ΕΘΝΙΚΟ ΚΑΙ
ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ

# Τεχνικές εξόρυξης Δεδομένων

## Άσκηση 1 - Read me

**Δημιουργία WordCloud**
Σε αυτό το ερώτημα δημιουργήθηκαν 5 wordCloud**,** 1 για κάθε κατηγορία. Πριν δημιουργηθούν τα αρχεία αυτά αφαιρέθηκαν από τα άρθρα οι ειδικές λέξεις stopWords=’english’ και επιπλέον κάποιες άλλες που παρατηρήθηκαν ανούσιες σύμφωνα με το ερώτημα. Για την εκτέλεση αυτή χρησιμοποιήθηκε η βιβλιοθήκη wordcloud+STOPWORDS

**Υλοποίηση Συσταδοποίησης (Clustering)**
Σε αυτό το ερώτημα υλοποιήθηκε ο αλγόριθμος k-means έτσι ώστε να ομαδοποιήσουμε τα δεδομένα μας , δηλαδή τα άρθρα σε 5 ομάδες και στην συνέχεια το αποτέλεσμα αυτό να αξιολογηθεί από το αρχείο εξόδου clustering_Kmeans.csv, το οποίο είναι και ζητούμενο της εργασίας. Αρχικά έγινε προεπεξεργασία των δεδομένων, δηλαδή τα άρθρα μας μετράπηκαν σε διάνυσματα όπου το κάθε διάνυσμα είναι ένα λεξικό και σαν στοιχεία έχει τον αριθμό των φορών που εμφανίζεται η λέξη στο κείμενο(CounterVectorizer). Στην συνέχεια τροποποιησαμε τα στοιχεία για να βάλουμε και μια επιπλέον παράμετρο , οπού είναι η βαρύτητα που έχει κάθε λέξη στο κείμενο(TFIDF-Vectorizer).
Το επόμενο βήμα ήταν να μειωθουν οι διαστάσεις του διανύσματος μέσω της LSI τεχνικής , όπου χρησιμοποιήθηκε η έτοιμη βιβλιοθήκη TruncadedSVD με επιλεγμένο αριθμό διαστάσεων 20 (δοκιμάστηκαν και άλλες τιμές της παραμέτρου αυτής). Και σε αυτήν την χρονική στιγμή είμαστε έτοιμοι να εκτελέσουμε τον k-means me k=5. Να τονίσουμε πως η υλοποίηση του αλγορίθμου έγινε από έτοιμη βιβλιοθήκη , την nltk.cluster με την KmeansClusterer συνάρτηση. Η συνάρτηση αυτή παρείχει την δυνατότητα να χρησιμοποιηθεί η cosine similarity μέθοδος μετρικής. Τέλος υλοποιήθηκε και ένα βήμα που αφορά την αξιολόγηση του αλγορίθμου όπου στο τέλος δημιουργήται το ζητούμενο αρχείο clustering_Kmeans.csv.

**Υλοποίηση Κατηγοριοποίησης (Classification)**
Στο ερώτημα υλοποιήθηκαν οι 4 ζητούμενοι αλγόριθμοι με σκοπό την κατηγοριοποιήση των δεδομένων. Οι 3 από τους 4 αλγορίθμους υλοποιήθηκαν από έτοιμες βιβλιοθήκες της python εκτός από τον KNN. Πριν την εκτέλεση των αλγορίθμων προηγήθηκε το βήμα του preprocessing. Για όλους τους αλγορίθμους χρησιμοποιήκε η ίδια μεθοδολογία. Η σειρά του preprocessing είναι πρώτα να φτιαχτεί ένα διάνυσμα για κάθε κείμενο με τον αριθμό των εμφανίσεων των λέξεων στο κείμενο(CounterVectorizer). Στην συνέχεια τροποποιησαμε τα στοιχεία για να βάλουμε και μια

επιπλέον παράμετρο , οπού είναι η βαρύτητα που έχει κάθε λέξη στο κείμενο(TFIDF-Vectorizer).
Το επόμενο βήμα ήταν να μειωθουν οι διαστάσεις του διανύσματος μέσω της LSI τεχνικής , όπου χρησιμοποιήθηκε η έτοιμη βιβλιοθήκη TruncadedSVD. Για κάθε ένα από τα 3 αυτά βήματα έγιναν αρκετά τεστ στις παραμέτρους αυτών των συναρτήσεων με σκοπό το καλύτερο αποτέλεσμα. Για παράδειγμα αφαιρέθηκαν οι ανούσιες λέξεις απο την stopword βιβλιοθήκη, χρησιμοποιήθηκαν οι μεταβλητές min_df kai max_df για να αποκλείσουμε τις πολύ συχνες λέξεις ή τις πολύ σπάνιες, δηλαδή λέξεις που δεν μας δίνουν κάποια ιδιαίτερη πληροφορία για την ανάλυση των άρθρων.
Επίσης δώθηκε περισσότερο έμφαση στην ανάλυση και τον πειραματεισμό με τις παραμέτρους των αλγορίθμων με σκοπό την καλύτερη κατανόηση τους. Στο επόμενο ερώτημα , δηλαδή στην υλοποίηση του δικού μας classifier δώθηκε περισσότερη έμφαση στο preprocessing (χρησιμοποιήθηκε και η τίτλος των αρθρων)με σκοπό την επίτευξη του καλύτερου αποτελέσματος.
1)Naive bayes
Στον αλγόριθμο αυτόν παραλείφθηκε το βήμα του dimensionality reduction αφού όπως ξέρουμε ο αλγόριθμος αυτός δεν βασίζεται στον υπολογισμό αποστάσεων και γενικά δεν χρειάζεται να ορίσει τα δεδομένα μας σε ένα χώρο d διαστάσεων αλλά το μόνο που μετράει είναι πιθανότητες. Δοκιμάστηκε ο Multinomial και Bernulli.
2)Random Forest
Ουσιαστικά εδώ χρησιμοποιούμε πολλά δέντρα αποφάσεων μαζεμένα. Ο αριθμός αυτός δοκιμάστηκε με 10,20,50,100. Το συμπέρασμα από αυτές τις δοκιμές ήταν ότι το αποτέλεσμα δεν είχε κάποια μεγάλη αλλοίωση.
Για παράδειγμα από 10 σε 50 είχε καλύτερο αποτέλεσμα αλλά από 50 σε 100 το αποτέλεσμα παρέμενε ίδιο.
3)Support Vector Machine
Στον αλγόριθμο αυτόν αν και παρατηρήθηκαν αρκετά καλά αποτελέσματα μπορούμε να παρατηρήσουμε ότι υστερούσε σε σχέση με τους άλλους με βάση τον χρόνο. Τα πειράματα που έγιναν με σε αυτήν την περίπτωση αφορούσαν κυριως της παραμέτρους kernel,C και gamma οι οποίες ζητήθηκαν και στην εκφώνηση της άσκησης. Για την πιο αποτελεσματική ερεύνα πάνω σε αυτές τις παραμέτρους έγινε προσπάθεια να χρησιμοποιηθεί η GridSearch βιβλιοθήκη με ορια για C [1,10] και gamma απο [1,10] παρατηρήθηκε οτι αναλογα με το ιδος του optimize για accuracy recall και των άλλων μετρικών η αριθμοι άλλαζαν αλλα η καλύτερη συναρτηση για similarity παρεμεινε ο rbf kernel
Γενικά παρατηρήθηκε στο kernel ότι η rbf είχε καλύτερη απόδοση σε σχέση με την linear.
4)KNN
Η υλοποίηση του αλγορίθμου αυτού έγινε χωρίς την βοήθεια της python. Αρχικά ορίσαμε το K , τον αριθμό δηλαδή των γειτόνων. Ο αλγόριθμος μετά υλοποίησε τα 3 συναρτήσεις. Την cosine similarity getNeighbors η αποια υπολλογίζει τις ομοιότητες μεταξυ των σημείων και κραταεί τις ποιο μεγάλες , δεδομένο από τα testData υπολογίσθηκε η απόσταση από κάθε trainData και τα αποτελέσματα αυτά τα κρατήσαμε σε μία ταξινομημένη λίστα και από αυτήν την λίστα κρατήσαμε τις K πιο μικρές αποστάσεις. Η μετρική απόστασης που χρησιμοποιείται είναι η cosine similarity.Η τελευταία συναρτηση getResponce είναι το βήμα με το οποίο γίνεται η πρόβλεψη. Με την μέθοδο Majority voting κάνουμε την επιλογή της κατηγορίας, δηλαδη μετράμε μέσα σε αυτους τους K γείτονες ποιές κατηγορίες εμφανίζονται περισσότερες φορές, και αυτήν προβλέπουμε ότι ανήκει το άρθρο η . Για τον υπολογισμο

Τέλος θα αναφέρουμε και τι έχουμε κάνει για την αξιολόγηση των αλγορίθμων. Για κάθε αλγόριθμο υπολογίζουμε accuracy/precision/recall/f1-score μέσα από 10-cross-validation και για το τελικό αποτέλεσμα που θα γραψουμε και στο ζητούμενο αρχείο EvaluationMetric_10fold.csv θα κρατήσουμε τον μέσο όρο των τελικών αποτελεσμάτων.

**Beat the Benchmark (bonus)**

Στο ερώτημα αυτό χρησιμοποιήθηκε ένας συνδυασμος αλγορίθμων , βασικά 2, ενός πιθανοτικού (naive bayes) και ενός machine learning algorithm του SVM. Αρχικα στο βήμα της προεπεξεργασίας δοκιμάστηκε η μέθοδος stemming η οποία μπορεί και κόβει κάποιες λέξεις ίδιας σημασίας , π.χ playing ,play και της βάζει σαν μία λέξη , αυτό βέβαια έχει και τον κίνδυνο όταν οι λέξεις δεν είναι ίδιας σημασίας και λόγω αρκετής χρονικης καθηστερησης στην εκτέλεση του προγράμματος η διαδικασία αυτή απορρίφθηκε. Γενικά μία παρατήρηση που πρέπει να γίνει είναι ότι ενώ δοκιμάστηκαν αρκετά πειράματα στην διαδικασία της προεπεξεργασίας η διαφορά στην απόδοδση ήταν μηδαμινή και ο λόγος οφείλεται κυρίως ότι οι αλγόριθμοι από μονοι τους χωρίς κάποια προεπεξερασία, είχαν αρκετό μεγάλο ποσοστό ευστοχίας. Επίσης χρησιμοποιήθηκε και ο τίτλος των άρθρων για την απόφαση της κατηγοριοποίησης . Μετά την εκτέλεση των αλγορίθμων στα προηγούμενα βήματα παρατηρήθηκε ότι ο random forest είχε την καλήτερη απόδοση και ο λόγος που γίνεται αυτο θεωρείται πως είναι γιατί ενώ την πρώτη φορά έχει περίπου ευστοχία στο 96% στα επόμενα 9 βήματα (10-cross-validation) αγγίζει το 100% γιατί μάλλον ξαναφτιάχνει καινούργιο μοντέλο όπου διορθώνει το παλιό μοντέλο του, αυτό όμως μπορεί να εμπεριέχει τον κίνδυνο για overfitting αν δοκιμάσουμε το μοντέλο σε καινούργια data. Επομένος φυσικό και επόμενο είναι αυτή απόδοση να μην μπορείς να την ξεπεράσεις αλλά ούτε υπάρχει λόγος να την χρησιμοποιήσεις στον δικό σου αλγόριθμο. Η βασική ιδέα του αλγόριθμου αυτού είναι η εξής. Εκτελούμε τον Naive bayes αλγόριθμο και τον εκπεδεύουμε και με βάση το content και με βάση το τίτλο. Το ίδιο κάνουμε και με το SVM. Οπότε στην ουσία έχουμε 4 predictions και το τελικό prediction βγαίνει με βάση κάποιων περιπτώσεων. Αρχική περίπτωση είναι ότι αν οι predictors από τα άρθρα έχουν προβλέψει την ίδια κατηγορία τότε αυτή είναι και η τελική πρόβλεψη για το συγκεκριμένο άρθρο, εναλακτίκα λαμβάνουμε υπόψην και τους predictors του τίτλου. Αυτό είχε σαν αποτέλεσμα να αυξήσουμε την απόδοση των αλγορίθμων αυτών. Για παράδειγμα ο NB είχε περίπου 94% ευστοχία και ο SVM 95% και τώρα έφτασε στο 96%. Ο κώδικας παράγει σαν output το testSet_categories.csv που είναι ζητούμενος της εργσιάς με το αυστηρό format που ζητήται.

**RocCurve**

Οι γραφικές παραστάσεις των ... βρίσκονται στον ομώνυμο φάκελο Οι γραφικές παραστάσεις παρουσιάζουν την micro average macro average roc curves δηλαδή για την πρώτη πήραμε όλα τα σημεία για κάθε μια από τις 5 κατηγορίες τα αθροίσαμε και διαιρέσαμε διά 5 , ενώ στην την δεύτερη πήραμε όλα τα σημεία από όλες τις κατηγορίες τα βάλαμε σε μια λίστα και κάναμε γραμμική παρεμβολή στα σημεία αυτά . Λίγα λόγια για την Roc curve είναι μια συνάρτηση η οποία έχει άξονα χ το false positive rate και άξονα y το true positive rate. Η συνάρτηση αυτή βγαίνει από την κατανομή τον σημείων σε ένα χώρο οπού έχει την πιθανότητα ενός σημείου να είναι false positive και true positive ορίζοντας ένα threshold κατηγοριοποιούμε το ποια σημεία είναι false positive και true positive η επιλογή πολλών τέτοιων thresholds μας δίνει σημεία της roc curve .

**Εκτέλεση**

Για την εκτέλεση όλων των αρχείων κώδικα έχει φτιαχτεί μία main στον φάκελο src/default με σκοπό να εκτελέσει και να παράγει όλα τα ζητούμενα αρχεία της άσκησης. Τα αρχεία αυτά δημιουργούνται στο φάκελο output. Πρώτα εκτελείτε ο myClassifier(MINE.py) για να πράγει το αρχείο testSet_categories.csv , μετά ο k-means που παράγει το αρχείο clustering_Kmeans.csv και τέλος οι classifiers που παράγουν το αρχείο EvaluationMetric_10fold.csv .

**Αποτελέσματα**

## Clustering

| | Business | Films | Football | Politics | Technology |
|---|---|---|---|---|---|
| **Cluster 1** | **0.05228519196** | **0.00848593122** | **0.00096123037** | **0.1118151323** | **0.91190316073** |
| **Cluster 2** | **0.00292504570** | **0.00223313979** | **0.97212431913** | **0.00037271711** | **0.01950235373** |
| **Cluster 3** | **0.00438756855** | **0.98302813756** | **0.00384492150** | **0.01006336191** | **0.02891728312** |
| **Cluster 4** | **0.92285191956** | **0.00446627959** | **0.01313681512** | **0.10510622437** | **0.03160726294** |
| **Cluster 5** | **0.01755027422** | **0.00178651183** | **0.00993271387** | **0.87327618337** | **0.00806993947** |

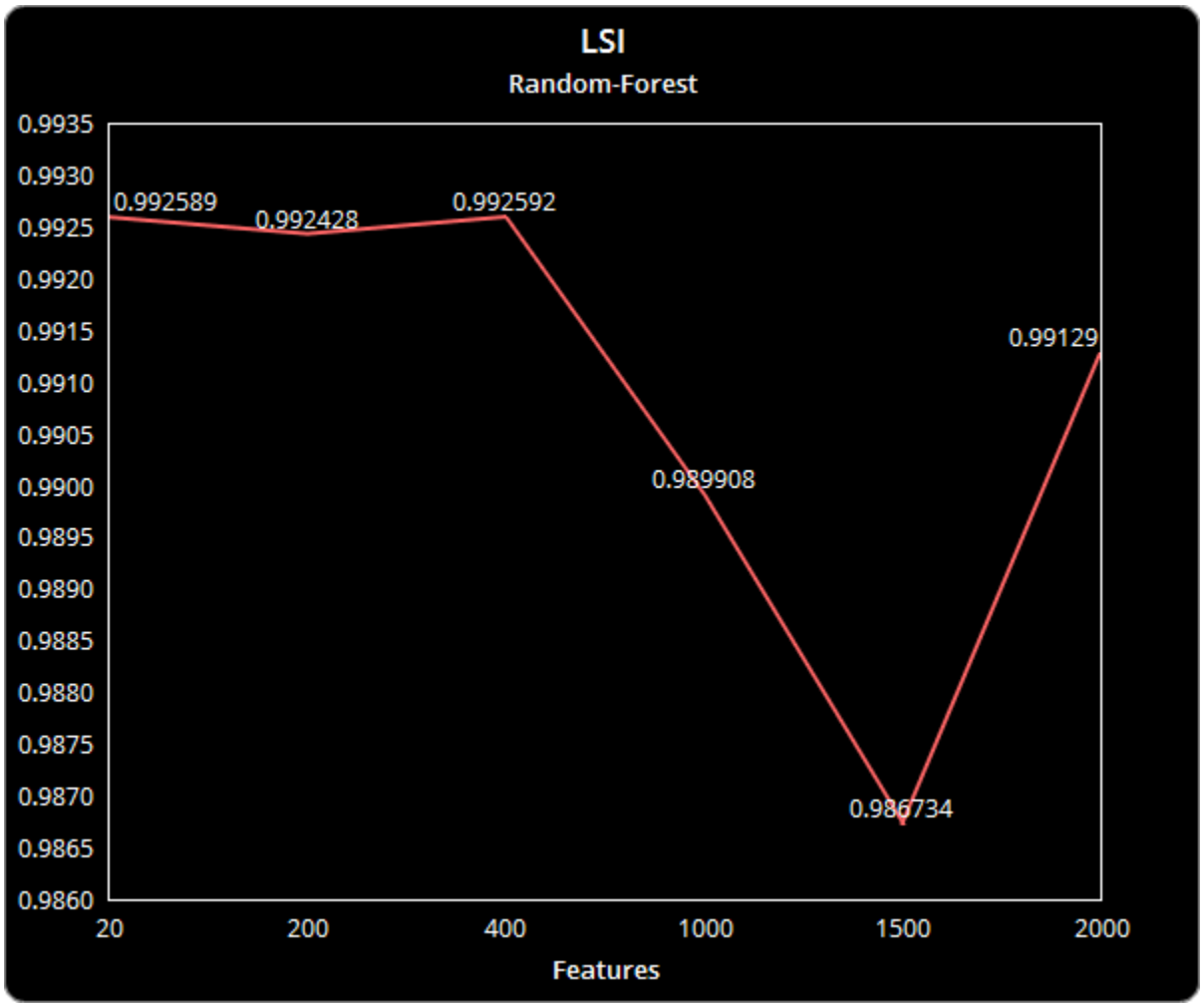
## Classification

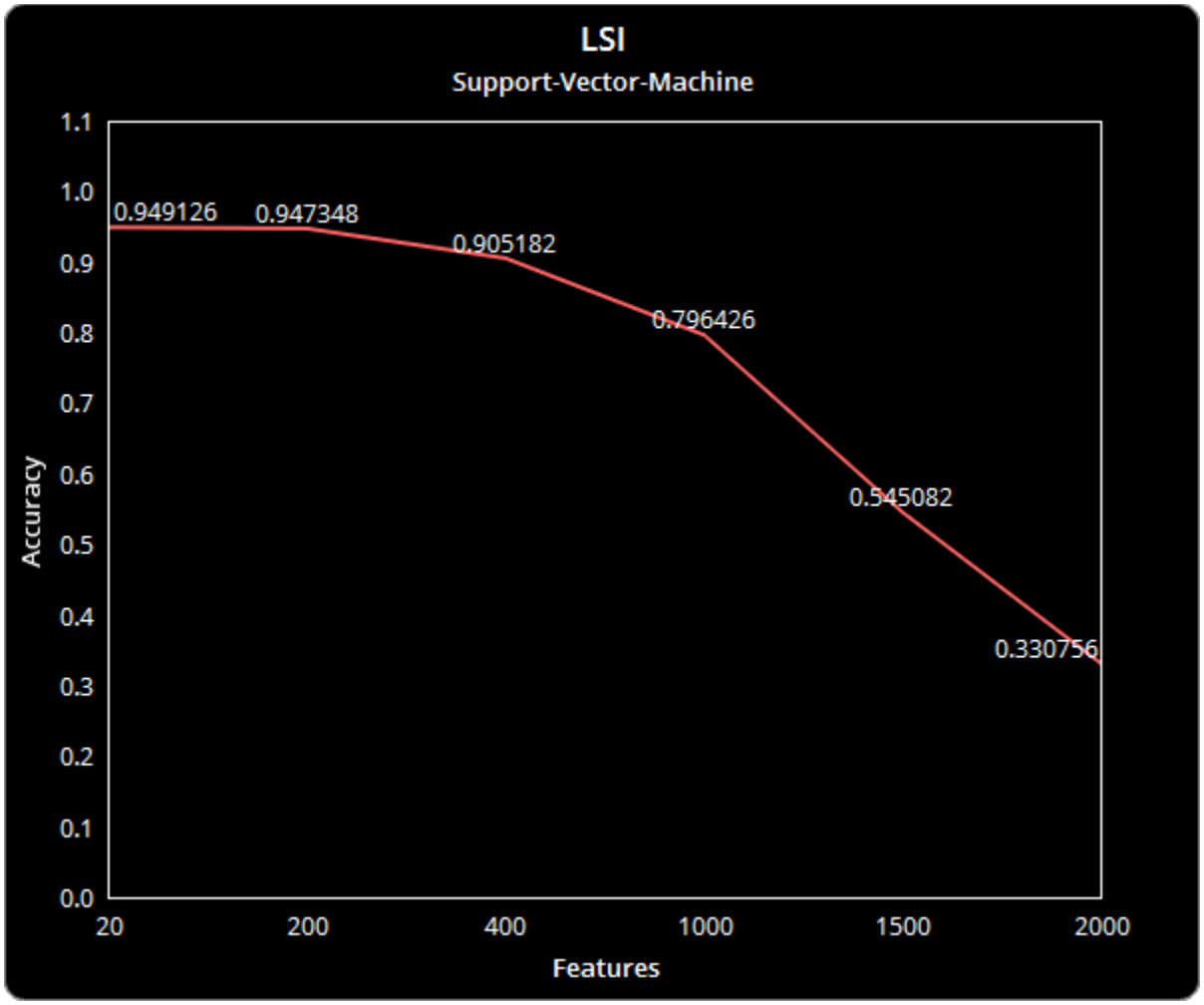
| Statistic Measure | Naive Bayes | Random Forest | SVM | KNN |
|---|---|---|---|---|
| **Accuracy** | **0.9419517586466** | **0.9960943856794142** | **0.943011519035859** | **0.9289977159908** |
| **Precision** | **0.9382983688874** | **0.9959454676765109** | **0.946578655004526** | **0.9319541447776** |
| **Recall** | **0.9368799520843** | **0.9958261635867451** | **0.930305642686298** | **0.9171042409198** |
| **F-Measure** | **0.9373542768873** | **0.995871101341957** | **0.936587875978114** | **0.9219736950026** |
| **AUC (Macro)** | **0.9949578234833** | **0.9991157578274151** | **0.996011781446017** | **0.9927912767639** |

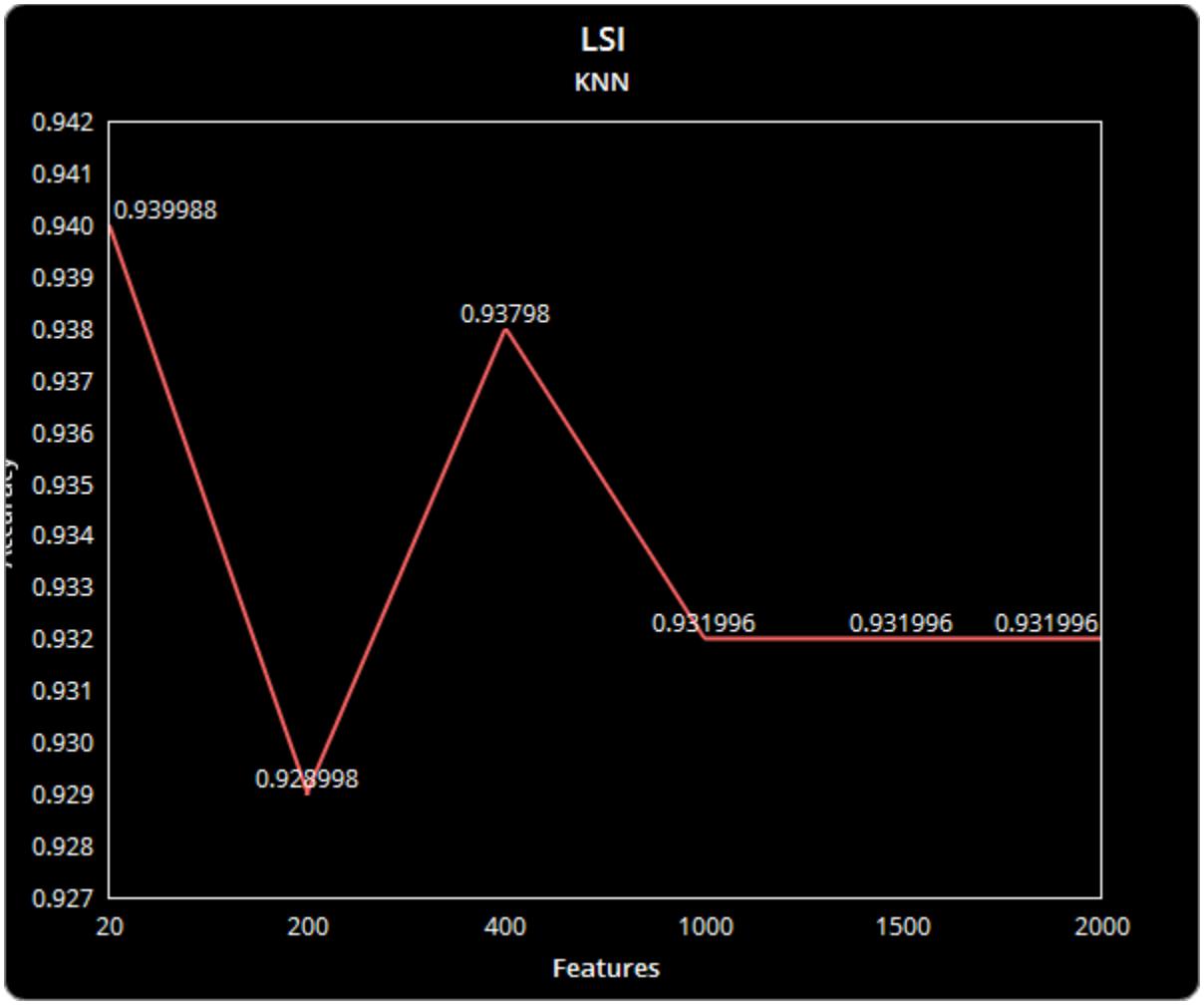
**Το αρχείο test set categories βρησκεται στον φάκελο outputFiles μαζί με τα άλλα αρχεία**

**LSI**

Εδώ παρουσιάζονται τα σχεδιαγράμματα για διαφορετικό αριθμό components σε σχέσει με το accuracy




LSI
Support-Vector-Machine
Accuracy
Features
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
20
200
400
1000
1500
2000
0.949126
0.947348
0.905182
0.796426
0.545082
0.330756




Συμπέρασμα παρατηρούμε ότι και ο Svm είναι αρκετά επιρρεπείς στον αριθμό των παραμέτρων των διανυσμάτων αυτό μπορεί να βελτιωθεί με την σωστή επιλογή του kernel function και των παραμέτρων C και Gamma . Χρησιμοποιώντας το kernel Trick μπορούμε να αυξήσουμε τις διαστάσεις για να προβάλουμε τα διανύσματα στον αυξημένο χώρου μας έτσι ώστε να μπορέσουμε να τα χωρίσουμε καλύτερα . Κανονικά θα έπρεπε να έχουμε μεγάλη αντοχή όσον άβαρα των Svm και τον αριθμό των χαρακτηριστικών των διανυσμάτων αλλά πειραματικά βλέπουμε ότι αυτό δεν ισχύει . Η σωστή επιλογή των παραμέτρων παραμένει θέμα μελέτης.

**Developers**

**AM:1115200800182**
**Βασίλης Χρόνης**

**AM:1115201100037**
**Φωκίων Ζηκίδης**