ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΜΣ «ΠΡΟΗΓΜΕΝΑ ΣΥΣΤΗΜΑΤΑ ΠΛΗΡΟΦΟΡΙΚΗΣ»

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΘΕΜΑ : Resampling Algorithms for the Class Imbalance Problem : A Case Study on Sentiment Analysis

Ονοματεπώνυμο : Παναγιώτης Μητσιάνης Α.Μ. : ΜΠΣΠ15054

Επιβλέπων : Γεώργιος Τσιχριντζής

Συνεπιβλέπων : Διονύσιος Σωτηρόπουλος

Πειραιάς 2017

Περιεχόμενα

1. Εισαγωγή

Το Class Imbalance Problem είναι ένα πρόβλημα της Αναγνώρισης Προτύπων, το οποίο προκύπτει στις περιπτώσεις που ένα σύνολο προτύπων εκπαίδευσης κλίνει περισσότερο προς μια κατηγορία κλάσης. Με άλλα λόγια δηλαδή, το πρόβλημα αυτό προκύπτει όταν μια κατηγορία του συνόλου αντιπροσωπεύεται από μεγάλο αριθμό παραδειγμάτων σε σχέση με το άλλο ή τα άλλα, τα οποία αντιπροσωπεύονται από λιγότερα.
Αυτό αποτελεί πρόβλημα, διότι οι περισσότεροι αλγόριθμοι μηχανικής μάθησης λειτουργούν καλύτερα όταν ο αριθμός των εμφανίσεων σε κάθε κλάση είναι περίπου ο ίδιος. Όταν ο αριθμός των εμφανίσεων σε μία τάξη υπερβαίνει κατά πολύ την άλλη ή τις άλλες, προκύπτουν προβλήματα. Εάν υπάρχει για παράδειγμα ένα σύνολο δεδομένων που αποτελείται από 10000 γνήσιες και 10 δόλιες συναλλαγές, ο ταξινομητής θα τείνει να χαρακτηρίσει παράνομες συναλλαγές ως γνήσιες συναλλαγές. Αν πρόκειται αυτό το παράδειγμα να αφορά τράπεζες ή εμπορικές εταιρίες, αυτό συνεπάγεται δυσαρεστημένοι πελάτες και κατά συνέπεια μείωση εσόδων και αξιοπιστίας για την εταιρία.
Το Class Imbalance Problem έκανε την εμφάνισή του στη βιβλιογραφία περίπου 15 χρόνια πριν. Η σημασία του όμως αυξήθηκε, καθώς όλο και περισσότεροι ερευνητές κατάλαβαν πως τα δεδομένα τους ήταν μη ισορροπημένα, με αποτέλεσμα να μην έχουν τη βέλτιστη απόδοση ταξινόμησης.
Τέτοια παραδείγματα είναι η ανίχνευση του δόλου σε τηλεφωνικές κλήσεις ή σε συναλλαγές με πιστωτική κάρτα. Επίσης ένα παρόμοιο πρόβλημα παρουσιάζεται όταν χρειάζεται να εντοπιστούν οι αναξιόπιστοι τηλεπικοινωνιακοί πελάτες, κι αυτό διότι αποτελούν πολύ μικρό κομμάτι του συνόλου.
Δεν είναι όμως μόνο ο δόλος, ο οποίος προκαλεί το πρόβλημα. Η προκατάληψη ή η μεροληψία ορισμένων ανθρώπων σε μια έρευνα ή σε μια επιλογή γενικότερα, μπορεί να προκαλέσει λάθος συμπεράσματα. Με άλλα λόγια δηλαδή, μερικοί άνθρωποι μπορεί να δώσουν μια άλλη απάντηση σε μια έρευνα ή να επιλέξουν να αγοράσουν ένα διαφορετικό προϊόν έναντι κάποιου άλλου, όχι επειδή το πιστεύουν αλλά επειδή λειτουργούν προκατειλημμένα γι' αυτό.
Ένας άλλος λόγος, ο οποίος μπορεί να δημιουργήσει μικρότερα δείγματα, είναι κάποιες σπάνιες περιπτώσεις, οι οποίες υπάρχουν σε ένα σύνολο δειγμάτων και πρέπει να ληφθούν υπόψη ώστε να έχουμε ασφαλή συμπεράσματα.
Σαν θεραπεία του Class Imbalance Problem είναι η αναδειγματοληψία (Resampling). Η αναδειγματοληψία λειτουργεί αλλάζοντας τις «ισορροπίες» στο σύνολο των δεδομένων είτε αυξάνοντας τον αριθμό των δειγμάτων στην κλάση μειοψηφίας (over-sampling) είτε μειώνοντας τον αριθμό των δειγμάτων

στην κλάση της πλειοψηφίας (under-sampling). Το αποτέλεσμα των δεδομένων που προκύπτει μετά την αναδειγματοληψία είναι πιο ισορροπημένο. Οι μέθοδοι αυτοί λοιπόν αναδειγματοληψίας, χρησιμοποιούν τέτοιες ευριστικές, οι οποίες προσπαθούν να προσεγγίσουν τη βέλτιστη κατανομή των δειγμάτων, ώστε να τα επεξεργαστούμε και να βγάλουμε ασφαλή συμπεράσματα για τα δεδομένα.

2. Resampling Techniques

2.1 Under-sampling Methods

Tomek Links

Η Tomek Links είναι μια under-sampling μέθοδος, η οποία απορρίπτει δείγματα από το σύνολο δεδομένων. Η μέθοδος αυτή λειτουργεί μεταξύ ζευγών δειγμάτων (x,y), και η οποία εξασφαλίζει ότι δεν υπάρχει σημείο z τέτοιο ώστε $d(x,z)<d(x,y)$ ή $d(y,z)<d(x,y)$, όπου $d(x,y)$ είναι η απόσταση μεταξύ αυτών των δύο δειγμάτων. Αν λοιπόν δύο δείγματα δημιουργούν ένα Tomek Links, τότε ένα από τα δύο δείγματα είναι είτε θόρυβος είτε είναι στο όριο της κλάσης. Τα δείγματα αυτά που πρέπει να εξαλειφθούν θα πρέπει να είναι αποκλειστικά από την κλάση πλειοψηφίας. Η μέθοδος αυτή είναι σημαντική στην αφαίρεση δεδομένων θορύβου από το σύνολο, τα οποία θα μπορούσαν να δημιουργήσουν λάθη ταξινόμησης. Από την άλλη μεριά όμως μπορεί να διαγραφούν δεδομένα τα οποία θα μπορούσαν να είναι χρήσιμα για την εξαγωγή συμπερασμάτων. Πρέπει να σημειωθεί όμως ότι η μέθοδος αυτή χρησιμοποιεί αλγόριθμο μεγάλης πολυπλοκότητας, ο οποίος μπορεί να εκτελείται πιο αργά σε σχέση με άλλους.

Condensed Nearest Neighbor rule

Η Condensed Nearest Neighbor (CNN) είναι μια under-sampling μέθοδος, η οποία συνδέεται κυρίως με την ταξινόμηση k-Nearest Neighbor (kNN). Ο στόχος αυτής τη μεθόδου είναι η απομάκρυνση στοιχείων από την κλάση πλειοψηφίας που απέχουν πάρα πολύ από τον επιθυμητό μας στόχο, ενώ τα εναπομείναντα στοιχεία παραμένουν για την εκπαίδευση του συστήματος. Η μέθοδος CNN επικεντρώνεται στην εύρεση ενός συνεπούς υποσυνόλου προτύπων εκπαίδευσης. Ένα υποσύνολο $E' \subseteq E$, χαρακτηρίζεται σύμφωνο με το E, αν εκπαιδεύοντας τον 1-NN ταξινομητή στο E', όλα τα στοιχεία στο E θα ταξινομηθούν σωστά. Ειδικότερα, ο CNN ξεκινά συγκεντρώνοντας τυχαία στοιχεία από την κλάση πλειοψηφίας και όλα τα στοιχεία από την E'. Στη συνέχεια, ο 1-NN βάσει του E' ταξινομεί το E. Στόχος όλης αυτής της

διαδικασίας είναι να εξαλειφθούν όλα εκείνα τα δεδομένα τα οποία θεωρούνται λιγότερο σημαντικά για την εκμάθηση του συστήματος.

One-Sided Selection

Η One-Sided Selection είναι μια άλλη under-sampling μέθοδος για την επίλυση του Class Imbalance Problem. Στόχος και αυτής τη μεθόδου είναι η τροποποίηση της κατανομής των αρνητικών δειγμάτων. Αυτή η μέθοδος λοιπόν, δημιουργεί ένα πιο αντιπροσωπευτικό υποσύνολο των αρνητικών δεδομένων τα οποία θα είναι πιο κατάλληλα για την εκπαίδευση του συστήματος. Συγκεκριμένα, η μέθοδος αυτή προσπαθεί να ελαχιστοποιήσει την ύπαρξη δεδομένων τα οποία έχουν θόρυβο ή βρίσκονται στο όριο μεταξύ των θετικών και των αρνητικών δειγμάτων. Αφαιρούνται επίσης αρνητικά δείγματα, τα οποία μπορεί να μην εμποδίζουν την ταξινόμηση, αλλά μπορεί να ληφθούν υπόψη από άλλα παραδείγματα και να μειωθεί έτσι η απόδοση. Ωστόσο, η διαδικασία αυτή αγνοεί ασφαλή αρνητικά δείγματα καθώς και το σύνολο των θετικών δειγμάτων. Σε αυτή την περίπτωση γίνεται προσπάθεια να εξαλειφθούν δείγματα, τα οποία έχουν μέσα τους θόρυβο. Αυτά τα δείγματα λοιπόν απομακρύνονται με τη βοήθεια της μεθόδου Tomek Links, η οποία αναφέρθηκε παραπάνω.
Ο OSS αλγόριθμος λειτουργεί όπως παρακάτω :

1. Έχουμε S το σύνολο των δεδομένων που έχουμε στη διάθεσή μας και C ένα προσωρινό σύνολο, το οποίο θα χρησιμοποιήσουμε στον αλγόριθμο.
2. Αρχικά το C περιλαμβάνει όλα τα θετικά δείγματα και ένα αρνητικό.
3. Γίνεται ταξινόμηση του συνόλου S με βάση τα στοιχεία του C και γίνεται σύγκριση των νέων labels με αυτά που προκύπτουν. Όσα δεν έχουν ταξινομηθεί σωστά, μεταφέρονται στο C.
4. Αφαιρούνται από το σύνολο C όλα τα αρνητικά στοιχεία για τα οποία ισχύει ο Tomek Links. Απομακρύνονται λοιπόν όλα τα στοιχεία τα οποία μπορεί να περιέχουν θόρυβο είτε να είναι στα όρια των κλάσεων. Ανακτώνται και όλα τα θετικά στοιχεία και όλα μαζί τοποθετούνται στον πίνακα T.

Neighborhood Cleaning Rule

Η Neighborhood Cleaning Rule είναι και αυτή μια under-sampling μέθοδος, η οποία λειτουργεί σαν επέκταση της OSS. Το βασικό μειονέκτημα της OSS είναι ότι χρησιμοποιεί τη μέθοδο CNN, η οποία είναι εξαιρετικά ευαίσθητη στο θόρυβο. Γενικά, η ύπαρξη δειγμάτων με θόρυβο εμποδίζει την απόδοση της ταξινόμησης. Το βασικό σκεπτικό πίσω από τον NCL είναι όμοιο με τον OSS, μιας και όλα τα δείγματα της κλάσης που μας ενδιαφέρει, θα ανακτηθούν από

το τελικό σύνολο εκπαίδευσης. Σε αντίθεση με τον OSS, ο NCL δίνει μεγαλύτερη έμφαση στον καθαρισμό και όχι στη μείωση των δεδομένων. Η ποιότητα των αποτελεσμάτων ταξινόμησης δε βασίζεται στο μέγεθος της κλάσης. Υπάρχουν μικρές κλάσεις, οι οποίες λειτουργούν σωστά και μεγάλες, οι οποίες είναι δύσκολο να ταξινομηθούν. Στόχος λοιπόν είναι η δημιουργία μιας μεθόδου under-sampling, η οποία να είναι αρκετά ακριβής παρόλο που τα δεδομένα μειώνονται.

2.2 Over-sampling Methods

Cluster-based Over-sampling

Η μέθοδος Cluster-based Over-sampling βασίζεται στην ομαδοποίηση. Συγκεκριμένα, αυτό επιτυγχάνεται με τη διαδοχική ομαδοποίηση των δεδομένων κάθε κλάσης πραγματοποιώντας τυχαίο over-sampling ομάδα-ομάδα.
Αρχικά, η μέθοδος αυτή, εφαρμόζει τον αλγόριθμο k-means τόσο στην κλάση μειοψηφίας, όσο και στην κλάση της πλειοψηφίας. Αφού λοιπόν γίνει η ομαδοποίηση της κάθε κλάσης, το επόμενο βήμα είναι το over-sampling κάθε μιας από τις επιμέρους ομάδες. Στην κλάση της πλειοψηφίας σε όλες τις επιμέρους ομάδες γίνεται over-sampling, εκτός από εκείνη που έχει τα περισσότερα στοιχεία, ώστε όλες οι ομάδες να αποκτήσουν τον ίδιο αριθμό στοιχείων. Στην κλάση μειοψηφίας, γίνεται over-sampling σε όλες τις επιμέρους ομάδες της κλάσης, ώστε όλες να αποκτήσουν ίδιο αριθμό στοιχείων με τη ομάδα με τα περισσότερα στοιχεία της κλάσης πλειοψηφίας. Θα ορίσουμε maxclasssize το μέγεθος της μεγαλύτερης κλάσης και Nmajorityclass τον αριθμό των ομάδων στην κλάση πλειοψηφίας. Επίσης στην κλάση μειοψηφίας, σε κάθε ομάδα εφαρμόζεται over-sampling, ώστε κάθε ομάδα να αποκτήσει maxclasssize*Nmajorityclass/Nmajorityclass στοιχεία, όπου Nmajorityclass είναι ο αριθμός των υποομάδων στην κλάση μειοψηφίας.

Synthetic Minority Over-Sampling Technique

Η τεχνική Synthetic Minority Over-Sampling Technique (SMOTE) ασχολείται και αυτή στην επίλυση του class imbalance problem. Το βασικό χαρακτηριστικό της SMOTE είναι ότι στην κλάση μειοψηφίας γίνεται over-sampling δημιουργώντας «συνθετικά» δείγματα και όχι αναπαράγοντας ήδη υπάρχοντα. Στην κλάση μειοψηφίας γίνεται over-sampling παίρνοντας κάθε δείγμα από την κλάση και δημιουργώντας συνθετικά δείγματα, τα οποία χρησιμοποιούν τους k Nearest Neighbors των δειγμάτων της κλάσης. Η αρχική υλοποίηση της SMOTE χρησιμοποιεί μόνο 5 Nearest Neighbors. Αν το

ποσοστό over-sampling που πρέπει να γίνει στην κλάση είναι για παράδειγμα 200%, τότε θα χρησιμοποιηθούν 2 Nearest Neighbors. Η δημιουργία των σύνθετων δειγμάτων γίνεται με την ακόλουθη διαδικασία: Αρχικά υπολογίζεται η διαφορά ανάμεσα στο δείγμα που μας ενδιαφέρει και τον Nearest Neighbor αυτού. Στη συνέχεια πολλαπλασιάζεται αυτή η διαφορά με έναν τυχαίο αριθμό μεταξύ του 0 και το αποτέλεσμα της οποίας προστίθεται στο διάνυσμα το οποίο εξετάζουμε.

Borderline-SMOTE

Η μέθοδος αυτή δημιουργήθηκε σαν επέκταση της αρχικής SMOTE over-sampling μεθόδου, με σκοπό να επιτευχθεί καλύτερη απόδοση ταξινόμησης. Αυτό επιτυγχάνεται προσπαθώντας να βρεθούν όσο το δυνατό καλύτερα τα όρια μεταξύ των κλάσεων. Τα στοιχεία, το οποία βρίσκονται κοντά σε αυτά τα όρια είναι πολύ πιο εύκολο να μην ταξινομηθούν σωστά σε σχέση με τα στοιχεία, το οποία είναι μακρύτερα από αυτά. Αρχικά εντοπίζονται τα στοιχεία, τα οποία βρίσκονται σε αυτά τα όρια και στη συνέχεια δημιουργούνται «συνθετικά» στοιχειά από αυτά. Αρχικά ορίζουμε T το σύνολο των δεδομένων που έχουμε στη διάθεσή μας και P και N τα σύνολα των θετικών και αρνητικών στοιχείων αντίστοιχα. P_{num} και n_{num} είναι ο αριθμός των θετικών και αρνητικών δεδομένων που έχουμε στη διάθεσή μας. Η υλοποίηση του αλγορίθμου ακολουθεί τα παρακάτω βήματα:
Βήμα 1: Για κάθε δείγμα p_i στην κλάση μειοψηφίας P, υπολογίζοντα οι m nearest neighbors εντός του συνόλου T. Ο αριθμός των δειγμάτων πλειοψηφίας που εμπεριέχονται στους m nearest neighbors ορίζεται ως m' τέτοιο ώστε $0 \leq m' < m$.
Βήμα 2: Αν $m'=m$ σημαίνει ότι και οι m nearest neighbors ανήκουν στην κλάση πλειοψηφίας, άρα είναι θόρυβος και αποκλείονται από τα παρακάτω βήματα. Αν $m/2 \leq m' < m$, αν δηλαδή ο αριθμός των nearest neighbors που ανήκουν στην κλάση πλειοψηφίας είναι μεγαλύτερος από αυτό που ανήκουν στην κλάση μειοψηφίας, τότε το στοιχείο αυτό p_i μεταφέρεται στον πίνακα DANGER. Αν $0 \leq m' < m/2$, τότε το στοιχείο p_i είναι ασφαλές και δε χρειάζεται να προχωρήσει στα επόμενα βήματα.
Βήμα 3: Τα στοιχεία που βρίσκονται στον πίνακα DANGER είναι αυτά που αναζητούσαμε και από τα οποία θα παραχθούν τα νέα «συνθετικά» στοιχεία με $0 \leq d_{num} \leq p_{num}$.
Βήμα 4: Στο σημείο αυτό, παράγονται $s^* d_{num}$ θετικά στοιχεία από τον πίνακα DANGER, όπου s είναι ένας ακέραιος στο διάστημα [1,k]. Για κάθε στοιχείο p_i', επιλέγονται s από τους k nearest neighbors. Αρχικά υπολογίζεται η διαφορά μεταξύ των στοιχείων p_i' και των s nearest neighbors και το αποτέλεσμα πολλαπλασιάζεται με έναν τυχαίο αριθμό στο διάστημα [0,1]. Το αποτέλεσμα αυτό τέλος προστίθεται στο στοιχείο εκείνο για το οποίο ενδιαφερόμαστε να δημιουργήσουμε ένα νέο συνθετικό.

Generative Over-sampling

Η μέθοδος Generative Over-sampling είναι και αυτή μια μέθοδος για την επίλυση του Class Imbalance Problem. Το βασικό στοιχείο αυτής της μεθόδου είναι η χρησιμοποίηση μια κατανομής πιθανοτήτων με αναδειγματοληψία. Το αρχικό λοιπόν σύνολο δεδομένων X χωρίζεται σε δύο σύνολα X_{train} και X_{test} , τα οποία αντιπροσωπεύουν τα στοιχεία για εκπαίδευση και έλεγχο αντίστοιχα. Το αρχικό σύνολο δεδομένων μπορεί να εκφραστεί ως η ένωση των συνόλων Pκαι Q όπου:

- P είναι ένα σύνολο σημείων από μία κλάση της οποίας η πιθανότητα διανομής είναι άγνωστη.
- Q είναι ένα σύνολο σημείων από μια δεύτερη κλάση της οποίας η πιθανότητα διανομής είναι επίσης άγνωστη.
- λ = | P |/| X | αντιστοιχεί στην πιθανότητα ένα σημείο να προέρχεται από το P.

Κάθε δυαδικός ταξινομητής θα πρέπει να έχει τη δυνατότητα να διακρίνει τα στοιχεία του P από τα στοιχεία του Q. Σε μια ειδική περίπτωση ενός δυαδικού προβλήματος ταξινόμησης, όπου οι κλάσεις είναι εξαιρετικά ασύμμετρες, μπορεί να γίνει τυποποίηση σε όρους P και Q, ορίζοντας P την κλάση μειοψηφίας και Q την κλάση πλειοψηφίας. Ο στόχος του over-sampling είναι η αύξηση του αριθμού των σημείων που προέρχονται από το P. Αυτό συνεπάγεται ότι σε ένα ιδανικό over-sampling θα πρέπει να προστεθούν σημεία στο X_{train} που έχουν εκδοθεί άμεσα από τη διανομή η οποία παρήγαγε το P. Μια θεμελιώδης προϋπόθεση της Generative Over-sampling είναι η ύπαρξη των διανομών πιθανότητας, οι οποίες μοντελοποιούν με ακρίβεια τις κατανομές δεδομένων.

Η μέθοδος Generative Over-sampling λειτουργεί όπως παρακάτω:

1. Επιλέγεται μια κατανομή πιθανότητας, προκειμένου να μοντελοποιήσει την κλάση μειοψηφίας.
2. Με βάση τα δεδομένα εκπαίδευσης, μαθαίνουμε τις παραμέτρους για την κατανομή της πιθανότητας.
3. Προστίθενται τεχνητά σημεία δεδομένων στο σύνολο δεδομένων στο οποίο έχει γίνει αναδειγματοληψία δημιουργώντας σημεία από την κατανομή πιθανοτήτων μέχρι τον επιθυμητό αριθμό των σημείων της κλάσης μειοψηφίας στο σύνολο εκπαίδευσης.

3. Sentiment Analysis Data Sets

Η ανάπτυξη των Big Data και Analysis τα τελευταία χρόνια, έχει αναγκάσει τους επαγγελματίες να αναζητήσουν νέες μεθόδους για να παράξουν νέες ιδέες συνυπολογίζοντας τη στάση των καταναλωτών απέναντι στα προϊόντα και τις υπηρεσίες ή και τα εμπορικά σήματα γενικότερα. Βλέπουμε τελευταία πλήθος εταιρειών να χρησιμοποιούν πολλά δεδομένα, τα οποία λαμβάνουν από τα μέσα κοινωνικής δικτύωσης. Αναπτύσσουν λοιπόν λογισμικό το οποίο ανακτά το κείμενο από τα μέσα αυτά κοινωνικής δικτύωσης και χρησιμοποιώντας εργαλεία μηχανικής μάθησης, προκύπτουν δεδομένα τα οποία χρησιμοποιούνται με σκοπό να παρακολουθήσουν και να προβλέψουν την κοινή γνώμη και την αγορά. Η ανάλυση όμως αυτής της πληροφορίας και η μετατροπή της σε χρήσιμη γνώση είναι μια εξαιρετικά δύσκολη διαδικασία ειδικά για την ελληνική γλώσσα, δεδομένης της πολυμορφίας της και της πολυπλοκότητάς της. Στην προκειμένη περίπτωση, για την παρούσα εργασία, χρησιμοποιήθηκαν δεδομένα από τις μεγάλες ελληνικές τράπεζες χρησιμοποιώντας το επίσημο API του Twitter .

Η αυξημένη χρήση των μέσων κοινωνικής δικτύωσης τα τελευταία χρόνια, έχει δημιουργήσει ένα τεράστιο πλούτο περιεχομένου, το οποίο μπορεί να χρησιμοποιηθεί για την social media analysis. Social media analytics (SMA) είναι μια εφαρμογή εξόρυξης δεδομένων και ανάκτησης πληροφορίας για την εξαγωγή συμπερασμάτων μεταξύ των μελών τέτοιων κοινωνικών μέσων, όπως τα μέσα κοινωνικής δικτύωσης και οι micro-blogging εφαρμογές.

Τα δεδομένα αυτά που προκύπτουν από τα μέσα κοινωνικής δικτύωσης αποτελούν για τις επιχειρήσεις μια πλούσια πηγή πληροφοριών, ώστε να λάβουν αποφάσεις για θέματα marketing, πολιτικής της εταιρείας, χρηματοδότησης, πολιτικές ασφαλείας αλλά και πρόβλεψη της εταιρικής απόδοσης.

Η sentiment analysis είναι ένα σημαντικό ερευνητικό πεδίο της SMA, η οποία επικεντρώνεται στην ανίχνευση συναισθημάτων ή γνωμών από αυτά τα μέλη. Η έρευνα έχει επικεντρωθεί στην ακρίβεια πρόβλεψης των κοινωνικών συναισθημάτων χρησιμοποιώντας αλγόριθμους μηχανικής μάθησης ή ταξινόμηση αυτών των συναισθημάτων και γνωμών βάσει ενός λεξικού.

Για την έρευνά μας, συλλέχθηκαν και αναλύθηκαν τυχαία μια σειρά από πάνω από 9.000 tweets κατά το χρονικό διάστημα μεταξύ 2013 και 2015, με τη χρήση του επίσημου API του Twitter. Η διαδικασία συλλογής δεδομένων επικεντρώθηκε στη συλλογή tweets που αναφέρονται σε τέσσερις κορυφαίες τράπεζες στην Ελλάδα, και συγκεκριμένα στην Εθνική Τράπεζα της Ελλάδος (ΕΤΕ), την Alpha Bank, την Τράπεζα Πειραιώς και τη Eurobank. Το αποτέλεσμα αυτό πραγματοποιήθηκε με τη χρησιμοποίηση του επίσημου API του Twitter χρησιμοποιώντας σαν λέξη-κλειδί στο φιλτράρισμα τους όρους "National Bank", "NBG", "Alpha Bank", "Piraeus Bank" και "Eurobank". Το σύνολο δεδομένων που προκύπτει, υποβάλλεται σε μια σειρά από καθαρισμό

και προ-επεξεργασίας δεδομένων. Η διαδικασία επεξεργασίας των δεδομένων περιλαμβάνει text tokenization, δηλαδή εξαγωγή των λέξεων μέσα από τις προτάσεις, διαγραφή των σημείων στίξης και των λέξεων με πάνω από 3 χαρακτήρες και εξαγωγή της ρίζας από κάθε λέξη.
Ως εκ τούτου, προκύπτει ένα σύνολο από 9552 αρχεία, όπου κάθε αρχείο περιλαμβάνει ένα σχόλιο από ένα μόνο tweet. Από αυτά λοιπόν τα 9552 σχόλια, 3132 είναι θετικά και 6460 είναι αρνητικά σχόλια.
Βασική προϋπόθεση ώστε να κάνουμε sentiment analysis μέσω του αλγόριθμου μηχανικής μάθησης, είναι να αποκτήσουμε μια μαθηματική αναπαράσταση του συνόλου τέτοια ώστε κάθε έγγραφο να μπορεί να παρασταθεί με ένα σημείο στον πολυδιάστατο χώρο. Η προσέγγιση αυτή έγινε με το πρότυπο VSM. Η βασική ιδέα είναι να μετατραπεί κάθε έγγραφο σε ένα πίνακα, ο οποίος να περιέχει μόνο τις λέξεις που περιέχονται στο έγγραφο και τη συχνότητα εμφάνισής τους. Κάθε έγγραφο αποτελείται από λέξεις, οι οποίες έχουν προέλθει από tokenization.
Η μορφοποίηση που γίνεται είναι όπως παρακάτω :

$$\varphi: d \to \varphi(d) = [tf(t_1, d), \dots, tf(t_M, d)] \in \mathbb{R}^M \ (1)$$

όπου $tf(t_i, d_j)$ είναι η συχνότητα με την οποία εμφανίζεται ο όρος t_i στο κείμενο d_j.

$$tf(t_i, d_j) = \frac{f(t_i, d_j)}{\max\{f(t, d_j): t \in d_j\}} \ (2)$$

όπου $f(t_i, d_j)$ είναι η απόλυτη συχνότητα εμφάνισης του όρου t_i στο αρχείο d_j.
Έχοντας λοιπό ένα σύνολο κειμένων D και ένα λεξικό T, όπως φαίνεται παρακάτω :

$$D = \{d_1, \dots, d_n\} \ (3)$$

$$T = \{t_1, \dots, t_M\} \ (4).$$

η προσέγγιση σύμφωνα με το πρότυπο VSM, είναι η δημιουργία ενός πίνακα όπως φαίνεται παρακάτω :

$$D = \begin{bmatrix} tf(t_1, d_1) & \cdots & tf(t_M, d_1) \\ \vdots & \ddots & \vdots \\ tf(t_1, d_n) & \cdots & tf(t_M, d_n) \end{bmatrix} (5)$$

Ενσωματώσαμε σε κάθε όρο το (tf-idf), κατά τον οποίο σε κάθε όρο t_i εφαρμόζεται ένα βάρος της μορφής :

$$w_i = idf(t_i, D) = \log \frac{|D|}{|\{d \in D: t_i \in d\}|} \ (6)$$

Αλγόριθμοι

Παρακάτω παρουσιάζονται οι αλγόριθμοι για under-sampling και over-sampling. Η ανάπτυξη και η εκτέλεση των αλγορίθμων αυτών έγινε σε περιβάλλον Matlab της έκδοσης R2013b.
Για κάθε αλγόριθμο, υπάρχουν δύο τμήματα κώδικα. Το πρώτο, το οποίο είναι της μορφής αλγόριθμος_Β είναι το τμήμα εκείνο, στο οποίο παράγονται τα δεδομένα τα οποία θα χρησιμοποιηθούν από τη συνάρτηση, η οποία ακολουθεί. Αυτά τα δεδομένα είναι κυρίως ο διαχωρισμός του συνόλου των δεδομένων σε θετικά και αρνητικά δείγματα και ο αριθμός των nearest neighbors (κυρίως για τους oversampling αλγόριθμους), ώστε να επιτύχουμε όσο το δυνατό ίδιο αριθμό θετικών και αρνητικών δειγμάτων στις δυο κλάσεις.
Για την υλοποίηση των αλγορίθμων χρησιμοποιήθηκαν ως είσοδοι δεδομένα, τα οποία προήλθαν από το Tweeter χρησιμοποιώντας το επίσημο API του. Τα δεδομένα λοιπόν τα έχουμε σε δύο αρχεία .mat, τα οποία τα εισάγουμε στους αλγόριθμους και παίρνουμε όλα τα δεδομένα τα οποία και επεξεργαζόμαστε. Τα δύο αρχεία είναι το sentiment_corpus_tfidf_vectors.mat και το sentiment_corpus_tfidf_vectors_labels.mat. Στο πρώτο αρχείο έχουμε το σύνολο των δεδομένων τα οποία έχουμε πάρει από το Tweeter και κάθε γραμμή αυτού του πίνακα αντιστοιχεί σε ένα σχόλιο που έχει γίνει στο Tweeter και μπορεί να παρασταθεί στο χώρο. Το δεύτερο αρχείο περιλαμβάνει ένα σύνολο από labels. Ο πίνακας αυτός έχει τόσα labels όσα και τα σχόλια στον πρώτο πίνακα. Ο πίνακας αυτός περιλαμβάνει τι τιμές 1 (θετικό σχόλιο), -1 (αρνητικό σχόλιο) και 0 (ουδέτερο σχόλιο).

3.1 Αλγόριθμοι under-sampling

Tomek Links

Στο τμήμα αυτού του αλγόριθμου, αφού γίνει ανάγνωση των δεδομένων από τα δύο .mat αρχεία, στη συνέχεια γίνεται η επεξεργασία τους. Αρχικά, γίνεται ο διαχωρισμός των δεδομένων σε θετικά και αρνητικά δεδομένα με τη βοήθεια των labels που έχουμε στη διάθεσή μας. Στη συνέχεια, εφαρμόζεται η συνάρτηση Tomek Links,η οποία βρίσκει όλα τα σημεία τα οποία δημιουργούν ζεύγη Tomek Links και τα τοποθετεί στον πίνακα T. Στη συνέχεια χρησιμοποιούμε τον πίνακα T, ώστε να αφαιρέσουμε από τον πίνακα των αρνητικών δειγμάτων που έχουμε στη διάθεσή μας, τα σημεία εκείνα που συμμετέχουν στα ζεύγη Tomek Links. Τέλος, λαμβάνουμε το νέο ισορροπημένο πλέον πίνακα X με τα θετικά και τα αρνητικά στοιχεία καθώς και τα αντίστοιχα labels αυτών.

TOMEK_LINKS_B

```
load('sentiment_corpus_tfidf_vectors.mat');
load('sentiment_corpus_tfidf_vectors_labels.mat');
%ola ta data
X1 = double(tfidf_vectors);
%ola ta labels
L = double(tfidf_vectors_labels);
Ipos=find(L==1);
Ineg=find(L==-1);
%ta thetika
Xpos=X1(Ipos,:);
%ta arnitika
Xneg=X1(Ineg,:);
%Tomek Links
[T]=tomek_links_function(Xneg);
N = size(Xneg,1);
T1=T(:);
T2=unique(T1);
T3=setdiff([1:N],T2);
NEG=Xneg(T3,:);
X=[NEG;Xpos];
n_neg=length(NEG);
l=n_neg+length(Xpos);
LABELS=zeros(l,1);
for i=1:1:n_neg
    LABELS(i)=-1;
end
for j=n_neg+1:1:l
    LABELS(j)=1;
end
```

Ακολουθεί η συνάρτηση που καλείται από τον παραπάνω αλγόριθμο. Σε αυτή τη συνάρτηση, υπολογίζεται η απόσταση των επιμέρους σημείων (x,y) και αν δεν υπάρχει σημείο z τέτοιο ώστε να ισχύει η συνθήκη d(x,z)<d(x,y) ή d(y,z)<d(x,y), όπου d η απόσταση μεταξύ των σημείων, τότε τα σημεία αυτά δημιουργούν ένα Tomek Links και απομακρύνονται από το δείγμα μας.

tomek_links_function

```
function [T]=tomek_links_function(Xneg)
N = size(Xneg,1);
% Initialize matrix T to the empty matrix.
T = [];
% Loop through the various pairs of datapoints indices (i,j)
such that j<i.
for i = 1:1:N
    for j = 1:1:i-1
        % fprintf('Processing pair: (%d,%d)\n',i,j);
```

```
        % Get the corresponding pair of datapoints.
        x_i = Xneg(i,:);
        x_j = Xneg(j,:);
        % Compute the distance between datapoints x_i and x_j.
        d_ij = dist(x_i,x_j');
        % Get a vector of indices pointing to the remaining
set of
        % datapoints.
        Irem = setdiff([1:N],[i j]);
        % Get the corresponding sub-matrix of remaining
datapoints.
        Xrem = Xneg(Irem,:);
        % Compute the distance for each pair of datapoints
formed between
        % x_i and any other element of Xrest.
        D_i_rem  = dist(x_i,Xrem');
        % Compute the distance for each pair of datapoints
formed between
        % x_j and any other element of Xrest.
        D_j_rem = dist(x_j,Xrem');
        % Form the sub-matrix of D_i_rem which contains
distances less than
        % d_ij.
        D_i_rem_less = D_i_rem(D_i_rem < d_ij);
        % Form the sub-matrix of D_j_rem which contains
distances less than
        % d_ij.
        D_j_rem_less = D_j_rem(D_j_rem < d_ij);
        % If both vectors D_i_rem_less and D_j_rem_less are
empty then a
        % new Tomek link has just been identified.
        if(isempty(D_i_rem_less)&&isempty(D_j_rem_less))
            T = [T;[i j]];
        end;
    end;
end;
```

One-Sided Selection (OSS)

Στο τμήμα αυτού του αλγόριθμου, αφού γίνει ανάγνωση των δεδομένων από τα δύο .mat αρχεία όπως αναφέραμε παραπάνω, στη συνέχεια γίνεται η επεξεργασία τους. Αρχικά, γίνεται ο διαχωρισμός των δεδομένων σε θετικά και αρνητικά δεδομένα με τη βοήθεια των labels που έχουμε στη διάθεσή μας. Στη συνέχεια εφαρμόζεται η συνάρτηση oss_function, η οποία επιστρέφει ένα νέο μικρότερο πίνακα με τα αρνητικά δείγματα. Σε αυτό το μικρότερο πλέον πίνακα αρνητικών δειγμάτων, εφαρμόζεται η συνάρτηση tomek_links_function, η οποία όπως παραπάνω βρίσκει τα ζεύγη στο δείγμα για τα οποία ισχύει η συνθήκη tomek links. Αφού βρεθούν τα ζεύγη αυτά, απομακρύνονται από το δείγμα και προκύπτει ο νέος πίνακας δεδομένων με τα αντίστοιχα labels αυτών.

OSS_B

```
load('sentiment_corpus_tfidf_vectors.mat');
load('sentiment_corpus_tfidf_vectors_labels.mat');
%ola ta data
X1 = double(tfidf_vectors);
%ola ta labels
L = double(tfidf_vectors_labels);
%ola ta thetika
Ipos=find(L==1);
%ola ta arnitika
Ineg=find(L==-1);
%pinakas me ola ta thetika stoixeia
Xpos=X1(Ipos,:);
%pinakas me ola ta arnitika stoixeia
Xneg1=X1(Ineg,:);
%arithmos thetikwn stoixeiwn
n_pos=length(Ipos);
%arithmos arnitikwn stoixeiwn
n_neg=length(Ineg);
%klisi tis sinartisis
[Xneg]=oss_function(X1,L,Ipos,Ineg,Xpos,Xneg1,n_pos,n_neg);
%akolouthei klisi Tomek Links
[T]=tomek_links_function(Xneg);

T1=T(:);
T2=unique(T1);
elements=length(Xneg);
T3=setdiff([1:elements],T2);
NEG=Xneg(T3,:);
X=[NEG;Xpos];
n_neg=length(NEG);
l=n_neg+length(Xpos);
LABELS=zeros(l,1);
for i=1:1:n_neg
    LABELS(i)=-1;
end
for j=n_neg+1:1:l
    LABELS(j)=1;
end
```

Ακολουθεί η συνάρτηση, η οποία καλείται από τον παραπάνω αλγόριθμο. Αρχικά δημιουργείται ένα δείγμα, το οποίο αποτελείται από όλα τα θετικά δείγματα και ένα τυχαίο αρνητικό δείγμα. Στη συνέχεια βρίσκουμε τον 1-NN συγκρίνοντας το μικρότερο αυτό δείγμα με το αρχικό και συγκρίνουμε τα labels αυτών που προκύπτουν με τα πραγματικά. Τα δείγματα, τα οποία δεν έχουν ταξινομηθεί σωστά, προστίθενται στο αρχικό μικρό δείγμα και προκύπτει έτσι ένας νέος μικρότερος πίνακας με αρνητικά δείγματα.

oss_function

```
Function
[Xneg]=oss_function(X1,L,Ipos,Ineg,Xpos,Xneg1,n_pos,n_neg)
%o pinakas C
X=zeros(n_pos+1,400);
for i=1:1:n_pos
    X(i,:)=Xpos(i,:);
end;
p=randperm(n_neg,1);
X(n_pos+1,:)=Xneg1(p,:);
%oi nearest neighbors
IDX = knnsearch(X,X1);
IDX2=zeros(length(L),1);
for k=1:1:length(L)
   IDX2(k)=L(IDX(k));
end;

Xneg=[];
for j=1:1:length(L)
    if (IDX2(j)~=L(j))
        X=[X;X1(j,:)];
        if (L(j)==-1)
            Xneg=[Xneg;X1(j,:)];
        end

    end;
end;
Xneg=[Xneg;X(n_pos+1,:)];
```

3.2 Αλγόριθμοι over-sampling

Cluster Based Over-sampling

Στο τμήμα αυτού του αλγόριθμου, αφού γίνει ανάγνωση των δεδομένων από τα δύο .mat αρχεία όπως αναφέραμε παραπάνω, στη συνέχεια γίνεται η επεξεργασία τους. Αρχικά, γίνεται ο διαχωρισμός των δεδομένων σε θετικά και αρνητικά δεδομένα με τη βοήθεια των labels που έχουμε στη διάθεσή μας. Στη συνέχεια ορίζουμε τον αριθμό των nearest neighbors που πρέπει να χρησιμοποιήσουμε, ώστε να επιτύχουμε τον ίδιο αριθμό θετικών και αρνητικών στοιχείων. Πόσες φορές δηλαδή πρέπει να αυξήσουμε κάθε ένα από τα σύνολα για να επιτευχθεί αυτός ο στόχος. Στη συνέχεια καλούμε τη συνάρτηση την οποία έχουμε δημιουργήσει παρακάτω και λαμβάνουμε το νέο ισορροπημένο πλέον πίνακα με τα θετικά και τα αρνητικά στοιχεία καθώς και τα αντίστοιχα labels αυτών.

CLUSTER_BASED_B

```
load('sentiment_corpus_tfidf_vectors.mat');
load('sentiment_corpus_tfidf_vectors_labels.mat');
%ola ta data
X1 = double(tfidf_vectors);
%ola ta labels
L = double(tfidf_vectors_labels);
%ola ta thetika
Ipos=find(L==1);
%ola ta arnitika
Ineg=find(L==-1);
%pinakas me ola ta thetika stoixeia
Xpos=X1(Ipos,:);
%pinakas me ola ta arnitika stoixeia
Xneg=X1(Ineg,:);
%arithmos thetikwn stoixeiwn
n_pos=length(Ipos);
%arithmos arnitikwn stoixeiwn
n_neg=length(Ineg);
%nearest neighbors gia to sinolo twn thetikwn stoixeiwn
nn_pos=3;
%nearest neighbors gia to sinolo twn arnitikwn stoixeiwn
stoixeiwn
nn_neg=2;
%klisi tis sinartisis
[POS1,NEG1,NEG2]=cluster_based_func(X1,L,Ipos,Ineg,Xpos,Xneg,n
_pos,n_neg,nn_pos,nn_neg);
%dimiourgia enos eniaiou pinaka stoixeiwn
%sto panw meros ta arnitika kai sto katw meros ta thetika
%stoixeia
NEG1=[NEG1;NEG2];
l_pos=length(POS1);
l_neg=length(NEG1);
l=l_pos+l_neg;
X=[NEG1;POS1];
%dimiourgia tou pinaka LABELS
%stis theseis pou exoume arnitika stoixeia vazoume -1 kai ekei
%pou exoume thetika stoixeia vazoume 1
LABELS=zeros(l,1);
for i=1:1:l_neg
    LABELS(i)=-1;
end
for j=l_neg+1:1:l
    LABELS(j)=1;
end
```

Ακολουθεί η συνάρτηση που καλείται από τον παραπάνω αλγόριθμο. Αρχικά γίνεται ο χωρισμός των θετικών στοιχείων (κλάση μειοψηφίας) σε τρεις ομάδες και των αρνητικών σε δύο ομάδες (κλάση πλειοψηφίας). Στη συνέχεια γίνεται αναζήτηση ποια είναι η ομάδα από τα αρνητικά στοιχεία με τα περισσότερα στοιχεία και κάνουμε over-sampling σε κάθε μια από τις ομάδες

αυτές, θετικές και αρνητικές, ώστε να επιτύχουμε τον αριθμό των στοιχείων της μέγιστης ομάδας κάνοντας τυχαία αναπαραγωγή δεδομένων από τα στοιχεία της εκάστοτε ομάδας και μόνο γι' αυτή.

---

cluster_based_func

```
function
[POS1,NEG1,NEG2]=cluster_based_func(X1,L,Ipos,Ineg,Xpos,Xneg,n
_pos,n_neg,nn_pos,nn_neg)
%xorismos twn thetikwn stoixeiwn se omades
POS = kmeans(Xpos,nn_pos);
P1=find(POS==1);
P2=find(POS==2);
P3=find(POS==3);
%1h omada
POS1=X1(P1,:);
%2h omada
POS2=X1(P2,:);
%3h omada
POS3=X1(P3,:);
n_POS1=size(POS1,1);
n_POS2=size(POS2,1);
n_POS3=size(POS3,1);
%xorismos twn arnitikwn stoixeiwn se omades
NEG = kmeans(Xneg,nn_neg);
N1=find(NEG==1);
N2=find(NEG==2);
NEG1=X1(N1,:);
NEG2=X1(N2,:);
n_NEG1=size(NEG1,1);
n_NEG2=size(NEG2,1);
max_NEG=max(n_NEG1,n_NEG2);
min_NEG=min(n_NEG1,n_NEG2);
DN_NEG=max_NEG-min_NEG;
if (length(NEG1)==max_NEG)
        DI_NEG=randi([1 n_NEG2],DN_NEG,1);
        NEG2=[NEG2;NEG2(DI_NEG,:)];

else
        DI_NEG=randi([1 n_NEG1],DN_NEG,1);
        NEG1=[NEG1;NEG1(DI_NEG,:)];

end
%euresi diaforas metaksi tou arithmou twn stoixeiwn kathe
%mikrooteris omadas
%apo ti megaliteri omada wste oles oi omades twn thetikwn
%stoixeiwn na exoun
%ton idio arithmo stoixeiwn
DN_POS1=fix((nn_neg/nn_pos)*max_NEG)-length(POS1);
DI_POS1=randi([1 n_POS1],DN_POS1,1);
POS1=[POS1;POS1(DI_POS1,:)];
```

```
DN_POS2=fix((nn_neg/nn_pos)*max_NEG)-length(POS2);
DI_POS2=randi([1 n_POS2],DN_POS2,1);
POS2=[POS2;POS2(DI_POS2,:)];

DN_POS3=fix((nn_neg/nn_pos)*max_NEG)-length(POS3);

DI_POS3=randi([1 n_POS3],DN_POS3,1);
POS3=[POS3;POS3(DI_POS3,:)];

POS1=[POS1;POS2];
POS1=[POS1;POS3];

end
```

Synthetic Minority Over-sampling Technique (SMOTE)

Σε αυτό τον αλγόριθμο αφού γίνει η ανάγνωση των δεδομένων από τα δύο αρχεία, γίνεται στη συνέχεια ο διαχωρισμός σε θετικά και αρνητικά δεδομένα με τη βοήθεια των labels που έχουμε στη διάθεσή μας. Κατόπιν υπολογίζουμε πόσες φορές είναι μεγαλύτερος ο πίνακας των αρνητικών στοιχείων (κλάση πλειοψηφίας) σε σχέση με των θετικών (κλάση μειοψηφίας), ώστε να υπολογίσουμε τον αριθμό των nearest neighbors που θα πρέπει να υπολογίσουμε για να πετύχουμε όσο το δυνατό πιο ισορροπημένες κλάσεις. Αφού γίνει η κλήση της συνάρτησης που έχει δημιουργηθεί παρακάτω, δημιουργείται ο νέος πίνακας με το νέο αριθμό θετικών στοιχείων και όλα τα αρνητικά στοιχεία, ορίζοντας και τα νέα αντίστοιχα labels.

SMOTE_B

```
load('sentiment_corpus_tfidf_vectors.mat');
load('sentiment_corpus_tfidf_vectors_labels.mat');
%ola ta data
X1 = double(tfidf_vectors);
%ola ta labels
L = double(tfidf_vectors_labels);
Ipos=find(L==1);
Ineg=find(L==-1);
%ta thetika
Xpos=X1(Ipos,:);
%ta arnitika
Xneg=X1(Ineg,:);
%arithmos thetikwn stoixeiwn
n_pos=length(Ipos);
%arithmos arnitikwn stoixeiwn
n_neg=length(Ineg);
%nearest neighbors
nn=round(n_neg/n_pos);
%klisi sinartisis
```

```
[synthetic]=smote_function(X1,L,Ipos,Ineg,Xpos,Xneg,n_pos,n_ne
g,nn);
%o pinakas twn thetikwn stoixeiwn, o opoios perilamvanei ola
%ta thetika mazi me ta nea sinthetika pou parixthisan
Xpos=[Xpos;synthetic];
%dimiourgia enos sinolikou pinaka me ta arnitika stoixeia
%sto panw meros kai ta thetika sto katw meros
X=[Xneg;Xpos];
l=n_neg+length(Xpos);
%dimiourgia tou pinaka LABELS
%stis theseis pou exoume arnitika stoixeia vazoume -1 kai ekei
%pou exoume thetika stoixeia vazoume 1
LABELS=zeros(l,1);
for i=1:1:n_neg
    LABELS(i)=-1;
end
for j=n_neg+1:1:l
    LABELS(j)=1;
end
```

Παρακάτω ορίζεται η συνάρτηση την οποία καλούμε παραπάνω. Στη συνάρτηση αυτή γίνεται η δημιουργία των νέων «συνθετικών» δεδομένων τα οποία θα έρθουν και θα προστεθούν στην κλάση των θετικών στοιχείων (κλάση μειοψηφίας). Αρχικά υπολογίζεται η διαφορά ανάμεσα στο δείγμα που μας ενδιαφέρει και τον Nearest Neighbor αυτού. Στη συνέχεια πολλαπλασιάζεται αυτή η διαφορά με έναν τυχαίο αριθμό μεταξύ του 0 και το αποτέλεσμα της οποίας προστίθεται στο διάνυσμα το οποίο εξετάζουμε. Αυτή η διαδικασία γίνεται για κάθε στοιχείο της κλάσης μειοψηφίας.

---

smote_function

```
function
[synthetic]=smote_function(X1,L,Ipos,Ineg,Xpos,Xneg,n_pos,n_ne
g,nn)
%euresi arithmou nearest neighbors wste na mporesoume na
paraksoume
%tosa synthetika stoixeia me skopo na proseggisoume ton
arithmo twn
%arnitikwn stoixeiwn
N1=n_neg/n_pos;
num_dig = 2;
N_rounded = round(N1*(10^num_dig))/(10^num_dig);
N=N_rounded*100;
if (N<100)
    n_pos=(N/100)*n_pos;
    N=100;
end
N=round(N/100);

NN=knnsearch(Xpos,Xpos,'k',nn);
```

```
gen_samples=1;

synthetic=zeros;

%sinartisi paragogis sinthetwn deigmatwn
K=round(N/nn);
for rows=1:1:n_pos
    while (K~=0)
        nnn=randperm(nn,1);

        for cols=1:1:400
        xmin=0;
        xmax=1;
        gap=xmin+rand(1,1)*(xmax-xmin);
        dif=Xpos(NN(nnn),cols)-Xpos(rows,cols);
        synthetic(gen_samples,cols)=Xpos(rows,cols)+gap*dif;
        end

        gen_samples=gen_samples+1;
        K=K-1;
    end
    K=round(N/nn);
end
end
```

Borderline – SMOTE

Σε αυτό τον αλγόριθμο, όπως και σε όλους τους παραπάνω αλγόριθμους over-sampling, γίνεται η ανάγνωση των δεδομένων που θα χρησιμοποιήσουμε για επεξεργασία. Κατόπιν γίνεται ο διαχωρισμός των δεδομένων σε θετικά και αρνητικά δεδομένα και κατόπιν γίνεται εύρεση πόσες φορές περισσότερο είναι μεγαλύτερος ο πίνακας των αρνητικών στοιχείων (κλάση πλειοψηφίας) από τα θετικά (κλάση μειοψηφίας), ώστε να δούμε πόσους nearest neighbors θα χρησιμοποιήσουμε προκειμένου να πλησιάσει ο αριθμός των στοιχείων της κλάσης μειοψηφίας τον αριθμό των στοιχείων της κλάσης πλειοψηφίας και να έχουμε ισορροπία. Ο αριθμός των nearest neighbors που προκύπτει από τους υπολογισμούς είναι ο αριθμός 2. Αυτός ο αριθμός όμως είναι ιδιαίτερα μικρός για να λειτουργήσει η απαραίτητη συνάρτηση και να παραχθούν νέα «συνθετικά» δεδομένα. Γι' αυτό ορίστηκε ο αριθμός των nearest neighbors 3 και η συνάρτηση λειτουργεί για κάθε ακέραιο αριθμό μεγαλύτερο του 3. Αφού γίνει η κλήση της συνάρτηση που έχουμε δημιουργήσει παρακάτω, δημιουργείται ο νέος πλέον πίνακας με το νέο αριθμό θετικών στοιχείων και όλα τα αρνητικά, καθώς και τα αντίστοιχα labels αυτών των στοιχείων.

BORDERLINE_SMOTE_B

```
load('sentiment_corpus_tfidf_vectors.mat');
load('sentiment_corpus_tfidf_vectors_labels.mat');
%ola ta data
X1 = double(tfidf_vectors);
%ola ta labels
L = double(tfidf_vectors_labels);
%ta thetika
Ipos=find(L==1);
%ta arnitika
Ineg=find(L==-1);
%o pinakas twn thetikwn stoixeiwn
Xpos=X1(Ipos,:);
%o pinakas twn arnitikwn stoixeiwn
Xneg=X1(Ineg,:);
%arithmos thetikwn stoixeiwn
n_pos=length(Ipos);
%arithmos arnitikwn stoixeiwn
n_neg=length(Ineg);
%arithmos nearest neighbors
%nn=fix(n_neg/n_pos);
nn=3;
%klisi sinartisis paragwgis sinthetwn stoixeiwn
[synthetic]=borderline_smote_function(X1,L,Ipos,Ineg,Xpos,Xneg
,n_pos,n_neg,nn);
%o pinakas twn thetikwn stoixeiwn, o opoios perilamvanei ola
%ta thetika mazi me ta nea sinthetika pou parixthisan
Xpos=[Xpos;synthetic];
%dimiourgia enos sinolikou pinaka me ta arnitika stoixeia
%sto panw meros kai ta thetika sto katw meros
X=[Xneg;Xpos];
l=n_neg+length(Xpos);
%dimiourgia tou pinaka LABELS
%stis theseis pou exoume arnitika stoixeia vazoume -1 kai ekei
%pou exoume thetika stoixeia vazoume 1
LABELS=zeros(l,1);
for i=1:1:n_neg
    LABELS(i)=-1;
end
for j=n_neg+1:1:l
    LABELS(j)=1;
end
```

Παρακάτω βλέπουμε τη συνάρτηση, της οποίας η κλήση έγινε παραπάνω. Για κάθε δείγμα στην κλάση μειοψηφίας, υπολογίζοντα οι m nearest neighbors εντός του συνόλου T. Ο αριθμός των δειγμάτων πλειοψηφίας που εμπεριέχονται στους m nearest neighbors ορίζεται ως m' τέτοιο ώστε 0≤m'<m και τοποθετείται στον πίνακα m. Αν m/2≤m'<m, αν δηλαδή ο αριθμός των nearest neighbors που ανήκουν στην κλάση πλειοψηφίας είναι μεγαλύτερος από αυτό που ανήκουν στην κλάση μειοψηφίας, τότε το στοιχείο αυτό

μεταφέρεται στον πίνακα DANGER. Στη συνέχεια, τα στοιχεία τα οποία έχουμε στον πίνακα DANGER είναι αυτά τα οποία θα χρησιμοποιήσουμε για να παράξουμε τα «συνθετικά» στοιχεία. Για κάθε λοιπόν στοιχείο, επιλέγονται s από τους k nearest neighbors. Αρχικά υπολογίζεται η διαφορά μεταξύ των στοιχείων που επεξεργαζόμαστε και των s nearest neighbors και το αποτέλεσμα πολλαπλασιάζεται με έναν τυχαίο αριθμό στο διάστημα [0,1]. Το αποτέλεσμα αυτό τέλος προστίθεται στο στοιχείο εκείνο για το οποίο ενδιαφερόμαστε να δημιουργήσουμε ένα νέο συνθετικό.

---

borderline_smote_function

```
function[synthetic]=borderline_smote_function(X1,L,Ipos,Ineg,X
pos,Xneg,n_pos,n_neg,nn)
NN=knnsearch(X1,Xpos,'k',nn);
m=zeros(n_pos,1);
%elegxos an oi nearest neighbors einai melos tis majority
%class
for i=1:1:n_pos
    for j=1:1:nn

        LIA=ismember(NN(i,j),Ineg);
        if (LIA==1)
            m(i)=m(i)+1;
        end

    end
end
a=0;
t=0;
DANGER=[];
for v=1:1:n_pos
    if (m(v)==nn)
        t=1;
    elseif (m(v)<nn&&m(v)>(nn/2))
        DANGER=[DANGER;Xpos(v,:)];

        a=1;
    elseif (m(v)>=0&&m(v)<(nn/2))
        t=2;
    else
        t=3;
    end
end
if (a==1)
    %sinartisi pou paragei ta sintheta deigmata
    [rows_num,cols_num]=size(DANGER);
    %ipologismos arithmou nearest neighbors
    nn1=fix(n_pos/rows_num);
    KNN=knnsearch(Xpos,DANGER,'k',nn1);
```

```
    synthetic=zeros;
    gen_samples=1;
    for i=1:1:rows_num
        s=randperm(nn1,1);
        s1=randi([1 nn1],1,nn1);
        for j=1:1:s
            for k=1:1:cols_num
            xmin=0;
            xmax=1;
            gap=xmin+rand(1,1)*(xmax-xmin);
            dif=Xpos(KNN(s1(j)),cols_num)-
DANGER(rows_num,cols_num);

           synthetic(gen_samples,cols_num)=DANGER(rows_num,cols
           _num)+gap*dif;
            end
            gen_samples=gen_samples+1;
        end
    end
end
end
```

4. Αποτελέσματα Μετρήσεων

Για την πραγματοποίηση των μετρήσεων χρησιμοποιήθηκε ένα δείγμα από 9.552 Tweets, τα οποία αντλήθηκαν από το επίσημο API του Tweeter.
Με σκοπό λοιπόν να προσεγγίσουμε όσο το δυνατό καλύτερα το sentiment classification, μετρήσαμε την ακρίβεια ταξινόμησης για κάθε έναν από τους παρακάτω ταξινομητές :

- SVM RBF
- Linear SVM
- Random Forest
- Multi-Layer Network
- Bayesian Network
- RBF Network (RBFN)
- Naïve Bayes
- Classification via Clustering

Η παραμετροποίηση του Multi-Layer Network ταξινομητή έγινε για αριθμό κρυφών επιπέδων 1, 3 και 5. Η παραμετροποίηση του Random Forest ταξινομητή έγινε για αριθμό δέντρων 10 και 15.
Παρακάτω παρουσιάζονται τα αποτελέσματα απόδοσης όλων των παραπάνω αλγορίθμων. Για την εύρεση αυτών των αποτελεσμάτων χρησιμοποιήθηκε το περιβάλλον WEKA 8 καθώς και το WEKA immune. Τα αποτελέσματα που προέκυψαν για κάθε αλγόριθμο, χρησιμοποιώντας διαφορετικό classifier κάθε φορά, παρουσιάζονται παρακάτω :

TOMEK LINKS

Ο SVM RBF classifier για το συγκεκριμένο αλγόριθμο, εκτελέστηκε με παραμέτρους cost = 1.0 και gamma = 3.2.

OVERALL CLASSIFICATION ACCURACY		
Classifier	**Corect Classification Rate**	**Incorrect Classification Rate**
SVM RBF	88,5235	11,4765
Random Forest (Trees Number = 15)	85,8449	14,1551
Random Forest (Trees Number = 10)	85,7143	14,2857
Multi-Layer Network (Hidden Layers=5)	73,6716	26,3284
Multi-Layer Network (Hidden Layers=3)	71,7117	28,2883
Multi-Layer Network (Hidden Layers=1)	69,9695	30,0305
SVM LINEAR	85,4094	14,5906
Bayesian Network	81,2064	18,7963
RBF Network	83,0357	16,9643
Naïve Bayes	78,6803	21,3197
Classification via Clustering	61,9120	38,0880

Παρατηρώντας τα αποτελέσματα, βλέπουμε ότι το καλύτερο ποσοστό θετικής ταξινόμησης επιτεύχθηκε από τον αλγόριθμο SVM RBF classifier με ποσοστό 88,5%. Η δεύτερη καλύτερη απόδοση πραγματοποιήθηκε από τον Random Forest (Trees Number = 15) με ποσοστό 85,8% και ακολουθεί ο Random Forest (Trees Number = 10) και ο SVM LINEAR με 85,7% και 85,4%. Την πέμπτη και έκτη καλύτερη απόδοση την έχουμε στους RBF Network classifier και Bayesian Network classifier με ποσοστά 83% και81,2% αντίστοιχα.

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.912	0.921
Random Forest (Trees Number = 15)	0.874	0.927
Random Forest (Trees Number = 10)	0.881	0.914
Multi-Layer Network (Hidden Layers=5)	0.735	0.960
Multi-Layer Network (Hidden Layers=3)	0.709	0.993
Multi-Layer Network (Hidden Layers=1)	0.697	0.991
SVM LINEAR	0.872	0.921

Bayesian Network	0.841	0.893
RBF Network	0.862	0.895
Naïve Bayes	0.810	0.898
Classification via Clustering	0.674	0.854

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.826	0.809
Random Forest (Trees Number = 15)	0.819	0.712
Random Forest (Trees Number = 10)	0.799	0.736
Multi-Layer Network (Hidden Layers=5)	0.752	0.257
Multi-Layer Network (Hidden Layers=3)	0.889	0.126
Multi-Layer Network (Hidden Layers=1)	0.796	0.075
SVM LINEAR	0.808	0.710
Bayesian Network	0.736	0.638
RBF Network	0.754	0.692
Naïve Bayes	0.714	0.549
Classification via Clustering	0.268	0.114

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.916	0.865
Random Forest (Trees Number = 15)	0.889	
Random Forest (Trees Number = 10)	0.897	
Multi-Layer Network (Hidden Layers=5)	0.833	0.706
Multi-Layer Network (Hidden Layers=3)	0.827	0.656
Multi-Layer Network (Hidden Layers=1)	0.818	0.579
SVM LINEAR	0.896	0.816
Bayesian Network	0.866	0.865
RBF Network	0.878	0.812
Naïve Bayes	0.852	0.746
Classification via Clustering	0.754	0.484

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.818	0.865
Random Forest (Trees Number = 15)	0.762	
Random Forest (Trees Number = 10)	0.766	
Multi-Layer Network (Hidden Layers=5)	0.383	0.706
Multi-Layer Network (Hidden Layers=3)	0.221	0.656
Multi-Layer Network (Hidden Layers=1)	0.137	0.579
SVM LINEAR	0.756	0.816
Bayesian Network	0.683	0.865
RBF Network	0.722	0.812
Naïve Bayes	0.621	0.832
Classification via Clustering	0.160	0.484

TIME COMPLEXITY METRICS		
Classifier	**TIME TO BUILD THE MODEL (secs)**	
SVM RBF	5.32	
Random Forest (Trees Number = 15)	110.43	
Random Forest (Trees Number = 10)	74.65	
Multi-Layer Network (Hidden Layers=5)	173.69	
Multi-Layer Network (Hidden Layers=3)	106.2	
Multi-Layer Network (Hidden Layers=1)	48.7	
SVM LINEAR	0.08	
Bayesian Network	0.47	
RBF Network	1.51	
Naïve Bayes	0.27	
Classification via Clustering	0.8	

One-Sided Selection (OSS)

Ο SVM RBF classifier για το συγκεκριμένο αλγόριθμο, εκτελέστηκε με παραμέτρους cost = 1.0 και gamma = 1.2.

OVERALL CLASSIFICATION ACCURACY		
Classifier	**Corect Classification Rate**	**Incorrect Classification Rate**
SVM RBF	91,2555	8,7445
Random Forest (Trees Number = 15)	88,7640	11,2360
Random Forest (Trees Number = 10)	88,5198	11,4802
Multi-Layer Network (Hidden Layers=5)	78,1876	21,8124
Multi-Layer Network (Hidden Layers=3)	77,1128	22,8872
Multi-Layer Network (Hidden Layers=1)	76,5022	23,4978
SVM LINEAR	89,4235	10,5765
Bayesian Network	86,2482	13,7518
RBF Network	86,8100	13,1900
Naïve Bayes	85,3933	14,6067
Classification via Clustering	67,1715	32,7308

Μελετώντας τα αποτελέσματα από τον One-Sided Selection παρατηρούμε ότι η καλύτερη απόδοση είναι στον αλγόριθμο SVM RBF classifier με ποσοστό επιτυχίας 91,2%. Η δεύτερη καλύτερη απόδοση είναι στον SVM LINEAR με ποσοστό 89,4% και ακολουθεί ο Random Forest (Trees Number=15) classifier με ποσοστό επιτυχίας 88,8%. Την τέταρτη και πέμπτη καλύτερη απόδοση, την έχουν ο Random Forest (Trees Number=10) και ο RBF Network classifier με ποσοστά επιτυχίας 88,5% και 86,2% αντίστοιχα.

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.925	0.964
Random Forest (Trees Number = 15)	0.905	0.953
Random Forest (Trees Number = 10)	0.911	0.942
Multi-Layer Network (Hidden Layers=5)	0.787	0.980
Multi-Layer Network (Hidden Layers=3)	0.770	1.000
Multi-Layer Network (Hidden Layers=1)	0.777	0.972
SVM LINEAR	0.914	0.952

Bayesian Network	0.896	0.928
RBF Network	0.925	0.901
Naïve Bayes	0.901	0.909
Classification via Clustering	0.756	0.845

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.863	0.746
Random Forest (Trees Number = 15)	0.816	0.674
Random Forest (Trees Number = 10)	0.787	0.702
Multi-Layer Network (Hidden Layers=5)	0.679	0.136
Multi-Layer Network (Hidden Layers=3)	0.963	0.027
Multi-Layer Network (Hidden Layers=1)	0.500	0.093
SVM LINEAR	0.818	0.707
Bayesian Network	0.735	0.649
RBF Network	0.702	0.762
Naïve Bayes	0.695	0.674
Classification via Clustering	0.182	0.112

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.944	0.855
Random Forest (Trees Number = 15)	0.928	
Random Forest (Trees Number = 10)	0.926	
Multi-Layer Network (Hidden Layers=5)	0.873	0.614
Multi-Layer Network (Hidden Layers=3)	0.870	0.555
Multi-Layer Network (Hidden Layers=1)	0.864	0.642
SVM LINEAR	0.932	0.829
Bayesian Network	0.912	0.885
RBF Network	0.913	0.850
Naïve Bayes	0.905	0.829
Classification via Clustering	0.798	0.478

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.800	0.855
Random Forest (Trees Number = 15)	0.738	
Random Forest (Trees Number = 10)	0.742	
Multi-Layer Network (Hidden Layers=5)	0.227	0.614
Multi-Layer Network (Hidden Layers=3)	0.053	0.555
Multi-Layer Network (Hidden Layers=1)	0.156	0.642
SVM LINEAR	0.759	0.829
Bayesian Network	0.689	0.885
RBF Network	0.731	0.850
Naïve Bayes	0.684	0.872
Classification via Clustering	0.139	0.479

TIME COMPLEXITY METRICS		
Classifier	**TIME TO BUILD THE MODEL (secs)**	
SVM RBF	2.08	
Random Forest (Trees Number = 15)	77.47	
Random Forest (Trees Number = 10)	50.75	
Multi-Layer Network (Hidden Layers=5)	123.93	
Multi-Layer Network (Hidden Layers=3)	81.65	
Multi-Layer Network (Hidden Layers=1)	38.41	
SVM LINEAR	0.16	
Bayesian Network	0.92	
RBF Network	2.37	
Naïve Bayes	0.31	
Classification via Clustering	1.2	

CLUSTER BASED

Ο SVM RBF classifier για το συγκεκριμένο αλγόριθμο, εκτελέστηκε με παραμέτρους cost = 1.0 και gamma = 0.1.

OVERALL CLASSIFICATION ACCURACY		
Classifier	**Corect Classification Rate**	**Incorrect Classification Rate**
SVM RBF	63,5279	36,4721
Random Forest (Trees Number = 15)	65,6032	34,3968
Random Forest (Trees Number = 10)	65,4479	34,5521
Multi-Layer Network (Hidden Layers=5)	50,9564	49,0436
Multi-Layer Network (Hidden Layers=3)	50,2647	49,7353
Multi-Layer Network (Hidden Layers=1)	51,3517	48,6483
SVM LINEAR	63,2667	36,7333
Bayesian Network	61,8974	38,1026
RBF Network	59,9704	40,0269
Naïve Bayes	54,0199	45,9801
Classification via Clustering	52,5941	47,4059

Παρατηρώντας τα αποτελέσματα βλέπουμε ότι το καλύτερο ποσοστό σωστής ταξινόμησης επιτεύχθηκε από τον Random Forest (Trees Number=15) με ποσοστό 65,6%. Το δεύτερο καλύτερο ποσοστό το βλέπουμε στον Random Forest (Trees Number=10) με ποσοστό 65,4% και ακολουθεί ο SVM RBF και ο SVM Linear με ποσοστά 63,5% και 63,3% αντίστοιχα.
Η δυσκολία σωστής ταξινόμησης είναι αρκετά μεγάλη, διότι έχουμε μεγάλη διαφορά ανάμεσα στον αριθμό των θετικών και αρνητικών δειγμάτων. Συγκεκριμένα, έχουμε 6.460 αρνητικά και 3.132 θετικά δείγματα, σχεδόν διπλάσιο αριθμό αρνητικών από θετικά δείγματα.
Πρέπει να σημειωθεί όμως ότι και τα χαμηλά ποσοστά απόδοσης σε αυτό τον αλγόριθμο οφείλονται και στον ίδιο τον τρόπο λειτουργίας του, κατά τον οποίο έχουμε over-sampling επαναλαμβάνοντας στοιχεία που ήδη υπάρχουν στο δείγμα και όχι δημιουργώντας νέα, όπως κάνουν οι άλλοι αλγόριθμοι over-sampling.

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.616	0.716
Random Forest (Trees Number = 15)	0.619	0.813
Random Forest (Trees Number = 10)	0.618	0.808

Multi-Layer Network (Hidden Layers=5)	0.514	0.359
Multi-Layer Network (Hidden Layers=3)	0.504	0.298
Multi-Layer Network (Hidden Layers=1)	0.514	0.494
SVM LINEAR	0.612	0.727
Bayesian Network	0.613	0.647
RBF Network	0.572	0.792
Naïve Bayes	0.523	0.909
Classification via Clustering	0.523	0.589

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.661	0.554
Random Forest (Trees Number = 15)	0.727	0.499
Random Forest (Trees Number = 10)	0.723	0.501
Multi-Layer Network (Hidden Layers=5)	0.507	0.660
Multi-Layer Network (Hidden Layers=3)	0.502	0.708
Multi-Layer Network (Hidden Layers=1)	0.513	0.533
SVM LINEAR	0.664	0.538
Bayesian Network	0.626	0.591
RBF Network	0.662	0.407
Naïve Bayes	0.653	0.172
Classification via Clustering	0.530	0.463

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.663	0.635
Random Forest (Trees Number = 15)	0.703	
Random Forest (Trees Number = 10)	0.701	
Multi-Layer Network (Hidden Layers=5)	0.423	0.523
Multi-Layer Network (Hidden Layers=3)	0.374	0.518
Multi-Layer Network (Hidden Layers=1)	0.504	0.534
SVM LINEAR	0.664	0.633
Bayesian Network	0.629	0.658

RBF Network	0.664	0.619
Naïve Bayes	0.664	0.551
Classification via Clustering	0.554	0.526

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.603	0.635
Random Forest (Trees Number = 15)	0.592	
Random Forest (Trees Number = 10)	0.592	
Multi-Layer Network (Hidden Layers=5)	0.574	0.523
Multi-Layer Network (Hidden Layers=3)	0.587	0.518
Multi-Layer Network (Hidden Layers=1)	0.523	0.534
SVM LINEAR	0.594	0.633
Bayesian Network	0.608	0.658
RBF Network	0.504	0.619
Naïve Bayes	0.272	0.621
Classification via Clustering	0.494	0.526

TIME COMPLEXITY METRICS		
Classifier	**TIME TO BUILD THE MODEL (secs)**	
SVM RBF	36.68	
Random Forest (Trees Number = 15)	2110.84	
Random Forest (Trees Number = 10)	1364.21	
Multi-Layer Network (Hidden Layers=5)	541.37	
Multi-Layer Network (Hidden Layers=3)	331	
Multi-Layer Network (Hidden Layers=1)	151.57	
SVM LINEAR	0.25	
Bayesian Network	3.17	
RBF Network	4.74	
Naïve Bayes	1	
Classification via Clustering	2.35	

SMOTE

Ο SVM RBF classifier για το συγκεκριμένο αλγόριθμο, εκτελέστηκε με παραμέτρους cost = 1.0 και gamma = 1.6.

OVERALL CLASSIFICATION ACCURACY		
Classifier	**Corect Classification Rate**	**Incorrect Classification Rate**
SVM RBF	93,2179	6,7821
Random Forest (Trees Number = 15)	93,6026	6,3974
Random Forest (Trees Number = 10)	93,1311	6,8689
Multi-Layer Network (Hidden Layers=5)	53,1892	46,8108
Multi-Layer Network (Hidden Layers=3)	57,9528	42,0472
Multi-Layer Network (Hidden Layers=1)	54,7771	45,2229
SVM LINEAR	89,3963	10,6037
Bayesian Network	85,1081	14,8919
RBF Network	84,3904	15,6096
Naïve Bayes	75,3745	24,6255
Classification via Clustering	55,3961	44,577

Από τη μελέτη των παραπάνω στοιχείων, βλέπουμε ότι την καλύτερη απόδοση την έχουμε με τον Random Forest (Trees Number=15) classifier με ποσοστό 93,6%. Η δεύτερη καλύτερη είναι με τον SVM RBF classifier με ποσοστό επιτυχίας 93,2% και ακολουθεί ο Random Forest (Trees Number=10) με ποσοστό επιτυχίας 93,1%. Την τέταρτη και πέμπτη καλύτερη απόδοση, την έχουν ο SVM Linear και ο Bayesian Network classifier με ποσοστά επιτυχίας 89,4% και 85,1% αντίστοιχα.

.

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.942	0.894
Random Forest (Trees Number = 15)	0.944	0.925
Random Forest (Trees Number = 10)	0.945	0.914
Multi-Layer Network (Hidden Layers=5)	0.421	0.300
Multi-Layer Network (Hidden Layers=3)	0	0
Multi-Layer Network (Hidden Layers=1)	0.421	0.200
SVM LINEAR	0.882	0.863
Bayesian Network	0.779	0.902
RBF Network	0.762	0.914

Naïve Bayes	0.653	0.884
Classification via Clustering	0.418	0.154

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.926	0.960
Random Forest (Trees Number = 15)	0.928	0.947
Random Forest (Trees Number = 10)	0.919	0.948
Multi-Layer Network (Hidden Layers=5)	0.580	0.700
Multi-Layer Network (Hidden Layers=3)	0.580	1
Multi-Layer Network (Hidden Layers=1)	0.580	0.800
SVM LINEAR	0.902	0.917
Bayesian Network	0.920	0.814
RBF Network	0.927	0.793
Naïve Bayes	0.887	0.659
Classification via Clustering	0.579	0.854

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.917	0.927
Random Forest (Trees Number = 15)	0.934	
Random Forest (Trees Number = 10)	0.929	
Multi-Layer Network (Hidden Layers=5)	0.350	0.521
Multi-Layer Network (Hidden Layers=3)	0	0.512
Multi-Layer Network (Hidden Layers=1)	0.271	0.509
SVM LINEAR	0.872	0.890
Bayesian Network	0.836	0.925
RBF Network	0.831	0.879
Naïve Bayes	0.751	0.785
Classification via Clustering	0.225	0.499

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.943	0.927
Random Forest (Trees Number = 15)	0.938	
Random Forest (Trees Number = 10)	0.933	
Multi-Layer Network (Hidden Layers=5)	0.634	0.522
Multi-Layer Network (Hidden Layers=3)	0.734	0.512
Multi-Layer Network (Hidden Layers=1)	0.672	0.509
SVM LINEAR	0.909	0.890
Bayesian Network	0.864	0.925
RBF Network	0.855	0.879
Naïve Bayes	0.756	0.854
Classification via Clustering	0.687	0.499

TIME COMPLEXITY METRICS		
Classifier	**TIME TO BUILD THE MODEL (secs)**	
SVM RBF	30.34	
Random Forest (Trees Number = 15)	391.84	
Random Forest (Trees Number = 10)	247.07	
Multi-Layer Network (Hidden Layers=5)	455.63	
Multi-Layer Network (Hidden Layers=3)	308.52	
Multi-Layer Network (Hidden Layers=1)	135.16	
SVM LINEAR	0.17	
Bayesian Network	3.6	
RBF Network	3.6	
Naïve Bayes	0.87	
Classification via Clustering	2.56	

BORDERLINE SMOTE

Ο SVM RBF classifier για το συγκεκριμένο αλγόριθμο, εκτελέστηκε με παραμέτρους cost = 1.0 και gamma = 2.1.

OVERALL CLASSIFICATION ACCURACY		
Classifier	**Corect Classification Rate**	**Incorrect Classification Rate**
SVM RBF	93,9249	6,0751
Random Forest (Trees Number = 15)	92,258	7,742
Random Forest (Trees Number = 10)	92,1145	7,8855
Multi-Layer Network (Hidden Layers=5)	58,1264	41,8736
Multi-Layer Network (Hidden Layers=3)	53,3401	46,6599
Multi-Layer Network (Hidden Layers=1)	54,3933	45,6067
SVM LINEAR	90,8126	9,1874
Bayesian Network	88,5571	11,4429
RBF Network	85,7513	14,2487
Naïve Bayes	76,9962	23,0038
Classification via Clustering	58,5665	41,3471

Μελετώντας τα αποτελέσματα από τον Borderline Smote παρατηρούμε ότι η καλύτερη απόδοση είναι στον αλγόριθμο SVM RBF classifier με ποσοστό επιτυχίας 93,9%. Η δεύτερη καλύτερη απόδοση είναι στον Random Forest (Trees Number=15) με 92,2% και ακολουθεί ο Random Forest (Trees Number=10) classifier με ποσοστό επιτυχίας 92,1%. Την τέταρτη και πέμπτη καλύτερη απόδοση, την έχουν ο SVM Linear και ο Bayesian Network classifier με ποσοστά επιτυχίας 90,8% και 88,6% αντίστοιχα.

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.951	0.924
Random Forest (Trees Number = 15)	0.920	0.894
Random Forest (Trees Number = 10)	0.928	0.881
Multi-Layer Network (Hidden Layers=5)	0.624	0.377
Multi-Layer Network (Hidden Layers=3)	0.556	0.260
Multi-Layer Network (Hidden Layers=1)	0.573	0.289
SVM LINEAR	0.919	0.891
Bayesian Network	0.905	0.858
RBF Network	0.819	0.913

Naïve Bayes	0.715	0.886
Classification via Clustering	0.710	0.271

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**PRECISION**	**RECALL**
SVM RBF	0.929	0.954
Random Forest (Trees Number = 15)	0.924	0.944
Random Forest (Trees Number = 10)	0.917	0.950
Multi-Layer Network (Hidden Layers=5)	0.563	0.779
Multi-Layer Network (Hidden Layers=3)	0.527	0.799
Multi-Layer Network (Hidden Layers=1)	0.534	0.791
SVM LINEAR	0.898	0.924
Bayesian Network	0.869	0.913
RBF Network	0.905	0.804
Naïve Bayes	0.856	0.657
Classification via Clustering	0.558	0.892

POSITIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.937	0.939
Random Forest (Trees Number = 15)	0.907	
Random Forest (Trees Number = 10)	0.904	
Multi-Layer Network (Hidden Layers=5)	0.470	0.667
Multi-Layer Network (Hidden Layers=3)	0.354	0.571
Multi-Layer Network (Hidden Layers=1)	0.384	0.581
SVM LINEAR	0.905	0.908
Bayesian Network	0.881	0.959
RBF Network	0.863	0.892
Naïve Bayes	0.791	0.786
Classification via Clustering	0.392	0.582

NEGATIVE CLASS CLASSIFICATION METRICS		
Classifier	**F-MEASURE**	**ROC**
SVM RBF	0.941	0.939
Random Forest (Trees Number = 15)	0.934	
Random Forest (Trees Number = 10)	0.933	
Multi-Layer Network (Hidden Layers=5)	0.654	0.667
Multi-Layer Network (Hidden Layers=3)	0.635	0.571
Multi-Layer Network (Hidden Layers=1)	0.638	0.581
SVM LINEAR	0.911	0.908
Bayesian Network	0.890	0.959
RBF Network	0.851	0.892
Naïve Bayes	0.744	0.896
Classification via Clustering	0.686	0.581

TIME COMPLEXITY METRICS		
Classifier	**TIME TO BUILD THE MODEL (secs)**	
SVM RBF	32.04	
Random Forest (Trees Number = 15)	1133.5	
Random Forest (Trees Number = 10)	700.41	
Multi-Layer Network (Hidden Layers=5)	423.81	
Multi-Layer Network (Hidden Layers=3)	275.07	
Multi-Layer Network (Hidden Layers=1)	115.91	
SVM LINEAR	0.31	
Bayesian Network	3.09	
RBF Network	6.15	
Naïve Bayes	0.81	
Classification via Clustering	1.75	

5. Conclusions and future works

Σε όλη αυτή την εργασία, με την ανάπτυξη των αλγορίθμων και τον υπολογισμό της απόδοσής τους, έγινε προσπάθεια να επιτευχθεί το sentiment analysis και στη συνέχεια το sentiment classification. Χρησιμοποιήθηκαν λοιπόν διάφοροι αλγόριθμοι, ώστε να επιτευχθεί αυτός ο στόχος, για τους οποίους έγινε και η μέτρηση της απόδοσής τους χρησιμοποιώντας διάφορους ταξινομητές.

Από τα αποτελέσματα που προέκυψαν μπορούμε να δούμε την αρκετά καλή απόδοση του SVM αλλά και του Random Forest στο θέμα της διαχείρισης της sentiment analysis και sentiment classification. Υπήρχαν και άλλοι classifiers με καλή απόδοση αλλά ο SVM αποδείχτηκε καλύτερος όσο αφορά τη διαχείριση δύο στοιχείων συναισθημάτων, όπως στην περίπτωσή μας τα θετικά και τα αρνητικά σχόλια.

Η μελλοντική έρευνα θα μπορούσε να επικεντρωθεί σε άλλες μεθόδους μηχανικής μάθησης, όπως η βιολογική ταξινόμηση των Artificial Immune Systems (AIS). Αυτή η μέθοδος έχει αποδειχτεί ότι είναι πιο αποτελεσματική στην ταξινόμηση σε τέτοιες περιπτώσεις ανισορροπίας μεταξύ των δεδομένων. Η μέθοδος αυτή βασίζεται στις αρχές και τις διαδικασίες του σπονδυλωτού ανοσοποιητικού συστήματος. Το πεδίο των Artificial Immune Systems (AIS), ασχολείται με τη δομή και τη λειτουργία του συστήματος στο υπολογιστικό σύστημα και τη διεύρυνση εφαρμογών για την επίλυση προβλημάτων μαθηματικών, μηχανικής και τεχνολογίας πληροφοριών. Το AIS είναι μια πολύ καλή περίπτωση ώστε να επεκταθεί η έρευνα της ταξινόμησης. Αυτό διότι βλέπουμε μια τάση των χρηστών να οδεύει προς την αρνητική τάξη των συναισθημάτων – γνωμών – επιλογών. Ένας ακόμα λόγος που έχει αξία για τη χρήση και κατ’ επέκταση τη μελέτη αυτών των συστημάτων είναι ότι έχουμε αρκετά αραιά στο χώρο σύνολα. Θα πρέπει λοιπόν να γίνει σωστή δειγματοληψία και ταξινόμηση των δειγμάτων και η χρήση των AIS θα μπορεί να βοηθήσει σημαντικά.

6. Βιβλιογραφία

Dionisios N. Sotiropoulos, George A. Tsihrintzis (2016, June). Machine learning paradigms. Artificial Immune Systems and their applications in software personalization.

Zan Huang, Hsinchun Chen, Daniel Zeng. Applying Associative Retrieval Techniques to Alleviate the Sparsity Problem in Collaborative Filtering

Batul J. Mirza, Benjamin J. Keller, Naren Ramakrishnan. Studying Recommendation Algorithms by Graph Analysis

Gediminas Adomavicius, Alexander Tuzhilin. Toward thw Next Generation of Recommender Systems: A Survey of the State-of-the-Art and possible extensions.

Dionisios N. Sotiropoulos, Chris D. Kounavis, Panos Kourothanassis, George M. Giaglis. What drives social sentiment? An entropic measure-based clustering approach towards identifying factors that influence social sentiment polarity.

Beau Piccart, Jan Struyf, Hendrik Blockeel. Alleviating the Sparsity Problem in Collaborative Filtering by using an Adapted Distance and a Graph-based Method.

Charu C. Aggarwal, Joel L. Wolf, Kun-Lung Wu, Philip S. Yu. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering.

Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kontor. Recommender Systems Handbook.

Lester Mackay (2009, October). Collaborative Filtering. Practical Machine Learning.

Jonathan L. Herlocker, Joseph A. Koustan, Loren G. Terveen, John T. Riedl. Evaluating Collaborative Filtering Recommender Systems.

Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, Irwin King. Recommender Systems with Social Regularizarion

David Z Liu, Gurbir Singh. Item Graph Based Recommendation System for Bookopolis.

Benjamin Marlin: Collaborative Filtering : A machine learning perspective.

https://www.mathworks.com/