Module 1 - Task 1

- Load the train data.
- Import Pandas and alias it as 'pd'.
- Read the CSV file movies training_data.csv into a Pandas DataFrame named 'train'.
- To import the 'training_data.csv' file, which is located in the root path of your project, you should use the following path: './training_data.csv'.
- Inspect the data by calling the variable 'train'.

Module 1 - Task 2

- Load the test data.
- Read the CSV file movies test_data.csv into a Pandas DataFrame named 'test'.
- To import the 'test_data.csv' file, which is located in the root path of your project, you should use the following path: './test_data.csv'.
- Inspect the data by calling the variable 'test'.

Module 1 - Task 3

- Finding Duplicates in train data.
- Calculate the number of duplicate rows in the DataFrame 'train' using the duplicated() method and then sum them up using the sum() method.
- Display the total number of duplicate rows, which is stored in the variable 'duplicates_train'.

Module 1 - Task 4

- Counting the Null Values in train data.
- Apply the .isnull() method to 'train' to identify and mark null values, returning a DataFrame with True/False values.
- Use the .sum() method on the result to count the number of null values in each column.
- Store the count of null values in the variable 'null_values_train'.

Module 1 - Task 5

- Finding Duplicates in test data.
- Calculate the number of duplicate rows in the DataFrame 'test' using the duplicated() method and then sum them up using the sum() method.
- Display the total number of duplicate rows, which is stored in the variable 'duplicates_test'.

Module 1 - Task 6

- Counting the Null Values in test data.
- Apply the .isnull() method to 'test' to identify and mark null values, returning a DataFrame with True/False values.
- Use the .sum() method on the result to count the number of null values in each column.
- Store the count of null values in the variable 'null_values_test'.

Module 2 - Task 1

- Calculate Ratio of 'TotVolDon' to 'NoDon'.
- Create a new variable named `ratio_totno`.
- Calculate the ratio by dividing the 'TotVolDon' column by the 'NoDon' column in the `train` dataframe.
- Store the result in the `ratio_totno` variable.

Module 2 - Task 2

- Drop Column 'TotVolDon' from the Train Dataframe.
- Drop the column named 'TotVolDon' from the `train` dataset.
- Use the `drop` method with the 'TotVolDon' column and axis set to 1 (indicating column) in the `train` dataset.
- Perform the operation in-place by setting the `inplace` parameter to True.

Module 2 - Task 3

- Prepare Data for Model Training.
- Create a variable `lastcoltarget` to store the 'DonMar2007' column from the `train` dataset.
- Drop the 'DonMar2007' column from the `train` dataset using the `drop` method with `axis=1` and `inplace=True`.
- Calculate the variable `no_period_first_donation` by dividing 'MonFirstDon' by 3.
- Calculate the variable `avg_don_per_period` by dividing 'NoDon' by `no_period_first_donation`.
- Insert the calculated 'AveDonPerPeriod' column at index 3 in the `train` dataset.
- Create a copy of the modified `train` dataset as `X` and the 'DonMar2007' column as `Y`.
- Use `train_test_split` to split the data into training and testing sets (X_train, X_test, y_train, y_test) with a test size of 20% and a random state of 42.

Module 3 - Task 1

- Logistic Regression Model Evaluation.
- Import necessary modules: `accuracy_score` from `sklearn.metrics` and `LogisticRegression` from `sklearn.linear_model`.
- Create a `LogisticRegression` model named `logistic_model`.
- Fit the model on the training data using the `fit` method.
- Make predictions on the test data(test) using the `predict` method.
- Calculate the accuracy of the model using the `accuracy_score` function.
- Round the accuracy value to 2 decimal places for better readability.

Module 3 - Task 2

- Logistic Regression Model Predictions.
- Use the trained logistic regression model (`logistic_model`) to make predictions on the test data(test).
- Create a DataFrame (`predictions_df`) to compare actual and predicted values, with columns 'Actual' and 'Predicted'.