Πρότζεκτ
Θεωρίας Αποφάσεων

Ακίνο-Ελλούλ Παύλος 5ο έτος
1059630
Αλ-Μπαμπούλ Γεώργιος 5ο έτος
1059631

Φεβρουάριος 2022

Το παρόν έγγραφο αποτελεί την αναφορά της εργασίας των φοιτητών Ακίνο-Ελλούλ Παύλου και Αλ-Μπαμπούλ Γεώργιου για το μάθημα Θεωρία Αποφάσεων του Τμήματος Μηχανικών Η/Υ και Πληροφορικής.

Contents

1 Εισαγωγή

Το πρότζεκτ που διαλέξαμε έχει τίτλο "Spam Or Ham". Συνοπτικά, σκοπός του πρότζεκτ είναι η υλοποίηση ενός Chat app με την ιδιότητα ότι θα αναγνωρίζει ανεπιθύμητα μηνύματα με την χρήση τεχνικών μηχανικής μάθησης. Οι τεχνικές που χρησιμοποιήσαμε ήταν το Logistic Regression, Random Forest, SVM, KNN και ένα νευρωνικό δίκτυο. Αρχικά αναλύουμε τον τρόπο με τον οποίο μετασχηματίσαμε τα δεδομένα ώστε να καθιστούν κατάλληλη είσοδος για τα μοντέλα που χρησιμοποιήσαμε αλλά και για την βελτιστοποίηση της ακρίβειας. Στην συνέχεια αναλύουμε την κάθε τεχνική και στο τέλος δίνουμε διάφορα αποδεκτά μετρικά για την κάθε μία. Τέλος, παρουσιάζουμε την εμφάνιση και λειτουργεία της εφαρμογής.

2 Pre-processing

2.1 CSV

Το dataset σε μορφή csv αρχικά είχε 5 στήλες περιπου, με την πρώτη να είναι το flag (ham or spam), και τις υπόλοιπες να είναι τα emails. Πριν κάνουμε οποιοδήποτε pre-processing με την Python, ενώσαμε τις στήλες με τα emails σε μία στήλη και αλλάξαμε το encoding σε utf-8.

2.2 Python Libraries

2.2.1 Doc2Vec

Η προεπεξεργασία αποτελείται απο 4 στάδια. Το tokenization των emails, την μετατροπή των tokens(Λέξεων) σε μικρά γράμματα, την διαγραφή των αριθμών, και την διαγραγή των αγγλικών stop words και των σημείων στίξης.Τέλος προσθέσαμε και stemming, στο preprocessing, αλλα παρατηρήσαμε πως δεν είχαμε πάντα βελτίωση των αποτελεσμάτων. Όλα αυτά γίνανε με την βοήθεια της βιβλιοθήκης nltk.

Αφού φέραμε τα emails σε μια αποδεκ΄τη μορφή. Τα περάσαμε στο Doc2Vec model, το οποίο μετέτρεψε τα tokenized emails σε vectors. Αρχικά το εκπαιδεύσαμε πάνω σε δικά μας δεδομένα, και ύστερα το χρησιμοποιούσαμε για να κάνουμε εκπαιδεύσουμε τα μοντέλα μας αλλά και για να μετατρέψουμε τα μηνύματα των χρηστών σε κατάλληλη μορφή για πρόβλεψη των spam or ham. Το Doc2Vec χρησιμοποιεί τις παρακάτω μεθόδους.

- Paragraph Vector - Distributed Memory (PV-DM)
- Paragraph Vector - Distributed Bag of Words (PV-DBOW)

2.2.2 Count Vectorizer

Μια άλλη μέθοδο embedding που δοκιμάσαμε ήταν η CountVectorizer της Sklearn η οποία χρησιμοποιεί Bag of Words. Παρατηρήσαμε πως έβγαζε καλύτερα accuracies στα μοντέλα μας.

3 Logistic Regression

3.1 Γενικά

Η λογιστική παλινδρόμηση είναι ένα γενικό στατιστικό μοντέλο, το οποίο αξιοποιεί μία λογιστική συνάρτηση για να υπολογίσει την πιθανότητα εμφάνισης μίας τιμής μίας εξαρτημένης τυχαίας μεταβλητής με την χρήση άλλων στατιστικά ανεξάρτητων τυχαίων μεταβλητών. Από μόνη της, δεν μπορεί να χρησιμοποιηθεί ως classifier αλλά μπορούμε να κατηγοριοποιήσουμε με βάση αν η πιθανότητα είναι μεγαλύτερη η μικρότερη από ένα cutoff value που ορίζουμε εμείς. Ως classifier είναι ένας από τους απλούστερους αλγόριθμους ML που μπορεί να χρησιμοποιηθεί για διάφορα προβλήματα ταξινόμησης όπως ανίχνευση πρόβλεψη διαβήτη, ανίχνευση καρκίνου. Χωρίζεται σε τρείς υποκατηγορίες, την binary, multinomial και την ordinal. Στην binary κατηγορία η εξαρτημένη τυχαία μεταβλητή είναι δυαδική και παίρνει τιμές όπως "0" ή "1", "pass" ή "fail", "ναι" ή "όχι" ή στην δικιά μας περίπτωση "spam" ή "ham". Στην multinomial κατηγορία, η εξαρτημένη ΤΜ μπορεί να έχει τρείς η περισσότερους μη ταξινομημένους τύπους όπως "Τύπος Α", "Τύπος Β", "Τύπος Γ". Τέλος, στην ordinal, η εξαρτημένη ΤΜ μπορεί να πάρει τρείς η περισσότερες ταξινομημένες τιμές όπως "κακό", "μέτριο", "καλό", "πολύ καλό".

3.2 Υλοποίηση

Στην δικιά μας υλοποίηση χρησιμοποιήσαμε την πρώτη (binary) κατηγορία με την χρήση των συναρτήσεων της βιβλιοθήκης sklearn για όλη την διαδικάσία. Αρχικά χωρίσαμε τα δεδομένα με την συνάρτηση train_test_split που χωρίζει τα δεδομένα σε ξεχωριστές λίστες τις Python για να χρησιμοποιηθούν αντίστοιχα (χρησιμοποιήσαμε το 70% των δεδομένων για την εκπαίδευση του μοντέλου και του 30% για testing). Έπειτα δημιουργήσαμε ένα instance της κλάσης LogisticRegression() και χρησιμοποιώντας την .fit() συνάρτηση-ιδιότητά της κάναμε fit τα train δεδομένα. Τέλος, χρησιμοποιώντας την .predict() κάναμε τα test. Για να μετρήσουμε την απόδοση χρησιμοποιήσαμε μετρικά τα οποία μας δίνονται από την βιβλιοθήκη όπως το f1 score. Τα αποτελέσματα μπορείτε να τα βρείτε στο κεφάλαιο 8

4 Random Forest

4.1 Γενικά

Το Random Forest ή αλλιώς το Random Decision Forests είναι μια μέθοδος ensemble εκμάθησης για επίτευξη στόχων όπως η ταξινόμηση και η παλινδρόμηση και λειτουργεί με την κατασκευή πολλών δέντρων αποφάσεων(Decision Trees) κατά το χρόνο εκπαίδευσης. Ως classifier, το classification γίνεται με την επιλογή της κλάσης στην οποία συμφωνεί η πλειοψηφία των δέντρων αποφάσεων. Για regression επιστρέφεται ο μέσος όρος των τιμών στις οποίες καταλήγουν τα δέντρα αποφάσεων.

4.2 Υλοποίηση

Για την δικιά μας υλοποίηση χρησιμοποιήσαμε πάλι την βιβλιοθήκη sklearn και συγκεκριμένα τις συναρτήσεις train_test_split και την κλάση RandomForestClassifier στης οποίας τον constructor δώσαμε δύο ορίσματα, τον αριθμό των δέντρων (n_estimators) και ένα random_state για να γίνει διαφορετική αρχικοποιήση κάθε φορά (κυρίως για εύρεση βέλτισου αριθμού δενδρων). Δεν δώθηκαν βάρη σε καμία κλάση καθώς στην δικιά μας περίπτωση το classification είναι binary. Στην συνέχεια, παρόμοια με πριν, χρησιμοποιούμε τις εσωτερικές μεθόδους fit() και predict() για την εκπαίδευση και το testing αντίστοιχα. Τέλος βγάζουμε διάφορα μετρικά τα οποία μπορείτε να δείτε στο κεφάλαιο 8

5 Support Vector Machine

5.1 Γενικά

Το "Support Vector Machine" (SVM) είναι ένας εποπτευόμενος αλγόριθμος μηχανικής μάθησης που μπορεί να χρησιμοποιηθεί τόσο για classification όσο και για regression. Ωστόσο, χρησιμοποιείται κυρίως σε προβλήματα classification. Στον αλγόριθμο SVM, σχεδιάζουμε κάθε στοιχείο δεδομένων ως ένα σημείο σε n-διάστατο χώρο (όπου n είναι ένας αριθμός χαρακτηριστικών που έχετε) με την τιμή κάθε χαρακτηριστικού να είναι η τιμή μιας συγκεκριμένης συντεταγμένης. Στη συνέχεια, πραγματοποιούμε ταξινόμηση βρίσκοντας το υπερεπίπεδο(hyperplane) που διαφοροποιεί πιο καλά τις δύο κατηγορίες.

5.2 Υλοποίηση

Για την υλοποίηση μας χρησιμοποιήσαμε την βιβλιοθήκη sklearn και συγκεκριμένα την κλάση SVC. Η οποία παρόμοια με πριν μας δίνει τις απαραίτητες συναρτήσεις για να κάνουμε train, test και score το συγκεκριμένο μοντέλο. Συγκεκριμένα εμείς χρησιμοποιήσαμε τον linear kernel για το svm καθώς το πρόβλημα μας είναι γραμμικό και έπειτα από έρευνα καταλήξαμε ότι δίνει τα καλύτερα score.

6 K-Nearest Neighbours

Ο αλγόριθμος k-nearest neighbours (KNN) είναι ένας αλγόριθμος εποπτευόμενης μηχανικής μάθησης που μπορεί να χρησιμοποιηθεί για την επίλυση classification όσο και regression προβλημάτων. Ο αλγόριθμος KNN υποθέτει ότι παρόμοια πράγματα υπάρχουν σε κοντινή απόσταση. Με άλλα λόγια, παρόμοια πράγματα είναι κοντά το ένα στο άλλο. Ο αλγόριθμος εξαρτάται από αυτή την υπόθεση ώστε να είναι χρήσιμος. Το KNN συλλαμβάνει την ιδέα της ομοιότητας υπολογίζοντας την απόσταση μεταξύ των σημείων σε ένα γράφημα. Η απόσταση μπορεί να διαφέρει ανάλογα με το πρόβλημα αλλά συνήθως επιλέγεται η Ευκλίδεια απόσταση.

6.1 Υλοποίηση

Στην υλοποίηση μας χρησιμοποιήσαμε την βιβλιοθήκη sklearn και συγκεκριμένα την κλάση KNeighborsClassifier στης οποίας τον constructor δώσαμε ως όρισμα τους γείτονες. Έπειτα με τις συναρτήση που αναφέρθηκαν πριν ακολουθήσαμε την ίδια διαδικασία για το train και το test. Στην περίπτωση μας, χρησιμοποιήσαμε 3 γείτονες γιατί μετά από διάφορα test runs είχε το μεγαλύτερο score.

7 Keras Neural Network

7.1 Υλοποίηση

Για την υλοποιήση του νευρωνικού μας δικτύου, χρησιμοποιήσαμε το sequential model από το Keras. Το sequential model είναι ουσιαστικά μια γραμμική στοίβα απο Layers, και μπορούμε με κατάλληλες μεθόδους να προσθέσουμε όσα επιθυμούμε. Στο παρών νευρωνικό δίκτυο έχουμε 3 layers. Το input layer το οποίο έχει input dimension 100, δηλαδή το μέγεθος του output vector του Doc2Vec. Το hidden layer kai το output layer, το οποίο εχει 1 νευρώνα καθώς το output είναι spam or ham. Ως Loss function ορίσαμε την binary crossentropy εφόσον το πρόβλημα ήταν binary classification.

8 Results

Method	Accuracy	F1	Recall	Precision
KNN	0.966	0.966	0.968	0.969
Logistic Regression	0.973	0.974	0.975	0.975
Random Forests	0.98	0.977	0.977	0.977
SVM	0.977	0.975	0.976	0.976
Keras NN	0.9751	0.8481	0.83	0.8975

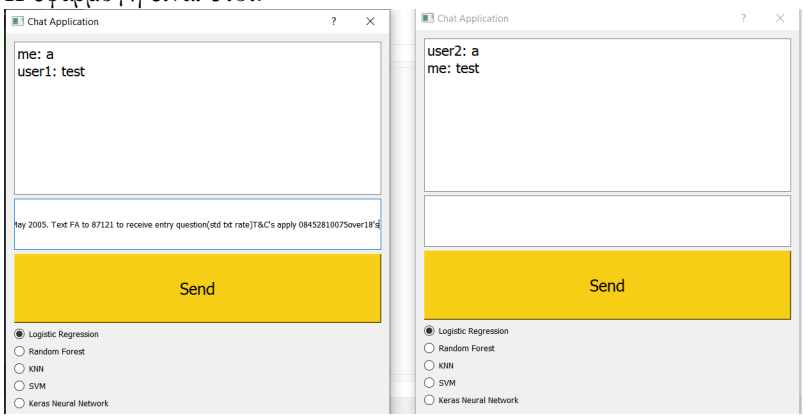
9 Chat app

Η εφαρμογή χωρίζεται σε 3 μέρη. Αρχικά έχουμε το αρχείο client.py το οποίο δημιουργεί το βασικό παράθυρο της εφαρμογής, έπειτα έχουμε τον φάκελο backend που έχει μέσα το αρχείο app.py το οποίο είναι ένα flask app που λειτουργεί ως api ανάμεσα στον client και την βάση δεδομένων. Ως βάση δεδομένων, χρησιμοποιήσαμε μία βάση mongodb με ένα collection (pendingMessages). Για να τρέξει κάποιος το πρότζεκτ θα πρέπει αρχικά να τρέξει το app.py και στην συνέχεια να τρέξει το client.py με ορίσματα username1 και username2. Οπότε οι εντολές θα ήταν:

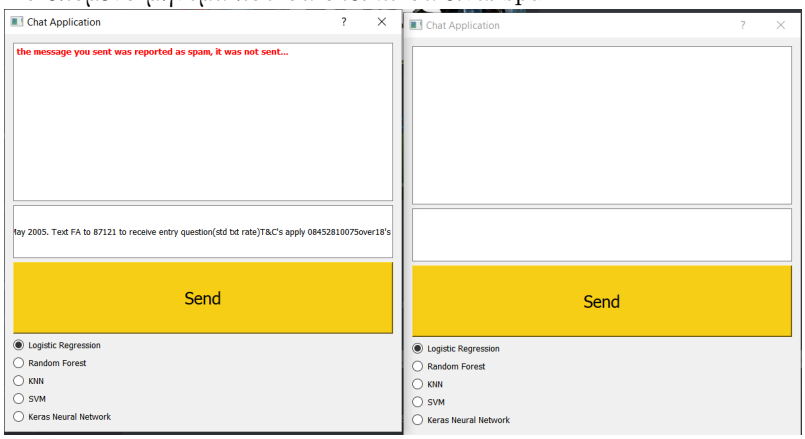
- cd backend.
- python app.py
- Σε άλλο τερματικό python client.py user1 user2

- Σε άλλο τερματικό python client.py user2 user1

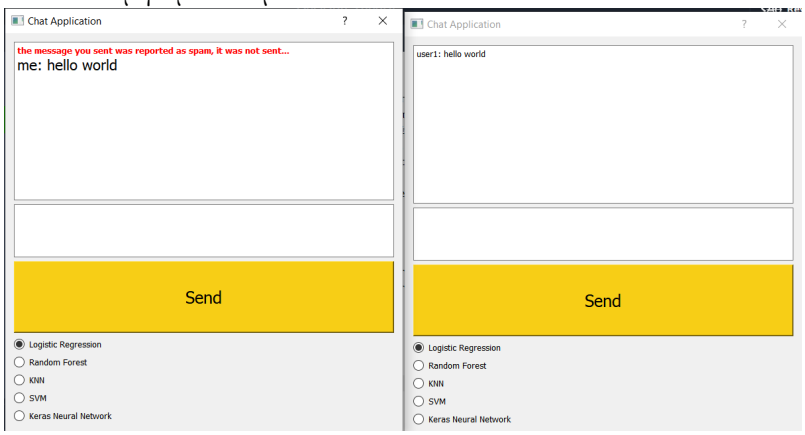
Η εφαρμογή είναι έτσι:



Το επόμενο μήνυμα που θα στείλω θα είναι spam:



Ενώ άλλα μηνύματα περνάνε κανονικά:



Τέλος μπορούμε να επιλέξουμε την μέθοδο με την οποία θα γίνει το classification

με τα radiobuttons που είναι κάτω από το κουμπί Send.

10 Project structure and comments

Στον φάκελο complete υπάρχουν ολα τα μοντελα ολοκληρωμένα και στο root υπάρχουν τα αποθηκευμένα μοντέλα μηχανικής μάθησης αλλα και του doc2vec. Στον φάκελο BOW, υπάρχουν τα ίδια μοντέλα πέρα του Keras, υλοποιημένα με Bag of Words.

Σχολιασμένα είναι τα παρακάτω αρχεία
complete/complete-keras
predict-keras