

# ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

## ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

### ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ


## Πρόγραμμα Μεταπτυχιακών Σπουδών
## Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών

**Ειδίκευση Λογισμικού και Πληροφοριακών Συστημάτων,**

**ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ**

# A Binary Classification Approach for Identifying Sexual Predators in Chat Messages Using Spark


**Σταύρος Ανδροκλής Δερζιωτης**
**Α.Μ. 20014**

**Εισηγητής: Βασίλειος Μάμαλης**


**Αθήνα, Ιούνιος 2023**

# A Binary Classification Approach for Identifying Sexual Predators in Chat Messages Using Spark

## Μέλη Εξεταστικής Επιτροπής συμπεριλαμβανομένου και του Εισηγητή

Η μεταπτυχιακή διπλωματική εργασία εξετάστηκε επιτυχώς από την κάτωθι Εξεταστική Επιτροπή:

| Α/α | ΟΝΟΜΑ ΕΠΩΝΥΜΟ | ΒΑΘΜΙΔΑ/ΙΔΙΟΤΗΤΑ | ΨΗΦΙΑΚΗ ΥΠΟΓΡΑΦΗ |
|---|---|---|---|
| 1 | Μάμαλης Βασίλειος | Καθηγητής | Vasileios Mamalis Digitally signed by Vasileios Mamalis Date: 2023.06.25 23:07:41 +03'00' |
| 2 | Καντζάβελου Ιωάννα | Επίκουρη Καθηγήτρια | Ioanna Kantzavelou Digitally signed by Ioanna Kantzavelou Date: 2023.06.26 13:46:02 +03'00' |
| 3 | Καρκαζής Παναγιώτης | Αναπληρωτής Καθηγητής | Panagiotis Karkazis Digitally signed by Panagiotis Karkazis Date: 2023.06.27 13:13:04 +03'00' |

**Ημερομηνία εξέτασης 21/06/2023**

A Binary Classification Approach for Identifying Sexual Predators in Chat Messages Using Spark

# ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΜΕΤΑΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Δερζιώτης Σταύρος Ανδροκλής του Νικολάου, με αριθμό μητρώου 20014 φοιτητής του Προγράμματος Μεταπτυχιακών Σπουδών Επιστήμη και Τεχνολογία της Πληροφορικής και των Υπολογιστών του Τμήματος Μηχανικών Πληροφορικής της Σχολής Μηχανικών του Πανεπιστημίου Δυτικής Αττικής, δηλώνω ότι:

«Είμαι συγγραφέας αυτής της μεταπτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο Δηλών

**(Κενό φύλλο)**

# ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της μεταπτυχιακής διπλωματικής μου εργασίας θα ήθελα για αρχή να ευχαριστήσω την σύζυγο μου Δήμητρα και την κόρη μου Σταματίνα. Η πρώτη για το ότι είναι πάντα δίπλα μου και με στηρίζει ενεργά σε όλες τις επιλογές που κάνω και η κόρη μου, που μας κάνει καθημερινά ευτυχισμένους και αλλάζει συνεχώς, όσο μεγαλώνει, τον τρόπο που αντιλαμβάνομαι τα πράγματα γύρω μου. Κομμάτι αυτής της αντίληψης συνέβαλε και στην επιλογή του συγκεκριμένου θέματος της παρούσας εργασίας.

Επίσης ξεχωριστά, τα λόγια δεν είναι αρκετά για να ευχαριστήσω τον επιβλέποντα καθηγητή μου Κο Μάμαλη που πέρα από την σημαντική του βοήθεια και υποστήριξη στην υλοποίηση της διπλωματικής μου εργασίας, αποτέλεσε πηγή ουσιαστικής επιρροής και έμπνευσης για την ενασχόληση μου με τα πεδία της μηχανικής μάθησης και κατανεμημένης/παράλληλης επεξεργασίας.

Τέλος θα ήθελα να ευχαριστήσω το Πανεπιστήμιο Δυτικής Αττικής και το Κέντρο Εθνικών Υποδομών Έρευνας και τεχνολογίας (GRNET). Χάρη στην συνεργασία τους είχα την δυνατότητα πρόσβασης στους υπολογιστικούς πόρους του “ Okeanos - Knossos” για την δημιουργία της υποστηρικτικής υποδομής με την οποία υλοποιήθηκε η εργασία μου.

**(Κενό φύλλο)**

# ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική έχει ως σκοπό την χρήση αλγόριθμων μηχανικής μάθησης με στόχο την πιθανή ταυτοποίηση περιπτώσεων παιδεραστίας ή πιθανής εκμετάλλευσης ανηλίκων σε chats μεταξύ χρηστών. Δίνεται έμφαση σε τεχνικές φιλτραρίσματος και προ επεξεργασίας κειμένων και θα παρουσιάζονται δοκιμές με διάφορους αλγόριθμούς ταξινόμησης της βιβλιοθήκης SPARK's MLLib.

Για την υλοποίηση δημιουργήθηκε συστοιχία υπολογιστών (cluster) με HDFS σύστημα αρχείων (HADOOP) και μηχανή ανάλυσης δεδομένων μεγάλης κλίμακας (SPARK)

Το υλικό (Dataset) που χρησιμοποιείται λόγω της φύσης του προβλήματος είναι αρκετά μη ισορροπημένο επομένως δίνεται έμφαση στις καταλληλότερες μετρικές αξιολόγησης ώστε να εξισορροπήσει την λεγόμενη «προκατάληψη των αλγορίθμων» ως προς την αδύναμη κλάση.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Apache Hadoop, Apache Spark, MlLib, Text Mining, Text Classification, Support Vector Machines, Logistic Regression, Grooming, Sexual Predator, HDFS, NLP

# ABSTRACT

The present thesis challenges the probability of sexual offenders identification in chat messages using machine learning algorithms. Text filtering and pre-processing techniques are emphasized and tests with various classification algorithms of Apache Spark's MlLib library will be presented. For the implementation, a large-scale data analysis engine (SPARK) was used in collaboration with an HDFS Hadoop Cluster. The dataset used due to the nature of the problem is quite unbalanced so emphasis is placed on the selection of the most qualified evaluation metric to balance the so-called "algorithm bias" towards the weak class.

# ΠΕΡΙΕΧΟΜΕΝΑ

## ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ ΚΑΙ ΠΙΝΑΚΩΝ

# 1. Εισαγωγή

Σε αυτό το κεφάλαιο αναλύεται το αντικείμενο της διπλωματικής εργασίας.

## 1.1 Συνοπτική Περιγραφή

Η παρούσα διπλωματική εργασία ασχολείται με το πρόβλημα της αποτελεσματικής ανίχνευσης περιπτώσεων σεξουαλικής εκμετάλλευσης ανηλίκων μεταξύ χρηστών στα μέσα κοινωνικής δικτύωσης χρησιμοποιώντας ως δεδομένα τα κείμενα που ανταλλάσσουν μεταξύ τους. Η συγκεκριμένη προσέγγιση εστιάζει στην μεθοδολογία και τα βήματα επίλυσης του προβλήματος, στις τεχνικές προεπεξεργασίας και φιλτραρίσματος κειμένου, στην δημιουργία χαρακτηριστικών (features), καθώς και στις δοκιμές και σύγκριση διαφόρων αλγόριθμων ταξινόμησης. Η υλοποίηση γίνεται σε Apache Spark με χρήση της βιβλιοθήκης για μηχανική μάθηση MLlib .

Η εφαρμογή μηχανικής μάθησης για την επίλυση προβλημάτων σχετικά με μοτίβα γραπτού λόγου μπορεί να προσφέρει αξιόπιστα μοντέλα πρόβλεψης συμπεριφοράς, ωστόσο η φύση του συγκεκριμένου προβλήματος καθιστά αναγκαία την ανθρώπινη παρέμβαση ως προς την εξακρίβωση αν τελικά μια συζήτηση μεταξύ δυο χρηστών θεωρείται περιστατικό ή και προσπάθεια σεξουαλικής εκμετάλλευσης. Γίνεται λοιπόν έτσι εύκολα αντιληπτό πως η συγκεκριμένη έρευνα παρουσιάζει μια μεθοδολογία που δύναται να εφαρμοστεί στην πρόληψη και αποτροπή πιθανών τέτοιων περιστατικών και όχι στην κατηγοριοποίηση τους ως αξιόποινες ή μη πράξεις.

# 2. Θεωρητικό υπόβαθρο

Σε αυτό το κεφάλαιο περιγράφονται έννοιες και τεχνολογίες που χρησιμοποιήθηκαν στην εργασία, οι οποίες είναι η βάση για την κατανόηση του έργου.

## 2.1 Author Profiling (Προφίλ Συγγραφέα)

Με τον όρο “Author Profiling” στο πεδίο της μηχανικής μάθησης αναφερόμαστε στην ανάλυση του γραπτού λόγου με υπολογιστικά μέσα με στόχο την εύρεση διαφόρων χαρακτηριστικών που προκύπτουν από το περιεχόμενο των κειμένων, το συντακτικό και γραμματικό ύφος καθώς και το λεξιλόγιο του συγγραφέα. Τέτοια χαρακτηριστικά μπορεί να είναι η ηλικία, το φύλο, η εθνικότητα, το μορφωτικό επίπεδο, η ανάλυση συναισθήματος, το επάγγελμα κ.α. Το author profiling έχει εφαρμογές σε διάφορα πεδία μελέτης και υπάρχουν περιπτώσεις για παράδειγμα στον κλάδο της εγκληματολογίας όπου ο στόχος είναι ακόμα και η πιθανή ταυτοποίηση του συγγραφέα. Η ταχεία ανάπτυξη του διαδικτύου καθώς και η προσβασιμότητά του από όλους έχουν συμβάλει θετικά στην ανάπτυξη του “author profiling” αφενός γιατί πλέον υπάρχει τεράστιο υλικό ανάλυσης (blogs, chatrooms, Social media, forums, email), αφετέρου γιατί αποτελεί πλέον βασικό εργαλείο ανάπτυξης, συνυφασμένο με το marketing σε κλάδους και βιομηχανίες όπως το εμπόριο, ο τουρισμός και η παροχή υπηρεσιών.

## 2.2 Σεξουαλική Παρενόχληση Ανηλίκων

Με τον αγγλικό όρο «grooming» αναφερόμαστε στο σύνολο των συμπεριφορών που εφαρμόζονται με σκοπό την προετοιμασία της πράξης που συνιστά σεξουαλική κακοποίηση ανηλίκου. Συνήθως τέτοιες συμπεριφορές δεν αναγνωρίζονται εύκολα πριν την πράξη της κακοποίησης καθώς ο δράστης καταφέρνει με επιτυχία να χειραγωγήσει το θύμα, την οικογένεια και την κοινότητα ώστε να κρύψει τις προθέσεις του και να αποφύγει τον εντοπισμό. Σύμφωνα με τους Craven, Brown και Gilchrist (1) στον ορισμό που αποδίδουν στο grooming αναφέρονται σε τρεις συγκεκριμένους στόχους που εφαρμόζονται από τον δράστη οι οποίοι είναι α)Απόκτηση πρόσβασης στο παιδί, β) κερδίζοντας την συμμόρφωση του και γ) δημιουργία και διατήρηση ενός περιβάλλοντος μυστικότητας. Οι στόχοι αυτοί εξυπηρετούν όχι μόνο την τέλεση της καταχρηστικής πράξης αλλά και να χρησιμοποιηθούν αργότερα ως αιτιολόγηση ή και άρνηση των πράξεων τους.

Οι δράστες – αναφέρονται και σαν predators – περιγράφονται ως ευγενικοί, γοητευτικοί και πρόθυμοι να βοηθήσουν. Επίσης πολλές φορές έχουν άμεση σχέση

με το οικογενειακό περιβάλλον του παιδιού και έχουν την πλήρη εμπιστοσύνη της οικογένειας. (2)

### 2.2.1 Στάδια Σεξουαλικής Παρενόχλησης

Η σχετική βιβλιογραφία για την σεξουαλική κακοποίηση υποδηλώνει πως η διαδικασία του grooming περιλαμβάνει μία σειρά από στάδια που εφαρμόζονται από τον δράστη στο θύμα. (3) (4) Εδώ πρέπει να τονιστεί πως είναι πολύ δύσκολο να ορίσουμε με ακρίβεια μια τέτοια διαδικασία καθώς έχει αποδειχθεί, ερευνώντας ήδη υπάρχουσες κακοποιητικές συμπεριφορές πως συναντάται μεγάλη απόκλιση ανάλογα με τα εμπλεκόμενα άτομα. (Whittle, Hamilton-Giachritsis, and Beech 2015). Ωστόσο, έχουν υπάρξει πολλές βιβλιογραφικές αναφορές που παρουσιάζουν κοινά συμπεριφορικά πρότυπα στις διαδικασίες της επιλογής του θύματος, στην απόκτηση πρόσβασης, στην δημιουργία κλίματος εμπιστοσύνης και στην απευαισθητοποίηση του θύματος σε ενέργειες σωματικής επαφής (5). Τα παραπάνω πρότυπα περιγράφουν λίγο αφηρημένα και γενικά τα στάδια του grooming, μιας και μπορούν να εφαρμοστούν και σε άλλες περιπτώσεις όπου δεν υπάρχει ένδειξη εκμετάλλευσης.

Το safechild.org (6)θέλοντας να προσδιορίζει μοναδικά το “grooming” προσθέτει κάποια ακόμα συμπεριφορικά στάδια όπως περιγράφονται παρακάτω.

- Επιλογή του θύματος
- Δημιουργία κλίματος εμπιστοσύνης και ευκαιρίες πρόσβασης
- Ικανοποίηση αναγκών
- Απομόνωση
- Μυστικότητα
- Έναρξη σεξουαλικής επαφής
- Έλεγχος της σχέσης

#### *Επιλογή του θύματος*

Το πρώτο στάδιο αναφέρεται στην επιλογή του θύματος που μπορεί να βασίζεται σε εξωτερικά χαρακτηριστικά, στην ηλικία, στον τρόπο ντυσίματος, στην ευκολία πρόσβασης ή ακόμα και σε οικογενειακές καταστάσεις όπως παιδιά μονογονεϊκών οικογενειών, καθώς αυτά τα παιδιά ενδέχεται να έχουν λιγότερη επίβλεψη από τον γονέα ή ακόμα ο γονέας να βασίζεται σε τρίτους για την άσκηση της γονικής

μέριμνας. Επίσης τα παιδιά θύματα ενδοοικογενειακής βίας ή με γονείς εξαρτημένους σε ναρκωτικές ουσίες έχουν μεγάλο κίνδυνο να τα προσεγγίσει κάποιος predator, καθώς αυτές οι καταστάσεις επίσης οδηγούν σε μειωμένη γονική επίβλεψη. Τέλος ο δράστης μπορεί να αναζητά ανήλικους με χαμηλή αυτοεκτίμηση, χαμηλή αυτοπεποίθηση, ανασφάλεια ή αφέλεια. Συνήθως αυτά τα παιδιά απομονώνονται από το κοινωνικό περίγυρο και είναι πιθανό να εμπιστευτούν κάποιον ξένο που προσφέρει αποδοχή. (Williams, Elliott, and Beech 2013). Το πρώτο στάδιο είναι μια πολύ σημαντική στρατηγικά σχεδιασμένη διαδικασία, που σηματοδοτεί την έναρξη μιας προσπάθειας παιδικής εκμετάλλευσης.

***Δημιουργία κλίματος εμπιστοσύνης και ευκαιρίες πρόσβασης***

Σε αυτό το στάδιο ο predator παρατηρεί το υποψήφιο θύμα και αξιολογεί τα τρωτά του σημεία ώστε να σχεδιάσει την τακτική που θα προσεγγίσει το παιδί με σκοπό να κερδίσει την εμπιστοσύνη του και να δημιουργήσει ευκαιρίες συνάντησης. Συχνά ο δράστης θα προσφέρει κατανόηση, ιδιαίτερη προσοχή και συμπόνια για την επίτευξη του στόχου του. Στο διαδίκτυο αυτό το στάδιο επιτυγχάνεται πιο αβίαστα καθώς ο δράστης μπορεί εύκολα να κρύψει την ταυτότητα του και να προσεγγίσει το πιθανό θύμα, χωρίς να είναι απαραίτητη η φυσική του παρουσία. Επίσης οι ανήλικοι τείνουν να εμπιστεύονται και να εκφράζονται πιο εύκολα σε έναν άγνωστο στο internet χωρίς τον φόβο της κριτικής ή της τιμωρίας που ενδέχεται να υπάρχει στο οικείο περιβάλλον τους. Έτσι το ψηφιακό μέσο αποκτά διττό ρόλο, αυτόν της ψηφιακής επικοινωνίας και του ψηφιακού κρησφύγετου, όπου οι “ένοικοι” μπορούν να έχουν πρόσβαση χωρίς κανέναν έλεγχο από τρίτους.

***Ικανοποίηση αναγκών***

Ο predator θα προσπαθήσει να χειραγωγήσει την σχέση με το πιθανό θύμα ώστε να φαίνεται πως είναι ο μόνος που καταλαβαίνει και ικανοποιεί τις ανάγκες του με κάποιο συγκεκριμένο τρόπο. Αντίστοιχα, θα προβάλει δικά του θέματα και προβλήματα παρουσιασμένα έτσι ώστε να πείσει τον ανήλικο πως μόνο αυτός μπορεί να καταλάβει.

***Απομόνωση***

Η απομόνωση μπορεί να οριστεί ως η προσπάθεια απομάκρυνσης του παιδιού από το περιβάλλον του, ώστε ο δράστης να αποκτήσει αποκλειστική πρόσβαση στο παιδί χωρίς την παρουσία τρίτων. Πέρα από την φυσική απομόνωση υπάρχει και η συναισθηματική, η οποία εργαλειοποείται συνήθως στο διαδίκτυο και έχει να κάνει με την δημιουργία τέτοιων συναισθημάτων αποτροπής και εμπάθειας προς το οικείο περιβάλλον του παιδιού, ώστε να αποθαρρυνθεί οποιαδήποτε προσπάθεια ουσιαστικής επικοινωνίας προς τους γονείς.

***Μυστικότητα***

Η δημιουργία μυστικότητας στην αρχή χρησιμοποιείται για την ενίσχυση της σχέσης του predator με το θύμα, αλλά αργότερα μπορεί να χρησιμοποιηθεί ως μέσο χειραγώγησης. Για την διατήρηση της μυστικότητας ο δράστης ενδέχεται να απειλήσει το θύμα με αποκάλυψη – δημοσιοποίηση, ευαίσθητου περιεχομένου, σωματική βλάβη στο ίδιο ή σε αγαπημένα του πρόσωπα ή δημιουργώντας ενοχές για πράξεις μη αποδεκτές από την κοινωνία κλπ.

***Έναρξη σεξουαλικής επαφής***

Σε αυτό το στάδιο ο δράστης ξεκινά την απευαισθητοποίηση του παιδιού στο άγγιγμα ξεκινώντας με απλά φαινομενικά αθώα αγγίγματα, όπως ένα χτύπημα στον ώμο, τα οποία σταδιακά μετατρέπονται σε σεξουαλικά. Στο διαδίκτυο η παραπάνω κλιμάκωση επιτυγχάνεται με τον δράστη να ξεκινάει συζητήσεις σεξουαλικού περιεχομένου, ερωτικών προτιμήσεων και ανταλλαγής ψηφιακού υλικού μέχρι να καταρρίψει όποιες αναστολές έχει το παιδί.

***Έλεγχος της σχέσης***

Οι δράστες βασίζονται στην μυστικότητα της σχέσης για να διασφαλίσουν πως το παιδί δεν θα αποκαλύψει την κακοποίηση. Τα παιδιά τις περισσότερες φορές φοβούνται να αποκαλύψουν μια τέτοια κατάσταση επειδή έχουν πειστεί πως κανένας δεν θα τους πιστέψει, νοιώθουν ντροπή ή φοβούνται για την ατομική τους ασφάλεια. Ο δράστης ασκεί τον έλεγχο της σχέσης με συχνές απειλές και εκφοβισμούς.

Τα παραπάνω στάδια παρουσιάζονται με μικρές αλλαγές μεταξύ διάφορων ερευνών, ωστόσο έχουν περισσότερες ομοιότητες στην βάση τους παρά διαφορές και δύναται να χρησιμοποιηθούν σαν ένα μοντέλο ορισμού του grooming με σκοπό την πιθανή ταυτοποίηση μιας τέτοιας συμπεριφοράς.

## 2.3 Apache Hadoop

Το Apache Hadoop είναι ένα σύνολο από βιβλιοθήκες λογισμικού ανοικτού κώδικα, που χρησιμοποιείται σε εφαρμογές ανάλυσης δεδομένων μεγάλης κλίμακας (Big Data). Το Hadoop αξιοποιεί χαμηλού κόστους υπολογιστικούς πόρους προσφέροντας κατανεμημένη επεξεργασία και αποθήκευση δεδομένων σε συνδυασμό με εξαιρετικές επιδόσεις. Βασικά χαρακτηριστικά του είναι η κλιμακωτή αρχιτεκτονική, που συμβάλει στην εύκολη επεκτασιμότητά του από μερικούς κόμβους σε χιλιάδες άλλους και η άμεση ανίχνευση και αποκατάσταση αστοχιών σε επίπεδο εφαρμογής, εξασφαλίζοντας υψηλή διαθεσιμότητα. Το Hadoop ως ολοκληρωμένο σύστημα αναλύεται σε τέσσερις κύριες συνιστώσες hadoop common, hdfs, map-reduce και yarn οι οποίες περιγράφονται παρακάτω.

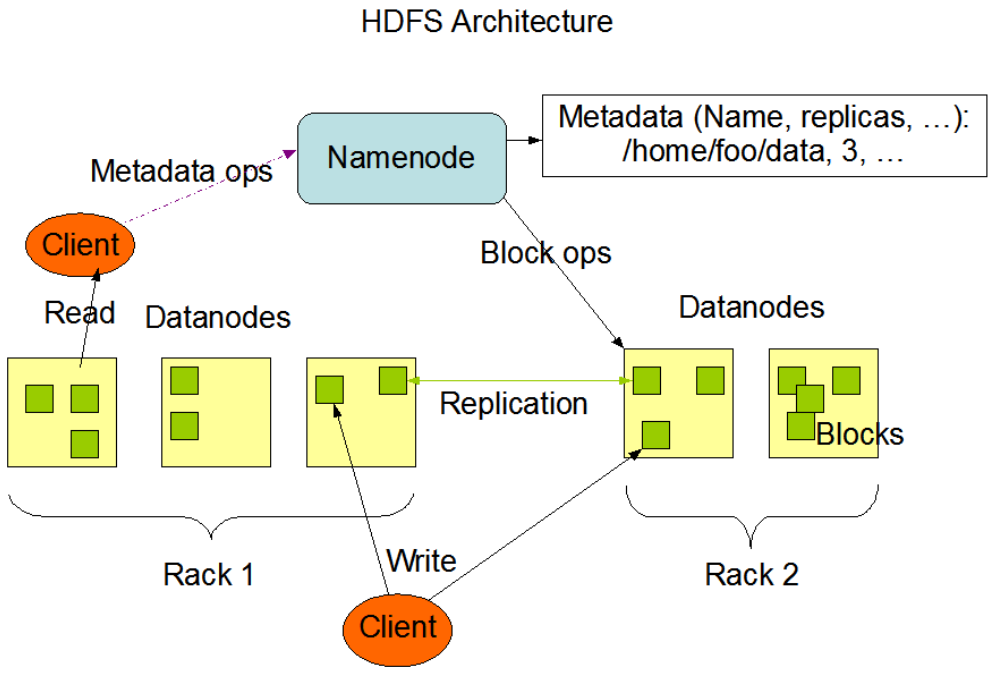
### *Hadoop Common*

Το HADOOP Common γνωστό και ως HADOOP Core μπορεί να χαρακτηριστεί ως ο πυρήνας του HADOOP όπου περιλαμβάνει όλες τις JAVA βιβλιοθήκες, που είναι απαραίτητες για την υποστήριξη τον υπόλοιπων συνιστωσών, hdfs, map-reduce και yarn.

### *Hdfs*

Το Hadoop χρησιμοποιεί το κατανεμημένο σύστημα αρχείων HDFS (Hadoop Distributed File System) για την επικοινωνία των δεδομένων μεταξύ τους. Η αρχιτεκτονική του HDFS ορίζεται ως master/slave. Δηλαδή ένας οι περισσότεροι κόμβοι του cluster έχουν τον ρόλο του master και διαχειρίζονται την τοποθεσία αποθήκευσης των δεδομένων καθώς και τα αιτήματα πρόσβασης σε αυτά και οι slaves στους οποίους αποθηκεύονται τοπικά τα δεδομένα, εκτελούν τις ζητούμενες διεργασίες και ενημερώνουν τους masters για τα νέα δεδομένα που προέκυψαν από τις διεργασίες. Στην περίπτωση του HDFS ως master αναφερόμαστε στους Namenodes και ως slaves στους Datanodes. Η λογική του HDFS είναι πως είναι προτιμότερο να φέρουμε την επεξεργασία κοντά στα δεδομένα και όχι όπως στις

παραδοσιακές αρχιτεκτονικές συστημάτων, όπου τα δεδομένα μεταφέρονται σε ξεχωριστούς κόμβους προς επεξεργασία και τα αποτελέσματα αποθηκεύονται πάλι πίσω. Η συγκεκριμένη τεχνική δεν μπορεί να εφαρμοστεί σε δεδομένα μεγάλης κλίμακας, καθώς θα υπερφορτώσει τους πόρους της υποδομής και η απόδοση του συστήματος θα είναι χαμηλή. Στο Hadoop οι διεργασίες διανέμονται και εκτελούνται στους Datanodes για τα δεδομένα που είναι αποθηκευμένα τοπικά σε αυτούς. Έτσι, μεγιστοποιείται η απόδοση του συστήματος και προσφέρεται η επεξεργασία μεγάλου όγκου δεδομένων. Τέλος, ένα ακόμα χαρακτηριστικό γνώρισμα του HDFS είναι η συνεχής διαθεσιμότητα (availability) του συστήματος μέσω της υψηλής ανοχής σε σφάλματα (fault tolerance) . Αυτό επιτυγχάνεται μέσω του replication, όπου τα δεδομένα χωρισμένα σε κομμάτια (blocks) αποθηκεύονται σε περισσότερους από έναν Datanodes, έτσι ώστε σε περίπτωση αποτυχίας κάποιου node να είναι συνεχώς διαθέσιμα. Το μέγεθος των blocks καθώς και ο αριθμός των αντιγράφων (replicas) ορίζονται στις παραμέτρους του HDFS.




*Σχήμα 1 Αρχιτεκτονική HDFS*

### *Map-Reduce*

Το Map-Reduce είναι ένα κατανεμημένης επεξεργασίας προγραμματιστικό μοντέλο, όπου τα δεδομένα προς επεξεργασία χωρίζονται σε δύο φάσεις την MAP και Reduce. Στην φάση MAP όπου τα δεδομένα εκφράζονται - μετατρέπονται σε

ζεύγη κλειδιού-τιμής και στη REDUCE που παίρνει τις πλειάδες δεδομένων που δημιουργήθηκαν από το MAP και τις συνδυάζει σε μικρότερες πλειάδες δεδομένων. Στο στάδιο του reduce που πάντα έπεται του MAP περιέχεται και άλλη μια φάση, το SHUFFLE, το οποίο αναλαμβάνει την ταξινόμηση και διανομή των δεδομένων στου Reducers. Μια διεργασία μπορεί να επαναλαμβάνει πολλές φορές τις παραπάνω φάσεις μέχρι την επίλυση του προβλήματος. Κάθε ολοκληρωμένη φάση MAP-REDUCE αντιστοιχεί σε ένα ολοκληρωμένο στάδιο (stage) της διεργασίας. Το μοντέλο του MAP-REDUCE κλιμακώνει την διαδικασία της επεξεργασίας σε πολλούς υπολογιστικούς κόμβους και το καθιστά ιδανικό για την επεξεργασία μεγάλων όγκων δεδομένων. Πάνω σε αυτό το μοντέλο είναι σχεδιασμένο και το hadoop το οποίο υλοποιεί την παραπάνω λογική με την χρήση του job tracker και του task tracker. Ο Job Tracker βρίσκεται στους master (Namenodes) κόμβους και είναι υπεύθυνος για την ανάθεση των διεργασιών στους slaves nodes (datanodes), καθώς και για την συνεχή επίβλεψή τους. Σε περίπτωση που ο job tracker αποτύχει, τότε όλοι οι υπολογισμοί για τους οποίους είναι υπεύθυνος θα πάψουν. Οι task trackers με την σειρά τους, εκτελούν τις διεργασίες που τους ανατέθηκαν στο μοντέλο του MAP-REDUCE, ενημερώνοντας για τα αποτελέσματα κάθε σταδίου (stage), καθώς και για την κατάσταση λειτουργίας τους με περιοδικές αναφορές πως δεν έχουν καταρρεύσει. Σε περίπτωση κατάρρευσης κάποιου datanode γίνεται αναδιανομή της διεργασίας που του είχε ανατεθεί σε επόμενο διαθέσιμο datanode.

***YARN (Yet An Other Recource Negotiator)***

Το Yarn έχει τον ρόλο του διαχειριστή πόρων (Resource Manager) και του χρονοπρογραμματισμού (scheduler) των εργασιών σε ένα κατανεμημένο σύστημα. Επίσης, είναι υπεύθυνο για την ανακατανομή των εργασιών σε περίπτωση που κάποιος κόμβος αποτύχει. Κάθε κόμβος κοινοποιεί κατά κάποιο τρόπο τους πόρους που μπορεί να διαθέσει και ο recource manager τους αξιοποιεί δυναμικά ανάλογα με τις ανάγκες κάθε εφαρμογής. Θεωρείται μία απο τις πιο σημαντικές συνιστώσες του Hadoop καθώς ο σωστή παραμετροποίηση έχει άμεσο αντίκτυπο στην απόδοση του συστήματος.

## 2.4 Apache Spark

Το Apache Spark είναι μια μηχανή ανοικτού λογισμικού σχεδιασμένη να τρέχει σε συστοιχίες υπολογιστών και προσφέρεται στην κατανεμημένη και παράλληλη επεξεργασία δεδομένων. Είναι υλοποιημένη σε γλώσσα προγραμματισμού Scala και παρέχει υπολογιστικές διεπαφές (APIs) για την διασύνδεσή του με άλλες γλώσσες προγραμματισμού όπως Java, Python και R.Σε αντίθεση με το Hadoop όπου τα ενδιάμεσα αποτελέσματα της επεξεργασίας αποθηκεύονται στους δίσκους των κόμβων, το spark παρέχει την ιδιότητα της αποθήκευσής τους στην κύρια μνήμη του κόμβου. Με αυτή την τεχνική τα δεδομένα είναι άμεσα προσβάσιμα προς επεξεργασία αν χρειαστούν σε μελλοντικούς υπολογισμούς και ο χρόνος απόκρισης του συστήματος μειώνεται στο ελάχιστο δυνατό, καθώς έτσι παρακάμπτεται η χρονοβόρα διαδικασία ανάκτησης από τον δίσκο. Το παραπάνω καθιστά το Spark εξαιρετικά αποδοτικό εργαλείο στην ανάλυση των δεδομένων ειδικά σε εφαρμογές με ανάγκες άμεσης επεξεργασίας σε πραγματικό χρόνο. Επίσης μέσω της συνιστώσας YARN που περιγράφηκε παραπάνω μπορεί να αναπτυχθεί σε περιβάλλον Hadoop σε επίπεδο αποθήκευσης δεδομένων συνδυάζοντας τις υψηλές επιδόσεις του με την υψηλή διαθεσιμότητα και την ανοχή σφαλμάτων του HDFS. Τέλος το Apache Spark ενσωματώνει στον πυρήνα του ένα σύνολο από διάφορα εργαλεία τα οποία μπορούν να χρησιμοποιηθούν μεμονωμένα η και συνδυαστικά μεταξύ τους καλύπτοντας ένα ευρύ φάσμα αναλύσεων. Τα εργαλεία αυτά συνθέτουν το «οικοσύστημα του Spark» όπως χαρακτηριστικά ονομάζεται και αποτελείται από τον πυρήνα του και τα spark SQL, Spark Streaming, MLlib και GraphX τα οποία περιγράφονται παρακάτω.

### 2.4.1 Spark RDD

Το Resilient Distributed Dataset (RDD) όπως περιγράφει και η ονομασία του προσφέρει μια αμετάβλητη και κατανεμημένη συλλογή δεδομένων οποιαδήποτε τύπου διαθέσιμα προς παράλληλη επεξεργασία. Τα δεδομένα που συνθέτουν ένα RDD μπορεί να προέρχονται από διαφορετικούς κόμβους, υποστηρίζουν τις γλώσσες προγραμματισμού Python, Scala, Java και επιπλέον δίνουν την δυνατότητα στους χρήστες να ορίσουν δικές τους κλάσεις ανάλογα με τις ανάγκες τους. Κάθε εργασία στο Spark ξεκινάει με ένα RDD το οποίο μπορεί να δημιουργηθεί είτε με τον παραλληλισμό μιας υπάρχουσας πηγής δεδομένων είτε

από τον μετασχηματισμό ενός προ-υπάρχοντος. Οι μόνες ενέργειες που επιτρέπονται στα RDD είναι αυτές του μετασχηματισμού(transformations) και υπολογισμού (actions). Οι υπολογισμοί (actions) αφορούν όλες τις εντολές εκτέλεσης σε ένα RDD με σκοπό την παραγωγή κάποιου αποτελέσματος. Οι μετασχηματισμοί αποτελούν στην ουσία ένα σύνολο ενεργειών που πραγματοποιούνται πάνω σε ένα υπάρχον RDD και μπορούν να οδηγήσουν στην δημιουργία ενός νέου. Επίσης βασικό γνώρισμα τους είναι ότι χρησιμοποιούν μια τεχνική που ονομάζεται lazy evaluation υλοποιούμενη από τον μηχανισμό DAG(Directed Acyclic Graph) όπου οι αλλαγές δεν πραγματοποιούνται άμεσα αλλά αποθηκεύονται σαν εντολές επεξεργασίας και εκτελούνται σειριακά όταν ζητηθεί η εκτέλεση κάποιου υπολογισμού (action). Με τον DAG επιτυγχάνεται η μέγιστη αποτελεσματικότητα εκτέλεσης της σειράς των εντολών μετασχηματισμού και χρησιμοποιείται επίσης ως μηχανισμός ανάκαμψης σε περίπτωση που κάποιος κόμβος αποτύχει. Δηλαδή σε περίπτωση αποτυχίας κάποιου κόμβου το Spark αναθέτει το partition σε νέο κόμβο και μέσω του DAG που έχει αποθηκεύσει την σειρά των μετασχηματισμών που ζητήθηκαν να υπολογιστούν εκ νέου τα αποτελέσματα.

Η αποθήκευση και ανάκτηση των δεδομένων προς επεξεργασία στην κύρια μνήμη κατά την διάρκεια μια διεργασίας είναι όπως αναφέρθηκε και παραπάνω το σημαντικό πλεονέκτημα του Apache Spark.Η διαδικασία αυτή ονομάζεται caching και υλοποιείται με δύο μεθόδους την persist() και την cache(). Με την persist() ορίζει ο χρήστης το μέσο αποθήκευσης ενώ στην cache το RDD αποθηκεύεται εξ ορισμού στην κύρια μνήμη. Επειδή όμως η κύρια μνήμη κάθε κόμβου αποτελείται από ένα πεπερασμένο μέγεθος δεν είναι δυνατή η ταυτόχρονη αποθήκευση όλων των RDD σε αυτή. Γιαυτό το λόγο το Apache Spark εφαρμόζει αυτόματα την πολιτική Least Recently Used (LRU), που σύμφωνα με αυτή τα partitions που χρησιμοποιούνται λιγότερο μεταφέρονται από την κύρια μνήμη στον σκληρό δίσκο. Τέλος, παρέχεται η δυνατότητα στον χρήστη δυναμικά να μεταφέρει κάποιο RDD στον δίσκο με την μέθοδο unpersist().

### 2.4.2 Spark Dataframe

Το Dataframe είναι μια δομή δεδομένων που οργανώνει τα δεδομένα σε έναν πίνακα δύο διαστάσεων από γραμμές και στήλες. Χρησιμοποιούνται πάρα πολύ σε

συστήματα ανάλυσης καθώς προσφέρουν μεγάλη ευελιξία στην αποθήκευση αλλά και την επεξεργασία των δεδομένων. Κάθε Dataframe περιέχει ένα σχήμα που προσδιορίζει το όνομα και τον τύπο κάθε στήλης. Τα Spark Dataframes μπορούν να ορίσουν κοινούς τύπους δεδομένων στις στήλες τους όπως Strings και Integers αλλά έχουν και δικούς τους που σχετίζονται αποκλειστικά με το Spark όπως τις Structs. Οι structs είναι λίστες από αντικείμενα τα οποία μπορούν να περιλαμβάνουν εμφωλευμένες στήλες. Αναλογικά τα Spark Dataframes μπορούν να παρομοιαστούν με σχεσιακές βάσης δεδομένων με μία μεγάλη διαφορά. Μια βάση δεδομένων συνήθως βρίσκεται σε μια συγκεκριμένη τοποθεσία ενώ ένα Dataframe βρίσκεται σε διάφορα σημεία μιας κατανεμημένης συστοιχίας υπολογιστών.

### 2.4.3 Οικοσύστημα Spark

***Spark Core***

Το Spark Core είναι ο πυρήνας του Spark και αποτελεί την βάση όλων των υποσυστημάτων που θα αναλύσουμε παρακάτω. Στις βασικές του λειτουργίες είναι ο χρονοπρογραμματισμός  η κατανομή και η επίβλεψη των εργασιών, η διαχείριση μνήμης, η αλληλεπίδραση με  άλλα συστήματα και η διαχείριση των αστοχιών.

***Spark SQL***

Το Spark έχει δυνατότητα επεξεργασίας ημιδομημένων δεδομένων από πηγές όπως csv, JSON κ.α. υποστηρίζοντας ερωτήματα (queries) αντίστοιχα με αυτά των παραδοσιακών σχεσιακών βάσεων δεδομένων. Μέσω του υποσυστήματος SPARK SQL μπορεί να δημιουργηθεί ένα ειδικού τύπου RDD με το όνομα Dataframe είτε από μία πηγή δεδομένων (Dataset) είτε από την μετατροπή ενός RDD. Τα Dataframes θυμίζουν πίνακες σε βάσεις δεδομένων όπου οι εγγραφές τους αναπαρίστανται με την μορφή στηλών και γραμμών όπου κάθε στήλη μπορεί να εμπεριέχει διακριτού τύπου δεδομένα. Στα Dataframes μπορούν να εφαρμοστούν εντολές αντίστοιχες της SQL όπως Select, Join, Delete κλπ. Επίσης, είναι δυνατή η μετατροπή ενός RDD σε Dataframe και αντίστροφα.

***Spark Streaming***

Το Spark Streaming χρησιμοποιείται στην αποθήκευση και επεξεργασία πραγματικού χρόνου ροών δεδομένων. Οι ροές μπορούν ταυτόχρονα να εισέρχονται από διαφορετικές πηγές δεδομένων και να αποθηκεύονται στην κύρια μνήμη κατόπιν διαίρεσης τους σε μικρά υποσύνολα σε μορφή RDD. Έτσι, εξασφαλίζονται όλα τα παραπάνω πλεονεκτήματα που παρέχει η πλατφόρμα (επεκτασιμότητα, ανοχή σε λάθη, διαθεσιμότητα) και σε συνεχής ροής δεδομένα.

***Graphx***

Το GraphX είναι η βιβλιοθήκη του Spark η οποία παρέχει λειτουργίες υποστήριξης και ανάλυσης γραφημάτων.

***MLLib (Machine Learning Library)***

Η συγκεκριμένη βιβλιοθήκη παρέχει μια πλήρη συλλογή αλγορίθμων μηχανικής μάθησης υλοποιημένους στο κατανεμημένο περιβάλλον του Apache Spark. Οι αλγόριθμοι που έχουν υλοποιηθεί μέχρι στιγμής υποστηρίζουν ένα μεγάλο εύρος λειτουργιών σχετικές με την μηχανική μάθηση όπως για παράδειγμα την δημιουργία χαρακτηριστικών (features), το φιλτράρισμα δεδομένων, την συσταδοποίηση (clustering) και την ταξινόμηση (classification). Αν και η ταχύτητα εκτέλεσης των αλγορίθμων εξασφαλίζεται ως ένα μεγάλο βαθμό από τα οφέλη της πλατφόρμας που περιγράφηκαν παραπάνω, η ποιότητα της γνώσης που παράγουν εξαρτάται σημαντικά τόσο από τον συνδυασμό της σωστής χρήσης και παραμετροποίησής τους, όσο και από την ποιότητα των δεδομένων που εισάγουμε. Ειδικότερα σχετικά με την ποιότητα των δεδομένων είναι σύνηθες στην αρχική τους μορφή να περιέχουν πολλές «άχρηστες» πληροφορίες που δεν συνεισφέρουν στο αποτέλεσμα της ανάλυσης - συχνά αναφέρονται και ως «θόρυβος». Ανάλογα με το είδος της εφαρμογής είναι πολύ σημαντικό το «φιλτράρισμα» και το «καθάρισμα» των δεδομένων. Στην παρούσα εργασία θα γίνει χρήση αλγόριθμων της MlliB σε όλα τα στάδια από τον «καθαρισμό» των δεδομένων μέχρι και την παραγωγή ενός μοντέλου πρόβλεψης ταξινόμησης.

## 2.5 Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing)

Το NLP ορίζεται ως ένα σύνολο τεχνικών που εφαρμόζονται ώστε να γίνει η ανθρώπινη γλώσσα κατανοητή σε υπολογιστικά συστήματα. Συνδυάζει την επιστήμη της γλωσσολογίας με την επιστήμη των υπολογιστών για την μελέτη των κανόνων και της δομής του γραπτού και προφορικού λόγου.

Το NLP εφαρμόζεται σε κείμενα που προέρχονται είτε από γραπτό είτε από τον προφορικό λόγο και τα αναλύει σε επίπεδα όπως το συντακτικό, την μορφολογία και την σημασιολογία τους. Για την ανάλυση ακολουθείται συνήθως μια διαδικασία από διαφορετικά βήματα και τεχνικές ανάλογα με το είδος κειμένων που έχουμε και το επιθυμητό αποτέλεσμα.

Ανεξαρτήτως τεχνικών και βημάτων το NLP μπορεί να χωριστεί σε δύο φάσεις: α) Προεπεξεργασία κειμένου και β) εφαρμογή κάποιου αλγόριθμου ταξινόμησης .Στο πρώτο στάδιο της προεπεξεργασίας γίνεται το φιλτράρισμα και ο καθαρισμός των δεδομένων. Το νόημα της προεπεξεργασίας είναι να αφαιρέσει όλα τα δεδομένα που δεν συμβάλουν στην δημιουργία ισχυρών χαρακτηριστικών (features) και να μειώσει το μέγεθος του συνόλου δεδομένων ώστε να βελτιώσει την απόδοση του αλγορίθμου της δεύτερης φάσης. Για την προεπεξεργασία αλλά και για την δημιουργία χαρακτηριστικών χρησιμοποιούνται διάφορες τεχνικές κάποιες από αυτές περιγράφονται συνοπτικά παρακάτω.

### *Bag Of Words (BoW)*

Με τον όρο bag of words αναφερόμαστε στη μέθοδο δημιουργίας ενός λεξικού (bag of words - τσάντα με λέξεις) αθροίζοντας τον αριθμό εμφάνισης κάθε λέξης σε ένα κείμενο, χωρίς να λαμβάνονται υπόψιν άλλα χαρακτηριστικά όπως η δομή ή η σειρά εμφάνισης. Αυτή η μέθοδος είναι πολύ χρήσιμη, καθώς μας δίνει την δυνατότητα να εντοπίσουμε ομοιότητες μεταξύ των κειμένων βασισμένες στις λέξεις που εμπεριέχουν και κατ’ επέκταση να προσδιορίσουμε το περιεχόμενο ή την θεματολογία τους. Συνήθως πριν της δημιουργίας ενός λεξικού BoW, προηγούνται διάφορες μέθοδοι φιλτραρίσματος και καθαρισμού κειμένου. Για παράδειγμα, αφαίρεση άρθρων, σημείων στίξης, επιρρημάτων, κ.α. που συνήθως δεν προσθέτουν ουσιαστικό νόημα στο θέμα του κειμένου. Τέλος, είναι πολύ συνηθισμένη η δημιουργία και εφαρμογή ενός υπάρχοντος λεξιλογίου ανάλογα με τα χαρακτηριστικά που θέλουμε να αποτυπώσουμε. Για παράδειγμα σε μια δυαδική

ταξινόμηση κειμένου για το αν η θεματολογία είναι τα αθλητικά ή όχι, μπορεί να χρησιμοποιηθεί ένα προσαρμοσμένο λεξιλόγιο με αθλητικούς όρους, όπου κάθε όρος θα αρχικοποιείται με την τιμή 0 (μηδέν).Κάθε φορά που θα υπάρχει ταύτιση μια λέξης του κειμένου με όρο του λεξιλογίου η τιμή του θα αυξάνεται κατά ένα μέχρι n, όπου n ένας ακέραιος αριθμός που υποδεικνύει την συχνότητα του όρου σε ένα κείμενο (TF – Term Frequency).

***TF-IDF (Term Frequency – Inverse Document Frequency)***

Το TF-IDF είναι μια πολύ συνηθισμένη τεχνική δημιουργίας χαρακτηριστικών και χρησιμοποιείται στην αποτύπωση της βαρύτητας μια λέξης σε μια συλλογή κειμένων. Η τιμή TF-IDF αυξάνεται αναλογικά με τον αριθμό που εμφανίζεται μια λέξη σε ένα κείμενο και αντισταθμίζεται με τον αριθμό των κειμένων που περιέχει η συλλογή δεδομένων. Είναι το αποτέλεσμα του συνδυασμού μεταξύ των στατιστικών όρων TF – Term Frequency και IDF Inverse Document Frequency. Με τον όρο TF αποτυπώνεται η βαρύτητα ενός όρου σε σχέση με την συχνότητα που εμφανίζεται σε ένα κείμενο.

*TF(t) = Αριθμός εμφανίσεων μια λέξης (t) σε ένα κείμενο / Σύνολο λέξεων κειμένου (1)*

Σε αντίθεση με το Term – Frequency το Inverse – document - frequency υπολογίζει την συχνότητα εμφάνισης μίας λέξης ενός συνόλου κειμένων, δηλαδή πόσο σπάνια ή συνηθισμένη είναι μία λέξη συγκριτικά με τον αριθμό εμφανίσεων της σε άλλα κείμενα της συλλογής. Όσο πιο κοντά στο 0 είναι η τιμή της τόσο πιο συνηθισμένη η λέξη. Πρακτικά δίνει μεγαλύτερη βαρύτητα στους όρους που εμφανίζονται λιγότερο αποφεύγοντας έτσι τον συνυπολογισμό όρων, όπως τα άρθρα, τα οποία συναντώνται πολύ συχνά στον γραπτό λόγο αλλά δεν προσφέρουν ουσιαστικά στην περιγραφή της θεματολογίας.

*IDF(t) = log_e (Σύνολο κειμένων / Σύνολο κειμένων που περιέχουν την λέξη (t)) (2)*

Πολλαπλασιάζοντας τα παραπάνω (1) και (2) προκύπτει η τιμή του TF-IDF.

### *Tokenization*

Tokenization είναι η διαδικασία μετατροπής κειμένου σε μικρότερα κομμάτια (προτάσεις, λέξεις, χαρακτήρες) με το όνομα tokens. Για παράδειγμα

*Αρχικό κείμενο -> "I Like Swimming In the Ocean"*
*Tokenized κείμενο "I", "Like", "Swimming", "In", "the", "Ocean"*

### *Stop-word-removal*

Στο στάδιο του Stop-word-Removal αφαιρούνται κοινές λέξεις, οι οποίες δεν προσφέρουν κάποιο νόημα στο κείμενο που απομένει. Για παράδειγμα

*Αρχικό κείμενο -> "I Like Swimming In the Ocean"*
*Stop-Word-Removed -> "Like, Swimming, Ocean"*

### *Stemming*

Το stemming μειώνει τον αριθμό των λέξεων ενός συνόλου δεδομένων, μετατρέποντας κάθε λέξη στη ρίζα της αφαιρώντας όλες τις παράγωγες. Για παράδειγμα

*Αρχικό κείμενο -> "wait, waited, waiting"*
*Stemmed -> "wait"*

### *Part-of-speech tagging (POS)*

Το POS αναφέρεται στην διαδικασία, όπου οι λέξεις χαρακτηρίζονται βάσει του ρόλου τους στο κείμενο (ουσιαστικό, ρήμα, αντικείμενο κλπ.) Για παράδειγμα

*Αρχικό κείμενο -> "I Like Swimming In the Ocean"*
*POS -> "Αντωνυμία, ρήμα, ρήμα, επίρρημα, άρθρο, ουσιαστικό"*

### *n-grams*

Με τον όρο n-grams αναφερόμαστε σε μία συνεχόμενη σειρά από n λέξεις. Ενα n-gram μεγέθους 1 ονομάζεται unigram, μεγέθους 2 bigram, μεγέθους 3 trigram κοκ. Για παράδειγμα

*Αρχικό κείμενο -> "I Like Swimming In the Ocean"*

*Bigrams -> "I Like", Like Swimming", "Swimming In", "In the", "the Ocean"*
*Trigrams -> "I Like Swimming", "Like Swimming In", "Swimming In the", "in the Ocean"*

Για να υπολογιστεί ο αριθμός των n-grams χρησιμοποιείται ο παρακάτω τύπος
***n-grams = X - (N-1)***

Όπου X ο αριθμός των λέξεων του κειμένου και N ο αριθμός του μεγέθους της σειράς λέξεων.

Τα n-grams μας δίνουν την δυνατότητα να αποτυπώσουμε φυσικές εκφράσεις λόγου, που μεμονωμένες λέξεις δεν μπορούν να αποδώσουν.

Στο δεύτερο στάδιο του NLP εφαρμόζεται κάποιος αλγόριθμος της επιλογής μας, για την εκπαίδευση του μοντέλου πρόβλεψης χρησιμοποιώντας τα χαρακτηριστικά (features) που δημιουργήθηκαν στην πρώτη φάση.

## 2.6 Ταξινόμηση

Στην μηχανική μάθηση με τον όρο ταξινόμηση αναφερόμαστε στην διαδικασία δημιουργίας μοντέλων πρόβλεψης (predictive models), τα οποία εφαρμόζονται σε σύνολα δεδομένων (datasets), για την κατηγοριοποίηση των εγγραφών τους σε συγκεκριμένες κλάσεις. Ένα απλό παράδειγμα ταξινόμησης είναι η κατηγοριοποίηση των email σε "spams" και "not spam".

Για την δημιουργία των μοντέλων χρησιμοποιούνται διάφοροι αλγόριθμοι ανάλογα με το είδος ταξινόμησης καθώς και του συνόλου των δεδομένων που έχουμε στην διάθεση μας. Οι αλγόριθμοι εκπαιδεύονται πάνω σε ένα συνήθως μεγάλο κομμάτι του συνόλου των δεδομένων -τα δεδομένα εκπαίδευσης- και δημιουργούν συσχετίσεις και μοτίβα ώστε να συνθέσουν το μοντέλο πρόβλεψης. Στη συνέχεια, χρησιμοποιώντας το παραχθέν μοντέλο ταιριάζουμε (fit) το υπόλοιπο κομμάτι του συνόλου τον δεδομένων -δεδομένα δοκιμής- ώστε να υπολογιστούν οι προβλέψεις.

Στο στάδιο της εκπαίδευσης ο αλγόριθμος για την δημιουργία του μοντέλου πρόβλεψης χρησιμοποιεί τα χαρακτηριστικά(features) και τις ετικέτες(labels).

Οι ετικέτες(labels) είναι στην ουσία οι κλάσεις που θέλουμε να ταξινομήσουμε τα δοκιμαστικά δεδομένα. Στο παράδειγμα με τα mails έχουμε δύο ετικέτες «spam» και «no spam».

Τα χαρακτηριστικά(features) είναι στοιχεία των δεδομένων που μπορούν να χρησιμοποιηθούν στην διαφοροποίηση των εγγραφών μεταξύ τους. Η σωστή επιλογή των χαρακτηριστικών συμβάλει άμεσα στην αποτελεσματικότητα του μοντέλου. Επίσης όσο μεγαλύτερο το πλήθος τον χαρακτηριστικών τόσο αυξάνεται οι πολυπλοκότητα. Στο παράδειγμα με τα emails παραπάνω πιθανά χαρακτηριστικά θα μπορούσαν να ήταν ο αποστολέας, το θέμα του email, ο αριθμός που εμφανίζεται κάποια λέξη πχ «προσφορά» ή «διαφήμιση» ή ακόμα και οι ώρες αποστολής. Όλα τα παραπάνω μπορούν να χρησιμοποιηθούν ώστε να διαχωρίσουν ένα mail σε spam ή σε «not spam».

Τόσο τα χαρακτηριστικά(features) όσο και οι ετικέτες (labels) πρέπει να αντιστοιχούνται με αριθμούς ώστε να μπορεί να τα επεξεργαστεί ο αλγόριθμος ταξινόμησης. Για παράδειγμα «spam = 0.0» και «no spam = 0.1»

Τα χαρακτηριστικά του συνόλου των δεδομένων και οι ετικέτες ταξινόμησης καθορίζουν άμεσα το είδος της ταξινόμησης που θέλουμε να εφαρμόσουμε. Παρακάτω παρουσιάζονται οι βασικές μορφές ταξινόμησης :

***Binary Classification –*** Με δύο πιθανές ετικέτες πχ ταξινόμηση email σε spam ή non-spam.

***Multi-Class Classification –*** Περισσότερες από δύο πιθανές ετικέτες όπου κάθε εγγραφή αντιστοιχεί σε μια ετικέτα, για παράδειγμα ταξινόμηση ατόμων σε ηλικιακή ομάδα (10-18, 19-30, 31 – 50, >50). Δηλαδή ένα άτομο που ανήκει στην ηλικιακή ομάδα 10-18 δεν μπορεί να ανήκει σε καμία άλλη.

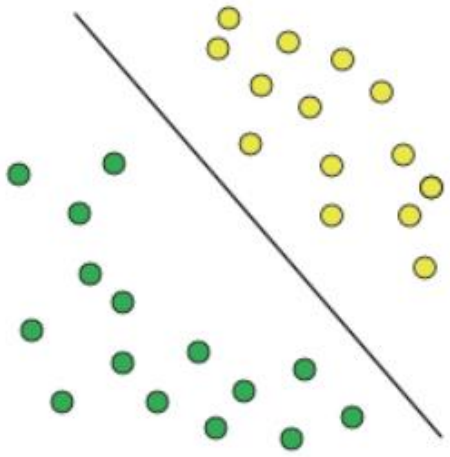
***Multi-Label Classification –*** Περισσότερες από δύο πιθανές ετικέτες πρόβλεψης, όπου κάθε εγγραφή αντιστοιχεί σε περισσότερες από μία ετικέτες. Για παράδειγμα ταξινόμηση είδους ταινιών (Κοινωνική, κωμωδία, θρίλερ, συναισθηματική). Μία ταινία μπορεί να ανήκει στις κατηγορίες συναισθηματική και κωμωδία ταυτόχρονα.

***Imbalanced Classification –*** Συνήθως με τον όρο μη ισορροπημένη ταξινόμηση αναφερόμαστε σε δυαδικού τύπου ταξινόμηση, όπου το σύνολο δεδομένων είναι μη ομοιόμορφα κατανεμημένο μεταξύ δύο κλάσεων-ετικετών. Για παράδειγμα στην

ταξινόμηση ανθρώπων σε δεξιόχειρες και αριστερόχειρες είναι φυσιολογικό τα δεδομένα της κλάσης «δεξιόχειρες» να είναι συντριπτικά περισσότερα από αυτά της κλάσης «αριστερόχειρες» αφού οι αριστερόχειρες αντιπροσωπεύουν μόλις το 10% του πληθυσμού. Αυτό μοιραία οδηγεί και στην λεγόμενη «προκατάληψη των αλγορίθμων ως προς την ασθενέστερη κλάση» όπου ο αλγόριθμος τείνει να ευνοεί την δυνατή κλάση στην ταξινόμηση των εγγραφών με αποτέλεσμα να μεγαλώνει το ποσοστό σφάλματος των προβλέψεων.

### 2.6.1 Support Vector Machine

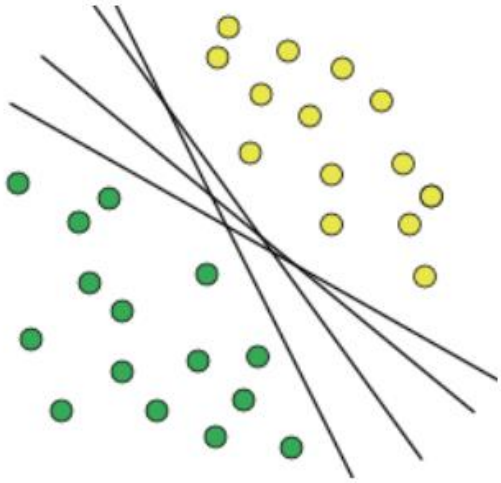
Οι αλγόριθμοι SVM εφαρμόζονται πολύ συχνά σε εφαρμογές μηχανικής μάθησης, όπου το ζητούμενο είναι η ταξινόμηση δεδομένων σε δύο ή περισσότερες κλάσεις. Βασίζονται σε μία απλή και εύκολα κατανοητή ιδέα, η οποία όμως είναι σε θέση να παράγει εξαιρετικά αποτελέσματα από άποψη επιδόσεων. Σκοπός των SVM είναι η εύρεση του βέλτιστου υπερεπιπέδου (hyperlane), που διαχωρίζει τα δεδομένα διαφόρων κλάσεων δημιουργώντας το μέγιστο περιθώριο (margin) μεταξύ υπερεπιπέδου και δεδομένων μια κλάσης. Για την κατανόηση του παραπάνω ακολουθεί μια χαρακτηριστική εικόνα ενός προβλήματος με πλήρως γραμμικά διαχωρίσιμα δεδομένα.



*Σχήμα 2 SVM Γραμμικά διαχωρίσιμο πρόβλημα*

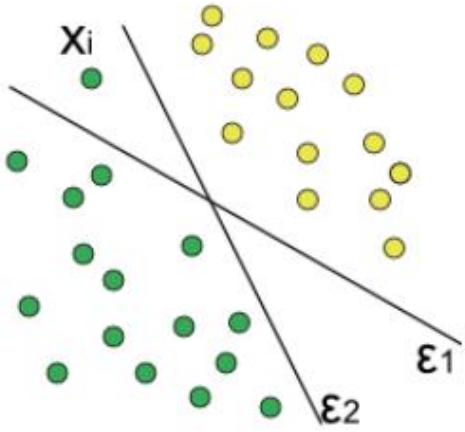
Η περίπτωση ενός πλήρως γραμμικά διαχωρίσιμου προβλήματος είναι αρκετά σπάνια αλλά μας βοηθάει να κατανοήσουμε την ιδέα του αλγορίθμου στην απλή του μορφή. Όπως φαίνεται και στην εικόνα τα δεδομένα ανήκουν σε δύο κλάσεις

(πράσινη και κίτρινη) και μπορούν να διαχωριστούν με μηδενική πιθανότητα σφάλματος με την εισαγωγή μιας ευθείας γραμμής (hyperlane) μεταξύ τους. Επίσης, όπως φαίνεται παρακάτω μπορούν να εισαχθούν περισσότερες από μία γραμμές, οι οποίες και πάλι να επιτυγχάνουν βέλτιστη ταξινόμηση.



*Σχήμα 2 SVM Ευθείες διαχωρισμού*

Στο παράδειγμα παραπάνω ας υποθέσουμε πως οι κλάσεις μπορούν να διαχωριστούν τέλεια με την χρήση δύο ευθείων ε1 και ε2. Εισάγοντας όμως έναν νέο πράσινο δεδομένο Xi παρατηρούμε ότι πλέον η ευθεία ε1 ταξινομεί το δεδομένο στο ημιεπίπεδο της κίτρινης κλάσης το οποίο είναι λάθος.




*Σχήμα 3 SVM Επιλογή Ευθείας διαχωρισμού*

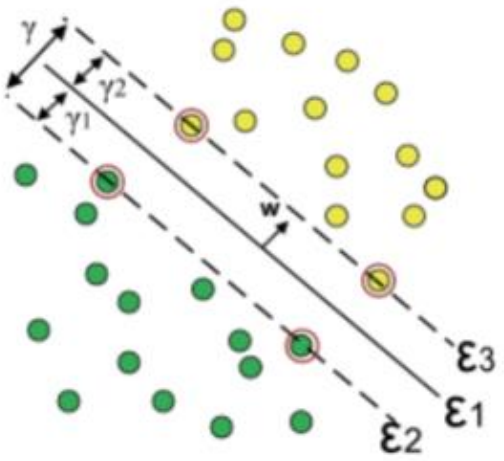
Επομένως επιλέγεται η ευθεία (hyperlane) ε2 καθώς διαχωρίζει τα δεδομένα στις υπάρχουσες κλάσεις με μηδενικό σφάλμα.

**Περιθώριο ταξινόμησης (margin)**

Στο παραπάνω πρόβλημα ας υποθέσουμε πως όλα τα δεδομένα $x_i \in \mathbb{R}^d$ και η ετικέτα κάθε δείγματος $y_i \in \{-1,+1\}$, έτσι το υπερεπίπεδο που διαχωρίζει τις κλάσεις ορίζεται από τον τύπο $w \cdot x + b = 0$, όπου w το διάνυσμα που ορίζει την κλίση του υπερεπιπέδου και b (bias) το διάνυσμα που αντιστοιχεί στην παράλληλη μετατόπιση του υπερεπιπέδου.

Ο βασικός στόχος του αλγορίθμου SVM όπως αναφέρθηκε είναι όχι μόνο να ταξινομεί σωστά τα υπάρχοντα δεδομένα αλλά και τα νέα δεδομένα που θα εισαχθούν. Για την επίτευξη του στόχου αυτού επιλέγονται τα δεδομένα κάθε κλάσης με την ελάχιστη απόσταση από το υπερεπίπεδο διαχωρισμού. Τα δεδομένα αυτά ονομάζονται διανύσματα υποστήριξης (support vectors). Με την σειρά τους τα διανύσματα υποστήριξης ορίζουν δύο νέα υπερεπίπεδα με την ελάχιστη απόσταση από το κεντρικό υπερεπίπεδο διαχωρισμού. Η απόσταση των δύο νέων υπερεπιπέδων από το κεντρικό ονομάζεται περιθώριο ταξινόμησης (margin) και ορίζεται με το σύμβολο γ. Όσο μεγαλύτερη η απόσταση του κεντρικού υπερεπιπέδου από το κοντινότερο δεδομένο κάθε κλάσης τόσο καλύτερη η απόδοση και η ανθεκτικότητα του παραχθέντος μοντέλου πρόβλεψης.

Για καλύτερη κατανόηση ακολουθεί το παρακάτω παράδειγμα όπου έχουμε δύο κλάσεις C1(πράσινα) και C2(κίτρινα) με στοιχεία $x_i \in \mathbb{R}^2$ και ετικέτες δείγματος $y_i \in \{-1,+1\}$.




*Σχήμα 4 SVM Περιθώρια ταξινόμησης*

Η ευθεία ε1 είναι το υπερεπίπεδο ταξινόμησης για την οποία ισχύει ε1:w·x+b=0

Οι ευθείες ε2 & ε3 είναι τα υπερεπίπεδα των διανυσμάτων υποστήριξης, τα οποία διαχωρίζουν τα δεδομένα κάθε κλάσης και ισαπέχουν από την ε1 (γ1 = γ2) και για τα οποία ισχύει

***ε2:w·x+b=−δ & ε3:w·x+b=δ***

Εδώ παρατηρούμε πως σαν διανύσματα υποστήριξης για κάθε κλάση έχουν επιλεγεί αυτά με την μικρότερη απόσταση από το κεντρικό υπερεπίπεδο.

Για χάριν ευκολίας αντικαθιστούμε το δ με 1 και έτσι έχουμε

***ε2:w·x+b=−1 & ε3:w·x+b=1***

Έτσι για τα δεδομένα Xi κάθε κλάσης έχουμε

***w·xi+b≤−1,για xi∈C1 (1)***
***w·xi+b≥1,για xi∈C2 (2)***

Για τα δεδομένα Xi που ανήκούν στην C1 ισχύει ***Yi = -1*** ενώ για τα δεδομένα Xi που ανήκουν στην C2 ισχύει ***Yi = 1***. Πολλαπλασιάζοντας τα δύο μέλη των σχέσεων (1) και (2) με τα Yi = -1 και Yi = 1 έχουμε

***yi(w·xi+b)≥1,για xi∈C1 (3) Και yi(w·xi+b)≥1,για xi∈C2 (4)***

Παρατηρώντας την (3) και (4) διακρίνουμε πως για κάθε δεδομένο Xi των κλάσεων ισχύει

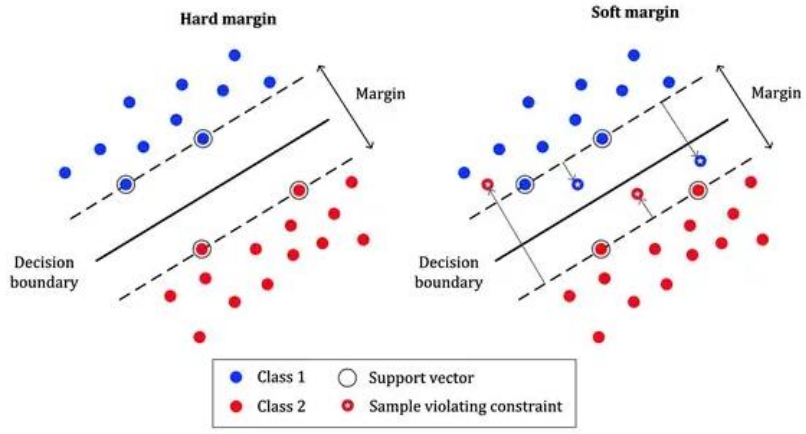
***yi(w·xi+b)≥1,∀xi (5)***

Τα Xi των κλάσεων C1 & C2 για τα οποία ισχύει ***w·xi+b=−1,xi∈C1*** και ***w·xi+b=1,xi∈C2*** είναι τα διανύσματα υποστήριξης της κάθε κλάσης.

…….

Στην περίπτωση που στο πρόβλημα μπορεί να επιτραπεί ένας μικρός αριθμός σφαλμάτων με σκοπό την επίτευξη μεγαλύτερου περιθωρίου (large margin) χρησιμοποιούνται μαλακά περιθώρια (soft margins). Χρησιμοποιώντας

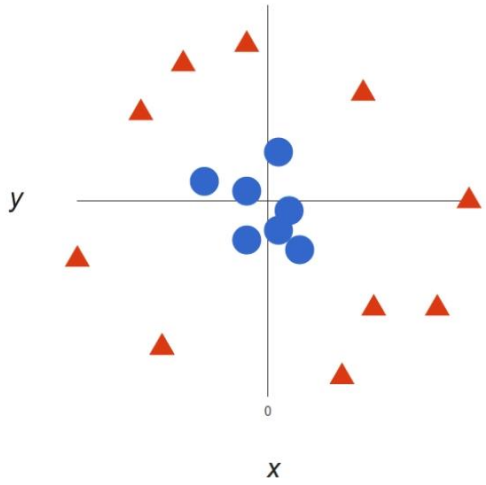
μαλακά περιθώρια επιλέγουμε κάποια δεδομένα -τα οποία αποτελούν ένα πολύ μικρό ποσοστό του δείγματος εκπαίδευσης- να βρίσκονται ανάμεσα στο κεντρικό υπερεπίπεδο και στα βοηθητικά υπερεπίπεδα όπως ορίστηκαν από τα διανύσματα υποστήριξης(Figure 1). Η αποδοχή αυτού του μικρού σφάλματος συμβάλει ουσιαστικά στην καλύτερη απόδοση του αλγορίθμου, καθώς μεγιστοποιεί τα περιθώρια και δημιουργεί πιο ανθεκτικά μοντέλα πρόβλεψης στα νέα δεδομένα που εισάγονται που ενδεχομένως να μην είναι γραμμικά διαχωρίσιμα. Επίσης η επιλογή στενών περιθωρίων (narrow margins) δεν θεωρείται καλή πρακτική, καθώς τα παραχθέντα μοντέλα πρόβλεψης στερούνται ευελιξίας εφόσον δημιουργούνται με πρότυπα αυστηρά προσαρμοζόμενα στα δεδομένα εκπαίδευσης.




*Σχήμα 5 SVM Soft - Hard Margin*

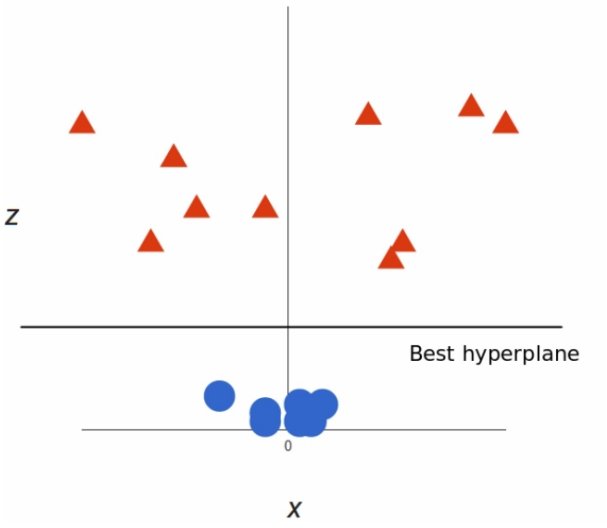
***Μη γραμμικά διαχωρίσιμα δεδομένα***

Τέλος μια πολύ συχνή μορφή προβλημάτων που καλούνται να επιλύσουν οι αλγόριθμοι πρόβλεψης είναι τα μη γραμμικά διαχωρίσιμα δεδομένα όπως φαίνονται παρακάτω.



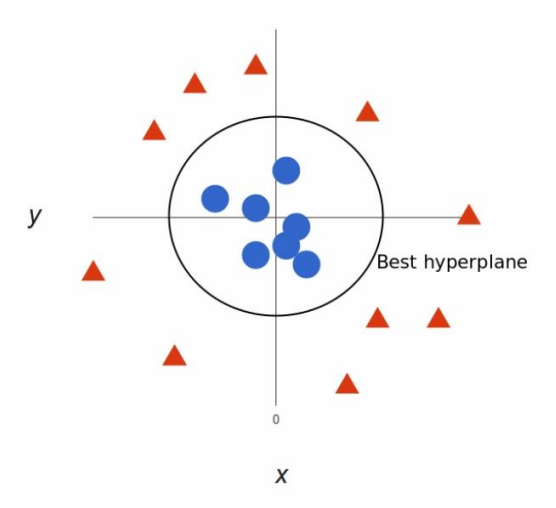

*Σχήμα 6 SVM Μη γραμμικά διαχωρίσιμα δεδομένα*

Είναι κατανοητό από το παραπάνω σχήμα πως δεν μπορούμε να ορίσουμε πάντα κάποιο υπερεπίπεδο που να ταξινομεί τα δεδομένα σε δύο κλάσεις. Για την επίλυση του προβλήματος θα πρέπει να μετασχηματίσουμε τα δεδομένα έτσι ώστε να αναπαρίστανται σε χώρο συντεταγμένων τριών διαστάσεων (X, Y, Z) και έτσι να οριστεί ένα υπερεπίπεδο παράλληλο με τον άξονα x όπως φαίνεται παρακάτω για ένα συγκεκριμένο z (πχ z=1).




*Σχήμα 7 SVM Αναπαράσταση δεδομένων σε χώρο τριών διαστάσεων*

Μετατρέποντας τώρα τα δεδομένα πίσω στις δύο διαστάσεις έχει δημιουργηθεί ένα όριο απόφασης (decision boundary) που είναι ένας κύκλος με περιφέρεια 1 και ταξινομεί τις δύο κλάσεις.

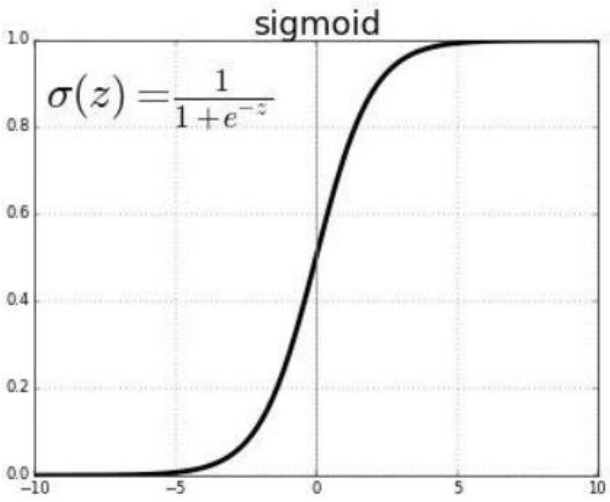



*Σχήμα 8 SVM Όριο Απόφασης - Συνάρτηση πυρήνα*

Η συγκεκριμένη μέθοδος αναφέρεται στην βιβλιογραφία και ως συνάρτηση πυρήνα (kernel trick) και επιτυγχάνεται μαθηματικά με τον τύπο Langrange.

### 2.6.2 Logistic Regression

Η λογιστική παλινδρόμηση είναι ένας στατιστικός αλγόριθμος κατηγοριοποίησης δεδομένων προϋποθέτοντας ότι αυτά ακολουθούν τη σιγμοειδή συνάρτηση. Εξετάζει δηλαδή την πιθανότητα των δεδομένων να ανήκουν σε κάποιο από τα δύο άκρα της σιγμοειδούς καμπύλης 0 ή 1 τα οποία αντιστοιχούν και σε μία κλάση.




*Σχήμα 9 Λογιστική Παλινδρόμηση*

Τα δύο άκρα της σιγμοειδούς καμπύλης τείνουν προς το 0 και το 1 που σημαίνει πως δεν λαμβάνουν ποτέ ακριβώς τις τιμές των άκρων της. Επειδή όμως

αναφερόμαστε σε προβλέψεις μας αρκεί η τιμή να είναι πιο κοντά σε κάποιο από τα δύο άκρα, ώστε να αντιστοιχηθεί και στην ανάλογη κλάση. Για το πόσο κοντά η μακριά είναι κάποια τιμή σε ένα από τα δύο άκρα η σιγμοειδής συνάρτηση ορίζει μια μεταβλητή z ως κατώφλι. Για κάθε τιμή μεγαλύτερη του z δίνεται πιθανότητα 1 και για κάθε τιμή μικρότερη από z δίνεται τιμή 0.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Η λογιστική παλινδρόμηση εφαρμόζεται πολύ συχνά σε εφαρμογές ταξινόμησης και διακρίνεται σε τρεις κατηγορίες. Την δυωνυμική παλινδρόμηση (binomial) όπου υπάρχουν μόνο δύο πιθανά αποτελέσματα, την πολυωνυμική παλινδρόμηση (Multinomial) με τρία ή περισσότερα πιθανά αποτελέσματα και την τακτική παλινδρόμηση όπου μια μεταβλητή μπορεί να λάβει τρεις ή περισσότερους ταξινομημένους τύπους ( θετικό, αρνητικό, ουδέτερο). (7)

### 2.7 Pipelines (Αγωγοί)

Τα Pipelines χρησιμεύουν στον αυτοματισμό της εκτέλεσης των διαδικασιών που ορίζονται ώστε να παραχθεί ένα μοντέλο πρόβλεψης. Οι διαδικασίες αφορούν τα στάδια προεπεξεργασίας των δεδομένων, δημιουργίας χαρακτηριστικών και συσχέτισης με χρήση αλγορίθμων ταξινόμησης.

Ένας αγωγός μπορεί να αποτελείται από n-αριθμό βημάτων μετασχηματισμού (transformers) και συσχέτισης (estimators) δεδομένων, που εκτελούνται διαδοχικά για την εκπαίδευση του μοντέλου. Ως transformers ορίζονται τα βήματα του αγωγού στα οποία με κάποιο τρόπο πραγματοποιείται μετασχηματισμός των δεδομένων. Οι estimators είναι οι αλγόριθμοι οι οποίοι εφαρμόζονται (fit) σε ένα dataframe με σκοπό την παραγωγή ενός transformer.

Εκτελούνται με μια οριζόμενη σειρά και η ροή τους δεν είναι μονόδρομη, αντίθετα είναι κυκλική και επιτρέπει την επανάληψη των διαδικασιών, το οποίο συμβάλει στην καλύτερη εκπαίδευση και επεκτασιμότητα του μοντέλου.

### 2.8 Μετρικές Αξιολόγησης

Για την αξιολόγηση της απόδοσης ενός μοντέλου ταξινόμησης χρησιμοποιούνται διάφορες μετρικές. Η επιλογή μιας μετρικής αξιολόγησης θα πρέπει να

ευθυγραμμίζεται με τον τελικό στόχο της ταξινόμησης και γι' αυτό είναι μία υποκειμενική επιλογή. Θεωρείται καλή τακτική να δοκιμάζονται διαφορετικές μετρικές με σκοπό την περαιτέρω βελτίωση του μοντέλου πρόβλεψης.

### *Confusion Matrix*

Τα αποτελέσματα μια ταξινόμησης δεδομένων μπορούν να αποτυπωθούν σε έναν πίνακα συσχέτισης(confusion matrix). Ο πίνακας συσχέτισης περιγράφει την απόδοση ενός μοντέλου ταξινόμησης σε δοκιμαστικά δεδομένα, όπου οι αληθινές τιμές είναι γνωστές .Επίσης αποτελεί την βάση για την εφαρμογή και άλλων μετρικών που θα δούμε παρακάτω. Ο πίνακας συσχέτισης συμπληρώνεται με τις παρακάτω τιμές

*(TP) True Positive - Αληθής πρόβλεψη που είναι όντως αληθής*

*(TN) True Negative – Ψευδής πρόβλεψη που είναι όντως ψευδής*

*(FP) False Positive – Αληθής πρόβλεψη που η αληθινή τιμή της είναι ψευδής*

*(FN) False Negative – Ψευδής πρόβλεψη που η αληθινή τιμής της είναι αληθής*

Για παράδειγμα στην ταξινόμηση των emails σε spam και not spam έχουμε τα αποτελέσματα του παρακάτω πίνακα

| | | Predicted Class: Spam | Predicted Class: Non-Spam |
|---|---|---|---|
| Actual Class | Spam | TP=45 | FN=20 |
| | Non-Spam | FP=5 | TN=30 |

(8)

*Σχήμα 10 Πίνακας Σύγχυσης ( Confusion Matrix )*

Ο άξονας X αναπαριστά τις προβλέψεις (predicted Class) και ο άξονας Y τις πραγματικές τιμές (Actual Class). Αναλύοντας τον πίνακα έχουμε

***TP -> 45 emails με πρόβλεψη spam και πραγματική τιμή spam***

***FP -> 5 emails με προβλέψη spam και πραγματική τιμή Non-Spam***

***FN -> 20 emails με πρόβλεψη Non-Spam και πραγματική τιμή Spam***

***TN -> 30 emails με προβλεψη Non-spam και πραγματική τιμή Non-Spam***

### *Accuracy*

Το accuracy υπολογίζει πόσο συχνά ένα μοντέλο ταξινόμησης προβλέπει τις σωστές τιμές. Προσδιορίζεται με την αναλογία των σωστών προβλέψεων ως προς το σύνολο των προβλέψεων. Είναι καλό να χρησιμοποιείται σε περιπτώσεις όπου τα δεδομένα είναι καλά ισορροπημένα και η βαρύτητα των θετικών και αρνητικών προβλέψεων είναι περίπου ίση.

***Accuracy = (TP+TN) / (TP+TN+FP+FN)***

### *Precision*

Το Precision αποτυπώνει πόσες από τις προβλέψεις ως «αληθείς» είναι σωστές και προσδιορίζεται από τον αριθμό των «αληθώς» θετικών ως προς το σύνολο των θετικών προβλέψεων. Το precision προτιμάται στις περιπτώσεις όπου οι «ψευδείς» θετικές προβλέψεις έχουν μεγαλύτερη βαρύτητα από τις «ψευδείς» αρνητικές.

***Precision = TP / (TP+FP)***

### *Recall (Sensitivity)*

Το Recall εξηγεί πόσες πραγματικά αληθείς τιμές κατάφερε το μοντέλο μας να προβλέψει και χρησιμοποιείται σε περιπτώσεις όπου η «ψευδής» αρνητική πρόβλεψη είναι πιο σημαντική από μία «ψευδής» θετική. Για παράδειγμα σε ένα ιατρικό σύστημα ελέγχου για covid-19 είναι πιο σημαντικός ο εντοπισμός του ιού ακόμα και με μεγάλη πιθανότητα λάθους από τον μη εντοπισμό του καθώς μια «ψευδώς» αρνητική πρόβλεψη συμβάλει δραστικά στην διασπορά του ιού αλλά και στην ανάρρωση του ασθενή. Το recall προσδιορίζεται με τον αριθμό τον «αληθώς» θετικών προβλέψεων ως προς το σύνολο των «αληθώς» θετικών και των «ψευδών» αρνητικών.

***Recall = TP / (TP+FN)***

### *F1 score*

Η βαθμολογία F1 είναι ο αρμονικός μέσος όρος του precision και του recall και έχει μέγιστη τιμή αναφοράς όταν το precision ισούται με το recall . Προτιμάται σε

περιπτώσεις όπου ο αριθμός των «αληθώς» αρνητικών είναι μεγάλος ή όταν τα «ψευδώς» θετικά με τα «ψευδώς» αρνητικά έχουν την ίδια βαρύτητα. Προσδιορίζεται από τον παρακάτω τύπο

***F1 = 2 * (Precision*Recall) / (Precision+Recall)***

### *F (β) Score*

Η βαθμολογία F1 δίνει ίση βαρύτητα στο precision και στο recall . Κάποιες φορές όμως χρειάζεται να δοθεί περισσότερη βαρύτητα στο recall έναντι του precision ή αντίστροφα. Για αυτό μπορούμε να δημιουργήσουμε μια σταθμισμένη f1 βαθμολογία, όπου ή μεταβλητή «β» διαχειρίζεται την αντιστάθμιση μεταξύ precision και recall.

***Fβ = (1+β^2) * (precision * recall) / ((β^2 *precision) + recall)***

Στο παραπάνω τύπο όταν β < 1 δίνεται μεγαλύτερη βαρύτητα στο recall και όταν β>1 στο precision.

### *AUC (under the ROC curve) – (κάτω από την καμπύλη ROC)*

ROC – Receiver Operator Characteristic

Η AUC ROC υποδηλώνει την ικανότητα του αλγόριθμου ταξινόμησης να διαχωρίσει δύο κλάσεις.

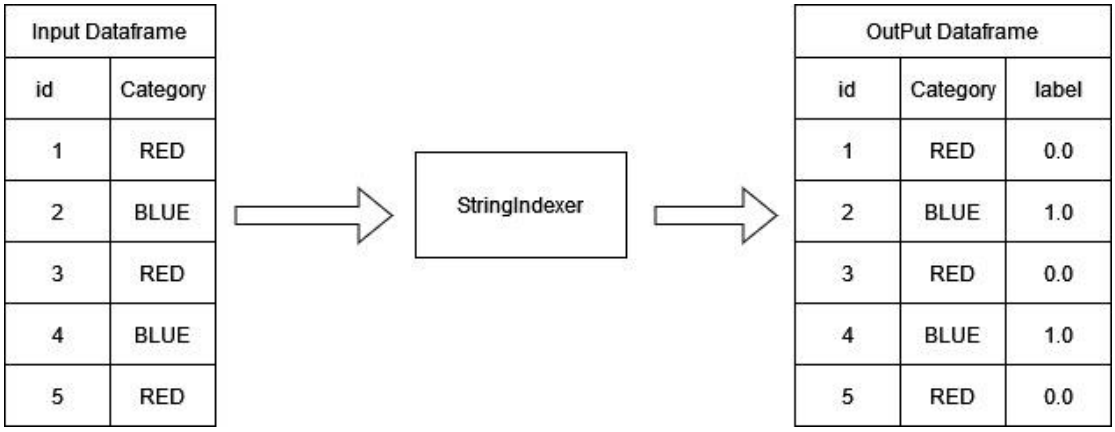
## 3. Machine Learning Library (MlLib)

Όπως διατυπώθηκε στο κεφάλαιο 2.4.2 η MlLib είναι ένα υποσύστημα του Apache ecosystem που περιλαμβάνει ένα πλήρες και αποδοτικό σύνολο από αλγόριθμους που σχετίζονται με την μηχανική μάθηση. Στο συγκεκριμένο κεφάλαιο θα παρουσιαστούν συνοπτικά οι αλγόριθμοι που χρησιμοποιήθηκαν στην εργασία μαζί με τις παραμέτρους τους.

### 3.1 StringIndexer

Ο StringIndexer αναλαμβάνει την μετατροπή μιας αλφαριθμητικής ή αριθμητικής στήλης ετικετών σε στήλη ευρετηρίων (indexes) . Στην περίπτωση που η στήλη εισαγωγής είναι αριθμητική πρώτα μετατρέπεται σε αλφαριθμητική και έπειτα δημιουργείται η στήλη με τα ευρετήρια. Οι τιμές των δεικτών των ευρετηρίων είναι

εξ ορισμού από 0 – n όπου n ο αριθμός των ετικετών. Η ταξινόμηση των ετικετών με τους δείκτες γίνεται με βάση την συχνότητα εμφάνισης όπου 0 η ετικέτα με την μεγαλύτερη συχνότητα. Στην περίπτωση που δύο ετικέτες παρουσιάζουν την ίδια συχνότητα εφαρμόζεται αλφαριθμητική ταξινόμηση μεταξύ τους.

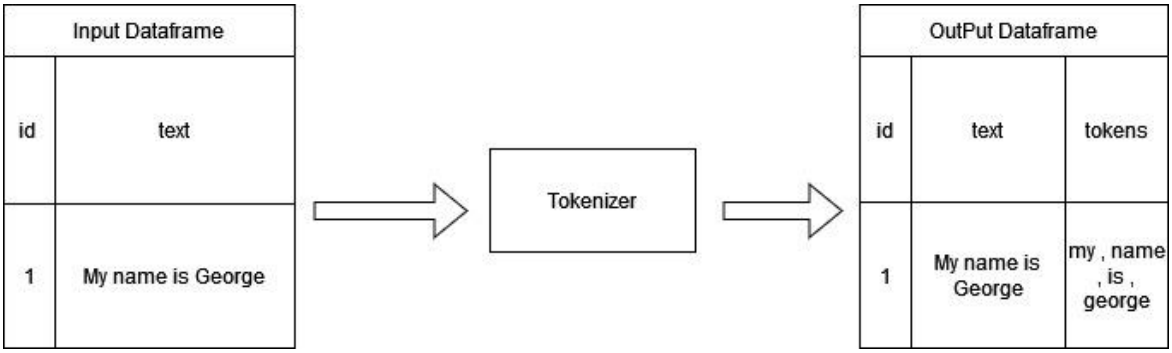



*Σχήμα 11 MlLib StringIndexer*

Η προεπιλεγμένη ταξινόμηση βάση συχνότητας μπορεί να αλλάξει με την παράμετρο ***stringOrderType*** που μπορεί να πάρει τις τιμές *frequencyDesc, frequencyAsc, alphabetDesc, alphabetAsc.*

### 3.2 Tokenizer

Η tokenizer μετατρέπει μια συμβολοσειρά εισόδου όπως για παράδειγμα ένα κείμενο σε πεζά και στην συνέχεια την χωρίζει βάση κενών(whitespaces) ώστε να δημιουργήσει μια στήλη εξόδου, όπου οι φράσεις παρουσιάζονται σαν μια ακολουθία λέξεων.

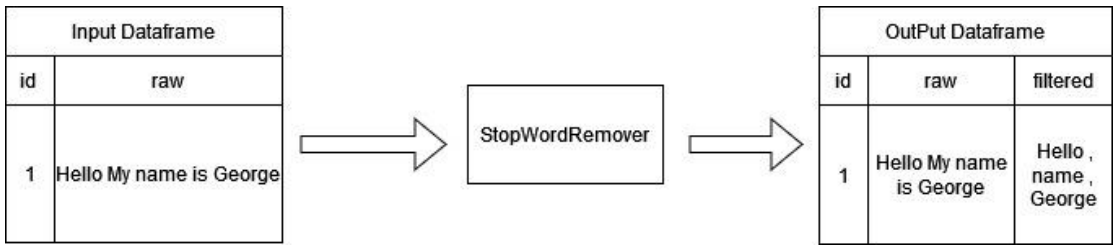



*Σχήμα 12 MlLib Tokenizer*

Επιπλέον η MlLib παρέχει και μια πιο εξειδικευμένη μέθοδο για την δημιουργία των tokens, την RegexTokenizer όπου σαν παράμετρος δίνεται η δυνατότητα εισαγωγής patterns σε Regular Expressions.

### 3.3 StopWordsRemover

Με τον όρο StopWords αναφερόμαστε σε μικρά τμήματα του λόγου με μεγάλη συχνότητα εμφάνισης που συνήθως δεν μεταφέρουν κάποιο ιδιαίτερο νόημα και επομένως συνίσταται να αφαιρούνται από τα δεδομένα εισόδου. Η StopWordsRemover κάνει ακριβώς αυτό παίρνοντας σαν δεδομένα εισόδου μια ακολουθία από λέξεις και αφαιρεί αυτές που προσδιορίζονται ως StopWords βάση μια λίστας που λαμβάνει ως παράμετρο. Η StopWordsRemover εξορισμού χρησιμοποιεί την λίστα με τις Αγγλικές StopWords αλλά υπάρχει επιλογή να χρησιμοποιηθούν και λίστες από άλλες γλώσσες που είναι διαθέσιμες καλώντας τις με την μέθοδο StopWordsRemover.loadDefaultStopWords(language). Η παράμετρος language μπορεί να πάρει τιμές όπως danish", "dutch", "english", "finnish", "french", "german", "hungarian", "italian", "norwegian", "portuguese", "russian", "spanish", "swedish" and "turkish".




*Σχήμα 13 MlLib StopWordRemover*

Ακόμα δίνεται η επιλογή της δημιουργίας νέας λίστας με StopWords ή επέκταση κάποια υφιστάμενης.

Νέα λίστα

*stopwordList = ["my", "test", "custom", "list"]*

*StopWordsRemover(inputCol='raw',outputCol='filtered', stopWords=stopwordList)*

*Επέκταση λίστας*

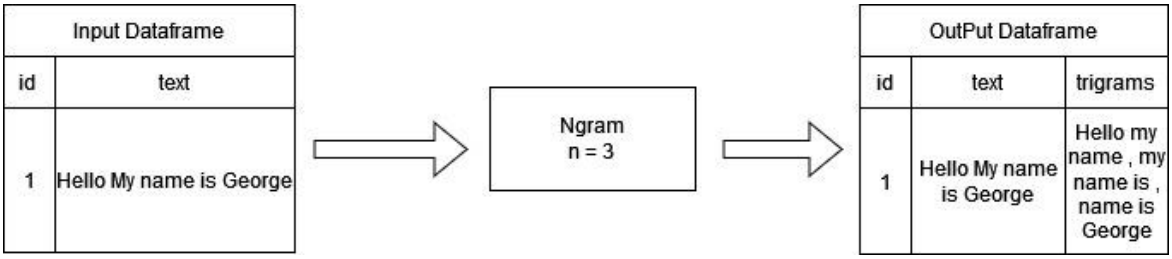
*stopwordList = [extend", "list", "example"]*

*StopWordsRemover.loadDefaultStopWords("english")*

*stopwordList.extend(StopWordsRemover().getStopWords())*
*stopwordList = list(set(stopwordList))*
*StopWordsRemover(inputCol='raw',outputCol='filtered', stopWords=stopwordList)*

### 3.4 ngram

Η ngram παίρνει ως είσοδο μια ακολουθία λέξεων και τις χωρίζει σε μικρότερες ακολουθίες μεγέθους n. Το n ορίζεται ως παράμετρος. Αν η ακολουθία εισόδου είναι μικρότερη του n τότε δεν παράγεται νέα ακολουθία ως έξοδος.

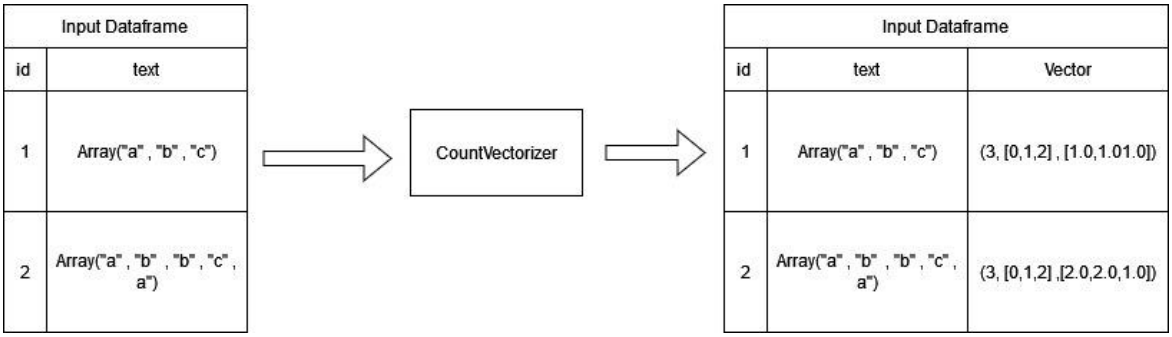



*Σχήμα 14 MlLib NGRAM*

### 3.5 CountVectorizer

Η CountVectorizer χρησιμοποιείται στην εξαγωγή ή και δημιουργία ενός λεξιλογίου από μία συλλογή κειμένων βασισμένων στην συχνότητα εμφάνισης των όρων – λέξεων και στην μετατροπή τους σε διανύσματα που μπορούν να αξιοποιηθούν ως δεδομένα εισόδου σε άλλους αλγόριθμους. Οι βασικές παράμετροι της είναι οι εξής:

**maxDF ->** Καθορίζει τον μέγιστο αριθμό διαφορετικών εγγράφων - κειμένων στα οποία θα μπορούσε να εμφανιστεί ένας όρος - λέξη για να συμπεριληφθούν στο λεξιλόγιο. Ένας όρος που εμφανίζεται περισσότερο από το όριο θα αγνοηθεί
**minDF ->** Καθορίζει τον ελάχιστο αριθμό διαφορετικών εγγράφων - κειμένων στα οποία πρέπει να εμφανίζεται ένας όρος - λέξη για να συμπεριληφθεί στο λεξιλόγιο
**minTF->** Χρησιμοποιείται ως φίλτρο για να μην συμπεριληφθούν σπάνιες λέξεις σε ένα έγγραφο. Για κάθε έγγραφο οι όροι – λέξεις με συχνότητα/ μέτρηση μικρότερη από το δεδομένο όριο αγνοούνται.
**VocabSize ->** μέγιστο μέγεθος του λεξικού




*Σχήμα 15 MlLib CountVectorizer*

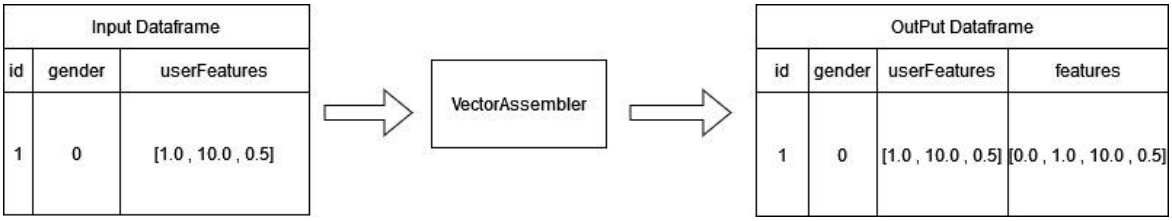
### 3.6 Idf

Η idf λαμβάνει ως είσοδο διανύσματα χαρακτηριστικών που έχουν δημιουργηθεί από την CountVectorizer και υπολογίζει ποια από αυτά εμφανίζονται λιγότερο. Με λίγα λόγια υπολογίζει την αντίστροφη συχνότητα όρων – λέξεων (tf - idf) σε μία συλλογή από έγγραφα – κείμενα όπως αυτή περιεγράφηκε στο κεφάλαιο 2.5

Μία βασική παράμετρός της είναι η minDocFreq που σχετίζεται με τον ελάχιστο αριθμό εγγράφων στα οποία πρέπει να εμφανίζεται ένας όρος για φιλτράρισμα.

### 3.7 VectorAssembler

Η VectorAssembler συνδυάζει λίστες χαρακτηριστικών που έχουν δημιουργηθεί από διαφορετικούς μετασχηματιστές – μεθόδους σε ένα ενιαίο διανυσμα χαρακτηριστικών που μπορεί να χρησιμοποιηθεί ως είσοδος για την εκπαίδευση των αλγόριθμων ταξινόμησης.




*Σχήμα 16 MlLib VectorAssembler*

### 3.8 LinearSVC

Ο LinearSVC είναι η υλοποίηση του αλγορίθμου support vector machines του Apache Spark για Binary Classification. Ο LinearSVC είναι ένας πολύ αποδοτικός αλγόριθμος και μέσω διαφόρων παραμέτρων που υποστηρίζει και συνοπτικά

περιγράφονται παρακάτω μας δίνεται η δυνατότητα να τον προσαρμόσουμε στα δεδομένα του Dataset.

**Threshold ->** Το όριο που ορίζεται για την μετατροπή των ακατέργαστών αποτελεσμάτων πρόβλεψης (raw predictions) που έχουν μορφή δεκαδικού σε ακέραιες προβλέψεις 0 και 1.

**Weights ->** Η βαρύτητα που υπολογίζεται για κάθε χαρακτηριστικό και χρησιμοποιείται σε δείγματα με ανισόρροπη κατανομή κλάσεων. Θέτοντας μεγαλύτερη βαρύτητα στην μειονοτική κλάση βοηθάμε τον αλγόριθμο στην δημιουργία ενός πιο ισορροπημένου μοντέλου πρόβλεψης μεταξύ τον κλάσεων.

**Intercept ->** Καθορίζει την τιμή της τομής ή αλλιώς του υπερεπιπέδου διαχωρισμού των κλάσεων.

**Standardization ->** Η τυποποίηση (Standardization) είναι μια μέθοδος κλιμάκωσης όπου οι τιμές επικεντρώνονται γύρω από το μέσο όσο με μία μικρή απόκλιση.

**regParam ->** Η παράμετρος τακτοποίησης (Reguralization parameter) αυξάνει τις πιθανότητες του μοντέλου να κάνει σωστές προβλέψεις σε νέα δεδομένα τα οποία δεν είχαν εκπαιδευτεί. Εάν δεν εφαρμοστεί τακτοποίηση το μοντέλο πρόβλεψης προσαρμόζεται υπερβολικά στα δεδομένα εκπαίδευσης και έτσι μειώνεται η ικανότητα γενίκευσης - όπως περιγράφεται στην βιβλιογραφία - η οποία είναι η δυνατότητα ενός μοντέλου να εφαρμόζεται έχοντας υψηλή απόδοση σε νέα δεδομένα που εισάγονται.

### 3.9 LogisticRegression

Αντίστοιχα η υλοποίηση του Logisticregression της MlLib υποστηρίζει Binary και Multi Class Classification και έχει κοινές παραμέτρους με τον LinearSVM τις Threshold, Weights, Standardization, regParam και Intercept. Επιπλέον μπορεί να λάβει περαιτέρω παραμετροποίηση με τις παρακάτω:

**Family ->** binomial για binary Classification, multinomial για MultiClass Classification

**numClasses ->** ο αριθμός πιθανών κλάσεων σε περίπτωση Multi Class classification

**numFeatures->** Αριθμός των χαρακτηριστικών

**iterations ->** Ο μέγιστος αριθμός επαναλήψεων

# 4. Δημιουργία Υποστηρικτικής Υποδομής

Για την ολοκλήρωση της παρούσας εργασίας χρησιμοποιήθηκαν υποδομές που παρέχονται στους φοιτητές του τμήματος από το κέντρο εθνικών υποδομών Έρευνας και τεχνολογίας (GRNET) σε μορφή IaaS (Infrastructure as a Service) με την ονομασία «Okeanos». Οι πόροι που δεσμευτήκαν αφορούσαν μια static public ip, 92 πυρήνες επεξεργασίας, 92GB μνήμης RAM και 630GB αποθηκευτικού χώρου. Τα παραπάνω κατόπιν διαφόρων δοκιμών διανέμισαν σε 11 εικονικά μηχανήματα (Virtual Machines) με λειτουργικό σύστημα Ubuntu Server 22.04.01 τα οποία αποτελούν και τους κόμβους του cluster. Κάθε κόμβος δεσμεύει από το σύνολο των πόρων 8 πυρήνες επεξεργασίας, 8GB RAM και 30GB αποθηκευτικού χώρου. Εγκαταστάθηκε και παραμετροποιήθηκε HADOOP 3.3.4 και ορίστηκε ένας κόμβος ως Namenode και οι υπόλοιποι 10 ως Datanodes. Επίσης εγκαταστάθηκε Apache Spark 3.3.0 και έγιναν οι απαραίτητες ρυθμίσεις ώστε να χρησιμοποιεί την επέκταση YARN του Hadoop για την διαχείριση πόρων και εφαρμογών. Τέλος χρησιμοποιήθηκε το Jupyter Notebook για την συγγραφή κώδικα σε γλώσσα pySpark. Το Jupyter Notebook είναι ένα διαδραστικό περιβάλλον ανάπτυξης εφαρμογών ανοικτού λογισμικού πολύ δημοφιλές για την χρήση του σε εφαρμογές μηχανικής μάθησης. Το μεγαλύτερο πλεονέκτημα σε σύγκριση με ένα τυπικό περιβάλλον IDE είναι πως δίνεται η δυνατότητα εγγραφής κομμάτια κώδικά ή εντολών σε blocks και η εκτέλεση κατά επιλογή. Η pySpark είναι μια διεπαφή του Apache Spark που επιτρέπει την ανάπτυξη εφαρμογών χρησιμοποιώντας την γλώσσα προγραμματισμού Python.

Η απόδοση του cluster σε γενικές γραμμές κρίθηκε επαρκής, ωστόσο δεν προτείνεται η χρήση της συγκεκριμένης υποδομής για περιβάλλοντα κατανεμημένης επεξεργασίας που χειρίζονται μεγάλο όγκο δεδομένων καθώς παρατηρήθηκαν πολλά προβλήματα στο δίκτυο ιδιαίτερα σε ώρες αιχμής. Βασικό κομμάτι ενός HDFS συστήματος είναι η μεταφορά δεδομένων μέσω δικτύου οπότε στις περιπτώσεις που το δίκτυο δεν μπορούσε να αποκριθεί στους κατάλληλους χρόνους παρουσιάζονταν προβλήματα στους nodes του cluster και κατά συνέπεια επηρεάζονταν αρνητικά και οι ταχύτητες περάτωσης. Για την επίλυση του συγκεκριμένου θέματος χρειάστηκε να αυξηθούν - από τις ρυθμίσεις του Hadoop- το μέγεθος κάθε αρχείου (Chunk Size) καθώς και οι χρόνοι αναμονής στο δίκτυο (network Timeout). Οι συγκεκριμένες ρυθμίσεις ελαχιστοποίησαν τα ποσοστά

αποτυχιών στους κόμβους αλλά μεγιστοποίησαν τους χρόνους υπολογισμού και εκτέλεσης. Η υποδομή «Okeanos» είναι σχεδιασμένη έτσι ώστε να εξυπηρετεί μεγάλο όγκο χειριστών σε γενικής φύσεως projects και εργαστηριακά μαθήματα οπότε είναι φυσικό να δίνει έμφαση στην σταθερότητα κα στην απρόσκοπτη - αδιάλειπτη λειτουργία του και όχι στις υψηλές επιδόσεις. Επομένως συνίσταται τέτοιου είδους projects να τρέχουν σε ειδικά διαμορφωμένες υποδομές συνδυάζοντας ταχύτητα και αξιοπιστία.

# 5. Ανάλυση Έφαρμογής

Στο κεφάλαιο που ακολουθεί γίνεται μια μικρή αναφορά σε σχετικές εργασίες στο παρελθόν, περιγράφονται τα δεδομένα εισαγωγής, αναλύεται η μεθοδολογία επίλυσης του προβλήματος, τα βήματα που ακολουθήθηκαν, καθώς και οι δοκιμές που έγιναν.

## 4.1 Σχετικές Έργασίες

Η πιθανή ταυτοποίηση περιπτώσεων παιδεραστίας είναι ένα θέμα που έχει απασχολήσει σε μεγάλο βαθμό την επιστημονική κοινότητα. Ένα χαρακτηριστικό παράδειγμα είναι ο διαγωνισμός με τίτλο “Sexual Predator Identification 2012” (9), που διοργανώθηκε από το εργαστήριο αξιολόγησης PAN* στο συνέδριο CLEF (Conference and Labs OF the Evaluation Forum) του 2012. Ο σκοπός ήταν η σύγκριση διαφόρων μεθοδολογιών για την αποτελεσματικότητά τους στην εύρεση πιθανών περιπτώσεων παιδικής εκμετάλλευσης.

Στον συγκεκριμένο διαγωνισμό το σύνολο δεδομένων που χρησιμοποιήθηκε είναι από την φύση του μη ισορροπημένο, δηλαδή μόλις το 1% των χρηστών είναι χαρακτηρισμένοι ως predators. Στην προσπάθεια φιλτραρίσματος ώστε να εξισορροπηθεί το σύνολο δεδομένων πολλοί διαγωνιζόμενοι εφάρμοσαν ταξινόμηση δύο σταδίων. Στο πρώτο στάδιο ο ταξινομητής έκανε διάκριση των συνομιλιών σε αυτές που περιλάμβαναν έναν predator και σε αυτές που δεν περιλάμβαναν κάποιον. Επιπλέον, σε αυτό το στάδιο επιλέγονταν οι συζητήσεις με μόνο δύο συνομιλητές και με περισσότερα από ν αριθμό μηνυμάτων. Συγκεκριμένα ν είναι ο αριθμός, που ο κάθε διαγωνιζόμενος έκρινε αρκετό ώστε η συζήτηση να έχει κάποιο νόημα για την δημιουργία χαρακτηριστικών. Για παράδειγμα μια συζήτηση μεταξύ δύο ατόμων που έχουν ανταλλάξει μόνο 4 μηνύματα δεν

θεωρείται καλό δείγμα για να μπορούμε να εξάγουμε χαρακτηριστικά, που πιθανόν διαφοροποιούν έναν predator από έναν non-predator. Στο δεύτερο στάδιο ύστερα από το φιλτράρισμα των δεδομένων γινόταν η ταξινόμηση του predator από τον non-predator σε επίπεδο χρήστη.

Στο στάδιο του φιλτραρίσματος παρατηρήθηκε σε αρκετές περιπτώσεις η χρήση προσαρμοσμένων λεξικών, όπως για παράδειγμα λεξικό από όρους σχετικούς με σεξουαλικό περιεχόμενο.

Στην επιλογή των χαρακτηριστικών η πλειονότητα των διαγωνιζομένων χρησιμοποίησε χαρακτηριστικά που μπορούν να διαχωριστούν σε δύο κύριες κατηγορίες, τα λεκτικά και τα συμπεριφορικά. Λεκτικά χαρακτηριστικά είναι αυτά που μπορούν να εξαχθούν από τα κείμενα. Για παράδειγμα unigrams, bigrams, TF-IDF, αριθμός emoticons κ.α. Τα συμπεριφορικά χαρακτηριστικά είναι αυτά που αποτυπώνουν την ενέργεια ενός συνομιλητή. Για παράδειγμα, πόσες φορές ξεκίνησε τη συζήτηση, πόσο χρόνο κάνει να απαντήσει, ο αριθμός των ερωτήσεων που έκανε κ.α.

Μια πολύ κοινή τακτική μεταξύ των διαγωνιζομένων ήταν η δημιουργία ενός σετ χαρακτηριστικών για κάθε συνομιλητή, που μπορούσε να τον προσδιορίσει ως πιθανό predator. Κάποιες άλλες προσπάθειες εστίασαν στην δημιουργία σετ χαρακτηριστικών, που προέκυπτε συνδυαστικά και από τους δύο συνομιλητές.

Για την ταξινόμηση χρησιμοποιήθηκαν διάφοροι αλγόριθμοι με τον SVM (Support Vector Machines) να εφαρμόζεται με τη μεγαλύτερη συχνότητα. Άλλοι αλγόριθμοι που χρησιμοποιήθηκαν είναι οι Neural NetWork Classifier, decision trees, k-NN, Naive Bayes και Random Forest.

Τέλος, για την αξιολόγηση των αποτελεσμάτων χρησιμοποιήθηκε το F0.5 score δίνοντας δηλαδή έμφαση στο recall καθώς θεωρήθηκε μεγαλύτερης βαρύτητας η πρόβλεψη της θετικής κλάσης (predators) έστω και σαν False Positive. Η επιλογή της συγκεκριμένης μετρικής αιτιολογείται απόλυτα, με την παραδοχή πως σε μια τέτοια εφαρμογή ο τελικός χαρακτηρισμός ενός χρήστη ως predator θα πραγματοποιείται μόνο κατόπιν αξιολόγησης από επαγγελματία επιβολής του νόμου.

**Το PAN είναι μια σειρά επιστημονικών εκδηλώσεων σχετικά με την εγκληματολογία και την στυλομετρία ψηφιακών κειμένων*

### 4.2 Περιγραφή Δεδομένων

Οι Giacomo Inches και Fabio Crestani οι οποίοι συνέθεσαν τα δεδομένα που χρησιμοποιούνται στην παρούσα εργασία είχαν σκοπό την δημιουργία ενός ρεαλιστικού συνόλου δεδομένων, όπου οι περιπτώσεις προσπάθειας σεξουαλικής εκμετάλλευσης ανηλίκων να είναι σημαντικά λιγότερες σε σχέση με το υπόλοιπο σύνολο. Σύμφωνα με την βιβλιογραφία [1] σε ένα παρόμοιο πεδίο ερευνών σχετικό με την ύπαρξη παιδόφιλων σε συστήματα (peer-to-peer) το ποσοστό δράσης παιδοφιλίας ανέρχεται στο 0,25%.Σεβόμενοι το παραπάνω σε συνάρτηση με την ανάγκη δημιουργίας ενός συνόλου δεδομένων που θα έκανε εφικτή την πιθανή ταυτοποίηση περιπτώσεων παιδικής εκμετάλλευσης με χρήση τεχνολογιών μηχανικής μάθησης, ενίσχυσαν το παραπάνω ποσοστό στο 3 - 4% του συνόλου. Πιο συγκεκριμένα οι διάλογοι που επιλέχθηκαν αντιπροσωπεύουν σε ένα πολύ μικρό ποσοστό (3-4%) τα True Positives (διάλογοι που εμπεριέχουν έναν predator) και έπειτα στην συντριπτική τους πλειονότητα αποτελούνται από False Positives (διάλογοι ερωτικού περιεχομένου) και False Negatives (διάλογοι διαφόρων θεμάτων μεταξύ χρηστών). Αναλύοντας περισσότερο την δομή του συνόλου των δεδομένων σε επίπεδο χρηστών μπορούμε να το χωρίσουμε στις παρακάτω κατηγορίες και υποκατηγορίες.

***A –*** *Συνομιλίες που περιέχουν τουλάχιστον έναν predator*

***A1*** *– Συνομιλίες μεταξύ predator – ανήλικου*

***A2*** *– Συνομιλίες μεταξύ predator – Χρήστη( συνήθως κάποιος ενήλικας που παριστάνει τον ανήλικο)*

***A3*** *– Συνομιλίες μεταξύ predator – Υπάλληλο επιβολής του νόμου με σκοπό την εύρεση και σύλληψη του πρώτου*

***B*** *– Συνομιλίες μεταξύ ενηλίκων.*

Για τις κατηγορίες A1 και A3 δεν ήταν δυνατή η συγκέντρωση υλικού, καθώς και στις δύο περιπτώσεις οι πηγές-συνήθως υπηρεσίες επιβολής του νόμου- όπως είναι εύκολα αντιληπτό δεν είναι πρόθυμες στην δημοσιοποίηση σχετικών δεδομένων λόγω της ευαισθησίας τους. Για την κατηγορία A2 χρησιμοποιήθηκε ως πηγή ο ιστότοπος www.perverted-justice.com, όπου φιλοξενεί πραγματικές συνομιλίες μεταξύ καταδικασμένων predators και εθελοντών που παρίσταναν τους ανήλικους. Αυτές οι συνομιλίες συνθέτουν τα True Positives του συνόλου. Για την Β κατηγορία χρησιμοποιήθηκαν συνομιλίες από το αποθετήριο του Omegle. Η

Omegle σαν υπηρεσία εξ ορισμού επιτρέπει την συνομιλία μεταξύ αγνώστων υπό ένα καθεστώς πλήρης ανωνυμίας, η οποία ενθαρρύνει τους συνομιλητές να εκφράζονται πιο ελεύθερα. Το συγκεκριμένο αποθετήριο ταιριάζει απόλυτα στο σύνολο των δεδομένων, καθώς εμπεριέχει αληθινές συζητήσεις ερωτικού περιεχομένου και ανάρμοστης λεκτικής συμπεριφοράς που θα μπορούσαν να χαρακτηριστούν ως False Positives στο συγκεκριμένο εγχείρημα. Τέλος, ένας τεράστιος όγκος συζητήσεων γενικού περιεχομένου αντλήθηκε από τα κανάλια http://www.irclog.org/ και http://krijnhoetmer.nl/irc-logs, που αφορούν συνομιλίες μεταξύ δύο ή και τριών συνομιλητών και αντιπροσωπεύουν το σύνολο των True Negatives της συλλογής δεδομένων.

Σε όλες τις συζητήσεις αποδόθηκε ένας μοναδικός αριθμός id και έγινε αντικατάσταση των ψευδώνυμων με ξεχωριστό κωδικό author-id. Για την περαιτέρω εξασφάλιση της ανωνυμίας των χρηστών αντικαταστάθηκαν επίσης και όλες οι διευθύνσεις ηλεκτρονικού ταχυδρομείου με μοναδικές ετικέτες. Επίσης επιλέχθηκαν οι συζητήσεις με ίσα ή λιγότερα των 150 μηνυμάτων, οι οποίες αποτελούσαν περισσότερο από το 90% της συλλογής δεδομένων. Τέλος αποφασίστηκε από τους δημιουργούς το υλικό να χωριστεί σε δύο μέρη. Το πρώτο σε ποσοστό 30% σαν δεδομένα εκπαίδευσης και το υπόλοιπο 70% σαν δεδομένα δοκιμής.

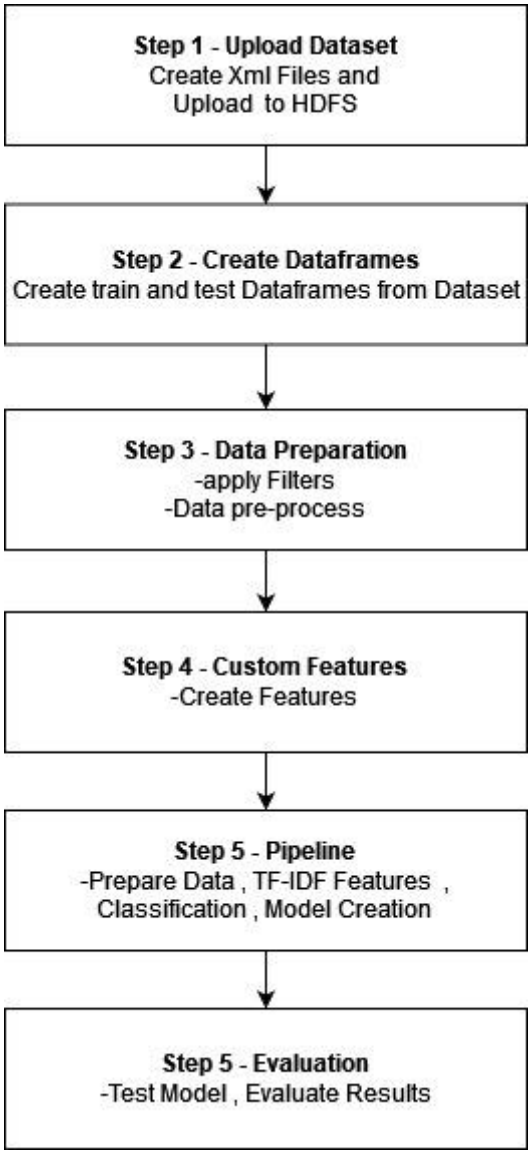
| | PJ<br>perverted-justice.com | krjin<br>krijnhoetmer.nl/irc-logs | irclog<br>irclog.org | omegle<br>omegle.inportb.com |
|---|---|---|---|---|
| #conversations | 11350 | 50510 | 28501 | 267261 |
| #conv. length ≤150 | 9076 | 48569 | 21896 | 265747 |
| (% all ) | (80%) | (96%) | (77%) | (99%) |
| Training set | | | | |
| #conv. length≤150 | 2723 | 14571 | 6569 | 43064 |
| ” and exactly 2 user | 984 | 2420 | 1146 | 41067 |
| (% training) | (36%) | (17%) | (17%) | (95%) |
| unique (perverted) users | 291 (142) | 2660 | 10613 | 84131 |
| Testing set | | | | |
| #conv. length≤150 | 5321 | 33998 | 15327 | 100482 |
| ” and exactly 2 user | 1887 | 5648 | 2673 | 95648 |
| (% testing) | (35%) | (17%) | (17%) | (95%) |
| unique (perverted) users | 440 (254) | 4358 | 17788 | 196130 |

*Σχήμα 17 Ανάλυση Dataset*

1 Latapy, M.: Quantifying Paedophile Queries in a Large P2P System. System pp. 401–405
(2011)

### 4.3 Μεθοδολογία Επίλυσης

Στο συγκεκριμένο κεφάλαιο θα παρουσιαστεί η μεθοδολογία επίλυσης καθώς και η προσέγγιση του προβλήματος. Για την καλύτερη κατανόηση της διαδικασίας παρουσιάζεται το παρακάτω σχήμα το οποιό χωρίζει την διαδικασία σε 5 διακριτά βήματα. Οι γραμμές κώδικα μαζί με τα σχόλια καθώς και οι οδηγίες για την εγκατάσταση του LAB θα ακολουθήσουν στα παραρτήματα του παρόντος εγγράφου.




*Σχήμα 18 Μεθοδολογία επίλυσης σε 5 βήματα*

### 4.3.1 Εισαγωγή Δεδομένων

Το αρχικό Dataset ήταν χωρισμένο σε δύο αρχεία train και test μορφής xml. Για την καλύτερη διαχείριση, μελέτη αλλά και την εισαγωγή στο Lab τα αρχεία αυτά χωρίστηκαν σε μικρότερα xml όπου κάθε ένα περιείχε την συζήτησή μεταξύ δύο η και παραπάνω χρηστών όπως φαίνεται στην εικόνα παρακάτω. Επίσης, η ονομασία κάθε νέου xml ήταν ένας αύξοντας αριθμός από το 1 ως το σύνολο ν των συνομιλιών. Για παράδειγμα 1.xml, 2.xml, 3.xml, …., n.xml.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<conversation id="380743a89b0163cef563c61c988ca1fd">
    <message line="1">
      <author>7cc27ec76df624c767d83b4a5d43fb17</author>
      <time>02:45</time>
      <text>ask me 5 questions and i will answer them truthfully</text>
    </message>
    <message line="2">
      <author>c18a32dc7dc8b612f47c3d21f6ccf5f0</author>
      <time>02:45</time>
      <text>you first.</text>
    </message>
    <message line="3">
      <author>7cc27ec76df624c767d83b4a5d43fb17</author>
      <time>02:45</time>
      <text>you want me to ask you questions first?</text>
    </message>
  </conversation>
```

*Σχήμα 19 Δείγμα Συζήτησης*

Παρατηρώντας την εικόνα βλέπουμε πως κάθε αρχείο περιλαμβάνει τα εξής στοιχεία

**Conversation Id ->** *Μοναδικό αλφαριθμητικό που αντιπροσωπεύει μια συζήτηση*

**Author ->** *Αλφαριθμητικό που αντικαθιστά το όνομα του χρήστη. Μπορεί να εμφανιστεί και σε άλλες συζητήσεις*

**Message line ->** *Αύξοντας αριθμός που υποδηλώνει τον αριθμό των μηνυμάτων μιας συζήτησης*

**Time ->** *Ώρα αποστολής της μορφής hh:mm*

**Text ->** *Το κείμενο του μηνύματος*

Επίσης συμπληρωματικά με το dataset των συνομιλιών υπάρχει ένα αρχείο κειμένου (txt) με τους χρήστες (authors) που χαρακτηρίζονται ως predators.Από το συγκεκριμένο δημιουργήθηκε το dataframe “truth_df”.

Τα παραπάνω αρχεία φορτώθηκαν στον master node και στην συνέχεια έγιναν εισαγωγή στο HDFS Cluster του Lab.Με την χρήση της βιβλιοθήκης “com.databricks.spark.xml” έγινε η μετατροπή των xml σε Spark Dataframes. Κατά το διάβασμα των αρχείων χρησιμοποιήθηκε στα παραχθέντα Dataframes “traindf” και “testdf” σαν στήλη η ονομασία του αρχείου (πχ 1.xml) το οποίο περιέγραφε μοναδικά την κάθε συνομιλία. Για το φιλτράρισμα αλλά και για την δημιουργία χαρακτηριστικών σε επόμενο στάδιο δημιουργήθηκαν οι παρακάτω στήλες.

**MsgNo ->** *Αριθμός μηνυμάτων που έστειλε ο χρήστης*

**totalMessages->** *Σύνολο μηνυμάτων συνομιλίας*

**_c1 ->** *Χαρακτηρισμός χρήστη ως Pred η NonP, προκύπτει από το left join των αρχείων traindf και testdf με το truth_df. Όπου δεν υπήρξε αντιστοίχιση συμπληρώθηκε στο πεδίο η τιμή NonP ενώ οι υπόλοιπες εγγραφές συμπληρώθηκαν με την τιμή pred.*

**Auth@File ->** *Μοναδικό πεδίο που χαρακτηρίζει κάθε γραμμή και προκύπτει από τις στήλες Author που είναι ο χρήστης και το Filename δηλαδή το όνομα αρχείου*

**wordcount ->** *Αριθμός λέξεων που έστειλε ο χρήστης*

**totalWords->** *Σύνολο λέξεων συνομιλίας*

**Role->** *Δημιουργία 3 κλάσεων με τιμές Casual, Pred, Victim*

Σύμφωνα με το κεφάλαιο 2.1 και τα στάδια σεξουαλικής παρενόχλησης, ο δράστης θα προσπαθήσει να απομονώσει το υποψήφιο θύμα ώστε να δημιουργήσει ευνοϊκές συνθήκες αλλά και να δημιουργήσει περιβάλλον μυστικότητας. Έτσι οι συζητήσεις που περιείχαν περισσότερους από δύο συνομιλητές αφαιρέθηκαν και από τα δύο dataframes (train και test). Παρακάτω το τελικό Dataframe όπως διαμορφώθηκε

| MsgNo | totalMessages | author | time | Filename | _c1 | Auth@File | texts | wordCount | totalWords | Role |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 048dd4ce22996dab9... | 04:36 | 278.xml | NonP | 048dd4ce22996dab9... | [rush fan?] | 2 | 5 | Casual |
| 2 | 3 | 1084df44f7bac0f4d... | 19:53 | 32588.xml | NonP | 1084df44f7bac0f4d... | [hi, asl] | 2 | 3 | Casual |
| 3 | 5 | 1201354159032f1c5... | 00:31 | 8515.xml | NonP | 1201354159032f1c5... | [hi, m 16, u] | 4 | 6 | Casual |
| 2 | 4 | 1f1820e556c34d410... | 12:16 | 37196.xml | NonP | 1f1820e556c34d410... | [heyy, asl ??? =)] | 4 | 6 | Casual |
| 2 | 3 | 2adca3dec87abe3a3... | 23:14 | 59500.xml | NonP | 2adca3dec87abe3a3... | [HAPPY QUAZIGGYZI... | 3 | 4 | Casual |
| 2 | 5 | 514e2568565f4f1d5... | 18:27 | 64859.xml | NonP | 514e2568565f4f1d5... | [hey, i dont like... | 9 | 12 | Casual |
| 3 | 4 | 573cd5c1b65b4d5c7... | 07:16 | 7855.xml | NonP | 573cd5c1b65b4d5c7... | [19 m, nd u./?, ?... | 5 | 9 | Casual |
| 12 | 24 | a6b89c5b4150eea92... | 22:33 | 35943.xml | NonP | a6b89c5b4150eea92... | [Heyy, :O, I WILL... | 35 | 71 | Casual |
| 3 | 4 | b62d34d946b77621a... | 04:33 | 51945.xml | NonP | b62d34d946b77621a... | [hi, h are u, 23m u] | 6 | 7 | Casual |
| 11 | 34 | bdf99c2791edb8f28... | 18:19 | 2920.xml | NonP | bdf99c2791edb8f28... | [hello, how r you... | 38 | 69 | Casual |
| 2 | 4 | c1e5da1bb7c81fc3c... | 23:55 | 22508.xml | NonP | c1e5da1bb7c81fc3c... | [hi, 23m] | 2 | 4 | Casual |
| 5 | 19 | c62283536cf6261e5... | 00:39 | 21099.xml | pred | c62283536cf6261e5... | [hi, are u there,... | 13 | 58 | pred |
| 9 | 25 | e06589a0e2bd2d8b5... | 05:11 | 19471.xml | NonP | e06589a0e2bd2d8b5... | [hi, anyone got a... | 67 | 206 | Casual |
| 2 | 4 | e6dca4abd8509a7f1... | 01:42 | 27844.xml | NonP | e6dca4abd8509a7f1... | [hi, aasl] | 2 | 7 | Casual |
| 2 | 3 | e73180abb1501f843... | 10:53 | 21668.xml | NonP | e73180abb1501f843... | [hello, 20 m] | 3 | 13 | Casual |
| 2 | 3 | ef57f16758e3a43f5... | 06:12 | 56098.xml | NonP | ef57f16758e3a43f5... | [hi, asl?] | 2 | 5 | Casual |
| 3 | 4 | f42ac83de02a56fb0... | 21:03 | 38881.xml | NonP | f42ac83de02a56fb0... | [hi, m, n u] | 4 | 5 | Casual |
| 2 | 3 | f6f7ef92d4f7604a1... | 16:22 | 47727.xml | NonP | f6f7ef92d4f7604a1... | [hi, asl] | 2 | 5 | Casual |
| 2 | 4 | 0a81c7678de300c0b... | 18:40 | 51043.xml | NonP | 0a81c7678de300c0b... | [hi, asl?] | 2 | 7 | Casual |
| 20 | 21 | 0afb1679c988c71d3... | 23:04 | 52107.xml | NonP | 0afb1679c988c71d3... | [?_?, ?_?, ?_?, ?... | 20 | 21 | Casual |

only showing top 20 rows

*Σχήμα 20 Dataframe Print*

| Dataframe | Σύνολο εγγραφών | Preds | NonP | Casual | Victims |
|---|---|---|---|---|---|
| train | 91202 | 1088 | 90114 | 89026 | 1088 |
| test | 211077 | 1887 | 209190 | 207303 | 1887 |

Πίνακας 1 Αποτελέσματα Πρώτου Σταδίου «Εισαγωγής Δεδομένων»

### 4.3.2 Προ-επεξεργασία Δεδομένων

Το στάδιο της προ-επεξεργασίας αποτελεί βασικό κομμάτι για την δημιουργία ενός αποτελεσματικού μοντέλου πρόβλεψης. Όπως αναφέρθηκε το συγκεκριμένο dataset παρουσιάζει μεγάλη ανισορροπία ως προς την κατανομή των δύο ζητούμενων κλάσεων. Επομένως το ζητούμενο σε αυτό το στάδιο είναι αφενός να εξισορροπηθεί στο καλύτερο βαθμό αυτή η κατανομή αφετέρου να γίνει ένας καθαρισμός των δεδομένων ώστε να αφαιρεθούν στοιχεία που ενδεχομένως δεν προσφέρονται στην δημιουργία ισχυρών χαρακτηριστικών που ακολουθεί σε επόμενο στάδιο.

Σχετικά με το φιλτράρισμα των συζητήσεων ορίστηκαν δύο συνθήκες που όπως αποδείχθηκε στο εργαστήριο επηρέασαν σε θετικό βαθμό το τελικό αποτέλεσμα. Η πρώτη συνθήκη αφορούσε την αφαίρεση συζητήσεων για τις οποίες ο συνολικός αριθμός μηνυμάτων που αντάλλαξαν ήταν μικρότερος των τριάντα (30). Αυτό βασίστηκε στην λογική πως για να μπορείς να αξιολογήσεις τις προθέσεις ενός συνομιλητή θα πρέπει μια συνομιλία να εξελιχθεί πέρα από το αρχικό στάδιο της αρχικής γνωριμίας και ανταλλαγής βασικών πληροφοριών. Η δεύτερη συνθήκη αφορά το ποσοστό συμμέτοχής του συνομιλητή. Για παράδειγμα παρατηρήθηκαν συνομιλίες με μεγάλο αριθμό μηνυμάτων (totalMessages) όπου μόνο ο ένας συνομιλητής είχε στείλει τον μεγαλύτερο όγκο των μηνυμάτων (>90%). Οι συγκεκριμένες περιπτώσεις ενώ ικανοποιούν την πρώτη συνθήκη δεν παρέχουν ικανοποιητικό βαθμό αλληλεπίδρασης η οποία θεωρείται αναγκαία στην εξέλιξη μιας συνομιλίας. Έτσι αφαιρέθηκαν όλες οι συζητήσεις από τις οποίες ο ένας συνομιλητής συμμετείχε με ποσοστό μικρότερο του 10%. Κατόπιν εφαρμογής των παραπάνω φίλτρων τα σύνολα διαμορφώθηκαν ως εξής

| Dataframe | Σύνολο εγγραφών | Preds | NonP | Casual | Victims |
|---|---|---|---|---|---|
| train | 8854 | 700 | 8150 | 7454 | 700 |
| test | 20106 | 1323 | 18782 | 17460 | 1323 |

Πίνακας 2 – Αποτελέσματα Φιλτραρίσματος Δεδομένων

Συγκρίνοντας τον πίνακα 1 με τον πίνακα 2 παρατηρούμε δύο σημαντικές διαφορές. Η πρώτη είναι η σημαντική μείωση των εγγραφών και η δεύτερη η σημαντική αύξηση του ποσοστού των Preds τόσο στο train αλλά και στο test dataframe από περίπου 1% σε περίπου 10% του συνόλου των εγγραφών.

Στην συνέχεια μελετώντας τις εγγραφές παρατηρήθηκε πως πολύ χρήστες επαναλάμβαναν λέξεις και προτάσεις χωρίς γραμματικό νόημα ή σχετικές με προώθηση προϊόντων ή παραπομπές σε εξωτερικούς συνδέσμους. Στον πίνακα παρακάτω, η στήλη “words” αφορά ένα trigram και το “word_cnt” το σύνολο εμφανίσεών του στην συγκεκριμένη συνομιλία που αντιστοιχεί στο “Filename”. Τέτοιες συνομιλίες χαρακτηρίστηκαν σαν spams και αφαιρέθηκαν από τα Dataframes. Κατόπιν διάφορων δοκιμών το όριο που ορίστηκε ώστε μια συνομιλία να χαρακτηριστεί ως spam είναι το word_cnt > 5.

| Filename | words | word_cnt |
|---|---|---|
| 62082.xml | pic 20enter funny... | 39 |
| 43166.xml | hurt feelings hurt | 25 |
| 9216.xml | stench death horr... | 41 |
| 9216.xml | carcass morning woke | 39 |
| 37639.xml | got address got | 35 |
| 31063.xml | amirite apos doesn | 440 |
| 26044.xml | white people good | 23 |
| 33609.xml | asl asl asl | 28 |
| 30809.xml | female male female | 49 |
| 4888.xml | dads lil black | 62 |
| 44955.xml | eid 1186205715247... | 37 |
| 55278.xml | blast play explosion | 525 |
| 55278.xml | lazer blast play | 525 |
| 64850.xml | asl asl asl | 28 |
| 4888.xml | real life[redacted] | 61 |
| 14789.xml | e5u e5u e5u | 31 |
| 4888.xml | life real life | 62 |
| 37384.xml | least five chats | 41 |
| 22296.xml | alderan chunks ev... | 40 |
| 9216.xml | peeled carcass mo... | 39 |

only showing top 20 rows

| Filename | words | word_cnt |
|---|---|---|
| 92538.xml | lolololololololol... | 25 |
| 131144.xml | php bd5f3a2940503... | 75 |
| 111651.xml | video games art | 101 |
| 111651.xml | see cosplay girl | 100 |
| 45441.xml | secks surprise butt | 31 |
| 26284.xml | girl[redacted]girl | 26 |
| 69730.xml | rasict rasict rasict | 32 |
| 139356.xml | kenya name ungara | 29 |
| 56710.xml | montel omgle lold... | 45 |
| 111651.xml | really cute love | 101 |
| 129552.xml | hard[redacted]hard | 21 |
| 77857.xml | www youtube user | 34 |
| 15729.xml | archives public p... | 26 |
| 15729.xml | public public html | 25 |
| 15729.xml | lists archives pu... | 26 |
| 146077.xml | [redacted]... | 23 |
| 2842.xml | [redacted] apos best | 24 |
| 24427.xml | asdf asdf asdf | 26 |
| 87372.xml | [redacted] cotton | 33 |
| 51920.xml | love love love | 751 |

*Σχήμα 21 Λέξεις Spam*

Κατόπιν εφαρμογής του παραπάνω φίλτρου ο πίνακας συνόλων διαμορφώθηκε ως εξής

| Dataframe | Σύνολο εγγραφών | Preds | NonP | Casual | Victims |
|---|---|---|---|---|---|
| Train | 8690 | 697 | 7993 | 7296 | 697 |
| Test | 19747 | 1312 | 18435 | 17123 | 1312 |

Πίνακας 3 Αποτελέσματα εφαρμογής φίλτρου SPAM

Τέλος αφαιρέθηκαν λέξεις με μήκος μεγαλύτερο του 12 καθώς στην συντριπτική πλειοψηφία τους παρατηρήθηκε πως δεν προσέφεραν κάποιο νόημα στην συζήτηση και επίσης δεν ακολουθούσαν κάποιο συντακτικό κανόνα. Για παράδειγμα οι λέξεις Heeyyyyyyyyyyyyy και heeeeyyyyyyyy. Προφανώς και οι δύο σχετίζονται με το αγγλικό «hey» αλλά στην προσπάθεια του κάθε συνομιλητή να τονίσει τον χαιρετισμό ή στα πλαίσια της χαλαρότητας και αφέλειας τέτοιων συνομιλιών αυτές οι λέξεις δεν ακολουθούν κάποιο μοτίβο και σίγουρα δεν ταυτοποιούν κάποιο συγκεκριμένο στυλ γραφής. Για παράδειγμα ο αριθμός τον «e» στις παραπάνω παραλλαγές είναι συμπτωματικός και αλλάζει συνεχώς ακόμα και αν προέρχεται από τον ίδιο συνομιλητή.

Τα παραπάνω φίλτρα ορίστηκαν κατόπιν μελέτης του Dataset και πολλαπλών δοκιμών με διάφορες τιμές. Προτείνεται σε κάθε τέτοιο εγχείρημα η λεπτομερής μελέτη ώστε να επιλεχθούν τα καλύτερα φίλτρα που να μεγιστοποιούν την απόδοση του classification παρακάτω. Καθαρισμός δεδομένων πραγματοποιείται και σε μετέπειτα στάδιο με χρήση βιβλιοθηκών της MlLib.

### 4.3.3 Δημιουργία Χαρακτηριστικών

Τα χαρακτηριστικά που χρησιμοποιήθηκαν μπορούν συνοπτικά να περιγραφούν στις παρακάτω τρεις κατηγορίες

Λεκτικά Χαρακτηριστικά

Τα λεκτικά χαρακτηριστικά σχετίζονται με τις λέξεις που εξάγονται από τις συνομιλίες και μπορούν να χρησιμοποιηθούν ως υποψήφια χαρακτηριστικά, ανάλογα με την συχνότητα εμφάνισης και την βαρύτητα τους στο κείμενο. Μια πολύ αποτελεσματική προσέγγιση είναι η χρήση του μοντέλου BoW (Bag Of Words) βασισμένο σε custom λεξικά αποτελούμενα από Unigrams, Bigrams, Trigrams.

Συμπεριφορικά Χαρακτηριστικά

Τα συμπεριφορικά χαρακτηριστικά είναι αυτά που υποδηλώνουν μια συμπεριφορά. Όπως η κυριαρχία σε μια συζήτηση από τον αριθμό μηνυμάτων που στέλνει ένας συνομιλητής, ο αριθμός των ερωτήσεων που κάνει για να δείξει ενδιαφέρον ή να μάθει πληροφορίες ή και ακόμα ο αριθμός των λέξεων που χρησιμοποιεί για να πετύχει κάποιο σκοπό.

Ψυχολογικά και γλωσσικά χαρακτηριστικά

Στην συγκεκριμένη κατηγορία περιλαμβάνονται τα χαρακτηριστικά προσανατολισμένα στην ψυχολογία και το λεξιλόγιο του δράστη. Για παράδειγμα αριθμός λέξεων ερωτικού περιεχομένου, αριθμός λέξεων ή φράσεων απαγόρευσης και μυστικότητας, μίμηση παιδικής γλώσσας (χρήση emoticons) κλπ.

Με βάση τις παραπάνω τρεις κατηγορίες δημιουργήθηκαν τα εξής features

MsgDom – Message Domination: ο λόγος μεταξύ του συνόλου των μηνυμάτων ενός χρήστη ως προς το σύνολο των μηνυμάτων μιας συνομιλίας.

WordDom – Word Domination: ο λόγος μεταξύ του συνόλου των λέξεων που χρησιμοποίησε ένα χρήστης ως προς το σύνολο των λέξεων μιας συζήτησης.

Το MsgDom όπως και το WordDom έχει να κάνει αφενός με την πρόθεση ενός συνομιλητή να κυριαρχήσει σε μια συζήτηση, αφετέρου με την προσπάθεια που καταβάλει ώστε να κρατήσει ζωντανό το ενδιαφέρον. Όσο πιο κοντά στο μηδέν το αποτέλεσμα των παραπάνω τόσο πιο ισορροπημένη θεωρητικά είναι μια συζήτηση.

SexWords: Το σύνολο των λέξεων ερωτικού περιεχομένου σε μία συνομιλία με την μέθοδο BagOfWords και δημιουργίας Custom λεξικού. Το συγκεκριμένο χαρακτηριστικό δημιουργήθηκε με την λογική πως οι πιθανοί predators τείνουν να κατευθύνουν την συνομιλία με επιμονή σε ερωτικό επίπεδο.

WhWords – Το σύνολο των αναφορικών αντωνυμιών σε μία συνομιλία που ξεκινάνε με Wh (what, who, where, which, whom κλπ.) με την μέθοδο BagOfWords και χρήσης Custom Λεξικού. Το συγκεκριμένο χαρακτηριστικό αποδείχθηκε πολύ σημαντικό στα αποτελέσματα της ταξινόμησης καθώς όπως φάνηκε οι πιθανοί predators με σκοπό να αντλήσουν πληροφορίες σχετικές με το υποψήφιο θύμα αλλά και τον περίγυρο του κάνουν πολλές ερωτήσεις του τύπου “What time you

finish school", "What time your parents come home", "Where are you now", What time will we meet" κλπ.

Age: Χαρακτηριστικό που υποδηλώνει την ηλικία των συνομιλητών και αντλείται από το περιεχόμενο των συνομιλιών με χρήση λεξικών unigrams και bigrams. (f 15, f 14, m 25, m23, female 22, female 14, 23 f κλπ).

NegVocab – Negative Vocabulary: Το σύνολο λέξεων και φράσεων που είναι πιθανό να προσδώσουν αρνητικό χαρακτήρα σε μία συνομιλία υπό του πρίσματος της παρούσας εργασίας. Δηλαδή φράσεις ή λέξεις όπου ο ένας συνομιλητής προσπαθεί να θέσει όρια, να δημιουργήσει μία κατάσταση μυστικότητας, να επιβάλει κάτι η ακόμα και να κερδίσει την εμπιστοσύνη του συνομιλητή ώστε να επιτύχει κάποιον σκοπό. Παραδείγματα τέτοιων είναι οι λέξεις-φράσεις "don't tell", "don't say", "tell anyone", "tell parents", "secret", "don't trust", "won't tell", "bad", "don't" κ.α. Το συγκεκριμένο λεξιλόγιο δημιουργήθηκε κατόπιν μελέτης των συνομιλιών καθώς και εφαρμογής της θεωρίας που παρουσιάστηκε παραπάνω σχετικά με τα στάδια σεξουαλικής εκμετάλλευσης στο κεφάλαιο 2.2.1.

Raw_Trigrams, Raw_Bigrams, Raw_Unigrams: Με χρήση της βιβλιοθήκης MlLib δημιουργήθηκαν αντίστοιχες στήλες με λέξεις - φράσεις μεγέθους 3, 2 και 1 αντίστοιχα.

Από τα παραπάνω δεν χρησιμοποιούνται στη δημιουργία του μοντέλου τα MsgDom, Age και bigrams καθώς δε φάνηκε να βελτιώνουν την απόδοση του αλγορίθμου ταξινόμησης και λειτουργούσαν επιβαρυντικά στο κόστος επεξεργασίας.

### 4.3.4 Classification – Ταξινόμηση

Το στάδιο της ταξινόμησης περιγράφει την διαδικασία παραγωγής του μοντέλου πρόβλεψης μέσα από την εφαρμογή διάφορων μεθόδων της βιβλιοθήκης MlLib. Οι συγκεκριμένες μέθοδοι συντέλεσαν στο περαιτέρω φιλτράρισμα των δεδομένων, τη δημιουργία και εφαρμογή χαρακτηριστικών και τέλος στην εφαρμογή των αλγορίθμων SVM (Support Vector Machine) και LR (logistic Regression) για την ταξινόμηση των δεδομένων σε κλάσεις.

Μελετώντας τις συνομιλίες παρατηρήθηκε πως στις περιπτώσεις όπου έχει εδραιωθεί ένα κλίμα εμπιστοσύνης μεταξύ συνομιλητών οι χρήστες τείνουν να

χρησιμοποιούν παρόμοιο λεξιλόγιο και εκφράσεις καθώς πλέον συμμετέχουν σε κοινή θεματολογία. Επιπροσθέτως, στο στάδιο του φιλτραρίσματος των δεδομένων επιλέχθηκαν συνομιλίες με μεγάλο αριθμό μηνυμάτων αλλά και σημαντικού ποσοστού συμμέτοχης. Έτσι θεωρήθηκε βέλτιστη η συγχώνευση των κειμένων που αφορούν ίδιες συνομιλίες καθώς και των υποψήφιων χαρακτηριστικών τους που δημιουργήθηκαν παραπάνω σε νέα Dataframes – trainFullConv και testFullConv. Η συγκεκριμένη παραδοχή δεν αλλάζει τον σκοπό της εργασίας, καθώς το ζητούμενο εξακολουθεί να είναι η πιθανή ανίχνευση περιπτώσεων εκμετάλλευσης ανηλίκων και αυτό μπορεί να επιτευχθεί μέσα από τον χαρακτηρισμό μιας συνομιλίας δυο ατόμων ως ύποπτη.

Επίσης, για την καλύτερη ισορροπία των δύο κλάσεων (Pred και NonP) δημιουργήθηκε η στήλη “classWeightCol”. Η συγκεκριμένη στήλη εισάγεται σαν παράμετρος στους αλγόριθμους ταξινόμησης της MlLib και χρησιμοποιείται σε περιπτώσεις όπου η κατανομή των κλάσεων δεν έχει ισορροπία (imbalance dataset). Η τιμή της στήλης για κάθε γραμμή ανάλογα με την κλάση που ανήκει υπολογίζεται με βάση μια ορισμένη(defined) μεταβλητή ισορροπίας. Στο συγκεκριμένο Dataset η τιμή της μεταβλητής ισορροπίας ορίστηκε βάση του τύπου balance_ratio = (total of positive class / train total rows) * 10 ο οποίος επιστρέφει την τιμή 0.790881. Για κάθε γραμμή της θετικής κλάσης (preds) η τιμή της στήλης “weightCol” συμπληρώθηκε με την τιμή της μεταβλητής balance_ratio και για κάθε γραμμή της αρνητικής κλάσης (NonP) η τιμή της στήλης υπολογίστηκε ως 1 – balance_ratio.

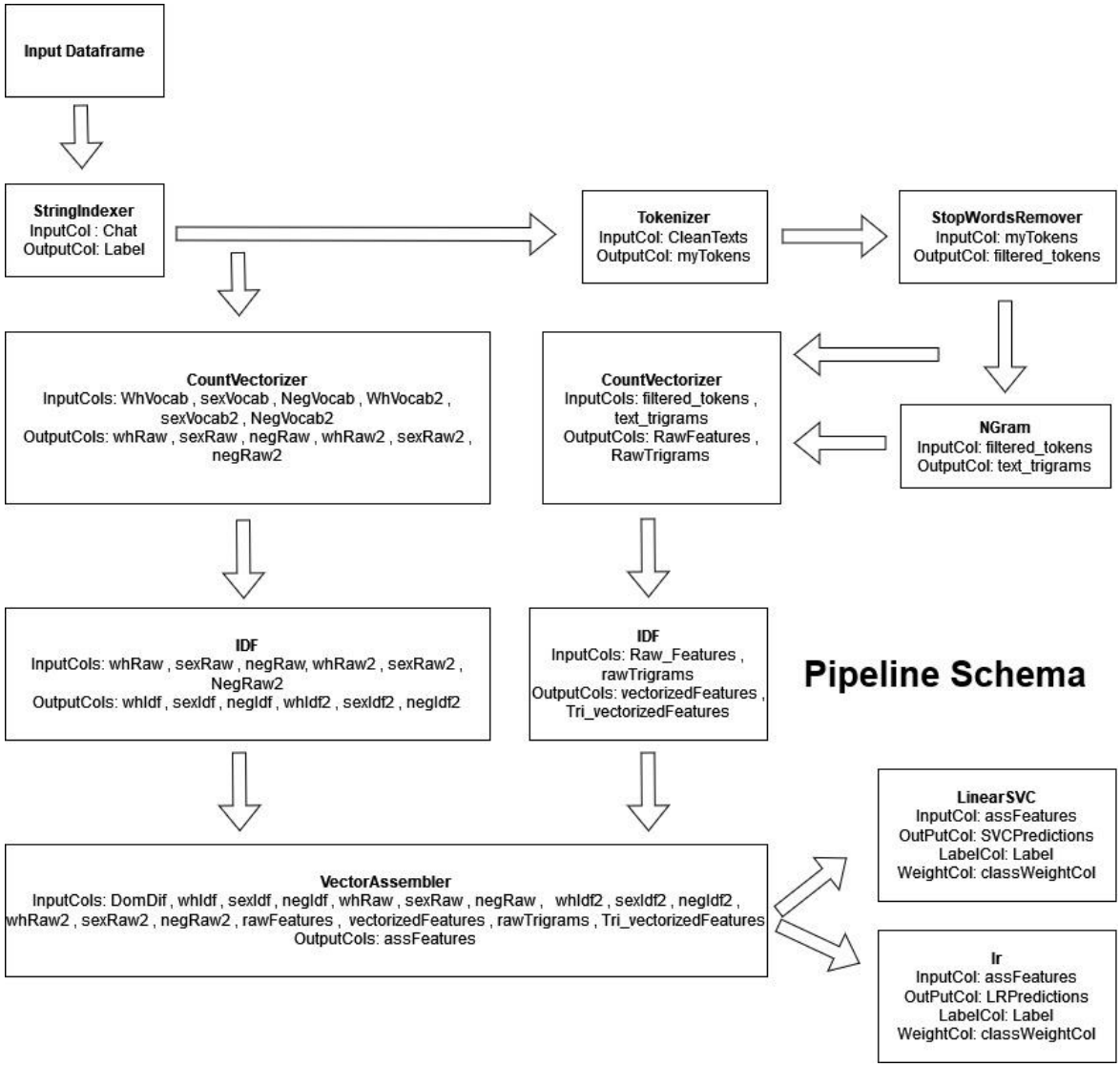
Τα νέα Dataframes αποτελούνται από τις παρακάτω στήλες οι οποίες προέρχονται από τα αρχικά train και test Dataframes, όπου A ο πρώτος συνομιλητής και B ο δεύτερος.

| Όνομα Στήλης | Περιγραφή |
|---|---|
| ClassWeightCol | pred -> classWeightCol = 1 * balance_ratio<br>Nonpred -> classWeightCol = 1 * (1 – balance_ratio) |
| Auth@file | Συνομιλητής A |
| _c1 | Ταυτοποίηση συνομιλίας ως pred ή NonPred |
| Wordcount | Σύνολο λέξεων A |

| | |
|---|---|
| TotalWords | Σύνολο λέξεων συνομιλίας |
| SexVocab | Λέξεις ερωτικού περιεχομένου A, |
| WhVocab | Wh-αντωνυμίες A |
| NegVocab | Λέξεις αρνητικού περιεχομένου A |
| Trigrams | Φράσεις μεγέθους 3 λέξεων A, |
| Auth@File2 | Συνομιλητής B |
| wordCount2 | Σύνολο λέξεων B |
| sexVocab2 | Λέξεις ερωτικού περιεχομένου B |
| whVocab2 | Wh-αντωνυμίες B |
| negVocab2 | Λέξεις αρνητικού περιεχομένου B |
| trigrams2 | Φράσεις μεγέθους 3 λέξεων B |
| Texts | Συγχωνευμένη συνομιλία A και B |
| DomDif | Η απόλυτη τιμή του αριθμού που προκύπτει από την παρακάτω πράξη και αφορά το ποσοστό συμμετοχής των συνομιλητών βάση του αριθμού των λέξεων που χρησιμοποιούν<br>wordcount/TotalWords – wordCount2/TotalWords |

Πίνακας 4 Στήλες trainFullConv - testFullConv

Για την ομαλοποίηση των βημάτων μέχρι την ταξινόμηση των δεδομένων δημιουργήθηκε ένα Pipeline, τα βήματα του οποίου φαίνονται στην παρακάτω εικόνα. Η χρήση Pipeline βοήθησε πολύ στο εργαστηριακό κομμάτι των δοκιμών, καθώς επέτρεψε την εκτέλεση των βημάτων ως μια ενιαία διαδικασία όπου δίνεται η δυνατότητα πραγματοποίησης αλλαγών στην παραμετροποίηση των αλγορίθμων χωρίς να επηρεάζεται η εκτέλεση των επιμέρους στοιχείων.




*Σχήμα 22 Pipeline Schema*

Όπως φαίνεται στο σχήμα ως είσοδο του Pipeline έχουμε τα dataframes τα οποία περιγράφτηκαν στην αρχή. Έπειτα καλείται η StringIndexer που δημιουργεί μια αριθμητική στήλη από μία στήλη αλφαριθμητικών. Σαν είσοδος της δίνεται η στήλη “Chat” που λαμβάνει τις δύο τιμές “NonP” και “Pred” οι οποίες αντιπροσωπεύουν τις δύο κλάσεις και βάση αυτών δημιουργεί την στήλη “Label” με τις διανυσματικές τιμές “0.0” και “1.0” για κάθε κλάση αντίστοιχα. Αυτή η μετατροπή είναι απαραίτητη προϋπόθεση για τους αλγόριθμους ταξινόμησης καθώς λειτουργούν μόνο με αριθμητικές τιμές. Συνεπώς, τόσο οι κλασεις όσο και τα χαρακτηριστικά που χρησιμοποιούνται σαν είσοδο σε αυτούς αναπαρίστανται σε αριθμητική μορφή.

Στην συνέχεια για τις στήλες WhVocab, sexVocab, NegVocab, WhVocab2, sexVcab2, NegVocab2 εκτελείται η CountVectorizer και στην συνέχεια η IDF. Με

την CountVectorizer υπολογίζεται η συχνότητα των όρων που περιέχονται σε κάθε στήλη αλλά και η μετατροπή τους σε διανύσματα. Μετά την εκτέλεση και της IDF έχουν δημιουργηθεί τα εξής χαρακτηριστικά whRaw, sexRaw, negRaw, whRaw2, sexRaw2, negRaw2, whidf, sexidf, negidf, whidf2, sexidf2, negidf2.

Για την στήλη “cleanTexts” καλείται η Tokenizer και στην συνέχεια η StopWordRemover ώστε να χωριστούν οι προτάσεις της στήλης σε όρους - λέξεις και να αφαιρεθούν από αυτές οι όροι – λέξεις που δεν γίνεται να αξιοποιηθούν ως χαρακτηριστικά (άρθρα, σύνδεσμοι κλπ). Το αποτέλεσμα της StopWordRemover στην ουσία είναι και τα unigrams που θα χρησιμοποιηθούν στην συνέχεια. Επίσης καλείται η ngram με n=3 για τον υπολογισμό των trigrams.

Στα unigrams εφαρμόζεται countVectorizer για την δημιουργία λεξικού 500 χαρακτηριστικών (vocabsize = 1000) εντάσσοντας σε αυτό λέξεις όρους με ελάχιστη συχνότητα 3 (minDF =3). Αντίστοιχα για τα trigrams δημιουργείται λεξικό μεγέθους 200 (vocabSize = 200) και ελάχιστη συχνότητα 2 (minDF=2). Και εδώ για κάθε νέα στήλη με χαρακτηριστικά που δημιουργήθηκε εφαρμόζεται η IDF.

Όλες οι στήλες με τα χαρακτηριστικά που δημιουργήθηκαν συνδυάζονται σε μία νέα στήλη “assFeatures” μέ την VectorAssembler η οποία είναι και η είσοδος για τους αλγόριθμους ταξινόμησης.

Τέλος εφαρμόζονται οι αλγόριθμοι ταξινόμησης lsvc (Linear Support Vector Machien) και lr (Logistic Regression) ώστε να συγκριθούν παράλληλα τα παραχθέντα μοντέλα. Και οι δύο αλγόριθμοι έχουν ως είσοδο την στήλη “assFeatures” για τα χαρακτηριστικά, την στήλη “label” για τις κλάσεις και την στήλη weightCol για την ισορροπία του δείγματος. Και στους δύο εφαρμόζεται τυποποίηση (standardization = true) και τακτοποίηση (regParam = 0.09) ενώ για τον lr θέτεται ως ανώτατο όριο επαναλήψεων το 4 (maxIter = 4).

## 5. Αποτελέσματα ταξινόμησης

Στο lab πραγματοποιήθηκαν πάρα πολλές δοκιμές τόσο στο κομμάτι της διαχείρισης των δεδομένων (φίλτρα, καθαρισμός, δημιουργία και εφαρμογή χαρακτηριστικών) όσο και στην εφαρμογή διαφόρων αλγόριθμων ταξινόμησης. Στο παρόν κεφάλαιο παρουσιάζονται μόνο αυτές με την βέλτιστη απόδοση.

## 5.1 Αξιολόγηση

Για την καλύτερη αξιολόγηση της προτεινόμενης μεθοδολογίας παρουσιάζονται τα αποτελέσματα ταξινομήσεων σε μελέτες περίπτωσης (Use Cases). Κάθε μελέτη περίπτωσης διαφέρει ως προς την παραμετροποίηση των αλγορίθμων που χρησιμοποιήθηκαν. Διατηρεί όμως ίδια τα αρχικά βήματα καθαρισμού και φιλτραρίσματος δεδομένων ώστε να παραμείνει κοινό το μέγεθος του dataset εκπαίδευσης και δοκιμής. Ο πίνακας που ακολουθεί περιγράφει συνοπτικά κάθε μελέτη περίπτωσης

| Όνομα | Περιγραφή |
|---|---|
| Use Case 1 | LR - Χωρίς μεταβλητή ισορροπίας, Χωρίς μεταβλητή γενίκευσης<br>SVM - Χωρίς μεταβλητή ισορροπίας, Χωρίς μεταβλητή γενίκευσης |
| Use Case 2 | LR – μεταβλητή ισορροπίας (weightCol), Χωρίς μεταβλητή γενίκευσης<br>SVM - μεταβλητή ισορροπίας (weightCol), Χωρίς μεταβλητή γενίκευσης |
| Use Case 3 | LR – μεταβλητή ισορροπίας (weightCol), με γενίκευση (regParam = 0.09)<br>SVM - μεταβλητή ισορροπίας (weightCol), με γενίκευση (regParam = 0.09) |

Πίνακας 5 UseCases

Για το υπολογισμό της απόδοσης του μοντέλου χρησιμοποιήθηκε η μετρική F1 και F(β) όπου β = 0.5. Η επιλογή του F1-Score χρησιμοποιείται καθώς έχει μεγάλη βαρύτητα το πλήθος των «ψευδώς» θετικών και «ψευδώς» αρνητικών. Δηλαδή οι περιπτώσεις pred όπου χαρακτηρίστηκαν σαν NonP και οι περιπτώσεις NonP που χαρακτηρίστηκαν σαν pred.

Επιπροσθέτως επειδή η πρόβλεψη της κλάσης pred είναι ουσιαστικής σημασίας τόσο για το ζητούμενο της παρούσας εργασίας αλλά και για τον ουσιαστικό χαρακτηρισμό της πράξης ως αξιόποινης υπολογίζεται και η F(0.5) ώστε να δοθεί έμφαση στο recall δηλαδή στην σωστή πρόβλεψη της κλάσης pred.

Παρακάτω οι πίνακες των αποτελεσμάτων για κάθε περίπτωση – Use Case που περιεγράφηκε.

| **Use Case 1** | | | |
|---|---|---|---|
| | f (1) Score | f (0.5) Score | Matrix Table |
| Logistic Regression | 0.95921 | 0.98374 | +-----+-------------+------<br>\|label\|lr_prediction\|count<br>+-----+-------------+------<br>\| 1.0\| 1.0\| 1223<br>\| 0.0\| 1.0\| 15<br>\| 1.0\| 0.0\| 89<br>\| 0.0\| 0.0\| 8427 |
| SVM | 0.87661 | 0.90590 | label\|lsvc_prediction\|count<br>------+---------------+------<br>1.0\| 1.0\| 1293<br>0.0\| 1.0\| 345<br>0.0\| 0.0\| 8097<br>1.0\| 0.0\| 19 |

Πίνακας 6 Αποτελέσματα Use Case 1

| **Use Case 2** | | | |
|---|---|---|---|
| | f (1) Score | f (0.5) Score | Matrix Table |
| Logistic Regression | 0.97685 | 0.98973 | \|label\|lr_prediction\|count\|<br>+-----+-------------+-----+<br>\| 1.0\| 1.0\| 1266\|<br>\| 0.0\| 1.0\| 14\|<br>\| 1.0\| 0.0\| 46\|<br>\| 0.0\| 0.0\| 8428\| |
| SVM | 0.96995 | 0.98533 | \|label\|lsvc_prediction\|count\|<br>+-----+---------------+-----+<br>\| 1.0\| 1.0\| 1259\|<br>\| 0.0\| 1.0\| 25\|<br>\| 1.0\| 0.0\| 53\|<br>\| 0.0\| 0.0\| 8417\|<br>+-----+---------------+-----+ |

Πίνακας 7 Αποτελέσματα Use Case 2

| **Use Case 3** | | | |
|---|---|---|---|
| | f (1) Score | f (0.5) Score | Matrix Table |
| Logistic Regression | 0.97598 | 0.99023 | +-----+-------------+-----+<br>\|label\|lr_prediction\|count\|<br>+-----+-------------+-----+<br>\| 1.0\| 1.0\| 1260\|<br>\| 0.0\| 1.0\| 10\|<br>\| 1.0\| 0.0\| 52\|<br>\| 0.0\| 0.0\| 8432\|<br>+-----+-------------+-----+ |
| SVM | 0.96993 | 0.98551 | +-----+---------------+-----+<br>\|label\|lsvc_prediction\|count\|<br>+-----+---------------+-----+<br>\| 1.0\| 1.0\| 1258\|<br>\| 0.0\| 1.0\| 24\|<br>\| 1.0\| 0.0\| 54\|<br>\| 0.0\| 0.0\| 8418\|<br>+-----+---------------+-----+ |

Πίνακας 8 Αποτελέσματα Use Case 3

Παρατηρώντας τα αποτελέσματα βλέπουμε πως στο σύνολο των αποτελεσμάτων ο LR είναι πιο αποδοτικός στο συγκεκριμένο έργο αλλά γενικότερα και οι δύο επιλεγμένοι αλγόριθμοι παρουσιάζουν υψηλά ποσοστά ταξινόμησης. Πιο συγκεκριμένα στην “Use Case 1” ο LR απέδωσε σημαντικά καλύτερα έναντι του SVM και στις δύο βαθμολογίες f(1) και f(0.5). Παρολαυτά ο SVM πέτυχε την υψηλότερη βαθμολογία στις προβλέψεις των “True Positives” για την περίπτωση “Use Case 1. Στην “Use Case 2” όπου εφαρμόστηκε η μεταβλητή ισορροπίας βελτιώθηκαν και των δύο τα αποτελέσματα όπου και πάλι με μικρή διαφορά ο LR είχε τις καλύτερες βαθμολογίες. Ωστόσο φαίνεται πως η μεταβλητή ισορροπίας βοήθησε σημαντικά τον SVM καθώς εμφάνισε τεράστια βελτίωση συγκριτικά με τα αποτελέσματα του “Use Case 1” μειώνοντας αισθητά τα “False Positives”. Τέλος στο “Use Case 3” όπου εφαρμόστηκε η μεταβλητή γενίκευσης βελτιώθηκε οριακά η απόδοση και των δύο στην βαθμολογία f(0.5) αλλά παρουσίασαν μια επίσης οριακή πτώση στην βαθμολογία f(1). Παρατηρώντας και τον πίνακα της συγκεκριμένης περίπτωσης βλέπουμε πως μειώθηκαν οι σωστές προβλέψεις για τις περιπτώσεις των “True Positives” αλλά και των “False Positives”.

## 6. Σύνοψη

Η παρούσα εργασία παρουσίασε μια προσπάθεια επίλυσης του προβλήματος της αναγνώρισης πιθανών περιπτώσεων παιδικής εκμετάλλευσης σε δικτυακές συνομιλίες μηνυμάτων κειμένου με χρήση της μηχανής Apache Spark.Η

μεθοδολογία περιλαμβάνει τεχνικές καθαρισμού κειμένων, φιλτραρίσματος εγγραφών, εφαρμογή τεχνικών επεξεργασίας φυσικής γλώσσας (NLP), εφαρμογή αλγόριθμων ταξινόμησης και χρήση της βιβλιοθήκης MlLib του Apache Spark. Χρησιμοποιήθηκαν τρεις διαφορετικές κατηγορίες χαρακτηριστικών τα λεκτικά, τα συμπεριφορικά και τα ψυχολογικά τα οποία είναι δυνατό να αντληθούν από τα κείμενα των συνομιλιών. Η επιλογή των χαρακτηριστικών που επιλέχθηκαν καθορίστηκε από τα αποτελέσματα των εργαστηριακών δοκιμών. Για την εφαρμογή των παραπάνω χρειάστηκε η δημιουργία Lab με πόρους που παρείχε το κέντρο εθνικών υποδομών Έρευνας και τεχνολογίας (GRNET) σε μορφή IaaS. Τέλος για την αξιολόγηση των αποτελεσμάτων χρησιμοποιήθηκαν ή f(1) και η f(0.5).

### 6.2 Συμπεράσματα και Παραδοχές

Για να φτάσουμε σε ένα γενικό συμπέρασμα για το αν το εγχείρημα είχε επιτυχία πρέπει να αναφερθούμε σε κάποιες παραδοχές σχετικά με τα δεδομένα που χρησιμοποιήθηκαν, οι οποίες ενδέχεται να επηρεάζουν σε κάποιο βαθμό τα αποτελέσματα. Η πιο σημαντική παραδοχή είναι πως το Dataset δεν περιλαμβάνει συνομιλίες μεταξύ δράστη και θύματος καθώς δεν είναι εφικτή η δημοσιοποίηση τέτοιων πληροφοριών από τους αρμόδιους φορείς. Αντίθετα, περιλαμβάνει συνομιλίες μεταξύ δράστη και χειριστή που προσποιείται το θύμα όπως περιγράφτηκε στο κεφάλαιο 4.2. Επίσης η δεύτερη παραδοχή είναι πως για να κριθεί μια συνομιλία ως δεδομένο εισόδου προς ταξινόμηση έπρεπε να πληροί το κριτήριο της διάρκειας, δηλαδή να αποτελείται από έναν συγκεκριμένο αριθμό μηνυμάτων και το κριτήριο της αλληλεπίδρασης δηλαδή να συμμετέχουν και οι δύο συνομιλητές ενεργά στην συζήτηση. Η τελευταία παραδοχή έχει να κάνει με την γλώσσα των δεδομένων. Οι συνομιλίες είναι στα Αγγλικά οπότε αυτό καθιστά την παραπάνω μεθοδολογία μη ικανή να εφαρμοστεί καθολικά σε περιπτώσεις όπου οι συνομιλίες είναι σε κάποια άλλη γλώσσα καθώς θα παρουσιάζουν λεκτικές και γλωσσικές ιδιομορφίες που πρέπει να ληφθούν υπόψιν στην δημιουργία των χαρακτηριστικών.

Αξιολογώντας τα αποτελέσματα του εργαστηρίου κάνοντας τις παραπάνω παραδοχές καταλήγουμε στο γενικό συμπέρασμα πως η ταυτοποίηση πιθανών περιπτώσεων σεξουαλικής εκμετάλλευσης ανηλίκων σε μηνύματα κειμένου είναι

εφικτή και μάλιστα με υψηλή ακρίβεια πρόβλεψης με χρήση της πλατφόρμας Apache Spark.

### 6.3 Προοπτικές

Η παρούσα εργασία ασχολείται με ένα θέμα που απασχολεί πολλές βαθμίδες της κοινωνίας και παραμένει διαχρονικά επίκαιρο. Είναι σημαντικό πέρα από την κατανόηση του προβλήματος και η προσπάθεια ανάδειξης λύσεων που θα βοηθήσουν τους άμεσα εμπλεκόμενους να προστατεύσουν τα παιδιά. Σε αυτό το πλαίσιο η προτεινόμενη μεθοδολογία θα μπορούσε να εφαρμοστεί από αναλυτές της αστυνομίας ώστε να αυξήσουν την αποτελεσματικότητά τους μειώνοντας τον χρόνο απόκρισης στην εύρεση «ύποπτων συμπεριφορών». Ταυτόχρονα η χρήση περισσότερων δεδομένων στην εκπαίδευση αλλά και η δυναμική προσαρμογή της μεθοδολογίας σε νέες τάσεις που εμφανίζονται ανά εποχή (πχ αργκό, greeklish) θα συνέβαλε στην περεταίρω βελτίωση του μοντέλου. Επίσης σε ένα μοντέλο λειτουργίας προσανατολισμένο προς τον τελικό χρήστη θα μπορούσε να υλοποιηθεί μία εφαρμογή με την δυνατότητα εκτίμησης επικινδυνότητας συνομιλιών. Αυτή η εφαρμογή θα μπορούσε να απευθύνεται σε γονείς, δασκάλους και κοινωνικούς λειτουργούς ώστε να μπορούν να εφαρμόσουν δράσεις για την καλύτερη προστασία και καθοδήγηση του ανηλίκου.

Όλα τα παραπάνω συμπεράσματα και προοπτικές είναι καθαρά προσωπικές εκτιμήσεις και απόψεις βασιζόμενες στα αποτελέσματα της παρούσας ακαδημαϊκής εργασίας. Συνεπώς δεν λαμβάνονται υπόψη το νομικό πλαίσιο εφαρμογής ενός τέτοιου εγχειρήματος, ο ποινικός κώδικας και η εφαρμογή του, καθώς και η ηθική διάσταση του θέματος σχετικά με τις περιπτώσεις λάθους προβλέψεων.

# 7. Παράρτημα Α’

## 7.1 Κώδικας ανάπτυξης

Παρακάτω ο κώδικας που αναπτύχθηκε.

### train.ipynb

```
import findspark
findspark.init()
#for warning PYARROW_IGNORE TIMEZONE
import os
os.environ["PYARROW_IGNORE_TIMEZONE"] = "1"
import pyspark
import random
from pyspark.sql import SparkSession
#used for schema
from pyspark.sql.types import StructType, StructField, StringType, IntegerType
from decimal import Decimal
#pandasQ
import pyspark.pandas as ps
#used for filename on reading data
from pyspark.sql import functions as F


#spark session
spark = SparkSession.builder\
.config("spark.executor.instances", "10")\
.config("spark.executor.cores", "3")\
.config("spark.executor.memory", "2g")\
.config("spark.default.parallelism", "600")\
.config("spark.driver.maxResultSize","2g")\
.appName('train-stage1').getOrCreate()
#messages line by line with columns author , time , text , Filename
train_messages = spark.read.format("com.databricks.spark.xml") \
    .option("rowTag", "message").load("hdfs:///prof/train/export/*") \
    .withColumn("Filename",F.input_file_name())
train_messages = train_messages.withColumn("FIlename" , F.split("Filename","/"))
train_messages = train_messages.withColumn("FIlename" ,
F.reverse("Filename").getItem(0))
#conversation id and messages with Filename
train_conversations = spark.read.format("com.databricks.spark.xml") \
    .option("rowTag", "conversation").load("hdfs:///prof/train/export/*") \
    .withColumn("Filename1",F.input_file_name())
train_conversations = train_conversations.withColumn("FIlename1" ,
F.split("Filename1","/"))
train_conversations = train_conversations.withColumn("FIlename1" ,
F.reverse("Filename1").getItem(0))
###join train_conversation with train_messages
###Every Row represent a text message from the conversation
###We have multiple same ConversationID and Filename and Authors
##Columns
#1.ConversationID
#2.Filename
#3.author
#4.messageText
#5.time
```

```
#6.line
traindf1 = train_messages.groupBy("FIlename").count()
traindf1 = traindf1.withColumnRenamed("FIlename","FIlename1")
#join traindf1 which is conversations and number of messages with traindf
traindf = traindf1.join(train_messages , train_messages["FIlename"] ==
traindf1["FIlename1"] , "inner").drop("FIlename1")

#rename count to totalMesages
traindf = traindf.withColumnRenamed("count" , "totalMessages")

###FINAL RDD Columns
#1.ConversationID -> has duplicates because tis message bellongs to a Conversation ID
#2.Filename -> has duplicates because many messages come from a File
#3.author -> has duplicates becasue same author reply in a conversation
#4.messageText -> unique (problably)
#5.time -> With no interest at the momment
#6.totalMessages -> number of total messages between conversations
#7.line -> message Count. Last is total messages with no interest at the moment
##group by conversation id and authors in order to find number of authors in
conversation
#Every converasartion  ID must have  exactly two authors for our project's goals so by
grouping and counting coversationID and authors we can count number of authors per
conversation
traindf2 = traindf.groupBy("Filename" , "author").count()
#now lets count ID assuming that if we find id more than 3 times it means that there are
more than two authors
traindf2 = traindf2.groupBy("Filename").count()
#we want only 2
traindf2 = traindf2.withColumnRenamed("Filename","Filename1").where("count == 2")
#join traindf2 which is conversations and number of authors with traindf
traindf = traindf2.join(traindf , traindf["Filename"] == traindf2["Filename1"] ,
"inner").drop("Filename1")
#rename count to totalParticipants
traindf = traindf.withColumnRenamed("count" , "totalParticipants")
###FINAL RDD Columns
#1.ConversationID -> has duplicates because tis message bellongs to a Conversation ID
#2.Filename -> has duplicates because many messages come from a File
#3.author -> has duplicates becasue same author reply in a conversation
#4.messageText -> unique (problably)
#5.time -> With no interest at the momment
#6.totalMessages -> number of total messages between conversations
#7.line -> message Count. Last is total messages with no interest at the moment
#8.totalParticipants -> must be exactly 2
#saving in csv
traindf.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/im
port/train/1")
traindf.show(1)
#loading
schema = StructType([ \
    StructField("totalParticipants",StringType(),True), \
    StructField("totalMessages",StringType(),True), \
    StructField("line",StringType(),True), \
    StructField("author",StringType(),True), \
    StructField("text",StringType(),True), \
    StructField("time",StringType(),True), \
    StructField("Filename",StringType(),True) \
                    ])
#Reading CSV files from HDFS
train = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/import/train/1/*")
train.show()
```

```
###Loading truth file with only predators
truthDF =
spark.read.format("csv").options(delimiter='\t').load("hdfs:///prof/train_truth.txt")
###Now we use truth file that contains author and the String "pred" which icludes only
the predators and left outer join with our main rdd
##now we know who are predators and we are nonpredator
train2 = train.join(truthDF,train["author"] == truthDF["_c0"],"left").drop("_c0")
###Replace Null values with the String "NonP"
train = train2.na.fill("NonP" , subset=["_c1"])
train.show()
from pyspark.sql.functions import col,lit,concat
#unique key creation author@Filename
train = train.withColumn("Auth@File",concat(col("author"),lit("@"),col("Filename")))
###Use of the unique key with duplicates above so we can Number of messages for each
author on a conversation
train2 = train.groupBy("Auth@File").count()
train2.show()
filename = train.select(F.col("Filename")).withColumnRenamed("Filename"
,"Filename2").where(F.col("_c1") == 'pred')
filename = filename.dropDuplicates(["Filename2"])
train3 = train.join(filename,train["Filename"] == filename["Filename2"],"left")
###Columns Renaming
train2 = train2.withColumnRenamed("Auth@File","Auth@File2")
train2 = train2.withColumnRenamed("count","MsgNo")
###Inner Join train with rdd that contains message number per author in a Conversation
train3 = train2.join(train3 , train3["Auth@File"] == train2["Auth@File2"] ,
"inner").drop("Auth@File2")
###Droping some columns
train3 = train3.drop("line")
#train = train.drop("time")
train3 = train3.drop("totalParticipants")
train3 = train3.na.fill("Casual" , subset=["Filename2"])
from pyspark.sql.functions import collect_list
###GroupBy AuthFile and agg to merge all the texts in one raw per author
groupedDF = train3.groupby('Auth@File').agg(collect_list('text').alias("texts"))
groupedDF = groupedDF.withColumnRenamed("Auth@File","Auth@File2")
groupedDF.show()
groupedDF.count()
###Join the above groupBy rdd with our main train  RDD
train4 = train3.join(groupedDF , train3["Auth@File"] == groupedDF["Auth@File2"] ,
"inner").drop("Auth@File2")
train4.show()
###Droping Some columns
train4 = train4.drop("text")
train4 = train4.dropDuplicates(["Auth@File"])
train4.count()
###texts column above is arry and CSV format can tsupport it. So we cast it to String
before saving
train5 = train4.withColumn("texts" , col("texts").cast("string"))
train = train5.na.drop()
from pyspark.sql.functions import split , size
import pyspark.sql.functions as f
###Counting words of every author in conversation in new column wordCount
wordCount = train.withColumn("wordCount" , f.size(f.split(f.col('texts'), ' ')))
###Summing totalWords of both authors in a conversation to find total Words in
conversation
totalWords = wordCount.groupBy("Filename").sum("wordCount")
###Join above results
totalWords = totalWords.withColumnRenamed("Filename","Filename3")
totalWords = wordCount.join(totalWords , totalWords["Filename3"] ==
wordCount["Filename"] , "inner").drop("Filename3")
totalWords = totalWords.withColumnRenamed("sum(wordCount)","totalWords")
totalWords.show()
```

```
from pyspark.sql import functions as F
from pyspark.sql.functions import *
train = totalWords.withColumn("Role", \
                        when((F.col("_c1") == 'pred') & (F.col('Filename2') != 
'Casual') , 'pred') \
                                .otherwise(when((F.col("_c1") == 'NonP') & 
(F.col('Filename2') != 'Casual') , 'victim') \
                                .otherwise('Casual')))
train.show()
train=train.drop('Filename2')
train.count()
train.filter(F.col("_c1") == 'pred').count()
#saving in csv
train.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/impo
rt/train/2")
#loading
schema = StructType([ \
    StructField("MsgNo",IntegerType(),True), \
    StructField("totalMessages",IntegerType(),True), \
    StructField("author",StringType(),True), \
    StructField("time",StringType(),True), \
    StructField("Filename",StringType(),True), \
    StructField("_c1",StringType(),True), \
    StructField("Auth@File",StringType(),True), \
    StructField("texts",StringType(),True), \
    StructField("wordCount",IntegerType(),True), \
    StructField("totalWords",IntegerType(),True), \
    StructField("Role",StringType(),True) \
                  ])

#Reading CSV files from HDFS
train = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/import/train/2/*")
```

### test.ipynb

```
import findspark
findspark.init()
#for warning PYARROW_IGNORE TIMEZONE
import os
os.environ["PYARROW_IGNORE_TIMEZONE"] = "1"
import pyspark
import random
from pyspark.sql import SparkSession
#used for schema
from pyspark.sql.types import StructType, StructField, StringType, IntegerType
from decimal import Decimal
#pandasQ
import pyspark.pandas as ps
#used for filename on reading data
from pyspark.sql import functions as F
#spark session
spark = SparkSession.builder\
.config("spark.executor.instances", "10")\
.config("spark.executor.cores", "3")\
.config("spark.executor.memory", "1g")\
.config("spark.default.parallelism", "600")\
.config("spark.driver.maxResultSize","2g")\
.appName('train-stage1').getOrCreate()
```

```
#messages line by line with columns author , time , text , Filename
test_messages = spark.read.format("com.databricks.spark.xml") \
    .option("rowTag", "message").load("hdfs:///prof/test/export/*") \
    .withColumn("Filename",F.input_file_name())
test_messages = test_messages.withColumn("FIlename" , F.split("Filename","/"))
test_messages = test_messages.withColumn("FIlename" , F.reverse("Filename").getItem(0))
#conversation id and messages with Filename
test_conversations = spark.read.format("com.databricks.spark.xml") \
    .option("rowTag", "conversation").load("hdfs:///prof/test/export/*") \
    .withColumn("Filename1",F.input_file_name())
test_conversations = test_conversations.withColumn("FIlename1" ,
F.split("Filename1","/"))
test_conversations = test_conversations.withColumn("FIlename1" ,
F.reverse("Filename1").getItem(0))
###join train_conversation with train_messages
###Every Row represent a text message from the conversation
###We have multiple same ConversationID and Filename and Authors
##Columns
#1.ConversationID
#2.Filename
#3.author
#4.messageText
#5.time
#6.line
testdf1 = test_messages.groupBy("FIlename").count()
testdf1 = testdf1.withColumnRenamed("FIlename","FIlename1")
#join traindf1 which is conversations and number of messages with traindf
testdf = testdf1.join(test_messages , test_messages["FIlename"] == testdf1["FIlename1"]
, "inner").drop("FIlename1")

#rename count to totalMesages
testdf = testdf.withColumnRenamed("count" , "totalMessages")

###FINAL RDD Columns
#1.ConversationID -> has duplicates because tis message bellongs to a Conversation ID
#2.Filename -> has duplicates because many messages come from a File
#3.author -> has duplicates becasue same author reply in a conversation
#4.messageText -> unique (problably)
#5.time -> With no interest at the momment
#6.totalMessages -> number of total messages between conversations
#7.line -> message Count. Last is total messages with no interest at the moment
##group by conversation id and authors in order to find number of authors in
conversation
#Every converasartion  ID must have  exactly two authors for our project's goals
#so by grouping and counting coversationID and authors we can count number of authors
per conversation
testdf2 = testdf.groupBy("FIlename" , "author").count()
#now lets count ID assuming that if we find id more than 3 times it means that there are
more than two authors.
testdf2 = testdf2.groupBy("FIlename").count()
#we want only 2
testdf2 = testdf2.withColumnRenamed("FIlename","FIlename1").where("count == 2")
#join traindf2 which is conversations and number of authors with traindf
testdf = testdf2.join(testdf , testdf["FIlename"] == testdf2["FIlename1"] ,
"inner").drop("FIlename1")
#rename count to totalParticipants
testdf = testdf.withColumnRenamed("count" , "totalParticipants")
###FINAL RDD Columns
#1.ConversationID -> has duplicates because tis message bellongs to a Conversation ID
#2.Filename -> has duplicates because many messages come from a File
#3.author -> has duplicates becasue same author reply in a conversation
#4.messageText -> unique (problably)
#5.time -> With no interest at the momment
#6.totalMessages -> number of total messages between conversations
```

```
#7.line -> message Count. Last is total messages with no interest at the moment
#8.totalParticipants -> must be exactly 2
#saving in csv
testdf.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/imp
ort/test/1")
testdf.show(1)
#loading
schema = StructType([ \
    StructField("totalParticipants",StringType(),True), \
    StructField("totalMessages",StringType(),True), \
    StructField("line",StringType(),True), \
    StructField("author",StringType(),True), \
    StructField("text",StringType(),True), \
    StructField("time",StringType(),True), \
    StructField("Filename",StringType(),True) \
                    ])
#Reading CSV files from HDFS
test = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/import/test/1/*")
test.show()
###Loading truth file with only predators
truthDF =
spark.read.format("csv").options(delimiter='\t').load("hdfs:///prof/testtruth.txt")
#truthDF.show()
truthDF.show()
###No we use truth file that contains author and the String "pred" which icludes only
the predators and left outer join with our main rdd
##now we know who are predators and we are nonpredator
test2 = test.join(truthDF,test["author"] == truthDF["_c0"],"left").drop("_c0" , "_c1" ,
"_c2" , "_c3")
test2.show(10)
###Replace Null values with the String "NonP"
test = test2.na.fill("NonP" , subset=["_c4"])
#train.show()
test.filter(F.col("_c4") == 'pred').show()
from pyspark.sql.functions import col,lit,concat
#unique key creation author@Filename
test = test.withColumn("Auth@File",concat(col("author"),lit("@"),col("Filename")))
#train.show()
###Use of the unique key with duplicates above so we can Number of messages for each
author on a conversation
test2 = test.groupBy("Auth@File").count()
test2.show()
filename = test.select(F.col("Filename")).withColumnRenamed("Filename"
,"Filename2").where(F.col("_c4") == 'pred')
filename = filename.dropDuplicates(["Filename2"])
test3 = test.join(filename,test["Filename"] == filename["Filename2"],"left")
test3.filter(F.col("_c4") == 'pred').show()
###Columns Renaming
test2 = test2.withColumnRenamed("Auth@File","Auth@File2")
test2 = test2.withColumnRenamed("count","MsgNo")
###Inner Join train with rdd that contains message numbe rper author in a Conversation
test3 = test2.join(test3 , test3["Auth@File"] == test2["Auth@File2"] ,
"inner").drop("Auth@File2")
###Droping some columns
test3 = test3.drop("line")
#train = train.drop("time")
test3 = test3.drop("totalParticipants")
test3 = test3.na.fill("Casual" , subset=["Filename2"])
from pyspark.sql.functions import collect_list
###GroupBy AuthFile and agg to merge all the texts in one raw per author
groupedDF = test3.groupby('Auth@File').agg(collect_list('text').alias("texts"))
```

```
groupedDF = groupedDF.withColumnRenamed("Auth@File","Auth@File2")
groupedDF.show()
groupedDF.count()
###Join the above groupBy rdd with our main train  RDD
test4 = test3.join(groupedDF , test3["Auth@File"] == groupedDF["Auth@File2"] ,
"inner").drop("Auth@File2")
test4.show()
###Droping Some columns
test4 = test4.drop("text")
test4 = test4.dropDuplicates(["Auth@File"])
test4.count()
###texts column above is arry and CSV format can tsupport it. So we cast it to String
before saving
test5 = test4.withColumn("texts" , col("texts").cast("string"))
test = test5.na.drop()
from pyspark.sql.functions import split , size
import pyspark.sql.functions as f
###Counting words of every author in conversation in new column wordCount
wordCount = test.withColumn("wordCount" , f.size(f.split(f.col('texts'), ' ')))
###Summing totalWords of both authors in a conversation to find total Words in
conversation
totalWords = wordCount.groupBy("Filename").sum("wordCount")
###Join above results
totalWords = totalWords.withColumnRenamed("Filename","Filename3")
totalWords = wordCount.join(totalWords , totalWords["Filename3"] ==
wordCount["Filename"] , "inner").drop("Filename3")
totalWords = totalWords.withColumnRenamed("sum(wordCount)","totalWords")
totalWords.show()
from pyspark.sql import functions as F
from pyspark.sql.functions import *
test = totalWords.withColumn("Role", \
                        when((F.col("_c4") == 'pred') & (F.col('Filename2') !=
'Casual') , 'pred') \
                              .otherwise(when((F.col("_c4") == 'NonP') &
(F.col('Filename2') != 'Casual') , 'victim') \
                              .otherwise('Casual')))
test.show()
test=test.drop('Filename2')
test.count()
#saving in csv
test.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/impor
t/test/2")
#loading
schema = StructType([ \
    StructField("MsgNo",IntegerType(),True), \
    StructField("totalMessages",IntegerType(),True), \
    StructField("author",StringType(),True), \
    StructField("time",StringType(),True), \
    StructField("Filename",StringType(),True), \
    StructField("_c1",StringType(),True), \
    StructField("Auth@File",StringType(),True), \
    StructField("texts",StringType(),True), \
    StructField("wordCount",IntegerType(),True), \
    StructField("totalWords",IntegerType(),True), \
    StructField("Role",StringType(),True) \
                  ])

#Reading CSV files from HDFS
test = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/import/test/2/*")
```

### filtering.ipynb

```
import findspark
findspark.init()
import os
os.environ["PYARROW_IGNORE_TIMEZONE"] = "1"
import pyspark
import random
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType
from decimal import Decimal
import pyspark.pandas as ps
#used for filename on reading data
from pyspark.sql import functions as F
#spark session
spark = SparkSession.builder\
.config("spark.executor.instances", "10")\
.config("spark.executor.cores", "3")\
.config("spark.executor.memory", "2g")\
.config("spark.default.parallelism", "600")\
.config("spark.driver.maxResultSize","2g")\
.appName('train-stage1').getOrCreate()
#Loading
schema = StructType([ \
    StructField("MsgNo",IntegerType(),True), \
    StructField("totalMessages",IntegerType(),True), \
    StructField("author",StringType(),True), \
    StructField("time",StringType(),True), \
    StructField("Filename",StringType(),True), \
    StructField("_c1",StringType(),True), \
    StructField("Auth@File",StringType(),True), \
    StructField("texts",StringType(),True), \
    StructField("wordCount",IntegerType(),True), \
    StructField("totalWords",IntegerType(),True), \
    StructField("Role",StringType(),True) \
                    ])

#Reading CSV files from HDFS
train = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/import/train/2/*")

#Reading CSV files from HDFS
test = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/import/test/2/*")
train.count()
test.count()
train count = 91202 test count = 211077
###Filtering Conversations that has at least two authors###
test_conversations = test.withColumnRenamed("Filename" ,
"Filename2").groupBy("Filename2").count()
train_conversations = train.withColumnRenamed("Filename" ,
"Filename2").groupBy("Filename2").count()
test_conversations = test_conversations.filter(F.col("count") == 2)
train_conversations = train_conversations.filter(F.col("count") == 2)
train = train_conversations.join(train , train_conversations["Filename2"] ==
train["Filename"] , "inner").drop("Filename2").drop("count")
test = test_conversations.join(test , test_conversations["Filename2"] ==
test["Filename"] , "inner").drop("Filename2").drop("count")
train.count()
test.count()

train count = 90730 test count = 210058
```

```
train preds = 1088 test preds = 1887

train.show()
test.show()
###Filtering out Conversation with less than 30 messages###
train =  train.filter(F.col("totalMessages") > 30)
test = test.filter(F.col("totalMessages") > 30)

train.show()
#Must Filter Conversations that at least one participant has less than 10% of messages.
That means that at least one participant not talking as expected
#therefore pred cant be possible
#Carefull this method will filter out authros from conversations .
#So there might be some conversation with only one author. The one with less of 10%
participation will discarded
train = train.filter(F.col("MsgNo") > F.col("totalMessages")/10)
test = test.filter(F.col("MsgNo") > F.col("totalMessages")/10)
train = train.na.drop()
test = test.na.drop()
train.show()
#saving in csv
train.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/filt
ering/train")
test.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/filte
ring/test")

#Loading
schema = StructType([ \
    StructField("MsgNo",IntegerType(),True), \
    StructField("totalMessages",IntegerType(),True), \
    StructField("author",StringType(),True), \
    StructField("time",StringType(),True), \
    StructField("Filename",StringType(),True), \
    StructField("_c1",StringType(),True), \
    StructField("Auth@File",StringType(),True), \
    StructField("texts",StringType(),True), \
    StructField("wordCount",IntegerType(),True), \
    StructField("totalWords",IntegerType(),True), \
    StructField("Role",StringType(),True) \
                    ])

#Reading CSV files from HDFS
train = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/filtering/train/*")

#Reading CSV files from HDFS
test = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/filtering/test/*")
```

### features.ipynb

```
import findspark
findspark.init()

import os
os.environ["PYARROW_IGNORE_TIMEZONE"] = "1"
import pyspark
import random
from pyspark.sql import SparkSession
from pyspark.sql.types import StructType, StructField, StringType, IntegerType ,
FloatType
```

```
from decimal import Decimal
import pyspark.pandas as ps
#used for filename on reading data
from pyspark.sql import functions as F
#spark session
spark = SparkSession.builder\
.config("spark.executor.instances", "10")\
.config("spark.executor.cores", "3")\
.config("spark.executor.memory", "2g")\
.config("spark.default.parallelism", "600")\
.config("spark.driver.maxResultSize","2g")\
.appName('train-stage1').getOrCreate()
#loading
schema = StructType([ \
    StructField("MsgNo",IntegerType(),True), \
    StructField("totalMessages",IntegerType(),True), \
    StructField("author",StringType(),True), \
    StructField("time",StringType(),True), \
    StructField("Filename",StringType(),True), \
    StructField("_c1",StringType(),True), \
    StructField("Auth@File",StringType(),True), \
    StructField("texts",StringType(),True), \
    StructField("wordCount",IntegerType(),True), \
    StructField("totalWords",IntegerType(),True), \
    StructField("Role",StringType(),True) \
                    ])

#Reading CSV files from HDFS
train = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/filtering/train/*")

#Reading CSV files from HDFS
test = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/filtering//test/*")

###FEATURES ENGINERING WORD DOMINATION AND MESSAGE DOMINATION###
#MsgDom (Message Domination calculates percentage of messages for each author in a
conversation
train = train.withColumn('msgDom', F.col('MsgNo') / F.col('totalMessages'))
test = test.withColumn('msgDom', F.col('MsgNo') / F.col('totalMessages'))

#word domination is trying to calculate the author that uses more words than the other
#in that case i beleive its not correct to just use total author words / total
conversation words because it may be missleading
#we might have to involve number of messages as well
#Under investigation to see what method is the best

train = train.withColumn('wordDom', F.col('wordCount') / F.col('totalWords'))
test = test.withColumn('wordDom', F.col('wordCount') / F.col('totalWords'))
#round message Domination and word Domination
train =train.withColumn("msgDom", F.round(train["msgDom"], 2))
test =test.withColumn("msgDom", F.round(test["msgDom"], 2))

#round message Domination
train =train.withColumn("wordDom", F.round(train["wordDom"], 2))
test =test.withColumn("wordDom", F.round(test["wordDom"], 2))
###END OF FEATURES ENGINERING WORD DOMINATION AND MESSAGE DOMINATION###

###REMOVE PUNCTUATION###
from pyspark.sql import functions as F
from pyspark.sql.functions import *
```

```
train = train.withColumn('ClearTexts', translate('texts', '!"#$%&\'()*+,-
./:;<=>?@[\\]^_{|}~', ''))
test = test.withColumn('ClearTexts', translate('texts', '!"#$%&\'()*+,-
./:;<=>?@[\\]^_{|}~', ''))
###END OF REMOVE PUNCTUATION###
test.show(1)
train.show(1)

#FEATURE ENGINERING SEXWORDS ###
#the sexword list
sexwordsList = ["sex", "dick", "lick", "nude" , "mouth" , "body" , "ass" , "tits" ,
"boobs" ,"boobies" , "cock" , "shit" , "damm" , "bitch" , "crap" , "piss" , "naked",
"dicks" , \
                "pussy" , "kiss" , "fuck" , "cum" ,  "hot" ,  "baby" ,  "came" ,
"underwear" , "shaved" , "69" , "asshole" , "fag" , "slut"  , "crap", "balls" , "penis",
\
                "cock" ,"spank" ,"aslaslasl" ,"finger" , "horny" , "fingers" , "sexy" ,
"bra" , "brest" , "nipples" , "tit" , "underwear" , "rub" , "rubbing" , "arsehole" ,
"wet" ,   \
                "aslaslasl" , "aslaslaslasl" , "daddy" , "photo" , "foto" , "skirt" ,
"skirts" , "tip" , "big" , "hard" , "erected" , "blush" , "candy" , "sweet" , "sweety" ,
"anal" , "anus" , "butt" , "doggy" , "behind",\
                "lick" , "porn" , "youporn" , "pornhub" , "tongue" , "bj" , "blowjob" ,
"brazzers" , "dildo" , "condom" , "vibrator" , "sperm" , "cuffs" , "whip" , "leather" ,
"hun" , "honey" , "size" , "bra" , "small",\
                "bras" , "cup" , "nips" , "sex" , "sexy" , "hot" , "hotty" , "sexxy" ,
"yummy" , "yummi" , "yum" , "sexxxy" , "touch" , "taste" , "tasty" , "male" , "female" ,
"girl" , "legs" , "lube" , "vazelin" , "rub" , "cream",\
                "heels" , "wife" , "women" , "man" , "husband" , "fucking" , "sucking" ,
"suck" , "cumming"
               ]

def convertCase(str):
    resStr=""
    donothing =""
    arr = str.split(" ")
    for x in arr:
        if x in sexwordsList:
            resStr =resStr + " " + x
        else:
            donothing = 1
    return resStr

from pyspark.sql.functions import col, udf
from pyspark.sql.types import StringType , IntegerType
# Converting function to UDF
convertUDF = udf(lambda z: convertCase(z),StringType())

train = train.withColumn("sexVocab" , convertUDF(col("ClearTexts")) )
test = test.withColumn("sexVocab" , convertUDF(col("ClearTexts")) )

train.show(1)
test.show(1)
#FEATURE ENGINERING wh-WORDS ###
#the whword list

whwordsList = [ "what" , "who" ,"where" , "which" , "whom" , "whose" , "when" , "why" ,
"how" ]

def convertCase(str):
    resStr=""
    donothing =""
    arr = str.split(" ")
    for x in arr:
```

```
        if x in whwordsList:
            resStr =resStr + " " + x
        else:
            donothing = 1
    return resStr

from pyspark.sql.functions import col, udf
from pyspark.sql.types import StringType , IntegerType
# Converting function to UDF
convertUDF = udf(lambda z: convertCase(z),StringType())
train = train.withColumn("WhVocab" , convertUDF(col("ClearTexts")) )
test = test.withColumn("WhVocab" , convertUDF(col("ClearTexts")) )

#END OF FEATURE ENGINERING whWORDS ###

train.count()
test.count()

#FEATURE ENGINERING PRED WORDS - MOST COMMON#
#FINDING COMMON WORDS AMONG PREDS
trainMostWords = train.withColumn('words', F.explode(F.split(F.col('ClearTexts'), '
'')))\
    .groupBy('Auth@File' , '_c1' , 'words')\
   .agg(F.count('*').alias('word_cnt'))

testMostWords = test.withColumn('words', F.explode(F.split(F.col('ClearTexts'), ' ')))\
    .groupBy('Auth@File' , '_c1' , 'words')\
   .agg(F.count('*').alias('word_cnt'))
trainMostWords.show(1)
testMostWords.show(1)

#WE WANT TO SEE ONLY WORDS LARGER THAN 3
trainMostWords =trainMostWords.filter((F.length("words") > 4) & (F.col("_c1") ==
'pred'))
testMostWords =testMostWords.filter((F.length("words") > 4) & (F.col("_c1") == 'pred'))
trainMostWords1 = trainMostWords.groupBy("words").count()
testMostWords1 = testMostWords.groupBy("words").count()

trainMostWords1.filter(F.col("count") > 50).show()
testMostWords1.filter(F.col("count") > 150).show()

#the PREDwORD list
prwordsList = [ "still" , "online" ,"could" , "today" , "gonna" , "wanna" , "night" ,
"again" , "going" , "doing" , "there" , "thats" , "really" , "anything"  , "cell" ,
"about" , "think" , "right" , "online" , "still", \
              "sorry" , "would" , "bout" , "school" , "wanna" , "stuff"]

def convertCase(str):
    resStr=""
    donothing =""
    arr = str.split(" ")
    for x in arr:
        if x in prwordsList:
            resStr =resStr + " " + x
        else:
            donothing = 1
    return resStr

from pyspark.sql.functions import col, udf
from pyspark.sql.types import StringType , IntegerType
# Converting function to UDF
convertUDF = udf(lambda z: convertCase(z),StringType())
train = train.withColumn("PrVocab" , convertUDF(col("ClearTexts")) )
test = test.withColumn("PrVocab" , convertUDF(col("ClearTexts")) )
```

```
train.show(1)
test.show(1)
#FEND OF FEATURE ENGINERING PRED WORDS - MOST COMMON#

#saving in csv
train.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/features/train")
#saving in csv
test.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/features/test")

#Loading
schema = StructType([ \
    StructField("MsgNo",IntegerType(),True), \
    StructField("totalMessages",IntegerType(),True), \
    StructField("author",StringType(),True), \
    StructField("time",StringType(),True), \
    StructField("Filename",StringType(),True), \
    StructField("_c1",StringType(),True), \
    StructField("Auth@File",StringType(),True), \
    StructField("texts",StringType(),True), \
    StructField("wordCount",IntegerType(),True), \
    StructField("totalWords",IntegerType(),True), \
    StructField("Role",StringType(),True), \
    StructField("msgDom",FloatType(),True), \
    StructField("wordDom",FloatType(),True), \
    StructField("ClearTexts",StringType(),True), \
    StructField("sexVocab",StringType(),True), \
    StructField("WhVocab",StringType(),True), \
    StructField("PrVocab",StringType(),True) \
                    ])
  #Reading CSV files from HDFS
train = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/features/train")

#Reading CSV files from HDFS
test = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/features/test")
train = train.na.fill("")
test = test.na.fill("")
train.count()
test.count()
###Bigrams - Trigrams###

stopwordList = ["<conversation" , "id=" , "</conversation>" , "AMIRITESO" , "MMMMM" , "LIKE" , "WHO" , "runescape" ,"account", "type" , "a9b326df4e6da61c5b6f5e1058be83a2", "puddi", "bastardfuck" , "puddipuddi", "claiming", \
                "http" , "https" , "amp", "damagelets" , "mmmmmmmmmmmmmmm"  , "bananas", "index", "completehtml", "source", "anlfjadfkandkadgajdjadgahkbdasjd", "lt3lt3",\
                "niggercocks" , "org" , "com" , "http" , "https" , "img" , "eateryou", "12x4312x800101error" , "ciarans" , "malefemale" , "worldfuck", "erectile" , "dysfunction" , "pleaserubbery" , "dirty", \
               "FAGGOT" , "OBAMA" ,  "FAGGOt"  , ":)," , "beeyetch" , "BANANAS" , "striengers", "httpwwwfacebookcomeventphpeid118620571524764" , "lyingstrangers" , \
                "?," , "[hi," , "Fixme.ps3" , "Shitbox" , "&gt;"  , "fixmeps3" , "mmmm", "chatting" , "random" , "stranger" , \
               "5c5b806fbd1826340209616ddb9ed767" , "e03aa9707bd13f180c517ae1a47e9da2",  "htmlspec" \
              ]
from pyspark.sql import functions as F
from pyspark.sql.functions import *
```

```
import pyspark.ml.feature
from pyspark.ml.feature import Tokenizer , StopWordsRemover , CountVectorizer , IDF ,
RegexTokenizer , NGram
StopWordsRemover.loadDefaultStopWords("english")
stopwordList.extend(StopWordsRemover().getStopWords())
stopwordList = list(set(stopwordList))
# Stages For the Pipeline pre processing
regexTokenizer = RegexTokenizer(inputCol='ClearTexts',outputCol='mytokens' ,
pattern="[\\p{L}\\w]+" , minTokenLength=3 , toLowercase = True , gaps=False)
#regex tokenizer for ages
regexTokenizer2 = RegexTokenizer(inputCol='ClearTexts',outputCol='age_tokens' ,
pattern="[\\p{L}\\w]+" , toLowercase = True , gaps=False)
#stop words remover excludes words that appear frequently and dont carry any meaning
stopwords_remover =
StopWordsRemover(inputCol='mytokens',outputCol='regex_filtered_tokens',
stopWords=stopwordList)
ngram = NGram(n=3, inputCol="regex_filtered_tokens", outputCol="trigrams")
bigram = NGram(n=2, inputCol="regex_filtered_tokens", outputCol="bigrams")
age_ngrams = NGram(n=2, inputCol="age_tokens", outputCol="age_bigrams")
train = regexTokenizer.transform(train)
train = stopwords_remover.transform(train)

train = regexTokenizer2.transform(train)
test = regexTokenizer2.transform(test)

test = regexTokenizer.transform(test)
test = stopwords_remover.transform(test)
train = ngram.transform(train)
test = ngram.transform(test)
train = bigram.transform(train)
test = bigram.transform(test)
train = age_ngrams.transform(train)
test = age_ngrams.transform(test)
test = test.drop("mytokens" , "age_tokens" , "regex_filtered_tokens")
train = train.drop("mytokens" , "age_tokens" , "regex_filtered_tokens")
train.show()
test.show()
###END OF BIGRAMS - TRIGRAMS###

train.count()
test.count()

###REMOVE SPAMS###

train_nonspam = train.withColumn('words', F.explode(F.col('trigrams')))\
    .groupBy("Filename" , 'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('word_cnt') > 5)

test_nonspam =test.withColumn('words', F.explode(F.col('trigrams')))\
    .groupBy("Filename" , 'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('word_cnt') > 5)
#test_nonspam.filter(F.col("word_cnt") > 20).show()
train_nonspam =train_nonspam.drop("word_cnt")
test_nonspam =test_nonspam.drop("word_cnt")

train_nonspam = train_nonspam.dropDuplicates(["Filename"])
test_nonspam = test_nonspam.dropDuplicates(["Filename"])
train_nonspam2 = train_nonspam.withColumn("spam", lit(1))
test_nonspam2 = test_nonspam.withColumn("spam", lit(1))
train_nonspam2.show()
train_nonspam2.count()
```

```
#join spams words results for train and test with main Dataset
train_nonspam2 = train_nonspam2.withColumnRenamed("Filename","Filename2")
test_nonspam2 = test_nonspam2.withColumnRenamed("Filename","Filename2")

train = train.join(train_nonspam2 , train["Filename"] == train_nonspam2["Filename2"] ,
"Leftouter").drop("Filename2")
test = test.join(test_nonspam2 , test["Filename"] == test_nonspam2["Filename2"] ,
"Leftouter").drop("Filename2")
train.count()
train = train.filter(F.col("spam").isNull())
test = test.filter(F.col("spam").isNull())
train = train.drop("spam" , "words")
test = test.drop("spam" , "words")
train.count()
test.filter(F.col("Role") == 'Casual').count()
test.count()
###END OF REMOVE SPAMS###

#the UNDERAGE LIST

under_ages = ["eleven" , "twelve" ,"thirteen" , "fourteen" , "fifteen" ,"sixteen",
"seventeen" , "am11" , "am12" , "am13" , "am14" , "am15" , "am16" , "am17",
"12","13","14","15","16","17",\
        "m11" , "m12" , "m13" , "m14" , "m15" , "m16" , "m17", "11m" , "12m" , "13m" ,
"14m" , "15m" , "16m" , "17m",\
        "f11" , "f12" , "f13" , "f14" , "f15" , "f16" , "f17", "11f" , "12f" , "13f" ,
"14f" , "15f" , "16f" , "17f",\
        "f 17", "f 16" , "f 15" , "f 14", "f 13" , "f 12", "f 11" ,"m 17", "m 16" , "m
15" , "m 14", "m 13" , "m 12", "m 11" ,\
         "17 f", "16 f" , "15 f" , "14 f", "13 f" , "12 f", "11 f" ,"17 m", "16 m" , "15
m" , "14 m", "13 m" , "12 m", "11 m", \
         "17 female", "16 female" , "15 female" , "14 female", "13 female" , "12
female", "11 female" ,"17 male", "16 male" , "15 male" , "14 male", "13 male" , "12
male", "11 male", \
        "female 17", "female 16" , "female 15" , "female 14", "female 13" , "female 12",
"female 11" ,"male 17", "male 16" , "male 15" , "male 14", "male 13" , "male 12", "male
11" , \
         "im 11" , "im 12" , "im 13" , "im 14" , "im 15" , "im 16" , "im 17" \
       ]

over_ages = ["eighteen" , "nineteen" ,"twenty" , "twentyone" , "twentytwo"
,"twentythree", "twentyfour" , "twentyfive" , "twentysix" , "twentyseven" ,
"twentyeight" , "twentynine" , "thirty" , "18" , "19" , "20" , "21" , "22" , "23" , "24"
, "25" ,"26","27","28","29","30","31", \

"32","33","34","35","36","37","38","39","40","41","42","43","44","45","am18" , "am19" ,
"am20" , "am21" , "am22" , "am23" , "am24", "am25" , "am26" , "am27" , "am28" , "am29" ,
"am30" , "am31" , "am32" , "am33" , "am34" , "am35" , "am36" , "am37" , "am38" , "am39"
, "am40",\
            "m18" , "m19" , "m20" , "m21" , "m22" , "m23" , "m24" , "m25" , "m26" ,
"m27" , "m28" , "m29" , "m30" , "m31" , "m32" , "m33" , "m34" , "m35" , "m36" , "m37" ,
"m38" , "m39" , "m40" ,\
            "18m" , "19m" , "20m" , "21m" , "22m" , "23m" , "24m" , "25m" , "26m" ,
"27m" , "28m" , "29m" , "30m" , "31m" , "32m" , "33m" , "34m" , "35m" , "36m" , "37m" ,
"38m" , "39m" , "40m" ,\
            "f18" , "f19" , "f20" , "f21" , "f22" , "f23" , "f24" , "f25" , "f26" ,
"f27" , "f28" , "f29" , "f30" , "f31" , "f32" , "f33" , "f34" , "f35" , "f36" , "f37" ,
"f38" , "f39" , "f40" ,\
            "18f" , "19f" , "20f" , "21f" , "22f" , "23f" , "24f" , "25f" , "26f" ,
"27f" , "28f" , "29f" , "30f" , "31f" , "32f" , "33f" , "34f" , "35f" , "36f" , "37f" ,
"38f" ,"39f" , "40f" ,\
            "f 18", "f 19" , "f 20" , "f 21", "f 22" , "f 23", "f 24" ,"f 25", "f 26" ,
"f 27" , "f 28", "f 29" , "f 30", "f 31" ,"f 32" ,"f 33" ,"f 34" ,"f 35" ,"f 36" ,"f 37"
,"f 38" ,"f 39" ,"f 40" ,\
```

```
            "m 18", "m 19" , "m 20" , "m 21", "m 22" , "m 23", "m 24" ,"m 25", "m 26" ,
"m 27" , "m 28", "m 29" , "m 30", "m 31" ,"m 32" ,"m 33" ,"m 34" ,"m 35" ,"m 36" ,"m 37"
,"m 38" ,"m 39" ,"m 40" ,\
             "18 f", "19 f" , "20 f" , "21 f", "22 f" , "23 f", "23 f" ,"25 f", "26 f" ,
"27 f" , "28 f", "29 f" , "30 f", "31 f" ,"32 f", "33 f" , "34 f" , "35 f", "36 f" , "37
f", "38 f" , "39 f" , "40 f",\
             "18 m", "19 m" , "20 m" , "21 m", "22 m" , "23 m", "23 m" ,"25 m", "26 m" ,
"27 m" , "28 m", "29 m" , "30 m", "31 m" ,"32 m", "33 m" , "34 m" , "35 m", "36 m" , "37
m", "38 m" , "39 m" , "40 m",\
             "18 female", "19 female" , "20 female" , "21 female", "22 female" , "23
female", "24 female" ,"25 female", "26 female" , "27 female" , "28 female", "29 female"
, "30 female", "31 female" ,\
             "32 female", "33 female" , "34 female" , "35 female", "36 female" , "37
female", "38 female" ,"39 female", "40 female" ,\
             "18 male", "19 male" , "20 male" , "21 male", "22 male" , "23 male", "24
male" ,"25 male", "26 male" , "27 male" , "28 male", "29 male" , "30 male", "31 male" ,\
             "32 male", "33 male" , "34 male" , "35 male", "36 male" , "37 male", "38
male" ,"39 male", "40 male" ,\
            "female 18", "female 19" , "female 20" , "female 21", "female 22" , "female
23", "female 24" ,"female 25", "female 26" , "female 27" , "female 28", "female 29" ,
"female 30", "female 31", \
            "female 32" , "female 33", "female 34" ,"female 35", "female 36" , "female
37" , "female 38", "female 39" , "female 40", "female 41", \
            "male 18", "male 19" , "male 20" , "male 21", "male 22" ,"male 23" , "male
24" ,"male 25", "male 26" , "male 27" , "male 28", "male 29" , "male 30", "male 31", \
            "male 32" , "male 33", "male 34" ,"male 35", "male 36" ,"male 37" , "male
38" , "male 39" ,"male 40" , "male 41",\
            "im 17" , "im 18" , "im 19" , "im 20" , "im 21" , "im 22" , "im 23" , "im
24" , "im 25" , "im 26" , "im 27" , "im 28" , "im 29" , "im 30" , "im 31" , "im 32" ,
"im 33" , "im 34" , "im 35" , "im 36" , "im 37" , "im 38" , "im 39" , "im 40"\
       ]
train.show(1)
train_under_ages = train.withColumn('words', F.explode(F.split(F.col('ClearTexts'), '
')))\
    .groupBy('author' ,'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('words').isin(under_ages))

test_under_ages = test.withColumn('words', F.explode(F.split(F.col('ClearTexts'), '
')))\
    .groupBy('author' ,'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('words').isin(under_ages))

train_under_ages_bigrams = train.withColumn('words', F.explode(F.col('age_bigrams')))\
    .groupBy('author' ,'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('words').isin(under_ages))

test_under_ages_bigrams = test.withColumn('words', F.explode(F.col('age_bigrams')))\
    .groupBy('author' ,'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('words').isin(under_ages))
train_over_ages = train.withColumn('words', F.explode(F.split(F.col('ClearTexts'), '
')))\
    .groupBy('author' ,'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('words').isin(over_ages))

test_over_ages = test.withColumn('words', F.explode(F.split(F.col('ClearTexts'), ' ')))\
    .groupBy('author' ,'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('words').isin(over_ages))
```

```
train_over_ages_bigrams = train.withColumn('words', F.explode(F.col('age_bigrams')))\
    .groupBy('author' ,'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('words').isin(over_ages))

test_over_ages_bigrams = test.withColumn('words', F.explode(F.col('age_bigrams')))\
    .groupBy('author' ,'words')\
   .agg(F.count('*').alias('word_cnt'))\
   .where(F.col('words').isin(over_ages))
train_under_ages_bigrams.count()
train_under_ages.count()
train_over_ages.count()
train_over_ages_bigrams.count()
train_under  = train_under_ages.union(train_under_ages_bigrams)
test_under  = test_under_ages.union(test_under_ages_bigrams)

train_over = train_over_ages.union(train_over_ages_bigrams)
test_over = test_over_ages.union(test_over_ages_bigrams)
train_under.count()
train_over.show()

train_under = train_under.withColumn("age", \
                        when((F.col("word_cnt") > 0) , 'U'))

test_under = test_under.withColumn("age", \
                        when((F.col("word_cnt") > 0) , 'U'))

train_over = train_over.withColumn("age", \
                        when((F.col("word_cnt") > 0) , 'O'))

test_over = test_over.withColumn("age", \
                        when((F.col("word_cnt") > 0) , 'O'))
train_under = train_under.withColumnRenamed("author","author2").drop("words","word_cnt")
test_under = test_under.withColumnRenamed("author","author2").drop("words","word_cnt")

train_over= train_over.withColumnRenamed("author","author2").drop("words","word_cnt")
test_over = test_over.withColumnRenamed("author","author2").drop("words","word_cnt")

test_under= test_under.dropDuplicates(["author2"])
train_under= train_under.dropDuplicates(["author2"])
test_over= test_over.dropDuplicates(["author2"])
train_over= train_over.dropDuplicates(["author2"])
train_age  = train_under.union(train_over)
test_age   = test_under.union(test_over)
train_under.count()
train_age= train_age.dropDuplicates(["author2"])
test_age= test_age.dropDuplicates(["author2"])
###END OF FEATURE ENGINERING UNDERAGE/ADULTS ###

train = train.join(train_age , train["author"] == train_age["author2"] ,
"left").drop("author2")
test= test.join(test_age , test["author"] == test_age["author2"] ,
"left").drop("author2")
train = train.na.fill("X" , subset=["age"])
test = test.na.fill("X" , subset=["age"])
train.count()
test.count()

train.select("bigrams").filter(F.col('_c1') == 'pred').show(10 , False)

###DICTATE WORDS - PHRASES###

negatives = ["dont tell" , "dont say" , "tell anyone" , "tell parents","our
secret","dont trust","trust me","didnt mean" ,\
```

```
        "wont tell" , "dont want" , "dont leave" , "your secret" , "new pics" , "trust"
, "secret" , "bad" , "sorry" , "dont" \
        ]
def convertCase(unig , bigr):
    resStr=""
    donothing =""
    arr = unig.split(" ")
    bigrams = " ".join(bigr)
    bi_arr = bigrams.split(" ")
    for x in arr:
        if x in negatives:
            resStr =resStr + " " + x
        else:
            donothing = 1
    for y in bi_arr:
        if y in negatives:
            resStr =resStr + " " + y
        else:
            donothing = 1 (10)
    return resStr

from pyspark.sql.functions import col, udf
from pyspark.sql.types import StringType , IntegerType

# Converting function to UDF

convertUDF = udf(lambda z , y: convertCase(z , y),StringType())
train = train.withColumn("NegVocab" , convertUDF(col("ClearTexts") , col("bigrams")) )
test = test.withColumn("NegVocab" , convertUDF(col("ClearTexts") , col("bigrams")) )
train.show()
test.show()
train = train.drop("age_bigrams")
test = test.drop("age_bigrams")
train.show(1)
test.show(1)
train = train.withColumn("trigrams" , col("trigrams").cast("string"))
test = test.withColumn("trigrams" , col("trigrams").cast("string"))

train = train.withColumn("bigrams" , col("bigrams").cast("string"))
test = test.withColumn("bigrams" , col("bigrams").cast("string"))
#saving in csv
test.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/featu
res/r2/test")
#saving in csv
train.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/feat
ures/r2/train")
train.show(1)
#Loading
schema = StructType([ \
    StructField("MsgNo",IntegerType(),True), \
    StructField("totalMessages",IntegerType(),True), \
    StructField("author",StringType(),True), \
    StructField("time",StringType(),True), \
    StructField("Filename",StringType(),True), \
    StructField("_c1",StringType(),True), \
    StructField("Auth@File",StringType(),True), \
    StructField("texts",StringType(),True), \
    StructField("wordCount",IntegerType(),True), \
    StructField("totalWords",IntegerType(),True), \
    StructField("Role",StringType(),True), \
    StructField("msgDom",FloatType(),True), \
    StructField("wordDom",FloatType(),True), \
    StructField("ClearTexts",StringType(),True), \
    StructField("sexVocab",StringType(),True), \
```

```
    StructField("WhVocab",StringType(),True), \
    StructField("PrVocab",StringType(),True), \
    StructField("trigrams",StringType(),True), \
    StructField("bigrams",StringType(),True), \
    StructField("age",StringType(),True), \
    StructField("NegVocab",StringType(),True) \
                    ])
  #Reading CSV files from HDFS
train = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/features/r2/train")

#Reading CSV files from HDFS
test = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/features/r2/test")


###BUILDING FULL TEST CSV ###
from pyspark.sql.functions import collect_list
train_conv =
train.groupby('Filename').agg(collect_list('ClearTexts').alias("full_texts"))
test_conv = test.groupby('Filename').agg(collect_list('ClearTexts').alias("full_texts"))
train_authors = train.groupby('Filename','author','_c1', 'Auth@File' , 'wordCount' ,
'totalwords' , 'sexVocab','age' , 'WhVocab' ,'PrVocab' , 'NegVocab' , 'bigrams' ,
'trigrams').count()
test_authors = test.groupby('Filename','author','_c1', 'Auth@File', 'wordCount' ,
'totalwords' , 'sexVocab','age' , 'WhVocab' , 'PrVocab' , 'NegVocab' , 'bigrams' ,
'trigrams').count()
train_authors.count()
test_authors.count()
train_authors = train_authors.withColumnRenamed("Filename" , "Filename2")
test_authors = test_authors.withColumnRenamed("Filename" , "Filename2")
train_conv_join = train_authors.alias("authors1").join(train_authors.alias("authors2") ,
train_authors["Filename2"] == train_authors["Filename2"] , "inner")\
.select("authors1.Auth@File","authors1.author" , "authors1.Filename2" , "authors1._c1" ,
"authors1.wordCount" , "authors1.totalwords" , "authors1.sexVocab" , "authors1.age" ,\
        "authors1.WhVocab" , "authors1.PrVocab" , "authors1.NegVocab" ,
"authors1.bigrams" , "authors1.trigrams" ,\
        F.col("authors2.Auth@File").alias("Auth@File2"),
F.col("authors2.author").alias("author2") , F.col("authors2._c1").alias("_c2")
,F.col("authors2.wordCount").alias("wordCount2") ,
F.col("authors2.sexVocab").alias("sexVocab2"),\

F.col("authors2.WhVocab").alias("WhVocab2"),F.col("authors2.PrVocab").alias("PrVocab2"),
F.col("authors2.NegVocab").alias("NegVocab2"),F.col("authors2.bigrams").alias("bigrams2"
),F.col("authors2.trigrams").alias("trigrams2"),\
       F.col("authors2.age").alias("age2"))
test_conv_join = test_authors.alias("authors1").join(test_authors.alias("authors2") ,
test_authors["Filename2"] == test_authors["Filename2"] , "inner")\
.select("authors1.Auth@File","authors1.author" , "authors1.Filename2" , "authors1._c1" ,
"authors1.wordCount" , "authors1.totalwords" , "authors1.sexVocab" , "authors1.age" ,\
        "authors1.WhVocab" ,"authors1.PrVocab", "authors1.NegVocab" , "authors1.bigrams"
, "authors1.trigrams" ,\
        F.col("authors2.Auth@File").alias("Auth@File2"),
F.col("authors2.author").alias("author2") , F.col("authors2._c1").alias("_c2")
,F.col("authors2.wordCount").alias("wordCount2") ,
F.col("authors2.sexVocab").alias("sexVocab2"),\

F.col("authors2.WhVocab").alias("WhVocab2"),F.col("authors2.PrVocab").alias("PrVocab2"),
F.col("authors2.NegVocab").alias("NegVocab2"),F.col("authors2.bigrams").alias("bigrams2"
),F.col("authors2.trigrams").alias("trigrams2"),\
       F.col("authors2.age").alias("age2"))
train_conv_join = train_conv_join.filter(F.col("author") != F.col("author2"))
```

```
test_conv_join = test_conv_join.filter(F.col("author") != F.col("author2"))
test_conv_join.count()
train_conv_join.count()

from pyspark.sql.functions import col,lit,concat
train_conv_join = train_conv_join.dropDuplicates(["Filename2"])
test_conv_join = test_conv_join.dropDuplicates(["Filename2"])
train_conv_join.show(1)
train_conv_join.filter(F.col("_c1") == 'NonP').show()
train_fconv = train_conv_join.join(train_conv , train_conv_join["Filename2"] ==
train_conv["Filename"] , "inner")
test_fconv = test_conv_join.join(test_conv , test_conv_join["Filename2"] ==
test_conv["Filename"] , "inner")
train_fconv.count()
train_fconv.show(1)
train = train_fconv.withColumn("texts" , col("full_texts").cast("string"))
train.show(1)

from pyspark.sql.functions import split , size
import pyspark.sql.functions as f
train= train_fconv.withColumn("ArrayOfString", col("full_texts").cast("string"))
test= test_fconv.withColumn("ArrayOfString", col("full_texts").cast("string"))
train = train.drop("full_texts")
test = test.drop("full_texts")
test.show(1)

from pyspark.sql import functions as F
from pyspark.sql.functions import *
train = train.withColumn('texts2', F.explode(F.split(F.col('ArrayOfString'), ' ')))\
        .where(length(col('texts2')) < 13)\


test = test.withColumn('texts2', F.explode(F.split(F.col('ArrayOfString'), ' ')))\
        .where(length(col('texts2')) < 13)

train.count()
test.count()
train = train.drop("texts2")
test = test.drop("texts2")
train.count()
test.count()
train = train.dropDuplicates(["Filename"])
test = test.dropDuplicates(["Filename"])
train.select("ArrayOfString").show(10 , False)
train.count()
test.count()
train.show(1)
test.show(1)

#saving in csv
train.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/feat
ures/fulltext/train")
#saving in csv
test.write.format("com.databricks.spark.csv").mode("overwrite").save("hdfs:///prof/featu
res/fulltext/test")
train.show(1)
#loading
schema = StructType([ \
    StructField("Auth@File",StringType(),True), \
    StructField("author",StringType(),True), \
    StructField("Filename2",StringType(),True), \
    StructField("_c1",StringType(),True), \
    StructField("wordCount",IntegerType(),True), \
    StructField("totalwords",IntegerType(),True), \
```

```
    StructField("sexVocab",StringType(),True), \
    StructField("age",StringType(),True), \
    StructField("WhVocab",StringType(),True), \
    StructField("PrVocab",StringType(),True), \
    StructField("NegVocab",StringType(),True), \
    StructField("bigrams",StringType(),True), \
    StructField("trigrams",StringType(),True), \
    StructField("Auth@File2",StringType(),True), \
    StructField("author2",StringType(),True), \
    StructField("_c2",StringType(),True), \
    StructField("wordCount2",IntegerType(),True), \
    StructField("sexVocab2",StringType(),True), \
    StructField("Whvocab2",StringType(),True), \
    StructField("Prvocab2",StringType(),True), \
    StructField("NegVocab2",StringType(),True), \
    StructField("bigrams2",StringType(),True), \
    StructField("trigrams2",StringType(),True), \
    StructField("age2",StringType(),True), \
    StructField("Filename",StringType(),True), \
    StructField("texts",StringType(),True) \
                    ])
  #Reading CSV files from HDFS
train = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/features/fulltext/train")

#Reading CSV files from HDFS
test = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/features/fulltext/test")
test.show()
test.filter(F.col("_c1") == "pred").count()
test.filter(F.col("_c1") == "NonP").count()
test.count()



def convertCase(str):
    resStr=""
    arr = str.split(" ")
    for x in arr:
        if len(x) < 13:
            resStr =resStr + " " + x
    return resStr
from pyspark.sql.functions import col, udf
from pyspark.sql.types import StringType , IntegerType
# Converting function to UDF
convertUDF = udf(lambda z: convertCase(z),StringType())

train = train.withColumn("cleanTexts", \
    convertUDF(col("texts")).alias("word") ) \
   .show(400 ,truncate=False)
train = train.withColumn("classWeightCol2", udf_calc(F.col('Chat')))


train.select("texts").show(10 , False)

train.show(10 , False)
```

**classification.ipynb**

```
import findspark
findspark.init()
#for warning PYARROW_IGNORE TIMEZONE
import os
os.environ["PYARROW_IGNORE_TIMEZONE"] = "1"
import pyspark
import random
from pyspark.sql import SparkSession
#used for schema
from pyspark.sql.types import StructType, StructField, StringType, IntegerType ,
FloatType
from decimal import Decimal
#pandasQ
import pyspark.pandas as ps
#used for filename on reading data
from pyspark.sql import functions as F
#spark session
spark = SparkSession.builder\
.config("spark.executor.instances", "10")\
.config("spark.executor.cores", "3")\
.config("spark.executor.memory", "2g")\
.config("spark.default.parallelism", "600")\
.config("spark.driver.maxResultSize","4g")\
.appName('train-stage1').getOrCreate()
#Loading
schema = StructType([ \
    StructField("Auth@File",StringType(),True), \
    StructField("author",StringType(),True), \
    StructField("Filename2",StringType(),True), \
    StructField("_c1",StringType(),True), \
    StructField("wordCount",IntegerType(),True), \
    StructField("totalwords",IntegerType(),True), \
    StructField("sexVocab",StringType(),True), \
    StructField("age",StringType(),True), \
    StructField("WhVocab",StringType(),True), \
    StructField("PrVocab",StringType(),True), \
    StructField("NegVocab",StringType(),True), \
    StructField("bigrams",StringType(),True), \
    StructField("trigrams",StringType(),True), \
    StructField("Auth@File2",StringType(),True), \
    StructField("author2",StringType(),True), \
    StructField("_c2",StringType(),True), \
    StructField("wordCount2",IntegerType(),True), \
    StructField("sexVocab2",StringType(),True), \
    StructField("Whvocab2",StringType(),True), \
    StructField("PrVocab2",StringType(),True), \
    StructField("NegVocab2",StringType(),True), \
    StructField("bigrams2",StringType(),True), \
    StructField("trigrams2",StringType(),True), \
    StructField("age2",StringType(),True), \
    StructField("Filename",StringType(),True), \
    StructField("texts",StringType(),True) \
                    ])
  #Reading CSV files from HDFS
train = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/features/fulltext/train")

#Reading CSV files from HDFS
test = spark.read.format("com.databricks.spark.csv") \
    .schema(schema) \
    .load("hdfs:///prof/features/fulltext/test")

train.count()
```

```
test.count()

#Function for removing word larger than 13 characters#

def convertCase(str):
    resStr=""
    donothing =""
    arr = str.split(" ")
    for x in arr:
        if len(x) > 13 :
            donothing = 1
        else:
            resStr =resStr + " " + x
    return resStr

from pyspark.sql.functions import col, udf
from pyspark.sql.types import StringType , IntegerType

# Converting function to UDF
convertUDF = udf(lambda z: convertCase(z),StringType())

train = train.withColumn("cleanTexts" , convertUDF(col("texts")) )
test = test.withColumn("cleanTexts" , convertUDF(col("texts")) )
test.show(1)

from pyspark.sql import functions as F
from pyspark.sql.functions import *

train = train.withColumn("Chat", \
                         when((F.col("_c1") == 'pred') | (F.col("_c2") == 'pred') ,
'pred') \
                          .otherwise('NonP'))
test = test.withColumn("Chat", \
                         when((F.col("_c1") == 'pred') | (F.col("_c2") == 'pred') ,
'pred') \
                          .otherwise('NonP'))
train.count()
test.count()

### ADULT - UNDERAGE ###
train = train.withColumn("AtoU", \
                         when((F.col("age") == 'O') & (F.col("age2") == 'U') , 1) \
                         .when((F.col("age2") == 'O') & (F.col("age") == 'U') , 1) \
                          .otherwise(0))
test = test.withColumn("AtoU", \
                       when((F.col("age") == 'O') & (F.col("age2") == 'U') , 1) \
                       .when((F.col("age2") == 'O') & (F.col("age") == 'U') , 1) \
                          .otherwise(0))
train.show(1000)
test.show(1)

### word domination ###
from pyspark.sql.functions import abs
train = train.withColumn("WordDom", F.col("wordCount") / F.col("totalwords"))
train = train.withColumn("WordDom2", F.col("wordCount2") / F.col("totalwords"))

test = test.withColumn("WordDom", F.col("wordCount") / F.col("totalwords"))
test = test.withColumn("WordDom2", F.col("wordCount2") / F.col("totalwords"))
train = train.withColumn("DomDif", abs(F.col("WordDom") - F.col("WordDom2")))
test = test.withColumn("DomDif", abs(F.col("WordDom") - F.col("WordDom2")))
train.show(1)
test.show(1)
```

```
balance_ratio_pred = 4299 / ( 2 * 340)
balance_ratio_NonP = 4299 / ( 2 * 3959)
#balance_ratio = 740 / 1000
balance_ratio = (340/4299) * 10

def weightsCal(str):
    if (str == 'pred'):
       x =   balance_ratio
    else:
       x = (1.0 - balance_ratio)
    return x

from pyspark.sql.types import IntegerType
from pyspark.sql.functions import udf
udf_calc = udf(weightsCal, StringType())
train = train.withColumn("classWeightCol2", udf_calc(F.col('Chat')))
train = train.withColumn("classWeightCol", F.round(train["classWeightCol2"], 2))

stopwordList = ["<conversation","id=","</conversation>" ,
"ya","u","i","oh","uh","p","d","k","yes","no","ur","ok","m","1","2","3","4","5","6","7",
"8","9","0",\
                "5c5b806fbd1826340209616ddb9ed767" , "e03aa9707bd13f180c517ae1a47e9da2"
, "htmlspec" , "malefemale" , "FAGGOT" , "FAGGOt" , ",îÇ,ñí,ñí,îÇ" , "00100000" ,
"PUDDIPUDDI" , "PUDDI" , "01110100"\
               ]
train = train.na.fill("")
test = test.na.fill("")

#classification

import pyspark.ml.feature
from pyspark.ml.feature import Tokenizer , StopWordsRemover , CountVectorizer , IDF ,
RegexTokenizer
from pyspark.ml.feature import StringIndexer , NGram
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.feature import OneHotEncoder

from pyspark.ml.feature import Word2Vec

from pyspark.ml import Pipeline
from pyspark.ml.regression import DecisionTreeRegressor
from pyspark.ml.feature import VectorIndexer
from pyspark.ml.evaluation import RegressionEvaluator

from pyspark.ml.classification import NaiveBayes
from sklearn.metrics import confusion_matrix

from pyspark.ml.feature import PCA
from pyspark.ml.linalg import Vectors

import pyspark.ml.feature
from pyspark.ml.feature import StringIndexer
StopWordsRemover.loadDefaultStopWords("english")
stopwordList.extend(StopWordsRemover().getStopWords())
stopwordList = list(set(stopwordList))

test.show(1)

from pyspark.sql.functions import split, col
train = train.withColumn("trigrams", split(col("trigrams"),","))
test = test.withColumn("trigrams", split(col("trigrams"),","))
train = train.withColumn("trigrams2", split(col("trigrams2"),","))
test = test.withColumn("trigrams2", split(col("trigrams2"),","))
```

```
train = train.withColumn("bigrams", split(col("bigrams"),","))
test = test.withColumn("bigrams", split(col("bigrams"),","))
train = train.withColumn("bigrams2", split(col("bigrams2"),","))
test = test.withColumn("bigrams2", split(col("bigrams2"),","))
train = train.withColumn("WhVocab", split(col("WhVocab"),","))
test = test.withColumn("WhVocab", split(col("WhVocab"),","))
train = train.withColumn("sexVocab", split(col("sexVocab"),","))
test = test.withColumn("sexVocab", split(col("sexVocab"),","))
train = train.withColumn("NegVocab", split(col("NegVocab"),","))
test = test.withColumn("NegVocab", split(col("NegVocab"),","))
train = train.withColumn("PrVocab", split(col("PrVocab"),","))
test = test.withColumn("PrVocab", split(col("PrVocab"),","))
train = train.withColumn("Whvocab2", split(col("Whvocab2"),","))
test = test.withColumn("Whvocab2", split(col("Whvocab2"),","))
train = train.withColumn("sexVocab2", split(col("sexVocab2"),","))
test = test.withColumn("sexVocab2", split(col("sexVocab2"),","))
train = train.withColumn("NegVocab2", split(col("NegVocab2"),","))
test = test.withColumn("NegVocab2", split(col("NegVocab2"),","))
train = train.withColumn("PrVocab2", split(col("PrVocab2"),","))
test = test.withColumn("PrVocab2", split(col("PrVocab2"),","))
train.show(1)
test.show(1)
labelEncoder = StringIndexer(inputCol='Chat' , outputCol='label')

# Stages For the Pipeline 2
#tokenizer breaks sentences to words
tokenizer = Tokenizer(inputCol='cleanTexts',outputCol='mytokens')
stopwords_remover = StopWordsRemover(inputCol='mytokens',outputCol='filtered_tokens',
stopWords=stopwordList)

trigram = NGram(n=3, inputCol="filtered_tokens", outputCol="text_Trigrams")
tri_vectorizer = CountVectorizer(inputCol='text_Trigrams',outputCol='rawTrigrams'
,minDF=2,vocabSize=200)
tri_idf = IDF(inputCol='rawTrigrams',outputCol='Tri_vectorizedFeatures')

wh_vectorizer = CountVectorizer(inputCol="WhVocab",outputCol='wh_raw')
sex_vectorizer = CountVectorizer(inputCol="sexVocab",outputCol='sex_raw')
neg_vectorizer = CountVectorizer(inputCol="NegVocab",outputCol='neg_raw')

wh_vectorizer2 = CountVectorizer(inputCol="Whvocab2",outputCol='wh_raw2')
sex_vectorizer2 = CountVectorizer(inputCol="sexVocab2",outputCol='sex_raw2')
neg_vectorizer2 = CountVectorizer(inputCol="NegVocab2",outputCol='neg_raw2')

wh_idf = IDF(inputCol='wh_raw',outputCol='wh_idf')
sex_idf = IDF(inputCol='sex_raw',outputCol='sex_idf')
neg_idf = IDF(inputCol='neg_raw',outputCol='neg_idf')

wh_idf2 = IDF(inputCol='wh_raw2',outputCol='wh_idf2')
sex_idf2 = IDF(inputCol='sex_raw2',outputCol='sex_idf2')
neg_idf2 = IDF(inputCol='neg_raw2',outputCol='neg_idf2')

vectorizer = CountVectorizer(inputCol='filtered_tokens',outputCol='rawFeatures'
,minDF=3,vocabSize=1000)
idf = IDF(inputCol='rawFeatures',outputCol='vectorizedFeatures')
#combines Features
assembler = VectorAssembler(inputCols=["DomDif","wh_raw","sex_raw","neg_raw", \
                                       "wh_raw2","sex_raw2","neg_raw2","rawFeatures", \

"wh_idf","sex_idf","neg_idf","wh_idf2","sex_idf2","neg_idf2",

"vectorizedFeatures","rawTrigrams","Tri_vectorizedFeatures"],outputCol="assfeatures")
from pyspark.ml.classification import LinearSVC
lsvc = LinearSVC(featuresCol='assfeatures',labelCol='label' , maxIter=5 ,standardization
= True , weightCol = 'classWeightCol' , regParam = 0.09 )
```

```
lsvc.setPredictionCol("lsvc_prediction")
lsvc.setRawPredictionCol("lsvc_rawprediction")
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol = 'assfeatures', labelCol = 'label', maxIter=5 ,
standardization = True , weightCol = 'classWeightCol' , regParam = 0.09)
lr.setPredictionCol("lr_prediction")
lr.setProbabilityCol("lr_probability")
lr.setRawPredictionCol("lr_rawprediction")
from pyspark.ml import Pipeline
pipeline = Pipeline(stages=[labelEncoder , tokenizer ,
stopwords_remover,trigram,tri_vectorizer, \

tri_idf,wh_vectorizer,sex_vectorizer,neg_vectorizer,wh_vectorizer2, \

sex_vectorizer2,neg_vectorizer2,wh_idf,sex_idf,neg_idf,wh_idf2,sex_idf2, \
                            neg_idf2,vectorizer,idf, assembler,lr,lsvc])
#Building model
main_model = pipeline.fit(train)
predictions = main_model.transform(test)
predictions.groupBy('label','lsvc_prediction').count().show()
predictions.groupBy('label','lr_prediction').count().show()
from sklearn.metrics import f1_score
y_true = predictions.select("label").collect()
y_pred_lr = predictions.select("lr_prediction").collect()
y_pred_lsvc = predictions.select("lsvc_prediction").collect()
f1_score(y_true, y_pred_lr)
f1_score(y_true, y_pred_lsvc)
from sklearn.metrics import fbeta_score
fbeta_score(y_true, y_pred_lr, average='macro', beta=0.5)
fbeta_score(y_true, y_pred_lsvc, average='macro', beta=0.5)
```

### 7.2 Οδηγίες Δημιουργίας Lab

Για την δημιουργία του lab δημιουργήθηκε cluster μεγέθους 11 εικονικών μηχανημάτων σε τοπικό δίκτυο . Σε κάθε μηχάνημα ακολουθήθηκε η διαδικασία που περιγράφεται παρακάτω

1. Εγκατάσταση Ubuntu 16.04 και update – upgrade σε 20.04.

```
sudo apt update
sudo apt upgrade
sudo apt dist-upgrade
sudo do-release-upgrade
```

2. Ενεργοποίηση Firewall και δημιουργία κανόνα για την πόρτα 22 – ssh

```
sudo ufw allow 22
sudo ufw enable
```

3. Απενεργοποίηση IPV6

```
sudo sysctl -w net.ipv6.conf.all.disable_ipv6=1
sudo sysctl -w net.ipv6.conf.default.disable_ipv6=1
sudo sysctl -w net.ipv6.conf.lo.disable_ipv6=1
```

4. Εγκατάσταση ssh , pdsh , java 8

```
sudo apt-get install ssh
sudo apt-get install pdsh
```

```
sudo apt-get install openjdk-8-jdk
```

5. Δημιουργία νέου χρήστη και γκρουπ

```
sudo addgroup hadoop
sudo adduser --ingroup hadoop prof
sudo usermod -a -G sudo prof
```

6. Παραμετροποίηση SSH

```
ssh-keygen -t rsa -P ""
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
ssh localhost
```

7. Εγκατάσταση HADOOP 3.3.1 (Δημιουργία καταλόγων , δικαιώματα κλπ)

```
Κατέβασμα Hadoop (wget) από την επίσημη σελίδα
sudo tar -xzvf hadoop-3.3.1.tar.gz
sudo mv hadoop-3.3.1 /usr/local/hadoop
sudo chown prof:hadoop -R /usr/local/hadoop

sudo mkdir -p /usr/local/hadoop_tmp/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_tmp/hdfs/datanode
sudo chown prof:hadoop -R /usr/local/hadoop_tmp/
```

8. Επεξεργασία αρχείου bashrc για τον χρήστη που δημιουργήθηκε (Οι ρυθμίσεις είναι και για Hadoop και για Spark)

```
sudo vi ~/.bashrc
```

Προσθήκη των παρακάτω στο τέλος του αρχείου

```
export PDSH_RCMD_TYPE=ssh
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

export JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64"
PATH=$PATH:$HOME/bin:$JAVA_HOME/bin
export PATH


#spark
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export SPARK_HOME=/opt/spark
export PATH=$PATH:$SPARK_HOME/bin:$SPARK_HOME/sbin
export PYSPARK_PYTHON=/usr/bin/python3
export YARN_CONF_DIR=$HADOOP_HOME/etc/hadoop
```

```
source ~/.bashrc
```

9. Ενημέρωση αρχείου hosts σε κάθε μηχάνημα με τις διευθύνσεις του τοπικού δικτύου και των hostnames

```
sudo gedit /etc/hosts
```

```
    10.0.0.1    HadoopMaster
    10.0.0.2    HadoopSlave1
    10.0.0.3    HadoopSlave2
    10.0.0.4    HadoopSlave3
    10.0.0.5    HadoopSlave4
    10.0.0.6    HadoopSlave5
    10.0.0.7    HadoopSlave6
    10.0.0.8    HadoopSlave7
    10.0.0.9    HadoopSlave8
    10.0.0.10   HadoopSlave9
    10.0.0.11   HadoopSlave10
```

10. Επεξεργασία αρχείων core-site.xml , hdfs-site.xml , yarn-site.xml , mapred-site.xml , workers , Hadoop-env.sh του Hadoop που βρίσκονται στο $HADOOP_HOME/etc/hadoop

**Core-site**

```
<configuration>
      <property>
      <name>fs.default.name</name>
      <value>hdfs://HadoopMaster:9000</value>
   </property>
</configuration>
```

**Hdfs-site**

```
<configuration>
<property>
      <name>dfs.replication</name>
      <value>3</value>
   </property>

   <property>
      <name>dfs.namenode.name.dir</name>
      <value>file:/usr/local/hadoop_tmp/hdfs/namenode</value>
   </property>

   <property>
      <name>dfs.datanode.data.dir</name>
      <value>file:/usr/local/hadoop_tmp/hdfs/datanode</value>
   </property>
<property>
<name>dfs.block.size</name>
<value>67108864</value>
</property>
</configuration>
```

**Yarn-site**

```
<configuration>

  <property>
    <name>yarn.resourcemanager.resource-tracker.address</name>
    <value> HadoopMaster:8025</value>
  </property>

  <property>
    <name>yarn.resourcemanager.scheduler.address</name>
    <value> HadoopMaster:8035</value>
  </property>

  <property>
    <name>yarn.resourcemanager.address</name>
    <value> HadoopMaster:8050</value>
  </property>
<!-- Site specific YARN configuration properties -->

<property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>256</value>
  </property>

  <property>
    <name>yarn.scheduler.maximum-allocation-mb</name>
    <value>6144</value>
  </property>

 <property>
    <name>yarn.scheduler.maximum-allocation-vcores</name>
    <value>6</value>
  </property>

<property>
    <name>yarn.nodemanager.resource.cpu-vcores</name>
    <value>6</value>
  </property>

 <property>
    <name>yarn.nodemanager.recource.memory-mb</name>
    <value>6144</value>
  </property>
 <property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
  </property>

 <property>
      <name>yarn.nodemanager.pmem-check-enabled</name>
```

```
    <value>false</value>
  </property>
</configuration>
```

Mapred-site

```
<configuration>
<property>
    <name>mapreduce.job.tracker</name>
    <value> HadoopMaster:5431</value>
  </property>

 <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Workers

```
HadoopSlave1
HadoopSlave2
HadoopSlave3
HadoopSlave4
HadoopSlave5
HadoopSlave6
HadoopSlave7
HadoopSlave8
HadoopSlave9
HadoopSlave10
```

Hadoop-env

Εύρεση της γραμμής

```
export JAVA_HOME=
```

και την αλλάζουμε με την παρακάτω

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

11. Από τον masternode εκτελούμε το παρακάτω για να αποθηκεύσουμε το rsa κλειδί ώστε να γινεται αυτόματα η σύνδεση ssh

```
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave1
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave2
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave3
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave4
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave5
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave6
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave7
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave8
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave9
ssh-copy-id -i $HOME/.ssh/id_rsa.pub prof@HadoopSlave10
```

12. Namenode format

```
hdfs namenode -format
```

## 13. Εγκατάσταση SPARK και Scala

Wget https://www.apache.org/dyn/closer.lua/spark/spark-3.4.0/spark-3.4.0-bin-hadoop3.tgz
#tar xvf spark-*
sudo mv spark-3.4.0-bin-hadoop3 opt/spark
sudo apt install scala git -y
sudo chown prof:hadoop -R /opt/spark

## 14. Επεξεργασία $SPARK_HOME/conf/spark-defaults.conf και workers

```
 spark.master   yarn
 spark.deploy.mode      cluster
 spark.driver.memory    4g
 spark.yarn.am.memory   2g
 spark.num.executors    40
 spark.executor.instaces 4

 spark.executor.memory 512m
 spark.driver.maxResultSize    1g

spark.jars.packages com.databricks:spark-
csv_2.10:1.3.0,com.databricks:spark-
xml_2.12:0.14.0,com.johnsnowlabs.nlp:spark-nlp_2.12:4.1.0
```

workers

```
HadoopSlave1
HadoopSlave2
HadoopSlave3
HadoopSlave4
HadoopSlave5
HadoopSlave6
HadoopSlave7
HadoopSlave8
HadoopSlave9
HadoopSlave10
```

## 15. Απαιτούμενα

sudo apt install python3-pip
pip install pyspark
pip install jupyterlab
pip install sklearn
pip install matplotlib
pip install nltk
pip install koala
pip install pandas
pip install findspark
pip install pyarrow
Openblass on all nodes
sudo apt-get install libopenblas-base
sudo update-alternatives --config libblas.so.3

# 8. Βιβλιογραφία

1. *Sexual Grooming of Children: Review of Literature.* **Craven, Samantha , Sarah Brown and Elizabeth Gilchrist.** 2006, Journal of Sexual Aggression, pp. 289-299.

2. *Identifying Child Molesters: Preventing Child Sexual Abuse by Recognizing the Patterns of Offenders.* **Van Dam, Carla.** 2001, Binghamton, NY: The Haworth Press.

3. *Examining the Modus Operandi of Sexual Offenders Against Children and Its Practical Implications.* **Leclerc, et al.** 2009, Aggression and Violent Behavior 14(1):5–12.

4. *Setting 'em up': Personal, Familial and Institutional Grooming in the Sexual Abuse of Children.* **McAlinden and Anne-Marie.** 2006, Social & Legal Studies, pp. 339-362.

5. *Stages of Sexual Grooming: Recognizing Potentially Predatory Behaviors of Child Molesters.* **Jeglic, Winters, Georgia M. and Elizabeth L.** 2016, Deviant Behavior.

6. **Kraizer, Dr.** safechild.org. *Τοποθεσία web safechild.org.* [Online] https://safechild.org/understanding-grooming/.

7. **Καρμαλής , Μάριος.** *Τεχνικές κατηγοριοποίησης για την εξάλειψη θορύβου απο αλγόριθμους συσταδοποίησης.* Πάτρα : Πανεπιστήμιο Πατρών, 2022.

8. **Sirsat, Manisha.** manisha-sirsat.blogspot. [Online] Απρίλιος 29, 2019. https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html.

9. **PAN.** Sexual Predator Identification 2012. *pan.webis.de.* [Online] 2012. https://pan.webis.de/clef12/pan12-web/sexual-predator-identification.html.

10. *An Apache Spark Implementation for SentimentAnalysis on Twitter Data.* **Alexandros, Baltas, Kanavos, Andreas and Tsakalidis, Athanasios.** 2016, Springer.

11. **Lüthe, Marvin.** GitHub. [Online] https://github.com/luethe007/Udacity_DSND_Spark.

12. **Spark, Apache.** Apache Spark. *Τοποθεσία Web του Apache Spark.* [Online] https://spark.apache.org/docs/latest/.

13. **Hadoop, Apache.** Apache Hadoop. *Τοποθεσία Web του Apache Hadoop.* [Online] https://hadoop.apache.org/.

14. **MohammadrezaEbrahimi.** *Automatic Identification of OnlinePredators in Chat Logs by Anomaly Detection and Deep Learning.* Quebec Canada : Concordia University, 2016.

15. **Mintz, Amanda.** *"WHO'S GOT THE POWER? "GAINING AND GRANTINGDOMINANCE IN CONVERSATION.* Waco Texas : Baylor University, 2014.

16. **Hadoop.org.** hadoop.apache.org. *https://hadoop.apache.org/.* [Online] https://hadoop.apache.org/.

17. **Thomas, Alex.** *Natural Language Processing with Spark NLp.* s.l. : O'Reilly.

18. **Xiangrui Meng, et al.** *MLlib: Machine Learning in Apache Spark.* s.l. : Journal of Machine Learning Research , 2016.

19. **Jeffrey Shafer, Scott Rixner and Alan L. Cox.** The Hadoop distributed filesystem: Balancing portability and performance. *IEEE.* March 2010.

20. **Chieh-Yen Lin, et al.** *Large-scale logistic regression and linear support vector machines using spark.* Washington, DC, USA : IEEE, 2014.

21. *Large Scale Implementations for TwitterSentiment Classification.* **Kanavos, Andreas, et al.** algorithms.

22. *Text Classification Algorithms: A Survey.* **Kowsari, Kamran, et al.** 2019, information.

23. *Spark NLP: Natural Language Understanding at Scale.* **Veysel, Kocaman and Talby, David.** 2021, Elsevier - Software Impacts.

24. **WeenKus.** github. *https://github.com/Weenkus/sexual-predator-classification-using-ensemble-classifiers/blob/master/notebooks/libs/profanity.txt.* [Online]

25. **V Srinivas Jonnalagadda, et al.** A Review Study of Apache Spark in Big Data Processing. *International Journal of Computer Science Trends and Technology (IJCST).* 2016, Vol. Volume 4.