Στην εργασία για το μέρος α’ υλοποιήθηκαν δύο αλγόριθμοι μηχανικής μάθησης ο Αφελής ταξινομητής Bayes και ο Logistic Regression. Θα εξηγηθεί παρακάτω το πως υλοποιήθηκε ο καθένας ξεχωριστά. Επίσης υλοποιήθηκαν αλγόριθμοι προεπεξεργασίας των δεδομένων οι οποίοι θα αναλυθούν και αυτοί. Τέλος, οι υλοποιήσεις μας συγκρίθηκαν με τους αντίστοιχους αλγόριθμους του Scikit-learn για το μέρος β’ και με ένα RNN για το μέρος γ’.

Τα αρχεία της εργασίας είναι δομημένα ως εξής:

Φάκελος data: τα δεδομένα που παράγονται έπειτα από την προεπεξεργασία των δεδομένων του IMDB.

Φάκελος x_reports: τα αποτελέσματα εκτέλεσης των αλγοριθμών.

Φάκελος src: τα source files γραμμένα σε python.

Ξεκινώντας με τον φάκελο src έχουμε τα αρχεία:

prepare_data.py**:** Περιέχονται οι αλγόριθμοι που προεπεξεργάζονται τα δεδομένα.

def stop_word_list(): Επιστρέφει την λίστα με τα stopwords του αρχείου stopwords.txt.

def clean_text(text):Αφαιρεί ειδικούς χαρακτήρες, πιθανά λάθη και όσες λέξεις από το stopwords περιέχει το κείμενο text.

def make_vocabulary(path, vocabulary, train_data_length): Δημιουργεί ένα λεξιλόγιο το οποίο περιέχει τις λέξεις από train_data_length αρχεία στο φάκελο path, μετά την εφαρμογή της συνάρτησης clean σε κάθε κείμενο. Το λεξιλόγιο για κάθε λέξη περιέχει και τον ποσοστό εμφάνισης της συνολικά.

def save_vocabulary(neg_pos, file_name, train_data_length): Αποθηκεύει το λεξιλόγιο στο file_name αρχείο.

def cut_words(m, n, k, file_to_read, file_to_write): Αρχικά διαβάζει τις λέξεις από το λεξιλόγιο του αρχείου file_to_read, έπειτα τις ταξινομεί σε φθίνουσα σειρά ανάλογα του ποσοστού εμφάνισης, διαγράφει τις n πιο συχνές και τις k πιο σπάνιες λέξεις και κρατάει τις m πιο συχνές από το καινούργιο λεξιλόγιο και τέλος αποθηκεύει το καινούργιο λεξιλόγιο στο αρχείο file_to_write.

def make_data_files(m, n ,k , train_data_length): Δημιουργεί τα αρχεία για τις δοθέντες παραμέτρους.

def make_vectors(file_to_read, neg_pos, train_test, data_length): Δημιουργεί τα διανύσματα για κάθε κείμενο.

def make_train_vectors(data_length): Δημιουργεί τα διανύσματα εκπαίδευσης.

def make_test_vectors(data_length): Δημιουργεί τα διανύσματα αξιολόγησης.

LogisticRegression.py:Η κλάση για την λογιστική παλινδρόμηση.

def __init__(self, learning_rate, reguralization, max_iterations): Αρχικοποίηση των παραπάνω παραμέτρων.

def sigmoid(self, t) : Υπολογίζει την πιθανότητα να είναι στην κατηγόρια 0 ή 1 βάσει του t όπου t είναι το διάνυσμα βαρών επί το διάνυσμα εισόδου.

def initialize(self, features_length): Αρχικοποίηση των βαρών με μία τυχαία τιμή από 0 ως 1.

def fit(self, x_train, y_train): Εκπαιδεύει το μοντέλο βάσει των x_train και y_train.

def predict(self, x_train): Ταξινομεί σε 0 ή 1 τα x_train.

logistic_regression_test.py: Οι αξιολογήσεις για την λογιστική παλινδρόμηση.

def prepare_data(m, n, k, data_length): Προετοιμάζει τα δεδομένα για τις παραπάνω παραμέτρους.

def test_model(model, x_train, y_train, x_test, y_test, filename): Δοκιμάζει το μοντέλο για τις παραπάνω παραμέτρους.

def custom_learning_curve(model, x_train, y_train, x_test, y_test, ,n_splits, filename) : Δημιουργεί το διάγραμμα για τις παραπάνω παραμέτρους.

def test_logistic_regression(): Δοκιμάζει τα μοντέλα λογιστικής παλινδρόμησης.

Bayes.py: Η κλάση για τον αφελή ταξινομητή Bayes.

def __init__(self): Είναι ο constructor της κλάσης.

def initialize(self): Αρχικοποίηση των λεξικών για την αποθήκευση των πιθανοτήτων και θέτει True την μεταβλητή initialized.

def classProbCaclc(self,y_train): Υπολογίζει τις πιθανότητες των κλάσεων(a priori), υπολογίζοντας την συχνότητα κάθε κλάσης στο σύνολο y_train και διαιρώντας τη συχνότητα αυτή με το πλήθος των δειγμάτων.

def featureprobCalc(self,x_train,y_train): Υπολογίζει τις πιθανότητες των χαρακτηριστικών(a posteriori) δεδομένης κάθε κλάσης στο σύνολο εκπαίδευσης και τις αποθηκεύει σε ένα λεξικό. Εφαρμόζει την ομαλοποίηση Laplace για να αποφευχθούν τα μηδενικά.

def fit(self, x_train, y_train): Εκπαιδεύει το μοντέλο βάσει των x_train και y_train.

def predict(self, x_train): Προβλέπει τις κλάσεις για ένα σύνολο δεδομένων εισόδου. Για κάθε δείγμα, υπολογίζει το σκορ για κάθε κλάση βάση της πιθανότητας κλάσης(class_prob) και τις πιθανότητες χαρακτηριστικών. Το σκορ προκύπτει από την πρόσθεση του αθροίσματος του λογαρίθμου της πιθανότητας κλάσης με το άθροισμα των λογαρίθμων των πιθανοτήτων των χαρακτηριστικών. Επιλέγει την κλάση με το μεγαλύτερο σκορ

ως προβλεπόμενη κλάση για κάθε δείγμα και την επιστρέφει ένα πίνακα με όλες τις προβλέψεις.

bayes_test.py: : Οι αξιολογήσεις για τον αφελή ταξινομητή Bayes.

def prepare_data(m, n, k, data_length): Προετοιμάζει τα δεδομένα για τις παραπάνω παραμέτρους.

def test_model(model, x_train, y_train, x_test, y_test, filename): Δοκιμάζει το μοντέλο για τις παραπάνω παραμέτρους.

def custom_learning_curve(model, x_train, y_train, x_test, y_test, ,n_splits, filename) : Δημιουργεί το διάγραμμα για τις παραπάνω παραμέτρους.

def test_bayes():Δοκιμάζει τα μοντέλα του αφελή ταξινομητή Bayes.

ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΚΤΕΛΕΣΗΣ

LogisticRegression

Μέρος α’:

Οι παράμετροι επιλέχθηκαν από δοκιμές σε ξεχωριστά δεδομένα εκπαίδευσης.

Τα αποτελέσματα εκτέλεσης για :

Features: 2000 excluding 100 most common and 100 least common

Logistic Regression Parameters:

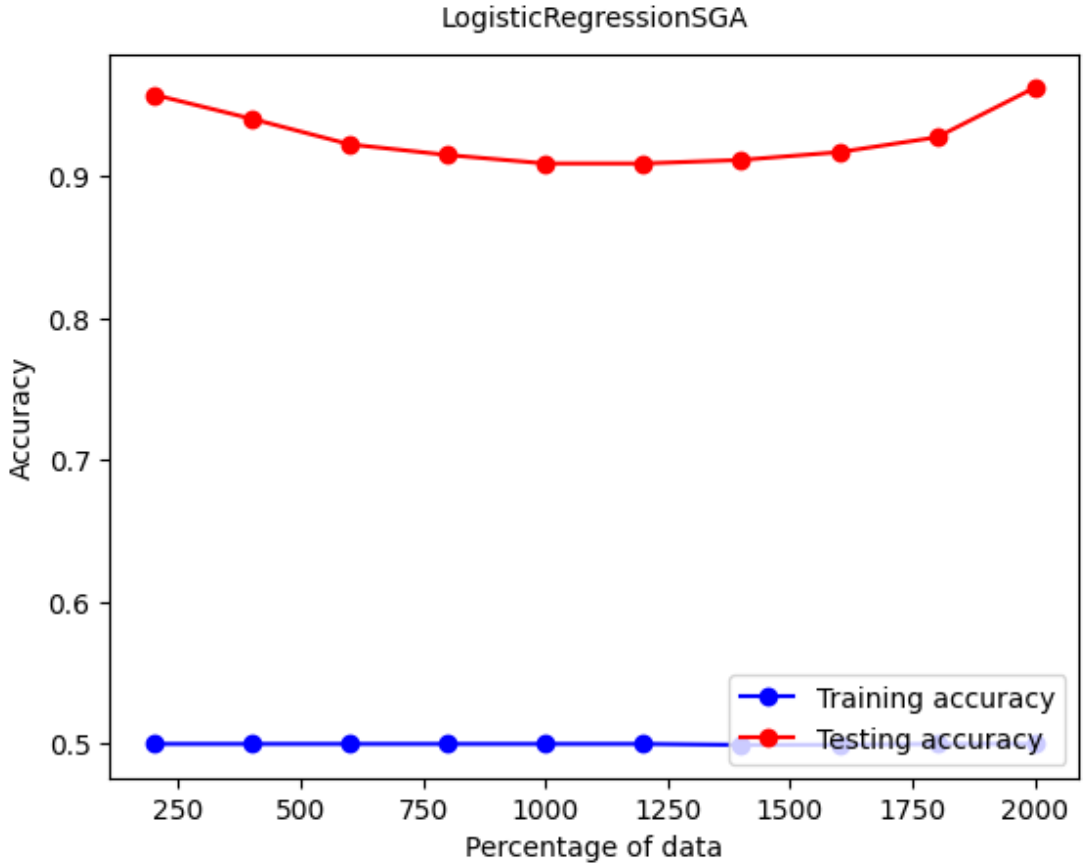
Data Length: 2000

Learning rate = 0.1

Regularization: 10

Max Iterations = 100.

```
Features:2000
Data Length:2000 of total 25000.
Data Percentage:8.0%.

Training Metrics for LogisticRegressionSGA:
              precision    recall  f1-score   support

           0       0.49      0.50      0.50       984
           1       0.51      0.50      0.50      1016

    accuracy                           0.50      2000
   macro avg       0.50      0.50      0.50      2000
weighted avg       0.50      0.50      0.50      2000

Testing Metrics for LogisticRegressionSGA:
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       974
           1       0.98      0.95      0.96      1026

    accuracy                           0.96      2000
   macro avg       0.96      0.96      0.96      2000
weighted avg       0.96      0.96      0.96      2000
```



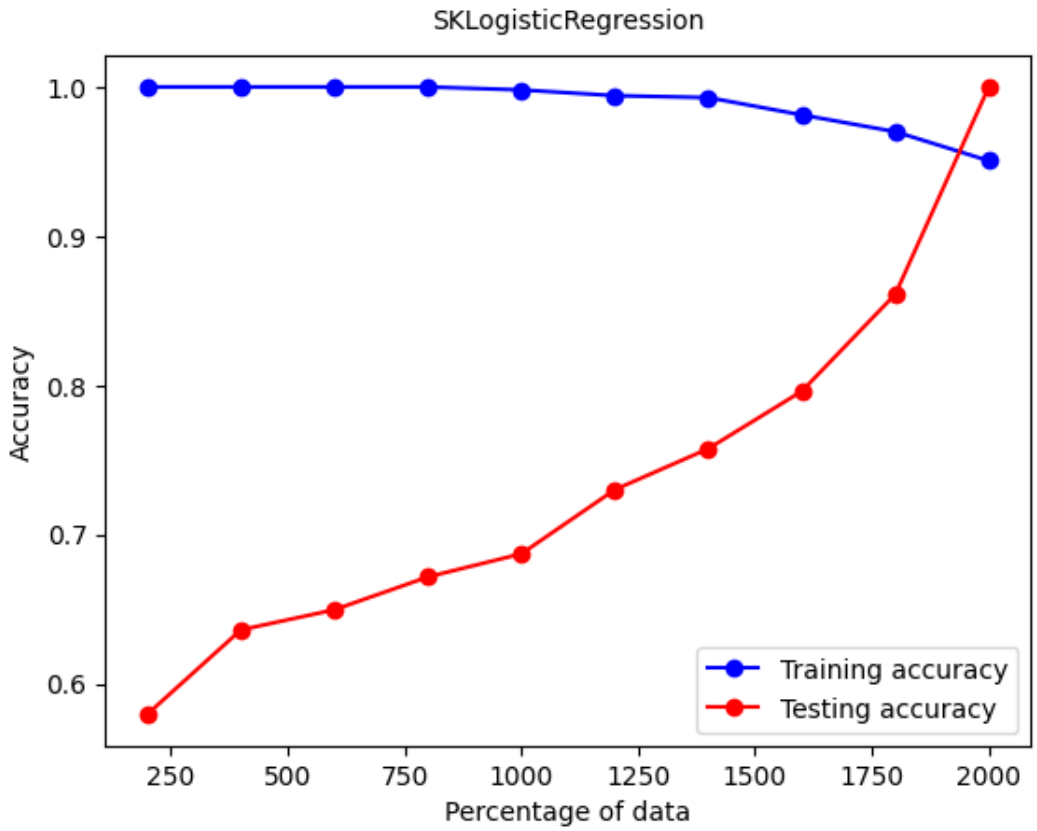
Όπως φαίνεται ο αλγόριθμος δεν πετυχαίνει πολύ καλή απόδοση κατά την εκπαίδευση, αλλά κατά την αξιολόγηση τα πηγαίνει πολύ καλύτερα. Αυτό συμπεραίνεται εύκολα από το παραπάνω διάγραμμα αλλά και από τα παραπάνω μετρικά.

Μέρος β’:

Εδώ θα συγκρίνουμε την παραπάνω υλοποίηση με την υλοποίηση του Scikit-learn.

Για τα ίδια ακριβώς δεδομένα έχουμε:

```
Features:2000
Data Length:2000 of total 25000.
Data Percentage:8.0%.

Training Metrics for SKLogisticRegression:
              precision    recall  f1-score   support

           0       0.96      0.94      0.95       984
           1       0.95      0.96      0.95      1016

    accuracy                           0.95      2000
   macro avg       0.95      0.95      0.95      2000
weighted avg       0.95      0.95      0.95      2000

Testing Metrics for SKLogisticRegression:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       981
           1       1.00      1.00      1.00      1019

    accuracy                           1.00      2000
   macro avg       1.00      1.00      1.00      2000
weighted avg       1.00      1.00      1.00      2000
```



Όπως φαίνεται ο αλγόριθμος πετυχαίνει πολύ καλή απόδοση κατά την εκπαίδευση, αλλά όσο αυξάνουμε τα δεδομένα αυτή μειώνεται. Από την άλλη, κατά την αξιολόγηση, όσο αυξάνουμε τα δεδομένα η απόδοση αυξάνεται.

Η υλοποίηση του Scikit-learn σε σχέση με την δικιά μας πετυχαίνει πολύ καλύτερες αποδόσεις και στα δεδομένα εκπαίδευσης και στα δεδομένα αξιολόγησης και αυτό συμπεραίνεται εύκολα από τα μετρικά αλλά και από τα διαγράμματα.

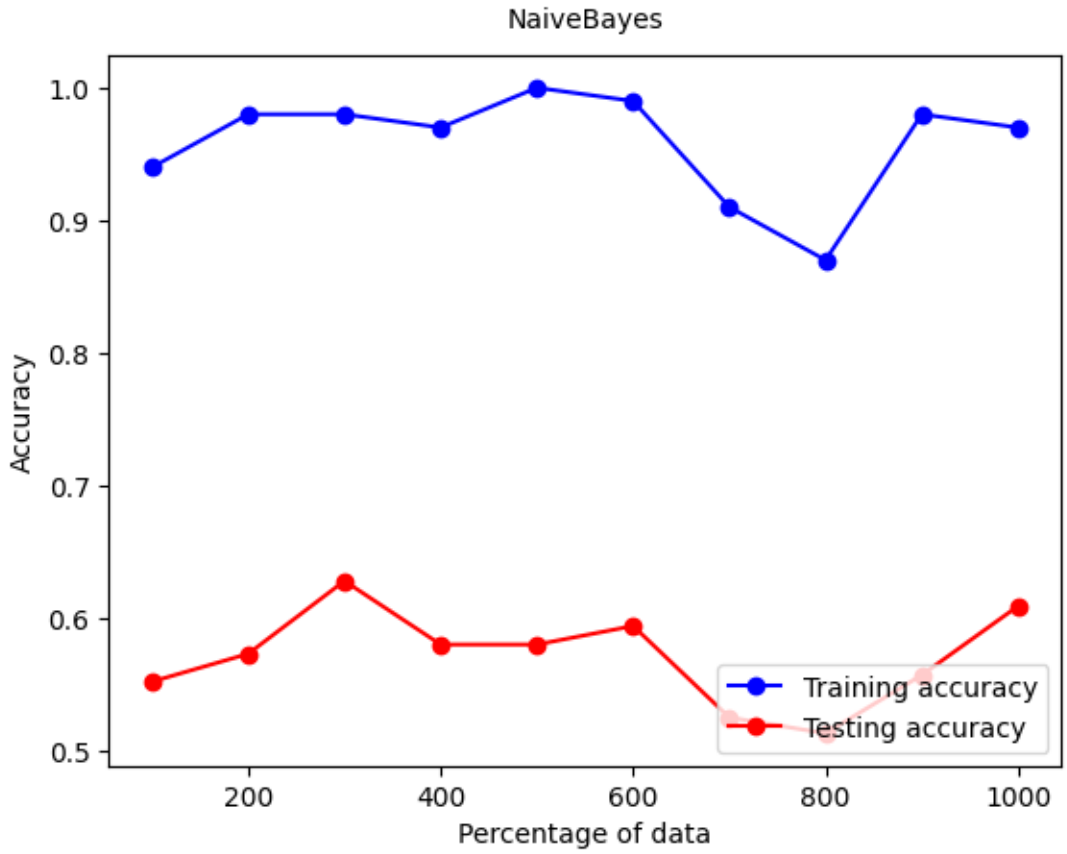
Bayes

Μέρος Α:

Οι παράμετροι επιλέχθηκαν από δοκιμές σε ξεχωριστά δεδομένα εκπαίδευσης.

Τα αποτελέσματα εκτέλεσης για: Features:1000

```
Features:1000
Data Length:1000 of total 25000.
Data Percentage:4.0%.

Training Metrics for NaiveBayes:
              precision    recall  f1-score   support

           0       0.53      0.65      0.58       493
           1       0.56      0.43      0.49       507

    accuracy                           0.54      1000
   macro avg       0.54      0.54      0.53      1000
weighted avg       0.54      0.54      0.53      1000

Testing Metrics for NaiveBayes:
              precision    recall  f1-score   support

           0       0.58      0.66      0.62       479
           1       0.64      0.56      0.60       521

    accuracy                           0.61      1000
   macro avg       0.61      0.61      0.61      1000
weighted avg       0.61      0.61      0.61      1000
```



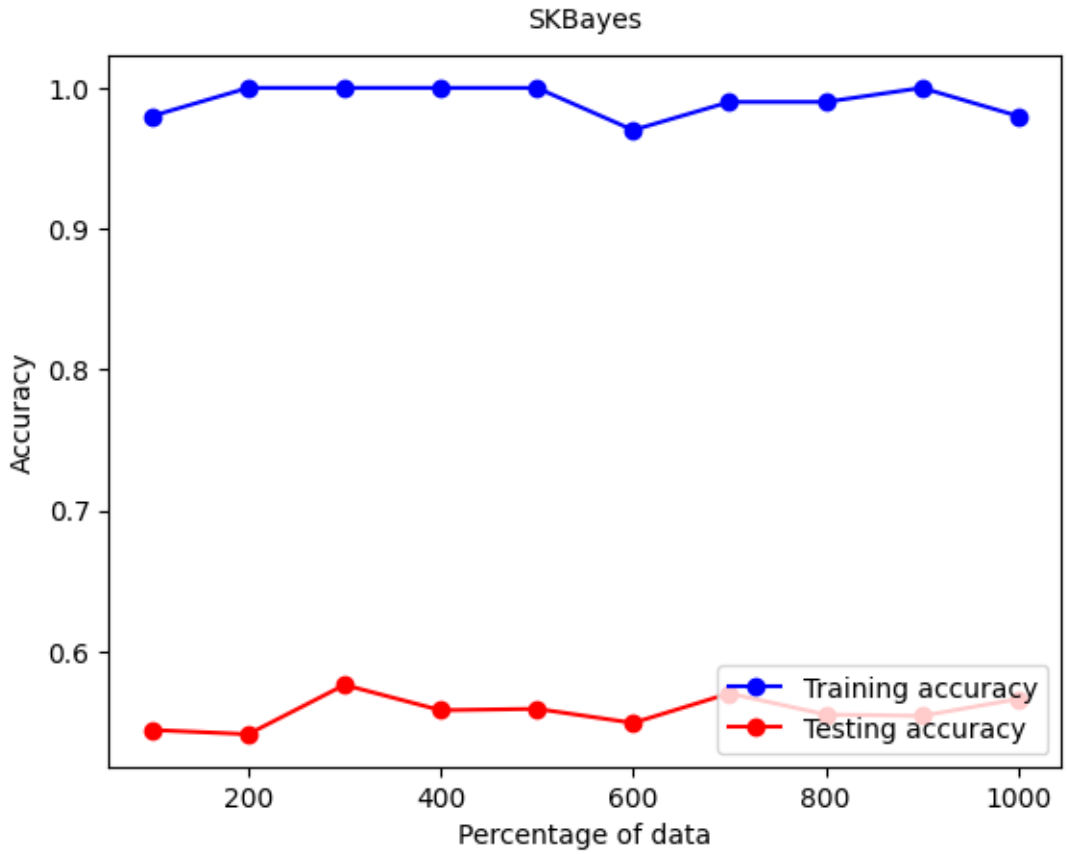
Από τα παραπάνω παρατηρείται ότι η ακρίβεια του αλγορίθμου κατά την εκπαίδευση είναι μεγαλύτερη σε σχέση με την ακρίβεια του κατά την αξιολόγηση. Αυτό μπορεί να υποδηλώνει πως ο ταξινομητής Bayes μοντελοποιεί καλά τα δεδομένα εκπαίδευσης αλλά όχι εξίσου καλά νέα δεδομένα.

Μέρος Β:

Εδώ θα συγκρίνουμε την παραπάνω υλοποίηση με την υλοποίηση του Scikit-learn.

Για τα ίδια ακριβώς δεδομένα έχουμε:

```
Features:1000
Data Length:1000 of total 25000.
Data Percentage:4.0%.

Training Metrics for SKBayes:
              precision    recall  f1-score   support

           0       0.55      0.38      0.45       493
           1       0.54      0.70      0.61       507

    accuracy                           0.54      1000
   macro avg       0.54      0.54      0.53      1000
weighted avg       0.54      0.54      0.53      1000

Testing Metrics for SKBayes:
              precision    recall  f1-score   support

           0       0.54      0.37      0.43       456
           1       0.58      0.73      0.65       544

    accuracy                           0.57      1000
   macro avg       0.56      0.55      0.54      1000
weighted avg       0.56      0.57      0.55      1000
```



Όπως παρατηρείται η υλοποίηση αυτή παρουσιάζει υψηλή ακρίβεια κατά την εκπαίδευση η οποία παραμένει σχετικά σταθερή όσο αυξάνεται το ποσοστό των δεδομένων. Από την άλλη, η ακρίβεια κατά την αξιολόγηση κυμαίνεται σε χαμηλά επίπεδα, αλλά βελτιώνεται όσο αυξάνονται τα δεδομένα, γεγονός που φανερώνει μια καλή γενίκευση του μοντέλου.

Μέρος Γ:

Για το μέρος Γ υλοποιήθηκε ένα RNN με παραμέτρους:

vocab_size = len(x_train[0]) // το μέγεθος του λεξιλογίου

embedding_dim = 50

max_sequence_length = len(x_train[0])

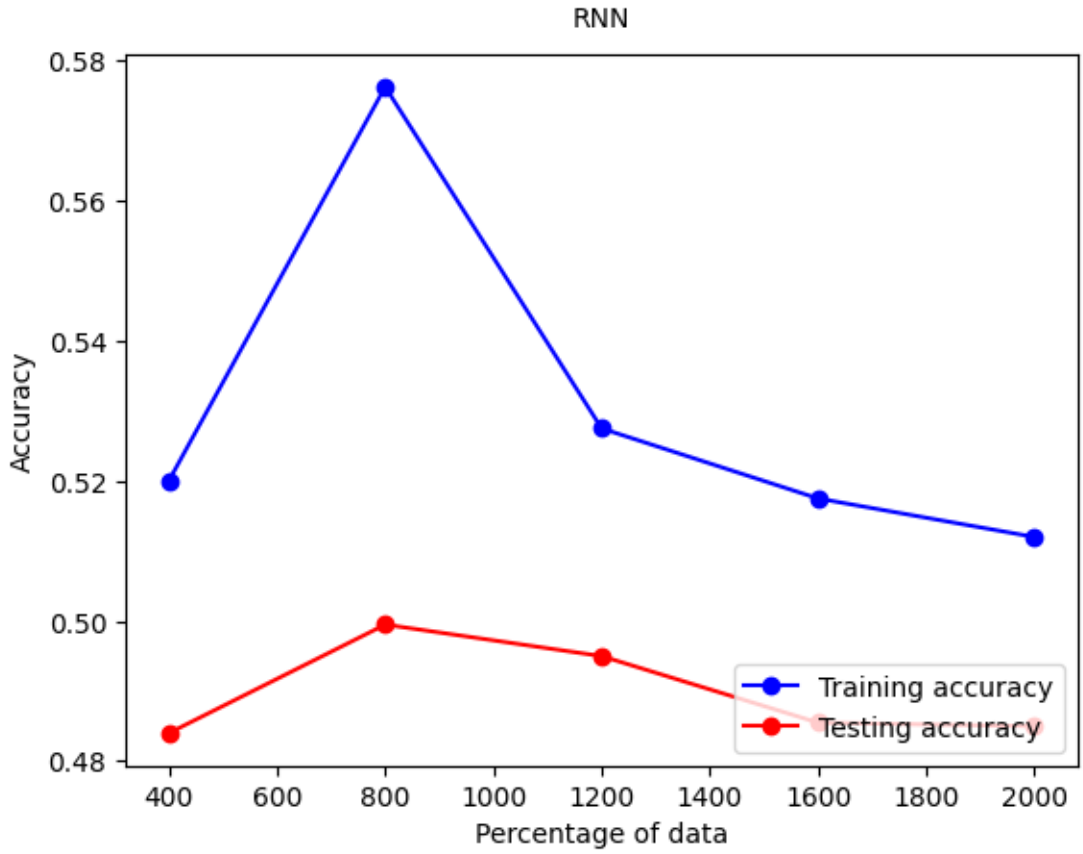
rnn_units = 50 // αριθμός των νευρώνων

run_epochs = 10 // αριθμός των εποχών

Ακολουθούν κάποια στατιστικά για το RNN:

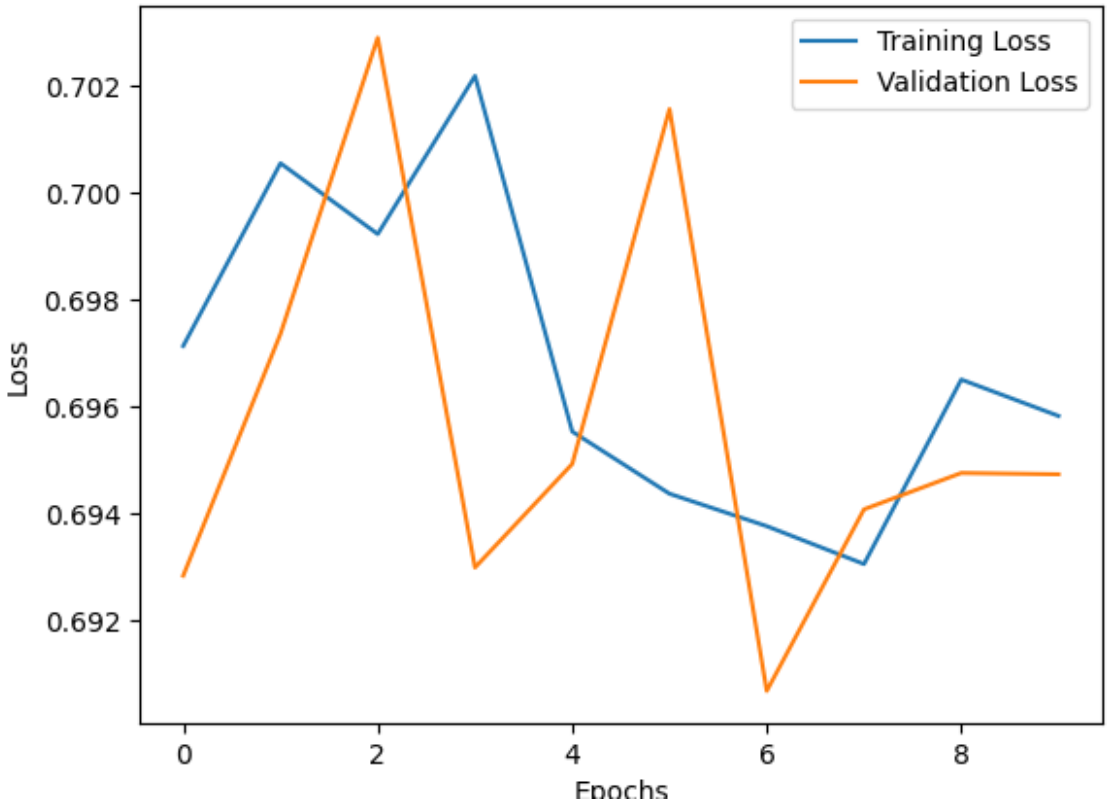
```
Features:2000
Data Length:2000 of total 25000.
Data Percentage:8.0%.

Training Metrics for RNN:
              precision    recall  f1-score   support

           0       0.51      0.23      0.32       984
```

```
           1       0.51      0.78      0.62      1016

    accuracy                           0.51      2000
   macro avg       0.51      0.51      0.47      2000
weighted avg       0.51      0.51      0.47      2000

Testing Metrics for RNN:
              precision    recall  f1-score   support

           0       0.48      0.21      0.29      1016
           1       0.49      0.77      0.59       984

    accuracy                           0.48      2000
   macro avg       0.48      0.49      0.44      2000
weighted avg       0.48      0.48      0.44      2000
```



Όπως φαίνεται το RNN μέτρια απόδοση και στα δεδομένα εκπαίδευσης αλλά και στα δεδομένα αξιολόγησης. Αυτό μπορεί να οφείλεται στις παραμέτρους που χρησιμοποιήθηκαν.

Τέλος το διάγραμμα μεταβολής σφάλματος:



ΕΚΤΕΛΕΣΗ ΠΡΟΓΡΑΜΜΑΤΟΣ

Για να τρέξετε τους αλγορίθμους τρέξτε τα αρχεία <name>_test στο φάκελο src.