Phase-4

Predicting IMDB Scores

Feature Engineering

Feature engineering is a crucial step in improving the performance of machine learning models for predicting IMDB scores or any other kind of regression problem. IMDb scores are typically associated with movie ratings, so the features you engineer can be related to various aspects of the movies.

Below python code divides the movies into long, medium and short based on movie's runtime.

```
data = {'Movie': ['Movie1', 'Movie2', 'Movie3'],
        'Runtime': [90, 125, 160]}
df = pandas.DataFrame(data)
bins = [0, 100, 150, float('inf')]
labels = ['Short', 'Medium', 'Long']
df['Runtime_Category'] = pd.cut(df['Runtime'], bins=bins, labels=labels)
category_mapping = {'Short': 1, 'Medium': 2, 'Long': 3}
df['Runtime_Category'] = df['Runtime_Category'].map(category_mapping)
print(df)
```

```
data = {'Movie': ['Movie1', 'Movie2', 'Movie3'],
        'Runtime': [90, 125, 160]}
df = pandas.DataFrame(data)
bins = [0, 100, 150, float('inf')]
labels = ['Short', 'Medium', 'Long']
df['Runtime_Category'] = pd.cut(df['Runtime'], bins=bins, labels=labels)
category_mapping = {'Short': 1, 'Medium': 2, 'Long': 3}
df['Runtime_Category'] = df['Runtime_Category'].map(category_mapping)
print(df)
```

```
    Movie  Runtime Runtime_Category
0  Movie1       90                1
1  Movie2      125                2
2  Movie3      160                3
```

Model Training:

The training process for predicting IMDB scores involves the following:

Data Splitting:

```
from sklearn.model_selection import train_test_split
```

```
X = data[['sentiment','Runtime']] # Features
y = data['IMDB Score']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

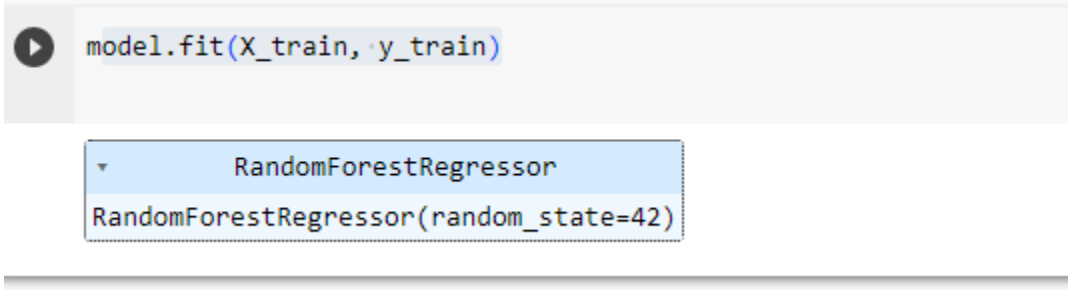
Model Selection:

```
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor(n_estimators=100, random_state=42)
```

Model Training:

```
odel.fit(X_train, y_train)
```



Evaluation:

The below python code for evaluating the data of IMDB:

```
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)
```

```python
from sklearn.metrics import mean_squared_error, r2_score

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)
```

```
Mean Squared Error: 1.6862085227621486
R-squared (R2) Score: -0.34047099938390657
```