PREDICTING IMDb SCORES

PHASE-4

Objective:

The objective of the project is to build the IMDb scores predicting model. In this phase we will continue building the IMDb scores prediction model by feature engineering, model training, evaluation.

Dataset Link:

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

Feature engineering:

Feature engineering refers to manipulation — addition, deletion, combination, mutation — of the data set to improve machine learning model training, leading to better performance and greater accuracy.

Creating new features gives a deeper understanding of the data and results in more valuable insights. When done correctly, feature engineering is one of the most valuable techniques of data science, but it is also one of the most challenging.

Feature engineering in ML contains mainly four processes:

- Feature Creation
- Transformations
- Feature Extraction
- Feature Selection

Feature Creation:

Feature creation is finding the most useful variables to be used in a predictive model. The process is subjective, and it requires human creativity and intervention.

Transformations:

The transformation step of feature engineering involves adjusting the predictor variable to improve the accuracy and performance of the model.

Feature Extraction:

Feature extraction is an automated feature engineering process that generates new variables by extracting them from the raw data. The main aim of this step is to reduce the volume of data so that it can be easily used and managed for data modelling. Feature extraction methods include cluster analysis, text analytics, edge detection algorithms, and principal components analysis (PCA).

Feature Selection:

Reducing the input variable to your model by using only relevant data and getting rid of noise in data. It is the process of automatically choosing relevant data and getting rid of noise in data.

Model training:

Model training is the phase in the data science development lifecycle where practitioners try to fit the best combination of weights and bias to a machine learning algorithm to minimize a loss function over the prediction range. Train the selected model using the training dataset. The model learns the relationships

between the input features (movie attributes) and the target variable (IMDb scores). There are 3 steps to train a machine learning model:

Step 1: Begin with existing data.

Step 2: Analysis data to identify patterns.

Step 3: Make predictions.

Evaluation:

To evaluate the model's performance, we can use metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE) or R-squared. These metrics measure how well your model's predictions match the actual IMDb scores.

Program:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error,
mean_squared_error, r2_score
from sklearn.ensemble import GradientBoostingRegressor

# loading the data set
```

```
data = pd.read_csv("NetflixOriginals.csv",encoding='latin-1')
print('\n')

print('DEVELOPMENT PART 2')
print('-------------------\n')
# Split data into features (X) and target (y)
X = data[['Genre','Premiere','Runtime', 'Language']]
y = data['IMDB Score']
# Split the 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)
print('Linear Regression model')
print('-----------------------\n')
# Choose a regression algorithm
model = LinearRegression()

# Initialize the Gradient Boosting Regressor
gb_regressor = GradientBoostingRegressor()
# Model Training
model.fit(X_train, y_train)
# Fit the model to your training data
gb_regressor.fit(X_train, y_train)

# Model Evaluation
y_pred = model.predict(X_test)
```

```
# Make predictions
predictions = gb_regressor.predict(X_test)

# by using LinearRegression method
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)



print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("R-squared:", r2)

print('\nGradient Boosting Regressor')
print('---------------------------\n')
# by using Gradient Boosting Regressor
mae = mean_absolute_error(y_test, predictions)
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)



print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
print(f'R-squared; {r2}’)
```

Output:

```
DEVELOPMENT PART 2
-------------------

Linear Regression model
------------------------

Mean Absolute Error: 0.8155336353328348
Mean Squared Error: 0.994440830285098
R-squared: 0.04191044117818454

Gradient Boosting Regressor
----------------------------

Mean Absolute Error: 0.7311737159548155
Mean Squared Error: 0.8821936141014356
R-squared: 0.1500545182888865
PS C:\Users\annsi\OneDrive\Documents>
```