Sri Lanka Institute of Information Technology

Auto Mobile Loan Approval System

- Fundamentals of Data Mining -

[IT3051]

Group ID/Name: 4 / Mining Minds

Mini Project – Final Report

Submitted by:

Fernando W.D.A	**- IT22223708**
Gunathilaka.P.A.S.R	**- IT22136824**
A.D.Oshadhi Vibodha	**- IT22363770**
Daraniyagala G.K	**- IT22360182**
Gamage D.M.G.P.K	**- IT22188472**

Submitted to:

Dr. Prasanna S. Haddela
Name of the supervisor/lecturer in charge

3rd October 2024
Date of submission

Video link : https://shorturl.at/hk8mK
Deployed link: https://shorturl.at/9Se7x
Github link: https://shorturl.at/hDFay

Contents

Declaration

Group Number	Student Name	IT Number	Signature
Y3.S1.WD.DS.01.01	**Fernando W.D.A**	**IT22223708**	[illegible]
Y3.S1.WD.DS.01.02	**Gunathilaka.P.A.S.R**	**IT22136824**	[illegible]
Y3.S1.WD.DS.01.01	**A.D.Oshadhi Vibodha**	**IT22363770**	[illegible]
Y3.S1.WD.DS.01.01	**Daraniyagala G.K**	**IT22360182**	[illegible]
Y3.S1.WD.DS.01.02	**Gamage D.M.G.P.K**	**IT22188472**	[illegible]

Abstract

Today, the whole world is progressing very fast. The main reason for this is the new technology that is improving day by day. Nowadays, many people cannot live without the help of technology. As a result, problems that were solved using traditional methods in the past have now been solved using new technology.

Often these problems arise in connection with one's work or daily life. For that, they seek the help of new technology to find the right solutions or to find suitable solutions to reduce the existing problematic situation.

We decided to provide solutions to such a problematic situation under this project. Under our project, we have worked to first identify their problem and pre-process the relevant data and inform the user about the specific solution to their problems or the strategies that can be taken to reduce the problematic situation.
Considering all the above points, we have decided to create an automobile loan approval system.

Acknowledgement

We are deeply indebted to Mr. Prasanna Sumathipala for the guidance and support given while we finished this module properly. Also, we are very grateful to have your advice and guidance necessary to complete the project successfully.

Introduction

In the current economic climate, characterized by a financial crisis, rising inflation, and increasing interest rates, consumers are facing significant financial pressure. This has created widespread challenges within the financial system, particularly affecting Non-Banking Financial Institutions (NBFIs).

NBFIs play a critical role in the economy, providing financial services such as investment consultancy, risk pooling, savings, and market brokerage. However, unlike traditional banks, they lack the strong support of central banks and are not subject to the same stringent regulations. This makes them more vulnerable to financial shocks, especially in times of economic uncertainty.

One of the key challenges facing NBFIs today is the increasing number of loan defaults, particularly in the **automobile loan** sector. With public transport often unreliable, private vehicle ownership has become essential for many households. As a result, automobile loans represent a significant portion of overall loans. However, these loans are highly susceptible to default, as they do not typically generate financial returns for consumers. This increase in defaults has severely impacted the profitability of NBFIs, as they are the primary lenders of auto loans and often serve clients with lower credit standards.

Given this, NBFIs are now focusing on enhancing their risk assessment models to better understand the financial strength of borrowers. By analyzing historical data on loan defaults, NBFIs can assess a client's loan repayment capacity before approving new loans. This predictive model will evaluate whether a client is likely to default on their automobile loan based on key factors such as income, loan amount, and credit history. By doing so, the model aims to reduce the financial risks associated with loan defaults.

This system will help NBFIs make informed lending decisions by accurately predicting a client's ability to repay loans within the designated period. This not only mitigates the risk of defaults but also ensures that the financial health of the institution remains stable, even in times of economic volatility. By proactively identifying high-risk clients, NBFIs can adjust loan terms or deny loans when necessary, thus reducing their exposure to potential losses.

In summary, NBFIs serve as a crucial resource for clients seeking automobile loans, offering lower credit standards and shorter-term contracts. However, these same advantages also attract a higher number of applicants, increasing the likelihood of defaults. By implementing a robust loan default prediction system, NBFIs can better manage their risk, protect their profit margins, and continue to serve clients effectively in a challenging financial environment.

Dataset Description

A Non-Banking Financial Institution (NBFI), also known as a Non-Bank Financial Company (NBFC), provides financial services similar to banks but operates without a full banking license and is not regulated by a national or international banking authority. These institutions offer services such as investments, risk management, contractual savings, and market brokering.
One such NBFI is facing profitability challenges due to an increase in defaults on vehicle loans. To improve their performance, the company aims to evaluate clients' ability to repay loans and identify the key factors that influence repayment behavior.
The objective is to predict whether a client will default on a vehicle loan. Using a given Test Dataset, the task is to predict the "Default" status for each client ID.Two datasets are provided the Train Dataset for model development and the Test Dataset for model evaluation.
The dataset consists of 121,856 records of past consumer data, detailing individuals who applied for and were approved for car loans, with information spread across 40 different features.

Dataset: https://www.kaggle.com/datasets/saurabhbagchi/dish-network-hackathon

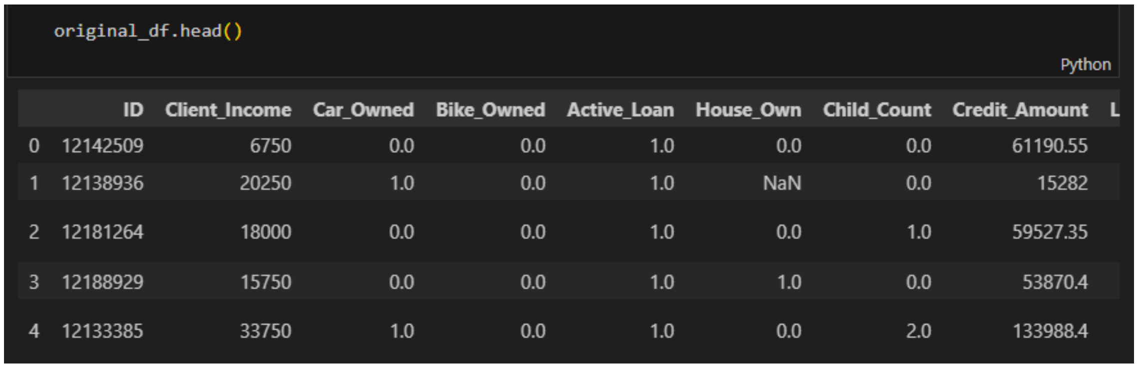
Data Preparation and Preprocessing

The raw data set must go through numerous phases such as:
1. Data cleaning
2. Data integration
3. Data reduction
4. Data transformation

to turn into a more intelligible format before feeding into the machine learning model to train the data. The procedures and methods used to preprocess data are described below.

Reading the original dataset

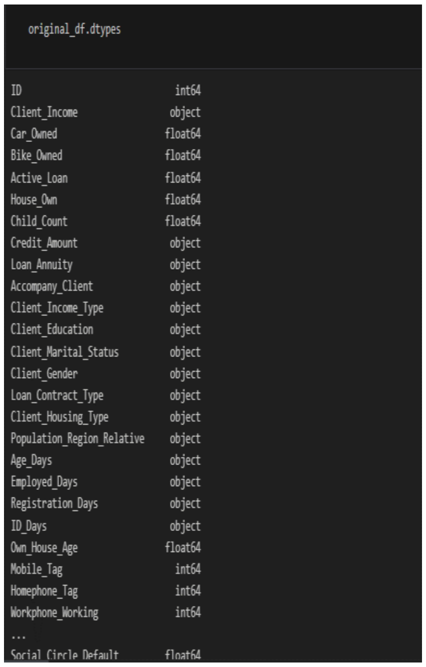
First, the original data are read to get an idea about the data and the attributes we are going to handle and thereby understand the dataset for model building. This will also help in understanding the types of data we are dealing with in the original dataset.



```python
original_df.head()
```

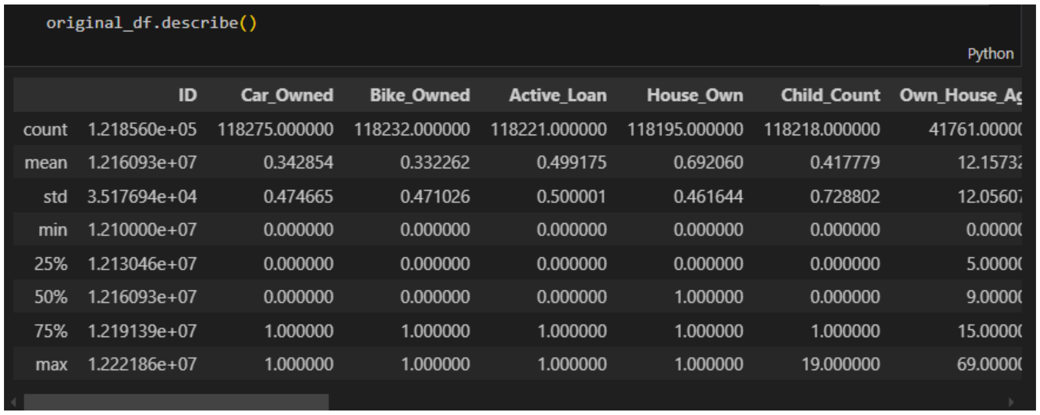
Python

```
         ID  Client_Income  Car_Owned  Bike_Owned  Active_Loan  House_Own  Child_Count  Credit_Amount  L
0  12142509           6750        0.0         0.0          1.0        0.0          0.0       61190.55
1  12138936          20250        1.0         0.0          1.0        NaN          0.0          15282
2  12181264          18000        0.0         0.0          1.0        0.0          1.0       59527.35
3  12188929          15750        0.0         0.0          1.0        1.0          0.0        53870.4
4  12133385          33750        1.0         0.0          1.0        0.0          2.0       133988.4
```



```python
original_df.dtypes
```

```
ID                            int64
Client_Income                object
Car_Owned                   float64
Bike_Owned                  float64
Active_Loan                 float64
House_Own                   float64
Child_Count                 float64
Credit_Amount                object
Loan_Annuity                 object
Accompany_Client             object
Client_Income_Type           object
Client_Education             object
Client_Marital_Status        object
Client_Gender                object
Loan_Contract_Type           object
Client_Housing_Type          object
Population_Region_Relative   object
Age_Days                     object
Employed_Days                object
Registration_Days            object
ID_Days                      object
Own_House_Age               float64
Mobile_Tag                    int64
Homephone_Tag                 int64
Workphone_Working             int64
...
Social_Circle_Default       float64
```

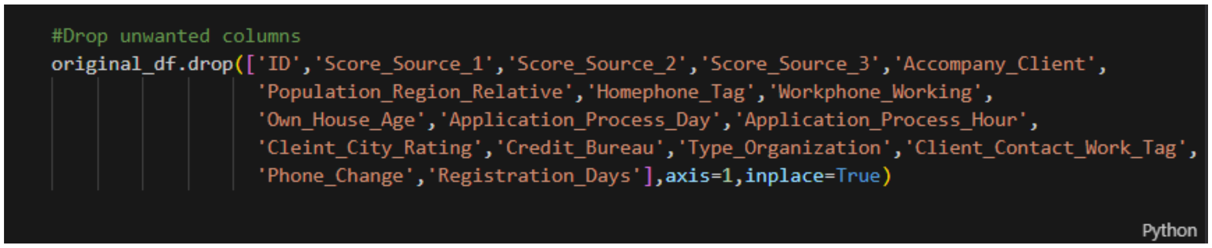


```
original_df.describe()
```

	ID	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Own_House_Ag
count	1.218560e+05	118275.000000	118232.000000	118221.000000	118195.000000	118218.000000	41761.0000
mean	1.216093e+07	0.342854	0.332262	0.499175	0.692060	0.417779	12.1573
std	3.517694e+04	0.474665	0.471026	0.500001	0.461644	0.728802	12.0560
min	1.210000e+07	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	1.213046e+07	0.000000	0.000000	0.000000	0.000000	0.000000	5.0000
50%	1.216093e+07	0.000000	0.000000	0.000000	1.000000	0.000000	9.0000
75%	1.219139e+07	1.000000	1.000000	1.000000	1.000000	1.000000	15.0000
max	1.222186e+07	1.000000	1.000000	1.000000	1.000000	19.000000	69.0000

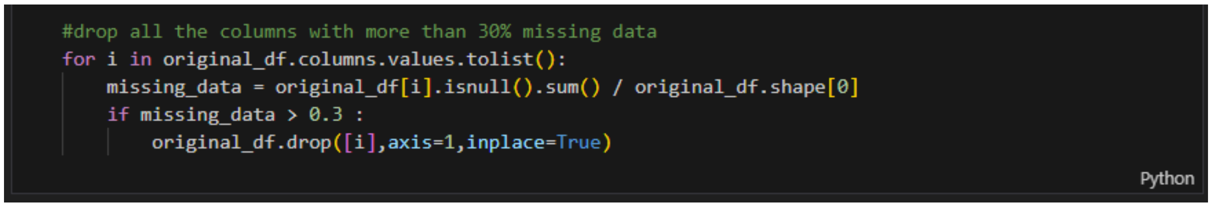
Dimensionality Reduction

Normally, it is done in two directions: row-wise for data sample reduction and column-wise for data variable reduction.
Here, column-wise reduction of variables of data is done. In this, model-irrelevant columns are dropped and those columns that contain more than 30% of rows as NULL are dropped because it may affect the accuracy of the model.
This is followed by removing columns like the client's id-a unique attribute that does not provide any predictive power, unambiguous columns such as 'scores1' and by conducting background research on the dataset. Furthermore, we check if a particular row contains more than 50% of null values. If so, we considered dropping that.



```
#Drop unwanted columns
original_df.drop(['ID','Score_Source_1','Score_Source_2','Score_Source_3','Accompany_Client',
                  'Population_Region_Relative','Homephone_Tag','Workphone_Working',
                  'Own_House_Age','Application_Process_Day','Application_Process_Hour',
                  'Cleint_City_Rating','Credit_Bureau','Type_Organization','Client_Contact_Work_Tag',
                  'Phone_Change','Registration_Days'],axis=1,inplace=True)
```

```
Client_Income                 0.029601
Car_Owned                     0.029387
Bike_Owned                    0.029740
Active_Loan                   0.029830
House_Own                     0.030044
Child_Count                   0.029855
Credit_Amount                 0.029806
Loan_Annuity                  0.039489
Client_Income_Type            0.030372
Client_Education              0.029912
Client_Marital_Status         0.028501
Client_Gender                 0.019802
Loan_Contract_Type            0.029962
Client_Housing_Type           0.030257
Age_Days                      0.029543
Employed_Days                 0.029945
ID_Days                       0.048976
Mobile_Tag                    0.000000
Client_Occupation             0.340032
Client_Family_Members         0.019777
Client_Permanent_Match_Tag    0.000000
Social_Circle_Default         0.508206
Default                       0.000000
dtype: float64
```



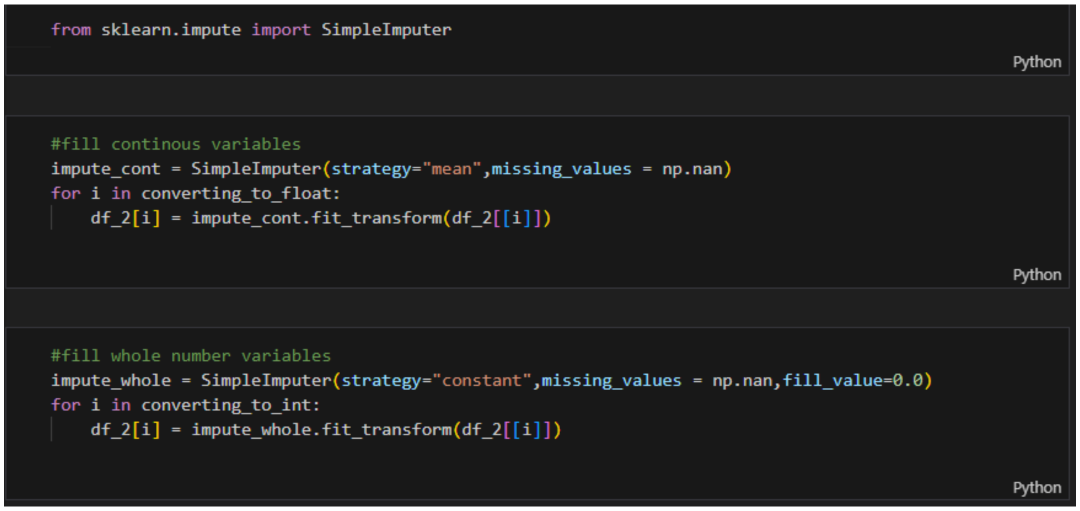
```python
#drop all the columns with more than 30% missing data
for i in original_df.columns.values.tolist():
    missing_data = original_df[i].isnull().sum() / original_df.shape[0]
    if missing_data > 0.3 :
        original_df.drop([i],axis=1,inplace=True)
```

```python
#check how may missing values are in each row and if it is greater than 11, drop all those rows
missing_values_per_row = original_df.isnull().sum(axis=1)
sorted_missing = missing_values_per_row.sort_values(ascending=False)
print(sorted_missing)
```

```
6468      7
56024     6
46452     5
34523     5
65991     5
         ..
50380     0
50379     0
50377     0
50376     0
121855    0
Length: 117094, dtype: int64
```

Null Value Handling and removal of duplicates

In general, there are two approaches to handling missing values when building operational data: the first one is to simply discard data samples containing missing values, because most data mining algorithms cannot handle data with missing values. Such a method is possible only when the proportion of the missing values is insignificant. The second is by applying the missing value imputation techniques to replace missing data using inference values. For imputation of missing continuous numerical variables, we use the **mean** strategy. Moreover, to impute whole number variables, we used **constant** strategy where a constant value of 0 is filled in every null value place. This might help our model from stop being biased to a certain value in each whole number variable columns.

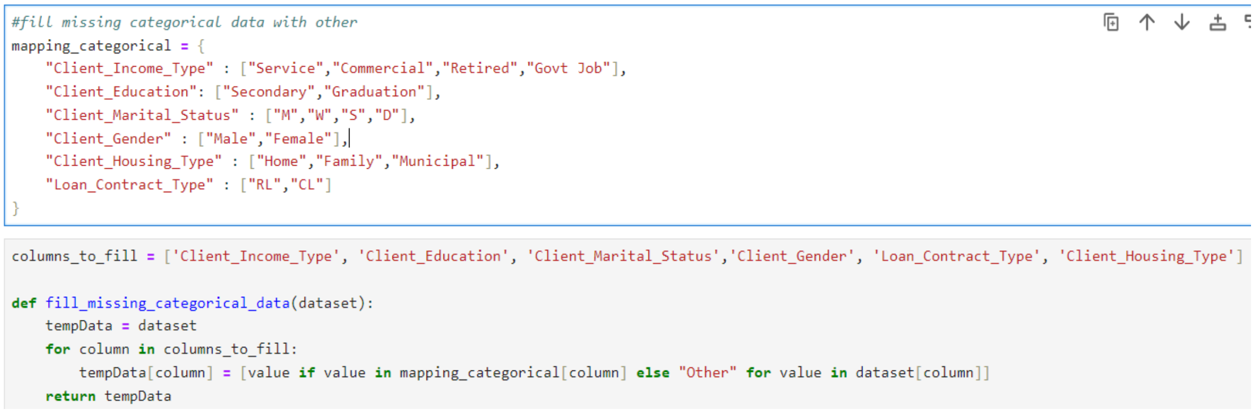


```python
from sklearn.impute import SimpleImputer
```

```python
#fill continous variables
impute_cont = SimpleImputer(strategy="mean",missing_values = np.nan)
for i in converting_to_float:
    df_2[i] = impute_cont.fit_transform(df_2[[i]])
```

```python
#fill whole number variables
impute_whole = SimpleImputer(strategy="constant",missing_values = np.nan,fill_value=0.0)
for i in converting_to_int:
    df_2[i] = impute_whole.fit_transform(df_2[[i]])
```

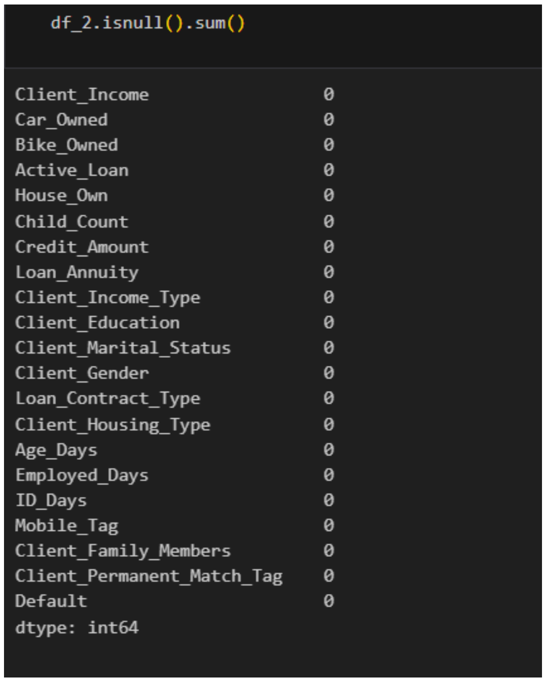
To fill in the null values in the categorical data, we first created a map by examining the groupings that appeared most frequently in each column. Next, we entered the value **Other** for every categorical column whose value was missing from the map.



```python
#fill missing categorical data with other
mapping_categorical = {
    "Client_Income_Type" : ["Service","Commercial","Retired","Govt Job"],
    "Client_Education": ["Secondary","Graduation"],
    "Client_Marital_Status" : ["M","W","S","D"],
    "Client_Gender" : ["Male","Female"],
    "Client_Housing_Type" : ["Home","Family","Municipal"],
    "Loan_Contract_Type" : ["RL","CL"]
}
```

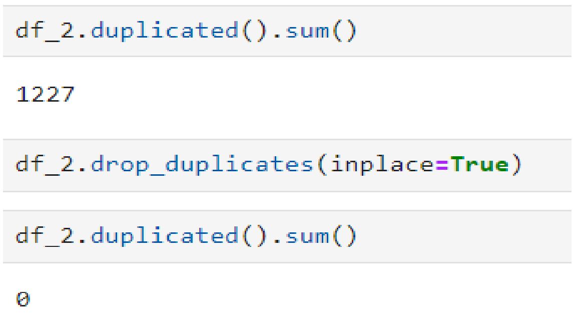
```python
columns_to_fill = ['Client_Income_Type', 'Client_Education', 'Client_Marital_Status','Client_Gender', 'Loan_Contract_Type', 'Client_Housing_Type']

def fill_missing_categorical_data(dataset):
    tempData = dataset
    for column in columns_to_fill:
        tempData[column] = [value if value in mapping_categorical[column] else "Other" for value in dataset[column]]
    return tempData
```

This is the appearance of our dataset after processing null values.



```
df_2.isnull().sum()

Client_Income                 0
Car_Owned                     0
Bike_Owned                    0
Active_Loan                   0
House_Own                     0
Child_Count                   0
Credit_Amount                 0
Loan_Annuity                  0
Client_Income_Type            0
Client_Education              0
Client_Marital_Status         0
Client_Gender                 0
Loan_Contract_Type            0
Client_Housing_Type           0
Age_Days                      0
Employed_Days                 0
ID_Days                       0
Mobile_Tag                    0
Client_Family_Members         0
Client_Permanent_Match_Tag    0
Default                       0
dtype: int64
```

We then looked to see if any values were duplicates. If so, get rid of them as they could make the model perform worse.



```
df_2.duplicated().sum()

1227

df_2.drop_duplicates(inplace=True)

df_2.duplicated().sum()

0
```

Finding outliers of numerical data and treat them

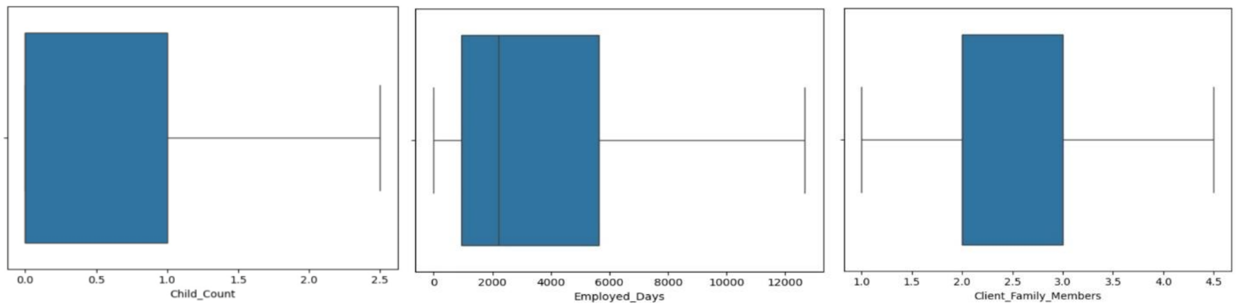
```python
def find_outlier(column):
    q1 = df_2[column].quantile(0.25)
    q3 = df_2[column].quantile(0.75)
    iqr = q3 - q1
    lb = q1 - 1.5 * iqr
    ub = q3 + 1.5 * iqr
    return lb,ub
```

```python
def treat_outliers(column):
    lb,ub = find_outlier(column)
    df_2[column] = np.where(df_2[i]<lb,lb,df_2[i])
    df_2[column] = np.where(df_2[i]>ub,ub,df_2[i])
```

```python
for i in ['Child_Count','Employed_Days','Client_Family_Members']:
    treat_outliers(i)
```

```python
for i in ['Child_Count','Employed_Days','Client_Family_Members']:
    sns.boxplot(data=df_2,x=i)
    plt.show()
```

After outlier treatments,

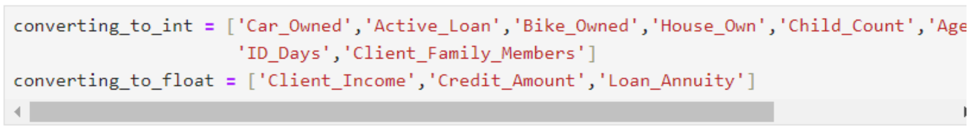


When reviewing all the box plot visualizations,

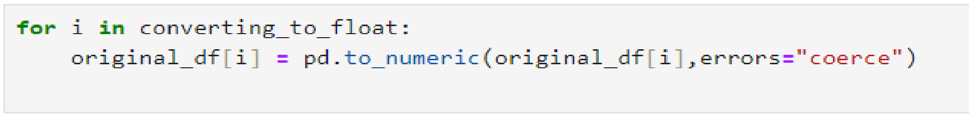
- Most of the features have a significant number of outliers.
- The boxplots for the 'Client Income', 'Credit Amount', 'Loan Annuity' features, after converting it to float, reveals the presence of several outliers, indicating extreme values in the data. We did not treat those outliers since they might affect the dataset as well as the model's performance for this problem.
- In the 'Bike Owned', 'Active Loan', and 'House Own', 'Car Owned' plots, the lower quartile is equal to the minimum and the upper quartile is equal to the maximum.
- The boxplots for 'Age Days' and 'Id Days' do not show outliers and features like 'child count', 'Employed Days', 'Client Family Members' show many outliers.
- After identifying the outliers, 'Child Count', 'Employed Days' and 'Client Family Members' were remedied and displayed in the respective ranges as mentioned above.

Data transformation and Normalization

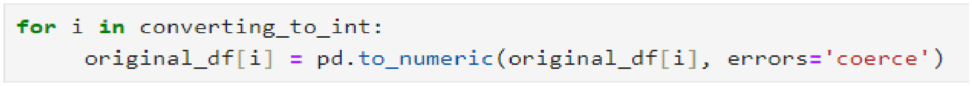
Some columns such as client income, credit amount, car owned etc. should be in numerical data type in nature but columns like those have the data type object in the dataset. Therefore, data type should be change in those columns.



```
converting_to_int = ['Car_Owned','Active_Loan','Bike_Owned','House_Own','Child_Count','Age
                     'ID_Days','Client_Family_Members']
converting_to_float = ['Client_Income','Credit_Amount','Loan_Annuity']
```

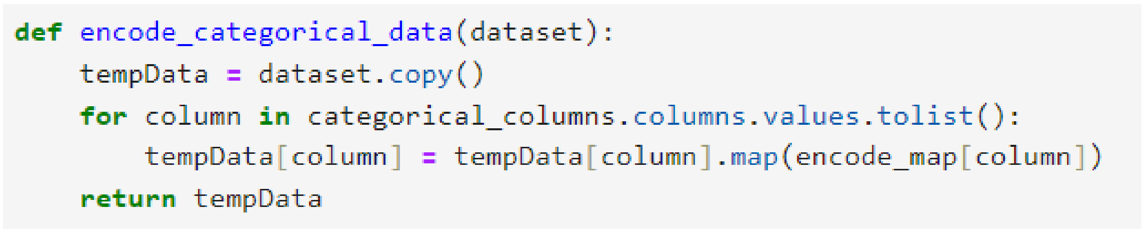


```
for i in converting_to_float:
    original_df[i] = pd.to_numeric(original_df[i],errors="coerce")
```

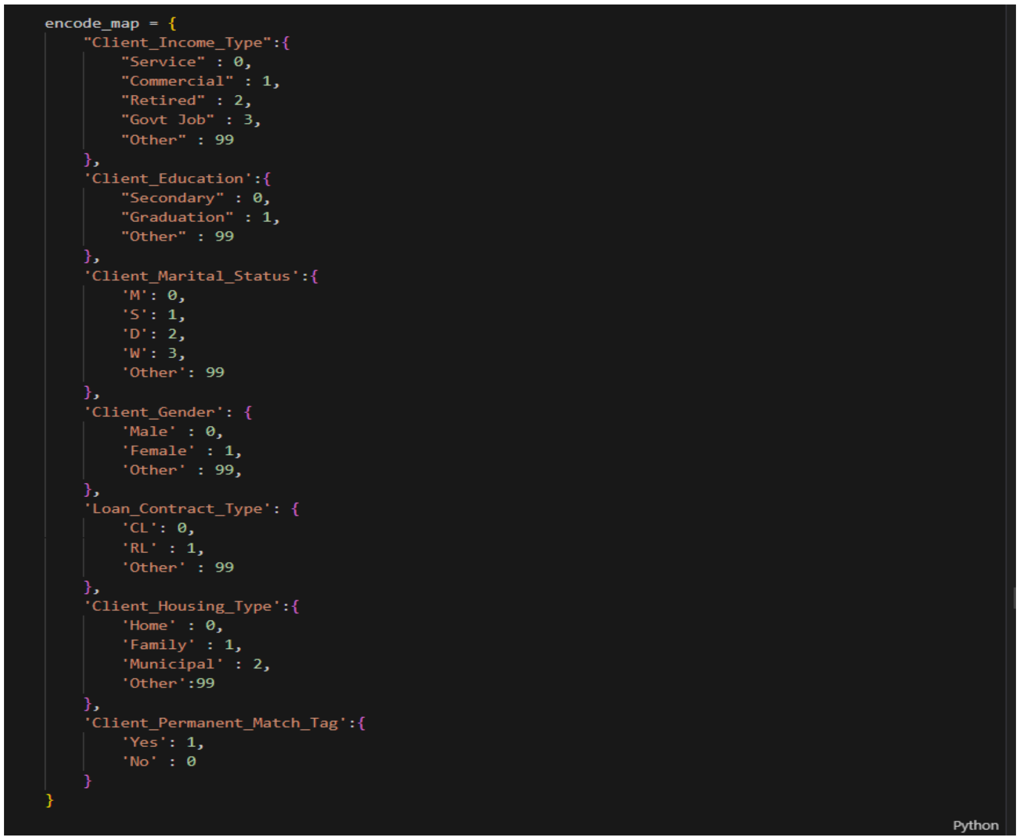


```
for i in converting_to_int:
     original_df[i] = pd.to_numeric(original_df[i], errors='coerce')
```

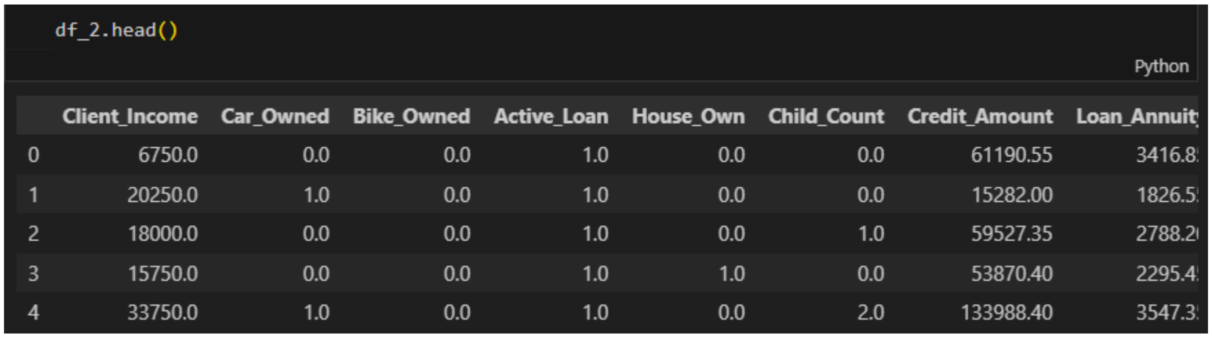
Categorical variables need to be encoded for machine learning models to learn. Label encoding and one hot encoding (Dummy encoding), are well-known techniques for converting to a numerical type. Both dummy coding and one hot encoding change one variable into many variables, whereas in label encoding, each unique value of the categorical column gets changed to a number.
But for our project, we applied **custom encoding** instead of one hot or label encoding. The goal of this manual encoding was to simplify the representation of the categories in a way that suits the context of data. The category Other was assigned with a value of 99 to clearly differentiate between common categories.



```
def encode_categorical_data(dataset):
    tempData = dataset.copy()
    for column in categorical_columns.columns.values.tolist():
        tempData[column] = tempData[column].map(encode_map[column])
    return tempData
```



```
encode_map = {
    "Client_Income_Type":{
        "Service" : 0,
        "Commercial" : 1,
        "Retired" : 2,
        "Govt Job" : 3,
        "Other" : 99
    },
    'Client_Education':{
        "Secondary" : 0,
        "Graduation" : 1,
        "Other" : 99
    },
    'Client_Marital_Status':{
        'M': 0,
        'S': 1,
        'D': 2,
        'W': 3,
        'Other': 99
    },
    'Client_Gender': {
        'Male' : 0,
        'Female' : 1,
        'Other' : 99,
    },
    'Loan_Contract_Type': {
        'CL': 0,
        'RL' : 1,
        'Other' : 99
    },
    'Client_Housing_Type':{
        'Home' : 0,
        'Family' : 1,
        'Municipal' : 2,
        'Other':99
    },
    'Client_Permanent_Match_Tag':{
        'Yes': 1,
        'No' : 0
    }
}
```



```
df_2.head()
```

	Client_Income	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Credit_Amount	Loan_Annuit
0	6750.0	0.0	0.0	1.0	0.0	0.0	61190.55	3416.8
1	20250.0	1.0	0.0	1.0	0.0	0.0	15282.00	1826.5
2	18000.0	0.0	0.0	1.0	0.0	1.0	59527.35	2788.2
3	15750.0	0.0	0.0	1.0	1.0	0.0	53870.40	2295.4
4	33750.0	1.0	0.0	1.0	0.0	2.0	133988.40	3547.3

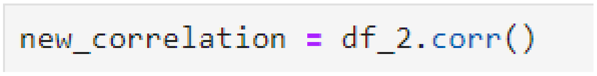
In order to improve the performance of the model, we used StandardScaler from the scikit-learn module to standardize numerical data. The mean and standard deviation are determined for every column by the standard scaler, which then normalizes the results to mean 0 and standard deviation 1.

```
normalizing_columns = ['Client_Income','Credit_Amount','Age_Days','ID_Days']

for i in normalizing_columns:
    scaler = StandardScaler()
    df_2[i] = scaler.fit_transform(df_2[[i]])
```

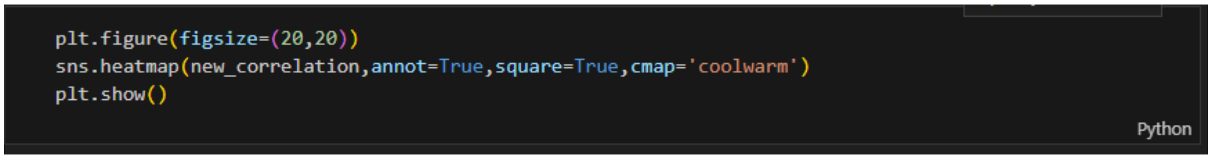
Feature Selection

There are still a lot of columns present in the dataset after performing the above steps and preparing them. The most recognizable step is the one for feature selection. Another synonym used is variable or attribute selection. Feature selection is the process through which the fewest possible useful attribute sets are selected. To choose the features, we used Correlation Coefficient score. The correlation coefficient implies a measure that gives the degree of a linear relationship between two variables. There should high correlation with the target variable, and none of the variables should be correlated among themselves.

When two variables are highly correlated it should use one for the prediction as the other doesn't give the model any further weight.



```
new_correlation = df_2.corr()
```

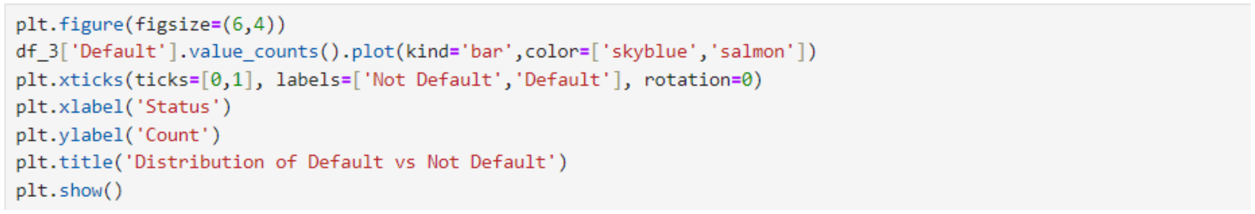


```
plt.figure(figsize=(20,20))
sns.heatmap(new_correlation,annot=True,square=True,cmap='coolwarm')
plt.show()
```

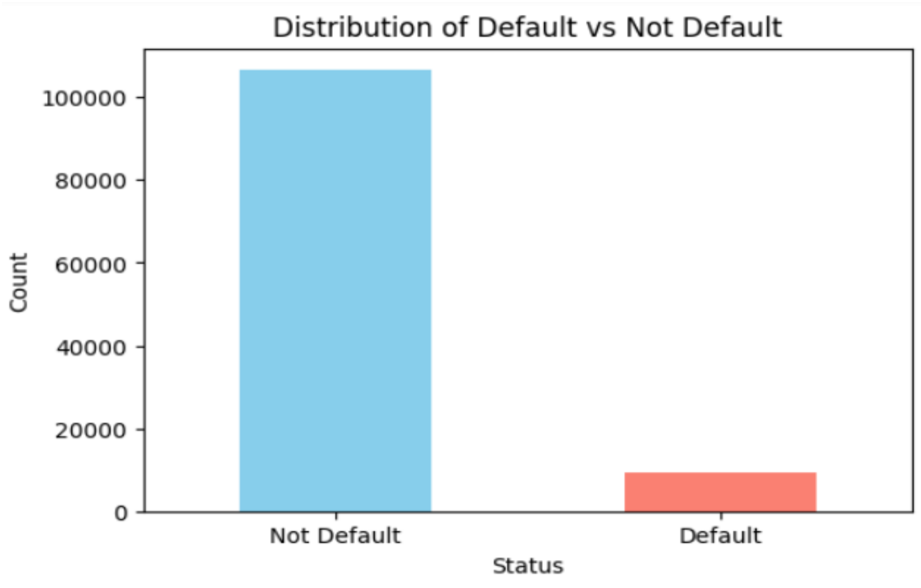
Python

Issue of an Imbalanced Data Set

The imbalance of the data set can be handled using various resampling techniques like under sampling, oversampling, SMOTE technique, etc.

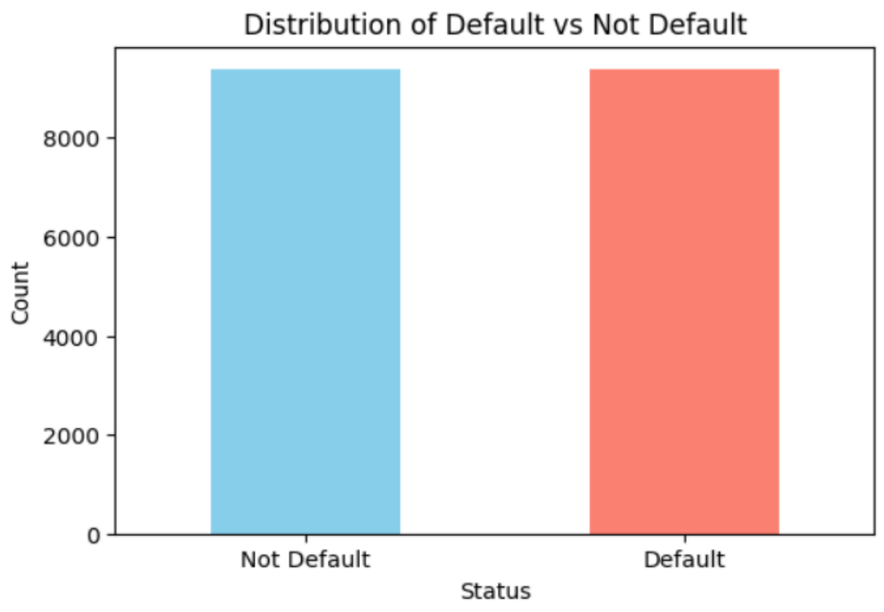


```
plt.figure(figsize=(6,4))
df_3['Default'].value_counts().plot(kind='bar',color=['skyblue','salmon'])
plt.xticks(ticks=[0,1], labels=['Not Default','Default'], rotation=0)
plt.xlabel('Status')
plt.ylabel('Count')
plt.title('Distribution of Default vs Not Default')
plt.show()
```



Solving the Issue of an Imbalanced Data Set

As already said, while picturing the step of data visualization, this data set is highly imbalanced. It had an over-skew of about 90% towards the Not Default type; this was a big problem since it affects the overall accuracy of the model and would be biased for class 0. Therefore, the under-sampling technique has been used here to balance the dataset.

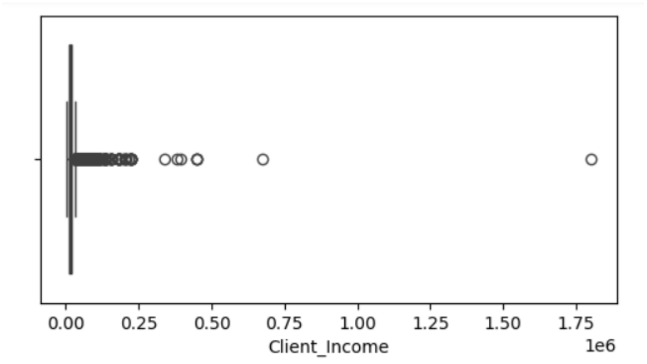


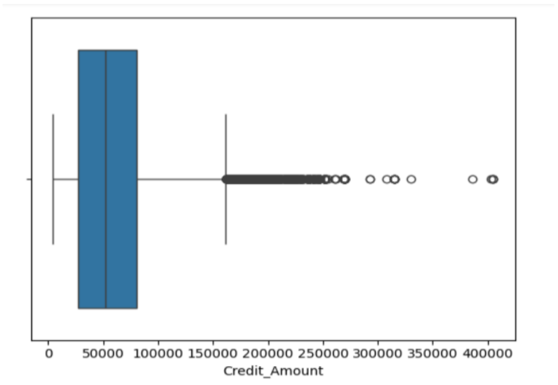
Data Visualization (Exploratory Data Analysis)

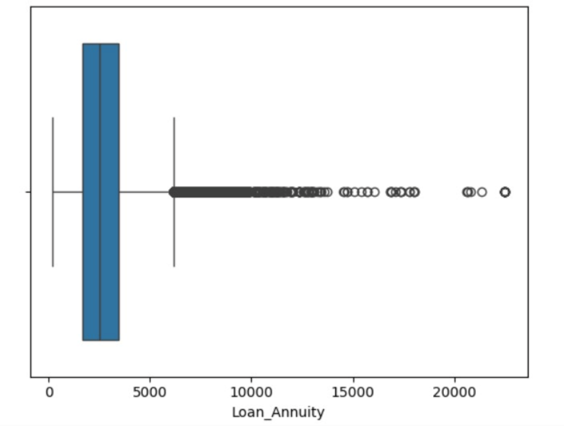
After completing basic data preprocessing, we move to the data visualization phase to get a rough idea about the dataset. The dataset contains both numerical and categorical variables. Categorical data is visualized using bar charts, while numerical data is represented through box plots, all created with the Seaborn library in Python.

Numerical Data Visualization

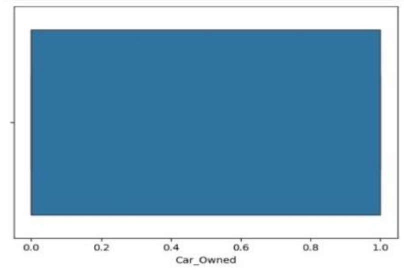
```
plt.figure(figsize=(6,3))
for i in converting_to_float:
    sns.boxplot(data=df_1,x=i)
    plt.show()
```

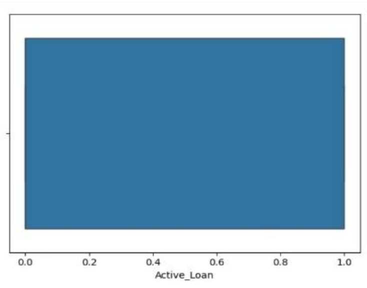


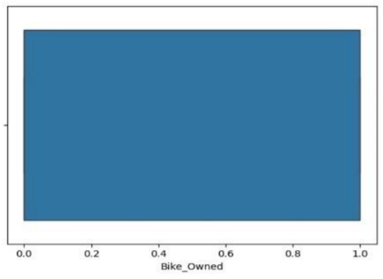


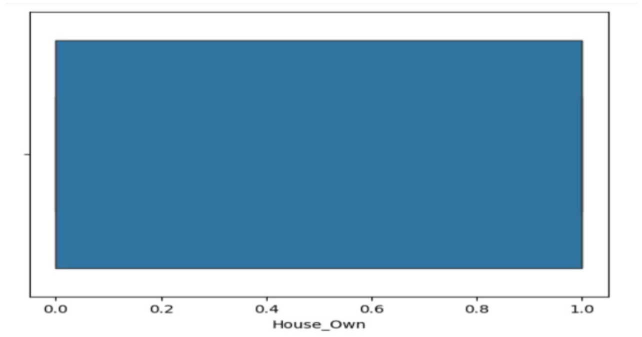


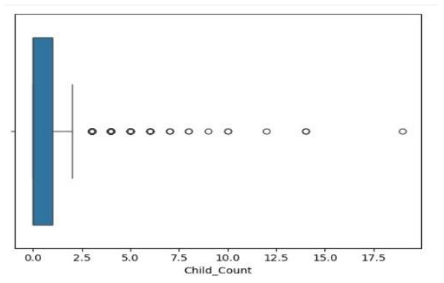
```python
for i in converting_to_int:
    sns.boxplot(data=df_1,x=i)
    plt.show()
```

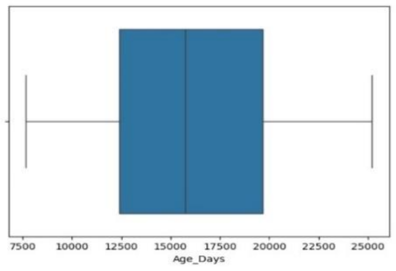


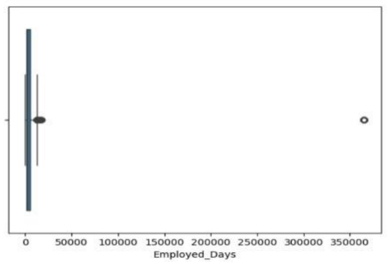


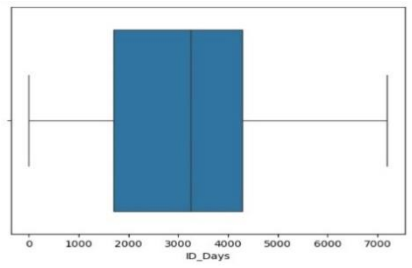


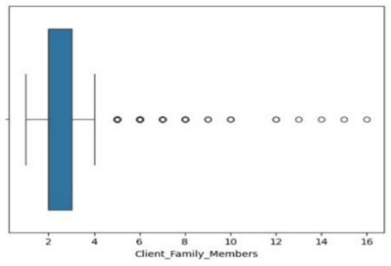




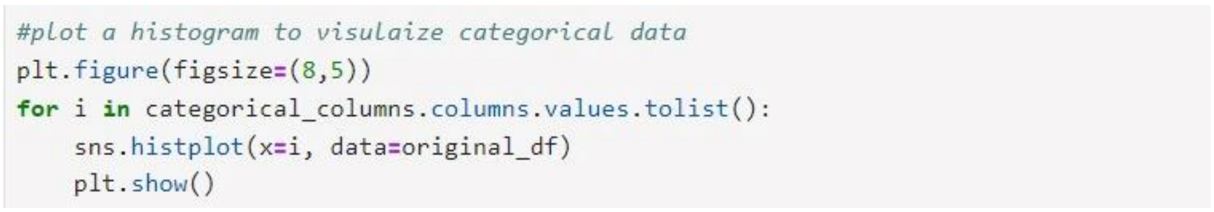




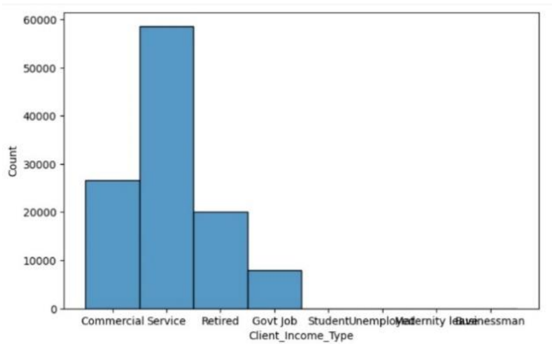


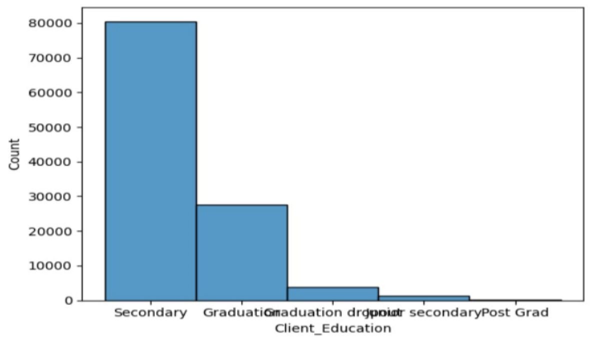


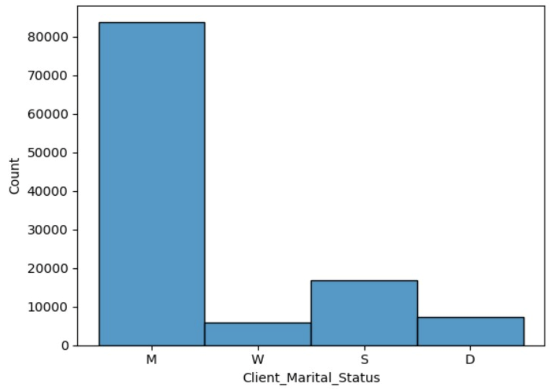
Categorical Data Visualization

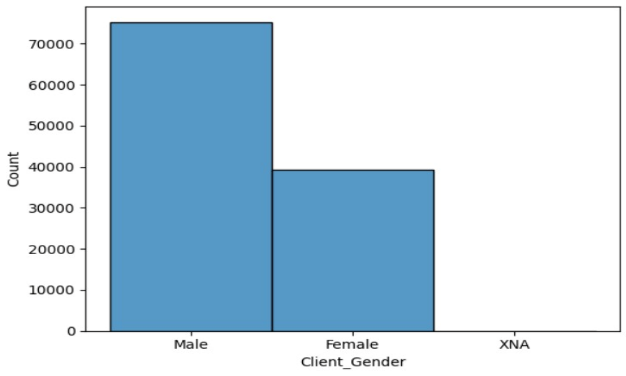


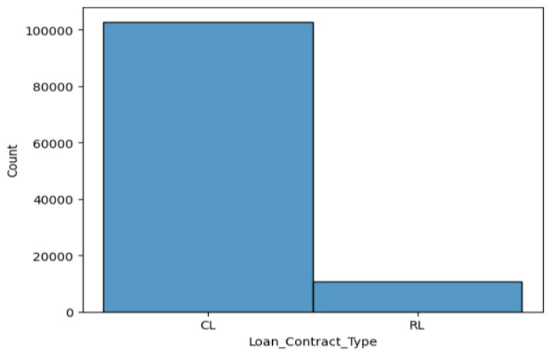
```
#plot a histogram to visulaize categorical data
plt.figure(figsize=(8,5))
for i in categorical_columns.columns.values.tolist():
    sns.histplot(x=i, data=original_df)
    plt.show()
```

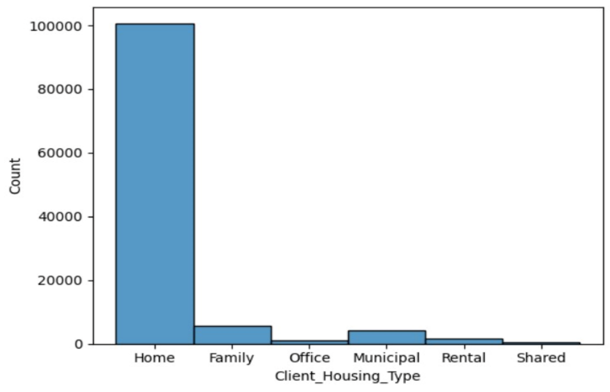


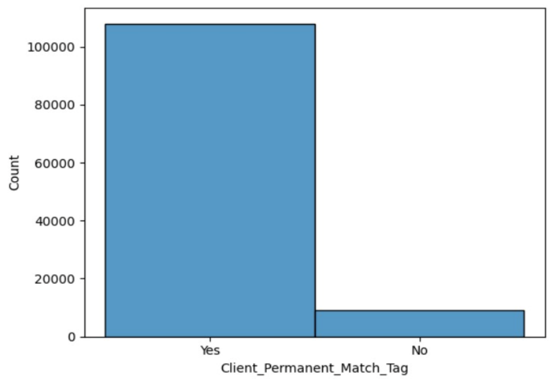








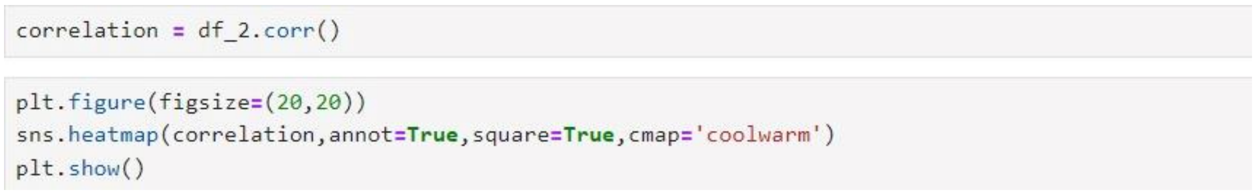




When reviewing all the graph visualizations,

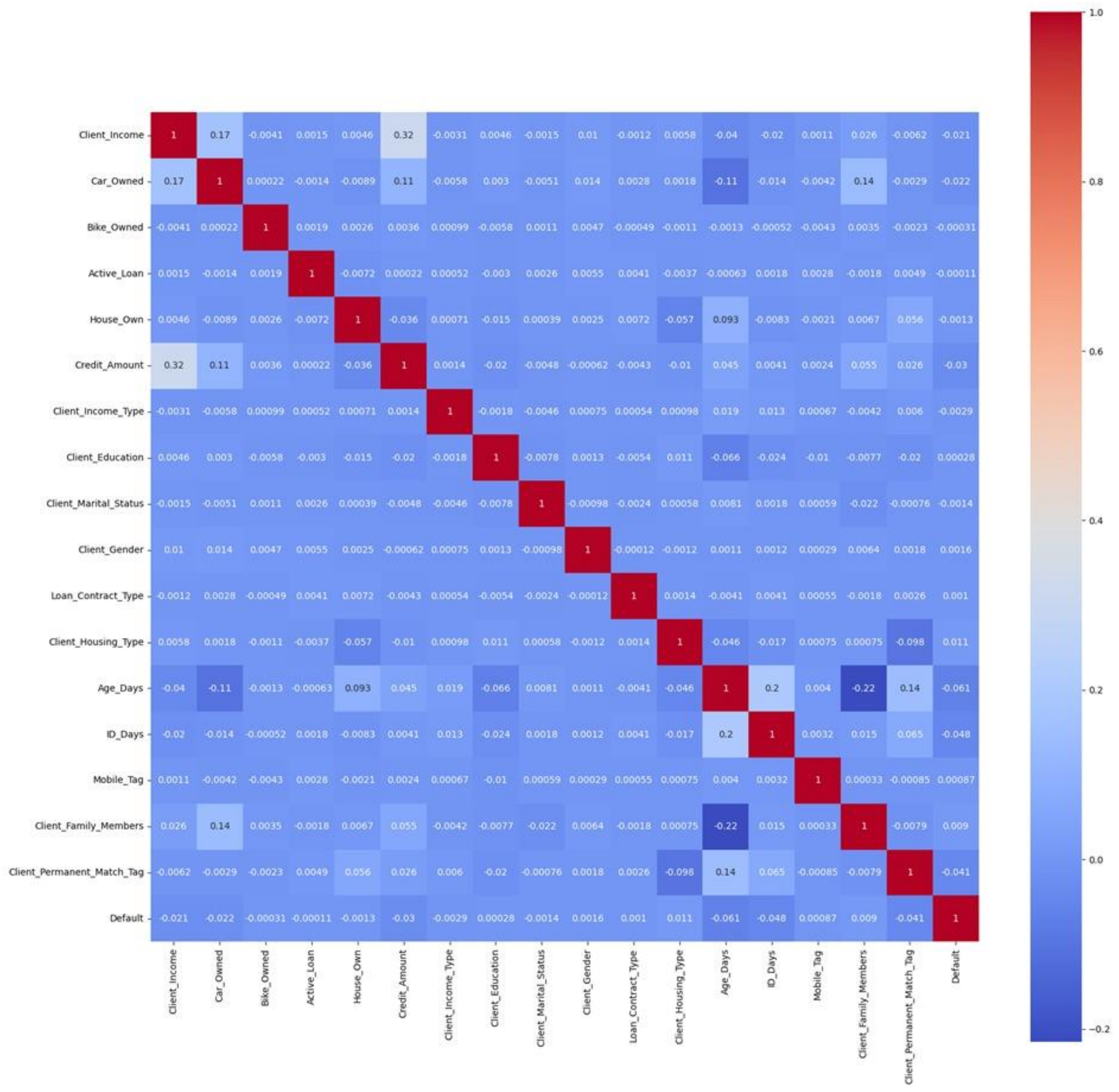
- Among the loan applicants, most of the clients have earned their income from service, with clients coming from commercial, retired and government employment backgrounds representing the majority. Meanwhile, the graph shows that students, businessman, the unemployed and those on maternity leave are underrepresented.
- The graph shows that most loan applicants have a secondary education, while fewer have graduate, postgraduate, or dropout degrees.
- The largest proportion of the population of loan obtainers was married. Compared to that, all other categories such as single, divorce and widowed receive less attention and widowed shows the least represented among applicants.
- There were more men than women who applied for the loan. A significant number of debts from both sexes were no longer in default.
- The highest amount of requests is for cash loans and out of them also the majority of the loans have not defaulted.
- The graph shows that most loan applicants live in their own homes (Home), followed by family homes (Family). Other housing types, such as offices, municipal housing, rentals, and shared housing, are less common among applicants.

Correlation heatmaps visualize the intensity of relationships between numerical variables. These plots help in identifying which variables are connected and the strength of their associations.



```
correlation = df_2.corr()
```

```
plt.figure(figsize=(20,20))
sns.heatmap(correlation,annot=True,square=True,cmap='coolwarm')
plt.show()
```



Client_Income
Car_Owned
Bike_Owned
Active_Loan
House_Own
Credit_Amount
Client_Income_Type
Client_Education
Client_Marital_Status
Client_Gender
Loan_Contract_Type
Client_Housing_Type
Age_Days
ID_Days
Mobile_Tag
Client_Family_Members
Client_Permanent_Match_Tag
Default
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

Data Mining Solutions

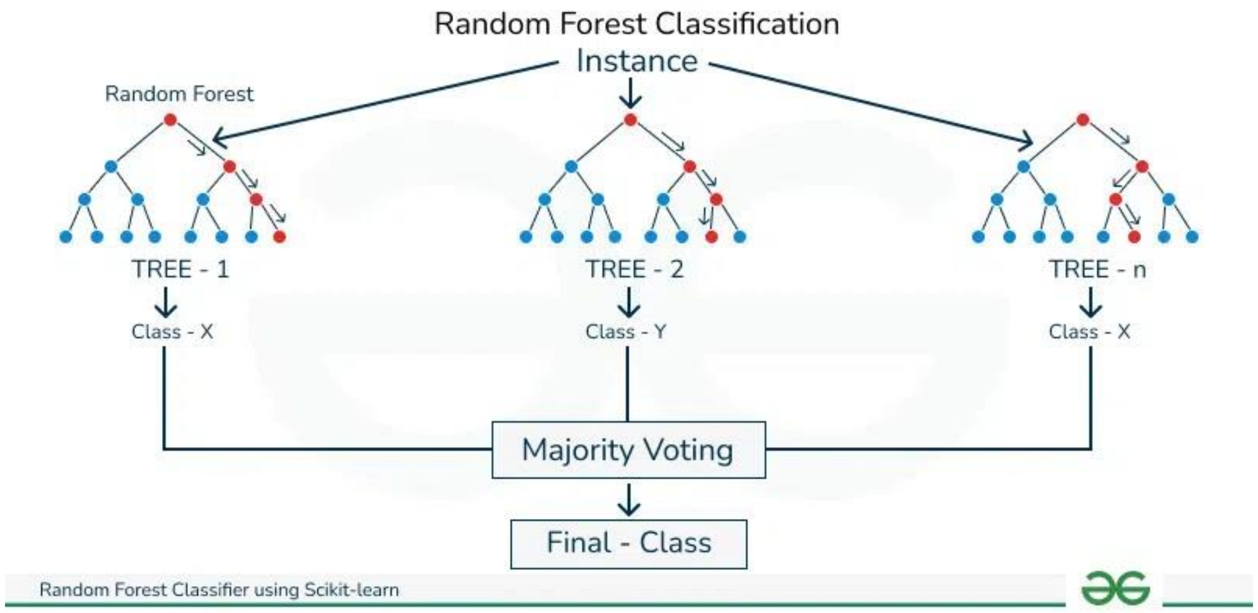
Machine Learning models may be described as a program which has been trained to detect relationships from new data and make a prediction. These models can be expressed in a mathematical equation which takes the request in some input data, makes some prediction on the input data and a put out an output. In the first place, these models learn over a set of data and subsequently are given an algorithm to reason over data, to extract pattern from feed data and learn from those data. Once these models get built, then it's possible to use them to forecast the new untrained data set.

In our case study for data mining, we focus on a Binary class classification task on credit risk prediction on whether a client will be able to pay his or her loans. The target variable represents two possible outcomes and '1' represents defaulted group and '0' represents non-defaulted group. To solve this problem, we used several machine learning algorithms on the dataset preprocessed by us.

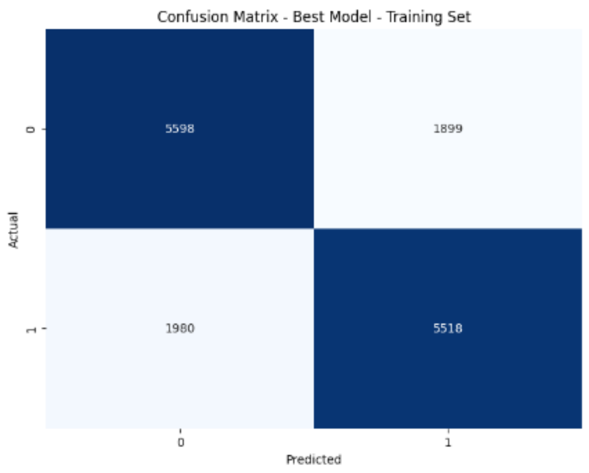
1. Random Forest
2. XGBoost (Extreme Gradient Boosting)
3. LightGBM (Light Gradient Boosting Machine)
4. Logistic Regression
5. CatBoost
6. Gaussian Naive Bayes (Gaussian NB)

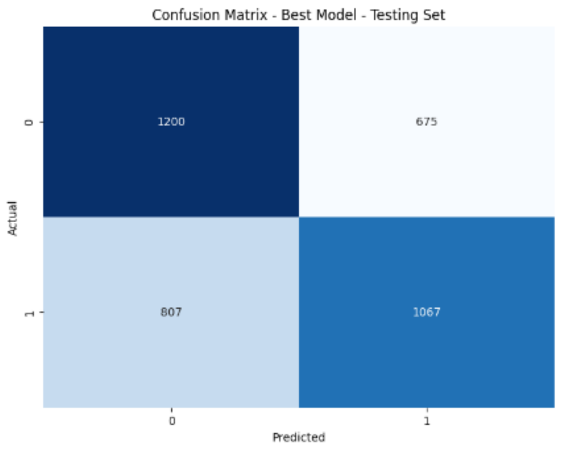
Random Forest

Random forests are an ensemble learning technique that builds a large number of decision trees during the training phase for tasks like regression, classification, and other applications. For classification output of the random forest can be the class chosen by majority vote. For a regression task, it would be the mean prediction. In regression tasks output is the mean or average prediction of individual trees. Random decision forests adjust for the tendency of decision trees to overfit their training set.It is a group of decision trees made out of random selection of the training set and in turn The final decision is made by collect the response of the different decision trees.

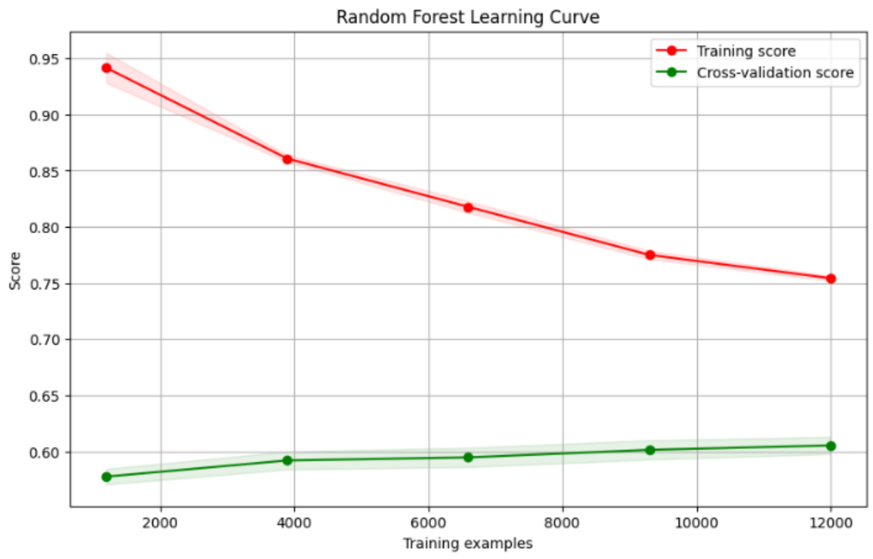


Random forest mitigates the problem of overfitting that is often associated with individual decision tree by producing the final predict on average of multiple decision trees. And also offers an option for the relative importance of each input feature or an importance score through which we can explore the importance of each input feature in prediction
Generally, Random Forest models have several hyperparameters the perform a significant role in how the model will perform including n_estimators (number of trees), max_depth (depth of trees) and max_features (number of features considered during splitting).
The below screenshots provide an instance where we used the Random Forest classifier and also include the training and test accuracy according to our preprocessed dataset.



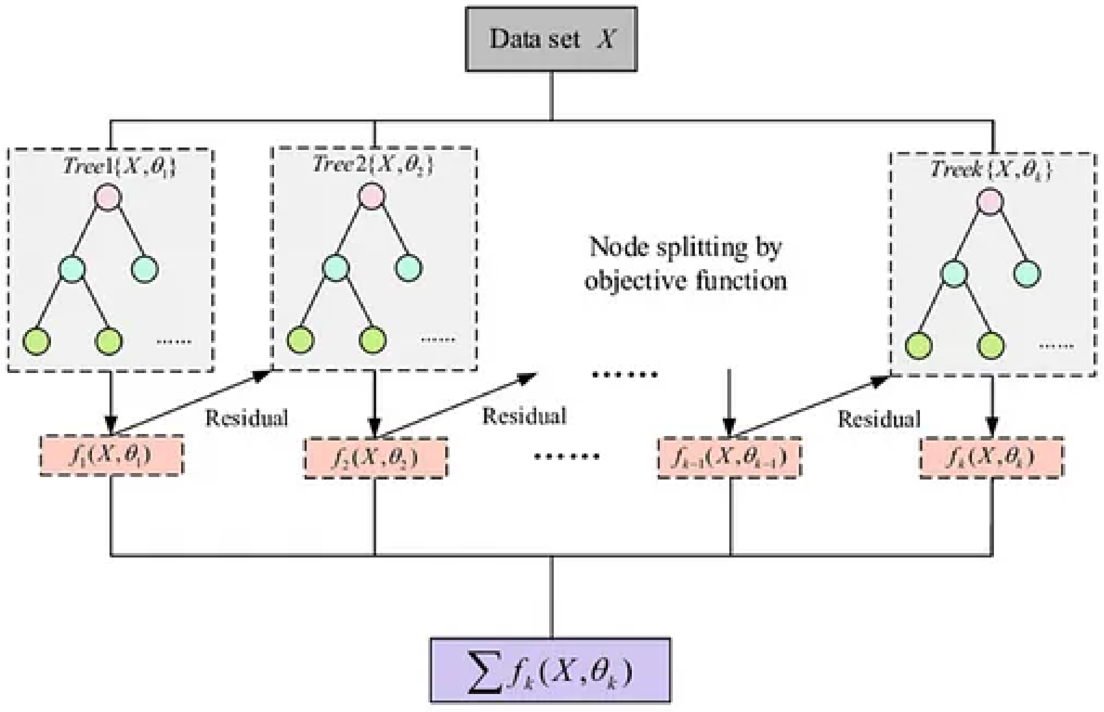


```
Best Model Train Accuracy: 0.7413137712570856
Best Model Test Accuracy: 0.604694585222726

Best Model Train Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.75      0.74      7497
           1       0.74      0.74      0.74      7498

    accuracy                           0.74     14995
   macro avg       0.74      0.74      0.74     14995
weighted avg       0.74      0.74      0.74     14995


Best Model Test Classification Report:
              precision    recall  f1-score   support

           0       0.60      0.64      0.62      1875
           1       0.61      0.57      0.59      1874

    accuracy                           0.60      3749
   macro avg       0.61      0.60      0.60      3749
weighted avg       0.61      0.60      0.60      3749
```



XGBoost (Extreme Gradient Boosting)

XGBoost is a highly optimized distributed gradients boosting libraries intended for scalable first-class machine learning training. It is that technique of machine learning where many weak classifiers are trained to make weak predictions and these are combined to make a strong prediction. XGBoost is an acronym of “Extreme Gradient Boosting” and has turned into one of the most demanding and accepted machine learning algorithms for two reasons, The XGBoost has immense capability to handle the huge data and it wins the high rank in the machine learning classification and regression tasks.

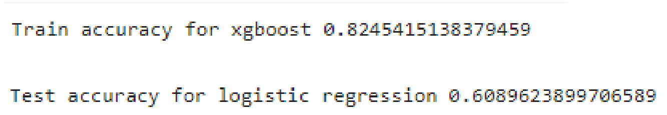


Flow chart of XGBoost

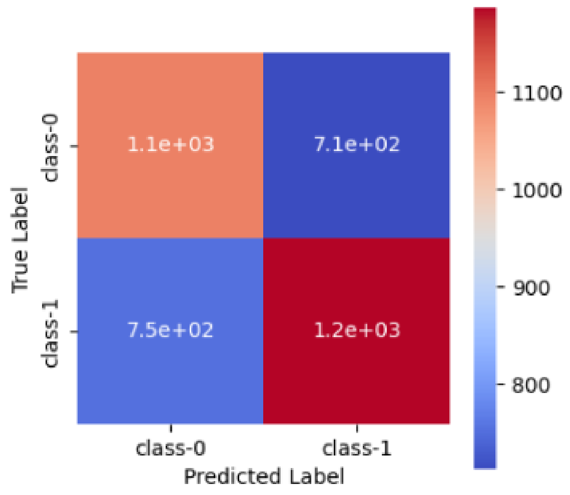
Unlike random forest where trees are built in parallel, XGboost built trees sequentially and every next tree tries to minimize the error made by the previous tree. The sequential boosting approach makes XGBoost special and very powerful in dealing with different large datasets.

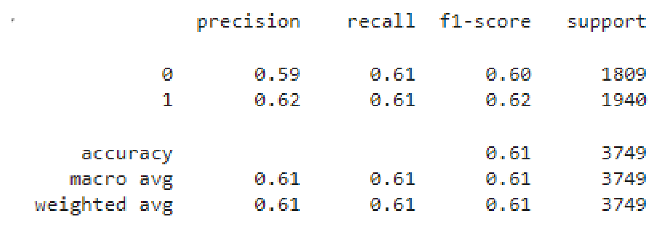
Additional to the improving accuracy, XGBoost is designed for fast training and there are numerous optimizations that allow for parallel computations on the multiple cores for construction of the trees and for the training phase. This makes it faster than most other global implementations of gradient boosting.XGBoost uses a more efficient method of pruning the trees. It controls the complexity using a property known as max depth, with which the model will not grow branches when the gain becomes a negative value, therefore minimizing overfitting.

The below screenshots provide an instance where we used the XGBoost classifier and also include the training and test accuracy according to our preprocessed dataset.



```
Train accuracy for xgboost 0.8245415138379459

Test accuracy for logistic regression 0.6089623899706589
```

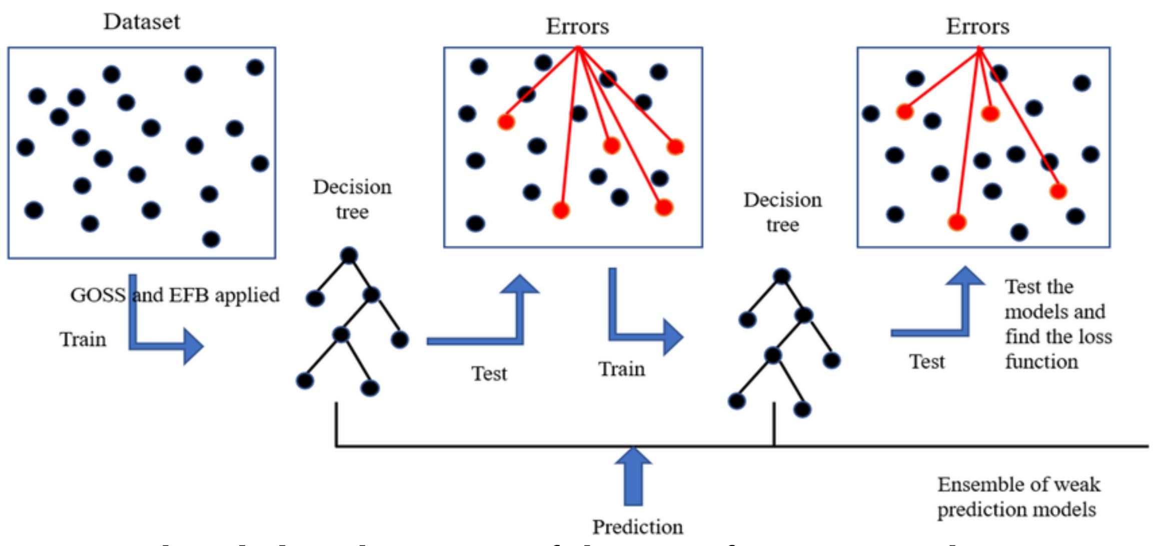




```
              precision    recall  f1-score   support

           0       0.59      0.61      0.60      1809
           1       0.62      0.61      0.62      1940

    accuracy                           0.61      3749
   macro avg       0.61      0.61      0.61      3749
weighted avg       0.61      0.61      0.61      3749
```

LightGBM (Light Gradient Boosting Machine)

LightGBM is an open-source gradient-boosting framework that is both distributed and high speed that is developed by Microsoft. It is econometric, scalable and accurate for producing results that are important in decision making. It is developed based on the decision trees to enhance the performance of the model and decrease the memory consumption. It also includes several new strategies, such as the GOSS gradient-based one-side sampling that retains only the instances with the big gradients during the training in order to save memory and time. Moreover, LightGBM uses a histogram-based approach for constructing the trees for high performance. These, combined with such optimizations as leaf-wise tree growth and proper formats of storing data, provide LightGBM with high performance that places it ahead of the other gradient-boosting frameworks.



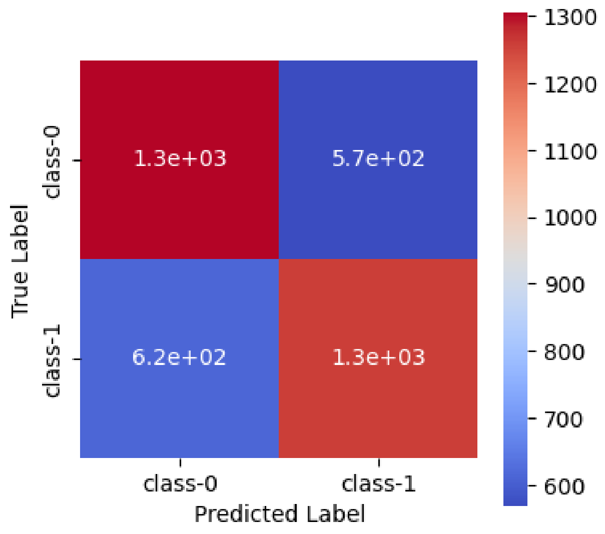
Histogram-based algorithm is one of the main features in LightGBM. Instead of assessing all the conceivable splits of a feature, as many other algorithms do, LightGBM sorts continuous feature values into sets of intervals of equal size (histograms). This makes it easier to decide on the best split and generally the training becomes faster.

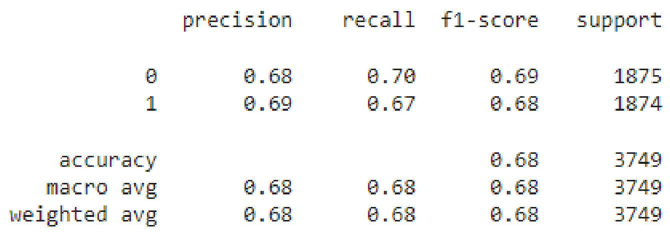
while in LightGBM trees expand leaf by leaf. It splits the leaf that gives the most reduction to loss, enabling greater depths and increased complexity thus increasing accuracy. But may be associated with increased tree depths, which might overfit if regulated poorly.And also this model can handle both categorical features and imbalanced data.

The below screenshots provide an instance where we used the LightGBM classifier and also include the training and test accuracy according to our preprocessed dataset.

Train accuracy for LightGBM - 0.6909636545515172

Test accuracy for LightGBM - 0.6841824486529742





```
              precision    recall  f1-score   support

           0       0.68      0.70      0.69      1875
           1       0.69      0.67      0.68      1874

    accuracy                           0.68      3749
   macro avg       0.68      0.68      0.68      3749
weighted avg       0.68      0.68      0.68      3749
```

Logistic Regression

Logistic regression is another classification algorithm of functions in machine learning as it forecasts a binary categorical variable using a logistic function.The logistic regression model applies the algorithm of odds function to the linear combination of the features in order to find the probability of occurrence of an event. The model then quantifies the probability to discrete binary states.

Sigmoid Function

Formula

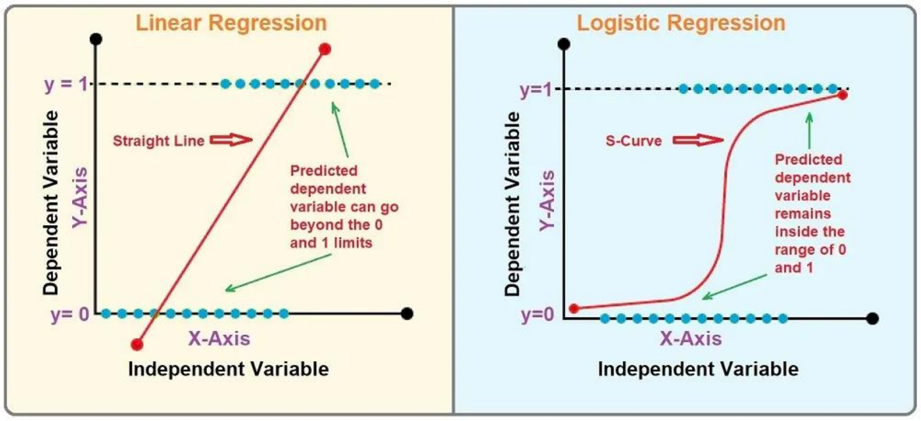
$$S(x) = \frac{1}{1 + e^{-x}}$$

$S(x)$ = sigmoid function

e = Euler's number

It transforms any actual value into another value within the interval 0,1. The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit. It forms a curve like the “S form.’

The S-form curve is named the sigmoid function or the logistic function.In logistic regression we have the threshold value that helps to define the probability of either 0 or 1. For example, a value greater than the threshold value approaches 1 and if the value is less than the threshold values then it approaches 0.



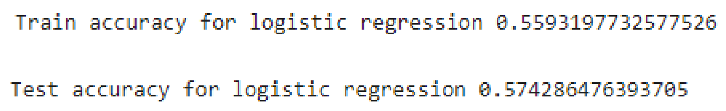
Types of Logistic Regression

Binomial: In binomial logistic regression, the value of the dependent variables can be only two types in nature.

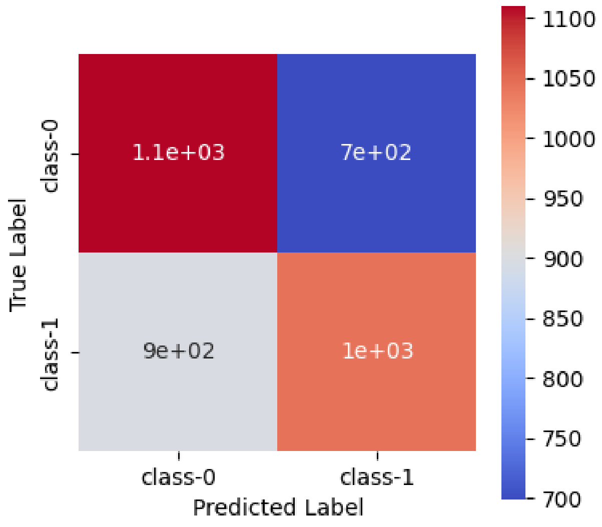
Multinomial: In multinomial logistic regression, it is possible to develop 3 or more measurement types that are not ordered by the dependent variable.

Ordinal: In ordinal logistic regression models, there can be 3 or more ordered types of dependent variables.

The below screenshots provide an instance where we used the Logistic Regression classifier and also include the training and test accuracy according to our preprocessed dataset.



```
Train accuracy for logistic regression 0.5593197732577526

Test accuracy for logistic regression 0.574286476393705
```

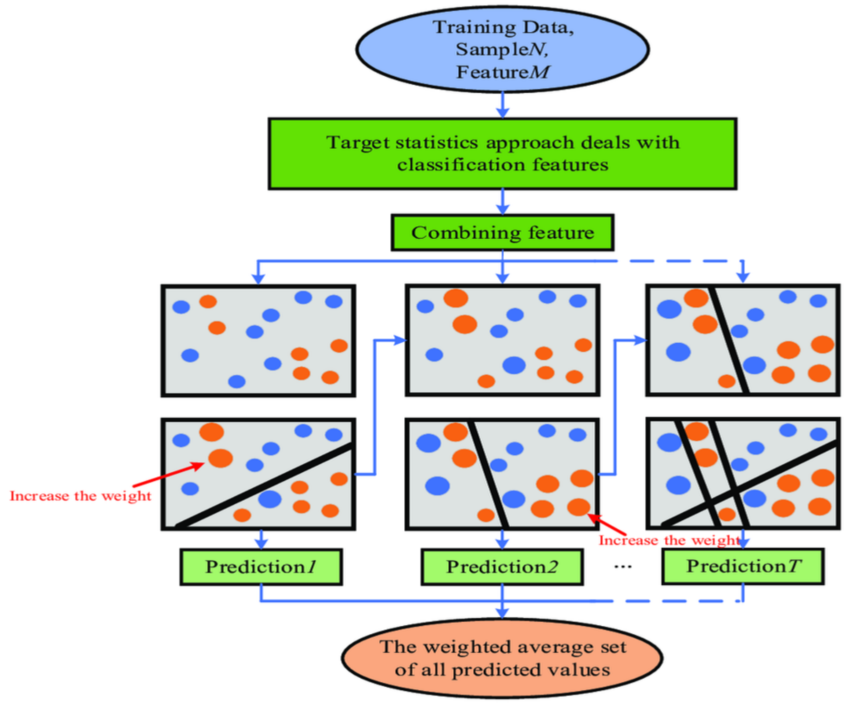


```
              precision    recall  f1-score   support

           0       0.55      0.61      0.58      1809
           1       0.60      0.54      0.57      1940

    accuracy                           0.57      3749
   macro avg       0.58      0.58      0.57      3749
weighted avg       0.58      0.57      0.57      3749
```

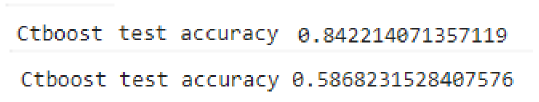
CatBoost

Categorical boosting is for problems like regression and classification containing a very large number of features in independent variables. Catboost is a variation of the gradient boosting algorithm which can deal with the categorical and numerical data as well. It does not need any feature encoding techniques like one hot encoder or label encoder to convert the categorical features into numeric features. It also employs an algorithm called symmetric weighted quantile sketch(SWQS) which has an ability to remove the missing values themselves to eliminate the problem of overfitting in the dataset and enhance the overall performance of the given dataset.

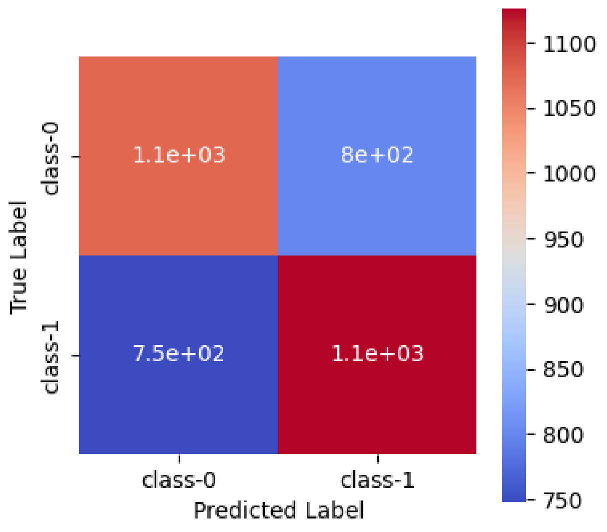
By building a CatBoost model, the creation of the trees is symmetrical, both for left and right sides. This leads to improved training and inference in addition to better results for data with high-cardinality categorical feats. Other benefits of the symmetric trees are related to model compression and optimization.



CatBoost has been optimized to work with both CPU and GPU. It is used to process large amounts of data which makes it faster than other gradient boosting algorithms and tries to minimize the problem of overfitting.
It includes cross-validation as a built-in option, which makes the process of hyperparameter tuning easier. It offers capabilities to perform hyperparameters search on Grid Search and Random Search all in auto mode.
The below screenshots provide an instance where we used the CatBoost classifier and also include the training and test accuracy according to our preprocessed dataset.



```
Ctboost test accuracy  0.842214071357119
Ctboost test accuracy 0.5868231528407576
```



```
              precision    recall  f1-score   support

           0       0.59      0.57      0.58      1875
           1       0.58      0.60      0.59      1874

    accuracy                           0.59      3749
   macro avg       0.59      0.59      0.59      3749
weighted avg       0.59      0.59      0.59      3749
```

Gaussian Naive Bayes (Gaussian NB)

Gaussian NB is a type of probabilistic machine learning algorithm that is normally used for the classification of objects. It is a Naive Bayes classifier type which is based on Bayes’ Theorem of Probability. What distinguishes the Gaussian version is the fact that implicit continuous features are assumed to be Gaussian distributed. It is especially easy, quick and accurate for some classification problems especially where there is at least a reasonable level of independence of the features.

Naive Bayes Classifier

Naive Bayes models are termed ‘naive’ because it’s assumed that every feature is independent of the other with regard to the class label, which is usually not the case. However, even if such an assumption is unrealistic, Naive Bayes yields surprisingly high results, especially in cases such as text categorization or spam detection.

Bayes' Theorem

Gaussian Naive Bayes involves calculation using Bayes’ Theorem that is a probability of an event concerning prior belief of an occasion. The formula is expressed as

The Formula For Bayes' Theorem Is

$$P(A|B) = \frac{P(A \bigcap B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)}$$

where:

$P(A) =$ The probability of A occurring

$P(B) =$ The probability of B occurring

$P(A|B) =$ The probability of A given B

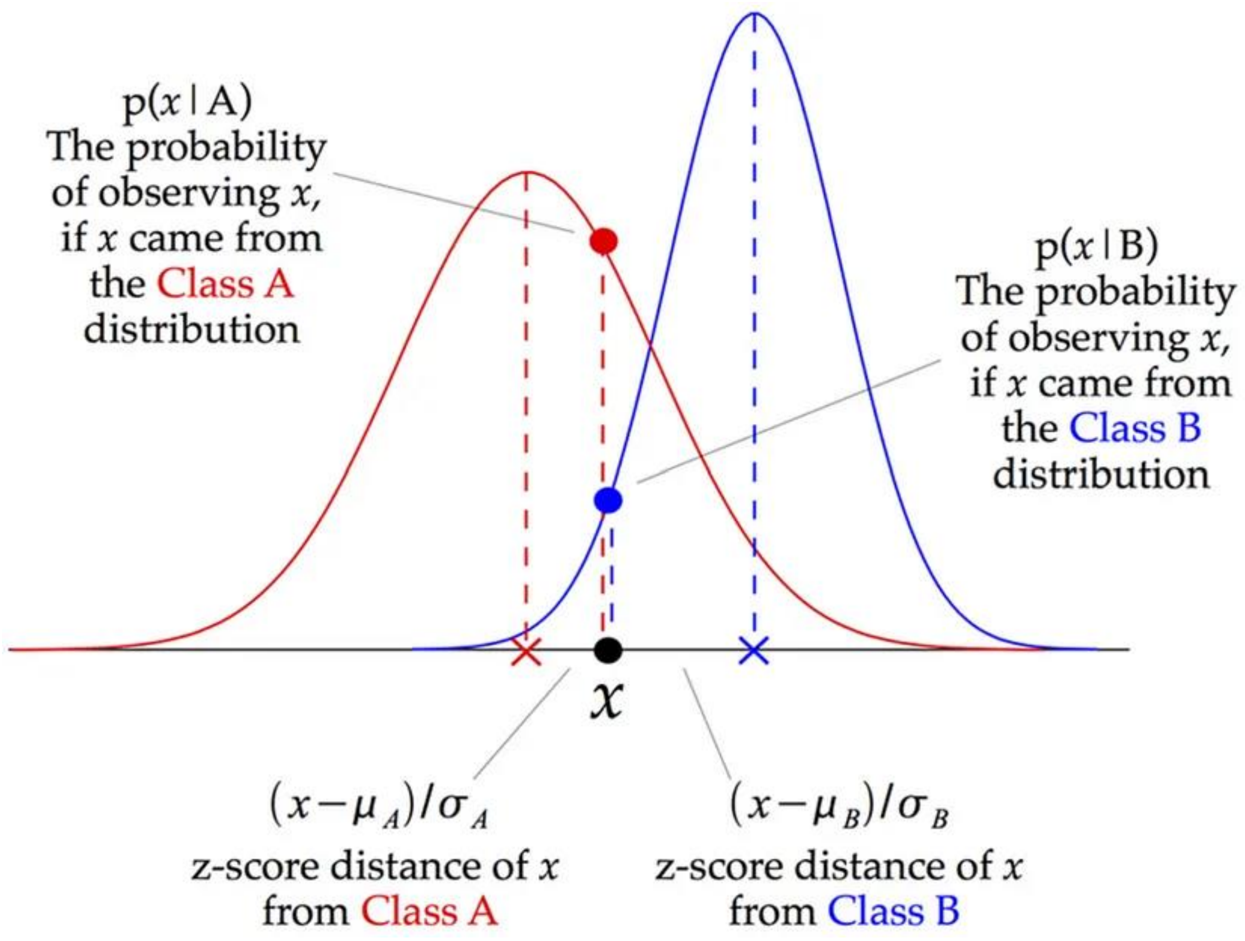
$P(B|A) =$ The probability of B given A

$P\left(A \bigcap B\right)) =$ The probability of both A and B occurring

Gaussian Distribution Assumption

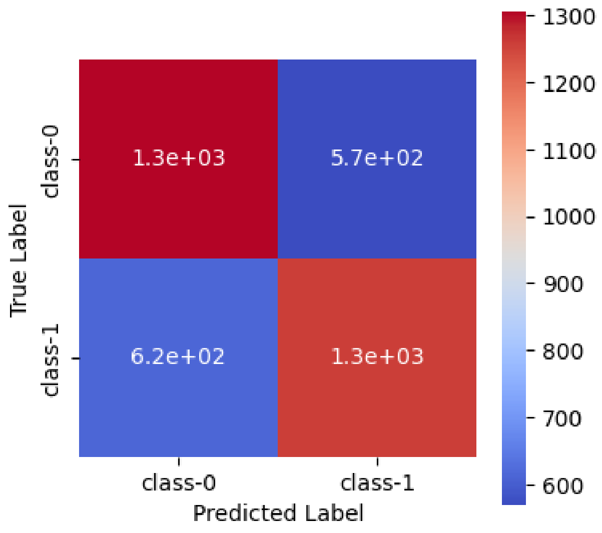
Gaussian Naive Bayes says that the likelihood of the features is assumed to follow normal distribution or in other word is Gaussian distribution. This assumption is important when handling such data that is in continuous form. For each feature, the likelihood is modeled as

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$


$p(x \mid A)$
The probability of observing x, if x came from the Class A distribution
$p(x \mid B)$
The probability of observing x, if x came from the Class B distribution
x
$(x-\mu_A)/\sigma_A$
z-score distance of x from Class A
$(x-\mu_B)/\sigma_B$
z-score distance of x from Class B

The below screenshots provide an instance where we used the Gaussian NB classifier and also include the training and test accuracy according to our preprocessed dataset.

Gaussian nb train accuracy - 0.5667
Gaussian nb test accuracy - 0.5615



```
              precision    recall  f1-score   support

           0       0.68      0.70      0.69      1875
           1       0.69      0.67      0.68      1874

    accuracy                           0.68      3749
   macro avg       0.68      0.68      0.68      3749
weighted avg       0.68      0.68      0.68      3749
```

Note

Although XGBoost and CatBoost models have the highest training accuracies and almost the same testing accuracies as Random Forest Classifier, the reason to choose Random Forest Classifier is that it has the best F1-scores for both the classes when compared among the other 5 models that we trained. So we choose Random forest model for further implementation.

Using Explainable AI on our finalized model

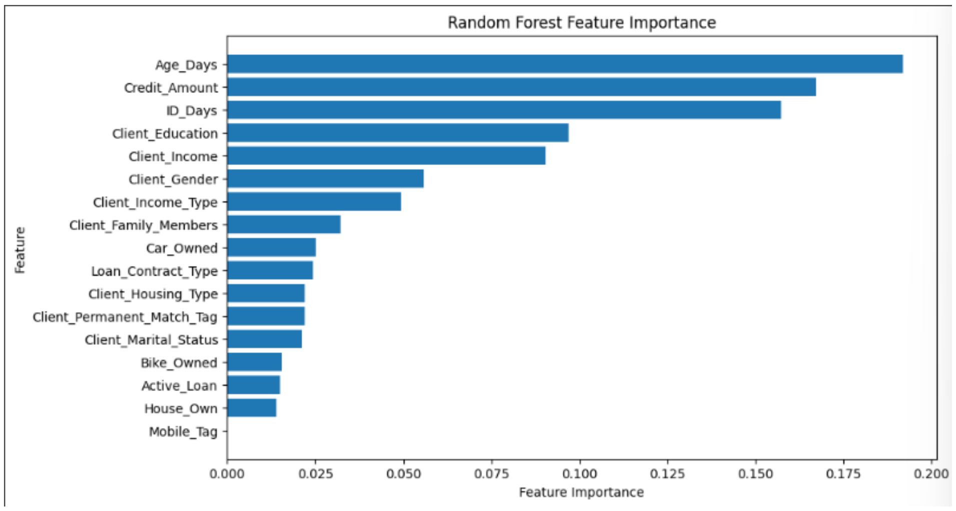
These days, many different businesses, including banking and healthcare, use machine learning and deep learning models. It is not an option to retain machine learning or deep learning models as black boxes due to their widespread use. So, we thought of utilizing **Explainable AI (XAI)** methodologies to acquire insights on how machine learning models perform with the data we offer. These make it possible for end users to understand and rely on the output produced by machine learning algorithms.

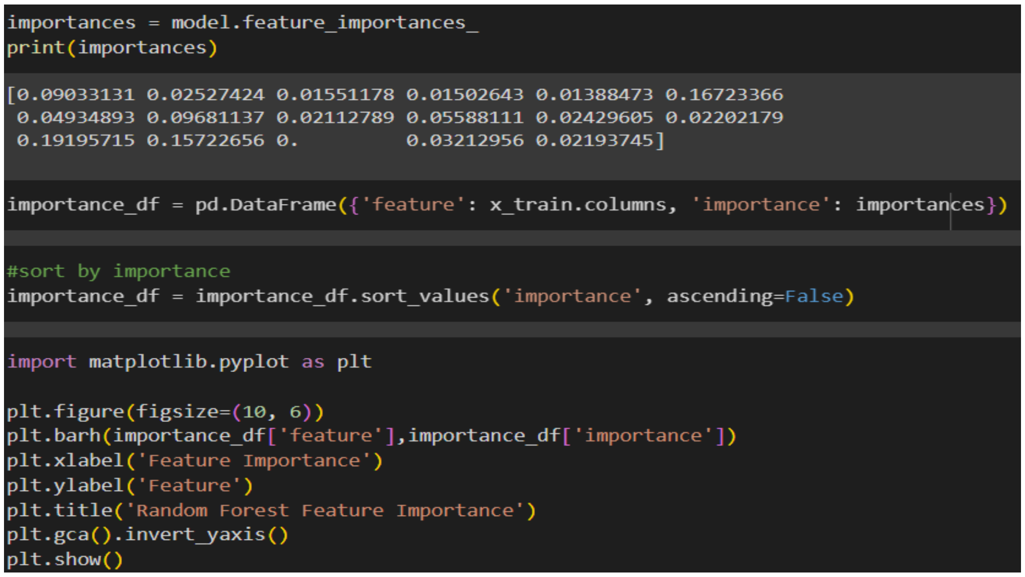
Here are a few Explainable AI beliefs which contribute to establishing trust.

1. **Transparency** - Making certain that stakeholders are aware of how the models make decisions.
2. **Fairness** - Ensuring that all individuals, especially those in protected groups (race, religion, gender, handicap, ethnicity), are treated equally in the decisions made by the models.
3. **Trust** - Determining how confident human users are in the AI system.
4. **Privacy** - Ensuring that private user data is safeguarded.
5. **Interpretability** - supplying comprehensible justifications for their forecasts and results.
6. **Robustness** - The ability to continue providing consistent and dependable performance in the face of uncertainty or unforeseen circumstances, despite changes in input data or model parameters.

   We applied 3 XAI techniques to **Random Forest Classifier**, our **final model**, to gain insight into how the model understands the data and how each feature influences the model's output.

Feature Importance





```
importances = model.feature_importances_
print(importances)
```

```
[0.09033131 0.02527424 0.01551178 0.01502643 0.01388473 0.16723366
 0.04934893 0.09681137 0.02112789 0.05588111 0.02429605 0.02202179
 0.19195715 0.15722656 0.         0.03212956 0.02193745]
```

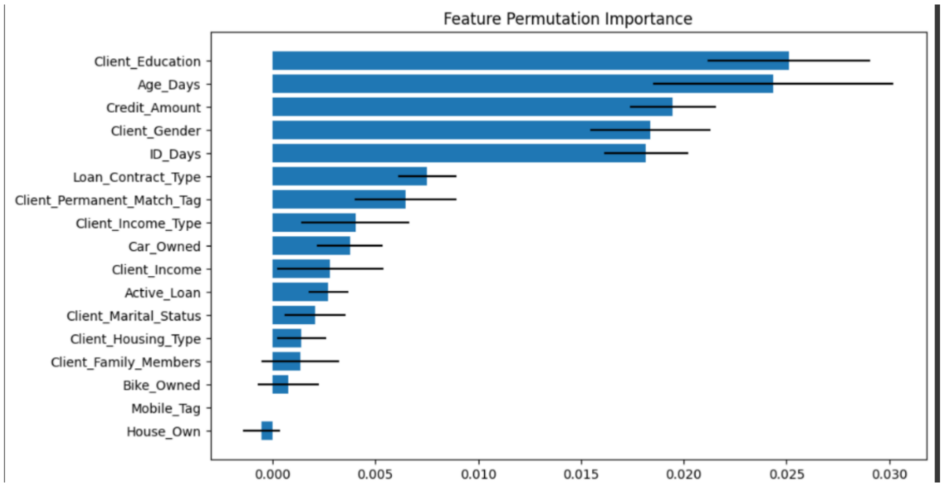
```
importance_df = pd.DataFrame({'feature': x_train.columns, 'importance': importances})
```

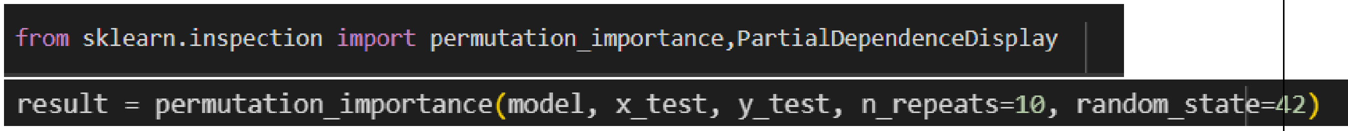
```
#sort by importance
importance_df = importance_df.sort_values('importance', ascending=False)
```

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.barh(importance_df['feature'],importance_df['importance'])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Random Forest Feature Importance')
plt.gca().invert_yaxis()
plt.show()
```

- **Age_Days, Credit_Amount, ID_Days** are the most influential features in predicting the target variable.
- **Client_Education,Client_Income** also shows a higher importance.
- **Bike_Owned, Active_Loan, House_Own** shows less importance indicating they have a very low impact on model's performance.

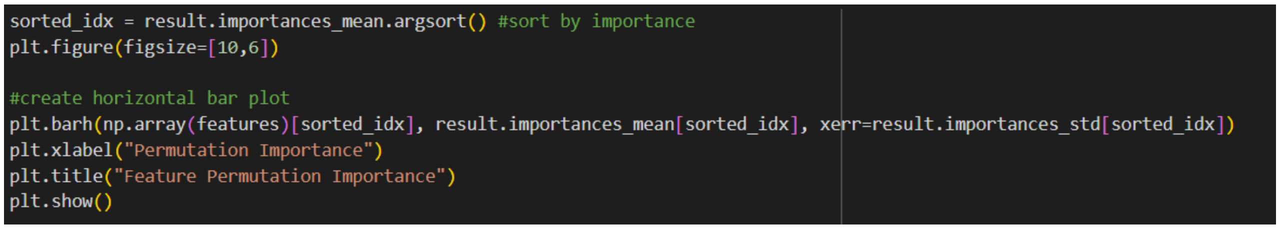
Feature Permutation Importance





```
from sklearn.inspection import permutation_importance,PartialDependenceDisplay
```

```
result = permutation_importance(model, x_test, y_test, n_repeats=10, random_state=42)
```

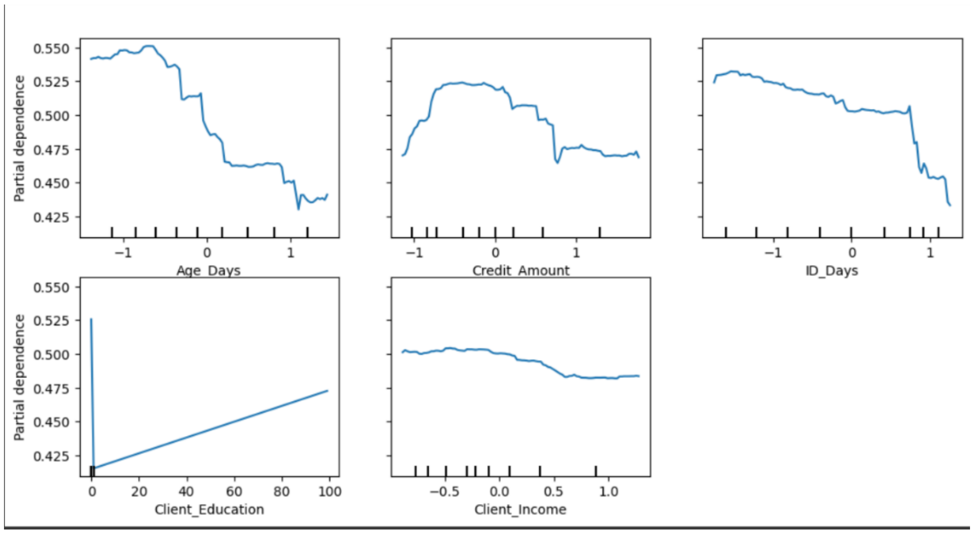


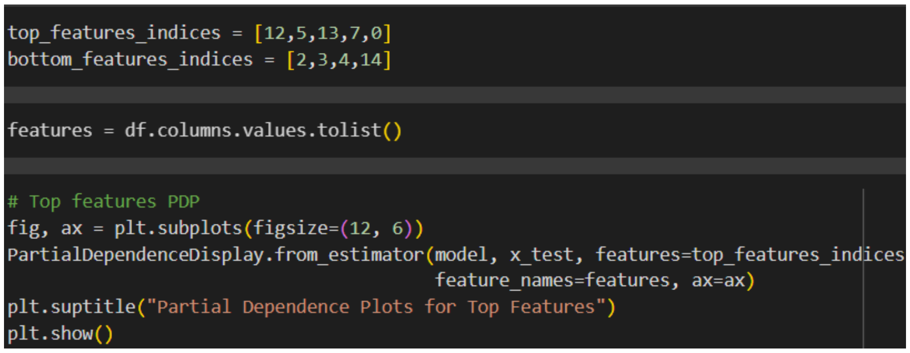
```
sorted_idx = result.importances_mean.argsort() #sort by importance
plt.figure(figsize=[10,6])

#create horizontal bar plot
plt.barh(np.array(features)[sorted_idx], result.importances_mean[sorted_idx], xerr=result.importances_std[sorted_idx])
plt.xlabel("Permutation Importance")
plt.title("Feature Permutation Importance")
plt.show()
```

This is a way to measure the importance of features in a model by observing how model's performance degrades when a specific feature is randomly shuffled keeping other features constant. The black bars show the variability in the importance scores across shuffling repetitions.

- When **Client_Education, Age_Days** features are shuffled, it causes a significant drop in model performance indicating these 2 features play a major role in determining the outcome of the model.
- **Loan_Contract_Type, Client_Permanent_Match_Tag** (Indication if client contact address does not match permanent address) and **Car_Owned** features still contribute to the model's performance but shuffling them has a small performance drop indicating the model relies less on them for predictions.

Partial Dependency Plots





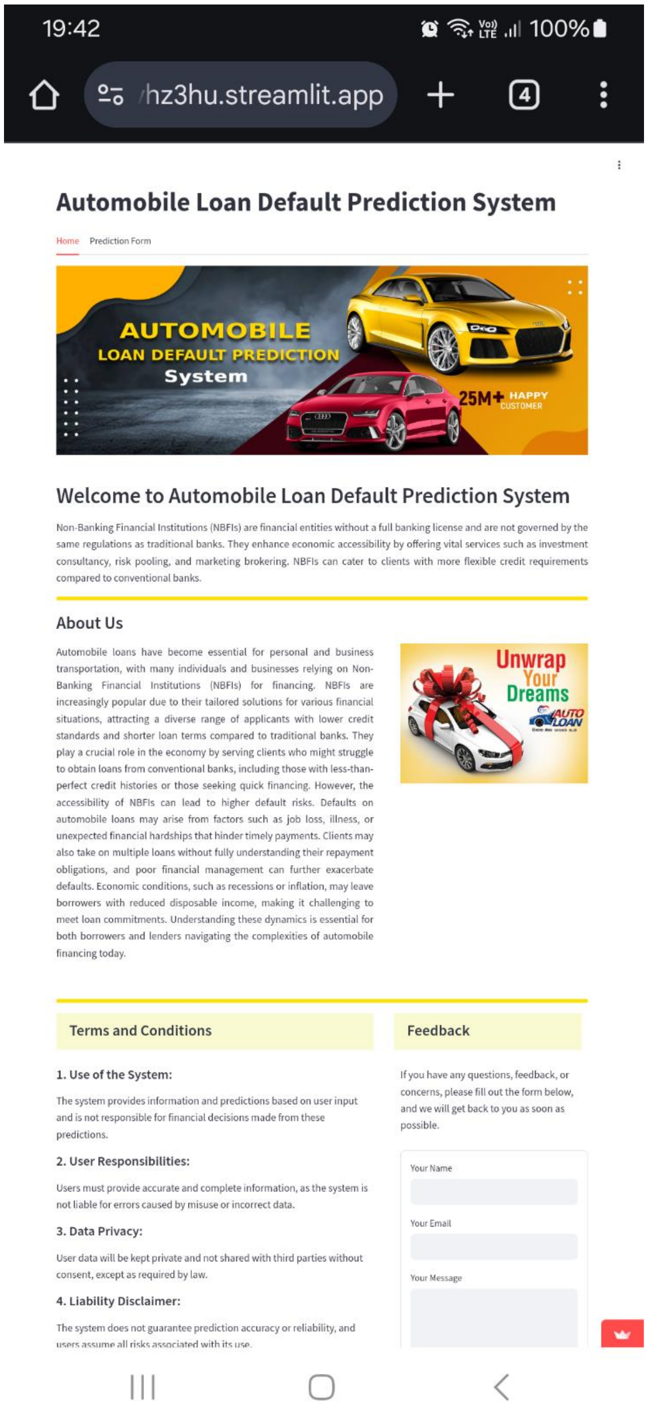
```
top_features_indices = [12,5,13,7,0]
bottom_features_indices = [2,3,4,14]

features = df.columns.values.tolist()

# Top features PDP
fig, ax = plt.subplots(figsize=(12, 6))
PartialDependenceDisplay.from_estimator(model, x_test, features=top_features_indices
                                        feature_names=features, ax=ax)
plt.suptitle("Partial Dependence Plots for Top Features")
plt.show()
```

The above figure illustrates the partial dependency plots for the top 5 features. As shown in the figure, it illustrates how each feature impacts the model's predicted probability of positive class which means the client defaulted on the loan in this problem.

- The curve exhibits a negative trend as **Age_Days** increases, suggesting a decline in the model's prediction probability. This suggests that as age grows, the model predicts a reduced probability for the positive class, meaning older borrowers are less likely to default on their loans.
- When **Client_Education** level grows, it indicates a positive upward trend indicating the probability of defaulting increases. If so, it may be because higher education levels are linked to riskier financial decisions or because larger loans have a higher default rate.

As machine learning algorithms become more complex, there is an increasing need for transparency and an understanding of model behavior, particularly in high stakes domains like finance. In this project, we applied XAI techniques to improve the interpretability of our machine learning models in the context of loan default prediction.

User Interfaces



19:42 100%

hz3hu.streamlit.app

Automobile Loan Default Prediction System

Home Prediction Form

Welcome to Automobile Loan Default Prediction System

Non-Banking Financial Institutions (NBFIs) are financial entities without a full banking license and are not governed by the same regulations as traditional banks. They enhance economic accessibility by offering vital services such as investment consultancy, risk pooling, and marketing brokering. NBFIs can cater to clients with more flexible credit requirements compared to conventional banks.

About Us

Automobile loans have become essential for personal and business transportation, with many individuals and businesses relying on Non-Banking Financial Institutions (NBFIs) for financing. NBFIs are increasingly popular due to their tailored solutions for various financial situations, attracting a diverse range of applicants with lower credit standards and shorter loan terms compared to traditional banks. They play a crucial role in the economy by serving clients who might struggle to obtain loans from conventional banks, including those with less-than-perfect credit histories or those seeking quick financing. However, the accessibility of NBFIs can lead to higher default risks. Defaults on automobile loans may arise from factors such as job loss, illness, or unexpected financial hardships that hinder timely payments. Clients may also take on multiple loans without fully understanding their repayment obligations, and poor financial management can further exacerbate defaults. Economic conditions, such as recessions or inflation, may leave borrowers with reduced disposable income, making it challenging to meet loan commitments. Understanding these dynamics is essential for both borrowers and lenders navigating the complexities of automobile financing today.

Terms and Conditions

1. Use of the System:

The system provides information and predictions based on user input and is not responsible for financial decisions made from these predictions.

2. User Responsibilities:

Users must provide accurate and complete information, as the system is not liable for errors caused by misuse or incorrect data.

3. Data Privacy:

User data will be kept private and not shared with third parties without consent, except as required by law.

4. Liability Disclaimer:

The system does not guarantee prediction accuracy or reliability, and users assume all risks associated with its use.

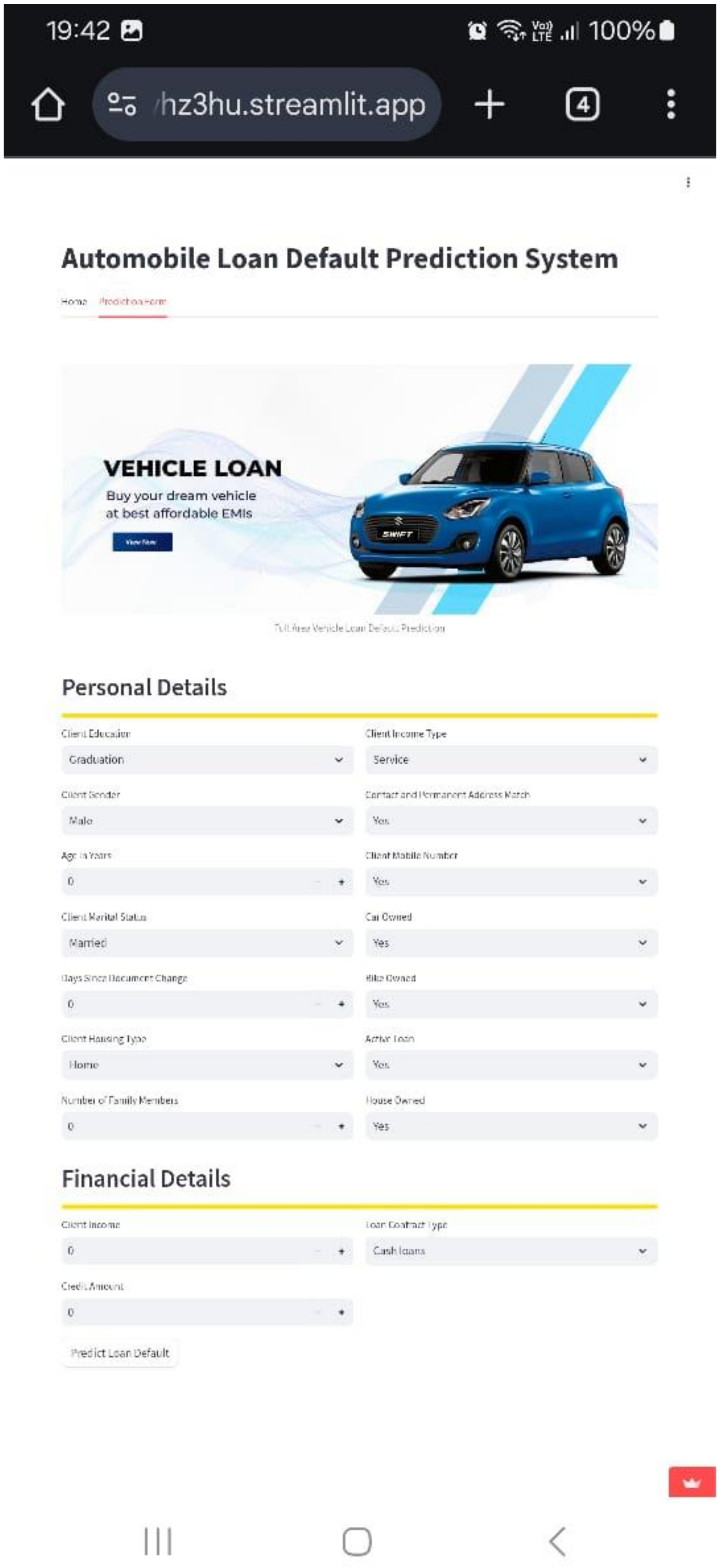
Feedback

If you have any questions, feedback, or concerns, please fill out the form below, and we will get back to you as soon as possible.

Your Name

Your Email

Your Message



Automobile Loan Default Prediction System

Home   Prediction Form

Full Area Vehicle Loan Default Prediction

Personal Details

Client Education	Graduation	Client Income Type	Service
Client Gender	Male	Contact and Permanent Address Match	Yes
Age in Years	0	Client Mobile Number	Yes
Client Marital Status	Married	Car Owned	Yes
Days Since Document Change	0	Bike Owned	Yes
Client Housing Type	Home	Active Loan	Yes
Number of Family Members	0	House Owned	Yes

Financial Details

Client Income	0	Loan Contract Type	Cash loans
Credit Amount	0		

Predict Loan Default

Conclusion

Our project was developed for a non-banking financial institution that operates without oversight from any national or international monetary authorities. For that, a suitable dataset was selected using the Kaggle platform under the title automobile loan default.

In our project, we developed a predictive model that evaluates a client's background to determine their likelihood of securing a loan. This model provides insights into the factors influencing a client's ability or inability to meet loan eligibility criteria.

One of the most challenging aspects of this project was data preprocessing, essential for preparing the data to fit into a predictive model. The raw dataset contained numerous inconsistencies, missing values, and irrelevant information that needed to be addressed. Initially, we removed unnecessary columns that did not contribute to the final prediction. We then handled missing values and resolved inconsistencies by cleaning the data. This included removing duplicate entries, handling null values, and filtering out specific rows and columns based on predefined criteria. Additionally, we converted categorical data into numerical formats to ensure compatibility with the model. To better understand the dataset, we created visualizations such as box plots and graphs, which helped identify outliers and distributions. We also examined correlations between different features in the dataset, allowing us to gain deeper insights into relationships within the data and enhancing our understanding of the overall dataset structure.

We had to further enhance the dataset by doing more preprocessing, and we used various techniques to balance the dataset.

For our project, we selected and implemented five different classification techniques to evaluate their performance on the chosen dataset. Each model was rigorously tested to determine its accuracy and effectiveness. After thorough experimentation and analysis, we identified that the Random Forest Classifier consistently outperformed the other models, providing the highest test accuracy on the preprocessed dataset. This model's ability to handle complex data patterns, combined with its robustness in managing both categorical and numerical variables, made it the most suitable choice for our classification task. The performance of the Random Forest Classifier demonstrated its effectiveness in delivering reliable predictions, solidifying it as the optimal model for our dataset.

After building, training, and deploying the predictive model, we developed a user-friendly interface using Streamlit to make the model easily accessible. This interface allows users to interact with the model by inputting the necessary customer data, such as financial and personal details, to receive real-time predictions on loan applications. The interface is intuitive and designed to guide users through the process seamlessly. By entering the required values, users can quickly assess whether a potential borrower is likely to default on a loan, helping them make informed lending decisions with greater confidence.

Finally, we hope that our project is a high-quality system that provides solutions to many problems in current society.

Reference

https://www.analyticsvidhya.com/blog/2022/11/hyperparameter-tuning-using-randomized-search/
https://www.geeksforgeeks.org/svm-hyperparameter-tuning-using-gridsearchcv-ml/
https://medium.com/@pwrxndr/catboost-classifier-a-simple-guide-for-everyone-48a2e3897251
https://medium.com/analytics-vidhya/how-to-improve-naive-bayes-9fa698e14cba
https://www.geeksforgeeks.org/cross-validation-and-hyperparameter-tuning-of-lightgbm-model/
https://medium.com/@dallanoce.fd/explainable-ai-a-complete-summary-of-the-main-methods-a28f9ab132f7
https://www.datacamp.com/tutorial/explainable-ai-understanding-and-trusting-machine-learning-models
https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/
https://www.geeksforgeeks.org/xgboost/
https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/
https://www.geeksforgeeks.org/catboost-ml/
https://www.geeksforgeeks.org/understanding-logistic-regression/
https://www.geeksforgeeks.org/gaussian-naive-bayes/
https://iq.opengenus.org/gaussian-naive-bayes/