MOVIE REVIEW ANALYIS

Using Supervised Learning Algorithms

Submitted for course

CSE4022 - NATURAL LANGUAGE PROCESSING

by

Maria Alex Kuzhippallil

16BCE2190

To faculty

Prof. Arivoli

SCOPE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Slot C1+TC1

ABSTRACT

The focus of the project is to perform sentiment analysis on IMBD movie dataset obtained from kaggle. Sentimental analysis is a concept the focuses on analysis by extracting the emotions, opinions of people towards a particular topic from a structured, semi-structured or unstructured textual data. The main objective is to examine the polarity of the review of the movie whether its negative review or positive review using natural language processing techniques. In this project we aim to use Sentiment Analysis on a set of movie reviews given by reviewers and try to understand what their overall reaction to the movie was, i.e. if they liked the movie or they hated it. Also trying to compare how gaussian naïve bayes, multinomial naïve bayes, logistic regression, ada boost classifier, random forest tree and support vector machine algorithms can categorize the polarity correctly. Identifying the model with the best accuracy for perfect prediction of the data reviews.

INTRODUCTION

Movie reviews are an important way to gauge the performance of a movie. While providing a rating to a movie states clearly how far people liked the movie and disliked the movie, a collection of movie reviews is what gives us a deeper qualitative insight on different aspects of the movie.

A textual movie review clearly mentions us about the weak and strong points of the movie and deeper analysis of a movie review explains if the movie in meets the expectations of the reviewer and whether the review is positive or negative. Sentiment Analysis is a sub field of machine learning which aims to extract subjective information from the textual reviews. The field of sentiment of analysis is closely tied to natural language processing and text mining. It can be used to determine the attitude of the reviewer with respect to various topics or the overall polarity of review.

Using sentiment analysis, we can find the state of mind of the reviewer while providing the review and understand if the person was “happy”, “sad”, “angry” and so on. In this project we aim to use Sentiment Analysis on a set of movie reviews given by reviewers and try to understand what their overall reaction to the movie was, i.e. if they liked the movie or they hated it. And using various supervised learning algorithms to understand how well the test data prediction matches with the existing output. Different models such as logistic regression, ada boost classifier, random forest tree, multinomial naïve bayes, gaussian naïve bayes, support vector machines are used for prediction and the main aim of this project is to identify which model predicts the output label with maximum accuracy.

LITERATURE SURVEY

The main aim of this project is to identify the underlying sentiment of a movie review on the basis of its textual information. In this project, we try to classify whether a person liked the movie or not based on the review they give for the movie. This is particularly useful in cases when the creator of a movie wants to measure its overall performance using reviews that critics and viewers are providing for the movie. The outcome of this project can also be used to create a recommender by providing recommendation of movies to viewers on the basis of their previous reviews. Another application of this project would be to find a group of viewers with similar movie tastes (likes or dislikes).

Rafeeque Pandarachalil et al., 2014 proposed a method for Twitter sentiment analysis using an unsupervised learning approach. They determined the Polarity of tweets is evaluated by using three sentiment lexicons-SenticNet, SentiWordNet, and SentislangNet. They used parallel python framework to implement this method.

Turney's applied on classification of reviews is perhaps the most approach to ours. He uesd a specific unsupervised learning technique based on the mutual information between document phrases and the words "excellent" and "poor", where as the mutual information is calculated using statistics gathered by a search engine. In contrast, we used several completely prior-knowledge-free supervised machine learning techniques with the goal of empathizing the inherent trouble of the task .

Zhang et al. applied word dependency structure to classify the sentiment using rule based semantic analysis. Maas et al. applied both supervised and un-supervised techniques by getting semantic term document information to learn word vector.

As a part of this project, we aim to study several feature extraction techniques used in text mining e.g. keyword spotting, lexical affinity and statistical methods, and understand their relevance to our problem. In addition to feature extraction, we also look into different classification techniques and explore how well they perform for different kinds of feature representations. We finally draw a conclusion regarding which combination of feature representations and classification techniques are most accurate for the current predictive task.

Prediction of Movies popularity Using Machine Learning Techniques Muhammad Hassan Latif, Hammad Afzal clearly explains various data mining techniques that is used for prediction of the data set into negative and positive comments using algorithms such as Logistic regression, Cart algorithms etc and also highlights the difficulty with preprocessing the data.

Movie Success Prediction Using Data Mining - Antara Upadhyay, Nivedita Kamath which describes the opinion of person on a particular topic and uses a website model to predict the movie review and it uses HTML, PHP and MYSQL.

Sentiment as a prior for movie rating prediction Battu Varshit explais various algorithm used for prediction such as recurrent networks and convolution networks for rating prediction using various techniques such as word embeddings, sentence embeddings, inclusion of priors to analyze the result.

Predicting Movie Success: Review of Existing Literature Arundeep Kaur, AP Gurpinder Kaur - e. There are number of parameters that may influence success of a movie like – time of its release, marketing gimmicks, lead actor, lead actress, director, producer, writer, music director – being some of the factors. The present study aims to develop a model based upon neural networks that

may help in predicting the success of a movie in advance thereby reducing certain level of uncertainty using neural networks.

METHODOLOGY USED

Naive Bayes Classifier

Naive Bayes is a classification algorithm for binary (two-class) and multi-class classification problems. The technique is easiest to understand when described using binary or categorical input values.

The Naive Bayes model involves a simplifying conditional independence assumption. That is a given class (where people don't express opinions in the same way; they use opinion words as positive or negative comments), the words are conditionally independent of each other. This assumption does not affect much the accuracy in text classification but makes really fast classification algorithms applicable for the problems.

Multinomial Naïve Bayes uses multinomial distribution and when the data has discrete values for example the review is either positive or negative.

Gaussian Naïve Bayes uses normal distribution which is specially used when the features are continuous that is the bag of words with unique terms of reviews.

Support Vector Machines

Support vector machines (SVMs) have been shown to be highly effective at traditional text categorization, they are large-margin, rather than probabilistic classifiers. In the two-category case, the basic idea behind the training procedure is to find a hyperplane, represented by vector hat not only separates the document vectors in one class from those in the other, but for which the

separation, or margin, is as large as possible. In this project mainly the task to find the hyperplane that separates positive and negative comments.

Logistic Regression

Logistic regression is a predictive analysis regression statistical concept used when the dependant variable is binary valued that is positive(1) and negative(0) . This method clearly describes the dependence of the bag of words in reviews with the label of the polarity of the movie review of IMBD data set.

Ada Boost Classifier

This method is used to boost the performance of decision trees for binary classified tree problems. This model divides data set into N models for which prediction is performed and based on the maximum votes the best prediction is created.

Random Forest Tree

This model is bagged decision tree model. In decision tree, the tree splits into smaller groups based on the features that is the reviews. In random forest tree multiple decisions tree are created and based on the majority vote.

MODULES

Data Input

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis.

Data Cleaning

Varied cleaning methods include removing html content, non-alphabetical characters, tokenizing the sentences and lemmatize each word to its lemma.

Features and Opinion Words Extraction

All opinion words are selected from the sentence. The system extracts all nouns, noun phrases, verbs and adjectives from the movie review and compares with the existing list of words. These words are classified on basis of their polarity. For Example “good” word is of positive polarity. On the other hand, features are selected on basis of number times occurrence of opinion words. If opinion word is an occurrence in review higher than the threshold value then it is added features list. For this system API is trained only for movie reviews with keyword and phrases dictionary which includes “good acting”, “solid story” and “awesome action”.

Identify Sentence Polarity

After extracting all features and Opinion words, it is very easy to find the polarity of the sentence. Sentence polarity follows the same rules as arithmetic expressions. A negative sentiment contains all negative opinion words and positive sentiment contain all positive opinion words. A negative sentiment may contain a positive opinion word. For Example: “This movie Story is not good” sentence in a movie review. In this sentence, “good” opinion word is of positive polarity but “not” is a negative word. Therefore, the overall polarity of this sentence will be negative.

Classification of Review

Once, review polarity is calculated. Review polarity percentage and polarity (Positive or Negative) classified and saved for further analysis. With further analysis, box office collection can be predicted and overall performance of movie can also be predicted.

Supervised Algorithms used for prediction

A) **Gaussian Naïve Bayes**

B) **Multinomial Naïve Bayes**

C) **Logistic Regression**

D) **Random Forest Tree**

E) **Ada Boost Classifier**

F) **Support Vector Machines**

CODE SNIPPETS

A) Libraries imported –

The main libraries used include sklearn, pandas, numpy and nltk for performing various operations such as data reading, data cleaning, preprocessing, splitting the data in train and test data set, performing cross validation etc. Implementing various algorithms such as Random Forest Tree, Logistic Regression, Ada Boost Classifier.

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.metrics import accuracy_score
from sklearn.ensemble import (RandomForestClassifier,GradientBoostingClassifier,AdaBoostClassifier)
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn import svm
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

import seaborn as sn

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.stem.snowball import SnowballStemmer
import re
import matplotlib.pyplot as plt
```

B) Data Reading and Cleaning

The data set is IMDB movie reviews and basic cleaning is performed by removing null values. Converting the strings into lower case characters also eliminating html tags and performing a series of morphological analysis carried out includes removing unnecessary stop word removal, stemming and lemmatization. The contractions of possessive terms etc are expanded.

```
review=pd.read_csv('C:\\Users\\Maria\\Desktop\\PYTHON\\codes\\imdb_master.csv',encoding="latin-1")

#REMOVING UNNECESARY COLUMNS AND UNNECESSARY (UNSUPERVISED)
review=review.drop(['Unnamed: 0','type','file'],axis=1)


#NEGATIVE AND POSITIVE COMMENTS
sum(review['label']=='pos')
sum(review['label']=='neg')

#REPLACING BINARY VALUES
review['label']=review['label'].map({'neg':0,'pos':1})

#DATA PREPROCESSING (Morphological Analysis)
#1. lower case
review['review']=review['review'].str.lower()


#2. removing html tags
count=-1
for s in review['review']:
    s= re.sub("<.*?>", "", s)
    count=count+1
    review['review'].iloc[count]=s
```

```
#Expansion
contractions_dict = {
  'didn\'t': 'did not',
  'don\'t': 'do not',
  'you\'ve':'you have',
  "ain't": "am not / are not / is not / has not / have not",
  "aren't": "are not / am not",
  "can't": "cannot",
  "can't've": "cannot have",
  "'cause": "because",
  "could've": "could have",
  "couldn't": "could not",
  "couldn't've": "could not have",
  "didn't": "did not",
  "doesn't": "does not",

  "hadn't": "had not",
  "hadn't've": "had not have",
  "hasn't": "has not",
  "haven't": "have not",
  "he'd": "he had / he would",
  "he'd've": "he would have",
  "he'll": "he shall / he will",
  "he'll've": "he shall have / he will have",
  "he's": "he has / he is",
  "how'd": "how did",
  "how'd'y": "how do you",
  "how'll": "how will",
  "how's": "how has / how is / how does",
  "i'd": "I had / I would",
  "i'd've": "I would have",
  "i'll": "I shall / I will",
  "i'll've": "I shall have / I will have",
  "i'm": "I am",
  "i've": "I have",
  "isn't": "is not",
  "it'd": "it had / it would",
  "it'd've": "it would have",
  "it'll": "it shall / it will",
  "it'll've": "it shall have / it will have",
  "it's": "it has / it is",
  "let's": "let us",
```

```
  "weren't": "were not",
  "what'll": "what shall / what will",
  "what'll've": "what shall have / what will have",
  "what're": "what are",
  "what's": "what has / what is",
  "what've": "what have",
  "when's": "when has / when is",
  "when've": "when have",
  "where'd": "where did",
  "where's": "where has / where is",
  "where've": "where have",
  "who'll": "who shall / who will",
  "who'll've": "who shall have / who will have",
  "who's": "who has / who is",
  "who've": "who have",
  "why's": "why has / why is",
  "why've": "why have",
  "will've": "will have",
  "won't": "will not",
  "won't've": "will not have",
  "would've": "would have",
  "wouldn't": "would not",
  "wouldn't've": "would not have",
  "y'all": "you all",
  "y'all'd": "you all would",
  "y'all'd've": "you all would have",
  "y'all're": "you all are",
  "y'all've": "you all have",
  "you'd": "you had / you would",
  "you'd've": "you would have",
  "you'll": "you shall / you will",
  "you'll've": "you shall have / you will have",
  "you're": "you are",
}
contractions_re = re.compile('(%s)' % '|'.join(contractions_dict.keys()))
def expand_contractions(s, contractions_dict=contractions_dict):
    def replace(match):
        return contractions_dict[match.group(0)]
    return contractions_re.sub(replace, s)
```

```
for i in range(0,50000):
    print(i)
    review.review.loc[i]=expand_contractions(review.review.loc[i])

review.to_csv('C:\\Users\\Maria\\Desktop\\PYTHON\\codes\\imdb_mastercleand.csv')
review=pd.read_csv('C:\\Users\\Maria\\Desktop\\PYTHON\\codes\\imdb_mastercleand.csv')
#3. Punctuation Removal and Stopword Removal
count=-1
stop_words=set(stopwords.words('english'))
for s in review['review']:
    count=count+1
    s=[word for word in s.split() if (word not in stop_words and word.isalpha())]
    s=' '.join(s)
    review['review'].iloc[count]=s
review.to_csv('C:\\Users\\Maria\\Desktop\\PYTHON\\codes\\imdb_mastercleand2.csv')
review=pd.read_csv('C:\\Users\\Maria\\Desktop\\PYTHON\\codes\\imdb_mastercleand2.csv')
review=review.drop(['Unnamed: 0'],axis=1)


#4. Lemmatization and Stemming
count=-1
stem=[]
lemma=WordNetLemmatizer()
stemmer=SnowballStemmer('english')
for s in review['review']:
    count=count+1
    print(count)
    word_token=word_tokenize(s)
    for word in word_token:
        stem.append(stemmer.stem(lemma.lemmatize(word)))
    s=' '.join(stem)
    review['review'].iloc[count]=s
    stem=[]

review.to_csv('C:\\Users\\Maria\\Desktop\\PYTHON\\codes\\imdb_mastercleand3.csv')
```

C) Dividing the data into test and training data set

Firstly vectorization is done in order to obtain bag of words which produces a data frame with columns containing unique words and rows containing different documents and the data set is divided into training and test with ratio 80:20 for performing supervised learning. And import warnings library to eliminate unnecessary future warnings presented during execution of various models.

```
#Fully cleaned data

review=pd.read_csv('C:\\Users\\Maria\\Desktop\\PYTHON\\codes\\imdb_mastercleand3.csv')
review=review.drop(['Unnamed: 0'],axis=1)

#Bag of Words
vect=CountVectorizer()
bow=vect.fit_transform(review.review)
bow=pd.DataFrame(bow.toarray(),columns=list(vect.get_feature_names()))
bow.shape
xtrbow,xtebow,ytrbow,ytebow=train_test_split(bow,review.label,test_size=0.20)


import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)


#models
models=[GaussianNB(),
          MultinomialNB(),
          LogisticRegression(),
          RandomForestClassifier(),
          AdaBoostClassifier()]
```

D) Applying different models for prediction – Gaussian Naïve Bayes, Multinomial Naïve Bayes, Logistic Regression, Random Forest Tree, Ada Boost Classifier, Support Vector Machines

For different models the mean accuracy of the model, standard deviation of accuracy and accuracy of the test data set and confusion matrix containing true positives, false positive, true negative and false negative are found out. So that we can find out which model is the most suitable for correctly classifying the movie reviews into negative and positive reviews.

```
2
3 from sklearn.metrics import confusion_matrix
4 for model in models:
5     xtrbow=xtrbow
6     ytrbow=ytrbow
7     xtebow=xtebow
8     ytebow=ytebow
9     score=cross_val_score(model,xtrbow,ytrbow,cv=5)
0     print('\n',model)
1     print('Mean accuracy for training set - ',score.mean())
2     print('Standard deviation of accuracy - ',score.std())
3     model.fit(xtrbow,ytrbow)
4     pred=model.predict(xtebow)
5     print("Accuracy of test data- ",accuracy_score(ytebow,pred))
6     data = confusion_matrix(ytebow,pred)
7     df_cm = pd.DataFrame(data, columns=np.unique(ytebow), index = np.unique(ytebow))
8     df_cm.index.name = 'Actual'
9     df_cm.columns.name = 'Predicted'
0     plt.figure(figsize = (10,7))
1     sn.set(font_scale=1.4)#for label size
2     sn.heatmap(df_cm, cmap="Blues", annot=True,annot_kws={"size": 16})
3
4 print("\nSVM ")
5 model=svm.SVC(kernel='linear',class_weight={0:0.60,1:0.40})
6 model.fit(xtrbow,ytrbow)
7 pred=model.predict(xtebow)
8 print("Accuracy score of test - ",accuracy_score(ytebow,pred))
9
0 df_cm = pd.DataFrame(confusion_matrix(ytebow,pred), columns=np.unique(ytebow), index = np.unique(ytebow))
1 df_cm.index.name = 'Actual'
2 df_cm.columns.name = 'Predicted'
3 import seaborn as sn
4 sn.heatmap(df_cm, cmap="Blues", annot=True)
```

RESULTS

Total number of negative and positive reviews

```
In [35]: sum(review['label']=='pos')
Out[35]: 25000

In [36]: sum(review['label']=='neg')
Out[36]: 25000
```

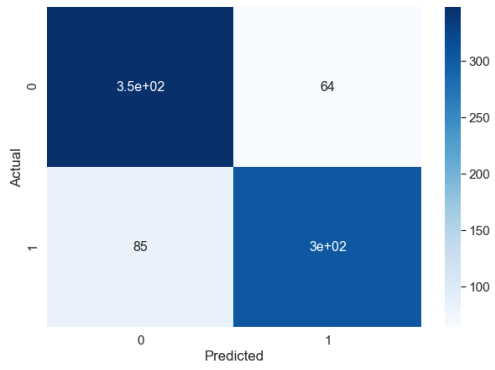
Bag of words

```
In [42]: vect=CountVectorizer()
    ...: bow=vect.fit_transform(review.review)
    ...: bow=pd.DataFrame(bow.toarray(),columns=list(vect.get_feature_names()))

In [43]: bow.shape
Out[43]: (4002, 17168)
```

```
In [45]: bow.head()
Out[45]:
   aa  aaaahhhhhh  aam  aap      ...        zucker  zugurt  zulu  zzzzzzzzzzzz
0   0           0    0    0      ...             0       0     0             0
1   0           0    0    0      ...             0       0     0             0
2   0           0    0    0      ...             0       0     0             0
3   0           0    0    0      ...             0       0     0             0
4   0           0    0    0      ...             0       0     0             0

[5 rows x 17168 columns]
```

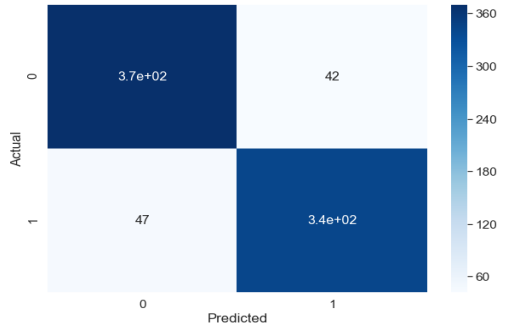
1) Model used for prediction – Gaussian Naïve Bayes

```
 GaussianNB(priors=None, var_smoothing=1e-09)
Mean accuracy for training set -  0.7641307527301093
Standard deviation of accuracy -  0.017326753413094264
Accuracy of test data-  0.8139825218476904
```



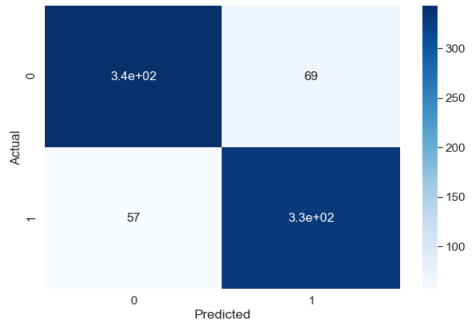
2) Model used for prediction – Multinomial Naïve Bayes

```
 MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
Mean accuracy for training set -  0.8866044266770672
Standard deviation of accuracy -  0.014584391546540941
Accuracy of test data-  0.8888888888888888
```



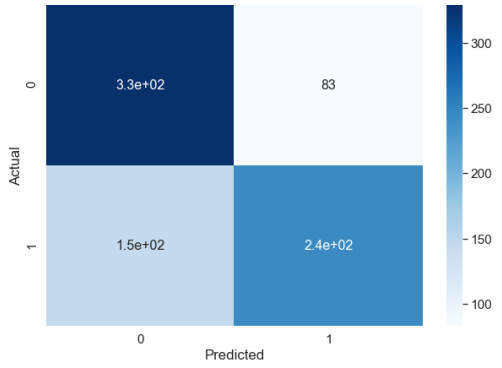
3) Model used for prediction – Logistic Regression

```
 LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
Mean accuracy for training set -  0.8544257020280812
Standard deviation of accuracy -  0.009963716768097536
Accuracy of test data-  0.8426966292134831
```



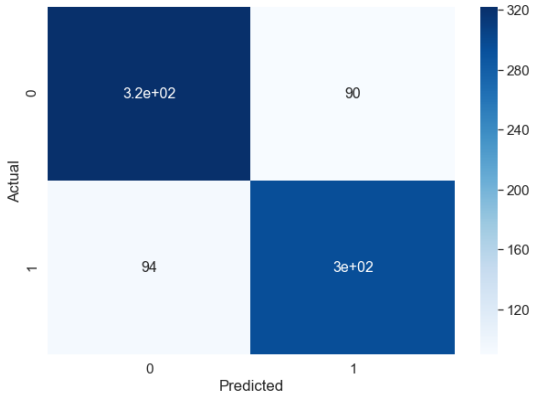
4) Model Used for prediction – Random Forest Tree

```
 RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators='warn', n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
Mean accuracy for training set -  0.7494539781591264
Standard deviation of accuracy -  0.0074096893931421964
Accuracy of test data-  0.714107365792759
```



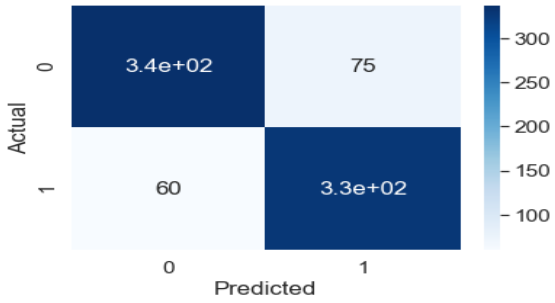
5) Model used for prediction – AdaBoost Classifier

```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None,
          learning_rate=1.0, n_estimators=50, random_state=None)
Mean accuracy for training set -  0.7869456903276131
Standard deviation of accuracy -  0.008189352748237781
Accuracy of test data-  0.7702871410736579
```



6) Model used for prediction – Support Vector machines

```
SVC(C=1.0, cache_size=200, class_weight={0: 0.6, 1: 0.4}, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=None,
  shrinking=True, tol=0.001, verbose=False)
Mean accuracy for training set -  0.8350550897035882
Standard deviation of accuracy -  0.011280683273024273
Accuracy score of test -  0.8314606741573034
```



CONCLUSION

Models	Gaussian NB	Multinomial NB	Logistic regression	Ada Boost Classifier	Random Forest Tree	Support Vector Machines
Accuracy of training set	0.7641	0.8866	0.8544	0.7869	0.7494	0.8350
Standard deviation	0.0173	0.0145	0.0099	0.0081	0.0074	0.0112
Accuracy of test set	0.8139	0.8888	0.8426	0.7702	0.7141	0.8314

Multinomial Naïve Bayes has achieved to bring the best result for prediction in training set and in test data set as well with very less standard deviation, the next algorithm which receives best result includes logistic regression and support vector machines which yields more than 80% accuracy. The least performance was offered by Random forest tree, Adaboost classifier and Gaussian Naïve Bayes due to the presence of so many bag of words which is used to determine the polarity of the reviews whether negative or positive.

Multinomial Naïve Bayes is the best approach since it uses feature engineering concept of extracting features of text and uses Bayes conditional probability to correctly classify as a negative or positive comment and also uses maximum likelihood hence its able to derive the accurate results. Tf-idf is used since it not only finds the occurrence of the word but also reflects the importance of a words. Random forest tree and AdaBoost tree produces comparatively less accuracy due to the presence of huge number of term frequency words which leads infinity randomized tree which makes it slower and makes it less accurate.

REFERENCES

[1] Prediction of Movies popularity Using Machine Learning Techniques Muhammad Hassan Latif†, Hammad Afzal††

[2] Movie Success Prediction Using Data Mining Antara Upadhyay, Nivedita Kamath

[3] Sentiment as a prior for movie rating prediction Battu Varshit

[4] Predicting Movie Success: Review of Existing Literature Arundeep Kaur, AP Gurpinder Kaur

[5] Sentiment Analysis for Movie Reviews Ankit Goyal, Amey Parulekar,