**Phase 5: Project documentation and submission**

**Project title: Predicting IMDb Scores**

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

**Problem Statement**

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.

**Design Thinking**

**1.Data Source**

The data source for this project will be the dataset available at the provided link: [Netflix Original Films IMDb Scores Dataset](https://www.kaggle.com/datasets/luiscorter/netflix-original-films-imdb-scores). This dataset contains essential information about movies, including IMDb scores, genre, premiere date, runtime, language, and more. It will serve as the foundation for building the predictive model.

**2.Data Preprocessing**

Data Preprocessing is a vital step in ensuring the quality and reliability of the dataset. The following actions will be taken during this phase:

* Data Cleaning: Identify and address missing or inconsistent data to ensure a clean dataset.
* Data Transformation: Convert categorical features such as genre and language into numerical representations using techniques like one-hot encoding or label encoding.
* Feature Scaling: Normalize or standardize numerical features to bring them to a consistent scale.

**3.Feature Engineering**

Feature Engineering aims to enhance the dataset's quality and enable the model to make more accurate predictions. Some key feature engineering strategies include:

* Extracting relevant information from the premiere date, such as year or month, to capture any time-related patterns that may affect IMDb scores.
* Creating new features based on genre information, such as the count of genres a movie belongs to.
* Handling outliers, if present, through transformations or removal to reduce their impact on the model.

**4.Model Selection**

Selecting an appropriate regression model is a pivotal decision. Several regression algorithms will be considered, including:

* Linear Regression: A simple and interpretable model to establish a baseline.
* Random Forest Regressor: A versatile ensemble model that can capture non-linear relationships.
* Support Vector Regression: Effective for handling complex datasets and outliers.
* Gradient Boosting Regressor: A powerful ensemble method for predictive accuracy.

The choice of the model will depend on factors like dataset size, complexity, and the model's performance during preliminary evaluations.

**5.Model Training**

Once the regression model is selected, it will be trained using the preprocessed data. The dataset will be divided into a training set and a testing set to assess the model's performance. Hyperparameter tuning may be necessary to optimize the model's predictive capabilities.

**6.Model Evaluation**

Model Evaluation is essential to determine how well the model predicts IMDb scores. Key evaluation metrics will include:

* Mean Absolute Error (MAE): This measures the average absolute difference between predicted and actual IMDb scores.
* Mean Squared Error (MSE): It quantifies the average squared difference between predicted and actual IMDb scores.
* R-squared (R2): An indicator of how well the model explains the variance in IMDb scores.

These metrics will provide insights into the model's accuracy and its ability to meet the project's objectives.

**Code Implementation**

#import necessary libararies

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error,r2\_score

import matplotlib.pyplot as plt

#Load the dataset

df = pd.read\_csv("C:/Users/91720/OneDrive/Documents/IMDb\_dataset.csv", encoding='ISO-8859-1')

#explore the loaded dataset to get an overview of its structure and contents:

#display the first rows of the dataframe

print(df.head())

Title Genre Premiere Runtime \

0 Enter the Anime Documentary August 5, 2019 58

1 Dark Forces Thriller August 21, 2020 81

2 The App Science fiction/Drama December 26, 2019 79

3 The Open House Horror thriller January 19, 2018 94

4 Kaali Khuhi Mystery October 30, 2020 90

IMDB Score Language

0 2.5 English/Japanese

1 2.6 Spanish

2 2.6 Italian

3 3.2 English

4 3.4 Hindi

#display basic information about the dataframe

print(df.info())

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

RangeIndex: 584 entries, 0 to 583

Data columns (total 6 columns):

# Column Non-Null Count Dtype

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

0 Title 584 non-null object

1 Genre 584 non-null object

2 Premiere 584 non-null object

3 Runtime 584 non-null int64

4 IMDB Score 584 non-null float64

5 Language 584 non-null object

dtypes: float64(1), int64(1), object(4)

memory usage: 27.5+ KB

None

#display summary statistics

print(df.describe())

Runtime IMDB Score

count 584.000000 584.000000

mean 93.577055 6.271747

std 27.761683 0.979256

min 4.000000 2.500000

25% 86.000000 5.700000

50% 97.000000 6.350000

75% 108.000000 7.000000

max 209.000000 9.000000

#data preprocessing

#check for missing values

print(df.isnull().sum())

Title 0

Genre 0

Premiere 0

Runtime 0

IMDB Score 0

Language 0

dtype: int64

#handle missing values

df=df.dropna()

#remove duplicates

df=df.drop\_duplicates()

#convert data types

#df['column\_name']=df['column\_name'].astype('desired\_type')

#save the dataframe to a new CSV file

df.to\_csv(r'C:\Users\91720\OneDrive\Documents\cleaned\_dataset.csv',index=False)

#Data Preprocessing and feature engineering

#Assume 'feature1','feature2,feature3' are selected features

X=df[['Title','Genre','Premiere','Runtime']]

y=df['IMDB Score']

#data splitting

X\_train,X\_temp,y\_train,y\_temp=train\_test\_split(x,y,test\_size=0.3,random\_state=42)

X\_val,X\_test,y\_val,y\_test=train\_test\_split(X\_temp,y\_temp,test\_size=0.5,random\_state=42)

#Reshaping of arrays

X\_train\_np=X\_train.to\_numpy()

X\_val\_np=X\_val.to\_numpy()

X\_test\_np=X\_test.to\_numpy()

X\_train\_np = print(X\_train\_np.reshape(1, -1))

X\_val\_np = print(X\_val\_np.reshape(1, -1))

X\_test\_np = print(X\_test\_np.reshape(1, -1))

[[ 86 110 111 42 137 94 48 112 97 94 96 117 94 92 98 108 102 79

108 96 94 136 90 126 98 94 100 95 97 155 149 47 100 107 89 86

103 86 14 97 121 119 104 89 83 128 98 120 100 87 55 40 39 142

106 91 89 96 119 131 95 89 63 120 92 98 90 100 101 102 89 105

115 109 10 105 109 28 97 90 111 89 15 99 98 105 108 82 16 147

106 151 117 71 144 101 82 126 94 116 102 136 123 97 30 80 103 105

100 99 72 96 98 132 148 39 130 121 106 41 114 19 129 96 34 124

107 91 99 96 101 103 108 209 108 80 39 78 106 98 105 103 23 101

104 86 86 72 97 122 86 21 118 100 102 94 102 94 87 24 95 100

96 104 100 98 97 125 73 9 101 94 94 78 104 92 32 120 60 113

97 96 83 97 53 74 106 39 121 81 102 90 80 103 101 80 83 93

122 97 40 54 112 140 132 91 84 88 97 27 96 104 52 114 70 100

90 90 85 41 116 95 107 121 99 118 32 97 98 89 126 102 49 87

106 94 15 99 124 95 89 98 70 106 56 133 105 49 125 97 98 99

102 83 96 96 84 112 97 88 99 95 99 114 117 99 82 98 85 89

90 124 79 108 94 114 93 99 74 108 92 85 136 105 139 101 90 92

125 83 30 119 104 104 79 113 98 78 92 94 115 88 121 80 83 7

117 100 111 98 107 76 95 95 19 134 96 107 101 80 103 97 31 107

112 103 105 73 21 83 90 113 90 95 4 124 106 131 105 94 104 37

115 106 92 153 111 83 124 85 114 97 144 81 124 112 98 85 100 17

110 92 95 95 17 95 101 125 116 90 104 85 92 85 105 93 93 112

64 102 138 92 97 103 40 86 83 75 111 108 97 91 107 125 101 117

112 73 123 123 87 92 64 117 101 107 109 112]]

[[112 144 47 100 106 95 120 129 57 89 70 51 120 85 102 101 104 92

76 40 132 112 135 91 105 81 23 58 108 99 97 95 87 99 112 88

58 89 82 89 85 134 87 118 118 90 89 64 98 25 114 90 98 116

99 87 37 107 149 101 97 36 13 101 90 28 100 87 89 102 121 70

106 98 86 92 26 130 31 91 121 11 89 93 118 103 86 92]]

[[ 95 83 46 108 100 104 90 83 100 58 64 92 106 28 100 94 12 97

114 92 107 114 98 103 111 102 131 100 83 84 95 142 93 114 136 118

102 86 101 72 30 94 60 91 97 121 58 85 102 90 104 85 113 94

93 118 132 97 23 79 44 139 80 96 112 94 125 120 40 116 149 92

79 104 86 91 97 89 94 81 95 119 45 151 114 105 108 95]]

#feature Scaling

Scaler=StandardScaler()

X\_train\_np=Scaler.fit\_transform(X\_train\_np)

X\_val\_np=Scaler.transform(X\_val\_np)

X\_test\_np=Scaler.transform(X\_test\_np)

#model training

model=LinearRegression()

model.fit(X\_train\_np,y\_train)

#model evaluation

y\_pred=model.predict(X\_test\_np)

mse=mean\_squared\_error(y\_test,y\_pred)

rmse=np.sqrt(mse)

r2=r2\_score(y\_test,y\_pred)

print("Mean Squared Error(MSE) value:",mse)

print("Root Mean Squared Error(RMSE) value:",rmse)

print("R-squared(R2)score:",r2)

**Why Linear regression in Model Training?**

Linear Regression can be a valid choice for a baseline model when predicting IMDb scores, especially if you want interpretability and computational efficiency. However, it's essential to assess the model's performance and consider more complex models if the linear assumption does not hold or if you aim for higher predictive accuracy. The choice of model should be based on the specific characteristics of the dataset and the project's objectives.

* **Interpretability:** Linear Regression is a straightforward and interpretable model. It provides clear insights into the relationships between input features (such as genre, premiere date, runtime, and language) and the IMDb scores. This interpretability can be valuable when you want to understand the impact of individual features on the outcome.
* **Simplicity:** Linear Regression assumes a linear relationship between the input features and the target variable (IMDb scores). While this is a simplification, it can be effective when the relationship is primarily linear or close to linear. Linear models are less prone to overfitting in such cases, making them a good starting point.
* **Speed and Efficiency:** Linear Regression is computationally efficient and can be trained relatively quickly, especially on large datasets. This efficiency can be advantageous when you want to perform quick initial experiments or require real-time predictions.
* **Baseline Model:** Linear Regression can serve as a useful baseline model. You can start with a Linear Regression model, assess its performance, and then compare it to more complex models like Random Forest or Gradient Boosting. If Linear Regression provides satisfactory results, it may save computational resources and suffice for your prediction task.

**Conclusion**

This document outlines the initial steps for addressing the problem of predicting IMDb scores for Netflix Original Films. By utilizing the provided dataset and following the data preprocessing, feature engineering, model selection, training, and evaluation phases, the project aims to deliver a predictive model that enhances the user experience by recommending highly rated films. Subsequent phases of the project will involve the practical implementation of these steps, fine-tuning the model, and providing a valuable solution for Netflix Original Films enthusiasts.