**KINGS ENGINEERING COLLEGE**

**PROJECT TITLE : Applied Data Science - Predicting IMDB scores**

The goal of predicting IMDb scores can be defined as follows:

**Project Goal:**

To develop an accurate and reliable predictive model for IMDb scores of movies and TV shows. The aim is to leverage various features such as genre, cast, crew, plot summary, production budget, and user reviews, among others, to forecast the IMDb rating of a movie or TV show. The prediction model should be able to analyze these factors and provide a numerical score that closely aligns with the actual IMDb ratings.

**Objectives:**

**Data Collection:** Gather comprehensive and relevant data from IMDb and other reliable sources, including information about movies and TV shows, such as cast and crew details, genres, plot summaries, user reviews, and IMDb ratings.

**Data Preprocessing:** Cleanse and preprocess the collected data, handling missing values, outliers, and ensuring consistency. Convert textual data into numerical or categorical features suitable for machine learning algorithms.

**Feature Selection:** Identify and select the most influential features that significantly contribute to predicting IMDb scores. This may involve feature engineering to create new meaningful features.

**Model Selection:** Explore and evaluate various machine learning algorithms such as regression models, decision trees, ensemble methods, or neural networks. Choose the most appropriate algorithm(s) for the prediction task based on performance metrics like Mean Absolute Error (MAE) or Root Mean Square Error (RMSE).

**Model Training:** Train the selected model(s) using a substantial portion of the dataset. Implement techniques like cross-validation to ensure the model's generalizability and prevent overfitting.

**Evaluation:** Assess the model's performance using appropriate evaluation metrics on a separate test dataset that the model has not seen before. Refine the model if necessary to improve accuracy and reliability.

**Deployment:** Develop a user-friendly interface or integrate the model into a platform where users can input movie or TV show details, and the system provides a predicted IMDb score. Ensure the deployment environment is scalable and reliable.

**Monitoring and Maintenance:** Implement a system for continuous monitoring of the model's performance. Regularly update the model using new data to ensure its predictions remain accurate over time. Address issues such as concept drift or changes in user preferences that might affect the model's performance.

**Interpretability:** Provide insights into which features most heavily influence the IMDb scores. This information can be valuable for filmmakers, producers, and other stakeholders to understand what aspects of a movie or TV show contribute most to its audience reception.

**Deploying a model for predicting IMDb scores publicly for reviews:**

**1. Choose a Deployment Platform:**

Select a suitable platform for deploying your predictive model. Options include cloud platforms like AWS, Google Cloud, or Azure, or using serverless computing services like AWS Lambda or Google Cloud Functions.

**2. API Development:**

Develop a RESTful API that interacts with your predictive model. This API will handle incoming requests, pass the required input data to the model, and return the predicted IMDb scores.

**3. Model Serialization:**

Serialize your trained machine learning model into a format that can be easily loaded by the API. Common formats include pickle (for Python), ONNX, or TensorFlow SavedModel.

**4. Input Validation:**

Implement input validation in your API to ensure that the incoming data meets the required format and constraints. Handle errors gracefully and provide meaningful error messages to users for invalid inputs.

**5. Security Measures:**

Implement security measures such as API key authentication, HTTPS, and input validation to prevent misuse and unauthorized access to the API.

**6. Rate Limiting:**

Implement rate limiting to prevent abuse of the API. Define limits on the number of requests a user or IP address can make within a specific time frame.

**7. Documentation:**

Create comprehensive API documentation that explains the endpoints, expected input formats, and response structures.Include examples to help users understand how to interact with the API.

**8. Testing:**

Thoroughly test the API using various test cases to ensure it behaves as expected and provides accurate predictions. Consider unit tests, integration tests, and endpoint tests.

**9. Monitoring and Logging:**

Set up monitoring and logging mechanisms to track API usage, errors, and performance metrics. Use tools like Prometheus, Grafana, or ELK stack for monitoring.

**10. Scaling:**

Design the deployment architecture to handle varying loads. Consider auto-scaling options offered by cloud platforms to automatically adjust resources based on demand.

**11. User Interface (Optional):**

Create a user-friendly web interface or mobile app that allows users to input movie or TV show details and displays the predicted IMDb score obtained from the API.

**12. Compliance and Regulations:**

Ensure compliance with relevant data protection regulations (such as GDPR) and obtain necessary permissions to use and process user data.

**13. Launch and Maintenance:**

Launch the API and continuously monitor its performance. Regularly update the model with new data to maintain prediction accuracy. Address any issues promptly to provide users with a reliable service.

**Develop a python script on applied data science predicting imdb scores:**

Developing a Python script for predicting IMDb scores involves several steps, including data loading, preprocessing, feature selection, model training, and prediction. Below is an example Python script using libraries such as pandas, scikit-learn, and joblib for model persistence. Please note that you'll need to have your dataset (CSV, Excel, etc.) containing relevant features for the prediction, and you should replace 'your\_dataset.csv' with the actual path to your dataset file.

**Import necessary libraries:**

python code

import pandas as pd

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error

import joblib

# Load the dataset (replace 'your\_dataset.csv' with the actual path to your dataset file)

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

# Data Preprocessing

# Example: Assuming 'budget', 'cast', 'director', 'genre', 'runtime' are important features

features = ['budget', 'cast', 'director', 'genre', 'runtime']

X = data[features]

y = data['imdb\_score']

# Convert categorical features to numerical using techniques like one-hot encoding

X = pd.get\_dummies(X, columns=['cast', 'director', 'genre'])

# 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)

# Model Training

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

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

# Model Evaluation

predictions = model.predict(X\_test)

mae = mean\_absolute\_error(y\_test, predictions)

print(f'Mean Absolute Error: {mae:.2f}')

# Save the trained model to a file for future use

joblib.dump(model, 'imdb\_score\_predictor.pkl')

**In this script:**

**Data Preprocessing:** Features like 'cast', 'director', and 'genre' are converted into numerical representations using one-hot encoding. Numerical features are kept as they are.

**Model Training:** A Random Forest Regressor model is used for prediction. You can experiment with other algorithms provided by scikit-learn based on your dataset and problem requirements.

**Model Evaluation:** Mean Absolute Error (MAE) is used as the evaluation metric. You can choose other metrics based on your preference.

**Model Persistence:** The trained model is saved to a file named 'imdb\_score\_predictor.pkl' using joblib. This saved model can be loaded and used for making predictions in the future without retraining the model.