Here is a structured outline for a big data analysis project on Movie Prediction using Python. I've placed relevant information and steps under each heading. The specific content can be adjusted depending on the dataset used.

Title of Project:

Movie Success Prediction Using Big Data Analysis

Objective:

The main objective of this project is to build a machine learning model that predicts the success of a movie based on several factors, such as budget, genre, cast, director, and more. Success can be measured by box office performance or ratings.

Data Source:

- The Movie Database (TMDb) API

- Kaggle Movie Datasets (e.g., IMDB dataset, TMDB box office prediction dataset)

- Box Office Mojo

- Rotten Tomatoes

Import Library:

python

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.linear\_model import LinearRegression

from sklearn.ensemble import RandomForestRegressor

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

Import Data:

```python

# Example: Load a dataset from a CSV file

movie\_data = pd.read\_csv('movies\_metadata.csv')

```

Describe Data:

```python

# Overview of the dataset

print(movie\_data.info())

print(movie\_data.describe())

print(movie\_data.head())

# Check for missing values

print(movie\_data.isnull().sum())

```

Explanation: The dataset may include columns like movie title, director, budget, revenue, genres, and ratings. Understanding missing data and the structure of the dataset is crucial.

Data Visualization:

```python

# Visualizing the distribution of budget vs revenue

sns.scatterplot(x='budget', y='revenue', data=movie\_data)

plt.title('Budget vs Revenue')

plt.show()

# Genre-based revenue

sns.barplot(x='genres', y='revenue', data=movie\_data)

plt.xticks(rotation=90)

plt.title('Revenue by Genre')

plt.show()

```

Data Preprocessing:

```python

# Dropping irrelevant columns

movie\_data = movie\_data.drop(['homepage', 'imdb\_id', 'overview', 'poster\_path'], axis=1)

# Handling missing values by dropping or filling in missing entries

movie\_data.fillna(movie\_data.mean(), inplace=True)

# Converting release\_date to datetime format

movie\_data['release\_date'] = pd.to\_datetime(movie\_data['release\_date'])

# Convert categorical columns (like genres) into numerical data using one-hot encoding

movie\_data = pd.get\_dummies(movie\_data, columns=['genres', 'original\_language'], drop\_first=True)

```

Define Target Variable (y) and Feature Variables (X):

```python

# Defining features and target variables

X = movie\_data[['budget', 'runtime', 'popularity', 'vote\_average', 'vote\_count']]

y = movie\_data['revenue']

# If we are doing classification (e.g., predicting success/failure)

# We can create a new target column 'success' based on revenue thresholds:

# y = movie\_data['revenue'].apply(lambda x: 1 if x > some\_threshold else 0)

```

Train Test Split:

```python

# Splitting the dataset into training and testing sets

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

```

Modeling:

1. Linear Regression:

```python

# Linear Regression Model

lr\_model = LinearRegression()

lr\_model.fit(X\_train, y\_train)

# Predictions

y\_pred\_lr = lr\_model.predict(X\_test)

```

2. Random Forest Regressor:

```python

# Random Forest Regressor Model

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

rf\_model.fit(X\_train, y\_train)

# Predictions

y\_pred\_rf = rf\_model.predict(X\_test)

```

Model Evaluation:

1. Linear Regression Evaluation:

```python

# Mean Squared Error & R-squared for Linear Regression

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

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

print(f"Linear Regression MSE: {mse\_lr}")

print(f"Linear Regression R2 Score: {r2\_lr}")

```

2. Random Forest Evaluation:

```python

# Mean Squared Error & R-squared for Random Forest

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

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

print(f"Random Forest MSE: {mse\_rf}")

print(f"Random Forest R2 Score: {r2\_rf}")

```

Prediction:

```python

# Predict the revenue for a new movie with certain features

new\_movie = np.array([[50000000, 120, 80, 7.5, 2000]]) # Budget, Runtime, Popularity, Vote Average, Vote Count

predicted\_revenue = rf\_model.predict(new\_movie)

print(f"Predicted Revenue for the new movie: ${predicted\_revenue[0]}")

```

Explanation:

1. Linear Regression: This model provides a basic interpretation of how the features affect the target, but it may not capture non-linear relationships.

2. Random Forest Regressor: This model often performs better for complex datasets due to its ability to capture non-linear patterns and interactions between variables.

3. Feature Importance: Random Forest can provide insights into the most important features influencing the revenue.

```python

# Feature importance from the Random Forest model

importances = rf\_model.feature\_importances\_

feature\_names = X.columns

sorted\_importances = sorted(zip(importances, feature\_names), reverse=True)

for importance, feature in sorted\_importances:

print(f"Feature: {feature}, Importance: {importance}")

```

This structure gives a comprehensive guide on setting up a movie success prediction project using Python, suitable for big data analysis.