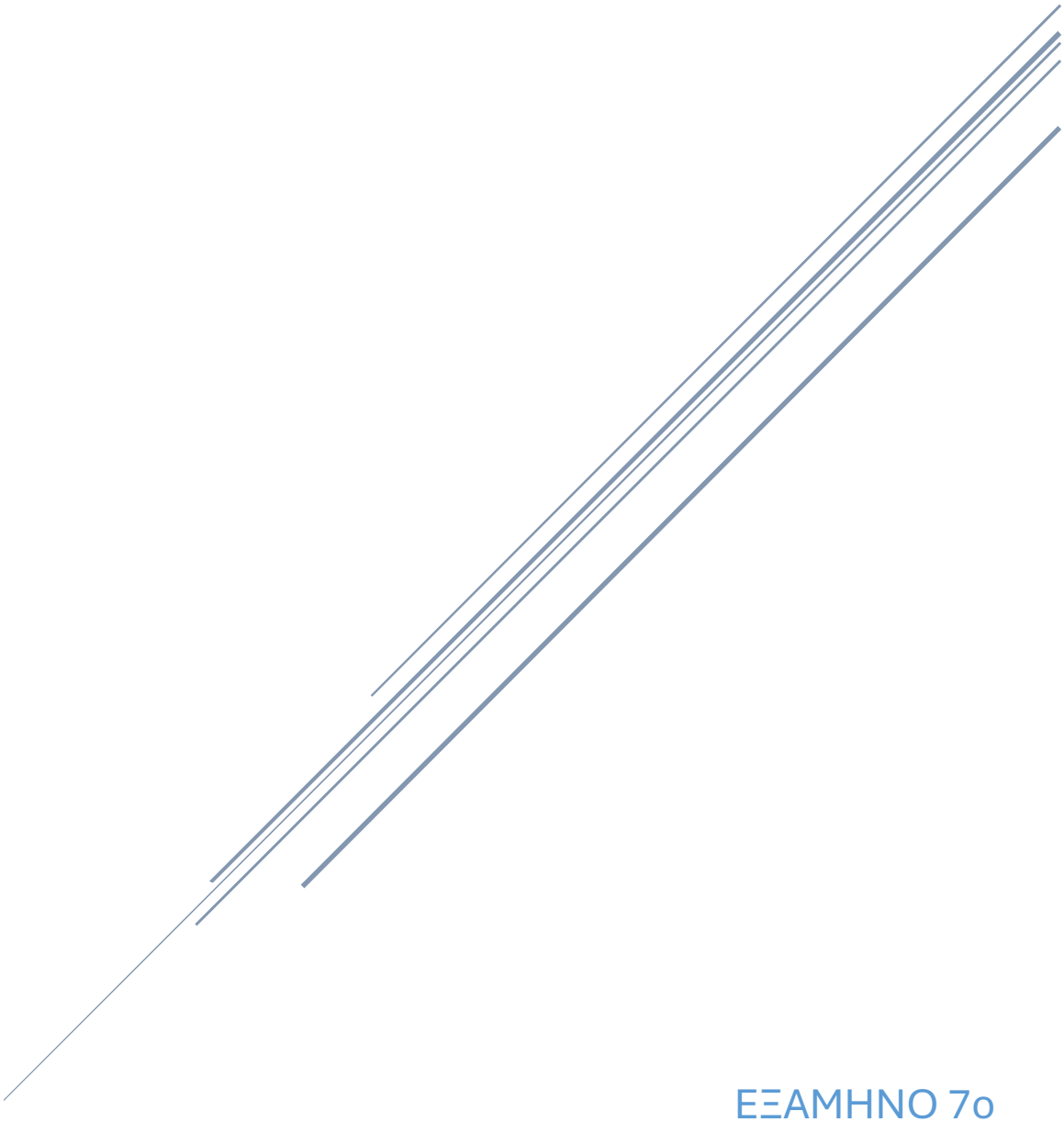
ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΝΑΛΥΣΗΣ ΔΕΔΟΜΕΝΩΝ

ΕΡΓΑΣΙΑ ΕΞΑΜΗΝΟΥ



ΕΞΑΜΗΝΟ 7ο
ΑΛΦΟΝΣΟΣ ΛΕΩΝΙΔΑΣ ΚΩΤΣΙΟΣ Ε21083

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η εργασία αυτή αναπτύχθηκε για το μάθημα Προηγμένα Θέματα Ανάλυσης Δεδομένων. Ο στόχος της εργασίας είναι η ανάπτυξη ενός συστήματος το οποίο θα αναγνωρίζει το συναίσθημα του χρήστη με βάση την κριτική που δίνεται για μία ταινία. Χρησιμοποιήσαμε για δεδομένα το σύνολο δεδομένων IMDB review dataset από το Kaggle. Αναπτύχθηκαν λοιπόν 2 διαφορετικά μοντέλα εκπαίδευσης τα οποία στην συνέχεια τα συγκρίναμε μεταξύ τους με βάση κάποιες μετρικές ώστε να αξιολογήσουμε ποιο είναι το ποιο αποδοτικό . Τέλος αναπτύχθηκε και μια web εφαρμογή στην οποία ενσωματώσαμε τα μοντέλα ώστε να τα δοκιμάσουμε τα αποτελέσματα τους σε νέες κριτικές.

Σε αυτό το documentation θα γίνει ο ορισμός του προβλήματος που μας δόθηκε , η αναλυτική παρουσίαση της ανάπτυξης των μοντέλων , η πειραματική μελέτη που κάναμε ,τα συμπεράσματά μας και η βιβλιογραφία που χρησιμοποιήσαμε.

Ορισμός Προβλήματος

Στην σημερινή εποχή της πληροφορίας , η ανάλυση συναισθήματος έχει αναδειχθεί ως ένα πολύ σημαντικό εργαλείο , ώστε να κατανοούμε τις απόψεις και τα συναισθήματα που κρύβονται πίσω από τον γραπτό λόγο. Στον τομέα λοιπόν της ψυχαγωγίας και του

κινηματογράφου όπου οι χρήστες εκφράζουν τις απόψεις και τα συναισθήματα τους σε κριτικές είναι πολύ σημαντικό να υπάρχει ένας τρόπος να κατηγοριοποιήσουμε αυτές τις κριτικές ώστε να μπορέσουμε , γρήγορα να έχουμε μια γενική εικόνα για την ταινία .

Ο στόχος είναι η εκπαίδευση ενός μοντέλου μηχανικής μάθησης που θα μπορεί να κατηγοριοποιεί τις κριτικές σε θετικές ή αρνητικές. Η επιτυχία του μοντέλου θα αξιολογηθεί με βάση διάφορες μετρικές, όπως η ακρίβεια κατηγοριοποίησης, ο χρόνος εκπαίδευσης και ο χρόνος κατηγοριοποίησης.

Μετά την εκπαίδευση και αξιολόγηση του μοντέλου, θα υλοποιηθεί μια εφαρμογή που θα δέχεται ως είσοδο ένα κείμενο κριτικής και θα επιστρέφει την κατηγορία στην οποία ανήκει το κείμενο, δηλαδή αν είναι θετικό ή αρνητικό. Αυτή η εφαρμογή θα μπορούσε να χρησιμοποιηθεί από πλατφόρμες κριτικών ταινιών για την αυτόματη κατηγοριοποίηση των κριτικών, προσφέροντας έτσι μια πιο άμεση και ακριβή εικόνα των απόψεων των χρηστών.

Παρουσίαση Μοντέλου

Η προσέγγιση που αναπτύχθηκε για την ανάλυση συναισθημάτων από τις κριτικές, βασίζεται σε τεχνικές επεξεργασίας φυσικής γλώσσας και σε τεχνικές ανάλυσης δεδομένων. Παρακάτω θα αναπτύξουμε όλα τα βήματα που ακολουθήσαμε για την ανάπτυξη αυτής της εφαρμογής.

1. Συλλογή Δεδομένων

Τα δεδομένα που συλλέξαμε από το dataset που μας δόθηκε αποτελούνται από 100.000 εγγραφές από τις οποίες οι 25.000 κατηγοριοποιούνται ως **θετικές ('pos'),** οι άλλες 25.000 ως **αρνητικές ('neg')** και οι 50.000 χωρίς να έχουν κατηγοριοποίηση **('unsup').** Αφού αναλύσαμε τις λεπτομέρειες του dataset διαγράψαμε τις εγγραφές οι οποίες δεν είχαν κατηγοριοποίηση και στην συνέχεια μετονομάσαμε τις 2 κατηγορίες σε **pos→1** και **neg → 0** .
(παρακάτω φαίνονται screenshots από τον πηγαίο κώδικά)

ΕΔΩ ΕΙΣΑΓΩ ΤΟ DATASET ΚΑΙ ΒΛΕΠΩ ΤΙΣ ΛΕΠΤΟΜΕΡΕΙΕΣ ΤΟΥ

```
df = pd.read_csv('imdb_master.csv',encoding="latin-1")
df.shape
df.head()
```

[2] ✓ 2.1s Python

	Unnamed: 0	type	review	label	file
0	0	test	Once again Mr. Costner has dragged out a movie...	neg	0_2.txt
1	1	test	This is an example of why the majority of acti...	neg	10000_4.txt
2	2	test	First of all I hate those moronic rappers, who...	neg	10001_1.txt
3	3	test	Not even the Beatles could write songs everyon...	neg	10002_3.txt
4	4	test	Brass pictures (movies is not a fitting word f...	neg	10003_3.txt

```
df.groupby(['label'])[['label']].count()
```

[4] ✓ 0.0s Python

label	label
neg	25000
pos	25000
unsup	50000

```
columns_to_drop = ['Unnamed: 0', 'type',
'file']
for column in columns_to_drop:
    if column in df.columns:
        df = df.drop(column, axis=1)

df.columns = ["review", "label",]
df.head()
```

[5] ✓ 0.0s Python

```
df = df[df.label != 'unsup']
```

[6] ✓ 0.0s Python

```
df.groupby(['label'])[['label']].count()
```

[7] ✓ 0.0s Python

label	label
neg	25000
pos	25000

Αντικαθηστώ τα labels με 0 και 1

```
df.label.replace({'pos':1,'neg':0},inplace=True)
```

[9] ✓ 0.0s Python

2. Προεπεξεργασία Δεδομένων

Για να προχωρήσουμε στην εκπαίδευση των μοντέλων υποβάλαμε τα δεδομένα μας σε επεξεργασία ώστε να βελτιώσουμε την ποιότητα και την συνέπεια των δεδομένων μας. Αρχικά αφαιρέσαμε τα html tags που μπορεί να είχαν τα δεδομένα μας και μετατρέψαμε όλους τους χαρακτήρες σε πεζούς. Στην συνέχεια αφαιρέσαμε τα σημεία στίξεις , χαρακτήρες όπως το # και το @ και οτιδήποτε δεν θα μας χρησίμευε στην

εκπαίδευση μας. Τέλος αφαιρέσαμε ένα σύνολο από stopwords λέξεις δηλαδή οι οποίες δεν προσφέρουν συναίσθημα στο κείμενο μας.

Αφού περάσαμε όλα τα δεδομένα μας από αυτή την επεξεργασία και αφαιρέσαμε της διπλές εγγραφές είμαστε έτοιμοι να περάσουμε στο επόμενο στάδιο της λύσης.

(παρακάτω φαίνονται screenshots από τον πηγαίο κώδικά)

Εδώ γίνεται το preprocess των reviews

```
import nltk
nltk.download('stopwords')
nltk.download('punkt_tab')

stop_words = set(stopwords.words('english')) # Δημιουργεί ένα σύνολο
(set) από stop words στα Αγγλικά.

def process(review):
    review = BeautifulSoup(review).get_text() #αφαιρούνται όλα τα
    HTML tags (π.χ., <br>, <p>)
    review = review.lower()#ολο το κείμενο μετατρέπεται σε πεζά για
    ομοιομορφία
    review = re.sub("[^a-zA-Z]",' ',review)# Αφαιρεί χαρακτήρες που
    δεν είναι γράμματα όπως αριθμούς ,σημεία στίξης
    review = re.sub(r"https\S+|www\S+|http\S+", '', review, flags =
    re.MULTILINE) # Αφαιρεί URLs.
    review = re.sub(r'\@w+|\#', '', review) # Αφαιρεί τα @ και τα #.
    review = re.sub(r'[^\w\s]', '', review)# Αφαιρεί χαρακτήρες
    εκτός από γράμματα και κενά .
    review_tokens = word_tokenize(review) # Διαχωρίζει το κείμενο σε
    ένα πίνακα από λέξεις
    filtered_review = [w for w in review_tokens if not w in
    stop_words] # Αφαιρεί τα stopwords
    return " ".join(filtered_review)
```

[11] ✓ 7.7s Python

ΒΛΕΠΟΥΜΕ ΤΑ DUPLICATES ENTRIES ΚΑΙ ΤΑ ΑΦΑΙΡΟΥΜΕ ΑΠΟ ΤΟ DATASET

```
duplicated_count = df.duplicated().sum()
print("Number of duplicate entries: ",
duplicated_count)
```

[14] ✓ 0.1s Python

```
Number of duplicate entries:  425
```

```
df = df.drop_duplicates('review')
```

[15] ✓ 0.0s Python

```
print(df.shape)
```

[16] ✓ 0.0s Python

```
(49575, 3)
```

```
columns_to_drop = ['word_count']
for column in columns_to_drop:
    if column in df.columns:
        df = df.drop(column, axis=1)

print(df.shape)
df.head()
```

[17] ✓ 0.0s Python

```
(49575, 2)
```

Εφαρμόζουμε το Process στα reviews

```
df.review = df['review'].apply(process)
```

✓ 58.1s Python

```
C:\Users\alfon\AppData\Local\Temp\ipykernel_20752\229932
  review = BeautifulSoup(review).get_text() #αφαιρούνται
```

```
def no_of_words(text):
    words= text.split()
    word_count = len(words)
    return word_count
df['word_count'] = df['review'].apply(no_of_words)
df.head(10)
```

✓ 0.4s Python

3. Αντιπροσώπευση Κειμένου

Για να επεξεργαστεί το μοντέλο μάθησης τα κείμενα μας πρέπει να χρησιμοποιήσουμε την τεχνική **TfidVectorize**r από την βιβλιοθήκη **scikit-learn**.

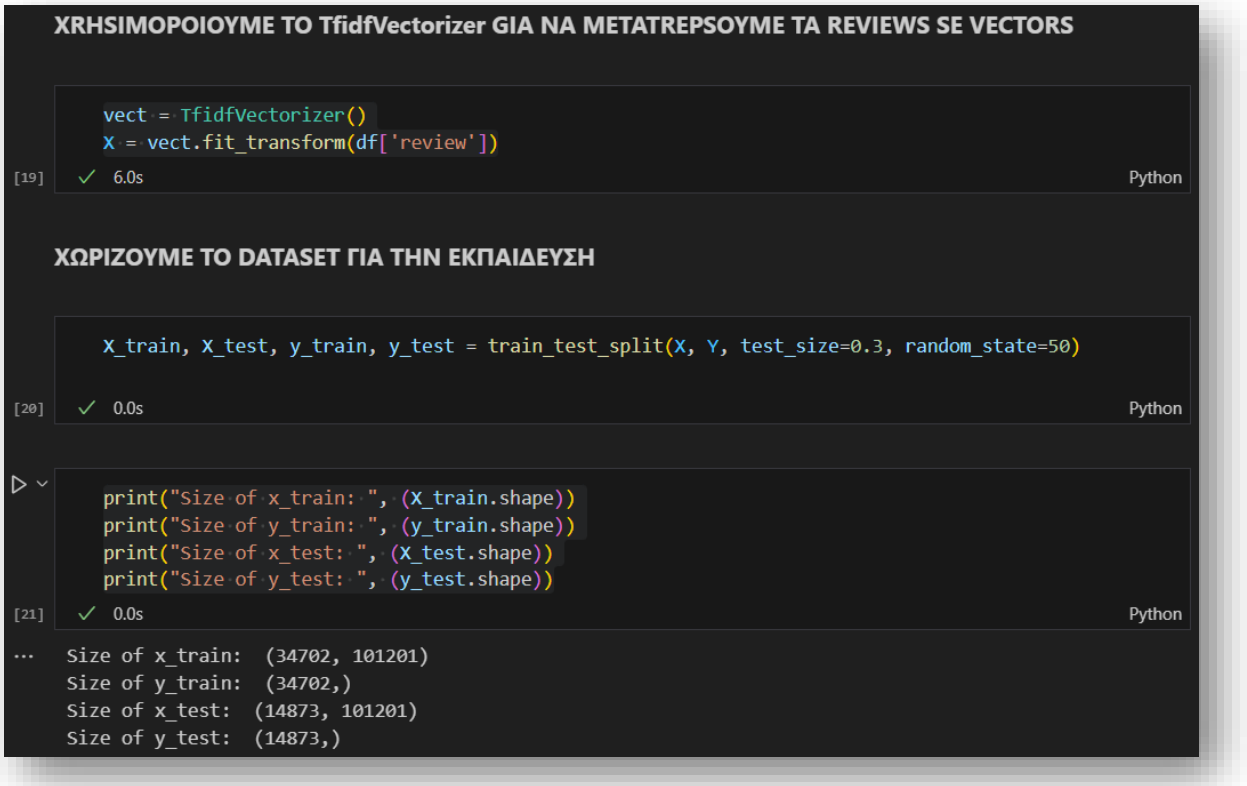
Το **TF-IDF** είναι μια τεχνική επεξεργασίας φυσικής γλώσσας που μετατρέπει το κείμενο σε αριθμητικά χαρακτηριστικά. Υπολογίζει πόσο σημαντική είναι μία λέξη μέσα σε ένα κείμενο , λαμβάνοντας υπόψη πόσο συχνά εμφανίζεται μέσα σε ένα κείμενο.

Το **TfidVectorizer** μετατρέπει το κείμενο σε μορφή κατάλληλη για είσοδο σε μοντέλα μάθησης.

Στην συνέχεια εισάγουμε τα reviews σε αυτή την μέθοδο ώστε να μετασχηματιστούν σε πίνακες όπου κάθε σειρά αντίστοιχή σε ένα κείμενο και κάθε στήλη σε μια λέξη από το λεξιλόγιο.

Τέλος χωρίζουμε τα δεδομένα μας σε **train** και **test data** . Το 30% των δεδομένων θα χρησιμοποιηθούν για testing και το 70% για training.

(παρακάτω φαίνονται screenshots από τον πηγαίο κώδικά)



4. Εκπαίδευση μοντέλων

Αναπτύξαμε 2 διαφορετικά μοντέλα μάθησης ένα **SVM** και ένα **NN νευρωνικό δίκτυο**.

Στο SVM χρησιμοποιήσαμε τον αλγόριθμό λογιστικής παλινδρόμησης **(Logistic Regression)** και βγάλαμε ακρίβεια **89.63%.**

Στο NN χρησιμοποιήσαμε μετά από πειραματική μελέτη ένα νευρωνικό δίκτυο με 3 επίπεδα :

- **DENSE(40 νευρώνες , συνάρτηση ενεργοποίησης 'relu')**
- **DENSE(20 νευρώνες , συνάρτηση ενεργοποίησης 'relu')**
- **DENSE(1 νευρώνες , συνάρτηση ενεργοποίησης 'sigmoid' για την μετατροπή της εξόδου σε 0 και 1)**

Για την εκπαίδευση μετά από πειραματική μελέτη επιλέξαμε **5 εποχές (epochs)** με **batch_size=32**.

Επίσης χρησιμοποιήσαμε ένα **early stopping** δηλαδή αν το val_loss δεν βελτιώνεται για 2 συνεχόμενες εποχές η εκπαίδευση σταματά.

Η ακρίβεια του NN είναι **89.21%**

Στην συνέχεια θα αξιολογήσουμε και θα συγκρίνουμε αναλυτικά τα 2 μοντέλα.

(παρακάτω φαίνονται screenshots από τον πηγαίο κώδικά)

```
from sklearn.svm import LinearSVC
svm_model = LinearSVC()
svm_model.fit(X_train, y_train)
```

[22] ✓ 0.7s

```
LinearSVC()
```

```
SVM Performance:
              precision    recall  f1-score   support

           0       0.90      0.89      0.89      7358
           1       0.89      0.91      0.90      7515

    accuracy                           0.90     14873
   macro avg       0.90      0.90      0.90     14873
weighted avg       0.90      0.90      0.90     14873

SVM Accuracy: 89.63%
```

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

nn_model = Sequential([
    Dense(40, activation='relu', input_shape=(X_train.shape
    [1],)),
    Dropout(0.4),
    Dense(20, activation='relu'),
    Dropout(0.4),
    Dense(1, activation='sigmoid')
])

nn_model.compile(optimizer='adam',
loss='binary_crossentropy', metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss', patience=2)

history = nn_model.fit(X_train, y_train, epochs=5,
batch_size=32, validation_data=(X_test, y_test), verbose=1,
callbacks=[early_stopping])

nn_loss, nn_accuracy = nn_model.evaluate(X_test, y_test,
verbose=0)
print("NN Accuracy: ", nn_accuracy)
```

✓ 4m 19.4s Python

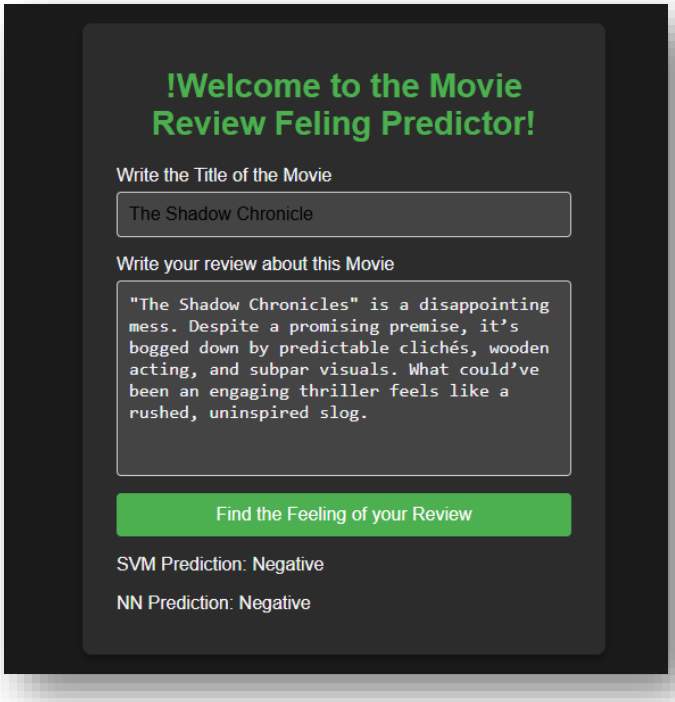
```
Epoch 1/5
1085/1085 ━━━━━━━━━━━━━━━━━━━━ 78s 70ms/step - accuracy: 0.7907 - loss: 0.4646 - val_accuracy: 0.9021 - val_loss: 0.2470
Epoch 2/5
1085/1085 ━━━━━━━━━━━━━━━━━━━━ 81s 75ms/step - accuracy: 0.9550 - loss: 0.1456 - val_accuracy: 0.8991 - val_loss: 0.2693
Epoch 3/5
1085/1085 ━━━━━━━━━━━━━━━━━━━━ 88s 81ms/step - accuracy: 0.9800 - loss: 0.0715 - val_accuracy: 0.8922 - val_loss: 0.3330
NN Accuracy:  0.8921535611152649
```

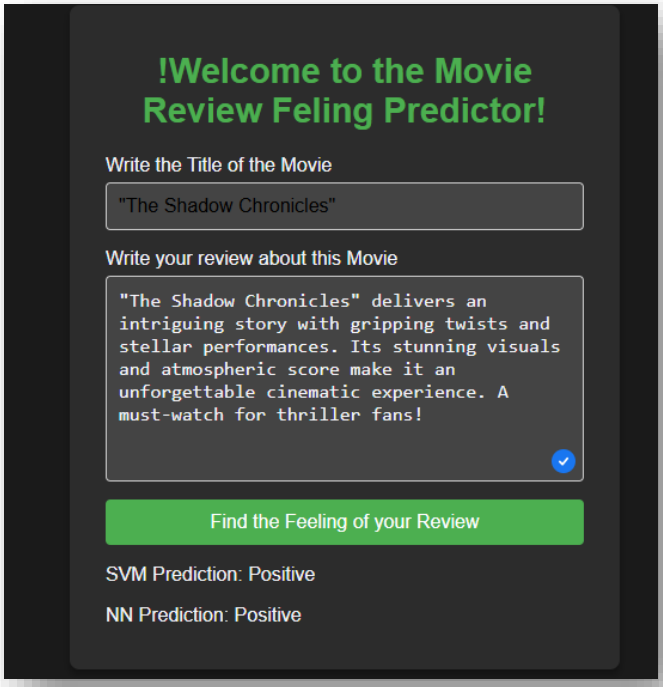
5. Αποθήκευση μοντέλων και Ανάπτυξη εφαρμογής

Αφού εκπαιδεύσαμε τα μοντέλα τα αποθηκεύσαμε μέσω της βιβλιοθήκης **pickle** ώστε στην συνέχεια να αναπτύξουμε την web εφαρμογή μας.

Η web εφαρμογή μας αναπτύχθηκε μέσω της **Flask** στην οποία εισάγουμε τα αποθηκευμένα μοντέλα μας δημιουργούμε μια **html** σελίδα στην οποία ο χρήστης θα μπορεί να εισάγει μία νέα δικιά του κριτική και να βλέπει αν η κριτική που έγραψε είναι θετική ή αρνητική.

(παρακάτω φαίνονται screenshots από την HTML σελίδα)





Αξιολόγηση Μοντέλων

Για να αξιολογήσουμε τα μοντέλα και να τα συγκρίνουμε θα χρησιμοποιήσουμε μετρικές όπως την ποιότητα του μοντέλου (Accuracy,Precision,Recall,F1),τον χρόνο εκπαίδευσης και τον χρόνο εκτέλεσης για νέα κείμενα.

Ας αρχίσουμε με την ποιότητα των μοντέλων παρακάτω φαίνεται ο αναλυτικός πίνακας των αποτελεσμάτων για τα 2 μοντέλα:

	SVM	**NN**
Accuracy:	**89.63%**	**89.22%**
Για κατ: **0**		
Precision:	0.90	0.90
Recall:	0.89	0.88
F1-Score:	0.89	0.89
Για κατ: **1**		
Precision:	0.89	0.88
Recall	0.91	0.91
F1-Score	0.90	0.89

Από τα παραπάνω αποτελέσματα παρατηρούμε ότι τα 2 μοντέλα είναι ισορροπημένα στα αποτελέσματα τους στο Precision και στο Recall και στις 2 κατηγορίες. Επίσης παρουσιάζουν μια παρόμοια απόδοση αλλά το SVM παρουσιάζει λίγο καλύτερο **Accuracy** (89.63% έναντι του 89.22%).

Στην συνέχεια ο χρόνος εκπαίδευσης των 2 μοντέλων είναι ευδιάκριτα πολύ διαφορετικός.

	SVM	**NN**
Χρόνος Εκπαίδευσης	0.7s	4m 19s

Η διαφορά αυτή δημιουργείται καθώς το SVM δεν χρειάζεται εποχές για την λειτουργεία του και επίσης είναι πολύ γρήγορο σε δεδομένα μεσαίου μεγέθους.

Ενώ το NN λόγω της αρχιτεκτονικής του , των πολλαπλών επιπέδων εκπαίδευσης και των 5 εποχών εκπαίδευσης χρειάζεται πολύ περισσότερο χρόνο.

Τέλος για να μετρήσουμε τον χρόνο εκτέλεσης, απαιτείται να εκτιμηθεί ο μέσος χρόνος που απαιτείται για την ταξινόμηση ενός κειμένου από το σύνολο δεδομένων ελέγχου.

- **SVM**: Ένα SVM συνήθως εκτελεί γρήγορα τις προβλέψεις, ειδικά όταν χρησιμοποιούμε έναν γραμμικό ταξινομητή.

- **NN**: Το NN είναι λίγο πιο αργό στην πρόβλεψη, καθώς περιλαμβάνει τη μετάδοση δεδομένων μέσα από πολλαπλά επίπεδα.

Η τελική σύγκριση λοιπόν φαίνεται στον παρακάτω πίνακα :

Κριτήριο	**SVM**	**NN**
Accuracy:	89.63%	89.22%
Precision/Recall/F1:	Παρόμοια(λίγο καλύτερο)	Παρόμοια
Χρόνος Εκπαίδευσης:	Γρηγορότερο(0.7s)	Πιο Αργό(4m19s)
Χρόνος Εκτέλεσης:	Γρηγορότερο	Πιο Αργό

Παρατηρούμε λοιπόν πως το μοντέλο SVM είναι καλύτερη επιλογή από το NN.

Συμπεράσματα

Μετά την ανάλυση που κάναμε είδαμε πως η μηχανική μάθηση και η ανάλυση δεδομένων με επιτυχία λύνουν προβλήματα όπως αυτό που θέσαμε δηλαδή την ανάλυση του συναισθήματος που έχει ένα κείμενο. Αξιολογήσαμε λοιπόν πως ένα SVM μοντέλο αποτελεί μια πιο αποτελεσματική λύση καθώς το πρόβλημά μας είχε δεδομένα χαμηλής πολυπλοκότητας. Τα νευρονικά δίκτυα είναι πιο χρήσιμα σε προβλήματα με καλύτερη γενίκευση και πολύπλοκα σύνολα δεδομένων, αλλά όπως είδαμε και αυτό πέρα από τον χρόνο εκπαίδευσής του τα πήγε πολύ καλά στην ανάλυση του συναισθήματος. Όμως όπως προαναφέρθηκε το SVM ξεχωρίζει σε προβλήματα δυαδικής αναζήτησης.

Βιβλιογραφία

TFIDF
https://scikit-learn.org/1.5/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

KAGGLE NOTEBOOKS
https://www.kaggle.com/datasets/utathya/imdb-review-dataset/code

ΔΙΑΦΑΝΕΙΕΣ % ΕΡΓΑΣΤΗΡΙΑ ΜΑΘΗΜΑΤΟΣ