These are the libraries that have been used:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

Loading our dataset.

The APP_PATH variable contains the current directory which is appended to the file name in the below command. To avoid any complications, make sure that the python file and the csv file are in the same directory. If the case is otherwise, provide the full path of the dataset file to avoid FileNotFoundError

```
data = pd.read_excel('New_Bank_loan_data.xlsx')
```

Handle missing values by dropping rows with NaN values

```
data.dropna(inplace=True)
```

Performing EDA testing to verify the data and Exploring the dataset

```
data.head()
data.describe()
data.info()
data.isnull().values.any()
Data.size
```

Extract target variable

```
y = data['Personal Loan'].astype(str)
data.drop(columns=['Personal Loan'], inplace=True)
```

Identifying the problems

```
X = data.drop('Personal Loan', axis=1)
y = data['Personal Loan']
problematic_cols = []
for col in X.select_dtypes(include=['object']).columns:
  try:
         label_encoder = LabelEncoder()
         X[col] = label_encoder.fit_transform(X[col])
  except:
         problematic_cols.append(col)
         print(f"Column '{col}' caused an error.")

print("Problematic Columns:", problematic_cols)
```

Split the data into train and eval sets

```
X_train, X_eval, y_train, y_eval = train_test_split(data, y, test_size=0.20,
random_state=0)
```

Preprocess categorical columns (one-hot encoding)

```
categorical_columns = ['Gender', 'Home Ownership']
X_train = pd.get_dummies(X_train, columns=categorical_columns,
drop_first=True)
X_eval = pd.get_dummies(X_eval, columns=categorical_columns, drop_first=True)
```

Add columns for missing values in evaluation set

```
for col in X_train.columns:
    if col not in X_eval.columns:
        X_eval[col] = 0
```

Reorder evaluation columns to match training columns

```
X_eval = X_eval[X_train.columns]
```

Scale numerical features

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_eval_scaled = scaler.fit_transform(X_eval)
```

Train a Logistic Regression model

```
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
precdicted_classes = model.predict(X_train_scaled)
```

Make predictions on eval data

```
y_pred = model.predict(X_eval_scaled)
accuracy = accuracy_score(y_eval, y_pred)
parameters = model.coef_
```

Calculate accuracy

```
print("Accuracy:", accuracy)
print(parameters)
```