Internship Project Title	Automate detection of different emotions from paragraphs and detect overall emotions.
Project Title	Automate detection of different emotions from paragraphs and detect overall emotions.
Name of the Company	TCS iON
Name of the Industry Mentor	Debashis Roy
Name of the Institute	Indian Institute Of Information Technology And Management – Kerala, Trivandrum

Start Date	End Date	Total Effort (hrs.)	Project Environment	Tools used
25/04/2022	22/06/2022	210	Jupyter Notebook	Python 3

Project Synopsis:

Sentiment analysis is the analysis of emotions and opinions from any form of text. Sentiment analysis is also termed as opinion mining. Sentiment analysis of the data is very useful to express the opinion of the mass or group or any individual. This technique is used to find the sentiment of the person with respect to a given source of content. Sentiments and opinions of individuals and mass group is important because that helps the company or the platform to improve or scale the services or products that are deployed. Social media and other online platforms contains a huge amount of the data in the form of tweets, reviews, blogs and updates on the status, posts, etc. In this paper, I have analysed the Movie reviews using various machine learning algorithms like Logistic Regression, Decision Tree, Random Forest etc.

Solution Approach:

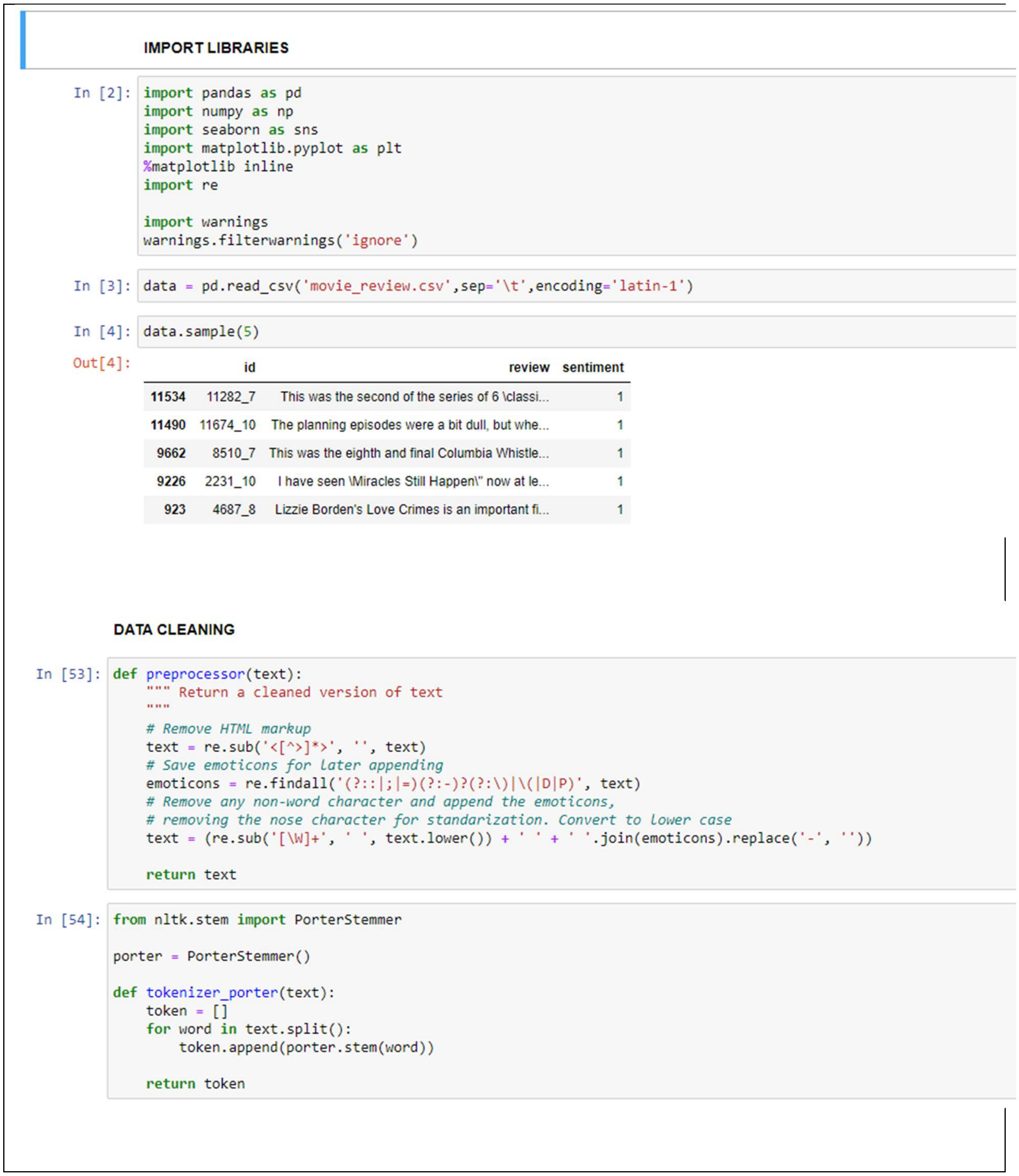
With the upcoming of various platforms such as Facebook, Twitter, LinkedIn, Instagram allow people to share their comments, opinions, feelings, reviews on a multitude of topics from education to entertainment. These platforms hold huge amount of data in the form of tweets, blogs, feedbacks and status updates, posts etc. Emotion analysis aims to identify the extremes of emotions such as happiness, grief, sadness, hate, anger, affection, and opinions. From texts, reviews, publications are available online on these platforms. Opinion mining finds the sentiment of the text in relation to a given content source. Sentiment analysis is complicated by slang words, misspellings, short forms, repeated characters, regional language usage, and new emojis coming soon. Therefore, an important task is to determine the appropriate sentiment of each word. Sentiment analysis is one of the most active and well studied areas of research in data mining.

Emotion analysis is applicable in almost all fields of business and society, as opinions are at the heart of most human activities and behaviors. Sentiment analysis is very popular because of its effectiveness. Thousands of documents can be processed for sentiment analysis. Since it is an efficient process that provides good accuracy, it therefore has many different applications:

- Improve the quality of a product or service: through opinion extraction, producers can gather user opinions, whether favorable or not, about their product or service, they can then improve and enhance the quality of their product or service.
- Recommendation system: By analyzing and categorizing people's opinions based on their preferences and concerns, the system can predict which items should be recommended and which are not should be recommended.
- Decision Making: People's sentiments, ideas, feelings are very important factor to make a decision. While buying an item the reviews of the previously purchased customers is been checked to make a decision to buy the product or not.
- Marketing research: Sentiment analysis technique results can be used to identify the patterns of customer behaviors and based on that new strategies and policies can be build to improve their business and products.

Phases of sentiment analysis:

Pre-Processing Phase: The data is first cleaned to reduce noise.
Feature Extraction: A token is given to the keywords and this token is now put under analysis.
Classification Phase: Based on different algorithms these keywords are put under certain category.



IMPORT LIBRARIES

```
In [2]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
        import re

        import warnings
        warnings.filterwarnings('ignore')
```

```
In [3]: data = pd.read_csv('movie_review.csv',sep='\t',encoding='latin-1')
```

```
In [4]: data.sample(5)
```

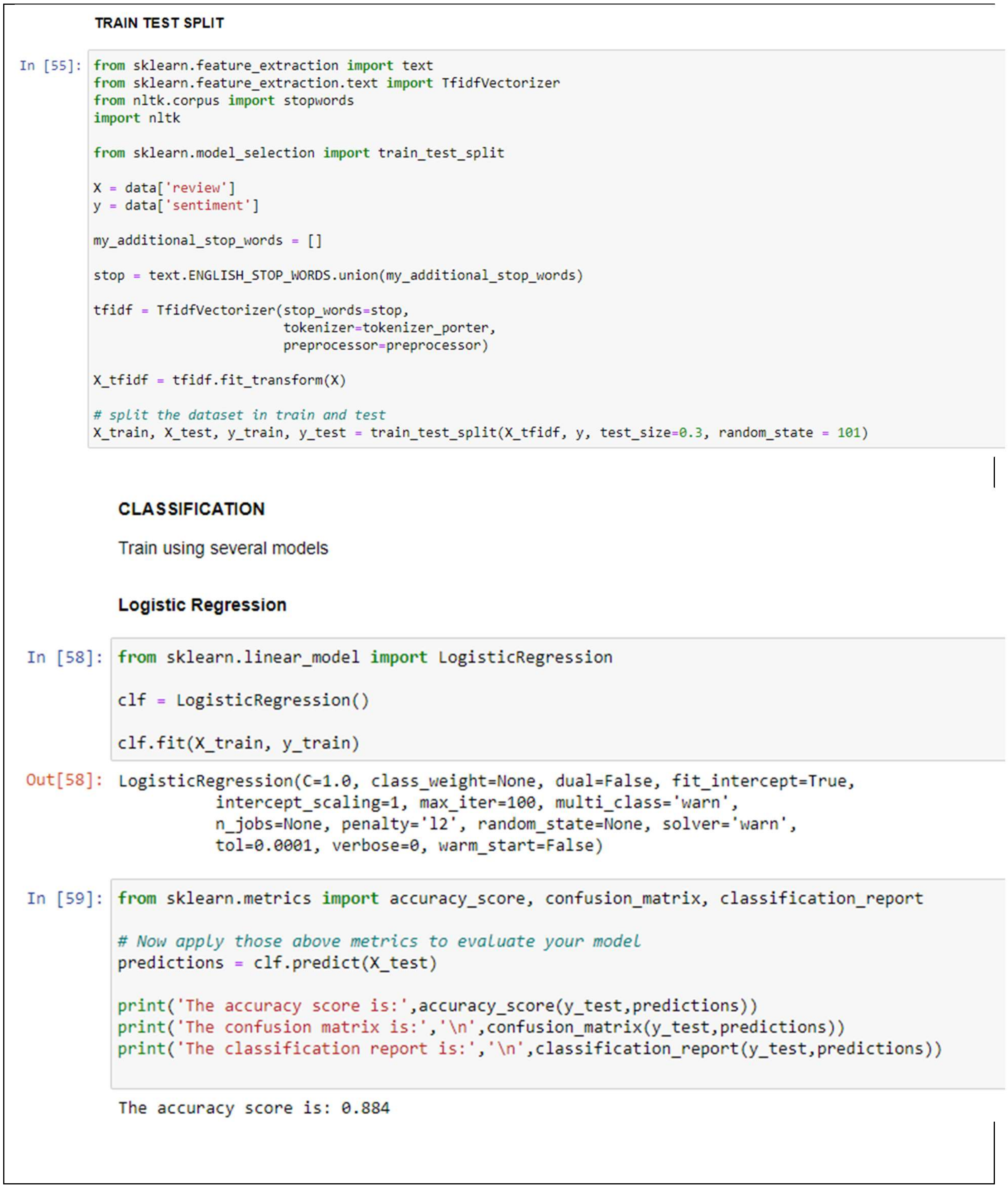
Out[4]:

	id	review	sentiment
11534	11282_7	This was the second of the series of 6 \classi...	1
11490	11674_10	The planning episodes were a bit dull, but whe...	1
9662	8510_7	This was the eighth and final Columbia Whistle...	1
9226	2231_10	I have seen \Miracles Still Happen\" now at le...	1
923	4687_8	Lizzie Borden's Love Crimes is an important fi...	1

DATA CLEANING

```
In [53]: def preprocessor(text):
             """ Return a cleaned version of text
             """
             # Remove HTML markup
             text = re.sub('<[^>]*>', '', text)
             # Save emoticons for later appending
             emoticons = re.findall('(?::|;|=)(?:-)?(?:\)|\(|D|P)', text)
             # Remove any non-word character and append the emoticons,
             # removing the nose character for standarization. Convert to lower case
             text = (re.sub('[\W]+', ' ', text.lower()) + ' ' + ' '.join(emoticons).replace('-', ''))

             return text
```

```
In [54]: from nltk.stem import PorterStemmer

         porter = PorterStemmer()

         def tokenizer_porter(text):
             token = []
             for word in text.split():
                 token.append(porter.stem(word))

             return token
```



TRAIN TEST SPLIT

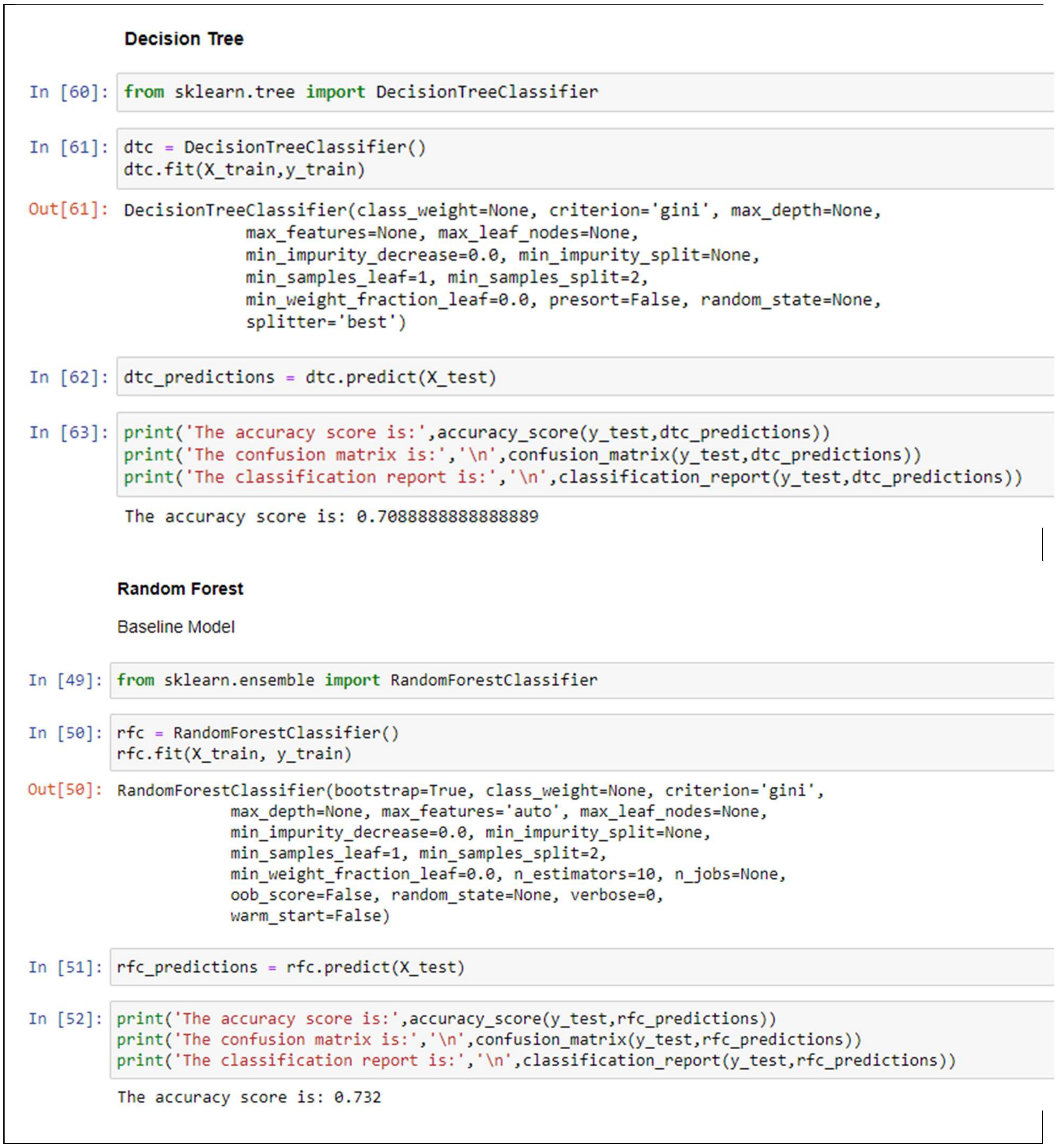
```
In [55]: from sklearn.feature_extraction import text
         from sklearn.feature_extraction.text import TfidfVectorizer
         from nltk.corpus import stopwords
         import nltk

         from sklearn.model_selection import train_test_split

         X = data['review']
         y = data['sentiment']

         my_additional_stop_words = []

         stop = text.ENGLISH_STOP_WORDS.union(my_additional_stop_words)

         tfidf = TfidfVectorizer(stop_words=stop,
                                 tokenizer=tokenizer_porter,
                                 preprocessor=preprocessor)

         X_tfidf = tfidf.fit_transform(X)

         # split the dataset in train and test
         X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.3, random_state = 101)
```

CLASSIFICATION

Train using several models

Logistic Regression

```
In [58]: from sklearn.linear_model import LogisticRegression

         clf = LogisticRegression()

         clf.fit(X_train, y_train)

Out[58]: 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)

In [59]: from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

         # Now apply those above metrics to evaluate your model
         predictions = clf.predict(X_test)

         print('The accuracy score is:',accuracy_score(y_test,predictions))
         print('The confusion matrix is:','\n',confusion_matrix(y_test,predictions))
         print('The classification report is:','\n',classification_report(y_test,predictions))

         The accuracy score is: 0.884
```



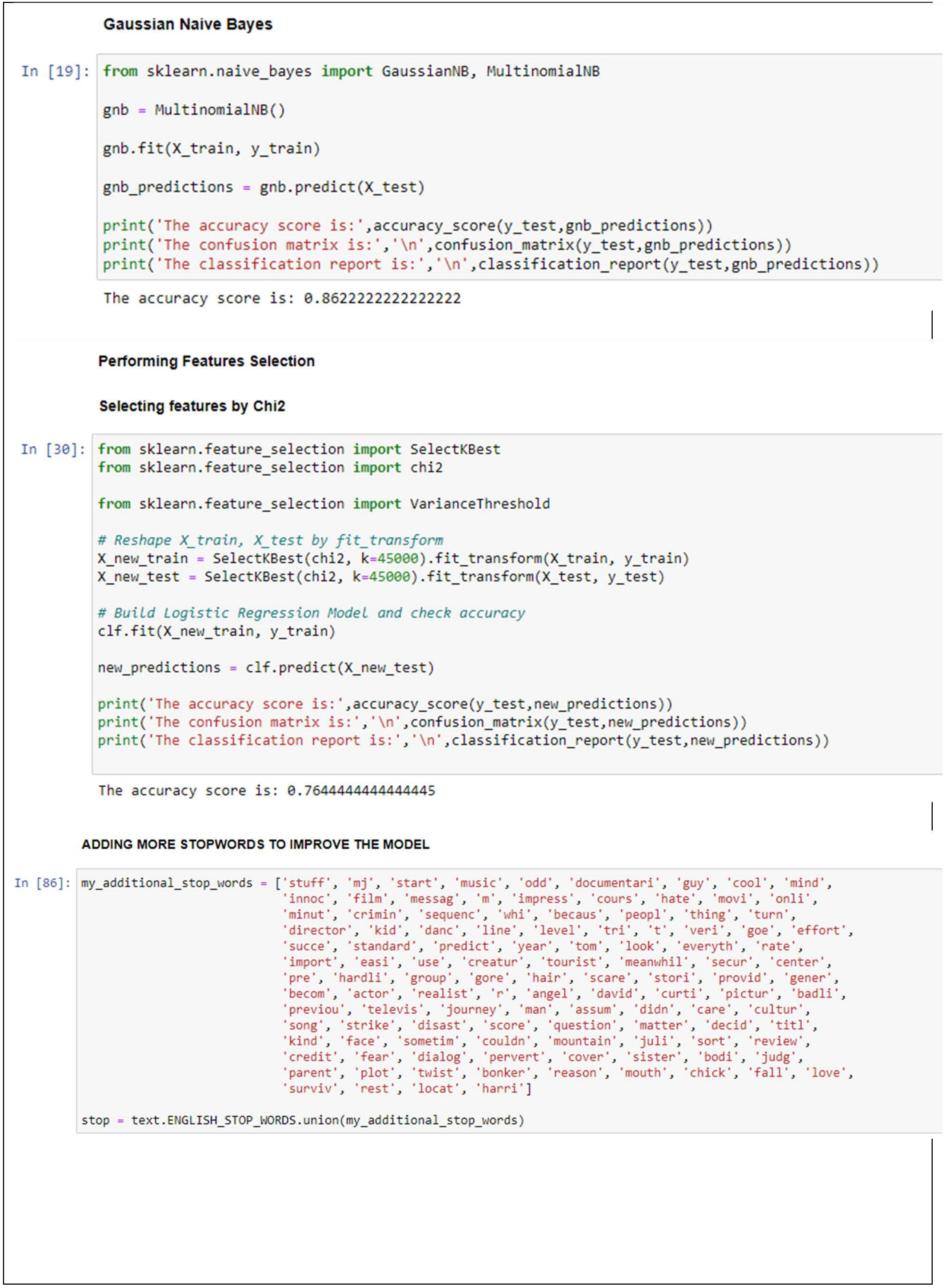
Decision Tree

```
In [60]: from sklearn.tree import DecisionTreeClassifier

In [61]: dtc = DecisionTreeClassifier()
         dtc.fit(X_train,y_train)

Out[61]: DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
                     max_features=None, 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, presort=False, random_state=None,
                     splitter='best')

In [62]: dtc_predictions = dtc.predict(X_test)

In [63]: print('The accuracy score is:',accuracy_score(y_test,dtc_predictions))
         print('The confusion matrix is:','\n',confusion_matrix(y_test,dtc_predictions))
         print('The classification report is:','\n',classification_report(y_test,dtc_predictions))

         The accuracy score is: 0.7088888888888889
```

Random Forest

Baseline Model

```
In [49]: from sklearn.ensemble import RandomForestClassifier

In [50]: rfc = RandomForestClassifier()
         rfc.fit(X_train, y_train)

Out[50]: 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=10, n_jobs=None,
                     oob_score=False, random_state=None, verbose=0,
                     warm_start=False)

In [51]: rfc_predictions = rfc.predict(X_test)

In [52]: print('The accuracy score is:',accuracy_score(y_test,rfc_predictions))
         print('The confusion matrix is:','\n',confusion_matrix(y_test,rfc_predictions))
         print('The classification report is:','\n',classification_report(y_test,rfc_predictions))

         The accuracy score is: 0.732
```



Gaussian Naive Bayes

```
In [19]: from sklearn.naive_bayes import GaussianNB, MultinomialNB

         gnb = MultinomialNB()

         gnb.fit(X_train, y_train)

         gnb_predictions = gnb.predict(X_test)

         print('The accuracy score is:',accuracy_score(y_test,gnb_predictions))
         print('The confusion matrix is:','\n',confusion_matrix(y_test,gnb_predictions))
         print('The classification report is:','\n',classification_report(y_test,gnb_predictions))
```

```
The accuracy score is: 0.8622222222222222
```

Performing Features Selection

Selecting features by Chi2

```
In [30]: from sklearn.feature_selection import SelectKBest
         from sklearn.feature_selection import chi2

         from sklearn.feature_selection import VarianceThreshold

         # Reshape X_train, X_test by fit_transform
         X_new_train = SelectKBest(chi2, k=45000).fit_transform(X_train, y_train)
         X_new_test = SelectKBest(chi2, k=45000).fit_transform(X_test, y_test)

         # Build Logistic Regression Model and check accuracy
         clf.fit(X_new_train, y_train)

         new_predictions = clf.predict(X_new_test)

         print('The accuracy score is:',accuracy_score(y_test,new_predictions))
         print('The confusion matrix is:','\n',confusion_matrix(y_test,new_predictions))
         print('The classification report is:','\n',classification_report(y_test,new_predictions))
```

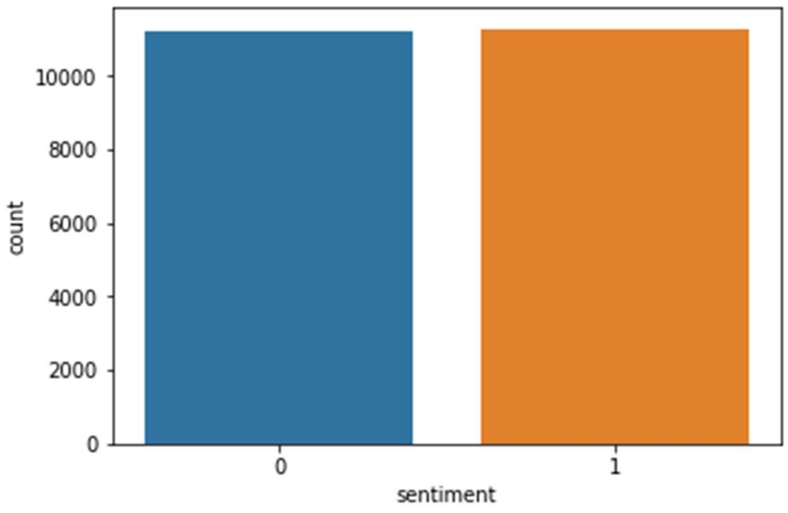
```
The accuracy score is: 0.7644444444444445
```

ADDING MORE STOPWORDS TO IMPROVE THE MODEL

```
In [86]: my_additional_stop_words = ['stuff', 'mj', 'start', 'music', 'odd', 'documentari', 'guy', 'cool', 'mind',
                                     'innoc', 'film', 'messag', 'm', 'impress', 'cours', 'hate', 'movi', 'onli',
                                     'minut', 'crimin', 'sequenc', 'whi', 'becaus', 'peopl', 'thing', 'turn',
                                     'director', 'kid', 'danc', 'line', 'level', 'tri', 't', 'veri', 'goe', 'effort',
                                     'succe', 'standard', 'predict', 'year', 'tom', 'look', 'everyth', 'rate',
                                     'import', 'easi', 'use', 'creatur', 'tourist', 'meanwhil', 'secur', 'center',
                                     'pre', 'hardli', 'group', 'gore', 'hair', 'scare', 'stori', 'provid', 'gener',
                                     'becom', 'actor', 'realist', 'r', 'angel', 'david', 'curti', 'pictur', 'badli',
                                     'previou', 'televis', 'journey', 'man', 'assum', 'didn', 'care', 'cultur',
                                     'song', 'strike', 'disast', 'score', 'question', 'matter', 'decid', 'titl',
                                     'kind', 'face', 'sometim', 'couldn', 'mountain', 'juli', 'sort', 'review',
                                     'credit', 'fear', 'dialog', 'pervert', 'cover', 'sister', 'bodi', 'judg',
                                     'parent', 'plot', 'twist', 'bonker', 'reason', 'mouth', 'chick', 'fall', 'love',
                                     'surviv', 'rest', 'locat', 'harri']

         stop = text.ENGLISH_STOP_WORDS.union(my_additional_stop_words)
```

Assumptions:

Data was collected from 22500 user-created movie reviews from web portal at https://www.kaggle.com/datasets/nltkdata/movie-review?select=movie_review.csv and is known as "Sentiment Polarity Dataset version 2.0": 11278 positive and 11222 negative processed reviews.



There are three columns in the movie review dataset:

- Id
- Review
- Sentiment

There are two classes in the movie review dataset:

- 1 : >= 7 rating
- 0: <5 rating

Project Diagrams:

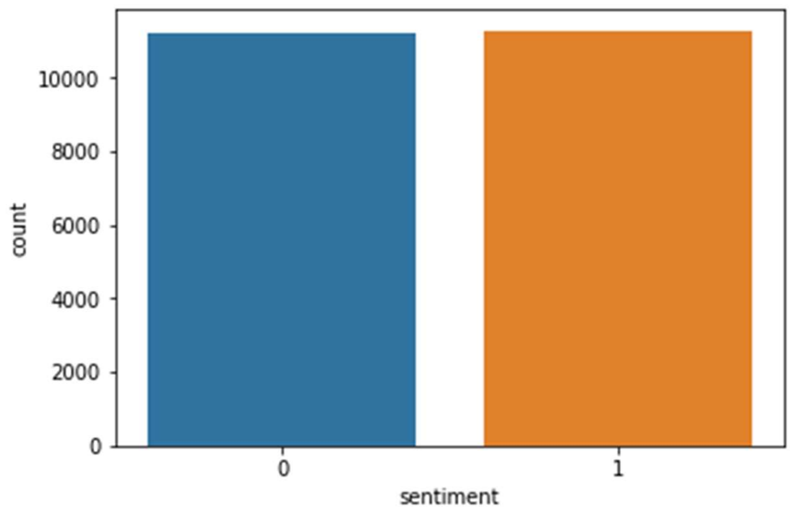


Fig 1: Data Distribution

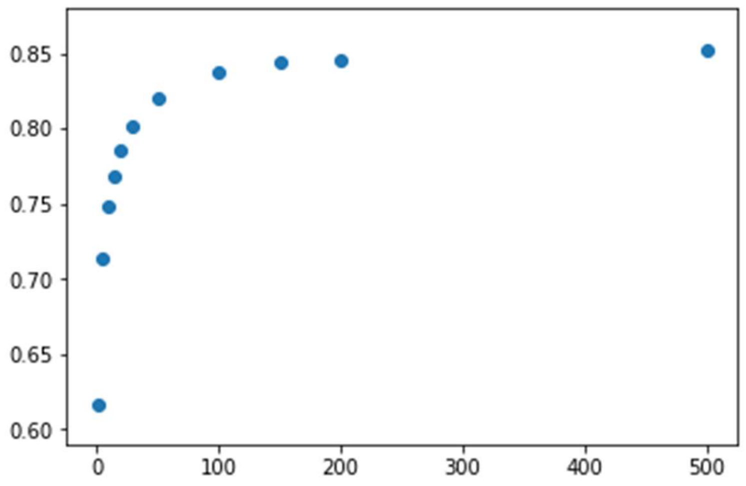


Fig 2: Scatter Plot – KNN Tuning Model

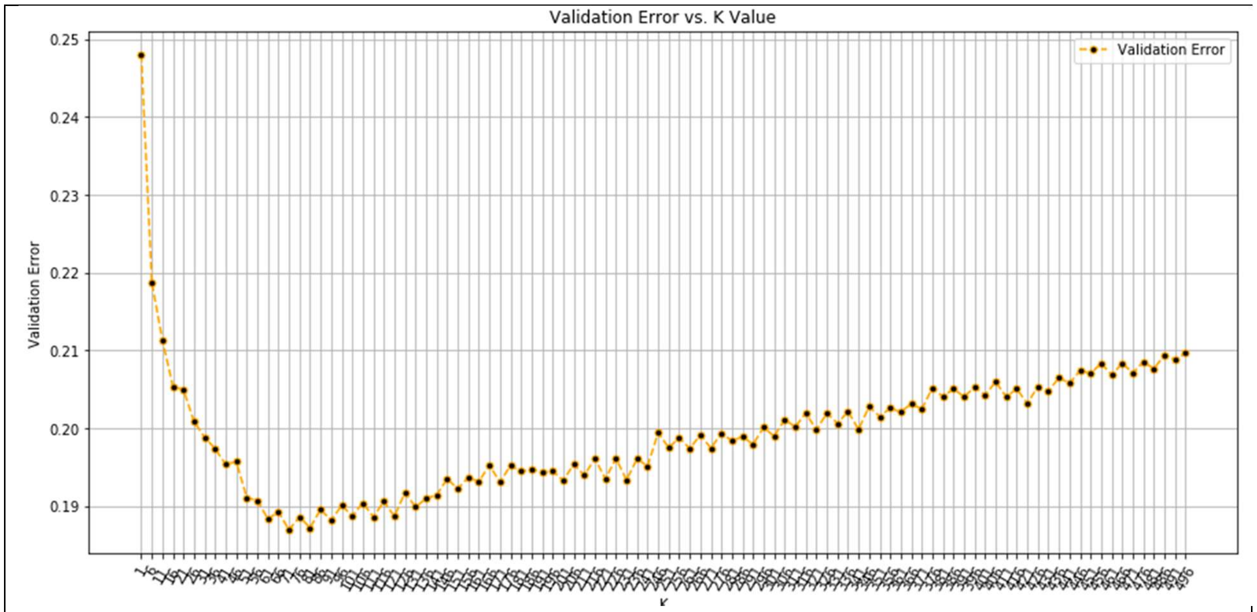


Fig 3: KNN – Validation Error vs K Value

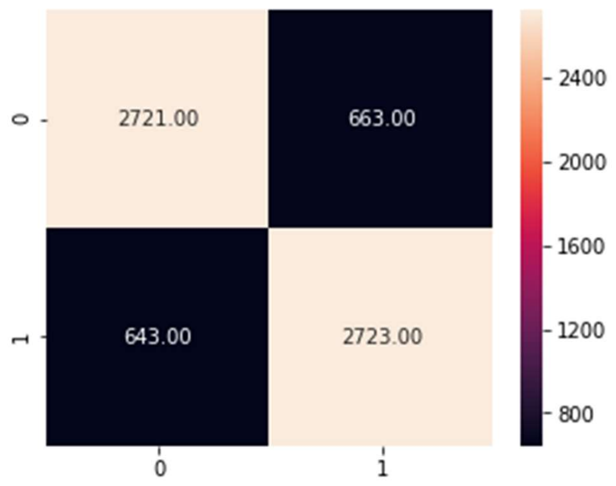


Fig 4: KNN Heatmap

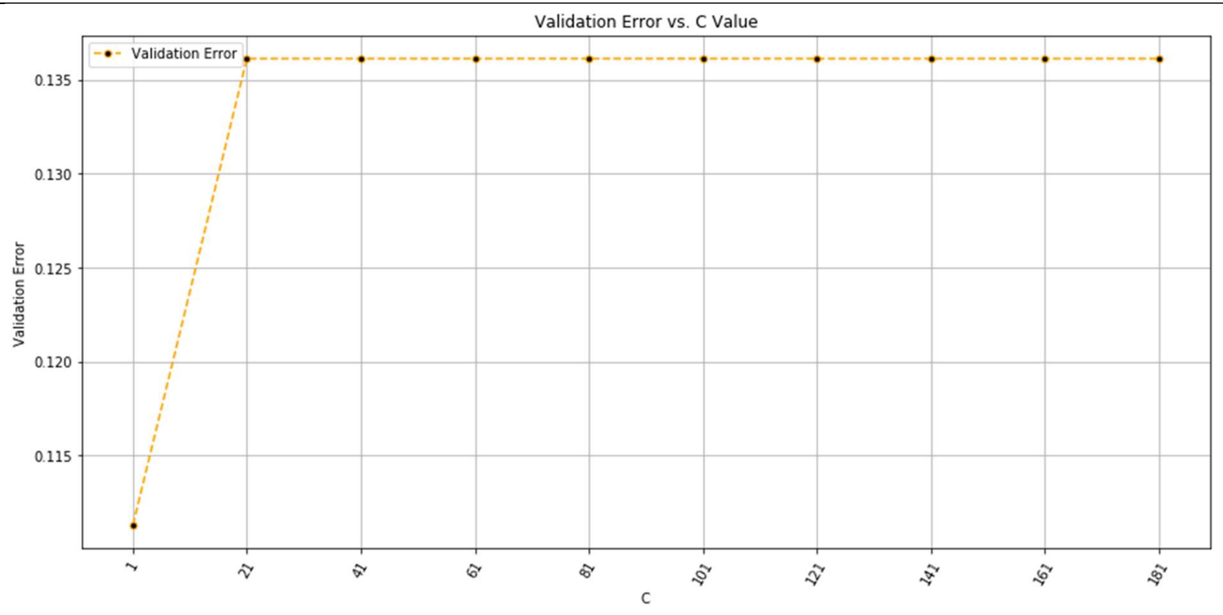


Fig 5: SVM – Validation Error vs C value

Algorithms:

Logistic Regression

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas:

Logit(pi) = 1/(1+ exp(-pi))
ln(pi/(1-pi)) = Beta_0 + Beta_1*X_1 + … + B_k*K_k

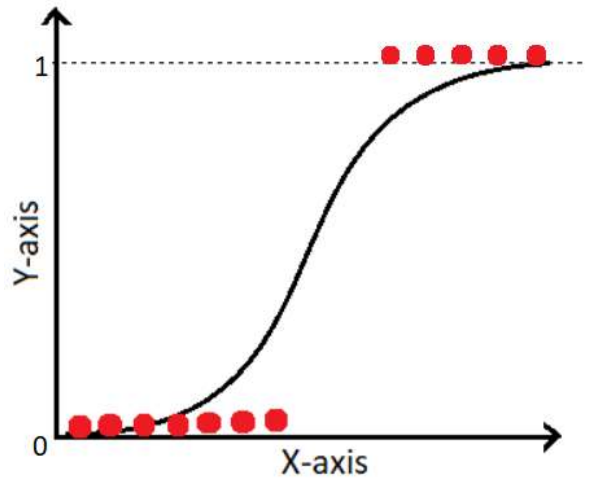


Fig 6: Logistic Regression

Decision Tree

Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves.

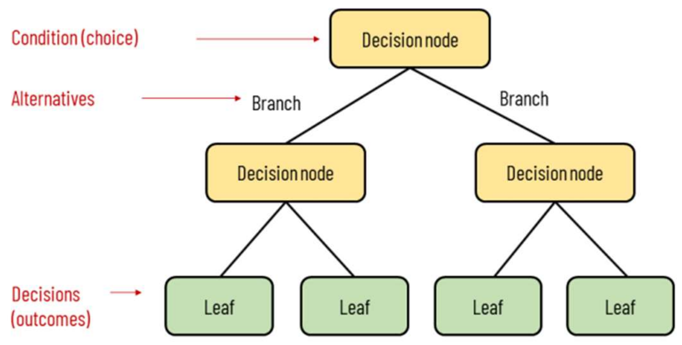


Fig 7: Decision Tree

Root Nodes – It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.
Decision Nodes – the nodes we get after splitting the root nodes are called Decision Node
Leaf Nodes – the nodes where further splitting is not possible are called leaf nodes or terminal nodes.

Random Forest

Random Forests are the learning method for classification and regression. It construct a number of decision trees at training time. To classify new case it sends the new case to each of the trees. Each tree perform classification and output a class. The output class is chosen based on majority voting that is the maximum number of similar class generated by various trees is considered as the output of the Random Forest. Random Forests are easy to learn and use for both professionals and laypeople with little research and programming required. It can easily be used by persons that don't have a strong statistical background.

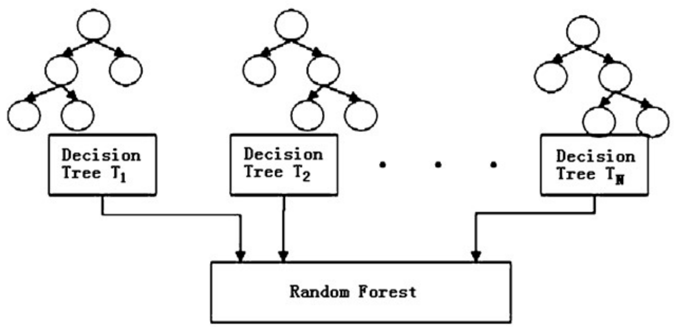


Fig 8: Random Forest

Naïve Bayes

It is a technique based on Bayes' Theorem. Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. This model is easy to build and particularly useful for very large datasets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.
P(C|X)= P(X|C) *P(C)/P(X)
P(C|X) is posterior probability of class C
P(C) is prior probability of class C
P(X|C) is probability of predictor given the class.
P(X) is prior probability of predictor

K- Nearest Neighbour

KNN is the simplest of all machine learning algorithms. The principle behind this method is to find a predefined number of training samples closest in distance to the new point and predict the label from these. The number of samples can be a user-defined constant or vary based on the local density of points. The distance can be any metric measure. Standard Euclidean distance is the most common choice for calculating the distance between two points. The Nearest Neighbours have been successful in a large number of classification and regression problems, including handwritten digits or satellite image processing and so on.

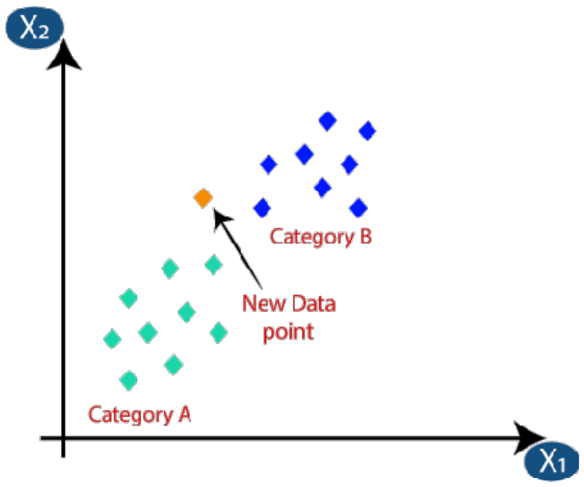


Fig 9: KNN

Support Vector Machine

Support Vector Machine" (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

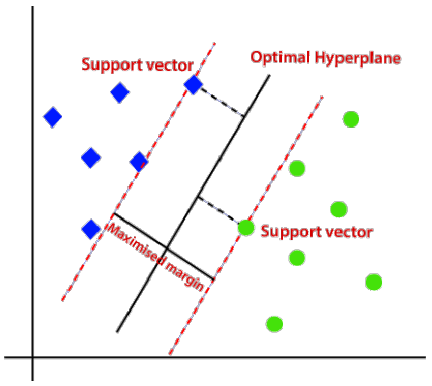


Fig 10: SVM

Outcome:

Logistic Regression Algorithm, Decision Tree Algorithm, Random Forest Algorithm, Navie Bayes Algorithm, K-Nearest Neighbour Algorithm and SVM were performed on the Data set. Results are presented in the form of figures and tables as shown below. Best accuracy was given by the Logistic Regression Algorithm.

ALGORITHM USED	ACCURACY
Logistic Regression	88.8%
Decision Tree	71.9%
Random Forest	85.7%
Naïve Bayes	66.8%
K-Nearest Neighbors	81.3%
Support Vector Machine	88.3%

Percentage Accuracy of Various Algorithms

Enhancement Scope:

In this project, various techniques were used to perceive the polarity of the movie reviews. The algorithms done were Logistic Regression, Decision Tree, Random Forest, Naïve Bayes, KNN, SVM. The high accuracy was given by Logistic Regression classifier. The Logistic Regression classifier completed with 88.8% accuracy.
As few algorithms had been tested, it's required to check other algorithms or create hybrid methods so that accuracy of the results can be extended.
Finding the polarity of the reviews can help in various domain. Intelligent systems can be developed which can provide the users with comprehensive reviews of movies, products, services etc. without requiring the user to go through individual reviews, he can directly take decisions based on the results provided by the intelligent systems.

Link to Code and executable file:
https://drive.google.com/drive/folders/1OhxiWJfYPUBkkAdZH7b0KfA78Zlcm3ms?usp=sharin
g