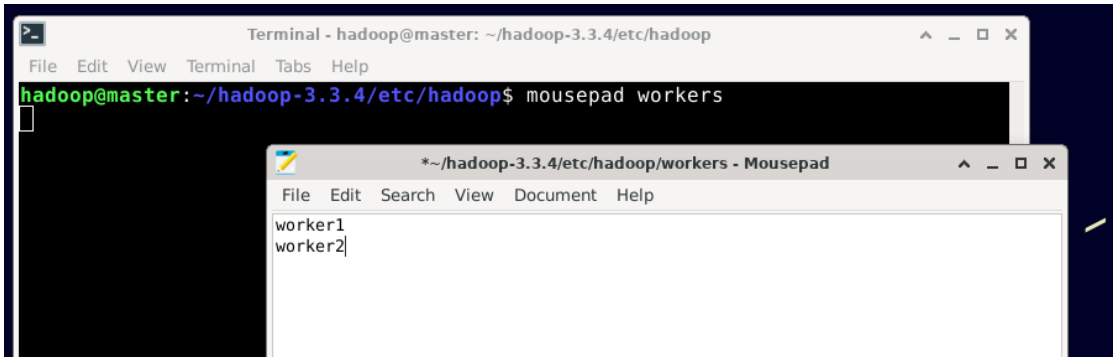
Καθορίζει την διεύθυνση (hostname και port) όπου δύναται η πρόσβαση στην διαδικτυακή διεπαφή του ResourceManager.

</description>

</property>

</configuration>

workers



Αφού έχουμε τελειώσει με τις ρυθμίσεις των κόμβων επιστρέφουμε στο home directory.

Τώρα θα δημιουργήσουμε χειροκίνητα τους φακέλους όπου θα αποθηκεύονται τα αρχεία των NameNode και DataNodes, καθώς και τα προσωρινά αρχεία όπως θέσαμε στις παραπάνω ρυθμίσεις.

Για να το κάνουμε αυτό τρέχουμε τις εξής εντολές:

➢ mkdir - p hdfs/namenode

Με την εντολή αυτήν θα δημιουργηθεί ένας φάκελος που θα ονομάζεται hdfs και μέσα σε αυτόν ένας άλλος, ο namenode.

➢ mkdir - p hdfs/datanode

Μέσα στον φάκελο hdfs θα δημιουργηθεί ένας νέος, ο datanode.

➢ mkdir - p hdfs/temp

Μέσα στον φάκελο hdfs θα δημιουργηθεί ένας νέος, ο temp.

Πατώντας cd hdfs και ls μπορούμε να ελέγξουμε αν έχουμε και τους τρεις φακέλους.

Το επόμενο βήμα είναι να δημιουργήσουμε και να αρχικοποιήσουμε τον NameNode.

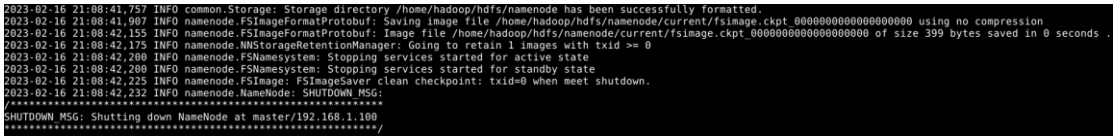
Από το root directory του hadoop τρέχουμε:

➢ cd bin

➢ ./hdfs namenode -format

και περιμένουμε να ολοκληρωθεί η εκτέλεση. Εάν στο τέλος δεν προκύψει κάποιο σφάλμα

θα έχουμε κάτι τέτοιο στο τερματικό:



Η εγκατάσταση του Apache Hadoop ολοκληρώθηκε.

Ανοίγουμε το αρχείο bashrc προς επεξεργασία και προσθέτουμε τις περιβαλλοντικές μεταβλητές του Hadoop (HDFS + YARN) ούτως ώστε να είναι ενήμερο το λειτουργικό σύστημα και οι διάφορες εφαρμογές.

➢ mousepad ~/.bashrc

Στο τέλος του bashrc προσθέτουμε τις γραμμές:

#####HADOOP CONFIGURATION#####

export HADOOP_HOME=/home/hadoop/hadoop-3.3.4

export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

export HADOOP_CONF_DIR="/home/hadoop/hadoop-3.3.4/etc/hadoop"

export HADOOP_LOG_DIR=$HADOOP_HOME/logs

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

Στο συγκεκριμένο σημείο προτείνεται να γίνει επανεκκίνηση της εικονικής μηχανής από το μενού του λειτουργικού συστήματος ή με την εντολή sudo reboot .

2.7 ΒΗΜΑ 5 - ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΡΥΘΜΙΣΗ ΤΟΥ ΛΟΓΙΣΜΙΚΟΥ APACHE SPARK

Το Hadoop αποτελεί την βάση πάνω από την οποία θα τρέξει το Spark, παρέχοντας ένα κατανεμημένο σύστημα αρχείων και έναν Resource Manager για την διαχείριση των πόρων της συστάδας.

Προχωράμε στην εγκατάσταση του Spark.

Για να κατεβάσουμε το Apache Spark θα πρέπει να επισκεφθούμε τον παρακάτω σύνδεσμο:

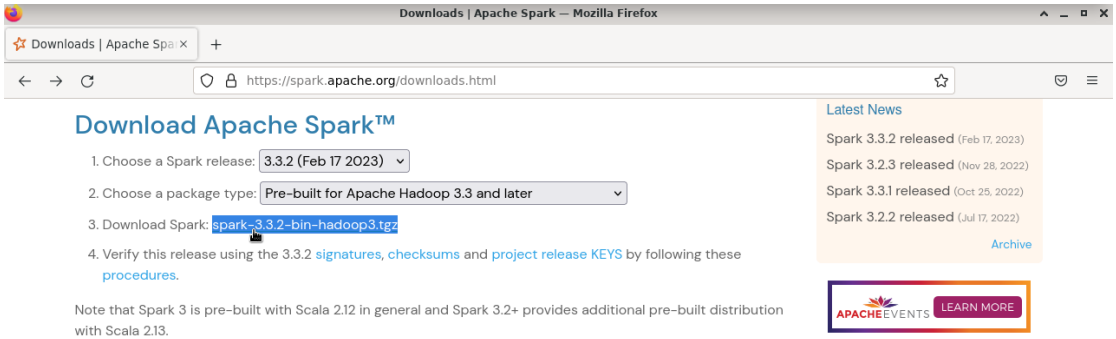
https://spark.apache.org/downloads.html

ΣΗΜΕΙΩΣΗ: Εάν ο αναγνώστης επιθυμεί να κατεβάσει την έκδοση που χρησιμοποιήθηκε στον παρόντα οδηγό θα την βρει εδώ:

https://archive.apache.org/dist/spark/spark-3.3.2/

Και θα πρέπει να προχωρήσει εδώ.

Στην ενότητα “Downloads” θα βρούμε μια σελίδα με τις τελευταίες εκδόσεις σε διαφορετικές μορφές αρχείων που μπορούμε να κατεβάσουμε.



Επιλέγουμε την πιο πρόσφατη έκδοση και τύπο πακέτου "Pre-built for Apache Hadoop 3.3 and later" εφόσον έχουμε χρησιμοποιήσει αντίστοιχη έκδοση για το Hadoop. Τέλος, κάνουμε κλικ στον σύνδεσμο "Download Spark".

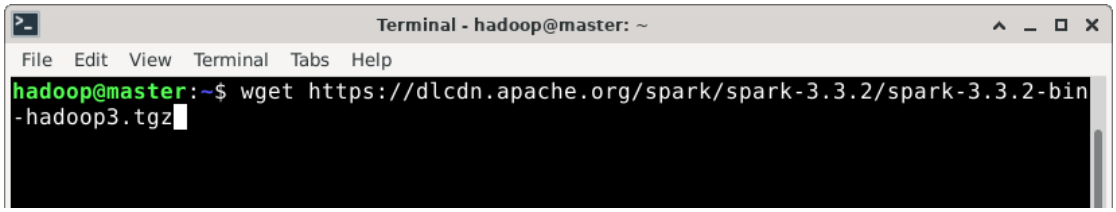
Στον επόμενη σελίδα κάνουμε δεξί κλικ στον πρώτο σύνδεσμο ή στον σύνδεσμο κάτω από την ένδειξη "**HTTP**" και επιλέγουμε αντιγραφή συνδέσμου (Copy link address).

Για όσους επέλεξαν την έκδοση Spark του οδηγού, τότε στην σελίδα που οδηγεί ο σύνδεσμος κάνουμε δεξί κλικ στο κουμπί με την ένδειξη "**spark-3.3.2-bin-hadoop3.tgz**" και επιλέγουμε αντιγραφή συνδέσμου (Copy link address).

Index of /dist/spark/spark-3.3.2

Name	Last modified	Size	Description
Parent Directory		-	
SparkR_3.3.2.tar.gz	2023-02-10 21:28	344K	
SparkR_3.3.2.tar.gz.asc	2023-02-10 21:28	687	
SparkR_3.3.2.tar.gz.sha512	2023-02-10 21:28	150	
pyspark-3.3.2.tar.gz	2023-02-10 21:28	268M	
pyspark-3.3.2.tar.gz.asc	2023-02-10 21:28	687	
pyspark-3.3.2.tar.gz.sha512	2023-02-10 21:28	151	
spark-3.3.2-bin-hadoop2.tgz	2023-02-10 21:28	261M	
spark-3.3.2-bin-hadoop2.tgz.asc	2023-02-10 21:28	687	
spark-3.3.2-bin-hadoop2.tgz.sha512	2023-02-10 21:28	158	
spark-3.3.2-bin-hadoop3-scala2.13.tgz	2023-02-10 21:28	292M	
spark-3.3.2-bin-hadoop3-scala2.13.tgz.asc	2023-02-10 21:28	687	
spark-3.3.2-bin-hadoop3-scala2.13.tgz.sha512	2023-02-10 21:28	168	
spark-3.3.2-bin-hadoop3.tgz	2023-02-10 21:28	285M	
spark-3.3.2-bin-hadoop3.tgz.asc	2023-02-10 21:28	687	

➢ wget [Σύνδεσμος για το Spark που αντιγράψαμε πριν]

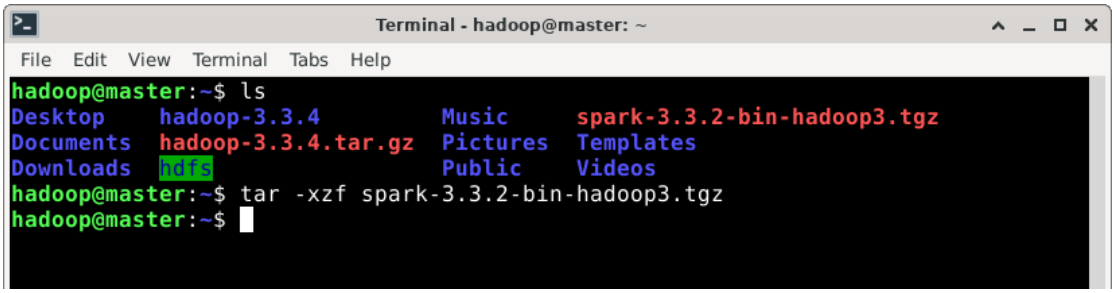


Περιμένουμε έως ότου κατέβει το συμπιεσμένο αρχείο που θα χρησιμοποιήσουμε για την εγκατάσταση. Για να βεβαιωθούμε ότι κατέβηκε επιτυχώς μπορούμε να χρησιμοποιήσουμε την εντολή:

➢ ls

Εάν δούμε ένα αρχείο με την εξής μορφή **spark-x.x.x-bin-hadoopx.tgz** σημαίνει πως όλα πήγαν καλά και μπορούμε να προχωρήσουμε στην εξαγωγή του φακέλου με τα αρχεία spark από το συμπιεσμένο αρχείο που κατεβάσαμε.

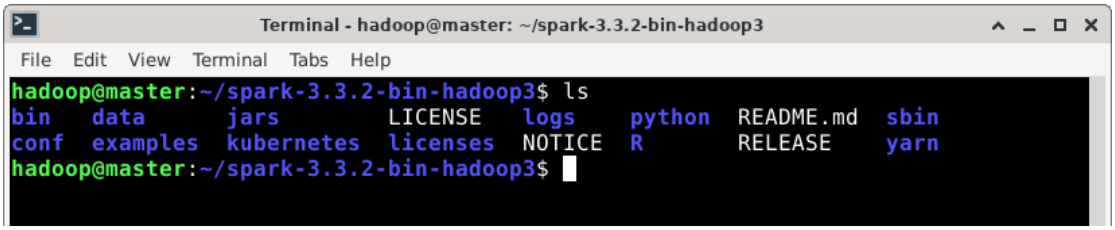
Στην παρακάτω εικόνα βλέπουμε τις εντολές που πρέπει να χρησιμοποιήσουμε:



Οι ρυθμίσεις του Spark είναι πολύ πιο σύντομη διαδικασία σε σχέση με το Hadoop. Αυτό συμβαίνει γιατί την διαχείριση των πόρων την αναλαμβάνει το YARN το οποίο στηρίζεται στις ρυθμίσεις του κόμβου που βρίσκονται στο περιβάλλον Hadoop.

Από το home directory τρέχουμε:

➢ cd spark-3.3.2-bin-hadoop3



Στην παραπάνω εικόνα βλέπουμε την ρίζα (root) του καταλόγου αρχείων του συστήματος spark..

Οι φάκελοι που θα μας απασχολήσουν σε αυτόν τον οδηγό είναι οι δύο φάκελοι:
1) **conf** με τις βασικές ρυθμίσεις και

2) **bin** με τα εκτελέσιμα binaries.

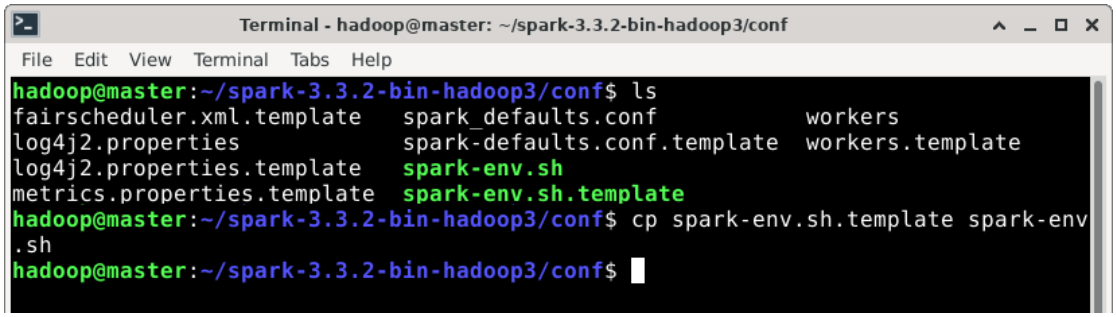
Σε αυτό το σημείο προτείνεται να πάρουμε λίγο χρόνο και να περιηγηθούμε στον κατάλογο αρχείων ούτως ώστε να αποκτήσουμε μια σχετική οικειότητα.

Από το root directory του Spark τρέχουμε:

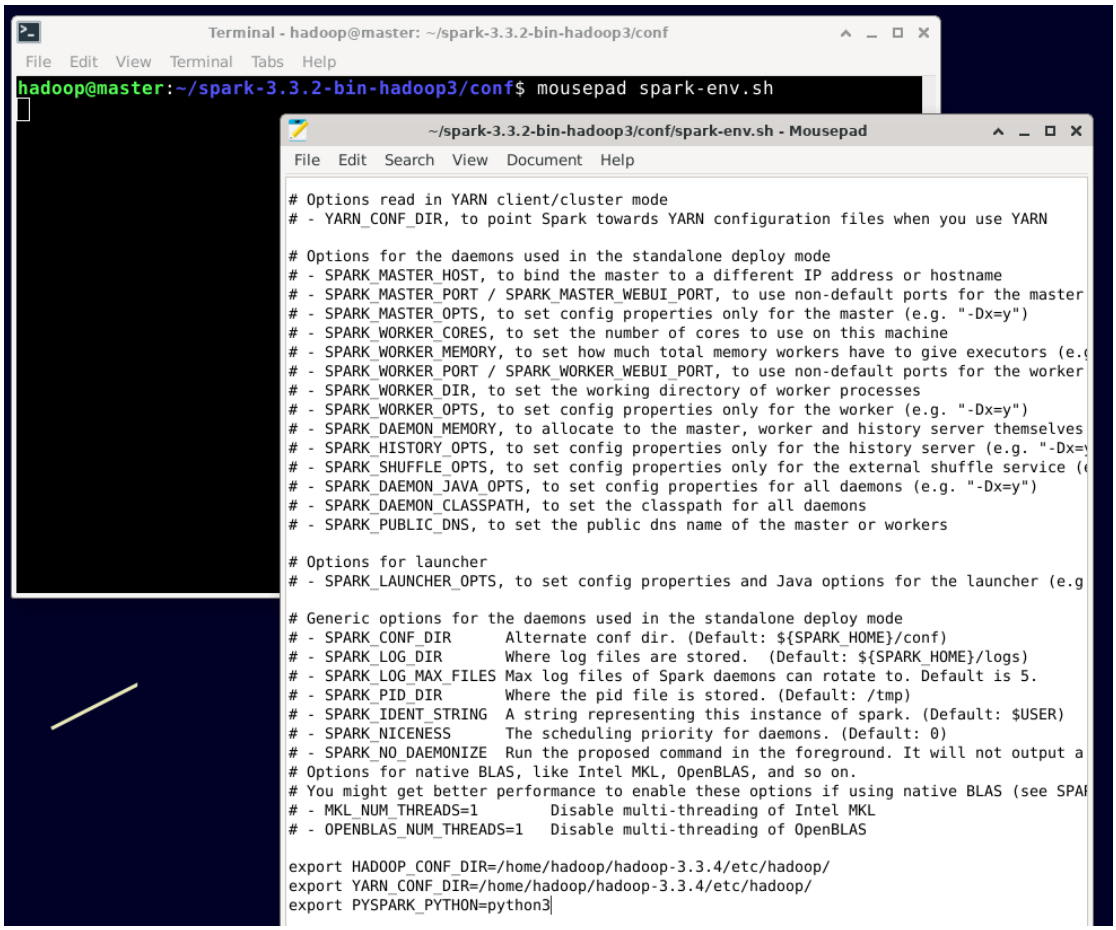
➢ cd conf

και δημιουργούμε ένα αντίγραφο του αρχείου "**spark-env.sh.template**" μετονομάζοντάς το σε "**spark-env.sh**" όπως βλέπουμε στην εικόνα με την εντολή

➢ cp spark-env.sh.template spark-env.sh



Έπειτα ανοίγουμε το αρχείο αυτό για επεξεργασία και προσθέτουμε την περιβαλλοντικές μεταβλητές του HDFS και YARN που οφείλει να γνωρίζει το Spark μαζί με την έκδοση Python που θα χρησιμοποιηθεί (εάν ο αναγνώστης επιλέξει να χρησιμοποιήσει το Spark με Python).



Για την επόμενη ρύθμιση, τρέχουμε την εντολή:

```
➢ cp spark-defaults.conf.template spark-defaults.conf
```

Έπειτα ανοίγουμε το αρχείο "**spark-defaults.conf"** για επεξεργασία και προσθέτουμε της εξής γραμμές:

```
# spark-submit defaults
spark.master yarn
spark.driver.memory 512m
spark.yarn.am.memory 512m
spark.executor.memory 512m

# spark history settings
spark.eventLog.enabled true
spark.eventLog.dir hdfs://namenode:9000/spark-logs
spark.history.provider org.apache.spark.deploy.history.FsHistoryProvider
spark.history.fs.logDirectory hdfs://namenode:9000/spark-logs
spark.history.fs.update.interval 10s
spark.history.ui.port 18080
```

Το πρώτο τμήμα έχει σχέση με τις παραμέτρους του εργαλείου γραμμής εντολών spark-submit τις οποίες μπορούμε να ορίσουμε by default από εδώ εάν θέλουμε χωρίς να τις προσθέτουμε κάθε φορά με την εκτέλεση της εντολής.

Το δεύτερο τμήμα ενεργοποιεί τον log server για το Spark. Όταν ο log server ενεργοποιηθεί, το Spark καταγράφει τα συμβάντα στον προκαθορισμένο κατάλογο για περαιτέρω ανάλυση.

Για να τρέξει ο log server ή history server, από το root directory του Spark τρέχουμε:

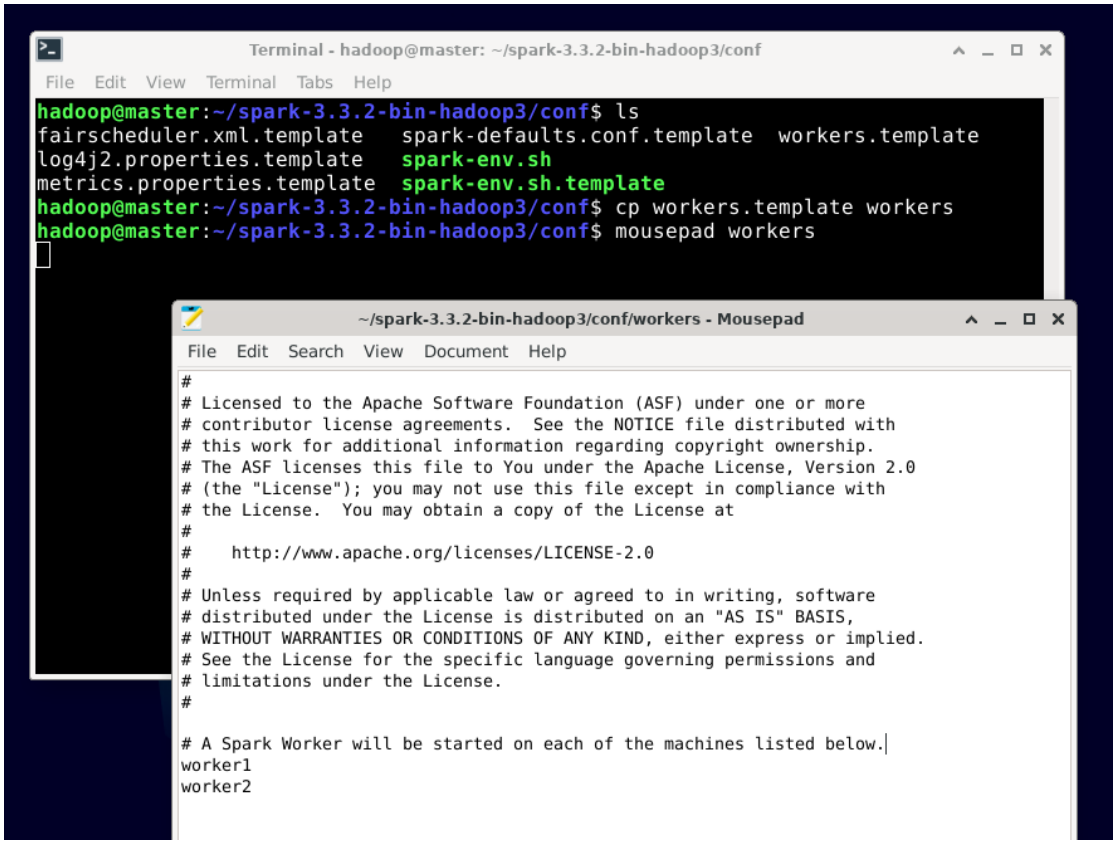
➢ sbin/start-history-server.sh

και για να σταματήσουμε την διεργασία:

➢ sbin/stop-history-server.sh

Όλες οι ρυθμίσεις στο αρχείο "**spark-defaults.conf**" είναι προαιρετικές.

Η επόμενη ρύθμιση έγκειται στο αρχείο "**workers**". Ανοίγουμε το αρχείο προς επεξεργασία και προσθέτουμε τους δύο κόμβους εργατών που θα δημιουργήσουμε:



Ανοίγουμε το αρχείο bashrc προς επεξεργασία και προσθέτουμε τις περιβαλλοντικές μεταβλητές του Hadoop (HDFS + YARN) ούτως ώστε να είναι ενήμερο το λειτουργικό σύστημα και οι διάφορες εφαρμογές.

➤ mousepad ~/.bashrc

Στο τέλος του bashrc προσθέτουμε τις γραμμές:

#####SPARK CONFIGURATION#####

export SPARK_HOME=/home/hadoop/spark-3.3.2-bin-hadoop3

export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin

Στο συγκεκριμένο σημείο προτείνεται να γίνει επανεκκίνηση της εικονικής μηχανής από το μενού του λειτουργικού συστήματος ή με την εντολή sudo reboot .

Η εγκατάσταση και ρύθμιση του Apache Spark έχει ολοκληρωθεί, όπως επίσης και του Hadoop.

Στο επόμενο βήμα θα δημιουργήσουμε μια συστάδα με τρεις εικονικές μηχανές.

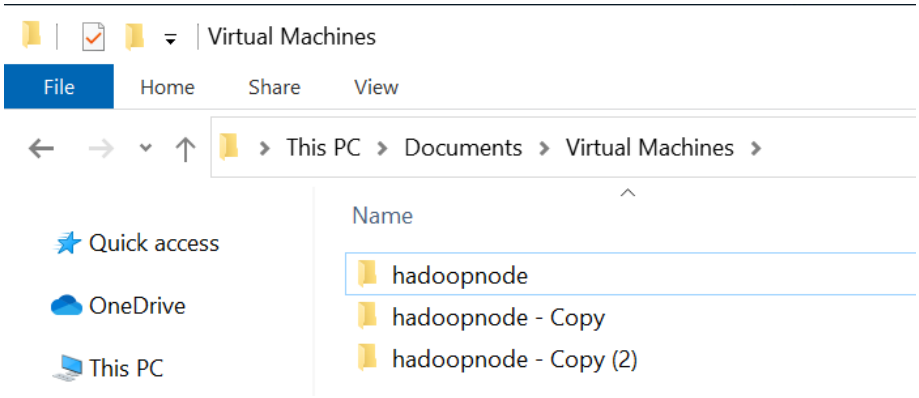
2.8 ΒΗΜΑ 6 - ΔΗΜΙΟΥΡΓΙΑ ΤΗΣ ΣΥΣΤΑΔΑΣ SPARK

Αυτό που έχουμε μέχρι στιγμής είναι ένας κόμβος ο οποίος περιέχει όλες τις ρυθμίσεις για το Spark, το YARN και το HDFS. Οι ρυθμίσεις όμως αυτές έγιναν με το σκεπτικό μιας συστάδας εικονικών υπολογιστών, οι οποίοι συνδέονται στο ίδιο δίκτυο και μπορούν να επικοινωνήσουν ούτως ώστε να γίνεται η απαραίτητη μεταφορά των δεδομένων από τον ένα κόμβο στον άλλο. Ο κόμβος Αρχηγός μέσω του YARN θα είναι εκείνος ο οποίος θα δίνει τις εντολές στους κόμβους εργατών για να δεσμεύσουν τους απαραίτητους πόρους έτσι ώστε να τρέξουν οι διάφορες εφαρμογές που απαιτούν κατανεμημένη εκτέλεση. Το πλάνο λοιπόν είναι να δημιουργήσουμε την συστάδα και β) να επιτρέψουμε στον κόμβο αρχηγό (master) όπου θα τρέχουμε τις εφαρμογές να επικοινωνεί με τους άλλους δύο κόμβους.

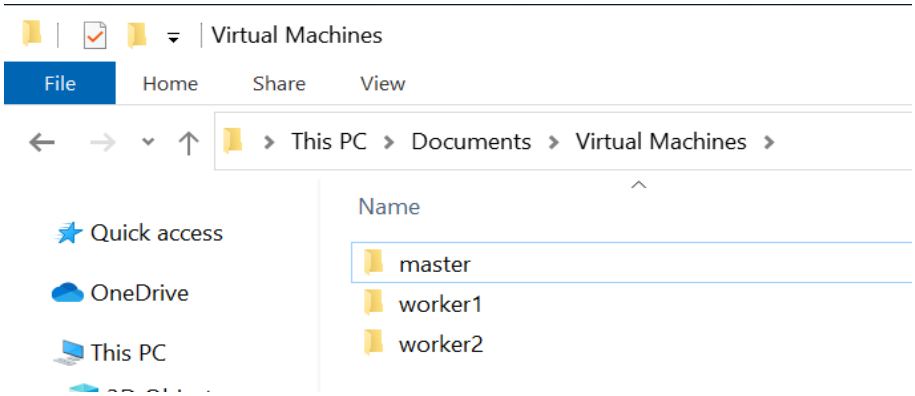
Στην φυσική μηχανή που χρησιμοποιούμε ως host για τις εικονικές μηχανές, βρίσκουμε τον φάκελο με τα αρχεία του κόμβου που δημιουργήσαμε στο πρώτο βήμα του οδηγού.

Στην περίπτωση του οδηγού το λειτουργικό σύστημα του υπολογιστή host είναι το Windows 10. Η ίδια λογική όμως εφαρμόζεται και για τα υπόλοιπα συστήματα.

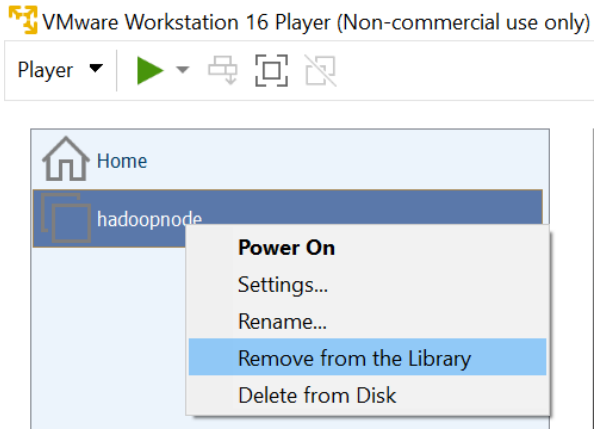
Αφού εντοπίσουμε τον φάκελο δημιουργούμε δύο αντίγραφα.



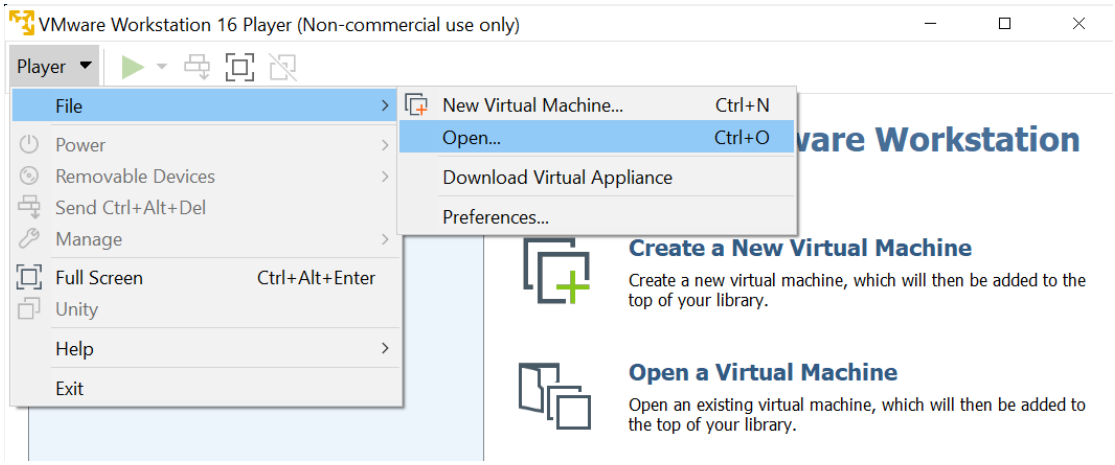
Μετονομάζουμε τους φακέλους σε "**master**", "**worker1**" και "**worker2**".

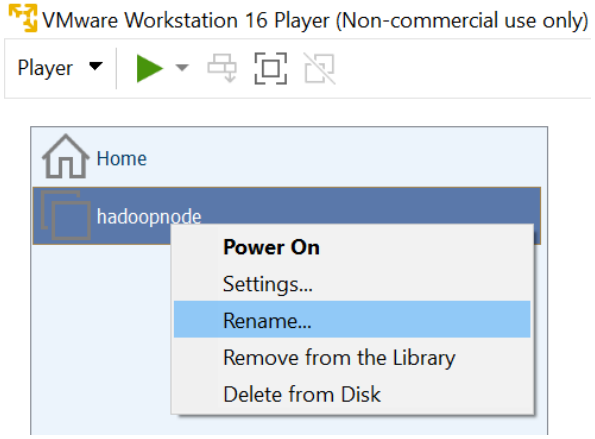


Έπειτα ανοίγουμε το πρόγραμμα **VMwareWorkstation Player** και διαγράφουμε την εικονική συσκευή από την βιβλιοθήκη του προγράμματος και ΟΧΙ από τον δίσκο.

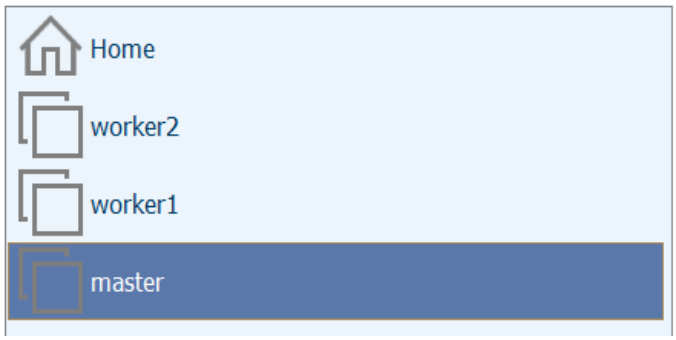


Το επόμενο βήμα είναι να εισαγάγουμε τις τρεις εικονικές μηχανές στην βιβλιοθήκη του VMWare Workstation Player μετονομάζοντας την κάθε μία:





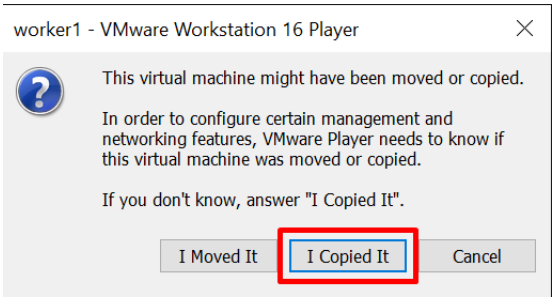
Μέχρι να έχουμε αυτόν τον κατάλογο:



Η συστάδα είναι έτοιμη. Προχωράμε στην συνδεσιμότητα.

Σε αυτό το σημείο για κάθε κόμβο εκτελούμε τα βήματα και τις εντολές με αστερίσκο [*] στο [Βήμα 3].

ΣΗΜΑΝΤΙΚΟ: Την πρώτη φορά που θα επιχειρήσουμε να εκκινήσουμε την κάθε εικονική μηχανή - κόμβο θα λάβουμε το παρακάτω μήνυμα. Και τις τρεις φορές θα επιλέξουμε "**I Copied It**"



Στους κόμβους **worker1** και **worker2** έχει απενεργοποιηθεί το παραθυρικό περιβάλλον για την εξοικονόμηση πόρων.

Χρησιμοποιήθηκαν οι εντολή:

➢ sudo systemctl set-default multi-user.target

και επανεκκίνηση του συστήματος.

Για την επαναφορά του παραθυρικού περιβάλλοντος χρησιμοποιείται η εντολή:

➢ sudo systemctl set-default graphical.target

και επανεκκίνηση του συστήματος.

Εάν η εικονική μηχανή δεν έχει εγκατεστημένο το πρωτόκολλο και την εφαρμογή open-ssh τρέχουμε την εντολή:

(Εάν η εγκατάσταση του οδηγού έχει ακολουθηθεί με ακρίβεια τότε ο αναγνώστης δεν χρειάζεται να εκτελέσει την επόμενη εντολή)

➢ sudo apt-get install openssh-server

Το πρωτόκολλο ssh λειτουργεί με κρυπτογραφημένα κλειδιά και επιτρέπει την πρόσβαση σε απομακρυσμένο υπολογιστή μέσα από το τερματικό (terminal).

Περισσότερες λεπτομέρειες για το πρωτόκολλο ssh υπάρχουν εδώ:

https://www.ssh.com/academy/ssh/openssh

Οι παρακάτω οδηγίες απευθύνονται ΜΟΝΟ για τον κόμβο **master**.

Αφού έχουμε εγκαταστήσει τον ssh server τρέχουμε την εντολή

➢ ssh-keygen -t rsa

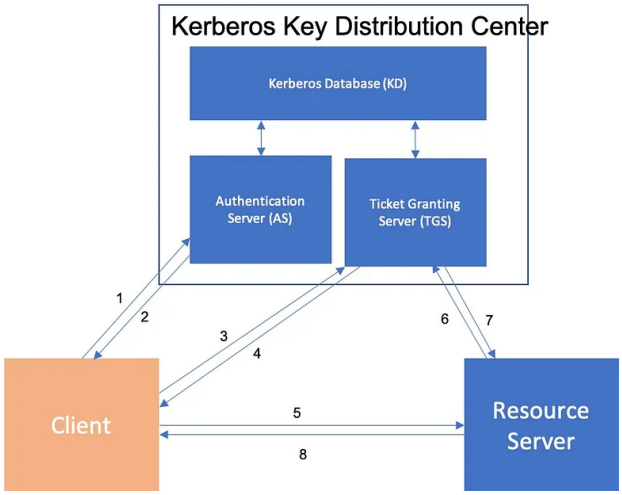
για να δημιουργήσουμε ένα ζεύγος κλειδιών RSA (Rivest-Shamir-Adleman) για χρήση με το SSH. Είναι σημαντικό να μην χρησιμοποιηθεί κωδικός κατά την δημιουργία κλειδιών.

Αυτό διότι κάθε φορά που θα απαιτείται πρόσβαση σε οποιονδήποτε κόμβο, ο χρήστης θα πρέπει να πληκτρολογεί τον κωδικό χειροκίνητα.

Σε ένα περιβάλλον παραγωγής είναι σημαντικό να υπάρχει κρυπτογράφηση ανώτερου επιπέδου από SSH. Στην περίπτωση του Hadoop προτείνεται η κρυπτογράφηση "**Kerberos**".

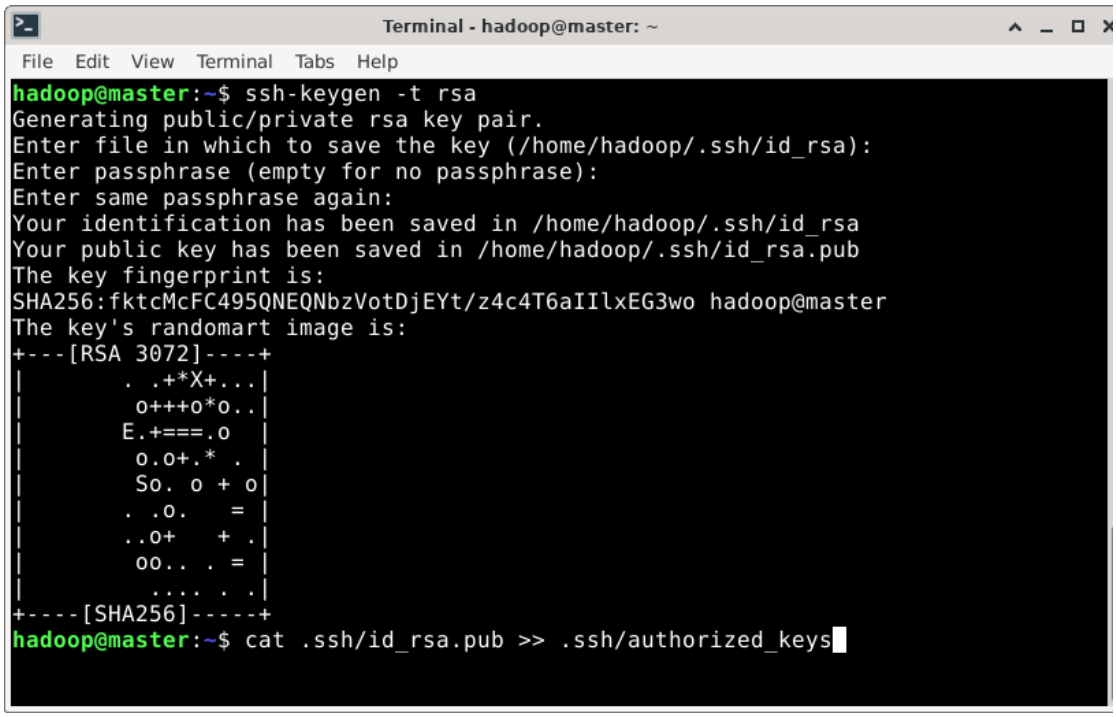
Password less communication using Kerberos

1. Client sends request enc(passwd(POST /jobs)) to AS
2. AS decrypts using the passwd from KD and sends back a Ticket Granting Ticket (TGT) which is encrypted using a shared key with TGS
3. Client sends the request along with TGT to TGS
4. TGS decrypts the TGT using the shared key with AS and sends back a token which is encrypted using another shared key between TGS and the Resource Server
5. Client then sends the request to Resource Server using along with the encrypted token
6. The Resource Server then decrypts the token using the shared key between Resource Server and TGT
7. TGT then send back a Valid response
8. Based on the outcome of step 7, the resource server sends back the resource response to the Client.

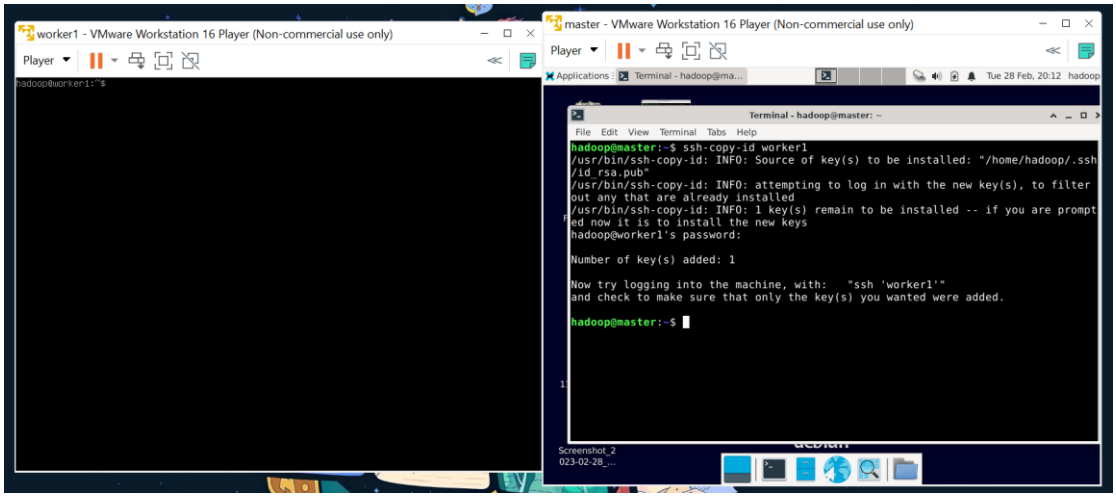


Θα συνεχίσουμε όμως με την χρήση του ssh.

Στην συνέχεια, όπως βλέπουμε και στην εικόνα αντιγράφουμε τα περιεχόμενα του αρχείου δημοσίων κλειδιών **id_rsa.pub** σε ένα νέο αρχείο με όνομα **authorized_keys**:

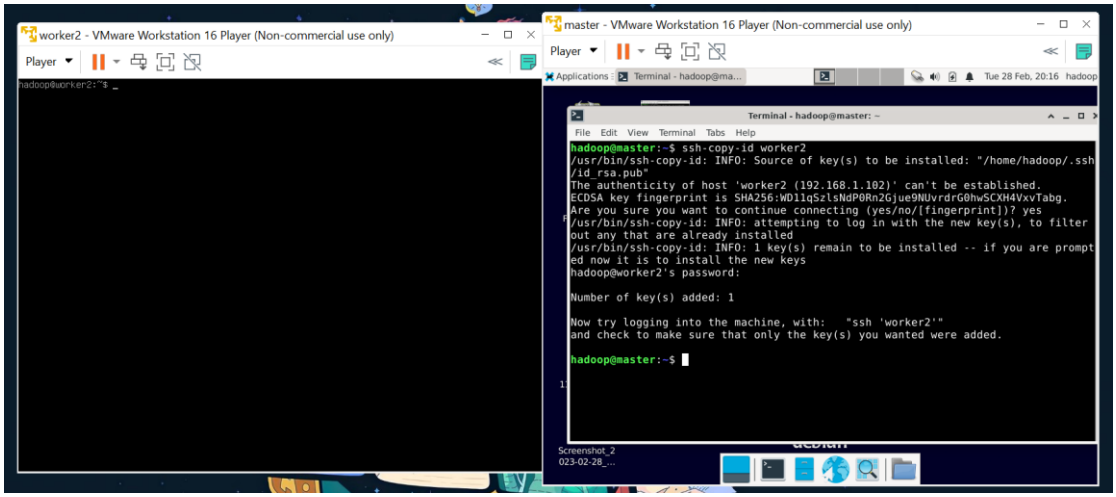


Μαζί με τον κόμβο **master** εκκινούμε τον κόμβο **worker1** και αντιγράφουμε το δημόσιο κλειδί RSA του **master** στον **worker1**.



Εάν λάβουμε μήνυμα για το αν θέλουμε να συνεχίσουμε με την σύνδεση επιλέγουμε yes.

Επαναλαμβάνουμε την διαδικασία για τον κόμβο worker2.



Στη συνέχεια δοκιμάζουμε να συνδεθούμε από τον **master** στους υπόλοιπους κόμβους με τις εντολές:

➢ ssh worker1

και

➢ exit

για να αποσυνδεθούμε.

➢ ssh worker2

και

➢ exit

για να αποσυνδεθούμε.

Η συστάδα Spark-YARN είναι έτοιμη.

Τα υπόλοιπα βήματα του οδηγού θα πραγματοποιηθούν μόνο στον master κόμβο.

2.9 ΒΗΜΑ 7 - ΔΟΚΙΜΗ ΤΟΥ APACHE HADOOP

Εφόσον όλοι οι κόμβοι βρίσκονται σε λειτουργία, από τον master ενεργοποιούμε το κατανεμημένο σύστημα αρχείων και το YARN.

Από το home directory τρέχουμε:

➢ cd hadoop-3.3.2/sbin

➢ ./start-dfs.sh

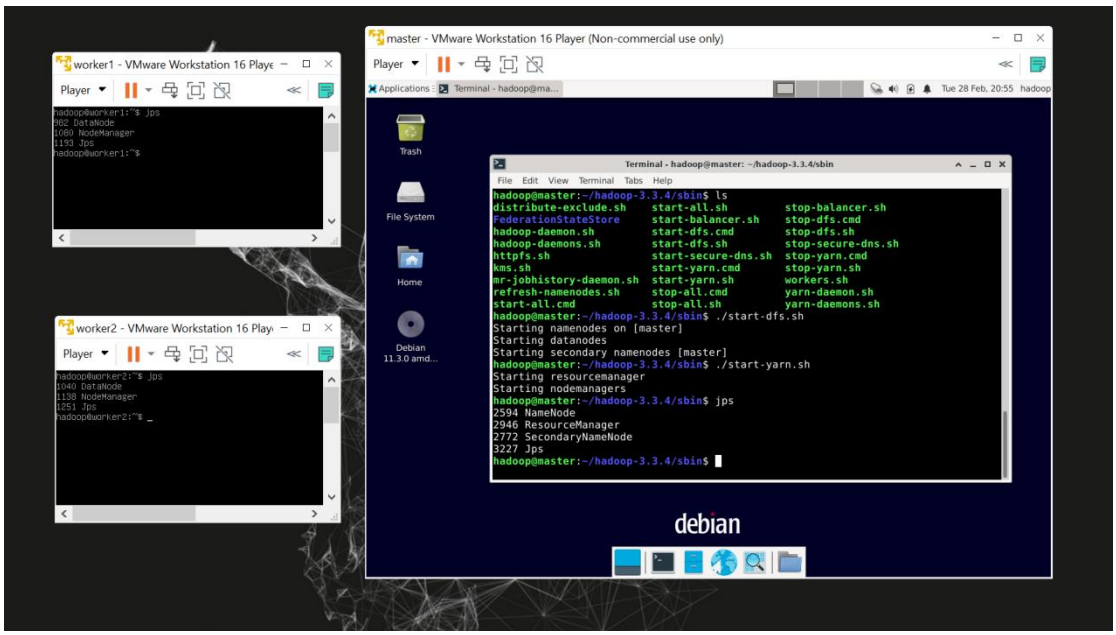
➢ ./start-yarn.sh

ΣΗΜΕΙΩΣΗ: Εάν έχουμε καταχωρήσει σωστά τις περιβαλλοντικές μεταβλητές Hadoop στο αρχείο bashrc τότε μπορούμε να τρέχουμε τις εντολές του Hadoop ανεξάρτητα από το directory που βρισκόμαστε και χωρίς να μπαίνουμε στον φάκελο sbin με την χρήση της εντολής cd .

Για να επιβεβαιώσουμε πως όλα λειτουργούν κανονικά ελέγχουμε ποιες εικονικές μηχανές Java τρέχουν στο σύστημα με την εντολή

➢ jps

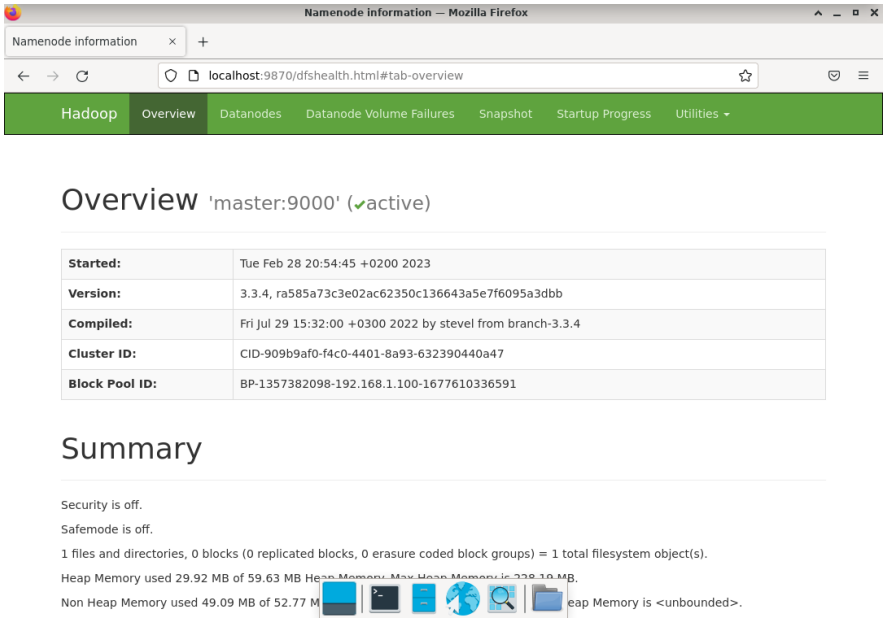
σε κάθε κόμβο. Η εικόνα που θα πρέπει να έχουμε θα μοιάζει με αυτήν:



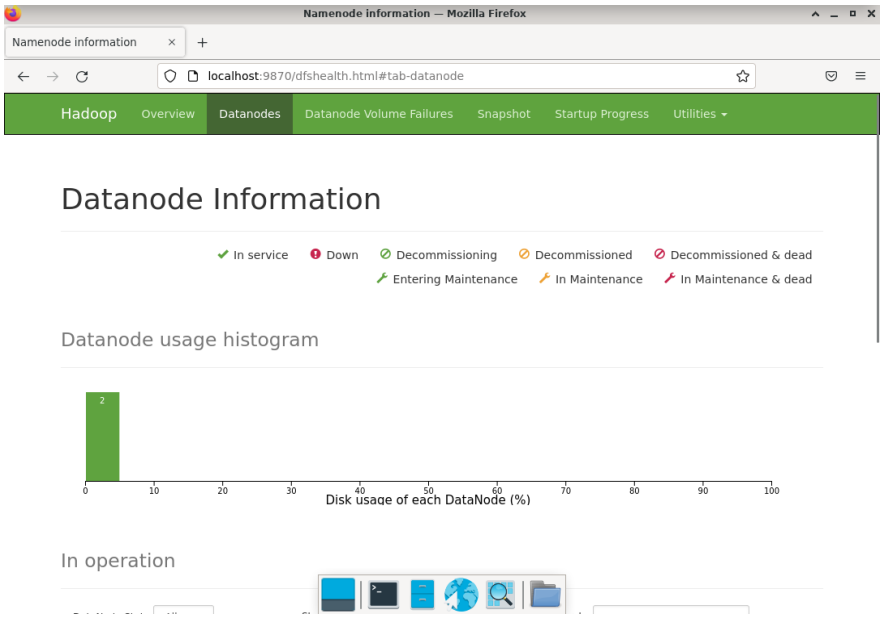
Για να έχουμε μια πιο ζωντανή εικόνα της συστάδας μπορούμε να ανοίξουμε το πρόγραμμα περιήγησης που χρησιμοποιούμε (π.χ. Mozilla Firefox) και να βάλουμε την διεύθυνση:

localhost:9870

Έτσι μπορούμε να έχουμε πρόσβαση στην διαδικτυακή διεπαφή του Hadoop.



Στην καρτέλα "**DataNodes**" βλέπουμε τους εν ενεργεία κόμβους.

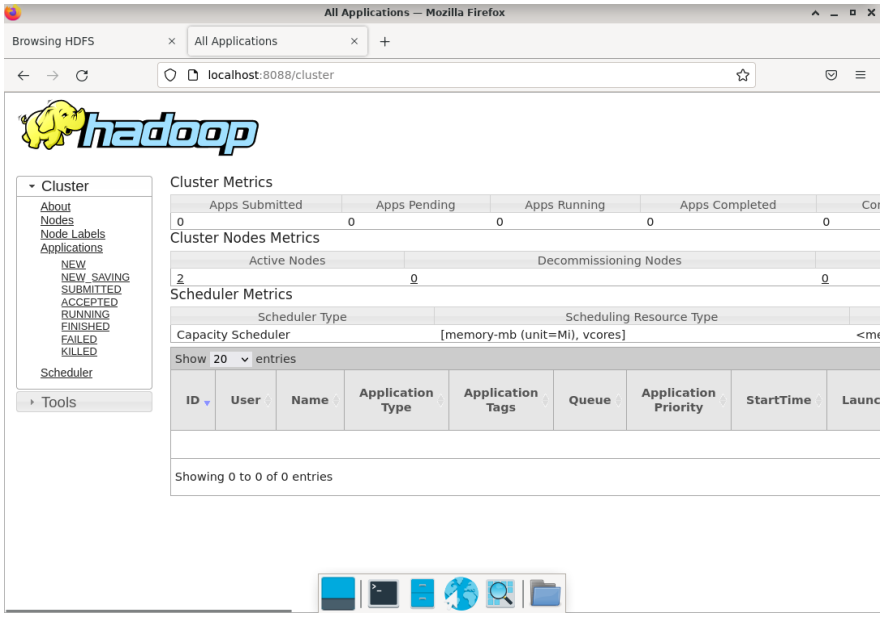


Η διαδικτυακή διεπαφή του YARN από προεπιλογή βρίσκεται στην διεύθυνση:

localhost:8088

Όμως μπορεί να ρυθμιστεί στο αρχείο **yarn-site.xml**

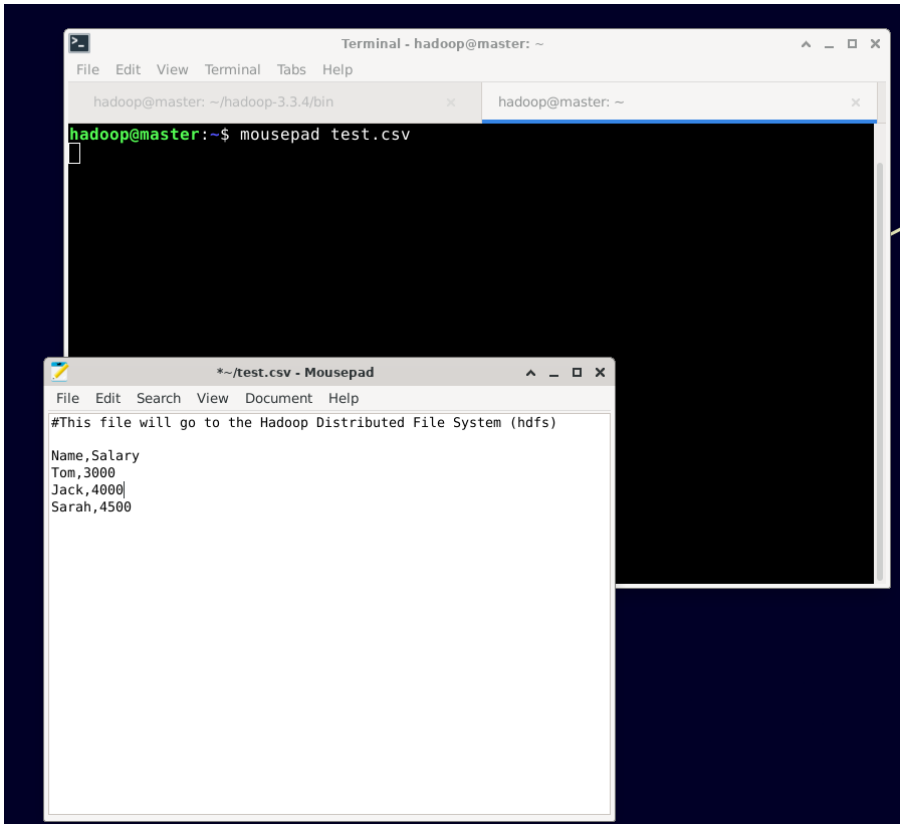
Στην καρτέλα "**Cluster**" βλέπουμε τους εν ενεργεία κόμβους, ενώ στην καρτέλα "**Scheduler**" τις ουρές που χρησιμοποιούνται και τον φόρτο εργασίας που έχουν.



Σε αυτό το σημείο θα φτιάξουμε ένα πειραματικό αρχείο csv και θα το τοποθετήσουμε στο κατανεμημένο σύστημα αρχείων Hadoop (HDFS).

Στο home directory τρέχουμε την εντολή:

➢ mousepad test.csv



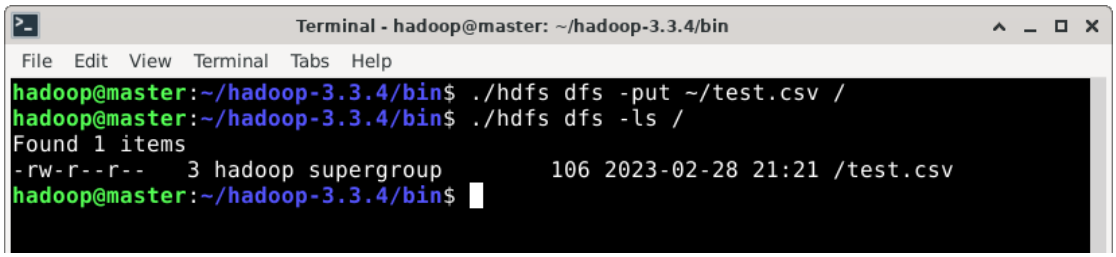
Στη συνέχεια, αφού αποθηκεύσουμε το αρχείο, από το sbin directory του Hadoop τρέχουμε την εντολή hadoop:

➢ ./hdfs dfs -put ~/test.csv /test.csv

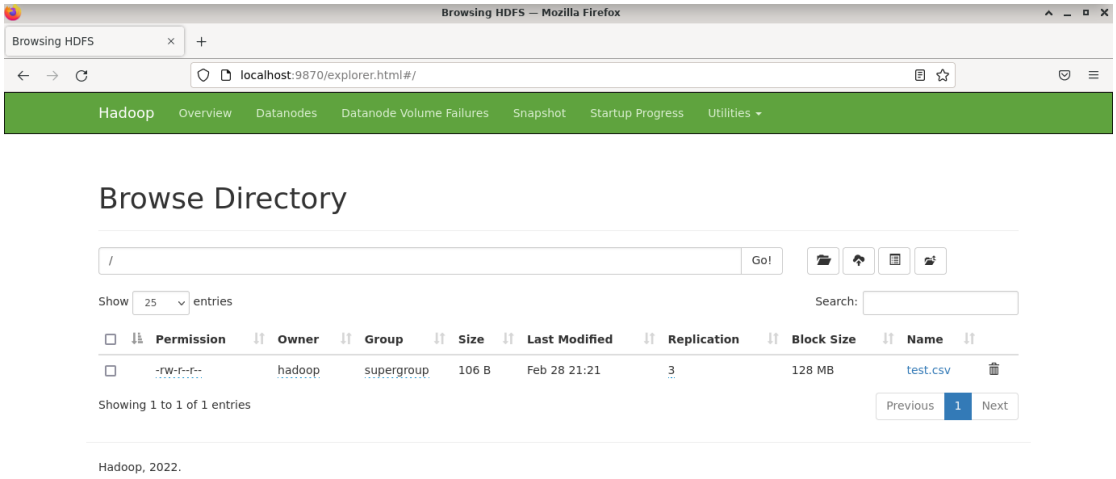
Για την αντιγραφή του αρχείου **test.csv** στον ριζικό φάκελο (root) του κατανεμημένου συστήματος αρχείων (HDFS).

➢ ./hdfs dfs -ls /

Η εντολή αυτή είναι η ίδια εντολή "ls" που είδαμε στα προηγούμενα βήματα με την διαφορά ότι εδώ θα εκτελεστεί στο HDFS. Το HDFS είναι συμβατό με όλες τις λειτουργίες ενός συστήματος αρχείων, όπως αντιγραφή/μετακίνηση/διαγραφή αρχείων και φακέλων ανάλογα με τα δικαιώματα που έχουμε.



Μπορούμε να έχουμε πρόσβαση στα αρχεία που βρίσκονται στο HDFS και από την διαδικτυακή διεπαφή Hadoop στην καρτέλα-μενού "**Utilities / Browse the file system**" που βρίσκεται στα δεξιά με τις ίδιες δυνατότητες που έχουμε και στο τερματικό.



2.10 ΒΗΜΑ 8 - ΔΟΚΙΜΗ ΤΟΥ APACHE SPARK

Στο όγδοο και τελευταίο βήμα θα προετοιμάσουμε το Spark για χρήση με γλώσσα Scala και Python. Επίσης θα τρέξουμε απλούς κώδικες και θα δούμε τα αποτελέσματά τους.

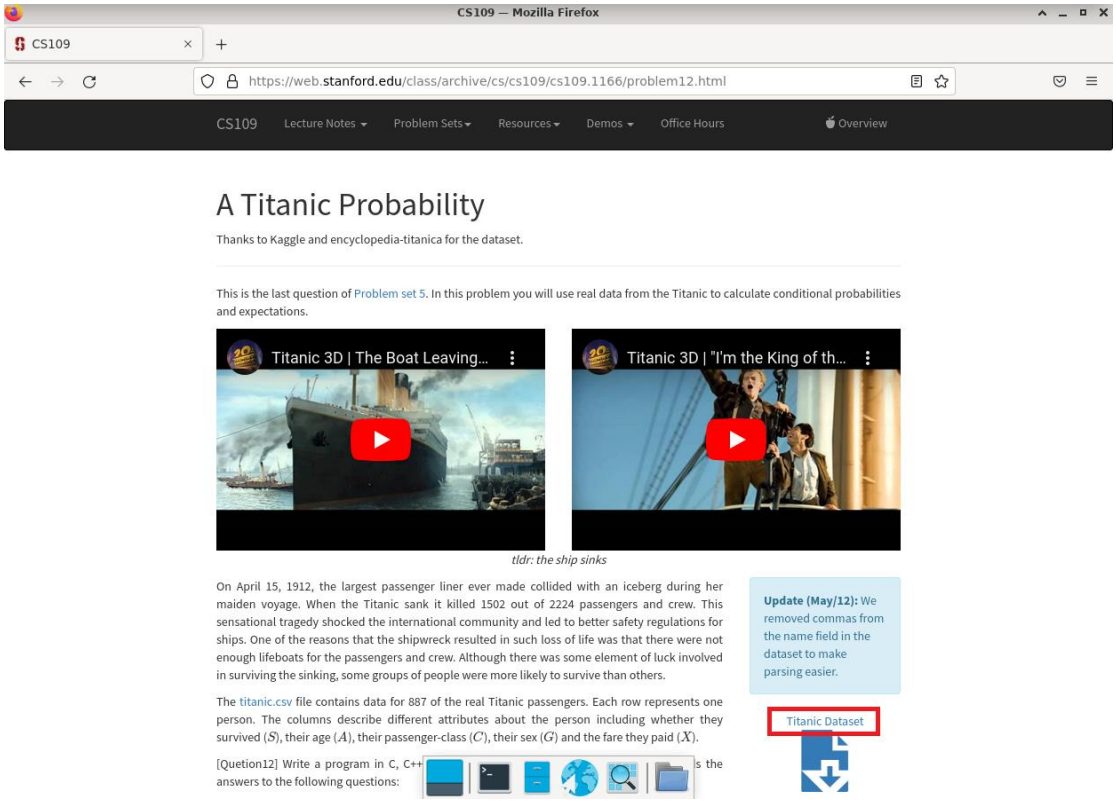
Για να δοκιμάσουμε την εκτέλεση σε κέλυφος Scala και Python θα πρέπει να κατεβάσουμε ένα δοκιμαστικό dataset σε μορφή csv.

Στον παρών οδηγό έχει χρησιμοποιηθεί το "**Titanic_Dataset**" του πανεπιστημίου Stanford.

Το "**Titanic_Dataset**" είναι ένα δημοφιλές σετ δεδομένων για μηχανική μάθηση που περιέχει τις πληροφορίες των επιβατών του πλοίου "Τιτανικός", συμπεριλαμβανομένων των δημογραφικών τους δεδομένων, την κατηγορία των εισιτηρίων τους, την καμπίνα τους και το ένδειξη επιβίωσής τους.

Μπορούμε να το κατεβάσουμε από εδώ:

https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/problem12.html



Στον σύνδεσμο με το όνομα του dataset, κάνουμε δεξί κλικ και επιλέγουμε αντιγραφή συνδέσμου (Copy link address).

Από το home directory τρέχουμε την εντολή:

➢ wget [Σύνδεσμος για το Titanic_Dataset]

Τέλος αντιγράφουμε το αρχείο στο HDFS για να βρίσκεται έτοιμο προς χρήση.

Από τον φάκελο sbin του hadoop directory τρέχουμε:

➢ ./hdfs dfs -put ~/titanic.csv /titanic.csv

2.10.1 ΔΟΚΙΜΗ ΤΟΥ APACHE SPARK ΜΕ SCALA SHELL

Για την εκτέλεση κώδικα σε γλώσσα scala θα πρέπει να εγκαταστήσουμε τις βιβλιοθήκες της γλώσσας καθώς και το SBT (Simple Build Tool) της Scala.

Το SBT είναι ένα εργαλείο δημιουργίας ανοιχτού κώδικα για έργα Scala και Java.

Υποστηρίζει την μεταγλώττιση μεγάλων project με εκατοντάδες ή χιλιάδες source files χωρίς να απαιτείται αυτό να γίνει χειροκίνητα.

Οι εντολές για την εγκατάσταση τους είναι οι εξής:

➢ sudo apt-get update

➢ sudo apt-get install apt-transport-https curl gnupg -yqq

➢ echo "deb https://repo.scala-sbt.org/scalasbt/debian all main" | sudo tee /etc/apt/sources.list.d/sbt.list

➢ echo "deb https://repo.scala-sbt.org/scalasbt/debian /" | sudo tee /etc/apt/sources.list.d/sbt_old.list

➢ curl -sL "https://keyserver.ubuntu.com/pks/lookup?op=get&search=0x2EE0EA64E40A89B84B2DF73499E82A75642AC823" | sudo -H gpg --no-default-keyring --keyring gnupg-ring:/etc/apt/trusted.gpg.d/scalasbt-release.gpg --import

➢ sudo chmod 644 /etc/apt/trusted.gpg.d/scalasbt-release.gpg

➢ sudo apt-get update

➢ sudo apt-get install sbt

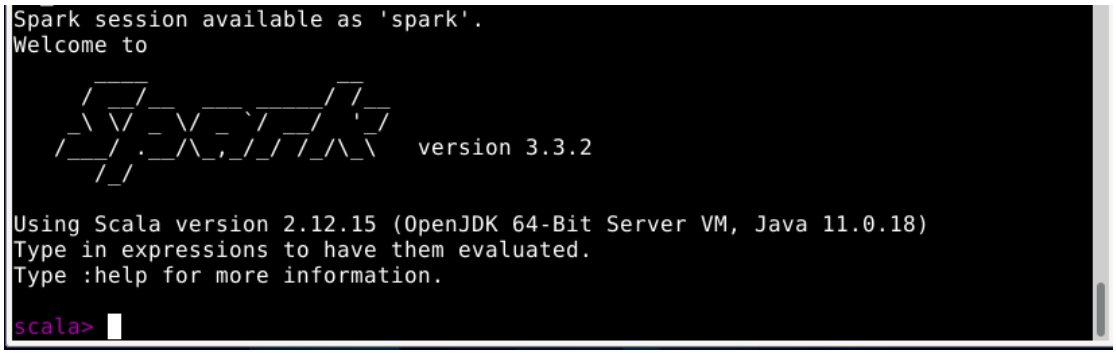
➢ sudo apt-get install scala

Για την εκτέλεση μιας εφαρμογής σε Scala χρησιμοποιούμε το spark-shell.

Είναι ένα διαδραστικό κέλυφος που λειτουργεί στο terminal.

Για να το ενεργοποιήσουμε, από τον φάκελο bin του spark directory πληκτρολογούμε την εντολή:

➢ ./spark-shell



Από εδώ μπορούμε είτε να τρέξουμε εντολές μια προς μία για πειραματικούς σκοπούς, είτε να υποβάλλουμε κώδικα προς εκτέλεση.

Κλείνουμε το spark-shell και προχωράμε με το να φτιάξουμε έναν δοκιμαστικό κώδικα σε γλώσσα scala.

Για να έχουμε τα αρχεία μας τακτοποιημένα δημιουργούμε ένα directory για το πρόγραμμα που θα φτιάξουμε. Μέσα στο directory αυτό θα φτιάξουμε subdirectories (υποφακέλους) όπως χρειάζεται το SBT για να πραγματοποιήσει την μεταγλώττιση του κώδικα σε αρχείο scala.

Από το home directory τρέχουμε:

➢ mkdir test_app

➢ mkdir -p test_app/src/main/scala

➢ cd test_app

➢ mousepad build.sbt

Για την δημιουργία αρχείου τύπου "**build**" που θα χρησιμεύσει πριν την μεταγλώττιση.

Στο αρχείο **build.sbt** γράφουμε:

name := “Scala Test”

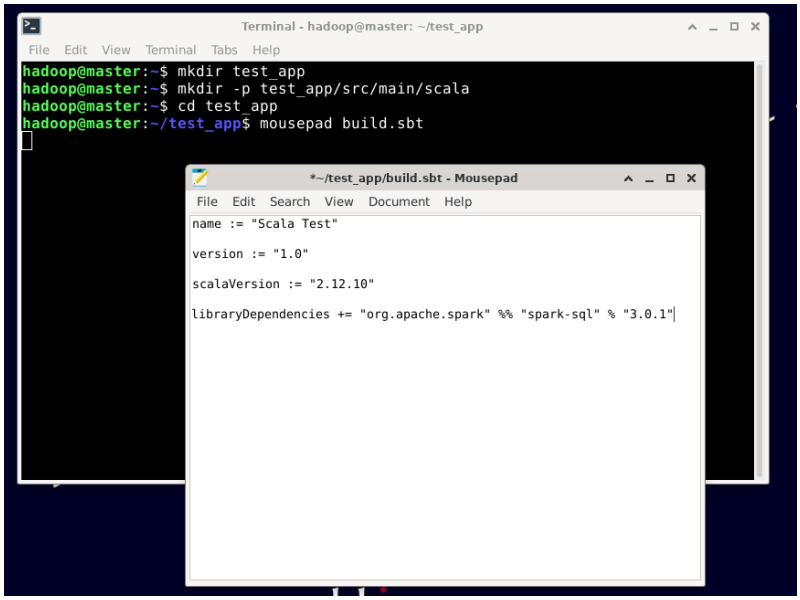
version := “1.0”

scalaVersion := “2.12.10”

libraryDependencies += “org.apache.spark” %% “spark-sql” % “3.0.1”

Όπου

- name: Το όνομα της εφαρμογής που φτιάχνουμε
- version: Η έκδοση της εφαρμογής που φτιάχνουμε
- scalaVersion: Η έκδοση της Scala που θα χρησιμοποιηθεί. Το Spark δεν είναι συμβατό με όλες τις εκδόσεις Scala.
- libraryDependencies: Οι βιβλιοθήκες spark που θα χρησιμοποιηθούν.



➢ mousepad src/main/scala/ScalaTest.scala

Δημιουργούμε τον κώδικα scala.

```
import org.apache.spark.sql.SparkSession

object ScalaTest {

    def main(args: Array[String]){

        val spark = SparkSession.builder.appName("Submitted").getOrCreate()
        val df = spark.read.format("csv").option("header", true).option("separator", ",").load("hdfs:///titanic.csv")

        df.show(10, false)

        spark.stop()

    }

}
```

Τα βήματα του αλγορίθμου είναι:

1) Εισαγωγή της βιβλιοθήκης Συνεδρίας Spark (SparkSession)
2) Δημιουργείται μια **Συνεδρία Spark (SparkSession)** με το όνομα “Submitted” και αποθηκεύεται για άμεση χρήση σε μια μεταβλητή με το όνομα “**spark**”. Η **Συνεδρία Spark** παρέχει κάποιες παραμέτρους που μπορεί να χρησιμοποιήσει ο χρήστης. Μία από αυτές είναι η παράμετρος “**read**”.
3) Η παράμετρος “**spark.read**” προέρχεται από το σύνολο παραμέτρων της **Συνεδρίας Spark** (ως μεταβλητή **spark**) και φέρει την παράμετρο “**csv**” ως

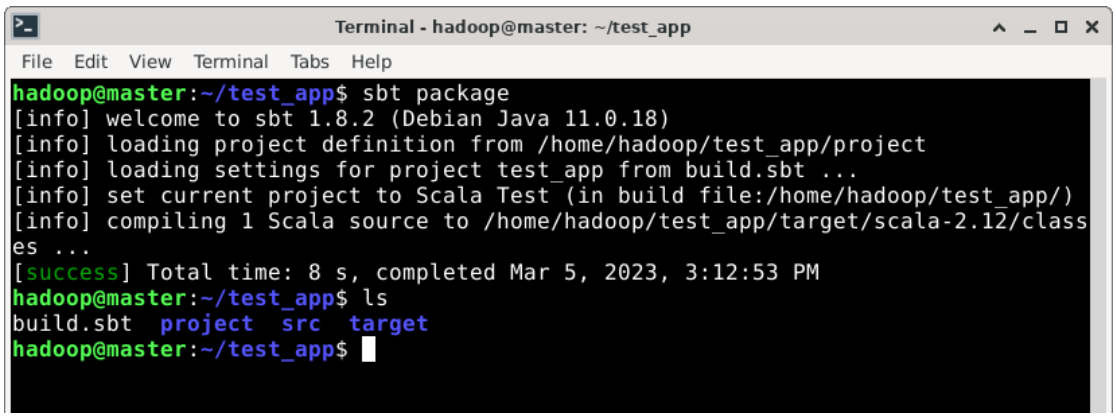
διευκρίνηση που σημαίνει ότι θα χρησιμοποιηθεί για την ανάγνωση ενός αρχείου "csv". Επίσης, όπως βλέπουμε χρησιμοποιούνται επιπλέων παράμετροι που βοηθούν στην σωστή ανάγνωση των δεδομένων. Στην προκειμένη περίπτωση έχουμε την παράμετρο "**.option("header", true)**" που δηλώνει ότι στο σετ δεδομένων υπάρχει κεφαλίδα και η παράμετρος "**.option("separator", ",")**" που δηλώνει ότι οι εγγραφές χωρίζονται με κόμμα. Τέλος, η παράμετρος "**.load("hdfs:///titanic.csv")**" δηλώνει το όνομα του αρχείου που θα φορτωθεί καθώς και η τοποθεσία του. Το συγκεκριμένο σετ δεδομένων είναι δομημένης μορφής με συγκεκριμένο σχήμα.

4) Σε μια μεταβλητή "**df**" αποθηκεύεται το αρχείο με την μορφή **spark DataFrame**.
5) Η εντολή "**df.show(10, false)**" εμφανίζει στην οθόνη τα 10 πρώτα αποτελέσματα από το **spark DataFrame** (Το false εδώ σημαίνει ότι στην οθόνη θα εμφανίζονται ολόκληρο το περιεχόμενο των στηλών του DataFrame).
6) Με την εντολή "**spark.stop()**" τερματίζει η εκτέλεση του προγράμματος.

Αποθηκεύουμε το αρχείο και τρέχουμε την εντολή:

➢ sbt package

για να πακεταριστεί ο κώδικας σε ένα διαμοιράσιμο αρχείο JAR (Java Archive).



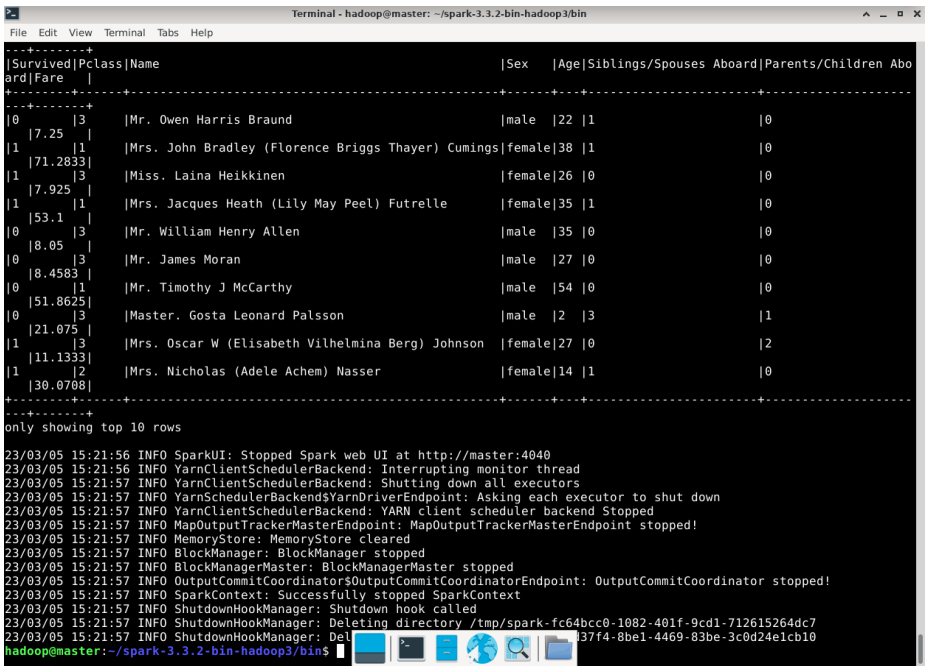
Για να υποβάλουμε το πρόγραμμα που δημιουργήσαμε προς εκτέλεση στο Spark, από το bin directory του Spark, χρησιμοποιούμε την εντολή:

➢ ./spark-submit --class ScalaTest --master yarn --queue dev ~/test_app/target/scala-2.12/scala-test_2.12-1.0.jar

Όπου:

- --class: Το πλήρες όνομα της κλάσης που περιέχει την κύρια μέθοδο της εφαρμογής Scala.
- --master: Το όνομα του Resource Manager.
- --queue: Η ουρά του YARN scheduler που θα χρησιμοποιηθεί.

Το αποτέλεσμα που παίρνουμε μοιάζει με αυτό:



2.10.2 ΔΟΚΙΜΗ ΤΟΥ APACHE SPARK ΜΕ ΤΟ PYSPARK

Για την χρήση του Spark με γλώσσα Python απαιτείται κάποια προεργασία, όπως και με την Scala.

Αρχικά θα πρέπει να έχουμε εγκατεστημένη την γλώσσα Python. Οι περισσότερες διανομές Linux έχουν προεγκατεστημένη την Python. Ακόμη κι έτσι όμως προτείνεται η εγκατάσταση και χρήση της πλατφόρμας Anaconda.

Η Anaconda είναι μια διανομή των γλωσσών προγραμματισμού Python και R, που στοχεύει στην απλοποίηση της διαχείρισης και της ανάπτυξης πακέτων. Η διανομή περιλαμβάνει πακέτα επιστήμης δεδομένων κατάλληλα για Windows, Linux και macOS.

Παρέχει επίσης κάποια χρήσιμα εργαλεία που χρησιμοποιούνται από επιστήμονες δεδομένων όπως το Spyder, το Rstudio και το Jupyter Notebook. Ένα άλλο σημαντικό χαρακτηριστικό είναι η ύπαρξη περιβαλλόντων. Μπορούμε να δημιουργήσουμε πολλά περιβάλλοντα στα οποία έχουμε πρόσβαση είτε από το γραφικό περιβάλλον της Anaconda είτε απευθείας από την γραμμή εντολών. Το κάθε περιβάλλον μπορεί να έχει πολλά πακέτα εγκατεστημένα χωρίς αυτά να αλληλεπιδρούν με τα υπόλοιπα περιβάλλοντα και έτσι μειώνεται η πιθανότητα σφαλμάτων και δυσλειτουργιών.

Η διανομή Anaconda είναι διαθέσιμη στον παρακάτω σύνδεσμο:

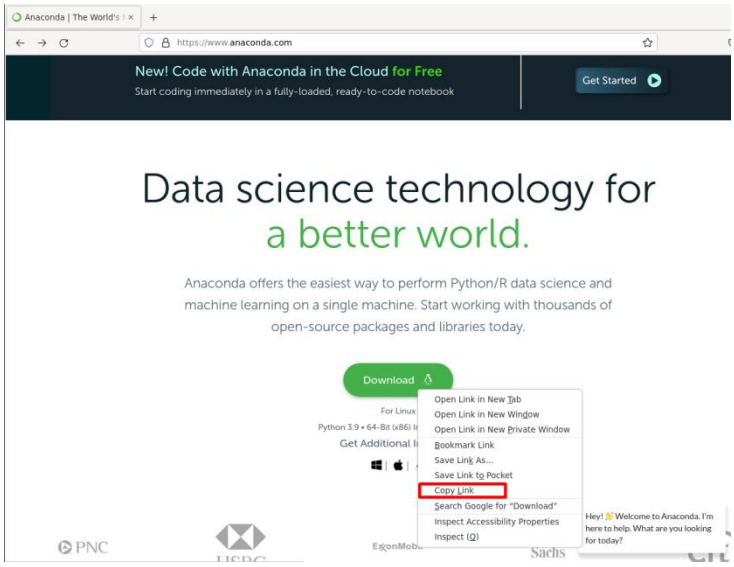
https://www.anaconda.com/

ΣΗΜΕΙΩΣΗ: Εάν ο αναγνώστης επιθυμεί να κατεβάσει την έκδοση που χρησιμοποιήθηκε στον παρών οδηγό θα την βρει εδώ:

https://repo.anaconda.com/archive/

Και θα πρέπει να προχωρήσει εδώ

Στην κεντρική σελίδα κάνουμε δεξί κλικ στο κουμπί με την ένδειξη “Download” και επιλέγουμε αντιγραφή συνδέσμου (Copy link address).



Για όσους επέλεξαν την έκδοση Anaconda του οδηγού, τότε στην σελίδα που οδηγεί ο σύνδεσμος κάνουμε δεξί κλικ στην έκδοση "**Anaconda3-2022.10-Linux-x86_64.sh**" και επιλέγουμε αντιγραφή συνδέσμου (Copy link address).

Index of /

Filename	Size	Last Modified	SHA256
.winzip/	-		<directory>
Anaconda3-2022.10-Linux-aarch64.sh	534.5M	2022-10-17 16:15:40	fbadbfae5992a8c96af0a4621262080eea44e22baee2
Anaconda3-2022.10-Windows-x86_64.exe	621.2M	2022-10-17 16:15:39	38b9d53a579843fe41fd05fd3c4f9ac3887f580e7bd9
Anaconda3-2022.10-Linux-x86_64.sh	737.6M	2022-10-17 16:15:39	e7ecbccbc197ebd7e1f211c59df2e37bc6959d081f22
Anaconda3-2022.10-MacOSX-x86_64.pkg	688.6M	2022-10-17 16:15:38	bd6147a59939009718ecc18ed6fd0cf1639dc1f1626a
Anaconda3-2022.10-MacOSX-arm64.sh	472.5M	2022-10-17 16:15:38	200700077db8eed762fbc996b830c3f8cc5a2bb7d6b2
Anaconda3-2022.10-MacOSX-x86_64.sh	681.6M	2022-10-17 16:15:37	dfcd1431a8206506799cb142b04d2db3be8a28671e5c
Anaconda3-2022.10-Linux-s390x.sh	282.4M	2022-10-17 16:15:37	f5ccc24aedab1f3f9cccf1945ca1061bee194fa42a21
Anaconda3-2022.10-Linux-ppc64le.sh	360.0M	2022-10-17 16:15:37	8fdebc79f63b74daad421a2674d43299fa9c5007d85c
Anaconda3-2022.10-MacOSX-arm64.pkg	484.1M	2022-10-17 16:15:36	4999ce8718c5d387940b1e213beb2c525e61eca94fd0
Anaconda3-2022.05-MacOSX-arm64.sh	304.8M	2022-06-07 12:40:25	a12119931945a9a1453993582259cc67318a9a75a157
Anaconda3-2022.05-MacOSX-arm64.pkg	316.4M	2022-06-07 12:40:24	0140970944a3e6088be5995ef7ce8525c1b2f8d5080e

Από το home directory:

➢ wget [Σύνδεσμος για το αρχείο Anaconda]

Για να μπορέσουμε να εγκαταστήσουμε πακέτα σε γραφικό περιβάλλον πρέπει προηγουμένως να τρέξουμε την εντολή:

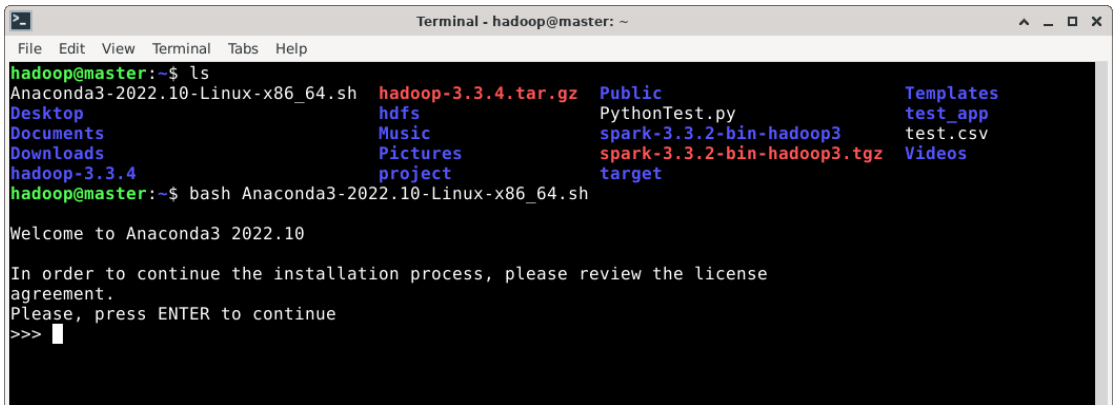
➢ apt-get install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1 libxcomposite1 libasound2 libxi6 libxtst6

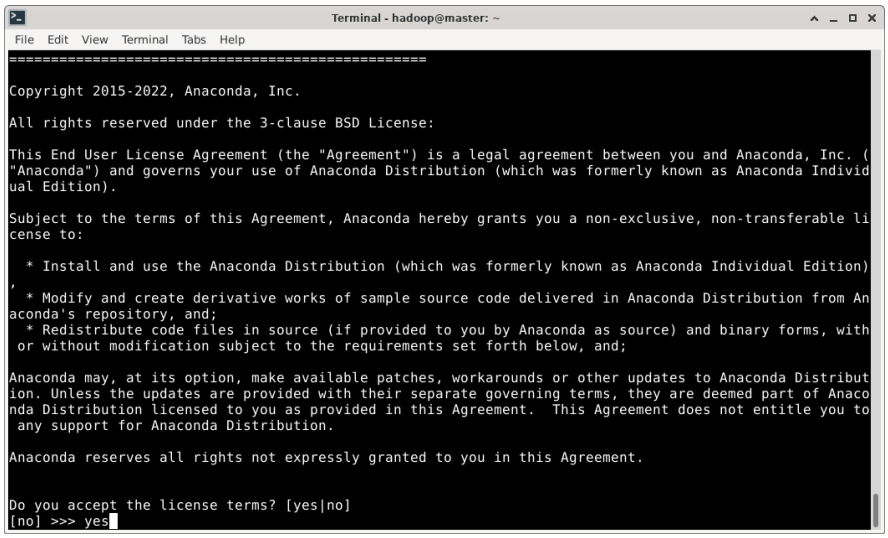
όπου εγκαθιστά τις απαραίτητες βιβλιοθήκες.

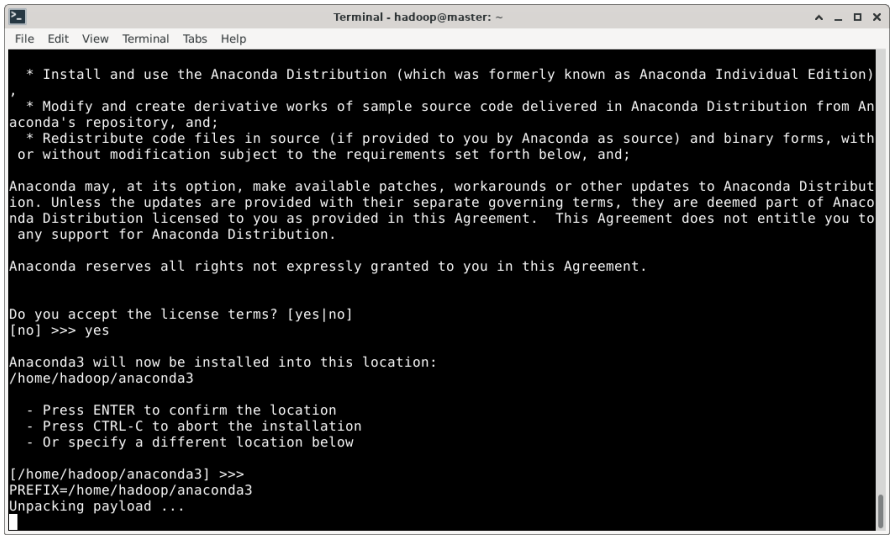
Από το home directory που αποθηκεύτηκε το αρχείο εγκατάστασης τρέχουμε:

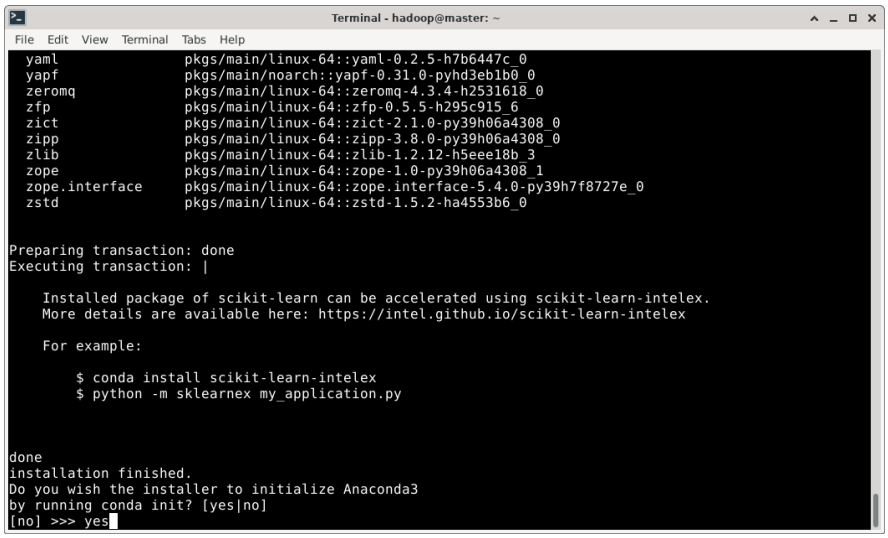
➢ bash ./Anaconda3-2020.05-Linux-x86_64.sh

και ακολουθούμε τις οδηγίες στις εικόνες:









Αφού εγκαταστάθηκε η Anaconda κλείνουμε το τερματικό και το ανοίγουμε ξανά. Παρατηρούμε στο τερματικό την ένδειξη “base”. Η ένδειξη αυτή αναφέρεται στο προεπιλεγμένο περιβάλλον Anaconda που είναι ενεργό. Για λόγους ασφαλείας προτιμούμε να μην χρησιμοποιούμε το προεπιλεγμένο βασικό περιβάλλον αλλά να δημιουργήσουμε ένα καινούριο.

Για να το καταφέρουμε αυτό χρησιμοποιούμε την εντολή:

➢ conda create --name pyspark_env

Έτσι φτιάξαμε το περιβάλλον που θα χρησιμοποιήσουμε με το Spark για Python.

Για να εξέλθουμε από το τρέχων περιβάλλον:

➢ conda deactivate

ενώ για να εισέλθουμε ξανά:

➢ conda activate pyspark_env

Εάν δοκιμάσουμε να κλείσουμε το terminal και να το ανοίξουμε ξανά παρατηρούμε ότι το έχει επανέλθει το βασικό περιβάλλον.

Σε περίπτωση που θέλουμε να έχουμε το περιβάλλον που δημιουργήσαμε ως προεπιλεγμένο τότε ανοίγουμε το αρχείο bashrc➢ mousepad ~/.bashrc

και γράφουμε στο τέλος του bashrc την εντολή:

conda activate pyspark_env

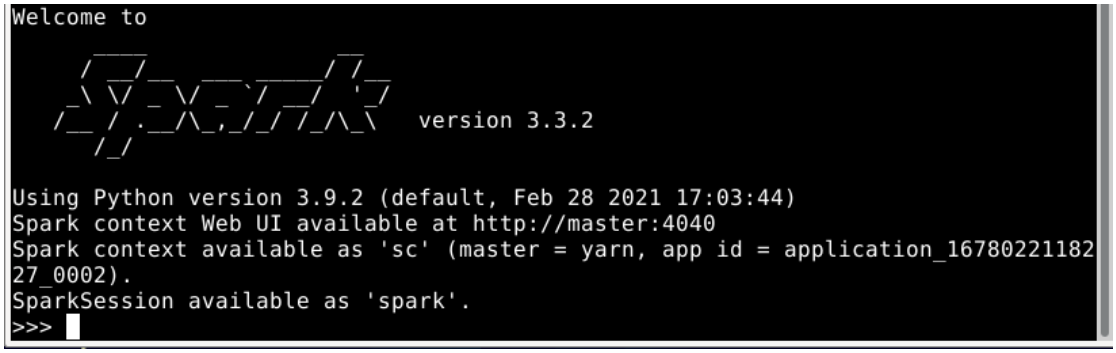
Έτσι κάθε φορά που θα αρχικοποιείται το κέλυφος του τερματικού θα ενεργοποιείται αυτόματα το περιβάλλον αυτό.

Για την εκτέλεση μιας εφαρμογής σε Scala χρησιμοποιούμε το pyspark.

Είναι ένα διαδραστικό κέλυφος που λειτουργεί στο terminal.

Για να το ενεργοποιήσουμε, από τον φάκελο bin του spark directory πληκτρολογούμε την εντολή:

➢ ./pyspark



Από εδώ μπορούμε είτε να τρέξουμε εντολές μια προς μία για πειραματικούς σκοπούς, είτε να υποβάλλουμε κώδικα προς εκτέλεση.

Κλείνουμε το pyspark shell και προχωράμε με το να φτιάξουμε έναν δοκιμαστικό κώδικα σε γλώσσα Python.

Από το home directory τρέχουμε:

➤ mousepad PythonTest.py

Δημιουργούμε τον κώδικα Python.

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Submitted").getOrCreate()

df = spark.read.format("csv").option("header", True).option("separator", ",").load("hdfs:///titanic.csv")

df.show(10, False)
```

Τα βήματα του αλγορίθμου είναι:

1) Εισαγωγή της βιβλιοθήκης Συνεδρίας Spark (SparkSession)
2) Δημιουργείται μια **Συνεδρία Spark (SparkSession)** με το όνομα "Submitted" και αποθηκεύεται για άμεση χρήση σε μια μεταβλητή με το όνομα "**spark**". Η **Συνεδρία Spark** παρέχει κάποιες παραμέτρους που μπορεί να χρησιμοποιήσει ο χρήστης. Μία από αυτές είναι η παράμετρος "**read**".
3) Η παράμετρος "**spark.read**" προέρχεται από το σύνολο παραμέτρων της **Συνεδρίας Spark** (ως μεταβλητή **spark**) και φέρει την παράμετρο "**format.("csv")**" ως διευκρίνηση που σημαίνει ότι θα χρησιμοποιηθεί για την ανάγνωση ενός αρχείου "csv". Επίσης, όπως βλέπουμε χρησιμοποιούνται επιπλέον παράμετροι που βοηθούν στην σωστή ανάγνωση των δεδομένων. Στην προκειμένη περίπτωση έχουμε την παράμετρο "**.option("header", True)**" που δηλώνει ότι στο σετ δεδομένων υπάρχει κεφαλίδα και η παράμετρος "**.option("separator", ",")**" που δηλώνει ότι οι εγγραφές χωρίζονται με κόμμα. Τέλος, η παράμετρος "**.load("hdfs:///titanic.csv")**" δηλώνει το όνομα του αρχείου που θα φορτωθεί καθώς και η τοποθεσία του. Το συγκεκριμένο σετ δεδομένων είναι δομημένης μορφής με συγκεκριμένο σχήμα.
4) Σε μια μεταβλητή "**df**" αποθηκεύεται το αρχείο με την μορφή **pyspark DataFrame**.
5) Η εντολή "**df.show(10, false)**" εμφανίζει στην οθόνη τα 10 πρώτα αποτελέσματα από το **pyspark DataFrame** (Το False εδώ σημαίνει ότι στην οθόνη θα εμφανίζονται ολόκληρο το περιεχόμενο των στηλών του DataFrame).

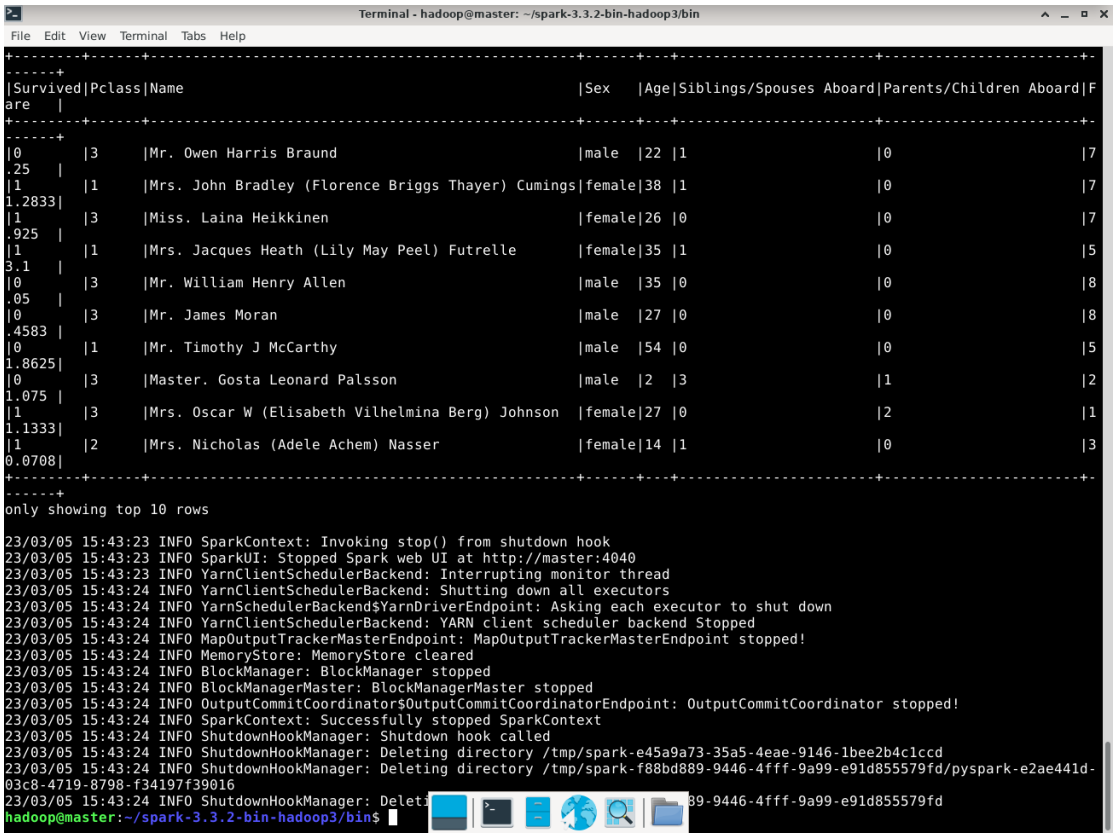
Για να υποβάλουμε το πρόγραμμα που δημιουργήσαμε προς εκτέλεση στο PySpark, από το bin directory του Spark, χρησιμοποιούμε την εντολή:

```
➢ ./spark-submit --master yarn --queue dev ~/PythonTest.py
```

Όπου:

- --master: Το όνομα του Resource Manager.
- --queue: Η ουρά του YARN scheduler που θα χρησιμοποιηθεί.

Το αποτέλεσμα που παίρνουμε μοιάζει με αυτό:



Το Anaconda διαθέτει κάποια πολύ χρήσιμα εργαλεία που χρησιμοποιούνται για την επιστήμη των δεδομένων. Αυτό με το οποίο θα ασχοληθούμε σε αυτό το σημείο είναι το Jupyter Notebook.

Το Jupyter Notebook χρησιμοποιείται από επιστήμονες δεδομένων, ερευνητές, εκπαιδευτικούς και προγραμματιστές σε διάφορους τομείς όπως η ανάλυση δεδομένων και η μηχανική μάθηση. Ο χρήστης μπορεί να γράψει και να εκτελέσει κώδικα και απεικονίσεις με αναπαραγόμενο τρόπο, καθώς η κάθε μία εντολή μπορεί να τρέξει ξεχωριστά και να παράγει αποτελέσματα. Χρησιμοποιείται για την ανάπτυξη δεδομένων και αλγορίθμων με συνδυασμούς κώδικα, κειμένου και πολυμεσικών στοιχείων.

Για να εγκαταστήσουμε το Jupyter Notebook στο περιβάλλον Anaconda που χρησιμοποιούμε βεβαιωνόμαστε πως βρισκόμαστε στο σωστό περιβάλλον και τρέχουμε την εντολή:

```
➢ conda install jupyter
```

Το Jupyter Notebook μπορεί να αντικαταστήσει το PySpark μέχρι ένα βαθμό αλλά χρησιμοποιείται κυρίως για ερευνητικούς σκοπούς καθώς αδυνατεί να διαχειριστεί μεγάλη ποσότητα δεδομένων.

Για την εκτέλεση κώδικα Pyspark σε Jupyter Notebook χρειαζόμαστε επίσης ένα πακέτο που περιλαμβάνει την βιβλιοθήκη findspark.

Η βιβλιοθήκη αυτή εισάγεται στην αρχή του κώδικα και έχει την ικανότητα να εντοπίζει και να αρχικοποιεί το περιβάλλον Spark στο Jupyter Notebook.

```
➢ conda install -c conda-forge findspark
```

Αν θελήσουμε να αναπτύξουμε μια εφαρμογή σε γλώσσα Python και να την τρέξουμε τοπικά στον κόμβο master (ή στον κόμβο που έχουμε επιλέξει να χρησιμοποιούμε για να δοκιμάζουμε κώδικα) χωρίς να χρειαζόμαστε ολόκληρη την συστάδα, τότε μπορούμε να χρησιμοποιήσουμε την βιβλιοθήκη της Python που ονομάζεται "**Pandas**".

Το Pandas είναι μια δημοφιλής βιβλιοθήκη ανοιχτού κώδικα για την επεξεργασία δεδομένων σε γλώσσα προγραμματισμού Python. Το Pandas παρέχει δύο κύριες δομές δεδομένων: τη σειρά (Series) και το πλαίσιο δεδομένων (DataFrame). Η Σειρά είναι μια μονοδιάστατη συλλογή δεδομένων με ετικέτα που μπορεί να κρατήσει οποιονδήποτε τύπο δεδομένων, ενώ το Πλαίσιο Δεδομένων είναι μια δομή παρόμοια με πίνακα δύο διαστάσεων που αποτελείται από γραμμές και στήλες, με ετικέτες στους άξονες. Το Pandas παρέχει μια μεγάλη ποικιλία συναρτήσεων για την επεξεργασία δεδομένων, συμπεριλαμβανομένων του φιλτραρίσματος, της επιλογής, της συγχώνευσης, της ομαδοποίησης και της

αναδιαμόρφωσης δεδομένων. Υποστηρίζει επίσης μια ποικιλία μορφών εισόδου/εξόδου, όπως CSV, Excel, βάσεις δεδομένων SQL και άλλα.

Για να εγκαταστήσουμε το Pandas και να μπορούμε να χρησιμοποιήσουμε την βιβλιοθήκη στο Jupyter Notebook υπάρχουν δύο τρόποι:

➢ pip install pandas

ή

➢ conda install pandas

Προτιμούμε τον δεύτερο τρόπο καθώς η εγκατάσταση γίνεται σε ένα συγκεκριμένο περιβάλλον.

Σε αυτό το σημείο μπορούμε να δοκιμάσουμε το Jupyter Notebook με ένα παράδειγμα κώδικα στο οποίο θα διαβάζουμε ένα dataset σε μορφή csv σαν pyspark dataframe και θα το μετατρέπουμε σε pandas.

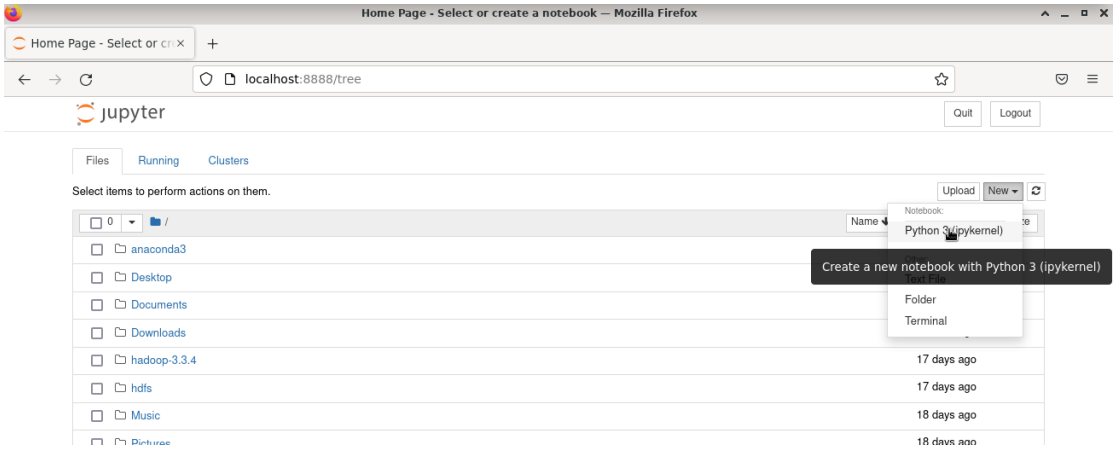
Ανοίγουμε ένα νέο instance του terminal και τρέχουμε την εντολή:

➢ jupyter notebook

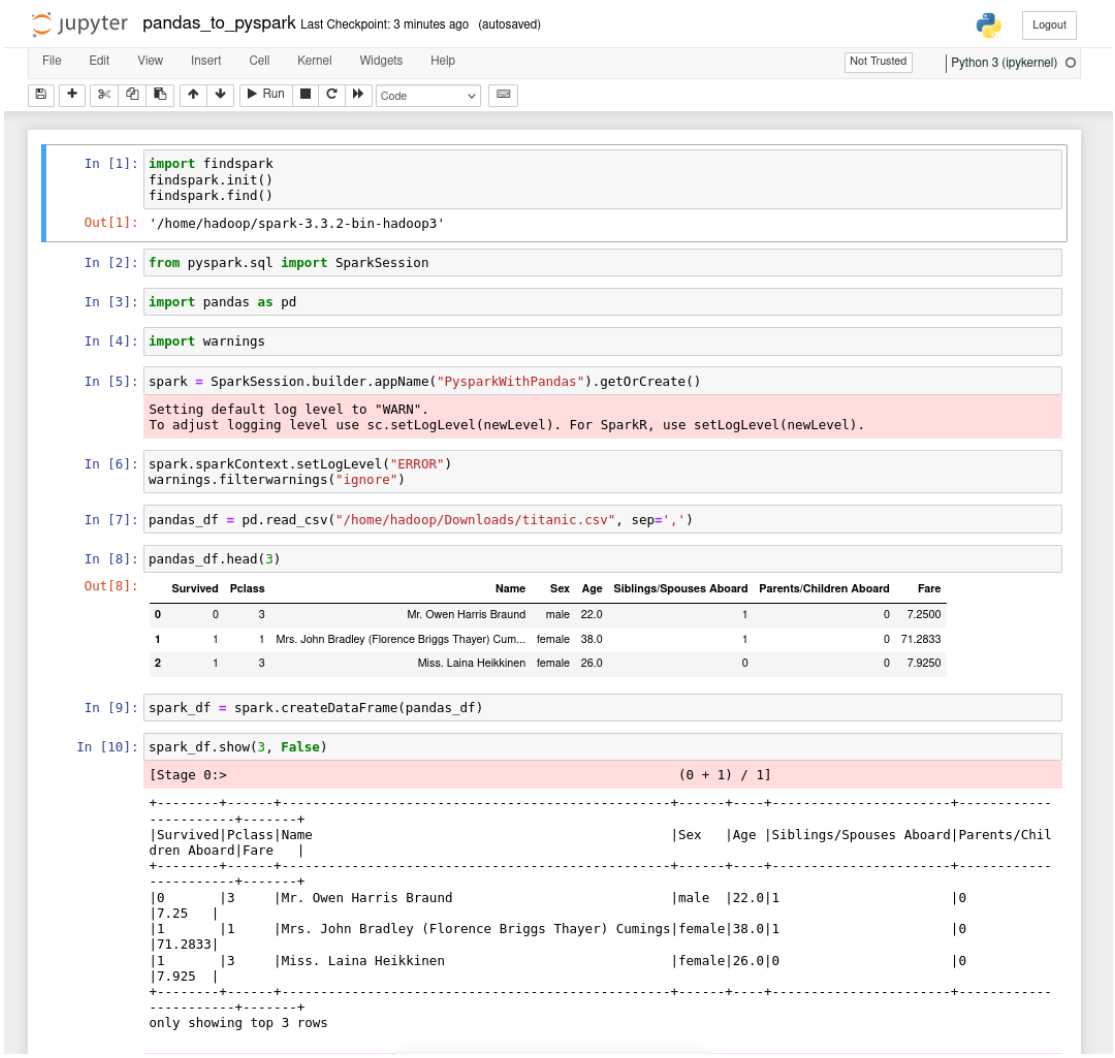
Τότε αυτόματα ανοίγει ο φυλλομετρητής (browser) το περιβάλλον εργασίας Jupyter στο directory από το οποίο τρέξαμε την εντολή. Σε περίπτωση που ο φυλλομετρητής δεν ανοίξει αυτόματα, πραγματοποιούμε την διαδικασία χειροκίνητα και κάνουμε αντιγραφή - επικόλληση τον σύνδεσμο που θα δούμε στο τερματικό που τρέχει το Jupyter Notebook.

Για να κλείσουμε το Jupyter Notebook πατάμε στο ίδιο τερματικό "ctrl + c" ή απλά κλείνουμε το τερματικό.

Στο δεξί μέρος του περιβάλλοντος εργασίας επιλέγουμε "**New**" και από το drop-down μενού επιλέγουμε "**Python 3**"



Στην καρτέλα που θα εμφανιστεί εισάγουμε τις εντολές του κώδικα που θέλουμε να τρέξουμε.



Στο παραπάνω παράδειγμα βλέπουμε το πως οι εντολές μπορούν να εκτελεστούν με αναπαραγόμενο τρόπο.

Τα βήματα από το 1 έως το 10 είναι τα εξής:

1) Εισαγωγή της βιβλιοθήκης "**findspark**" και αρχικοποίηση του Spark.
2) Εισαγωγή της βιβλιοθήκης "**SparkSession**" του PySpark.
3) Εισαγωγή της βιβλιοθήκης "**Pandas**".
4) Εισαγωγή της βιβλιοθήκης προειδοποιήσεων (warnings) της Python. (προαιρετικό: για την καλύτερη εμφάνιση των εντολών στο παράδειγμα).
5) Δημιουργία SparkSession με όνομα "PysparkWithPandas".
6) Αλλαγή του επιπέδου προειδοποιήσεων στο Spark και στην Python (προαιρετικό: για την καλύτερη εμφάνιση των εντολών στο παράδειγμα).
7) Δημιουργία ενός **Pandas DataFrame** από το αρχείο "**titanic.csv**" το οποίο όπως μπορούμε να δούμε αυτή τη φορά φορτώνεται τοπικά και όχι από το HDFS.
8) Εμφάνιση των πρώτων 3 γραμμών του **Pandas DataFrame**.
9) Μετατροπή του **Pandas DataFrame** σε **PySpark DataFrame**.
10) Εμφάνιση των πρώτων 3 γραμμών του **PySpark DataFrame**.

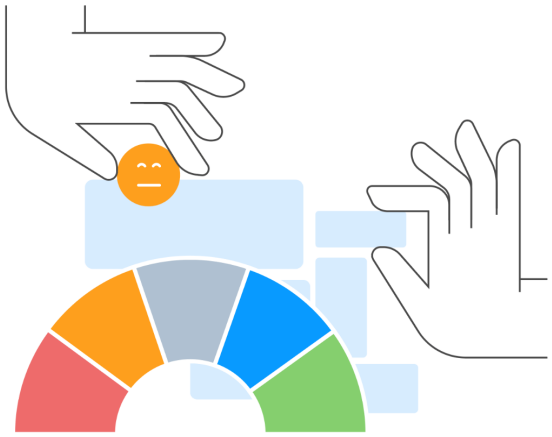
Αυτό ήταν το τέλος του οδηγού εγκατάστασης Spark με YARN και HDFS.

ΚΕΦΑΛΑΙΟ 3

3.1 Η ΤΕΧΝΙΚΗ ΤΗΣ ΑΝΑΛΥΣΗΣ ΣΥΝΑΙΣΘΗΜΑΤΟΣ (SENTIMENT ANALYSIS)

Η ανάλυση συναισθήματος η οποία αναφέρεται και ως εξόρυξη γνώμης (Opinion Mining) είναι μια τεχνική επεξεργασίας φυσικής γλώσσας (Natural Language Processing - NLP) που περιλαμβάνει τη χρήση αλγορίθμων και μοντέλων μηχανικής μάθησης για τον εντοπισμό και την εξαγωγή υποκειμενικών πληροφοριών από δεδομένα κειμένου. Ο στόχος της ανάλυσης συναισθήματος είναι να προσδιοριστεί ο συναισθηματικός τόνος ή το συναίσθημα ενός κειμένου, το οποίο μπορεί να είναι θετικό, αρνητικό ή ουδέτερο. Η διαδικασία της ανάλυσης συναισθήματος συνήθως περιλαμβάνει πολλά βήματα, τα οποία είναι η χρήση εξόρυξης δεδομένων, η προεπεξεργασία κειμένου (π.χ. αφαίρεση συνδετικών λέξεων (stopwords) ή δημιουργία λημμάτων) και η χρήση μηχανικής μάθησης (ML) για την εξόρυξη συναισθήματος από ένα σύνολο δεδομένων.

Τα συστήματα ανάλυσης συναισθήματος χρησιμοποιούν αναλυτικά στοιχεία κειμένου για να αναλύσουν διαδικτυακές πηγές, όπως μηνύματα ηλεκτρονικού ταχυδρομείου, αναρτήσεις ιστολογίου, διαδικτυακές κριτικές, εισιτήρια υποστήριξης πελατών, άρθρα ειδήσεων, απαντήσεις σε έρευνες, μελέτες περιπτώσεων, συνομιλίες ιστού, tweets, φόρουμ και σχόλια. Οι γνώσεις που αποκτώνται από την ανάλυση συναισθήματος μπορούν να χρησιμοποιηθούν για διάφορους σκοπούς, όπως παρακολούθηση επωνυμίας, έρευνα αγοράς, εξυπηρέτηση πελατών και διαχείριση φήμης. Εκτός από τον προσδιορισμό του συναισθήματος, η ανάλυση συναισθημάτων μπορεί να εξαγάγει την πολικότητα ή την ποσότητα της θετικής και αρνητικότητας, το θέμα και τον κάτοχο της γνώμης μέσα στο κείμενο. Αυτή η προσέγγιση χρησιμοποιείται για την ανάλυση διαφόρων τμημάτων του κειμένου, όπως ένα πλήρες έγγραφο ή μια παράγραφο, πρόταση ή δευτερεύουσα πρόταση. ([77])



Εικόνα 3.1: Ανάλυση Συναισθήματος (Sentiment Analysis) και κατηγοριοποίηση συναισθημάτων

Ακολουθούν ορισμένοι τύποι ανάλυσης συναισθήματος:

Ανάλυση συναισθήματος βάσει πρόθεσης:

Αυτός ο τύπος ανάλυσης συναισθήματος στοχεύει στον προσδιορισμό της πρόθεσης πίσω από ένα κείμενο. Εάν δηλαδή το κείμενο εκφράζει κάποιο παράπονο, ένα αίτημα ή μια πρόταση. Η ανάλυση συναισθήματος με βάση την πρόθεση μπορεί να βοηθήσει τις επιχειρήσεις να κατανοήσουν τις ανάγκες των πελατών τους και να λάβουν τις κατάλληλες ενέργειες για την εκπλήρωσή τους.

Λεπτομερής ανάλυση συναισθήματος:

Αυτός ο τύπος ανάλυσης συναισθήματος αναλύει την ένταση ή τη δύναμη του συναισθήματος που εκφράζεται σε ένα κείμενο. Για παράδειγμα, μια κριτική μπορεί να κατηγοριοποιηθεί ως "θετική", αλλά η λεπτομερής ανάλυση συναισθήματος μπορεί να δείξει εάν η κριτική είναι ήπια θετική ή εξαιρετικά θετική. Η λεπτομερής ανάλυση συναισθήματος μπορεί να προσφέρει πιο λεπτομερείς γνώσεις σχετικά με τις απόψεις και τις προτιμήσεις των πελατών.

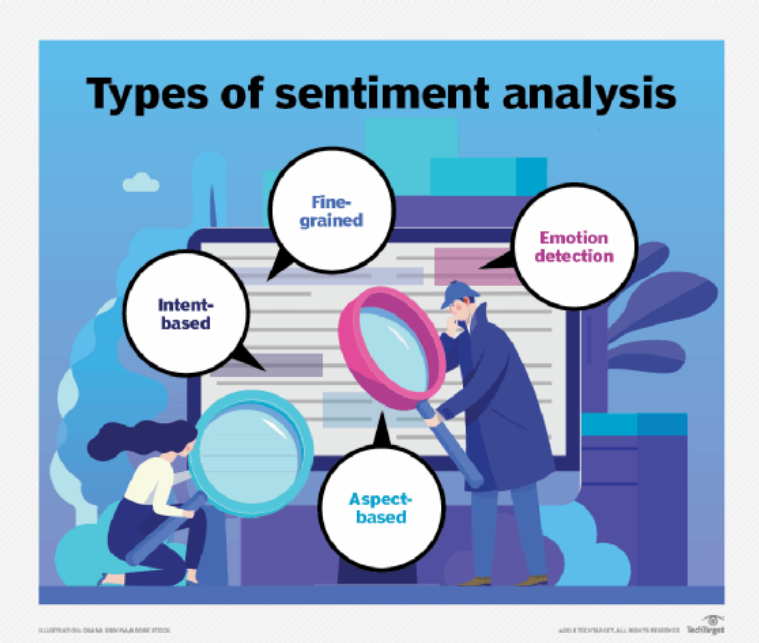
Ανίχνευση συναισθημάτων:

Αυτός ο τύπος ανάλυσης συναισθημάτων προσδιορίζει τα συναισθήματα που εκφράζονται σε ένα κείμενο, όπως χαρά, θυμός, λύπη, φόβος και έκπληξη. Η

ανίχνευση συναισθημάτων μπορεί να είναι χρήσιμη σε πολλές εφαρμογές, όπως η παρακολούθηση μέσων κοινωνικής δικτύωσης, η εξυπηρέτηση πελατών και η υγειονομική περίθαλψη. Για παράδειγμα, η ανίχνευση συναισθημάτων μπορεί να βοηθήσει τις επιχειρήσεις να εντοπίσουν πελάτες που εκφράζουν απογοήτευση ή δυσαρέσκεια και να ανταποκριθούν σε αυτούς έγκαιρα και με ενσυναίσθηση.

Ανάλυση συναισθήματος βάσει πτυχών:

Αυτός ο τύπος ανάλυσης συναισθήματος εστιάζει σε συγκεκριμένες πτυχές ή χαρακτηριστικά ενός προϊόντος, υπηρεσίας ή επωνυμίας και αναλύει το συναίσθημα που σχετίζονται με κάθε πτυχή. Για παράδειγμα, μια ανάλυση συναισθήματος που βασίζεται σε πτυχές και αφορά ένα smartphone μπορεί να αναλύσει το συναίσθημα που σχετίζεται με την κάμερα, τη διάρκεια ζωής της μπαταρίας, τη σχεδίαση και την απόδοση. Η ανάλυση συναισθήματος βάσει πτυχών μπορεί να βοηθήσει τις επιχειρήσεις να εντοπίσουν τομείς βελτίωσης και να δώσουν προτεραιότητα στην βελτίωση συγκεκριμένων χαρακτηριστικών των προϊόντων τους.



Εικόνα 3.1.1: Είδη Ανάλυσης Συναισθήματος.

Πώς λειτουργεί όμως η ανάλυση συναισθήματος;

Η ανάλυση συναισθήματος χρησιμοποιεί μοντέλα μηχανικής μάθησης για την εκτέλεση ανάλυσης κειμένου της ανθρώπινης γλώσσας. Οι μετρήσεις που χρησιμοποιούνται έχουν σχεδιαστεί για να ανιχνεύουν εάν το συνολικό συναίσθημα ενός κειμένου είναι θετικό, αρνητικό ή ουδέτερο.

Εικόνα 3.1.2: Χρήση της τεχνικής Εξόρυξης Δεδομένων (Data Mining).

Συλλογή δεδομένων

Η συλλογή δεδομένων προς ανάλυση επιτυγχάνεται με μια διαδικασία που ονομάζεται Εξόρυξη Δεδομένων (Data Mining) η οποία έχει γίνει ευρέως χρησιμοποιούμενη.

Το κείμενο που αναλύεται αναγνωρίζεται και συλλέγεται από αυτοματοποιημένους αλγορίθμους (bots) γνωστοί και ως scrappers ή miners. Σκοπός των bots είναι να εξάγουν χρήσιμες πληροφορίες από συγκεκριμένα σημεία και πηγές που μπορεί να βρίσκονται οπουδήποτε στο δίκτυο με ανοιχτή πρόσβαση ανάλογα με τις ανάγκες.

Οι πληροφορίες που συλλέγονται μπορούν να χρησιμοποιηθούν σε εφαρμογές επιχειρηματικής ευφυΐας (Business Intelligence - BI) και προηγμένες εφαρμογές ανάλυσης που περιλαμβάνουν ανάλυση ιστορικών δεδομένων, καθώς και σε εφαρμογές ανάλυσης πραγματικού χρόνου που εξετάζουν δεδομένα ροών (data streams) καθώς δημιουργούνται ή συλλέγονται.

Μια ιδιαίτερα δημοφιλής πηγή εξόρυξης δεδομένων αποτελούν τα κοινωνικά δίκτυα, όπως το Facebook, το Instagram και το Twitter.

Σύμφωνα με δημοσίευση του πανεπιστημίου Lahore, μια έρευνα με θέμα την χρήση του Twitter από αρχηγούς του Παγκόσμιου Πολιτικού Φόρουμ (G7) που περιλαμβάνουν την Κίνα, της Ηνωμένες Πολιτείες της Αμερικής, την Γαλλία, τον Καναδά, την Ιταλία και την Ευρωπαϊκή Ένωση και η οποία διεξήχθη από της 17 Νοεμβρίου 2019 έως της 17 Μαρτίου 2020 ανακάλυψε 203 tweets σχετικά με τον Covid-19 τα οποία έγιναν viral.

Από τα 203 μηνύματα, τα 166 κατηγοριοποιήθηκαν ως ενημερωτικά, τα 48 διέθεταν συνδέσμους σε κρατικές πηγές, τα 19 χαρακτηρίστηκαν ως εμψυχωτικά και τα 4 από αυτά ήταν πολιτικής φύσεως.

Ο λογαριασμός του Προέδρου των Ηνωμένων Πολιτειών Ντόναλντ Τραμπ στο Twitter αποτελούσε ανέκαθεν πηγή πολλών αντιπαραθέσεων. Η συχνή χρήση του ιστοχώρου αλλά και η επιρροή που ασκεί στα κοινωνικά δίκτυα χρησιμοποιώντας αθέμιτες πρακτικές, έχουν οδηγήσει σε πολλές επιστημονικές μελέτες που αξιολογούν τα μηνύματά του στο Twitter.

Χαρακτηριστικό παράδειγμα αποτελεί η ιστορία του σκανδάλου της Cambridge Analytica, όπου σύμφωνα με άρθρο της Guardian αλλά και πολλών ακόμη ειδησεογραφικών μέσων, κατά την δεκαετία του 2010 η βρετανική εταιρεία ανάλυσης δεδομένων, συνθέτοντας ένα υποτιθέμενο «τεστ προσωπικότητας» προσέλκυσε περίπου 320.000 χρήστες, οι οποίοι μέσω των προσωπικών τους λογαριασμών στο Facebook παραχώρησαν στην εταιρεία πλήρη πρόσβαση, τόσο στα δικά τους προσωπικά δεδομένα, όσο και σε αυτά των διαδικτυακών τους φίλων. Με αυτόν τον τρόπο, η Cambridge Analytica συγκέντρωσε τα προσωπικά δεδομένα περισσότερων από 87.000.000 χρηστών του Facebook και τα χρησιμοποίησε για να υλοποιήσει μια καμπάνια για την εκλογή του Ντόναλντ Τραμπ στις εκλογές του 2016. Τα «likes» σε αναρτήσεις και σε σελίδες που ακολουθούσαν οι χρήστες, ήταν αρκετά, ώστε ο αλγόριθμος της Cambridge Analytica να καταφέρει να χτίσει ένα πανίσχυρο μοντέλο ανάλυσης συναισθήματος και προσωπικότητας. Τότε άνοιξε ο δρόμος για τις στοχευμένες διαφημιστικές εκστρατείες. Το σκάνδαλο της κατάχρησης αυτής των δεδομένων αποκάλυψε το 2018 ο Κρίστοφερ Γουάιλι, πρώην εργαζόμενος της Cambridge Analytica. ([75]), ([76])

Εικόνα 3.1.3: Ο προέδρος των Ηνωμένων Πολιτειών Donald Trump έχει χρησιμοποιήσει την επιστήμη των δεδομένων σαν μέσο χειραγώγησης των μαζών.

Καθαρισμός δεδομένων

Τα δεδομένα υποβάλλονται σε επεξεργασία και καθαρίζονται με την αφαίρεση των άχρηστων τμημάτων και των τμημάτων της γραφής που δεν έχουν νόημα σχετικά με το συναίσθημα του κειμένου. Αυτό περιλαμβάνει συντομεύσεις όπως το “I'm”, λέξεις που θεωρούνται ως μη επαρκείς πληροφορίες (π.χ. τα άρθρα), σημεία στίξης, URL, ειδικοί χαρακτήρες και κεφαλαία γράμματα.

- Κανονικοποίηση κειμένου: Το πρώτο βήμα στον καθαρισμό δεδομένων είναι η κανονικοποίηση του κειμένου, η οποία περιλαμβάνει τη μετατροπή του κειμένου σε τυπική μορφή. Αυτό μπορεί να περιλαμβάνει την κατάργηση των σημείων στίξης, τη μετατροπή κειμένου σε πεζά και την επέκταση των συστολών (π.χ. "don’t" σε "dont").

- Tokenization: Μόλις το κείμενο έχει κανονικοποιηθεί, μπορεί να γίνει διακριτικό, το οποίο περιλαμβάνει τη διάσπαση του κειμένου σε μεμονωμένες λέξεις ή φράσεις. Αυτό μπορεί να γίνει χρησιμοποιώντας μια ποικιλία τεχνικών, όπως ο διαχωρισμός του κειμένου σε κενό διάστημα ή η χρήση εργαλείων επεξεργασίας φυσικής γλώσσας.

- Διακοπή κατάργησης λέξης: Οι λέξεις διακοπής είναι συνηθισμένες λέξεις που είναι απίθανο να είναι χρήσιμες στην ανάλυση συναισθήματος, όπως "το"

ή "και". Η αφαίρεση των λέξεων τερματισμού μπορεί να βοηθήσει στη βελτίωση της ακρίβειας της ανάλυσης μειώνοντας το θόρυβο στα δεδομένα.

- Αποκοπή καταλήξεων (Stemming) και Λημματοποίηση (Lemmatization): Το Stemming και το Lemmatization είναι τεχνικές που χρησιμοποιούνται για τη μείωση των λέξεων στη βασική τους μορφή. Αυτό μπορεί να βοηθήσει στη μείωση του αριθμού των μοναδικών λέξεων στα δεδομένα, γεγονός που μπορεί να κάνει την ανάλυση πιο αποτελεσματική. Το Stemming περιλαμβάνει την αφαίρεση των επιθημάτων από τις λέξεις, ενώ η λημματοποίηση περιλαμβάνει τη μετατροπή των λέξεων στη βασική τους μορφή χρησιμοποιώντας ένα λεξικό.

- Ορθογραφικός έλεγχος και διόρθωση: Ο ορθογραφικός έλεγχος και η διόρθωση μπορούν να σας βοηθήσουν να διασφαλίσετε ότι το κείμενο είναι ακριβές και συνεπές. Αυτό μπορεί να περιλαμβάνει τη χρήση ενός εργαλείου ορθογραφικού ελέγχου ή τη μη αυτόματη εξέταση του κειμένου για σφάλματα.

- Χειρισμός άρνησης: Η άρνηση είναι μια σημαντική πτυχή της ανάλυσης συναισθήματος, καθώς μπορεί να αλλάξει την πολικότητα μιας δήλωσης. Ο χειρισμός της άρνησης περιλαμβάνει τον προσδιορισμό λέξεων άρνησης, όπως "όχι" ή "ποτέ" και την τροποποίηση της πολικότητας των συσχετισμένων λέξεων ανάλογα.

Εξαγωγή χαρακτηριστικών

Ένας αλγόριθμος μηχανικής μάθησης εξάγει αυτόματα λειτουργίες κειμένου για να εντοπίσει αρνητικά, θετικά ή ουδέτερα συναισθήματα. Οι προσεγγίσεις Μηχανικής Μάθησης που χρησιμοποιούνται περιλαμβάνουν την τεχνική bag-of-words που παρακολουθεί την εμφάνιση λέξεων σε ένα κείμενο και την πιο εξειδικευμένη τεχνική ενσωμάτωσης λέξεων βαθιάς εκμάθησης (Deep Learning) που χρησιμοποιεί νευρωνικά δίκτυα για την ανάλυση λέξεων, την συσχέτιση περίπλοκων σχέσεων μεταξύ λέξεων και φράσεων, όπως και την αναγνώριση ιδιωματισμών.

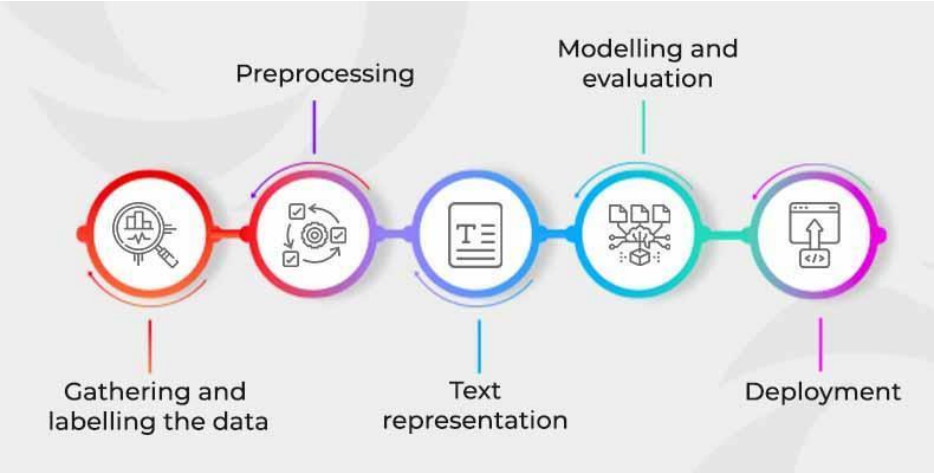
([68]), ([69])

Επιλογή μοντέλου Μηχανικής Μάθησης (ML)

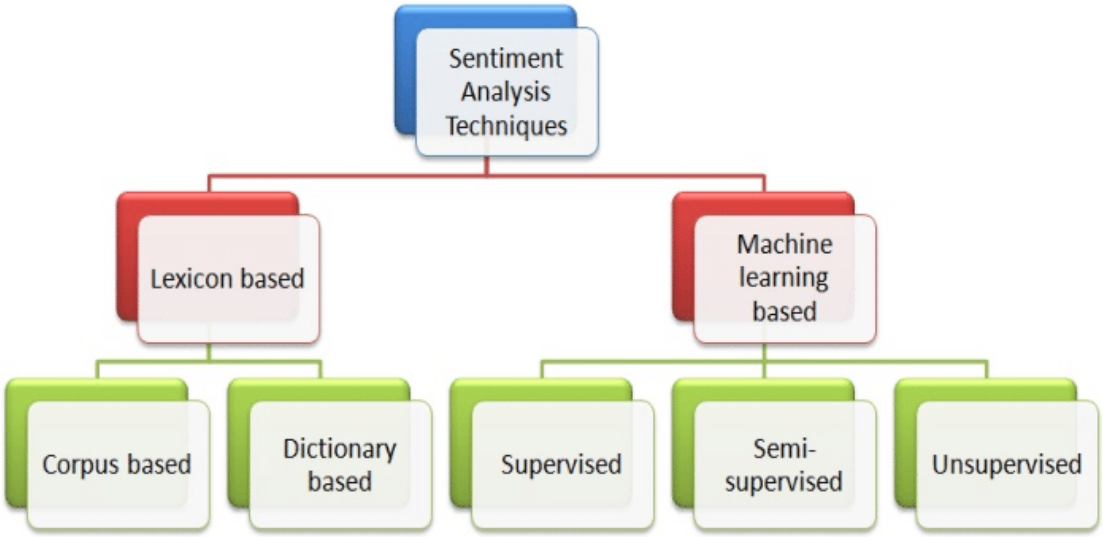
Ένα εργαλείο ανάλυσης συναισθήματος βαθμολογεί το κείμενο χρησιμοποιώντας ένα μοντέλο μηχανικής μάθησης είτε βασισμένο σε κανόνες, είτε αυτοματοποιημένο, είτε υβριδικό.

Τα συστήματα που βασίζονται σε κανόνες εκτελούν ανάλυση συναισθήματος με βάση προκαθορισμένους κανόνες που βασίζονται σε ένα λεξικό και χρησιμοποιούνται συχνά σε τομείς όπως η νομοθεσία και η ιατρική όπου απαιτείται υψηλός βαθμός ακρίβειας και ανθρώπινου ελέγχου. Τα αυτόματα συστήματα χρησιμοποιούν τεχνικές βαθιάς μάθησης για να εκπαιδευτούν από μεγάλα σύνολα δεδομένων. Ένα υβριδικό μοντέλο συνδυάζει και τις δύο προσεγγίσεις και γενικά θεωρείται ότι είναι το πιο ακριβές μοντέλο. Αυτά τα μοντέλα προσφέρουν διαφορετικές προσεγγίσεις για την ανάθεση βαθμολογιών συναισθήματος σε κομμάτια κειμένου.

Ταξινόμηση συναισθημάτων. Μόλις επιλεγεί ένα μοντέλο και χρησιμοποιηθεί για την ανάλυση του κειμένου, αποδίδει μια βαθμολογία συναισθήματος στο κείμενο, η οποία μεταφράζεται σε θετικό, αρνητικό ή ουδέτερο συναίσθημα.
([70]), ([71]), ([74])



Εικόνα 3.1.4: Βήματα εφαρμογής Μηχανικής Μάθησης.



Εικόνα 3.1.5: Τεχνικές εφαρμογής Μηχανικής Μάθησης.

Μέθοδοι και Αλγόριθμοι Μηχανικής Μάθησης για την ανάλυση συναισθήματος

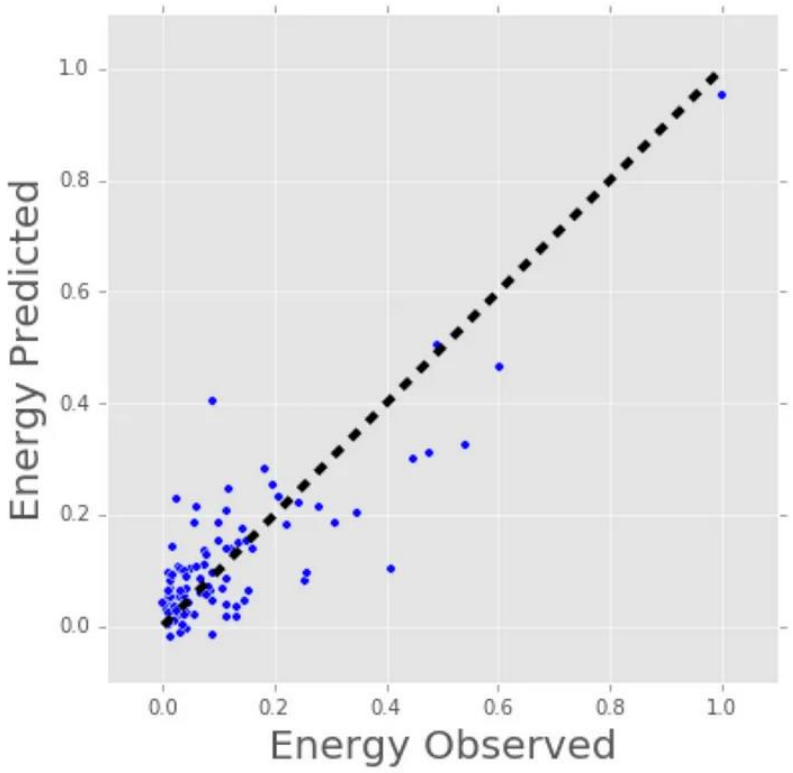
Παλινδρόμηση (Regression)

Η μέθοδος της παλινδρόμησης ανήκει στην κατηγορία της εποπτευόμενης Μηχανικής Μάθησης (Supervised ML). Βοηθά στην πρόβλεψη ή την επεξήγηση μιας συγκεκριμένης αριθμητικής τιμής με βάση ένα σύνολο προηγούμενων δεδομένων, για παράδειγμα στην πρόβλεψη της τιμής ενός ακινήτου με βάση προηγούμενα δεδομένα τιμολόγησης για παρόμοια ακίνητα.

Η απλούστερη μέθοδος είναι η γραμμική παλινδρόμηση όπου χρησιμοποιούμε τη μαθηματική εξίσωση (y = m * x + b) για να μοντελοποιήσουμε ένα σύνολο δεδομένων. Εκπαιδεύουμε ένα μοντέλο γραμμικής παλινδρόμησης με πολλά ζεύγη δεδομένων (x, y) υπολογίζοντας τη θέση και την κλίση μιας γραμμής που ελαχιστοποιεί τη συνολική απόσταση μεταξύ όλων των σημείων δεδομένων και της γραμμής. Με άλλα λόγια, υπολογίζουμε την κλίση (m) και την τομή y (b) για μια γραμμή που προσεγγίζει καλύτερα τις παρατηρήσεις στα δεδομένα.

Ένα παράδειγμα γραμμικής παλινδρόμησης είναι η πρόβλεψη της κατανάλωσης ενέργειας (σε kWh) ορισμένων κτιρίων, συγκεντρώνοντας την ηλικία του κτιρίου, τον αριθμό των ορόφων, τα τετραγωνικά μέτρα και τον αριθμό του εντοιχισμένου εξοπλισμού. Επειδή υπάρχουν περισσότερες από μία είσοδοι (ηλικία, τετραγωνικά μέτρα, κ.λπ.…), χρησιμοποιείται γραμμική παλινδρόμηση πολλαπλών μεταβλητών. Η αρχή είναι η ίδια με μια απλή γραμμική παλινδρόμηση ένα προς ένα, αλλά σε αυτήν την περίπτωση η «γραμμή» που δημιουργείται εμφανίζεται σε πολυδιάστατο χώρο με βάση τον αριθμό των μεταβλητών.

Το παρακάτω διάγραμμα δείχνει πόσο καλά ταιριάζει το μοντέλο γραμμικής παλινδρόμησης στην πραγματική κατανάλωση ενέργειας του κτιρίου.



Εικόνα 3.1.6: Γράφημα μοντέλου Μηχανικής Μάθησης με την χρήση της μεθόδου Παλινδρόμησης.

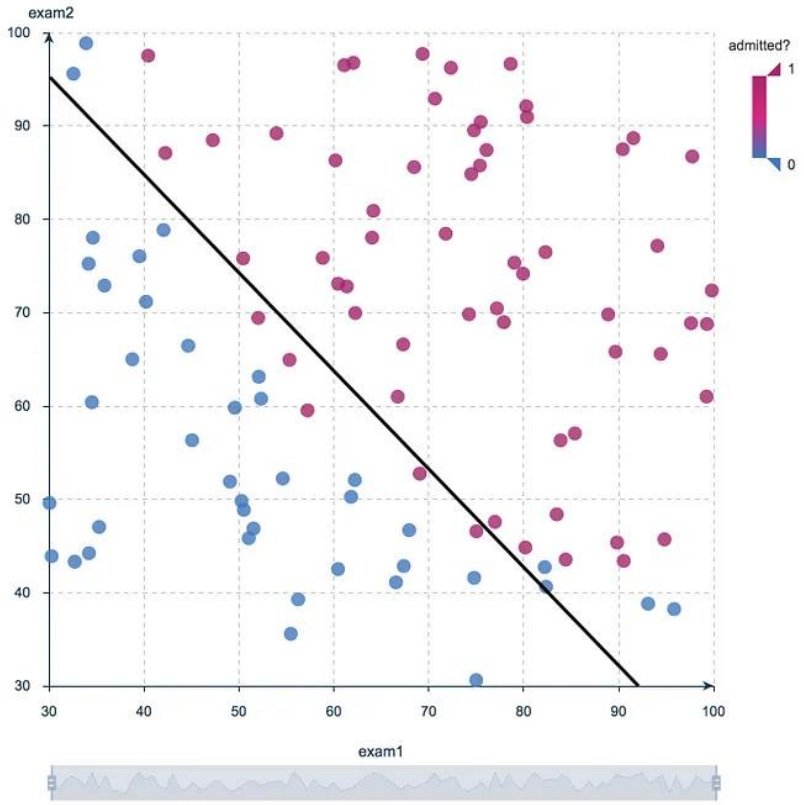
Επίσης μπορεί να χρησιμοποιηθεί γραμμική παλινδρόμηση για να υπολογιστεί το βάρος κάθε παράγοντα που συμβάλλει στην τελική πρόβλεψη της ενέργειας που καταναλώνεται. Δηλαδή, σε μια φόρμουλα, μπορεί να προσδιοριστεί εάν η ηλικία, το μέγεθος ή το ύψος είναι πιο σημαντικά.

Οι τεχνικές παλινδρόμησης κυμαίνονται από απλές (όπως η γραμμική παλινδρόμηση) εώς πολύπλοκες (όπως η κανονικοποιημένη γραμμική παλινδρόμηση, η πολυωνυμική παλινδρόμηση, τα δέντρα απόφασης, και τα νευρωνικά δίκτυα).

Ταξινόμηση ή Κατηγοριοποίηση

Μια άλλη κατηγορία εποπτευόμενης Μηχανικής Μάθησης, η μέθοδος ταξινόμησης προβλέπει ή εξηγεί μια τιμή κλάσης. Για παράδειγμα, μπορεί να βοηθήσει στην πρόβλεψη εάν ένας διαδικτυακός πελάτης θα αγοράσει ένα προϊόν ή όχι. Η έξοδος μπορεί να είναι "ναι" ή "όχι". Μία μέθοδος ταξινόμησης θα μπορούσε να βοηθήσει να αξιολογηθεί εάν μια εικόνα περιέχει ένα αυτοκίνητο ή ένα φορτηγό. Σε αυτήν την περίπτωση, η έξοδος θα είναι 3 διαφορετικές τιμές: 1) η εικόνα περιέχει ένα αυτοκίνητο, 2) η εικόνα περιέχει ένα φορτηγό ή 3) η εικόνα δεν περιέχει ούτε αυτοκίνητο ούτε φορτηγό. Ο απλούστερος αλγόριθμος ταξινόμησης είναι η λογιστική παλινδρόμηση η οποία όμως δεν είναι μέθοδος παλινδρόμησης. Η λογιστική παλινδρόμηση εκτιμά την πιθανότητα εμφάνισης ενός γεγονότος με βάση μία ή περισσότερες εισαγωγές. Για παράδειγμα, μια λογιστική παλινδρόμηση μπορεί να λάβει ως εισόδους δύο βαθμολογίες εξετάσεων για έναν μαθητή προκειμένου να εκτιμηθεί η πιθανότητα ο μαθητής να γίνει δεκτός σε μια συγκεκριμένη σχολή. Επειδή η εκτίμηση είναι μια πιθανότητα, η έξοδος είναι ένας αριθμός μεταξύ 0 και 1, όπου το 1 αντιπροσωπεύει πλήρη βεβαιότητα. Για τον μαθητή, εάν η εκτιμώμενη πιθανότητα είναι μεγαλύτερη από 0,5, τότε προβλέπεται ότι θα γίνει δεκτός. Εάν η εκτιμώμενη πιθανότητα είναι μικρότερη από 0,5, προβλέπεται ότι θα απορριφθεί.

Το παρακάτω διάγραμμα απεικονίζει τις βαθμολογίες των προηγούμενων μαθητών μαζί με το αν έγιναν δεκτοί. Η λογιστική παλινδρόμηση μας επιτρέπει να σχεδιάσουμε μια γραμμή που αντιπροσωπεύει το όριο απόφασης.

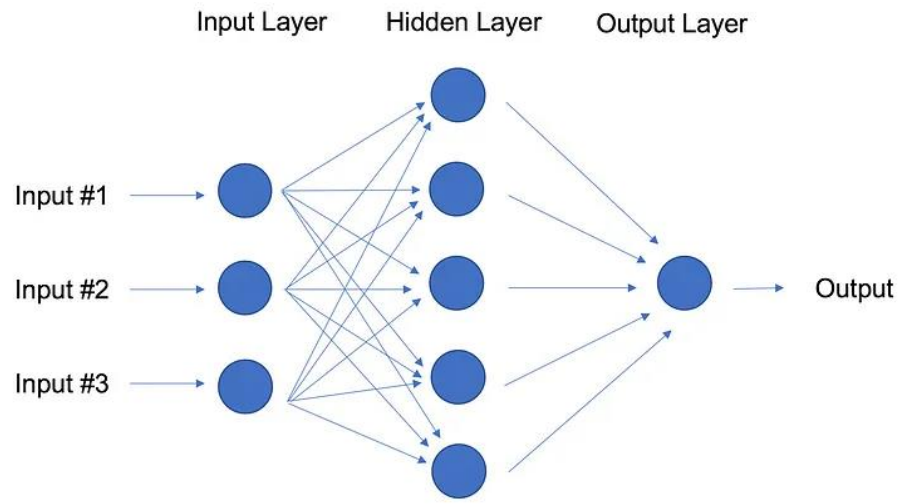


Εικόνα 3.1.7: Γράφημα μοντέλου Μηχανικής Μάθησης με την χρήση της μεθόδου Ταξινόμησης / Κατηγοριοποίησης.

Η λογιστική παλινδρόμηση θεωρείται ως το απλούστερο μοντέλο ταξινόμησης.

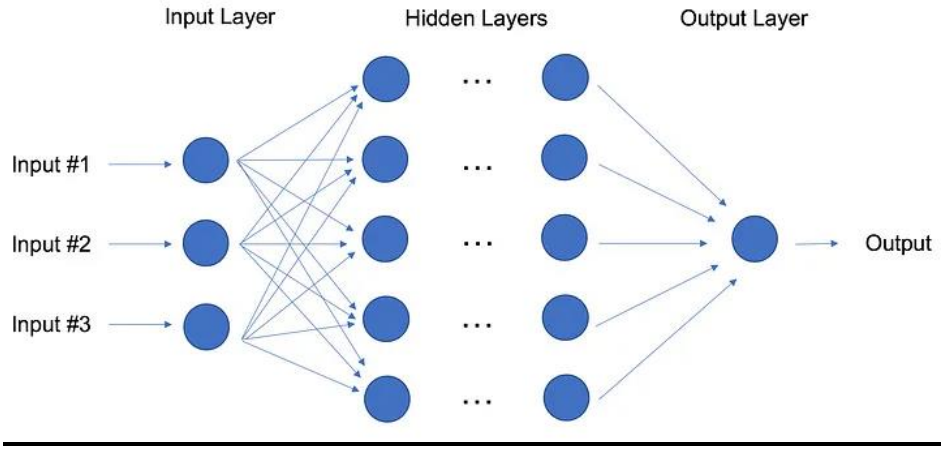
Νευρωνικά δίκτυα και βαθιά μάθηση

Σε αντίθεση με την γραμμική και λογιστική παλινδρόμηση που θεωρούνται γραμμικά μοντέλα, ο στόχος των νευρωνικών δικτύων είναι να συλλάβουν μη γραμμικά μοτίβα στα δεδομένα προσθέτοντας στρώματα παραμέτρων στο μοντέλο. Στην παρακάτω εικόνα, το απλό νευρωνικό δίκτυο έχει τρεις εισόδους. Ένα αρχικό στρώμα τιμών εισόδων, ένα κρυφό στρώμα με πέντε παραμέτρους και ένα στρώμα εξόδου.



Εικόνα 3.1.8: Μοντέλο απλού Νευρωνικού Δικτύου.

Η δομή των νευρωνικών δικτύων είναι αρκετά ευέλικτη για να δημιουργήσει γραμμική και λογιστική παλινδρόμηση. Ο όρος Βαθιά Μάθηση (Deep Learning) προέρχεται από ένα νευρωνικό δίκτυο με πολλά κρυφά στρώματα και μεγάλη ποικιλία αρχιτεκτονικών.



Εικόνα 3.1.9: Μοντέλο Βαθιάς Μάθησης.

Για μεγαλύτερη απόδοση, οι τεχνικές βαθιάς εκμάθησης απαιτούν πολλά δεδομένα — και πολλή υπολογιστική ισχύ, καθώς η μέθοδος αυτοσυντονίζει πολλές παραμέτρους μέσα σε τεράστιες αρχιτεκτονικές. Αυτό συνεπάγεται στο γεγονός πως οι επαγγελματίες βαθιάς μάθησης συχνά χρειάζονται ιδιαίτερα ισχυρούς υπολογιστές ενισχυμένους με δυνατές GPU (μονάδες γραφικής επεξεργασίας). Οι τεχνικές βαθιάς

μάθησης είναι εξαιρετικά επιτυχημένες στους τομείς της όρασης (ταξινόμηση εικόνων), του κειμένου, του ήχου και του βίντεο. Τα πιο κοινά πακέτα λογισμικού για βαθιά εκμάθηση είναι το Tensorflow και το PyTorch.

Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing - NLP)

Η Επεξεργασία Φυσικής Γλώσσας δεν θεωρείται τόσο μια μέθοδος μηχανικής μάθησης αλλά περισσότερο μια ευρέως χρησιμοποιούμενη τεχνική για την προετοιμασία κειμένου για μηχανική μάθηση. Πολλά από τα έγγραφα κειμένου διαφόρων μορφών είναι συχνά είναι γεμάτα τυπογραφικά λάθη, χαρακτήρες που λείπουν και άλλες λέξεις που θα έπρεπε να φιλτραριστούν. Αυτή τη στιγμή, το πιο δημοφιλές πακέτο για την επεξεργασία κειμένου είναι το NLTK (Natural Language ToolKit), που δημιουργήθηκε από ερευνητές στο Stanford. Ο απλούστερος τρόπος για την αντιστοίχιση κειμένου σε αριθμητική αναπαράσταση είναι ο υπολογισμός της συχνότητας κάθε λέξης μέσα σε κάθε έγγραφο κειμένου. Σαν ένας πίνακα ακεραίων όπου κάθε σειρά αντιπροσωπεύει ένα έγγραφο κειμένου και κάθε στήλη αντιπροσωπεύει μια λέξη. Αυτή η αναπαράσταση τύπου πίνακα των συχνοτήτων λέξεων ονομάζεται Πίνακας Όρου Συχνότητας (Term Frequency Matrix - TFM). Με τον πίνακα αυτό, μπορεί να δημιουργηθεί μια άλλη αναπαράσταση πίνακα ενός εγγράφου κειμένου διαιρώντας κάθε καταχώρηση στον πίνακα με το βάρος του πόσο σημαντική είναι κάθε λέξη σε ολόκληρο το σύνολο των εγγράφων. Αυτή η μέθοδος, ονομάζεται Όρος Συχνότητας Αντίστροφης Συχνότητας Εγγράφου (Term Frequency Inverse Document Frequency - TFIDF) και συνήθως λειτουργεί καλύτερα για εργασίες μηχανικής εκμάθησης.

Naive Bayes

Ο Naive Bayes είναι ο πιο απλός αλγόριθμος που μπορεί να εφαρμοστεί στα δεδομένα. Όπως υποδηλώνει το όνομα (“naive” = “αφελής”), ο αλγόριθμος αυτός κάνει μια υπόθεση, καθώς όλες οι μεταβλητές στο σύνολο δεδομένων είναι "Αφελείς", δηλαδή δεν συσχετίζονται μεταξύ τους. Ο Naive Bayes είναι ένας πολύ δημοφιλής αλγόριθμος ταξινόμησης που χρησιμοποιείται κυρίως για τη λήψη της βασικής ακρίβειας του συνόλου δεδομένων.

Το θεώρημα Bayes παρέχει έναν τρόπο υπολογισμού της μεταγενέστερης πιθανότητας, P(c|x), από τα P(c), P(x) και P(x|c). Ο ταξινομητής Naive Bayes

υποθέτει ότι η επίδραση της τιμής ενός προγνωστικού δείκτη (x) σε μια δεδομένη κατηγορία (c) είναι ανεξάρτητη από τις τιμές άλλων προβλέψεων. Αυτή η υπόθεση ονομάζεται ανεξαρτησία υπό όρους τάξης.

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Εικόνα 3.1.10: Φόρμουλα Θεωρήματος Bayes.

- P(c|x) είναι η οπίσθια πιθανότητα της κλάσης (στόχος) δεδομένου προγνωστικού (χαρακτηριστικό).
- Το P(c) είναι η προηγούμενη πιθανότητα της κλάσης.
- P(x|c) είναι η πιθανότητα που είναι η πιθανότητα μιας προβλεπόμενης κλάσης.
- Το P(x) είναι η προηγούμενη πιθανότητα του προγνωστικού.

Για την εκπαίδευση ενός μοντέλου Naive Bayes, υπολογίζουμε πρώτα τις προηγούμενες πιθανότητες κάθε ετικέτας κλάσης με βάση τη συχνότητα εμφάνισης στα δεδομένα εκπαίδευσης. Στη συνέχεια υπολογίζουμε την πιθανότητα κάθε χαρακτηριστικού για κάθε ετικέτα κλάσης, η οποία είναι η πιθανότητα παρατήρησης αυτού του χαρακτηριστικού με δεδομένη την ετικέτα κλάσης. Τέλος, χρησιμοποιούμε το θεώρημα του Bayes για να υπολογίσουμε την μεταγενέστερη πιθανότητα κάθε ετικέτας κλάσης για μια νέα είσοδο με βάση τα χαρακτηριστικά της και καταχωρούμε την είσοδο στην κλάση με την υψηλότερη οπίσθια πιθανότητα.

Ένα από τα πλεονεκτήματα του Naive Bayes είναι η απλότητα και η αποτελεσματικότητά του στην εκπαίδευση και την πρόβλεψη, ειδικά για δεδομένα

υψηλών διαστάσεων όπως το κείμενο. Μπορεί επίσης να χειριστεί δεδομένα που λείπουν και λειτουργεί καλά με μικρά σετ προπόνησης. Ωστόσο, η αφελής υπόθεση της ανεξαρτησίας των χαρακτηριστικών μπορεί να μην ισχύει σε όλες τις περιπτώσεις και ο αλγόριθμος μπορεί να μην έχει καλή απόδοση εάν τα χαρακτηριστικά έχουν υψηλή συσχέτιση.

Αλγόριθμος Κ-Κοντινών Γειτόνων (K-Nearest Neighbors - K-NN)

Ο K-Nearest Neighbors είναι ένας μη παραμετρικός αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται για εργασίες ταξινόμησης και παλινδρόμησης. Ο αλγόριθμος λειτουργεί βρίσκοντας τα Κ πιο κοντινά σημεία δεδομένων εκπαίδευσης (δηλαδή, τους πλησιέστερους γείτονες) σε ένα νέο σημείο δεδομένων χρησιμοποιώντας την πλειοψηφία των ετικετών κλάσεων των γειτόνων Κ για ταξινόμηση ή τον μέσο όρο των τιμών τους για παλινδρόμηση.

Στην περίπτωση ταξινόμησης, ο K-NN εκχωρεί την ετικέτα κλάσης που εμφανίζεται πιο συχνά μεταξύ των Κ πλησιέστερων γειτόνων στο νέο σημείο δεδομένων. Η τιμή του Κ, η οποία αντιπροσωπεύει τον αριθμό των γειτόνων που πρέπει να ληφθούν υπόψη, επιλέγεται συνήθως με διασταυρωμένη επικύρωση (cross-validation) ή άλλες μεθόδους επιλογής μοντέλου. Μια μικρή τιμή του Κ οδηγεί σε ένα πιο ευέλικτο όριο απόφασης, ενώ μια μεγαλύτερη τιμή του Κ οδηγεί σε ένα πιο ομαλό όριο απόφασης. Στην περίπτωση παλινδρόμησης, ο K-NN προβλέπει την τιμή του νέου σημείου δεδομένων ως τον μέσο όρο των τιμών των Κ πλησιέστερων γειτόνων. Αυτή η μέθοδος μπορεί να είναι χρήσιμη για την εκτίμηση μιας συνεχούς μεταβλητής ή την πρόβλεψη της τιμής ενός σπιτιού με βάση τις τιμές παρόμοιων σπιτιών στη γειτονιά. Ο K-NN είναι ένας τεμπέλης (lazy) αλγόριθμος εκμάθησης, που σημαίνει ότι δεν μαθαίνει ρητά ένα μοντέλο από τα δεδομένα εκπαίδευσης, αλλά αντ' αυτού απομνημονεύει τα δεδομένα εκπαίδευσης για να κάνει προβλέψεις κατά το χρόνο εκτέλεσης. Αυτό καθιστά τη φάση εκπαίδευσης υπολογιστικά φθηνή, αλλά η φάση πρόβλεψης μπορεί να είναι αργή, ειδικά για μεγάλα σύνολα δεδομένων. Ο K-NN απαιτεί μια μέτρηση απόστασης για τη μέτρηση της ομοιότητας μεταξύ σημείων δεδομένων, η οποία υπολογίζεται με την χρήση του τύπου της Ευκλείδειας Απόστασης:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

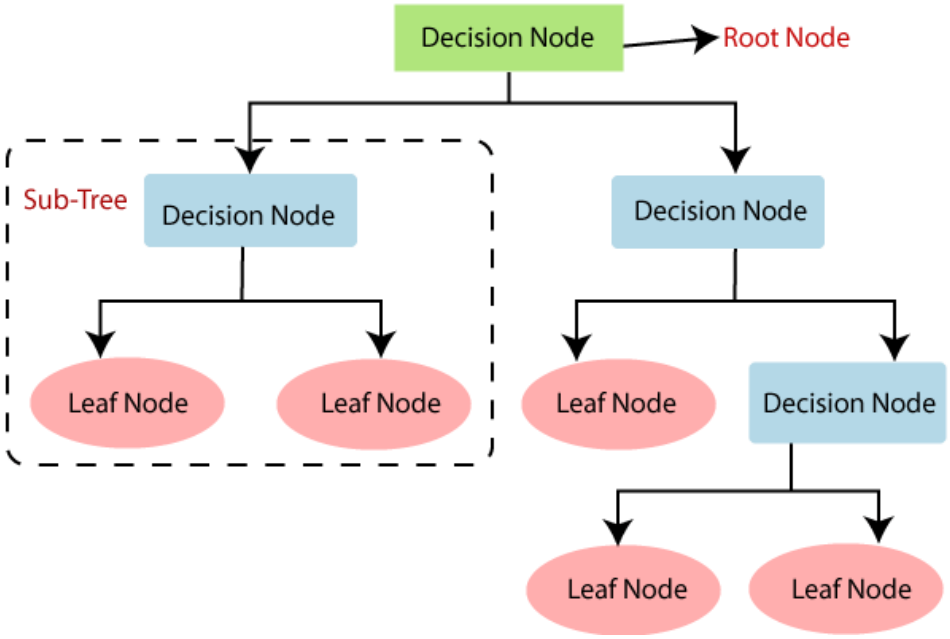
Εικόνα 3.1.11: Φόρμουλα Ευκλείδειας Απόστασης.

Μπορεί όμως να μην είναι πάντα κατάλληλη για όλους τους τύπους δεδομένων ή τους τομείς. ([72])

Decision Tree

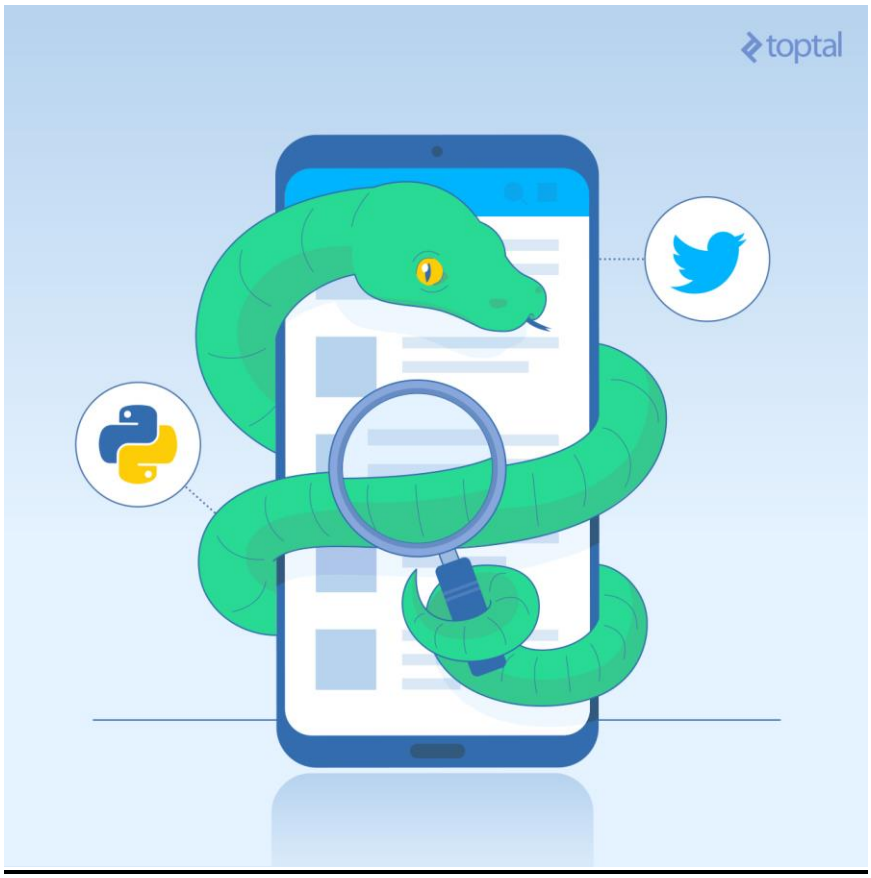
Ο αλγόριθμος Decision Tree είναι ένας εποπτευόμενος αλγόριθμος μηχανικής μάθησης που χρησιμοποιείται τόσο για εργασίες ταξινόμησης όσο και για εργασίες παλινδρόμησης. Ο αλγόριθμος λειτουργεί διαχωρίζοντας αναδρομικά τα δεδομένα εκπαίδευσης σε υποσύνολα με βάση τις τιμές των χαρακτηριστικών εισόδου, με στόχο τη μεγιστοποίηση της καθαρότητας των υποσυνόλων σε σχέση με την μεταβλητή στόχο. Στην περίπτωση της ταξινόμησης, κάθε εσωτερικός κόμβος του δέντρου αντιπροσωπεύει μια απόφαση που βασίζεται σε ένα από τα χαρακτηριστικά εισόδου, ενώ κάθε κόμβος φύλλου αντιπροσωπεύει μια ετικέτα κλάσης. Η απόφαση λαμβάνεται ακολουθώντας τη διαδρομή από τη ρίζα του δέντρου σε έναν κόμβο φύλλου που αντιστοιχεί στη δεδομένη είσοδο. Το όριο απόφασης αντιπροσωπεύεται από το σύνολο κανόνων απόφασης που ορίζουν τις διαδρομές από τη ρίζα σε κάθε κόμβο φύλλου. Στην περίπτωση της παλινδρόμησης, ο αλγόριθμος λειτουργεί με παρόμοιο τρόπο, αλλά προβλέπει μια συνεχή τιμή αντί για μια ετικέτα κλάσης. Κάθε κόμβος φύλλου αντιπροσωπεύει τον μέσο όρο της μεταβλητής στόχου για το υποσύνολο δεδομένων εκπαίδευσης που έφτασε σε αυτόν τον κόμβο. Ο αλγόριθμος του δέντρου αποφάσεων έχει το πλεονέκτημα ότι είναι εύκολο να ερμηνευτεί και να εξηγηθεί, καθιστώντας τον μια δημοφιλή επιλογή σε πολλές εφαρμογές. Επιπλέον, τα δέντρα αποφάσεων μπορούν να χειριστούν τόσο κατηγορίες όσο και αριθμητικά χαρακτηριστικά, καθώς και δεδομένα που λείπουν. Ωστόσο, τα δέντρα απόφασης

μπορεί να υποφέρουν από υπερβολική προσαρμογή, ειδικά όταν το δέντρο είναι πολύ περίπλοκο ή όταν υπάρχουν θορυβώδη ή άσχετα δεδομένα. ([73])



Εικόνα 3.1.6: Γράφημα Λειτουργίας του αλγορίθμου Decision Tree (Δέντρου Απόφασης).

3.2 ΕΞΟΡΥΞΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΕΦΑΡΜΟΓΗ ΤΗΣ ΑΝΑΛΥΣΗΣ ΣΥΝΑΙΣΘΗΜΑΤΟΣ ΜΕ ΧΡΗΣΗ ΤΟΥ APACHE SPARK



Εικόνα 3.2: Απεικόνιση της χρήσης γλώσσας Python με δεδομένα από την κοινωνική πλατφόρμα Twitter.

Σε αυτήν την ενότητα, θα γίνει παρουσίαση ενός συστήματος το οποίο αποτελείται από δύο κύρια μέρη. Κατά το πρώτο μέρος, πραγματοποιείται εξόρυξη δεδομένων κειμένου (text) μέσω της πλατφόρμας κοινωνικής δικτύωσης Twitter (τα δεδομένα αυτά αποτελούνται από αναρτήσεις χρηστών χρησιμοποιώντας παραμέτρους ως φίλτρα αναζήτησης). Για τις ανάγκες της εργασίας αυτής, χρησιμοποιήθηκε dataset μεγέθους 3GB το οποίο συλλέχθηκε σε χρονικό διάστημα των σαράντα (40) λεπτών.
Κατά το δεύτερο μέρος, τα συλλεχθέντα δεδομένα υπόκεινται σε διαδικασίες καθαρισμού και εξομάλυνσης και στην συνέχεια αναλύονται έτσι ώστε να γίνει

εξαγωγή του κυρίαρχου συναισθήματος μέσα από το κάθε κάθε κείμενο. Η διαδικασία αυτή διεκπεραιώθηκε εντός του χρονικού διαστήματος των έξι (6) λεπτών. Το προαναφερθέν σύστημα περιλαμβάνει δύο προγράμματα γραμμένα σε γλώσσα Python και ανεπτυγμένα για χρήση των δυνατοτήτων του λογισμικου Spark. Σύμφωνα με τις πληροφορίες του Κεφαλαίου 1, τα προγράμματα αυτά δύναται να τρέξουν τόσο σε local mode, όσο και σε cluster mode με cluster manager το YARN. Πιο αναλυτικά:

3.2.1 pyspark_scrapper.py

Όγκος δεδομένων: 3GB	Χρόνος: 40’	Τύποι Δεδομένων: αδόμητα - ημιδομημένα

Στην αρχή του αλγορίθμου γίνεται εισαγωγή των απαραίτητων βιβλιοθηκών της Python, έτσι ώστε να μπορούν να εκτελεστούν οι κύριες λειτουργίες του προγράμματος.
Οι βιβλιοθήκες που θα χρησιμοποιηθούν εδώ είναι οι εξής:

os

Η βιβλιοθήκη os παρέχει την δυνατότητα αλληλεπίδρασης με το λειτουργικό σύστημα. Προσφέρει μια σειρά από λειτουργίες για την εκτέλεση διαφόρων εργασιών που σχετίζονται με αυτό, όπως λειτουργίες αρχείων και καταλόγου, διαχείριση διεργασιών, μεταβλητές περιβάλλοντος και άλλα. ([78])

pyspark.sql

Η βιβλιοθήκη pyspark παρέχει στο Apache Spark ένα API υψηλού επιπέδου για εργασία με δεδομένα μεγάλης κλίμακας και επιτρέπει την ανάπτυξη εφαρμογών Spark με χρήση της γλώσσας Python. Η επέκταση pyspark.sql παρέχει την δυνατότητα εκτέλεσης ερωτημάτων (queries) τύπου SQL, χειρισμού δεδομένων και ανάλυσης σε σύνολα δεδομένων μεγάλης κλίμακας. ([79])

snscrape

Η βιβλιοθήκη Snscrape απλοποιεί τη διαδικασία εξόρυξης δεδομένων από πλατφόρμες κοινωνικής δικτύωσης παρέχοντας μια ενοποιημένη διεπαφή. Επιτρέπει

την συλλογή αναρτήσεων, σχολίων και άλλων πληροφοριών. Χρησιμοποιείται συχνά για την ανάλυση συναισθήματος, την παρακολούθηση τάσεων και άλλων εφαρμογών που απαιτούν ανάλυση δεδομένων από διάφορες πλατφόρμες κοινωνικών μέσων. ([80])

Εφόσον φορτωθούν στο σύστημα οι βιβλιοθήκες, δηλώνονται οι συναρτήσεις που θα χρησιμοποιηθούν στον κυρίως κώδικα (main). Η συνάρτηση collect() αναλαμβάνει την εξόρυξη των δεδομένων μέσω του Twitter API και την αποθήκευσή τους σε αρχείο με την μορφή csv.

Η συνάρτηση file_size() επιστρέφει την τιμή του μεγέθους από το αρχείο csv που δημιουργήθηκε σε megabytes.

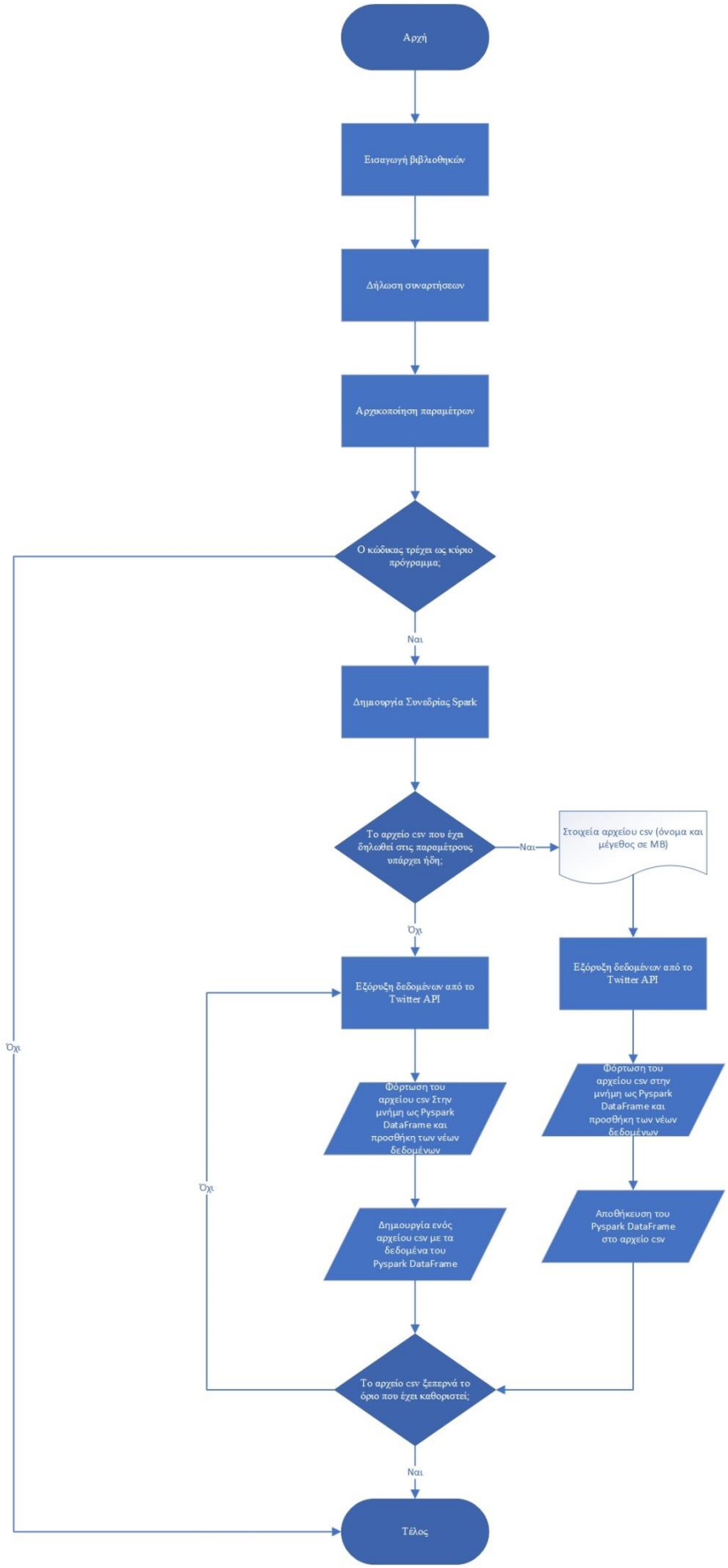
Στην συνέχεια αρχικοποιούνται οι καθολικές (global) μεταβλητές των οποίων τις τιμές μπορεί να καθορίσει ο χρήστης ανάλογα με τις απαιτήσεις του. Κάποιες βασικές παράμετροι είναι η μεταβλητή *keyword* που ορίζει το αντικείμενο της αναζήτησης στο Twitter, η *filename* που καθορίζει το όνομα του αρχείου csv που θα δημιουργηθεί ή θα φορτωθεί στην μνήμη και η *limit* που αναφέρεται στο όριο που θέλουμε να φτάσει το αρχείο και η *maxTweets*. Η μεταβλητή αυτή θέτει το μέγιστο όριο των tweets που θα συλλεχθούν ανά απόπειρα εξόρυξης. Η default τιμή της είναι η 5000 διότι η διεπαφή του Twitter θέτει ορισμένους περιορισμούς ως προς την εξυπηρέτηση των αιτημάτων των εφαρμογών.

Προχωράμε στο κυρίως (main) πρόγραμμα[3] όπου δημιουργείται η Συνεδρία Spark (Spark Session). Πλέον η εφαρμογή μπορεί να κάνει χρήση των εντολών Pyspark που βρίσκονται στην συνάρτηση collect(). Η συνάρτηση αυτή όπως αναφέρθηκε προηγουμένως, είναι υπεύθυνη για να δημιουργήσει ή να φορτώσει στην μνήμη το αρχείο csv στο οποίο θα αποθηκευτούν τα δεδομένα εξόρυξης που κρατώνται σε μία προσωρινή λίστα, αφού πρώτα η λίστα αυτή μετατραπεί σε Pyspark Dataset (βλ. Κεφάλαιο 1 - «Dataset» στη σελίδα 31).

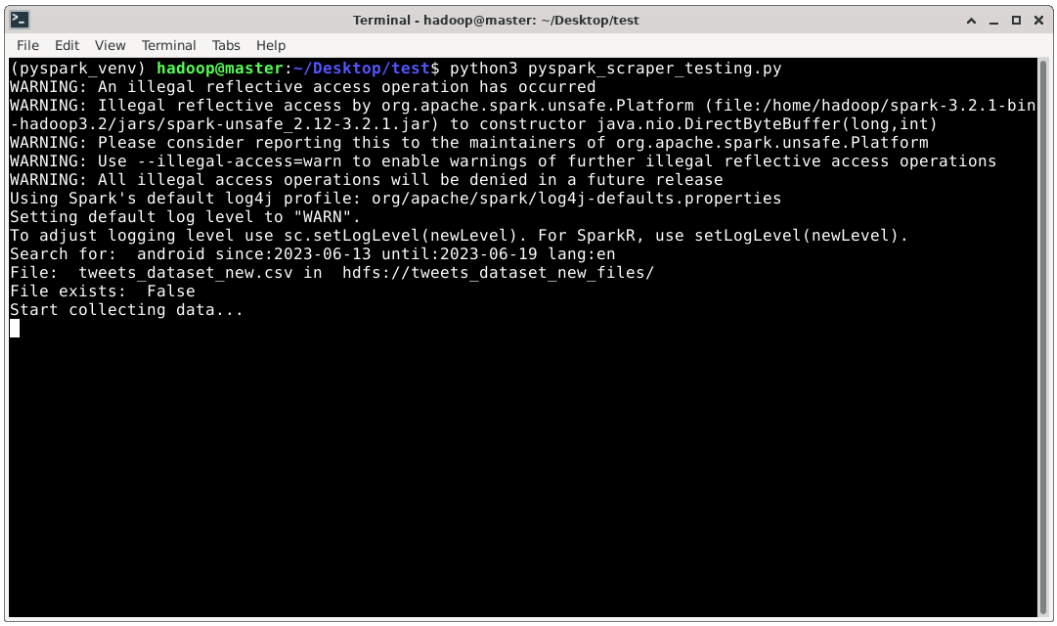
[3] Οι εφαρμογές PySpark, όπως και οι εφαρμογές Python, έχουν ένα κύριο σημείο εισόδου. Η συνθήκη if __name__ == "__main__" χρησιμοποιείται για να προσδιορίσει το σημείο εισόδου και να διασφαλίσει ότι η κύρια λογική εκτελείται μόνο όταν η εφαρμογή εκτελείται απευθείας.
Η Συνεδρία Spark (SparkSession) δημιουργείται συνήθως μέσα στο μπλοκ if __name__ == "__main__" για να διασφαλιστεί ότι αρχικοποιείται μόνο όταν η εφαρμογή εκτελείται απευθείας. Αυτό βοηθά στην αποφυγή πολλαπλών δημιουργιών SparkSession κατά την εισαγωγή της λειτουργικής μονάδας ως βιβλιοθήκης, η οποία μπορεί να οδηγήσει σε προβλήματα απόδοσης.

Κάνοντας χρήση του Pyspark API αρχικοποιούνται επιπλέον παράμετροι που καθορίζουν τα μονοπάτια αρχείων στο σύστημα HDFS («Hadoop Distributed File System» – Βλ. Κεφάλαιο 1 στη σελίδα 36, στη σελίδα 40, στη σελίδα 43 και Κεφάλαιο 2 σελ. στη σελίδα 126) για να είναι σε θέση το πρόγραμμα να γνωρίζει που θα πραγματοποιήσει την όποια αναζήτηση. Σε αυτό το σημείο πρέπει να αναφερθεί ότι το σετ δεδομένων που δημιουργείται από την εφαρμογή μπορεί να αποθηκευτεί και στο τοπικό σύστημα αρχείων του κόμβου στον οποίο θα τρέξει η εφαρμογή σε local mode. Με αυτόν τον τρόπο όμως δεν γίνεται αξιοποίηση των δυνατοτήτων παράλληλης επεξεργασίας δεδομένων που προσφέρει το Spark όπως επίσης και του κατανεμημένου συστήματος αποθήκευσης του Hadoop. Λεπτομέρειες σχετικά με τις λειτουργίες εκτέλεσης του Spark αναφέρονται στα Κεφάλαια 1 και 2.

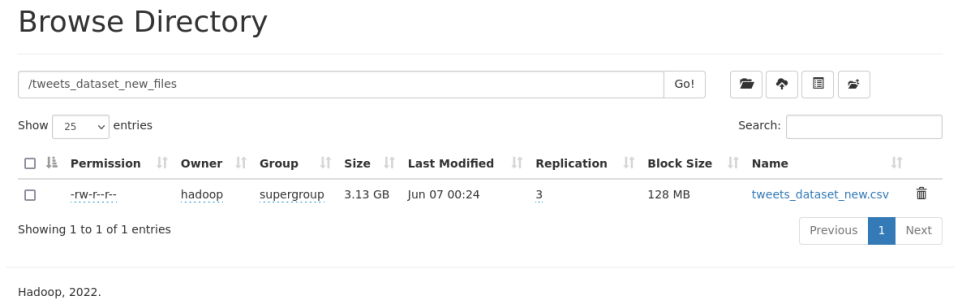
Στο τελευταίο μέρος του κώδικα βλέπουμε έναν βρόχο while, όπου εξετάζεται το μέγεθος αρχείου csv. Εάν αυτό ξεπερνά το όριο που έχουμε θέσει στην μεταβλητή *limit*, τότε το πρόγραμμα τερματίζει. Εάν όχι εκτελείται ξανά η συνάρτηση collect() για να συλλέξει τα επόμενα 5000 tweets και να τα γράψει στο αρχείο csv που έχει οριστεί.


Αρχή
Εισαγωγή βιβλιοθηκών
Δήλωση συναρτήσεων
Αρχικοποίηση παραμέτρων
Ο κώδικας τρέχει ως κύριο πρόγραμμα;
Ναι
Όχι
Δημιουργία Συνεδρίας Spark
Το αρχείο csv που έχει δηλωθεί στις παραμέτρους υπάρχει ήδη;
Ναι
Όχι
Στοιχεία αρχείου csv (όνομα και μέγεθος σε MB)
Εξόρυξη δεδομένων από το Twitter API
Φόρτωση του αρχείου csv στην μνήμη ως Pyspark DataFrame και προσθήκη των νέων δεδομένων
Αποθήκευση του Pyspark DataFrame στο αρχείο csv
Εξόρυξη δεδομένων από το Twitter API
Φόρτωση του αρχείου csv Στην μνήμη ως Pyspark DataFrame και προσθήκη των νέων δεδομένων
Δημιουργία ενός αρχείου csv με τα δεδομένα του Pyspark DataFrame
Όχι
Το αρχείο csv ξεπερνά το όριο που έχει καθοριστεί;
Ναι
Τέλος

Εικόνα 3.3: Το διάγραμμα ροής του αλγορίθμου "pyspark scraper"



Εικόνα 3.4: Εκτέλεση της εφαρμογής "pyspark scraper".



Εικόνα 3.5: Στοιχεία του dataset που συλλέχθηκε στο κατανεμημένο σύστημα αρχείων του Hadoop.

3.2.2 sentiment_analysis.py

Όγκος δεδομένων: 3GB	Χρόνος: 6’	Τύποι Δεδομένων: ημιδομημένα - δομημένα

Στην αρχή του αλγορίθμου γίνεται εισαγωγή των απαραίτητων βιβλιοθηκών της Python, έτσι ώστε να μπορούν να εκτελεστούν οι κύριες λειτουργίες του προγράμματος.

Οι βιβλιοθήκες που θα χρησιμοποιηθούν εδώ είναι οι εξής:

os

(βλ. παραπάνω)

pyspark.pandas

Η pyspark.pandas είναι μια μονάδα (module) επέκτασης που παρέχεται από το PySpark ως ένα παρόμοιο API με αυτό του Pandas αλλά για εργασία με κατανεμημένα δεδομένα μεγάλης κλίμακας σε αντίθεση με το Pandas που χρησιμοποιεί την μνήμη σε ένα μόνο μηχάνημα.
Τα Pandas και Pyspark Pandas παρέχουν ένα σύνολο λειτουργιών και μεθόδων για του χειρισμό δεδομένων, συμπεριλαμβανομένων λειτουργιών φιλτραρίσματος,συγκέντρωσης, ένωσης και μετασχηματισμού. ([81])

numpy

Η βιβλιοθήκη NumPy είναι μια θεμελιώδης βιβλιοθήκη για αριθμητικούς υπολογισμούς της γλώσσας Python. Παρέχει υποστήριξη για πολυδιάστατους και μη πίνακες, μαζί με ένα σύνολο μαθηματικών συναρτήσεων για την αποτελεσματική λειτουργία τους. ([82])

textblob

Η TextBlob είναι μια βιβλιοθήκη που παρέχει ένα εύχρηστο API για την εκτέλεση εργασιών επεξεργασίας φυσικής γλώσσας (NLP – Natural Language Processing). Είναι χτισμένο πάνω σε άλλες δημοφιλείς βιβλιοθήκες όπως η NLTK (Natural Language Toolkit) και χρησιμοποιείται για εργασίες όπως η ανάλυση συναισθήματος, η προσθήκη ετικετών σε μέρος του λόγου, η εξαγωγή ουσιαστικών φράσεων, η μετάφραση και άλλα. ([83])

wordcloud

Η βιβλιοθήκη WordCloud είναι μια δημοφιλής βιβλιοθήκη που χρησιμοποιείται για τη δημιουργία σύννεφων με λέξεις από δεδομένα κειμένου. Ένα σύννεφο λέξεων

είναι μια οπτική αναπαράσταση δεδομένων κειμένου, όπου το μέγεθος και το χρώμα των λέξεων καθορίζονται από τη συχνότητα ή τη σημασία τους στο κείμενο εισαγωγής. ([84])

re

Η βιβλιοθήκη re (Regular Expression) είναι μια ενσωματωμένη βιβλιοθήκη Python που παρέχει υποστήριξη για κανονικές εκφράσεις. Οι κανονικές εκφράσεις είναι μοτίβα που χρησιμοποιούνται για την αντιστοίχιση και το χειρισμό συμβολοσειρών κειμένου. Η βιβλιοθήκη επιτρέπει την αναζήτηση, αντιστοίχιση και χειρισμό κειμένου χρησιμοποιώντας τα μοτίβα αυτά. ([85])

matplotlib.pyplot

Η μονάδα (module) matplotlib.pyplot είναι μέρος της βιβλιοθήκης Matplotlib, η οποία είναι μια βιβλιοθήκη οπτικοποίησης δεδομένων. Παρέχει μια διεπαφή παρόμοια με το MATLAB για τη δημιουργία ενός ευρέος φάσματος στατικών, κινούμενων και διαδραστικών πλοκών. ([86])

seaborn

Η Seaborn είναι μια βιβλιοθήκη οπτικοποίησης δεδομένων χτισμένη στην Matplotlib. Παρέχει μια διεπαφή υψηλού επιπέδου για τη δημιουργία ενημερωτικών στατιστικών γραφικών. Το Seaborn είναι ιδιαίτερα χρήσιμο για τη δημιουργία οπτικά ελκυστικών απεικονίσεων για στατιστική ανάλυση. ([87])

nltk.wordnetlemmatizer

Η NLTK (Natural Language Toolkit) είναι μια βιβλιοθήκη για επεξεργασία φυσικής γλώσσας (NLP). Προσφέρει μια συλλογή βιβλιοθηκών και εργαλείων για εργασίες όπως το tokenization, το stemming (διαδικασία εύρεσης της λεκτικής ρίζας), το tagging (διαδικασία ορισμού ετικετών στις λέξεις μιας πρότασης, υποδεικνύοντας τη γραμματική τους κατηγορία, όπως ουσιαστικό, ρήμα, επίθετο κ.λπ.), η ανάλυση συναισθήματος και άλλα. Η NLTK χρησιμοποιείται ευρέως στον ακαδημαϊκό χώρο και τη βιομηχανία για έρευνα και ανάπτυξη επεξεργασίας της φυσικής γλώσσας.
Η επέκταση WordNetLemmatizer είναι μέρος της βιβλιοθήκης NLTK. Είναι ένα εργαλείο που χρησιμοποιείται για τη λημματοποίηση (lemmatization) λέξεων, που σημαίνει μείωση των λέξεων στη βάση ή τη μορφή λεξικού τους, γνωστή ως λήμμα. Η λημματοποίηση είναι χρήσιμη σε διάφορες εργασίες επεξεργασίας φυσικής γλώσσας (NLP) για την κανονικοποίηση των λέξεων και τη μείωση τους σε μια κοινή μορφή ανάλυσης. ([88])

sklearn

Η Scikit-learn γνωστή και ως sklearn, είναι μια δημοφιλής βιβλιοθήκη μηχανικής εκμάθησης. Παρέχει ένα ευρύ φάσμα εργαλείων και αλγορίθμων για διάφορες εργασίες μηχανικής μάθησης, όπως ταξινόμηση, παλινδρόμηση, ομαδοποίηση, μείωση διαστάσεων, επιλογή μοντέλου και προεπεξεργασία. ([89])

langdetect

Η Langdetect είναι μια βιβλιοθήκη που χρησιμοποιείται για την ανίχνευση γλώσσας σε κείμενο. Παρέχει έναν απλό τρόπο αναγνώρισης της γλώσσας ενός κειμένου ή εγγράφου. Η βιβλιοθήκη βασίζεται σε συχνότητες χαρακτήρων n-grams (μία n-gram

είναι μια συνεχόμενη ακολουθία n στοιχείων από ένα δείγμα κειμένου ή ομιλίας) και αλγόριθμους μηχανικής μάθησης. ([90])

translators.googletrans
Η βιβλιοθήκη googletrans είναι μία επέκταση σε γλώσσα Python για το Google Translate API. Επιτρέπει την ενσωμάτωση των λειτουργιών του Google Translate σε εφαρμογές Python. Μεταφράζει κείμενο από μια γλώσσα σε άλλη καθορίζοντας τη γλώσσα προέλευσης και προορισμού και υποστηρίζει ένα ευρύ σύνολο γλωσσών. ([90])

pyspark.sql
(βλ. παραπάνω)

hdfs.insecureclient
Η hdfs.InsecureClient είναι μια κλάση στη βιβλιοθήκη hdfs που παρέχει μια διεπαφή για αλληλεπίδραση με το κατανεμημένο σύστημα αρχείων Hadoop (HDFS) χωρίς να απαιτείται έλεγχος ταυτότητας.
Η κλάση hdfs.InsecureClient επιτρέπει την εκτέλεση λειτουργιών στο HDFS, όπως δημιουργία καταλόγων, καταχώρηση αρχείων, αποστολή και λήψη αρχείων, διαγραφή αρχείων, ανάγνωση και εγγραφή περιεχομένων αρχείων. ([91])

Αφού φορτωθούν στο σύστημα οι βιβλιοθήκες, δηλώνονται οι συναρτήσεις που θα χρησιμοποιηθούν στον κυρίως κώδικα (main).

Υπάρχουν συναρτήσεις που αναλαμβάνουν την μορφοποίηση και την διαλογή του κειμένου. Αυτές είναι οι cleanData(), formatData() και lemmatizeData().

- cleanData(): Καθαρισμός των δεδομένων κειμένου από συγκεκριμένους χαρακτήρες και λέξεις.
- formatData(): Μορφοποίηση του κειμένου απομακρύνοντας τις συνδετικές λέξεις.
- lemmatizeData(): Λημματοποίηση λέξεων (βλ. Κεφάλαιο 3.1 στη σελίδα 157).

Άλλες συναρτήσεις πραγματοποιούν την ανάλυση συναισθήματος πάνω στα μορφοποιημένα δεδομένα. Αυτές είναι οι getSubjectivity(), getPolarity() και getAnalysis().

- getSubjectivity(): Δημιουργία βαθμών αντικειμενικότητας των δεδομένων κειμένου.
- getPolarity(): Δημιουργία βαθμών πολικότητας των δεδομένων κειμένου.
- getAnalysis(): Έλεγχος για αρνητικά, θετικά και ουδέτερα αποτελέσματα των δεδομένων κειμένου με βάση τις βαθμολογήσεις και χαρακτηρισμός αυτών ως «Αρνητικά», «Θετικά» ή «Ουδέτερα».

Η συνάρτηση langDetect() αναγνωρίζει την γλώσσα του κειμένου που έχει λάβει στην είσοδο και μεταφράζει το κείμενο στα αγγλικά με χρήση του Google Translate API.

Η συνάρτηση create_wordcloud() δημιουργεί το σύννεφο λέξεων από την λίστα που έχει λάβει ως είσοδο.

Η συνάρτηση model_Evaluate() πραγματοποιεί την αξιολόγηση του μοντέλου εκμάθησης που έχει λάβει ως είσοδο.

Προχωράμε στο κυρίως (main) πρόγραμμα όπου δημιουργείται η Συνεδρία Spark (Spark Session). Πλέον η εφαρμογή μπορεί να κάνει χρήση των εντολών Pyspark.

Κάνοντας χρήση του Pyspark API αρχικοποιούνται παράμετροι που καθορίζουν τα μονοπάτια αρχείων στο σύστημα HDFS για να είναι σε θέση το πρόγραμμα να γνωρίζει που θα πραγματοποιήσει την όποια αναζήτηση. Εκτός από τις παραμέτρους των μονοπατιών αρχικοποιούνται και οι κλάσεις των βιβλιοθηκών μετάφρασης (translators) και αναγνώρισης γλώσσας (langdetect), καθώς επίσης και η λίστα με τις συνδετικές λέξεις (stopwords) που θα χρησιμοποιηθούν για το φιλτράρισμα των δεδομένων κειμένου.

Το επόμενο μέρος του προγράμματος περιλαμβάνει τις μετατροπές (transformations) των δεδομένων που έχουν εισαχθεί, το φιλτράρισμα αυτών και την εφαρμογή της τεχνικής ανάλυσης συναισθήματος.

Το πρόγραμμα δέχεται σαν είσοδο ένα μονοδιάστατο σετ δεδομένων (dataset) το οποίο δηλαδή αποτελείται από μία στήλη (column) με δεδομένα κειμένου (text). Αφού το σετ δεδομένων φορτωθεί στην μνήμη από το αρχείο csv που έχουμε ορίσει στο πρόγραμμα να αναζητήσει, εκχωρείται σε ένα pyspark dataframe με όνομα *data_to_df*. Κατόπιν, γίνεται αναγνώριση για το αν υπάρχει κεφαλίδα (header). Εάν υπάρχει, τότε η εγγραφή της κεφαλίδας απορρίπτεται από το pyspark dataframe και η νέα κεφαλίδα που εκχωρείται ονομάζεται «rawData». Έπειτα γίνονται απόρριψη όλες οι διπλότυπες εγγραφές και το dataframe αποθηκεύεται στην κρυφή μνήμη για ευκολότερη πρόσβαση σε μελλοντικές μετατροπές αλλά και ανοχή σφαλμάτων. Στη συνέχεια, εφαρμόζεται η συνάρτηση cleanData() σε κάθε εγγραφή της στήλης «rawData» για να παραχθεί μια νέα στήλη με όνομα «cleanedData» και η οποία αποτελείται από τα αρχικά δεδομένα τα οποία όμως έχουν υποστεί φιλτράρισμα από συγκεκριμένους χαρακτήρες και λέξεις.

Το επόμενο κομμάτι του κώδικα έχει δημιουργηθεί για πειραματική χρήση καθώς η λειτουργία του εξαρτάται από τον μικρό αριθμό των εγγραφών σε ένα dataset. Εδώ

πραγματοποιείται η αναγνώριση της γλώσσας και η μετάφραση των δεδομένων κειμένου με χρήση του Google Translator API από οποιαδήποτε υποστηριζόμενη γλώσσα (Αναφορικά με το σχετικό documentation: https://py-googletrans.readthedocs.io/en/latest/) στην Αγγλική.

Η χρήση του συγκεκριμένου API όμως υπόκειται σε περιορισμούς. Συγκεκριμένα οι αιτήσεις ανά ώρα δεν θα πρέπει να ξεπερνούν τις 1000 σύμφωνα με το σχόλιο ενός χρήστη στον ιστότοπο Stack Overflow (https://stackoverflow.com/questions/56672411/googletrans-api-error-daily-limit-or-blocked-ip) το οποίο και επιβεβαιώθηκε με την χρήση. Άρα οι εγγραφές του dataset θα πρέπει να είναι 1000 ή λιγότερες.

Εάν η μετάφραση μέσω της συνάρτησης langDetect() ολοκληρωθεί με επιτυχία, παράγεται μία νέα στήλη με τα μεταφρασμένα δεδομένα η οποία αντικαθιστά την προηγούμενη (cleanedData) με το ίδιο όνομα.

Στη συνέχεια γίνεται φιλτράρισμα του dataframe από κενές εγγραφές και εφαρμόζονται οι συναρτήσεις getSubjectivity(), getPolarity() και getAnalysis() που δημιουργούν τις στήλες «Subjectivity», «Polarity» και «Analysis». Σε αυτό το σημείο έχουμε εισάγει με επιτυχία τα αποτελέσματα της τεχνικής ανάλυσης συναισθήματος στο dataframe. Τα επόμενα βήματα περιλαμβάνουν πρόσθετες ενέργειες που θα αξιοποιήσουν τα δεδομένα αυτά.

Στην συνέχεια λοιπόν, πραγματοποιούνται υπολογισμοί για τα ποσοστά των αρνητικών και θετικών αποτελεσμάτων επί του συνόλου των δεδομένων που θα εξαχθούν στο τελικό αρχείο excel με τα αποτελέσματα της εκτέλεσης του αλγορίθμου.

Στο επόμενο βήμα γίνεται εκτέλεση της συνάρτησης formatData() για την μορφοποίηση και προετοιμασία των δεδομένων της στήλης «cleanedData» για λημματοποίηση καθώς και την δημιουργία μιας νέας στήλης «formattedData». Η στήλη αυτή θα μετατραπεί στην συνέχεια σε RDD (βλ. Κεφάλαιο 1) που θα περιλαμβάνει τις λέξεις προς λημματοποίηση και θα εφαρμοστεί μια flatMap μετατροπή σε κάθε ένα στοιχείο κάνοντας χρήση της συνάρτησης lemmatizeData(). Έτσι η κάθε εγγραφή x1, x2, x3... θα υποστεί την μετατροπή lemmatized(x1), lemmatized(x2), lemmatized(x3)…

Τα νέα δεδομένα θα αποθηκευτούν σε μία λίστα Python για μετέπειτα χρήση με το όνομα «wordlist» που θα περιλαμβάνει τα λήμματα των λέξεων.

Στη συνέχεια ο αλγόριθμος παίρνει τα δεδομένα από τις στήλες «cleanedData» και «Polarity» όπου οι τιμές στην στήλη Polarity είναι διαφορετικές του 0. Μετονομάζει την στήλη «cleanedData» σε «text» και δημιουργεί ένα νέο dataframe που ονομάζεται *train_df*.

Στο *train_df* δημιουργείται μια νέα στήλη με όνομα «target». Εδώ θα πρέπει να αναφερθεί ότι τα δεδομένα της στήλης «Polarity» είναι τύπου float και οι τιμές τους κυμαίνονται από -1 εώς 1.

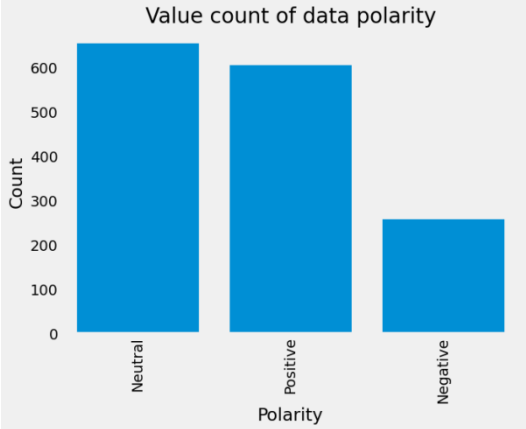
Όπου η αρχική τιμή της στήλης «Polarity» είναι μεγαλύτερη του 0 τότε η νέα τιμή γίνεται 1 σε μορφή integer, ενώ σε τιμές ίσες ή μικρότερες του 0 η τελική τιμή γίνεται 0. Η στήλη «Polarity» απορρίπτεται από το *train_df* και τη θέση της παίρνει η στήλη με τις νέες τιμές και όνομα «target». Άρα το dataframe *train_df* αποτελείται από τις στήλες «text» και «target» και περιέχει τα δεδομένα που θα χρησιμοποιηθούν αργότερα για την τροφοδότηση του μοντέλου εκμάθησης.

Στο επόμενο μέρος του προγράμματος δημιουργούνται γραφήματα των δεδομένων από τα δύο dataframes που έχουν προκύψει, το «data_to_df» και το «train_df», τα οποία εξάγονται είτε τοπικά στο μηχάνημα που τρέχει το πρόγραμμα (σε local mode), είτε τοπικά στο κατανεμημένο σύστημα αρχείων του Hadoop (σε cluster mode).

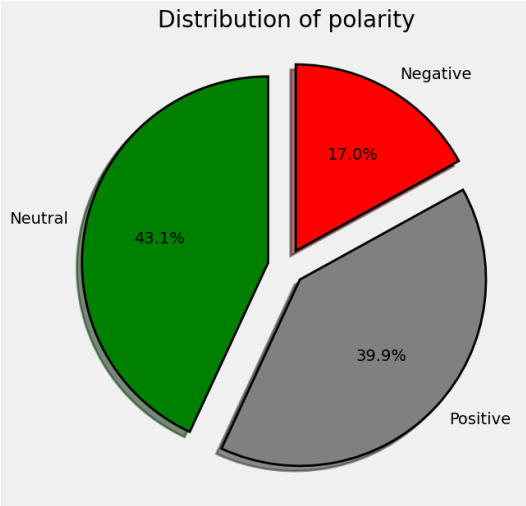
Τα dataframes του pyspark όμως δεν υποστηρίζουν λειτουργίες ανάλυσης δεδομένων που παρέχει η γλώσσα Python με ποικίλες βιβλιοθήκες για pandas dataframes. Την λύση σε αυτό το στάδιο, καλείται να δώσει η βιβλιοθήκη pyspark.pandas. Έτσι, τα παραπάνω pyspark dataframes μετατρέπονται σε pyspark.pandas dataframes (τα οποία για λόγους ευκολίας θα αναφέρονται στο εξής ως pandas dataframes) και μετονομάζονται σε *data_pdf* και *train_pdf* αντίστοιχα.

Στη συνέχεια δημιουργούνται τα εξής γραφήματα:

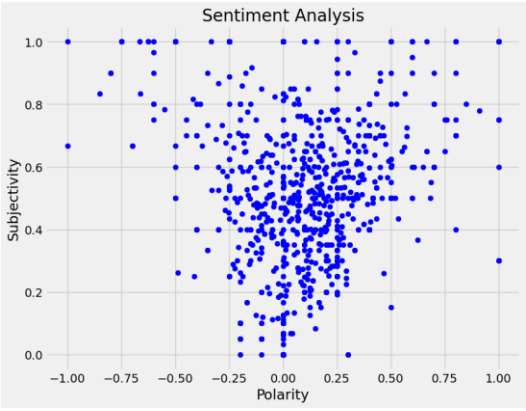
1. Διάγραμμα ράβδων για την εμφάνιση του ποσοστού της συναισθηματικής πολικότητας των δεδομένων. Στην ανάλυση συναισθήματος, η πολικότητα αναφέρεται στο συναίσθημα που εκφράζεται σε ένα κομμάτι κειμένου. Αντιπροσωπεύει το υποκείμενο θετικό, αρνητικό ή ουδέτερο συναίσθημα που μεταφέρεται από το κείμενο. Η πολικότητα συχνά ποσοτικοποιείται σε αριθμητική κλίμακα, όπου οι θετικές τιμές υποδηλώνουν θετικό συναίσθημα, οι αρνητικές τιμές υποδηλώνουν αρνητικό συναίσθημα και οι μηδενικές ή κοντά στο μηδέν τιμές δείχνουν ουδέτερο συναίσθημα. [Εικονα 3.7]
2. Διάγραμμα πίτας για την εμφάνιση του ποσοστού διανομής της συναισθηματικής πολικότητας. [Εικονα 3.8]
3. Διάγραμμα διασποράς της πολικότητας ως προς την υποκειμενικότητα. Στην ανάλυση συναισθήματος, η υποκειμενικότητα αναφέρεται στον βαθμό στον οποίο ένα κομμάτι κειμένου εκφράζει υποκειμενικό ή γνωμικό περιεχόμενο και όχι αντικειμενικές πληροφορίες. Οι υποκειμενικές εκφράσεις αντικατοπτρίζουν προσωπικές απόψεις, πεποιθήσεις, συναισθήματα, ή αξιολογήσεις. Συχνά περιλαμβάνουν τη χρήση λέξεων όπως «αγάπη», «μίσος», «καλό», «κακό», «υπέροχο» και ούτω καθεξής. [Εικονα 3.9]
4. Διάγραμμα των 10 συχνότερα χρησιμοποιούμενων λέξεων που παράγεται από την λίστα *wordlist* με τα λήμματα των λέξεων. [Εικονα 3.10]
   Τα γραφήματα ή αλλιώς σύννεφα λέξεων της βιβλιοθήκης wordcloud που απεικονίζουν τη συχνότητα λέξεων σε ένα σύνολο δεδομένων κειμένου. Ένα σύννεφο λέξεων απεικονίζει τις λέξεις που εμφανίζονται πιο συχνά σε ένα κείμενο, όπου το μέγεθος κάθε λέξης αντιπροσωπεύει τη συχνότητα ή τη σημασία της στο κείμενο. [Εικονα 3.11]



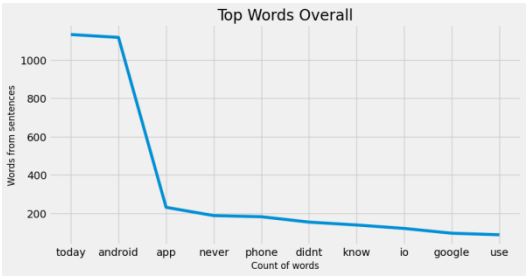
Εικόνα 3.7: Διάγραμμα ράβδων της συναισθηματικής πολικότητας των δεδομένων.



Εικόνα 3.8: Διάγραμμα πίτας της συναισθηματικής πολικότητας.



Εικόνα 3.9: Διάγραμμα διασποράς πολικότητας – υποκειμενικότητας.



Εικόνα 3.10: Διάγραμμα των 10 συχνότερα χρησιμοποιούμενων λέξεων.

Εικόνα 3.11: Σύννεφο λέξεων (word cloud) της συχνότητας λέξεων σε ένα σύνολο δεδομένων κειμένου.

Το τελευταίο μέρος έχει δημιουργηθεί και αυτό για πειραματική χρήση. Στο στάδιο ελέγχεται ο αριθμός των εγγραφών του *train_pdf* εάν ξεπερνά τις 200 εγγραφές (ενδεικτικός αριθμός) ούτως ώστε να μπορεί να τροφοδοτήσει επαρκώς το μοντέλο μηχανικής μάθησης που χρησιμοποιείται. Όσο μεγαλύτερος είναι ο όγκος των δεδομένων, τόσο πιο αποτελεσματική θα είναι και η εκμάθηση.
Από το *train_pdf* διαχωρίζοται τα χαρακτηριστικά εισόδου (input features) και ο στόχος (target). Τα δεδομένα αυτά παρέχονται από τις στήλες «text» και «target» που αναφέρθηκαν σε προηγούμενο βήμα.

Στην μηχανική μάθηση, οι όροι "χαρακτηριστικά εισόδου" (input features) και "στόχος" (target) αναφέρονται σε διαφορετικά μέρη ενός συνόλου δεδομένων που χρησιμοποιούνται για την εκπαίδευση ενός μοντέλου μηχανικής μάθησης. Τα χαρακτηριστικά εισόδου, γνωστά και ως ανεξάρτητες μεταβλητές ή προγνωστικοί παράγοντες, είναι οι μεταβλητές που χρησιμοποιούνται για την πρόβλεψη της μεταβλητής στόχου. Αντιπροσωπεύουν τα χαρακτηριστικά ή τις ιδιότητες των δεδομένων που χρησιμοποιούνται για να κάνουν μια πρόβλεψη.
Για παράδειγμα, σε ένα μοντέλο που προβλέπει τις τιμές των κατοικιών με βάση το μέγεθός τους, το χαρακτηριστικό εισόδου θα ήταν το μέγεθος του σπιτιού. Η μεταβλητή στόχος, γνωστή και ως εξαρτημένη μεταβλητή, είναι η μεταβλητή που προβλέπεται από το μοντέλο μηχανικής μάθησης, αντιπροσωπεύει το αποτέλεσμα ή την ετικέτα που προσπαθεί να προβλέψει το μοντέλο. Στο παράδειγμα της τιμής κατοικίας, η μεταβλητή στόχος θα ήταν η τιμή του σπιτιού. ([112])

Διαχωρίζεται λοιπόν το 95% των δεδομένων για εκμάθηση και το 5% για δοκιμές στους άξονες X και Y (X_train, X_test, y_train, y_test). Έπειτα, δημιουργείται μια οντότητα (instance) του TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer, και στην συνέχεια εισάγονται τα δεδομένα προς εκμάθηση.

Ο TF-IDF (Term Frequency-Inverse Document Frequency) Vectorizer εκχωρεί μια αριθμητική βαθμολογία σε κάθε λέξη ή όρο σε ένα έγγραφο, η οποία αντικατοπτρίζει πόσο σημαντικός είναι ο όρος για το έγγραφο στο πλαίσιο μιας συλλογής εγγράφων. Η βαθμολογία υπολογίζεται με βάση δύο παράγοντες:

1) Την Συχνότητα Όρου (TF): Ο αριθμός των φορών που εμφανίζεται ένας όρος σε ένα έγγραφο.
2) Την Αντίστροφη Συχνότητα Εγγράφων (IDF): Ένα μέτρο του πόσο σπάνιος ή κοινός είναι ένας όρος σε όλα τα έγγραφα σε ένα σώμα. Αυτό υπολογίζεται ως ο λογάριθμος του συνολικού αριθμού εγγράφων στο σώμα διαιρεμένο με τον αριθμό των εγγράφων που περιέχουν τον όρο.
Ο TF-IDF Vectorizer συνδυάζει τις βαθμολογίες TF και IDF για να δημιουργήσει μια ενιαία βαθμολογία για κάθε όρο σε ένα έγγραφο.
Το διάνυσμα (vector) που προκύπτει για ένα έγγραφο περιέχει τις βαθμολογίες TF-IDF για όλους τους όρους του εγγράφου.
Ο TF-IDF Vectorizer μπορεί να χρησιμοποιηθεί για την αναπαράσταση ενός εγγράφου ή μιας συλλογής εγγράφων σαν αριθμητικός πίνακας, όπου κάθε σειρά αντιπροσωπεύει ένα έγγραφο και κάθε στήλη αντιπροσωπεύει έναν όρο. Αυτός ο πίνακας μπορεί στη συνέχεια να χρησιμοποιηθεί ως είσοδος σε διάφορους αλγόριθμους μηχανικής μάθησης για εργασίες όπως η ταξινόμηση, η ομαδοποίηση και η ανάκτηση πληροφοριών.
([108]), ([109]), ([110]), ([111])

Κατόπιν ακολουθεί μετασχηματισμός των δεδομένων με την χρήση του TF-IDF Vectorizer και κωδικοποίηση των ετικετών σε αριθμητική μορφή.

Ο Κωδικοποιητής Ετικετών (LabelEncoder) χρησιμοποιείται για τη μετατροπή κατηγορικών ετικετών σε αριθμητικές ετικέτες. Εφαρμόζεται στα δεδομένα του άξονα y.

Το πρόγραμμα παρέχει την δυνατότητα αποθήκευσης του μοντέλου εκμάθησης σε αρχείο της μορφής plk. Εάν το αρχείο αυτό υπάρχει ήδη, τότε φορτώνεται στην μνήμη και το μοντέλο Bernoulli Naive Bayes που είχε αποθηκευτεί στο αρχείο. Εάν το αρχείο δεν βρεθεί, το μοντέλο Bernoulli Naive Bayes δημιουργείται από την αρχή και γίνεται εκπαίδευση στα δεδομένα.

Ο Bernoulli Naive Bayes (BNB) είναι ένας πιθανοτικός αλγόριθμος μηχανικής μάθησης που βασίζεται στο θεώρημα Bayes και μοντελοποιεί τις πιθανότητες κάθε χαρακτηριστικού στο διάνυσμα εισόδου να υπάρχει ή να απουσιάζει στην κλάση εξόδου.
Χρησιμοποιείται για προβλήματα ταξινόμησης, ειδικά σε ταξινόμηση κειμένου και φιλτράρισμα ανεπιθύμητων μηνυμάτων.
Είναι μια παραλλαγή του αλγορίθμου Naive Bayes, ο οποίος υποθέτει ότι τα χαρακτηριστικά είναι υπό όρους ανεξάρτητα μεταξύ τους, δεδομένης της ετικέτας κλάσης.

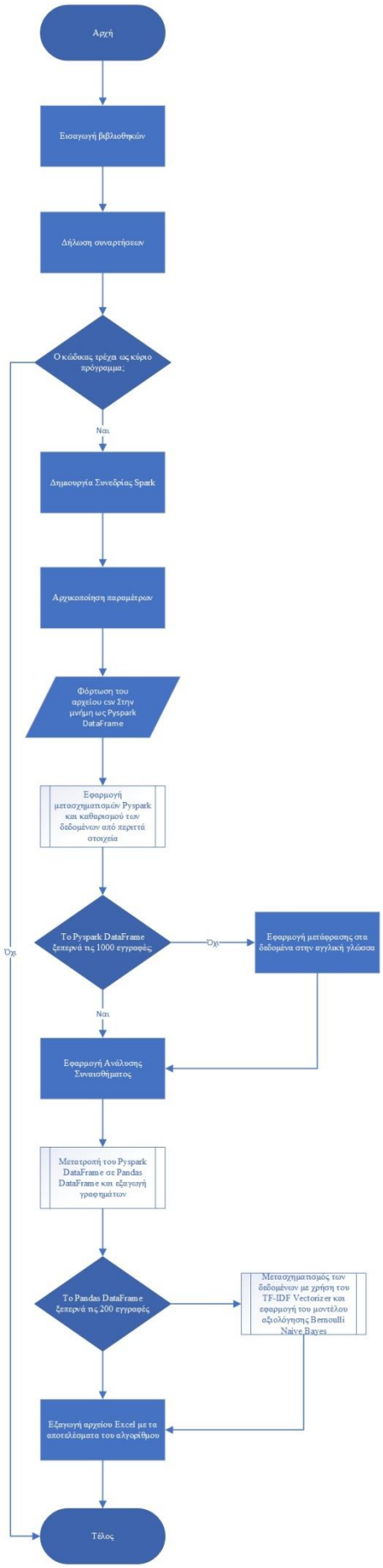
Στον Bernoulli Naive Bayes, τα χαρακτηριστικά είναι δυαδικά, δηλαδή αντιπροσωπεύουν την παρουσία ή την απουσία μιας συγκεκριμένης λέξης στο κείμενο. ([104]), ([105])

Τέλος, εξάγεται ένα γράφημα το οποίο απεικονίζει την Καμπύλης ROC-AUC για το μοντέλο και ενός αρχείου excel με τα τελικά αποτελέσματα του dataset [Εικόνα 3.12].

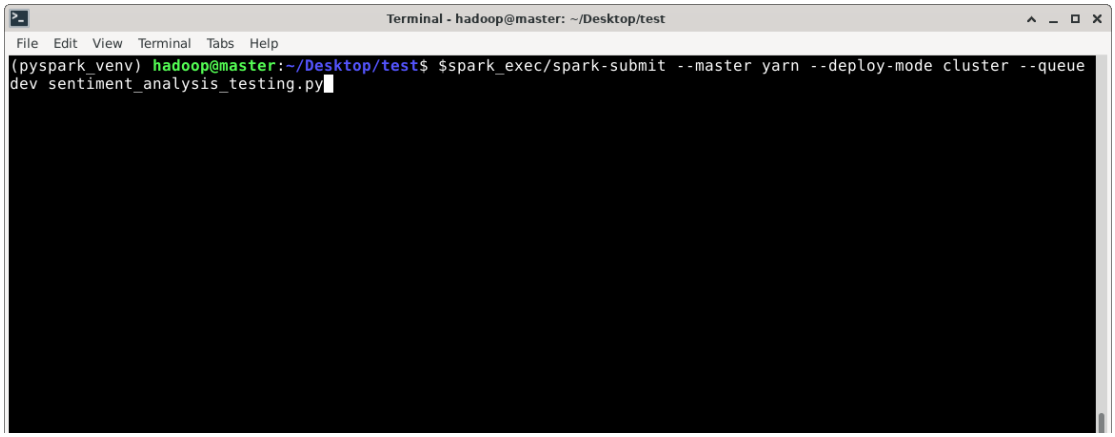
Total Rows	Positive Data	Negative Data	Trained Words
1518	39.9%	17.0%	3302

Εικόνα 3.12: Τα τελικά αποτελέσματα που παρήχθησαν απο το dataset σε μορφή αρχείου excel.

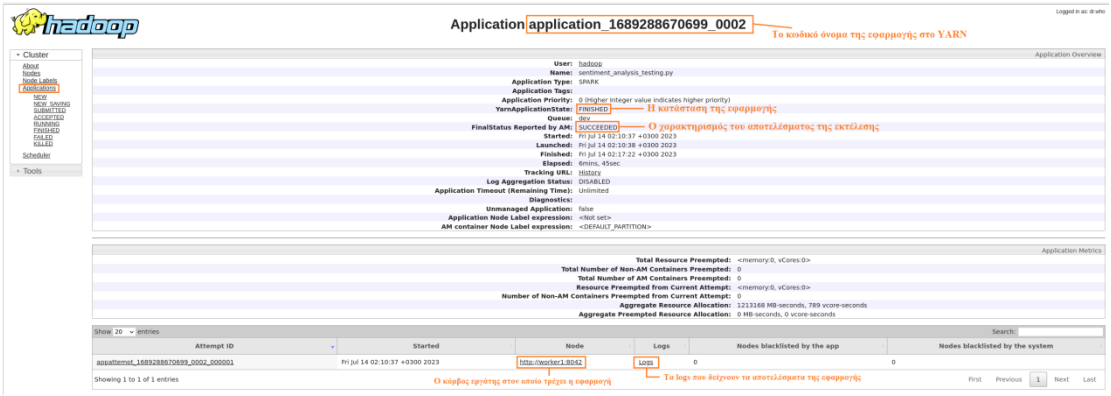
Η καμπύλη ROC-AUC είναι μια γραφική αναπαράσταση της απόδοσης ενός μοντέλου δυαδικής (binary) ταξινόμησης. Το "ROC" σημαίνει Χαρακτηριστικά Λειτουργίας Δέκτη (Receiver Operating Characteristic) και το AUC σημαίνει Περιοχή Κάτω από την Καμπύλη (Area Under the Curve). Η καμπύλη ROC-AUC δημιουργείται με τη σχεδίαση του πραγματικού θετικού ποσοστού (TPR) έναντι του ψευδώς θετικού ποσοστού (FPR) σε διαφορετικά κατώτατα όρια ταξινόμησης. Το TPR είναι ο λόγος των σωστά ταξινομημένων θετικών περιπτώσεων προς τον συνολικό αριθμό των θετικών περιπτώσεων και το FPR είναι ο λόγος των λανθασμένα ταξινομημένων αρνητικών περιπτώσεων προς τον συνολικό αριθμό των αρνητικών περιπτώσεων. Η καμπύλη ROC-AUC παρέχει μια οπτική αναπαράσταση του πόσο καλά ένα δυαδικό μοντέλο ταξινόμησης είναι σε θέση να διακρίνει μεταξύ θετικών και αρνητικών περιπτώσεων. Ένας τέλειος ταξινομητής εχει TPR 1 και FPR 0, με αποτέλεσμα AUC 1. Ένας τυχαίος ταξινομητής, από την άλλη πλευρά, θα είχε TPR και FPR που είναι και οι δύο κοντά στο 0.5, με αποτέλεσμα μια AUC της τάξης του 0.5. ([106]), ([107])
Κατά την διάρκεια εκτέλεσης μιας εφαρμογής Spark, μπορούμε να συνδεθούμε στην διαδικτυακή διεπαφή (Web UI) που παρέχει το YARN (βλ. Κεφάλαιο 2) και να παρακολουθήσουμε την κατάσταση της εκτέλεσης, όπως επίσης και διάφορες μετρήσεις (metrics) που μπορεί να μας ενδιαφέρουν. Παρακάτω στις εικόνες 3.14 και 3.15 μπορούμε να δούμε στιγμιότυπα από το Web Interface του Hadoop YARN.

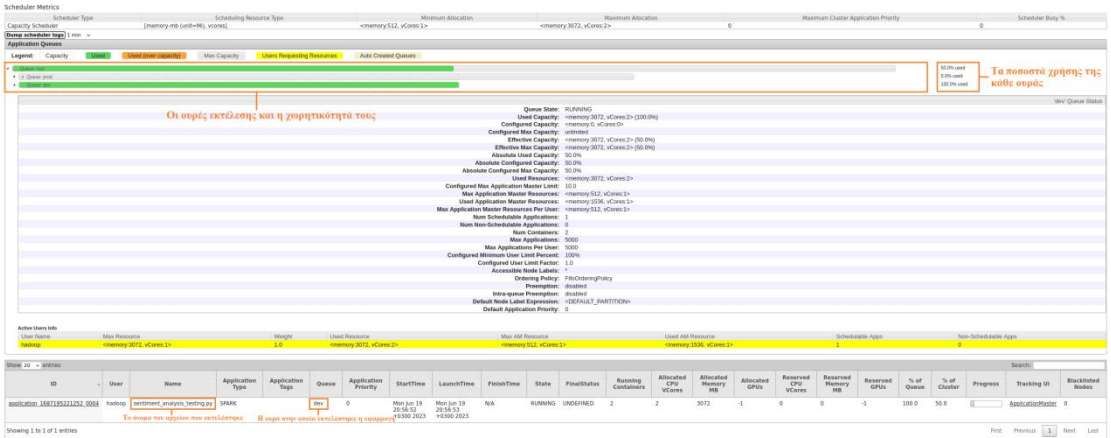

Αρχή
Εισαγωγή βιβλιοθηκών
Δήλωση συναρτήσεων
Ο κώδικας τρέχει ως κύριο πρόγραμμα;
Ναι
Όχι
Δημιουργία Συνεδρίας Spark
Αρχικοποίηση παραμέτρων
Φόρτωση του αρχείου csv Στην μνήμη ως Pyspark DataFrame
Εφαρμογή μετασχηματισμών Pyspark και καθαρισμού των δεδομένων από περιττά στοιχεία
Το Pyspark DataFrame ξεπερνά τις 1000 εγγραφές
Όχι
Εφαρμογή μετάφρασης στα δεδομένα στην αγγλική γλώσσα
Ναι
Εφαρμογή Ανάλυσης Συναισθήματος
Μετατροπή του Pyspark DataFrame σε Pandas DataFrame και εξαγωγή γραφημάτων
Το Pandas DataFrame ξεπερνά τις 200 εγγραφές
Μετασχηματισμός των δεδομένων με χρήση του TF-IDF Vectorizer και εφαρμογή του μοντέλου αξιολόγησης Bernoulli Naive Bayes
Εξαγωγή αρχείου Excel με τα αποτελέσματα του αλγορίθμου
Τέλος

Εικόνα 3.6: Το διάγραμμα ροής του αλγορίθμου "sentiment analysis".



Εικόνα 3.13: Εκτέλεση της εφαρμογής "sentiment analysis" σε cluster mode με YARN resource manager.





Εικόνα 3.14: Τα στοιχεία και οι μετρήσεις (metrics) του spark job στον capacity scheduler του YARN.

Εικόνα 3.15: Το output log της εφαρμογής "sentiment analysis" στο YARN.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] 'How much data is generated each day?', World Economic Forum, 17 Απρίλιος 2019. https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/

[2] ‘Difference between Structured, Semi-structured and Unstructured data’, GeeksforGeeks, 5 Νοέμβριος 2018. https://www.geeksforgeeks.org/difference-between-structured-semi-structured-and-unstructured-data/

[3] ‘What is Structured Data?’, TIBCO Software. https://www.tibco.com/reference-center/what-is-structured-data

[4] ‘What is Structured Data? - Structured Data Explained - AWS’, Amazon Web Services, Inc. https://aws.amazon.com/what-is/structured-data/

[5] ‘What Is Unstructured Data?’, MongoDB. https://www.mongodb.com/unstructured-data

[6] ‘What is Unstructured Data?’, GeeksforGeeks, 15 Απρίλιος 2019. https://www.geeksforgeeks.org/what-is-unstructured-data/

[7] ‘What is Semi-structured data?’, GeeksforGeeks, 12 Απρίλιος 2019. https://www.geeksforgeeks.org/what-is-semi-structured-data/

[8] ‘What Is Semi-Structured Data?’, MonkeyLearn Blog, 16 Νοέμβριος 2020. https://monkeylearn.com/blog/semi-structured-data/

[9] N. Miloslavskaya και A. Tolstoy, ‘Big Data, Fast Data and Data Lake Concepts’, Procedia Computer Science, τ. 88, σσ. 300–305, Ιανουαρίου 2016, doi: 10.1016/j.procs.2016.07.439.

[10] ‘What is a Data Mart | IBM’. https://www.ibm.com/topics/data-mart

[11] ‘What is a Data Mart? | Oracle’. https://www.oracle.com/autonomous-database/what-is-data-mart/

[12] ‘Data Mart vs. Data Warehouse’, Panoply. https://panoply.io/data-warehouse-guide/data-mart-vs-data-warehouse/

[13] ‘Data Mart in Data Warehouse and its Benefits | Complete Guide’. http://www.xenonstack.com/insights/data-mart

[14] ‘What is a data pipeline | IBM’. https://www.ibm.com/topics/data-pipeline

[15] ‘Data Pipeline’, Hazelcast. https://hazelcast.com/glossary/data-pipeline/

[16] B. Singhal και A. Aggarwal, 'ETL, ELT and Reverse ETL: A business case Study', στο 2022 Second International Conference on Advanced Technologies in Intelligent Control, Environment, Computing & Communication Engineering (ICATIECE), Δεκεμβρίου 2022, σσ. 1–4. doi: 10.1109/ICATIECE56365.2022.10046997.

[17] R. Jaffe, 'Library Guides: Metadata Creation: 1. What are metadata?' https://guides.library.ucsc.edu/c.php?g=618773&p=4306381

[18] 'IBM Documentation', 18 Μάρτιος 2022. https://www.ibm.com/docs/en/iis/11.5?topic=jobs-metadata

[19] S. Sagiroglu και D. Sinanc, 'Big data: A review', στο 2013 International Conference on Collaboration Technologies and Systems (CTS), Μαΐου 2013, σσ. 42–47. doi: 10.1109/CTS.2013.6567202.

[20] 'What Is Big Data?' https://www.oracle.com/big-data/what-is-big-data/

[21] 'Big data', Wikipedia. 30 Ιούνιος 2023. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://en.wikipedia.org/w/index.php?title=Big_data&oldid=1162719705

[22] 'What Is Big Data? Definition and Best Practices', Spiceworks. https://www.spiceworks.com/tech/big-data/articles/what-is-big-data/

[23] 'Apache Spark History'. https://www.programsbuzz.com/article/apache-spark-history

[24] E. Shaikh, I. Mohiuddin, Y. Alufaisan, και I. Nahvi, 'Apache Spark: A Big Data Processing Engine', στο 2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM), Νοεμβρίου 2019, σσ. 1–6. doi: 10.1109/MENACOMM46666.2019.8988541.

[25] 'Features of Apache Spark | Apache Spark Tutorial'. https://www.knowledgehut.com/tutorials/apache-spark-tutorial/apache-spark-features

[26] 'Components of Apache Spark', GeeksforGeeks, 20 Ιούνιος 2020. https://www.geeksforgeeks.org/components-of-apache-spark/

[27] D. Team, ‘Apache Spark Ecosystem - Complete Spark Components Guide’, DataFlair, 1 Μάιος 2017. https://data-flair.training/blogs/apache-spark-ecosystem-components/

[28] D. Team, ‘Apache Spark RDD vs DataFrame vs DataSet’, DataFlair, 10 Μάιος 2017. https://data-flair.training/blogs/apache-spark-rdd-vs-dataframe-vs-dataset/

[29] ‘RDD vs DataFrames and Datasets: A Tale of Three Apache Spark APIs’, Databricks, 14 Ιούλιος 2016. https://www.databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html

[30] ‘What is Spark SQL?’, Databricks, 18 Μάιος 2018. https://www.databricks.com/glossary/what-is-spark-sql

[31] www.facebook.com/sandeep.dayananda, ‘Spark SQL Tutorial | Understanding Spark SQL With Examples’, Edureka, 2 Ιανουάριος 2017. https://www.edureka.co/blog/spark-sql-tutorial/

[32] ‘What is Spark Streaming?’, Databricks, 18 Μάιος 2018. https://www.databricks.com/glossary/what-is-spark-streaming

[33] ‘What is a Machine Learning Library (MLlib)?’, Databricks, 18 Μάιος 2018. https://www.databricks.com/glossary/what-is-machine-learning-library

[34] ‘What is Spark GraphX? Everything You Need To Know | Simplilearn’, Simplilearn.com, 1 Δεκέμβριος 2022. https://www.simplilearn.com/spark-graphx-article

[35] ‘Apache Spark Architecture - Detailed Explanation’, InterviewBit, 3 Ιούνιος 2022. https://www.interviewbit.com/blog/apache-spark-architecture/

[36] S. P. M. Says, ‘Apache Spark Architecture | Distributed System Architecture Explained’, Edureka, 28 Σεπτέμβριος 2018. https://www.edureka.co/blog/spark-architecture/

[37] Naveen (NNK), ‘What is Apache Spark Driver?’, Spark By {Examples}, 23 Νοέμβριος 2022. https://sparkbyexamples.com/spark/what-is-apache-spark-driver/

[38] S. Hussain, ‘Understanding the working of Spark Driver and Executor’, Knoldus Blogs, 27 Δεκέμβριος 2019. https://blog.knoldus.com/understanding-the-working-of-spark-driver-and-executor/

[39] ‘Spark Executor | How Apache Spark Executor Works? | Uses’, EDUCBA, 31 Ιούλιος 2020. https://www.educba.com/spark-executor/

[40] rimmalapudi, ‘What is Spark Executor’, Spark By {Examples}, 5 Μάρτιος 2023. https://sparkbyexamples.com/spark/what-is-spark-executor/

[41] D. Team, ‘Apache Spark Executor for Executing Spark Tasks’, DataFlair, 17 Ιανουάριος 2018. https://data-flair.training/blogs/spark-executor/

[42] ‘HDFS Architecture Guide’. https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[43] ‘What Is Hadoop Namenode and Why Is It a Single Point of Failure | Saturn Cloud Blog’, 13 Ιούνιος 2023. https://saturncloud.io/blog/what-is-hadoop-namenode-and-why-is-it-a-single-point-of-failure/

[44] P. Majumder, ‘An Overview of HDFS: NameNodes and DataNodes’, Analytics Vidhya, 21 Απρίλιος 2022. https://www.analyticsvidhya.com/blog/2022/04/an-overview-of-hdfs-namenodes-and-datanodes/

[45] ‘What is HDFS? Hadoop Distributed File System overview’, Data Management. https://www.techtarget.com/searchdatamanagement/definition/Hadoop-Distributed-File-System-HDFS

[46] ‘How Does Namenode Handles Datanode Failure in Hadoop Distributed File System?’, GeeksforGeeks, 17 Σεπτέμβριος 2019. https://www.geeksforgeeks.org/how-does-namenode-handles-datanode-failure-in-hadoop-distributed-file-system/

[47] T. Team, ‘Apache Spark Cluster Manager: YARN, Mesos and Standalone’, TechVidvan, 11 Ιανουάριος 2018. https://techvidvan.com/tutorials/spark-cluster-manager-yarn-mesos-and-standalone/

[48] D. Team, ‘Apache Spark Cluster Managers - YARN, Mesos & Standalone’, DataFlair, 28 Απρίλιος 2017. https://data-flair.training/blogs/apache-spark-cluster-managers-tutorial/

[49] ‘What are the types of cluster managers in Spark?’, Educative: Interactive Courses for Software Developers. https://www.educative.io/answers/what-are-the-types-of-cluster-managers-in-spark

[50] ‘Component tools’, Kubernetes. https://kubernetes.io/docs/reference/command-line-tools-reference/

[51] ‘Concepts’, Kubernetes. https://kubernetes.io/docs/concepts/

[52] ‘Apache Hadoop 3.3.6 – Apache Hadoop YARN’. https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html

[53] ‘Apache Mesos’, Apache Mesos. https://mesos.apache.org/

[54] ‘Cluster Mode Overview - Spark 3.4.1 Documentation’. https://spark.apache.org/docs/latest/cluster-overview.html

[55] Naveen (NNK), ‘Spark Deploy Modes - Client vs Cluster Explained’, Spark By {Examples}, 10 Μάρτιος 2021. https://sparkbyexamples.com/spark/spark-deploy-modes-client-vs-cluster/

[56] Sephinreji, ‘Understanding Spark Deployment Modes: Client vs Cluster vs Local’, Medium, 14 Φεβρουάριος 2023. https://medium.com/@sephinreji98/understanding-spark-cluster-modes-client-vs-cluster-vs-local-d3c41ea96073

[57] ‘Spark Standalone Mode - Spark 3.4.1 Documentation’. https://spark.apache.org/docs/latest/spark-standalone.html

[58] ‘Spark Local Mode | ArcGIS GeoAnalytics Engine | ArcGIS Developers’, ArcGIS GeoAnalytics Engine. https://developers.arcgis.com/geoanalytics/install/local_mode/

[59] ‘Apache Spark Architecture - From Basics to Advance’. https://intellipaat.com/blog/tutorial/spark-tutorial/spark-architecture/?US

[60] D. Team, ‘How Apache Spark Works - Run-time Spark Architecture’, DataFlair, 10 Απρίλιος 2017. https://data-flair.training/blogs/how-apache-spark-works/

[61] ‘Apache Spark Architecture’, Encora. https://www.encora.com/insights/apache-spark-architecture

[62] L. Thorp, ‘Apache Spark — Multi-part Series: Spark Architecture’, Medium, 21 Ιανουάριος 2022. https://towardsdatascience.com/apache-spark-multi-part-series-spark-architecture-461d81e24010

[63] ‘Submitting Applications - Spark 3.4.1 Documentation’. https://spark.apache.org/docs/latest/submitting-applications.html

[64] Naveen (NNK), ‘Spark Submit Command Explained with Examples’, Spark By {Examples}, 23 Σεπτέμβριος 2020. https://sparkbyexamples.com/spark/spark-submit-command/

[65] ‘Submitting Spark applications | CDP Public Cloud’. https://docs.cloudera.com/runtime/7.2.10/running-spark-applications/topics/spark-submitting-apps.html? (

[66] D. Tobin, ‘Spark vs Hadoop MapReduce: 5 Key Differences’, Integrate.io. https://www.integrate.io/blog/apache-spark-vs-hadoop-mapreduce/

[67] ‘Hadoop vs. Spark: In-Depth Big Data Framework Comparison’, Data Management. https://www.techtarget.com/searchdatamanagement/feature/Hadoop-vs-Spark-Comparing-the-two-big-data-frameworks

[68] ‘Fig. 2. Sentiment Analysis Techniques’, ResearchGate. https://www.researchgate.net/figure/Sentiment-Analysis-Techniques_fig2_341017994

[69] G. Goyal, ‘Twitter Sentiment Analysis Using Python | Introduction & Techniques’, Analytics Vidhya, 11 Ιούνιος 2021. https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/

[70] N. C. Dang, M. N. Moreno-García, και F. De la Prieta, ‘Sentiment Analysis Based on Deep Learning: A Comparative Study’, Electronics, τ. 9, τχ. 3, Art. τχ. 3, Μαρτίου 2020, doi: 10.3390/electronics9030483.

[71] J. Castañón, ‘10 Machine Learning Methods that Every Data Scientist Should Know’, Medium, 5 Σεπτέμβριος 2019. https://towardsdatascience.com/10-machine-learning-methods-that-every-data-scientist-should-know-3cc96e0eeee9

[72] O. Harrison, ‘Machine Learning Basics with the K-Nearest Neighbors Algorithm’, Medium, 14 Ιούλιος 2019. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

[73] ‘Decision Tree Algorithm in Machine Learning - Javatpoint’. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[74] Y. Fan, ‘Introduction of a big data machine learning tool — SparkML’, Yurong Fan, 10 Ιανουάριος 2017. https://yurongfan.wordpress.com/2017/01/10/introduction-of-a-big-data-machine-learning-tool-sparkml/

[75] J. C. Wong, ‘The Cambridge Analytica scandal changed the world – but it didn’t change Facebook’, The Guardian, 18 Μάρτιος 2019. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.theguardian.com/technology/2019/mar/17/the-cambridge-analytica-scandal-changed-the-world-but-it-didnt-change-facebook

[76] T. Gross, ‘Whistleblower Explains How Cambridge Analytica Helped Fuel U.S. 'Insurgency'’, NPR, 8 Οκτώβριος 2019. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://www.npr.org/2019/10/08/768216311/whistleblower-explains-how-cambridge-analytica-helped-fuel-u-s-insurgency

[77] M. Rambocas και J. Gama, ‘Marketing Research: The Role Of Sentiment Analysis’, FEP Working Papers, Art. τχ. 489, Απριλίου 2013. [Έκδοση σε ψηφιακή μορφή]. Διαθέσιμο στο: https://ideas.repec.org//p/por/fepwps/489.html

[78] ‘os — Miscellaneous operating system interfaces’, Python documentation. https://docs.python.org/3/library/os.html

[79] Naveen (NNK), ‘PySpark SQL with Examples’, Spark By {Examples}, 19 Δεκέμβριος 2022. https://sparkbyexamples.com/pyspark/pyspark-sql-with-examples/

[80] ‘snscrape: A social networking service scraper’.

[81] Naveen (NNK), ‘Pandas API on Spark | Explained With Examples’, Spark By {Examples}, 9 Αύγουστος 2022. https://sparkbyexamples.com/pyspark/pandas-api-on-apache-spark-pyspark/

[82] ‘Python Numpy: Tutorial, What It is, Library - Javatpoint’. https://www.javatpoint.com/numpy-tutorial

[83] ‘TextBlob: Simplified Text Processing — TextBlob 0.16.0 documentation’. https://textblob.readthedocs.io/en/dev/

[84] ‘Generating Word Cloud in Python’, GeeksforGeeks, 11 Μάιος 2018. https://www.geeksforgeeks.org/generating-word-cloud-python/

[85] ‘re — Regular expression operations’, Python documentation. https://docs.python.org/3/library/re.html

[86] ‘What Is Matplotlib In Python? How to use it for plotting?’, ActiveState. https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/

[87] ‘Introduction to Seaborn - Python’, GeeksforGeeks, 31 Μάιος 2020. https://www.geeksforgeeks.org/introduction-to-seaborn-python/

[88] ‘Python | Lemmatization with NLTK’, GeeksforGeeks, 6 Νοέμβριος 2018. https://www.geeksforgeeks.org/python-lemmatization-with-nltk/

[89] ‘Scikit Learn Tutorial’. https://www.tutorialspoint.com/scikit_learn/index.htm

[90] ‘Googletrans: Free and Unlimited Google translate API for Python — Googletrans 3.0.0 documentation’. https://py-googletrans.readthedocs.io/en/latest/

[91] ‘Python InsecureClient Examples, hdfs.InsecureClient Python Examples - HotExamples’. https://python.hotexamples.com/examples/hdfs/InsecureClient/-/python-insecureclient-class-examples.html

[92] J.-G. Perrin, Spark in Action, Second Edition: Covers Apache Spark 3 with Examples in Java, Python, and Scala, 2nd edition. Shelter Island, NY: Manning, 2020.

[93] D. S. S. R. Mengle και M. Gurmendez, Mastering Machine Learning on AWS: Advanced machine learning in Python using SageMaker, Apache Spark, and TensorFlow. Packt Publishing, 2019.

[94] G. Maas και F. Garillot, Stream Processing with Apache Spark: Mastering Structured Streaming and Spark Streaming, 1st edition. Sebastopol, CA: O'Reilly Media, 2019.

[95] 'Data Analytics with Spark Using Python (Addison-Wesley Data & Analytics Series): Aven, Jeffrey: 9780134846019: Amazon.com: Books'. https://www.amazon.com/Analytics-Spark-Using-Python-Addison-Wesley/dp/013484601X/ref=as_li_ss_tl?dchild=1&keywords=apache+spark&qid=1601579528&s=books&sr=1-31&linkCode=sl1&tag=solutionsre04-20&linkId=0e86d071dfe73e76a557b33d4b1ee112&language=en_US

[96] J. Luraschi, K. Kuo, και E. Ruiz, Mastering Spark with R: The Complete Guide to Large-Scale Analysis and Modeling, 1st edition. Beijing China ; Sebastopol, CA: O'Reilly Media, 2019.

[97] J. Damji, B. Wenig, T. Das, και D. Lee, Learning Spark: Lightning-Fast Data Analytics, 2nd edition. Sebastopol, CA: O'Reilly Media, 2020.

[98] T. Maheshwari, 'All About Spark- Jobs, Stages and Tasks', Analytics Vidhya, 24 Σεπτέμβριος 2022. https://www.analyticsvidhya.com/blog/2022/09/all-about-spark-jobs-stages-and-tasks/

[99] 'Difference between Star Schema and Snowflake Schema', GeeksforGeeks, 27 Μάιος 2019. https://www.geeksforgeeks.org/difference-between-star-schema-and-snowflake-schema/

[100] 'What is a Data Warehouse? | IBM'. https://www.ibm.com/topics/data-warehouse

[101] 'What is a Data Warehouse?' https://www.oracle.com/database/what-is-a-data-warehouse/

[102] ‘What is a Data Lake? Data Lake vs. Warehouse | Microsoft Azure’. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-a-data-lake/

[103] ‘Introduction to Data Lakes’, Databricks, 21 Μάρτιος 2023. https://www.databricks.com/discover/data-lakes

[104] ‘sklearn.naive_bayes.BernoulliNB’, scikit-learn. https://scikit-learn/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html

[105] ‘1.9. Naive Bayes’, scikit-learn. https://scikit-learn/stable/modules/naive_bayes.html (ημερομηνία πρόσβασης 6 Ιούλιος 2023).

[106] A. Bhandari, ‘Guide to AUC ROC Curve in Machine Learning : What Is Specificity?’, Analytics Vidhya, 15 Ιούνιος 2020. https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

[107] S. Narkhede, ‘Understanding AUC - ROC Curve’, Medium, 15 Ιούνιος 2021. https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[108] ‘Understanding TF-IDF for Machine Learning’, Capital One. https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/

[109] ‘Understanding TF-IDF (Term Frequency-Inverse Document Frequency)’, GeeksforGeeks, 20 Ιανουάριος 2021. https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

[110] V. Jayaswal, ‘Text Vectorization: Term Frequency — Inverse Document Frequency (TFIDF)’, Medium, 4 Οκτώβριος 2020. https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d

[111] N. C. R. Riego και D. B. Villarba, ‘Utilization of Multinomial Naive Bayes Algorithm and Term Frequency Inverse Document Frequency (TF-IDF Vectorizer) in Checking the Credibility of News Tweet in the Philippines’. arXiv, 30 Μάιος 2023. doi: 10.48550/arXiv.2306.00018.

[112] joydeep bhattacharjee, ‘Some Key Machine Learning Definitions’, Technology at Nineleaps, 20 Νοέμβριος 2017. https://medium.com/technology-nineleaps/some-key-machine-learning-definitions-b524eb6cb48

[113] R. Kitchin και G. McArdle, ‘What makes Big Data, Big Data? Exploring the ontological characteristics of 26 datasets’, *Big Data & Society*, τ. 3, τχ. 1, σ. 2053951716631130, Ιουνίου 2016, doi: 10.1177/2053951716631130.

[114] G. S. Reddy, R. Srinivasu, M. P. C. Rao, και S. R. Rikkula, ‘DATA WAREHOUSING, DATA MINING, OLAP AND OLTP TECHNOLOGIES ARE ESSENTIAL ELEMENTS TO SUPPORT DECISION-MAKING PROCESS IN INDUSTRIES’, τ. 02, τχ. 09, 2010.

[115] J. J. Santanna, R. Durban, A. Sperotto, και A. Pras, ‘Inside booters: An analysis on operational databases’, στο *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Μαΐου 2015, σσ. 432–440. doi: 10.1109/INM.2015.7140320.

[116] ‘Relational Databases and Mixed Workloads’, *MarkLogic*, 13 Νοέμβριος 2015. https://www.marklogic.com/blog/relational-databases-and-mixed-workloads/

[117] ‘OLTP’, *Corporate Finance Institute*. https://corporatefinanceinstitute.com/resources/data-science/oltp/

[118] ‘Operational Database (Advantages, Disadvantages, Features & Examples) - DatabaseTown’, 15 Δεκέμβριος 2022. https://databasetown.com/operational-database/

[119] ‘OLTP Database vs OLAP Data Warehouse - What’s the Difference’. https://www.bizdata.com.au/blogpost.php?p=olap-gets-you-from-database-to-data-warehouse

ΠΡΟΣΑΡΤΗΜΑ

pyspark_scraper.py

```
# Εισαγωγή των απαραίτητων βιβλιοθηκών

import os
os.environ["PYARROW_IGNORE_TIMEZONE"] = "1"

from os.path import exists
from pathlib import Path

from pyspark.sql import SparkSession
from pyspark import SparkContext

import snscrape.modules.twitter as sntwitter

from datetime import datetime, timedelta


# Δημιουργία συνάρτησης που συνδέεται με το Twitter API και χρησιμοποιώντας συγκεκριμένα φίλτρα διαβάζει tweets και τα γράφει σε ένα αρχείο csv στο κατανεμημένο σύστημα αρχείων του Hadoop
def collect():
    global total_path
    global file_exists
    for i, tweet in enumerate(sntwitter.TwitterSearchScraper(parameters).get_items()):
        if i>maxTweets:
            break
        #tweets_list.append([tweet.rawContent]) # Συλλογή όλων των πληροφοριών ενός tweet (ID:String, Content:String, User:String, Date: DateTime, Likes:Integer, Retweets: Integer, Replies: Integer, URL: String, Media:String or List of Strings)
```

```
        tweets_list.append([tweet.content]) # Συλλογή μόνο των
πληροφοριών από το πεδίο "Content"

    tweets_to_rdd = spark.sparkContext.parallelize(tweets_list)
    tweets_to_df = spark.createDataFrame(tweets_to_rdd)
    if file_exists:
                        tweets_to_df.write.mode("append").option("header",
False).csv(total_path)
    else:
        print('Creating the csv file...')
        tweets_to_df.write.option("header", False).csv(total_path)
                                                    file_exists           =
sc._jvm.org.apache.hadoop.fs.FileSystem.get(sc._jsc.hadoopConfiguration
()).exists(sc._jvm.org.apache.hadoop.fs.Path(total_path))


# Δημιουργία συνάρτησης που υπολογίζει το μέγεθος αρχείου csv σε
megabytes
def file_size():
    global path
    global hdfs_root
                                                size                      =
int(spark._jvm.org.apache.hadoop.fs.FileSystem.get(spark._jsc.hadoopCon
figuration()).getContentSummary(spark._jvm.org.apache.hadoop.fs.Path(hd
fs_root + path)).getLength()) / 1024 ** 2
    size = float('{:.2f}'.format(size))
    return size

# Η λίστα που θα αποθηκευτούν προσωρινά τα tweets και το όριο των
tweets ανά session με το API του Twitter
tweets_list = []
maxTweets = 5000


# Παράμετροι που ορίζονται από τον χρήστη πριν την εκτέλεση του
προγράμματος
until = datetime.now() - timedelta(days=1) # "Εώς [ημερομηνία]"
since = datetime.now() - timedelta(days=7) # "Από [ημερομηνία]"
limit = 3000.0 # Το όριο μεγέθους του αρχείου που θα δημιουργηθεί
keyword = 'android' # Η λέξη "κλειδί" της αναζήτησης.
hdfs_root = "hdfs://master:9000/" # Ο root κατάλογος του μονοπατιού
hdfs.
filename = "tweets_dataset_new" # Το όνομα του αρχείου csv που θα
δημιουργηθεί.
dir = "user/hadoop/" + filename + ".csv_files/" # Το μονοπάτι του
αρχείου csv που θα δημιουργηθεί στο hdfs
```

```
# Μορφοποίηση των ημερομηνιών και δημιουργία της τελικής μορφής των παραμέτρων που θα χρησιμοποιηθούν ως φίλτρα αναζήτησης
since_day = since.strftime("%Y:%m:%d").replace(":", "-")
until_day = until.strftime("%Y:%m:%d").replace(":", "-")
parameters = keyword + ' since:' + since_day + ' until:' + until_day + ' lang:en'


if __name__ == "__main__":

    # Δημιουργία Συνεδρίας Spark με όνομα "pysparkscraper"
    spark = SparkSession.builder.appName('pysparkscraper').getOrCreate()
    spark.conf.set("spark.sql.execution.arrow.pyspark.enabled", "true")
    sc= spark.sparkContext
    sc.setLogLevel("ERROR")


    print('Search for: ', parameters)

    # Σύνθεση των τελικών μονοπατιών στο hdfs
    hdfs = spark._jvm.org.apache.hadoop.fs.FileSystem.get(spark._jsc.hadoopConfiguration())
    file = filename + ".csv"
    path = dir + file
    total_path = hdfs_root + dir + file

    # Ανίχνευση του αρχείου csv
    file_exists = sc._jvm.org.apache.hadoop.fs.FileSystem.get(sc._jsc.hadoopConfiguration()).exists(sc._jvm.org.apache.hadoop.fs.Path(total_path))



    print('File: ', file, 'in ', 'hdfs://' + dir)

    print('File exists: ', file_exists)

    # Εάν το αρχείο υπάρχει τυπώνεται το όνομά του και το μέγεθός του (σε MB). Εάν όχι, ξεκινά η συλλογή των tweets.
    if file_exists:
        print('File size: ', file_size(path), '/', limit, ' MB')
    else:
        print('Start collecting data...')
        collect()
```

```
    size = file_size()

    try:
        # Όσο το μέγεθος του αρχείου csv είναι μικρότερο α΄πό το όριο
που έχουμε ορίσει επαναλαμβάνεται η διαδικασία συλλογής των tweets
        while size <= limit:
            collect()
            size = file_size()
            print('File size: ', file_size(), '/', limit, ' MB')


    except:
        print('The program was interrupted.')


    if size > limit:
        print('Program Ended: File limit reached.')




#*****************************************************************
```

sentiment_analysis.py

```
# Εισαγωγή των απαραίτητων βιβλιοθηκών

import os
os.environ["PYARROW_IGNORE_TIMEZONE"] = "1"

import pyspark
import pyspark.pandas as pd

import numpy as np

from textblob import TextBlob
from wordcloud import WordCloud
import re
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
```

```
import nltk
from nltk import WordNetLemmatizer


from collections.abc import Iterable

import sklearn.utils as utils
from sklearn import preprocessing
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import roc_curve, auc

from langdetect import DetectorFactory, detect
import translators as ts

import string

from pyspark.sql import SparkSession
from pyspark import SparkContext
from pyspark.sql import functions as F
from pyspark.sql.types import *

# Φιλτράρισμα των προειδοποιήσεων
import warnings
warnings.filterwarnings('ignore')

import pickle
import io
from hdfs import InsecureClient

#----------------------------------------------------------------------------


# Προσδιορισμός ενός σετ με όλες τις συνδετικές λέξεις στα Αγγλικά
stopwordlist = ['a', 'about', 'above', 'after', 'again', 'ain', 'all',
'am', 'an', 'and','any','are', 'as', 'at', 'be', 'because', 'been',
'before', 'being', 'below', 'between','both', 'by', 'can', 'd', 'did',
'do', 'does', 'doing', 'down', 'during', 'each','few', 'for', 'from',
'further', 'had', 'has', 'have', 'having', 'he', 'her', 'here', 'hers',
'herself', 'him', 'himself', 'his', 'how', 'i', 'if', 'in',
'into','is', 'it', 'its', 'itself', 'just', 'll', 'm', 'ma', 'me',
'more', 'most','my', 'myself', 'now', 'o', 'of', 'on', 'once', 'only',
'or', 'other', 'our', 'ours','ourselves', 'out', 'own', 're', 's',
'same', 'she', 'shes', 'should', 'shouldve', 'so', 'some', 'such', 't',
'than', 'that', 'thatll', 'the', 'their', 'theirs', 'them',
```

```
'themselves', 'then', 'there', 'these', 'they', 'this', 'those',
'through', 'to', 'too','under', 'until', 'up', 've', 'very', 'was',
'we', 'were', 'what', 'when', 'where', 'which', 'while', 'who', 'whom',
'why', 'will', 'with', 'won', 'y', 'you', 'youd', 'youll', 'youre',
'youve', 'your', 'yours', 'yourself', 'yourselves', 'the', 'no', 'not',
'but']


# Συνάρτηση για τον καθαρισμό των δεδομένων
def cleanData(text):
    text = re.sub('@[A-Za-z0-9_]+', '', text) # Διαγράφονται τα
mentions από το Twitter
    text = re.sub('[0-9]+', '', text) # Διαγράφονται οι αριθμοί
    text = re.sub(r'"', '', text) # Διαγράφονται τα κενά
    text = "".join(a for a in text if a not in "'") # Διαγράφονται οι
απόστροφοι
    text = re.sub(r'(?<=[a-zA-Z])/(?=[a-zA-Z])', ' \g<0> ', text) #
Προστίθεται κενό ανάμεσα σε δύο λέξεις με “/” : Π.χ. η φράση "Red/Blue
theme" μετατρέπεται σε "Red / Blue theme"
    text = re.sub(r'(?<=[a-zA-Z])_(?=[a-zA-Z])', ' \g<0> ', text) #
Προστίθεται κενό ανάμεσα σε δύο λέξεις με “_” : Π.χ. η φράση "Red_Blue
theme" μετατρέπεται σε "Red _ Blue theme"
    text = re.sub(r'(.)1+', r'1', text) # Διαγράφονται οι
επαναλαμβανόμενοι χαρακτήρες
    text = re.sub('#','',text) # Διαγράφονται όλοι οι χαρακτήρες “#”
    text = re.sub('RT[\s]+','',text) # Διαγράφονται όλες οι σημάνσεις
“RT” από τα ReTweets
    text = re.sub('https?:\/\/\S+', '', text) # Διαγράφονται τα URLs
    text = re.sub('\n',' ',text) # Αντικαθίστανται τις σημάνσεις νέας
γραμμής “\n” με κενό
    text = text.strip() # Απομακρύνονται τα κενά στην αρχή και στο
τέλος της φράσης
    return text


# Συνάρτηση για τον εντοπισμό της γλώσσας
def langDetect(data):
    translation = ts.google(data) # Χρήση του Google Translator API για
μετάφραση των δεδομένων
    return translation

# Συνάρτηση για την μορφοποίηση του κειμένου. Φιλτράρονται οι
συνδετικές λέξεις.
def formatData(text):
    text =  text.translate(translator)
    text = text.lower()
    text = " ".join([word for word in text.split() if word not in
STOPWORDS])
    return text
```

```
# Συνάρτηση λημματοποίησης λέξεων
def lemmatizeData(text):
    newtext = [lm.lemmatize(word) for word in text.split()]
    return newtext


# Συνάρτηση που μετατρέπει μια εμφωλευμένη λίστα σε ενιαία
def flatten(list):
    for item in list:
        if isinstance(item, Iterable) and not isinstance(item, str):
            for x in flatten(item):
                yield x
        else:
            yield item



# Συνάρτηση δημιουργίας αντικειμενικότητας
def getSubjectivity(text):
    return TextBlob(text).sentiment.subjectivity


# Συνάρτηση δημιουργίας πολικότητας
def getPolarity(text):
    return TextBlob(text).sentiment.polarity



# Δημιουργία συνάρτησης που ελέγχει τα αρνητικά, ουδέτερα και θετικά
αποτελέσματα των αναλύσεων
def getAnalysis(score):
    if score<0:
        return 'Negative'
    elif score ==0:
        return 'Neutral'
    else:
        return 'Positive'



# Δημιουργία συνάρτησης για το wordcloud
def create_wordcloud(text):
    allWords = ' '.join([data for data in text])
    wordCloud = WordCloud(background_color='white', width=800,
height=500, random_state=21, max_font_size=130).generate(allWords)
    plt.figure(figsize=(20,10))
    plt.imshow(wordCloud)
```

```
    plt.axis('off')
    global local_check
    if "local" not in local_check:
        plt.savefig(filename + '_wordcloud.png', bbox_inches='tight')
        client.upload(dir, filename + '_wordcloud.png', overwrite=True)
    else:
        plt.savefig(ldir + filename + '_wordcloud.png',
bbox_inches='tight')



# Αξιολόγηση Μοντέλου Bernoulli Naive Bayes
def model_Evaluate(model):
    # Προβλέπονται οι τιμές για το Test dataset
    y_pred = model.predict(X_test)
    # Υπολογίζει και σχεδιάζει τον Πίνακα Σύγχυσης. Ο Πίνακας Σύγχυσης
χρησιμοποιείται για την αξιολόγηση της απόδοσης ενός μοντέλου μηχανικής
μάθησης.
    cf_matrix = confusion_matrix(y_test, y_pred)
    categories = ['Negative','Positive']
    group_names = ['True Neg','False Pos', 'False Neg','True Pos']
    group_percentages = ['{0:.2%}'.format(value) for value in
cf_matrix.flatten() / np.sum(cf_matrix)]
    labels = [f'{v1}n{v2}' for v1, v2 in
zip(group_names,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cf_matrix, annot = labels, cmap = 'Blues',fmt = '',
    xticklabels = categories, yticklabels = categories)
    plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad =
10)
    plt.ylabel("Actual values" , fontdict = {'size':14}, labelpad = 10)
    plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)
    global local_check
    if "local" not in local_check:
        plt.savefig(filename + '_model_evaluation.png',
bbox_inches='tight')
        client.upload(dir, filename + '_model_evaluation.png',
overwrite=True)
    else:
        plt.savefig(ldir + filename + '_model_evaluation.png',
bbox_inches='tight')



# Η συνάρτηση model_Evaluate παίρνει ένα μοντέλο ως παράμετρο, το οποίο
είναι το εκπαιδευμένο μοντέλο που θέλουμε να αξιολογήσουμε στο
δοκιμαστικό σύνολο δεδομένων. Η συνάρτηση χρησιμοποιεί πρώτα τη μέθοδο
πρόβλεψης του μοντέλου για να προβλέψει τις ετικέτες για το δοκιμαστικό
σύνολο δεδομένων X_test και τις αποθηκεύει στη μεταβλητή y_pred.
```

```
Υπολογίζει τον πίνακα σύγχυσης για τις προβλέψεις και τις αληθινές ετικέτες χρησιμοποιώντας τη συνάρτηση confusion_matrix από το scikit-learn και αποθηκεύει το αποτέλεσμα στο cf_matrix. Ρυθμίζει την μορφοποίηση για τους σχολιασμούς στο διάγραμμα μήτρας σύγχυσης, ορίζοντας τις κατηγορίες (ετικέτες κλάσεων), τα ονόματα ομάδων (τα κελιά του πίνακα σύγχυσης) και τα ποσοστά ομάδων (τα ποσοστά κάθε κελιού σε σχέση με το σύνολο αριθμός προβλέψεων). Δημιουργεί μια γραφική παράσταση χάρτη θερμότητας του πίνακα σύγχυσης χρησιμοποιώντας τη συνάρτηση heatmap του seaborn, με σχολιασμούς που δείχνουν τα ονόματα και τα ποσοστά των ομάδων. Οι ετικέτες tick x και y ορίζονται στις κατηγορίες (ετικέτες κλάσης). Τέλος, η γραφική παράσταση αποθηκεύεται ως αρχείο εικόνας PNG με το όνομα αρχείου που καθορίζεται από τη μεταβλητή ονόματος αρχείου.



#-----------------------------------------------------------------------



# Έλεγχος για το αν ο κώδικας εκτελείται ως κύριο πρόγραμμα ή έχει εισαχθεί ως τμήμα σε άλλο πρόγραμμα. Ο κώδικας τρέχει μόνο στην πρώτη περίπτωση.
if __name__ == "__main__":


    #Ρυθμίσεις κρυφής μνήμης cache
    cache_settings = pyspark.StorageLevel.MEMORY_AND_DISK_2
#Χρησιμοποιείται η αποθήκευση στην προσωρινή μνήμη αλλά και στον δίσκο


    #Δημιουργία Συνεδρίας Spark (SparkSession) με όνομα “sentiment_analysis”
    spark = SparkSession.builder.appName('sentiment_analysis').getOrCreate()
    spark.conf.set("spark.sql.execution.arrow.pyspark.enabled", "true")
    sc = spark.sparkContext
    sc.setLogLevel("ERROR")

    # Το όνομα του χρήστη Hadoop
    hadoopuser = "hadoop"
    # Η διεύθυνση του master κόμβου της συστάδας
    hdfs_root = "hdfs://master:9000"
    # Το όνομα του αρχείου csv που θα φορτωθεί στην μνήμη (χωρίς την κατάληξη του αρχείου)
    filename = "tweets_dataset_new"
    # Το όνομα του μονοπατιού που βρίσκεται το αρχείο csv
    dir = "/" + filename + "_files/" # Το μονοπάτι του αρχείου csv στο hdfs
```

```
    # Η ονομασία του αρχείου pkl που θα αποθηκευτούν τα δεδομένα από το μοντέλο αξιολόγησης (χωρίς την κατάληξη του αρχείου)
    modelname = "BNBmodel"


    # Δημιουργείται μία σύνδεση πελάτη με τον hdfs server
    client = InsecureClient('http://master:9870')

    # Έλεγχος εάν το πρόγραμμα τρέχει σε local mode
    local_check = sc.master
    if "local" not in local_check:
        print("Spark application is running in cluster mode")
        model_path = "/trained_models/" #"user/" + hadoopuser + "/trained_models/" # Το σημείο στο hdfs που θα αποθηκευτούν τα δεδομένα από το μοντέλο αξιολόγησης
        model_full_path = hdfs_root + model_path + modelname + ".pkl" # Ολόκληρο το μονοπάτι του hdfs που θα αποθηκευτούν τα δεδομένα από το μοντέλο αξιολόγησης

        # Ενεργοποίηση του HDFS Api για το Pyspark το οποίο παρέχει πρόσβαση στο σύστημα αρχείων του Hadoop
        hdfs = spark._jvm.org.apache.hadoop.fs.FileSystem.get(spark._jsc.hadoopConfiguration())

        # Δημιουργείται κατάλογος για τα αρχεία που θα εξαχθούν στο HDFS
        if not hdfs.exists(spark._jvm.org.apache.hadoop.fs.Path(model_path)):
            client.makedirs(model_path, permission="777")
        if not hdfs.exists(spark._jvm.org.apache.hadoop.fs.Path(dir)):
            client.makedirs(dir, permission="777")

        nltk.download('omw-1.4', download_dir='/usr/lib/nltk_data')

    else:
        print("Spark application is running in local mode")
        ldir = "./" + filename + "_results/" # Το directory που θα χρησιμοποιηθεί για εκτέλεση σε local mode
        model_full_path = "./" + modelname + ".pkl"

        # Δημιουργείται κατάλογος για τα αρχεία που θα εξαχθούν
        if not os.path.exists(ldir):
            os.mkdir(ldir)

        nltk.download('omw-1.4')
```

```
    STOPWORDS = set(stopwordlist)
    # Εισαγωγή της κλάσης WordNetLemmatizer από την βιβλιοθήκη nltk
(Natural Language Toolkit)
    lm = nltk.WordNetLemmatizer()
    # Δημιουργία λίστας στίξεων.
    punctuations_list = string.punctuation
    # Αφαίρεση όλων των χαρακτήρων στη λίστα στίξεων από μια
συμβολοσειρά
    translator = str.maketrans('', '', punctuations_list)


    # Ορίζεται την τιμή σποράς (seed) για την κλάση DetectorFactory της
βιβλιοθήκης "langdetect" σε 0. Η βιβλιοθήκη langdetect παρέχει έναν
τρόπο ανίχνευσης της γλώσσας δεδομένων κειμένου. Η κλάση
DetectorFactory χρησιμοποιείται για τη διαμόρφωση της συμπεριφοράς της
διαδικασίας ανίχνευσης γλώσσας, όπως η προεπιλεγμένη γλώσσα στην οποία
θα επανέλθει εάν δεν μπορεί να εντοπιστεί γλώσσα ή ο παράγοντας
εξομάλυνσης που θα χρησιμοποιηθεί κατά τον υπολογισμό των πιθανοτήτων
γλώσσας. Ρυθμίζοντας την τιμή σποράς σε 0, αρχικοποιείτε τη γεννήτρια
τυχαίων αριθμών που χρησιμοποιείται από τον ανιχνευτή γλώσσας με μια
σταθερή τιμή. Αυτό διασφαλίζει ότι τα αποτελέσματα της διαδικασίας
ανίχνευσης γλώσσας είναι ντετερμινιστικά, που σημαίνει ότι θα είναι τα
ίδια κάθε φορά που εκτελείτε τον ανιχνευτή γλώσσας με το ίδιο κείμενο
εισαγωγής και διαμόρφωση.

    DetectorFactory.seed = 0


    # Δημιουργία ενός pyspark dataframe διαβάζοντας το αρχείο που
ορίστηκε παραπάνω
    data_to_df = spark.read.option("header", True).csv(dir + filename +
".csv")

    # Αποθηκεύεται το όνομα της κεφαλίδας του dataframe σε ξεχωριστό
σημείο
    header = data_to_df.columns[0]


    # Μετονομασία της κεφαλίδας σε "rawData"
    data_to_df = data_to_df.withColumnRenamed(header, "rawData")
    # Αυτή η γραμμή καταργεί τυχόν σειρές από το DataFrame όπου η τιμή
στη στήλη rawData είναι ίση με την τιμή που είναι αποθηκευμένη στη
μεταβλητή κεφαλίδας. Έτσι, εάν η πρώτη σειρά των αρχικών δεδομένων
περιέχει πραγματικά δεδομένα αντί για όνομα στήλης, θα το αφαιρεθεί από
το DataFrame. Ωστόσο, εάν η πρώτη σειρά των αρχικών δεδομένων περιέχει
πραγματικό όνομα στήλης, θα αφαιρεθεί η πρώτη σειρά του ονόματος στήλης
και τα δεδομένα θα με΄ίνουν ανέπαφα.
    data_to_df = data_to_df.filter(data_to_df.rawData != header)
    # Φιλτράρισμα των κενών (null) τιμών
```

```
    data_to_df = data_to_df.filter(data_to_df.rawData.isNotNull())


    # Φιλτράρισμα των διπλών εγγραφών και αποθήκευση του pyspark
DataFrame στην κρυφή μνήμη έτσι ώστε να μπορεί να αποφευχθεί ο
επανυπολογισμός του από την πηγή του κάθε φορά που χρησιμοποιείται,
κάτι που μπορεί να βελτιώσει την απόδοση και την ανοχή σε σφάλματα.
    data_to_df = data_to_df.distinct().persist(cache_settings)


    #Εμφανίζονται οι πρώτες 5 γραμμές (για εκτέλεση σε local mode)
    data_to_df.show(5)


    # Εφαρμόζεται η συνάρτηση "cleanData" σε κάθε γραμμή των δεδομένων
και δημιουργείται μια νέα στήλη με το όνομα "cleanedData". Ο τύπος των
τιμών της στήλης "cleanedData" είναι String
    cleaned = F.udf(lambda q: cleanData(q), StringType())
    data_to_df = data_to_df.withColumn("cleanedData",
cleaned(F.col("rawData")))


    data_to_df.show(5)


    # ΠΕΙΡΑΜΑΤΙΚΗ ΧΡΗΣΗ
    # Εάν οι γραμμές του pyspark DataFrame "data_to_df" δεν ξεπερνούν
τις 2000 εγγραφές:
    if data_to_df.count() <= 1000:


        try:

            # Επιλέγεται ένας πολύ μικρός αριθμός εγγραφών από το
"data_to_df" (συγκεκριμένα οι πρώτες 5) για δοκιμή με χρήση του Google
Translate API και δημιουργείται ένα νέο pyspark DataFrame το
"data_to_df2". Ο λόγος που χρησιμοποιείται ένας τόσο μικρός αριθμός
είναι διότι η δωρεάν έκδοση του API φέρει περιορισμό στην χρήση με
500,000 χαρακτήρες / ημέρα.
            data_to_df2 = data_to_df.head(5).persist(cache_settings)
            data_to_df2 = spark.createDataFrame(data_to_df2)

            print("Data for testing: ")
            data_to_df2.show()
```

```
            # Εφαρμόζεται η συνάρτηση "langDetect" στο pyspark
DataFrame "data_to_df2" με την δημιουργία μιας νέας στήλης
"cleanedData_new"
            translated = F.udf(lambda q: langDetect(q), StringType())
            data_to_df2 = data_to_df2.withColumn("cleanedData_new",
translated(F.col("cleanedData")))

            # Αποσύρεται η στήλη "cleanedData" και η "cleanedData_new"
μετονομάζεται σε "cleanedData" και παίρνει τη θέση της με τα
μεταφρασμένα δεδομένα
            data_to_df2 = data_to_df2.drop("cleanedData")
            data_to_df2 =
data_to_df2.withColumnRenamed("cleanedData_new", "cleanedData")


            print("Translated data for testing: ")
            data_to_df2.show()

            # Το "data_to_df2" αποδεσμεύεται από την προσωρινή μνήμη
            data_to_df2.unpersist()


        except:


            print("Could not apply translation on the dataframe")



    # Φιλτράρισμα των καθαρισμένων δεδομένων από του "data_to_df" από
τυχόν κενές (null) τιμές
    data_to_df = data_to_df.filter(data_to_df.cleanedData.isNotNull())



    # Εφαρμόζονται οι συναρτήσεις "getSubjectivity" και "getPolarity"
σε κάθε γραμμή των δεδομένων και δημιουργούνται δυο νέες στήλες
"Subjectivity" και "Polarity" αντίστοιχα. Ο τύπος των τιμών και των δύο
στηλών είναι Float
    subjectivity = F.udf(lambda q: getSubjectivity(q), FloatType())
    polarity = F.udf(lambda q: getPolarity(q), FloatType())


    data_to_df = data_to_df.withColumn("Subjectivity",
subjectivity(F.col("cleanedData")))
    data_to_df.show(5)
```

```
    data_to_df = data_to_df.withColumn("Polarity",
polarity(F.col("cleanedData")))
    data_to_df.show(5)

    # Απόσυρση της στήλης "rawData" από το "data_to_df"
    data_to_df = data_to_df.drop("rawData")



    # Εφαρμόζεται η συνάρτηση "getAnalysis" σε κάθε γραμμή των
δεδομένων και δημιουργείται μια νέα στήλη με το όνομα "Analysis". Ο
τύπος των τιμών της στήλης "Analysis" είναι String
    analysis = F.udf(lambda q: getAnalysis(q), StringType())
    data_to_df = data_to_df.withColumn("Analysis",
analysis(F.col("Polarity")))

    data_to_df.show(5)

    # Φιλτράρισμα των δεδομένων. Κρατούνται οι εγγραφές της όπου οι
έχουν τιμή "Analysis = Positive", δηλαδή τα θετικά αποτελέσματα
    positiveData = data_to_df.filter(data_to_df.Analysis ==
"Positive").select(F.col("cleanedData"))

    # Υπολογισμός του ποσοστού % του συνόλου των θετικών αποτελεσμάτων
ως προς το γενικό σύνολο
    percentage = round((positiveData.count() / data_to_df.count()) *
100, 1)
    positiveData = str(percentage) + "%"

    # Φιλτράρισμα των δεδομένων. Κρατούνται οι εγγραφές της όπου οι
έχουν τιμή "Analysis = Negative", δηλαδή τα αρνητικά αποτελέσματα
    negativeData = data_to_df.filter(data_to_df.Analysis ==
"Negative").select(F.col("cleanedData"))

    # Υπολογισμός του ποσοστού % του συνόλου των αρνητικών
αποτελεσμάτων ως προς το γενικό σύνολο
    percentage = round((negativeData.count() / data_to_df.count()) *
100, 1)
    negativeData = str(percentage) + "%"


    # Εφαρμογή της συνάρτησης "formatData" για την μορφοποίηση και
προετοιμασία του κειμένου για ληματοποίηση καθώς και δημιουργία νέας
στήλης "formattedData"
    formatted = F.udf(lambda q: formatData(q))
    data_to_df = data_to_df.withColumn("formattedData",
formatted(F.col("cleanedData")))
```

```
    # Από την στήλη "formattedData" του "data_to_df" δημιουργείται ένα
στοιχείο RDD που αποτελείται από μια λίστα των ληματοποιημένων λέξεων
    data_to_rdd = data_to_df.select("formattedData").rdd.flatMap(lambda
list:[lemmatizeData(item) for item in list])
    # Μετατροπή του παραπάνω RDD σε λίστα Python.
    wordlist = data_to_rdd.mapPartitions(lambda
list:flatten(list)).collect()

    # Απόσυρση της στήλης "formattedData" στο "data_to_df"
    data_to_df = data_to_df.drop("formattedData")


    # Δημιουργία ενός pyspark DataFrame με όνομα "train_df"
χρησιμοποιώντας τις στήλες "cleanedData" και "Polarity", όπου η τιμή
"Polarity" είναι διαφορετική του 0 και αποθήκευση σε μια νέα στήλη με
όνομα "text". Το "train_df" θα χρησιμοποιηθεί αργότερα για την χρήση
ενός μοντέλου μηχανικής μάθησης.
    train_df = data_to_df.select(data_to_df["cleanedData"],
data_to_df["Polarity"]) \
            .filter(data_to_df.Polarity != 0.0) \
            .withColumnRenamed("cleanedData", "text") \
            .persist(cache_settings)

    # Δημιουργία μιας νέας στήλης στο "train_df" με όνομα "target".
Όπου οι τιμές της στήλης "Polarity" είναι μεγαλύτερες του 0 η νέα τιμή
στην στήλη "target" γίνεται 1. Σε διαφορετική περίπτωση η τιμή γίνεται
0. Ο τύπος των τιμών της στήλης "target" είναι Integer.
    train_df = train_df.withColumn("target", \
                        F.when(train_df["Polarity"] > 0.0,
1).otherwise(0).cast(IntegerType()))

    # Απόσυρση της στήλης "Polarity" από το "train_df"
    train_df = train_df.drop("Polarity")



#-------------------------------------------------------------------


    # Μετατροπή του Pyspark DataFrame σε Pandas DataFrame για να
δημιουργηθούν τα αρχεία των αποτελεσμάτων ανάλυσης δεδομένων
    data_pdf = data_to_df.toPandas()
    train_pdf = train_df.toPandas()



    # Αποδέσμευση όλων των παραπάνω Pyspark DataFrames που είχαν
αποθηκευτεί στην κρυφή μνήμη
```

```
    data_to_df.unpersist()
    data_to_rdd.unpersist()
    train_df.unpersist()



    # Σχεδιασμός ενός γραφήματος ράβδων για την εμφάνιση του αριθμού
του συναισθήματος δεδομένων
    fig = plt.figure(figsize=(7,5))
    xlabel = ['Positive','Negative','Neutral']
    plt.bar(xlabel,data_pdf['Analysis'].value_counts())
    data_pdf['Analysis'].value_counts().plot(kind='bar')
    plt.title('Value count of data polarity')
    plt.ylabel('Count')
    plt.xlabel('Polarity')
    plt.grid(False)
    if "local" not in local_check:
        plt.savefig(filename + '_sentiment_count.png',
bbox_inches='tight')
        client.upload(dir, filename + '_sentiment_count.png',
overwrite=True)
    else:
        plt.savefig(ldir + filename + '_sentiment_count.png',
bbox_inches='tight')



    # Σχεδιασμός ενός γραφήματος πίτας για την εμφάνιση του ποσοστού
διανομής της πολικότητας
    fig = plt.figure(figsize=(7,7))
    colors = ('green', 'grey', 'red')
    wp={'linewidth':2, 'edgecolor': 'black'}
    tags=data_pdf['Analysis'].value_counts()
    explode = (0.1,0.1,0.1)
    tags.plot(kind='pie', autopct='%1.1f%%', shadow=True,
colors=colors, startangle=90, wedgeprops=wp, explode=explode, label='')
    plt.title('Distribution of polarity')
    if "local" not in local_check:
        plt.savefig(filename + '_sentiment_percentage.png',
bbox_inches='tight')
        client.upload(dir, filename + '_sentiment_percentage.png',
overwrite=True)
    else:
        plt.savefig(ldir + filename + '_sentiment_percentage.png',
bbox_inches='tight')
```

```
    # Σχεδιασμός της πολικότητας και της υποκειμενικότητας σε ένα
διάγραμμα διασποράς
    plt.figure(figsize=(9,7))
    for i in range(0,data_pdf.shape[0]):
        plt.scatter(data_pdf['Polarity'][i],data_pdf['Subjectivity'][i]
, color='blue')
    plt.title('Sentiment Analysis')
    plt.xlabel('Polarity')
    plt.ylabel('Subjectivity')
    if "local" not in local_check:
        plt.savefig(filename + '_polarity_subjectivity.png',
bbox_inches='tight')
        client.upload(dir, filename + '_polarity_subjectivity.png',
overwrite=True)
    else:
        plt.savefig(ldir + filename + '_polarity_subjectivity.png',
bbox_inches='tight')



    # Σχεδιασμός ενός wordcloud για τα θετικά αποτελέσματα
    posData = data_pdf.loc[data_pdf['Analysis']=='Positive',
'cleanedData']
    create_wordcloud(posData)


    # Σχεδιασμός ενός wordcloud για τα αρνητικά αποτελέσματα
    negData = data_pdf.loc[data_pdf['Analysis']=='Negative',
'cleanedData']
    create_wordcloud(negData)



    # Το αντικείμενο Series που προκύπτει περιέχει τον αριθμό εμφάνισης
κάθε λέξης στα αρχικά δεδομένα εισόδου της λίστας λέξεων
    lem = pd.DataFrame(wordlist)
    lem = lem[0].value_counts()



    #Σχεδιασμός των 10 πιο χρησιμοποιούμενων λέξεων
    lem = lem[:10]
    plt.figure(figsize=(10,5))
    plt.plot(lem.index, lem.values)
    plt.title('Top Words Overall')
    plt.xlabel('Count of words', fontsize=12)
    plt.ylabel('Words from sentences', fontsize=12)
    if "local" not in local_check:
        plt.savefig(filename + '_top_words.png', bbox_inches='tight')
```

```
        client.upload(dir, filename + '_top_words.png', overwrite=True)
    else:
        plt.savefig(ldir + filename + '_top_words.png',
bbox_inches='tight')



    # ΠΕΙΡΑΜΑΤΙΚΗ ΧΡΗΣΗ
    # Έλεγχος για το αν ο αριθμός των εγγραφών στο "train_pdf" είναι
μεγαλύτερος από 200:
    if len(train_pdf.index) > 200:


        # Διαχωρίζοται τα χαρακτηριστικά εισόδου (input features) και ο
στόχος (target). Στην μηχανική μάθηση, οι όροι "χαρακτηριστικά εισόδου"
και "στόχος" αναφέρονται σε διαφορετικά μέρη ενός συνόλου δεδομένων που
χρησιμοποιούνται για την εκπαίδευση ενός μοντέλου μηχανικής μάθησης. Τα
χαρακτηριστικά εισόδου, γνωστά και ως ανεξάρτητες μεταβλητές ή
προγνωστικοί παράγοντες, είναι οι μεταβλητές που χρησιμοποιούνται για
την πρόβλεψη της μεταβλητής στόχου. Αντιπροσωπεύουν τα χαρακτηριστικά ή
τις ιδιότητες των δεδομένων που χρησιμοποιούνται για να κάνουν μια
πρόβλεψη. Για παράδειγμα, σε ένα μοντέλο που προβλέπει τις τιμές των
κατοικιών με βάση το μέγεθός τους, το χαρακτηριστικό εισόδου θα ήταν το
μέγεθος του σπιτιού. Η μεταβλητή στόχος, γνωστή και ως εξαρτημένη
μεταβλητή, είναι η μεταβλητή που προβλέπεται από το μοντέλο μηχανικής
μάθησης. Αντιπροσωπεύει το αποτέλεσμα ή την ετικέτα που προσπαθεί να
προβλέψει το μοντέλο. Στο παράδειγμα της τιμής κατοικίας, η μεταβλητή
στόχος θα ήταν η τιμή του σπιτιού.
        X=train_pdf.text
        y=train_pdf.target


        #Διαχωρίζεται το 95% των δεδομένων για εκμάθηση και το 5% για
δοκιμές
        X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size = 0.05, random_state =26105111)



        #Δημιουργία μιας οντότητας (instance) του TF-IDF (Term
Frequency-Inverse Document Frequency) Vectorizer, και στην συνέχεια
εισαγωγή των δεδομένων προς εκμάθηση
        vectorizer = TfidfVectorizer(use_idf=True, max_features=500000)
        vectorizer.fit(X_train)


        # Το TF-IDF (Term Frequency-Inverse Document Frequency)
Vectorizer εκχωρεί μια αριθμητική βαθμολογία σε κάθε λέξη ή όρο σε ένα
```

```
έγγραφο, η οποία αντικατοπτρίζει πόσο σημαντικός είναι ο όρος για το
έγγραφο στο πλαίσιο μιας συλλογής εγγράφων. Η βαθμολογία υπολογίζεται
με βάση δύο παράγοντες:
1) Την Συχνότητα Όρου (TF): Ο αριθμός των φορών που εμφανίζεται ένας
όρος σε ένα έγγραφο.
2) Την Αντίστροφη Συχνότητα Εγγράφων (IDF): Ένα μέτρο του πόσο σπάνιος
ή κοινός είναι ένας όρος σε όλα τα έγγραφα σε ένα σώμα. Αυτό
υπολογίζεται ως ο λογάριθμος του συνολικού αριθμού εγγράφων στο σώμα
διαιρεμένο με τον αριθμό των εγγράφων που περιέχουν τον όρο.
Το TF-IDF Vectorizer συνδυάζει τις βαθμολογίες TF και IDF για να
δημιουργήσει μια ενιαία βαθμολογία για κάθε όρο σε ένα έγγραφο. Το
διάνυσμα (vector) που προκύπτει για ένα έγγραφο περιέχει τις
βαθμολογίες TF-IDF για όλους τους όρους του εγγράφου. Το TF-IDF
Vectorizer μπορεί να χρησιμοποιηθεί για την αναπαράσταση ενός εγγράφου
ή μιας συλλογής εγγράφων σαν αριθμητικός πίνακας, όπου κάθε σειρά
αντιπροσωπεύει ένα έγγραφο και κάθε στήλη αντιπροσωπεύει έναν όρο.
Αυτός ο πίνακας μπορεί στη συνέχεια να χρησιμοποιηθεί ως είσοδος σε
διάφορους αλγόριθμους μηχανικής μάθησης για εργασίες όπως η ταξινόμηση,
η ομαδοποίηση και η ανάκτηση πληροφοριών.
```

```
        # Μετασχηματισμός των δεδομένων με την χρήση του TF-IDF
Vectorizer
        X_train = vectorizer.transform(X_train)
        X_test  = vectorizer.transform(X_test)


        # Έλεγχος της μορφής των ετικετών του άξονα y.
        if y_train.dtype != 'int' or y_train.dtype != 'float':
            # Εφαρμόζεται κωδικοποίηση των ετικετών σε αριθμητική
μορφή.
            # Ο Κωδικοποιητής Ετικετών (LabelEncoder) χρησιμοποιείται
για τη μετατροπή κατηγορικών ετικετών σε αριθμητικές ετικέτες. Θα
πρέπει να εφαρμόζεται στα δεδομένα του άξονα y
            label_encoder = preprocessing.LabelEncoder()
            y_train = label_encoder.fit_transform(y_train)
            y_test  = label_encoder.fit_transform(y_test)
            print("Labels have been converted to numerical format using
label encoding")


        feature_words = len(vectorizer.get_feature_names_out()) #Αυτή η
γραμμή υπολογίζει τον αριθμό των μοναδικών λέξεων που χρησιμοποιήθηκαν
από τον vectorizer


        if "local" not in local_check:
```

```
            # Έλεγχος για το αν υπάρχει το αρχείο του μοντέλου
αξιολόγησης του οποίου τη θέση ορίσαμε στην αρχή
            file_exists =
sc._jvm.org.apache.hadoop.fs.FileSystem.get(sc._jsc.hadoopConfiguration
()).exists(sc._jvm.org.apache.hadoop.fs.Path(model_full_path))

            # Εάν το αρχείο plk βρεθεί:
            if file_exists:
                # Φορτώνεται στην μνήμη το μοντέλο Bernoulli Naive
Bayes που είχε αποθηκευτεί στο αρχείο. Πρόκειται για έναν τύπο μοντέλου
που υποθέτει ότι τα χαρακτηριστικά είναι δυαδικά (δηλαδή 0 ή 1).
                model_data =
spark.sparkContext.binaryFiles(model_full_path).take(1)[0][1]
                model_data = io.BytesIO(model_data)
                BNBmodel = pickle.load(model_data)
            else:
                BNBmodel = BernoulliNB() # Εάν το αρχείο δεν βρεθεί, το
μοντέλο Bernoulli Naive Bayes δημιουργείται από την αρχή

        else:

            # Έλεγχος για το αν υπάρχει το αρχείο του μοντέλου
αξιολόγησης του οποίου τη θέση ορίσαμε στην αρχή
            file_exists = os.path.exists(model_full_path)
            # Εάν το αρχείο plk βρεθεί:
            if file_exists:
                 # Φορτώνεται στην μνήμη το μοντέλο Bernoulli Naive
Bayes που είχε αποθηκευτεί στο αρχείο
                with open(model_full_path, 'rb') as file:
                    model_data = file.read()

                model_data = io.BytesIO(model_data)
                BNBmodel = pickle.load(model_data)
            else:
                BNBmodel = BernoulliNB() # Εάν το αρχείο δεν βρεθεί, το
μοντέλο Bernoulli Naive Bayes δημιουργείται από την αρχή



        # Ο Bernoulli Naive Bayes (BNB) είναι ένας πιθανοτικός
αλγόριθμος μηχανικής μάθησης που βασίζεται στο θεώρημα Bayes.
Χρησιμοποιείται για προβλήματα ταξινόμησης, ειδικά σε ταξινόμηση
κειμένου και φιλτράρισμα ανεπιθύμητων μηνυμάτων. Είναι μια παραλλαγή
του αλγορίθμου Naive Bayes, ο οποίος υποθέτει ότι τα χαρακτηριστικά
είναι υπό όρους ανεξάρτητα μεταξύ τους, δεδομένης της ετικέτας κλάσης.
Στον Bernoulli Naive Bayes, τα χαρακτηριστικά είναι δυαδικά, δηλαδή
αντιπροσωπεύουν την παρουσία ή την απουσία μιας συγκεκριμένης λέξης στο
κείμενο.
```

```
        # Μοντελοποιεί τις πιθανότητες κάθε χαρακτηριστικού στο
διάνυσμα εισόδου να υπάρχει ή να απουσιάζει στην κλάση εξόδου


        # Εφαρμογή του Μοντέλου Bernoulli Naive Bayes και εκπαίδευση με
τα δεδομένα
        BNBmodel = BernoulliNB()
        BNBmodel.fit(X_train, y_train)
        model_Evaluate(BNBmodel)
        y_pred1 = BNBmodel.predict(X_test)


        # Η καμπύλη ROC-AUC είναι μια γραφική αναπαράσταση της απόδοσης
ενός μοντέλου δυαδικής (binary) ταξινόμησης. Το "ROC" σημαίνει
Χαρακτηριστικά Λειτουργίας Δέκτη (Receiver Operating Characteristic)
και το AUC σημαίνει Περιοχή Κάτω από την Καμπύλη (Area Under the
Curve). Η καμπύλη ROC-AUC δημιουργείται με τη σχεδίαση του πραγματικού
θετικού ποσοστού (TPR) έναντι του ψευδώς θετικού ποσοστού (FPR) σε
διαφορετικά κατώτατα όρια ταξινόμησης. Το TPR είναι ο λόγος των σωστά
ταξινομημένων θετικών περιπτώσεων προς τον συνολικό αριθμό των θετικών
περιπτώσεων και το FPR είναι ο λόγος των λανθασμένα ταξινομημένων
αρνητικών περιπτώσεων προς τον συνολικό αριθμό των αρνητικών
περιπτώσεων. Η καμπύλη ROC-AUC παρέχει μια οπτική αναπαράσταση του πόσο
καλά ένα δυαδικό μοντέλο ταξινόμησης είναι σε θέση να διακρίνει μεταξύ
θετικών και αρνητικών περιπτώσεων. Ένας τέλειος ταξινομητής έχει TPR 1
και FPR 0, με αποτέλεσμα AUC 1. Ένας τυχαίος ταξινομητής, από την άλλη
πλευρά, θα είχε TPR και FPR που είναι και οι δύο κοντά στο 0.5, με
αποτέλεσμα μια AUC της τάξης του 0.5.


        # Σχεδιασμός της Καμπύλης ROC-AUC για το μοντέλο
        fpr, tpr, thresholds = roc_curve(y_test, y_pred1)
        roc_auc = auc(fpr, tpr)
        plt.figure()
        plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve
(area = %0.2f)' % roc_auc)
        plt.xlim([0.0, 1.0])
        plt.ylim([0.0, 1.05])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.title('ROC CURVE')
        plt.legend(loc="lower right")
        if "local" not in local_check:
            plt.savefig(filename + '_roc_curve.png',
bbox_inches='tight')
            client.upload(dir, filename + '_roc_curve.png',
overwrite=True)
        else:
```

```
            plt.savefig(ldir + filename + '_roc_curve.png',
bbox_inches='tight')



        # Σειριοποίηση του μοντέλου σε bytes
        model_bytes = pickle.dumps(BNBmodel)


        # Εγγραφή των σειριοποιημένων δεδομένων σε ένα αρχείο pkl
        if "local" not in local_check:
            output_stream =
spark._jvm.org.apache.hadoop.fs.FileSystem.get(spark._jsc.hadoopConfigu
ration()).create(spark._jvm.org.apache.hadoop.fs.Path(model_full_path),
True)
            output_stream.write(model_bytes)
            output_stream.close()
        else:
            with open(model_full_path, 'wb') as file:
                output_stream = pickle.Pickler(file)
                output_stream.dump(model_bytes)


        print(f"Trained model saved to:{model_full_path}")


    else:
        feature_words = 0



    # Δημιουργείται ένα αρχείο excel με τα τελικά αποτελέσματα
    data = np.array([[data_pdf.shape[0], positiveData, negativeData,
feature_words]])
    data_sum = pd.DataFrame(data, columns=['Total Rows', 'Positive
Data', 'Negative Data', 'Trained Words'])
    if "local" not in local_check:
        data_sum.to_excel(filename + '_data_sum.xlsx')
        client.upload(dir, filename + '_data_sum.xlsx', overwrite=True)
    else:
        data_sum.to_excel(ldir + filename + '_data_sum.xlsx')



    print('Sentiment Analysis finished.')
```