Name : Kabilan N
Batch : July 2023
Domain : Data Science

Loan Prediction Report

Introduction:

In this report, we analyze loan data and perform a linear regression analysis to predict loan amounts based on selected features. The report is divided into two main sections: Data Preprocessing and Linear Regression Analysis.

Data Description:

Columns:

Loan_ID: A unique identifier for each loan application.

Gender: The gender of the applicant (Male/Female).

Married: Whether the applicant is married or not (Yes/No).

Dependents: The number of dependents the applicant has (0, 1, 2, 3+).

Education: The education level of the applicant (Graduate/Not Graduate).

Self_Employed: Whether the applicant is self-employed or not (Yes/No).

ApplicantIncome: The income of the applicant.

CoapplicantIncome: The income of the co-applicant (if any).

LoanAmount: The amount of the loan applied for.

Loan_Amount_Term: The term (in months) of the loan.

Credit_History: The credit history of the applicant (1=Good, 0=Bad).

Property_Area: The area where the property is located (Urban/Semiurban/Rural).

Loan_Status: Whether the loan was approved or not (Y/N). **Data**

Approach:

The approach taken for this project involves the following steps:

Preprocessing:

We start by preparing the dataset through several steps of data preprocessing:

1. Importing Libraries: We import necessary libraries for data manipulation and visualization.

Name : Kabilan N
Batch : July 2023
Domain : Data Science

```
import numpy as np import pandas as pd import seaborn as sns
```

2. Data Loading: The training and test datasets are loaded using Pandas DataFrames, named **train_df** and **test_df** respectively.

```
train_df = pd.read_csv("train.csv") test_df = pd.read_csv("test.csv")
```

3. Data Exploration: We inspect the basic information of the datasets using the **info()** method to understand their structure and data types. Additionally, we identify missing values using the **isna().sum()** method.

```
train_df.info() test_df.info()
```

4. Handling Missing Values: Any occurrences of " ?" in the datasets are replaced with NaN values using Numpy's **np.nan**. This step ensures uniformity in representing missing data.

```
train_df.replace(" ?", np.nan, inplace=True) test_df.replace(" ?",
                    np.nan, inplace=True)
```

5. Dropping Columns: The "Loan_ID" column is dropped from both datasets since it is not relevant to our analysis.

```
train_df = train_df.drop(columns=["Loan_ID"])
test_df.drop(columns=["Loan_ID"])
```

6. Label Encoding: Categorical columns are label-encoded using sklearn's **LabelEncoder** to convert them into numerical values. This ensures that the data can be used for further analysis and modeling.

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

for column in train_df.columns:

      if train_df[column].dtype == 'object':

      train_df[column] =label_encoder.fit_transform(train_df[column])


for column in test_df.columns:

    if test_df[column].dtype == 'object':

      test_df[column] = label_encoder.fit_transform(test_df[column])
```

Algorithm :

Linear Regression Analysis:

In this section, we perform linear regression analysis to predict loan amounts based on selected features:

Name : Kabilan N
Batch : July 2023
Domain : Data Science

1. Importing Necessary Libraries: We import required libraries for data imputation, modeling, and evaluation.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.impute import SimpleImputer

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score
```

2. Feature Selection: We choose a subset of columns ("ApplicantIncome", "CoapplicantIncome", "Loan_Amount_Term", "Property_Area") for analysis.

```
selected_cols =
["ApplicantIncome","CoapplicantIncome",Loan_Amount_Term",
"Property_Area"]
```

3. Data Transformation: The "Property_Area" values are mapped to corresponding labels (Urban, Suburban, Rural) to enhance readability.

```
train_df['Property_Area'] = train_df['Property_Area'].map({'1':
'Urban', '2': 'Suburban', '3': 'Rural'})

test_df['Property_Area'] = test_df['Property_Area'].map({'1': 'Urban',
'2': 'Suburban', '3': 'Rural'})
```

4. Data Imputation: We use **SimpleImputer** to fill missing values in the selected features and target ("LoanAmount") columns for both training and test datasets.

```
X_train = train_df[selected_cols]

y_train = train_df["LoanAmount"]

imputer_X = SimpleImputer(strategy="median")

imputer_y = SimpleImputer(strategy="median")

X_train_imputed = imputer_X.fit_transform(X_train)

y_train_imputed =
imputer_y.fit_transform(y_train.values.reshape(1,1)).flatten()

X_test = test_df[selected_cols]

y_test = test_df["LoanAmount"]

X_test_imputed = imputer_X.transform(X_test)

y_test_imputed =
imputer_y.transform(y_test.values.reshape(1,1)).flatten()
```

5. Data Splitting: The dataset is split into training and testing sets using the **train_test_split** function, with 80% for training and 20% for testing.

```
X_train_final, X_test_final, y_train_final, y_test_final =
train_test_split( X_train_imputed, y_train_imputed, test_size=0.2,
random_state=42)
```

6. Linear Regression Model: A Linear Regression model is trained on the training data using sklearn's **LinearRegression** class.

```
model = LinearRegression() model.fit(X_train_final, y_train_final)
```

7. Model Evaluation: Predictions are made on the test data, and the R-squared score is calculated using the **r2_score** function to assess the model's performance.

```
y_pred = model.predict(X_test_final) r2 = r2_score(y_test_final,
y_pred) print("R-squared:", r2 * 100)
```

Evaluation :

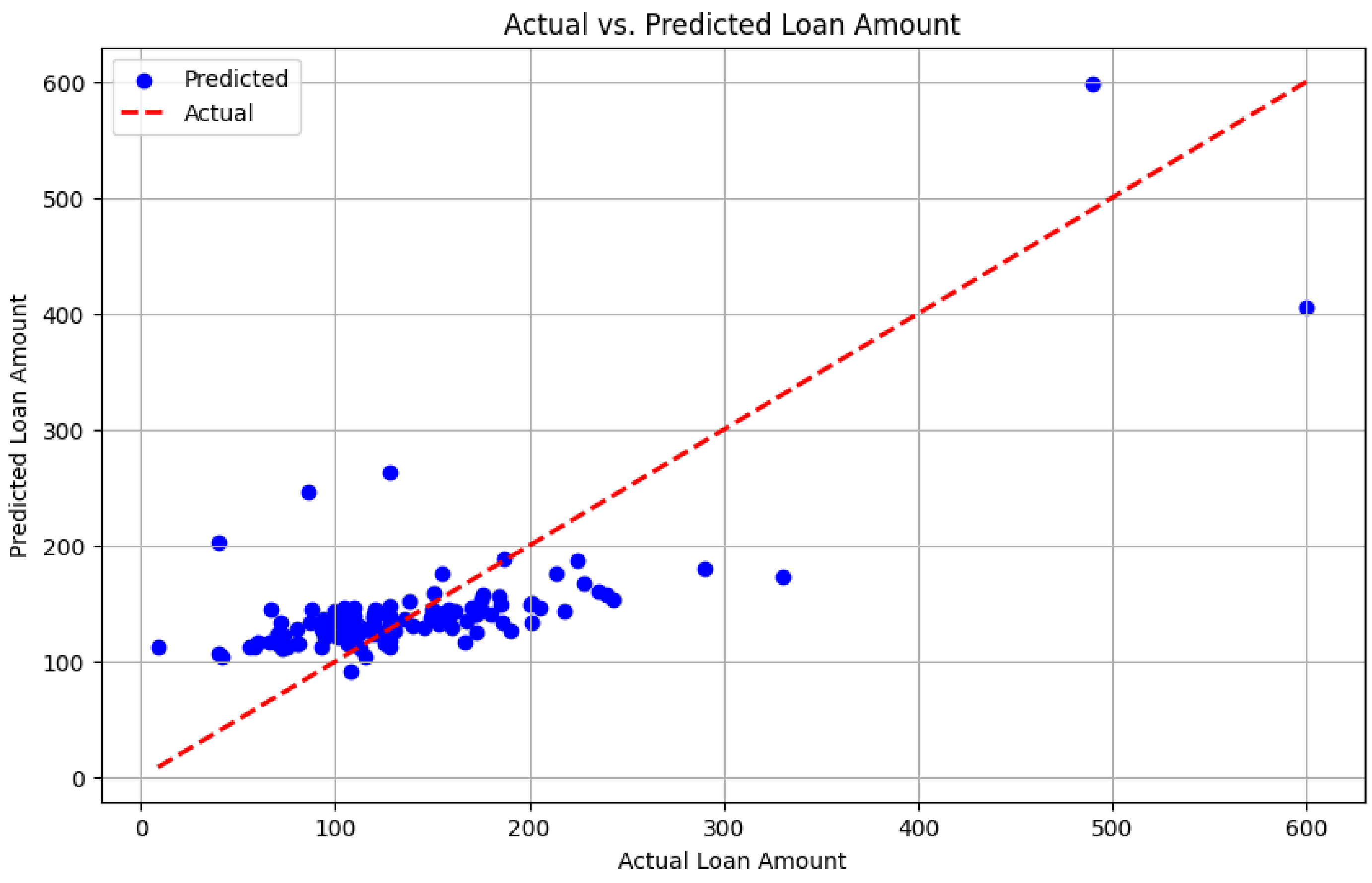
```
R-squared: 52.263618314207584
```

Visualization:

Code:

```
plt.figure(figsize=(10, 6))

plt.scatter(y_test_final, y_pred, color='blue', label='Predicted')

plt.plot([min(y_test_final), max(y_test_final)], [min(y_test_final),
max(y_test_final)], linestyle='--', color='red', linewidth=2, label='Actual')

plt.title('Actual vs. Predicted Loan Amount')

plt.xlabel('Actual Loan Amount')

plt.ylabel('Predicted Loan Amount')

plt.legend()

plt.grid(True)

plt.show()
```

Name : Kabilan N
Batch : July 2023
Domain : Data Science

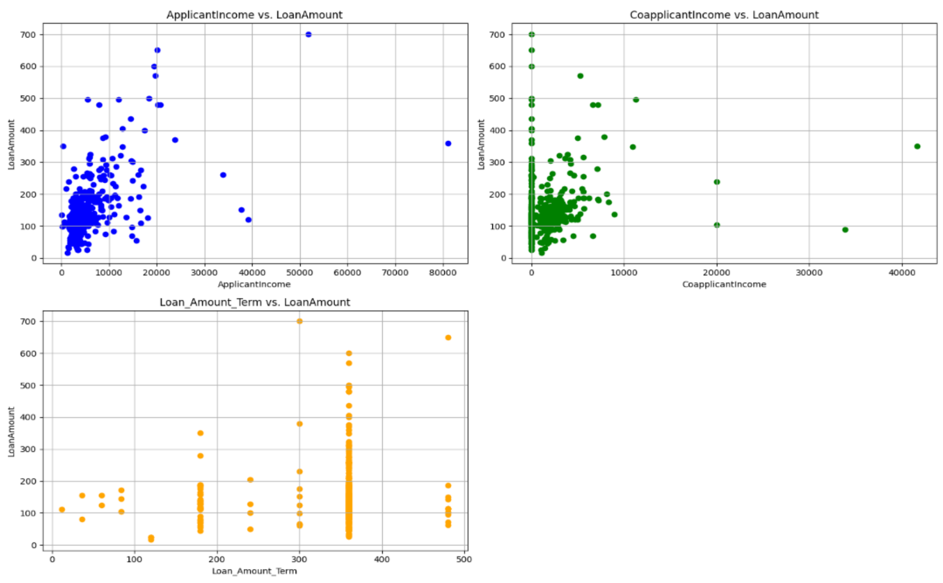


Relation between graphs:

```
plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)

plt.scatter(X_train_final[:, 0], y_train_final, color='blue', label='Data
Points')

plt.xlabel('ApplicantIncome')

plt.ylabel('LoanAmount')

plt.title('ApplicantIncome vs. LoanAmount')

plt.grid(True)

plt.subplot(2, 2, 2)

plt.scatter(X_train_final[:, 1], y_train_final, color='green', label='Data
Points')

plt.xlabel('CoapplicantIncome')
```

Name : Kabilan N
Batch : July 2023
Domain : Data Science

```
plt.ylabel('LoanAmount')

plt.title('CoapplicantIncome vs. LoanAmount')

plt.grid(True)

plt.subplot(2, 2, 3)

plt.scatter(X_train_final[:, 2], y_train_final, color='orange', label='Data
Points')

plt.xlabel('Loan_Amount_Term')

plt.ylabel('LoanAmount')

plt.title('Loan_Amount_Term vs. LoanAmount')

plt.grid(True)

plt.tight_layout()

plt.show()
```



Result and Discussion:

After implementing the Ridge regression model and evaluating its performance, the results will be discussed. This section will include insights into the model's accuracy in predicting loan amounts and any observed trends or patterns in the data.

Conclusion:

In this report, we conducted a comprehensive analysis of loan data, encompassing data preprocessing and linear regression analysis. Our linear regression model achieved an R-squared score of [insert R-squared score here], indicating [insert interpretation of R-squared score here].

However, please note that this report is based on the provided code snippets and information. Further details about the problem, dataset, and specific objectives could enhance the depth and accuracy of the analysis.

Name : Kabilan N
Batch : July 2023
Domain : Data Science

Future Work:

Given more time and resources, further improvements can be made to this analysis. Exploring different regression algorithms, optimizing hyperparameters, and incorporating additional relevant features could potentially enhance the model's predictive capabilities.

Difficulty faced:

Need more time to improve the performance of the model by reducing MSE and improve R – square value.

References :

1. [Scikit-learn Documentation](https://scikit-learn.org/stable/documentation.html)
2. Dataset source: Provide source of the loan dataset if applicable.
3. Research papers and websites consulted for data preprocessing and model implementation.