ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ-ΙΙ
ΕΡΓΑΣΙΑ 1
2020-21

Ονοματεπώνυμο: Βάιος Λύτρας

Α.Μ: 1115201800101

ΑΣΚΗΣΗ 1

$MSE(w) = \frac{1}{m}\sum_{i=1}^{m}(h_w(x^{(i)}) - y^{(i)})^2 = \frac{1}{m}\sum_{i=1}^{m}(w \cdot x^{(i)} - y^{(i)})^2$

Θέτουμε τώρα ότι: $k(w) = w \cdot x^{(i)} - y^{(i)}$

Άρα: $MSE(w) = \frac{1}{m}\sum_{i=1}^{m} k_i(w)^2 = \frac{1}{m}k^T k$ αφού

$$\begin{bmatrix} k_1 & k_2 & \dots & k_n \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \dots \\ k_n \end{bmatrix} = \sum_{i=1} k_i{}^2$$

Οπότε $MSE(w) = \frac{1}{m}k^T k$

$= \frac{1}{m}(wx - y)^T(wx - y)$

$= \frac{1}{m}(w^T x^T - y^T)(wx - y)$

$= \frac{1}{m}(w^T x^T wx - w^T x^T y - y^T wx + y^T y)$

Ξέρουμε ότι: $(y^T xw)^T = w^T x^T y$ και επίσης μπορούμε να παρατηρήσουμε ότι αυτός είναι $1X1$ πίνακας οπότε $y^T xw = w^T x^T y$.

Άρα $MSE(w) = \frac{1}{m}(w^T x^T wx - 2w^T x^T y + y^T y)$

$\nabla_w MSE(w) = \frac{1}{m}(\nabla_w w^T x^T wx - 2\nabla_w w^T x^T y + \nabla_w y^T y)$

$= \frac{1}{n}(2x^T xw - 2x^T y + 0)$

$= \frac{w}{n}(x^T xw - x^T y)$

$= \frac{2}{n}(x^T(xw - y))$

*Σημείωση: παραπάνω χρησιμοποιήθηκε και ο μαθηματικός κανόνας του ανάστροφου ότι $x^T x = x^2$

ΑΣΚΗΣΗ 2 (ΕΠΕΞΗΓΗΣΕΙΣ)

ΤΡΕΞΙΜΟ ΕΦΑΡΜΟΓΗΣ

Η τελική εφαρμογή είναι ολόκληρη μέσα σε ένα μεγάλο κελί κώδικα στο τέλος του notebook και μπορείτε να αλλάξετε το test set αλλάζοντας το path στην μεταβλητή 'testset' κάτω από τα imports. Παρακαλώ να κοιτάξετε καλά το path γιατί όπως το έχω τώρα χρειάζεται ένας φάκελος με όνομα 'data' που εκεί μέσα έβαζα εγώ τα datasets. Επίσης ολόκληρος ο κώδικας θα κάνει ώρα να τρέξει πιο πολύ λόγω του σχεδιασμού των learning curves (περίπου 1 με 2 λεπτά). Επίσης πάνω από τις καμπύλες στην έξοδο θα εκτυπωθούν και τα recall, precision και f1 scores.

ΕΠΕΞΗΓΗΣΗΤΟΥ ΥΠΟΛΟΙΠΟΥ ΚΩΔΙΚΑ

Πάνω από το τελευταίο και τεράστιο κελί κώδικα που είναι και ο τελικός υπάρχουν μικρότερα κελιά με ξηγήσεις τα οποία χρησιμοποιούσα για να δοκιμάσω αυτά που θα εξηγήσω παρακάτω. Δηλαδή έχω 1 κελί για κάθε vectorizer για να επιλέγω όποιον θέλω κάθε φορά και ένα κελί που είναι το training έτσι ώστε να πειράζω τις μεταβλητές και να τρέχω μόνο αυτό, ένα κελί που εκτυπώνει τα scores κλπ. Τρέχοντας τα κατάλληλα κελιά κώδικα και αλλάζοντας λίγο τον κώδικα σε κάποια σημεία που εξηγείται με σχόλια μπορείτε να δείτε τα νούμερα (scores) και τις καμπύλες που θα εξηγήσω παρακάτω για κάθε δοκιμή και πως έφτασα στο τελικό αποτέλεσμα.

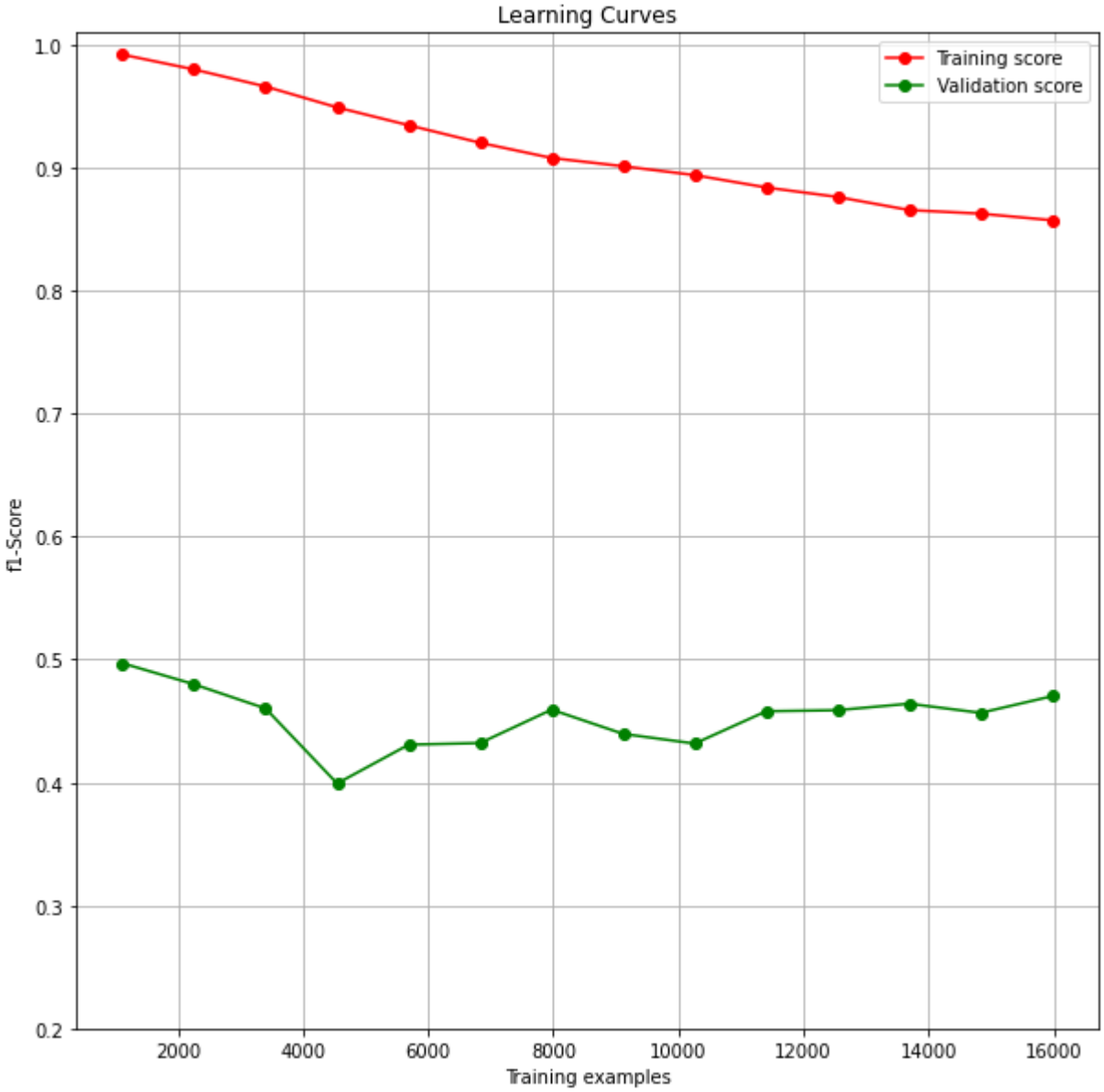
ΑΝΑΛΥΤΙΚΗ ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΚΑΘΑΡΙΣΜΟΥ ΔΕΔΟΜΕΝΩΝ ΠΟΥ ΑΝΑΦΕΡΕΤΑΙ ΠΑΡΑΚΑΤΩ

Ως καθαρισμό των δεδομένων μου επέλεξα να βάλω απλό καθαρισμό από stopwords χρησιμοποιώντας την βιβλιοθήκη NLTK και αφαίρεση των συνδέσμων (links) από τα tweets γιατί δεν αποσκοπούν πουθενά και εκτός του ότι βρωμίζουν το dataset κάνουν και το μοντέλο πιο αργό επειδή έχει να επεξεργαστεί extra αχρείαστα tokens. Εκτός από αφαίρεση των links αφαιρούνται από το dataset τα σύμβολα '#' , '@' , '.' , ',' έτσι ώστε να

έχουμε μόνο καθαρές λέξεις. Επίσης δοκίμασα να βγάλω όλα τα σημεία στίξης από το κείμενο ή να κάνω stemming αλλά δεν το κράτησα στο τελικό μοντέλο γιατί έριχναν το score (εξηγείται παρακάτω).

ΠΡΟΟΔΟΣ ΤΗΣ ΕΡΓΑΣΙΑΣ ΜΕ BULLETS

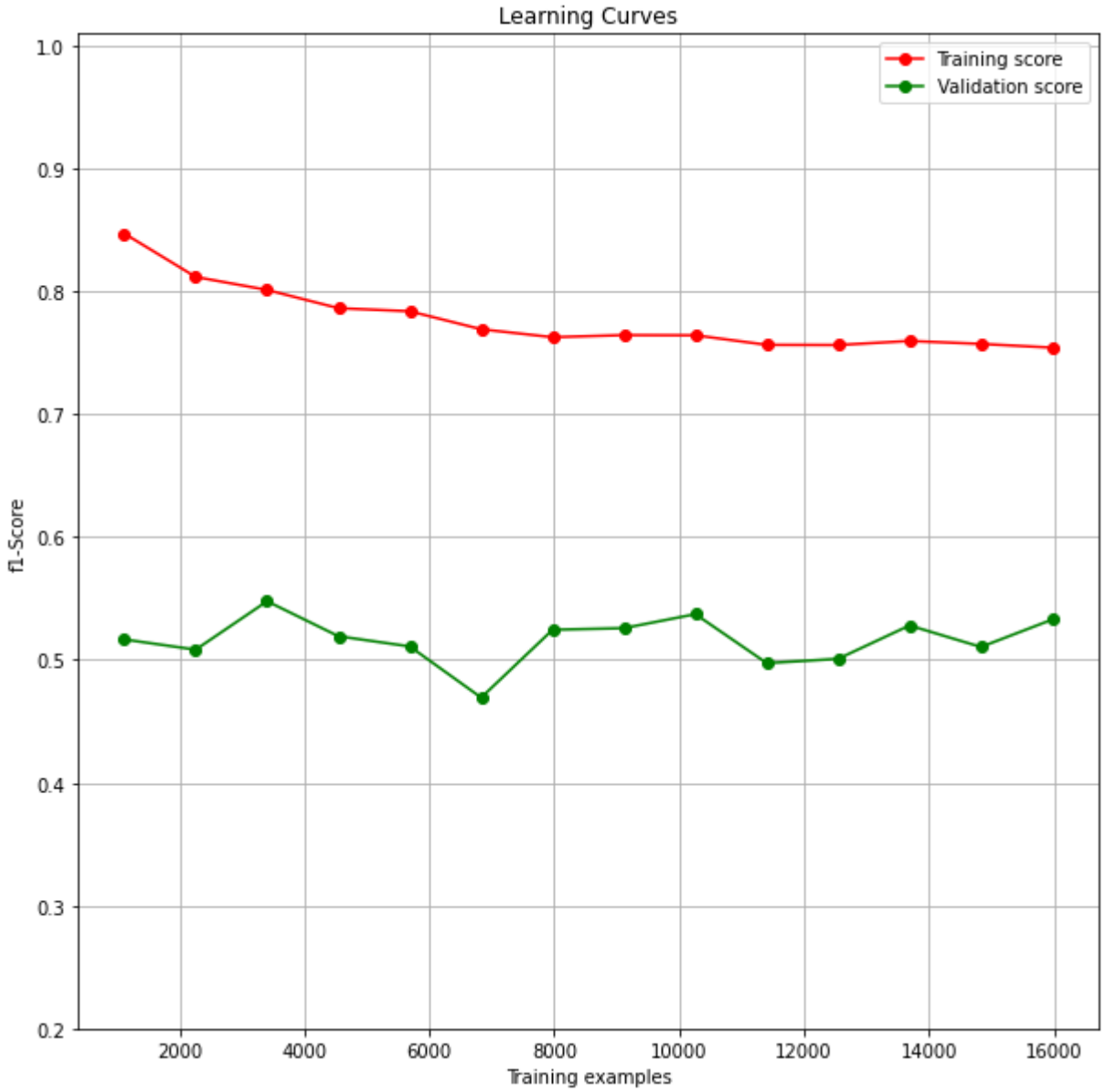
- Αρχικά ξεκίνησα γράφοντας τον βασικό κώδικα για το διάβασμα αρχείων, φτιάξιμο καμπύλων κλπ. και χρησιμοποίησα count-vectorizer χωρίς καθόλου data-cleaning για να δω αν λειτουργούν όλα όπως πρέπει. Αυτό έπιανε score κοντά στο 47% το οποίο είναι πολύ χαμηλό οπότε δεν του έδωσα παραπάνω σημασία. Με τις πρώτες δοκιμές όμως έμαθα ότι το training set βγάζει περίπου 7000-8000 features by default στους vectorizer.
- Η επόμενη βελτίωση αφού διάβασα για την συνάρτηση εκμάθησης στο (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html) ήταν να βάλω το όρισμα **class_weight = 'balanced'** στην logistic_regression επειδή σκέφτηκα ότι θα κάνει πιο ομαλό distribution στα weights και θα βοηθήσει αλλά έριξε το score οπότε την έβγαλα τελείως (δοκιμάστηκε ξανά αργότερα με τους άλλους vectorizers αλλά πάλι έριχνε το score).
- Πρόσθεσα καθαρισμό δεδομένων με stopwords από την βιβλιοθήκη NLTK και το score στο testset ανέβηκε στο 48% και δεν κατάφερα να το ανεβάσω άλλο ούτε να διορθωθεί τελείως το τρομερό overfit που έχει, αν και με κάποιες ρυθμίσεις διορθώνεται κάπως αλλά όχι τελείως και το score του παραμένει πολύ κακό. Οι τελικές καμπύλες του είναι:



Ανεβάζοντας ή κατεβάζοντας τον αριθμό των features στον count_vectorizer έπεφτε το score οπότε ο ιδανικός αριθμός είναι κοντά στο 1000.

- Η επόμενη δοκιμή ήταν με hashing_vectorizer στις default ρυθμίσεις και χωρίς καθαρισμό δεδομένων έπιασε σκορ 65%. Κάτι το οποίο χωρίς έξτρα καθαρισμό κλπ είναι ένα πολύ καλό νούμερο σε σύγκριση με τον count.
- Επόμενη κίνηση μου ήταν να ανεβάσω τον αριθμό των features κατά 2000 την φορά και το score αυξάνονταν επίσης με αποτέλεσμα να πιάνει το 67% στα 8000 features.
- Αυτό που έμενε ήταν να βάλω καθαρισμό δεδομένων και stopwords που έκανε το score 69.8%.
- Η επόμενη προσθήκη ήταν το όρισμα **strip_accents = 'unicode'** στον vectorizer μου το οποίο βοηθάει στην κανονικοποίηση των δεδομένων και αύξησε το score του μοντέλου στο 71.07%.

- Προφανώς το τελευταίο βήμα ήταν να δοκιμάσω τον TF-IDF vectorizer ο οποίος με την πρώτη δοκιμή που έκανα ήταν μαζί με καθαρισμό των δεδομένων έπιασε score κάτω από 50% και την παρακάτω πολύ κακή καμπύλη:

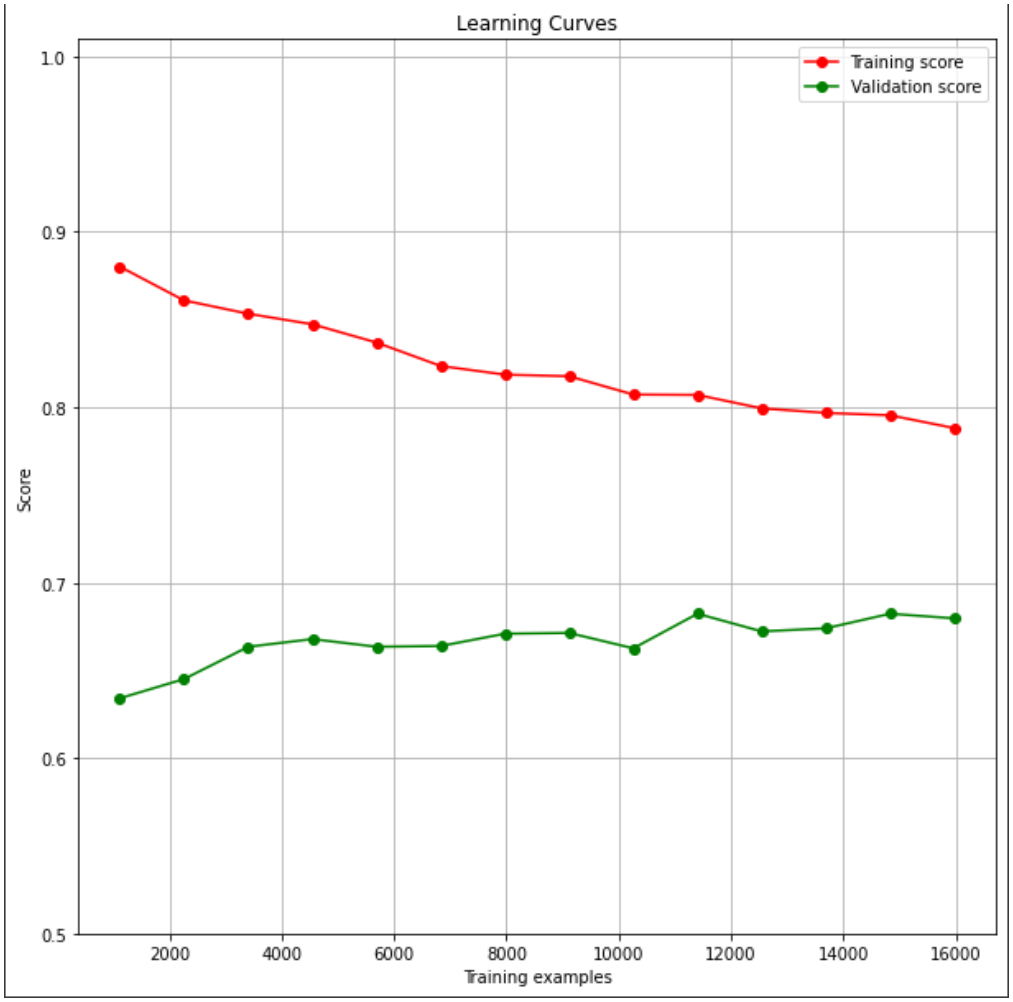


Δοκίμασα να του ανεβάσω ή να του κατεβάσω τα features και το score παρέμενε απίστευτα κακό οπότε δεν ασχολήθηκα άλλο με αυτόν τον vectorizer. Από εδώ και πέρα χρησιμοποιούσα τον TF-IDF και τον count μόνο όταν δοκίμαζα κάτι στον hashing (που είχα καταλήξει ότι είναι ο καλύτερος) και του έριχνε την απόδοση. Αυτό το έκανα γιατί αν έριχνε την απόδοση και στους άλλους 2 τότε καταλάβαινα ότι αυτό που κάνω γενικά δεν βοηθάει το μοντέλο μου.

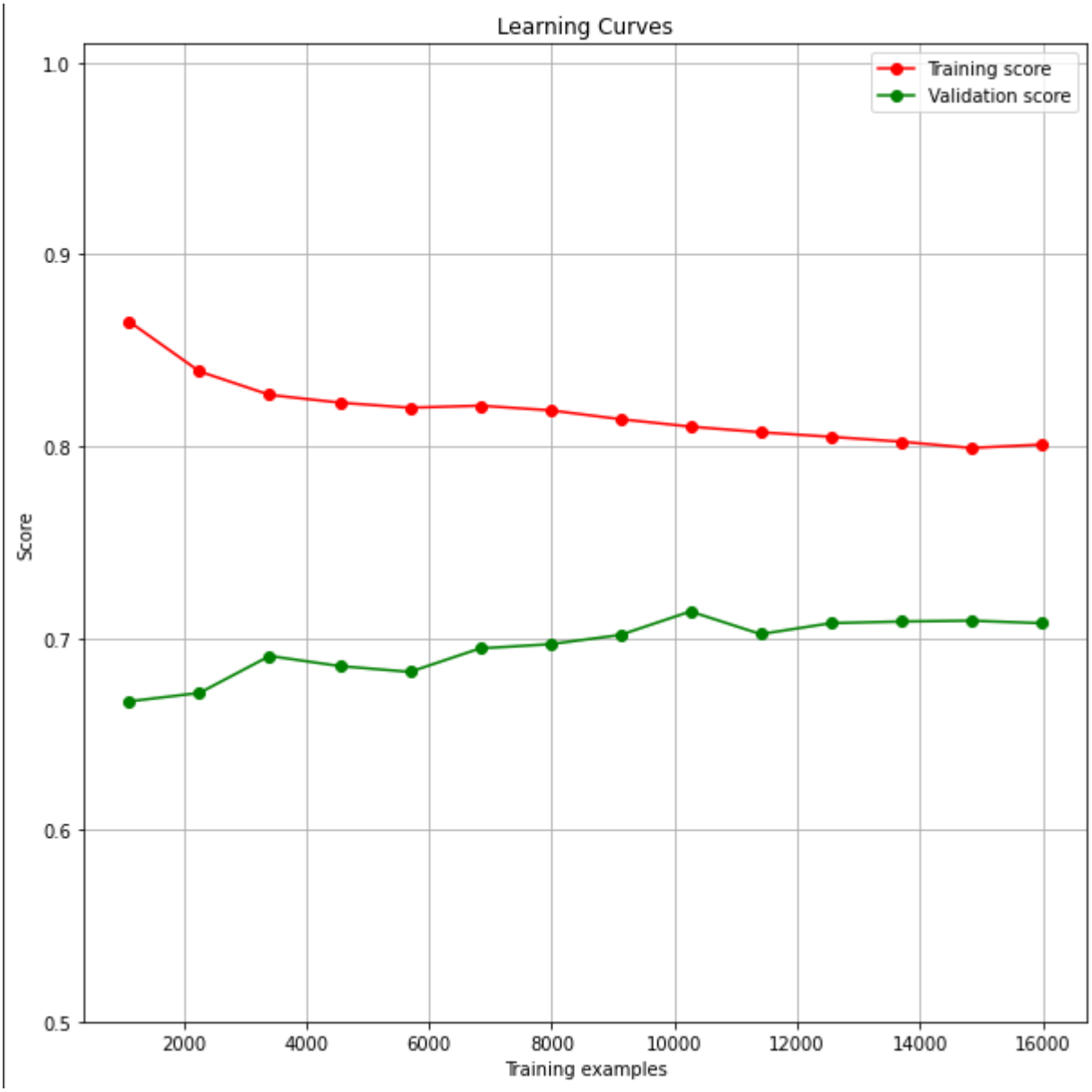
- Από εδώ και κάτω δοκίμασα οτιδήποτε είδαμε στο φροντιστήριο και ότι διάβασα στο documentation των συναρτήσεων για να ανεβάσω το 71.07% αλλά δεν τα κατάφερα. Ξεκίνησα κάνοντας έξτρα καθαρισμό δεδομένων με stemming το οποίο έριξε πάρα πολύ το score (κατά περίπου 3-5%) το οποίο μου έκανε εντύπωση αλλά αφού διάβασα λίγα παραδείγματα προτάσεων που γινόντουσαν stem από το google

τότε κατάλαβα ότι το stemming ίσως δεν είναι και τόσο βοηθητικό όταν θες να κάνεις classify 2 πράγματα που είναι εντελώς αντίθετα

- Επίσης δοκίμασα lemmatization το οποίο είναι πιο αυστηρό από το stemming οπότε εξίσου δεν βοήθησε καθόλου
- Μετά δοκίμασα να αφαιρέσω από το κείμενο όλα τα σημεία στίξης χρησιμοποιώντας RegexTokenizer() (από την NLTK) στον καθαρισμό των δεδομένων αλλά μου έριξε το score κατά 3%. Δοκιμάστηκε επίσης μαζί με stemming και επίσης το score έπεσε.
- Επόμενη δοκιμή ήταν η αφαίρεση των αριθμών από το κείμενο επειδή κοίταξα την βάση και είδα κάποιους αριθμούς σε κάποια tweets το οποίο επίσης δεν βοήθησε και το score έμεινε σταθερό.
- Διαφορετικός solver στην logistic_regression (επειδή ο default είναι ο lbfgs) δεν αύξησε καθόλου το score
- Μετά δοκίμασα να κατεβάσω το tolerance σε 0.0000001 (default^2) στην εκμάθηση του μοντέλου γιατί το default είναι 0.0001 και μου έριξε το score κατά 0.1%.
- Τέλος δοκίμασα να παίξω με το ngram_range του vectorizer και πάλι μου έριχνε την απόδοση (περίπου 4% το (1,2) και 6% το (1,3)) και επίσης έκαναν μεγάλο overfit οι καμπύλες και ήταν και μακριά η μία από την άλλη:



- Εν τέλει μπορεί το μοντέλο που κατέληξα να έχω στα χέρια μου να έπιανε το καλύτερο score αλλά από το γράφημά του παρατήρησα ότι κάνει ένα μικρό overfit και οι καμπύλες πάλι είναι σχετικά μακριά:



Αλλά κατάφερα να το βελτιώσω ρίχνοντας τον αριθμό των features αισθητά (3.000 από 8.000) επειδή είχαμε δει στο μάθημα ότι τα πολλά features δεν είναι καλό για λίγα δεδομένα και το μοντέλο πλέον δεν κάνει overfit. Εδώ βέβαια συνέβη κάτι αναπάντεχο καθώς έβαλα τα features=3000 ενώ πριν ανέφερα ότι τα δοκίμαζα ανεβάζοντάς τα κατά 2000 τη φορά οπότε δεν έβαλα ποτέ 3000. Περίμενα λοιπόν να γλυτώσω το overfit με αυτή την αλλαγή και μαζί με αυτό να χάσω λίγο από το score. Εν τέλει το μοντέλο απέφυγε το overfit αλλά ανέβηκε και το score του κατά 0.3% και πήγε 71.34%! Το οποίο είναι και το τελικό (τα 3000 features σε συνδυασμό με τα παραπάνω επειδή τα ξαναδοκίμασα πάλι δεν βοήθησαν γιατί έριχναν αρκετά το score ή έκαναν σημαντικό overfit).

Σημείωση και πόρισμα: Αυτά που δοκίμασα ήταν αρκετά και όχι μόνο όσα είδαμε στο φροντιστήριο αλλά και κάποια έξτρα τα οποία κανένα από αυτά δεν κατάφερε να μου βελτιώσει το μοντέλο οπότε τελικά ίσως το training set να είναι πολύ μικρό για να καταφέρει οποιοδήποτε μοντέλο ένα καλύτερο score από 75% στο παρόν πρόβλημα.

ΤΕΛΙΚΟ ΚΑΛΥΤΕΡΟ SCORE ΚΑΙ LEARNING CURVES

```
f1-micro:  71.34092900964066 %
f1-weighted score:  70.4319025648829 %
accuracy score:  71.34092900964066 %
```