Table of Contents

Index

Introduction

We have decided to do a sentiment analysis for the reviews of movies as received by IMDB, using various Classifiers and compare them on the basis of accuracy score, precision, recall and F score. We will be generating a classification report, Confusion matrix from the given input and then try to compare accuracy between the classifiers.

Before going any further let's understand some basic Terminologies –

Natural Language Processing (NLP)

It is a subfield of Computer Science Linguistics and artificial intelligence dealing with the interactions between computers and human languages (natural languages).

Natural Language Processing enables the interaction between humans and computers in the former's language and scales. NLP enables the computer to do tasks such as reading text, interpreting, hear speech, measure sentiment of the speech and determine important parts.

Sentiment Analysis

It refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

Existing approaches to sentiment analysis can be grouped into three main categories: knowledge-based techniques, statistical methods, and hybrid approaches.

- Knowledge-based techniques classify text by affect categories based on the presence of unambiguous affect words such as happy, sad, afraid, and bored. Some knowledge bases not only list obvious affect words, but also assign arbitrary words a probable "affinity" to particular emotions.
- Statistical methods leverage elements from machine learning such as latent semantic analysis, support vector machines, "bag of words", "Pointwise Mutual Information" for Semantic Orientation, and deep learning. More sophisticated methods try to detect the holder of a sentiment (i.e., the person who maintains that affective state) and the target (i.e., the entity about which the affect is felt). To mine the opinion in context and get the feature about which the speaker has opined, the grammatical relationships of words are used. Grammatical dependency relations are obtained by deep parsing of the text.
- Hybrid approaches leverage both machine learning and elements from knowledge representation such as ontologies and semantic networks in order to detect semantics that are expressed in a subtle manner, e.g., through the analysis of concepts that do not explicitly convey relevant information, but which are implicitly linked to other concepts that do so.

Classification Algorithms

Classification is the process of recognizing, understanding, and grouping ideas and objects into pre-set categories or "sub-populations." Using pre-categorized training datasets, machine learning programs use a variety of algorithms to classify future datasets into categories.

Classification algorithms in machine learning use input training data to predict the likelihood that subsequent data will fall into one of the predetermined categories. One of the most common uses of classification is filtering emails into "spam" or "non-spam."

In short, classification is a form of "pattern recognition," with classification algorithms applied to the training data to find the same pattern (similar words or sentiments, number sequences, etc.) in future sets of data.

Using classification algorithms, which we'll go into more detail about below, text analysis software can perform things like sentiment analysis to categorize unstructured text by polarity of opinion (positive, negative, neutral, and beyond).

Technology Used

Software and Libraries

The software used for the implementation of this project is Ipython Notebook, its basically an IDE that allows running different snippets of the code together, the libraries used for the successful implementation of the project are –

Pandas – It is a powerful Python library used for data analysis and manipulation, we used it to create data frames in order to perform various pre-processing and classification algorithms.

Numpy – It is a python library dealing with the mathematical operations on arrays and matrices, it is way faster than inbuilt methods.

Scikit Learn – It is a machine learning library for Python Language.

NLTK – It is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language.

Concepts

Natural Language Processing –

Lemmatization – It is a process of grouping together the different forms of the word so that they can be analyzed together as a single unit, Text pre-processing can be done by both Stemming and Lemmatization but Lemmatization is preferred more since it does morphological analysis of the words. Function used – WordNetLemmatizer().

Stop words - These are the words that have to be filtered out from the sentence in order to make efficient sentiment analysis of the sentence, these are the words such as we, will, for, the and others.

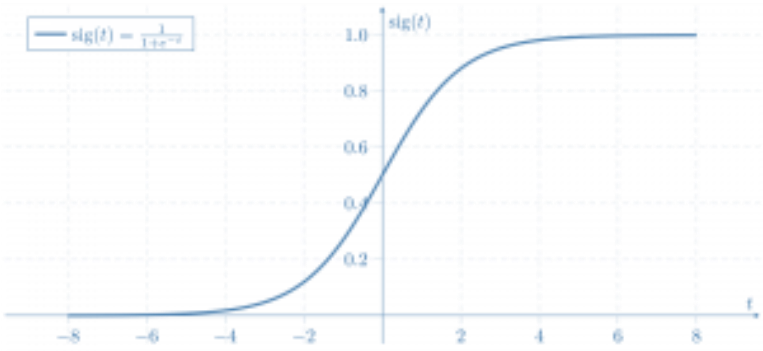
Tfid Vectorization – The process to transforming text to feature that can be used as an input to the classifier. In each vector the numbers represent the tf-idf scores.

Classification Algorithms -

Logistic Regression – It is a supervised classification algorithm; in a classification algorithm the target value y can only take discreet values depending on the input x.

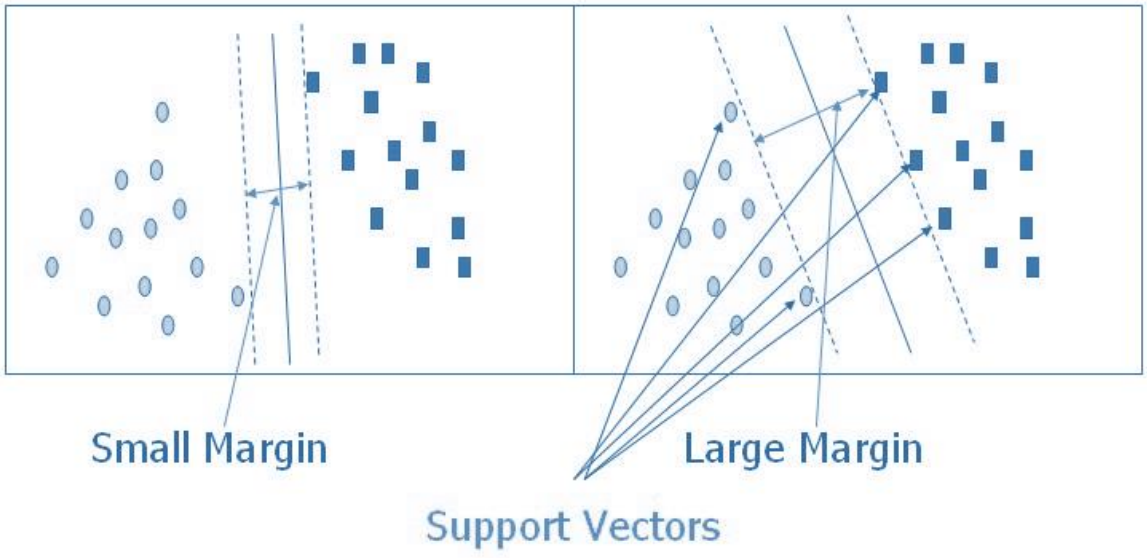
It's called regression cause its based on a regression model, just like linear regression models data using a linear function, the logistic regression models data based on sigmoid function.

$$g(z) = \frac{1}{1+e^{-z}}$$



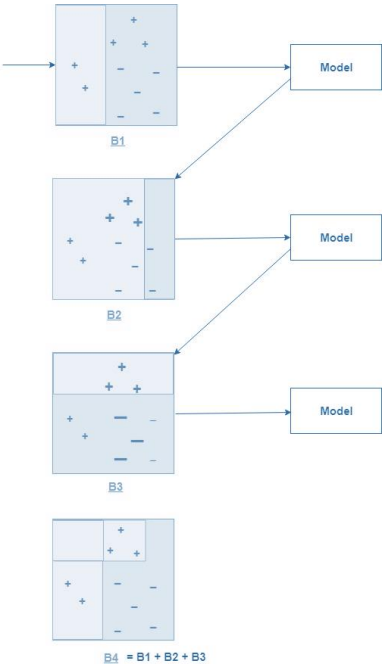
It becomes a classification algorithm only when decision threshold is considered, the decision threshold can be seen as a threshold value that separates one class from another, hence we can see this is a binary classification solution.

SVM (Support Vector Machines) - The support vector machine is a model used for both classification and regression problems though it is mostly used to solve classification problems. Instead of finding a decision threshold like Linear regression this algorithm creates a hyperplane or line(decision boundary) which separates data into classes. It



uses the kernel trick to find the best line separator (decision boundary that has same distance from the boundary point of both classes). It is a clear and more powerful way of learning complex non linear functions.

AdaBoost classifier – Boosting is an ensemble learning technique which focuses on building a strong classifier from a number of weak classifiers, done by building models using weak models. A model is built from train data, then another model correcting the mistakes in the first model is built, the process continues till the complete dataset is predicted correctly or limit on the number of models is reached.



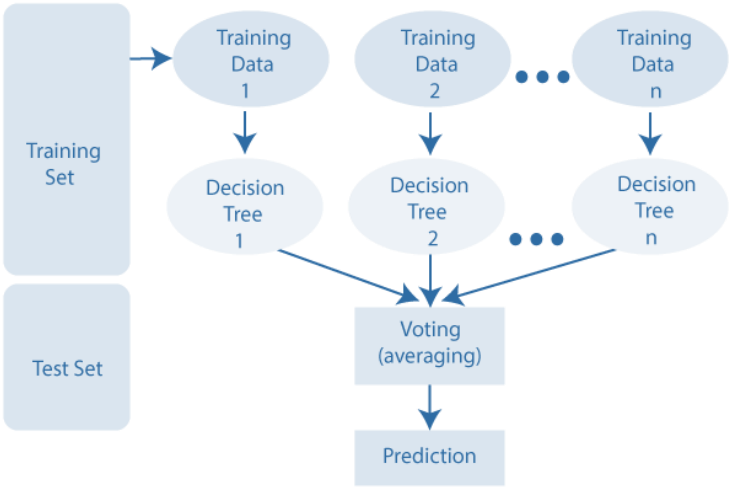
AdaBoost was first successful boosting algorithm implemented for binary classification problems.

Naïve Bayes Classifier – These are the collection of classification algorithms, based on Bayes' Theorem. They are the collection of algorithms where every feature being classified is independent of each other, and every feature is given same weight or importance.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Bayes' theorem finds the probability of occurrence of event A given that another even B has already occurred.

Random Forest Classifier – It is a supervised machine learning algorithm used for classification.



It creates a set of decision trees from the randomly selected subset of training set and then collects the votes from different decision trees to make the final prediction.

Code

Language Used => Python (3.8)

```
# importing dataset collection libraries
import pandas as pd
import numpy as np
import glob, os, string, re

# importing Natural Language Processing Libraries
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# impoting Evaluvation Metrics
from sklearn.metrics import classification_report, accuracy_score, conf
usion_matrix

# importing classification algorithms
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import LinearSVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB


# importing dataset
from os import listdir
from os.path import isfile, join

onlyfiles = [f for f in listdir("aclImdb/train/pos") if isfile(join("ac
lImdb/train/pos", f))]

#extracting names
import os
name = []

for j in onlyfiles:
    os.path.splitext(j)
    name.append(os.path.splitext(j)[0])

#reading all the reviews for the dataframe
review =[]
for path in onlyfiles:
    txt = open("aclImdb/train/pos/"+path, encoding="utf8")
    review.append(txt.read())

#creating dataframe
movie_dict = {"review": review, "target":1}
movie_df = pd.DataFrame(data = movie_dict)
```

	review	target
0	Bromwell High is a cartoon comedy. It ran at t...	1
1	Homelessness (or Houselessness as George Carli...	1
2	Brilliant over-acting by Lesley Ann Warren. Be...	1
3	This is easily the most underrated film inn th...	1
4	This is not the typical Mel Brooks film. It wa...	1

Data Frame Looks Like

```
#doing same for negatives
from os import listdir
from os.path import isfile, join

onlyfiles = [f for f in listdir("aclImdb/train/neg") if isfile(join("ac
lImdb/train/neg", f))]
import os
name = []

for j in onlyfiles:
    os.path.splitext(j)
    name.append(os.path.splitext(j)[0])
review =[]
for path in onlyfiles:
    txt = open("aclImdb/train/neg/"+path, encoding="utf8")
    review.append(txt.read())
movie_dict = {"review": review, "target":0}
movie_df_neg = pd.DataFrame(data = movie_dict)
```

	review	target
0	Story of a man who has unnatural feelings for ...	0
1	Airport '77 starts as a brand new luxury 747 p...	0
2	This film lacked something I couldn't put my f...	0
3	Sorry everyone,,, I know this is supposed to b...	0
4	When I was little my parents took me along to ...	0

Data Frame with Negative sentences

```
#merging both datasets
movie_df = movie_df_neg.append(movie_df, ignore_index = True)


#data preprocessing function :
lemma = WordNetLemmatizer()
stops = set(stopwords.words('english'))

def text_prep(text):
    no_punct = [char for char in text if char not in string.punctuation
]
    text = "".join(no_punct)
    text = [lemma.lemmatize(text, pos='v') for text in text.lower().spl
it() if text not in stops]
    text = " ".join(text)
```

```
    return (text)

#preprocessing training data
movie_df['prep_review'] = movie_df['review'].apply(lambda x: text_prep(
x))
movie_df[['prep_review', 'target']].head()
```

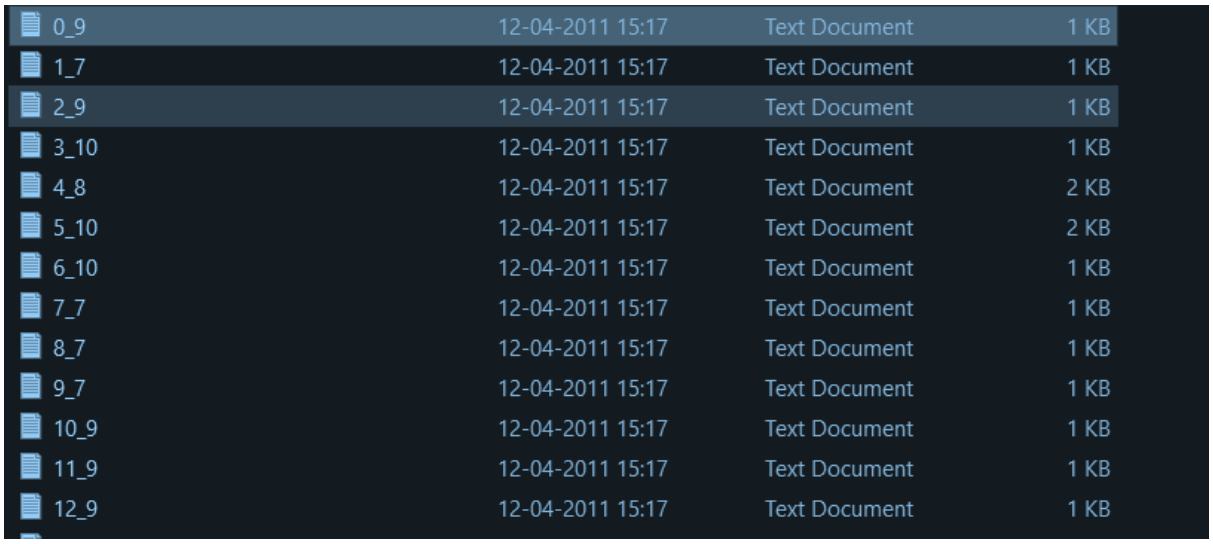
	prep_review	target
0	story man unnatural feel pig start open scene ...	0
1	airport 77 start brand new luxury 747 plane lo...	0
2	film lack something couldnt put finger first c...	0
3	sorry everyone know suppose art film wow hand ...	0
4	little parent take along theater see interiors...	0

After Preprocessing Data

```
#doing it for testing data now
onlyfiles = [f for f in listdir("aclImdb/test/pos") if isfile(join("acl
Imdb/test/pos", f))]
review =[]
for path in onlyfiles:
    txt = open("aclImdb/test/pos/"+path, encoding="utf8")
    review.append(txt.read())

movie_dict = {"review": review, "target":1}
movie_df_test = pd.DataFrame(data = movie_dict)

movie_df_test
onlyfiles = [f for f in listdir("aclImdb/test/neg") if isfile(join("acl
Imdb/test/neg", f))]
review =[]
for path in onlyfiles:
    txt = open("aclImdb/test/neg/"+path, encoding="utf8")
    review.append(txt.read())

movie_dict = {"review": review, "target":0}
movie_df_test_neg = pd.DataFrame(data = movie_dict)

movie_df_test_neg
movie_df_test = movie_df_test_neg.append(movie_df_test, ignore_index =
True)
movie_df_test['prep_review'] = movie_df_test['review'].apply(lambda x:
text_prep(x))
movie_df_test[['prep_review', 'target']].head()
```

```
#now vectorizing train data
tfidf = TfidfVectorizer()

x_train = tfidf.fit_transform(movie_df['prep_review'])
y_train = movie_df['target']

#vectorizing test data

x_test = tfidf.transform(movie_df_test['prep_review'])
y_test = movie_df_test['target']


# now training prediction classifiers
# Logistic Regression
# Linear SVC Classifier
# Ada Boost Classifier
# Naive Bayes Classifier
# Random Forest Classifier
print(x_train.shape, x_test.shape)


#function to fit various classification models
def model (classifier , x_train, y_train, x_test, y_test):
    cmodel = classifier
    cmodel.fit(x_train, y_train)
    cmodel_pred = cmodel.predict(x_test)
    return [cmodel.score(x_train , y_train) , accuracy_score(y_test, cm
odel_pred), classification_report(y_test, cmodel_pred), confusion_matri
x(y_test, cmodel_pred)]


classifiers = [LogisticRegression() , LinearSVC(), AdaBoostClassifier(n
_estimators = 100), MultinomialNB(), RandomForestClassifier(n_estimator
s=100, random_state = 42, n_jobs = -1)]
names = ["Logistic Regression" , "SVM", "AdaBoost", "Naive Bayes", "Ran
dom Forests"]
for i in range(len(classifiers)):
    [score, accuracy, report, matrix] = model(classifiers[i], x_train,
y_train, x_test, y_test )
    print('*****************************', names[i], '*****************
***************')
    print("Classifier Score = ",  score)
    print("Accuracy score = ", accuracy)
    print("Classification Report \n  ")
    print(report)
    print("Confusion Matrix \n ")
    print(matrix)
    print('************************************************************
*************** \n \n')
```

Dataset



These are some of the files in the dataset, the prefix to _ denoted movie id, and the suffix denotes the movie rating.

The contents of a file are the review about the movie with the given ratings.

The full dataset for the IMDB reviews can be downloaded from

CLICK TO GET THE DATASET

Output

```
**************************** Logistic Regression ********************************
Classifier Score =  0.93488
Accuracy score =  0.88124
Classification Report

              precision    recall  f1-score   support

           0       0.88      0.88      0.88     12500
           1       0.88      0.88      0.88     12500

    accuracy                           0.88     25000
   macro avg       0.88      0.88      0.88     25000
weighted avg       0.88      0.88      0.88     25000

Confusion Matrix

[[10986  1514]
 [ 1455 11045]]
*************************************************************************
```

Results for Logistic regression

```
**************************** SVM ********************************
Classifier Score =  0.9926
Accuracy score =  0.871
Classification Report

              precision    recall  f1-score   support

           0       0.86      0.88      0.87     12500
           1       0.88      0.86      0.87     12500

    accuracy                           0.87     25000
   macro avg       0.87      0.87      0.87     25000
weighted avg       0.87      0.87      0.87     25000

Confusion Matrix

[[11019  1481]
 [ 1744 10756]]
*************************************************************************
```

Results for SVM classifier

```
****************************** AdaBoost ********************************
Classifier Score =  0.83932
Accuracy score =  0.83064
Classification Report

              precision    recall  f1-score   support

           0       0.85      0.81      0.83     12500
           1       0.82      0.85      0.83     12500

    accuracy                           0.83     25000
   macro avg       0.83      0.83      0.83     25000
weighted avg       0.83      0.83      0.83     25000

Confusion Matrix

[[10104  2396]
 [ 1838 10662]]
```

Results for AdaBoost Classifier

```
****************************** Naive Bayes ********************************
Classifier Score =  0.91912
Accuracy score =  0.8316
Classification Report

              precision    recall  f1-score   support

           0       0.80      0.88      0.84     12500
           1       0.87      0.78      0.82     12500

    accuracy                           0.83     25000
   macro avg       0.83      0.83      0.83     25000
weighted avg       0.83      0.83      0.83     25000

Confusion Matrix

[[11000  1500]
 [ 2710  9790]]
***************************************************************************
```

Results for Naïve Bayes

```
**************************** Random Forests ******************************
Classifier Score =  1.0
Accuracy score =  0.847
Classification Report

              precision    recall  f1-score   support

           0       0.84      0.86      0.85     12500
           1       0.86      0.83      0.84     12500

    accuracy                           0.85     25000
   macro avg       0.85      0.85      0.85     25000
weighted avg       0.85      0.85      0.85     25000

Confusion Matrix

[[10774  1726]
 [ 2099 10401]]
**********************************************************************
```

Results for Random Forest Classifier

Conclusion

We took a dataset that contained reviews for movies in separate files. We used file handling technique to extract dataset then created the data frame using it. We used NLP techniques such as removing stop words, lemmatization to pre-process data. We then Vectorized the data and applied various classification algorithms. Random Forests and Naïve Bayes show the signs of overfitting, while the best performances were given by logistic regression and SVM, however we will be keeping SVM above Logistic cause of it being a better classification algorithm for such dataset.

In the end the order will be –

SVM

Logistic Regression

Random Forests classifier

Naïve Bayes Classifier

AdaBoost Classifier

References

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

Li Y., Bontcheva K., Cunningham H. (2005) SVM Based Learning System for Information Extraction. In: Winkler J., Niranjan M., Lawrence N. (eds) Deterministic and Statistical Methods in Machine Learning. DSMML 2004. Lecture Notes in Computer Science, vol 3635. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11559887_19

Robert E. Schapire. 1999. A brief introduction to boosting. In Proceedings of the 16th international joint conference on Artificial intelligence - Volume 2 (IJCAI'99). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1401–1406.

Web sources –

Sentiment Analysis (stanford.edu)

Understanding Logistic Regression - GeeksforGeeks

Naive Bayes Classifiers - GeeksforGeeks

www.Wikipedia.com

Random Forest Classifier using Scikit-learn - GeeksforGeeks

Classifying data using Support Vector Machines(SVMs) in Python - GeeksforGeeks