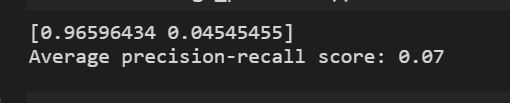
**Μέρος 1**

Χρησιμοποιήθηκαν οι παρακάτω βιβλιοθήκες της python οι οποίες περιέχονται και το pipy  
Pandas, sklearn, numpy. Για την εγκατάσταση τους πρέπει να τρέξουμε την εντολή pip install [το όνομα του αντίστοιχου πακέτου] στην κονσόλα του συστήματος μας.   
Για την υλοποίηση του πρώτου ερωτήματος χρησιμοποιήθηκαν jupyter notebooks. Αρχικά για το Α χρησιμοποιήσαμε έτοιμες συναρτήσεις plotting και άντλησης πληροφοριών που παρέχει η βιβλιοθήκη pandas. Για κάθε χαρακτηριστικό είδαμε ένα bar plot και το πεδίο τιμών της στήλης καθώς και συνολικά διάφορες πληροφορίες με την χρήση της describe().

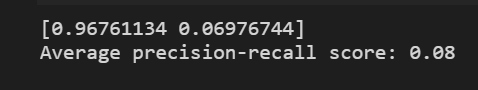
Όσον αφορά το Β1 η αφαίρεση της στήλης έγινε με την εντολή dropna της βιβλιοθήκης pandas. Για το B2, παρόμοια με πρίν χρησιμοποιήθηκε η εντολή fillna(γεμίζει, όπου υπάρχει NaN) όπου το value argument πήρε την τιμή csv.mean() που υπολογίζει τον μέσο όρο της στήλης. Για το 3ο ερώτημα χρησιμοποιήθηκε η κλάση linearRegressor του sklearn. Αρχικά κάναμε προεπεξεργασία στα δεδομένα μας μετατρέποντας όλα τα string σε int με την χρήση του labelEncoder επίσης από την βιβλιοθήκη του sklearn. Έπειτα με την συνάρτηση fit δημιουργήσαμε το μοντέλο και το ¨ ταιριάξαμε¨ στα δεδομένα μας. Έπειτα με την μέθοδο predict υπολογίσαμε τις τιμές του bmi σύμφωνα με το linear regression. Για να μην γίνει αλλοίωση των δεδομένων, χρησιμοποιήθηκε πάλι η fillna με value την καινούρια στήλη που δημιουργήθηκε, οπότε από το αρχικό dataframe γέμισαν μόνο η γραμμές που δεν είχαν δεδομένα και γέμισαν με την αντίστοιχη πρόβλεψη.   
Τέλος, το τέταρτο ερώτημα έγινε με τον ακριβώς ίδιο τρόπο με το τρίτο, με την διαφορά ότι χρησιμοποιήσαμε την κλάση KNeighborsClassifier.

Για την υλοποίηση του Γ ερωτήματος, παρόμοια με την Β3 και Β4 χρησιμοποιήθηκε η κλάση RandomForestRegressor στην οποία δώσαμε τα training data αφού τα χωρίσαμε με την χρήση της train\_test\_split που επιστρέφει τα αντίστοιχα label και data για training και testing. Έπειτα καλέσαμε την συνάρτηση predict για να μας υπολογίσει το stroke σύμφωνα με τα test\_data. Για τις μετρικές, χρησιμοποιήσαμε το test\_labels που είναι η πραγματικές τιμές του stroke για τα test\_data. Όποτε συγκρίναμε την πρόβλεψη με τα πραγματικά δεδομένα.

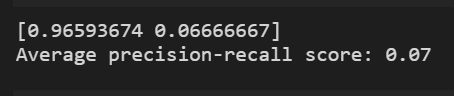
Για το Β1, το f1 και το precision-recall:



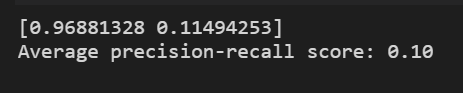
Για το Β2 το f1 και το precision-recall:



Για το Β3 το f1 και το precision-recall:



Για το Β4 το f1 και το precision-recall:



**Μέρος 2**

**Βιβλιοθήκες που χρησιμοποιήθηκαν**

**Tensorflow** -

pip install tensorflow

**Genism** -

pip install gensim

**Nltk –**

pip install nltk

import nltk

nltk.download()

Λήψη όλων των πακέτων διότι δεν μου δόθηκε επιλογή

**Pandas**

**Numpy**

**Random**

**IDE – VSCode – Jupyter notebooks**

**Mέρος 2**

**Word Embeddings**

**Αρχείο μοντέλου – doc2vecModel.d2v**

**Αρχείο πηγαίου κώδικα – doc2vec.py**

Χρησιμοποιήθηκε η βιβλιοθήκη Doc2Vec , όπου στο μοντέλο δόθηκαν τα emails ως tokenized arrays.

1) Μετατροπή σε string όλα τα rows των emails

2) Χρήση word\_tokenizer από (nltk.tokenize)

3) Μετατροπή όλων των γραμμάτων σε μικρά

Ύστερα τα rows ανακατεύθηκαν.

Εισάχθηκαν στο σύστημα ως λεξιλόγιο και το μοντέλο αποθηκεύτηκε στο ίδιο directory.

**Neural Network**

**Αρχείο εκπαιδευμένου μοντέλου – kerasModel**

**Αρχείο πηγαίου κώδικα – keras.py**

Χρησιμοποιήθηκε το νευρωνικό δίκτυο Keras sequential.

Τα δεδομένα δόθηκαν στο νευρωνικό δίκτυο αφού πέρασαν από το Doc2vec , ώστε να έρθουν σε μια μορφή vector.

Υπάρχει μια συνάρτηση για τον χωρισμό των δεδομένων στις 2 ομάδες και κάποιες συναρτήσεις για τις μετρικές.

**Παρατηρήσεις**

Η βασική μας παρατήρηση ήταν πως, αν τα δεδομένα δεν ήταν ανακατεμένα το testing accuracy έβγαινε πολύ μικρό διότι το dataset δεν ήταν ομοιόμορφο, με αποτέλεσμα το μοντέλο να εκπαιδευόταν σε πολλά not spam emails , και το testing σε spam. Όταν τα ανακάτευα με την χρήση της εντολής “shuffle” τα δεδομένα γινόντουσαν ομοιόμορφα και το testing accuracy αυξανόταν.