

**Πατούχας Ιωάννης , 3200149**

**Μπίλιας Γεώργιος , 3200278**

**Παπαγεωργίου Γεώργιος , 3200227**

Εργασία 2 ( Μηχανική Μάθηση )

Πρώτα από όλα φορτώσαμε το dataset μέσω της βιβλιοθήκης του tensorflow και πιο συγκεκριμένα με το “keras” :

tf.keras.datasets.imdb.load\_data(num\_words=4000,skip\_top=3 )

H συνάρτηση αυτή δέχεται ως όρισμα τη μεταβλητή num\_words (m) η οποία προσδιορίζει πόσες λέξεις θα χρησιμοποιηθούν από το αναλοίωτο vocabulary που περιλαμβάνετα στο dataset. Στη δικια μας περίπτωση το 4000 επιφέρει τα καλύτερα αποτελέσματα. Το όρισμα skip\_top αναφέρεται στον αριθμό των πιο συχνων λέξεων που θα αγνοηθούν (skipped) , μεταβλητη n.

Πριν ξεκινήσουμε την αναφορά των αλγορίθμων είναι απαραίτητο να αναφέρουμε το χειρισμό αγνόησης των πιο σπάνιων λέξεων (k).

Αφότου δημιουργήσουμε το vocabulary χρησιμοποιούμε ένα dictionary το οποίο το αρχικοποιούμε με 0 και στη συνέχεια μετράμε πόσες φορές συναντάμε κάθε λέξη στο vocabulary. Έπειτα το ταξινομούμε (με αύξουσα σειρά) με τη χρήση της βιβλιοθήκης collections.OrderedDict

sorted\_dict = dict(sorted(dictonary.items(), key=lambda item: item[1]),reverse = True)

και αφαιρούμε τις λέξεις που εμφανίζονται λιγότερα.

counter = 0

k = 5

for x in sorted\_dict:

  print(x,sorted\_dict[x])

for x in list(sorted\_dict):

  print(x,sorted\_dict.pop(x,None))

  counter =  counter + 1

Στη συνέχεια δημιουργούμε νεο vocabulary με βάση το dictionary το οποιο υα νε και αυτο που θα χρησιμοποιήσουμε στην μετατροπή σε binary dataset για την εκπαίδευση των αλγορίθμων

Ακολουθούν οι εξηγήσεις , αναφορές και συγκρίσεις των αλγορίθμων , ID3 , Naïve Bayes (Bernoulli) και Logistic Regression

Bernoulli Naïve Bayes

C = (θετικο η αρνητικο) , Χ = 0,1

O Αλγόριθμος προϋποθέτει ότι οι τιμές Xn είναι ανεξάρτητες μεταξύ τους. Στη δική μας περίπτωση συμβολίζουν την ύπαρξη η όχι μίας λέξης.

Ο αλγόριθμος δέχεται ως όρισμα την μεταβλητή (α) η οποία χρησιμοποιείται αργότερα για την αποφυγή μηδενισματος πιθανοτήτων η ακόμη και απροσδιοριστίας , καθώς και το dataset (X) στο οποίο θα εκαπιδευτεί μαζί με τα επιθυμητά αποτελέσματα (y).

self.prop\_of\_each\_class = self.y\_count\_each / self.y\_count\_each.sum()

Στη περίπτωση μας και είναι 0.5 και τα δύο εφόσον τα data είναι split 50/50. Στην απο κάτω γραμμή λογαριθμίζουμε το αποτέλεσμα για μελλοντικές πράξεις

prob\_ofX\_knowingY = np.zeros([self.categories, self.num\_of\_words])

Για την αναπαρασταση πιθανοτητες με γνωση δεδομενων υα χρησιμοποιησουμε πινακα δυο διαστασεων τον οποιο αρχικοποιουμε με 0 .

Ο αριθμός των στηλων είναι ο αριθμος των λέξεων δηλαδη ||X|| και ο αριθμως των γραμμων είναι ισος με τον αριθμο των περιπτωσεων (2) επομένως

prob\_ofX\_knowingY[0][1]

συμβολίζει τη πιθανοτητα τη πιθανοτητα ύπαρξης της λέξης Χ2 με γνώση ότι ανήκει στη κατηγορία 0.

Δουλεύουμε για

X\_filtered = X[y==0,:] # take the rows where class is 0

Aθροίζουμε κάθε στήλη των δεδομένων του πίνακα X o οποίος τώρα περιέχει μόνο τις γραμμες κατηγοριας 0 και αποθηκευουμε τα αποτελέσματα στη μεταβλητη num1+α και τη διαιρουμε με των αριθμο των δεδομένων κατηγορίας 0 (12500) + 2\*α. Έπειτα αποθηκεύουμε το αποτέλεσμα στη πρώτη γραμμή του πίνακα prob\_ofXknowingY

Το ιδιο κάνουμε για .

Τέλος λογαριθμιζουμε τον πίνακα probofXknowingY , και αποθηκεύουμε το αποτέλεσμα στη μεταβλητή self.logged\_positive.

Ολη αυτη η διαδικασια ηταν για την πιθανοτητα να υπαρχει η λέξη. Για το self.logged\_negative μπορούμε απλώς να χρησιμοποιήσουμε τον τυπο

Όλα τα παραπάνω ήταν για τη συνάρτηση fit()

Τώρα θα εξηγήσουμε την συνάρτηση predict() η οποία χρησιμοποιεί τις πιθανότητες που υπολόγισε η συναρτηση fit().

Χρησιμοποιούμε τις πιθανοτητες που μάθαμε και με τους κατάληλλους πολλ/σμους με το transpose του Χ του οποιου θέλουμε να προβλέψουμε με τη χρήση της συνάρτησης .dot()

**Ξέρουμε** από θεωρία πιθανοτήτων ότι

Άρα log\_joint\_likelihoods = log\_likelihoods + self.logged\_prop\_of\_each\_class

Tέλος βρίσκουμε το set των max values με την συνάρτηση arg max για κάθε στήλη και επιστρέφουμε τις προβλέψεις.

Αποτελέσματα

SkLearn Naïve Bayes

Calendar

Description automatically generated

Naive Bayes

Calendar

Description automatically generated

Από τα παραπάνω συμπεράσματα παρατηρούμε ότι ο αλγόριθμος μας έχει μονάχα 1% απόκλιση. Ακολουθεί η εξήγηση των διαγραμμάτων.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Στο τομέα του accuracy παρατηρούμε ότι οι δύο γραφικές είναι σχεδόν ολόιδιες και συγκεκριμενα για 15000 λέξεις υπάρχει μεγάλη απότομη πτώση απο τον δικο μας αλγοριθμο. Απο κει και πέρα ο δικος μας αυξάνεται λογαριθμικα ενω του scit learn ξαναμειωνεται

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Οι γραφικες του f1 είναι παρόμοιες με accuracy

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Οσον αφορά το recall στον δικο μας αλγοριθμο η αποσταση είναι πολυ μικρη μεταξυ training και recall που σημαίνει πολύ μικρό “overfitting” ενω απο το scit learn υπαρχει περισσοτερο overfitting.

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Tελος στο precision παλι ο δικος μας αλγοριθμος έχει λιγότερο overfitting (σχεδον καθολου) σε σχεση με του scit – learn

ID3

Ο αλγόριθμος ID3 κατασκευάζει δέντρα απόφασης. Κάθε φορά επιλέγετε η κατάλληλη ιδιότητα που συμφέρει και τα δεδομένα χωρίζονται σε 2 υπόδεντρα με βάση αυτή. Επεκτείνει κάθε φορά τους κόμβους που χρειάζονται μέχρι να καταλήξει σε φύλλο απόφασης.

Στον αλγόριθμο χρησιμοποιείται και μια κλάση Node για την κατασκευή του δέντρου

Ο αλγόριθμος δέχεται σαν ορίσματα τον μικρότερο αριθμό ιδιοτήτων για spilt ( min\_samples\_split), μέγιστο βάθος του δέντρου(max\_depth),και τον αριθμό των ιδιοτήτων που θα χρησιμοποιήσουμε(number\_Of\_features).

Στην εκπαίδευσή απλά κατασκευάζουμε το δέντρο απόφασης σπάζοντας καταλληλά τα δεδομένα μας.

Στον id3 συνήθως δεν χρειάζεται να χρησιμοποιήσουμε όλες τις ιδιότητες που μας δίνονται (συνήθως παίρνουμε μόνο τις πιο σημαντικές). Επίσης περιορίζουμε το βάθος του δέντρου που θα φτάσουμε για να αποφύγουμε να κάνουμε overfitting. Τέλος μας νοιάζει να μπορούμε να αναγνωρίσουμε ότι είμαστε σε κόμβο φύλλου δηλαδή σε κάθε κόμβο έχουμε μόνο 1 ιδιότητα.

Κριτήριο τερματισμού με βάση τα πιο πάνω:

if(depth>=self.max\_depth or n\_labels==1 or

n\_samples<self.min\_samples\_split)

Για να βρούμε ποια ιδιότητα και τιμή είναι η κατάλληλη για σπάσιμο χρησιμοποιούμε μια βοηθητική συνάρτηση find\_bestSplit.

Για κάθε ιδιότητα των παραδειγμάτων υπολογίζουμε το κέρδος πληροφορίας(information gain) για κάθε τιμή της( σε εμάς οι τιμές μας είναι το 0 και 1) και κρατάμε την ιδιότητα με το καλύτερο information gain.

Για τον υπολογισμό του information gain χρησιμοποιούμε την συνάρτηση information\_gain.

Πρώτο βήμα υπολογίζουμε την εντροπία(entropy) του πατερά και των παιδιών που θα πρόκυπταν μετά το σπάσιμο με την συγκεκριμένη ιδιότητα και τιμή.

Entropy:

  s = 0

        count = np.bincount(y)

        p = count / len(y)

        for p in p:

          if p>0:

            s= s + p\*np.log(p)

        return -s

στην συνέχεια το information gain υπολογίζεται ως εξής:

information\_gain = parent\_entropy - childEntropy

β)

ID3

Calendar

Description automatically generated

sklearn ID3

Calendar

Description automatically generated

Από τα παραπάνω παρατηρούμε ότι ο δικός μας αλγόριθμος τα πηγαίνει πολύ καλά σε σχέση με αυτόν της sklearn αφού έχουν μικρές διάφορες .

ID3

Chart, line chart

Description automatically generated

sklearn ID3

Chart, line chart

Description automatically generated

Παρατηρούμε ότι οι 2 γραφικές είναι ιδίες. Παρατηρούμε επίσης ότι υπάρχει λίγο overfitting λόγο της απόστασης των δυο ευθείων .

ID3

Chart, line chart

Description automatically generated

sklearn ID3

Chart, line chart

Description automatically generated

Παρατηρούμε ότι οι γραφικές για το f1 είναι σχετικά καλύτερες από αυτές του accuracy

ID3

Chart, line chart

Description automatically generated

sklearn ID3

Chart, line chart

Description automatically generated

Παρατηρούμε ότι στο recall οι δυο γραφικές είναι ιδίες με τις 2 γραμμές να είναι αρκετά κοντά μεταξύ τους. Η γραφική του training κατηφορίζει απότομα στις 10000 λέξεις.

ID3

Chart, line chart

Description automatically generated

sklearn ID3

Chart, line chart

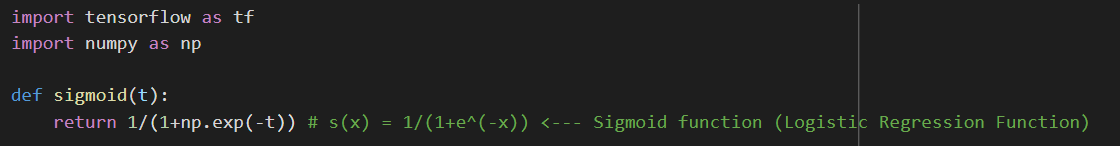
Description automatically generated

Παρατηρούμε ότι οι δυο γραφικές έχουν απόσταση μεταξύ τους. Με αποτέλεσμα να κάνει overfitting μαθαίνοντας ίσως καλυτέρα τα training και να μην τα πηγαίνει τόσο καλά στα νέα παραδείγματα.

Τέλος για όλες τις γραφικές αναφέρω ότι και στους δυο αλγορίθμους έχουμε περιορίσει το βάθος του δέντρου σε 10 γιατί αν αφήναμε την default τιμή δεν είχε νόημα η γραφική έχοντας αρκετά μεγάλο overfitting.

Logistic Regression

Αρχικά, δημιουργούμε την παρακάτω **σιγμοειδής συνάρτηση** (συνάρτηση λογιστικής)



Στην συνέχεια, όπως φαίνεται και στην παρακάτω εικόνα, δημιουργούμε μια κλάση **Logistic Regression** όπου την αρχικοποιούμε (στην function **\_init\_**) με το **learning\_rate** ως 0.3, το λ (**lmd**) ως 0.4, τον αριθμό των αναδρομών (**number\_of\_iterations**) σε 1000 και το βάρος και την προκατάληψη (**w και b**) σε None.

Το initialization της φαίνεται παρακάτω:

Εικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

Μετά μέσα στην κλάση **Logistic Regression** δημιουργούμε 2 functions

Aρχικά, δημιουργούμε την **fit**. Σε αύτη την συνάρτηση δίνουμε 3 παραμέτρους. Τον self και τα δεδομένα x και y.

Με τον x, χρησιμοποιώντας το .shape, αρχικοποιούμε τον αριθμό των δειγμάτων και χαρακτηριστικών των δεδομένων του. Ακολούθως, αρχικοποιούμε τα βάρη του Logistic Regression ως ένα πίνακα array με μέγεθος τον αριθμό των χαρακτηριστικών και θέτουμε όλα τα στοιχεία του ως 0 καθώς και την προκατάληψη του ως 0.

Στην συνέχεια, γίνεται ένα loop ίσο με τον αριθμό των αναδρομών. Εκεί μέσα κάθε φορά υπολογίζεται το αποτέλεσμα της γραμμικής παλινδρόμησης και μπαίνει ως παράμετρος στη σιγμοειδή συνάρτηση.

Επιπροσθέτως, υπολογίζουμε την κατάβαση dw και db για να υπολογίσουμε την κατάβαση κλίσης και συνεπώς το σφάλμα.

Στο τέλος, αφαιρούμε απτό αρχικό βάρος τον πολλαπλασιασμό του dw και του learning rate. Με ανάλογο τρόπο αφαιρούμε και απτή προκατάληψη αλλά αντί του dw χρησιμοποιούμε το db.

Με αυτόν τον τρόπο έχουμε επιτύχει να κάνουμε κανονικοποιήση

Εικόνα που περιέχει κείμενο, χλόη, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματαΗ fit αναπαριστάται στην παρακάτω εικόνα:

Τέλος, δημιουργούμε και την predict. Η predict κάνει χρήση του τύπου της γραμμικής παλινδρόμησης με βάση τα βάρη που έχουν υπολογιστεί στην συνάρτηση fit. Τα αποτελέσματα της στην συνέχεια μπαίνουν ως παράμετρος στην σιγμοειδή συνάρτηση. Εν συνεχεία, παίρνουμε τα αποτελέσματα που προέκυψαν απ΄την χρήση της σιγμοειδής συνάρτησης και τα βάζουμε σε μία λίστα class\_prediction με τον ανάλογο τρόπο: αν το y μέσα στο logistic\_prediction είναι <= 0,5 τότε μπαίνει στο class\_prediction ως 0 αλλιώς μπαίνει ως 1.

Εικόνα που περιέχει κείμενο, εσωτερικό, στιγμιότυπο οθόνης

Περιγραφή που δημιουργήθηκε αυτόματαΗ συνάρτηση predict:

ΣΗΜΕΙΩΣΗ: Η παράμετρος λ έχει τεθεί ως 0.4, ο αριθμός των αναδρομών ως 1000 και το learning\_rate ως 0.3. Αυτό προέκυψε ύστερα από αρκετό τεστάρισμα με διάφορους αριθμούς στις παραμέτρους για την εύρεση ενός βέλτιστου αποτελέσματος.

**ΑΠΟΤΕΛΈΣΜΑΤΑ**

**Logistic Regression**

**Εικόνα που περιέχει κείμενο, πίνακας αποτελεσμάτων, μέτρο, συσκευή

Περιγραφή που δημιουργήθηκε αυτόματα**

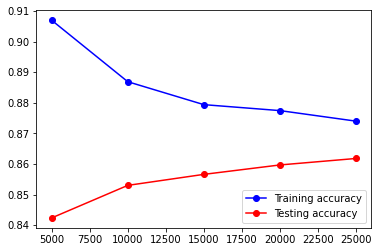
**Scit-Learn Logistic Regression**

**Εικόνα που περιέχει κείμενο, μέτρο, συσκευή

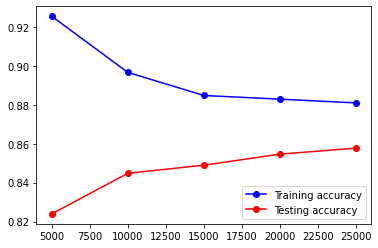
Περιγραφή που δημιουργήθηκε αυτόματα**

Παρατηρούμε ότι τα αποτελέσματα των δύο αλγορίθμων είναι σχετικά πανομοιότυπα. Έχουν τα ίδια average αποτελέσματα και το ίδιο accuracy. Υπάρχει μόνο μια απόκλιση 1% στο precision και στο recall των 1, όπου ο κώδικας υπερισχύει.

Accuracy



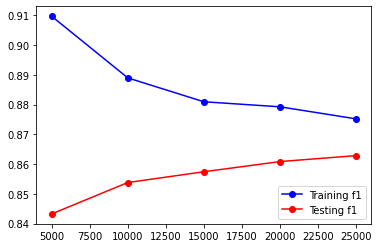
Δική μας



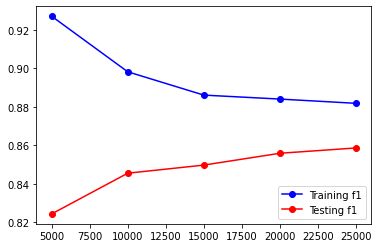
Scit-Learn

Παρατηρούμε ότι οι δυο γραφικές είναι πανομοιότυπες. Η απόσταση μεταξύ του training και testing όσο πάει και συγκλίνει. Η μόνη ουσιαστική διαφορά είναι πως το δικό μας training ξεκινάει από περίπου 91% ενώ του Scit-Learn από περίπου 92% και το δικό μας testing ξεκινάει από 84% ενώ του Scit-Learn από 82%. Ίσως ο δικός μας να συγκλίνει και λίγο παραπάνω.

F1

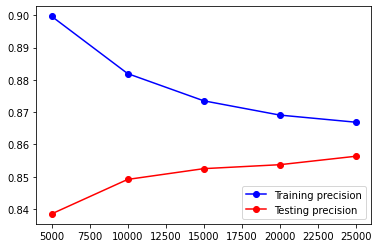


Δική μας

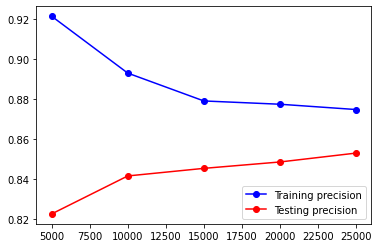


Scit-Learn

Στην f1 παρατηρούμε όπως και παραπάνω πως και οι δύο γραφικές συγκλίνουν με ανάλογο τρόπο και πως είναι πανομοιότυπες με σχετική απόκλιση 1-2% στο ξεκίνημα. Η δική μας φαίνεται να συγκλίνει περισσότερο.

Precision

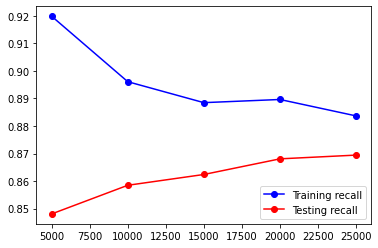
Δική μας



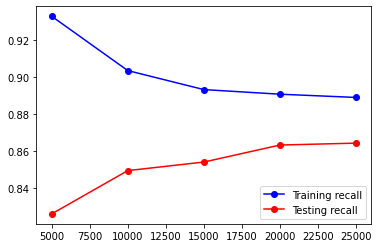
Scit-Learn

Στην precision παρατηρούμε πως όπως και στις παραπάνω γραφικές οι training και testing όσο πάει και συγκλίνουν. Υπάρχει μια διαφορά 2% ως προς το πού ξεκινούν όπου στο training υπερισχύει η δική τους ενώ εμάς στο testing. Επίσης παρατηρούμε πως η δική μας συγκλίνει μέχρι την απόσταση περίπου 1,5% ενώ η δική τους λίγο παραπάνω από 2%.

Recall



Δική μας



Scit-Learn

Στην recall παρατηρούμε πως του scit-Learn είναι σχετικά πιο αποδοτική στο training με σχετική απόδοση <1%, ενώ όσο αφορά το testing η δική μας αποδίδει καλύτερα με απόδοση <1% επίσης. Και οι δύο γραφικές συγκλίνουν με ανάλογο τρόπο.

Μερος Γ

***Σημείωση: Για την δημιουργία των στατιστικών στοιχείων σε σχεση με τα προηγουμενα 2 τρεξαμε το μερος Γ πολλες φορες καθε φορα με διαφορετικο μηκος dataset αντι να φτιαχναμε ειδικη συναρτηση που θα ταν πιο ιδανικη λογω χρονου.Επισης λειπουν οι γραφικες για training , οι παρακάτω είναι από τα δεδομενα testing . Η δημιουργια των γραφηματων εγινε με χρηση της plot***

Chart, line chart

Description automatically generated

Στη γραφικη loss παρατηρουμε γραμμικη μειωση σε συναρτηση των εποχών (epochs)

RNN

Chart, line chart

Description automatically generated

Scit learn

Chart, line chart

Description automatically generated

Δικος μας

Chart, line chart

Description automatically generated

Στο accuracy παρατηρουμε ότι στο RNN όσο αυξάνουμε το μέγεθος του dataset έχουμε όλο περισσότερη ακρίβεια αλλά αθξάνεται λογαριθμικά

RNN

Chart, line chart

Description automatically generated

Naïve Bayes(Scit learn)

Chart, line chart

Description automatically generated

Naïve Bayes(Δικος μας)

Chart, line chart

Description automatically generated

Παρόμοια παρατήρηση μπορούμε να κάνουμε και για το f1

RNN

Chart, line chart

Description automatically generated

Naïve Bayes(scit learn)

Chart, line chart

Description automatically generated

Naïve Bayes(Δικος μας)

Chart, line chart

Description automatically generated

Για το recall σε σχεση με τον Naïve Bayes το rnn αθξάνεται συνέχως ενώ στα αλλα παρατηρούνται πτώσεις

RNN

Chart, line chart

Description automatically generated

Naïve Bayes(Scit learn)

Chart, line chart

Description automatically generated

Naïve Bayes ( Δικός μας)

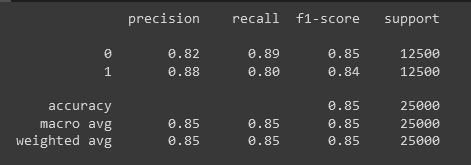
Chart, line chart

Description automatically generated

Τελος στο precision παρατηρουμε ότι το Rnn έχει μεγάλη πτώση προς το τελος παρόμοια με τις αλλες δυο

Πινακες

RNN



Naïve Bayes (scit learn)

Calendar

Description automatically generated

Naïve Bayes (Δικος μας)

Calendar

Description automatically generated