PROJECT REPORT

SENTIMENTAL ANALYSIS OF IMDb MOVIE REVIEWS

BY

A.LOHIT - 180330004
K.SRI CHARITHA – 180330027
K.JAYATHI - 180330003
K.VAISHNAVI-180330026

Table of contents:

INTRODUCTION:

SENTIMENT ANALYSIS ON IMDb MOVIE REVIEWS:

Sentiment analysis is a natural language processing problem where the intent of a movie is predicted using the reviews .
Here, a IMDB dataset containing around 25,000 reviews(good /bad)is taken and is trained and tested using different deep learning libraries like tensor flow, keras, matplot lib(for plotting)etc. Here we use multi-layer perceptron model where it consists of 3 layers, Input layer for intialisation of data, hidden layer for computing the data and output layer to produce the results.
Tensorflow is an open source library especially used in neural networks for perception,

prediction and creation of data .we can develop, train and build any complex models using this platform .It is a math library and can perform different operations on multi-dimensional data arrays.
Keras is a neural network library which basically is designed for faster experimentation with the deep neural networks ,it is modular and extensible platform .This platform is user-friendly and allows us to try different ideas. it supports multiple back-end neural network computation.

LITERATURE SURVEY

AUTHORS

1.Byran Tan
2.Ankit Goyal and Amey Parulekar
3.Lakshmi pathi N
4.Angcar Li
5.Mohammad Wasil

- https://www.kaggle.com/lakshmi25npathi/sentiment-analysis-of-imdb-movie-reviews
- https://cseweb.ucsd.edu/classes/wi15/cse255-a/reports/fa15/003.pdf
- https://towardsdatascience.com/imdb-reviews-or-8143fe57c825
- https://www.andrew.cmu.edu/user/angli2/li2019sentiment.pdf
- https://github.com/MohammadWasil/Sentiment-Analysis-IMDb-Movie-Review

REFERENCE:

- Sentiment Analysis – Wikipedia – https://en.wikipedia.org/wiki/Sentiment_analysis
- Natural Language Processing from Scratch - http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/35671.pdf
- Pang, Bo; Lee, Lillian; Vaithyanathan, Shivakumar (2002). "Thumbs up? Sentiment Classification using Machine Learning Techniques". Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).
- NLTK Stopwords Corpus: http://www.nltk.org/book/ch02.html

The IMDB data set reviews are entrenched by gathering data from Kaggle.

- We evaluated the model using confusion matrix

- We used word cloud to plot the negative and positive words from the reviews.
- We tried different algorithms to predict the accuracy and Using LTSM model we predicted the review is positive(1) or Negative (0).

NOVELTY OF THE WORK

SENTIMENT ANALYSIS ON IMDb MOVIE REVIEWS:

we have framed this analysis as natural language accessing problem. The input layer consists of over 25,000 reviews, the output layer consists of the epoch values for every single data and gives the accuracy.

CODE:

At first ,we have trained the data, displayed the records and classifies the reviews as good(0) or bad(1).we have checked the average length of a review and plotted it.

Using the keras library we have developed a multi-layer perceptron model with 250 hidden layers which compute the input layer data and 1 output layer which reviews the movie as good or bad

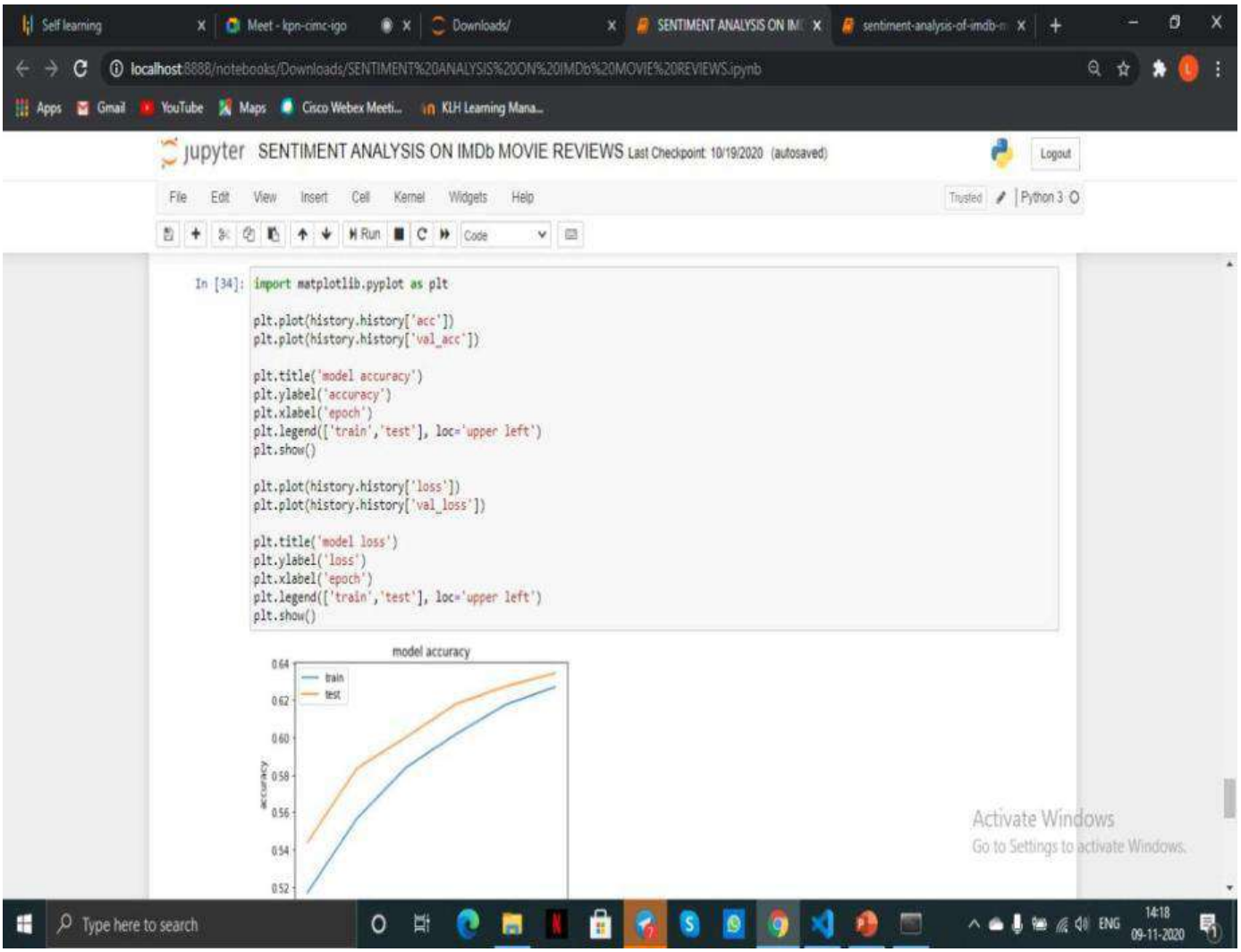
For final evaluation of the model, we have used the fit method by defining the batch size and number of epochs for the model.

So by using these different machine learning algorithms we have got the accuracy as shown below

So here we can clearly observe that Logistic regression and SVM gave the best accuracy compared to the others algorithms.

Algorithm	Accuracy
Logistic Regression	86.9
Naïve baiyes	82.1
Random forest	80.8
Support Vector Machine	87.0
Ensemble	84.9

IMPLEMENTATION WITH CODE



jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

```
imdb_data=pd.read_csv('../Input/IMDB Dataset.csv')
print(imdb_data.shape)
imdb_data.head(10)
```

```
(50000, 2)
```

Out[2]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive

Exploratery data analysis

In [3]:
```
#Summary of the dataset
imdb_data.describe()
```

Out[3]:

	review	sentiment
count	50000	50000
unique	49582	2
top	Loved today's show!!! It was a variety and not ...	positive

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

In [4]:
```
#sentiment count
imdb_data['sentiment'].value_counts()
```

Out[4]:
```
positive    25000
negative    25000
Name: sentiment, dtype: int64
```

We can see that the dataset is balanced.

Spliting the training dataset

In [5]:
```
#split the dataset
#train dataset
train_reviews=imdb_data.review[:40000]
train_sentiments=imdb_data.sentiment[:40000]
#test dataset
test_reviews=imdb_data.review[40000:]
test_sentiments=imdb_data.sentiment[40000:]
print(train_reviews.shape,train_sentiments.shape)
print(test_reviews.shape,test_sentiments.shape)
```

```
(40000,) (40000,)
(10000,) (10000,)
```

Text normalization

In [6]:
```
#Tokenization of text
tokenizer=ToktokTokenizer()
#Setting English stopwords
stopword_list=nltk.corpus.stopwords.words('english')
```

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Removing html strips and noise text

In [7]:
```
#Removing the html strips
def strip_html(text):
    soup = BeautifulSoup(text, "html.parser")
    return soup.get_text()

#Removing the square brackets
def remove_between_square_brackets(text):
    return re.sub('\[[^]]*\]', '', text)

#Removing the noisy text
def denoise_text(text):
    text = strip_html(text)
    text = remove_between_square_brackets(text)
    return text
#Apply function on review column
imdb_data['review']=imdb_data['review'].apply(denoise_text)
```

Removing special characters

In [8]:
```
#Define function for removing special characters
def remove_special_characters(text, remove_digits=True):
    pattern=r'[^a-zA-z0-9\s]'
    text=re.sub(pattern,'',text)
    return text
#Apply function on review column
imdb_data['review']=imdb_data['review'].apply(remove_special_characters)
```

"Text stemming "

Type here to search

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

" Text stemming "

In [9]:
```
#Stemming the text
def simple_stemmer(text):
    ps=nltk.porter.PorterStemmer()
    text= ' '.join([ps.stem(word) for word in text.split()])
    return text
#Apply function on review column
imdb_data['review']=imdb_data['review'].apply(simple_stemmer)
```

Removing stopwords

In [10]:
```
#set stopwords to english
stop=set(stopwords.words('english'))
print(stop)

#removing the stopwords
def remove_stopwords(text, is_lower_case=False):
    tokens = tokenizer.tokenize(text)
    tokens = [token.strip() for token in tokens]
    if is_lower_case:
        filtered_tokens = [token for token in tokens if token not in stopword_list]
    else:
        filtered_tokens = [token for token in tokens if token.lower() not in stopword_list]
    filtered_text = ' '.join(filtered_tokens)
    return filtered_text
#Apply function on review column
imdb_data['review']=imdb_data['review'].apply(remove_stopwords)
```

```
{'those', 'she', 'should', 'you', 'won', "hadn't", 'will', 'whom', 'yourself', 'y', 'below', 'down', "shan't", 'we', 'your', 'o
ff', 'haven', 'were', 'only', "doesn't", 'no', 'from', "couldn't", 'of', "hasn't", 'very', "she's", 'hers', 'hadn', 'wasn', 'ou
rselves', "won't", 'what', 'are', 'a', 'its', 'o', 'been', 'once', 'mustn', 'after', 'not', "that'll", 'this', 'by', 'shan', "s
houldn't", 'themselves', 't', 'was', 'me', 'herself', 'ma', "mightn't", 'shouldn', 'if', 'and', 're', "needn't", "you'll", 'the
```

Type here to search

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
'himself', 'ours', 'but', "haven't", "aren't", 'against', 'wouldn', 'at', "mustn't", 'between', 'where', 'both', 'him', 'aren',
'he', 'weren', 'or', 'why', 'it', 'further', 'most', 'myself', 'isn', 'his'}
```

Normalized train reviews

In [11]:
```
#normalized train reviews
norm_train_reviews=imdb_data.review[:40000]
norm_train_reviews[0]
#convert dataframe to string
#norm_train_string=norm_train_reviews.to_string()
#Spelling correction using Textblob
#norm_train_spelling=TextBlob(norm_train_string)
#norm_train_spelling.correct()
#Tokenization using Textblob
#norm_train_words=norm_train_spelling.words
#norm_train_words
```

Out[11]:
```
'one review ha mention watch 1 Oz episod youll hook right thi exactli happen meth first thing struck Oz wa brutal unflinch scen
e violenc set right word GO trust thi show faint heart timid thi show pull punch regard drug sex violenc hardcor classic use wo
rdit call OZ nickNam given oswald maximum secur state penitentari focus mainli emerald citi experiment section prison cell glas
s front face inward privaci high agenda Em citi home manyaryan muslim gangsta latino christian italian irish moreso scuffl deat
h stare dodgi deal shadi agreement never far awayi would say main appeal show due fact goe show wouldnt dare forget pretti pict
ur paint mainstream audienc forget charm forget romanceoz doesnt mess around first episod ever saw struck nasti wa surreal coul
dnt say wa readi watch develop tast Oz got accustom high level graphic violenc violenc injustic crook guard wholl sold nickel 1
nmat wholl kill order get away well manner middl class inmat turn prison bitch due lack street skill prison experi watch Oz may
becom comfort uncomfort viewingthat get touch darker side'
```

Normalized test reviews

In [12]:
```
#Normalized test reviews
norm_test_reviews=imdb_data.review[40000:]
norm_test_reviews[45005]
```


jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
norm_test_reviews=imdb_data.review[40000:]
norm_test_reviews[45005]
#convert dataframe to string
#norm_test_string=norm_test_reviews.to_string()
#spelling correction using Textblob
#norm_test_spelling=TextBlob(norm_test_string)
#print(norm_test_spelling.correct())
#Tokenization using Textblob
#norm_test_words=norm_test_spelling.words
#norm_test_words
```

Out[12]:
```
'read review watch thi piec cinemat garbag took least 2 page find somebodi els didnt think thi appallingli unfunni montag wasnt
acm humour 70 inde ani era thi isnt least funni set sketch comedi ive ever seen itll till come along half skit alreadi done inf
init better act monti python woodi allen wa say nice piec anim last 90 second highlight thi film would still get close sum mind
less drivelridden thi wast 75 minut semin comedi onli world semin realli doe mean semen scatolog humour onli world scat actual
fece precursor joke onli mean thi handbook comedi tit bum odd beaver niceif pubesc boy least one hand free hevent found playboy
exist give break becaus wa earli 70 way sketch comedi go back least ten year prior onli way could even forgiv thi film even med
e wa gunpoint retro hardli sketch clown subtli pervert children may cut edg circl could actual funni come realli quit sad kept
go throughout entir 75 minut sheer belief may save genuin funni skit end gave film 1 becaus wa lower scoreand onli recommend in
somniac coma patientsor perhap peopl suffer lockjawtheir jaw would final drop open disbelief'
```

"Bags of words model "

It is used to convert text documents to numerical vectors or bag of words

In [31]:
```
#Count vectorizer for bag of words
cv=CountVectorizer(min_df=0,max_df=1,binary=False,ngram_range=(1,3))
#transformed train reviews
cv_train_reviews=cv.fit_transform(norm_train_reviews)
#transformed test reviews
cv_test_reviews=cv.transform(norm_test_reviews)

print('BOW_cv_train:',cv_train_reviews.shape)
print('BOW_cv_test:',cv_test_reviews.shape)
```

Activate Windows
Go to Settings to activate Windows.

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

```
cv=CountVectorizer(min_df=0,max_df=1,binary=False,ngram_range=(1,3))
#transformed train reviews
cv_train_reviews=cv.fit_transform(norm_train_reviews)
#transformed test reviews
cv_test_reviews=cv.transform(norm_test_reviews)

print('BOW_cv_train:',cv_train_reviews.shape)
print('BOW_cv_test:',cv_test_reviews.shape)
#vocab=cv.get_feature_names()-toget feature names
```

```
BOW_cv_train: (40000, 6209089)
BOW_cv_test: (10000, 6209089)
```

Term Frequency-Inverse Document Frequency model (TFIDF)

It is used to convert text documents to matrix of tfidf features.

In [18]:
```
#Tfidf vectorizer
tv=TfidfVectorizer(min_df=0,max_df=1,use_idf=True,ngram_range=(1,3))
#transformed train reviews
tv_train_reviews=tv.fit_transform(norm_train_reviews)
#transformed test reviews
tv_test_reviews=tv.transform(norm_test_reviews)
print('Tfidf_train:',tv_train_reviews.shape)
print('Tfidf_test:',tv_test_reviews.shape)
```

```
Tfidf_train: (40000, 6209089)
Tfidf_test: (10000, 6209089)
```

Labeling the sentiment text

In [14]:
```
#labeling the sentient data
lb=LabelBinarizer()
```


jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

```
Tfidf_train: (40000, 6209089)
Tfidf_test: (10000, 6209089)
```

Labeling the sentiment text

In [14]:
```
#labeling the sentient data
lb=LabelBinarizer()
#transformed sentiment data
sentiment_data=lb.fit_transform(imdb_data['sentiment'])
print(sentiment_data.shape)
```

```
(50000, 1)
```

Split the sentiment tdata

In [16]:
```
#Spliting the sentiment data
train_sentiments=sentiment_data[:40000]
test_sentiments=sentiment_data[40000:]
print(train_sentiments)
print(test_sentiments)
```

```
[[1]
 [1]
 [1]
 ...
 [1]
 [0]
 [0]]
[[0]
 [0]
 [0]
 ...
```

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

```
#Fitting the model for Bag of words
lr_bow=lr.fit(cv_train_reviews,train_sentiments)
print(lr_bow)
#Fitting the model for tfidf features
lr_tfidf=lr.fit(tv_train_reviews,train_sentiments)
print(lr_tfidf)
```

```
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=500,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=42, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=500,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=42, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

Logistic regression model performane on test dataset

In [33]:
```
#Predicting the model for bag of words
lr_bow_predict=lr.predict(cv_test_reviews)
print(lr_bow_predict)
##Predicting the model for tfidf features
lr_tfidf_predict=lr.predict(tv_test_reviews)
print(lr_tfidf_predict)
```

```
[0 0 0 ... 0 1 1]
[0 0 0 ... 0 1 1]
```

Accuracy of the model

In [34]:
```
#Accuracy score for bag of words
```


jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

In [34]:
```
#Accuracy score for bag of words
lr_bow_score=accuracy_score(test_sentiments,lr_bow_predict)
print("lr_bow_score :",lr_bow_score)
#Accuracy score for tfidf features
lr_tfidf_score=accuracy_score(test_sentiments,lr_tfidf_predict)
print("lr_tfidf_score :",lr_tfidf_score)
```

```
lr_bow_score : 0.7512
lr_tfidf_score : 0.75
```

Print the classification report

In [35]:
```
#Classification report for bag of words
lr_bow_report=classification_report(test_sentiments,lr_bow_predict,target_names=['Positive','Negative'])
print(lr_bow_report)

#Classification report for tfidf features
lr_tfidf_report=classification_report(test_sentiments,lr_tfidf_predict,target_names=['Positive','Negative'])
print(lr_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.75      0.75      0.75      4993
    Negative       0.75      0.75      0.75      5007

    accuracy                           0.75     10000
   macro avg       0.75      0.75      0.75     10000
weighted avg       0.75      0.75      0.75     10000

              precision    recall  f1-score   support

    Positive       0.74      0.77      0.75      4993
    Negative       0.76      0.73      0.75      5007
```

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Confusion matrix

In [36]:
```
#confusion matrix for bag of words
cm_bow=confusion_matrix(test_sentiments,lr_bow_predict,labels=[1,0])
print(cm_bow)
#confusion matrix for tfidf features
cm_tfidf=confusion_matrix(test_sentiments,lr_tfidf_predict,labels=[1,0])
print(cm_tfidf)
```

```
[[3768 1239]
 [1249 3744]]
[[3663 1344]
 [1156 3837]]
```

Stochastic gradient descent or Linear support vector machines for bag of words and tfidf features

In [38]:
```
#training the linear svm
svm=SGDClassifier(loss='hinge',max_iter=500,random_state=42)
#fitting the svm for bag of words
svm_bow=svm.fit(cv_train_reviews,train_sentiments)
print(svm_bow)
#fitting the svm for tfidf features
svm_tfidf=svm.fit(tv_train_reviews,train_sentiments)
print(svm_tfidf)
```

```
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=True,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge',
              max_iter=500, n_iter_no_change=5, n_jobs=None, penalty='l2',
              power_t=0.5, random_state=42, shuffle=True, tol=0.001,
              validation_fraction=0.1, verbose=0, warm_start=False)
SGDClassifier(alpha=0.0001, average=False, class_weight=None,
```

Self learning | Meet - kpn-cimc-igo | Downloads/ | sentiment-analysis-of-imdb-mo

localhost:8888/notebooks/Downloads/sentiment-analysis-of-imdb-movie-reviews.ipynb

Apps | Gmail | YouTube | Maps | Cisco Webex Meeti... | KLH Learning Mana...

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
print(svm_bow_predict)
#Predicting the model for tfidf features
svm_tfidf_predict=svm.predict(tv_test_reviews)
print(svm_tfidf_predict)
```

```
[1 1 0 ... 1 1 1]
[1 1 1 ... 1 1 1]
```

Accuracy of the model

In [40]:
```
#Accuracy score for bag of words
svm_bow_score=accuracy_score(test_sentiments,svm_bow_predict)
print("svm_bow_score :",svm_bow_score)
#Accuracy score for tfidf features
svm_tfidf_score=accuracy_score(test_sentiments,svm_tfidf_predict)
print("svm_tfidf_score :",svm_tfidf_score)
```

```
svm_bow_score : 0.5829
svm_tfidf_score : 0.5112
```

Print the classification report

In [41]:
```
svm_bow_report=classification_report(test_sentiments,svm_bow_predict,target_names=['Positive','Negative'])
print(svm_bow_report)
svm_tfidf_report=classification_report(test_sentiments,svm_tfidf_predict,target_names=['Positive','Negative'])
print(svm_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.94      0.18      0.30      4993
    Negative       0.55      0.99      0.70      5007

    accuracy                           0.58     10000
   macro avg       0.74      0.58      0.50     10000
```

Activate Windows
Go to Settings to activate Windows.

Type here to search | ENG | 14:17 09-11-2020

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

Print the classification report

In [41]:
```
svm_bow_report=classification_report(test_sentiments,svm_bow_predict,target_names=['Positive','Negative'])
print(svm_bow_report)
svm_tfidf_report=classification_report(test_sentiments,svm_tfidf_predict,target_names=['Positive','Negative'])
print(svm_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.94      0.18      0.30      4993
    Negative       0.55      0.99      0.70      5007

    accuracy                           0.58     10000
   macro avg       0.74      0.58      0.50     10000
weighted avg       0.74      0.58      0.50     10000

              precision    recall  f1-score   support

    Positive       1.00      0.02      0.04      4993
    Negative       0.51      1.00      0.67      5007

    accuracy                           0.51     10000
   macro avg       0.75      0.51      0.36     10000
weighted avg       0.75      0.51      0.36     10000
```

Plot the confusion matrix

In [42]:
```
#confusion matrix for bag of words
cm_bow=confusion_matrix(test_sentiments,svm_bow_predict,labels=[1,0])
print(cm_bow)
#confusion matrix for tfidf features
```


jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

```
[[5007    0]
 [4888  105]]
```

Multinomial Naive Bayes for bag of words and tfidf features

In [43]:
```
#training the model
mnb=MultinomialNB()
#fitting the svm for bag of words
mnb_bow=mnb.fit(cv_train_reviews,train_sentiments)
print(mnb_bow)
#fitting the svm for tfidf features
mnb_tfidf=mnb.fit(tv_train_reviews,train_sentiments)
print(mnb_tfidf)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Model performance on test data

In [44]:
```
#Predicting the model for bag of words
mnb_bow_predict=mnb.predict(cv_test_reviews)
print(mnb_bow_predict)
#Predicting the model for tfidf features
mnb_tfidf_predict=mnb.predict(tv_test_reviews)
print(mnb_tfidf_predict)
```

```
[0 0 0 ... 0 1 1]
[0 0 0 ... 0 1 1]
```

Accuracy of the model

In [45]:
```
#Accuracy score for bag of words
```

Activate Windows
Go to Settings to activate Windows.

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

```
[0 0 0 ... 0 1 1]
[0 0 0 ... 0 1 1]
```

Accuracy of the model

In [45]:
```
#Accuracy score for bag of words
mnb_bow_score=accuracy_score(test_sentiments,mnb_bow_predict)
print("mnb_bow_score :",mnb_bow_score)
#Accuracy score for tfidf features
mnb_tfidf_score=accuracy_score(test_sentiments,mnb_tfidf_predict)
print("mnb_tfidf_score :",mnb_tfidf_score)
```

```
mnb_bow_score : 0.751
mnb_tfidf_score : 0.7509
```

Print the classification report

In [46]:
```
#Classification report for bag of words
mnb_bow_report=classification_report(test_sentiments,mnb_bow_predict,target_names=['Positive','Negative'])
print(mnb_bow_report)
#Classification report for tfidf features
mnb_tfidf_report=classification_report(test_sentiments,mnb_tfidf_predict,target_names=['Positive','Negative'])
print(mnb_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.75      0.76      0.75      4993
    Negative       0.75      0.75      0.75      5007

    accuracy                           0.75     10000
   macro avg       0.75      0.75      0.75     10000
weighted avg       0.75      0.75      0.75     10000
```

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

mnb_bow_score : 0.751
mnb_tfidf_score : 0.7509

Print the classification report

In [46]:
```
#Classification report for bag of words
mnb_bow_report=classification_report(test_sentiments,mnb_bow_predict,target_names=['Positive','Negative'])
print(mnb_bow_report)
#Classification report for tfidf features
mnb_tfidf_report=classification_report(test_sentiments,mnb_tfidf_predict,target_names=['Positive','Negative'])
print(mnb_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.75      0.76      0.75      4993
    Negative       0.75      0.75      0.75      5007

    accuracy                           0.75     10000
   macro avg       0.75      0.75      0.75     10000
weighted avg       0.75      0.75      0.75     10000
```

```
              precision    recall  f1-score   support

    Positive       0.75      0.76      0.75      4993
    Negative       0.75      0.74      0.75      5007

    accuracy                           0.75     10000
   macro avg       0.75      0.75      0.75     10000
weighted avg       0.75      0.75      0.75     10000
```

Plot the confusion matrix

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

Plot the confusion matrix

In [47]:
```
#confusion matrix for bag of words
cm_bow=confusion_matrix(test_sentiments,mnb_bow_predict,labels=[1,0])
print(cm_bow)
#confusion matrix for tfidf features
cm_tfidf=confusion_matrix(test_sentiments,mnb_tfidf_predict,labels=[1,0])
print(cm_tfidf)
```

```
[[3736 1271]
 [1219 3774]]
[[3729 1278]
 [1213 3780]]
```

Let us see positive and negative words by using WordCloud.

Word cloud for positive review words

In [48]:
```
#word cloud for positive review words
plt.figure(figsize=(10,10))
positive_text=norm_train_reviews[1]
WC=WordCloud(width=1000,height=500,max_words=500,min_font_size=5)
positive_words=WC.generate(positive_text)
plt.imshow(positive_words,interpolation='bilinear')
plt.show
```

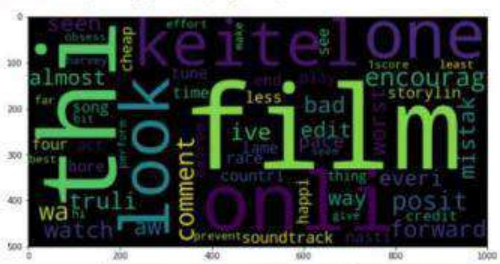
Out[48]: <function matplotlib.pyplot.show(*args, **kw)>

jupyter sentiment-analysis-of-imdb-movie-reviews Last Checkpoint: 11/02/2020 (autosaved)

Word cloud for negative review words

In [49]:
```
#word cloud for negative review words
plt.figure(figsize=(10,10))
negative_text=norm_train_reviews[0]
WC=WordCloud(width=1000,height=500,max_words=500,min_font_size=5)
negative_words=WC.generate(negative_text)
plt.imshow(negative_words,interpolation='bilinear')
plt.show
```

Out[49]: <function matplotlib.pyplot.show(*args, **kw)>



jupyter SENTIMENT ANALYSIS ON IMDb MOVIE REVIEWS Last Checkpoint: 10/19/2020 (autosaved)

In [35]:
```
instance = X[5]
print(instance)
```

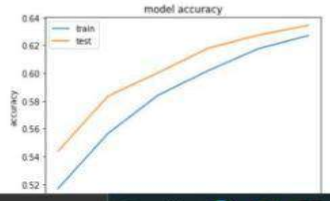
```
Probably my all time favorite movie story of selflessness sacrifice and dedication to noble cause but it not preachy or boring
It just never gets old despite my having seen it some or more times in the last years Paul Lukas performance brings tears to my
eyes and Bette Davis in one of her very few truly sympathetic roles is delight The kids are as grandma says more like dressed u
p midgets than children but that only makes them more fun to watch And the mother slow awakening to what happening in the world
and under her own roof is believable and startling If had dozen thumbs they all be up for this movie
```

In [36]:
```
instance = tokenizer.texts_to_sequences(instance)

flat_list = []
for sublist in instance:
    for item in sublist:
        flat_list.append(item)

flat_list = [flat_list]

instance = pad_sequences(flat_list, padding='post', maxlen=maxlen)

model.predict(instance)
```

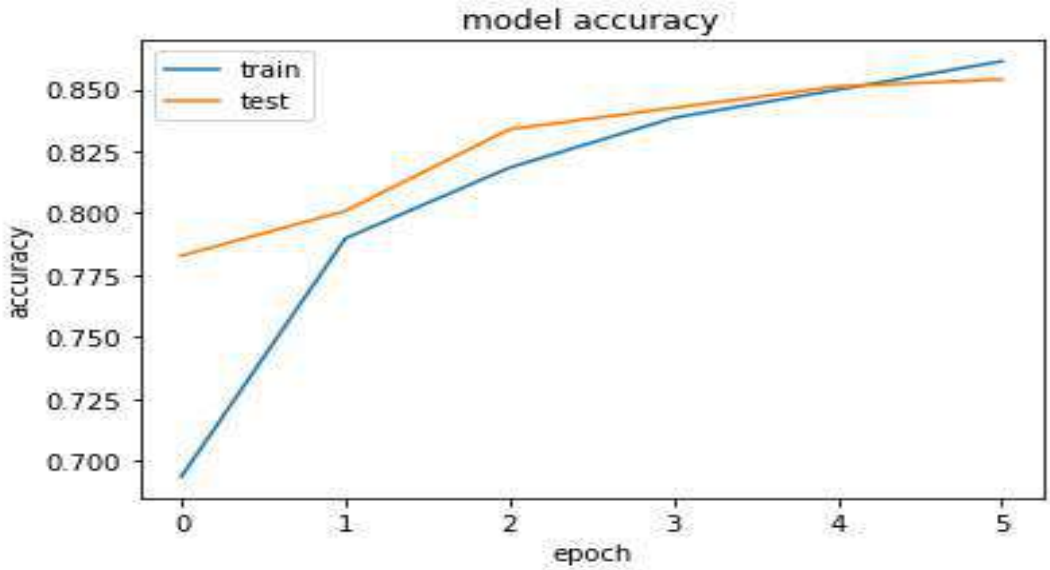
Out[36]:
```
array([[0.67628795]], dtype=float32)
```

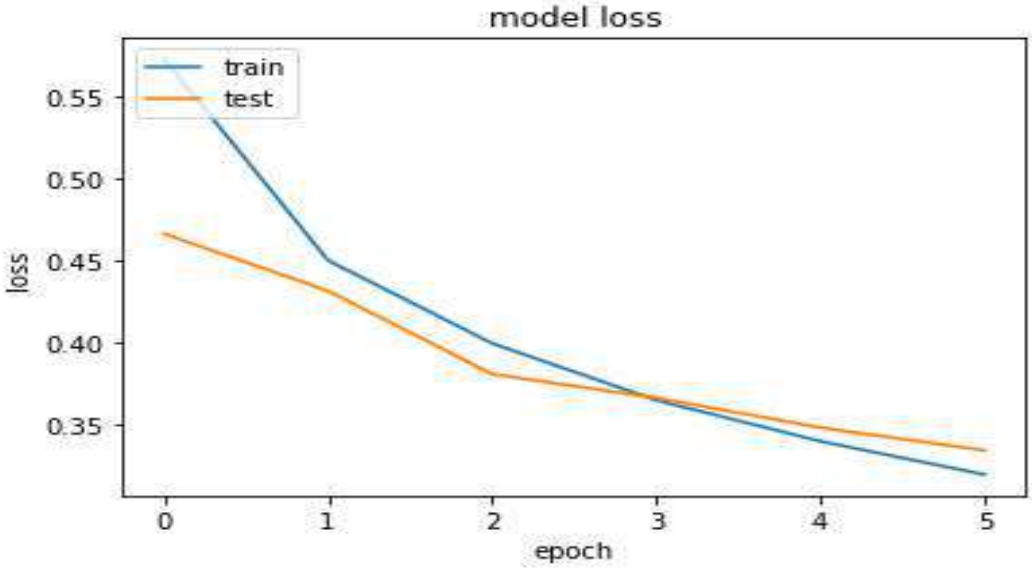
jupyter SENTIMENT ANALYSIS ON IMDb MOVIE REVIEWS Last Checkpoint: 10/19/2020 (autosaved)

In [34]:
```
import matplotlib.pyplot as plt

plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])

plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])

plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train','test'], loc='upper left')
plt.show()
```



RESULT AND ANALYSIS





```
instance = X[57]
print(instance)
```

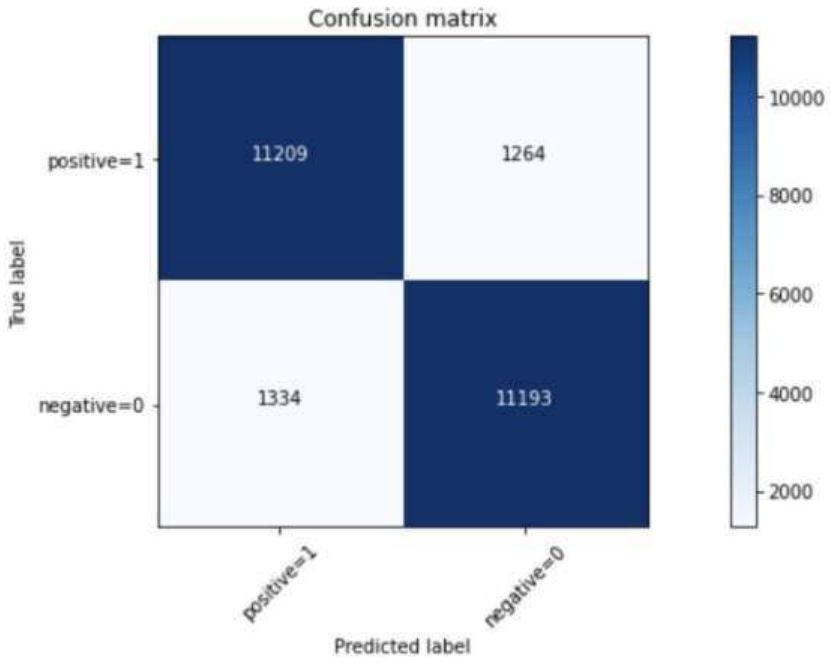
```
I laughed all the way through this rotten movie It so unbelievable woman leaves her husband after many years of marriage has br
eakdown in front of real estate office What happens The office manager comes outside and offers her job Hilarious Next thing yo
u know the two women are going at it Yep they re lesbians Nothing rings true in this Lifetime for Women with nothing better to
do movie Clunky dialogue like don want to spend the rest of my life feeling like had chance to be happy and didn take it doesn
help There a wealthy distant mother who disapproves of her daughter new relationship sassy black maid unbelievable that in the
year film gets made in which there a sassy black maid Hattie McDaniel must be turning in her grave The woman has husband who fr
eaks out and wants custody of the snotty teenage kids Sheesh No cliche is left unturned
```

```
instance = tokenizer.texts_to_sequences(instance)

flat_list = []
for sublist in instance:
    for item in sublist:
        flat_list.append(item)

flat_list = [flat_list]

instance = pad_sequences(flat_list, padding='post', maxlen=maxlen)

model.predict(instance)
```

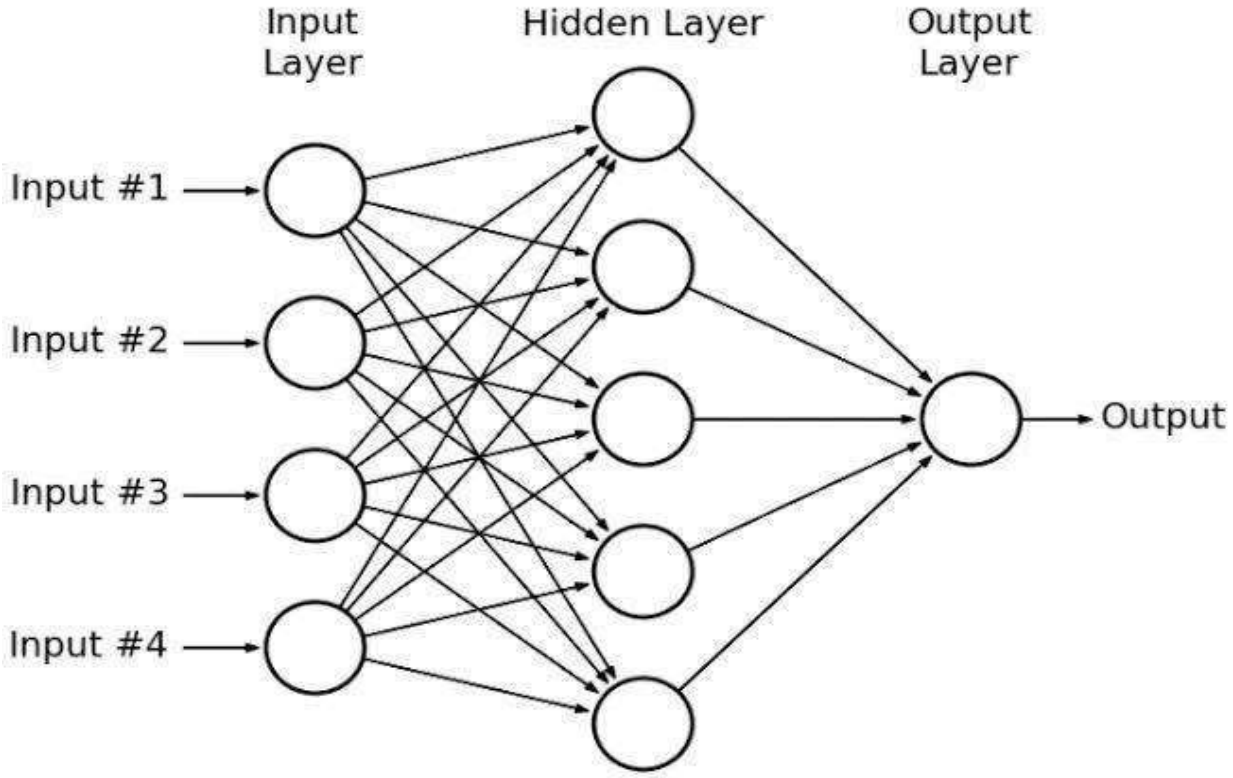
```
array([[0.6609535]], dtype=float32)
```

```
svm_bow_report=classification_report(test_sentiments,svm_bow_predict,target_names=['Positive','Negative'])
print(svm_bow_report)
svm_tfidf_report=classification_report(test_sentiments,svm_tfidf_predict,target_names=['Positive','Negative'])
print(svm_tfidf_report)
```

```
              precision    recall  f1-score   support

    Positive       0.94      0.18      0.30      4993
    Negative       0.55      0.99      0.70      5007

    accuracy                           0.58     10000
   macro avg       0.74      0.58      0.50     10000
weighted avg       0.74      0.58      0.50     10000

              precision    recall  f1-score   support

    Positive       1.00      0.02      0.04      4993
    Negative       0.51      1.00      0.67      5007

    accuracy                           0.51     10000
   macro avg       0.75      0.51      0.36     10000
weighted avg       0.75      0.51      0.36     10000
```



ARCHITECTURE



CONCLUSION

We discovered the IMDB sentiment analysis dataset for natural language processing.
we learned how to develop deep learning models for sentiment analysis including:
How to develop a large neural network model for sentiment analysis.
How to develop a one-dimensional convolutional neural network model for sentiment analysis.

How to load and review the IMDB dataset within Keras.

we have tried different Types of Algorithms like

- Naive Bayes
- Logistic Regression
- Ensemble
- Support Vector Machine
- Random Forest

So here we can clearly observe that Logistic regression and SVM gave the best accuracy compared to the others algorithms.