

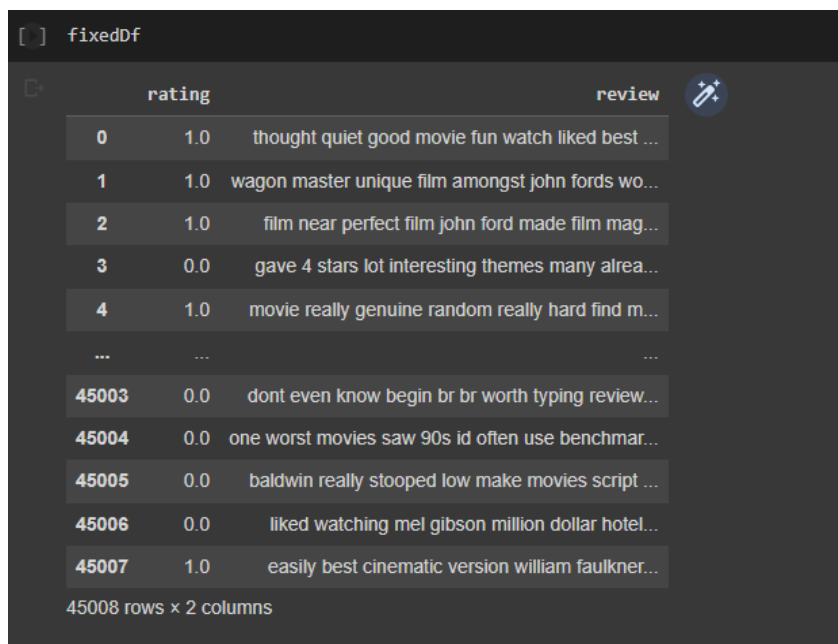
ΘΕΟΦΑΝΟΠΟΥΛΟΣ ΜΙΧΑΗΛ

111 520 18 00 053

Εργασία 2 Τεχνητή Νοημοσύνη II

Για το καθάρισμα των δεδομένων χρησιμοποίησα την εξής προσέγγιση:

1. Αρχικά κρατάω μόνο τις στήλες των *rating* και *review*, καθώς θα χρειαστούμε τη πρώτη για το *training* του μοντέλου και τη δεύτερη για να γίνει το *prediction*.
2. Στη συνέχεια, αντικαθιστούμε όλα τα *reviews* με είτε 0.0 είτε 1.0 εάν είναι 5 αντίστοιχα, ώστε να μπορεί να γίνει το *prediction*.
3. Ύστερα, αφαιρούμε από τα κείμενα των *reviews* όλα τα *emojicons* και τα σύμβολα που μπορεί να περιέχονται ώστε να υπάρχει σκέτο κείμενο. Πέραν αυτών, αφαιρούμε τα *links*, *url's*, σημεία στίξης και τους αριθμούς από το κείμενο.
4. Εκτελούμε τη διαδικασία του *Tokenization* στα *reviews*.
5. Εκτελούμε τη διαδικασία του *Stemming* στα *reviews*.
6. Εκτελούμε τη διαδικασία του *Lemmatization* στα *reviews*. Αφού έχουμε τελειώσει με το καθάρισμα των δεδομένων, κρατάμε σε δυο ξεχωριστούς πίνακες τη στήλη *review* και τη στήλη *rating*. Η μορφή στην οποία βρίσκονται αυτή τη στιγμή τα δεδομένα μας είναι η εξής:



	rating	review
0	1.0	thought quiet good movie fun watch liked best ...
1	1.0	wagon master unique film amongst john fords wo...
2	1.0	film near perfect film john ford made film mag...
3	0.0	gave 4 stars lot interesting themes many alrea...
4	1.0	movie really genuine random really hard find m...
...
45003	0.0	dont even know begin br br worth typing review...
45004	0.0	one worst movies saw 90s id often use benchmar...
45005	0.0	baldwin really stooped low make movies script ...
45006	0.0	liked watching mel gibson million dollar hotel...
45007	1.0	easily best cinematic version william faulkner...

45008 rows x 2 columns

Παρατηρούμε ότι τα reviews είναι πλήρως καθαρισμένα και έχουμε κρατήσει μόνο τις σημαντικές για την επεξεργασία μας λέξεις.

Το επόμενο βήμα είναι να χωρίσουμε τα δεδομένα μας σε training και testing set, με σκοπό να προχωρήσουμε στη δημιουργία του μοντέλου μας.

Εάν δωθεί graders_data (**TODO**: store your data set in this variable), τότε ως training set κρατάμε ολόκληρο το data set που έχουμε και ως test set κρατάμε το data set του grader. Διαφορετικά, κάνουμε split το set σε 80-20.

Ύστερα από τη διαδικασία του καθαρισμού των δεδομένων, ακολουθεί η εξής προσέγγιση:

1. Εκτελούμε τη διαδικασία του vectorization για κάθε review. Στο πρώτο μοντέλο επιτυγχάνεται αξιοποιώντας GloVe pre-trained words embedding vectors, ενώ στο δεύτερο μοντέλο χρησιμοποιείται ένας TF-IDF vectorizer.
2. Αρχικοποιούμε ένα νευρωνικό δίκτυο και στα 2 μοντέλα μας.
3. Ορίζουμε το loss function και τον optimizer μαζί με τις αντίστοιχες παραμέτρους του που θα χρησιμοποιήσει το κάθε μοντέλο μας. Το loss function που χρησιμοποιούμε στην υλοποίηση μας είναι το CrossEntropyLoss, καθώς το πρόβλημα μας είναι τύπου multiclass classification.
4. Εκπαιδεύουμε το μοντέλο μας για έναν αριθμό από epochs. Σε κάθε epoch υπολογίζουμε το:
 - i. Validation loss
 - ii. Validation accuracy
 - iii. Validation f1-scoreκαι τα εκτυπώνουμε κατάλληλα.
5. Δοκιμάζουμε και αξιολογούμε το μοντέλο μας, κατασκευάζοντας και παρουσιάζοντας τα loss vs epoch curves καθώς και τα ROC curves για κάθε μια κλάση ώστε να εμφανίζουμε τη συμπεριφορά του μοντέλου μας κατά την εκπαίδευση και να βγάζουμε τα κατάλληλα συμπεράσματα.

Για το GloVe embeddings, επέλεξα να αξιοποιήσω το αρχείο **glove.42B.300d.zip** καθώς με το συγκεκριμένο λόγω του μεγέθους του παρατήρησα καλύτερα αποτελέσματα στα αποτελέσματά μου.

Ύστερα από το κατέβασμα του αρχείου, κατασκευάζουμε ένα dictionary που περιέχει την αναπαράσταση της κάθε λέξης. Για την αναπαράστασή της κάθε λέξης χρησιμοποιούμε την εξής προσέγγιση: κάθε γραμμή του αρχείου έχει ως πρώτη λέξη το κλειδί και μετά οι υπόλοιπες λέξεις αναπαριστούν τη λέξη. Τελικά προκύπτει ένα dictionary που περιέχει όλες τις λέξεις σαν κλειδιά και τις αντίστοιχες αναπαραστάσεις σαν τιμές των αντίστοιχων κλειδιών.

Για να δημιουργήσουμε την αναπαράσταση του κάθε review αξιοποιώντας το ανάλογο dictionary η προσέγγιση είναι η εξής:

1. Αναπαριστάμε το κάθε review ως μια λίστα από strings, πιο συγκεκριμένα tokens.

2. Για κάθε μια λέξη από τη λίστα λέξεων ενός review, βρίσκουμε το vector που την αναπαριστά μέσα στο dictionary.
3. Αθροίζουμε τις λίστες του κάθε review και τελικά υπολογίζουμε ένα μέσο διάνυσμα το οποίο αναπαριστά το κάθε review.
4. Τελικά έχουμε για κάθε review μια αναπαράσταση του ως ένα διάνυσμα fixed μεγέθους (by default 200) το οποίο θα μας βοηθήσει στην μετέπειτα υλοποίηση μας.

Για το TF-IDF Vectorizer, επέλεξα να χρησιμοποιήσω τις εξής παραμέτρους:

1. `ngram_range = (1,3)`
2. `max_features = 300`

Κάνουμε fit τον vectorizer μας χρησιμοποιώντας το training set μας και ύστερα κάνουμε transform το testing set μας.

Για τα μοντέλα μου έχω ορίσει τη κλάση Net η οποία αναπαριστά το νευρωνικό μας δίκτυο.

Η συγκεκριμένη κλάση με την αρχικοποίηση της ορίζει τις βασικές παραμέτρους του δικτύου μας, όπως τον αριθμό των layers του δικτύου καθώς και τις διαστάσεις του κάθε layer, τα activation functions του κάθε layer, τις τεχνικές regularization που θα εφαρμοστούν μεταξύ των layers.

Για κάθε μοντέλο μαζί με την αρχικοποίηση του νευρωνικού μας δικτύου ορίζουμε το loss function και τον optimizer που θα χρησιμοποιηθεί μετέπειτα.

Στη συνέχεια, μετατρέπουμε τα δεδομένα μας σε tensors, δημιουργούμε ένα ενιαίο tensor dataset τόσο για το training όσο και για το testing set και τέλος δημιουργούμε τους data loaders μας.

Υστερα, εκπαιδεύουμε το μοντέλο μας αξιοποιώντας τη συνάρτηση train_model δίνοντας σαν όρισμα τον επιθυμητό αριθμό epochs. Για κάθε epoch:

1. Καθαρίζουμε τα αποθηκευμένα gradients από προηγούμενα epochs.
2. Χρησιμοποιούμε το μοντέλο μας με το τρέχον batch.
3. Υπολογίζουμε το loss function.
4. Εφαρμόζουμε backpropagation αξιοποιώντας το loss που υπολογίσαμε.
5. Ανανεώνουμε τα weighs του μοντέλου μας με βάση τα gradients που προέκυψαν από το backpropagation.
6. Αποθηκεύουμε τα predictions ώστε να αξιολογήσουμε στη συνέχεια το μοντέλο μας.

Υστερα, αξιολογούμε το μοντέλο μας, και για κάθε batch του validation set:

1. Χρησιμοποιούμε το μοντέλο μας για το τρέχον batch.

2. Υπολογίζουμε το *loss function*.
3. Αποθηκεύουμε ξανά τα *prediction* του *validation set*.

Τέλος, υπολογίζουμε και αξιολογούμε το μοντέλο μας χρησιμοποιώντας τις εξής μετρικές:

1. *Train loss*
2. *Validation loss*
3. *Train F1-Score*
4. *Validation F1-Score*
5. *Validation Accuracy*

Για το πρώτο μας μοντέλο *Word Embeddings / Feed-forward NN*

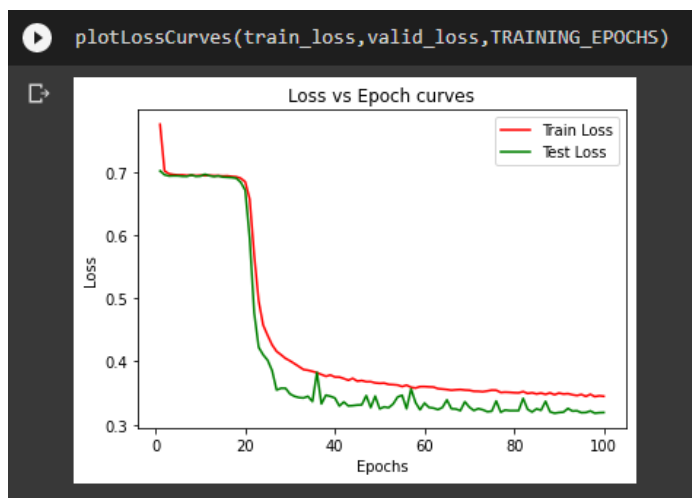
Οι παράμετροι που χρησιμοποιούμε είναι οι εξής:

1. *Loss-function = CrossEntropyLoss*
2. *Optimizer = SGD*
3. *Learning_rate = 0.0025*
4. *Training_epochs = 100*
5. *Layer1 size = 64*
6. *Layer2 size = 32*
7. *Layer3 size = 16*
8. *Output_layer size = 3*

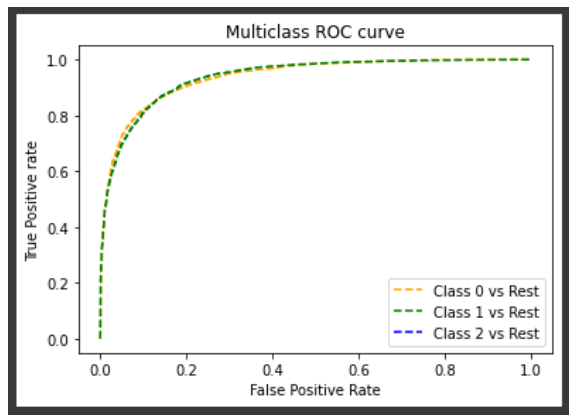
Κατά τη διαδικασία του *training* έχουμε το εξής *output*:

Epoch	0:	Train Loss = 0.77554	Validation Loss = 0.70182	Accuracy = 49.7223	Train-f1 = 0.4846	Valid-F1 = 0.3304
Epoch	1:	Train Loss = 0.70177	Validation Loss = 0.69564	Accuracy = 49.7112	Train-f1 = 0.5007	Valid-F1 = 0.3301
Epoch	2:	Train Loss = 0.69721	Validation Loss = 0.69379	Accuracy = 50.2888	Train-f1 = 0.5002	Valid-F1 = 0.3365
Epoch	3:	Train Loss = 0.69599	Validation Loss = 0.69408	Accuracy = 49.7112	Train-f1 = 0.4982	Valid-F1 = 0.3301
Epoch	4:	Train Loss = 0.69526	Validation Loss = 0.69403	Accuracy = 49.7112	Train-f1 = 0.5035	Valid-F1 = 0.3301
Epoch	5:	Train Loss = 0.69532	Validation Loss = 0.69330	Accuracy = 50.2888	Train-f1 = 0.4940	Valid-F1 = 0.3365
Epoch	6:	Train Loss = 0.69460	Validation Loss = 0.69321	Accuracy = 49.7112	Train-f1 = 0.5007	Valid-F1 = 0.3301
Epoch	7:	Train Loss = 0.69480	Validation Loss = 0.69524	Accuracy = 49.7112	Train-f1 = 0.4997	Valid-F1 = 0.3301
Epoch	8:	Train Loss = 0.69426	Validation Loss = 0.69327	Accuracy = 50.2888	Train-f1 = 0.5048	Valid-F1 = 0.3365
Epoch	9:	Train Loss = 0.69474	Validation Loss = 0.69357	Accuracy = 50.2888	Train-f1 = 0.4964	Valid-F1 = 0.3365
Epoch	10:	Train Loss = 0.69433	Validation Loss = 0.69639	Accuracy = 50.2888	Train-f1 = 0.5028	Valid-F1 = 0.3365
Epoch	11:	Train Loss = 0.69456	Validation Loss = 0.69422	Accuracy = 49.7112	Train-f1 = 0.4962	Valid-F1 = 0.3301
Epoch	12:	Train Loss = 0.69422	Validation Loss = 0.69289	Accuracy = 50.2888	Train-f1 = 0.5056	Valid-F1 = 0.3365
Epoch	13:	Train Loss = 0.69419	Validation Loss = 0.69389	Accuracy = 50.2888	Train-f1 = 0.5056	Valid-F1 = 0.3365
Epoch	14:	Train Loss = 0.69373	Validation Loss = 0.69216	Accuracy = 50.2888	Train-f1 = 0.5080	Valid-F1 = 0.3365
Epoch	15:	Train Loss = 0.69393	Validation Loss = 0.69172	Accuracy = 50.2888	Train-f1 = 0.5049	Valid-F1 = 0.3365
Epoch	16:	Train Loss = 0.69300	Validation Loss = 0.69125	Accuracy = 49.7112	Train-f1 = 0.5148	Valid-F1 = 0.3301
Epoch	17:	Train Loss = 0.69243	Validation Loss = 0.69022	Accuracy = 50.2888	Train-f1 = 0.5183	Valid-F1 = 0.3365
Epoch	18:	Train Loss = 0.69017	Validation Loss = 0.68387	Accuracy = 68.1293	Train-f1 = 0.5307	Valid-F1 = 0.6586
Epoch	19:	Train Loss = 0.68458	Validation Loss = 0.67127	Accuracy = 60.2644	Train-f1 = 0.5582	Valid-F1 = 0.5313
Epoch	20:	Train Loss = 0.65718	Validation Loss = 0.59482	Accuracy = 76.2053	Train-f1 = 0.6260	Valid-F1 = 0.7608
Epoch	21:	Train Loss = 0.56910	Validation Loss = 0.47725	Accuracy = 79.2935	Train-f1 = 0.7231	Valid-F1 = 0.7921
Epoch	22:	Train Loss = 0.49595	Validation Loss = 0.42249	Accuracy = 81.6374	Train-f1 = 0.7721	Valid-F1 = 0.8156
Epoch	23:	Train Loss = 0.45766	Validation Loss = 0.41061	Accuracy = 81.5708	Train-f1 = 0.7968	Valid-F1 = 0.8135
Epoch	24:	Train Loss = 0.44133	Validation Loss = 0.40201	Accuracy = 82.0595	Train-f1 = 0.8069	Valid-F1 = 0.8187
Epoch	25:	Train Loss = 0.42628	Validation Loss = 0.38553	Accuracy = 83.2926	Train-f1 = 0.8152	Valid-F1 = 0.8319
Epoch	26:	Train Loss = 0.41597	Validation Loss = 0.35483	Accuracy = 84.7256	Train-f1 = 0.8217	Valid-F1 = 0.8473
Epoch	27:	Train Loss = 0.41053	Validation Loss = 0.35761	Accuracy = 84.8811	Train-f1 = 0.8256	Valid-F1 = 0.8486
Epoch	28:	Train Loss = 0.40478	Validation Loss = 0.35753	Accuracy = 84.6256	Train-f1 = 0.8296	Valid-F1 = 0.8458
Epoch	29:	Train Loss = 0.40096	Validation Loss = 0.34847	Accuracy = 85.1589	Train-f1 = 0.8282	Valid-F1 = 0.8514
Epoch	30:	Train Loss = 0.39646	Validation Loss = 0.34479	Accuracy = 85.3699	Train-f1 = 0.8323	Valid-F1 = 0.8536
Epoch	31:	Train Loss = 0.39185	Validation Loss = 0.34278	Accuracy = 85.4255	Train-f1 = 0.8341	Valid-F1 = 0.8543
Epoch	32:	Train Loss = 0.38719	Validation Loss = 0.34221	Accuracy = 85.2477	Train-f1 = 0.8347	Valid-F1 = 0.8522
Epoch	33:	Train Loss = 0.38606	Validation Loss = 0.34472	Accuracy = 84.9034	Train-f1 = 0.8357	Valid-F1 = 0.8488
Epoch	34:	Train Loss = 0.38428	Validation Loss = 0.33652	Accuracy = 85.4699	Train-f1 = 0.8373	Valid-F1 = 0.8547
Epoch	35:	Train Loss = 0.38215	Validation Loss = 0.38272	Accuracy = 83.2482	Train-f1 = 0.8387	Valid-F1 = 0.8309
Epoch	36:	Train Loss = 0.37933	Validation Loss = 0.33279	Accuracy = 85.7920	Train-f1 = 0.8397	Valid-F1 = 0.8579
Epoch	37:	Train Loss = 0.37649	Validation Loss = 0.34665	Accuracy = 84.4923	Train-f1 = 0.8420	Valid-F1 = 0.8440
Epoch	38:	Train Loss = 0.37847	Validation Loss = 0.34483	Accuracy = 85.5365	Train-f1 = 0.8402	Valid-F1 = 0.8553
Epoch	39:	Train Loss = 0.37541	Validation Loss = 0.34215	Accuracy = 85.0033	Train-f1 = 0.8402	Valid-F1 = 0.8497
Epoch	40:	Train Loss = 0.37526	Validation Loss = 0.32947	Accuracy = 85.8032	Train-f1 = 0.8427	Valid-F1 = 0.8580
Epoch	41:	Train Loss = 0.37286	Validation Loss = 0.33578	Accuracy = 85.6032	Train-f1 = 0.8432	Valid-F1 = 0.8559

και παρατηρούμε πως τα *loss curves* των *training* και *testing set* είναι τα εξής:



Τα *ROC curves* είναι τα εξής:



Και τέλος τα αποτελέσματα των μετρικών είναι τα εξής:

```

Accuracy: 86.04%
f1 score: 86.03%
Precision: 86.06%
Recall: 86.04%

```

	precision	recall	f1-score	support
0.0	0.85	0.87	0.86	4527
1.0	0.87	0.85	0.86	4475
accuracy			0.86	9002
macro avg	0.86	0.86	0.86	9002
weighted avg	0.86	0.86	0.86	9002

Συμπεραίνουμε ότι το μοντέλο μας δεν κάνει ούτε *overfitting* ούτε *underfitting*. Η χρήση του *word embeddings* αύξησε αρκετά την απόδοση του μοντέλου μας και χρησιμοποιώντας ένα απλό μοντέλο με προσαρμοσμένες κατάλληλα παραμέτρους πετύχαμε ένα καλό αποτέλεσμα.

Για το δεύτερο μοντέλο μας TF-IDF / Feed-forward NN:

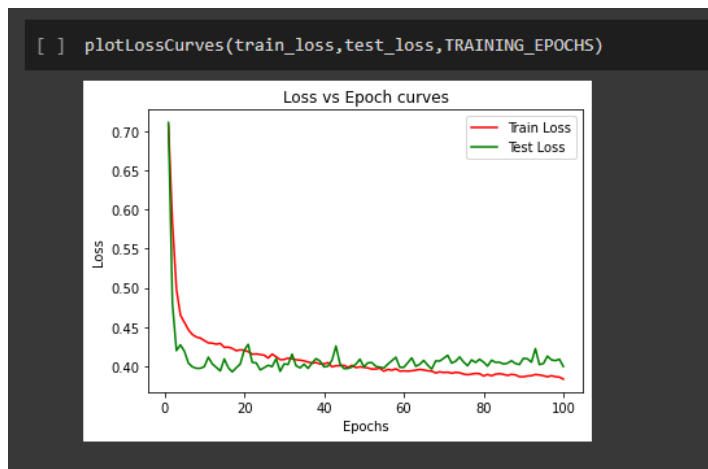
Οι παράμετροι που χρησιμοποιούμε είναι οι εξής:

1. *Loss-function* = *CrossEntropyLoss*
2. *Optimizer* = *SGD*
3. *Learning_rate* = 0.0025
4. *Training_epochs* = 100
5. *Layer1 size* = 64
6. *Layer2 size* = 32
7. *Layer3 size* = 16
8. *Output_layer size* = 3

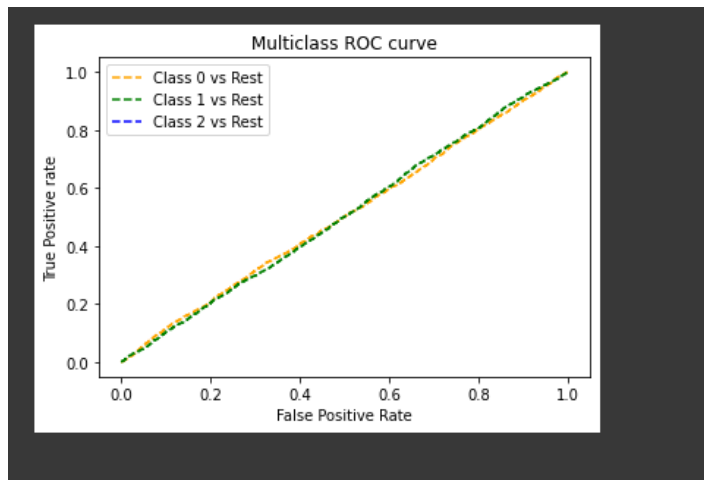
Κατά τη διαδικασία του *training* έχουμε το εξής *output*:

Epoch	0:	Train Loss = 0.70704	Validation Loss = 0.71080	Accuracy = 49.5668	Train-f1 = 0.5074	Valid-F1 = 0.3285
Epoch	1:	Train Loss = 0.58234	Validation Loss = 0.47967	Accuracy = 77.5272	Train-f1 = 0.6012	Valid-F1 = 0.7701
Epoch	2:	Train Loss = 0.49863	Validation Loss = 0.41985	Accuracy = 80.9931	Train-f1 = 0.7678	Valid-F1 = 0.8094
Epoch	3:	Train Loss = 0.46512	Validation Loss = 0.42711	Accuracy = 80.6487	Train-f1 = 0.7889	Valid-F1 = 0.8055
Epoch	4:	Train Loss = 0.45588	Validation Loss = 0.41911	Accuracy = 80.6154	Train-f1 = 0.7960	Valid-F1 = 0.8055
Epoch	5:	Train Loss = 0.44639	Validation Loss = 0.40393	Accuracy = 81.6485	Train-f1 = 0.7977	Valid-F1 = 0.8164
Epoch	6:	Train Loss = 0.44037	Validation Loss = 0.39916	Accuracy = 81.7929	Train-f1 = 0.8023	Valid-F1 = 0.8179
Epoch	7:	Train Loss = 0.43711	Validation Loss = 0.39727	Accuracy = 81.6596	Train-f1 = 0.8028	Valid-F1 = 0.8164
Epoch	8:	Train Loss = 0.43580	Validation Loss = 0.39723	Accuracy = 81.3708	Train-f1 = 0.8037	Valid-F1 = 0.8133
Epoch	9:	Train Loss = 0.43266	Validation Loss = 0.39945	Accuracy = 81.8152	Train-f1 = 0.8062	Valid-F1 = 0.8181
Epoch	10:	Train Loss = 0.42965	Validation Loss = 0.41139	Accuracy = 81.9040	Train-f1 = 0.8091	Valid-F1 = 0.8189
Epoch	11:	Train Loss = 0.42933	Validation Loss = 0.40285	Accuracy = 81.3153	Train-f1 = 0.8066	Valid-F1 = 0.8123
Epoch	12:	Train Loss = 0.42800	Validation Loss = 0.39842	Accuracy = 81.9485	Train-f1 = 0.8084	Valid-F1 = 0.8194
Epoch	13:	Train Loss = 0.42900	Validation Loss = 0.39395	Accuracy = 82.0818	Train-f1 = 0.8075	Valid-F1 = 0.8206
Epoch	14:	Train Loss = 0.42401	Validation Loss = 0.40920	Accuracy = 81.5041	Train-f1 = 0.8111	Valid-F1 = 0.8150
Epoch	15:	Train Loss = 0.42426	Validation Loss = 0.39807	Accuracy = 81.6374	Train-f1 = 0.8116	Valid-F1 = 0.8162
Epoch	16:	Train Loss = 0.42261	Validation Loss = 0.39269	Accuracy = 81.9040	Train-f1 = 0.8115	Valid-F1 = 0.8190
Epoch	17:	Train Loss = 0.41971	Validation Loss = 0.39795	Accuracy = 81.2597	Train-f1 = 0.8126	Valid-F1 = 0.8115
Epoch	18:	Train Loss = 0.42080	Validation Loss = 0.40265	Accuracy = 81.8485	Train-f1 = 0.8112	Valid-F1 = 0.8183
Epoch	19:	Train Loss = 0.42035	Validation Loss = 0.42039	Accuracy = 81.6818	Train-f1 = 0.8137	Valid-F1 = 0.8168
Epoch	20:	Train Loss = 0.41844	Validation Loss = 0.42782	Accuracy = 81.4930	Train-f1 = 0.8127	Valid-F1 = 0.8147
Epoch	21:	Train Loss = 0.41525	Validation Loss = 0.40487	Accuracy = 81.2507	Train-f1 = 0.8140	Valid-F1 = 0.8122
Epoch	22:	Train Loss = 0.41566	Validation Loss = 0.40414	Accuracy = 81.6374	Train-f1 = 0.8164	Valid-F1 = 0.8164
Epoch	23:	Train Loss = 0.41478	Validation Loss = 0.39519	Accuracy = 82.0040	Train-f1 = 0.8152	Valid-F1 = 0.8199
Epoch	24:	Train Loss = 0.41400	Validation Loss = 0.39808	Accuracy = 81.3597	Train-f1 = 0.8164	Valid-F1 = 0.8134
Epoch	25:	Train Loss = 0.41031	Validation Loss = 0.40109	Accuracy = 81.8707	Train-f1 = 0.8193	Valid-F1 = 0.8187
Epoch	26:	Train Loss = 0.41535	Validation Loss = 0.39940	Accuracy = 82.0040	Train-f1 = 0.8167	Valid-F1 = 0.8199
Epoch	27:	Train Loss = 0.41196	Validation Loss = 0.40964	Accuracy = 81.4152	Train-f1 = 0.8164	Valid-F1 = 0.8129
Epoch	28:	Train Loss = 0.40824	Validation Loss = 0.39333	Accuracy = 82.1706	Train-f1 = 0.8199	Valid-F1 = 0.8217
Epoch	29:	Train Loss = 0.40849	Validation Loss = 0.40290	Accuracy = 81.1264	Train-f1 = 0.8193	Valid-F1 = 0.8110
Epoch	30:	Train Loss = 0.41024	Validation Loss = 0.40210	Accuracy = 81.5152	Train-f1 = 0.8193	Valid-F1 = 0.8148
Epoch	31:	Train Loss = 0.40916	Validation Loss = 0.41526	Accuracy = 80.9376	Train-f1 = 0.8191	Valid-F1 = 0.8081
Epoch	32:	Train Loss = 0.40827	Validation Loss = 0.40043	Accuracy = 81.5152	Train-f1 = 0.8209	Valid-F1 = 0.8150
Epoch	33:	Train Loss = 0.40770	Validation Loss = 0.39791	Accuracy = 81.6374	Train-f1 = 0.8208	Valid-F1 = 0.8164
Epoch	34:	Train Loss = 0.40668	Validation Loss = 0.40247	Accuracy = 81.9151	Train-f1 = 0.8201	Valid-F1 = 0.8188
Epoch	35:	Train Loss = 0.40535	Validation Loss = 0.39712	Accuracy = 81.6485	Train-f1 = 0.8194	Valid-F1 = 0.8165
Epoch	36:	Train Loss = 0.40385	Validation Loss = 0.40402	Accuracy = 80.8820	Train-f1 = 0.8230	Valid-F1 = 0.8083
Epoch	37:	Train Loss = 0.40454	Validation Loss = 0.40959	Accuracy = 80.9376	Train-f1 = 0.8216	Valid-F1 = 0.8088
Epoch	38:	Train Loss = 0.40281	Validation Loss = 0.40654	Accuracy = 81.6818	Train-f1 = 0.8191	Valid-F1 = 0.8168
Epoch	39:	Train Loss = 0.40288	Validation Loss = 0.39902	Accuracy = 81.6152	Train-f1 = 0.8222	Valid-F1 = 0.8161
Epoch	40:	Train Loss = 0.40448	Validation Loss = 0.40005	Accuracy = 81.9707	Train-f1 = 0.8212	Valid-F1 = 0.8195
Epoch	41:	Train Loss = 0.39907	Validation Loss = 0.40754	Accuracy = 80.5710	Train-f1 = 0.8243	Valid-F1 = 0.8040
Epoch	42:	Train Loss = 0.40050	Validation Loss = 0.42557	Accuracy = 79.6934	Train-f1 = 0.8257	Valid-F1 = 0.7951

και παρατηρούμε ότι τα *loss curves* των training και testing set είναι τα εξής:



Τα ROC curves είναι τα εξής:



και τέλος τα αποτελέσματα των μετρικών είναι τα εξής:

```

Accuracy: 50.01%
f1 score: 50.00%
Precision: 50.00%
Recall: 50.01%

```

	precision	recall	f1-score	support
0.0	0.50	0.52	0.51	4540
1.0	0.50	0.48	0.49	4462
accuracy			0.50	9002
macro avg	0.50	0.50	0.50	9002
weighted avg	0.50	0.50	0.50	9002

Τα μοντέλο μας δεν παρουσιάζει φαινόμενα *overfitting/underfitting*. Η χρήση ενός απλού δικτύου, όπως και στο προηγούμενο μοντέλο μας μας βοήθησε να πετύχουμε ένα καλό αποτέλεσμα και σε συνδυασμό με τη χρήση του *TF-IDF Vectorizer* η απόδοση μας ήταν αρκετά καλή.

Γενικό συμπέρασμα:

Τα αποτελέσματα που προκύπτουν από το πρώτο μοντέλο είναι καλύτερα σε σχέση με τα αποτελέσματα του δεύτερου μοντέλου. Με λίγα λόγια δηλαδή χρησιμοποιώντας και στις 2 περιπτώσεις ένα αρκετά απλό μοντέλο, πετύχαμε καλύτερη ακρίβεια στη πρώτη περίπτωση με τη χρήση δηλαδή των *word embeddings* καθώς γίνεται καλύτερη αξιοποίηση των απλουστευμένων *reviews* από ότι στο δεύτερο μοντέλο που χρησιμοποιούμε έναν *TF-IDF Vectorizer*. Ωστόσο, και τα 2 μοντέλα μας αποδείχθηκαν εξίσου αποδοτικά και σε συνδυασμό με τον επαρκή καθαρισμό των δεδομένων μας, πήραμε και στις 2 περιπτώσεις παρόμοια *loss* και *ROC curves*.