Τεχνητή Νοημοσύνη 2η Εργασία

Ταξινόμηση κριτικών IMDB

- Σοφία-Ζωή Σωτηρίου : 3210192
- Ευάγγελος Λευτάκης 3200093
- Ρέα Σκλήκα 3210181

Αρχικοποίηση Δεδομένων

Αφού λάβουμε το imdb dataset μαζί το λεξικό word index του μέσω του Keras, προσθέτουμε στο λεξικό τις λέξεις [pad], [bos], [oov] που είναι tokens με ειδική σημασία για την αναπαράσταση κειμένου. Έπειτα με την χρήση του countVectorizer φιλτράρουμε τις λέξεις και κρατάμε μόνο αυτές που εμφανίζονται τουλάχιστον 100 φορές (min_df) και παράλληλα μετατρέπουμε τις προτάσεις σε δυαδικές αναπαραστάσεις από 0 για απουσία και 1 για εμφάνιση μιας λέξης σε κάθε πρόταση. Αλλάζουμε τον τύπο τον δεδομένων από float64 που είναι το default σε integer για βελτιωμένη απόδοση. Τέλος, μετατρέπουμε τα δεδομένα μάθησης και ελέγχου σε dense arrays με την χρήση της toarray() καθώς by default λαμβάνουμε ένα sparse array από τον count vectorizer η οποία μορφή δεν είναι συμβατή με πολλές από τις διαδικασίες που απαιτούνται για τις διάφορες μεθόδους τις εργασίας.

```python
from sklearn.feature_extraction.text import CountVectorizer

(x_train_imdb, y_train), (x_test_imdb, y_test) = tf.keras.datasets.imdb.load_data()

word_index = tf.keras.datasets.imdb.get_word_index()
index2word = dict((i + 3, word) for (word, i) in word_index.items())
index2word[0] = '[pad]'
index2word[1] = '[bos]'
index2word[2] = '[oov]'
x_train_imdb = np.array([' '.join([index2word[idx] for idx in text]) for text in x_train_imdb])
x_test_imdb = np.array([' '.join([index2word[idx] for idx in text]) for text in x_test_imdb])

binary_vectorizer = CountVectorizer(binary=True, min_df=100,stop_words="english")
x_train = binary_vectorizer.fit_transform(x_train_imdb)
x_test = binary_vectorizer.transform(x_test_imdb)
print(
    'Vocabulary size:', len(binary_vectorizer.vocabulary_)
)
x_train = x_train.astype(int)
x_test = x_test.astype(int)

# x_train = x_train.toarray()
# x_test = x_test.toarray()
```

1ο Μέρος

Για να κάνουμε την κατηγοριοποίηση των κριτικών από το dataset μας στο πρώτο κομμάτι της εργασίας αναπτύξαμε APIs για τους αλγορίθμους μηχανικής μάθησης Naive Bayes, Logisitic Regression και AdaBoost και να αξιολογήσουμε τις επιδόσεις τους.

Logistic regression:

Ο αλγόριθμος που υλοποιήσαμε είναι λογιστικής παλινδρόμησης με L2 κανονικοποίηση και κανόνα ανανέωσης βαρών:

$$w_l \leftarrow (1 - 2 \cdot \lambda \cdot \eta) \cdot w_l + \eta \cdot \sum_{i=1}^{m} \left[y^{(i)} - P\left(c_+ | \vec{x}^{(i)}\right) \right] x_l^{(i)}$$

Η διεπαφή αποτελείται από τις παρακάτω μεθόδους:

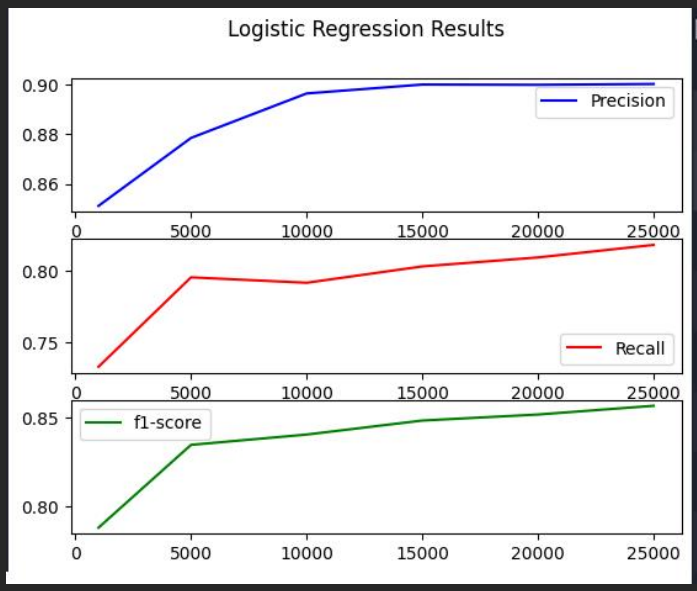
- __init__(*epochs* ,*learning_rate* ,*threshold* ,*regularization_factor*): Παίρνει ως ορίσματα τις διάφορες υπερπαραμέτρους όπως αριθμός επαναλήψεων, ρυθμός μάθησης παράγοντας κανονικοποίησης και threshold. Κατασκευάζει και επιστρέφει τον ταξινομητή με τα ορίσματα που προσδιορίσαμε.
- fit(x_train_input,y_train_input,learning_curve): Παίρνει ως ορίσματα τα δεδομένα εκπαίδευσης και μια boolean τιμή σε περίπτωση που θέλουμε να παραχθεί καμπύλη μάθησης. Αρχικά διαιρούμε τα δεδομένα σε εκπαίδευσης και επικύρωσης 80-20 αντίστοιχα. Για κάθε εποχή τα παραδείγματα αναδιατάσσονται τυχαία προς αποφυγή overfitting και να μην δημιουργηθούν γνώσεις που έχουν να κάνουν με την διάταξη των παραδειγμάτων εκπαίδευσης. Για την ενημέρωση των βαρών χρησιμοποιούμε Stochastic Gradient Ascent χωρίς mini batches. Επιστρέφει διάφορες μετρικές που παράγονται από την evaluate() (εμφανίζεται παρακάτω
- predict(x_test): Παίρνει ως όρισμα τα δεδομένα ελέγχου. Με τον τύπο τις σιγμοειδούς καμπύλης $P(C = 1) = \frac{1}{1+e^{-w*x}}$ υπολογίζουμε την πιθανότητα κάθε παράδειγμα ελέγχου να είναι στην θετική κατηγορία. Έπειτα αν η πιθανότητα αυτή είναι μεγαλύτερη από το threshold θωρούμε ότι η κριτική είναι θετική. Επιστρέφει έναν πίνακα y_predicted με μήκος ίσο με τα παραδείγματα ελέγχου ο οποίος αποτελείται από 0 και 1 για εκτιμόμενα θετική ή αρνητική κριτική.
- predict_prob(x_test): Το ίδιο με την predict απλά επιστρέφει τις εκτιμώμενες πιθανότητες για τα δεδομένα ελέγχου χωρίς να τα κατατάσσει.
- initialize_weights(): Αρχικοποιούμε τα βάρη σε τυχαίους αριθμούς πολύ κοντά στο μηδέν.
- update_weights(x_train, y_train): Χρησιμοποιώντας sga με τον τύπο ανανεώνουμε τα βάρη με τον τύπο ανανέωσης.

- evaluate(y_test, y_predicted):
  Συγκρίνουμε τα αποτελέσματα που εκτιμήθηκαν με τα πραγματικά και επιστρέφουμε ένα πίνακα με μετρικές όπως [precision,recall,f1]
- get_params(): Υλοποιούμε για συμβατότητά με μεθόδους του sci-kit learn όπως learning_curve()
- classification_diagrams(x_train, y_train, x_test, y_test, train_sizes): Παράγει τρία plots για την πορεία των precision, recall, f1 score συναρτήσει του μεγέθους των δεδομένων εκπαίδευσης.
- custom_classification_report(*x_train, y_train,x_test, y_test, train_sizes*): Καλεί την classification_diagrams() και την my_learning_curve() για να δώσει μια πλήρη εικόνα των αποτελεσμάτων μάθησης του ταξινομητή που π.
- Άλλες Βοηθητικές μέθοδοι όπως set_threshold() για αλλαγή του threshold ενός ταξινομητή, sigmoid() για υπολογισμό του τύπου .

Αποτελέσματα:

Ο ταξινόμητής που χρησιμοποιήσαμε για τα δεδομένα μάθησης τελικά είχε τις ύπερ παραμέτρους:

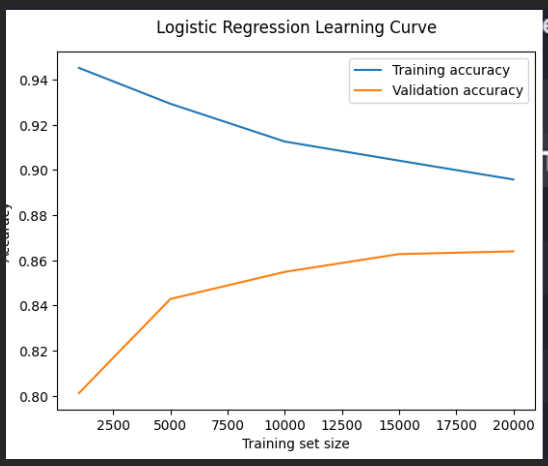
- Threshold: του οποίου δώσαμε διάφορες τιμές και καταλήξαμε ότι κάτι κοντά στο (0.4,0.6) έχει καλά αποτελέσματα και μια καλή ισορροπία ανάμεσα σε precision και recall.
- Regularization Factor λ: βάζουμε χαμηλό παράγοντα κανονίκοποίησης (0.001) καθώς για το πρόβλημα μας που είναι sentiment analysis επωφελούμαστε από την καλή αφομοίωση των δεδομένων εκπαίδευσης και αποφεύγοντας το overfitting
- Learning Rate: Αφήνουμε την τιμή στο default που έχουμε ορίσει 0.001 που προσφέρει πιο αργό ρυθμό μάθησης αλλά περιορίζει το ενδεχόμενο να χάσουμε κατά πολύ τα βέλτιστα βαρύ και να ταλαντευόμαστε γύρω από την μέγιστη τιμή πιθανοφάνειας
- Epochs: Επιλέξαμε 100 εποχές κρίνοντας από τα αποτελέσματα της καμπύλης μάθησης.



Logistic Regression Results:

Training Set Size	Precision	Recall	F1-Score
1000	0.851184	0.73304	0.787707
5000	0.878480	0.79520	0.834768
10000	0.896349	0.79144	0.840634
15000	0.899919	0.80280	0.848590
20000	0.899813	0.80904	0.852016
25000	0.900141	0.81776	0.856975

Στην καμπύλη μάθησης παρατηρούμε την μείωση της ακρίβειας στα δεδομένα εκπαίδευσης που συνάδει με την μείωση του overfitting, ενώ αυξάνεται η ακρίβεια στα Development δεδομένα που δείχνει ότι ο αλγόριθμος μαθαίνει.



earning Curve Data:

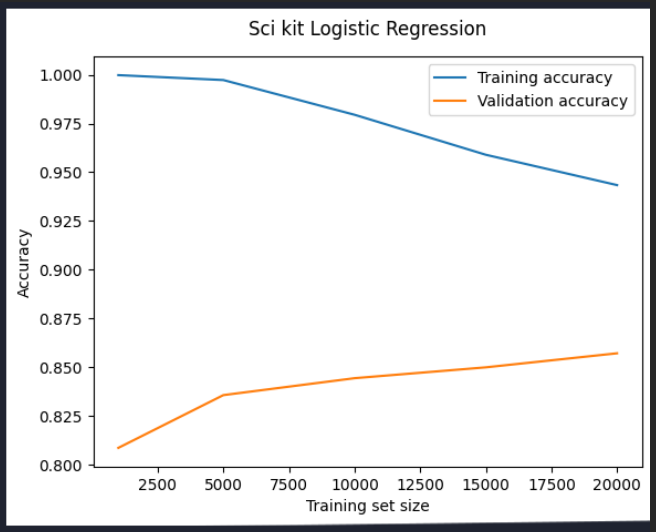
Training Set Size	Training Accuracy	Validation Accuracy
1000	0.945200	0.80120
5000	0.929320	0.84288
10000	0.912600	0.85488
15000	0.904133	0.86276
20000	0.895790	0.86396

2ο Μέρος

Σε αυτό το σημείο θα συγκρίνουμε τις υλοποιήσεις των αλγορίθμων μας με τις υλοποιήσεις του sci-kit learn. Για αυτή την σύγκριση πέρα τον μετρικών ακρίβειας όπως accuracy, precision, recall και f1 score έχουμε υλοποιήσει διάφορα διαγράμματα όπως precision-recall καμπύλη, ROC καμπύλη και καμπύλη μάθησης.

Logistic regression:

Καμπύλες μάθησης:



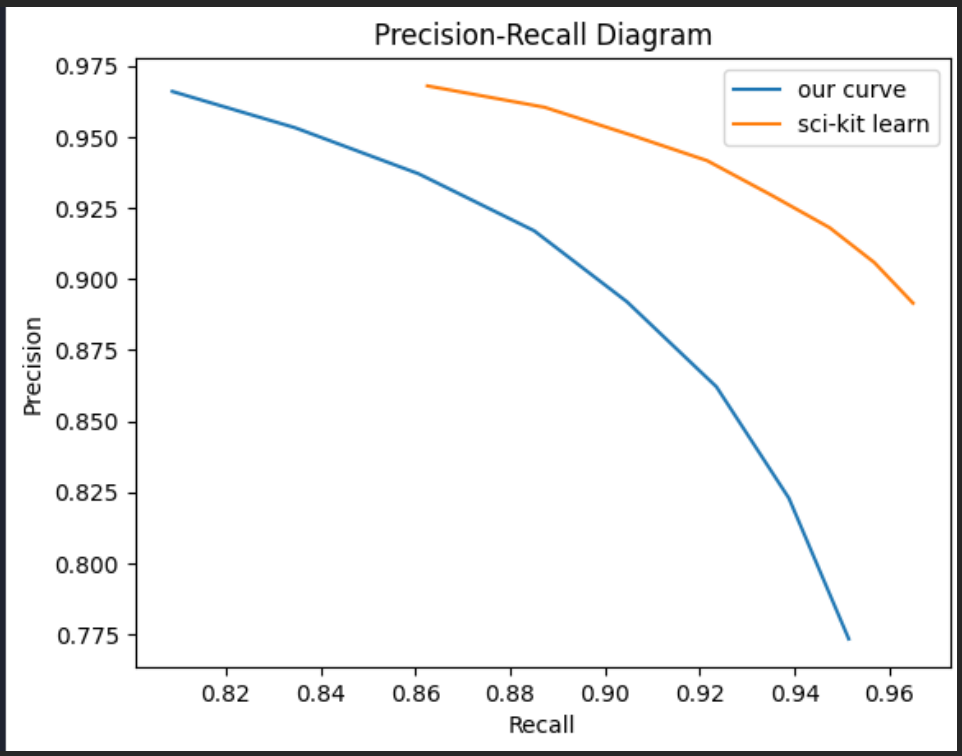
Learning Curve Data:

Training Set Size	Training Accuracy	Validation Accuracy
1000	0.945000	0.80168
5000	0.929400	0.84244
10000	0.912080	0.85588
15000	0.904173	0.86208
20000	0.896850	0.86380

Learning Curve Data:

Training Set Size	Training Accuracy	Validation Accuracy
1000	0.999800	0.80872
5000	0.997280	0.83576
10000	0.979480	0.84444
15000	0.958973	0.85000
20000	0.943420	0.85720

Precision-Recall διάγραμμα:



Καμπύλες ROC: