Τεχνητή Νοημοσύνη 2

Εργασία 1

Δημητρακόπουλος Κωνσταντίνος

ΑΜ: 1115201500034

Ερώτημα 1

Γνωρίζουμε:

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

Οπότε:

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

$$= \frac{2}{m}\sum_{i=1}^{m}\left(h_x(x^{(i)}) - y^{(i)}\right)\nabla_w\left(\sum_{i=1}^{m}\left(h_w(x^{(i)}) - y^{(i)}\right)\right)$$

$$= \frac{2}{m}\sum_{i=1}^{m}(wx^{(i)} - y^{(i)})\,\nabla_w\left(\sum_{i=1}^{m}(wx^{(i)} - y^{(i)})\right)$$

$$= \frac{2}{m}\left[\sum_{i=1}^{m}(wx^{(i)} - y^{(i)})\right]\sum_{i-1}^{m}x^{(i)}$$

$$= \frac{2}{m}\sum_{i-1}^{m}x^{(i)}\left[w\sum_{i-1}^{m}x^{(i)} - \sum_{i-1}^{m}y^{(i)}\right]$$

$$= \frac{2}{m}(X^T(Xw - y))$$

Ερώτημα 2

Το τελικό μοντέλο βρίσκεται στα τρία πρώτα κελιά του παραδοτέου notebook.

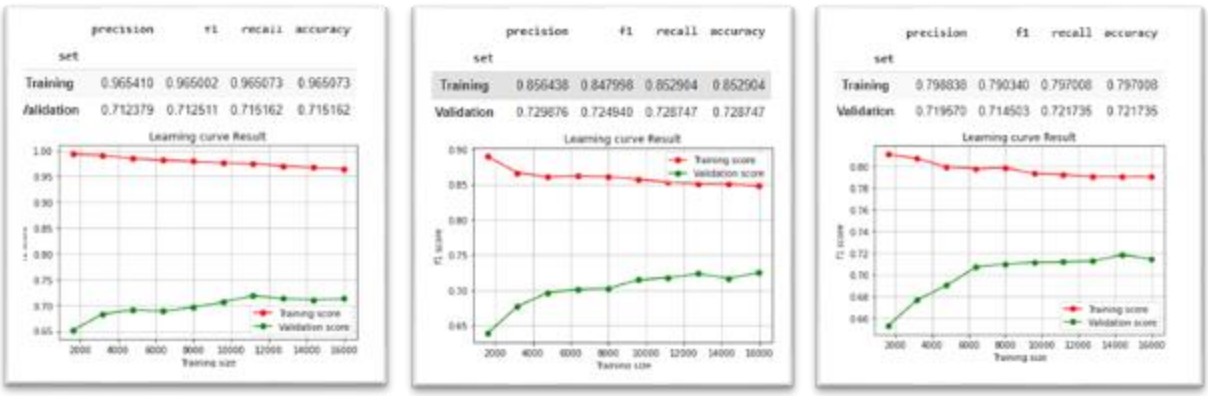
Η μεταβλητή validation_set_path κρατά το μονοπάτι του validation set και με αλλαγή της μπορεί να γίνει αξιολόγηση του test set.

Επιλογή Vectorizer

Αρχικά εξετάστηκαν οι παρακάτω vectorizers, με εισαγωγή του training και validation dataset σε έναν απλό multinomial logistic regression ταξινομητή με τις προκαθορισμένες παραμέτρους.

- CountVectorizer: υλοποίηση Bag Of Words
- Tf-IDF Vectorizer: υλοποίηση όπου χρησιμοποιεί τη συχνότητα κάθε όρου στο κάθε κείμενο σε σχέση με τη συχνότητα εμφάνισης σου στα υπόλοιπα κείμενα.
- HashingVectorizer: εύκολα κλιμακούμενη υλοποίηση, χωρίς dictionary, με τεχνική hashing.

Το παραγόμενο μοντέλο κάνει overfit σύμφωνα με τα αποτελέσματα:



Από αριστερά προς τα δεξιά: CountVectorizer - tfidfVectorizer - HashingVectorizer

Μετά από επεξεργασία των παραμέτρων των vectorizers απομακρύνεται ένα μέρος του θορύβου από το dataset και αποφεύγεται η κατάσταση overfitting στους CountVectorizer και TfIdfVectorizer.

Συγκεκριμένα:

- min_df, max_df: Θέτοντας το ελάχιστο όριο document frequency σε 0.01 και το μέγιστο σε 0.4 αποκόπτουμε ένα μέρος του dataset που αποτελεί θόρυβο για τον ταξινομητή.
- ngrams_range: Θέτοντας αυτή τη τιμή σε (1,5) περιέχουμε ως όρους συνδυασμούς των λέξεων ανά n ενισχύοντας τον ταξινομητή.

Τα αποτελέσματα φαίνονται παρακάτω:

Από αριστερά προς τα δεξιά: CountVectorizer - tfidfVectorizer - HashingVectorizer

Επίσης δοκιμάστηκαν οι vectorizers με τις παρακάτω τιμές παραμέτρων χωρίς να μεταβάλλουν σημαντικά το αποτέλεσμα:

→ TfidfVectorizer(min_df=0.01, max_df=0.99, ngram_range=(1,5), max_features=1000, smooth_idf=True, norm='l2', sublinear_tf=True, use_idf=False)
→ TfidfVectorizer(norm='l2', use_idf=True, smooth_idf=False, sublinear_tf=True, lowercase=True)
→ CountVectorizer(min_df=0.01, max_df=0.4, ngram_range=(1,5), max_features=1000)
→ CountVectorizer(lowercase=True, stop_words='english', min_df=0.2, binary=False)
→ CountVectorizer(lowercase=False, stop_words='english', binary=False)
→ CountVectorizer(lowercase=True, stop_words='english', binary=False, ngram_range=(1,2))
→ HashingVectorizer(n_features=2**12)

Data Preprocessing

Εφαρμόζεται προεπεξεργασία στα δεδομένα πριν δοθούν ως είσοδο στους vectorizers. Σκοπός αυτού του βήματος είναι η αφαίρεση πληροφορίας όπου δεν είναι χρήσιμη για τον ταξινομητή και ο μετασχηματισμός όρων σε κοινές σημασιολογικές ομάδες (λήμμα - πεζά γράμματα). Ως επιθυμητό αποτέλεσμα είναι η διατήρηση της πληροφορίας που θα ενισχύσει τη διαφορετικότητα μετα ξύ των κλάσεων και την ικανότητα του ταξινομητή να κατανέμει ορθά τα tweet στις κλάσεις αυτές.

Συγκεκριμένα έχουν υλοποιηθεί οι εξής συναρτήσεις που μετασχηματίζουν το dataset:

1. Αφαίρεση των συνδέσμων http - bit.ly
2. Αφαίρεση των μη αλφαριθμητικών χαρακτήρων.
3. Μετασχηματισμό όλων των κεφαλαίων χαρακτήρων σε πεζούς.
4. Lemmatize των λέξεων.
5. Αφαίρεση των english stop words.
6. Αφαίρεση των hashtags #[A-Za-z0-9]
7. Αφαίρεση των mentions @[A-Za-z0-9]

Έχουν ελεγχθεί διάφοροι συνδυασμοί των παραπάνω φίλτρων και τελικά κρατήθηκε ο καλύτερος από αυτούς: {1,3,4,5}

Τα αποτελέσματα φαίνονται στα παρακάτω διαγράμματα:

Από αριστερά προς τα δεξιά: CountVectorizer - tfidfVectorizer

Feature Engineering

Πραγματοποιήθηκε εξαγωγή πληροφοριών από τα δεδομένα του dataset, όπου περιγράφουν στατιστικά το κάθε tweet και το προσδιορίζουν λεξικά. Σκοπός αυτού του βήματος είναι η αναγνώριση ενός γλωσσολογικού μοτίβου που θα ενισχύσει τη διαφορετικότητα μεταξύ των κλάσεων και την ικανότητα του ταξινομητή να κατανέμει ορθά τα tweet στις κλάσεις αυτές. Οι νέες αυτές πληροφορίες εισάγονται σαν νέα χαρακτηριστικά στον τελικό πίνακα διανυσμάτων.

Συγκεκριμένα τα νέα χαρακτηριστικά για κάθε tweet είναι τα παρακάτω:

1. Πλήθος των χαρακτήρων.
2. Πλήθος των λέξεων.
3. Πλήθος των κεφαλαίων χαρακτήρων.
4. Πλήθος των λέξεων που αποτελούνται μόνο από κεφαλαίους χαρακτήρες.
5. Πλήθος των σημείων στίξης.
6. Πλήθος των hashtags.
7. Πλήθος των mentions.
8. Πλήθος των stop words.
9. Πλήθος μοναδικών λέξεων.
10. Συχνότητα μοναδικών λέξεων.
11. Συχνότητα stop words.
12. Μέσο μήκος λέξεων.
13. Συχνότητα ουσιαστικών, επιθέτων, ρημάτων και επιρρημάτων.

Έχουν ελεγχθεί διάφοροι συνδυασμοί των παραπάνω χαρακτηριστικών και τελικά κρατήθηκε ο καλύτερος από αυτούς: {10,11,12,13}

Τα αποτελέσματα φαίνονται στα παρακάτω διαγράμματα:

Από αριστερά προς τα δεξιά: CountVectorizer - tfidfVectorizer

Για την υλοποίηση των χαρακτηριστικών χρησιμοποιήθηκαν οι πηγές:

- https://www.analyticsvidhya.com/blog/2021/04/a-guide-to-feature-engineering-in-nlp/
- https://medium.com/swlh/nlp-all-them-features-every-feature-that-can-be-extracted-from-text-7032c0c87dee

Dimensionality Reduction

Πραγματοποιήθηκαν δοκιμές για μείωση της διάστασης του παραγόμενου πίνακα διανυσμάτων, με χρήση του αλγόριθμου PCA, και την εξερεύνηση τυχόν καλύτερων αποτελεσμάτων. Ως αποτέλεσμα της χρήσης των παραμέτρων min_df και max_df, οι ήδη υπάρχουσες διαστάσεις είναι λίγες. Παρόλα αυτά μειώνοντας περαιτέρω τη διάσταση του πίνακα, διατηρούμε το πλήθος των χαρακτηριστικών που δημιουργήθηκαν στην παραπάνω ενότητα, αλλάζοντας τη βαρύτητα τους.

Οι δοκιμές κατέληξαν σε μείωση της απόδοσης των μετρικών και συνεπώς δεν συμπεριλήφθηκαν στο τελικό μοντέλο.

Τα αποτελέσματα:

Από αριστερά προς τα δεξιά: CountVectorizer - tfidfVectorizer

Softmax Regression

Τα παραπάνω αποτελέσματα έχουν εξαχθεί από τον multinomial logistic regression ταξινομητή με προκαθορισμένες παραμέτρους. Σε αυτή την ενότητα θα ελεγχθούν η λειτουργικότητα και ο έλεγχος των κατάλληλων παραμέτρων για την μεγιστοποίηση των αποτελεσμάτων των μετρικών.

Παράμετρος penalty:

- L2 - Ελαχιστοποίηση της συνάρτησης κόστους:

$$\min_{w,c} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

- L1 - Ελαχιστοποίηση της συνάρτησης κόστους:

$$\min_{w,c} \|w\|_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1).$$

- Elasticnet - Ελαχιστοποίηση της συνάρτησης κόστους:

$$\min_{w,c} \frac{1-\rho}{2} w^T w + \rho \|w\|_1 + C \sum_{i=1}^{n} \log(\exp(-y_i(X_i^T w + c)) + 1),$$

  όπου η παράμετρος ρ ρυθμίζεται μέσω της παραμέτρου l1_ratio του μοντέλου.

Παράμετρος C:

Η παράμετρος C που φαίνεται στις παραπάνω συναρτήσεις κόστους.

Παράμετρος solver:

Ο αλγόριθμος που θα χρησιμοποιηθεί στο πρόβλημα (newton-cg, lbfgs, saga, sag).

Μετά από δοκιμές διαφόρων τιμών και εκτέλεση του αλγορίθμου GridSearchSV για το δοθέν training και validation dataset οι βέλτιστες τιμές των παραμέτρων είναι οι εξής:

- C: 100 -> TfIdfVectorizer , 0,1 -> CountVectorizer
- Solver: newton-cg
- Penalty: l2

Τα αποτελέσματα που προέκυψαν φαίνονται στα παρακάτω διαγράμματα:

Από αριστερά προς τα δεξιά: CountVectorizer - tfidfVectorizer

Τελικό Μοντέλο

Διαμόρφωση

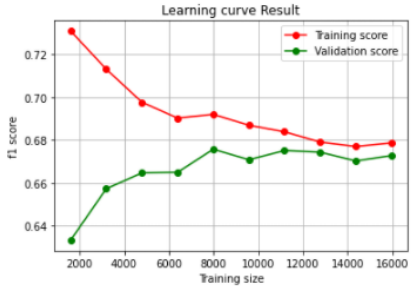
Το μοντέλο το οποίο επιλέχθηκε ως τελικό διαμορφώνεται ως εξής:

- Data Preprocessing :
  - → Αφαίρεση των συνδέσμων http - bit.ly
  - → Μετασχηματισμό όλων των κεφαλαίων χαρακτήρων σε πεζούς.
  - → Lemmatize των λέξεων.
  - → Αφαίρεση των english stop words.
- Feature Extraction:
  - → Συχνότητα μοναδικών λέξεων.
  - → Συχνότητα stop words.
  - → Μέσο μήκος λέξεων.
  - → Συχνότητα ουσιαστικών, επιθέτων, ρημάτων και επιρρημάτων.
- Vectorizer:
  - → TF-IDF Vectorizer
- Softmax Parameters:
  - → C: 100
  - → Solver: newton-cg
  - → Penalty: l2

Αποτελέσματα

Αποτελέσματα του τελικού μοντέλου:

set	precision	f1	recall	accuracy
Training	0.680184	0.678775	0.692288	0.692288
Validation	0.673559	0.672845	0.686678	0.686678



Υλοποίηση Βοηθητικών Συναρτήσεων

Κατά την ανάπτυξη της εργασίας υλοποιήθηκαν οι εξής συναρτήσεις:

- freq_pos:

  Προσθέτει τέσσερις επιπλέον στήλες στο dataframe που δέχεται όπου περιέχουν τη συχνότητα εμφάνισης ουσιαστικών, επιθέτων, ρημάτων και επιρρημάτων.

- get_scores:

  Δέχεται τους πίνακες Y των validation και training set με τα αληθινά labels και τους αντίστοιχους πίνακες με τα labels των προβλέψεων. Επιστρέφει ένα dataframe με τα αποτελέσματα των μετρικών precision, recall, f1, accuracy σε έλεγχο του training και test set.

- f1_learning_curve:

  Δημιουργεί τον πίνακα ts που περιέχει τα μεγέθη του training set σύμφωνα με τα οποία θα γίνει το training για τον τελικό σχηματισμό της learning curve.

  Δημιουργεί τον πίνακα test_fold όπου περιέχει μια ακολουθία από -1 κι έπειτα από 0, όπου υποδεικνύει ποιες γραμμές του πίνακα που θα εισαχθεί στη συνάρτηση learning_curve αποτελούν το training set και το validation set. Επίσης υποδεικνύει πως δεν θα γίνει cross validation αλλά μόνο έλεγχος στο δοθέν training και validation set.

  Καλείται η συνάρτηση learning_curve όπου για κάθε δοθέν μέγεθος του training set κάνει train το μοντέλο, έλεγχο στο ίδιου μεγέθους training set και έλεγχο σε όλο το validation set.

  Στη συνέχεια τυπώνεται η learning curve συμφωνα με τη πηγή https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html#sphx-glr-auto-examples-model-selection-plot-learning-curve-py

- display_false_predictions:

  Για κάθε λάθος πρόβλεψη τυπώνει τη σωστή κλάση, την κλάση που υπέδειξε το μοντέλο και τα δεδομένα του tweet.