



Ελληνικό Ανοικτό Πανεπιστήμιο
Σχολή Θετικών Επιστημών και Τεχνολογίας
Πρόγραμμα Σπουδών: Πληροφορική

Τεχνικές Ταξινόμησης/Συσταδοποίησης Μεγάλων Συλλογών
Διαδικτυακών Δεδομένων
HOU-CS-UGP-2016-9

Πτυχιακή Εργασία

του

ΚΩΝΣΤΑΝΤΙΝΟΥ Ν. ΜΠΑΖΑΚΟΥ

Επιβλέπων: Ιωάννης Αναγνωστόπουλος
Αναπληρωτής Καθηγητής

Πάτρα, Ιούλιος 2017



Ελληνικό Ανοικτό Πανεπιστήμιο
Σχολή Θετικών Επιστημών και Τεχνολογίας
Πρόγραμμα Σπουδών: Πληροφορική

Τεχνικές Ταξινόμησης/Συσταδοποίησης Μεγάλων Συλλογών
Διαδικτυακών Δεδομένων
HOU-CS-UGP-2016-9

Πτυχιακή Εργασία

του

ΚΩΝΣΤΑΝΤΙΝΟΥ Ν. ΜΠΑΖΑΚΟΥ

Επιβλέπων: Ιωάννης Αναγνωστόπουλος
Αναπληρωτής Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 15η Ιουλίου 2017.

(Υπογραφή)

(Υπογραφή)

(Υπογραφή)

.....
Ιωάννης Αναγνωστόπουλος
Αναπληρωτής Καθηγητής
Πανεπιστήμιο Θεσσαλίας

.....
Βασίλειος Πλαγιανάκος
Αναπληρωτής Καθηγητής
Πανεπιστήμιο Θεσσαλίας

.....
Χρήστος Νικόλαος Αναγνωστόπουλος
Αναπληρωτής Καθηγητής
Πανεπιστήμιο Αιγαίου

Πάτρα, Ιούλιος 2017



Ελληνικό Ανοικτό Πανεπιστήμιο
Σχολή Θετικών Επιστημών και Τεχνολογίας
Πρόγραμμα Σπουδών: Πληροφορική

Copyright © – All rights reserved Κωνσταντίνος Μπαζάκος, 2017.

Με την επιφύλαξη παντός δικαιώματος.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα.

Το περιεχόμενο αυτής της εργασίας δεν απηχεί απαραίτητα τις απόψεις του Τμήματος, του Επιβλέποντα, ή της επιτροπής που την ενέχρινε.

Τπεύθυνη Δήλωση

Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας, και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών Πληροφορικής του Ελληνικού Ανοικτού Πανεπιστημίου.

(Τπογραφή)

.....
Κωνσταντίνος Μπαζάκος

Περίληψη

Το γενικό θεωρητικό αντικείμενο που αναπτύσσεται σε αυτήν την εργασία είναι η μηχανική μάθηση και εφαρμογές της όπως (classification/clustering). Η μηχανική μάθηση θα μπορούσε να περιγραφεί ως η διαδικασία εκείνη κατά την οποία ένα σύστημα βελτιώνεται συνεχώς ως προς την απόδοση του, κατά την διάρκεια εκτέλεσης μιας συγκεκριμένης εργασίας, χωρίς να απαιτείται ο εκ νέου προγραμματισμός του. Σύμφωνα με αυτήν την προσέγγιση, σκοπός της είναι η δημιουργία συστημάτων που διαθέτουν ικανότητες μάθησης και βελτίωσης της απόδοσής τους σε ορισμένους τομείς μέσω της αξιοποίησης προηγούμενης γνώσης και εμπειρίας.

Αρχικά γίνεται αναφορά στα σημαντικότερα σημεία της θεωρίας όπως η ανάκτηση της πληροφορίας, ο διανυσματικός χώρος, μέθοδοι μηχανικής μάθησης, έννοιες απόδοσης (ανάληση και ακρίβεια) και οι αλγόριθμοι που εφαρμόζονται στα πλαίσια της εργασίας.

Επειτα παρουσιάζονται οι πηγές δεδομένων που χρησιμοποιήθηκαν, η διαδικασία της απαύγουμενης προεπεξεργασίας των δεδομένων καθώς και οι διαδικασίες εμπλουτισμού του dataset Amazon Movies Reviews με τα ground truth labels.

Στην συνέχεια αντιμετωπίζονται δύο διαφορετικά προβλήματα, αυτό της ταξινόμησης και της συσταδοποίησης. Και στις δύο αυτές περιπτώσεις χρησιμοποιούνται οι πηγές δεδομένων που αναλύθηκαν εκτενώς στο τρίτο κεφάλαιο. Στο πρόβλημα της ταξινόμησης εφαρμόζονται πέντε διαφορετικά μοντέλα (Naive Bayes, Random Forest, Logistic Regression, K-nearest neighbors και Support vector machine) ενώ για την υλοποίηση της συσταδοποίησης χρησιμοποιείται η LDA και η μέθοδος μέσω των πινάκων—μασκών συνάφειας. Σε όλες τις περιπτώσεις παρατίθενται τα αποτελέσματα των αλγορίθμων καθώς και κάποιες γραφικές παραστάσεις όπου ήταν εφικτό.

Τέλος παρουσιάζονται αξιολογήσεις των αποτελεσμάτων των μοντέλων που υλοποιήθηκαν και για τα δύο προβλήματα. Ο σχολιασμός γίνεται ανά πηγή δεδομένων και ανά μοντέλο. Λαμβάνονται υπόψη οι τιμές των μετρικών της ακρίβειας, ανάλησης καθώς και η τιμή της f1-score. Τέλος αναφέρονται κάποια γενικά συμπεράσματα.

Το σύνολο του κώδικα βρίσκεται δημοσιευμένο σε Git Repository [6].

Λέξεις Κλειδιά

Μηχανική μάθηση, Επιβλεπόμενη μάθηση, Μή επιβλεπόμενη μάθηση, Ταξινόμηση, Συσταδοποίηση, Εξόρυξη πληροφορίας

Abstract

This project's object is to study and develop mechanisms to address the problem of large Internet data collections classification/clustering. Existing algorithms will also be examined relating to the classification/clustering of large collections of Web objects (websites, blog posts, social media posts etc) with traditional textual analysis algorithms in the information retrieval field but also with methods of structure identification/representation.

All the code written for this thesis is available on Git Repository [6].

Keywords

Machine learning, Supervised Learning, Unsupervised Learning, Information Retrieval, Data mining, Classification, Clustering, Labeled data, Python, IPython Notebook

στην Μαρία

Ευχαριστίες

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου στον επιβλέποντα της εργασίας καθηγητή κο Αναγνωστόπουλος Ιωάννη. Όχι απλώς μου έδωσε την ευκαιρία να κάνω μια πολύ ενδιαφέρουσα πτυχιακή εργασία, αλλά κατάφερε να με οδηγήσει στο να κάνουμε κατά τη γνώμη μου σημαντική δουλειά χωρίς να με πιέσει ούτε στο ελάχιστο.

Επίσης ευχαριστώ τον φίλο και μεταπτυχιακό φοιτητή του Τμήματος ΗΜΜΥ Δ.Π.Θ Δήμητρα Μάρκο για τις συμβουλές του και την καθοδήγησή του.

Τέλος θα ήθελα να ευχαριστήσω την σύζυγο μου Κατερίνα για την στήριξη και την υπομονή της.

Περιεχόμενα

Περίληψη	i
Abstract	iii
Ευχαριστίες	vii
Περιεχόμενα	xi
Κατάλογος Σχημάτων	xvi
Κατάλογος Πινάκων	xvii
1 Εισαγωγή	1
1.1 Αντικείμενο της διπλωματικής	1
1.2 Οργάνωση του τόμου	1
1.3 Συνεισφορά της Πτυχιακής Εργασίας	2
1.4 Συμπεράσματα	3
1.5 Μελλοντική έρευνα	4
2 Μέθοδοι μηχανικής μάθησης για ταξινόμηση/συσταδοποίηση πληροφορίας	5
2.1 Ανάκτηση Πληροφορίας: Ταξινόμηση και Συσταδοποίηση	5
2.2 Το Μοντέλο Διανυσματικού Χώρου	6
2.3 Ο αλγόριθμος tf-idf	8
2.4 Μέθοδοι Μηχανικής Μάθησης για Ταξινόμηση Κειμενικής Πληροφορίας	9
2.4.1 Επιβλεπόμενη μάθηση (Supervised learning)	9
2.4.2 Μη επιβλεπόμενη μάθηση (Unsupervised learning)	10
2.4.3 Ενισχυτική μάθηση (Reinforcement learning)	11
2.4.4 Η μέθοδος Cross-Validation	11
2.5 Μέθοδοι Αξιολόγησης – Απόδοσης	12
2.5.1 Βασικές Έννοιες Απόδοσης	12
2.5.2 Ανάληση και Ακρίβεια (Recall and Precision)	13
2.6 Αλγόριθμοι που χρησιμοποιήθηκαν	17

2.6.1	Support vector machine	17
2.6.2	Random forest	19
2.6.3	K-Nearest Neighbors	20
2.6.4	Logistic Regression	22
2.6.5	Naive Bayes	23
2.6.6	Latent Dirichlet Allocation	24
3	Πηγές και συλλογές δεδομένων που χρησιμοποιήθηκαν (Datasets)	27
3.1	Six Categories of Amazon Product Reviews	27
3.1.1	Περιγραφή δομής της Πηγής	28
3.1.2	Χρησιμότητα και εφαρμογές	30
3.2	Amazon movie reviews dataset	30
3.2.1	Περιγραφή δομής της Πηγής	31
3.2.2	Χρησιμότητα και εφαρμογές	32
3.3	Επεξεργασία των Dataset	32
3.4	Συνεισφορά της Πτυχιακής	33
3.4.1	Συλλογή των labels και εμπλουτισμός του dataset	33
3.4.2	Ιεραρχική δομή των labels	35
3.4.3	Οδηγίες λήψης των labels και διαδικασίας εμπλουτισμού του αρχικού Dataset.	37
4	Αποτελέσματα – Μετρήσεις	39
4.1	Πλατφόρμα – Βιβλιοθήκες	39
4.1.1	Γλώσσα προγραμματισμού Python	39
4.1.2	Βιβλιοθήκη Μηχανικής μάθησης Scikit-learn	41
4.1.3	IPython Notebook Environment	42
4.2	Προβλήματα Ταξινόμησης – Μεθοδολογία και αποτελέσματα	42
4.2.1	Ταξινόμηση στο Six Categories of Amazon Product Reviews Dataset	42
4.2.2	Ταξινόμηση στο Amazon movie reviews Dataset	58
4.3	Προβλήματα Συσταδοποίησης – Μεθοδολογία και αποτελέσματα	74
4.3.1	Η μέθοδος Latent Dirichlet Allocation (Topic model)	74
4.3.2	Συσταδοποίηση μέσω Πινάκων – Μασκών Συνάφειας	86
5	Αξιολόγηση – Συμπεράσματα	95
5.1	Αξιολόγηση προβλημάτων ταξινόμησης	95
5.1.1	Αξιολόγηση στο “Six Categories of Amazon Product Reviews Dataset”	95
5.1.2	Αξιολόγηση στο “Amazon Movies Reviews Dataset”	98
5.1.3	Συμπεράσματα Ταξινόμησης	103
5.2	Αξιολόγηση προβλημάτων συσταδοποίησης	108
5.2.1	Το μοντέλο Latent Dirichlet allocation (LDA)	108
5.2.2	Η μέθοδος Πινάκων/Μασκών Συνάφειας	108
5.3	Γενικά Συμπεράσματα	114

A' Κώδικας σε γλώσσα προγραμματισμού Python	115
A'.1 Εμπλουτισμός του dataset με τα συλλεχθέντα labels	115
A'.2 Εξαγωγή των labels σε format ιεραρχικής δομής (tree structure) σε JSON αρχείο	116
A'.3 Δημιουργία μασκών/πινάκων Συνάφειας	118
A'.4 Μέθοδος συλλογής δεδομένων (scraping) από τις σελίδες των προϊόντων της Amazon.com	120
Βιβλιογραφία	129

Κατάλογος Σχημάτων

1.1	Το λογότυπο του ΕΑΠ σε Word Cloud που παράχθηκε από τα συλλεχθέντα labels του Amazon movie reviews dataset	3
2.1	Συσχετισμένα έγγραφα στον διανυσματικό χώρο	7
2.2	Έγγραφα στον διανυσματικό χώρο σε απόσταση μεταξύ τους	7
2.3	Μοντέλο Δένδρου αποφάσεων (decision tree)	10
2.4	Γραφική παράσταση συσταδοποίησης τριών συστάδων	11
2.5	Η συνηθέστερη μορφή της μεθόδου k-fold cross-validation, όπου $k = 10$	12
2.6	Αναπαράσταση των συνόλων των εγγράφων	13
2.7	Περίπτωση χαμηλής ανάχλησης και χαμηλής ακρίβειας	14
2.8	Περίπτωση χαμηλής ανάχλησης και υψηλής ακρίβειας	14
2.9	Περίπτωση υψηλής ανάχλησης και χαμηλής ακρίβειας	15
2.10	Περίπτωση υψηλής ανάχλησης και υψηλής ακρίβειας	15
2.11	Γραφική παράσταση ανάχλησης/ακρίβειας	16
2.12	Καμπύλη ανάχλησης - ακρίβειας με την τεχνική της παρεμβολής	17
2.13	Πιθανά υπερεπίπεδα που διαχωρίζουν σωστά τα αντικείμενα	18
2.14	Επιλογή του βέλτιστου υπερεπιπέδου	18
2.15	Βήματα κατασκευής ενός Random forest	19
2.16	Βήματα ενός K-NN αλγορίθμου	21
2.17	Περιπτώσεις K-nn με $k = 1, k = 2$ και $k = 3$	21
2.18	Περίπτωση K-nn με λανθασμένη ταξινόμηση	22
2.19	Σιγμοειδής συνάρτηση	23
2.20	Αναπαράσταση επιπέδων στην LDA	25
3.1	Σχήμα διάρθρωσης φωκέλων του Dataset	28
3.2	Παράδειγμα προϊόντος με επισημασμένα τα labels	33
3.3	Παράδειγμα προϊόντος στο οποίο δεν έχουν καταχωρηθεί labels.	34
3.4	Ιεραρχική δομή των labels.	35
4.1	Διάγραμμα Γλωσσών προγραμματισμού στην προσφορά εργασίας	39
4.2	Χάρτης αλγορίθμων – μοντέλων της Scikit-learn	41
4.3	Απόσπασμα από το IPython Notebook	42
4.4	Αποτέλεσμα της εντολής df.head()	43

4.5 Αποτέλεσμα της εντολής df.head(5)	44
4.6 Προσθήκη στήλης με το μήκος του κάθε review	45
4.7 Ιστόγραμμα βασισμένο στο μήκος των reviews	45
4.8 Ιστόγραμμα βασισμένο στο μήκος των reviews με λογαριθμημένο τον άξονα y	46
4.9 Ιστόγραμμα μήκους review ανά κατηγορία προϊόντος	46
4.10 Πλήθος των reviews ανά κατηγορία προϊόντος	47
4.11 Πέντε πρώτα και πέντε τελευταία instances των reviews	48
4.12 Training set και testing set	49
4.13 Δημιουργία Document-term Matrix	49
4.14 Απόδοση ακρίβειας Naive Bayes	50
4.15 Πίνακας σύγχυσης Naive Bayes	51
4.16 Απόδοση ακρίβειας 10-Fold Cross Validation του Naive Bayes	51
4.17 Απόδοση ακρίβειας Logistic regression	52
4.18 Πίνακας σύγχυσης Logistic regression	52
4.19 Απόδοση ακρίβειας 10-Fold Cross Validation του Logistic regression	53
4.20 Απόδοση ακρίβειας Random Forest	53
4.21 Πίνακας σύγχυσης Random Forest	54
4.22 Απόδοση ακρίβειας 10-Fold Cross Validation του Random Forest	54
4.23 Απόδοση ακρίβειας του K-nn για τιμές του k από 1 έως 25	55
4.24 Απόδοση ακρίβειας του K-nn	56
4.25 Πίνακας σύγχυσης του K-nn	56
4.26 Απόδοση ακρίβειας 10-Fold Cross Validation του K-nn	57
4.27 Απόδοση ακρίβειας Support vector machine	57
4.28 Πίνακας σύγχυσης Support vector machine	58
4.29 Απόδοση ακρίβειας 10-Fold Cross Validation του Support vector machine	58
4.30 Δεδομένα στα οποία θα εφαρμοσθούν τα μοντέλα ταξινόμησης του Amazon movie reviews Dataset	59
4.31 Κατανομή των reviews ανά κατηγορία	60
4.32 Διαμόρφωση ομαλότερης κατανομής των reviews ανά κατηγορία	61
4.33 Νέα (ομαλή) κατανομή των reviews ανά κατηγορία	61
4.34 Κατανομή των reviews ανά κατηγορία με λογαριθμημένο τον άξονα y	62
4.35 Ιστόγραμμα ανά κατηγορία προϊόντων	63
4.36 Ορισμός id's στις κατηγορίες	64
4.37 Πέντε πρώτα και πέντε τελευταία instances των reviews	64
4.38 Διαμοίραση σε training set και testing set	65
4.39 Στοιχεία διανυσματικού πίνακα	65
4.40 Απόδοση ακρίβειας Naive Bayes	66
4.41 Πίνακας σύγχυσης Naive Bayes	67
4.42 Απόδοση ακρίβειας 10-Fold Cross Validation του Naive Bayes	67
4.43 Απόδοση ακρίβειας Logistic regression	68
4.44 Πίνακας σύγχυσης Logistic regression	68

4.45 Απόδοση ακρίβειας 10-Fold Cross Validation του Logistic regression	69
4.46 Απόδοση ακρίβειας Random Forest	69
4.47 Πίνακας σύγχυσης Random Forest	70
4.48 Απόδοση ακρίβειας 10-Fold Cross Validation του Random Forest	70
4.49 Απόδοση ακρίβειας του K-nn για τιμές του k από 1 έως 25	71
4.50 Απόδοση ακρίβειας K-nn	71
4.51 Πίνακας σύγχυσης K-nn	72
4.52 Απόδοση ακρίβειας 10-Fold Cross Validation του K-nn	72
4.53 Απόδοση ακρίβειας Support vector machine	73
4.54 Πίνακας σύγχυσης Support vector machine	73
4.55 Απόδοση ακρίβειας 10-Fold Cross Validation του Support vector machine .	74
4.56 Προσπέλαση και φόρτωση δεδομένων από τα JSON αρχεία	77
4.57 Data preprocessing	78
4.58 Διαδικασία αντιστοίχησης εξαγόμενων topics με τις πραγματικές κατηγορίες .	79
4.59 Πλήθος topics από 3 έως 6	79
4.60 Πλήθος topics από 7 έως 10	80
4.61 Πλήθος topics από 11 έως 12	80
4.62 Intertopic Distance Map for 3 topics	81
4.63 Intertopic Distance Map for 4 topics	81
4.64 Intertopic Distance Map for 5 topics	82
4.65 Intertopic Distance Map for 6 topics	82
4.66 Intertopic Distance Map for 7 topics	83
4.67 Intertopic Distance Map for 8 topics	83
4.68 Intertopic Distance Map for 9 topics	84
4.69 Intertopic Distance Map for 10 topics	84
4.70 Intertopic Distance Map for 11 topics	85
4.71 Intertopic Distance Map for 12 topics	85
4.72 Τυχαίος διαχωρισμός σε training και testing sets	87
4.73 Αντιστοίχιση κατηγοριών με αριθμούς (id's)	88
4.74 Μάσκα συνάφειας για τον Logistic Regression	89
4.75 Ποσοστά απόδοσης ανά κατηγορία για το Logistic Regression	89
4.76 Μάσκα συνάφειας για τον Random Forest	90
4.77 Ποσοστά απόδοσης ανά κατηγορία για το Random Forest	90
4.78 Μάσκα συνάφειας για τον Naive Bayes	91
4.79 Ποσοστά απόδοσης ανά κατηγορία για το Naive Bayes	91
4.80 Μάσκα συνάφειας για τον K-nearest neighbors	92
4.81 Ποσοστά απόδοσης ανά κατηγορία για το K-nearest neighbors	92
4.82 Μάσκα συνάφειας για τον Support vector machine	93
4.83 Ποσοστά απόδοσης ανά κατηγορία για το Support vector machine	93

5.1	Αποδόσεις μοντέλων στο “Six Categories of Amazon Product Reviews Dataset” - Training / Testing set	104
5.2	Αποδόσεις μοντέλων στο “Six Categories of Amazon Product Reviews Dataset” - 10-Fold Cross Validation	104
5.3	Συγκεντρωτικό γράφημα αποδόσεων στο “Six Categories of Amazon Product Reviews Dataset” – TT vs 10-Fold Cross Validation	105
5.4	Συγκεντρωτικές αποδόσεις ανά μοντέλο στο “Six Categories of Amazon Prod- uct Reviews Dataset”	105
5.5	Αποδόσεις μοντέλων στο “Amazon Movies Reviews Dataset” – Training/Testing set	106
5.6	Αποδόσεις μοντέλων στο “Amazon Movies Reviews Dataset” – 10-Fold Cross Validation	106
5.7	Συγκεντρωτικό γράφημα αποδόσεων στο “Amazon Movies Reviews Dataset” – TT vs 10-Fold Cross Validation	107
5.8	Συγκεντρωτικές αποδόσεις ανά μοντέλο στο “Amazon Movies Reviews Dataset”	107
5.9	Αποδόσεις της LDA ανά πλήθος εξαγόμενων topics	108
5.10	Πίνακας/μάσκα συνάφειας του Naive Bayes	109
5.11	Πίνακας/μάσκα συνάφειας του Random Forest	110
5.12	Πίνακας/μάσκα συνάφειας του Logistic Regression	111
5.13	Πίνακας/μάσκα συνάφειας του K-nearest neighbors	112
5.14	Πίνακας/μάσκα συνάφειας του Support vector machine	113

Κατάλογος Πινάκων

2.1	Πιθανές καταστάσεις ανάκτησης της πληροφορίας	13
2.2	Τιμές ανάκλησης και ακρίβειας παραδείγματος	16
3.1	Στατιστικά στοιχεία δεδομένων του Six Categories of Amazon Product Reviews dataset	28
3.2	Στατιστικά στοιχεία δεδομένων του Amazon movie reviews dataset	31
5.1	Ανάλυση απόδοσης του μοντέλου Naive Bayes	96
5.2	Ανάλυση απόδοσης του μοντέλου Logistic Regression	96
5.3	Ανάλυση απόδοσης του μοντέλου Random Forest	97
5.4	Ανάλυση απόδοσης του μοντέλου K-nearest neighbors	97
5.5	Ανάλυση απόδοσης του μοντέλου Support vector machine	98
5.6	Ανάλυση απόδοσης του μοντέλου Naive Bayes	99
5.7	Ανάλυση απόδοσης του μοντέλου Logistic Regression	100
5.8	Ανάλυση απόδοσης του μοντέλου Random Forest	101
5.9	Ανάλυση απόδοσης του μοντέλου K-nearest neighbors	102
5.10	Ανάλυση απόδοσης του μοντέλου Support vector machine	103

Κεφάλαιο 1

Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Το γενικό θεωρητικό αντικείμενο που αναπτύσσεται σε αυτήν την εργασία είναι η μηχανική μάθηση και κάποιες εφαρμογές της (classification/clustering). Η μηχανική μάθηση θα μπορούσε να περιγραφεί ως η διαδικασία εκείνη κατά την οποία ένα σύστημα βελτιώνεται συνεχώς ως προς την απόδοση του, κατά την διάρκεια εκτέλεσης μιας συγκεκριμένης εργασίας, χωρίς να απαιτείται ο εκ νέου προγραμματισμός του. Σύμφωνα με αυτήν την προσέγγιση, σκοπός της είναι η δημιουργία συστημάτων που διαθέτουν ικανότητες μάθησης και βελτίωσης της απόδοσής τους σε ορισμένους τομείς μέσω της αξιοποίησης προηγούμενης γνώσης και εμπειρίας.

Ο Mitchell έχει δώσει τον παρακάτω γενικός ορισμός Μηχανικής Μάθησης:

“Ενα πρόγραμμα υπολογιστή λέμε ότι μαθαίνει από την εμπειρία E ως προς κάποια χλάση εργασιών T και μέτρο απόδοσης P , αν η απόδοσή του σε εργασίες από το T , όπως μετριέται από το P , βελτιώνεται μέσω της εμπειρίας E .”. [8]

1.2 Οργάνωση του τόμου

Η παρούσα πτυχιακή εργασία που φέρει τον τίτλο “Τεχνικές Ταξινόμησης και Συσταδοποίησης Μεγάλων Συλλογών Διαδικτυακών Δεδομένων” περιλαμβάνει τα παρακάτω κεφάλαια:

- Εισαγωγή
- Μέθοδοι μηχανικής μάθησης για ταξινόμηση/συσταδοποίηση πληροφορίας
- Πηγές και συλλογές δεδομένων που χρησιμοποιήθηκαν (Datasets)
- Αποτελέσματα - Μετρήσεις
- Αξιολόγηση - Συμπεράσματα

Ακολουθεί μία σύντομη επισκόπηση των κεφαλαίων ξεκινώντας από το πρώτο κεφάλαιο το οποίο περιλαμβάνει εισαγωγικές έννοιες πάνω στα αντικείμενα που καλύπτονται στην πτυχιακή εργασία.

Στο δεύτερο κεφάλαιο γίνεται αναφορά στα σημαντικότερα σημεία της θεωρίας όπως η ανάκτηση της πληροφορίας, ο διανυσματικός χώρος, μέθοδοι μηχανικής μάθησης, έννοιες απόδοσης (ανάληση και ακρίβεια) και οι αλγόριθμοι που εφαρμόσθηκαν στα πλαίσια της εργασίας.

Στο τρίτο κεφάλαιο παρουσιάζονται οι πηγές δεδομένων που χρησιμοποιήθηκαν, η διαδικασία της απαίτουμενης προεπεξεργασίας των δεδομένων καθώς και οι διαδικασίες εμπλουτισμού του dataset Amazon Movies Reviews με τα ground truth labels.

Στο τέταρτο κεφάλαιο αντιμετωπίζονται δύο διαφορετικά προβλήματα, αυτό της ταξινόμησης και της συσταδοποίησης. Και στις δύο αυτές περιπτώσεις χρησιμοποιούνται οι πηγές δεδομένων που αναλύθηκαν εκτενώς στο τρίτο κεφάλαιο. Στο πρόβλημα της ταξινόμησης εφαρμόζονται πέντε διαφορετικά μοντέλα (Naive Bayes, Random Forest, Logistic Regression, K-nearest neighbors και Support vector machine) ενώ για την υλοποίηση της συσταδοποίησης χρησιμοποιείται η LDA και η μέθοδος μέσω των πινάκων—μασκών συνάρφειας. Σε όλες τις περιπτώσεις παρατίθενται τα αποτελέσματα των αλγορίθμων καθώς και κάποιες γραφικές παραστάσεις όπου ήταν εφικτό.

Στο πέμπτο κεφάλαιο παρουσιάζονται αξιολογήσεις των αποτελεσμάτων των μοντέλων που υλοποιήθηκαν και για τα δύο προβλήματα. Ο σχολιασμός γίνεται ανά πηγή δεδομένων και ανά μοντέλο. Λαμβάνονται υπόψη οι τιμές των μετρικών της ακρίβειας, ανάλησης καθώς και η τιμή της f1-score. Τέλος αναφέρονται κάποια γενικά συμπεράσματα.

1.3 Συνεισφορά της Πτυχιακής Εργασίας

Μία βασική συνεισφορά της παρούσας πτυχιακής, έγκειται στον εμπλουτισμό του “Amazon movie reviews Dataset” το οποίο διανέμεται από το Stanford University, με ιεραρχικές ετικέτες (labels) όπως αποτυπώνονται στις ιστοσελίδες της Amazon. Η προσθήκη ετικετών επαλήθευσης (Ground truth) βελτιώνει κατά πολύ την αρχική έκδοση τη συλλογής διότι προσφέρει μία καλύτερη ποιότητα εκπαίδευσης στα μοντέλα ταξινομητών/συσταδοποιητών στα οποία θα χρησιμοποιηθεί. Η νέα αυτή εμπλουτισμένη πηγή δεδομένων φιλοξενείται ήδη στην συλλογή των Datasets της Kaggle.com¹ και επίσης είναι διαθέσιμο προς αξιοποίηση στην κοινότητα, μέσω git repository στο GitHub.com [5]. Επίσης βρισκόμαστε σε διαδικασία ανάρτησης του στην επίσημη ιστοσελίδα του Stanford University και πιο συγκεκριμένα στην σελίδα της ερευνητικής ομάδας SNAP (Stanford Network Analysis Project)².

¹<https://www.kaggle.com/thebuzz/ground-truth-labels-amazon-movie-reviews-dataset>

²<https://snap.stanford.edu/data/web-Movies.html>



Σχήμα 1.1: Το λογότυπο του EAII σε Word Cloud που παράχθηκε από τα συλλεχθέντα labels του Amazon movie reviews dataset

1.4 Συμπεράσματα

Όπως αναφέρεται και στο πέμπτο κεφάλαιο, παρόλο που οι αποδόσεις των μοντέλων ήταν αρκετά υψηλές, σημαντικό ρόλο παίζουν τα δεδομένα με τα οποία υλοποιείται η φάση της εκπαίδευσης. Επίσης επαληθεύεται ότι το κάθε μοντέλο δεν είναι κατάλληλο για κάθε τύπο δεδομένων. Κάποια είναι πιο αποδοτικά με αριθμητικά δεδομένα και κάποια με κειμενικά. Αυτό φαίνεται και από την διαχύμανση των αποδόσεων στις υλοποιήσεις των μοντέλων. Υπήρχε μια σχετική διαχύμανση στις προβλέψεις τους.

Επίσης παρατηρήσαμε ότι δεν υπάρχει μοναδική λύση στα προβλήματα μηχανικής μάθησης που αναλύονται στο τέταρτο κεφάλαιο. Υπάρχει αρκετά μεγάλη ποικιλία διαφορετικών προ-

σε γγίσεων ως προς την αντιμετώπιση του προβλήματος, με κάποιες να υπερτερούν σε απόδοση και άλλες όχι.

1.5 Μελλοντική έρευνα

Παράλληλα με την ραγδαία εξέλιξη της τεχνολογίας, αυξάνονται και οι ανάγκες σε ευφυή συστήματα που να προσομοιώνουν την ανθρώπινη διάσθηση/αντίληψη. Αυτός είναι ένας λόγος για τον οποίο υπάρχει μια συνεχής έρευνα στον τομέα της τεχνητής νοημοσύνης και σε όλα τα ερευνητικά πεδία που ανήκουν στον τομέα αυτό. Ενδεικτικά ακολουθούν μερικά αντικείμενα στα οποία η τεχνητή νοημοσύνη θα συμβάλει καίρια στην ανάπτυξη τους αλλά και στην αποτελεσματικότητα τους.

- Self driving vehicles (Αυτοκινούμενα οχήματα)
- Real time translation (Μετάφραση σε πραγματικό χρόνο)
- Classifying DNA sequences (Ταξινόμηση ακολουθιών του DNA)
- Bioinformatics (Βιοπληροφορική)
- Sentiment Analysis/Opinion Mining
- Structural health monitoring (Παρακολούθηση υγείας)
- Object recognition in Computer vision (Αναγνώριση αντικειμένων από υπολογιστή)
- Brain-machine interfaces
- Medical diagnosis
- Computational Advertising & finance (Υπολογιστική διαφήμιση & χρηματοοικονομικά)
- Detecting credit card fraud (Εντοπισμός απάτης με χρήση πιστωτικών καρτών)

Φαίνεται εξαιρετικά ενδιαφέρον το μέλλον και οι προοπτικές στον τομέα της τεχνητής νοημοσύνης και της μηχανικής μάθησης.

Κεφάλαιο 2

Μέθοδοι μηχανικής μάθησης για ταξινόμηση/συσταδοποίηση πληροφορίας

Στο δεύτερο αυτό κεφάλαιο θα αναφερθούν όμως στα ζητήματα πάνω στον τομέα της μηχανικής μάθησης καθώς και ποια μοντέλα - αλγόριθμοι χρησιμοποιήθηκαν για την υλοποίηση των πειραμάτων μηχανικής μάθησης της παρούσας εργασίας.

2.1 Ανάκτηση Πληροφορίας: Ταξινόμηση και Συσταδοποίηση

Τα τελευταία χρόνια το διαδίκτυο μπήκε σχεδόν σε όλα τα σπίτια και έγινε καθημερινό εργαλείο ενημέρωσης αλλά και ψυχαγωγίας. Η ταχεία ανάπτυξη του επηρέασε όχι μόνο την δομή των σύγχρονων ιστοσελίδων αλλά και την αυξανόμενη διαδραστικότητα τους. Παράλληλα εξελίσσονται οι αλγόριθμοι των μηχανών αναζήτησης αλλά και αναπτύσσονται νέοι ακόμη πιο αποδοτικοί και «έξυπνοι» ώστε να προσελκύουν περισσότερους χρήστες με ολοένα και σωστότερα αποτελέσματα στις αναζητήσεις τους.

Ο τομέας της ανάκτησης και εξόρυξης πληροφορίας βρίσκει άμεση εφαρμογή στις μηχανές αναζήτησης. Η ραγδαία και απότομη αύξηση των ιστότοπων τα τελευταία χρόνια, δίνει το έναυσμα στους μηχανικούς πληροφορικής που ασχολούνται με την έρευνα στο συγκεκριμένο πεδίο, να υλοποιήσουν αλγορίθμους που επηρεάζουν σημαντικά τον τρόπο με τον οποίο λειτουργούν οι μηχανές αναζήτησης.

Σίγουρα δεν αρκεί μόνο η ανάκτηση της πληροφορίας αλλά έχει μεγάλη σημασία και η μορφοποίηση αυτής. Αυτή πρέπει να είναι αναγνώσιμη όχι τόσο μόνο από το ανθρώπινο μάτι, αλλά κυρίως από τα προγράμματα και τις εφαρμογές στις οποίες πρόκειται να την επεξεργασθούν και στην συνέχεια να την παρουσιάσουν στους χρήστες οι οποίοι την αναζήτησαν.

Οι δύο πιο διαδεδομένες μέθοδοι – διαδικασίες στην εξόρυξη δεδομένων είναι η κατηγοριοποίηση (classification) και η συσταδοποίηση (clustering) και με αυτές τις δύο θα ασχοληθούμε

στα επόμενα κεφάλαια.

2.2 Το Μοντέλο Διανυσματικού Χώρου

Έχει αποδειχθεί ότι στα συστήματα ανάκτησης πληροφορίας, επιτυγχάνεται καλύτερη συσταδοποίηση όταν η κατανομή των δεδομένων στον χώρο γίνεται όσο το δυνατόν λιγότερο κοντά το ένα από το άλλο. Από το παραπάνω προκύπτει πως η αξία ενός τέτοιου συστήματος θα μπορούσε να εκφραστεί ως μία συνάρτηση πυκνότητας του χώρου των αντικειμένων μιας και αυτή η πυκνότητα επηρεάζει την απόδοση στην ανάκτηση της πληροφορίας.

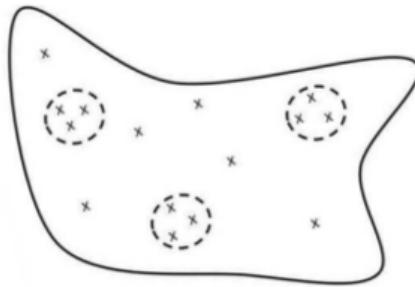
Θεωρούμε E_i έγγραφα σε έναν χώρο εγγράφων όπου το κάθε έγγραφο i , χαρακτηρίζεται από έναν ή περισσότερους όρους T_i . Κάθε όρος i χαρακτηρίζεται επίσης από μία τιμή (έστω από 0 έως 1) η οποία δηλώνει το βάρος του συγκεκριμένου όρου στο έγγραφο που ανήκει. Το πλήθος των όρων υποδηλώνει και τον αριθμό των διαστάσεων του χώρου των εγγράφων. Συνεπώς το κάθε έγγραφο μπορεί να αναπαρασταθεί από ένα διάνυσμα n διαστάσεων όπως το παρακάτω:

$$E_i = (w_{i1}, w_{i2}, \dots, w_{in}) \quad (2.1)$$

όπου το w_{ij} είναι το βάρος του j -οστού όρου στο έγγραφο i .

Πλέον μπορούμε, έχοντας δύο έγγραφα με τα αντίστοιχα διανύσματα τους, να υπολογίσουμε έναν συντελεστή ομοιότητας μεταξύ τους, έστω $\rho(E_i, E_j)$ ο οποίος υποδηλώνει την ομοιότητα μεταξύ των όρων των δύο εγγράφων. Η σχέση αυτή θα μπορούσε να είναι το εσωτερικό γινόμενο των διανυσμάτων ή η αντίστροφη συνάρτηση της γωνίας μεταξύ των. Έτσι σε περίπτωση ίδιων εγγράφων θα είχαμε γωνία 0° ή η αντίστροφη συνάρτηση θα έπαιρνε την μέγιστη τιμή της. Επίσης δύο έγγραφα με παρόμοια διανύσματα αναπαρίστανται από κοντινά σημεία στον χώρο με την μεταξύ τους απόσταση να είναι ανάλογη της σχετικότητας τους. Η ιδιαίτερη διάταξη των εγγράφων στον χώρο θα ήταν μία διάταξη τέτοια που οι τιμές της ανάλησης και της ακρίβειας θα έπαιρναν τις βέλτιστες τιμές τους.

Όταν δεν έχουμε προηγούμενη γνώση για τα έγγραφα μας, ένας ιδανικός χώρος θα ήταν αυτός όπου τα έγγραφα που σχετίζονται με αναζήτησεις του χρήστη, θα ήταν τοποθετημένα πολύ κοντά μεταξύ τους. Με αυτή τη διάταξη, η κάθε αναζήτηση θα επέστρεψε όλα τα σχετικά έγγραφα μαζί ενώ τα υπόλοιπα, μη σχετιζόμενα με την αναζήτηση, έγγραφα θα βρίσκονται ιδιαίτερα σε πιο απομακρυσμένα σημεία του χώρου. Μία αναπαράσταση της παραπάνω περιγραφής φαίνεται στην εικόνα 2.1:



Σχήμα 2.1: Συσχετισμένα έγγραφα στον διανυσματικό χώρο

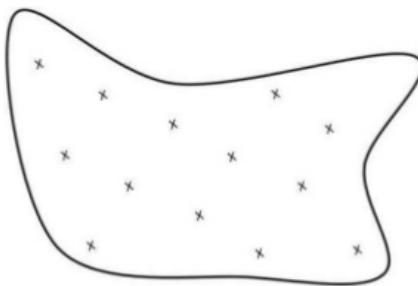
Βέβαια, μία τέτοια διάταξη θεωρείται ιδαινική μιας και οι συστάδες των εγγράφων που υπάρχουν στον χώρο σχετίζονται όμεσα με ανάλογες αναζητήσεις του χρήστη, κάτι το οποίο είναι αδύνατο να εφαρμοσθεί κατά την δημιουργία του χώρου. Αυτό συμβαίνει διότι κατά την διαδικασία της συσταδοποίησης δεν υπάρχει γνώση ή ιστορικό σχετικό με τις αναζητήσεις και τις ανακτήσεις εγγράφων.

Αυτό που θα μπορούσε να βελτιώσει την διάταξη του χώρου είναι η μεγιστοποίηση των αποστάσεων μεταξύ των μη σχετικών εγγράφων και πιο συγκεκριμένα, σε μία συλλογή n εγγράφων θα πρέπει να ελαχιστοποιηθεί η παρακάτω συνάρτηση:

$$F = \sum_{i=1}^n \sum_{j=1}^n o(E_i, E_j), i \neq j \quad (2.2)$$

όπου $o(E_i, E_j)$ είναι η ομοιότητα μεταξύ των εγγράφων i και j .

Όσο χαμηλότερη τιμή έχει η συνάρτηση F , τόσο μικρότερη είναι η ομοιότητα μεταξύ των δύο εγγράφων i και j . Έτσι έχουμε ως αποτέλεσμα την αύξηση της μέσης απόστασης των αντικειμένων στον χώρο και εξασφαλίζεται ότι σε αναζήτηση του χρήστη θα ανακτηθούν μόνο τα έγγραφα που βρίσκονται αρκετά κοντά σε αυτήν και όχι όλα τα γειτονικά έγγραφα. Με αυτόν τον τρόπο αυξάνεται η ακρίβεια αλλά και η ανάκληση του συστήματος. Μία αναπαράσταση της παραπάνω περιγραφής φαίνεται στην εικόνα 2.2:



Σχήμα 2.2: Έγγραφα στον διανυσματικό χώρο σε απόσταση μεταξύ τους

Πρακτικά ο υπολογισμός της συνάρτησης F είναι μεγάλης σχετικά πολυπλοκότητας μιας και για μία συλλογή n εγγράφων θα πρέπει να υπολογισθούν n^2 έγγραφα. Έτσι θα θεωρήσου-

με τις συστάδες των εγγράφων που θα χαρακτηρίζονται από το κέντρο της κάθε ομάδας, οπότε σε έναν χώρο οι ομάδες των εγγράφων θα αναπαρίστανται από το κέντρο της συστάδας σχετικών εγγράφων που ανήκουν. Το μέσο βάρος Σ των εγγράφων, έστω m στο πλήθος, που ανήκουν σε μία κατηγορία K μπορεί να οριστεί από την παρακάτω σχέση:

$$\Sigma_j = \frac{\sum_{i=1}^m E_{ij}}{m}, E_i \in K \quad (2.3)$$

Συνεπώς, όπως υπολογίσθηκε το κέντρο μίας συστάδας εγγράφων από τα έγγραφά της, έτσι μπορεί να υπολογισθεί και το κέντρο του χώρου από την μέση τιμή των βαρών στα κέντρα των συστάδων που περιέχει. Η πυκνότητα του χώρου σε έναν συσταδοποιημένο χώρο εγγράφων, υπολογίζεται από το σύνολο των συντελεστών ομοιότητας του εγγράφου και του κέντρου του χώρου. Εκφράζεται από την παρακάτω σχέση:

$$\sum_{i=1}^n o(\Sigma^*, E_i) \quad (2.4)$$

όπου Σ^* θεωρείται το κέντρο του χώρου.

Συνεπώς, για να επιτευχθεί η βέλτιστη απόδοση ενός συστήματος, ο χώρος του θα πρέπει να αποτελείται από συστάδες σχετικών εγγράφων, η απόσταση των οποίων εντός της συστάδας θα πρέπει να μεγιστοποιείται ενώ η απόσταση μεταξύ των ίδιων των συστάδων στον χώρο να ελαχιστοποιείται. Έτσι εξασφαλίζεται η βέλτιστη απόδοση του συστήματος με τις αντίστοιχες τιμές ακρίβειας και ανάλησης.

2.3 Ο αλγόριθμος tf-idf

Στα συστήματα ανάκτησης πληροφοριών και εξόρυξης κειμένου χρησιμοποιείται ο δημοφιλής αλγόριθμος $tf - idf$ (term frequency-inverse document frequency) [10]. Ο αλγόριθμος αυτός χρησιμοποιεί δύο τιμές, η μεν πρώτη tf υποδηλώνει τη συχνότητα του όρου t εντός ενός εγγράφου d , που ανήκει σε κάποια συλλογή και η δε δεύτερη idf τη συχνότητα του όρου t σε όλα τα έγγραφα της συλλογής. Η κύριος ρόλος της αντίστροφης συχνότητας εγγράφων είναι να αντισταθμίζεται η τελική τιμή ενός όρου σε περιπτώσεις λέξεων που εκ των πραγμάτων έχουν εξαιρετικά υψηλή συχνότητα, όπως για παράδειγμα ένα άρθρο ή ένας σύνδεσμος (και, ή, το κλπ).

Ο υπολογισμός της συχνότητας όρου, πέρα από την τετριμμένη καταμέτρηση της λέξης εντός του εγγράφου, κανονικοποιείται με διάφορους τρόπους προς αποφυγή κάποιας ακούσιας μεριοληψίας (μεγάλο ή μικρό κείμενο, συνηθισμένη ή ασυνήθιστη λέξη κλπ) διαιρώντας την συχνότητα με το μήκος του εγγράφου:

$$tf(t, d) = \frac{f_{t,d}}{\sum_k f_{k,d}} \quad (2.5)$$

όπου $f_{t,d}$ είναι η συχνότητα του όρου και ο παρονομαστής είναι το μήκος του εγγράφου (το συνολικό πλήθος των λέξεων του).

Η αντίστροφη συχνότητα εγγράφων είναι ένα μέτρο με το οποίο προσδιορίζεται η ποσότητα της πληροφορίας που πας παρέχει ο όρος t , δηλαδή το πόσο συχνά ή σπάνια εμφανίζεται ο όρος t σε όλα τα έγγραφα της συλλογής D . Στον αριθμητή έχουμε το πλήθος των εγγράφων και στον παρονομαστή το πλήθος των εγγράφων στα οποία περιέχετε ο όρος t [2]. Έπειτα παίρνουμε τον λογάριθμο του λόγου:

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \quad (2.6)$$

όπου στον παρονομαστή προστίθεται ο αριθμός 1 για να αποφύγουμε πιθανή διαίρεση με το μηδέν στην περίπτωση που ο όρος t δεν εμφανίζεται καθόλου σε όλη τη συλλογή.

Ο τελικός υπολογισμός του αλγορίθμου προκύπτει από την παρακάτω σχέση:

$$tf - idf(t, d, D) = tf(t, d) \cdot idf(t, D) \quad (2.7)$$

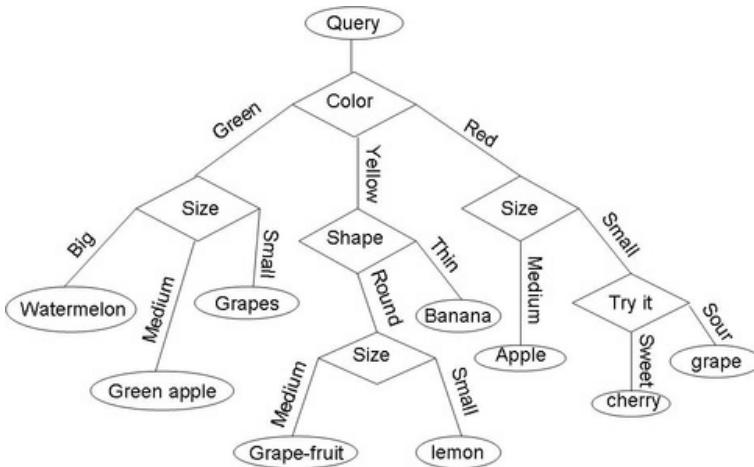
όπου η υψηλή τιμή του επιτυγχάνεται από υψηλή συχνότητα όρου (tf) και χαμηλή αντίστροφη συχνότητα εγγράφων (idf), έτσι οι κοινές λέξεις βαθμολογούνται χαμηλά.

2.4 Μέθοδοι Μηχανικής Μάθησης για Ταξινόμηση Κειμενικής Πληροφορίας

Ο τομέας της Επιστήμης Υπολογιστών που ασχολείται με την ευφυή συμπεριφορά υπολογιστικών μηχανών είναι η Μηχανική Μάθηση (Machine Learning). Βρίσκει εφαρμογή σε πολλά πεδία όπως για παράδειγμα σε μηχανές αναζήτησης, σε αναγνώριση προτύπων, εξόρυξη πληροφορίας συναισθημάτων από δεδομένα κειμένου, ιατρική διάγνωση κα. Τα συστήματα μηχανικής μάθησης έχουν έναν τρόπο να μαθαίνουν και να βελτιώνονται. Οι τρόποι με τους οποίους υλοποιούνται αυτά τα συστήματα είναι οι παρακάτω.

2.4.1 Επιβλεπόμενη μάθηση (Supervised learning)

Όταν εφαρμόζεται η μέθοδος της λεγόμενης επιβλεπόμενης μάθησης (supervised learning), παράγεται ένα σύστημα από την επόπτευση των εκπαιδευτικών δεδομένων (training set) το οποίο χρησιμοποιείται για την κατηγοριοποίηση των επόμενων άγνωστων δεδομένων εισόδου. Σε αυτήν την περίπτωση, το training set αποτελείται από ζεύγη δεδομένων όπου το ένα στοιχείο είναι η είσοδος στο σύστημα (input) ενώ το δεύτερο είναι η επιθυμητή έξοδος. Κατά την εκπαίδευση του συστήματος, εισάγεται ένα αρκετά μεγάλο σύνολο από τέτοια ζεύγη με σκοπό να το «ετοιμάσουν» ώστε να είναι σε θέση να προβλέπει επιτυχώς τις εξόδους σε άγνωστα σε αυτό δεδομένα.



Σχήμα 2.3: Μοντέλο Δένδρου αποφάσεων (decision tree)

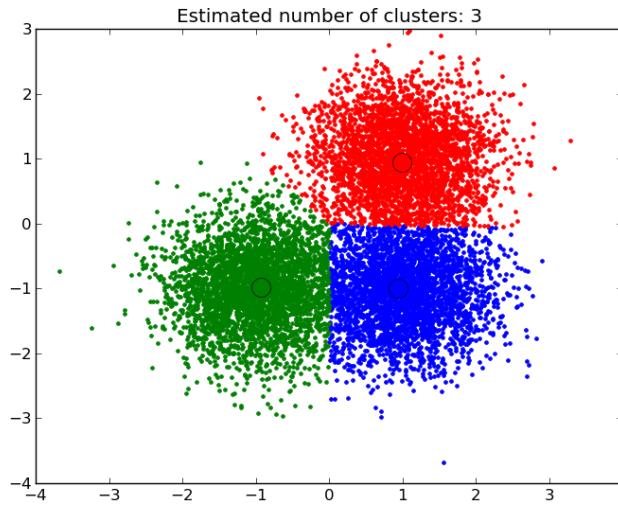
Τα βήματα που απαιτούνται για την αντιμετώπιση ενός προβλήματος με επιβλεπόμενη μάθηση είναι τα παρακάτω:

- Καθορισμός με ρητό τρόπο του τύπου δεδομένων του training set.
- Προετοιμασία αυτού καθώς και της επιθυμητής εξόδου σε κάθε ζεύγος δεδομένων.
- Επιλογή κατάλληλου αλγορίθμου που θα υλοποιήσει το επιθυμητό μοντέλο.
- Εκτέλεση της διαδικασίας μάθησης με τα δεδομένα εκμάθησης.
- Αξιολόγηση ακρίβειας του συστήματος, ώστε να δούμε πως συμπεριφέρεται και πόσο σωστές είναι οι προβλέψεις του σε άγνωστα προς αυτό δεδομένα.

Για την επιλογή του κατάλληλου αλγόριθμου στο 3ο βήμα, πρέπει να ληφθούν υπόψη τόσο το πλήθος των δεδομένων εκπαίδευσης όσο και ο τύπος δεδομένων αυτών (κειμενική πληροφορία, αριθμοί, διανύσματα κλπ).

2.4.2 Μη επιβλεπόμενη μάθηση (Unsupervised learning)

Στην μέθοδο αυτή (συσταδοποίηση - clustering) υλοποιείται η λεγόμενη μη-επιβλεπόμενη μάθηση (unsupervised learning) όπου και πάλι παράγεται ένα σύστημα από την είσοδο των εκπαίδευτικών δεδομένων (training set) το οποίο σε αυτήν την περίπτωση, αποτελείται μόνο από το στοιχείο εισόδου (input) χωρίς απολύτως καμία άλλη πληροφορία επιθυμητής εξόδου.



Σχήμα 2.4: Γραφική παράσταση συσταδοποίησης τριών συστάδων

Κατά την εκπαίδευση του συστήματος, εισάγεται ένα αρκετά μεγάλο σύνολο από δεδομένα εκπαίδευσης με σκοπό να το «ετοιμάσουν» ώστε να είναι σε θέση να αναγνωρίζει πρότυπα και να ομαδοποιεί τα άγνωστα σε αυτό δεδομένα σε συστάδες που έχουν κοινά γνωρίσματα και ιδιότητες.

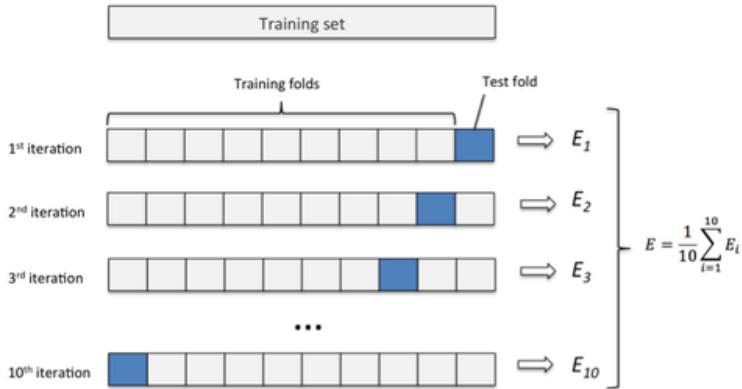
2.4.3 Ενισχυτική μάθηση (Reinforcement learning)

Βρίσκεται εφαρμογή κυρίως σε συστήματα που χρησιμοποιούν ευφυείς πράκτορες οι οποίοι εκπαιδεύονται δια της μεθόδου της επιβραβεύσεως. Όσο καλύτερη απόφαση λάβουν, τόσο μεγαλύτερη και η επιβράβευση αυτών. Αυτή η διαδικασία, μετά από την φάση της εκπαίδευσης, έχει ως αποτέλεσμα ένα εξαιρετικά ευφυή και αποδοτικό σύστημα που λαμβάνει σωστές και έξυπνες αποφάσεις με υψηλό ποσοστό επιτυχίας. Η διαφορά της από την επιβλεπόμενη μάθηση είναι ότι δεν χρησιμοποιεί γνώση που να είναι γνωστή από πριν αλλά βελτιώνεται σταδιακά κατά την διάρκεια της εκπαίδευσης.

2.4.4 Η μέθοδος Cross-Validation

Η μέθοδος Cross-Validation χρησιμοποιείται κατά κύριο λόγο στην αξιολόγηση ή ακόμη και στην σύγχριση αλγορίθμων. Ουσιαστικά διαιρεί τα διαθέσιμα δεδομένα σε δύο υποσύνολα, το μεν πρώτο το χρησιμοποιεί για την εκπαίδευση του μοντέλου και το δε δεύτερο για την αξιολόγηση - επιβεβαίωση ότι η εκπαίδευση υλοποιήθηκε επιτυχώς.

Στην κλασική εκδοχή της μεθόδου, που ονομάζεται k -fold cross-validation, το k αντιπροσωπεύει στο πλήθος των όμοιων σε μέγεθος μερών στα οποία θα μοιραστούν τα αρχικά δεδομένα. Έπειτα λαμβάνει χώρα ένας βρόγχος k επαναλήψεων όπου σε κάθε μία από αυτές, το υποσύνολο που δεσμεύτηκε για την αξιολόγηση είναι διαφορετικό ενώ τα υπόλοιπα $k - 1$ μέρη, χρησιμοποιούνται για την εκπαίδευση του μοντέλου.



Σχήμα 2.5: Η συνηθέστερη μορφή της μεθόδου k-fold cross-validation, όπου $k = 10$

Μετά το τέλος της διαδικασίας, υπολογίζεται ο μέσος όρος των ποσοστών επιτυχίας που προέκυψε σε κάθε μία από τις επαναλήψεις, όπως φαίνεται στον τύπο της εικόνας 2.5

2.5 Μέθοδοι Αξιολόγησης – Απόδοσης

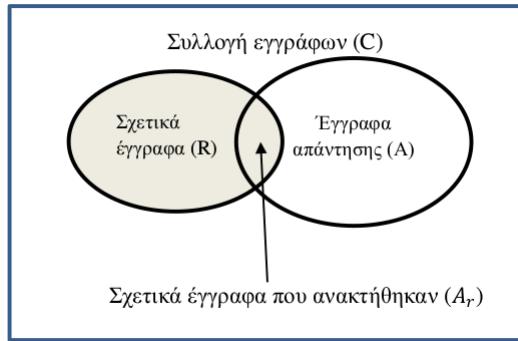
Τα συστήματα ανάκτησης πληροφοριών ανταποκρίνονται στα ερωτήματα των χρηστών επιστρέφοντας σε αυτούς αποτελέσματα σχετικά με τα ερωτήματα που δέχονται. Η επάρκεια αυτών των αποτελεσμάτων ωστόσο, πάντα όμως είναι ένα θέμα το οποίο πρέπει να αναλύεται και να αξιολογείται διότι συμβάλλουν στην αποτελεσματικότητα του συστήματος.

Παρακάτω παρουσιάζονται οι βασικές μετρικές εκτίμησης - απόδοσης ενός συστήματος διαχείρισης και ανάκτησης πληροφορίας οι οποίες είναι η ανάκληση (recall) και η ακρίβεια (precision). Οι μετρικές αυτές μας βοηθούν να καταλάβουμε πότε ένα σύστημα έχει καλύτερη απόδοση από ένα άλλο [13].

Πολλοί παράγοντες επηρεάζουν την αποτελεσματικότητα ενός συστήματος, όπως για παράδειγμα τα δεδομένα που διαχειρίζεται το σύστημα, το είδος των ερωτημάτων που υποβάλλονται από τους χρήστες και το μοντέλο ανάκτησης που χρησιμοποιείται και η ποιότητα εκπαίδευσής του.

2.5.1 Βασικές Έννοιες Απόδοσης

Έστω ότι σε ένα σύστημα με μία συλλογή εγγράφων C και από ένα ερώτημα q προς την C , έχουμε ως απόκριση τα δεδομένα που αποτελούν το σύνολο της A . Θεωρητικά έχουμε γνώση για το εάν το έγγραφο d είναι σχετικό ή όχι με το ερώτημα q . Έχοντας επίσης ένα σύνολο R που αποτελείται από όλα τα έγγραφα της συλλογής C που σχετίζονται με το ερώτημα q , συμβολίζουμε με A_r το υποσύνολο του A που περιέχει τα σχετικά ως προς το q έγγραφα.



Σχήμα 2.6: Αναπαράσταση των συνόλων των εγγράφων

Στο πίνακα 2.1 φαίνονται οι πιθανές καταστάσεις ενός εγγράφου ως προς ένα ερώτημα προς το σύστημα. Οι ιδιαικές περιπτώσεις είναι τα έγγραφα να ανήκουν στα υποσύνολα A_r ή $C - (R \cup A)$, δηλαδή στο κάτω αριστερό τεταρτημόριο ή στο πάνω δεξιό τεταρτημόριο του πίνακα.

	Ανακτηθέν	Μη ανακτηθέν
Μη σχετικό	Μη σχετικά έγγραφα που έχουν ανακτηθεί $A - A_r$	Μη σχετικά έγγραφα που δεν έχουν ανακτηθεί $C - (R \cup A)$
Σχετικό	Σχετικά έγγραφα που έχουν ανακτηθεί $A - A_r$	Σχετικά έγγραφα που δεν έχουν ανακτηθεί $R - A$

Πίνακας 2.1: Πιθανές καταστάσεις ανάκτησης της πληροφορίας

2.5.2 Ανάκληση και Ακρίβεια (Recall and Precision)

Ανάκληση (Recall) είναι ο λόγος του πλήθους των σχετικών εγγράφων που ανακτήθηκαν προς το συνολικό πλήθος σχετικών εγγράφων της συλλογής:

$$\text{recall}(q) = \frac{\# \text{ of fetched related documents}}{\# \text{ of related documents in collection}} \quad (2.8)$$

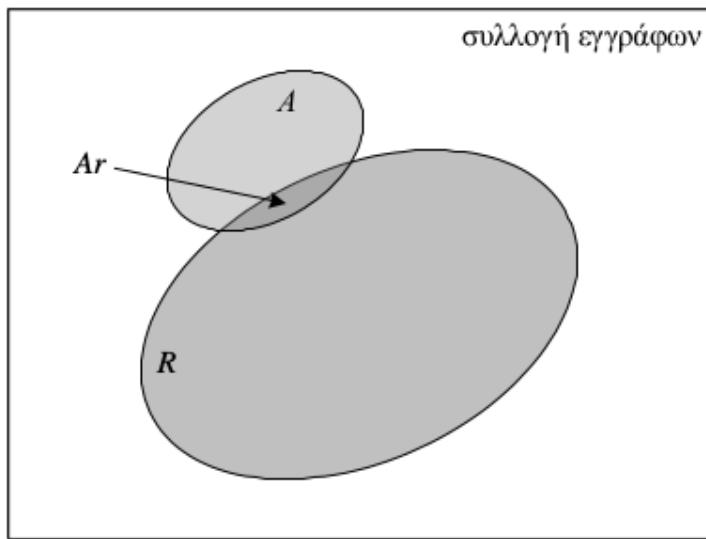
Ουσιαστικά από την ανάκληση έχουμε το ποσοστό των σχετικών εγγράφων που ανακτήθηκαν βάσει του ερωτήματος, σε σχέση με όλα τα σχετικά έγγραφα που περιέχει η συλλογή.

Ακρίβεια (Precision) είναι ο λόγος του πλήθους των σχετικών εγγράφων που ανακτήθηκαν προς τον συνολικό πλήθος των εγγράφων που ανακτήθηκαν:

$$\text{precision}(q) = \frac{\# \text{ of fetched related documents}}{\# \text{ of fetched documents}} \quad (2.9)$$

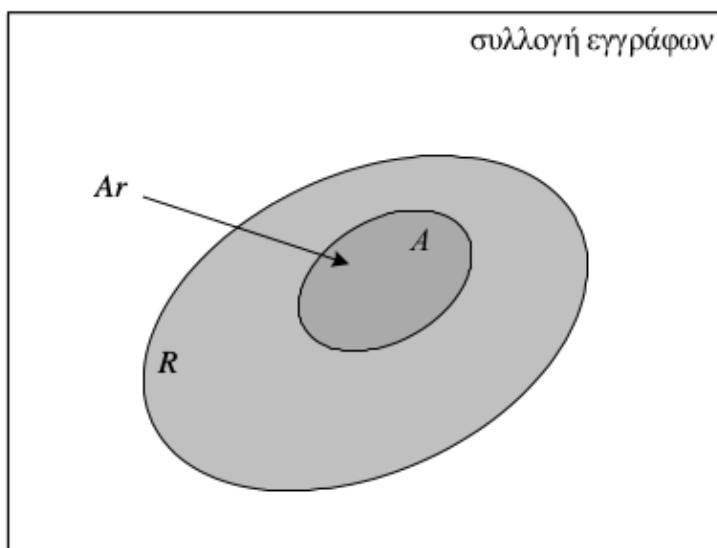
Παρακάτω παρουσιάζονται οι 4 περιπτώσεις συσχέτισης ανάκλησης – ακρίβειας σε ένα υποτιθέμενο σύστημα ανάκτησης πληροφορίας.

Στην εικόνα 2.7 έχουμε χαμηλή ανάχληση και χαμηλή ακρίβεια. Αυτό σημαίνει ότι πιθανότατα πολύ λίγα σχετικά έγγραφα θα ανακτηθούν από το σύστημα. Μία τέτοια περίπτωση υποδηλώνει χαμηλή ποιότητα στην αξία των πληροφοριών που ανακτούνται.



Σχήμα 2.7: Περίπτωση χαμηλής ανάχλησης και χαμηλής ακρίβειας

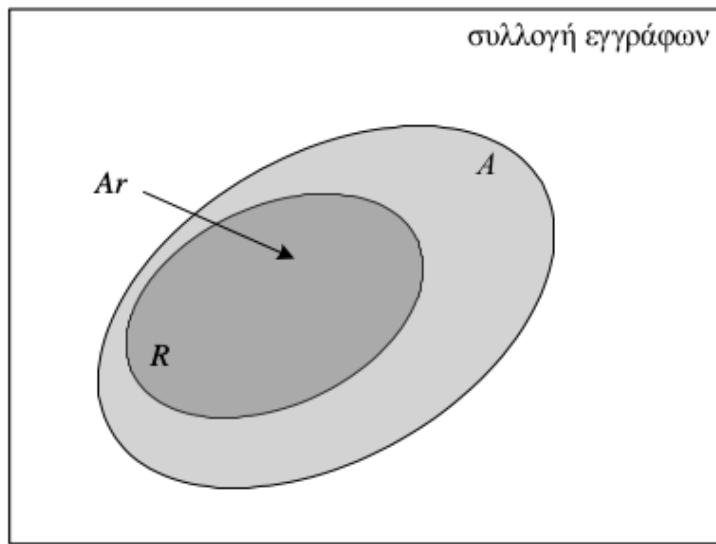
Στην εικόνα 2.8 έχουμε την περίπτωση όπου η ακρίβεια είναι στο 100% σε αντίθεση με την ανάχληση που έχει πολύ χαμηλή τιμή. Αυτό σημαίνει ότι ανακτήθηκαν καλής ποιότητας αποτελέσματα από το σύστημα αλλά λίγα σε σχέση με το πλήθος των σχετικών εγγράφων που περιλαμβάνονται στην συλλογή.



Σχήμα 2.8: Περίπτωση χαμηλής ανάχλησης και υψηλής ακρίβειας

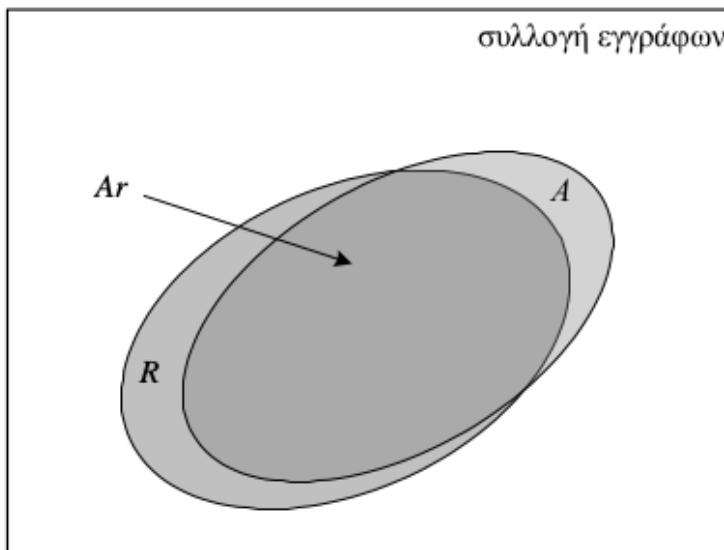
Στην εικόνα 2.9 παρουσιάζεται το ακριβώς αντίθετο από την περίπτωση της εικόνας 2.8

πιο πάνω. Έχουμε την ανάχληση να έχει ποσοστό στο 100% ενώ η ακρίβεια έχει χαμηλή τιμή. Αυτό σημαίνει ότι ανακτήθηκαν όλα τα σχετικά έγγραφα από το σύστημα αλλά και πολλά άλλα τα οποία δεν είναι σχετικά και άρα δεν έπρεπε να ανακτηθούν.



Σχήμα 2.9: Περίπτωση υψηλής ανάχλησης και χαμηλής ακρίβειας

Και τέλος στην εικόνα 2.10 φαίνεται η ιδανικότερη περίπτωση όπου ανάχληση και ακρίβεια έχουν ιδιαιτέρως υψηλά ποσοστά.



Σχήμα 2.10: Περίπτωση υψηλής ανάχλησης και υψηλής ακρίβειας

Καταλήγουμε στο συμπέρασμα ότι ένα σύστημα ανάχλησης και ακρίβειας κυμαίνεται μεταξύ της 2ης και 3ης περίπτωσης με ιδανικότερη την 4η.

Ακολουθεί ένα παράδειγμα: Ας θεωρήσουμε ότι για ένα ερώτημα q , τα σχετικά έγγραφα

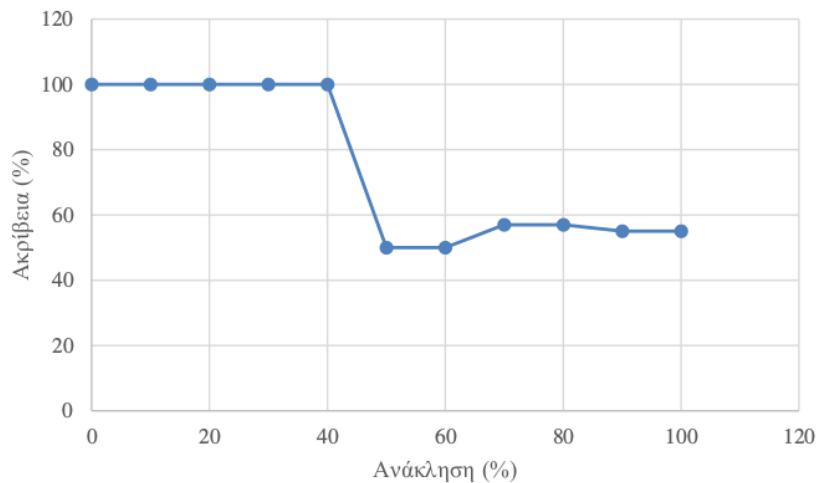
είναι τα παρακάτω: d_1, d_4, d_5, d_8 και d_9 . Ωστόσο όμως, το σύστημα επιστρέφει ως αποτέλεσμα τα $d_4, d_5, d_2, d_3, d_7, d_9, d_8, d_6$ και d_1 .

Στον παρακάτω πίνακα φαίνονται οι τιμές της ανάκλησης και της ακρίβειας από το παραπάνω αποτέλεσμα (Κεφάλαιο 2 του [12]).

A/A	Έγγραφο	Σχετικό	Ανάκληση	Ακρίβεια
1	d_4	Ναι	20%	100%
2	d_5	Ναι	40%	100%
3	d_2	Όχι	40%	66%
4	d_3	Όχι	40%	50%
5	d_7	Όχι	40%	40%
6	d_9	Ναι	60%	50%
7	d_8	Ναι	80%	57%
8	d_6	Όχι	80%	50%
9	d_1	Ναι	100%	55%

Πίνακας 2.2: Τιμές ανάκλησης και ακρίβειας παραδείγματος

Παρακάτω φαίνεται η γραφική παράσταση της καμπύλης στης σχέσης μεταξύ ανάκλησης και ακρίβειας:



Σχήμα 2.11: Γραφική παράσταση ανάκλησης/ακρίβειας

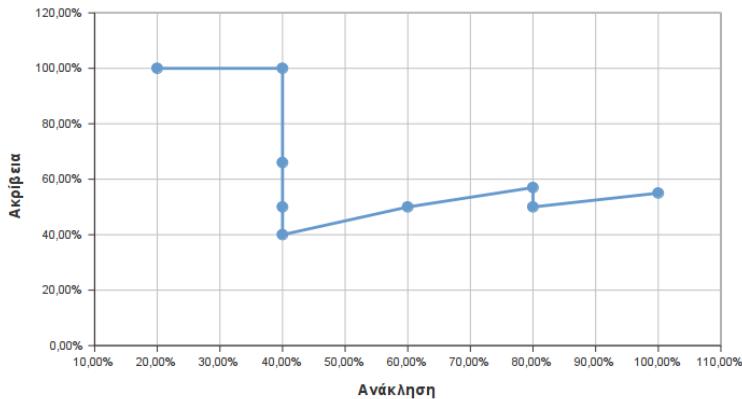
Η παραπάνω καμπύλη συνήθως διαμορφώνεται χρησιμοποιώντας πάντα 11 σημεία ανάκλησης (0%, 10%, ..., 100%) και αντιστοιχίζοντας τες με τις τιμές της ακρίβειας. Στην περίπτωση που δεν υπάρχει αντίστοιχη τιμή ανάκλησης, γίνεται χρήση της τεχνικής της παρεμβολής (interpolation). Με την τεχνική αυτή αντιστοιχίζεται η τιμή ανάκλησης για την οποία δεν υπάρχει τιμή ακρίβειας, η μέγιστη τιμή ακρίβειας που υπάρχει για το αμέσως επόμενο επίπεδο ανάκλησης (Κεφάλαιο 2 σελ. 29-30 του [12]).

Δηλαδή, όταν r_i το i -οστό επίπεδο ανάκλησης, με $p(r)$ την πραγματική τιμή τις ακρίβειας

για το επίπεδο της ανάκλησης r και με $\pi(i)$ την παρεμβαλλόμενη τιμή της ακρίβειας για το επίπεδο r_i , τότε η τιμή του $\pi(r_i)$ θα είναι:

$$\pi(r_i) = \max_{(r_i \leq r \leq r_{i+1})} (p(r)) \quad (2.10)$$

Με εφαρμογή της τεχνικής της παρεμβολής, η καμπύλη διαμορφώνεται ως εξής:



Σχήμα 2.12: Καμπύλη ανάκλησης - ακρίβειας με την τεχνική της παρεμβολής

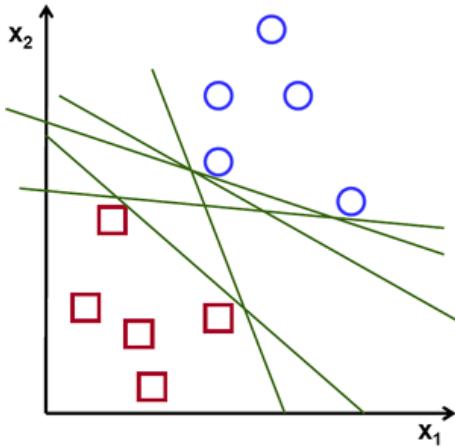
2.6 Αλγόριθμοι που χρησιμοποιήθηκαν

Παρακάτω ακολουθεί μία παρουσίαση των αλγορίθμων και των μοντέλων που επιλέχθηκαν για την υλοποίηση των πειραμάτων μηχανικής μάθησης στην παρούσα εργασία.

2.6.1 Support vector machine

Τα Support Vector Machines θεωρούνται από τις πλέον αξιόλογες τεχνικές ταξινόμησης, βασίζονται δε στη Στατιστική θεωρία εκμάθησης και παρουσιάζουν πολύ καλά αποτελέσματα σε πρακτικές εφαρμογές όπως η αναγνώριση χειρόγραφων εγγράφων. Επίσης εφαρμόζονται και με πολυδιάστατα δεδομένα χωρίς να αντιμετωπίζουν προβλήματα διαστατικότητας [3].

Κατά κύριο λόγο, τα SVM επιλύουν προβλήματα ταξινόμησης δύο κλάσεων με εξαιρετικά αποτελεσματικό τρόπο, ωστόσο όμως, με κατάλληλες τροποποιήσεις και επεκτάσεις της βασικής μεθοδολογίας, επιλύουν εξίσου αποδοτικά και άλλα προβλήματα όπως αυτό της παλινδρόμησης (regression) ή και αναγνώρισης προτύπων. Παρακάτω ακολουθεί ανάλυση της λογικής των SVM πάνω στο πρόβλημα των μεγίστων περιθωρίων των υπερεπιπέδων (Hyperplanes).



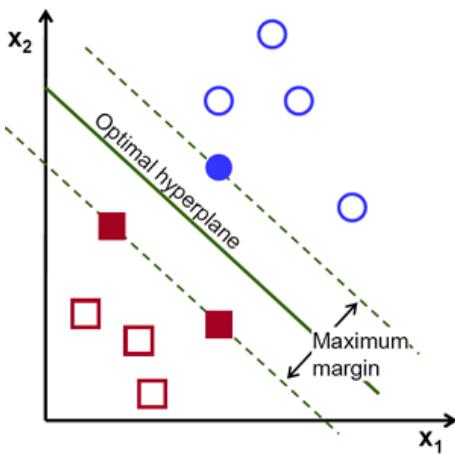
Σχήμα 2.13: Πιθανά υπερεπίπεδα που διαχωρίζουν σωστά τα αντικείμενα

Στην εικόνα 2.13 βλέπουμε ένα σύνολο στοιχείων από αντικείμενα δύο διαφορετικών κατηγοριών, η μεν πρώτη αντιπροσωπεύεται από τα τετράγωνα και η δε δεύτερη από τους κύκλους. Επίσης είναι εμφανές ότι το σύνολο των στοιχείων είναι γραμμικά διαχωρίσιμο από θεωρητικά άπειρα hyperplanes τέτοια ώστε όλα τα τετράγωνα να υπάρχουν στην μια πλευρά του hyperplane ενώ όλοι οι κύκλοι από την άλλη.

Αν και φαίνομενικά όλα τα hyperplanes διαχωρίζουν σωστά τους δύο τύπους κατηγοριών, υπάρχει αβεβαιότητα σχετικά με το πως θα συμπεριφερθεί μία τυχαία επιλογή hyperplane από τις τόσες διαθέσιμες, στα άγνωστα παραδείγματα του test set.

Το κριτήριο με το οποίο πρέπει επιλεχθεί το hyperplane με την βέλτιστη απόδοση είναι το εξής:

Η επιλογή hyperplane που βρίσκεται πολύ κοντά στα στοιχεία του συνόλου δεν θεωρείται καλή διότι θα είναι ευαίσθητη στον θόρυβο και δεν θα γενικευτεί σωστά. Ο στόχος μας είναι να βρεθεί τέτοιο hyperplane ώστε η γραμμή του να περνά όσο το δυνατόν πιο μακριά από όλα τα στοιχεία του επιπέδου.



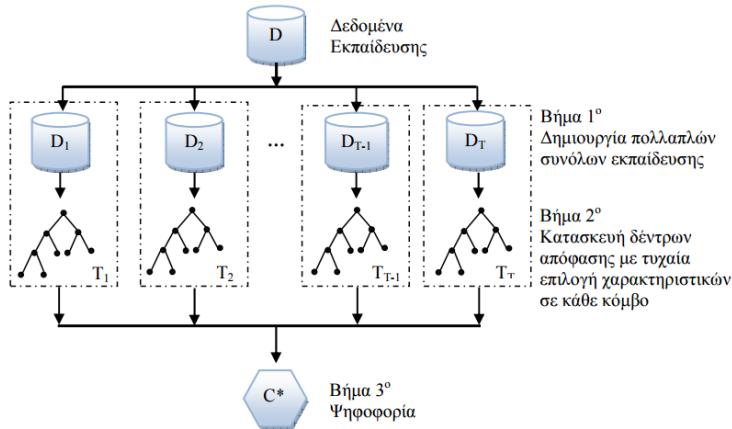
Σχήμα 2.14: Επιλογή του βέλτιστου υπερεπιπέδου

Στην εικόνα 2.14 βλέπουμε την επιλογή του αλγορίθμου SVM η οποία έχει το μέγιστο περιθώριο μεταξύ των κοντινότερων στοιχείων του επιπέδου.

Καλό θα ήταν να αναφερθεί ότι στο παραπάνω παράδειγμα έγινε εφαρμογή του αλγορίθμου στο Καρτεσιανό επίπεδο και όχι σε πολυδιάστατο χώρο. Η επιλογή αυτή έγινε για λόγους απλότητας και καλύτερης κατανόησης του προβλήματος. Η γενική μορφή του αλγορίθμου ωστόσο απευθύνεται σε προβλήματα ταξινόμησης σε χώρους με πάνω από δύο διαστάσεις.

2.6.2 Random forest

Ο αλγόριθμος Random Forest είναι μία σχετικά νέα τεχνική ταξινόμησης δεδομένων η οποία είναι βασίζεται στην μέθοδο των Decision Trees (Δέντρα Απόφασης). Στην ουσία, ένα σύνολο από decision trees δημιουργούν ένα Random Forest μοντέλο. Η μέθοδος προτάθηκε από τον Leo Breiman [9].



Σχήμα 2.15: Βήματα κατασκευής ενός Random forest

Αρχικά το σύνολο των δεδομένων εκπαίδευσης ανατίθεται στην ρίζα του δέντρου απόφασης κατά την κατασκευή του. Στους ενδιάμεσους κόμβους ανατίθενται υποσύνολα των δεδομένων εκπαίδευσης τα οποία διαχωρίζονται, βάσει εφαρμογής κάποιας κατάλληλης μεθόδου, σε μικρότερα υποσύνολα τόσα στο πλήθος όσα και τα παιδιά του επόμενου επιπέδου στο δέντρο.

Συνήθως η μέθοδος διαχωρισμού των δεδομένων εκπαίδευσης αφορά ένα υποσύνολο των χαρακτηριστικών τους και η επιλογή του βέλτιστου διαχωρισμού γίνεται σύμφωνα με ένα κατάλληλο μέτρο (Gini index, misclassification error, εντροπία). Τα δέντρα του δάσους αναπτύσσονται στο μέγιστο μέγεθος τους, χωρίς «κλάδεμα».

Τυπάρχει μία ειδική κατηγορία των Random Forests στην οποία χρησιμοποιούνται για ταξινομητές δέντρα απόφασης. Αυτή είναι η μέθοδος bagging στην οποία η τυχαιότητα ενσωματώνεται στο μοντέλο μέσω της τυχαίας επιλογής N δεδομένων εκπαίδευσης, με επαναποιητήση, από το αρχικό σύνολο εκπαίδευσης.

Η διαδικασία της ταξινόμησης των άγνωστων δεδομένων υλοποιείται μέσω της διάσχισης όλων των δέντρων του δάσους από την ρίζα προς κάποιο από τα φύλλα. Έπειτα, υπολογίζονται τα αποτελέσματα όλων των δέντρων και η τελική κατηγορία ταξινόμησης επιλέγεται βάσει ενός

πλειοψηφικού συστήματος ψηφοφορίας. Κάθε παράδειγμα ανατίθεται στην κατηγορία με τη μεγαλύτερη συχνότητα.

Ακολουθούν τα κυριότερα πλεονεκτήματα των Random Forests:

- Αν και εκπαιδεύονται σε σύνολα δεδομένων υψηλής διάστασης όπως κείμενα ή εικόνες, δεν εμφανίζουν σημαντικό βαθμό υπερεκπαίδευσης (overfitting).
- Περιορισμένο σφάλμα γενίκευσης κατά βάση εξαιτίας του μεγάλου πλήθους δέντρων στο δάσος γεγονός το οποίο έχει ως αποτέλεσμα την σπάνια εμφάνιση υπερεκπαίδευσης.
- Ολοκλήρωση αλγορίθμου σε σταθερό αριθμό βημάτων. Αποφυγή επαναληπτικών διαδικασιών εκπαίδευσης.

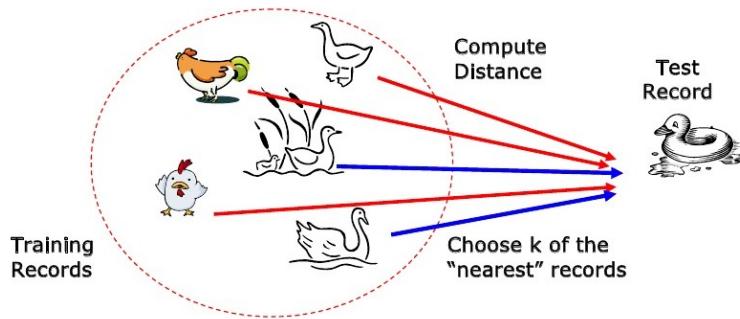
Παρακάτω βλέπουμε κάποια σημαντικά μειονεκτήματα:

- Υψηλό υπολογιστικό κόστος.
- Η εξαιρετικά λεπτομερής παραμετροποίηση του συχνά προκαλεί σύγχυση κατά την κατασκευή του μοντέλου. Δημαρχός ο χρήστης πρέπει να προσδιορίσει π.χ. πλήθος δέντρων, βαθμός κόμβων, μέγεθος δεδομένων εκπαίδευσης, συνθήκη τερματισμού διαμέρισης των κόμβων.
- Έλλειψη ευελιξίας μιας και για την εισαγωγή μιας νέας κατηγορίας, απαιτείται η κατασκευή του μοντέλου από την αρχή.
- Τα νέα προς μελέτη δεδομένα πρέπει να διασχίσουν όλα τα δέντρα του δάσους για την εκτίμηση της κατηγορίας τους.

2.6.3 K-Nearest Neighbors

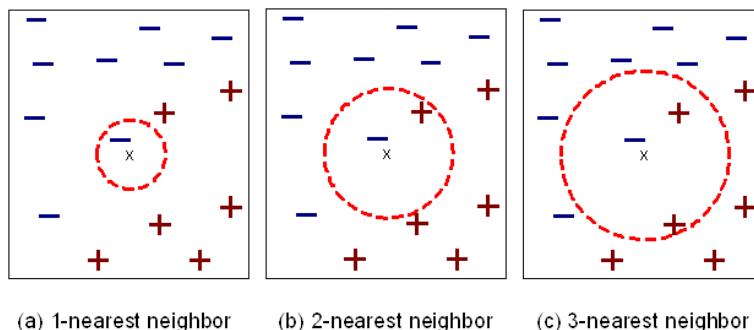
Ο αλγόριθμος Πλησιέστερου Γείτονα (Nearest Neighbor) θεωρείται από τους πιο απλούς αλγορίθμους στον τομέα της μηχανικής μάθησης. Η ιδιαιτερότητα του είναι ότι καθυστερεί την διαδικασία της μοντελοποίησης – διαμόρφωσης του training set μέχρι αυτό να είναι απαραίτητο για την ταξινόμηση – classification του test set. Λόγω της παραπάνω ιδιότητας κατατάσσεται σε μία κατηγορία ταξινομητών τους λεγόμενους «τεμπέλικους ταξινομητές μάθησης» (lazy learners) [11].

Η παραπάνω τεχνική μπορεί να βελτιωθεί και να γίνει πιο ευέλικτη εάν βρεθούν όλα εκείνα τα παραδείγματα του training set τα οποία έχουν σχετικά όμοια χαρακτηριστικά γνωρίσματα και ιδιότητες με την προς ταξινόμηση εγγραφή η οποία μέχρι πριν ήταν άγνωστη στον ταξινομητή. Το σύνολο αυτών των παραδειγμάτων, τα οποία είναι γνωστά σαν πλησιέστεροι γείτονες (nearest neighbors), χρησιμοποιούνται για να προσδιοριστεί η επικέτα κατηγορίας της προς ταξινόμηση εγγραφής. Η αιτιολόγηση του συγκεκριμένου παραδείγματος μπορεί να εξηγηθεί καλύτερα με την κλασική πλέον φράση: «Εάν περπατά σαν πάπια και φωνάζει σαν πάπια και μοιάζει με πάπια τότε είναι πάπια».



Σχήμα 2.16: Βήματα ενός K-NN αλγορίθμου

Οι ταξινομητές πλησιέστερων γειτόνων (nearest neighbour) αναπαριστούν κάθε αντικείμενο σαν ένα σημείο δεδομένων σε ένα d -διάστατο χώρο, όπου d εκτός από τον αριθμό των διαστάσεων είναι και το πλήθος των χαρακτηριστικών γνωρισμάτων. Σε μία τυχαία είσοδο προς ταξινόμηση, έστω z , μπορούμε να υπολογίσουμε το πόσο «κοντά» βρίσκεται στα υπόλοιπα σημεία δεδομένων (data points) του training set. Οι k -nearest neighbours της παραπάνω εισόδου z , αναφέρονται στα k σημεία τα οποία βρίσκονται πιο κοντά σε αυτή.

Σχήμα 2.17: Περιπτώσεις K-nn με $k = 1, k = 2$ και $k = 3$

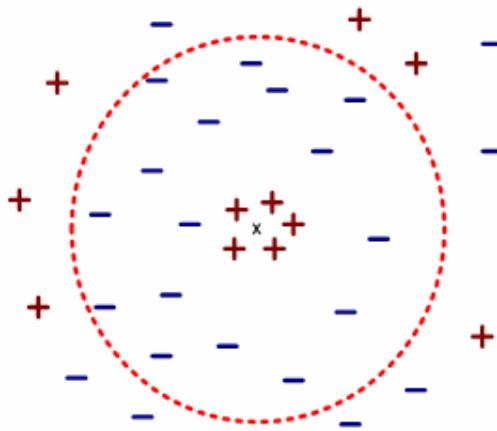
Στην εικόνα 2.17 φαίνονται οι περιπτώσεις όπου το k έχει ως τιμές τους αριθμούς 1, 2 και 3 για ένα σημείο δεδομένων το οποίο βρίσκεται στο κέντρο του κάθε κύκλου. Η απόφαση για την πρόβλεψη της ταξινόμησης του σημείου θα βασιστεί στις ετικέτες κατηγορίας των γειτονικών του σημείων. Στην περίπτωση όπου το σύνολο των γειτονικών στοιχείων ανήκει σε διαφορετικές κατηγορίες, τότε το σημείο δεδομένων ταξινομείται στην κατηγορία που ανήκει η πλειοψηφία των κοντινότερων γειτόνων.

Παρακάτω φαίνεται πόσο σημαντικό είναι η επιλογή της κατάλληλης τιμής για τον k . Στην Εικόνα 2.17 στην περίπτωση (a), ο 1-nearest neighbour του σημείου δεδομένων είναι ένα αρνητικό παράδειγμα. Επομένως το σημείο δεδομένων ταξινομείται στην αρνητική κατηγορία. Εάν το $k = 3$, (αριθμός των nearest neighbours), όπως βλέπουμε στην περίπτωση (c), τότε η γειτονιά περιέχει δύο θετικά και ένα αρνητικό παράδειγμα. Βάσει πλειοψηφίας, το προς ταξινόμηση σημείο θα κατηγοριοποιηθεί στην θετική κατηγορία. Στην περίπτωση ισοπαλίας μεταξύ δύο κατηγοριών, όπως στην περίπτωση (b), γίνεται μία τυχαία επιλογή από τις δύο κατηγορίες.

ταξινομείται σε αυτή το σημείο δεδομένων μας.

Συνεπώς, εάν στο k αποδοθεί μία σχετικά χαμηλή τιμή, τότε ο ταξινομητής nearest neighbour είναι ευαίσθητος στο θόρυβο διότι ο χώρος που λαμβάνεται υπόψιν είναι ιδιαίτερα μικρός και οποιαδήποτε αλλαγή μπορεί να αλλοιώσει τα δεδομένα των γειτόνων με αποτέλεσμα να έχουμε μία λάθος ταξινόμηση.

Από την άλλη ωστόσο, αν το k είναι ιδιαιτέρως μεγάλο, ο ταξινομητής nearest neighbour μπορεί να προβλέψει λανθασμένη κατηγορία για το προς εξέταση αντικείμενο. Αυτό συμβαίνει λόγω του μεγέθους της λίστας των γειτόνων, η οποία, εξαιτίας της μεγάλης τιμής του k , θα είναι πολύ μεγάλη και θα λαμβάνει υπόψη τις επικέτες κατηγορίας των πολύ μακρινών γειτόνων, οι οποίοι πιθανότατα να μην έχουν τόσο όμοια χαρακτηριστικά γνωρίσματα παρόλο που βρίσκονται μέσα την περιοχή δειγμάτων.



Σχήμα 2.18: Περίπτωση K-nn με λανθασμένη ταξινόμηση

Στο παράδειγμα της εικόνας 2.18, οι κοντινοί γείτονες γύρω από το προς εξέταση σημείο είναι όλοι θετικοί, ωστόσο όμως στην πλειοφηφία τους οι γείτονες είναι ταξινομημένοι στην αρνητική κατηγορία και ας βρίσκονται μακριά του. Αυτό έχει σαν αποτέλεσμα την λανθασμένη ταξινόμηση του σημείου δεδομένων ως αρνητικό.

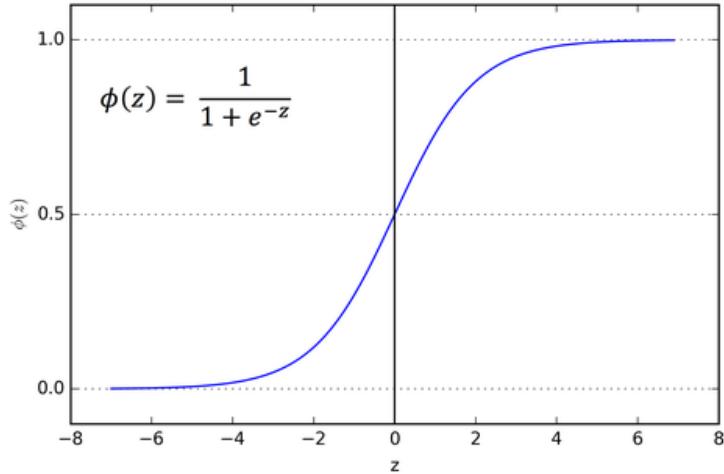
2.6.4 Logistic Regression

Το μοντέλο λογιστικής παλινδρόμησης (logistic regression) είναι ένα μη γραμμικό μοντέλο του οποίου τα σφάλματα δεν ακολουθούν κανονική κατανομή και η μεταβλητή απόκρισης είναι διακριτή. Η συνηθέστερη χρήση αυτού του μοντέλου αφορά την πρόβλεψη απουσίας ή παρουσίας ενός χαρακτηριστικού. Ουσιαστικά πρόκειται για μία γενικευμένη μορφή της γραμμικής παλινδρόμησης (linear regression) με την διαφορά ότι η εξαρτημένη μεταβλητή παίρνει δύο τιμές (τιμή 0 όταν απουσιάζει το χαρακτηριστικό ή τιμή 1 όταν υπάρχει).

Συγκεκριμένα, στο λογιστικό υπόδειγμα γίνεται χρήση της γνωστής λογιστικής συνάρτησης:

$$f(z) = \frac{e^z}{1 + e^z} = \frac{1}{1 + e^{-z}} \quad (2.11)$$

η οποία υπολογίζει την πιθανότητα την απουσίας ή παρουσίας του υπό μελέτη χαρακτηριστικού.



Σχήμα 2.19: Σιγμοειδής συνάρτηση

Το βασικότερο μειονέκτημα του μοντέλου της λογιστικής παλινδρόμησης είναι ότι κατά την κατασκευή του πρέπει να αποφεύγεται η χρησιμοποίηση μεταβλητών που παρουσιάζουν μεγάλη συσχέτιση. Επίσης, αν και δεν απαιτείται οι μεταβλητές να ακολουθούν την χανονική κατανομή, στην περίπτωση που αυτές χαρακτηρίζονται από ακραία μη-χανονικότητα, τότε πιθανότατα τα αποτελέσματα του μοντέλου δεν θα είναι αξιόπιστα.

Ένα από τα πλεονεκτήματα της λογιστικής παλινδρόμησης είναι ότι μπορεί να εκτιμηθεί η σπουδαιότητα κάθε χαρακτηριστικού στο εξαγόμενο αποτέλεσμα. Πολλοί ερευνητές χρησιμοποιούν το μοντέλο λογιστικής παλινδρόμησης για να επιλέξουν ποια από τα αρχικά χαρακτηριστικά θα χρησιμοποιηθούν στην ανάλυση και ποια όχι. Έπειτα δημιουργείται ένα νέο μοντέλο ταξινόμησης με την ίδια ή κάποια άλλη τεχνική. Η εξαίρεση των μη σημαντικών χαρακτηριστικών (feature selection) πιθανότατα θα συμβάλει στην βελτίωση της προβλεπτικής ικανότητας του μοντέλου.

2.6.5 Naive Bayes

Πρόκειται για μια τεχνική ταξινόμησης βασισμένη στο Θεώρημα του Bayes στην οποία γίνεται η παραδοχή της ανεξαρτησίας μεταξύ των χαρακτηριστικών γνωρισμάτων ενός αντικειμένου. Δηλαδή, ένας ταξινομητής Naive Bayes κάνει την υπόθεση ότι η παρουσία ενός συγκεκριμένου χαρακτηριστικού σε μια κατηγορία δεν σχετίζεται με την παρουσία οποιουδήποτε άλλου χαρακτηριστικού. Για παράδειγμα, ένα φρούτο μπορεί να θεωρηθεί λεμόνι αν είναι κίτρινο, στρογγυλό και περίπου 10 εκατοστά σε διάμετρο.

Ακόμη και αν αυτά τα χαρακτηριστικά εξαρτώνται το ένα από το άλλο ή από την ύπαρξη άλλων χαρακτηριστικών, συμβάλλουν ανεξάρτητα στην πιθανότητα αυτό το λαχανικό να είναι λεμόνι και για αυτό το λόγος ο αλγόριθμος είναι γνωστός ως 'Naive'. Το μοντέλο Naive Bayes κατασκευάζεται εύκολα και έχει αποδειχθεί ιδιαίτερα χρήσιμο για πολύ μεγάλα σύνολα

δεδομένων. Λόγω της απλότητας του, έχει γίνει αρκετά δημοφιλής και σε μερικές περιπτώσεις ξεπερνάει ακόμη και πολύ εξελιγμένες μεθόδους ταξινόμησης.

Μια παραλλαγή της κλασσικής μεθόδου έχει μελετηθεί [4] και αποδείχθηκε ωφέλιμη ως προς την απόδοση άλλα και την εξοικονόμηση υπολογιστικών πόρων. Συγκεκριμένα έχουν εισαχθεί κάποιες ιδιότητας του αλγορίθμου tf-idf, δηλαδή η συχνότητα της κάθε λέξης εντός του εγγράφου (αυξημένη σημαντικότητα) καθώς και η συχνότητα της λέξης σε όλα τα έγγραφα της συλλογής (μειωμένη σημαντικότητα).

Η εφαρμογή του Naive Bayes μοντέλου σε προβλήματα ταξινόμησης κειμενικής πληροφορίας προϋποθέτει την ύπαρξη ενός λεξικού που περιέχει τις λέξεις εκείνες από τις οποίες αποτελείται το training set. Στην περίπτωση μας πρόκειται για τις στήλες του διανυσματικού πίνακα (document-term matrix).

Τυποθέτουμε ότι το πρόβλημα που αντιμετωπίζεται είναι η ταξινόμηση εγγράφων στις κατηγορίες K_1, K_2, \dots, K_i . Η πιθανότητα ένα νέο έγγραφο δ να ανήκει στην κατηγορία K_i είναι:

$$f(d, P(K_i)) = \frac{\# \text{ of documents in category } K_i}{\# \text{ of all documents in collection}} \quad (2.12)$$

Η κάθε λέξη w_j του εξεταζόμενου εγγράφου d συγκρίνεται με το λεξικό που δημιουργήθηκε από την διαδικασία εκπαιδεύσεως και εντοπίζονται οι περιπτώσεις n_w όπου η λέξη ταιριάζει σε μία κατηγορία K_i .

Η δεσμευμένη πιθανότητα η λέξη w_j να περιέχεται στην κατηγορία K_i είναι:

$$P(w_j|K_i) = \frac{n_w}{n} \quad (2.13)$$

και η πιθανότητα το υπό εξέταση έγγραφο d να ανήκει στην κατηγορία K_i είναι:

$$P(d|K_i) = \prod_{j=1}^n P(w_j|K_i) \quad (2.14)$$

Τέλος, η πιθανότητα η κατηγορία K_i να περιέχει το έγγραφο d , έχοντας ήδη την γνώση ότι έχει ταξινομηθεί σε αυτή, είναι:

$$P(K_i|d) = \frac{P(K_i) \cdot P(d|K_i)}{P(d_i)} \quad (2.15)$$

Η μέγιστη δεσμευμένη πιθανότητα $P(K_i|d)$ θα υλοποιήσει και την πρόβλεψη της κατηγορίας στην οποία τελικά θα ταξινομηθεί το έγγραφο d .

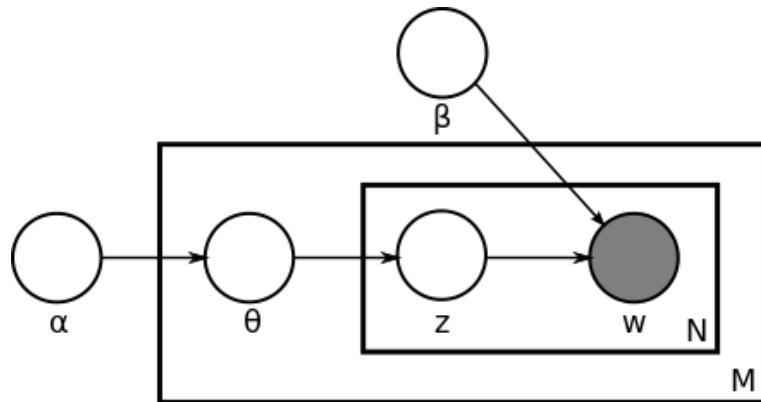
2.6.6 Latent Dirichlet Allocation

Η διαδικασία της ομαδοποίησης εγγράφων ή κειμενικής πληροφορίας (text clustering) θα μπορούσε να οριστεί ως ο διαχωρισμός των κειμένων φυσικής γλώσσας σε ξεχωριστές κατηγορίες η οποίες μας είναι εξ αρχής άγνωστες. Αφενός η ανάγκη για οργάνωση του συνεχώς αυξανόμενου όγκου πληροφοριών και αφετέρου της τεχνογνωσίας της μηχανικής μάθησης που επιτρέπει την ανάπτυξη περισσότερων ιδιαιτέρων «ευφυών» μεθόδων, οδηγούν στην συνεχή ανάπτυξη και βελτίωση όλο και αποδοτικότερων συστημάτων αυτόματης ομαδοποίησης

εγγράφων. Πρώτη φορά παρουσιάστηκε από τους David Blei, Andrew Ng και Michael I. Jordan το 2003 [1].

Ο αλγόριθμος Λανθάνουσας Ανάθεσης Dirichlet, ή Latent Dirichlet Allocation (LDA), είναι ένα γενικευμένο στατιστικό μοντέλο το οποίο ανήκει στην κατηγορία αλγορίθμων της επεξεργασίας της φυσικής γλώσσας (NLP) και έχει την δυνατότητα να ανιχνεύει κοινά θεματικά αντικείμενα από μέχρι πρότινος άγνωστα σε αυτό δεδομένα, ομαδοποιώντας τα ανά θεματικές ενότητες κοινών χαρακτηριστικών (topics). Εαν, για παράδειγμα, τα δεδομένα είναι έγγραφα κειμένου, κάνει εκτιμήσεις και υποθέσεις ότι κάθε έγγραφο ανήκει σε ένα μικρό σύνολο θεματικών ενοτήτων και πως κάθε λέξη αποδίδεται σε μία από αυτές τις ενότητες με κάποια κατανομή.

Πιο συγκεκριμένα, η κάθε θεματική ενότητα ορίζεται ως μία κατανομή σε ένα σύνολο από λέξεις, το οποίο παράγεται από την συλλογή των εγγράφων. Το κάθε έγγραφο αποτελείται από λέξεις οι οποίες όμως ανήκουν σε διαφορετικά θέματα. Για παράδειγμα από δύο θέματα (έστω t_1 και t_2) που έχουν διαφορετικά λεξιλόγια το καθένα, δημιουργούνται διαφορετικά έγγραφα (έστω d_1 , d_2 και d_3) των οποίων η κάθε λέξη προέρχεται είτε από το ένα θέμα ή από το άλλο με κάποια πιθανότητα.



Σχήμα 2.20: Αναπαράσταση επιπέδων στην LDA

Στο σχήμα 2.20 παραπάνω, φαίνονται τα τρία επίπεδα στην αναπαράσταση της LDA: τα α και β είναι παράμετροι που αναφέρονται στη συλλογή των κειμένων και υπολογίζονται μία φορά κατά τη δημιουργία της συλλογής κειμένων. Η μεταβλητή θ αναφέρεται στο έγγραφο και υπολογίζεται επίσης μία φορά για κάθε έγγραφο. Τέλος, οι μεταβλητές z και w αναφέρονται στις λέξεις και υπολογίζονται μία φορά για κάθε λέξη που περιέχεται στο υπό εξέταση έγγραφο. N είναι το έγγραφο που αποτελείται από τις λέξεις και M η συλλογή των εγγράφων [14].

Κεφάλαιο 3

Πηγές και συλλογές δεδομένων που χρησιμοποιήθηκαν (Datasets)

Στο κεφάλαιο αυτό περιγράφουμε τις πηγές δεδομένων πάνω στις οποίες υλοποιήθηκαν τα πειράματα και οι δοκιμές των μοντέλων και των αλγορίθμων. Επίσης παρουσιάζονται οι περιγραφές των δομών, η διαδικασία της ετοιμασίας των δεδομένων (pre-processing). Επίσης περιγράφεται η διαδικασία εμπλουτισμού του dataset της παραγράφου.

3.1 Six Categories of Amazon Product Reviews

Για τις ανάγκες τις πτυχιακής αυτής, χρησιμοποιήσαμε το Six Categories of Amazon Product Reviews το οποίο διανέμεται από το University of Illinois at Urbana-Champaign¹ και περιλαμβάνει κριτικές προϊόντων τα οποία ανήκουν στις έξι κατηγορίες που ακολουθούν:

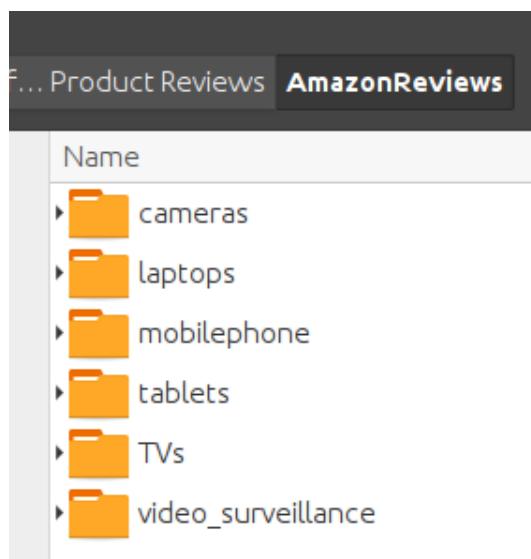
- Camera
- Mobile phone
- TV
- Laptop
- Tablet
- Video surveillance system

¹<http://bit.ly/wang296Data>

Πλήθος κριτικών	1,116,526
Πλήθος χρηστών	439,136
Πλήθος προϊόντων	17,219
Χρήστες με πάνω από 50 σχόλια	671
Μέσο μήκος review	667
Περίοδος	Ιουν 1999 - Ιουν 2014

Πίνακας 3.1: Στατιστικά στοιχεία δεδομένων του Six Categories of Amazon Product Reviews dataset

Τα δεδομένα είναι αποθηκευμένα σε αρχεία τύπου JSON (JavaScript Object Notation) τα οποία βρίσκονται μέσα σε φακέλους όπου ο κάθε φάκελος αποτελεί το σύνολο των εγγραφών μίας κατηγορίας.



Σχήμα 3.1: Σχήμα διάρθρωσης φακέλων του Dataset

3.1.1 Περιγραφή δομής της Πηγής

Το κάθε αρχείο έχει ως όνομα το ASIN² του προϊόντος και για περιλαμβάνει τα παρακάτω στοιχεία για το καθένα:

1. "ProductInfo": {
2. "Price": "Unavailable",
3. "Features": "14.1 Megapixels with 720p HD",
4. "Name": "Casio Exilim EX-ZS10 Silver 14 MP",
5. "ImgURL": "http://img-amzn.com/image/SX.jpg",
6. "ProductID": "B004HYGDJ2" }

²<https://www.amazon.com/gp/seller/asin-upc-isbn-info.html>

όπου:

- Γραμμή 1.: ProductInfo: Έναρξη περιγραφής του προϊόντος.
- Γραμμή 2.: Price: Η τιμή του προϊόντος.
- Γραμμή 3.: Features: Κάποια χαρακτηριστικά.
- Γραμμή 4.: Name: Ονομασία.
- Γραμμή 5.: ImgURL: Το URL της φωτογραφίας του προϊόντος.
- Γραμμή 6.: ProductID: Είναι το ASIN το οποίο καθορίζει μονοσήμαντα το εκάστοτε προϊόν σε ολόκληρο το σύστημα της Amazon.

επίσης περιλαμβάνει και μία λίστα με χριτικές όπως παρακάτω:

```

1. "Reviews": [
2.   {
3.     "Title": "CASIO EXILIM EX-ZS10",
4.     "Author": "A. Solano",
5.     "ReviewID": "R385TOYIAYOHRU",
6.     "Overall": "2.0",
7.     "Content": "Very poor quality images",
8.     "Date": "July 29, 2011"
9.   },
10.  ....
11.  {
12.    "Title": "Sample of the video recording",
13.    "Author": "W. Taylor",
14.    "ReviewID": "RN3Q7RCZOYRBH",
15.    "Overall": "4.0",
16.    "Content": "Here is an example situations",
17.    "Date": "March 28, 2011"
18.  }
19.]
```

όπου:

- Γραμμή 1.: Reviews: Έναρξη περιγραφής της χριτικής του προϊόντος.
- Γραμμή 3.: Title: Ο τίτλος της χριτικής.
- Γραμμή 4.: Author: Ο συγγραφέας της χριτικής (Ονομ/μο χρήστη - πελάτη).
- Γραμμή 5.: ReviewID: Το id της χριτικής.
- Γραμμή 6.: Overall: Βαθμολογία.
- Γραμμή 7.: Content: Το κείμενο της χριτικής.
- Γραμμή 8.: Date: Η ημερομηνία δημοσίευσης.
- Γραμμές 11-18.: Επαναλαμβάνεται η ίδια δομή για όλες τις χριτικές που υπάρχουν στο αρχείο.

Σχετικά με την πληρότητα των εγγραφών του dataset, υπήρξαν αρχετά reviews με special characters ή ακόμη και διαφορετικών κωδικοποιήσεων με αποτέλεσμα να χρειάζονται συχνά διαδικασίες καιναρισμού των δεδομένων. Ο εντοπισμός αυτών των εγγραφών δεν ήταν πάντα εύκολος, μιας και το πλήθος των instances είναι πάνω από 500,000. Επίσης υπήρχαν αρχετά μονολεκτικά reviews που αποπροσανατόλιζαν τα μοντέλα και μάλιστα σε γλώσσες όπως η Ισπανική ή ακόμη και ιδεογράμματα από Ασιατικές γλώσσες.

3.1.2 Χρησιμότητα και εφαρμογές

Η συγκεκριμένη πηγή μπορεί να χρησιμοποιηθεί σε διάφορες τεχνικές μηχανικής μάθησης όπως classification ή clustering, από τις οποίες μπορεί να προκύψουν χρήσιμα συμπεράσματα, όπως για παράδειγμα σε ποιες κατηγορίες προϊόντων υπάρχουν τα περισσότερα ή πλουσιότερα - βιοηθητικά reviews. Επίσης προσφέρεται και για Sentiment Analysis βάσει των σχολίων των χρηστών. Εξίσου πολύτιμη είναι η πληροφορία που εξάγεται σχετικά με το πόσο βιοηθητικά είναι τα παρεχόμενα reviews η οποία και μπορεί να χρησιμοποιηθεί για να επιφέρει πιθανή αύξηση των πωλήσεων στα πλαίσια του marketing της εκάστοτε εταιρείας.

Πιο συγκεκριμένα στην εργασία [15] οι συγγραφείς προτείνουν ως άμεσα υλοποιόμενες και εκμεταλλεύσιμες εφαρμογές των αποτελεσμάτων τους ως:

- Σύνοψη κριτικών (aspect opinion summarization) όπου αυτό το είδος ενδελεχούς πληροφορίας θα ήταν εξαιρετικά βιοηθητικό και χρήσιμο στους χρήστες ώστε να προκύψει μια έγκυρη και συνοπτική πληροφορία αξιολόγησης για ένα προϊόν που όμως εξάγεται από ένα εξαιρετικά μεγάλο πλήθος κριτικών.
- Ταξινόμηση προϊόντων βάσει κριτικών δηλαδή να προβάλει στον χρήστη κριτικές από χρήστες που είχαν παρόμοιο τρόπο αξιολόγησης όπως αυτός με αποτέλεσμα να τον βιοθίσει να λάβει μια απόφαση πιθανώς πιο σωστή.
- Ανάλυση της συμπεριφοράς με την οποία βαθμολόγησε ο χρήστης ώστε να αποφασίζεται ποιοι παράγοντες τον επηρέασαν προκειμένου να υποβάλλει την κριτική - αξιολόγηση του.

Επίσης, στην εργασία [16], η πηγή δεδομένων χρησιμοποιείται για την δημιουργία ενός μοντέλου (LARAM) το οποίο εξαιτίας της γενικότητας του, μπορεί να βρει εφαρμογή σχεδόν σε όλες τις πηγές που περιλαμβάνουν δεδομένα κείμενου με κριτικές - γνώμες. Τέλος, το εν λόγω μοντέλο μπορεί να χρησιμοποιηθεί εξίσου και στις ανωτέρω αναφερόμενες εφαρμογές.

3.2 Amazon movie reviews dataset

Παράλληλα με την παραπάνω πηγή, χρησιμοποιήσαμε και το Amazon movie reviews dataset το οποίο διανέμεται από το Stanford University³ και αποτελείται από κριτικές ταινιών (CD, DVD κλπ) που διατίθενται από την Amazon. Τα δεδομένα καλύπτουν μια περίοδο άνω

³<https://snap.stanford.edu/data/web-Movies.html>

των 10 ετών (από τον Αύγουστο του 1997 μέχρι και τον Οκτώβριο του 2012) και περιλαμβάνει περίπου 8 εκατομμύρια κριτικές. Αυτές περιλαμβάνουν πληροφορίες για προϊόντα και χρήστες, αξιολογήσεις και μια αναλυτική παρουσίαση.

Πλήθος κριτικών	7,911,684
Πλήθος χρηστών	889,176
Πλήθος προϊόντων	253,059
Χρήστες με πάνω από 50 σχόλια	16,341
Μέσο μήκος review	101
Περίοδος	Αυγ 1997 - Οκτ. 2012

Πίνακας 3.2: Στατιστικά στοιχεία δεδομένων του Amazon movie reviews dataset

3.2.1 Περιγραφή δομής της Πηγής

Ακολουθεί μία περιγραφή της δομής αναφορικά με κάθε εγγραφή του δατασετ όπως διατίθεται στο αποθετήριο SNAP.

Παρακάτω ακολουθεί αναλυτική περιγραφή των πεδίων ανά γραμμή:

- 1.product/productId: B00006HAXW
- 2.review/userId: A1RSDE90N6RSZF
- 3.review/profileName: Joseph M. Kotow
- 4.review/helpfulness: 9/9
- 5.review/score: 5.0
- 6.review/time: 1042502400
- 7.review/summary: Pittsburgh - Home of the OLDIES
- 8.review/text: I have all of the doo wop DVD's and this one is as good or better than the 1st ones. Remember once these performers are gone, we'll never get to see them again. Rhino did an excellent job and if you like or love doo wop and Rock n Roll you'll LOVE this DVD!!

όπου:

Γραμμή 1.: product/productId: Είναι το asin (footnote # 2 στην σελίδα 28) του εκάστοτε προϊόντος πχ: B00006HAXW με URL το <http://amazon.com/dp/B00006HAXW>.

Γραμμή 2.: review/userId: Το id του εκάστοτε χρήστη πχ: A1RSDE90N6RSZF.

Γραμμή 3.: review/profileName: Όνομα χρήστη.

Γραμμή 4.: review/helpfulness: Αναλογία χρηστών που βρήκαν την κριτική βοηθητική.

Γραμμή 5.: review/score: Βαθμολογία του προϊόντος.

Γραμμή 6.: review/time: Ημερομηνία-ώρα που υποβλήθηκε η κριτική (σε unix time format).

Γραμμή 7.: review/summary: Περίληψη κριτικής.

Γραμμή 8.: review/text: Πλήρες κείμενο της κριτικής.

Σχετικά με την πληρότητα των εγγραφών του dataset, συχνό ήταν το φαινόμενο να υπάρχουν html tags εντός του κειμένου των κριτικών. Το γεγονός αυτό δημιούργησε αρκετά προβλήματα κατά την διαδικασία της συσταδοποίησης καθώς κατά την προσπέλαση των δεδομένων από το μοντέλο, προκειμένου να υπολογίσει τις συστάδες, θεώρησε ως μεμονωμένο cluster το σύνολο αυτών των tags. Και σε αυτό το dataset υπήρχαν αρκετά μονολεκτικά reviews που αποπροσανατόλιζαν τα μοντέλα και μάλιστα σε γλώσσες όπως η Ισπανική ή ακόμη και Ιδεογράμματα από Ασιατικές γλώσσες.

3.2.2 Χρησιμότητα και εφαρμογές

Το συγκεκριμένο dataset μπορεί να χρησιμοποιηθεί (όπως και το προηγούμενο) σε διάφορες τεχνικές μηχανικής μάθησης όπως classification ή clustering, από τις οποίες μπορεί να προκύψουν χρήσιμα συμπεράσματα όπως για παράδειγμα σε ποιες κατηγορίες προϊόντων υπάρχουν τα περισσότερα ή πλουσιότερα - βιοηθητικά reviews. Επίσης προσφέρεται και για Sentiment Analysis βάσει των σχολίων των χρηστών. Εξίσου πολύτιμη είναι η πληροφορία που εξάγεται σχετικά με το πόσο βιοηθητικά είναι τα παρεχόμενα reviews η οποία και που μπορεί να χρησιμοποιηθεί για να επιφέρει πιθανή αύξηση των πωλήσεων στα πλαίσια του marketing της εκάστοτε εταιρείας.

Μια σημαντική εργασία που δείχνει την χρησιμότητα, καθώς και εφαρμογές που μπορούν να προκύψουν από την ανάλυση του dataset είναι η [7]. Εκεί οι συγγραφείς δημιούργησαν ένα recommendation system όπου καταφέρνουν να προτείνουν ένα προϊόν στον χρήστη, όχι μόνο βάσει των προτιμήσεων του, αλλά ακόμη λαμβάνοντας υπόψη και την εμπειρία του χρήστη καθώς και την καταναλωτική του συμπεριφορά. Φαίνεται ότι όχι μόνο προκύπτουν καλύτερες προτάσεις για τους χρήστες αλλά δίνεται η δυνατότητα να μελετηθεί και η συμπεριφορά - εμπειρία του χρήστη σε αυτό το μεγάλο dataset.

3.3 Επεξεργασία των Dataset

Και στα δύο παραπάνω dataset, εφαρμόσθηκε μία σειρά από διαδικασίες προεπεξεργασίας (data pre-processing) ώστε οι προβλέψεις και αποτελέσματα που θα προκύψουν από τα μοντέλα ταξινόμησης και συσταδοποίησης να είναι όσο το δυνατόν αμερόληπτα (unbiased).

Τα στάδια προεπεξεργασίας είναι τα παρακάτω:

1. Μετατροπή του κειμένου σε μικρά γράμματα (lower case).
2. Διαίρεση του κειμένου σε μεμονομένες λέξεις (tokenization).
3. Αφαίρεση των stop words στην Αγγλική γλώσσα.
4. Αφαίρεση των λέξεων με μήκος μικρότερο ή ίσο με 3.
5. Εφαρμογή του αλγορίθμου Porter Stemmer για να απομείνουν μοναδικές λέξεις στην ρίζα τους.

3.4 Συνεισφορά της Πτυχιακής

Μία βασική συνεισφορά της παρούσας πτυχιακής, έγκειται στον εμπλουτισμό της πηγής που περιγράφεται στην ενότητα 3.2 με ιεραρχικές ετικέτες (labels) όπως αποτυπώνονται στις ιστοσελίδες της Amazon (βλ. Εικόνα 3.2).



Σχήμα 3.2: Παράδειγμα προϊόντος με επισημασμένα τα labels

3.4.1 Συλλογή των labels και εμπλουτισμός του dataset

Η διαδικασία στηρίχθηκε σε τεχνικές μαζικού web scraping και ήταν ιδιαίτερα χρονοβόρα και απαιτητική όσον αφορά τις απαραίτητες υποδομές και υπηρεσίες αναζήτησης, συλλογής, αποθήκευσης και επεξεργασίας. Κατά τις διαδικασίες αυτές αντιμετωπίστηκαν αρκετά προβλήματα, τα οποία αντιμετωπίστηκαν με τις παρακάτω ενδεικτικές δράσεις και λύσεις:

- Συγγραφή κατάλληλου κώδικα (Python) προς εντοπισμό και συλλογή των labels.
- Εκτέλεση διαδικασιών στα πλαίσια της οριθής πρακτικής και δεοντολογίας ώστε να μην επιβαρυνθεί το σύστημα και οι υποδομές της Amazon⁴⁵.
- Αντιμετώπιση περιπτώσεων όπου το σύστημα αντιμετώπιζε τα http requests ως κακόβουλα (π.χ. Denial-of-Service κ.λπ.) και απαιτούσε επιβεβαίωση μέσω Captcha code.
- Περιορισμοί φυσικού εξοπλισμού από μέρους μας για την παράλληλη συλλογή των δεδομένων από περισσότερους Η/Υ.

⁴http://bit.ly/CoU_LaA - Amazon.com Legal Policies - Conditions of Use (License And Access)

⁵<http://bit.ly/amznApiLic> - Amazon.com Product Advertising API License Agreement - Usage req (p)

- Περιορισμοί Εικονικών Μηχανών (VMs) και σχετικών υπηρεσιών από την υποδομή Okeanos⁶ του ΕΔΕΤ.
- Προβληματικές εγγραφές στο dataset:
 - Υπάρξη εγγραφών όπου τα asin's οδηγούσαν σε HTTP 404 error οπότε και παραβλέφθηκαν. Επίσης, υπάρχει ένα αρκετά μεγάλο υποσύνολο του dataset το οποίο δεν είχε labels στην αντίστοιχη σελίδα των προϊόντων (βλ. Εικόνα 3.3).
 - Υπάρξη εγγραφών όπου τα asin's είχαν ως πρώτο χαρακτήρα το 0. Αυτό δημιούργησε πρόβλημα κατά την επεξεργασία του csv αρχείου από tabular-oriented tool (π.χ. Excel), αφού χανόταν η σωστή ταυτότητα του asin (leading zeros) με αποτέλεσμα να αλλοιώνεται η πληροφορία και να οδηγεί σε λανθασμένο url (HTTP 404 error).



Σχήμα 3.3: Παράδειγμα προϊόντος στο οποίο δεν έχουν καταχωριθεί labels.

Στην ενότητα Α΄.4, σελίδα 120 φαίνεται ο επισημασμένος και κατάλληλα σχολιασμένος κώδικας που υλοποιήθηκε για την συλλογή των δεδομένων. Επιλέχθηκε η γλώσσα προγραμματισμού Python και κατάλληλες βιβλιοθήκες για την επίτευξη του στόχου.

⁶<http://bit.ly/OkeanosProject> (Προαπαιτεί ενεργή σύνδεση στο Okeanos)

3.4.2 Ιεραρχική δομή των labels

Μία ενδεικτική εικόνα της ιεραρχικής δομής των συλλεχθέντων labels φαίνεται στην Εικόνα 3.4. Πιο συγκεκριμένα, έχει υλοποιηθεί μία τυπική ομαδοποίηση των root labels κάτιο το οποίο εφαρμόστηκε αναδρομικά σε κάθε επίπεδο ξεκινώντας από το πρώτο label και συνεχίζοντας μέχρι και το τελευταίο.

Η ιεραρχική δομή στο σύνολο του dataset φαίνεται εδώ [5].⁷

```

▼ Arts, Crafts & Sewing {7}
  ▼ Crafting {4}
    ▼ Craft Supplies {2}
      Craft Bells : null
    ▼ Cutting Tools {1}
      Scissors : null
  ▼ Paper & Paper Crafts {3}
    ▼ Embossing {1}
      Embossing Folders : null
  ▼ Paper {2}
    Decorative Paper : null
    Origami Paper : null
    Punches : null
  ▼ Sculpture Supplies {1}
    Molding & Casting : null
  ▼ Weaving & Spinning {3}
    Ball Winders : null
    Spinning Wheels : null
    Weaving Loom Tools & Accessories : null
  ▼ Knitting & Crochet {2}
    Knitting Kits : null
    Knitting Patterns : null

```

Σχήμα 3.4: Ιεραρχική δομή των labels.

Ενδεικτικά ακολουθεί είσοδος από labels, που ανήκουν σε 10 τυχαία προϊόντα, και η ιεραρχική αποτύπωση τους που προκύπτει από τον αλγόριθμο ιεραρχικής απόδοσης των labels, ενότητα Α'.2 σελ. 116:

Input:

1. ["Arts, Crafts & Sewing", "Crafting", "Craft Supplies", "Craft Bells"]

⁷<http://codebeautify.org/jsonviewer/cbdc82b5>

2. ["Arts, Crafts & Sewing", "Crafting", "Craft Supplies", "Cutting Tools", "Scissors"]
 3. ["Arts, Crafts & Sewing", "Crafting", "Paper & Paper Crafts", "Paper", "Decorative Paper"]
 4. ["Arts, Crafts & Sewing", "Crafting", "Paper & Paper Crafts", "Paper", "Origami Paper"]
 5. ["Arts, Crafts & Sewing", "Crafting", "Paper & Paper Crafts", "Punches"]
 6. ["Arts, Crafts & Sewing", "Crafting", "Sculpture Supplies", "Molding & Casting"]
 7. ["Arts, Crafts & Sewing", "Crafting", "Weaving & Spinning", "Ball Winders"]
 8. ["Arts, Crafts & Sewing", "Crafting", "Weaving & Spinning", "Spinning Wheels"]
 9. ["Arts, Crafts & Sewing", "Knitting & Crochet", "Knitting Kits"]
 10. ["Arts, Crafts & Sewing", "Knitting & Crochet", "Knitting Patterns"]

Output:

```
1 {
2     "root": {
3         "Arts, Crafts & Sewing": {
4             "Crafting": {
5                 "Craft Supplies": {
6                     "Craft Bells": null,
7                     "Cutting Tools": {
8                         "Scissors": null
9                     }
10                },
11                "Paper & Paper Crafts": {
12                    "Paper": {
13                        "Decorative Paper": null,
14                        "Origami Paper": null
15                    },
16                    "Punches": null
17                },
18                "Sculpture Supplies": {
19                    "Molding & Casting": null
20                },
21                "Weaving & Spinning": {
22                    "Ball Winders": null,
23                    "Spinning Wheels": null
24                }
25            }
26        }
27    }
28}
```

```

25     },
26     "Knitting & Crochet": {
27       "Knitting Kits": null,
28       "Knitting Patterns": null
29     }
30   }
31 }
32 }
```

3.4.3 Οδηγίες λήψης των labels και διαδικασίας εμπλουτισμού του αρχικού Dataset.

Ακολουθούν σύντομες οδηγίες περί του τρόπου με τον οποίο μπορεί να ανακτηθεί το εμπλουτισμένο dataset που περιγράφεται στην ενότητα 3.2.

1. Λήψη του αρχικού dataset από την ιστοσελίδα του Stanford University⁸ του οποίου το μέγεθος είναι περίπου 3.3 GB σε συμπιεσμένο αρχείο.
2. Λήψη του csv αρχείου όπου βρίσκονται τα δεδομένα των labels σε μονοσήμαντη αντιστοίχηση με το εκάστοτε ASIN. Η νέα αυτή εμπλουτισμένη πηγή δεδομένων φιλοξενείται ήδη στην συλλογή των Datasets της Kaggle.com⁹ και επίσης είναι διαθέσιμο προς αξιοποίηση στην κοινότητα, μέσω git repository στο GitHub.com [5]. Επίσης βρισκόμαστε σε διαδικασία ανάρτησης του στην επίσημη ιστοσελίδα του Stanford University και πιο συγκεκριμένα στην σελίδα της ερευνητικής ομάδας SNAP (Stanford Network Analysis Project).
3. Εκτέλεση του κώδικα που στο παράρτημα (Ενότητα A'.1, σελίδα 115), ο οποίος θα δημιουργήσει ένα νέο συμπιεσμένο αρχείο στο οποίο θα έχει προστεθεί η επιπλέον πληροφορία των labels σε κάθε instance όπως περιγράφεται παρακάτω.

Μετά την ολοκλήρωση των παραπάνω βημάτων, μπορεί να χρησιμοποιηθεί το αρχείο `output.txt.gz` το οποίο για να παραχθεί χρειάστηκαν περίπου 40 λεπτά. Η νέα δομή της κάθε εγγραφής θα έχει την παρακάτω μορφή (στην γραμμή No 9 η προσθήκη των labels):

1. `product/productId: B00006HAXW`
2. `review/userId: A1RSDE90N6RSZF`
3. `review/profileName: Joseph M. Kotow`
4. `review/helpfulness: 9/9`
5. `review/score: 5.0`
6. `review/time: 1042502400`

⁸<https://snap.stanford.edu/data/web-Movies.html>

⁹<https://www.kaggle.com/thebuzz/groud-truth-labels-amazon-movie-reviews-dataset>

7.review/summary: Pittsburgh - Home of the OLDIES

8.review/text: I have all of the doo wop DVD's and this one is as good or better than the 1st ones. Remember once these performers are gone, we'll never get to see them again. Rhino did an excellent job and if you like or love doo wop and Rock n Roll you'll LOVE this DVD!!

9.product/categories: ['CDs & Vinyl', 'Pop', 'Oldies', 'Doo Wop']

Κεφάλαιο 4

Αποτελέσματα – Μετρήσεις

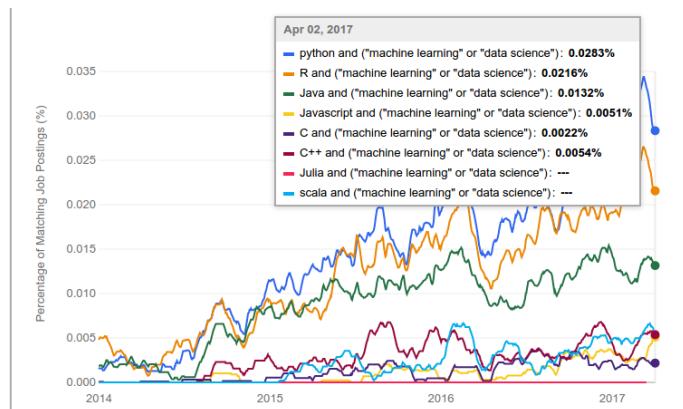
Στο κεφάλαιο αυτό περιγράφονται οι τεχνολογίες που χρησιμοποιήθηκαν, οι βιβλιοθήκες και η γλώσσα προγραμματισμού με την βοήθεια των οποίων ολοκληρώθηκε η υλοποίηση των πειραμάτων και των δοκιμών. Επίσης αναπτύχθηκαν οι μεθοδολογίες με τις οποίες αντιμετωπίστηκαν τα προβλήματα ταξινόμησης και συσταδοποίησης καθώς και τα αποτελέσματα που προέκυψαν. Το σύνολο του κώδικα βρίσκεται δημοσιευμένο σε Git Repository [6].

4.1 Πλατφόρμα – Βιβλιοθήκες

Για τις ανάγκες τις πτυχιακής αυτής, επιλέχθηκε ως γλώσσα προγραμματισμού η Python και για την μοντελοποίηση των προβλημάτων ταξινόμησης και συσταδοποίησης, η βιβλιοθήκη μηχανικής μάθησης Scikit-Learn. Το περιβάλλον δε, στο οποίο εφαρμόσθηκαν τα παραπάνω, είναι το IPython Notebook από το Jupyter Project.

4.1.1 Γλώσσα προγραμματισμού Python

Η Python είναι μια γενική, ευέλικτη και δημοφιλής γλώσσα. Αυτό έχει ως αποτέλεσμα να υπάρχουν διαθέσιμες πολλές βιβλιοθήκες με τις οποίες μπορεί να αντιμετωπισθεί μεγάλο εύρος προβλημάτων, από ανάπτυξη web περιεχόμενου μέχρι και επιστημονικές εφαρμογές.



Σχήμα 4.1: Διάγραμμα Γλωσσών προγραμματισμού στην προσφορά εργασίας¹

Μερικά πλεονεκτήματα της Python είναι τα εξής:

- Απλό συντακτικό.
- Εύκολη στην εκμάθηση.
- Δωρεάν και OpenSource.
- Γλώσσα Υψηλού Επιπέδου.
- Αντικειμενοστρεφής.
- Επεκτάσιμη.
- Μεγάλη κοινότητα προγραμματιστών.

Συγκεκριμένα, για τα προβλήματα που αντιμετωπίστηκαν στην πτυχιακή (Classification, Clustering, Text mining κ.λπ.), επιλέχθηκε μία σειρά από βιβλιοθήκες, οι κυριότερες των οποίων είναι οι παρακάτω:

- Scikit-learn: machine learning in Python
- Gensim: Topic modelling for humans²
- Natural Language Toolkit — NLTK ³
- Pandas ⁴
- NumPy ⁵
- BeautifulSoup ⁶
- Matplotlib: Python plotting ⁷
- Requests: HTTP for Humans ⁸

Παράλληλα δε, χρησιμοποιήθηκαν και άλλες βιβλιοθήκες βοηθητικού χαρακτήρα όπως για παράδειγμα stop-words ⁹ os ¹⁰, json ¹¹, csv ¹², pyLDAvis ¹³ κ.α.

¹<http://indeedhi.re/2qDoFR7>

²<https://radimrehurek.com/gensim/>

³<http://www.nltk.org>

⁴<https://pandas.pydata.org>

⁵<https://numpy.org>

⁶<https://www.crummy.com/software/BeautifulSoup/>

⁷<https://matplotlib.org>

⁸<https://docs.python-requests.org>

⁹<https://pypi.python.org/pypi/stop-words>

¹⁰<https://docs.python.org/3/library/os.html>

¹¹<https://docs.python.org/3/library/json.html>

¹²<https://docs.python.org/3/library/csv.html>

¹³<https://github.com/bmabey/pyLDAvis>

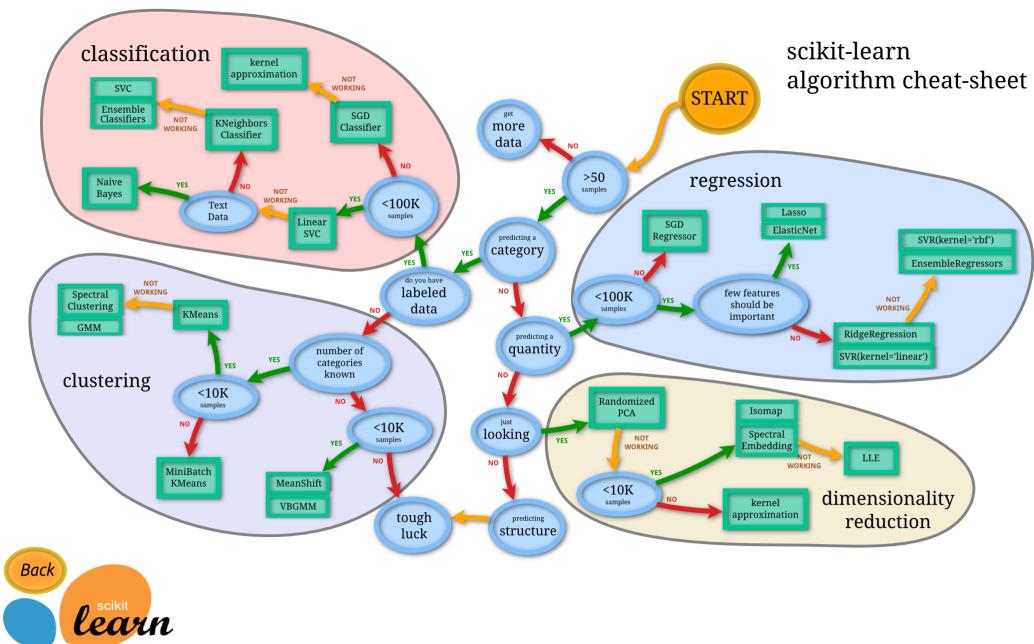
4.1.2 Βιβλιοθήκη Μηχανικής μάθησης Scikit-learn

Ιδιαίτερη μνεία θα πρέπει να γίνει για την βιβλιοθήκη scikit-learn¹⁴, η οποία περιλαμβάνει όλα τα απαραίτητα, και όχι μόνο, εργαλεία για την υλοποίηση μοντέλων μηχανικής μάθησης όπως:

- Classification.
- Regression.
- Clustering.
- Dimensionality reduction.
- Model selection.
- Preprocessing.

Άλλα πλεονεκτήματα της συγκεκριμένης βιβλιοθήκης είναι η απλότητα και πληρότητά της, η προσβασιμότητα και επαναχρησιμοποίηση σε ποικίλες εφαρμογές καθώς και ότι είναι βιβλιοθήκη ανοικτού κώδικα.

Ενδεικτικά, χρησιμοποιείται από εταιρείες/οργανισμούς όπως οι Spotify, INRIA (French Institute for Research in Computer Science and Control), Evernote, Booking.com κ.α.



Σχήμα 4.2: Χάρτης αλγορίθμων – μοντέλων της Scikit-learn

¹⁴<https://scikit-learn.org/>

4.1.3 IPython Notebook Environment

Το περιβάλλον στο οποίο έγιναν τα πειράματα και οι εφαρμογή των αλγορίθμων είναι το IPython Notebook (Jupyter project¹⁵) το οποίο είναι μια web εφαρμογή ανοικτού κώδικα στην οποία μπορεί να εκτελεσθεί Python κώδικας, να προβληθούν μαθηματικές σχέσεις, διαγράμματα κ.α. Οι πιο συνηθισμένες χρήσεις είναι για εφαρμογές καθαρισμού και μετασχηματισμού δεδομένων (data cleaning and transformation), στατιστική μοντελοποίηση (statistical modeling), καθώς και μηχανικής μάθησης (machine learning).

```
In [18]: import datetime
print('Hello there!')
print(datetime.date.today().strftime('Today is %d %b %Y'))
Hello there!
Today is 01 Jun 2017
```

Σχήμα 4.3: Απόσπασμα από το IPython Notebook

4.2 Προβλήματα Ταξινόμησης – Μεθοδολογία και αποτελέσματα

Παρακάτω, παρουσιάζεται όλη η διαδικασία αντιμετώπισης του προβλήματος της ταξινόμησης σε περιβάλλον IPython Notebook, η μεθοδολογία που ακολουθήθηκε καθώς και τα αποτελέσματα που προέκυψαν. Αρχικά, θα δούμε το dataset που περιγράφεται στην παράγραφο 3.1 (Six Categories of Amazon Product Reviews) και κατόπιν εκείνο της παραγράφου 3.2 (Amazon movie reviews dataset).

Το σύνολο του κώδικα βρίσκεται δημοσιευμένο σε Git Repository [6].

4.2.1 Ταξινόμηση στο Six Categories of Amazon Product Reviews Dataset

Ο κώδικας που υλοποιεί την παρακάτω ταξινόμηση βρίσκεται στο Ipython Notebook με όνομα chapter4_2_1.ipynb στο αποθετήριο [6].

Αρχικά, έγινε εισαγωγή όλων των απαραίτητων βιβλιοθηκών για την υλοποίηση της ταξινόμησης.

```
1 %matplotlib inline
2 import matplotlib.pyplot as plt
3 from __future__ import print_function
4 from sklearn.feature_extraction.text import CountVectorizer
5 from sklearn.feature_extraction.text import TfidfTransformer
6 from sklearn.cross_validation import train_test_split
7 from sklearn.metrics import classification_report, f1_score
```

¹⁵<http://jupyter.org/>

```

8 from sklearn.metrics import accuracy_score, confusion_matrix
9 from sklearn import metrics
10 from sklearn.naive_bayes import MultinomialNB
11 from sklearn.ensemble import RandomForestClassifier
12 import numpy as np
13 from sklearn.neighbors import KNeighborsClassifier
14 from sklearn.cross_validation import cross_val_score
15 from sklearn.pipeline import Pipeline
16 import pandas as pd

```

Στην συνέχεια, φορτώθηκε το csv αρχείο που περιέχει τα δεδομένα του dataset.

```

1 df = pd.read_csv('reviews.csv')
2 df.shape

```

Τα δεδομένα φορτωθήκαν σε ένα dataframe αντικείμενο της βιβλιοθήκης Pandas και με την μέθοδο `df.shape` μπορούμε να δούμε το πλήθος των instances και τον αριθμό των features που έχει. Στην προκειμένη περίπτωση έχουμε 27320 instances και 14 features. Εδώ θα πρέπει να αναφερθεί τα πειράματα υλοποιήθηκαν σε ένα υποσύνολο του αρχικού dataset διότι υπήρξαν προβλήματα όπως περιορισμένη μνήμη κλπ. Με την εντολή `df.head()` βλέπουμε τις πρώτες γραμμές του dataframe.

In [3]:	<code>df.head()</code>					
Out[3]:	raw_review	id	review	real_category	lda_catogories	lda_catogories_12
0	I love this phone so much! I had this phone f...	B006QMZCT0_0	[u'love', u'phone', u'much', u'phone', u'year'...]	mobilephone	[('cameras', 0.353), ('tablets', 0.169), ('lap...]	[('cameras', 0.101), ('tablets', 0.102), ('TVs'...]
1	This phone often just freezes or turns itself ...	B006QMZCT0_1	[u'phone', u'often', u'just', u'freez', u'turn...]	mobilephone	[('cameras', 0.104), ('tablets', 0.039), ('lap...]	[('cameras', 0.01), ('tablets', 0.004), ('TVs'...]

Σχήμα 4.4: Αποτέλεσμα της εντολής `df.head()`

Μερικά στοιχεία των δεδομένων που έχουμε στο dataframe βλέπουμε παρακάτω με την εντολή `df.info()`

```

1 <class 'pandas.core.frame.DataFrame'>
2 RangeIndex: 27320 entries, 0 to 27319
3 Data columns (total 14 columns):
4 raw_review      27320 non-null object
5 id              27320 non-null object
6 review          27320 non-null object

```

```

7 real_category      27320 non-null object
8 lda_catogories    27320 non-null object
9 ....
10 dtypes: object(14)
11 memory usage: 2.9+ MB

```

Οι παραπάνω στήλες με τίτλο lda_catogories χρατάνε δεδομένα από προηγούμενα πειράματα που θα αναλυθούν στην ενότητα 4.3.1, σελίδα 74.

Στην συνέχεια θα εφαρμοσθεί μία επιλογή από τα απαραίτητα features για την δημιουργία ενός νέου dataframe, το οποίο θα χρησιμοποιηθεί και για την υλοποίηση των πειραμάτων. Επιλέγονται μόνο οι στήλες raw_review που ουσιαστικά είναι το κείμενο του review και το real_category που είναι η κατηγορία στην οποία ανήκει το προϊόν.

```

1 reviews_df = df[['raw_review', 'real_category']].copy()
2 reviews_df.head(5)

```

In [6]: # examine the first and last 5 rows
reviews_df.head(5)

Out[6]:

	raw_review	real_category
0	I love this phone so much! I had this phone f...	mobilephone
1	This phone often just freezes or turns itself ...	mobilephone
2	What I would have like to know before I purcha...	mobilephone
3	I bought two of these phones for my kids. Bot...	mobilephone
4	Liked the phone, but after a while I started h...	mobilephone

Σχήμα 4.5: Αποτέλεσμα της εντολής df.head(5)

Στην συνέχεια, για να έχουμε μία εικόνα σχετικά με το πόσο μακροσκελή είναι τα reviews του dataset, θα προσθέσουμε μία επιπλέον στήλη με τίτλο length η οποία θα περιέχει το μήκος του κάθε review.

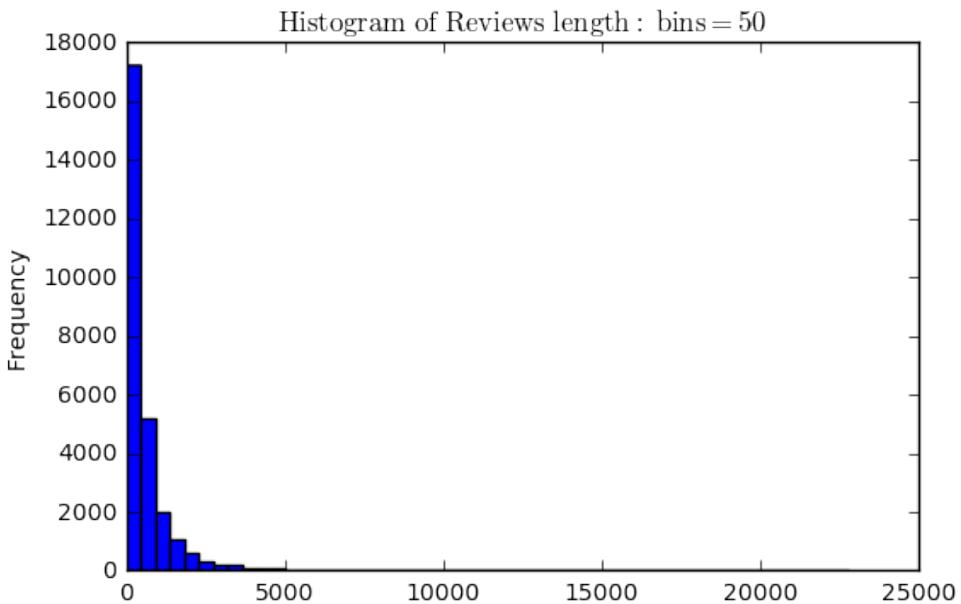
```
In [8]: reviews_df['length'] = reviews_df['raw_review'].map(lambda text: len(text))
reviews_df.head()
```

Out[8]:

	raw_review	real_category	length
0	I love this phone so much! I had this phone f...	mobilephone	1081
1	This phone often just freezes or turns itself ...	mobilephone	281
2	What I would have like to know before I purcha...	mobilephone	936
3	I bought two of these phones for my kids. Bot...	mobilephone	126
4	Liked the phone, but after a while I started h...	mobilephone	416

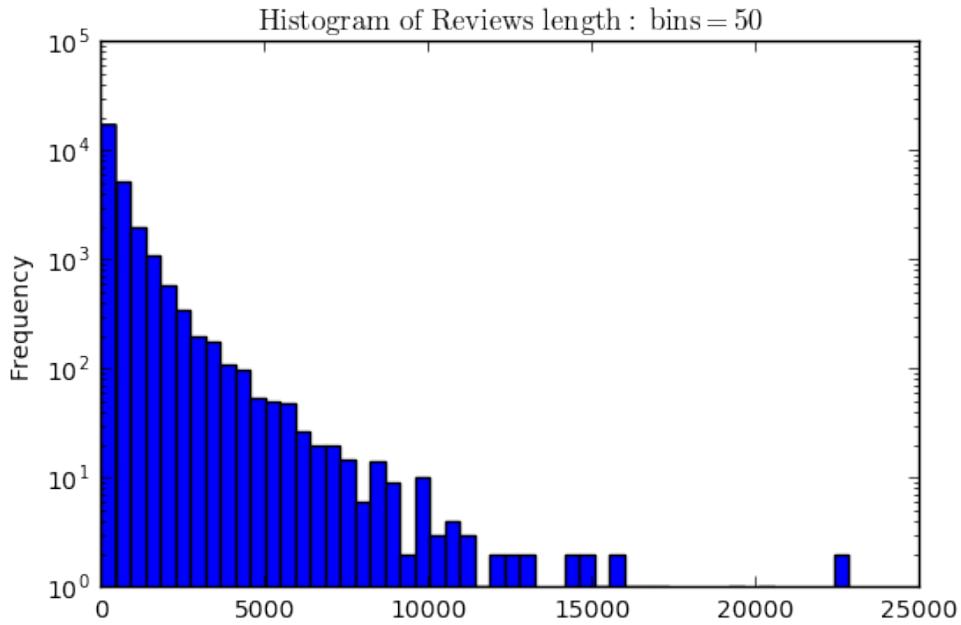
Σχήμα 4.6: Προσθήκη στήλης με το μήκος του κάθε review

Από το μήκος των reviews έχει παραχθεί το παρακάτω ιστόγραμμα της εικόνας 4.7. Βλέπουμε ότι λίγα από αυτά είναι ιδιαιτέρως μακροσκελή (και άρα λεπτομερή), ενώ στην πλειοψηφία τους τα reviews είναι μικρά σε περιεχόμενο.

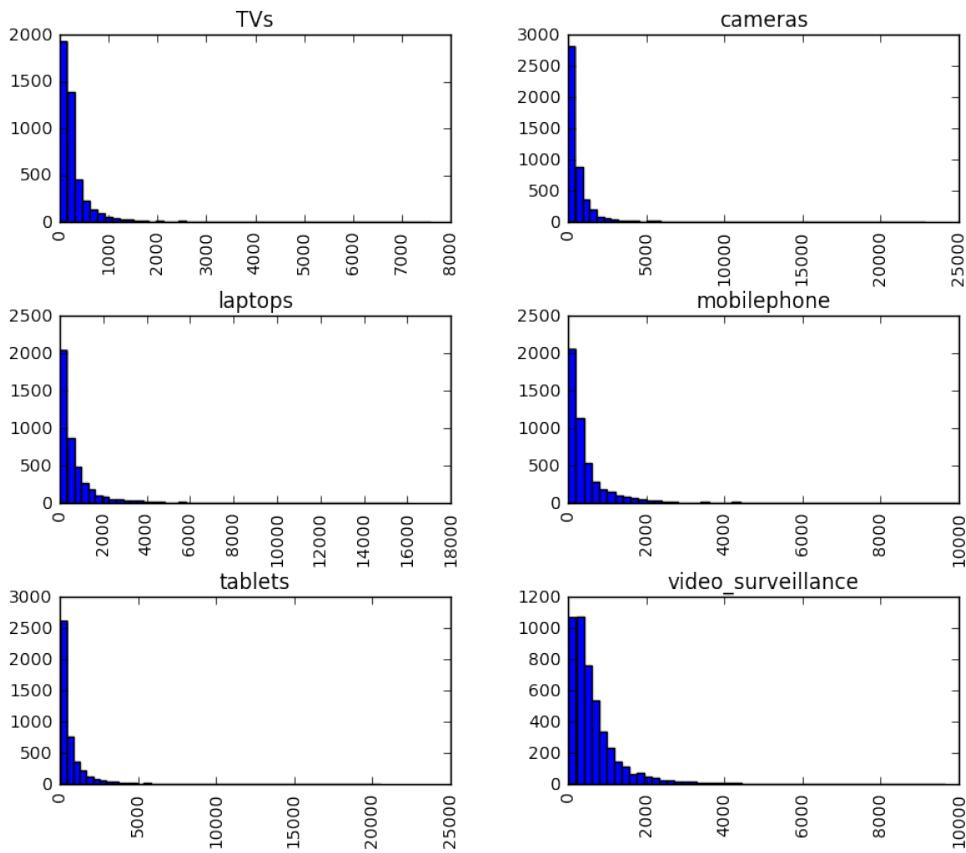


Σχήμα 4.7: Ιστόγραμμα βασισμένο στο μήκος των reviews

Στο παρακάτω ιστόγραμμα (Εικόνα 4.8), έχουμε λογαριθμήσει τον άξονα για να έχουμε καλύτερη εικόνα ως προς την συχνότητα στο μήκος των reviews.



Σχήμα 4.8: Ιστόγραμμα βασισμένο στο μήκος των reviews με λογαριθμημένο τον άξονα για την Ακολουθεί το αντίστοιχο ιστόγραμμα ανά κατηγορία προϊόντων:



Σχήμα 4.9: Ιστόγραμμα μήκους review ανά κατηγορία προϊόντος

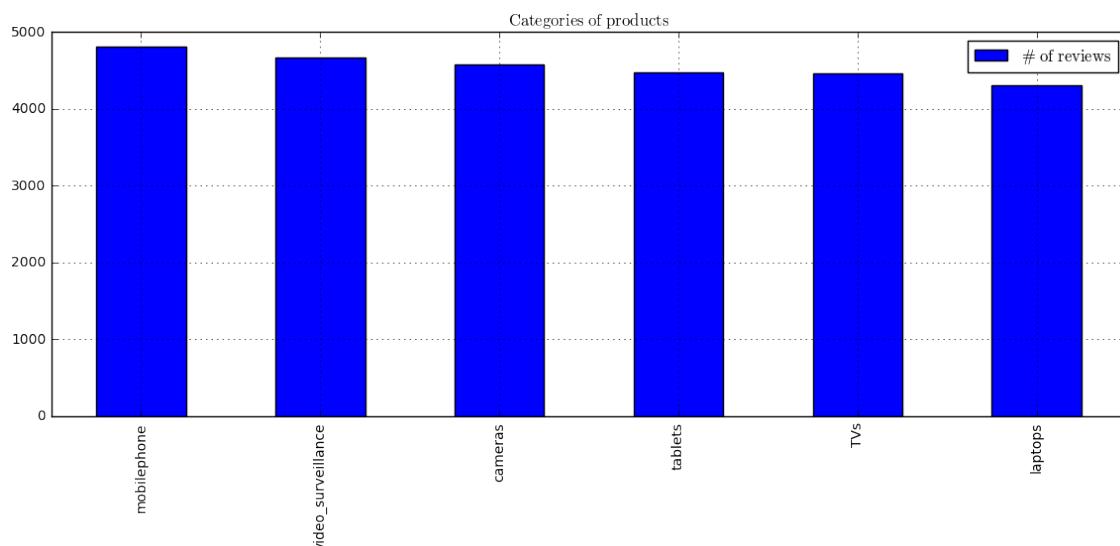
Από τα παραπάνω ιστογράμματα, θα μπορούσαμε να συμπεράνουμε ότι οι πιο περιγραφικοί και λεπτομερής χρήστες είναι αυτοί που υπέβαλαν reviews για προϊόντα της κατηγορίας video surveillance.

Συνεχίζοντας την ανάλυση των reviews και εκτελώντας την παρακάτω εντολή επιστρέφεται ο παρακάτω πίνακας με στατιστικά στοιχεία σχετικά με το μήκος των reviews:

```
1 reviews_df.length.describe()
```

```
1 count      27320.000000
2 mean       628.336896
3 std        995.436464
4 min        1.000000
5 25%       155.000000
6 50%       304.000000
7 75%       683.250000
8 max       22828.000000
9 Name: length, dtype: float64
```

Στο παρακάτω διάγραμμα (εικόνα 4.10) φαίνεται το πλήθος των reviews ανά κατηγορία (label) προϊόντων:



Σχήμα 4.10: Πλήθος των reviews ανά κατηγορία προϊόντος

Στην συνέχεια, για να μπορέσουμε να υλοποιήσουμε τα μοντέλα ταξινόμησης θα πρέπει να αντιστοιχίσουμε τις τιμές της στήλης real_category με μία αριθμητική τιμή. Η αντιστοίχηση υλοποιείται με την εισαγωγή μίας νέας στήλης με όνομα real_category_num. Έτσι εκτελώντας τον παρακάτω κώδικα, το dataframme στο οποίο θα εφαρμόσουμε τα πειράματα θα έχει την παρακάτω μορφή (Εικόνα 4.11):

```

1 # convert label to a numerical variable
2 d = {'mobilephone':0, 'cameras':1, 'video_surveillance':2, 'TVs':3,
3 'tablets':4, 'laptops':5}
4 reviews_df['real_category_num'] = reviews_df.real_category.map(d)

```

Οπότε, το dataframe στο οποίο θα εφαρμόσουμε τα πειράματα θα έχει την παρακάτω μορφή:

In [13]:	reviews_df.head()																														
Out[13]:	<table border="1"> <thead> <tr> <th></th> <th>raw_review</th> <th>real_category</th> <th>length</th> <th>real_category_num</th> </tr> </thead> <tbody> <tr><td>0</td><td>I love this phone so much! I had this phone f...</td><td>mobilephone</td><td>1081</td><td>0</td></tr> <tr><td>1</td><td>This phone often just freezes or turns itself ...</td><td>mobilephone</td><td>281</td><td>0</td></tr> <tr><td>2</td><td>What I would have like to know before I purcha...</td><td>mobilephone</td><td>936</td><td>0</td></tr> <tr><td>3</td><td>I bought two of these phones for my kids. Bot...</td><td>mobilephone</td><td>126</td><td>0</td></tr> <tr><td>4</td><td>Liked the phone, but after a while I started h...</td><td>mobilephone</td><td>416</td><td>0</td></tr> </tbody> </table>		raw_review	real_category	length	real_category_num	0	I love this phone so much! I had this phone f...	mobilephone	1081	0	1	This phone often just freezes or turns itself ...	mobilephone	281	0	2	What I would have like to know before I purcha...	mobilephone	936	0	3	I bought two of these phones for my kids. Bot...	mobilephone	126	0	4	Liked the phone, but after a while I started h...	mobilephone	416	0
	raw_review	real_category	length	real_category_num																											
0	I love this phone so much! I had this phone f...	mobilephone	1081	0																											
1	This phone often just freezes or turns itself ...	mobilephone	281	0																											
2	What I would have like to know before I purcha...	mobilephone	936	0																											
3	I bought two of these phones for my kids. Bot...	mobilephone	126	0																											
4	Liked the phone, but after a while I started h...	mobilephone	416	0																											
In [120]:	reviews_df.tail()																														
Out[120]:	<table border="1"> <thead> <tr> <th></th> <th>raw_review</th> <th>real_category</th> <th>length</th> <th>real_category_num</th> </tr> </thead> <tbody> <tr><td>27315</td><td>While the Chromebook is quick, light, portable...</td><td>laptops</td><td>689</td><td>5</td></tr> <tr><td>27316</td><td>I purchased this for my 11 year old grandson f...</td><td>laptops</td><td>1125</td><td>5</td></tr> <tr><td>27317</td><td>i did not realize the operating system would n...</td><td>laptops</td><td>519</td><td>5</td></tr> <tr><td>27318</td><td>This is the worst laptop I've ever seen in my ...</td><td>laptops</td><td>123</td><td>5</td></tr> <tr><td>27319</td><td>It does not do everything I need it to. I wil...</td><td>laptops</td><td>132</td><td>5</td></tr> </tbody> </table>		raw_review	real_category	length	real_category_num	27315	While the Chromebook is quick, light, portable...	laptops	689	5	27316	I purchased this for my 11 year old grandson f...	laptops	1125	5	27317	i did not realize the operating system would n...	laptops	519	5	27318	This is the worst laptop I've ever seen in my ...	laptops	123	5	27319	It does not do everything I need it to. I wil...	laptops	132	5
	raw_review	real_category	length	real_category_num																											
27315	While the Chromebook is quick, light, portable...	laptops	689	5																											
27316	I purchased this for my 11 year old grandson f...	laptops	1125	5																											
27317	i did not realize the operating system would n...	laptops	519	5																											
27318	This is the worst laptop I've ever seen in my ...	laptops	123	5																											
27319	It does not do everything I need it to. I wil...	laptops	132	5																											

Σχήμα 4.11: Πέντε πρώτα και πέντε τελευταία instances των reviews

Πλέον, μετά την σύντομη επισκόπηση των δεδομένων μας, είμαστε σε θέση να απομονώσουμε τις στήλες raw_review και real_category_num, από τις οποίες η μεν πρώτη θα είναι αυτή που θα εκπαιδεύσει τα μοντέλα μας, η δε δεύτερη θα είναι η πληροφορία για το που πραγματικά ανήκει το κάθε review.

```

1 # how to define X and y for use with COUNTVECTORIZER
2 X = reviews_df.raw_review
3 y = reviews_df.real_category_num

```

Έπειτα, χωρίζουμε τα νέα μας δεδομένα, σε δύο υποσύνολα (X με 21,856 εγγραφές και y με 5,464) και σε αναλογία 80% - 20% ώστε να χρησιμοποιηθεί το 80% για την εκπαίδευση (training set) των μοντέλων και το υπόλοιπο 20% για την αξιολόγηση (testing set).

```
In [16]: # split X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=0)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

(21856,)
(5464,)
(21856,)
(5464,)
```

Σχήμα 4.12: Training set και testing set

Συνεχίζουμε στην μετατροπή των reviews από μορφή κειμένου σε διανυσματικό πίνακα με στήλες τις λέξεις που προέκυψαν ως οι πιο “σημαντικές” από την κλάση CountVectorizer(), γραμμές τα reviews και με δεδομένα στα κελιά την συχνότητα της κάθε λέξης που αντιστοιχεί στην στήλη. Μάλιστα, προς εξοικονόμηση μνήμης, αποθηκεύονται μόνο οι συντεταγμένες του πίνακα που περιέχουν τιμή μεγαλύτερη του 0 (Sparse matrix).

```
In [17]: # instantiate CountVectorizer
vect = CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))

In [18]: # learn training data vocabulary, then use it to create a document-term matrix
X_train_dtm = vect.fit_transform(X_train)

In [19]: # examine the document-term matrix
X_train_dtm

Out[19]: <21856x334982 sparse matrix of type '<type 'numpy.int64'>'  
with 1951975 stored elements in Compressed Sparse Row format>

In [20]: # transform testing data (using fitted vocabulary) into a document-term matrix
X_test_dtm = vect.transform(X_test)
X_test_dtm

Out[20]: <5464x334982 sparse matrix of type '<type 'numpy.int64'>'  
with 367097 stored elements in Compressed Sparse Row format>
```

Σχήμα 4.13: Δημιουργία Document-term Matrix

Βλέπουμε ότι ο διανυσματικός πίνακας που προέκυψε (Document-term Matrix¹⁶) αποτελείται από 334,982 στήλες.

Στην συνέχεια, εξετάζονται με την παρακάτω σειρά οι εξής αλγόριθμοι ταξινόμησης:

- Naive Bayes
- Random Forest
- Logistic Regression
- K-nearest neighbors
- Support vector machine

Για κάθε έναν από αυτούς, θα εκτελεσθεί μία σειρά από βήματα μέσω των οποίων θα ολοκληρώνεται κάθε φορά η υλοποίηση του μοντέλου ταξινόμησης. Τα βήματα είναι τα παρακάτω:

¹⁶https://en.wikipedia.org/wiki/Document-term_matrix

1. Αρχικοποίηση μοντέλου.
2. Εκπαίδευση με τα δεδομένα X_train_dtm και y_train.
3. Διαδικασία πρόβλεψης με είσοδο το X_test_dtm.
4. Υπολογισμός ακρίβειας των προβλέψεων του προηγούμενου βήματος.
5. Υπολογισμός και εμφάνιση τού πίνακα σύγχυσης.
6. Επιβεβαίωση ότι τα αποτελέσματα μας δεν είναι biased βάσει k-Fold Cross Validation, με $k = 10$.

Naive Bayes

Στον παρακάτω κώδικα εκτελούνται τα πρώτα 3 βήματα:

```

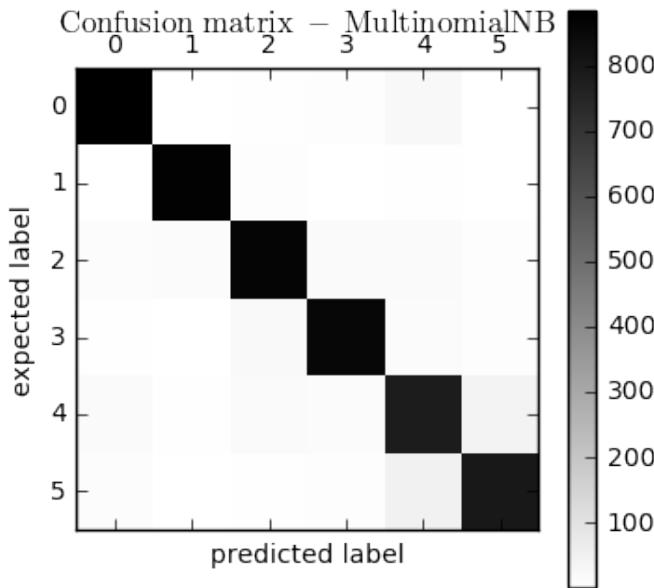
1 # 1. instantiate a Multinomial Naive Bayes model
2 nb = MultinomialNB()
3 # 2. train the model using X_train_dtm
4 nb.fit(X_train_dtm, y_train)
5 # 3. make class predictions for X_test_dtm
6 y_pred_class = nb.predict(X_test_dtm)

```

Βλέπουμε ότι το ποσοστό ακρίβειας προέκυψε στα 91.69%. Πιο κάτω φαίνεται και ο πίνακας σύγχυσης (Εικόνα 4.14) με τις τιμές αλλά και σε γράφημα (Εικόνα 4.15).

In [25]:	# calculate accuracy of class predictions res = metrics.accuracy_score(y_test, y_pred_class) models['NB'].append(res) print("Accuracy: %0.2f %%" % (res * 100))
	Accuracy: 91.69 %
In [26]:	# print the confusion matrix metrics.confusion_matrix(y_test, y_pred_class)
Out[26]:	array([[852, 3, 20, 3, 48, 5], [2, 843, 29, 1, 16, 2], [5, 2, 915, 4, 12, 6], [4, 1, 56, 803, 39, 11], [19, 1, 29, 3, 827, 19], [5, 3, 8, 2, 96, 770]])

Σχήμα 4.14: Απόδοση ακρίβειας Naive Bayes



Σχήμα 4.15: Πίνακας σύγχυσης Naive Bayes

Παρακάτω φαίνεται και το τελευταίο βήμα, όπου και στην εφαρμογή του 10-Fold Cross Validation η απόδοση των επιτυχών προβλέψεων είναι σχεδόν στα ίδια επίπεδα (90.24%) με την training/testing μέθοδο.

```
In [142]: pipeline = Pipeline([
    # strings to token integer counts
    ('bow', CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))),
    # integer counts to weighted TF-IDF scores
    ('tfidf', TfidfTransformer()),
    # train on TF-IDF vectors w/ Naive Bayes classifier
    ('classifier', nb),
])

In [30]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
                               X, # training data
                               y, # training labels
                               cv=10, # split data randomly into 10 parts: 9 for training, 1 for scoring
                               scoring='accuracy', # which scoring metric?
                               n_jobs=-1, # -1 = use all cores = faster
)
print(scores)

[ 0.88925439  0.93199269  0.93448023  0.90483163  0.91764275  0.92347126
  0.93994874  0.89967045  0.86556777  0.81721612]

In [31]: models['NB'].append(scores.mean())
print("Accuracy: %.2f %% (+/- %.2f %%)" % (scores.mean() * 100, scores.std() * 2 * 100))

Accuracy: 90.24 % (+/- 7.15 %)
```

Σχήμα 4.16: Απόδοση ακρίβειας 10-Fold Cross Validation του Naive Bayes

Logistic regression

Στον παρακάτω κώδικα εκτελούνται τα πρώτα 3 βήματα:

```
1 # 1. instantiate a logistic regression model
2 logreg = LogisticRegression()
```

```

3 # 2. train the model using X_train_dtm
4 logreg.fit(X_train_dtm, y_train)
5 # 3. make class predictions for X_test_dtm
6 y_pred_class = logreg.predict(X_test_dtm)

```

Βλέπουμε πως κάτω ότι το ποσοστό ακριβείας προέκυψε στα 93.89%. Κατ' αντιστοιχία με τον παραπάνω αλγόριθμο, φαίνεται και ο αντίστοιχος πίνακας σύγχυσης με τις τιμές αλλά και σε γράφημα.

```

In [35]: # calculate accuracy
res = metrics.accuracy_score(y_test, y_pred_class)
models['LR'].append(res)
print("Accuracy: %.2f %%" % (res * 100))

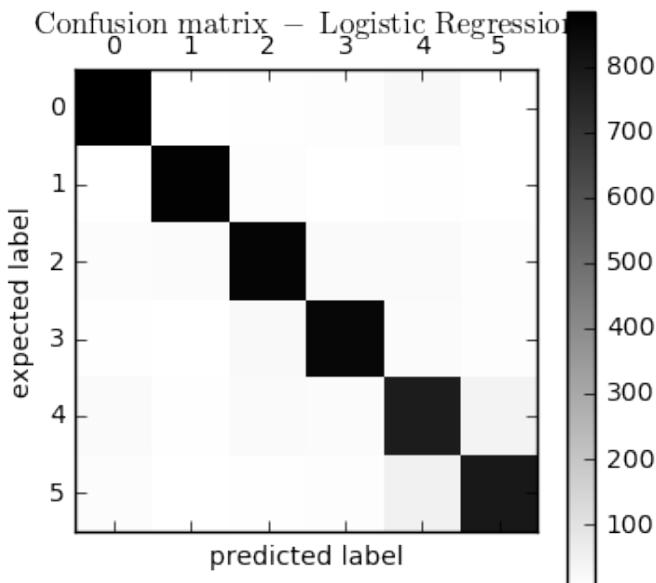
Accuracy: 93.89 %

In [36]: # print the confusion matrix
metrics.confusion_matrix(y_test, y_pred_class)

Out[36]: array([[892,    5,    4,    5,   22,    3],
   [    3, 876,    7,    1,    3,    3],
   [ 10,    9, 873,   25,   17,   10],
   [  2,    2,   19, 873,   11,    7],
   [ 15,    8,   15,   15, 802,   43],
   [  6,    3,    5,    4,   52, 814]])

```

Σχήμα 4.17: Απόδοση ακριβείας Logistic regression



Σχήμα 4.18: Πίνακας σύγχυσης Logistic regression

Παρακάτω φαίνεται και το τελευταίο βήμα, όπου και στην εφαρμογή του 10-Fold Cross Validation η απόδοση των επιτυχών προβλέψεων είναι σχεδόν στα ίδια επίπεδα (92.7%) με την training/testing μέθοδο.

```
In [39]: pipeline = Pipeline([
    ('bow', CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))),
    ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores
    ('classifier', logreg), # train on TF-IDF vectors w/ Logistic Regression classifier
])
<ipython-input-39-1234567890>
```

```
In [40]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
                               X, # training data
                               y, # training labels
                               cv=10, # split data randomly into 10 parts: 9 for training, 1 for scoring=accuracy, # which scoring metric?
                               n_jobs=-1, # -1 = use all cores = faster
)
print(scores)
```

```
[ 0.93567251  0.94771481  0.94070278  0.92606149  0.95204978  0.94653973
  0.95862321  0.92859758  0.89047619  0.84322344]
```

```
In [41]: models['LR'].append(scores.mean())
print("Accuracy: %0.2f %% (+/- %0.2f %%)" % (scores.mean() * 100, scores.std() * 2 * 100))
```

```
Accuracy: 92.70 % (+/- 6.65 %)
```

Σχήμα 4.19: Απόδοση ακρίβειας 10-Fold Cross Validation του Logistic regression

Random Forest

Στον παρακάτω κώδικα εκτελούνται τα πρώτα 3 βήματα:

```
1 # 1. instantiate a Random Forest model
2 rf = RandomForestClassifier()
3 # 2. train the model using X_train_dtm
4 rf.fit(X_train_dtm, y_train)
5 # 3. make class predictions for X_test_dtm
6 y_pred_class = rf.predict(X_test_dtm)
```

Βλέπουμε πως κάτω ότι το ποσοστό ακρίβειας προέκυψε στα 86.84% (βλ. Εικόνα 4.20 και Εικόνα 4.21 αντίστοιχα).

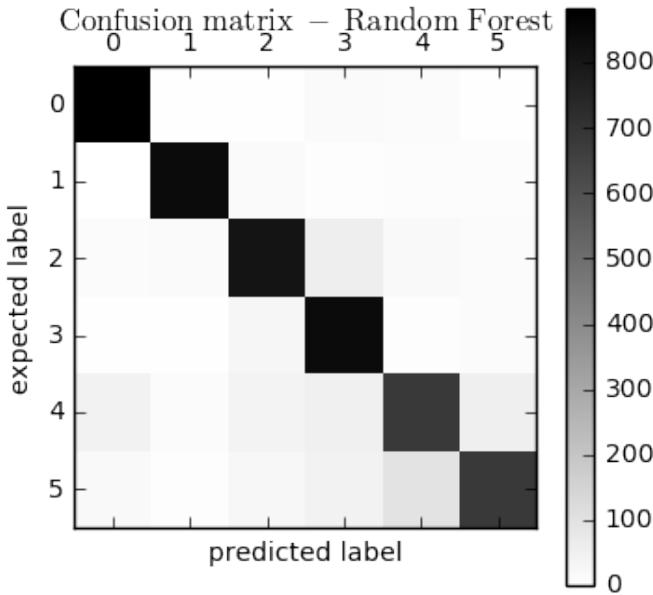
```
In [162]: # calculate accuracy
res = metrics.accuracy_score(y_test, y_pred_class)
models['RF'].append(res)
print("Accuracy: %0.2f %%" % (res * 100))

Accuracy: 86.84 %

In [163]: # print the confusion matrix
metrics.confusion_matrix(y_test, y_pred_class)

Out[163]: array([[883,    4,    6,   18,   14,    6],
                  [  0, 843,   18,    8,   12,   12],
                  [ 16,   18, 813,   61,   22,   14],
                  [  5,    6,   34, 843,    9,   17],
                  [ 46,   16,   44,   55, 681,   56],
                  [ 21,   10,   28,   46,   97, 682]])
```

Σχήμα 4.20: Απόδοση ακρίβειας Random Forest



Σχήμα 4.21: Πίνακας σύγχυσης Random Forest

Παρακάτω φαίνεται και το τελευταίο βήμα, όπου και στην εφαρμογή του 10-Fold Cross Validation η απόδοση των επιτυχών προβλέψεων είναι σχεδόν στα ίδια επίπεδα (86.21%) με την training/testing μέθοδο.

```
In [173]: pipeline = Pipeline([
    ('bow', CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))),
    ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores
    ('classifier', rf), # train on TF-IDF vectors w/ Random Forest classifier
])

```

```
In [174]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
                                X, # training data
                                y, # training labels
                                cv=10, # split data randomly into 10 parts: 9 for training, 1 for sc
                                scoring='accuracy', # which scoring metric?
                                n_jobs=-1, # -1 = use all cores = faster
)
print(scores)

[ 0.85818713  0.90310786  0.8817716   0.84590044  0.88396779  0.87879897
 0.91468327  0.86634932  0.84542125  0.74285714]
```

```
In [175]: models['RF'].append(scores.mean())
print("Accuracy: %0.2f %% (+/- %0.2f %%)" % (scores.mean() * 100, scores.std() * 2 * 100))

Accuracy: 86.21 % (+/- 9.03 %)
```

Σχήμα 4.22: Απόδοση ακρίβειας 10-Fold Cross Validation του Random Forest

K-nearest neighbors

Σε αυτό το μοντέλο πρέπει να αποφασιστεί η τιμή που θα δώσουμε στο k . Το k αντιπροσωπεύει το πλήθος των γειτόνων που θα εξετασθούν για να ληφθεί η απόφαση της ταξινόμησης για το εξεταζόμενο review. Θα δοκιμάσουμε την απόδοση του μοντέλου με τιμή του k από 1

έως και 25 και όταν επιλέξουμε αυτή που θα μας επιστρέψει το μέγιστο ποσοστό επιτυχίας.

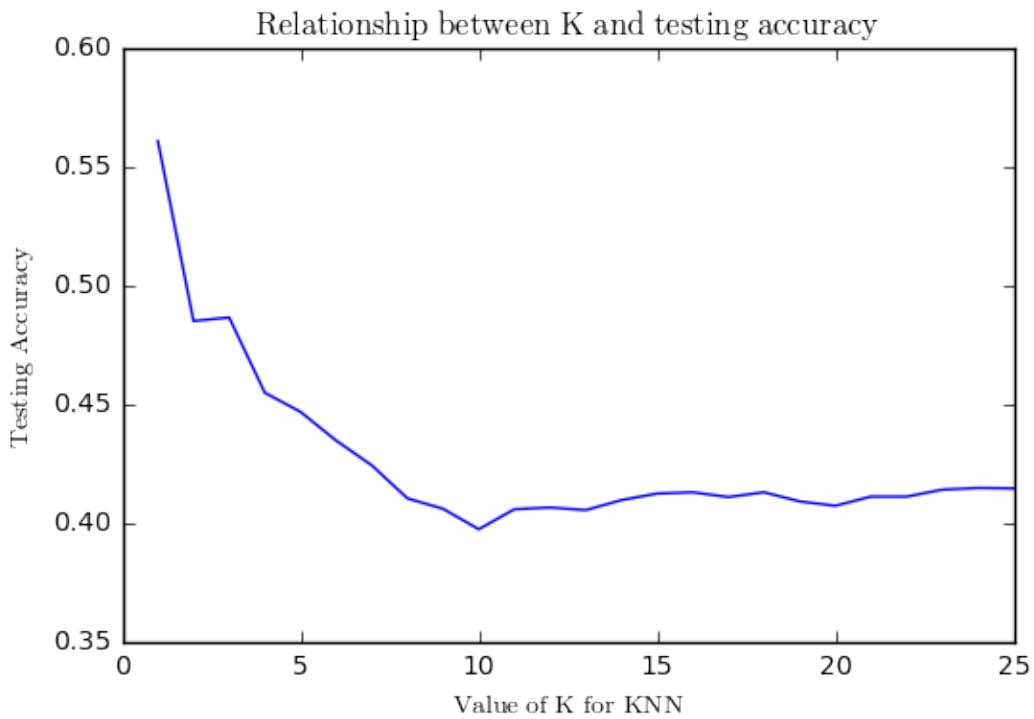
Αυτό επιτυγχάνεται με τον παρακάτω κώδικα:

```

1 # try K=1 through K=25 and record testing accuracy
2 k_range = list(range(1, 26))
3 knnscores = []
4 for k in k_range:
5     knn = KNeighborsClassifier(n_neighbors=k)
6     knn.fit(X_train_dtm, y_train)
7     y_pred = knn.predict(X_test_dtm)
8     knnscores.append(metrics.accuracy_score(y_test, y_pred))

```

Το γράφημα που προκύπτει από τις τιμές του k και των ποσοστών ακριβείας αποτυπώνεται στην Εικόνα 4.23.



Σχήμα 4.23: Απόδοση ακριβείας του K-nn για τιμές του k από 1 έως 25

Παρατηρούμε ότι η υψηλότερη επίδοση προκύπτει όταν το $k = 1$ και όσο αυτό αυξάνεται, η απόδοση λαμβάνει χαμηλότερες τιμές.

Στον παρακάτω κώδικα εκτελούνται τα πρώτα τέσσερα βήματα καθώς επίσης και ο υπολογισμός του πίνακα σύγχυσης:

```
In [51]: # try K=1, record testing accuracy
# 1. instantiate a KNeighborsClassifier model
knn = KNeighborsClassifier(n_neighbors=1)
# 2. train the model using X_train_dtm
knn.fit(X_train_dtm, y_train)
# 3. make class predictions for X_test_dtm
y_pred_class = knn.predict(X_test_dtm)
# 4. Accuracy calculation
res = metrics.accuracy_score(y_test, y_pred_class)
models['KNN'].append(res)
print("Accuracy: {:.2f} %".format(res * 100))

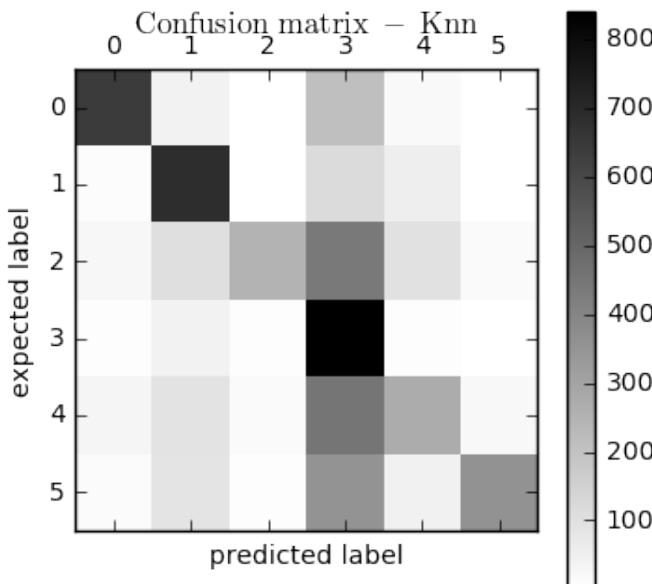
Accuracy: 56.06 %
```

```
In [52]: # print the confusion matrix
metrics.confusion_matrix(y_test, y_pred_class)
```

```
Out[52]: array([[647,   46,    1, 212,   23,    2],
       [ 13, 692,    3, 122,   59,    4],
       [ 28, 107, 250, 440, 101,   18],
       [  9,   44,    8, 841,   10,    2],
       [ 34,   93,   18, 455, 274,   24],
       [ 16,   92,   10, 358,   49, 359]])
```

Σχήμα 4.24: Απόδοση ακρίβειας του K-nn

Βλέπουμε πως πάνω ότι το ποσοστό ακρίβειας προέκυψε στα 56.06%. Ακολουθεί το γράφημα του πίνακα σύγχυσης (Εικόνα 4.25):



Σχήμα 4.25: Πίνακας σύγχυσης του K-nn

Παρακάτω, φαίνεται και το τελευταίο βήμα, όπου και στην εφαρμογή του 10-Fold Cross

Validation, όπου (όπως και για τους άλλους αλγορίθμους) η απόδοση των επιτυχών προβλέψεων είναι σχεδόν στα ίδια επίπεδα (48.84%) με την training/testing μέθοδο.

```
In [55]: pipeline = Pipeline([
    ('bow', CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))),
    ('tfidf', TfidfTransformer()),
    ('classifier', knn), # train on TF-IDF vectors w/ Knn classifier
])

In [56]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
                                 X, # training data
                                 y, # training labels
                                 cv=10, # split data randomly into 10 parts: 9 for training, 1 for sc
                                 scoring='accuracy', # which scoring metric?
                                 n_jobs=-1, # -1 = use all cores = faster
)
print(scores)

[ 0.56359649  0.54442413  0.54245974  0.41691069  0.79099561  0.4544123
  0.54522153  0.39069938  0.34615385  0.28937729]

In [57]: models['KNN'].append(scores.mean())
print("Accuracy: %0.2f %% (%+- %0.2f %%)" % (scores.mean() * 100, scores.std() * 2 * 100))

Accuracy: 48.84 % (+/- 26.99 %)
```

Σχήμα 4.26: Απόδοση ακρίβειας 10-Fold Cross Validation του K-nn

Support vector machine

Στον παρακάτω κώδικα εκτελούνται τα πρώτα 3 βήματα:

```
1 # 1. instantiate a Support vector machine (SVM) model
2 svm_clf = svm.SVC(kernel='linear')
3 # 2. train the model using X_train_dtm
4 svm_clf.fit(X_train_dtm, y_train)
5 # 3. make class predictions for X_test_dtm
6 y_pred_class = svm_clf.predict(X_test_dtm)
```

Βλέπουμε πως κάτω ότι το ποσοστό ακρίβειας προέκυψε στα 92.75% (βλ. εικόνα 4.27). Τιδια συμπεράσματα με τους προηγούμενους αλγορίθμους συνάγονται και για την εφαρμογή του 10-Fold Cross Validation (93.42%) (βλ. εικόνα 4.29).

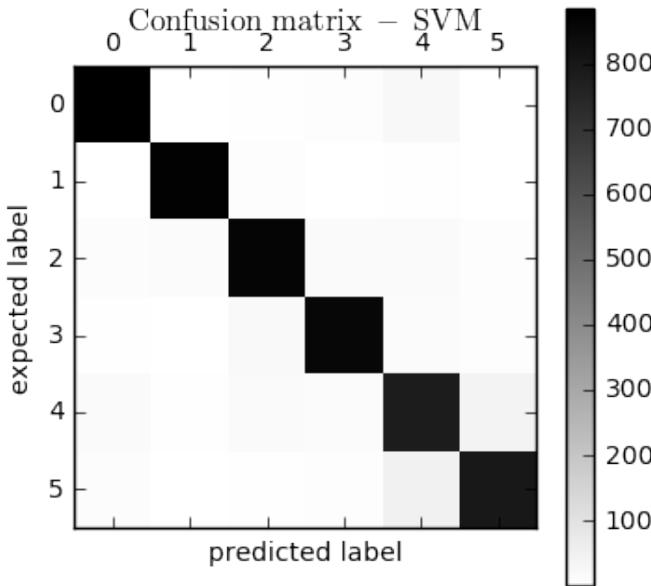
```
In [60]: # calculate accuracy
res = metrics.accuracy_score(y_test, y_pred_class)
models['SVM'].append(res)
print("Accuracy: %0.2f %%" % (res * 100))

Accuracy: 92.75 %

In [61]: # print the confusion matrix
metrics.confusion_matrix(y_test, y_pred_class)

Out[61]: array([[885,    1,    6,   11,   27,    1],
               [  2, 872,    9,    3,    5,   2],
               [ 13,   16, 865,   20,   21,    9],
               [  6,    1,   22, 858,   17,   10],
               [ 21,    7,   21,   16, 788,   45],
               [ 12,    4,    6,   10,   52, 800]])
```

Σχήμα 4.27: Απόδοση ακρίβειας Support vector machine



Σχήμα 4.28: Πίνακας σύγχυσης Support vector machine

```
In [64]: pipeline = Pipeline([
    ('bow', CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))),
    ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores
    ('classifier', svm_clf), # train on TF-IDF vectors w/ SVM classifier
])
<...>

In [65]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
                                X, # training data
                                y, # training labels
                                cv=10, # split data randomly into 10 parts: 9 for training, 1 for score
                                scoring='accuracy', # which scoring metric?
                                n_jobs=-1, # -1 = use all cores = faster
)
print(scores)

[ 0.94152047  0.95648995  0.94948755  0.93850659  0.95095168  0.94617356
  0.96594654  0.93299158  0.8996337  0.86007326]

In [66]: models['SVM'].append(scores.mean())
print("Accuracy: %0.2f %% (%+- %0.2f %%)" % (scores.mean() * 100, scores.std() * 2 * 100))
Accuracy: 93.42 % (+/- 5.98 %)
```

Σχήμα 4.29: Απόδοση ακρίβειας 10-Fold Cross Validation του Support vector machine

4.2.2 Ταξινόμηση στο Amazon movie reviews Dataset

Ο κώδικας που υλοποιεί την παρακάτω ταξινόμηση βρίσκεται στο Ipython Notebook με όνομα chapter4_2_2.ipynb στο αποθετήριο [6].

Αρχικά, έγινε εισαγωγή όλων των απαραίτητων βιβλιοθηκών για την υλοποίηση της ταξινόμησης.

```
1 # for Python 2: use print only as a function
2 %matplotlib inline
3 import matplotlib.pyplot as plt
4 from __future__ import print_function
```

```

5 from sklearn.feature_extraction.text import CountVectorizer
6 from sklearn.feature_extraction.text import TfidfTransformer
7 from sklearn.cross_validation import train_test_split
8 from sklearn.metrics import classification_report, f1_score
9 from sklearn.metrics import accuracy_score, confusion_matrix
10 from sklearn import metrics
11 from sklearn.naive_bayes import MultinomialNB
12 from sklearn.ensemble import RandomForestClassifier
13 import numpy as np
14 from sklearn.neighbors import KNeighborsClassifier
15 from sklearn.cross_validation import cross_val_score
16 from sklearn.cross_validation import cross_val_predict
17 from sklearn.pipeline import Pipeline
18 import ast
19 import os, csv, gc
20 import pandas as pd
21 import random
22 from sklearn import svm

```

Έπειτα φορτώθηκε το csv αρχείο που περιέχει τα δεδομένα του dataset.

In [2]:	df = pd.read_csv('cd3.csv') df.shape																																				
Out[2]:	(103149, 5)																																				
In [3]:	df.head()																																				
Out[3]:	<table border="1"> <thead> <tr> <th></th><th>review/text</th><th>product/categories</th><th>root_class</th><th>length</th><th>root_class2</th></tr> </thead> <tbody> <tr> <td>0</td><td>Not many "indie" bands get to have their own D...</td><td>['CDs & Vinyl', 'Alternative Rock', 'Indie & L...</td><td>CDs & Vinyl</td><td>507</td><td>Alternative Rock</td></tr> <tr> <td>1</td><td>I had a Death Cab for Cutie phase, so this DVD...</td><td>['CDs & Vinyl', 'Alternative Rock', 'Indie & L...</td><td>CDs & Vinyl</td><td>241</td><td>Alternative Rock</td></tr> <tr> <td>2</td><td>Every rock band needs to make at least one tou...</td><td>['CDs & Vinyl', 'Alternative Rock', 'Indie & L...</td><td>CDs & Vinyl</td><td>1221</td><td>Alternative Rock</td></tr> <tr> <td>3</td><td>I saw this last night on the Encore channel an...</td><td>['CDs & Vinyl', 'Alternative Rock', 'Indie & L...</td><td>CDs & Vinyl</td><td>1880</td><td>Alternative Rock</td></tr> <tr> <td>4</td><td>i love the dvd the only thing that irkes me is...</td><td>['CDs & Vinyl', 'Alternative Rock', 'Indie & L...</td><td>CDs & Vinyl</td><td>458</td><td>Alternative Rock</td></tr> </tbody> </table>		review/text	product/categories	root_class	length	root_class2	0	Not many "indie" bands get to have their own D...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	507	Alternative Rock	1	I had a Death Cab for Cutie phase, so this DVD...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	241	Alternative Rock	2	Every rock band needs to make at least one tou...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	1221	Alternative Rock	3	I saw this last night on the Encore channel an...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	1880	Alternative Rock	4	i love the dvd the only thing that irkes me is...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	458	Alternative Rock
	review/text	product/categories	root_class	length	root_class2																																
0	Not many "indie" bands get to have their own D...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	507	Alternative Rock																																
1	I had a Death Cab for Cutie phase, so this DVD...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	241	Alternative Rock																																
2	Every rock band needs to make at least one tou...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	1221	Alternative Rock																																
3	I saw this last night on the Encore channel an...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	1880	Alternative Rock																																
4	i love the dvd the only thing that irkes me is...	['CDs & Vinyl', 'Alternative Rock', 'Indie & L...	CDs & Vinyl	458	Alternative Rock																																

Σχήμα 4.30: Δεδομένα στα οποία θα εφαρμοσθούν τα μοντέλα ταξινόμησης του Amazon movie reviews Dataset

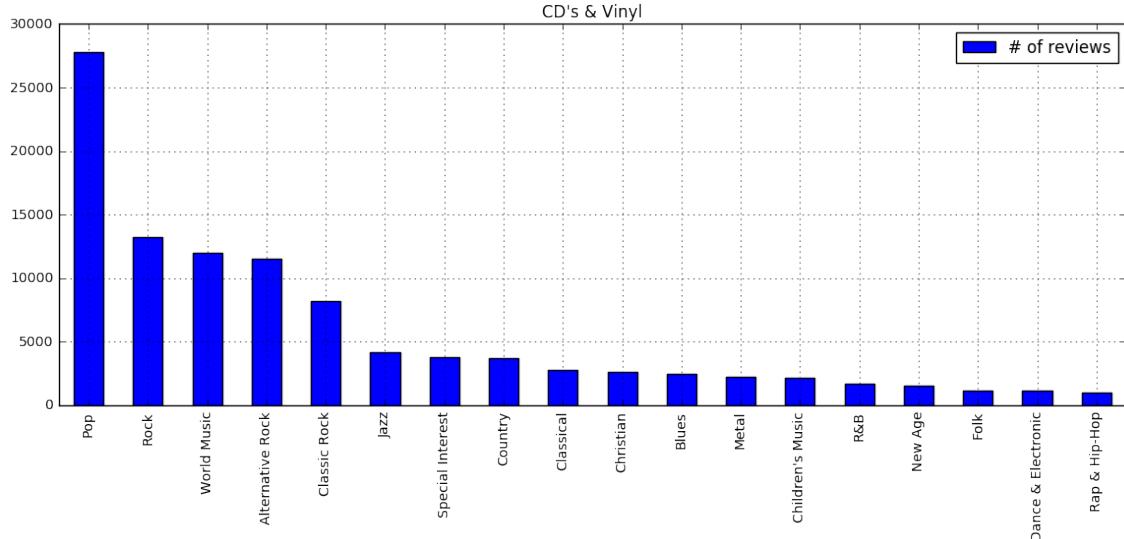
Τα δεδομένα φορτωθήκαν σε ένα dataframe αντικείμενο της βιβλιοθήκης Pandas και με την μέθοδο df.shape() βλέπουμε ότι το πλήθος των instances είναι 103,149 και τα features 5.

Με την εντολή df.head() βλέπουμε τις πρώτες γραμμές του dataframe. Η στήλη review/text περιέχει το κείμενο του review, μετά έχουμε την λίστα με τα labels, έπειτα το πρώτο label, το μήκος του review και τέλος το δεύτερο label από την αρχική λίστα.

Εδώ, θα πρέπει να αναφερθεί ότι τα πειράματα υλοποιήθηκαν σε ένα υποσύνολο του αρχικού dataset, διότι υπήρξαν προβλήματα σε πόρους μνήμης. Το αρχικό dataset αποτελούνταν από περίπου 7 εκατομμύρια reviews οπότε επιλέχθηκε το υποσύνολο της κατηγορίας “CDs &

Vinyl". Η ταξινόμηση υλοποιήθηκε ως προς το δεύτερο label (root_class2).

Παρακάτω φαίνεται η κατανομή των reviews ανά κατηγορία (δεύτερο label στην λίστα):



Σχήμα 4.31: Κατανομή των reviews ανά κατηγορία

Είναι εμφανές ότι η δεν υπάρχει ομαλή κατανομή σε κάθε κατηγορία, οπότε προχωρήσαμε σε μία τυχαία επιλογή ενός υποσυνόλου του κάθε label. Η επιλογή υλοποιήθηκε έτσι ώστε να έχουμε περίπου 1,000 reviews σε κάθε κατηγορία.

Θα εκτελέσουμε την παρακάτω μέθοδο για να καταφέρουμε την ομαλή κατανομή:

```

1 # Η συνάρτηση ranDF δέχεται ως ορίσματα ένα dataframme (df), ένα string
2 # (col) που αντιπροσωπεύει το όνομα μίας στήλης του df και δύο integers
3 # (a και b) που αντιπροσωπεύουν άνω κάι κάτω όρια στο πλήθος των instances.
4 # Επιστρέφει ένα dataframme, αντίγραφο αυτού της εισόδου με την διαφορά
5 # ότι το πλήθος των instances του νέου dataframme θα κυμαίνεται μεταξύ
6 # των τιμών του a και b.
7 def ranDF(df, col, a, b):
8     # Η label_list αποθηκεύει μια λίστα με τα labels του dataset
9     label_list = df[col].value_counts().to_dict().keys()
10    # Η rows αποθηκεύει μια λίστα από τυχαία indexes των instances του
11    # dataset που ανήκουν στο πρώτο label. Το πλήθος των indexes
12    # κυμαίνεται μεταξύ των τιμών των μεταβλητών a και b για λόγους
13    # διαχύμανσης των πλήθων τους
14    rows = np.random.choice(df[df[col] == label_list[0]].index.values,
15                            random.randint(a, b))
16    # Δημιουργείται ένα dataframme από τα instances των παραπάνω indexes
17    sampled_df = df.ix[rows]
18    # Το παραπάνω dataframme εμπλουτίζεται με instances από όλα τα labels
19    # με παρόμοιο πλήθος σε κάθε label

```

```

20     for f in label_list[1:]:
21         rows = np.random.choice(df[df[col] == f].index.values,
22                               random.randint(a, b))
23         sampled_df = sampled_df.append(df.ix[rows], ignore_index=True)
24
25     return sampled_df

```

Καλούμε την παραπάνω συνάρτηση με τις εξής παραμέτρους:

```
In [7]: # Δημιουργία του dataset (dataframe) στο οποίο οριοθετείται το πλήθος των
# reviews για κάθε label μεταξύ 970 και 1027 instances
df = ranDF(df, 'root_class2', 970, 1027)
```

```
In [8]: # Πλήθος του τελικού
df.shape
```

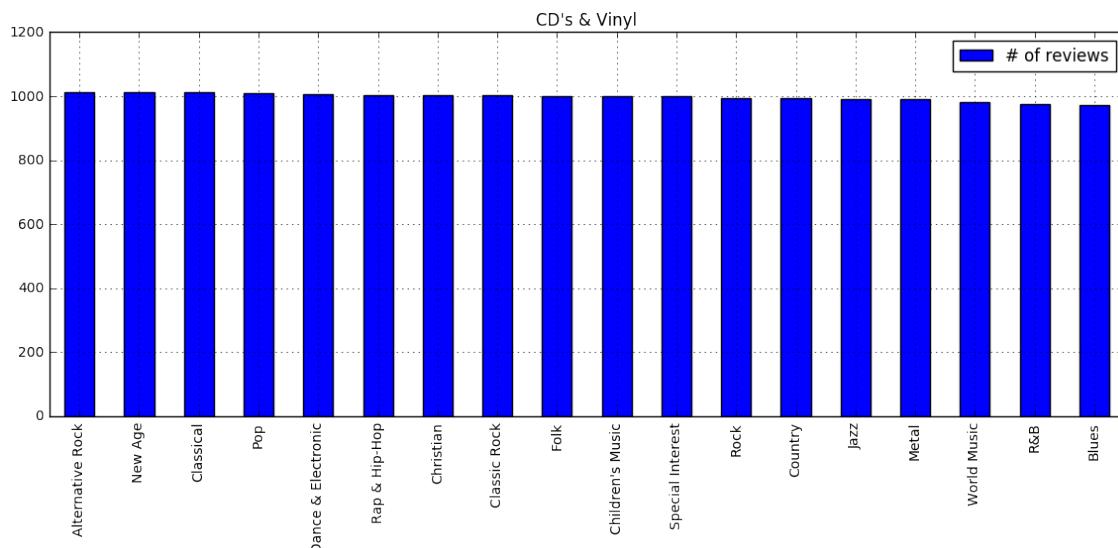
```
Out[8]: (17960, 5)
```

```
In [10]: # Πληροφορίες για το μήκος των reviews
df.length.describe()
```

```
Out[10]: count    17960.000000
mean      717.957795
std       829.015221
min       25.000000
25%      237.000000
50%      460.000000
75%      890.000000
max      17971.000000
Name: length, dtype: float64
```

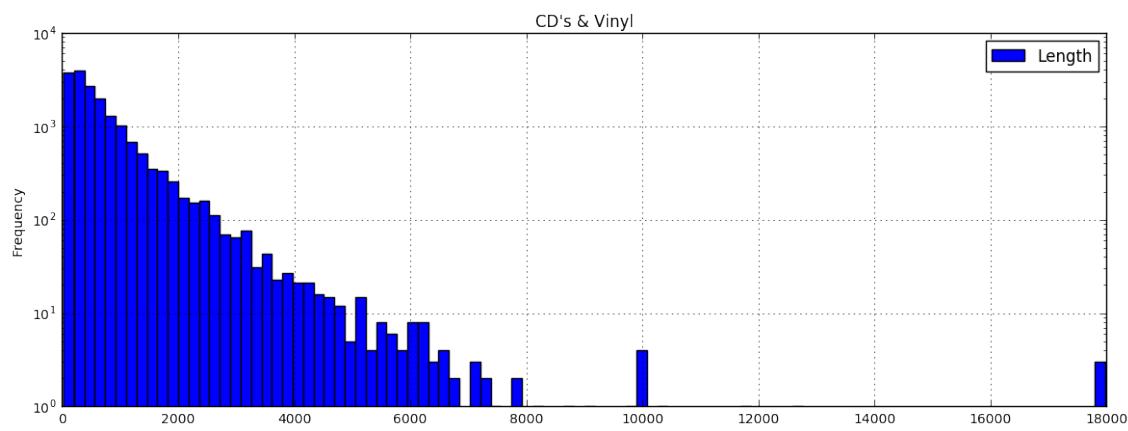
Σχήμα 4.32: Διαμόρφωση ομαλότερης κατανομής των reviews ανά κατηγορία

Βλέπουμε παραπάνω ότι το νέο dataframe έχει πλέον 17,960 instances.



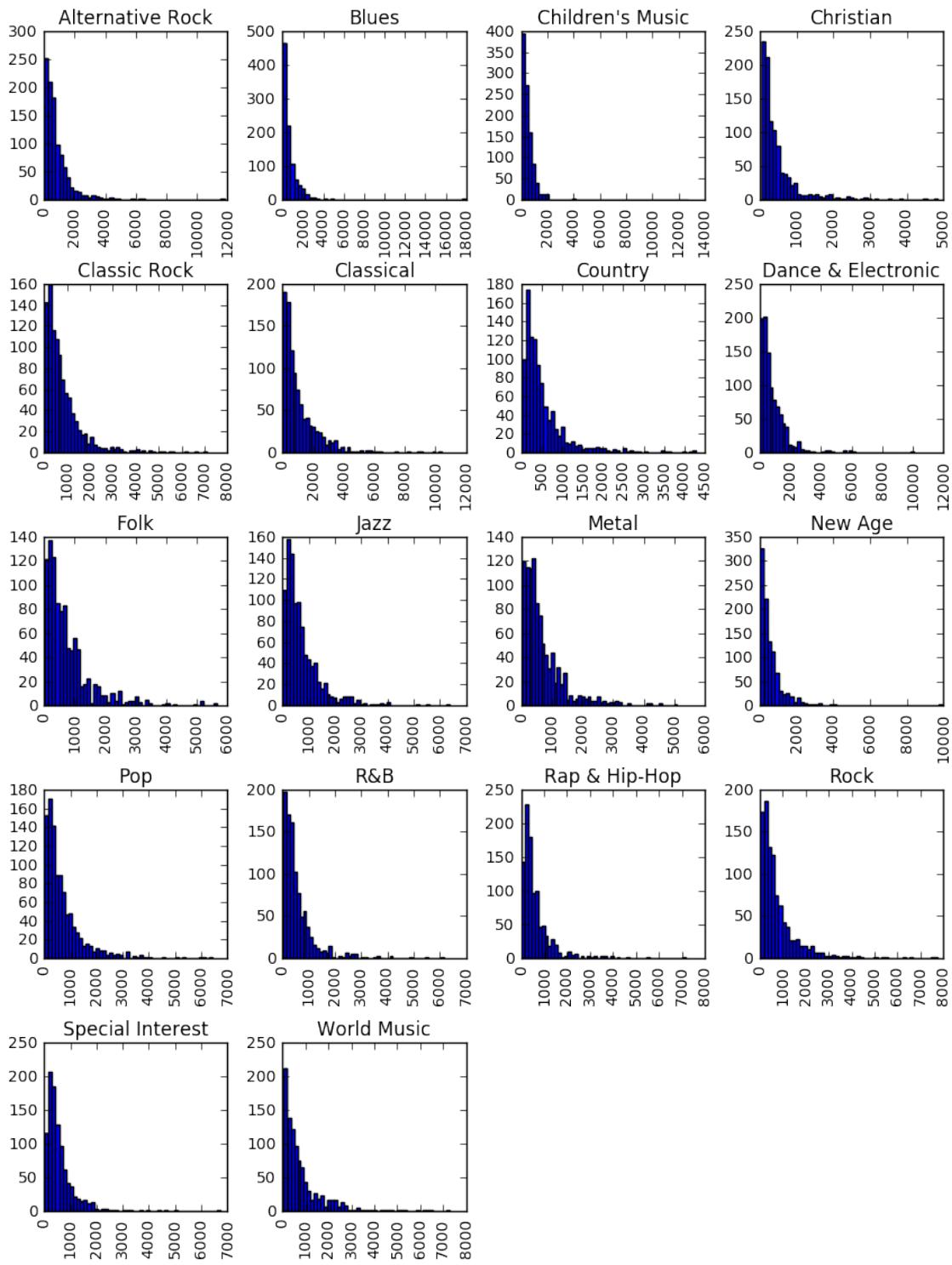
Σχήμα 4.33: Νέα (ομαλή) κατανομή των reviews ανά κατηγορία

Από το μήκος των reviews έχει παραχθεί το παρακάτω ιστόγραμμα στο οποίο έχουμε λογαριθμήσει τον άξονα y για να έχουμε καλύτερη εικόνα ως προς την συχνότητα:



Σχήμα 4.34: Κατανομή των reviews ανά κατηγορία με λογαριθμημένο τον άξονα y

Ακολουθεί το αντίστοιχο ιστόγραμμα ανά κατηγορία προϊόντων:



Σχήμα 4.35: Ιστόγραμμα ανά κατηγορία προϊόντων

Από τα παραπάνω ιστογράμματα, θα μπορούσαμε να συμπεράνουμε ότι τα περισσότερα αναλυτικά reviews αφορούν τις κατηγορίες Alternative Rock, Rock και Classical.

Στην συνέχεια, για να μπορέσουμε να υλοποιήσουμε τα μοντέλα ταξινόμησης θα πρέπει να αντιστοιχήσουμε τις τιμές της στήλης `root_class2` με μία αριθμητική τιμή. Η αντιστοίχηση

υλοποιείται με την εισαγωγή μίας νέας στήλης με όνομα real_category_num.

Έπισι, αρχικά ορίζουμε τα id's των κατηγοριών και εισάγουμε την νέα στήλη στο dataframe ως εξής:

```
In [13]: # Αριθμηση των label (id's)
tempDict = {}
for item, value in enumerate(distr.to_dict().keys()):
    tempDict[value] = item
tempDict

Out[13]: {'Alternative Rock': 0,
'Blues': 15,
'Children's Music': 5,
'Christian': 1,
'Classic Rock': 17,
'Classical': 2,
'Country': 4,
'Dance & Electronic': 11,
'Folk': 16,
'Jazz': 6,
'Metal': 7,
'New Age': 10,
'Pop': 9,
'R&B': 3,
'Rap & Hip-Hop': 12,
'Rock': 14,
'Special Interest': 8,
'World Music': 13}

In [14]: # Προσθήκη νέας στήλης real_category_num με το id των label που ανήκει το κάθε instance
df['real_category_num'] = df.root_class2.map(tempDict)
```

Σχήμα 4.36: Ορισμός id's στις κατηγορίες

Πλέον, το dataframe στο οποίο θα εφαρμόσουμε τα πειράματα θα έχει την παρακάτω μορφή:

	review/text	product/categories	root_class	length	root_class2	real_category_num
0	If you want a real eye popping, ear bleeding ...	['CDs & Vinyl', 'Alternative Rock', 'American ...']	CDs & Vinyl	165	Alternative Rock	0
1	I love the video collection. I think the sound...	['CDs & Vinyl', 'Alternative Rock', 'New Wave ...']	CDs & Vinyl	421	Alternative Rock	0
2	Movie is way better than the recent remake. Bl...	['CDs & Vinyl', 'Alternative Rock', 'Hardcore ...']	CDs & Vinyl	111	Alternative Rock	0
3	I always thought Boy George was weirder than h...	['CDs & Vinyl', 'Alternative Rock', 'New Wave ...']	CDs & Vinyl	879	Alternative Rock	0
4	I admit I did not get into New Order until Blu...	['CDs & Vinyl', 'Alternative Rock', 'New Wave ...']	CDs & Vinyl	497	Alternative Rock	0

	review/text	product/categories	root_class	length	root_class2	real_category_num
17955	I think this DVD stink. The picture quality is...	['CDs & Vinyl', 'Classic Rock', 'Arena Rock']	CDs & Vinyl	547	Classic Rock	17
17956	Technically this is a great disc. Great audio ...	['CDs & Vinyl', 'Classic Rock', 'Album-Oriente...']	CDs & Vinyl	216	Classic Rock	17
17957	the dvd is featuring two optional movies from...	['CDs & Vinyl', 'Classic Rock', 'Arena Rock']	CDs & Vinyl	1493	Classic Rock	17
17958	The Audio CD is more musical than the DVD, perh...	['CDs & Vinyl', 'Classic Rock', 'Album-Oriente...']	CDs & Vinyl	73	Classic Rock	17
17959	This dvd will not disappoint! It's Van in area	['CDs & Vinyl', 'Classic Rock', 'Album-Oriente...']	CDs & Vinyl	155	Classic Rock	17

Σχήμα 4.37: Πέντε πρώτα και πέντε τελευταία instances των reviews

Μετά την σύντομη επισκόπηση των δεδομένων μας, είμαστε σε θέση απομονώσουμε τις στήλες review/text και real_category_num, από τις οποίες η μεν πρώτη θα είναι αυτή που θα εκπαιδεύσει τα μοντέλα μας, η δε δεύτερη θα είναι η πληροφορία για το που πραγματικά ανήκει το κάθε review.

```
1 # how to define X and y for use with COUNTVECTORIZER
2 X = df['review/text']
3 y = df.real_category_num
```

Έπειτα, χωρίζουμε τα νέα μας δεδομένα, σε δύο υποσύνολα (X με 14,368 εγγραφές και y με 3,592) και σε αναλογία 80% - 20% ώστε να χρησιμοποιηθεί το 80% για την εκπαίδευση (training set) των μοντέλων και το υπόλοιπο 20% για την αξιολόγηση (testing set).

```
In [20]: # split X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=42)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)

(14368,)
(3592,)
(14368,)
(3592,)
```

Σχήμα 4.38: Διαμοίραση σε training set και testing set

Συνεχίζουμε στην μετατροπή των reviews από μορφή κειμένου σε διανυσματικό πίνακα με στήλες τις λέξεις που προέκυψαν ως οι πιο “σημαντικές” από την κλάση CountVectorizer(), γραμμές τα reviews και με δεδομένα στα κελιά την συχνότητα της κάθε λέξης που αντιστοιχεί στην στήλη. Μάλιστα, προς εξοικονόμηση μνήμης, αποθηκεύονται μόνο οι συντεταγμένες του πίνακα που περιέχουν τιμή μεγαλύτερη του 0.

```
In [21]: # instantiate CountVectorizer (with the default parameters)
vect = CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))

In [22]: # learn training data vocabulary, then use it to create a document-term matrix
try:
    X_train_dtm = vect.fit_transform(X_train)
except UnicodeDecodeError as inst:
    print (type(inst))
    print (inst)

In [23]: X_train_dtm

Out[23]: <14368x290955 sparse matrix of type '<type 'numpy.int64'>'>
         with 1522129 stored elements in Compressed Sparse Row format>

In [24]: # transform testing data (using fitted vocabulary) into a document-term matrix
X_test_dtm = vect.transform(X_test)
X_test_dtm

Out[24]: <3592x290955 sparse matrix of type '<type 'numpy.int64'>'>
         with 287299 stored elements in Compressed Sparse Row format>
```

Σχήμα 4.39: Στοιχεία διανυσματικού πίνακα

Βλέπουμε ότι ο διανυσματικός πίνακας που προέκυψε (document-term matrix) αποτελείται από 287,299 στήλες.

Θα εξεταστούν με την παρακάτω σειρά οι εξής αλγόριθμοι ταξινόμησης:

- Naive Bayes
- Random Forest
- Logistic Regression
- K-nearest neighbors
- Support vector machine

Σε κάθε έναν από αυτούς θα εκτελεσθεί μία σειρά από βήματα, μέσω των οποίων θα ολοκληρώνεται κάθε φορά η υλοποίηση του μοντέλου ταξινόμησης. Τα βήματα είναι τα παρακάτω:

1. Αρχικοποίηση μοντέλου
2. Εκπαίδευση με τα δεδομένα X_train_dtm και y_train
3. Διαδικασία πρόβλεψης με είσοδο το X_test_dtm
4. Υπολογισμός ακρίβειας των προβλέψεων του προηγούμενου βήματος
5. Υπολογισμός και εμφάνιση τού πίνακα σύγχυσης
6. Επιβεβαίωση ότι τα αποτελέσματα μας δεν είναι biased βάσει k-Fold Cross Validation, με $k = 10$.

Naive Bayes

Στον παρακάτω κώδικα εκτελούνται τα πρώτα 3 βήματα:

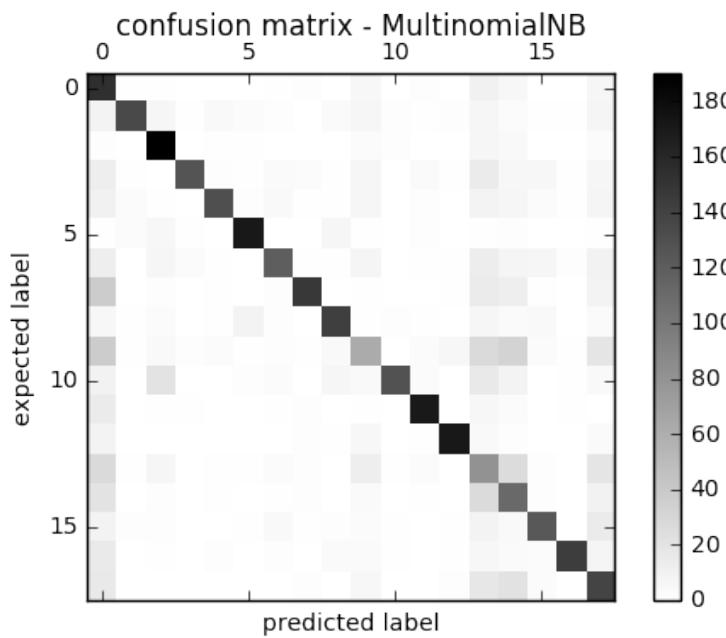
```
1 # 1. instantiate a Multinomial Naive Bayes model
2 nb = MultinomialNB()
3 # 2. train the model using X_train_dtm
4 nb.fit(X_train_dtm, y_train)
5 # 3. make class predictions for X_test_dtm
6 y_pred_class = nb.predict(X_test_dtm)
```

Βλέπουμε ότι το ποσοστό ακρίβειας προέκυψε στα 68.26%. Πιο κάτω φαίνεται και ο πίνακας σύγχυσης σε γράφημα (Εικόνα 4.40).

```
In [29]: # calculate accuracy of class predictions
res = metrics.accuracy_score(y_test, y_pred_class)
models['NB'].append(res)
print("Accuracy: %.2f %%" % (res*100))
```

Accuracy: 68.26 %

Σχήμα 4.40: Απόδοση ακρίβειας Naive Bayes



Σχήμα 4.41: Πίνακας σύγχυσης Naive Bayes

Παρακάτω φαίνεται και το τελευταίο βήμα, όπου και στην εφαρμογή του 10-Fold Cross Validation η απόδοση των επιτυχών προβλέψεων είναι σχεδόν στα ίδια επίπεδα (71.44%) με την training/testing μέθοδο.

```
In [32]: pipeline = Pipeline([
    ('bow', CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))),
    ('tfidf', TfidfTransformer()),
    ('classifier', nb), # train on TF-IDF vectors w/ Naive Bayes classifier
])

```

```
In [33]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
                               X, # training data
                               y, # training labels
                               cv=10, # split data randomly into 10 parts: 9 for training, 1 for score
                               scoring='accuracy', # which scoring metric?
                               n_jobs=-1, # -1 = use all cores = faster
)
print(scores)
[ 0.71721085  0.70304709  0.7163515   0.72024472  0.72575251  0.70886782
  0.71149554  0.71205357  0.71356784  0.7150838 ]
```

```
In [34]: res = scores.mean()
models['NB'].append(res)
print("Accuracy: %0.2f %% (+/- %0.2f %%)" % (scores.mean()*100, scores.std() * 2 * 100))
Accuracy: 71.44 % (+/- 1.18 %)
```

Σχήμα 4.42: Απόδοση ακρίβειας 10-Fold Cross Validation του Naive Bayes

Logistic regression

Στον παρακάτω κώδικα εκτελούνται τα πρώτα 3 βήματα:

```
1 # 1. instantiate a logistic regression model
2 logreg = LogisticRegression()
```

```

3 # 2. train the model using X_train_dtm
4 logreg.fit(X_train_dtm, y_train)
5 # 3. make class predictions for X_test_dtm
6 y_pred_class = logreg.predict(X_test_dtm)

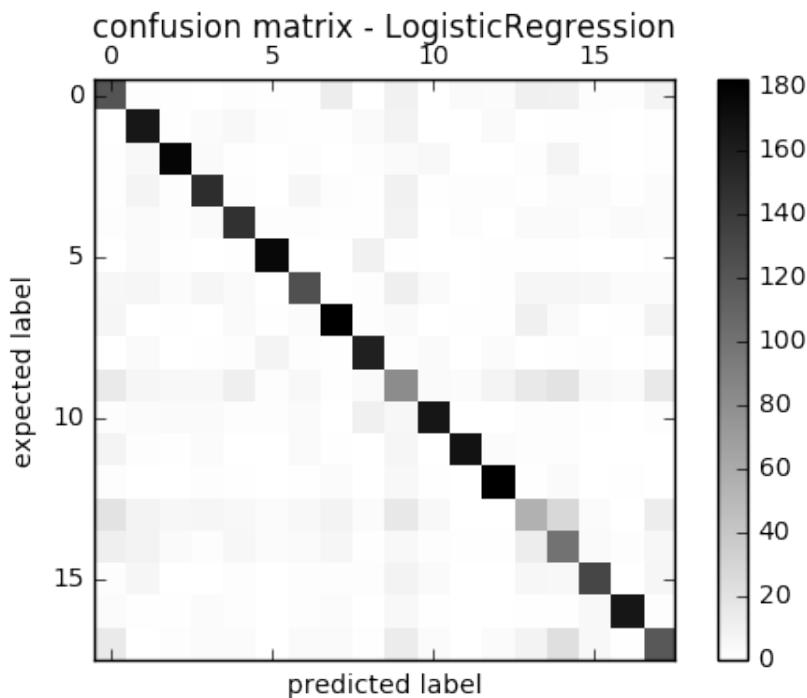
```

Βλέπουμε πως κάτω ότι το ποσοστό ακρίβειας προέκυψε στα 71.58%. Επίσης φαίνεται και ο πίνακας σύγχυσης σε γράφημα:

```
In [38]: # calculate accuracy
res = metrics.accuracy_score(y_test, y_pred_class)
models['LR'].append(res)
print("Accuracy: %0.2f %%" % (res*100))

Accuracy: 71.58 %
```

Σχήμα 4.43: Απόδοση ακρίβειας Logistic regression



Σχήμα 4.44: Πίνακας σύγχυσης Logistic regression

Παρακάτω φαίνεται και το τελευταίο βήμα, όπου και στην εφαρμογή του 10-Fold Cross Validation η απόδοση των επιτυχών προβλέψεων είναι σχεδόν στα ίδια επίπεδα (72.54%) με την training/testing μέθοδο.

```
In [41]: pipeline = Pipeline([
    ('bow', CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))),
    ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores
    ('classifier', logreg), # train on TF-IDF vectors w/ Logistic Regression classifier
])
<ipython-input-41-1234567890>

In [42]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
                               X, # training data
                               y, # training labels
                               cv=10, # split data randomly into 10 parts: 9 for training, 1 for scoring
                               scoring='accuracy', # which scoring metric?
                               n_jobs=-1, # -1 = use all cores = faster
)
print(scores)

[ 0.73990039  0.71523546  0.7246941   0.72858732  0.7335563   0.72169548
  0.71819196  0.72209821  0.71356784  0.73631285]

In [43]: res = scores.mean()
models['LR'].append(res)
print("Accuracy: %0.2f %% (+/- %0.2f %%)" % (scores.mean()*100, scores.std() * 2 * 100))

Accuracy: 72.54 % (+/- 1.70 %)
```

Σχήμα 4.45: Απόδοση ακρίβειας 10-Fold Cross Validation του Logistic regression

Random Forest

Στον παρακάτω κώδικα εκτελούνται τα πρώτα 3 βήματα:

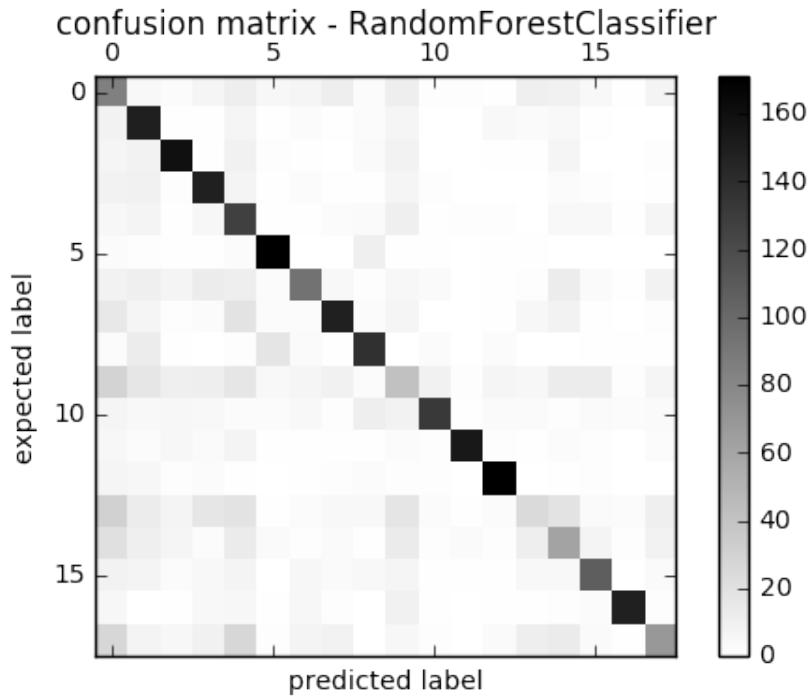
```
1 # 1. instantiate a Random Forest model
2 rf = RandomForestClassifier()
3 # 2. train the model using X_train_dtm
4 rf.fit(X_train_dtm, y_train)
5 # 3. make class predictions for X_test_dtm
6 y_pred_class = rf.predict(X_test_dtm)
```

Βλέπουμε πως κάτω ότι το ποσοστό ακρίβειας προέκυψε στα 59.44%. Επίσης φαίνεται και ο πίνακας σύγχυσης σε γράφημα.

```
In [45]: # calculate accuracy of class predictions
res = metrics.accuracy_score(y_test, y_pred_class)
models['RF'].append(res)
print("Accuracy: %0.2f %%" % (res*100))

Accuracy: 59.44 %
```

Σχήμα 4.46: Απόδοση ακρίβειας Random Forest



Σχήμα 4.47: Πίνακας σύγχυσης Random Forest

Παραχάτω φαίνεται και το τελευταίο βήμα, όπου και στην εφαρμογή του 10-Fold Cross Validation η απόδοση των επιτυχών προβλέψεων είναι ακριβώς στα ίδια επίπεδα (59.44%) με την training/testing μέθοδο.

```
In [48]: pipeline = Pipeline([
    ('bow', CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))),
    ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores
    ('classifier', rf), # train on TF-IDF vectors w/ Random forest classifier
])

```

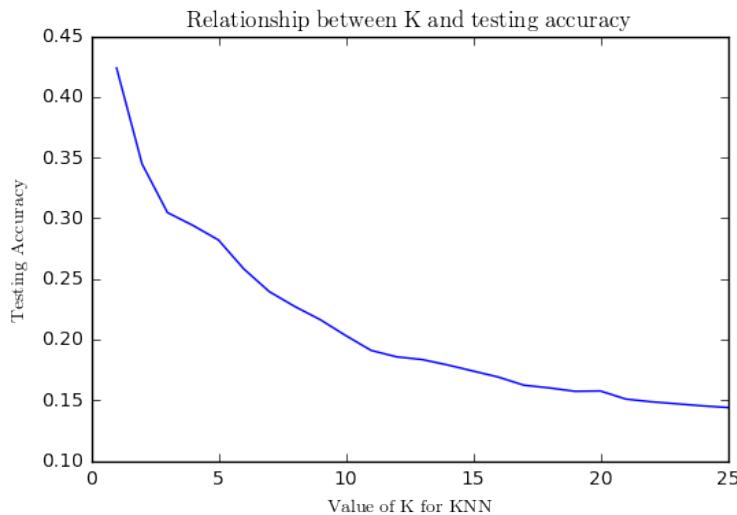
```
In [49]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
                               X, # training data
                               y, # training labels
                               cv=10, # split data randomly into 10 parts: 9 for training, 1 for s
                               scoring='accuracy', # which scoring metric?
                               n_jobs=-1, # -1 = use all cores = faster
)
print(scores)
[ 0.60044272  0.5700831   0.5945495   0.60122358  0.61538462  0.59286113
  0.59542411  0.59207589  0.59073143  0.59106145]
```

```
In [50]: res = scores.mean()
models['RF'].append(res)
print("Accuracy: %0.2f %% (+/- %0.2f %%)" % (scores.mean()*100, scores.std() * 2 * 100))
Accuracy: 59.44 % (+/- 2.14 %)
```

Σχήμα 4.48: Απόδοση ακριβειας 10-Fold Cross Validation του Random Forest

K-nearest neighbors

Σε αυτό το μοντέλο θα πρέπει όπως και πιο πάνω να αποφασιστεί η τιμή που θα δώσουμε στο k . Το k αντιπροσωπεύει το πλήθος των γειτόνων που θα εξετασθούν για να ληφθεί η απόφαση της ταξινόμησης για το εξεταζόμενο review. Όπως και στο προηγούμενο dataset, θα δοκιμάσουμε πάλι την επίδοση του μοντέλου με τιμή του k από 1 έως 25 και θα επιλέξουμε αυτή που θα μας επιστρέψει την μέγιστο ποσοστό επιτυχών προβλέψεων. Το γράφημα που προκύπτει από τις τιμές του k και των ποσοστών ακρίβειας είναι το παρακάτω:



Σχήμα 4.49: Απόδοση ακρίβειας του K-nn για τιμές του k από 1 έως 25

Βλέπουμε, όπως και πιο πάνω, ότι η υψηλότερη απόδοση προκύπτει όταν το $k = 1$ και όσο αυτό αυξάνεται, η απόδοση παίρνει χαμηλότερες τιμές.

Στον παρακάτω κώδικα εκτελούνται τα πρώτα 3 βήματα:

```

1 # 1. instantiate a KNN model for K=1
2 knn = KNeighborsClassifier(n_neighbors=1)
3 # 2. train the model using X_train_dtm
4 knn.fit(X_train_dtm, y_train)
5 # 3. make class predictions for X_test_dtm
6 y_pred_class = knn.predict(X_test_dtm)

```

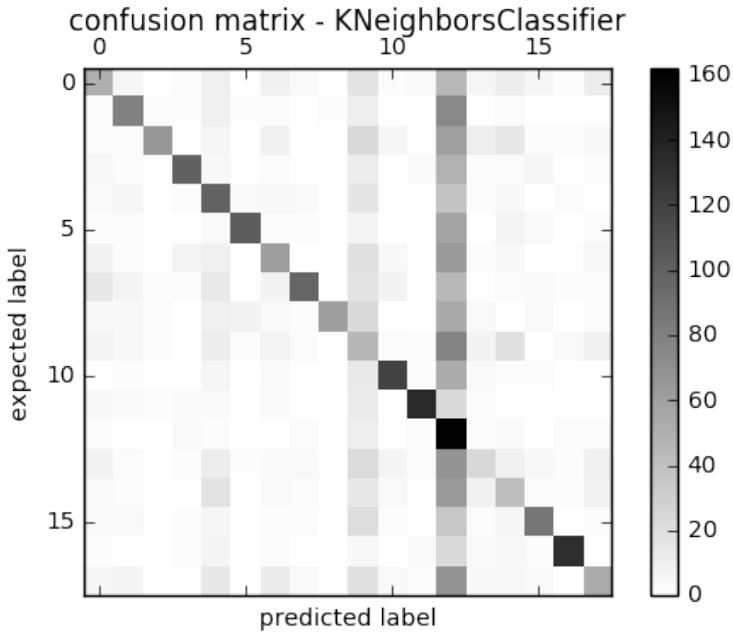
Βλέπουμε πιο κάτω ότι το ποσοστό ακρίβειας προέκυψε στα 42.37%. Επίσης φαίνεται και ο πίνακας σύγχυσης σε γράφημα.

In [52]:

```
# calculate accuracy of class predictions
res = metrics.accuracy_score(y_test, y_pred_class)
models['KNN'].append(res)
print("Accuracy: %0.2f %%" % (res * 100))
```

Accuracy: 42.37 %

Σχήμα 4.50: Απόδοση ακρίβειας K-nn



Σχήμα 4.51: Πίνακας σύγχυσης K-nn

Παρακάτω φαίνεται και το τελευταίο βήμα, όπου και στην εφαρμογή του 10-Fold Cross Validation η απόδοση των επιτυχών προβλέψεων είναι ακριβώς στα ίδια επίπεδα (40.57%) με την training/testing μέθοδο.

```
In [55]: Pipeline([
    CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3)), # string
    TfidfTransformer()), # integer counts to weighted TF-IDF scores
    classifier, knn), # train on TF-IDF vectors w/ KNN classifier

In [56]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
                               X, # training data
                               y, # training labels
                               cv=10, # split data randomly into 10 parts: 9 for training, 1 for
                               scoring='accuracy', # which scoring metric?
                               n_jobs=-1, # -1 = use all cores = faster
                               )
print(scores)
[ 0.41505257  0.39224377  0.37931034  0.40378198  0.42307692  0.41215839
  0.40513393  0.40290179  0.41708543  0.40614525]

In [57]: res = scores.mean()
models['KNN'].append(res)
print("Accuracy: %.2f %% (+/- %.2f %%)" % (scores.mean() * 100, scores.std() * 2 * 100))
Accuracy: 40.57 % (+/- 2.41 %)
```

Σχήμα 4.52: Απόδοση ακρίβειας 10-Fold Cross Validation του K-nn

Support vector machine

Στον παρακάτω κώδικα εκτελούνται τα πρώτα 3 βήματα:

```
1 # 1. instantiate a Support vector machine (SVM) model
2 svm_clf = svm.SVC(kernel='linear')
```

```

3 # 2. train the model using X_train_dtm
4 svm_clf.fit(X_train_dtm, y_train)
5 # 3. make class predictions for X_test_dtm
6 y_pred_class = svm_clf.predict(X_test_dtm)

```

Βλέπουμε πως κάτω ότι το ποσοστό ακρίβειας προέκυψε στα 65.59%. Επίσης φαίνεται και ο πίνακας σύγχυσης σε γράφημα (Εικόνα 4.54).

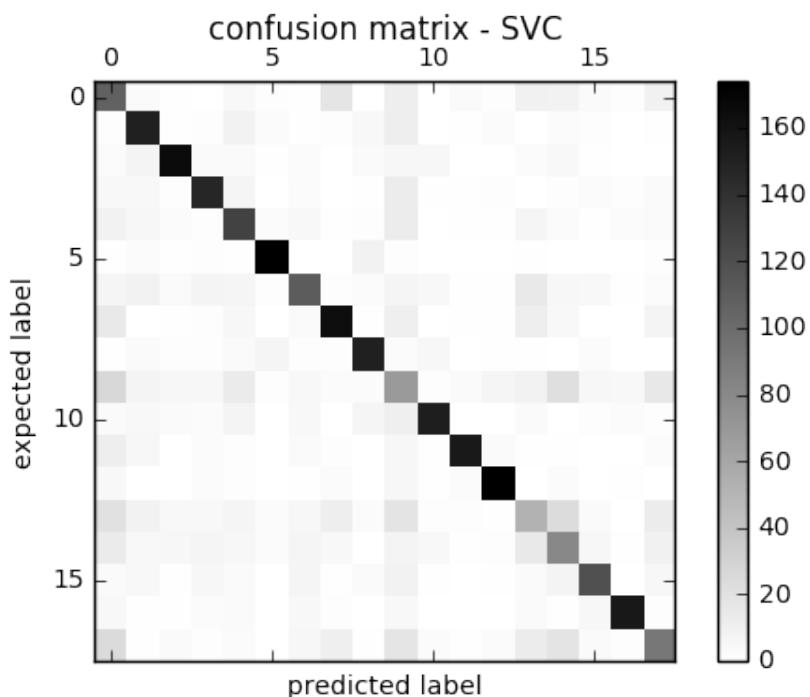
```

In [60]: # calculate accuracy
res = metrics.accuracy_score(y_test, y_pred_class)
models['SVM'].append(res)
print("Accuracy: %0.2f %%" % (res * 100))

Accuracy: 65.59 %

```

Σχήμα 4.53: Απόδοση ακρίβειας Support vector machine



Σχήμα 4.54: Πίνακας σύγχυσης Support vector machine

Παρακάτω φαίνεται και το τελευταίο βήμα, όπου και στην εφαρμογή του 10-Fold Cross Validation η απόδοση των επιτυχών προβλέψεων είναι σχεδόν στα ίδια επίπεδα (73.54%) με την training/testing μέθοδο.

```
In [63]: pipeline = Pipeline([
    ('bow', CountVectorizer(max_df=0.7, min_df=2, stop_words='english', ngram_range=(1, 3))), #
    ('tfidf', TfidfTransformer()), # integer counts to weighted TF-IDF scores
    ('classifier', svm_clf), # train on TF-IDF vectors w/ SVM classifier
])

In [64]: scores = cross_val_score(pipeline, # steps to convert raw messages into models
X, # training data
y, # training labels
cv=10, # split data randomly into 10 parts: 9 for training, 1 for scoring
scoring='accuracy', # which scoring metric?
n_jobs=-1, # -1 = use all cores = faster
)
print(scores)

[ 0.74377421  0.71966759  0.74860957  0.73414905  0.74080268  0.73508087
 0.73158482  0.73381696  0.72473479  0.74134078]
In [65]: res = scores.mean()
models['SVM'].append(res)
print("Accuracy: %0.2f %%" % (scores.mean() * 100, scores.std() * 2 * 100))

Accuracy: 73.54 % (+/- 1.66 %)
```

Σχήμα 4.55: Απόδοση ακρίβειας 10-Fold Cross Validation του Support vector machine

4.3 Προβλήματα Συσταδοποίησης – Μεθοδολογία και αποτελέσματα

Παρακάτω παρουσιάζεται όλη η διαδικασία αντιμετώπισης του προβλήματος της συστηματοποίησης σε περιβάλλον IPython Notebook, η μεθοδολογία που ακολουθήθηκε και τα αποτελέσματα που προέκυψαν. Αρχικά θα δούμε το dataset που περιγράφεται στην ενότητα 3.1 (Six Categories of Amazon Product Reviews) στο οποίο εφαρμόσθηκε η μέθοδος Latent Dirichlet Allocation και κατόπιν εκείνο της ενότητας 3.2 (Amazon movie reviews dataset), όπου η συσταδοποίηση υλοποιήθηκε μέσα από τους πίνακες-μάσκες συνάφειας.

4.3.1 Η μέθοδος Latent Dirichlet Allocation (Topic model)

Τα δεδομένα στα οποία θα εφαρμόσουμε συσταδοποίηση προέρχονται από έξι ήδη προκαθορισμένες κατηγορίες (mobilephone, cameras, video_surveillance, TVs, tablets και laptops), οπότε ο αριθμός των topics (clusters) που θα επιλέξουμε να παραχθούν θα είναι από 3 έως και 12 ($n/2$ έως και $2 \cdot n$, με $n = 6$) έτσι ώστε να υπάρχει μία ολοκληρωμένη προσέγγιση ως προς το αρχικό και επιβεβαιωμένο πλήθος από topics. Ο κώδικας που υλοποιεί την συσταδοποίηση με την μέθοδο LDA βρίσκεται στο Ipython Notebook με όνομα chapter4_3_1.ipynb στο αποθετήριο [6].

Για την υλοποίηση της συσταδοποίησης με την μέθοδο LDA, θα εκτελεσθεί μία σειρά από βήματα μέσω των οποίων θα ολοκληρωθεί η εξαγωγή των topics.

Τα βήματα είναι τα παρακάτω:

1. Εισαγωγή απαραίτητων βιβλιοθηκών.
2. Άνοιγμα αρχείων και φόρτωση στην μνήμη των δεδομένων από το dataset

3. Επεξεργασία (preprocessing) των δεδομένων πριν την εφαρμογή κάποιου μοντέλου όπως παρακάτω:

- (α') Μετατροπή του κειμένου σε lowercase
- (β') Μετατροπή του κειμένου σε λίστα από λέξεις (bag of words – tokenization)
- (γ') Αφαίρεση των stop words (λέξεις που δεν προσφέρουν πληροφορία στην ανάλυση του κειμένου όπως προθέσεις, άρθρα κλπ)
- (δ') Αφαίρεση των λέξεων που το μήκος τους είναι μικρότερο του 3.
- (ε') Εφαρμογή του αλγορίθμου Porter Stemming¹⁷

4. Εξαγωγή των topics από την LDA για πλήθος από 3 έως και 12.

5. Μετατροπή του topic σε format τέτοιο ώστε να είναι δυνατή η αντληση πληροφορίας των κυρίαρχων λέξεων. Με άλλα λόγια, το topic όταν εξάγεται από την LDA έχει την παρακάτω μορφή:

`(0, u'0.025*"summer" + u'0.021*"distanc" + ... + u'0.005*"shot" + 0.001*"help")`

όπου στην αρχή υπάρχει ένας ακέραιος που είναι το id του topic και στην συνέχεια ακολουθεί μία συμβολοσειρά (string) όπου περιέχει μία σειρά από ζεύγη βαθμολογιών/πιθανοτήτων και λέξεων που αντιστοιχούν στο εν λόγω topic.

Μετά την μετατροπή, το παραπάνω αλλάζει σε:

`(0, 'summer': 0.025, 'distanc': 0.021, ... , 'shot': 0.005, 'help': 0.001)`

όπου ο ακέραιος παρέμεινε όπως πριν αλλά η συμβολοσειρά μετατράπηκε σε ένα python dict από το οποίο μπορούμε να αντλήσουμε την πληροφορία που περιέχει το topic.

6. Εφαρμόζουμε μια βαθμολόγηση όλων των reviews και υπολογίζουμε την απόδοση που πετυχαίνουν σε κάθε ένα topic. Αυτή η υλοποίηση έχει ως είσοδο το κείμενο του review και την κατηγορία στην οποία ανήκει με την εξής μορφή:

`[[u'love', u'phone', u'much', ..., u'perfect'], 0],`

όπου το πρώτο στοιχείο είναι το επεξεργασμένο κείμενο σε μορφή bags of words και το δεύτερο στοιχείο το id της κατηγορίας όπου ανήκει. Μετά την βαθμολόγηση προστίθεται η πληροφορία των επιδόσεων σε κάθε topic, οπότε η μορφή του αλλάζει όπως παρακάτω:

`[[u'love', u'phone', u'much', ..., u'perfect'], 0, [(0, 0.189), (1, 0.01), ..., (5, 0.086)]]`

όπου το επιπλέον στοιχείο περιέχει την βαθμολογία που πέτυχε το review στο κάθε topic.

Αυτός ο βαθμός προκύπτει από την συχνότητα της κάθε λέξεως πολλαπλασιασμένη με τον βαθμό που έχει αυτή βάσει της εξαγωγής από την LDA.

¹⁷<https://tartarus.org/martin/PorterStemmer/>

7. Υπολογίζεται η αντιστοίχηση των topics που προέκυψαν από την LDA με τις κατηγορίες που έχουμε από το dataset. Αυτή η αντιστοίχηση είναι αναγκαία διότι είναι πιθανό, ιδιαίτερα όταν επιλέγουμε να παραχθούν πολλά topics, να έχουμε κατηγορίες που δεν αντιστοιχίζονται με αυτές που έχουμε στο dataset, π.χ. ενώ έχουμε κατηγορίες προϊόντων, να προκύψει ένα topic που αφορά δυσαρεστημένους πελάτες ή διαδικασίες επιστροφής προϊόντων κ.λπ.
8. Γίνεται αξιολόγηση και υπολογίζεται το ποσοστό επιτυχούς πρόβλεψης για κάθε review συγχρίνοντας την κατηγορία που ανήκει με αυτήν (της LDA) που κατάφερε την μέγιστη βαθμολογία. Εάν αυτές οι δύο συμπίπτουν, έχουμε σωστή πρόβλεψη.

Αρχικά, έγινε εισαγωγή όλων των απαραίτητων βιβλιοθηκών για την υλοποίηση της συσταδοποίησης.

```

1 from nltk.tokenize import RegexpTokenizer
2 from stop_words import get_stop_words
3 from nltk.stem.porter import PorterStemmer
4 from gensim import corpora, models
5 import gensim
6 import os, json
7 import pyLDAvis.gensim
8 from bs4 import BeautifulSoup
9 from random import randint
10 import numpy as np
11 import pandas as pd
12 import matplotlib.pyplot as plt

```

Στην συνέχεια, δηλώνονται οι βοηθητικές συναρτήσεις που υλοποιήθηκαν για την ολοκλήρωση της συσταδοποίησης. Αυτές είναι η `get_topic_weight()`, η `score()` και η `get_eval()`. Ο κώδικας τους φαίνεται στο Ipython Notebook με όνομα `chapter4_3_1.ipynb` στο αποθετήριο [6].

Στην συνέχεια, φορτώνουμε το dataset, το οποίο όπως αναφέρθηκε παραπάνω, είναι αποθηκευμένο σε JSON αρχεία σε φακέλους ανά κατηγορία. Οπότε έχουμε τον παρακάτω κώδικα:

```
In [3]: # Δημιουργεί μία λίστα με τα ονόματα των φακέλων που βρίσκονται μέσα στο directory data/
folder_names = os.listdir("data/")

# Για κάθε φάκελο
for f in folder_names:
    # Φτιάξε μία λίστα με τα ονόματα των αρχείων που βρίσκονται μέσα στον φάκελο f
    file_names = os.listdir("data/" + f)
    # Αποθήκευσε το όνομα της κατηγορίας και το id αυτής
    dict_topics[folder_names.index(f)] = f
    sum_of_reviews = 0
    # Για κάθε αρχείο
    for it in file_names:
        # Άνοιξε το
        with open("data/" + f + '/' + it) as data_file:
            data = json.load(data_file)
            # Ενημέρωση αθροίσματος του πλήθους των Reviews
            sum_of_reviews += len(data['Reviews'])
            # Για κάθε Review
            for i in data['Reviews']:
                try:
                    # Αφαιρεσε όλα τα html tags
                    soup = BeautifulSoup(i['Content'], "lxml")
                    #Ενημέρωση του doc_set και για το topic από το οποίο προέρχεται το review
                    doc_set.append((soup.get_text(), folder_names.index(f)))
                except:
                    pass
        print sum_of_reviews, ' reviews in ', f
print 'There are ',len(doc_set), ' reviews in the dataset!'
print 'Categories: ', dict_topics
```

Σχήμα 4.56: Προσπέλαση και φόρτωση δεδομένων από τα JSON αρχεία

Κατά την διάρκεια της φόρτωσης, τυπώνονται κάποια στατιστικά στοιχεία όπως για παράδειγμα το πλήθος των reviews ανά κατηγορία. Τα στοιχεία είναι τα εξής:

```
1 4814 reviews in mobilephone
2 4578 reviews in cameras
3 4677 reviews in video_surveillance
4 4461 reviews in TVs
5 4484 reviews in tablets
6 4320 reviews in laptops
7 There are 27320 reviews in the dataset!
8 Categories:{  
9     0: 'mobilephone',  
10    1: 'cameras',  
11    2: 'video_surveillance',  
12    3: 'TVs',  
13    4: 'tablets',  
14    5: 'laptops'  
15}
```

Παρακάτω φαίνεται η διαδικασίας της επεξεργασίας:

```
In [4]: # Βρόγχος που διαπερνάει την λίστα των reviews
for item in doc_set:
    try:
        # κατακερματισμός σε λέξεις και αφαίερεση κεφαλαίων γραμματών
        raw = item[0].lower()
        tokens = tokenizer.tokenize(raw)
        # Αφαίρεση των stop words
        stopped_tokens0 = [i for i in tokens if not i in en_stop]
        # Αφαίρεση λέξεων με μήκος < 3
        stopped_tokens = [i for i in stopped_tokens0 if len(i) >= 4]
        # Εφαρμογή αλγορίθμου Porter Stemming
        stemmed_tokens = [p_stemmer.stem(i) for i in stopped_tokens]
        # Καταχώρηση του επεξεργασμένου κειμένου
        texts.append(stemmed_tokens)
        texts_eval.append([stemmed_tokens, item[1]])
    except:
        pass
# turn our tokenized documents into a id <-> term dictionary
dictionary = corpora.Dictionary(texts)
# convert tokenized documents into a document-term matrix
corpus = [dictionary.doc2bow(text) for text in texts]
```

Σχήμα 4.57: Data preprocessing

Η εξαγωγή των topics υλοποιείται με το παρακάτω κώδικα:

```
1 # Πλήθος λέξεων που θα περιέχει το κάθε topic
2 num_of_words = 250
3 # Λίστες που θα κρατάνε τα ενδιάμεσα αποτελέσματα του βρόγχου
4 model_topics_list = []
5 model_topics = []
6 review_score = []
7 n = 6
8 for num_t in range(n/2, n*2 + 1):
9     # Δημιουργία των topics σε αντικείμενο κλάσης από την LDA
10    temp_model = gensim.models.ldamodel.LdaModel(corpus,
11        num_topics=int(num_t), id2word = dictionary, passes=20)
12    model_topics.append(temp_model)
13    #Εξαγωγή των topics
14    temp_model_topics = temp_model.print_topics( num_topics=int(num_t),
15        num_words=num_of_words)
16    # Εγγραφή των topics σε αρχείο
17    with open("OutputNew.txt", "a") as text_file:
18        text_file.write("\t\tres%s: %s\n\n\n"
19                      % (num_t, temp_model_topics))
20    #Εξαγωγή των βαρών των λέξεων ανά topic σε dictionary
21    temp_topic_weight = get_topic_weight(temp_model_topics)
22    #Ενημέρωση της αρχικής λίστας με την βαθμολογία κάθε review ανά topic
23    temp_review_score = score(texts_eval, temp_topic_weight)
```

 24 `review_score.append(temp_review_score)`

Η αντιστοίχηση μεταξύ των εξαγόμενων topics το προηγούμενου βήματος με τις πραγματικές κατηγορίες επιτυγχάνεται με τον παρακάτω κώδικα, αλλά και με ταυτόχρονη υποκειμενική παρατήρηση:

```
In [180]: # Σάρωση όλων των πιθανών map μεταξύ LDA topic και πραγματικού για να δω που προκύπτει μέγιστη απόδοση
for tt in range(10):
    print 'topic', tt + 3
    theD = {}
    s = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
    # Για κάθε topic
    for ga in range(tt + 3):
        mmaaxx = 0
        # Για κάθε topic του dataset
        for gb in range(6):
            # Λίστα με topics προς αγνόηση
            d = s[1:]
            # Αφορίσου τον υπό εξέταση topic για να μην αγνοηθεί
            d.remove(ga)
            # Συνδιασμός αντιστοίχισης των 2 topics
            di = {ga: gb}
            # Βαθμολογία που πέτυχε ο παραπάνω συνδιασμός
            rrr = get_eval(di, review_score[tt], d)
            # Ελεγχος μέγιστης βαθμολογίας και αποθήκευση αυτής
            if rrr[1] > mmaaxx:
                mmaaxx = rrr[1]
                percc = (rrr[1], rrr[2], round(rrr[2]/float(rrr[3]) * 100, 3))
                mmaaxxa = ga
                mmaaxxb = gb
                theD[ga] = gb
        print percc, mmaaxxa: mmaaxxb, rrr
    print ' - ', theD, ' - ', get_eval(theD, review_score[tt], [])
```

Σχήμα 4.58: Διαδικασία αντιστοίχησης εξαγόμενων topics με τις πραγματικές κατηγορίες

Πλέον, έχοντας τις αντιστοιχήσεις μεταξύ των δύο ομάδων των topics, μπορούμε να υπολογίσουμε το ποσοστό επιτυχούς πρόβλεψης και για τις δέκα περιπτώσεις που εξετάσαμε (για $n = 3, 4, \dots, 12$)

```
In [181]: print '3 topics'
top3a = get_eval({0: 1, 1: 4, 2: 5}, review_score[0], [])
print top3a[0], '% accuracy, ', 100 - round(top3a[2]/float(top3a[3])*100, 2), '% loss'
3 topics
44.674 % accuracy,  0.0 % loss

In [184]: print '4 topics'
top4a = get_eval({0: 5, 1: 0, 2: 4, 3: 1}, review_score[1], [])
print top4a[0], '% accuracy, ', 100 - round(top4a[2]/float(top4a[3])*100, 2), '% loss'
4 topics
58.367 % accuracy,  0.0 % loss

In [192]: print '5 topics'
top5a = get_eval({0: 3, 1: 4, 2: 5, 3: 1, 4: 0}, review_score[2], [])
print top5a[0], '% accuracy, ', 100 - round(top5a[2]/float(top5a[3])*100, 2), '% loss'
5 topics
65.359 % accuracy,  0.0 % loss

In [220]: print '6 topics'
top6a = get_eval({0: 0, 1: 2, 2: 1, 3: 5, 4: 4, 5: 3}, review_score[3], [])
print top6a[0], '% accuracy, ', 100 - round(top6a[2]/float(top6a[3])*100, 2), '% loss'
6 topics
75.278 % accuracy,  0.0 % loss
```

Σχήμα 4.59: Πλήθος topics από 3 εώς 6

```
In [205]: print '7 topics'
top7a = get_eval({0: 3, 1: 0, 2: 2, 3: 5, 4: 4, 5: 1, 6: 4},review_score[4],[])
print top7a[0],'% accuracy, ', 100 - round(float(top7a[2])/float(top7a[3])*100,2),'% loss'
7 topics
70.842 % accuracy,  0.0 % loss

In [210]: print '8 topics'
top8a = get_eval({0: 4, 1: 5, 2: 4, 3: 0, 4: 4, 5: 3, 6: 2, 7: 1},review_score[5],[])
print top8a[0],'% accuracy, ', 100 - round(float(top8a[2])/float(top8a[3])*100,2),'% loss'
8 topics
80.326 % accuracy,  0.0 % loss

In [215]: print '9 topics'
top9a = get_eval({0: 5, 1: 5, 2: 3, 3: 4, 4: 3, 5: 4, 6: 2, 7: 1, 8: 0},review_score[6],[])
print top9a[0],'% accuracy, ', 100 - round(float(top9a[2])/float(top9a[3])*100,2),'% loss'
9 topics
76.321 % accuracy,  0.0 % loss

In [217]: print '10 topics'
top10a = get_eval({0: 4, 1: 4, 2: 4, 3: 5, 4: 0, 5: 5, 6: 3, 7: 4, 8: 1, 9: 5},review_score[7],[])
print top10a[0],'% accuracy, ', 100 - round(float(top10a[2])/float(top10a[3])*100,2),'% loss'
10 topics
67.376 % accuracy,  0.0 % loss
```

Σχήμα 4.60: Πλήθος topics από 7 εώς 10

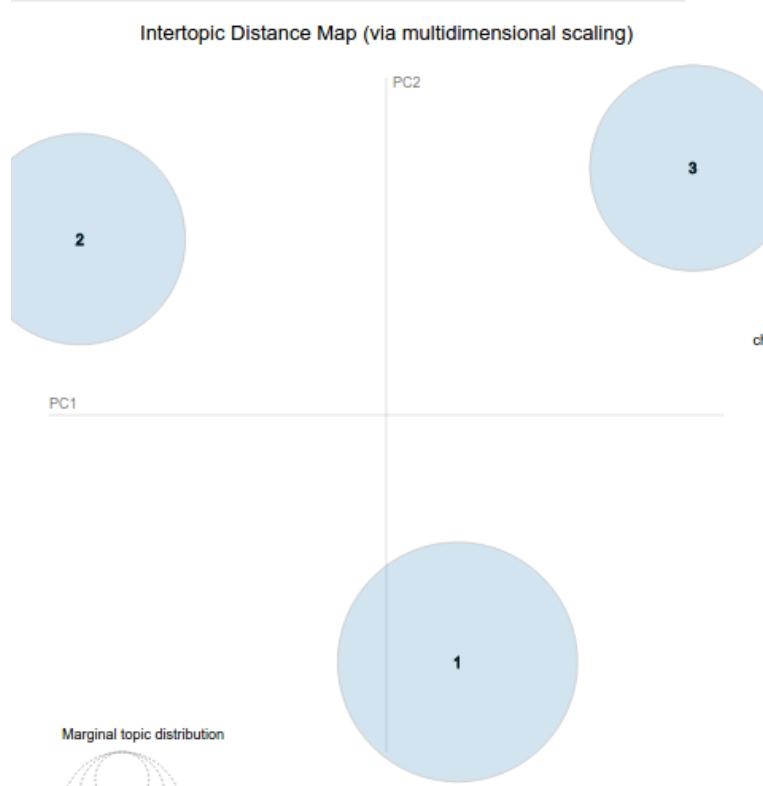
```
In [218]: print '11 topics'
top11a = get_eval({0: 5, 1: 1, 2: 5, 3: 4, 4: 3, 5: 0, 6: 1, 7: 0, 8: 4, 9: 2, 10: 5},review_score[8],[])
print top11a[0],'% accuracy, ', 100 - round(float(top11a[2])/float(top11a[3])*100,2),'% loss'
11 topics
82.943 % accuracy,  0.0 % loss

In [219]: print '12 topics'
top12a = get_eval({0: 1, 1: 4, 2: 3, 3: 1, 4: 1, 5: 5, 6: 5, 7: 1, 8: 2, 9: 5, 10: 0, 11: 4}, review_score[9],[])
print top12a[0],'% accuracy, ', 100 - round(float(top12a[2])/float(top12a[3])*100,2),'% loss'
12 topics
74.451 % accuracy,  0.0 % loss
```

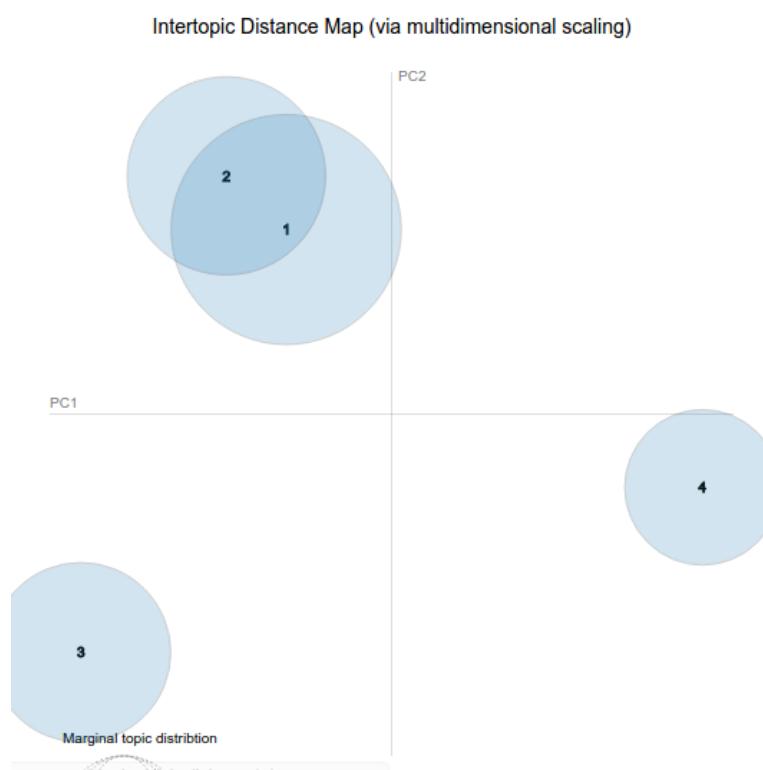
Σχήμα 4.61: Πλήθος topics από 11 εώς 12

Το ποσοστό απώλειας που φαίνεται παραπάνω είναι το πλήθος των reviews που δεν ελέχθησαν διότι υεωρήθηκε ότι δεν ανήκουν σε κάποιο από τα εξαγόμενα τοπικις. Έγινε προσπάθεια να μην υπάρξει καθόλου απώλεια τέτοιου είδους.

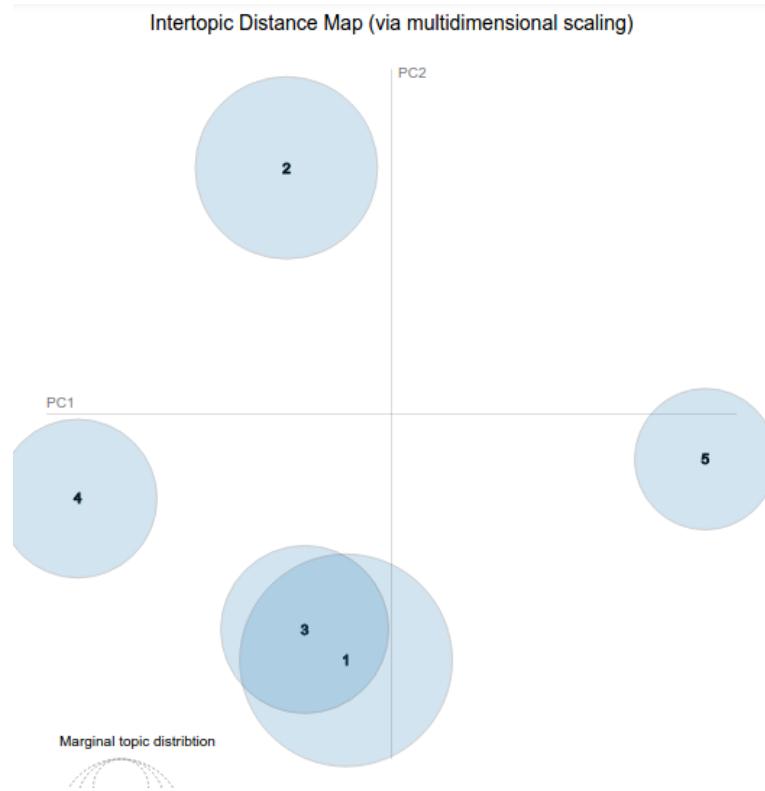
Ο χάρτης αποστάσεων (Intertopic Distance Map) των topics είναι ο παρακάτω και υλοποιείται από την βιβλιοθήκη pyLDAvis:



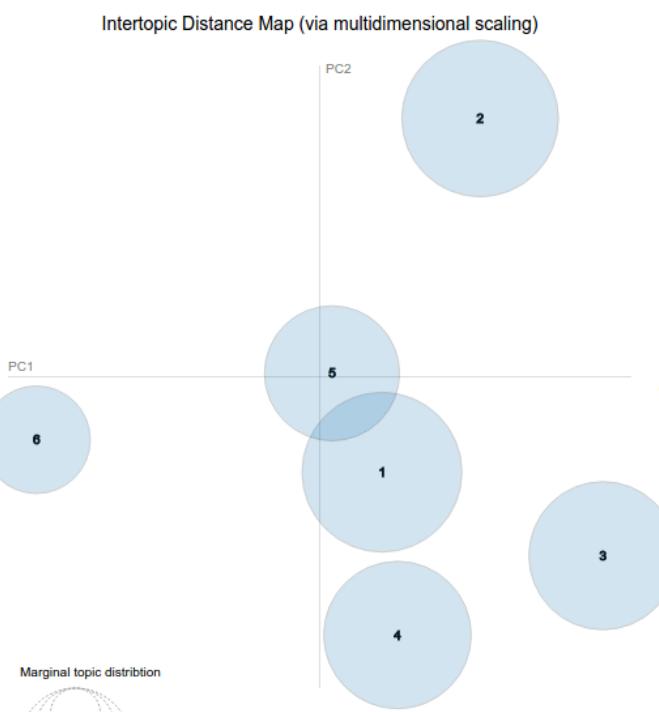
Σχήμα 4.62: Intertopic Distance Map for 3 topics



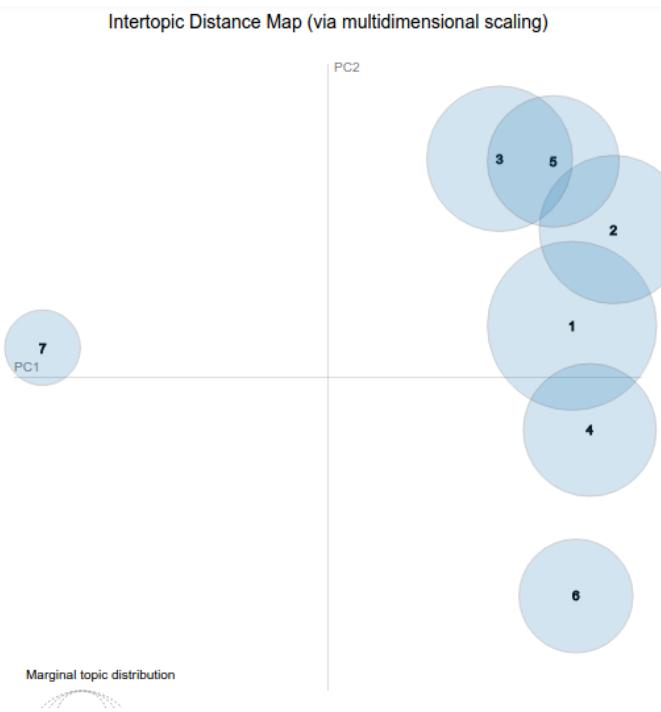
Σχήμα 4.63: Intertopic Distance Map for 4 topics



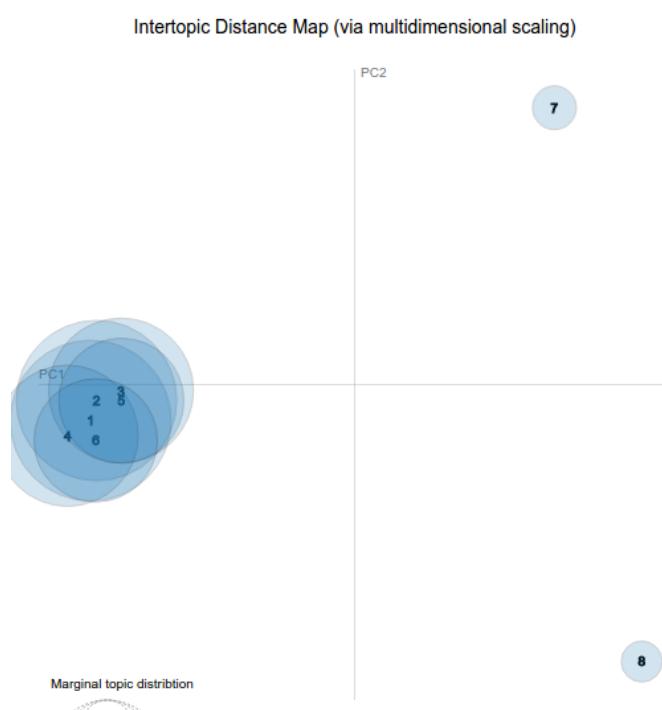
Σχήμα 4.64: Intertopic Distance Map for 5 topics



Σχήμα 4.65: Intertopic Distance Map for 6 topics



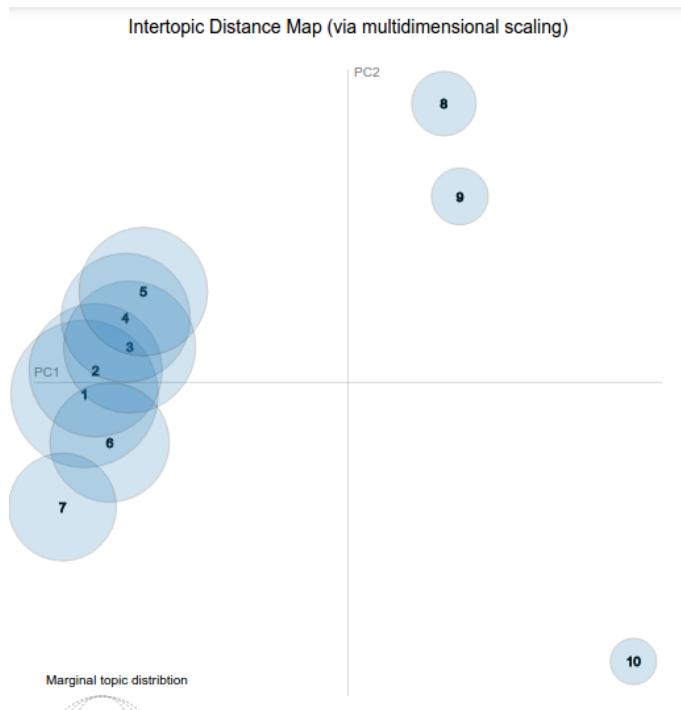
Σχήμα 4.66: Intertopic Distance Map for 7 topics



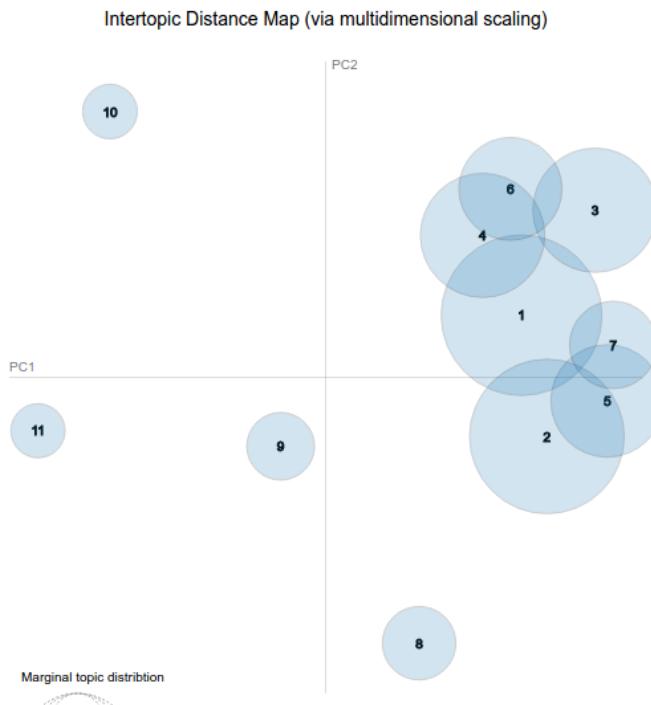
Σχήμα 4.67: Intertopic Distance Map for 8 topics



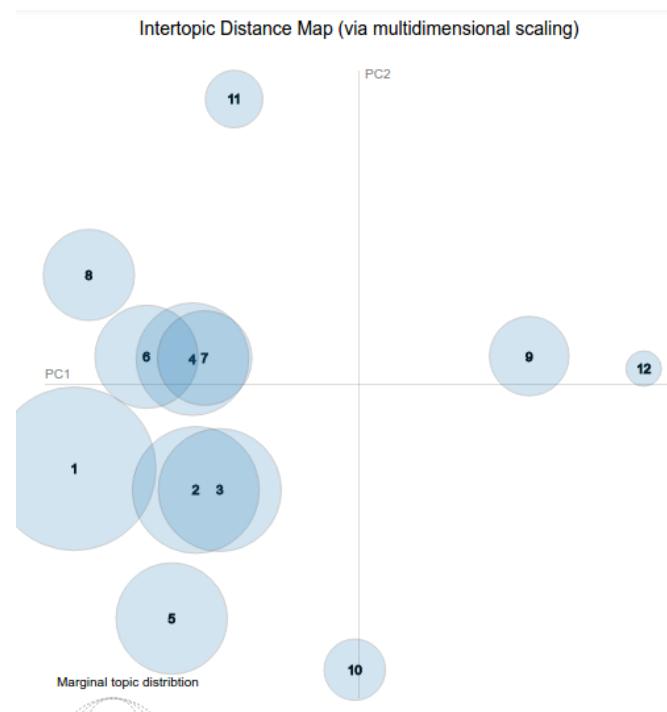
Σχήμα 4.68: Intertopic Distance Map for 9 topics



Σχήμα 4.69: Intertopic Distance Map for 10 topics



Σχήμα 4.70: Intertopic Distance Map for 11 topics



Σχήμα 4.71: Intertopic Distance Map for 12 topics

4.3.2 Συσταδοποίηση μέσω Πινάκων – Μασκών Συνάφειας

Ο πίνακας συνάφειας είναι ένας πίνακας, έστω A , μεγέθους $n \times n$ του οποίου οι γραμμές και οι στήλες αντιπροσωπεύουν τις κατηγορίες που έχει το dataset, άρα όπου n το πλήθος των κατηγοριών. Οι πληροφορίες που αποθηκεύει είναι ο λόγος του πλήθους των reviews που ενώ στην πραγματικότητα ανήκουν στην κατηγορία της γραμμής π.χ 1, αυτά ταξινομήθηκαν εσφαλμένα στην κατηγορία της στήλης x , όπου $x \neq 1$, προς το άθροισμα των εσφαλμένα ταξινομημένων reviews.

Η συγκεκριμένη μέθοδος συσταδοποίησης υλοποιήθηκε στο ίδιο dataset που εφαρμόσθηκε η ταξινόμηση της ενότητας 4.2.2.

Αρχικά, το dataset μοιράστηκε με τυχαίο τρόπο σε training και testing set σε αναλογία 80% και 20% αντίστοιχα. Εφαρμόσθηκαν και τα 5 μοντέλα που χρησιμοποιήθηκαν και στην ταξινόμηση της παραγράφου 4.2.2 από τα αποτελέσματα των οποίων πετυχαίνουμε και την συσταδοποίηση βάσει των λανθασμένων ταξινομήσεων όπως περιγράφηκε παραπάνω.

Η παρακάτω διαδικασία ακολουθείται για κάθε ένα από τα μοντέλα (Naive Bayes, Random Forest, Logistic Regression, K-nearest neighbors και Support vector machine).

Διατρέχουμε έναν βρόγχο για την κάθε κατηγορία που έχουμε (Rock, Blues, Folk κ.λπ.) και ακολουθούμε τα παρακάτω βήματα για να συμπληρωθεί ο πίνακας συνάφειας:

1. Παίρνουμε μόνο τα reviews από το testing set που ανήκουν στην υπό έλεγχο κατηγορία της τρέχουσας επανάληψης.
2. Υλοποιείται η εκπαίδευση του τρέχοντος μοντέλου στο training set.
3. Υπολογίζονται τα ποσοστά των λανθασμένων προβλέψεων ανά κατηγορία και συμπληρώνονται τα αποτελέσματα στην σχετική γραμμή του πίνακα. Το τρέχων βήμα επαναλαμβάνεται για κάθε κατηγορία.
4. Στο τέλος του βρόγχου, ο πίνακας θα είναι συμπληρωμένος και στις 18 γραμμές.

Παρακάτω φαίνεται η προετοιμασία του dataset και ο τυχαίος διαχωρισμός του σε training και testing sets.

Συσταδοποίηση μέσω Πινάκων - Μασκών Συνάφειας

```
In [69]: # Δημιουργία αντιγράφου από το dataset
map_df = df.copy()

# Μέγεθος dataset
print(map_df.shape)

# Υπολογισμός πλήθους instances το 80% του αρχικού
a = int(map_df.shape[0]*.8)

# Τυχαία επιλογή από τα id's για να αποτελέσουν το training set
index_list = np.random.choice(df.index.values, a, replace=False)

# Δημιουργία του training dataframe
map_df_train = map_df.ix[index_list]

# Δημιουργία του test dataframe
map_df_test = map_df.drop(map_df.index[[index_list]])
print(map_df_train.shape, map_df_test.shape)

# Ορισμός των X και y
X_map_train = map_df_train['review/text']
y_map_train = map_df_train.real_category_num

# Δημιουργία του διανυσματικού πίνακα
X_map_train_dtm = vect.fit_transform(X_map_train)
```

(17960, 6)
(14368, 6) (3592, 6)

Σχήμα 4.72: Τυχαίος διαχωρισμός σε training και testing sets

Ακολουθεί η δημιουργία των μοντέλων και η εκπαίδευση αυτών.

```
1 # Λίστα που αποθηκεύει τα μοντέλα ταξινόμησης
2 map_models = []
3 mdl = LogisticRegression()
4 map_models.append(mdl)
5 mdl = RandomForestClassifier()
6 map_models.append(mdl)
7 mdl = MultinomialNB()
8 map_models.append(mdl)
9 mdl = KNeighborsClassifier(n_neighbors=1)
10 map_models.append(mdl)
11 mdl = svm.SVC(kernel='linear', C=1)
12 map_models.append(mdl)
13 # Εκπαίδευση των μοντέλων
14 for m in map_models:
15     m.fit(X_map_train_dtm, y_map_train)
```

Αντιστοιχίζουμε τις κατηγορίες με αριθμούς (id's)

```
In [71]: # Αντιστοίχιση των labels με αριθμούς (id's)
inv_map = {v: k for k, v in tempDict.iteritems()}
inv_map

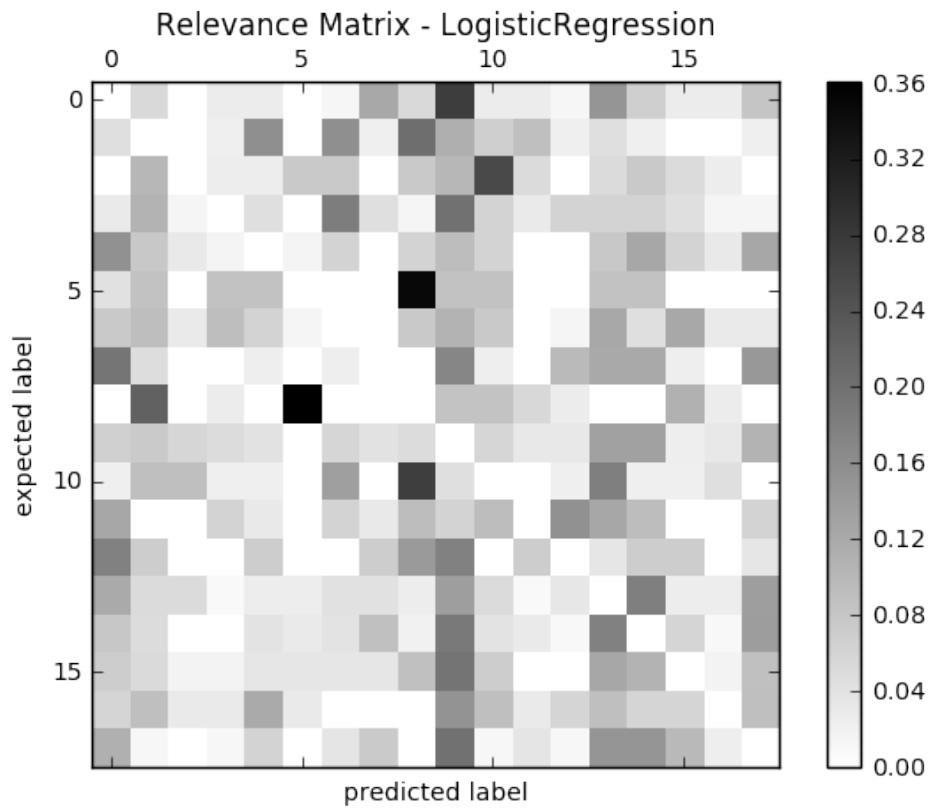
Out[71]: {0: 'Alternative Rock',
1: 'Christian',
2: 'Classical',
3: 'R&B',
4: 'Country',
5: "Children's Music",
6: 'Jazz',
7: 'Metal',
8: 'Special Interest',
9: 'Pop',
10: 'New Age',
11: 'Dance & Electronic',
12: 'Rap & Hip-Hop',
13: 'World Music',
14: 'Rock',
15: 'Blues',
16: 'Folk',
17: 'Classic Rock'}
```

Σχήμα 4.73: Αντιστοίχιση κατηγοριών με αριθμούς (id's)

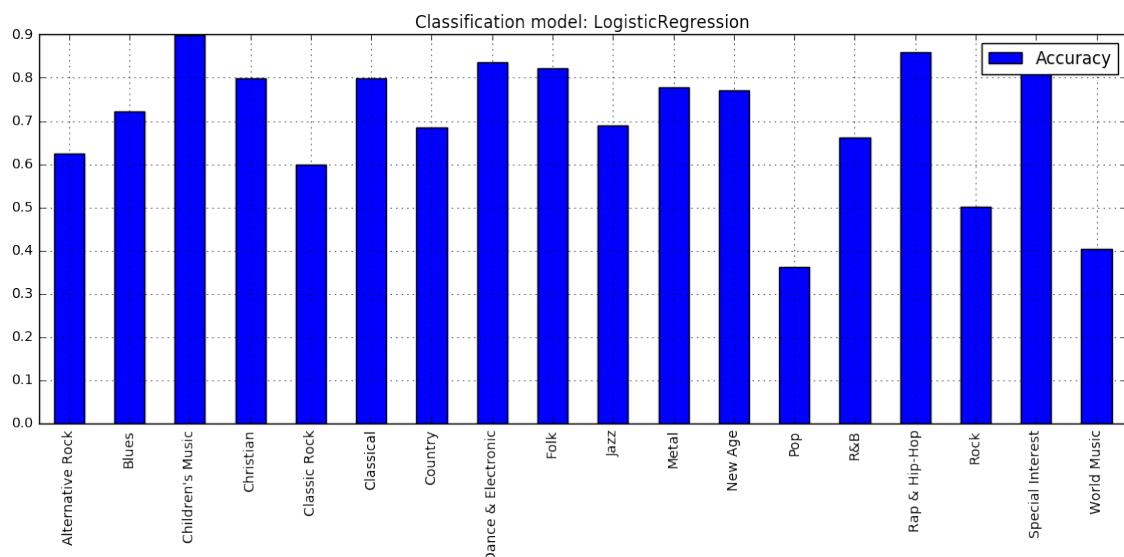
Ο βρόγχος με τον οποίον δημιουργείται ο πίνακας συνάφειας για κάθε μοντέλο φαίνεται στην ενότητα Α'.3, σελίδα 118.

Παραχάτω φαίνονται οι πίνακες συνάφειας και το διάγραμμα των ποσοστών επιτυχών προβλέψεων ανά κατηγορία (δηλαδή σε dataset που είχε μόνο reviews της εν λόγω κατηγορίας) και κατά μοντέλο:

Logistic Regression

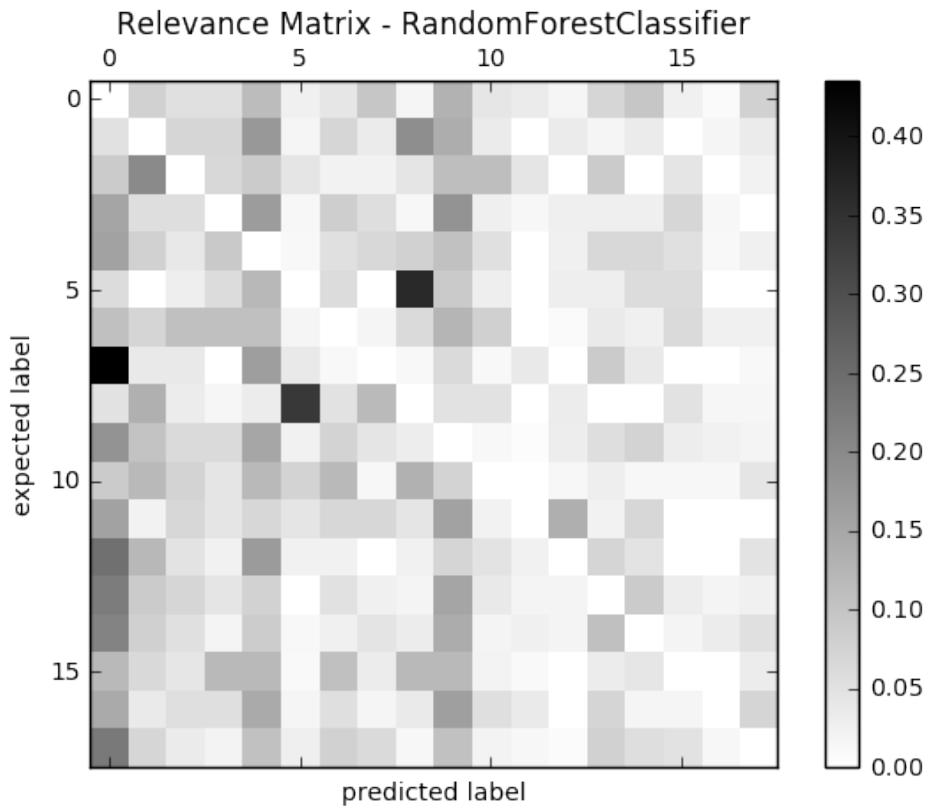


Σχήμα 4.74: Μάσκα συνάρειας για τον Logistic Regression

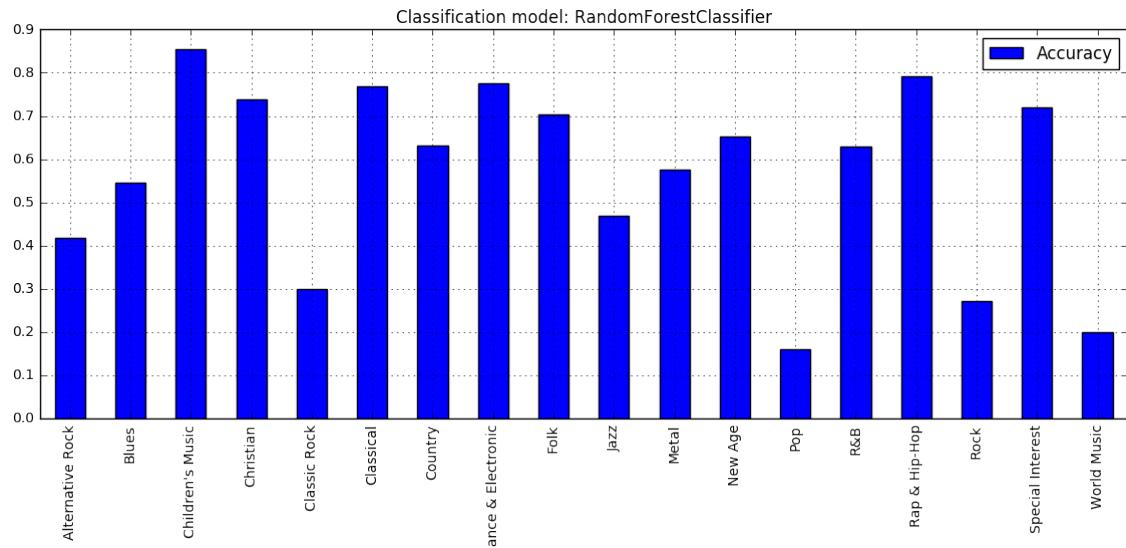


Σχήμα 4.75: Ποσοστά απόδοσης ανά κατηγορία για το Logistic Regression

Random Forest

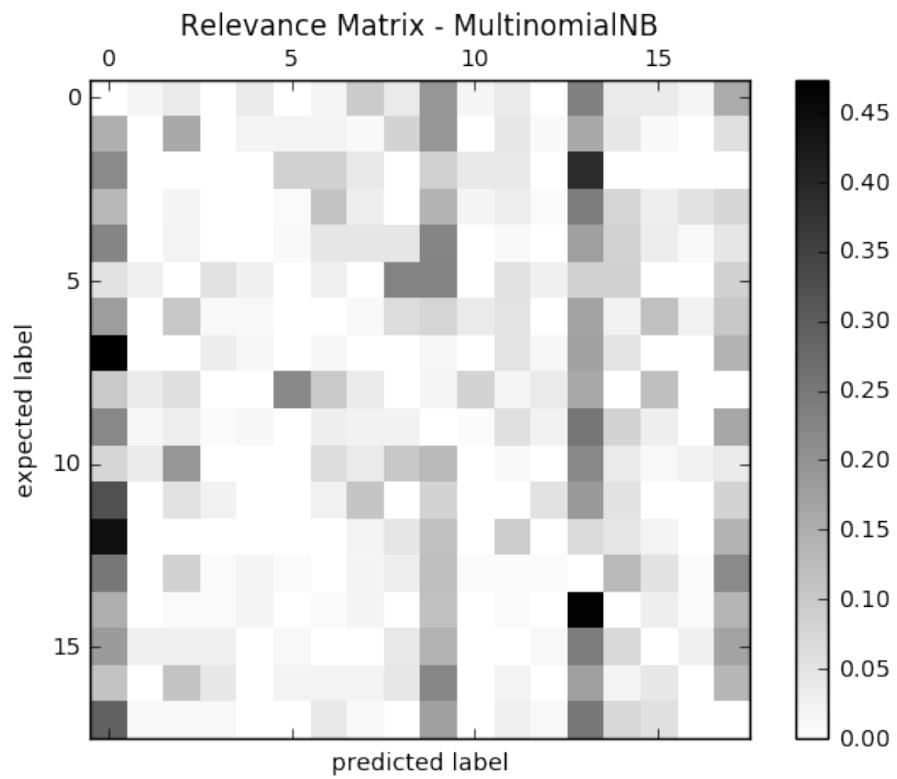


Σχήμα 4.76: Μάσκα συνάφειας για τον Random Forest

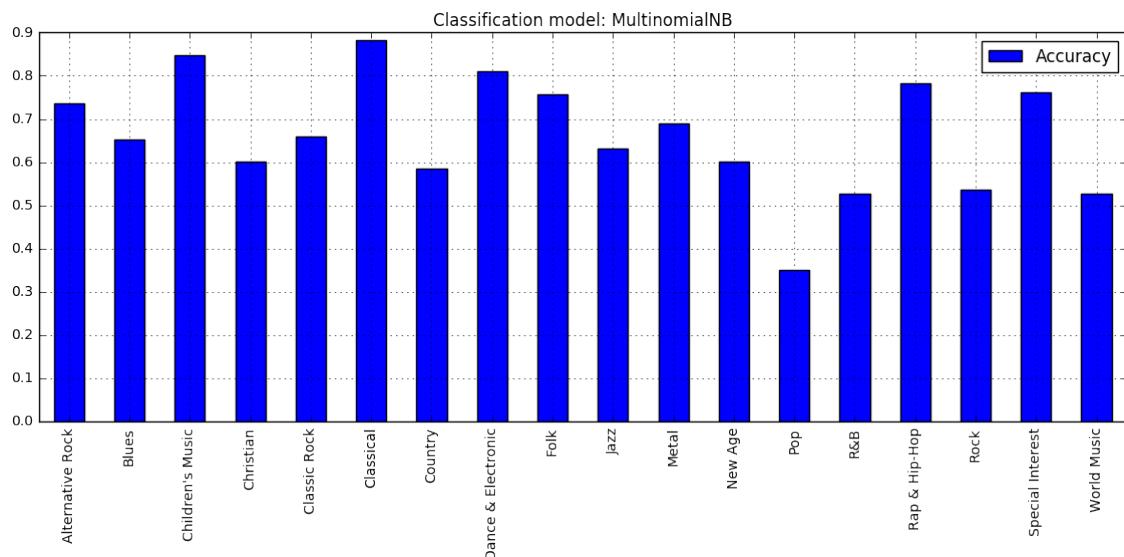


Σχήμα 4.77: Ποσοστά απόδοσης ανά κατηγορία για το Random Forest

Naive Bayes

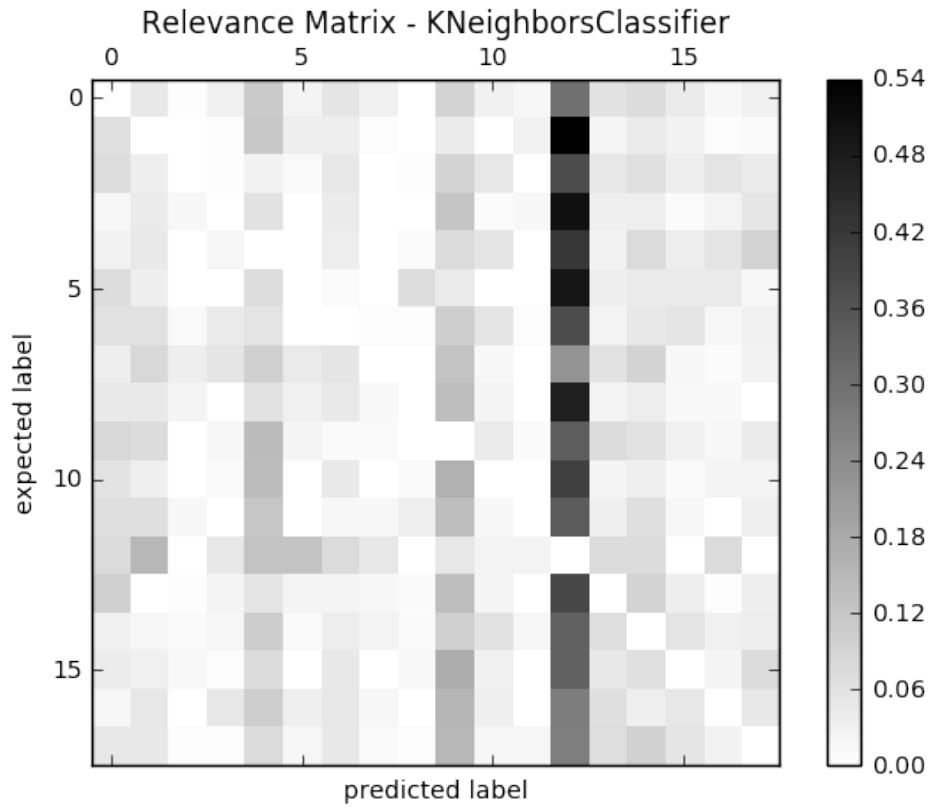


Σχήμα 4.78: Μάσκα συνάφειας για τον Naive Bayes

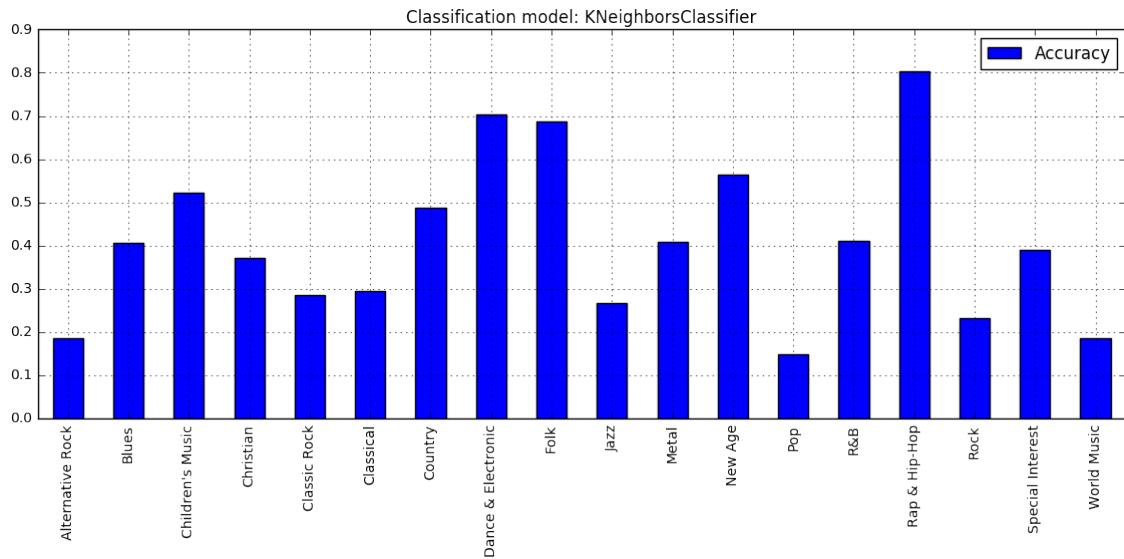


Σχήμα 4.79: Ποσοστά απόδοσης ανά κατηγορία για το Naive Bayes

K-nearest neighbors

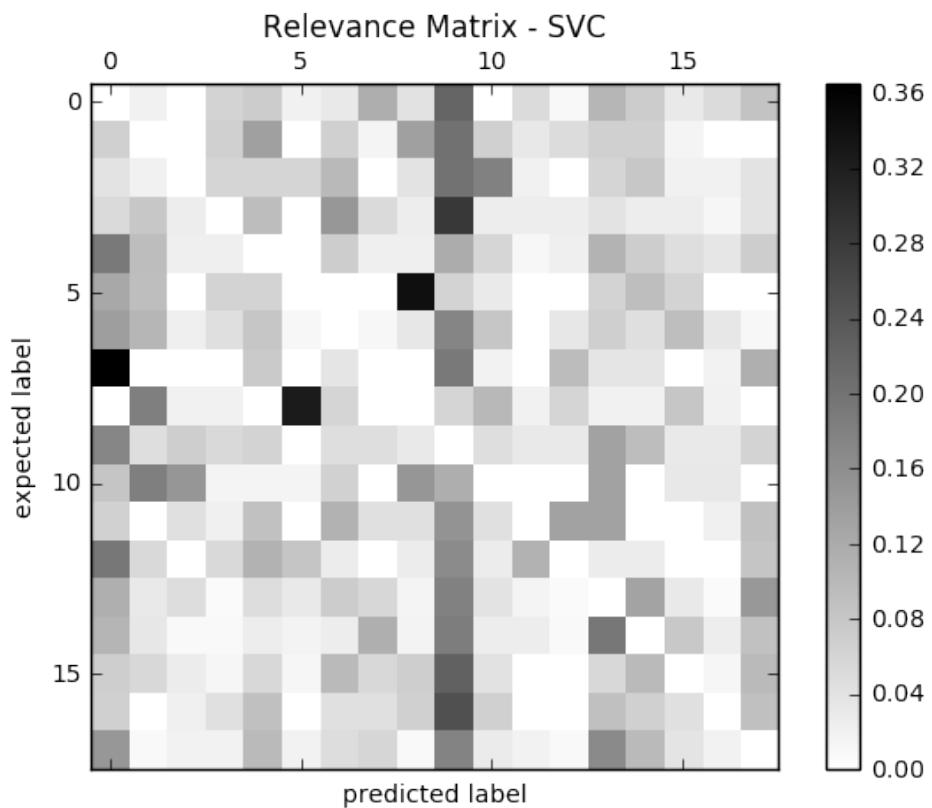


Σχήμα 4.80: Μάσκα συνάρτησης για τον K-nearest neighbors

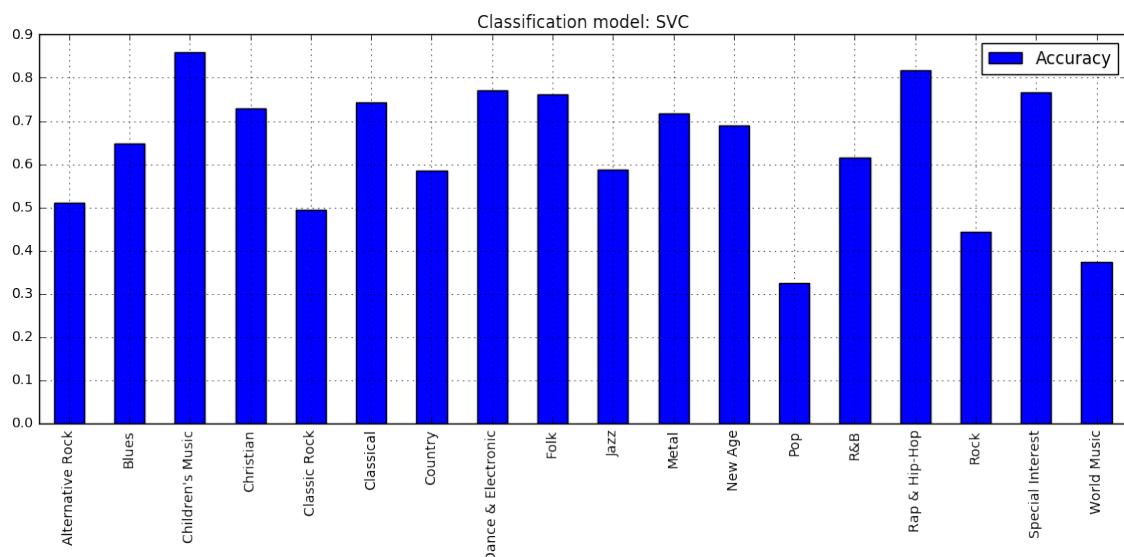


Σχήμα 4.81: Ποσοστά απόδοσης ανά κατηγορία για το K-nearest neighbors

Support vector machine



Σχήμα 4.82: Μάσκα συνάφειας για τον Support vector machine



Σχήμα 4.83: Ποσοστά απόδοσης ανά κατηγορία για το Support vector machine

Κεφάλαιο 5

Αξιολόγηση – Συμπεράσματα

Στο δεύτερο κεφάλαιο έγινε αναφορά στα σημαντικότερα σημεία της θεωρίας όπως η ανάκτηση της πληροφορίας, ο διανυσματικός χώρος, μέθοδοι μηχανικής μάθησης, έννοιες απόδοσης (ανάληση και ακρίβεια) και οι αλγόριθμοι που εφαρμόσθηκαν στα πλαίσια της εργασίας. Στο τρίτο κεφάλαιο παρουσιάσθηκαν οι πηγές δεδομένων που χρησιμοποιήθηκαν, η διαδικασία της απαιτούμενης προεπεξεργασίας των δεδομένων καθώς και οι διαδικασίες εμπλουτισμού του Amazon Movies Reviews Dataset με τα ground truth labels. Στο τέταρτο κεφάλαιο παρουσιάστηκαν δύο διαφορετικά προβλήματα, αυτό της ταξινόμησης και της συσταδοποίησης. Και στις δύο αυτές περιπτώσεις χρησιμοποιήσαμε τις πηγές δεδομένων που αναλύθηκαν εκτενώς στο τρίτο κεφάλαιο. Στο πρόβλημα της ταξινόμησης εφαρμόσαμε πέντε διαφορετικά μοντέλα (Naive Bayes, Random Forest, Logistic Regression, K-nearest neighbors και Support vector machine) ενώ για την υλοποίηση της συσταδοποίησης εφαρμόσθηκε η LDA και η μέθοδος μέσω των πινάκων–μασκών συνάφειας. Σε όλες τις περιπτώσεις παρατέθηκαν τα αποτελέσματα των αλγορίθμων καθώς και κάποιες γραφικές παραστάσεις όπου ήταν εφικτό.

5.1 Αξιολόγηση προβλημάτων ταξινόμησης

Παρακάτω ακολουθούν αξιολογήσεις των αποτελεσμάτων των μοντέλων που υλοποιήθηκαν ανά πηγή δεδομένων.

5.1.1 Αξιολόγηση στο “Six Categories of Amazon Product Reviews Dataset”

Ακολουθούν αξιολογήσεις ταξινόμησης ανά μοντέλο ταξινομητή. Λαμβάνονται υπόψη οι τιμές των μετρικών της ακρίβειας, ανάλησης καθώς και η τιμή της f1-score. Η f1-score μπορεί να ερμηνευτεί ως ένας σταθμισμένος μέσος όρος της ακρίβειας και της ανάλησης, όπου η βαθμολογία της κυμαίνεται μεταξύ 0 και 1. Υπολογίζεται από τον παρακάτω τύπο:

$$F_1 = 2 \cdot \frac{1}{\frac{1}{recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (5.1)$$

Το μοντέλο Naive Bayes

Το ποσοστό επιτυχίας του ταξινομητή Naive Bayes στο παραπάνω dataset είναι 91.69%. Στον πίνακα βλέπουμε τις τιμές της ανάκλησης, της ακρίβειας αλλά και της τιμής f1-score ανά κλάση/κατηγορία. Παρατηρούμε ότι η κλάση 4 έχει την χαμηλότερη τιμή ακρίβειας ενώ η 5 την χαμηλότερη ανάκληση. Η υψηλότερη τιμή της f1-score την πέτυχε στην κλάση 1. Επίσης παρατηρούμε ότι η κατανομή των δειγμάτων ανά κλάση είναι ιδιαίτερα ομαλή.

class	precision	recall	f1-score	support
0	0.96	0.92	0.94	931
1	0.99	0.94	0.97	893
2	0.87	0.97	0.91	944
3	0.98	0.88	0.93	914
4	0.80	0.92	0.85	898
5	0.95	0.87	0.91	884
avg / total	0.92	0.92	0.92	5464

Πίνακας 5.1: Ανάλυση απόδοσης του μοντέλου Naive Bayes

Κατά την αξιολόγηση του μοντέλου Naive Bayes με την μέθοδο 10-cross validation, πέτυχε 90.24% με τυπική απόκλιση $+/- 7.15\%$.

Το μοντέλο Logistic Regression

Ο ταξινομητής Logistic Regression πέτυχε ποσοστό ακριβείας 93.89% με την μέθοδο εκπαίδευσης training/testing set (80% – 20%). Στον πίνακα βλέπουμε ότι η κλάση 1 έχει την υψηλότερη τιμή ακρίβειας αλλά και ανάκλησης. Η δε κλάση 4 έχει τις χαμηλότερες τιμές και στις δύο μετρήσεις. Η υψηλότερη τιμή της f1-score την πέτυχε στην κλάση 1.

class	precision	recall	f1-score	support
0	0.96	0.96	0.96	931
1	0.97	0.98	0.98	893
2	0.95	0.92	0.94	944
3	0.95	0.96	0.95	914
4	0.88	0.89	0.89	898
5	0.93	0.92	0.92	884
avg / total	0.94	0.94	0.94	5464

Πίνακας 5.2: Ανάλυση απόδοσης του μοντέλου Logistic Regression

Κατά την αξιολόγηση του μοντέλου με την μέθοδο 10-cross validation, πέτυχε 92.70% με τυπική απόκλιση $+/- 6.65\%$.

Το μοντέλο Random Forest

Από τον πίνακα απόδοσης του ταξινομητή Random Forest βλέπουμε ότι η κλάση 1 έχει την υψηλότερη τιμή ακρίβειας αλλά και ανάκλησης. Η δε κλάση 4, όπως και πριν, έχει τις χαμηλότερες τιμές και στις δύο μετρήσεις. Η υψηλότερη τιμή της f1-score την πέτυχε στην κλάση 1. Κατά την μέθοδο εκπαίδευσης training/testing set (80% – 20%) πέτυχε ποσοστό ακρίβειας 86.84%.

class	precision	recall	f1-score	support
0	0.89	0.93	0.91	931
1	0.94	0.95	0.95	893
2	0.89	0.88	0.88	944
3	0.84	0.94	0.89	914
4	0.82	0.80	0.81	898
5	0.92	0.81	0.86	884
avg / total	0.89	0.88	0.88	5464

Πίνακας 5.3: Ανάλυση απόδοσης του μοντέλου Random Forest

Ο ταξινομητής Random Forest αξιολογήθηκε με την μέθοδο 10-cross validation και πέτυχε 86.21% με τυπική απόκλιση +/– 9.03%.

Το μοντέλο K-nearest neighbors

Ο ταξινομητής K-nearest neighbors πέτυχε ποσοστό ακρίβειας 56.06% με την μέθοδο εκπαίδευσης training/testing set (80% – 20%). Αυτή η απόδοση ήταν η χαμηλότερη από όλα τα μοντέλα που υλοποιήσαμε. Στον πίνακα βλέπουμε ότι οι χαμηλότερη τιμή ακρίβειας βρίσκεται στην κλάση 3 ενώ ανάκλησης στην κλάση 2. Η δε κλάση 5 έχει την υψηλότερη ακρίβεια ενώ η κλάση 3 την υψηλότερη ανάκληση. Η υψηλότερη τιμή της f1-score την πέτυχε στην κλάση 0.

class	precision	recall	f1-score	support
0	0.87	0.69	0.77	931
1	0.64	0.77	0.70	893
2	0.86	0.26	0.41	944
3	0.35	0.92	0.50	914
4	0.53	0.31	0.39	898
5	0.88	0.41	0.56	884
avg / total	0.69	0.56	0.55	5464

Πίνακας 5.4: Ανάλυση απόδοσης του μοντέλου K-nearest neighbors

Κατά την αξιολόγηση του μοντέλου με την μέθοδο 10-cross validation, πέτυχε 48.84% με τυπική απόκλιση +/– 26.99%.

Το μοντέλο Support vector machine

Από τον πίνακα απόδοσης του ταξινομητή Support vector machine βλέπουμε ότι η κλάση 1 έχει την υψηλότερη τιμή ακρίβειας αλλά και ανάλησης. Η δε κλάση 4, όπως και πριν, έχει τις χαμηλότερες τιμές και στις δύο μετρήσεις. Η υψηλότερη τιμή της f1-score την πέτυχε στην κλάση 1. Κατά την μέθοδο εκπαίδευσης training/testing set (80% – 20%) πέτυχε ποσοστό ακρίβειας 92.75%.

class	precision	recall	f1-score	support
0	0.94	0.95	0.95	931
1	0.97	0.98	0.97	893
2	0.93	0.92	0.92	944
3	0.93	0.94	0.94	914
4	0.87	0.88	0.87	898
5	0.92	0.90	0.91	884
avg / total	0.93	0.93	0.93	5464

Πίνακας 5.5: Ανάλυση απόδοσης του μοντέλου Support vector machine

Ο ταξινομητής Support vector machine αξιολογήθηκε με την μέθοδο 10-cross validation και πέτυχε 93.42% με τυπική απόκλιση +/– 5.98%.

5.1.2 Αξιολόγηση στο “Amazon Movies Reviews Dataset”

Ακολουθούν κατά τον ίδιο τρόπο με παραπάνω, αξιολογήσεις ταξινόμησης ανά μοντέλο ταξινομητή. Λαμβάνονται υπόψη οι τιμές των μετρικών της ακρίβειας, ανάλησης καιών και η τιμή της f1-score.

Το μοντέλο Naive Bayes

Το ποσοστό επιτυχίας του ταξινομητή Naive Bayes στο παραπάνω dataset είναι 68.26%. Στον πίνακα βλέπουμε τις τιμές της ανάλησης, της ακρίβειας αλλά και της τιμής f1-score ανά κλάση/κατηγορία. Παρατηρούμε ότι στις κλάσεις 9 και 13 έχει χαμηλές τιμές σε ακρίβεια και ανάληση. Επίσης βλέπουμε ότι στην κλάση 0, ενώ έχει χαμηλή τιμή ακρίβειας, η ανάληση είναι υψηλή. Αυτό σημαίνει πως έγινε σωστή εκτίμηση του ταξινομητή στα περισσότερα σχετικά έγγραφα, απλά ταξινομήθηκαν στην ίδια κλάση και άλλα πολλά έγγραφα που δεν ανήκουν σε αυτή. Η υψηλότερη τιμή της f1-score την πέτυχε στην κλάση 5. Επίσης παρατηρούμε ότι η κατανομή των δειγμάτων ανά κλάση είναι ιδιαίτερα ομαλή.

class	precision	recall	f1-score	support
0	0.38	0.81	0.51	192
1	0.91	0.68	0.78	198
2	0.74	0.89	0.81	214
3	0.94	0.64	0.76	199
4	0.88	0.68	0.77	192
5	0.89	0.87	0.88	196
6	0.78	0.61	0.69	197
7	0.89	0.64	0.74	231
8	0.81	0.76	0.78	189
9	0.43	0.29	0.35	214
10	0.93	0.61	0.74	211
11	0.88	0.85	0.86	201
12	0.92	0.85	0.88	202
13	0.29	0.43	0.34	189
14	0.41	0.61	0.49	182
15	0.78	0.69	0.73	180
16	0.95	0.73	0.83	196
17	0.50	0.67	0.57	209
avg / total	0.74	0.68	0.70	3592

Πίνακας 5.6: Ανάλυση απόδοσης του μοντέλου Naive Bayes

Κατά την αξιολόγηση του μοντέλου Naive Bayes με την μέθοδο 10-cross validation, πέτυχε 71.44% με τυπική απόκλιση +/− 1.18%.

To μοντέλο Logistic Regression

Ο ταξινομητής Logistic Regression πέτυχε ποσοστό ακριβείας 71.58% με την μέθοδο εκπαίδευσης training/testing set (80% – 20%). Στον πίνακα βλέπουμε ότι στις κλάσεις 9 και 13 έχει χαμηλές τιμές σε ακρίβεια αλλά και ανάκληση. Στις κλάσεις 5, 11, 12 και 16 έχει υψηλές τιμές απόδοσης. Η υψηλότερη τιμή της f1-score την πέτυχε στην κλάση 5.

class	precision	recall	f1-score	support
0	0.56	0.64	0.60	192
1	0.70	0.83	0.76	198
2	0.84	0.83	0.84	214
3	0.76	0.75	0.76	199
4	0.73	0.76	0.74	192
5	0.88	0.89	0.89	196
6	0.72	0.63	0.67	197
7	0.79	0.79	0.79	231
8	0.78	0.84	0.81	189
9	0.39	0.38	0.39	214
10	0.83	0.79	0.81	211
11	0.93	0.84	0.88	201
12	0.86	0.90	0.88	202
13	0.39	0.30	0.34	189
14	0.45	0.55	0.49	182
15	0.77	0.73	0.75	180
16	0.90	0.85	0.87	196
17	0.61	0.57	0.59	209
avg / total	0.72	0.72	0.71	3592

Πίνακας 5.7: Ανάλυση απόδοσης του μοντέλου Logistic Regression

Κατά την αξιολόγηση του μοντέλου με την μέθοδο 10-cross validation, πέτυχε 72.54% με τυπική απόκλιση +/ - 1.70%.

To μοντέλο Random Forest

Από τον πίνακα απόδοσης του ταξινομητή Random Forest βλέπουμε ότι οι κλάσεις 0, 9, 13, 14 και 17 έχουν ιδιαίτερα χαμηλές τιμές ακρίβειας αλλά και ανάλησης. Οι κλάσεις 2, 5, 11, 12 και 16 έχουν υψηλές τιμές. Παρατηρούμε στην κλάση 4 μία αρκετά υψηλή διαφορά μεταξύ ακρίβειας και ανάληση σε αντίθεση με τις υπόλοιπες κλάσεις. Η υψηλότερη τιμή της f1-score την πέτυχε στην κλάση 11. Κατά την μέθοδο εκπαίδευσης training/testing set (80% – 20%) πέτυχε ποσοστό ακριβείας 59.44%.

class	precision	recall	f1-score	support
0	0.29	0.44	0.35	192
1	0.52	0.76	0.62	198
2	0.70	0.75	0.72	214
3	0.61	0.75	0.67	199
4	0.42	0.67	0.52	192
5	0.77	0.87	0.82	196
6	0.61	0.48	0.53	197
7	0.71	0.65	0.67	231
8	0.69	0.73	0.71	189
9	0.25	0.20	0.22	214
10	0.79	0.63	0.70	211
11	0.92	0.77	0.84	201
12	0.81	0.84	0.83	202
13	0.27	0.13	0.18	189
14	0.36	0.34	0.35	182
15	0.65	0.60	0.62	180
16	0.91	0.76	0.83	196
17	0.50	0.33	0.40	209
avg / total	0.60	0.59	0.59	3592

Πίνακας 5.8: Ανάλυση απόδοσης του μοντέλου Random Forest

Ο ταξινομητής Random Forest αξιολογήθηκε με την μέθοδο 10-cross validation και πέτυχε 59.44% με τυπική απόκλιση $+/- 2.14\%$. Τίδια απόδοση με την μέθοδο training/testing set.

Το μοντέλο K-nearest neighbors

Ο ταξινομητής K-nearest neighbors πέτυχε ποσοστό ακριβείας 42.37% με την μέθοδο εκπαίδευσης training/testing set (80% – 20%). Αυτή η απόδοση ήταν, όπως και πιο πάνω, η χαμηλότερη από όλα τα μοντέλα που υλοποιήσαμε. Στον πίνακα βλέπουμε ότι οι κλάσεις με τις χαμηλές τιμές ακριβείας και ανόλησης είναι οι 0, 6, 9, 12, 13, 14 και 17. Οι κλάση 8 έχει την υψηλότερη ακριβεία ενώ η 12 την υψηλότερη ανάληση. Η υψηλότερη τιμή της f1-score την πέτυχαν οι κλάσεις 11 και 16 με ίδια απόδοση 0,74.

class	precision	recall	f1-score	support
0	0.38	0.27	0.31	192
1	0.56	0.40	0.47	198
2	0.80	0.31	0.45	214
3	0.75	0.50	0.60	199
4	0.40	0.52	0.45	192
5	0.81	0.53	0.64	196
6	0.43	0.31	0.36	197
7	0.73	0.42	0.54	231
8	0.91	0.33	0.48	189
9	0.14	0.21	0.17	214
10	0.71	0.56	0.63	211
11	0.82	0.67	0.74	201
12	0.15	0.80	0.26	202
13	0.27	0.13	0.18	189
14	0.29	0.23	0.26	182
15	0.64	0.48	0.55	180
16	0.83	0.67	0.74	196
17	0.43	0.25	0.32	209
avg / total	0.56	0.42	0.45	3592

Πίνακας 5.9: Ανάλυση απόδοσης του μοντέλου K-nearest neighbors

Κατά την αξιολόγηση του μοντέλου με την μέθοδο 10-cross validation, πέτυχε 40.57% με τυπική απόκλιση $+/- 2.41\%$.

To μοντέλο Support vector machine

Από τον πίνακα απόδοσης του ταξινομητή Support vector machine βλέπουμε ότι οι κλάσεις με υψηλές αποδόσεις ακριβείας και ανόλησης είναι οι 5, 8, 11, 12 και 16. Η δε κλάσεις 0, 9, 13, 14 και 17, έχουν τις χαμηλότερες τιμές και στις δύο μετρήσεις. Η υψηλότερη τιμή της f1-score την πέτυχε στην κλάση 5. Κατά την μέθοδο εκπαίδευσης training/testing set (80% – 20%) πέτυχε ποσοστό ακριβείας 65.59%.

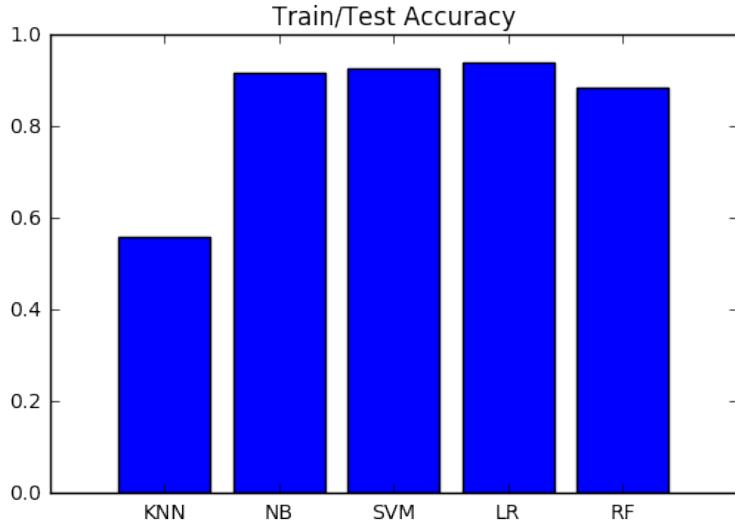
class	precision	recall	f1-score	support
0	0.40	0.56	0.47	192
1	0.66	0.76	0.71	198
2	0.80	0.78	0.79	214
3	0.71	0.74	0.73	199
4	0.58	0.67	0.62	192
5	0.87	0.89	0.88	196
6	0.63	0.56	0.60	197
7	0.72	0.71	0.71	231
8	0.78	0.80	0.79	189
9	0.30	0.32	0.31	214
10	0.82	0.73	0.77	211
11	0.89	0.78	0.83	201
12	0.86	0.86	0.86	202
13	0.35	0.28	0.31	189
14	0.42	0.45	0.43	182
15	0.72	0.66	0.69	180
16	0.89	0.80	0.84	196
17	0.53	0.44	0.48	209
avg / total	0.66	0.66	0.66	3592

Πίνακας 5.10: Ανάλυση απόδοσης του μοντέλου Support vector machine

Ο ταξινομητής Support vector machine αξιολογήθηκε με την μέθοδο 10-cross validation και πέτυχε 73.54% με τυπική απόκλιση + / - 1.66%.

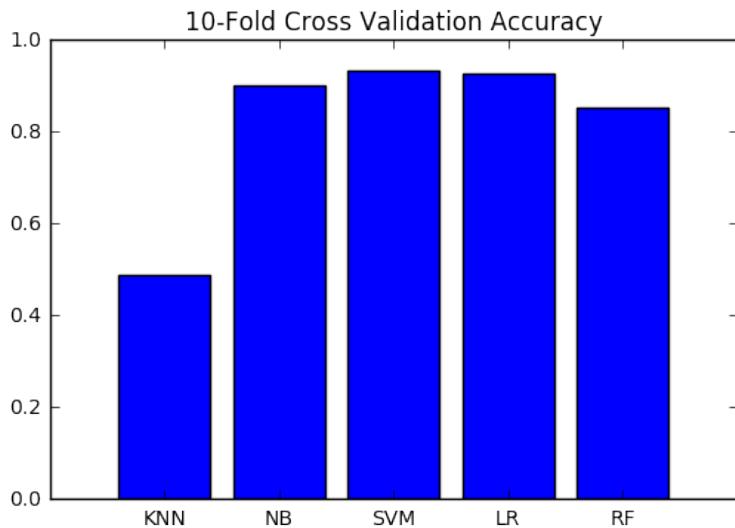
5.1.3 Συμπεράσματα Ταξινόμησης

Παρακάτω ακολουθούν διαγράμματα με τις αποδόσεις των μοντέλων ανά πηγή δεδομένων με πρώτο το “Six Categories of Amazon Product Reviews Dataset”.



Σχήμα 5.1: Αποδόσεις μοντέλων στο “Six Categories of Amazon Product Reviews Dataset”
- Training / Testing set

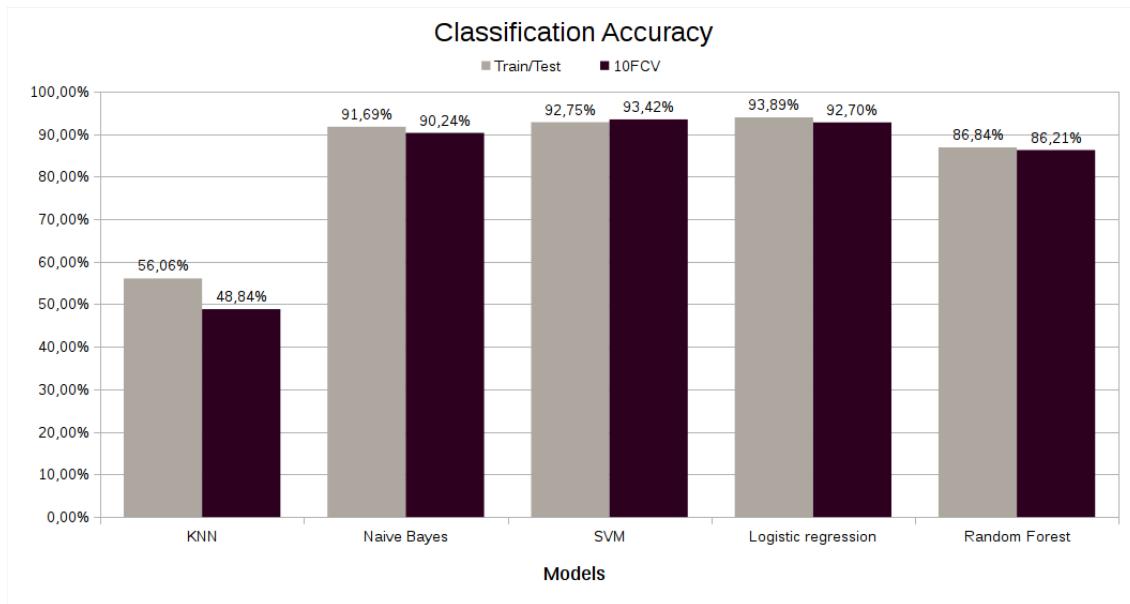
Βλέπουμε στο πρώτο dataset ότι όλα τα μοντέλα είχαν παρόμοια συμπεριφορά και απόδοση με εξαίρεση τον K-nearest neighbors ο οποίος είχε αισθητά χαμηλότερη απόδοση. Αυτό αποδεικνύει ότι δεν είναι όλα τα μοντέλα εξίσου κατάλληλα για όλα τα είδη των προς ταξινόμηση δεδομένων. Επίσης η επιλογή του $k = 1$ φαίνεται να προκάλεσε υπερεκπαίδευση (overfitting).



Σχήμα 5.2: Αποδόσεις μοντέλων στο “Six Categories of Amazon Product Reviews Dataset”
- 10-Fold Cross Validation

Στο παραπάνω γράφημα βλέπουμε τις αποδόσεις των μοντέλων στην αξιολόγηση τους με την μέθοδο k-cross validation, όπου $k = 10$. Αυτό που μπορεί να βγει ως συμπέρασμα είναι πως υπάρχει σχεδόν ίδια απόδοση με την μέθοδο training/testing set που σημαίνει πως η εκπαίδευση των ταξινομητών από τα training sets έγινε αμερόληπτα (unbiased training).

Ακολουθεί ένα συγκεντρωτικό γράφημα για καλύτερη οπτική αντίληψη των παραπάνω:



Σχήμα 5.3: Συγκεντρωτικό γράφημα αποδόσεων στο “Six Categories of Amazon Product Reviews Dataset” – TT vs 10-Fold Cross Validation

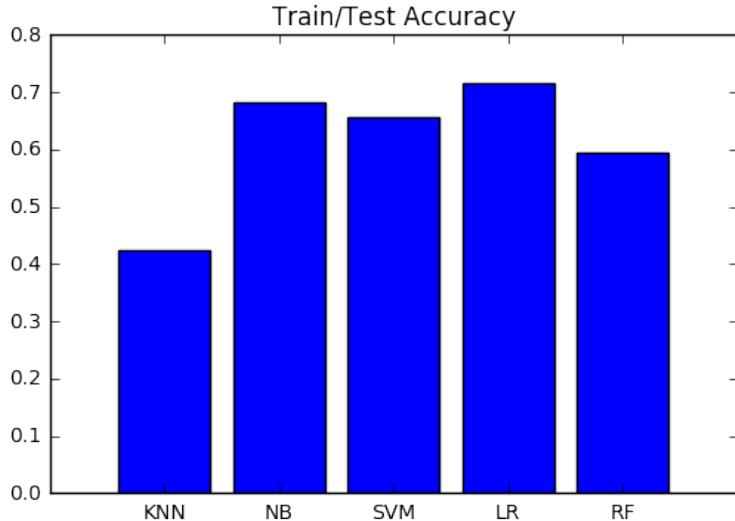
	Naive Bayes			Logistic Regression			Random Forest			K-NN			SVM		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
0. Mobilephone	0,96	0,92	0,94	0,96	0,96	0,96	0,89	0,93	0,91	0,87	0,69	0,77	0,94	0,95	0,95
1. Cameras	0,99	0,94	0,97	0,97	0,98	0,98	0,94	0,95	0,95	0,64	0,77	0,7	0,97	0,98	0,97
2. Video Surveillance	0,87	0,97	0,91	0,95	0,92	0,94	0,89	0,88	0,88	0,86	0,26	0,41	0,93	0,92	0,92
3. Tvs	0,98	0,88	0,93	0,95	0,96	0,95	0,84	0,94	0,89	0,35	0,92	0,5	0,93	0,94	0,94
4. Tablets	0,8	0,92	0,85	0,88	0,89	0,89	0,82	0,8	0,81	0,53	0,31	0,39	0,87	0,88	0,87
5. Laptops	0,95	0,87	0,91	0,93	0,92	0,92	0,92	0,81	0,86	0,88	0,41	0,56	0,92	0,9	0,91
Average	0,93	0,92	0,92	0,94	0,94	0,94	0,88	0,89	0,88	0,69	0,56	0,56	0,93	0,93	0,93

P: Precision, R: Recall, F: F_1 – Score

Σχήμα 5.4: Συγκεντρωτικές αποδόσεις ανά μοντέλο στο “Six Categories of Amazon Product Reviews Dataset”

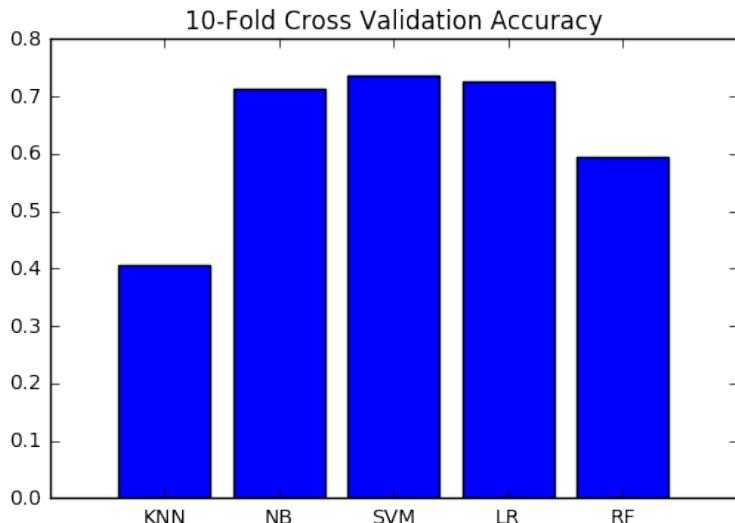
Στην παραπάνω εικόνα φαίνονται οι τιμές ακρίβειας (P), ανάκλησης (R) και F1-Score (F) για κάθε μοντέλο και για κάθε κλάση. Βλέπουμε μία σχετική ομαδοποίηση και κοινή “συμπεριφορά” των μοντέλων στις κλάσεις. Η μόνη διαφοροποίηση έρχεται από τον K-nearest neighbors του οποίου η απόδοση είναι συγχριτικά χαμηλότερη και με μεγαλύτερη διακύμανση μεταξύ των κλάσεων.

Ακολουθούν τα διαγράμματα με τις αποδόσεις των μοντέλων στο “Amazon Movies Reviews Dataset”



Σχήμα 5.5: Αποδόσεις μοντέλων στο “Amazon Movies Reviews Dataset” – Training/Testing set

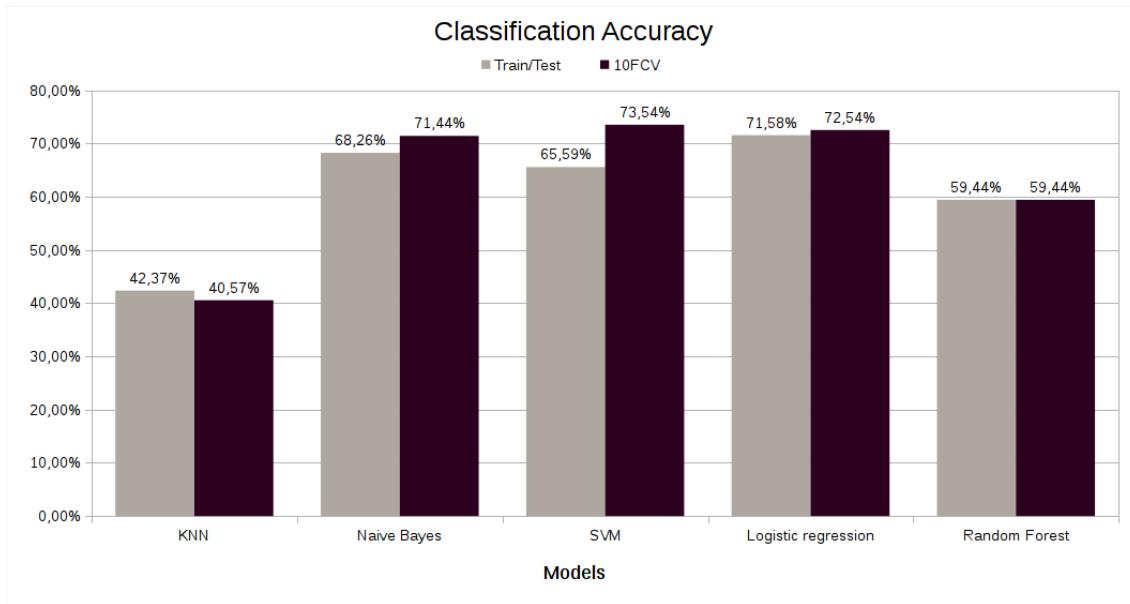
Βλέπουμε ότι και στο δεύτερο dataset, όλα τα μοντέλα έχουν σχεδόν παρόμοια συμπεριφορά και απόδοση με εξαίρεση και πάλι τον K-nearest neighbors ο οποίος έχει αισθητά χαμηλότερη απόδοση.



Σχήμα 5.6: Αποδόσεις μοντέλων στο “Amazon Movies Reviews Dataset” – 10-Fold Cross Validation

Στο παραπάνω γράφημα βλέπουμε τις αποδόσεις των μοντέλων στην αξιολόγηση τους με την μέθοδο k-cross validation, όπου $k = 10$. Αυτό που μπορεί να βγει ως συμπέρασμα είναι πως υπάρχει παρόμοια απόδοση με την μέθοδο training/testing set που σημαίνει ότι εκπαίδευση των ταξινομητών από τα training sets έγινε και πάλι αμερόληπτα (unbiased training).

Ακολουθεί ένα συγκεντρωτικό γράφημα για καλύτερη οπτική αντίληψη των παραπάνω:



Σχήμα 5.7: Συγκεντρωτικό γράφημα αποδόσεων στο “Amazon Movies Reviews Dataset” – TT vs 10-Fold Cross Validation

		Naive Bayes			Logistic Regression			Random Forest			K-NN			SVM		
		P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
0. Alternative Rock		0,38	0,81	0,51	0,56	0,64	0,6	0,29	0,44	0,35	0,38	0,27	0,31	0,4	0,56	0,47
1. Christian		0,91	0,68	0,78	0,7	0,83	0,76	0,52	0,76	0,62	0,56	0,4	0,47	0,66	0,76	0,71
2. Classical		0,74	0,89	0,81	0,84	0,83	0,84	0,7	0,75	0,72	0,8	0,31	0,45	0,8	0,78	0,79
3. R&B		0,94	0,64	0,76	0,76	0,75	0,76	0,61	0,75	0,67	0,75	0,5	0,6	0,71	0,74	0,73
4. Country		0,88	0,68	0,77	0,73	0,76	0,74	0,42	0,67	0,52	0,4	0,52	0,45	0,58	0,67	0,62
5. Children's Music		0,89	0,87	0,88	0,88	0,89	0,89	0,77	0,87	0,82	0,81	0,53	0,64	0,87	0,89	0,88
6. Jazz		0,78	0,61	0,69	0,72	0,63	0,67	0,61	0,48	0,53	0,43	0,31	0,36	0,63	0,56	0,6
7. Metal		0,89	0,64	0,74	0,79	0,79	0,79	0,71	0,65	0,67	0,73	0,42	0,54	0,72	0,71	0,71
8. Special Interest		0,81	0,76	0,78	0,78	0,84	0,81	0,69	0,73	0,71	0,91	0,33	0,48	0,78	0,8	0,79
9. Pop		0,43	0,29	0,35	0,39	0,38	0,39	0,25	0,2	0,22	0,14	0,21	0,17	0,3	0,32	0,31
10. New Age		0,93	0,61	0,74	0,83	0,79	0,81	0,79	0,63	0,7	0,71	0,56	0,63	0,82	0,73	0,77
11. Dance & Electronic		0,88	0,85	0,86	0,93	0,84	0,88	0,92	0,77	0,84	0,82	0,67	0,74	0,89	0,78	0,83
12. Rap & Hip-Hop		0,92	0,85	0,88	0,86	0,9	0,88	0,81	0,84	0,83	0,15	0,8	0,26	0,86	0,86	0,86
13. World Music		0,29	0,43	0,34	0,39	0,3	0,34	0,27	0,13	0,18	0,27	0,13	0,18	0,35	0,28	0,31
14. Rock		0,41	0,61	0,49	0,45	0,55	0,49	0,36	0,34	0,35	0,29	0,23	0,26	0,42	0,45	0,43
15. Blues		0,78	0,69	0,73	0,77	0,73	0,75	0,65	0,6	0,62	0,64	0,48	0,55	0,72	0,66	0,69
16. Folk		0,95	0,73	0,83	0,9	0,85	0,87	0,91	0,76	0,83	0,83	0,67	0,74	0,89	0,8	0,84
17. Classic Rock		0,5	0,67	0,57	0,61	0,57	0,59	0,5	0,33	0,4	0,43	0,25	0,32	0,53	0,44	0,48
Average		0,74	0,68	0,70	0,72	0,72	0,71	0,60	0,59	0,59	0,56	0,42	0,45	0,66	0,66	0,66

P: Precision, R: Recall, F: F_1 – Score

Σχήμα 5.8: Συγκεντρωτικές αποδόσεις ανά μοντέλο στο “Amazon Movies Reviews Dataset”

Στον παραπάνω πίνακα, όπως και παραπάνω, φαίνονται οι τιμές ακρίβειας (P), ανάχλησης (R) και F_1 -Score (F) για κάθε μοντέλο και για κάθε κλάση. Βλέπουμε εξίσου μία σχετική ομαδοποίηση και κοινή “συμπεριφορά των μοντέλων” στις κλάσεις. Και πάλι ο K-nearest neighbors διαφέρει σημαντικά από την μέση απόδοση των υπολοίπων αλγορίθμων, με χαμη-

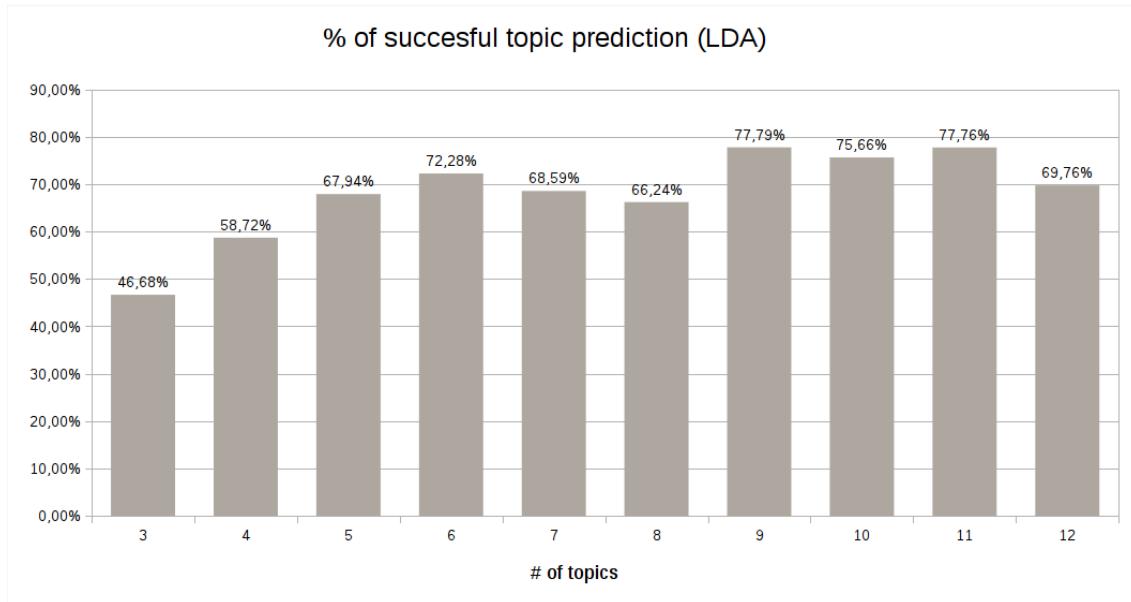
λότερα ποσοστά απόδοσης.

5.2 Αξιολόγηση προβλημάτων συσταδοποίησης

Παρακάτω ακολουθούν αξιολογήσεις των αποτελεσμάτων των μοντέλων που υλοποιήθηκαν ανά μέθοδο/μοντέλο συσταδοποίησης.

5.2.1 Το μοντέλο Latent Dirichlet allocation (LDA)

Ακολουθεί διάγραμμα με τις αποδόσεις της LDA ανά πλήθος εξαγόμενων topics. Η πηγή δεδομένων που χρησιμοποιήθηκε είναι η “Six Categories of Amazon Product Reviews”.



Σχήμα 5.9: Αποδόσεις της LDA ανά πλήθος εξαγόμενων topics

Βλέπουμε ότι η υψηλότερη απόδοση επιτεύχθηκε στην εξαγωγή εννέα topics ενώ η χαμηλότερη στα τρία topics. Υπενθυμίζεται ότι οι αρχικές κατηγορίες των προϊόντων είναι έξι. Παρατηρούμε πως όσο το πλήθος των topics πλησιάζει στον αριθμό 6, τόσο αυξάνεται η απόδοση.

Η αντικειμενική αντιστοίχιση των topics με τις αρχικές κατηγορίες κρύβει μεγάλη πιθανότητα σφάλματος με αποτέλεσμα ίσως κάποιο χαμηλό ποσοστό απόδοσης. Επίσης η διαφοροποίηση μεταξύ των κατηγοριών δεν ήταν και τόσο μεγάλη από την άποψη ότι πρόκειται για ηλεκτρικές συσκευές. Είναι λογικό και αναμενόμενο να μοιάζουν οι κριτικές μεταξύ διαφορετικών ηλεκτρικών συσκευών μιας και μοιράζονται πολλές ιδιότητες.

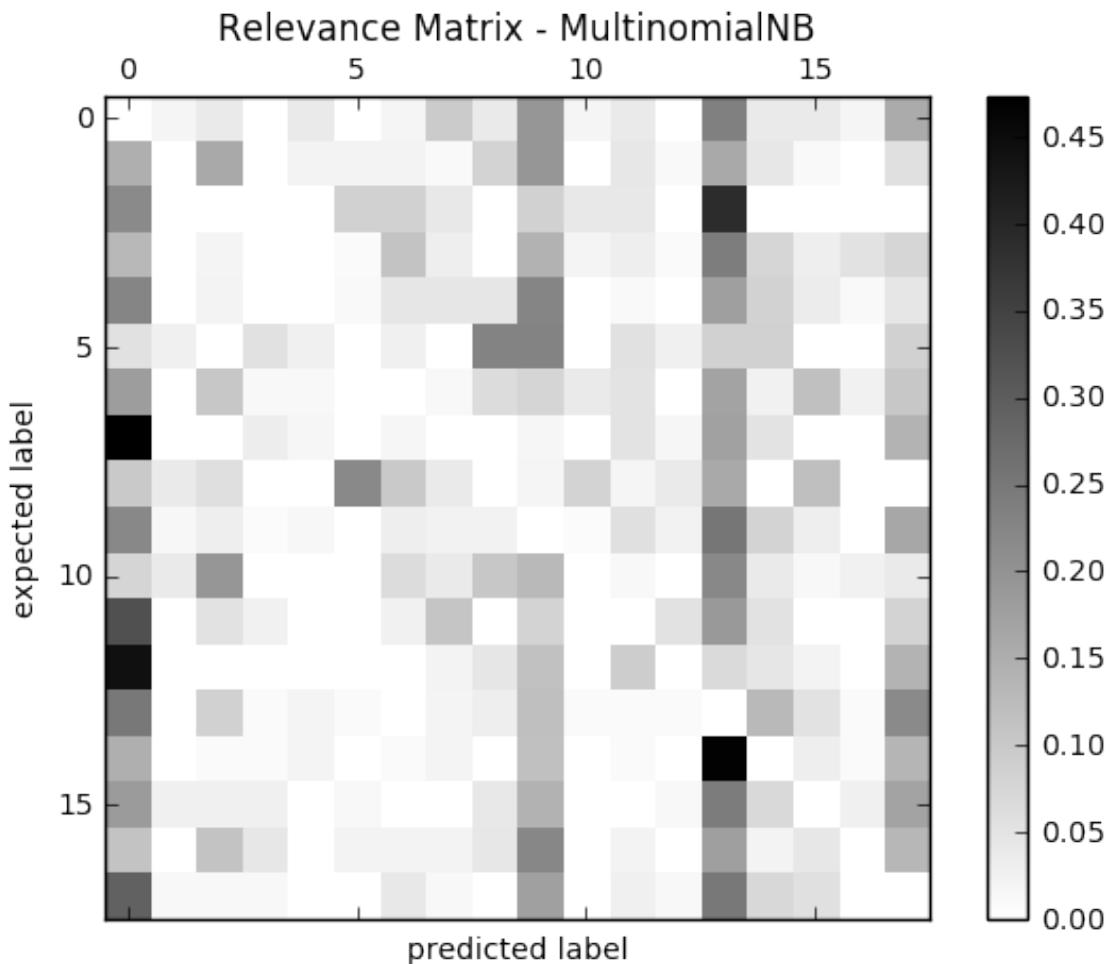
5.2.2 Η μέθοδος Πινάκων/Μασκών Συνάφειας

Θα δούμε τις μάσκες συνάφειας που προέκυψαν από την υλοποίηση των ταξινομητών και βάσει των οποίων εφαρμόσθηκε η συσταδοποίηση των εγγραφών. Θα έλεγα ότι αυτό μας

δείχνει ότι αυτές οι κλάσεις έχουν αρκετές ομοιότητες (βάσει των χαρακτηριστικών τους) και δεν μπορούν εύκολα να διακριθούν μεταξύ τους (ιδιαίτερα αν η πρόβλεψη αφορά την κλάση 0 και την 9). Τώρα όσο πιο σκούρο είναι το χρώμα σε ένα misclassified instance, τόσο πιο χοντά είναι η predicted class σε μία 2η, 3η, κλπ. επιλογή (πέραν αυτής που ανήκει).

Το μοντέλο Naive Bayes

Ο πίνακας/μάσκα συνάφειας που δημιουργήθηκε είναι ο παρακάτω:

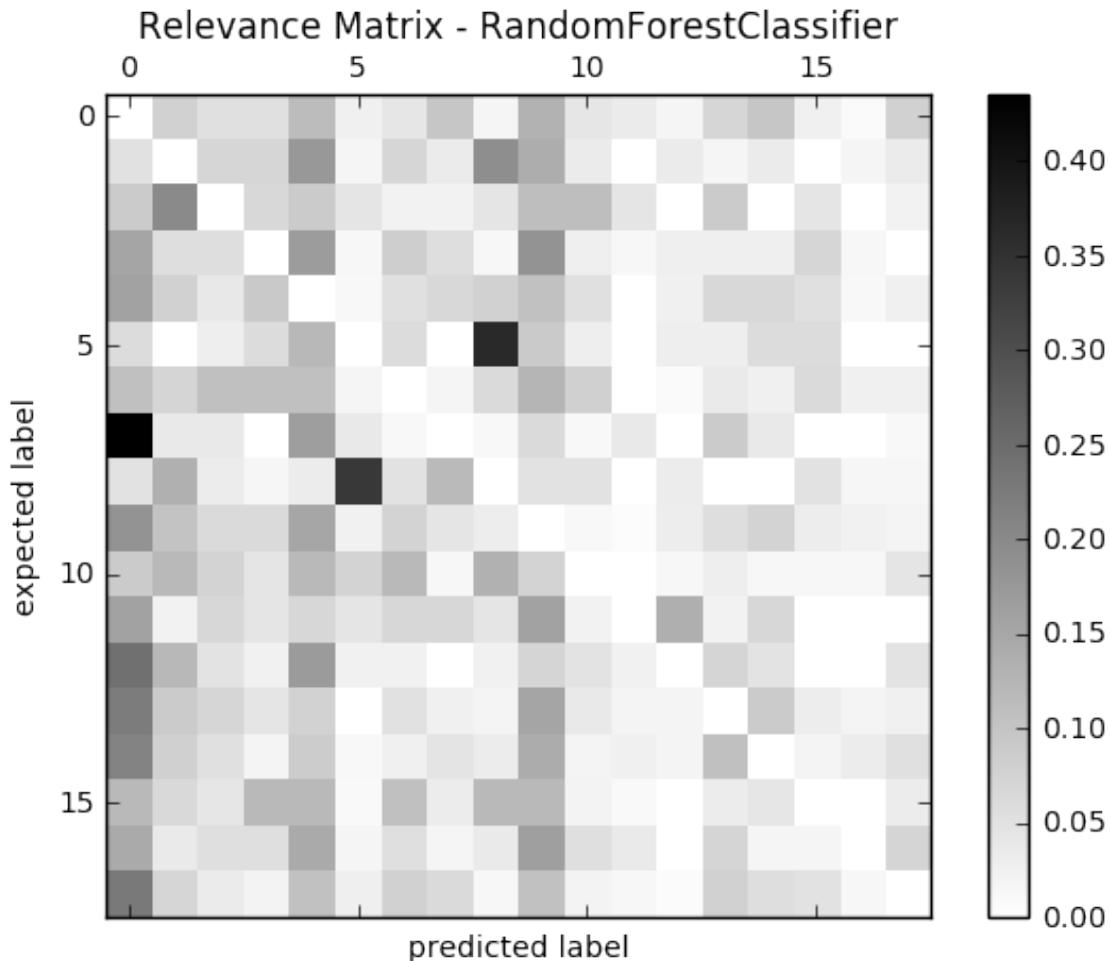


Σχήμα 5.10: Πίνακας/μάσκα συνάφειας του Naive Bayes

Παρατηρούμε ότι οι κλάσεις 0, 9, 13 και 17 έχουν αποπροσανατολίσει το μοντέλο με αποτέλεσμα να ταξινομήσει εσφαλμένα αρκετές εγγραφές σε αυτές τις κατηγορίες. Θα μπορούσαμε να συμπεράνουμε πως οι παραπάνω κλάσεις έχουν κάποια κοινά χαρακτηριστικά και ιδιότητες βάσει των οποίων προέκυψε η παραπάνω ομαδοποίηση. Παρατηρούμε επίσης ότι τα περισσότερο σκούρα σημεία ανήκουν σε αυτές τις ίδιες κλάσεις γεγονός που ενισχύει τον παραπάνω ισχυρισμό (της ομοιότητας των κλάσεων).

Το μοντέλο Random Forest

Ακολουθεί ο πίνακας/μάσκα συνάφειας που δημιουργήθηκε:

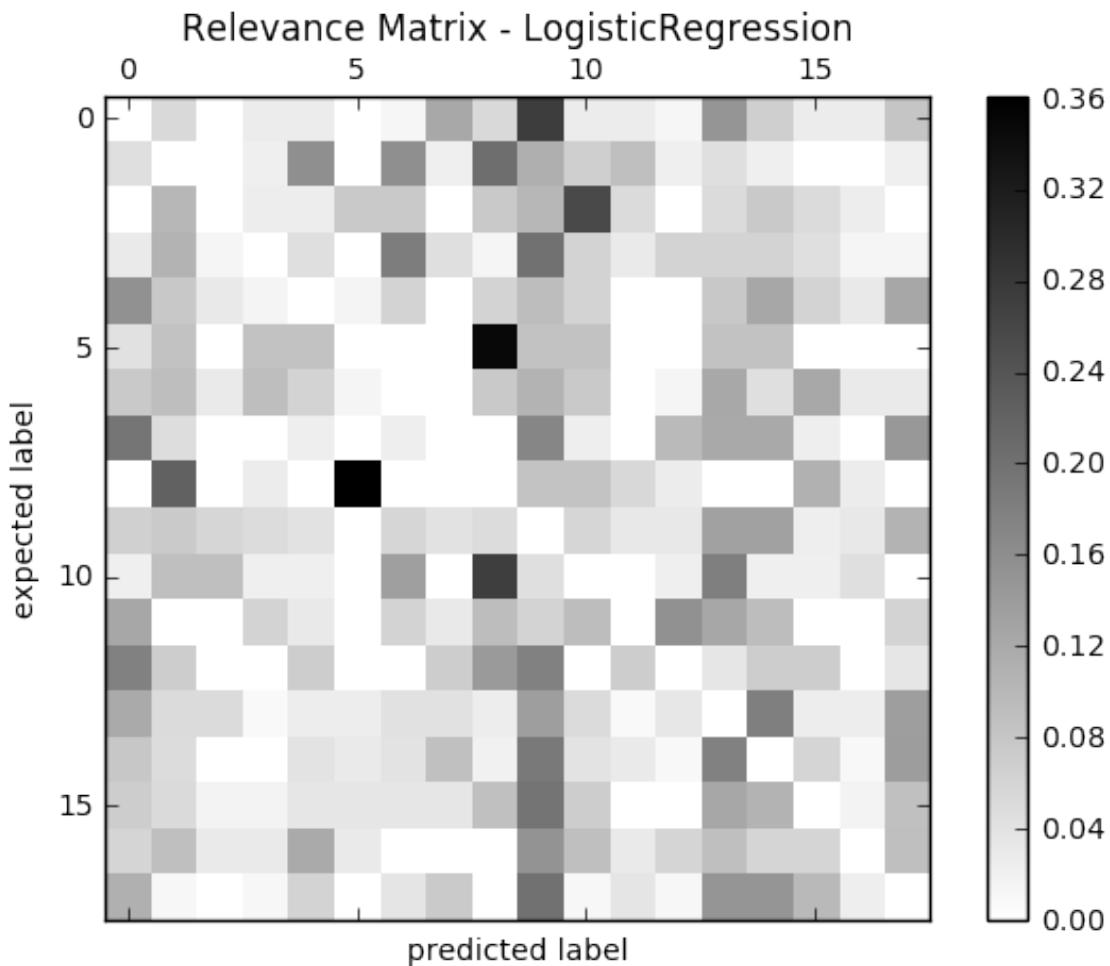


Σχήμα 5.11: Πίνακας/μάσκα συνάφειας του Random Forest

Παρατηρούμε ότι οι κλάσεις 0, 4 και 9 έχουν συλλέξει τα περισσότερα λανθασμένα ταξινομημένα instances. Επίσης φαίνεται ότι η πλειοψηφία των εγγραφών έχουν ταξινομηθεί στις πρώτες 9 κλάσεις. Από την κλάση 10 μέχρι και την τελευταία, όλο και λιγότερες εγγραφές ταξινομήθηκαν. Συμπεραίνουμε πως οι πρώτες κλάσεις, σύμφωνα με τον Random Forest, έχουν κάποια κοινά χαρακτηριστικά και ιδιότητες. Παρατηρούμε επίσης ότι τα περισσότερο σκούρα σημεία ανήκουν σε κλάσεις με αριθμό μικρότερο του 8.

To μοντέλο Logistic Regression

Ο πίνακας/μάσκα συνάφειας που δημιουργήθηκε είναι ο παρακάτω:

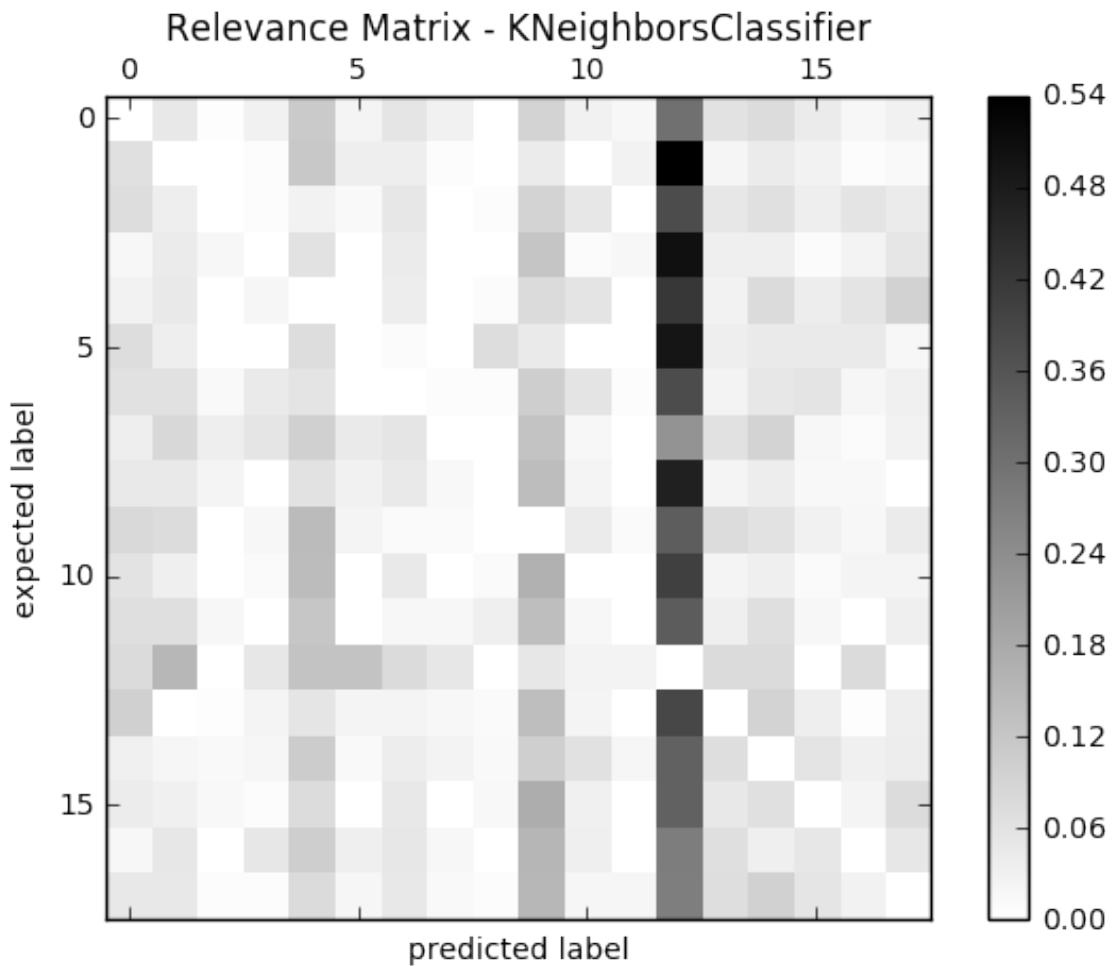


Σχήμα 5.12: Πίνακας/μάσκα συνάφειας του Logistic Regression

Παρατηρούμε ότι οι κλάσεις 2, 3, 7, 11, 12 και 16 έχουν συγκεντρώσει τα λιγότερα έγγραφα σε αντίθεση με τις 0, 1, 5, 8, 9 και 10 όπου συγκέντρωσαν τα περισσότερα. Η εσφαλμένη αυτή ταξινόμηση μας υποδηλώνει στην μεν πρώτη περίπτωση (κλάσεις 2, 3, 7, 11, 12 και 16) ότι οι κλάσεις αυτές διαφέρουν αρκετά από την αναμενόμενη κλάση ωστόσο όμως έχουν κοινά χαρακτηριστικά μεταξύ τους. Στην δεύτερη περίπτωση (κλάσεις 0, 1, 5, 8, 9 και 10) συμπεραίνουμε πως πάλι οι κλάσεις έχουν κοινά στοιχεία, όχι όμως μόνο μεταξύ τους αλλά και με την αναμενόμενη κατηγορία.

Το μοντέλο K-nearest neighbors

Ακολουθεί ο πίνακας/μάσκα συνάφειας που δημιουργήθηκε:

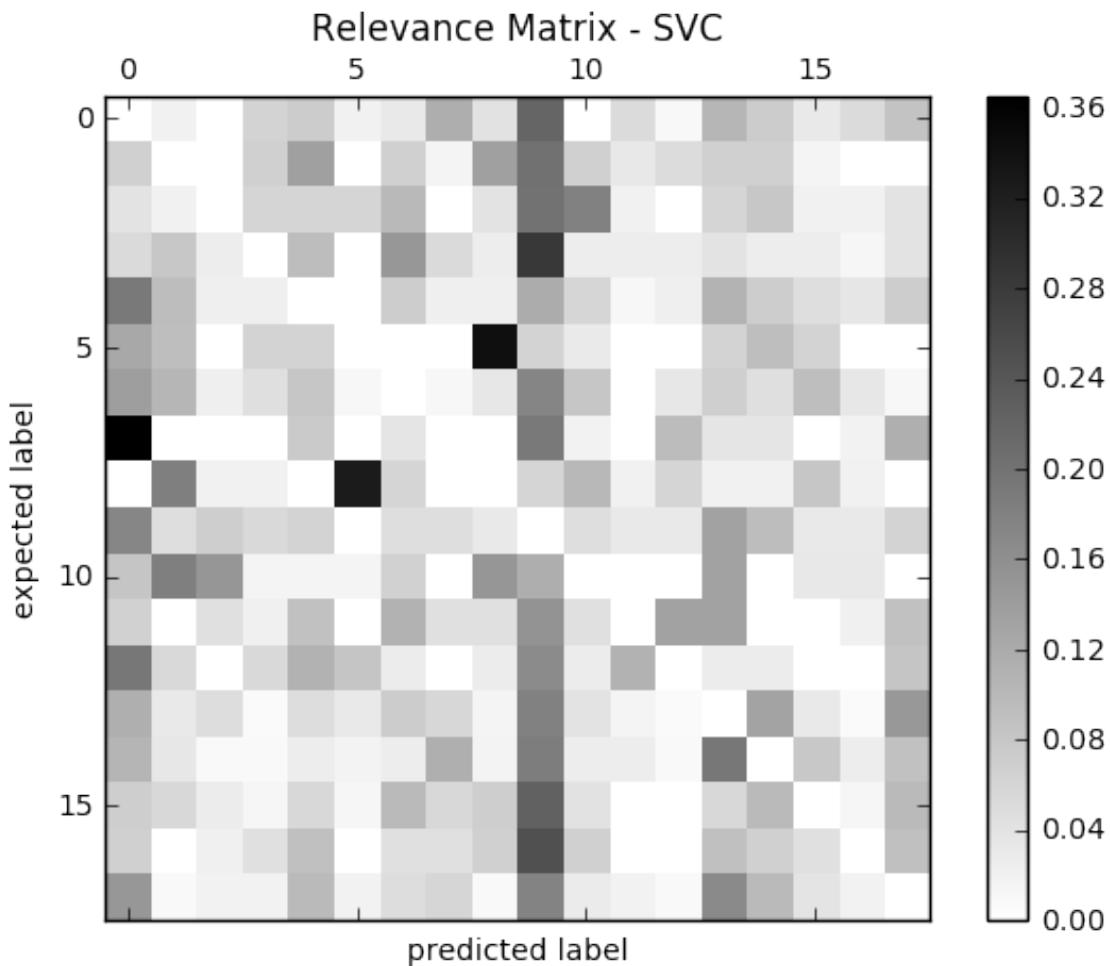


Σχήμα 5.13: Πίνακας/μάσκα συνάφειας του K-nearest neighbors

Εδώ παρατηρούμε ότι οι κλάσεις 0, 1, 4, 9 και 12 έχουν συλλέξει τα περισσότερα λανθασμένα ταξινομημένα instances με κυρίαρχη την κλάση 12 όπως είναι εμφανές. Γενικά εάν εξαιρέσουμε την κλάση 12 και ίσως την 4 και την 9, τα υπόλοιπα instances ακολουθούν μία σχετικά κανονική κατανομή μεταξύ των λανθασμένων κλάσεων.

To μοντέλο Support vector machine

Ο πίνακας/μάσκα συνάφειας που δημιουργήθηκε είναι ο παρακάτω:



Σχήμα 5.14: Πίνακας/μάσκα συνάφειας του Support vector machine

Παρατηρούμε ότι οι κλάσεις 2, 5, 7, 11, 12 και 16 έχουν συγκεντρώσει τα λιγότερα έγγραφα σε αντίθεση με τις 0, 1, 4, 9, 10, 13 και 14 όπου συγκέντρωσαν τα περισσότερα. Η εσφαλμένη αυτή ταξινόμηση μας υποδηλώνει στην μεν πρώτη περίπτωση (κλάσεις 2, 5, 7, 11, 12 και 16) ότι οι κλάσεις αυτές διαφέρουν αρκετά από την αναμενόμενη κλάση ωστόσο όμως έχουν κοινά χαρακτηριστικά μεταξύ τους. Στην δεύτερη περίπτωση (κλάσεις 0, 1, 4, 9, 10, 13 και 14) συμπεραίνουμε πως πάλι οι κλάσεις έχουν κοινά στοιχεία, όχι όμως μόνο μεταξύ τους αλλά και με την αναμενόμενη κατηγορία.

5.3 Γενικά Συμπεράσματα

Εφαρμόσαμε τεχνικές ταξινόμησης και συσταδοποίησης σε δύο αρκετά γνωστά datasets. Στην ταξινόμηση επιλέξαμε πέντε δημοφιλή μοντέλα (Naive Bayes, Random Forest, Logistic Regression, K-nearest neighbors και Support vector machine) για να υλοποιήσουμε τα πειράματα και τις δοκιμές. Στην συσταδοποίηση επιλέξαμε δύο διαφορετικές μεθόδους σε διαφορετικά δατασετς. Αυτό της LDA στο “Six Categories of Amazon Product Reviews” και την μέθοδο των πινάκων/μασκών συνάφειας στο “Amazon Movies Reviews Dataset”. Έπειτα παρατηρήσαμε τα αποτελέσματα και τα αξιολογήσαμε.

Βλέπουμε ότι, παρόλα τα υψηλά ποσοστά απόδοσης των μοντέλων, υπάρχουν αρκετοί παράγοντες που συμβάλουν υλοποίηση ενός επιτυχημένου μηχανισμού ταξινόμησης ή συσταδοποίησης. Παίζουν σπουδαίο ρόλο τα δεδομένα με τα οποία εκπαιδεύουμε τα μοντέλα μας, πόσο “καυθαρά” είναι και σωστά δομημένα. Επίσης το εκάστοτε μοντέλο δεν είναι κατάλληλο για όλους τους τύπους δεδομένων. Αυτό φάνηκε από την διακύμανση των αποδόσεων στις υλοποιήσεις των μοντέλων. Δεν υπήρχε παρόμοια συμπεριφορά στις προβλέψεις τους.

Η λύση στα προβλήματα μηχανικής μάθησης δεν είναι μοναδική. Υπάρχουν πολλές διαφορετικές προσεγγίσεις, άλλες πιο αποδοτικές και άλλες λιγότερο. Δύσκολα όμως θα βρεθεί μοντέλο εκπαιδευμένο που να έχει 100% απόδοση επιτυχημένων προβλέψεων. Γιαυτό το λόγο υπάρχει μεγάλο ενδιαφέρον στο συγκεχριμένο τομέα με την έρευνα να συνεχίζεται βελτιώνοντας τα ήδη υπάρχοντα μοντέλα άλλα και δημιουργώντας νέα ακόμη πιο αποδοτικά και ευέλικτα.

Παράρτημα Α'

Κώδικας σε γλώσσα προγραμματισμού Python

Το σύνολο του κώδικα βρίσκεται δημοσιευμένο σε Git Repository [6].

A'.1 Εμπλουτισμός του dataset με τα συλλεχθέντα labels

Όπως αναφέρθηκε στην ενότητα 3.4.3, σελίδα 37.

```
1 import gzip
2 import csv
3 import ast
4
5 def look_up(asin, diction):
6     try:
7         return diction[asin]
8     except KeyError:
9         return []
10
11 def load_labels():
12     labels_dictionary = {}
13     with open('labels.csv', mode='r') as infile:
14         csvreader = csv.reader(infile)
15         next(csvreader)
16         for rows in csvreader:
17             labels_dictionary[rows[0]] = ast.literal_eval(rows[1])
18     return labels_dictionary
19
20 def parse(filename):
21     labels_dict = load_labels()
22     f = gzip.open(filename, 'r')
```

```

23     entry = {}
24     for l in f:
25         l = l.strip()
26         colonPos = l.find(':')
27         if colonPos == -1:
28             yield entry
29             entry = {}
30             continue
31         eName = l[:colonPos]
32         rest = l[colonPos+2:]
33         entry[eName] = rest
34         if eName == 'product/productId':
35             entry['product/categories'] = look_up(rest, labels_dict)
36     yield entry
37
38 if __name__ == "__main__":
39     try:
40         print ("Parsing dataset...\\nPlease be patient, this
41         will take a while...")
42         with gzip.open('output.txt.gz', 'wb') as fo:
43             for e in parse("movies.txt.gz"):
44                 for i in e:
45                     fo.write('%s: %s\\n' % (i, e[i]))
46                     fo.write("\\n")
47         print ("New enriched dataset has been exported successfully!\\n
48         File name: output.txt.gz")
49     except Exception as inst:
50         print type(inst)
51         print inst.args
52         print inst

```

A'.2 Εξαγωγή των labels σε format ιεραρχικής δομής (tree structure) σε JSON αρχείο

Όπως αναφέρθηκε στην ενότητα 3.4.2, σελίδα 35.

```

1 # coding: utf-8
2 import json
3 import csv
4 import ast
5

```

```
6 def print_tree(datalist):
7     try:
8         temp_list = dataList[:]
9         for s in temp_list[0]:
10            if not s:
11                dataList[0].remove(s)
12
13        empty = 0
14        for lista in dataList[0]:
15            if lista:
16                empty += 1
17        if empty == 0:
18            return
19
20        result = {}
21        firsts = []
22        for m in dataList:
23            for n in m:
24                firsts.append(n[0])
25        firsts = list(set(firsts))
26        for i in firsts:
27            temp = []
28            for j in dataList:
29                for l in j:
30                    #print
31                    #print i, j[0]
32                    if i == l[0]:
33                        s = l[:]
34                        s.pop(0)
35                        temp.append(s)
36            result[i] = print_tree([temp])
37        return result
38    except Exception as e:
39        print e, dataList
40
41 print('Loading data from labels.csv...')
42 theList = []
43 with open('labels.csv', mode='r') as infile:
44     csvreader = csv.reader(infile)
45     next(csvreader)
46     for rows in csvreader:
```

```

47         cat = ast.literal_eval(rows[1])
48
49     if cat not in theList:
50         theList.append(cat)
51
52 # Uncomment the following code block if you want included the ASINs.
53 # Use theListASIN list.
54 #
55 #print('Loading data from labels.csv...')
56 #theListASIN = []
57 #with open('labels.csv', mode='r') as infile:
58 #    csvreader = csv.reader(infile)
59 #    next(csvreader)
60 #    for rows in csvreader:
61 #        cat = ast.literal_eval(rows[1])
62 #
63 #        cat.append(rows[0])
64 #        theListASIN.append(cat)
65 print('Data have been loaded successfully!\n'
66       'Exporting data to JSON file...')
67
68 with open('export_labels_tree.json', 'w') as outfile:
69     json.dump({'root':print_tree([theList])}, outfile,\n
70                sort_keys=True, indent=4)
71
72 # Uncomment the following two lines if you want included the categories
73 # but also the ASINs
74 # If that is your choice, keep in mind that it will take some time to
75 # export a tree with ~v253k paths
76 #with open('export_labels_ASINs_tree.json', 'w') as outfile:
77 #    json.dump({'root':print_tree([theListASIN])}, outfile,
78 #               sort_keys=True, indent=4)
79 print('JSON file is ready!')

```

A'.3 Δημιουργία μασκών/πινάκων Συνάφειας

Όπως αναφέρθηκε στην ενότητα 4.3.2, σελίδα 86.

```

1 # Λίστα με τα id's των labels
2 size = sorted(tempDict.values())
3 # Λίστες που χρατάνε τα ενδιάμεσα αποτελέσματα των αποδόσεων και άλλων

```

```

4 # δεδομένων
5 acc_list = []
6 as_list = []
7 dfs = []
8 # Αρχικός βρόγχος που διαπερνά όλα τα μοντέλα ταξινόμησης
9 # Για κάθε μοντέλο
10 for m in map_models:
11     # Dictionary που χρατά τις αποδόσεις των μοντέλων
12     acc = {}
13     # Αρχικοποίηση του πίνακα με τιμές 0
14     A = np.zeros(shape=(len(size), len(size)))
15     print(str(m) + '\n')
16     # Για κάθε label
17     for i in size:
18         # Λεκτικό label
19         label = inv_map[i]
20         print('Label: ', label, ' - ', i)
21         # Φιλτράρισμα του dataset μόνο για το τρέχων label
22         XX = map_df_test[map_df_test.real_category_num == i]
23             ['review/text']
24         yy = map_df_test[map_df_test.real_category_num == i]
25             ['real_category_num']
26         XX_dtm = vect.transform(XX)
27         # Διαδικασία πρόβλεψης του τρέχοντος μοντέλου
28         cvpredicted = m.predict(XX_dtm)
29         print(XX_dtm.shape)
30         # Υπολογισμός ποσοστού απόδοσης σωστών προβλέψεων
31         ac = metrics.accuracy_score(yy, cvpredicted)
32         print("Accuracy: %0.2f %" % (ac * 100))
33         acc[label] = ac
34         y_mis = np.asarray(yy)
35         # Υπολογισμός των λανθασμένα ταξινομημένων reviews (χρατάει
36         # τα id's το misclassified)
37         misclassified = np.where(y_mis != cvpredicted)
38         print('Misclassified entries: ', len(misclassified[0]))
39         # Υπολογισμός των σωστά ταξινομημένων reviews (χρατάει τα
40         # id's το classified)
41         classified = np.where(y_mis == cvpredicted)
42         print('Well-classified entries: ', len(classified[0]))
43         # Dataframe με τα misclassified reviews
44         misclassified_df = df.ix[misclassified[0]]

```

```

45     # Δημιουργία λίστας με την λανθασμένη επιλογή label
46     misList = []
47     for d, value in enumerate(cvpredicted.tolist()):
48         if value != i:
49             misList.append(value)
50     # Προσθήκη στήλης με την λανθασμένη πρόβλεψη του κάθε review
51     misclassified_df['mispred'] = misList
52     # Ο πίνακα res αποθηκεύει για κάθε label πόσα reviews συγκέντρωσε
53     # και το διαιρώ με το πλήθος των
54     # misclassified για να έχω το τελικό ποσοστό σε κάθε κατηγορία
55     res = misclassified_df.mispred.value_counts() / len(misclassified[0])
56     # Ο παραπάνω πίνακας αφορά ένα μόνο label (το τρέχων) οπότε
57     # καταχωρείται η γραμμή στον πίνακα αποτελεσμάτων A
58     A[i] = res.as_matrix(size)
59     # Το στοιχείο της διαγωνίου παίρνει την τιμή 0
60     A[i][i] = 0
61     print('-----')
62     print
63     asaf = pd.DataFrame(data=A, index=size, columns=size)
64     nd = asaf.fillna(0)
65     # Αποθηκεύονται τα δεδομένα πριν περάσουμε στο επόμενο μοντέλο
66     dfs.append(nd)
67     acc_list.append(acc)
68     as_list.append(A)

```

A'.4 Μέθοδος συλλογής δεδομένων (scraping) από τις σελίδες των προϊόντων της Amazon.com

Όπως αναφέρθηκε στην ενότητα 3.4.1, σελίδα 33.

```

1 # coding: utf-8
2 # Εισαγωγή χρήσιμων βιβλιοθηκών
3 import pandas as pd
4 import pprint
5 import gzip
6 from lxml import html
7 import csv
8 import requests
9 from exceptions import ValueError
10 import time
11 from random import randint

```

```
12
13 # Μέθοδος που παίρνει ως όρισμα το url του προιόντος και επιστρέφει
14 # μία λίστα python με τα labels που αντιστοιχούν στο υπό εξέταση asin.
15 def AmzonParser(url):
16     # Μετρητής για το πόσες φορές θα προκύψει Captcha control.
17     global countRobots
18     # Λίστα από headers που εναλλάσσονται για να προσομοιωθεί όσο το
19     # δυνατόν καλύτερα η επαναλαμβανόμενη αποστολή
20     # των http requests
21     headers = [
22         {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) \
23             AppleWebKit/537.36 (KHTML, like Gecko) \
24             Chrome/54.0.2840.90 Safari/537.36'},
25         {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel \
26             Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, \
27             like Gecko) Chrome/41.0.2227.1 Safari/537.36'},
28         {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; \
29             WOW64; rv:40.0) Gecko/20100101 Firefox/40.1'},
30         {'User-Agent': 'Mozilla/5.0 (Windows NT 6.3; rv:36.0) \
31             Gecko/20100101 Firefox/36.0'},
32         {'User-Agent': 'Mozilla/5.0 (X11; OpenBSD i386) \
33             AppleWebKit/537.36 (KHTML, like Gecko) \
34             Chrome/36.0.1985.125 Safari/537.36'}
35     ]
36     # Μικρή παύση
37     ##time.sleep(randint(4, 6))
38     # Αποστολή http request
39     page = requests.get(url,
40         headers=headers[randint(0, len(headers) - 1)])
41     while True:
42         time.sleep(1)
43         try:
44             # Δημιουργία του html αντικειμένου και επιλογή του
45             # κατάλληλου tag όπου εμφανίζονται τα labels
46             doc = html.fromstring(page.content)
47             XPATH_CATEGORY =
48                 '//a[@class="a-link-normal a-color-tertiary"]//text()'
49             RAW_CATEGORY = doc.xpath(XPATH_CATEGORY)
50             noCategories = '//h1[@id="btf-product-details"]//text()'
51             rawNoCategories = doc.xpath(noCategories)
52
```

```

53 # Έλεγχος για Captcha
54 if doc.findtext('.//title') == 'Robot Check':
55     countRobots += 1
56     print str(countRobots) + 'th Robot Check pause:',
57             time.strftime("%d/%m/%y %H:%M"),
58             time.localtime(time.time()))
59 #time.sleep(countRobots * 120)
60 ##time.sleep(randint(4, 6))
61 # Μικρές παύσεις σε περίπτωση captha
62 if countRobots == 4:
63     global start_time, start_time2
64     start_time = time.time()
65     start_time2 = time.time()
66     countRobots = 0
67 # Επιστρέφει την λέξη Captcha και τερματίζει η μέθοδος
68 return ['Captcha']
69 categories = []
70 # Μεταβλητή που ορίζει το βάθος (στην ιεραρχική τους δομή)
71 # των labels που θα συλλεχθούν
72 depth = 5
73 # Σε περίπτωση που δεν υπάρχουν labels
74 if not RAW_CATEGORY and rawNoCategories:
75     categories.append('None')
76 elif not rawNoCategories and not RAW_CATEGORY:
77     # print 'Unable to fetch categories...'
78     categories.append('Unable to fetch categories...')
79 # Διαφορετικά, εάν υπάρχουν labels
80 else:
81     # Ενημερώνεται η λίστα με τα labels
82     for index, i in enumerate(RAW_CATEGORY)
83         if RAW_CATEGORY else None:
84             if index < depth:
85                 categories.append(i.strip())
86             pass
87 # Έλεγχος για http error codes
88 if page.status_code != 200:
89     if page.status_code == 404:
90         categories = ['HTTP 404']
91     else:
92         print page.status_code
93         raise ValueError('captha')

```

```
94         data = categories
95
96     return data
97
98 except Exception as e:
99     print e
100
101 # Μέθοδος που δέχεται ως ορίσματα ένα url που οδηγεί σε csv αρχείο και
102 # τον τρόπο ανοίγματος του αρχείου
103 # επιστρέφει λίστα με τα δεδομένα του csv αφαιρόντας τυχόν διπλές
104 # εγγραφές
105 def listFromCSV(url, mode):
106     mylist = []
107     with open(url, mode=mode) as initfile:
108         csvreader = csv.reader(initfile)
109         next(csvreader)
110         for rows in csvreader:
111             mylist.append(rows[0].strip())
112     return list(set(mylist))
113
114 # Μέθοδος που παίρνει ως άρισμα ένα asin, ολοκληρώνει την διαδικασία
115 # συλλογής των labels και
116 # ανάλογα με την έκβαση του htto reques (αποδεκτό, 404, captcha κλπ)
117 # ενημερώνει τα αντίστοιχα csv
118 # αρχεία που κρατάνε τα asins και φορτώνονται κάθε φορά που εκτελείται
119 # ο αλγόριθμος
120 def parse2(asin):
121     # Έλεγχος για το εάν το υπό εξέταση asin έχει ήδη ελεχθέι και
122     # βρίσκεται στην λίστα που τηρούνται τα
123     # αποδεκτά asin's και τα λανθασμένα (404 error κλπ)
124     accept = asin in acceptedASINs.keys()
125     unaccept = asin in errorASINs.keys()
126     # Σε περίπτωση που το υπό εξέταση asin δεν έχει ελεγχθεί ποτέ
127     if not accept and not unaccept:
128         # Λίστα με πλήθος μηδενικών για την αντιμετώπιση προβλήματος
129         # με asin που ξεκινούσαν με 0
130         zeros = {0:'', 1:'0', 2:'00', 3:'000', 4:'0000', 5:'00000',
131         6:'000000'}
132         # Εάν το μήκος του asin είναι < 10 (που σημαίνει ότι κάποιο
133         # leading zero χάθηκε)
134         if len(str(asin)) < 10:
135             # Πρόσθισε τα ελλείποντα μηδενικά στην αρχή του asin
136             asin = zeros[10-len(asin)] + asin
```

```

135          # Κάλεσε την AmzonParser με το κατάλληλο url και αποθήκευσε
136          # στην λίστα tempResult
137          # τα labels που αντιστοιχούν στο υπό εξέταση asin
138          tempResult = AmzonParser("https://www.amazon.com/dp/" + asin)
139          # Έλεγχος για το εάν το παραπάνω request δεν έγινε αποδεκτό
140          cond1 = tempResult[0] in \
141                  ['None', 'HTTP 404', 'Captcha', \
142                  'Unable to fetch categories...']
143          # Εάν δεν έγινε αποδεκτό
144          if cond1:
145              # Περίπτωση Captcha και ενημέρωση του αντίστοιχου αρχείου
146              if tempResult[0] == 'Captcha':
147                  captcha_list.append(asin)
148                  with open('csvexport/captcha_list.csv', 'wb') as myfile:
149                      fieldnames = ['ASIN']
150                      wr = csv.DictWriter(myfile, fieldnames=fieldnames)
151                      wr.writerow({'ASIN': 'ASIN'})
152                      for j in captcha_list:
153                          wr.writerow({'ASIN': j.strip()})
154              # Περίπτωση 404 error ή κάποιου άλλου και ενημέρωση του
155              # αντίστοιχου αρχείου
156          else:
157              errorASINs[asin] = tempResult
158              print tempResult[0], '—', len(errorASINs), '—',
159              len(errorASINs[asin]), '—', "http://www.amazon.com/dp/" \
160                  + asin, '—', time.strftime("%d/%m/%y %H:%M",
161                  time.localtime(time.time())), '—', (time.time() -
162                  start_time2) / 60
163
164              with open('csvexport/ASINerror.csv', 'ab') as csvfile:
165                  fieldnames = ['ASIN', 'Categories', 'timestamp']
166                  writer = csv.DictWriter(csvfile, \
167                      fieldnames=fieldnames)
168                  writer.writerow({'ASIN': str(asin).strip(), \
169                      'Categories': errorASINs[asin], 'timestamp': \
170                          time.strftime("%d/%m/%y %H:%M", \
171                          time.localtime(time.time()))})
172
173          # Άλλιώς το asin είναι αποδεκτό
174          acceptedASINs[asin] = tempResult
175          # Τυπώνονται κάποια ενημερωτικά στοιχεία

```

```
175         print 'OK -', len(acceptedASINs), '-',
176         len(acceptedASINs[asin]),
177         '_', "http://www.amazon.com/dp/" + asin,
178         '_', time.strftime("%d/%m/%y %H:%M",
179         time.localtime(time.time())), '_',
180         (time.time() - start_time2) / 60
181     # Ενημερώνεται το αντίστοιχο αρχείο με τα labels που
182     # συλλέχθηκαν
183     with open('csvexport/ASINaccepted2.csv', 'ab') as csvfile:
184         fieldnames = ['ASIN', 'Categories', 'timestamp']
185         writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
186         writer.writerow({'ASIN': str(asin).strip(),
187                         'Categories': acceptedASINs[asin], 'timestamp':
188                         time.strftime("%d/%m/%y %H:%M",
189                         time.localtime(time.time()))})
190     # Μετά τον έλεγχο του asin ως προς το αν είναι αποδεκτό ή όχι
191     # αφαιρείται από την λίστα που χρατάει όλα τα προς εξέταση asins
192     temp_list.remove(asin)
193     # Ενημερώνεται το αρχείο που χρατάει τα asins προς εξέταση
194     with open('csvexport/ypoloipa.csv', 'wb') as myfile:
195         fieldnames = ['ASIN']
196         wr = csv.DictWriter(myfile, fieldnames=fieldnames)
197         wr.writerow({'ASIN': 'ASIN'})
198         for j in temp_list:
199             wr.writerow({'ASIN': j.strip()})
200
201 # Σβήσιμο των δεδομένων του αρχείου που χρατάει τα asins από captcha
202 # ώστε να είναι έτοιμο να δεχθεί τα νέα asins
203 def delCaptchaList(url):
204     with open(url, 'wb') as myfile:
205         fieldnames = ['ASIN']
206         wr = csv.DictWriter(myfile, fieldnames=fieldnames)
207         wr.writerow({'ASIN': 'ASIN'})
208         for j in []:
209             wr.writerow({'ASIN': j.strip()})
210
211 # Μέθοδος που επιστρέφει ένα python dictionary το οποίο παράγεται από
212 # το αρχείο του ορίσματος
213 def dictFromCSV(url):
214     dict = {}
215     with open(url, mode='r') as infile:
```

```

216         csvreader = csv.reader(infile)
217         next(csvreader)
218         for rows in csvreader:
219             dict[rows[0]] = rows[1]
220     return dict
221
222 # Μέθοδος που τυπώνει το ποσοστό προόδου
223 def printPercentage (per):
224     percentage = ''
225     for p in range(0, per):
226         percentage += '*'
227     for p in range(0, 100 - per):
228         percentage += '_'
229     print "Progress: %.2f%% %s" % (per, percentage)
230
231 if __name__ == "__main__":
232     # Παίρνουμε όλα τα asins από το dataset και αφαιρούνται τα διπλά
233     # Το 404.csv χρατάει αυτά που επιστρέφουν 404 error
234     12 = listFromCSV('csvexport/404.csv', 'r')
235     # Το ypoloipa.csv χρατάει αυτά που δεν ολοκληρώθηκαν για οποιοδήποτε
236     λόγο στην προηγούμενη εκτέλεση
237     # του προγράμματος
238     13 = listFromCSV('csvexport/ypoloipa.csv', 'r')
239     # Το captcha_list.csv χρατάει αυτά που επιστρέφουν έλεγχο captcha
240     14 = listFromCSV('csvexport/captcha_list.csv', 'r')
241     # Γίνεται η ένωση των παραπάνω λιστών
242     initlistset = 12 + 13 + 14
243
244     # Σβήσιμο των asins του captcha ώστε να είναι έτοιμη να δεχθεί τα νέα
245     # asins που θα επιστρέψουν έλεγχο captcha
246     delCaptchaList('csvexport/captcha_list.csv')
247
248     # Φορτώνουμε στην λίστα acceptedASINs όλα όσα έχουμε ήδη συλλέξει
249     acceptedASINs = dictFromCSV('csvexport/ASINaccepted2.csv')
250     # Το πλήθος αυτών
251     lenacceptedASINs = len(acceptedASINs)
252     # Φορτώνουμε στην λίστα errorASINs όλα όσα έχουμε ήδη συλλέξει και ήταν
253     # 404 error
254     errorASINs = dictFromCSV('csvexport/ASINerror.csv')
255     # Το πλήθος αυτών
256     lenerrorASINs = len(errorASINs)

```



```
295                     lenerrorASINs)
296                     i += 1
297                     time.sleep(int(restTime/2))
298                     #time.sleep(randint(4, 6))
299                     start_time = time.time()
300                     start_time2 = time.time()
301                     # Διαχείριση exceptions
302             except:
303                 print 'pass'
304                 pass
305             print 'End of list'
306         except Exception as e:
307             print e
308             raise ValueError(e)
```

Βιβλιογραφία

- [1] David M. Blei, Andrew Y. Ng και Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.
- [2] Ioannis Charalampopoulos και Ioannis Anagnostopoulos. A comparable study employing WEKA clustering/classification algorithms for web page classification. Στο *15th Panhellenic Conference on Informatics, PCI 2011, Kastoria, Greece, September 30 - October 2, 2011*, σελίδες 235–239, 2011.
- [3] Corinna Cortes και Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, 1995.
- [4] P. Domingos και M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–137, 1997.
- [5] Bazakos Konstantinos. Addition of ground truth labels on amazon movie reviews dataset. GitHub repository: <https://github.com/bazakoskon/labels-on-Amazon-movie-reviews-dataset>, 2017.
- [6] Bazakos Konstantinos. Classification/clustering techniques for large web data collections. GitHub repository: <https://github.com/bazakoskon/Classification-clustering-Thesis>, 2017.
- [7] Julian John McAuley και Jure Leskovec. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. Στο *Proceedings of the 22Nd International Conference on World Wide Web, WWW '13*, σελίδες 897–908, New York, NY, USA, 2013. ACM.
- [8] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1η έκδοση, 1997.
- [9] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986.
- [10] Karen Sparck Jones. Document retrieval systems. κεφάλαιο A Statistical Interpretation of Term Specificity and Its Application in Retrieval, σελίδες 132–142. Taylor Graham Publishing, London, UK, UK, 1988.

- [11] Γούδας Θεοδόσιος. Διπλωματική Εργασία, Συστήματα Υποστήριξης Αποφάσεων Δι-άγνωσης Βαλβιδοπαθειών με χρήση καρδιακών ήχων, Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων, Παν. Αιγαίου, 2007.
- [12] Ιωάννης Μανωλόπουλος, Απόστολος Ν. Παπαδόπουλος. Ανάκτηση Πληροφορίας, Αρι-στοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Σχολή Θετικών Επιστημών, Τμήμα Πληροφορι-κής, 2010.
- [13] Σπυρίδων Κάτσαμπος. Πτυχιακή Εργασία, Εφαρμογή Διαχείρισης Πληροφορίας σε Πη-γές Νομικών Κειμένων, Τμήμα Πληροφορικής με Εφαρμογές στη Βιοϊατρική, Παν. Θεσ-σαλίας, 2015.
- [14] Ταμπώχ Αριάννα. Πτυχιακή Εργασία, Αναγνώριση κίνησης σε video με τη χρήση ομα-δοποίησης και του αλγόριθμου latent dirichlet allocation, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Σχολή Θετικών Επιστημών, Τμήμα Πληροφορικής. 2009.
- [15] Hongning Wang, Yue Lu και Chengxiang Zhai. Latent aspect rating analysis on review text data: A rating regression approach. Στο *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, σελίδες 783–792, New York, NY, USA, 2010. ACM.
- [16] Hongning Wang, Yue Lu και ChengXiang Zhai. Latent aspect rating analysis without aspect keyword supervision. Στο *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, σελίδες 618–626, New York, NY, USA, 2011. ACM.

Ευρετήριο όρων

- Ακρίβεια, 13
Ανάληση, 13
Βιβλιοθήκες, 39
Διανυσματικός Χώρος, 6
Ενισχυτική μάθηση, 11
Επιβλεπόμενη μάθηση, 9
Μάσκα Συνάφειας, 86
Μη επιβλεπόμενη μάθηση, 10
Πίνακας Συνάφειας, 86
Πηγές δεδομένων, 27
Συσταδοποίηση, 74, 86
Ταξινόμηση, 42, 58
εξόρυξης πληροφορίας, 5
κατηγοριοποίηση, 5
συσταδοποίηση, 5
Classification, 42, 58
Clustering, 74, 86
Cross-Validation, 11
IPython Notebook, 42
K-Nearest Neighbors, 20
K-nearest neighbors, 54, 71, 97, 101, 112
LDA, 24, 74, 108
Latent Dirichlet Allocation, 24, 74
Latent Dirichlet allocation, 108
Logistic Regression, 22, 96, 99, 111
Logistic regression, 51, 67
Naive Bayes, 23, 50, 66, 96, 98, 109
Precision, 13
Python, 39
Random Forest, 53, 69, 97, 100, 110
Random forest, 19
Recall, 13
Reinforcement learning, 11
Scikit-learn, 41
Supervised learning, 9
Support vector machine, 17, 57, 72, 98, 102, 113
Topic model, 74
Unsupervised learning, 10
data pre-processing, 32
dataset, 27
tf-idf, 8





ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ: ΠΛΗΡΟΦΟΡΙΚΗ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τεχνικές Ταξινόμησης/Συσταδοποίησης
Μεγάλων Συλλογών Διαδικτυακών Δεδομένων
HOU-CS-UGP-2016-915-7

Κωνσταντίνος Ν. Μπαζάκος

ΠΑΤΡΑ

ΙΟΥΛΙΟΣ 2017



ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΣΠΟΥΔΩΝ: ΠΛΗΡΟΦΟΡΙΚΗ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τεχνικές Ταξινόμησης/Συσταδοποίησης
Μεγάλων Συλλογών Διαδικτυακών Δεδομένων
HOU-CS-UGP-2016-915-7

Κωνσταντίνος Ν. Μπαζάκος

ΠΑΤΡΑ

ΙΟΥΛΙΟΣ 2017

