Πανεπιστήμιο Πειραιώς

Σχολή Τεχνολογιών Πληροφορικής και Τηλεπικοινωνιών
Τμήμα Ψηφιακών Συστημάτων

Μάθημα: Προηγμένα Θέματα Ανάλυσης Δεδομένων

Διδάσκων: Μαρία Χαλκίδη

Ονόματα Μελών Ομάδας	ΑΜ
Αντώνης Γάντζος	E18030

Αναφορά Εξαμηνιαίας Εργασίας

1) Abstract

Σκοπός της συγκεκριμένης εργασίας είναι η επιτυχημένη υλοποίηση μίας εφαρμογής που διαχωρίζει ψευδείς από αληθείς ειδήσεις. Για την υλοποίηση της εφαρμογής πρώτα χρειάζεται η εκπαίδευση ενός μοντέλου, το οποίο ενσωματώνεται στην εφαρμογή και εκτελεί τις προβλέψεις. Για ην εκπαίδευση του μοντέλου χρησιμοποιήθηκε ένα training dataset, στο οποίο εφαρμόστηκαν οι αλγόριθμοι κατηγοριοποίησης (classifiers) Support Vector Machine (S.V.M.) και Λογιστική Παλινδρόμηση (Logistic Regression). Στη συνέχεια συγκρίθηκαν οι αποδόσεις του κάθε αλγορίθμου πάνω σε διάφορες μετρικές και ο πιο αποδοτικός αλγόριθμος ήταν αυτός που ενσωματώθηκε στο μοντέλο. Αυτή η διαδικασία εκτελείται αφού γίνει η κατάλληλη προ-επεξεργασία στο σετ δεδομένων, που χρησιμοποιείται για την εκπαίδευση του μοντέλου. Για την υλοποίηση της εργασίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού python.

1.1) Περιγραφή συνόλου δεδομένων

Το σετ δεδομένων που χρησιμοποιείται για την υλοποίηση της εργασίας είναι ένα σύνολο ειδήσεων. Κάθε είδηση αποτελεί και μία εγγραφή στο σύνολο δεδομένων. Κάθε εγγραφή αποτελείται από μερικά χαρακτηριστικά. Συγκεκριμένα τα χαρακτηριστικά (features) του σετ είναι τα εξής 4 :

- id : αριθμός εγγραφής.
- title : τίτλος της είδησης.
- author : το όνομα του συγγραφέα που έγραψε το κείμενο της είδησης.
- text : το κείμενο της είδησης.

Ακόμα κάθε εγγραφή περιλαμβάνει και ένα label, το οποίο παίρνει 2 μόνο τιμές, το 0 και το 1. Οι τιμές αυτές συμβάλλουν στη κατηγοριοποίηση μιας είδησης ως αληθής ή ψευδής, καθώς όποια εγγραφή του dataset έχει label=0 θεωρείται αληθής και όποια έχει label=0 ψευδής. Αυτά είναι συνοπτικά τα χαρακτηριστικά που αποτελούν κάθε εγγραφή που υπάρχει στο σύνολο δεδομένων.

1.2) Λίγα Λόγια για τους Αλγορίθμους Κατηγοριοποίησης

Για την εκπαίδευση του μοντέλου, ώστε να κατηγοριοποιεί σωστά μία είδηση ως αληθή ή ψευδή χρησιμοποιήθηκαν 2 αλγόριθμοι κατηγοριοποίησης, ο SVM και ο αλγόριθμος Λογιστικής Παλινδρόμησης. Παρακάτω ακολουθεί μία συνοπτική θεωρητική περιγραφή των αλγορίθμων για να γίνει κατανοητός, ο τρόπος με τον οποίο λειτουργούν και διαχωρίζουν τα δεδομένα.

- **Logistic Regression**: ο αλγόριθμος αυτός εκτιμά απευθείας τη πρόγνωση ενός στιγμιότυπου δεδομένων χ χρησιμοποιώντας τις τιμές των χαρακτηριστικών του. Η βασική ιδέα της λογιστικής παλινδρόμησης είναι η χρήση μιας συνάρτησης πρόβλεψης η οποία αναπαριστά τα δεδομένα του χ ως εξής : $\boldsymbol{P(y=1|x;w) = \frac{1}{1+e^{-z}}}$, όπου $z= g(w^Tx)$, με το w^T να ορίζεται ως ένα διάνυσμα που περιλαμβάνει όλες τις παραμέτρους w για κάθε χ (feature) που υπάρχει στο σετ δεδομένων. Για παράδειγμα στο συγκεκριμένο σετ δεδομένων το διάνυσμα αυτό περιλαμβάνει 4 παραμέτρους, βρίσκεται δηλαδή στο R^4. Το αποτέλεσμα της συνάρτησης είναι η πιθανότητα να ισχύει y=1 και ισχύει όποτε $g(z)\geq 0$. Διαφορετικά το label ης εγγραφής κατηγοριοποιείται ως 0.
  Επειδή λοιπόν η λογιστική παλινδρόμηση υπολογίζει τις εκ των υστέρων πιθανότητες χωρίς να κάνει κάποια υπόθεση για τις υπό συνθήκη πιθανότητες είναι πολύ γενική, μπορεί δηλαδή να χρησιμοποιηθεί σε πολλές περιπτώσεις κατηγοριοποίησης δεδομένων.

- **Support Vector Machines (SVM) :**
  Μια Μηχανή Διανυσμάτων Υποστήριξης είναι ένα μοντέλο κατηγοριοποίησης το οποίο βασίζεται στην αρχή της δομικής ελαχιστοποίησης κίνδυνου και λειτουργεί εκπαιδεύοντας γραμμικά η μη γραμμικά όρια απόφασης (margins), στο χώρο των χαρακτηριστικών με σκοπό να διαχωρίσει τις κατηγορίες. Επίσης παρέχει πολύ ισχυρές δυνατότητες προσαρμογής, ελέγχει δηλαδή τη πολυπλοκότητα του μοντέλου έτσι ώστε να εξασφαλίζεται καλή απόδοση χωρίς να δημιουργούνται προβλήματα υπέρ-προσαρμογής (overfitting), που είναι και ο λόγος που ο συγκεκριμένος classifier επιλέχθηκε σε αυτή την εργασία. Ένας SVM classifier είναι λοιπόν, ένα διαχωριστικό μοντέλο το οποίο επηρεάζεται μόνο από τα στιγμιότυπα εκπαίδευσης που βρίσκονται κοντά στα όρια των κατηγοριών που θέλει να χωρίσει.

2) Εκπαίδευση του μοντέλου

2.1) Προ-Επεξεργασία Δεδομένων

Το μοντέλο εκπαίδευσης δεδομένων υλοποιείται με τη χρήση ενός προγράμματος. Για τη κατασκευή του προγράμματος χρησιμοποιείται η γλώσσα προγραμματισμού python. Αρχικά εισάγονται οι απαραίτητες βιβλιοθήκες που θα χρησιμοποιηθούν στη προ-επεξεργασία των δεδομένων και στη συνέχεια στην εκπαίδευση των κατηγοριοποιητών και στην αξιολόγηση τους.

```
#this program detects real (0) and fake (1) news
#importing the libraries
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string
```

Έπειτα εκτελείται μετατροπή στα labels. Κάθε label με τιμή 0, αντικαθίσταται με το string ‘Real’ και αντίστοιχα κάθε label με value 1 αντικαθίσταται με το string ‘Fake’. Αυτή η διαδικασία εκτελείται με σκοπό τη καλύτερη κατανόηση του τι αντιπροσωπεύουν οι τιμές 0 και 1 σε κάθε label.

```
#load the data
#read the loaded data we will use for the training
df=pd.read_csv('train.csv')
conversion_dict= {0:"Real" , 1:"Fake"}
df['label']=df['label'].replace(conversion_dict)
df.head()
```

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	Fake
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	Real
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	Fake
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	Fake
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	Fake

Στη συνέχεια γίνεται έλεγχος για τυχών διπλό-εγγραφές και missing values και αφαίρεση τους από το dataset με σκοπό την εξομάλυνση των δεδομένων.

Ακόμα γίνεται επιλογή των πιο απαραίτητων χαρακτηριστικών για την εκπαίδευση του μοντέλου. Επιλέγονται ως πιο σημαντικά και κρίσιμα για την εκπαίδευση τα χαρακτηριστικά title και author, καθώς από τον τίτλο μίας είδησης και τον συγγραφέα του άρθρου είναι πιο εύκολο να διασταυρώσει κανείς πηγές για να ενημερωθεί για την εγκυρότητα τις είδησης, από το να εκτελέσει διασταύρωση πηγών στο ίδιο το κείμενο. Τα 2 χαρακτηριστικά αυτά συνδυάζονται σε μία ενιαία στήλη. Παρακάτω φαίνεται ενδεικτικά η μορφή ης καινούριας στήλης.

```
#combine the ccolumns that are imporant for the training of he program
df['comb']=df['author'] + ' ' +df['title']
df.head()
```

	id	title	author	text	label	comb
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	Fake	Darrell Lucus House Dem Aide: We Didn't Even S...
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	Real	Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	Fake	Consortiumnews.com Why the Truth Might Get You...
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	Fake	Jessica Purkiss 15 Civilians Killed In Single ...
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	Fake	Howard Portnoy Iranian woman jailed for fictio...

2.2) Χρήση TF-IDF

Τελευταίο βήμα για τη προ-επεξεργασία των δεδομένων είναι η μετατροπή το του text που δίνεται ως input στο μοντέλο σε ένα διάνυσμα. Κάθε τιμή του διανύσματος αντιπροσωπεύει τη συχνότητα κάθε λέξης του κειμένου. Για την εκτέλεση αυτής της μετατροπής χρησιμοποιείται ένας TF-IDF (Text Frequency-Inverse Document Frequency). Ο όρος αυτός αναφέρεται σε έναν text vectorizer, που μετατρέπει το κείμενο σε ένα χρήσιμο, για τον αλγόριθμο κατηγοριοποίησης, διάνυσμα. Συνδυάζει 2 έννοιες τη Συχνότητα Κειμένου (Text Frequency) και την Αντίστροφη Συχνότητα Εγγράφου (Inverse Document Frequency). Ως συχνότητα ορίζεται ο αριθμός εμφανίσεων ενός συγκεκριμένου όρου μέσα στο κείμενο και ενδεικνύει το μέγεθος της σημαντικότητας του όρου αυτού για το κείμενο. Το διάνυσμα συχνοτήτων δομείται ως εξής. Ο αριθμός των σειρών του διανύσματος είναι ο αριθμός των εγγράφων και ο αριθμός των στηλών είναι ο αριθμός του κάθε ξεχωριστού όρου που υπάρχει σε κάθε κείμενο, για όλα τα έγγραφα. Συχνότητα Εγγράφων είναι ο αριθμός των εγγράφων που περιέχουν έναν συγκεκριμένο όρο. Αυτό δείχνει πόσο συνηθής είναι η εμφάνιση του όρου αυτού στα έγγραφα που έχουν δοθεί στον αλγόριθμο. Αντίστροφη συχνότητα δεδομένων είναι το ‘βάρος’, δηλαδή η σημαντικότητα που δίνεται σε κάθε όρο σε ένα κείμενο. Το βάρος ενός όρου μειώνεται αν η συχνότητα του ανά τα κείμενα που έχουν δοθεί είναι μεγάλη. Ουσιαστικά το διάνυσμα που δημιουργείται αποτυπώνει το πόσο σημαντική είναι κάθε λέξη που χρησιμοποιείται σε κάθε κείμενο. Αυτό συμβάλλει στον διαχωρισμό τον λέξεων και επιτρέπει στον αλγόριθμο να μην πάρει υπόψιν τους λιγότερο σημαντικούς όρους (πχ σημεία στίξης, σύνδεσμοι, οριστικά και αόριστα άρθρα και άλλα) για την

καηγοριοποίηση του αποτελέσματος. Με την ενσωμάτωση του TF-IDF vectorizer και τη μετατροπή των δεδομένων το κομμάτι της προ-επεξεργασίας τελειώνει.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vect=TfidfVectorizer(stop_words='english', max_df=0.7)
#stop words are useless words that will be needed for the data
```

2.3) Εκπαίδευση των Classifiers

Επόμενο βήμα, τώρα που τα δεδομένα είναι στη μορφή που χρειάζεται για την εκπαίδευση των αλγορίθμων, είναι η υλοποίηση της διαδικασίας της εκπαίδευσης. Χρησιμοποιείται η μέθοδος **train_test_split,** για τον διαχωρισμό του dataset σε 2 κομμάτια. Ορίζεται ότι το 80% των εγγραφών του συνόλου δεδομένων θα χρησιμοποιηθούν για την εκπαίδευση των αλγορίθμων και το 20% θα αποτελούν το test size, δηλαδή θα είναι δεδομένα που δε θα χρησιμοποιηθούν για την εκπαίδευση, αλλά για τη δοκιμή των αλγορίθμων μετά τη διαδικασία της εκπαίδευσης για να δοκιμαστεί η αποδοτικότητά τους. Από το σύνολο δεδομένων κρατούνται μόνο τα χαρακτηριστικά title και author για την εκπαίδευση των αλγορίθμων. Τα διαχωριζόμενα δεδομένα εισάγονται στον TF-IDF vectorizer.

```
#split the data to training and testing data
#80% of the data is for training
#20% of the data is for testing
from sklearn.model_selection import train_test_split
x=df['title']+ '' +df['author']
x_train, x_test, y_train, y_test=train_test_split(x, df['label'], test_size=0.20, random_state=0)
tfid_x_train = vect.fit_transform(x_train)
tfid_x_test = vect.transform(x_test)
```

2.4) Προβλέψεις Αλγορίθμων και Μετρικές Αξιολόγησης

Έχοντας χωρίσει και εισάγει τα δεδομένα ορθά στο μοντέλο, ξεκινάει πλέον ο ορισμός των classifiers. Όπως προ-αναφέρθηκε χρησιμοποιούνται οι αλγόριθμοι κατηγοριοποίησης Logistic Regression και SVM. Εισάγονται στο πρόγραμμα με τη χρήση της βιβλιοθήκης **sklearn**. Από την αρχή εκπαίδευσης του κάθε αλγορίθμου μέχρι το τέλος, έχουν οριστεί για τον κάθε έναν 2 μεταβλητές start και end που χρησιμοποιούνται για μέτρηση του χρόνου σε δευτερόλεπτα που παίρνει στον κάθε αλγόριθμο να κάνει fit τα δεδομένα. Αφού τα δεδομένα εκπαίδευσης εισαχθούν και στους 2 classifiers και η εκπαίδευση του κάθε αλγορίθμου ολοκληρωθεί καλούνται να κάνουν προβλέψεις για όλα τα δεδομένα ελέγχου, το 20% του συνολικού dataset, που ορίστηκε κατά τον διαχωρισμό των δεδομένων. Τα αποτελέσματα των προβλέψεων παραδίδονται παρακάτω.

```
#print the predictions of the classifiers
#for the predictions here the data given to the algorithms is already known
print("Predictions of the Logistic Regression model : ",LR_clf.predict(tfid_x_test))
print("Predictions of the SVM model : ",SVM_clf.predict(tfid_x_test))


print("Actual values: ",y_test.values)
```

```
Predictions of the Logistic Regression model :  ['Real' 'Real' 'Fake' ... 'Real' 'Real' 'Fake']
Predictions of the SVM model :  ['Real' 'Real' 'Real' ... 'Real' 'Real' 'Real']
Actual values:  ['Real' 'Real' 'Fake' ... 'Real' 'Real' 'Fake']
```

Οι αλγόριθμοι αξιολογήθηκαν με βάση την αποδοτικότητα τους σε 3 διαφορετικές μετρικές, την ακρίβεια κατηγοριοποίησης, δηλαδή του ποσοστό επιτυχημένης κατηγοριοποίησης για κάθε classifier, τον χρόνο εκπαίδευσης και τον χρόνο εκτέλεσης. Για ον υπολογισμό του χρόνου εκτέλεσης υλοποιήθηκε μία συνάρτηση που καλεί τους 2 αλγορίθμους να εκτελέσουν προβλέψεις για 1000 τυχαία δεδομένα (ενδεικτικά επιλέγονται τα 1000 πρώτα). Από την αρχή μέχρι το τέλος της συνάρτησης ο χρόνος εκτέλεσης των προβλέψεων για κάθε αλγόριθμο μετριέται με την ίδια μεθοδολογία που εφαρμόζεται για τη μέτρηση του χρόνου εκπαίδευσης. Για τη μέτρηση του χρόνου χρησιμοποιείται η βιβλιοθήκη **time.** Σημειώνεται ότι τα αποτελέσματα ενδέχεται να διαφέρουν από μέτρηση σε μέτρηση αλλά με διαφορά ελάχιστων κλασμάτων του δευτερολέπτου, τόσο μικρή που θεωρείται αμελητέα.

Αναλυτικά τα αποτελέσματα των μετρήσεων του χρόνου εκπαίδευσης και ακρίβειας παραδίδονται παρακάτω.

```
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt

#evaluate the algorithms on their accuracy, training time and classificaion time and present the results on a graph

LRprediction=LR_clf.predict(tfid_x_test)
SVMpred=SVM_clf.predict(tfid_x_test)
LRaccuracy=accuracy_score(LRprediction, y_test)
SVMaccuracy=accuracy_score(SVMpred, y_test)

print("On a metric scale from 0 to 1, the accuracy of the Logistic Regression Algorithm is :", LRaccuracy )
print("On a metric scale from 0 to 1, the accuracy of the SVM Algorithm is :", SVMaccuracy)

print("\n ")

print("The time it took to train the Logistic Regression Algorithm is :", LR_training_time, "seconds")
print("The time it took to train the SVM Algorithm is :", SVM_training_time, "seconds")

print("\n ")
names = ["Logistic Regression","SVM"]
scores = [LRaccuracy,SVMaccuracy]
train_times=[LR_training_time, SVM_training_time]
```

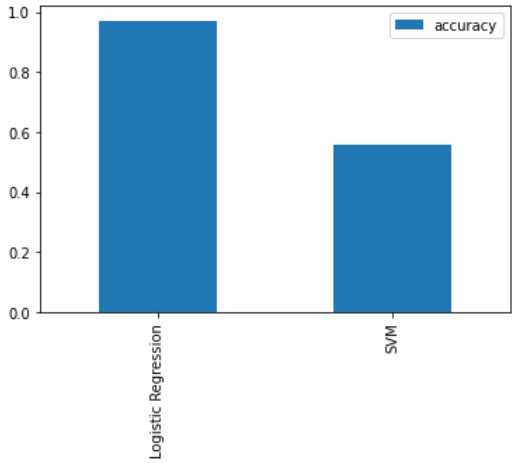
```
On a metric scale from 0 to 1, the accuracy of the Logistic Regression Algorithm is : 0.9710144927536232
On a metric scale from 0 to 1, the accuracy of the SVM Algorithm is : 0.5578342904019689


The time it took to train the Logistic Regression Algorithm is : 0.4878504276275635 seconds
The time it took to train the SVM Algorithm is : 30.18468713760376 seconds
```

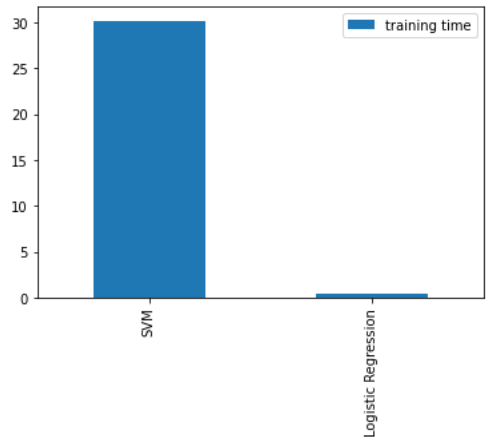
Παραδίδεται ακόμα και η μέτρηση για τον χρόνο εκτέλεσης.

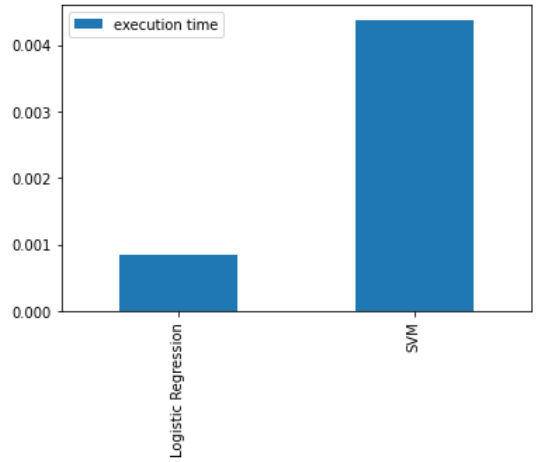
```python
def fake_news_det_LR(df):
    i=0
    exec_time=[]
    for data in df:
        start3=time.time()
        data=[data]
        vectorized_input_data = vect.transform(data)
        prediction = LR_clf.predict(vectorized_input_data)
        stop3=time.time()
        exec_time.append(stop3-start3)
        i=i+1
        if i>=999:
            break
    return exec_time

def fake_news_det_SVM(df):
    i=0
    exec_time=[]
    for data in df:
        start3=time.time()
        data=[data]
        vectorized_input_data = vect.transform(data)
        prediction = SVM_clf.predict(vectorized_input_data)
        stop3=time.time()
        exec_time.append(stop3-start3)
        i=i+1
        if i>=999:
            break
    return exec_time
```

Συναρτήσεις που εκτελούν τις προβλέψεις των classifiers για τις 1000 πρώτες εγγραφές των δεδομένων ελέγχου.

Τέλος υλοποιήθηκε και η μέτρηση των μοντέλων με τη μορφή διαγραμμάτων, χρησιμοποιώντας τη βιβλιοθήκη pandas. Αναλυτικά τα διαγράμματα παραδίδονται παρακάτω.







2.4)Ενσωμάτωση εκπαιδευμένου μοντέλου

Από τις μετρήσεις παρατηρείται ότι η Λογιστική Παλινδρόμηση είναι ο βέλτιστος classifier από τους 2. Επομένως αυτός θα είναι που θα ενσωματωθεί και στην εφαρμογή. Για την ενσωμάτωση χρησιμοποιείται η βιβλιοθήκη **pickle,** που επιτρέπει την αποθήκευση του εκπαιδευμένου μοντέλου σε ένα αρχείο. Μετά τη αποθήκευση του αρχείου που περιέχει το μοντέλο σειρά έχει η υλοποίηση της εφαρμογής.

```
import pickle
#save the Logistic Regression model on a pickle file to use in the web application later
with open('model', 'wb') as f:
    pickle.dump(LR_clf, f)
```

3) Υλοποίηση Εφαρμογής.

Το τελευταίο βήμα που απαιτείται για την επιτυχημένη εκπόνηση της εργασίας. Για την υλοποίηση της εφαρμογής. Χρησιμοποιείται η υπηρεσία Flask για την κατασκευή της, κάνοντας import την αντίστοιχη βιβλιοθήκη. Για την εκτέλεση του κώδικα και την χρήση της εφαρμογής χρησιμοποιείται στο τερματικό η εντολή **run flask,** και η εφαρμογή τρέχει στον localhost. Τέλος για την υλοποίηση της εφαρμογής ακολουθούνται ακριβώς τα ίδια βήματα προ-επεξεργασίας δεδομένων με αυτά που αναφέρθηκαν στην ενότητα 2, με τη διαφορά ότι δεν εφαρμόζεται ξανά εκπαίδευση στον classifier, καθώς για αυτό υπάρχει έτοιμο το αποθηκευμένο μοντέλο classifier, το οποίο και ενσωματώνεται στην εφαρμογή. Για την εμφάνιση της ιστοσελίδας χρησιμοποιείται απλός html κώδικας, που συνδέεται στην εφαρμογή.

```
app = Flask(__name__  , template_folder='html')
#make the vectorizer
vect=TfidfVectorizer(stop_words='english', max_df=0.7)
#load the saved classifier model
loaded_model = pickle.load(open('model', 'rb'))
#open the training dataframe
dataframe = pd.read_csv('train.csv')
#drop the duplicates and missing values
dataframe.drop_duplicates(inplace=True)
dataframe.dropna(axis=0, inplace=True)
x = dataframe['title']+ '' +dataframe['author']
y = dataframe['label']
#split tha data
x_train, x_test,y_train, y_test = train_test_split(x, dataframe['label'], test_size=0.20, random_state=0)
```

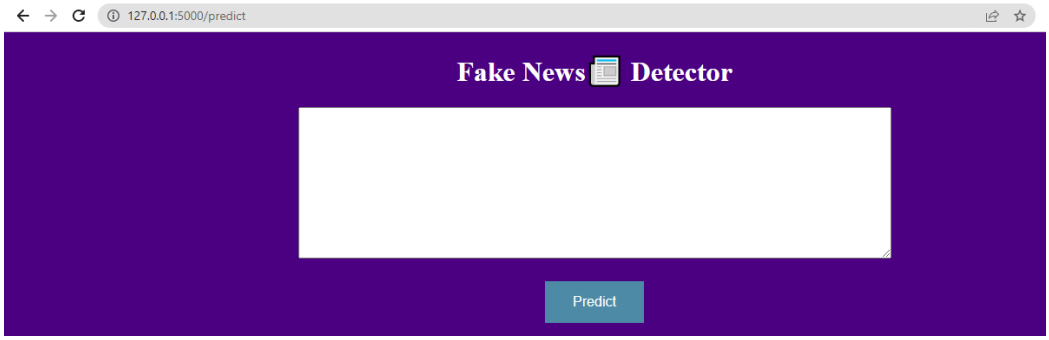
Παρακάτω παραδίδεται η συνάρτηση που χρησιμοποιείται για να γίνονται προβλέψεις από το ενσωματωμένο μοντέλο, παρατηρείται ότι ο τρόπος είναι παρόμοιος με αυτόν που χρησιμοποιήθηκε στην ενότητα 2 για τη μέτρηση του χρόνου εκπαίδευσης.

```
def fake_news_det(news):
    news=str(news)
    tfid_x_train = vect.fit_transform(x_train)
    tfid_x_test = vect.transform(x_test)
    input_data = [news]
    vectorized_input_data = vect.transform(input_data)
    prediction = loaded_model.predict(vectorized_input_data)
    return prediction
```

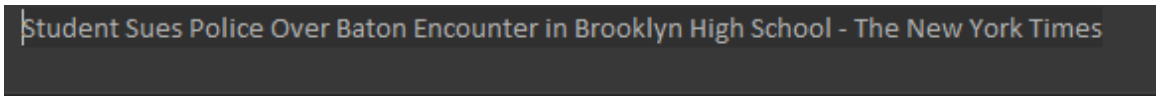
Οι μέθοδοι @app.route αποτελούν τις συνολικές λειτουργίες της ιστοσελίδας. Στη συγκεκριμένη περίπτωση, λόγω απλής ιστοσελίδας που εκτελεί μόνο μία λειτουργία, τους διαχωρισμούς δεδομένων σε αληθείς ή ψευδείς. Χρησιμοποιείται η μέθοδος GET, γιατί η μέθοδος αυτή επιστρέφει ένα αποτέλεσμα στον χρήστη, στη περίπτωση αυτή επιστρέφει την απάντηση της κατηγοριοποίησης της είδησης.

```python
@app.route('/', methods=['POST' , 'GET'])
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST' , 'GET'])
def predict():
    if request.method == 'POST':
        message = request.form['message']
        pred = fake_news_det(message)
        print(pred)
        return render_template('index.html', prediction=pred)
    else:
        return render_template('index.html', prediction="Something went wrong")

if __name__ == '__main__':
    app.run(debug=True)
```

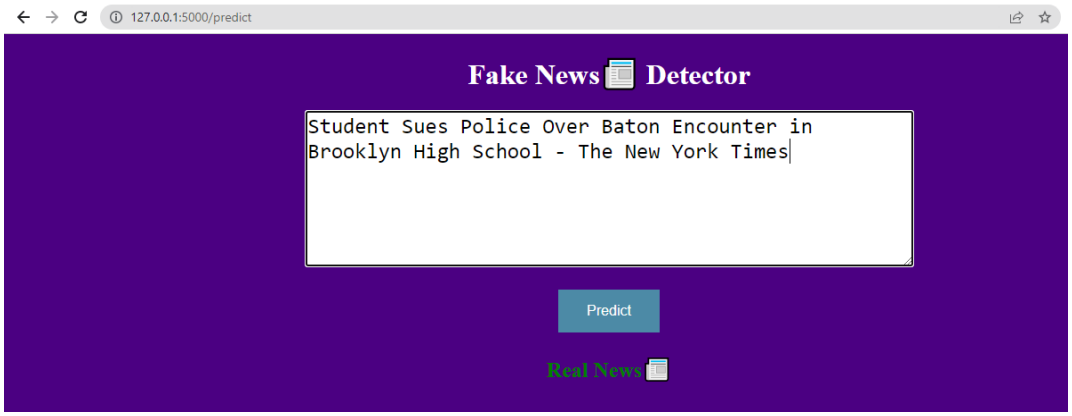
Παρακάτω παραδίδεται η τελική εικόνα της εφαρμογής.



Γίνονται ενδεικτικά μερικές δοκιμές, χρησιμοποιώντας τυχαίες εγγραφές στο σύνολο δεομένων. Παίρνεται αυθαίρετα μία εγγραφή πχ η εγγραφή με id=2065 στο σύνολο δεδομένων. Ο τίτλος της είδησης φαίνεται παρακάτω.



Παρατηρείται ακόμα ότι το label της είδησης είναι 0, άρα η είδηση είναι αληθής. Ο τίτλος της είδησης γίνεται επικόλληση στο textarea της εφαρμογής για να συγκριθεί η πραγματική απάντηση με τη πρόβλεψη του αλγορίθμου που υλοποιήθηκε στην εφαρμογή.



Παρατηρείται ότι η πρόβλεψη της εφαρμογής συμπίπτει με την κατηγοριοποίηση της εγγραφής στο σύνολο δεδομένων. Δοκιμάζεται με τον ίδιο τρόπο παραπάνω φορές (περίπου 15-25), για την επιβεβαίωση ότι η εφαρμογή δουλεύει σωστά.

4) Βιβλιογραφία

[1] scikit-learn: https://scikitlearn.org/stable/install.html
[2] Εισαγωγή στην Εξόρυξη Δεδομένων – Εκδόσεις Τζιόλα
[3] Εξόρυξη και Ανάλυση Δεδομένων-Εκδόσεις Κλειδάριθμος
[4] Logistic Regression : https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[5] SVM : https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[6] https://youtu.be/z_mNVoBcMjM