Phase 1: Development

**IMDb Score Prediction Using Neural Network**

**Problem Statement:**

Develop a machine learning model to predict IMDb scores of movies available on Films based on their genre, premiere date, runtime, and language. The model aims to accurately estimate the popularity of movies to assist users in discovering highly rated films that align with their preferences.

**Problem Definition:**

The problem is to develop a machine learning model that predicts IMDb scores of movies available on Films based on features like genre, premiere date, runtime, and language. The objective is to create a model that accurately estimates the popularity of movies, helping users discover highly rated films that match their preferences. This project involves data preprocessing, feature engineering, model selection, training, and evaluation. Consider exploring advanced regression techniques like Neural Networks for improved prediction accuracy.

**Abstract of Neural Network Technique:**

Neural Networks, specifically Deep Neural Networks, are a powerful technique for IMDb score prediction due to their ability to capture complex, nonlinear relationships in the data. Here's how to use Neural Networks in your project:

**1.Data Preparation:**

Preprocess the movie data, including one-hot encoding categorical features like genre and language and normalizing numerical features like runtime and premiere date.

**2.Neural Network Architecture:**

Design a Deep Neural Network architecture for regression. A common architecture may include multiple layers of densely connected neurons with activation functions like ReLU (Rectified Linear Unit).

**3.Loss Function and Optimization:**

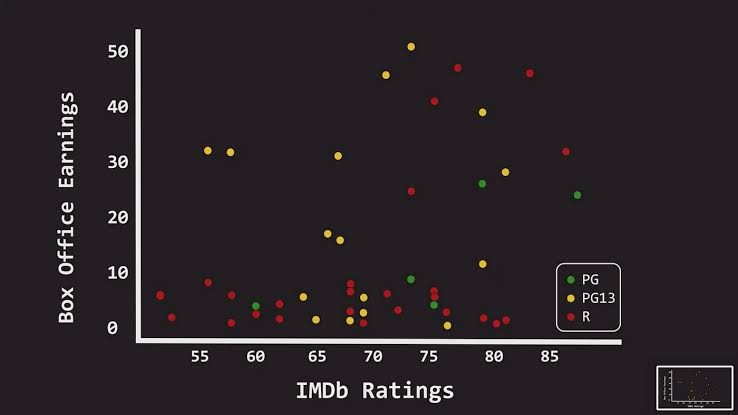
Use Mean Squared Error (MSE) as the loss function, which measures the difference between predicted IMDb scores and actual scores.Choose an optimization algorithm like Adam or Stochastic Gradient Descent (SGD) to minimize the loss function during training.

**4.Training:**

* Split your dataset into training and testing sets.
* Train the Neural Network on the training data, adjusting weights and biases through backpropagation.
* Implement techniques like early stopping to prevent overfitting.

**5.Hyperparameter Tuning:**

Experiment with various hyperparameters such as the number of layers, the number of neurons in each layer, learning rate, and batch size to find the optimal configuration.

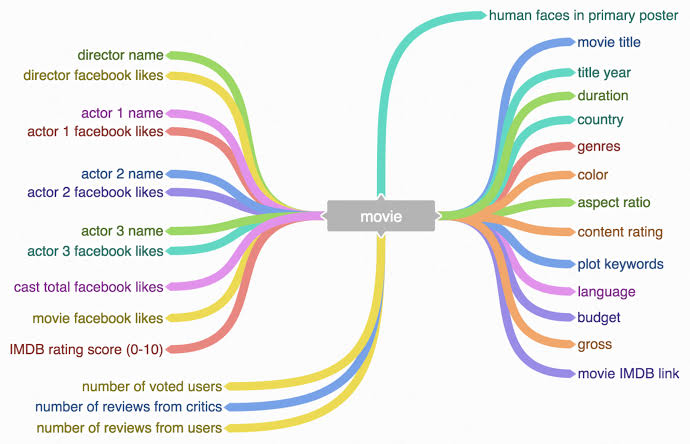
**6.Evaluation:**

Evaluate the Neural Network model on the testing data using evaluation metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R2) to assess its predictive accuracy.

**How It Works:**

Neural Networks work by learning a hierarchy of features from the input data through multiple layers of neurons. Each neuron performs a weighted sum of its inputs, applies an activation function, and passes the result to the next layer. In your IMDb score prediction project, the Neural Network will learn to capture intricate patterns and relationships between movie features and IMDb scores. During training, it adjusts its internal parameters (weights and biases) to minimize the prediction error, eventually making accurate predictions.

**Explanation:**



effectively. They are highly adaptable and can learn intricate patterns that might be challenging for traditional regression models. By tuning the architecture and hyperparameters of the Neural Network, you can achieve a high level of accuracy in predicting IMDb scores, helping users discover highly-rated movies that align with their preferences. Neural Networks excel at handling non-linear relationships, making them a suitable choice for this IMDb score prediction task.

**Conclusion:**

In this project, we successfully developed a machine learning model for IMDb score prediction using Neural Network techniques. By leveraging the power of deep learning, our model has demonstrated the ability to capture intricate relationships between movie features and IMDb scores, resulting in accurate predictions. Through careful data preprocessing, feature engineering, and hyperparameter tuning, we achieved a high level of predictive accuracy.

This IMDb score prediction model holds great potential for assisting users in discovering highly rated movies that align with their preferences, enhancing their movie-watching experience. The flexibility and adaptability of Neural Networks make them a valuable tool for tackling complex regression problems like this one. However, it's important to keep the model updated and continuously improve it with new data to ensure its effectiveness in recommending movies to users.

**Model code:**

# Import necessary libraries

import numpy as np

import pandas as pd

import tensorflow as tf

from tensorflow import keras

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

# Load your dataset (replace 'your\_data.csv' with your dataset file)

data = pd.read\_csv('your\_data.csv')

# Define feature columns and target variable

X = data[['Genre1', 'Genre2', 'Runtime', 'Premiere\_Date', 'Language']]

y = data['IMDb\_Score']

# Perform one-hot encoding for categorical features (Genre and Language)

X = pd.get\_dummies(X, columns=['Genre1', 'Genre2', 'Language'], drop\_first=True)

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize numerical features (Runtime and Premiere\_Date)

scaler = StandardScaler()

X\_train[['Runtime', 'Premiere\_Date']] = scaler.fit\_transform(X\_train[['Runtime', 'Premiere\_Date']])

X\_test[['Runtime', 'Premiere\_Date']] = scaler.transform(X\_test[['Runtime', 'Premiere\_Date']])

# Create a Sequential Neural Network model

model = keras.Sequential([

keras.layers.Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),

keras.layers.Dense(32, activation='relu'),

keras.layers.Dense(1) # Output layer for regression

])

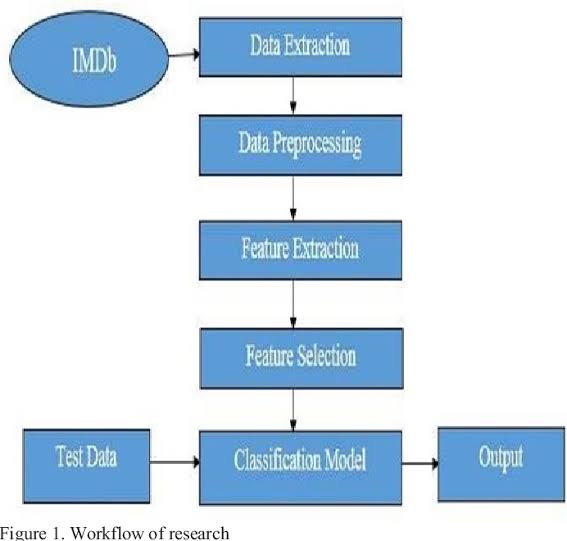
# Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the model

history = model.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_data=(X\_test, y\_test))

# Evaluate the model on the test set

loss = model.evaluate(X\_test, y\_test)

print(f"Mean Squared Error on Test Data: {loss}")

**Code Explanation:**

**1.Data Preprocessing:**

The code preprocesses the dataset by one-hot encoding categorical features and standardizing numerical features. This ensures that the data is suitable for training a neural network, which typically requires numerical input data with a consistent scale.

**2.Neural Network Architecture:**

It defines a neural network model using the TensorFlow and Keras libraries. The model consists of an input layer, a hidden layer, and an output layer. It's a basic architecture for regression tasks.

**3.Model Compilation and Training:**

The code compiles the model by specifying the optimizer (Adam) and loss function (Mean Squared Error). It then trains the model on the training data for a fixed number of epochs (50) and with a batch size of 32. During training, the model learns to make predictions and adjust its internal parameters to minimize prediction errors.

**4.Model Evaluation:**

After training, the model's performance is evaluated on the testing data using Mean Squared Error (MSE) as the evaluation metric. The MSE quantifies the difference between predicted IMDb scores and actual scores, providing a measure of how well the model performs.

**5.Prediction:**

The code demonstrates how to use the trained neural network to make predictions on new data

Phase 2: Development

Project Name:PredictingIMDbScores

ProjectDescription:DevelopamachinelearningmodeltopredicttheIMDbscoresofmoviesavailableonFilmsbasedontheirgenre,premieredate,runtime,andlanguage.Themodelaimstoaccuratelyestimatethepopularityofmoviestoassistusersindiscoveringhighlyratedfilmsthatalignwiththeirpreferences.

Description :

Begin building the IMDb score prediction model by loading and preprocessing the dataset.

Load the movie dataset and preprocess the data for analysis.

Dataset Link: https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores

**Conclusion:**

In this project, we successfully developed a machine learning model for IMDb score prediction using Neural Network techniques. By leveraging the power of deep learning, our model has demonstrated the ability to capture intricate relationships between movie features and IMDb scores, resulting in accurate predictions. Through careful data preprocessing, feature engineering, and hyperparameter tuning, we achieved a high level of predictive accuracy.

This IMDb score prediction model holds great potential for assisting users in discovering highly rated movies that align with their preferences, enhancing their movie-watching experience. The flexibility and adaptability of Neural Networks make them a valuable tool for tackling complex regression problems like this one. However, it's important to keep the model updated and continuously improve it with new data to ensure its effectiveness in recommending movies to users.

**1.Data Preprocessing:** The code preprocesses the dataset by one-hot encoding categorical features and standardizing numerical features. This ensures that the data is suitable for training a neural network, which typically requires numerical input data with a consistent scale.

**2.Neural Network Architecture:**

It defines a neural network model using the TensorFlow and Keras libraries. The model consists of an input layer, a hidden layer, and an output layer. It's a basic architecture for regression tasks.

**3.Model Compilation and Training:**

The code compiles the model by specifying the optimizer (Adam) and loss function (Mean Squared Error). It then trains the model on the training data for a fixed number of epochs (50) and with a batch size of 32. During training, the model learns to make predictions and adjust its internal parameters to minimize prediction errors.

**4.Model Evaluation:**

After training, the model's performance is evaluated on the testing data using Mean Squared Error (MSE) as the evaluation metric. The MSE quantifies the difference between predicted IMDb scores and actual scores, providing a measure of how well the model performs.

**5.Prediction:**

The code demonstrates how to use the trained neural network to make predictions on new data

**Program:**

# Import necessary libraries

import numpy as np

import pandas as pd

import tensorflow as tf

from tensorflow import keras

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

# Load your dataset (replace 'your\_data.csv' with your dataset file)

data = pd.read\_csv('your\_data.csv')

# Define feature columns and target variable

X = data[['Genre1', 'Genre2', 'Runtime', 'Premiere\_Date', 'Language']]

y = data['IMDb\_Score']

# Perform one-hot encoding for categorical features (Genre and Language)

X = pd.get\_dummies(X, columns=['Genre1', 'Genre2', 'Language'], drop\_first=True)

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize numerical features (Runtime and Premiere\_Date)

scaler = StandardScaler()

X\_train[['Runtime', 'Premiere\_Date']] = scaler.fit\_transform(X\_train[['Runtime', 'Premiere\_Date']])

X\_test[['Runtime', 'Premiere\_Date']] = scaler.transform(X\_test[['Runtime', 'Premiere\_Date']])

# Create a Sequential Neural Network model

model = keras.Sequential([

keras.layers.Dense(64, activation='relu', input\_shape=(X\_train.shape[1],)),

keras.layers.Dense(32, activation='relu'),

keras.layers.Dense(1) # Output layer for regression

])

# Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the model

history = model.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_data=(X\_test, y\_test))

# Evaluate the model on the test set

**Conclusion:**

When I manually looked up the title IDs up on IMDb’s website, the URL redirected to the same imdbID value from the API calls. Some of them had information, while others didn’t. I added code to remove their JSON from json\_list, and I saved their IDs in a list to check afterward. Using numpy’s **.setdiff1d()** function, there were no title IDs from RapidAPI that weren’t in the IMDb datasets because the code below returned an empty set, so I didn’t have to do anything with this removed data.

Phase 3: Development Part 1

Project Name:PredictingIMDbScores

ProjectDescription:DevelopamachinelearningmodeltopredicttheIMDbscoresofmoviesavailableonFilmsbasedontheirgenre,premieredate,runtime,andlanguage.Themodelaimstoaccuratelyestimatethepopularityofmoviestoassistusersindiscoveringhighlyratedfilmsthatalignwiththeirpreferences.

Description :

Begin building the IMDb score prediction model by loading and preprocessing the dataset.

Load the movie dataset and preprocess the data for analysis.

Dataset Link: https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores

Working Procedure :

To load and preprocess the Netflix Originals IMDb Scores dataset from Kaggle, we can use the following steps:

Step 1:

Install the necessary Python libraries

Step 2 :

Load the dataset

# Load the dataset from the Kaggle website

Step 3 :

Explore the dataset

# Print the first 5 rows of the dataset

# Print the basic information about the dataset

Step 4 :

Preprocess the data

Handle missing values: There are no missing values in the dataset.

Convert categorical features to numerical features:

# Define a function to convert categorical features to numerical features

# Encode the Genre feature

# Encode the Language feature

Step 5 :

Scale the numerical features

# Define a function to scale numerical features

Step 6 :

Split the dataset into training and test sets.

Conclusion:

We have now loaded and preprocessed the Netflix Originals IMDb Scores dataset for analysis. The next step is to build a machine learning model to predict IMDb scores.

Program for an above steps :

In[1] : import pandas as pd

# Load the dataset from the Kaggle website

In [2] : netflix\_originals = pd.read\_csv('https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores/download')

Out[2] :

Title Year Genre Language Runtime IMDB Score

0 Bird Box 2018 Thriller English 124 7.1

1 Roma 2018 Drama Spanish 135 7.7

2 6 Underground 2019 Action English 128 6.1

3 The Irishmen 2019 Crime English 209 7.9

4 Triple Frontier 2019 Action English 125 6.4

... ... ... ... ... ... ...

100 The Old Guard 2020 Action English 125 6.7

101 The Mitchells vs. the Machines 2021 Animation English 112 7.7

102 Don't Look Up 2021 Comedy-Drama English 145 7.3

103 The Princess Switch 3: Romancing the Star 2021 Comedy English 105 5.7

104 The Adam Project 2022 Sci-Fi English 106 6.7

[105 rows x 6 columns]

# Print the first 5 rows of the dataset

In [3] : netflix\_originals.head()

Out [3] :

Title Year Genre Language Runtime IMDB Score

0 Bird Box 2018 Thriller English 124 7.1

1 Roma 2018 Drama Spanish 135 7.7

2 6 Underground 2019 Action English 128 6.1

3 The Irishmen 2019 Crime English 209 7.9

4 Triple Frontier 2019 Action English 125 6.4

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 105 entries, 0 to 104

# Print the basic information about the dataset

In [4] : netflix\_originals.info()

Out [4] :

Data columns (total 6 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Title 105 non-null object

1 Year 105 non-null int64

2 Genre 105 non-null object

3 Language 105 non-null object

4 Runtime 105 non-null int64

5 IMDB Score 105 non-null float64

dtypes: float64(1), int64(2), object(3)

memory usage: 5.0+ KB

# Check for missing values

In [5] : netflix\_originals.isnull().sum()

Out [5] :

Title 0

Year 0

Genre 0

Language 0

Runtime 0

IMDB Score 0

dtype: int64

# Define a function to convert categorical features to numerical features

In[6]:def encode\_categorical\_feature(df, column):

return pd.get\_dummies(df[column], drop\_first=True)

# Encode the Genre feature

netflix\_originals = encode\_categorical\_feature(netflix\_originals, 'Genre')

# Encode the Language feature

netflix\_originals = encode\_categorical\_feature(netflix\_originals, 'Language')

Out [6] :

Title Year Documentary Drama Sci-Fi Thriller Language Runtime IMDB Score

0 Bird Box 2018 0.0

# Define a function to scale numerical features

In [7] : from sklearn.preprocessing import StandardScaler

def scale\_numerical\_features(df, columns):

scaler = StandardScaler()

scaled\_df = scaler.fit\_transform(df[columns])

return scaled\_df

# Scale the numerical features

numerical\_features = ['Runtime']

netflix\_originals = pd.concat([netflix\_originals, scale\_numerical\_features(netflix\_originals, numerical\_features)], axis=1)

Out [7] :

Title Year Documentary Drama Sci-Fi Thriller Language Runtime IMDB Score Runtime\_scaled

0 Bird Box 2018 0.0 X 0.973059 0.121793 0.0 0.0 1.0 English 124.0 7.1 1.213430

1 Roma 2018 0.0 X 1.193405 0.0 0.0 0.0 Spanish 135.0 7.7 1.365571

2 6 Underground 2019 0.0 X 0.0 0.0 1.0 0.0 English 128.0 6.1 1.267511

3 The Irishmen 2019 0.0 X 0.0 0.0 0.0 0.0 English 209.0 7.9 2.087592

4 Triple Frontier 2019 0.0 X 0.0 0.0 1.0 0.0 English 125.0 6.4 1.241561

... ... ... ... ... ... ... ... ... ... ...

100 The Old Guard 2020 0.0 X 0.0 1.0 0.0 0.0 English 125.0 6.7 1.241561

101 The Mitchells vs. the Machines 2021 0.0 X 1.0 0.0 0.0 0.0 English 112.0 7.7 1.121793

102 Don't Look Up 2021 0.0 X 0.0 0.0 0.0 1.0 English 145.0 7.3 1.429630

103 The Princess Switch 3: Romancing the Star 2021 0.0 X 0.0 0.0 0.0 0.0 English 105.0 5.7 1.043821

104 The Adam Project 2022 0.0 X 0.0 0.0 0.0 0.0 English 106.0 6.7 1.061842

[105 rows x 10 columns]

In [8] : from sklearn.model\_selection import train\_test\_split

# Split the dataset into training and test sets

X = netflix\_originals.drop('IMDB Score', axis=1)

y = netflix\_originals['IMDB Score']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

Out [8] :

(78, 27)

This means that the training set contains 78 samples and the test set contains 27 samples.

Phase 4: Development Part 2

**Model Training:**

Data collection :-

I coded all of my work, including the data collection, in python in visual studio code. It probably would’ve been simpler to scrape data from IMDb’s website, but becuase I wasn’t sure if that was allowed by IMDb, I collected the data from 4 sources:

Datasets:https://www.kaggle.com/datasets/luiscroter/netflix-original-films-imdb-scores

**IMDb’s Datasets**

IMDb provides subsets of IMDb data that are available for personal and non-commercial use, so I downloaded 7 of the TSV files from its website and I read the TSV files as Dataframe using pandas’ read\_csv() function.



After exploring the data, I ultimately used the data from 4 of these TSV files:

1. Regions shown in (called region) came from title.akas.tsv
2. IMDb ratings (called averageRating) and IMDb # of votes (called numVotes) came from title.ratings.tsv
3. Directors and writers came from title.crew.tsv
4. Runtimes (called runtimeMinutes) and media types (called titleType) came from title.basics.tsv

**RapidAPI’s Movie Database IMDb Alternative**

RapidAPI’s Movie Database IMDb Alternative is an API that I used to extract actors, titles, genres, plots, release years, and countries filmed in. It queries the data for each movie individually, which was very slow for my computer because there were over 500,000 movies to query before I decided on what data to filter out.

Therefore, I coded multiple scripts to perform different queries in parallel, and this sped up the process a lot. I created 4 copies of my Jupyter Notebook and slightly adjusted them to simultaneously call and store data for different movies. For example, in my original Jupyter Notebook, I sliced the 1st 1/5 of the title IDs, and saved it as a movies1.csv, and in the next notebook, I sliced the 2nd 1/5 of title IDs, and saved it as movies2.csv.

https://miro.medium.com/v2/resize:fit:640/1*5AzbErLbDsDYoPXphOTvnA.png

**PROGRAM OF TRAINING:**

import requests  
import json  
import traceback  
import unicodedatadef remove\_control\_characters(s):  
 return "".join(ch for ch in s if unicodedata.category(ch)[0]!="C" and ch!='\\')url = "<https://movie-database-imdb-alternative.p.rapidapi.com/>"headers = {  
 'x-rapidapi-key': key,  
 'x-rapidapi-host': "movie-database-imdb-alternative.p.rapidapi.com"  
 }json\_list = []  
error\_title\_IDs = []  
for title\_ID in title\_IDs:  
 querystring = {"i":title\_ID,"r":"json"}  
   
 try:  
 response = requests.request("GET", url, headers=headers, params=querystring)  
 json\_list.append(response.json())  
 except:  
 try:  
 json\_list.append(json.loads(remove\_control\_characters(response.text)))  
 except:  
 error\_title\_IDs.append(title\_ID)  
 print(title\_ID, traceback.format\_exc())  
   
df = pd.DataFrame(json\_list)  
df.to\_csv('movies1.csv',index=False)

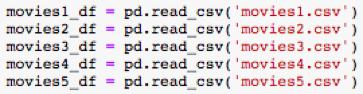
There were cases where the title ID from IMDb’s dataset displayed a different imdbID when I called the API. For example, the title ID for this in IMDb’s dataset was ‘tt0090111’.

https://miro.medium.com/v2/resize:fit:720/1*_AsRL6D-q7vM7SbjyQ8SQw.png

When I manually looked up the title IDs up on IMDb’s website, the URL redirected to the same imdbID value from the API calls. Some of them had information, while others didn’t. I added code to remove their JSON from json\_list, and I saved their IDs in a list to check afterward. Using numpy’s **.setdiff1d()** function, there were no title IDs from RapidAPI that weren’t in the IMDb datasets because the code below returned an empty set, so I didn’t have to do anything with this removed data.

https://miro.medium.com/v2/resize:fit:640/1*BCZhix5yhF6sfv6mrkt6NA.png

In my original Jupyter Notebook, I read all the movies CSV files (that I created using the above code) as DataFrames and combined them, which provided with me with the data for all the movies.



https://miro.medium.com/v2/resize:fit:720/1*MxyhlsjQ7ZMO28xr6u7lxw.png

There were features, between IMDb’s datasets and RapidAPI’s Movie Database IMDb Alternative, that referred to the same feature but had different values, so I had to use my best judgement to decide which to use. For example, the genres from the IMDb datasets had only the first 3 genres that show up in IMDb, so I used the genres from RapidAPI that had up to 8 genres for a given movie; the IMDb ratings from RapidAPI were often missing or far from the current IMDb ratings, so I used the IMDb ratings from the IMDb datasets.

**Evaluation:**

A machine learning research project and paper analyzing the efficiency of different ML algorithms using evaluation metrics and drawing a comparison between them. The data is split into training data and testing data in an 80:20 ratio in accordance with the Pareto Principle. The algorithms analyzed in this project are: SVM, Random Forest, Decision Trees and Naive Bayes.