Sri Lanka Institute of Information Technology

- Fundamentals of Data Mining -

[IT3051]

Mini Project – Final Report

Group – G07

Jayasooriya C.A	**-**	**IT20250942**
Amanullath M.U	**-**	**IT20155520**
Gavindya N.A.C	**-**	**IT20409982**
Bandara T.M.Y.M	**-**	**IT20492052**
Rathnaweera R.P.W.G	**-**	**IT20237554**

Automobile Loan Approval System

GitHub link – Mobi_Loan_Git_Repository

Deployed link – Mobi Loan – Intelligent Loan System

Video Links -

Drive Link Containing the Recorded Presentation

YouTube Link Containing the Recorded Presentation

Contents

Declaration

This project report, or any portion of it, was not copied from the internet or any other source, nor was it based on work produced by any organization, institution, another institute, or previous student project team at the Sri Lanka Institute of Information Technology.

Student Number	Name	Signature
IT20250942	**Jayasooriya C.A**	[illegible]
IT20155520	**Amanullath M.U**	[illegible]
IT20409982	**Gavindya N.A.C**	[illegible]
IT20492052	**Bandara T.M.Y.M**	[illegible]
IT20237554	**Rathnaweera R.P.W.G**	[illegible]

Abstract

The world is now dealing with a number of technical issues, but there are also a significant number of them that are being resolved thanks to cutting-edge technology. Most of the time, these issues are comprised of difficulties that people in this world deal with on a daily basis at work or in settings that are familiar to us.

Our aim with this short project is to use Data Mining and Machine Learning approaches to provide remedies to such an issue. Through our answer, we offer a technical method that might be used to either resolve the issue or lessen its effects on the populace and society.

We, therefore, discuss our solution to the following issue.

- **Automobile loan approval system.**

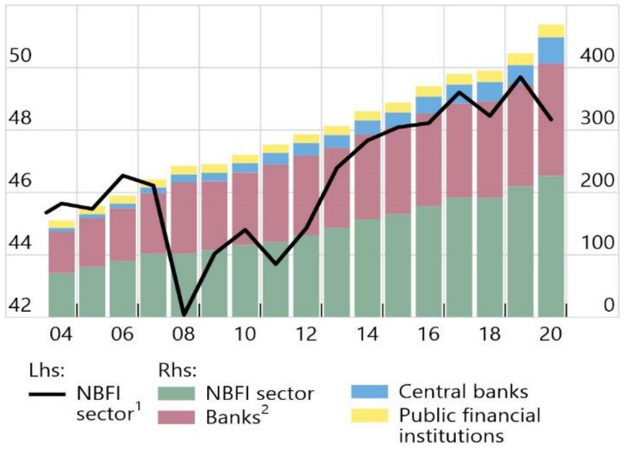
Acknowledgment

We are very grateful to Mr. Prasanna Sumathipala for his assistance and direction while we finished this module and for allowing us to produce this project report. We would like to express our sincere gratitude to Ms. Madhusha Gunarathne in particular for taking the time to review and accept our project proposal.

Introduction

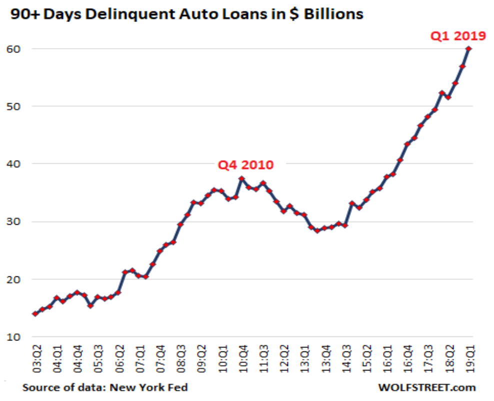
In today's context due to the current financial crisis and with the increased inflation rate and the increase in the interest rates, consumers have been stretched thin with regard to financial strength. This has created issues within the financial system.

Banks and Non-banking Financial Institutions (NBFI) are the pillars of a country's economy, if these institutions are to fail, the entire economy will face the threat of collapse. With the above-said reasons, one of the main threats to the current financial system is customers defaulting on the obtained loans. This has mainly affected the NBFIs as they do not have the strongest support from the Central Banks like Banking institutions.



Effect of financial institutions due to loan default

With the problems and inadequacy in public transport and every household needing a private vehicle, out of the overall loans, a high volume of loans belongs to the Automobile / Auto loan category, and they are the most vulnerable for customers defaulting, as the majority do not provide any financial return. Thus, defaulting on automobile loans has increased significantly in recent times and NBFI profit margins have been hit largely due to this as they are the prominent lenders of auto loans.



We at group CYWAA identified a market opportunity to help NBFIs by developing an intelligent software intelligence solution that can predict the customer loan defaulting possibility and provide a system for the NBFIs to approve or reject customer loan requests with high accuracy.

a collaborative effort of

CYWAA

G-07

An NBFI is a non-financial organization that lacks a full banking license and is not under the supervision of any national or international banking regulatory bodies. The business can provide financial services such as market brokerage, risk pooling, contractual savings, and investing.

These companies have had financial hardships as a result of the unanticipated loss and have failed to mark profit effectively due to the rise in loan defaults in the category of auto loans. To avoid these issues, the business has chosen to assess the client's loan repayment capacity and comprehend the proportional weighting of each factor that affects a borrower's capacity to return a loan. The model will determine whether the client's request would default (fail to repay the loan) on the car payment based on observations by studying previous data on lending loans to clients.

This might lessen the financial loss that could result from a client's payment default by enabling the business to estimate whether the client can afford the amount in the allotted time and extend the loan.

- Thus, **Mobiloan** was produced, Mobiloan is an intelligent software solution that addresses the aforementioned business requirement and assists NBFIs in decision-making.
- Mobiloan utilizes a predictive model which was custom-built and tuned for the specific task and trained with a live dataset that mimics the business requirements.

MOBI

LOAN

Data Description

The dataset chosen contains data from a non-banking financial institution (NBFI) or non-bank financial company (NBFC) is a Financial Institution that does not have a full banking license or is not supervised by a national or international banking regulatory agency. NBFC facilitates bank-related financial services, such as investment, risk pooling, contractual savings, and market brokering.

The aforementioned NBFI is struggling to mark profits due to an increase in defaults in the vehicle loan category. The company aims to determine the client's loan repayment abilities and understand the relative importance of each parameter contributing to a borrower's ability to repay the loan.

The data collection includes 121856 rows of historical information about consumers who applied for a car loan and were approved, together with 40 different fields (or attributes).

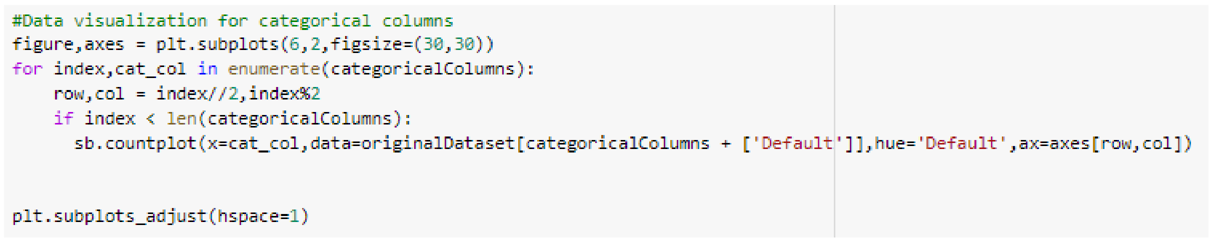
- Data set: [Automobile Loan Default Dataset | Kaggle]

Data Visualization and Preprocessing

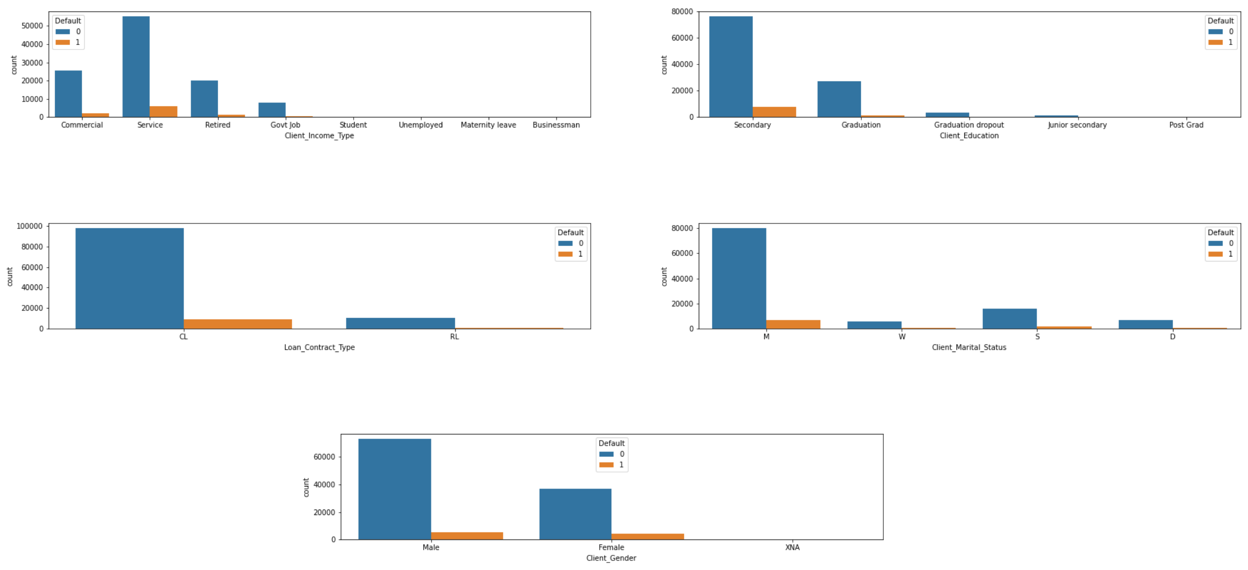
Data visualization

The data set includes both numerical and category data. Categorical data has been visualized as bar charts and numerical data as box plots using the Seaborn Python library (S, n.d.).

Categorical Data Visualization



```
#Data visualization for categorical columns
figure,axes = plt.subplots(6,2,figsize=(30,30))
for index,cat_col in enumerate(categoricalColumns):
    row,col = index//2,index%2
    if index < len(categoricalColumns):
      sb.countplot(x=cat_col,data=originalDataset[categoricalColumns + ['Default']],hue='Default',ax=axes[row,col])


plt.subplots_adjust(hspace=1)
```

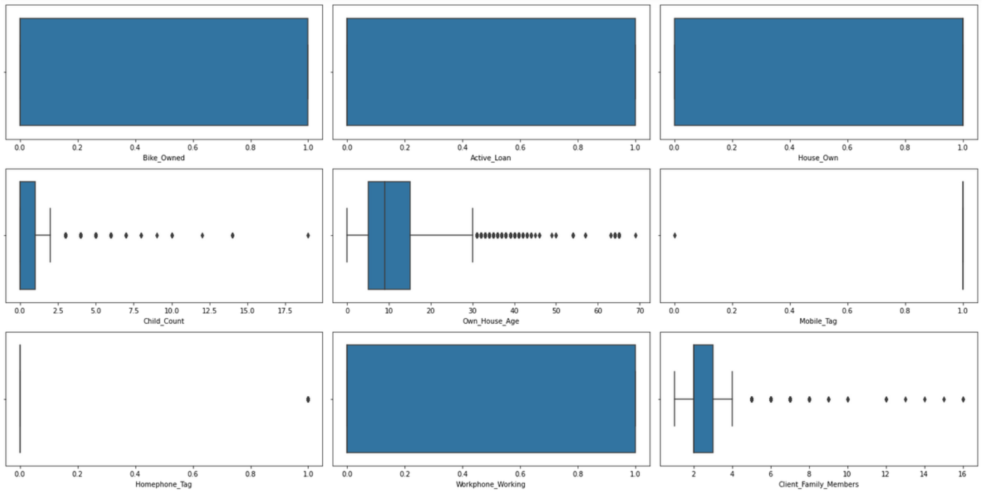


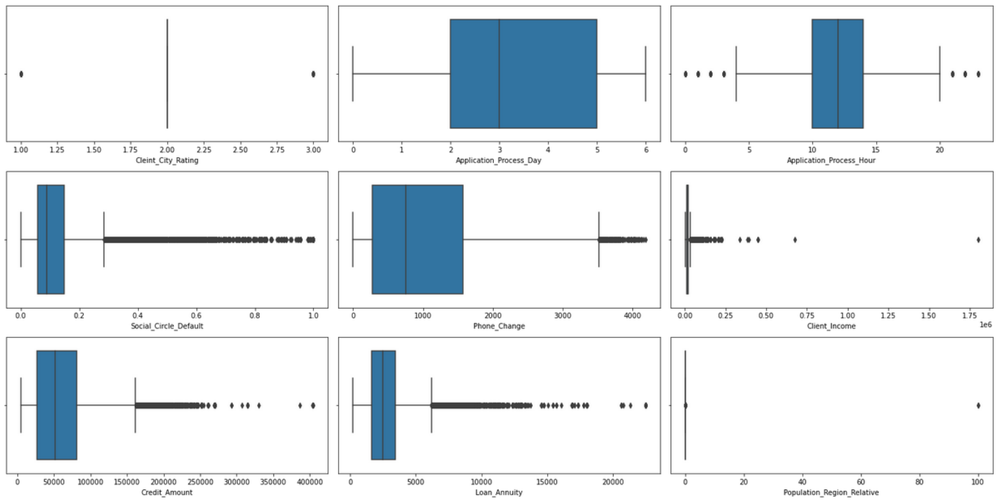
According to the above graphs,

1. Most of the clients earned their incomes from the Service category following commercial and retired categories and the majority of the clients from the said categories did not default on their payments of the loan.
2. The majority of the clients have completed their secondary education followed by graduates and dropouts, there is a minority of clients who are post-grads and in junior secondary education. And most of the said clientele has not defaulted on their obtained loans
3. The largest proportion of the population of loan obtainers was married and is approximately 2/3 of the population. These proportions were followed by Single, Divorced, and Widowed respectively.

4. There were more men than women who applied for the loan. A significant number of debts from both sexes were no longer in default.
5. The highest amount of requests is for cash loans and out of them also the majority of the loans have not defaulted.

Numerical Data Visualization

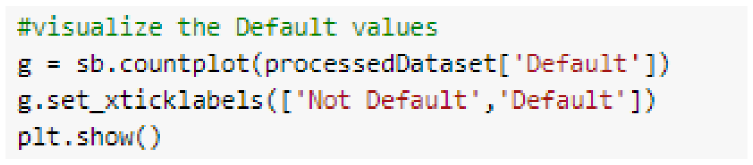




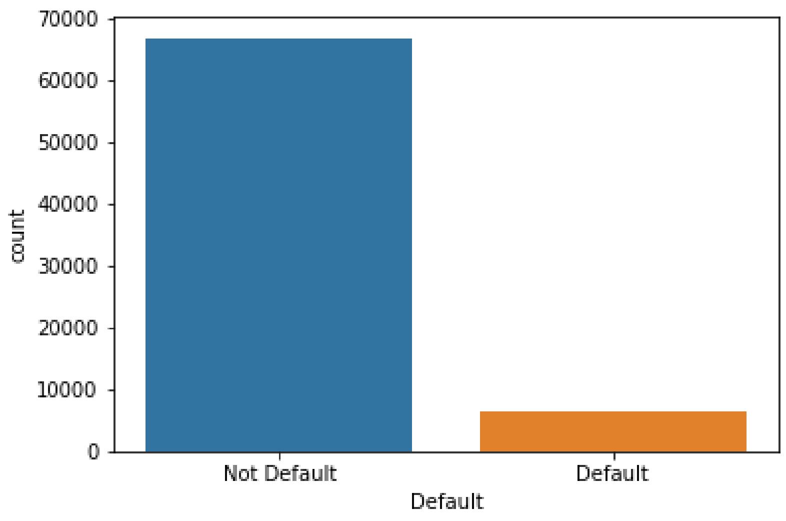
When considering all box plot graphs,

1. Most of the variables contain a considerable amount of outliers
2. In the 'Bike_Owned', 'Active_Loan', and 'House_Own' plots, the lower quartile is equal to the minimum and the upper quartile is equal to the maximum.
3. In certain boxplots above the plot is present as a line because a majority of the values in those variables are the same value (almost over 2/3 of the values

Target variable visualization



```
#visualize the Default values
g = sb.countplot(processedDataset['Default'])
g.set_xticklabels(['Not Default','Default'])
plt.show()
```



As it is visible the data set is not balanced as the data are biased towards the Not Default class category.

This could be concluded by referring to the above chart where it is visible, the difference between the 2 class values. About 90% of the data are biased towards Not Default.

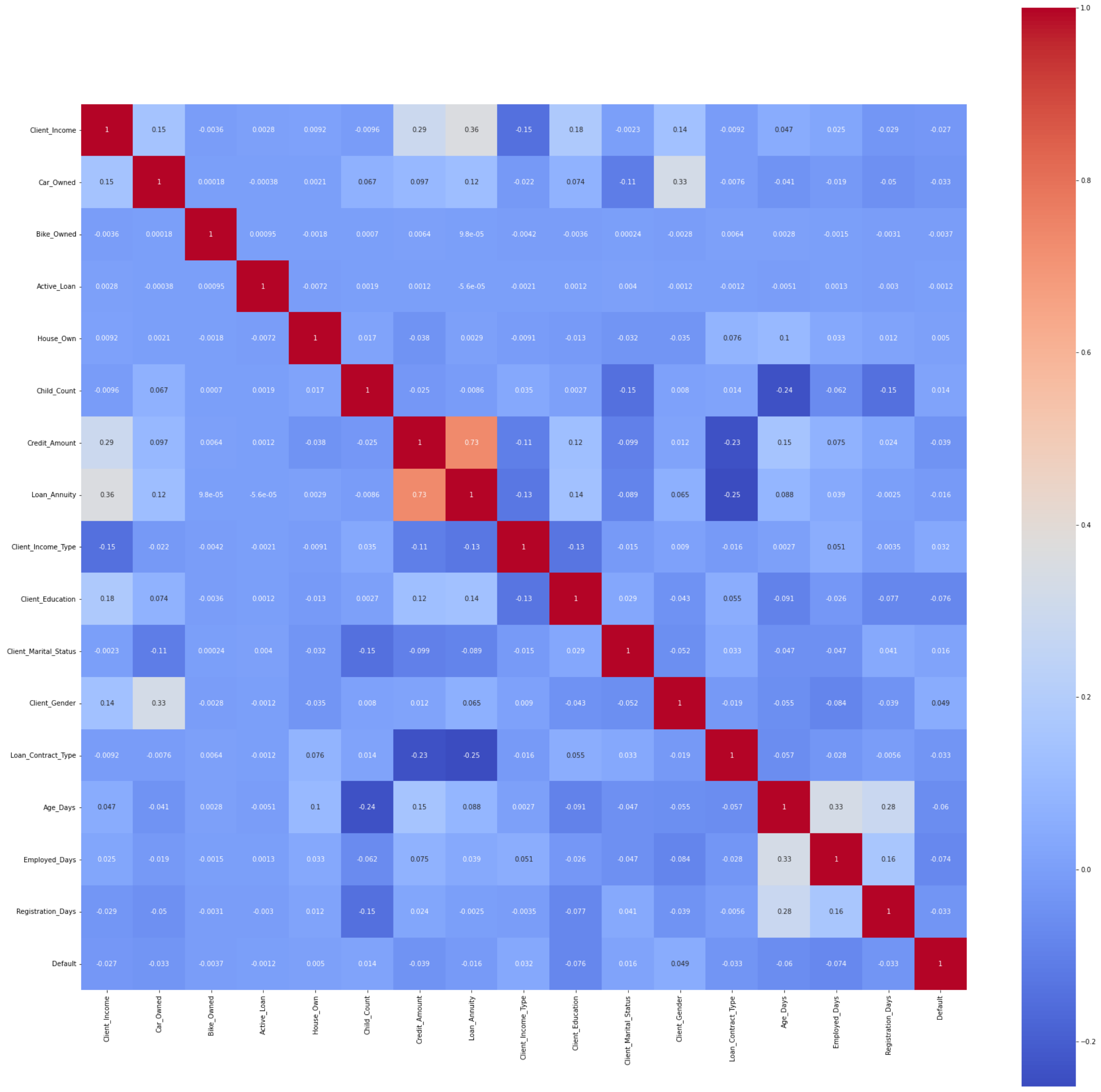
Overcoming the Issue if Dataset Imbalance and Balancing the Data

The imbalance of the data set can be handled using various resampling techniques like under-sampling, oversampling, SMOTE technique, etc. (Wu, 2022).

It has been elaborately explained in the latter parts of the report, how the dataset has been balanced and prepared to omit accuracy and other issues that could be present due to the imbalance of the data.

Correlation Heatmap

Correlation heatmaps are a type of plot that visualize the strength of relationships between numerical variables (Kumar, 2022). Correlation plots are used to understand which variables are related to each other and the strength of this relationship.



Data Preparation and Preprocessing

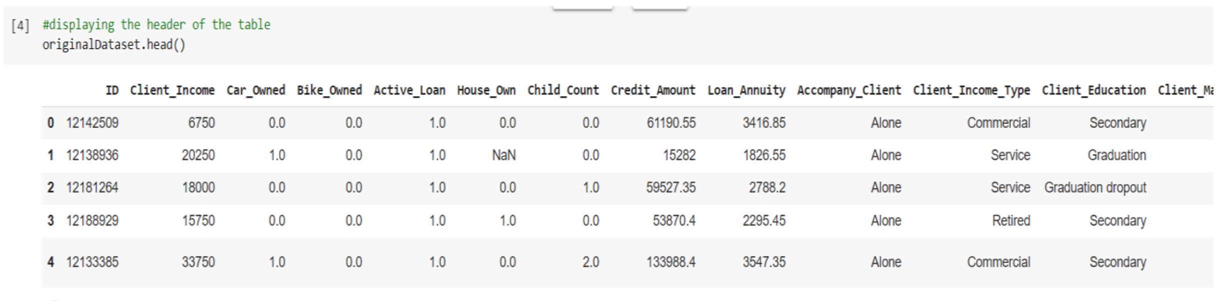
The raw data set must go through numerous phases such as:

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 (Xenonstack, 2018). The procedures and methods used to preprocess data are described below.

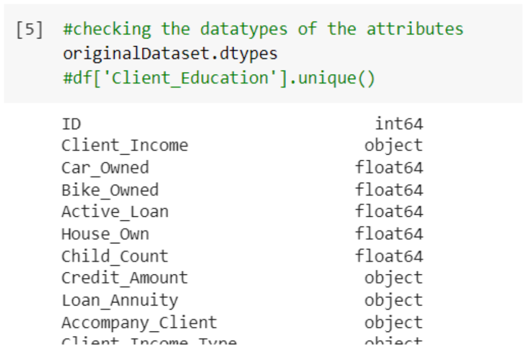
Reading the original dataset

Initially, the original dataset is read to get an overview of the data and the attributes that we are going to handle and to understand the dataset for the model building. This also helps in understanding the data types that we are dealing with in the original dataset

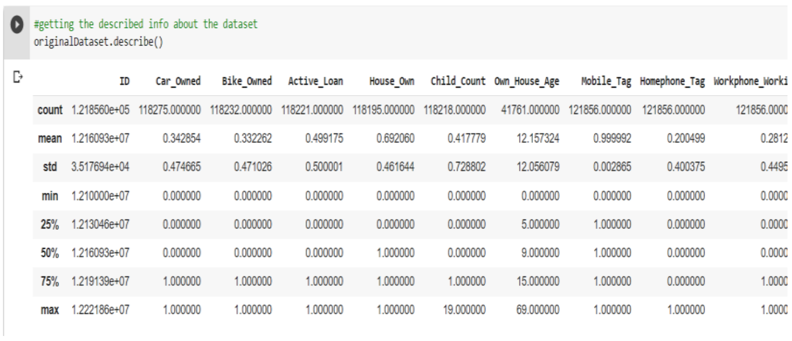


```
[4] #displaying the header of the table
    originalDataset.head()
```

	ID	Client_Income	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Credit_Amount	Loan_Annuity	Accompany_Client	Client_Income_Type	Client_Education	Client_Ma
0	12142509	6750	0.0	0.0	1.0	0.0	0.0	61190.55	3416.85	Alone	Commercial	Secondary	
1	12138936	20250	1.0	0.0	1.0	NaN	0.0	15282	1826.55	Alone	Service	Graduation	
2	12181264	18000	0.0	0.0	1.0	0.0	1.0	59527.35	2788.2	Alone	Service	Graduation dropout	
3	12188929	15750	0.0	0.0	1.0	1.0	0.0	53870.4	2295.45	Alone	Retired	Secondary	
4	12133385	33750	1.0	0.0	1.0	0.0	2.0	133988.4	3547.35	Alone	Commercial	Secondary	



```
[5] #checking the datatypes of the attributes
    originalDataset.dtypes
    #df['Client_Education'].unique()

    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
```



```
#getting the described info about the dataset
originalDataset.describe()
```

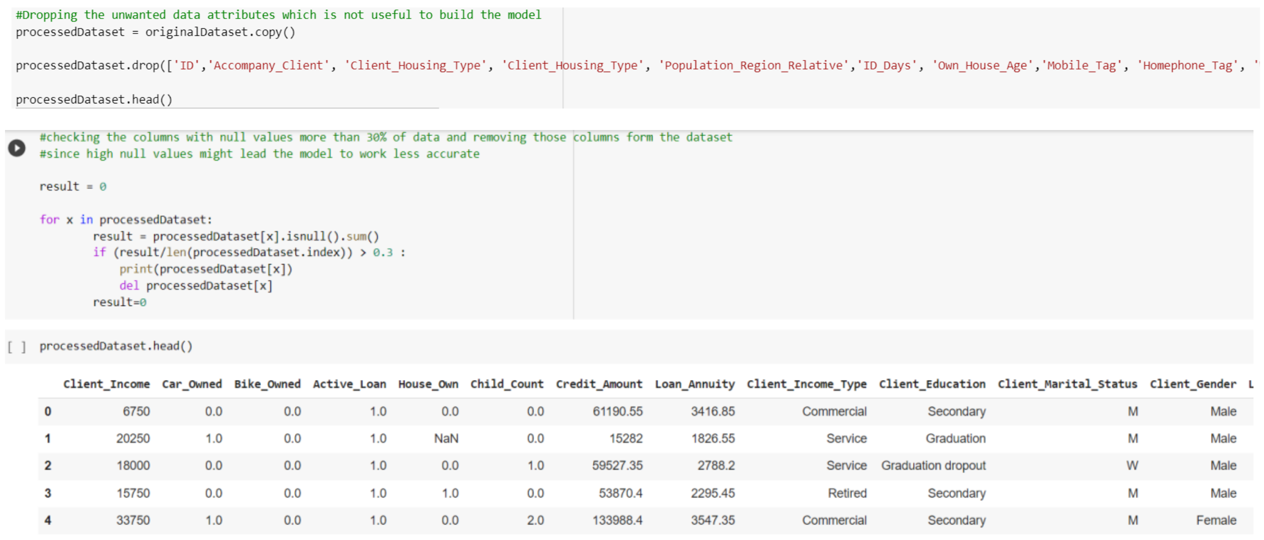
	ID	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Own_House_Age	Mobile_Tag	Homephone_Tag	Workphone_Worki
count	1.218560e+05	118275.000000	118232.000000	118221.000000	118195.000000	118218.000000	41761.000000	121856.000000	121856.000000	121856.0000
mean	1.216093e+07	0.342854	0.332262	0.499175	0.692060	0.417779	12.157324	0.999992	0.200499	0.2812
std	3.517694e+04	0.474665	0.471026	0.500001	0.461644	0.728802	12.056079	0.002865	0.400375	0.4495
min	1.210000e+07	0.000000	0.000000	0.000000	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	5.000000	1.000000	0.000000	0.0000
50%	1.216093e+07	0.000000	0.000000	0.000000	1.000000	0.000000	9.000000	1.000000	0.000000	0.0000
75%	1.219139e+07	1.000000	1.000000	1.000000	1.000000	1.000000	15.000000	1.000000	0.000000	1.0000
max	1.222186e+07	1.000000	1.000000	1.000000	1.000000	19.000000	69.000000	1.000000	1.000000	1.0000

Dimensionality Reduction

Data reduction is typically conducted in two directions, i.e., row-wise for data sample reduction and column-wise for data variable reduction (geeksforgeeks, 2022).

Here, column-wise for data variable reduction is done. The irrelevant columns for the model are dropped and the columns which have greater than 30% of rows as NULL is dropped as this could hinder the accuracy of the model.

Then columns like the client's id a unique attribute that does not have predictive power and unambiguous columns like 'scores1' are removed from the data set.



```
#Dropping the unwanted data attributes which is not useful to build the model
processedDataset = originalDataset.copy()

processedDataset.drop(['ID','Accompany_Client', 'Client_Housing_Type', 'Client_Housing_Type', 'Population_Region_Relative','ID_Days', 'Own_House_Age','Mobile_Tag', 'Homephone_Tag', '

processedDataset.head()
```

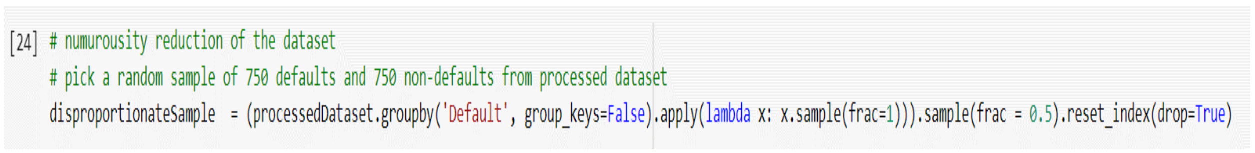
```
#checking the columns with null values more than 30% of data and removing those columns form the dataset
#since high null values might lead the model to work less accurate

result = 0

for x in processedDataset:
        result = processedDataset[x].isnull().sum()
        if (result/len(processedDataset.index)) > 0.3 :
            print(processedDataset[x])
            del processedDataset[x]
        result=0
```

```
[ ]  processedDataset.head()
```

	Client_Income	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Credit_Amount	Loan_Annuity	Client_Income_Type	Client_Education	Client_Marital_Status	Client_Gender	L
0	6750	0.0	0.0	1.0	0.0	0.0	61190.55	3416.85	Commercial	Secondary	M	Male	
1	20250	1.0	0.0	1.0	NaN	0.0	15282	1826.55	Service	Graduation	M	Male	
2	18000	0.0	0.0	1.0	0.0	1.0	59527.35	2788.2	Service	Graduation dropout	W	Male	
3	15750	0.0	0.0	1.0	1.0	0.0	53870.4	2295.45	Retired	Secondary	M	Male	
4	33750	1.0	0.0	1.0	0.0	2.0	133988.4	3547.35	Commercial	Secondary	M	Female	

Numerosity Reduction

Here, a function is written to reduce the number of rows. As the distribution of data is in different proportions, Stratified sampling is used and only 50% of the original rows are used proportionately.



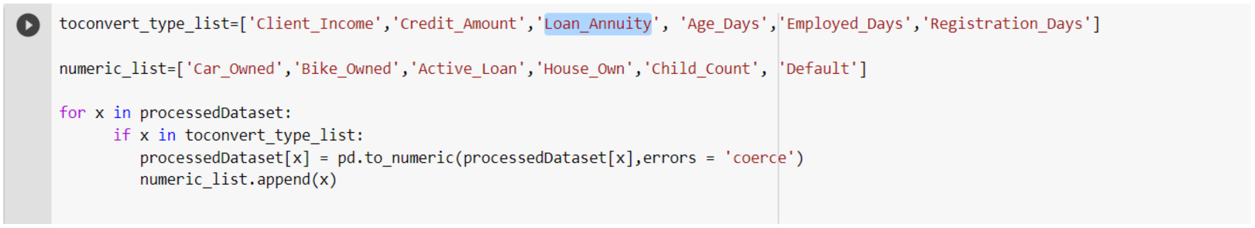
```
[24] # numurousity reduction of the dataset
     # pick a random sample of 750 defaults and 750 non-defaults from processed dataset
     disproportionateSample  = (processedDataset.groupby('Default', group_keys=False).apply(lambda x: x.sample(frac=1))).sample(frac = 0.5).reset_index(drop=True)
```

Data Transformation and Normalization

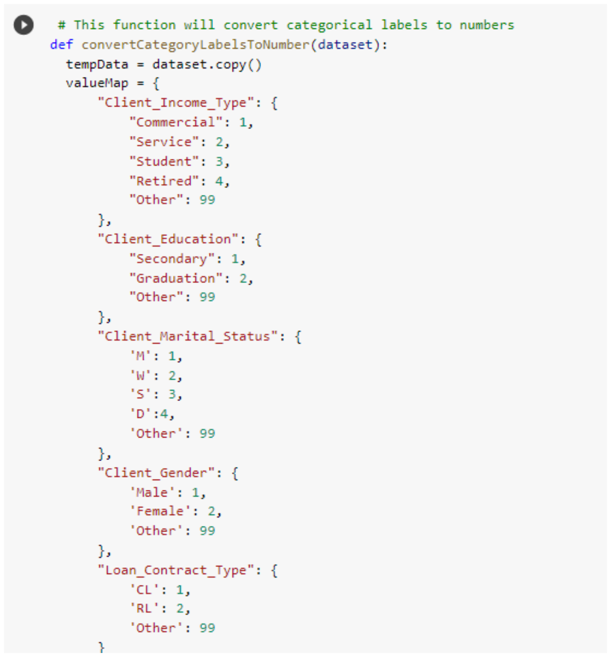
Data transformation can be applied to transform categorical variables into numerical ones to facilitate the development of prediction models (galaktika-soft, n.d.).

Well-known techniques for converting to numerical type include label encoding, one hot encoding, and Dummy coding. Dummy coding and one hot encoding both change one variable into numerous variables, as opposed to label encoding, which turns each unique value of the categorical column into a number. Label encoding was employed in this situation.

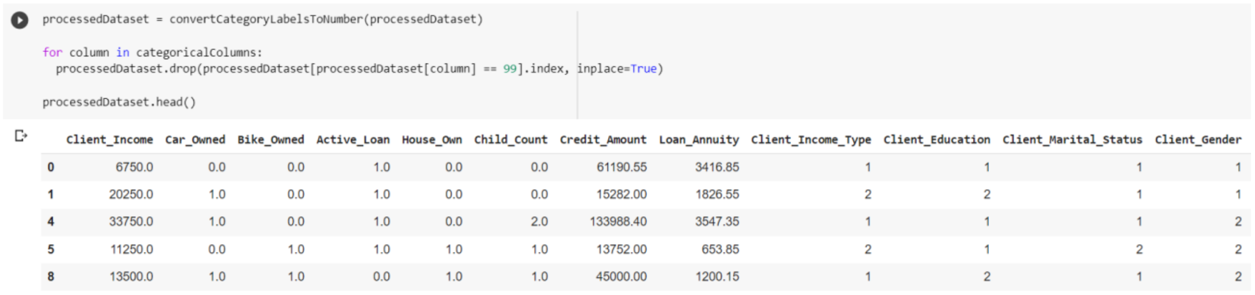
The columns like 'Client_Income', 'Credit_Amount', and 'Loan_Annuity' are converted to numerical types which were originally categorical type columns in the original dataset.



```
toconvert_type_list=['Client_Income','Credit_Amount','Loan_Annuity', 'Age_Days','Employed_Days','Registration_Days']

numeric_list=['Car_Owned','Bike_Owned','Active_Loan','House_Own','Child_Count', 'Default']

for x in processedDataset:
      if x in toconvert_type_list:
         processedDataset[x] = pd.to_numeric(processedDataset[x],errors = 'coerce')
         numeric_list.append(x)
```



```
 # This function will convert categorical labels to numbers
def convertCategoryLabelsToNumber(dataset):
  tempData = dataset.copy()
  valueMap = {
      "Client_Income_Type": {
          "Commercial": 1,
          "Service": 2,
          "Student": 3,
          "Retired": 4,
          "Other": 99
      },
      "Client_Education": {
          "Secondary": 1,
          "Graduation": 2,
          "Other": 99
      },
      "Client_Marital_Status": {
          'M': 1,
          'W': 2,
          'S': 3,
          'D':4,
          'Other': 99
      },
      "Client_Gender": {
          'Male': 1,
          'Female': 2,
          'Other': 99
      },
      "Loan_Contract_Type": {
          'CL': 1,
          'RL': 2,
          'Other': 99
      }
```



```
processedDataset = convertCategoryLabelsToNumber(processedDataset)

for column in categoricalColumns:
  processedDataset.drop(processedDataset[processedDataset[column] == 99].index, inplace=True)

processedDataset.head()
```

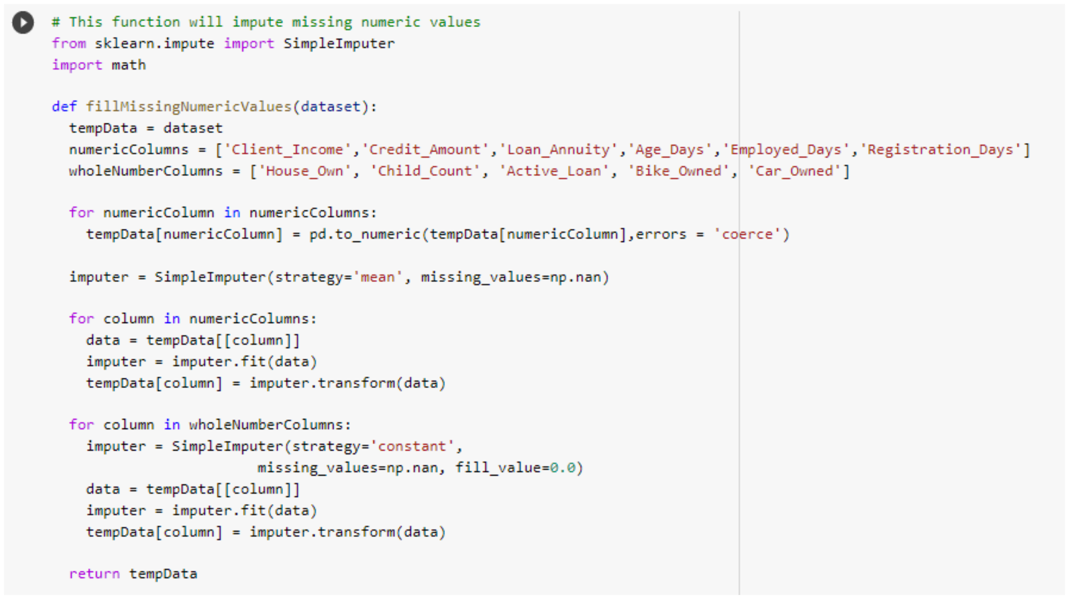
	Client_Income	Car_Owned	Bike_Owned	Active_Loan	House_Own	Child_Count	Credit_Amount	Loan_Annuity	Client_Income_Type	Client_Education	Client_Marital_Status	Client_Gender
0	6750.0	0.0	0.0	1.0	0.0	0.0	61190.55	3416.85	1	1	1	1
1	20250.0	1.0	0.0	1.0	0.0	0.0	15282.00	1826.55	2	2	1	1
4	33750.0	1.0	0.0	1.0	0.0	2.0	133988.40	3547.35	1	1	1	2
5	11250.0	0.0	1.0	1.0	1.0	1.0	13752.00	653.85	2	1	2	2
8	13500.0	1.0	1.0	0.0	1.0	1.0	45000.00	1200.15	1	2	1	2

Null Value Handling

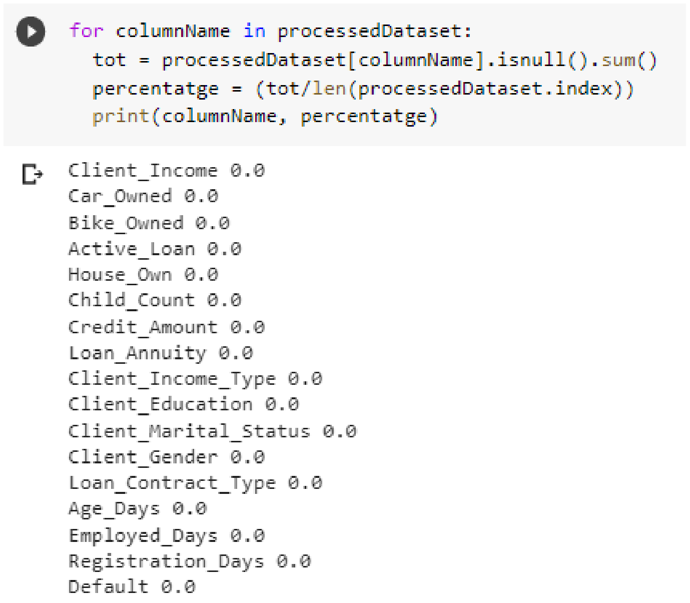
There are two general ways to handle missing values in building operational data. The first is to simply discard data samples with missing values as most data mining algorithms cannot handle data with missing values (Sucky, n.d.). Such a method is only applicable when the proportion of missing values is insignificant. The second is to apply missing value imputation methods to replace missing data with inference values.

Thus, in this step columns containing more than 30% of null values, such as "Own house age" and "Social Circle Default," are not included in the data collection.

To impute missing values from other columns, the mean technique is utilized.



```python
# This function will impute missing numeric values
from sklearn.impute import SimpleImputer
import math

def fillMissingNumericValues(dataset):
  tempData = dataset
  numericColumns = ['Client_Income','Credit_Amount','Loan_Annuity','Age_Days','Employed_Days','Registration_Days']
  wholeNumberColumns = ['House_Own', 'Child_Count', 'Active_Loan', 'Bike_Owned', 'Car_Owned']

  for numericColumn in numericColumns:
    tempData[numericColumn] = pd.to_numeric(tempData[numericColumn],errors = 'coerce')

  imputer = SimpleImputer(strategy='mean', missing_values=np.nan)

  for column in numericColumns:
    data = tempData[[column]]
    imputer = imputer.fit(data)
    tempData[column] = imputer.transform(data)

  for column in wholeNumberColumns:
    imputer = SimpleImputer(strategy='constant',
                  missing_values=np.nan, fill_value=0.0)
    data = tempData[[column]]
    imputer = imputer.fit(data)
    tempData[column] = imputer.transform(data)

  return tempData
```



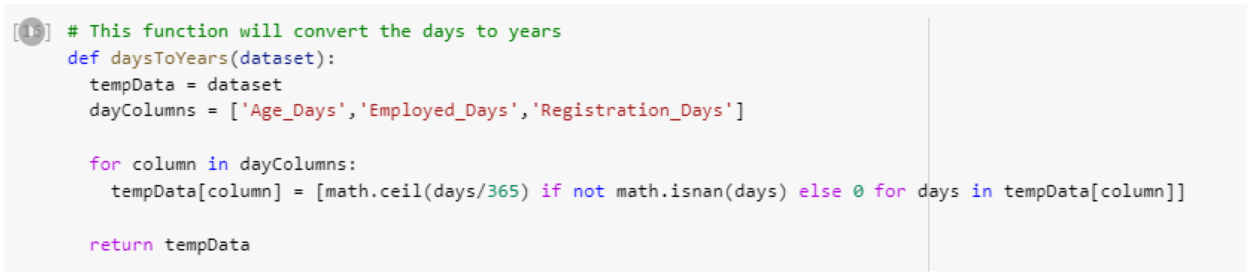
```python
for columnName in processedDataset:
  tot = processedDataset[columnName].isnull().sum()
  percentatge = (tot/len(processedDataset.index))
  print(columnName, percentatge)
```

```
Client_Income 0.0
Car_Owned 0.0
Bike_Owned 0.0
Active_Loan 0.0
House_Own 0.0
Child_Count 0.0
Credit_Amount 0.0
Loan_Annuity 0.0
Client_Income_Type 0.0
Client_Education 0.0
Client_Marital_Status 0.0
Client_Gender 0.0
Loan_Contract_Type 0.0
Age_Days 0.0
Employed_Days 0.0
Registration_Days 0.0
Default 0.0
```

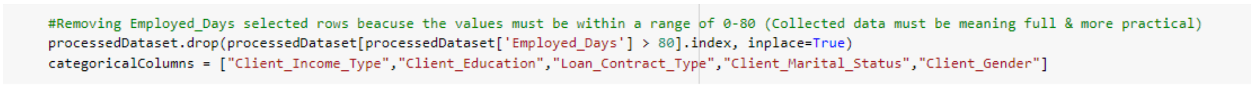
Converting The Age from Days to Years and Removing Incorrect Age Rows

It was observed that the dataset records the data in days rather than in years, hence, it was decided to convert this column data into years.

After the conversion of the data, it was observed that there are ages that exceed the maximum human life span. Hence, they were considered incorrect data. An average lifespan was considered as 80 years and we dropped the rows that record the age greater than 80.



```
[15] # This function will convert the days to years
     def daysToYears(dataset):
       tempData = dataset
       dayColumns = ['Age_Days','Employed_Days','Registration_Days']

       for column in dayColumns:
         tempData[column] = [math.ceil(days/365) if not math.isnan(days) else 0 for days in tempData[column]]

       return tempData
```



```
#Removing Employed_Days selected rows beacuse the values must be within a range of 0-80 (Collected data must be meaning full & more practical)
processedDataset.drop(processedDataset[processedDataset['Employed_Days'] > 80].index, inplace=True)
categoricalColumns = ["Client_Income_Type","Client_Education","Loan_Contract_Type","Client_Marital_Status","Client_Gender"]
```

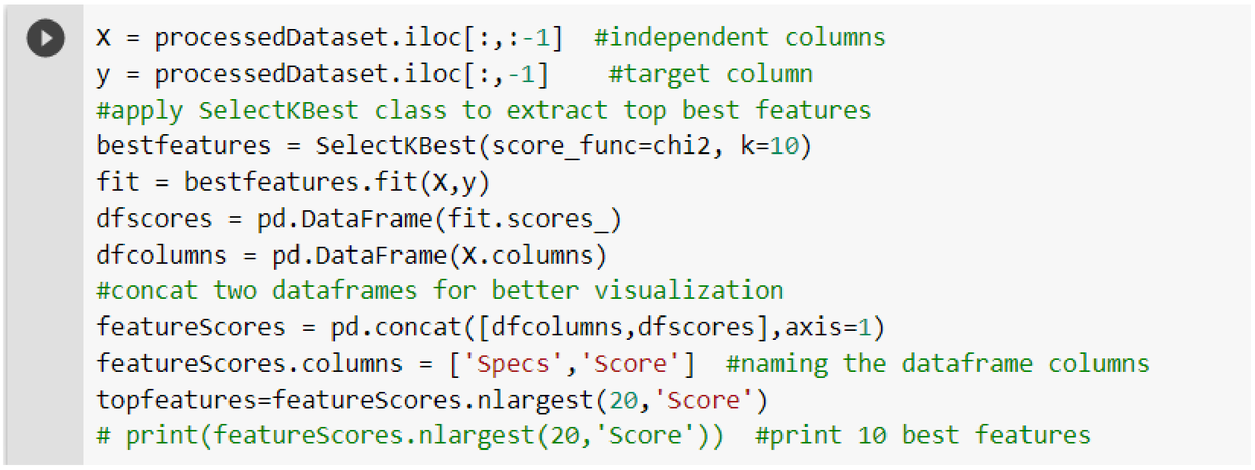
Feature Selection

There are still a lot of columns in the data set after carrying out the aforementioned procedures and preparing them. The following step, feature selection, is the most noticeable one. An alternative term for this is choosing variables or characteristics. Feature selection is the process of selecting the fewest possible useful attribute sets. The following techniques can be used to choose features:

- Information Gain filtering
- Chi-square Test
- Fisher's Score
- Correlation Coefficient score
- K-highest scores

The Correlation Coefficient test's k-highest scores were used to pick the features in this case. The correlation coefficient is a metric used to evaluate a linear relationship between two variables. High correlations with the goal variable and no correlations between the variables themselves make for good model variables. One variable should be utilized for prediction when two are highly correlated because the other doesn't give the model any additional weight. The Pearson correlation has been used for this (Brownlee, 2020).

Even when the cutoff value is set to an absolute 0.5, the correlation coefficient scores can only be used to exclude three attributes. The k-highest scores strategy is used to obtain the best attributes.



```
X = processedDataset.iloc[:,:-1]  #independent columns
y = processedDataset.iloc[:,-1]    #target column
#apply SelectKBest class to extract top best features
bestfeatures = SelectKBest(score_func=chi2, k=10)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']  #naming the dataframe columns
topfeatures=featureScores.nlargest(20,'Score')
# print(featureScores.nlargest(20,'Score'))  #print 10 best features
```

From the above, the best features were selected to proceed with the model building.

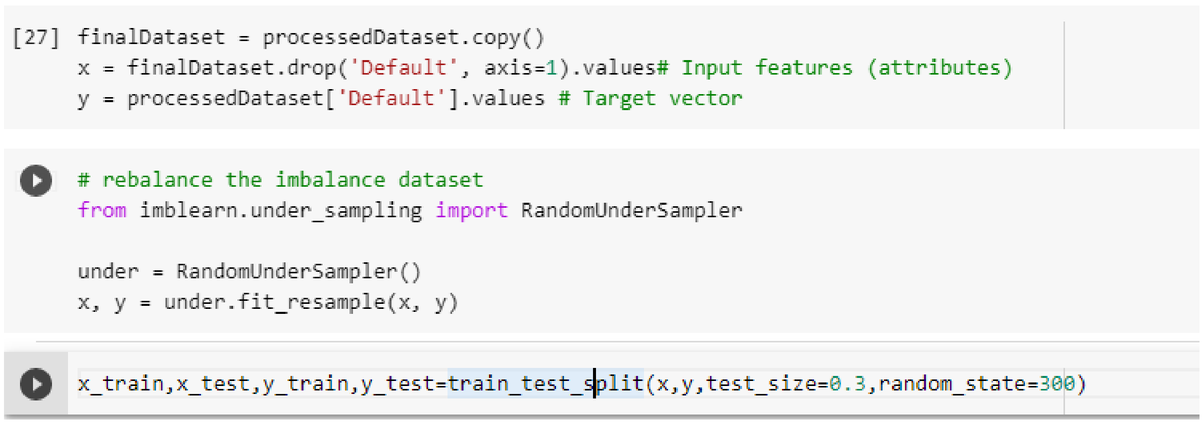
After selecting the top best features, the data set will be divided into the train (x -without target class) and test (y-only target class) vectors.

Solving the Issue of an Imbalanced Data Set

As mentioned, and displayed in the data visualization step, the data set is highly imbalanced.

The target was about 90% skewed towards the Not Default type, this was a major issue as this affects the overall accuracy of the model.

Therefore, to rebalance the data set under-sampling technique has been used here



```
[27] finalDataset = processedDataset.copy()
     x = finalDataset.drop('Default', axis=1).values# Input features (attributes)
     y = processedDataset['Default'].values # Target vector
```

```
# rebalance the imbalance dataset
from imblearn.under_sampling import RandomUnderSampler

under = RandomUnderSampler()
x, y = under.fit_resample(x, y)
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=300)
```

Proposed Data Mining Solutions

Data mining is considered the process of extracting useful information from a vast amount of data. It's used to discover new, accurate, and useful patterns in the data, looking for meaning and relevant information for the organization or individual who needs it. It's a tool used by humans.

On the other hand, machine learning is the process of discovering algorithms that have improved courtesy of experience derived from data. It's the design, study, and development of algorithms that permit machines to learn without human intervention. It's a tool to make machines smarter, eliminating the human element (but not eliminating humans themselves; that would be wrong).

Classification is a technique to categorize data into a given number of classes. This problem needs to be categorized as the 'Default' and 'Not Default' classes. The used classification models are,

1. Support Vector Machine
2. Logistic Regression
3. Decision Tree
4. Random Forest
5. Gaussian Naive Bayes

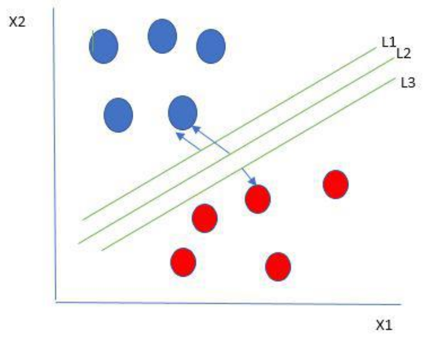
Each model was built with hyperparameter tuning which is a technique used to find correct hyperparameters for machine learning or deep learning models. The goal of hyperparameter tuning is to get the maximum performance out of models. Grid Search hyperparameter tuning method has been used here.
For each model testing and training accuracies and other metrics like precision, recall, and F1 scores are derived.

Support Vector Machine Algorithm

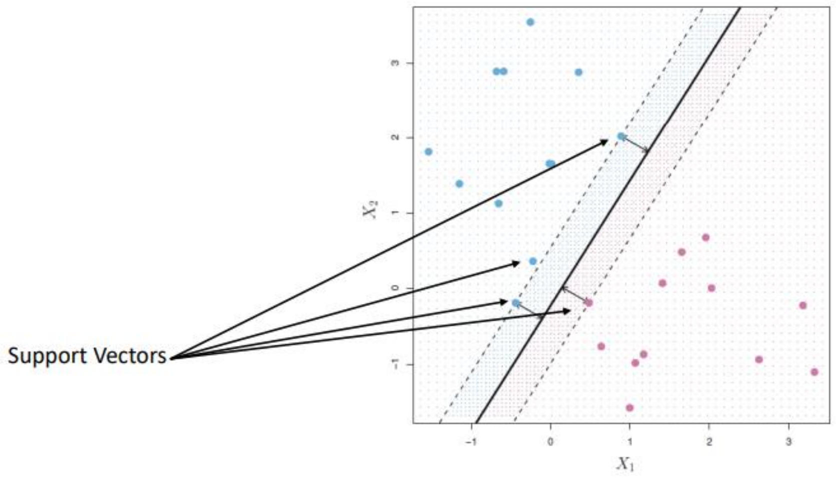
Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it's best suited for classification. The objective of the SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three (Ray, 2017).

So how do we choose the best line or in general the best hyperplane that segregates our data points?

- One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.

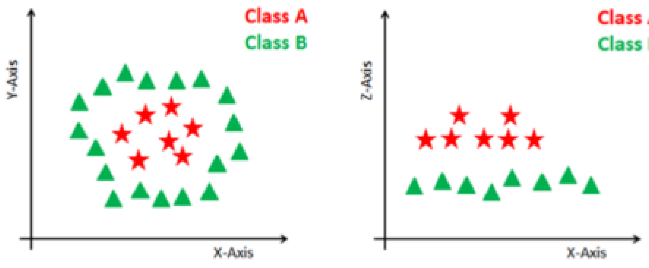


- Points on the margin or close to the margin are called Support Vectors.



So, we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the maximum-margin hyperplane/hard margin. So, from the above figure, we choose L2.

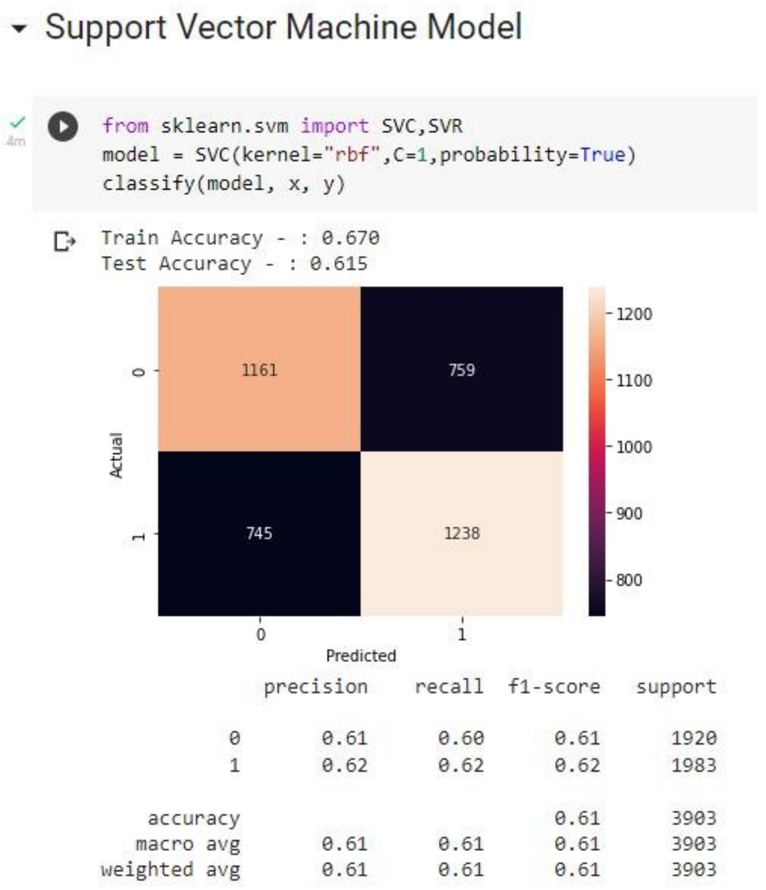
Some problems can't be solved using linear hyperplane, as shown in the figure below. In such a situation, SVM uses a kernel trick to transform the input space to a higher dimensional space as shown on the right.



SVM Kernel

The SVM kernel is a function that takes low-dimensional input space and transforms it into higher-dimensional space, ie it converts nonseparable problems to separable problems. It is mostly useful in non-linear separation problems. Simply put the kernel, does some extremely complex data transformations and then finds out the process to separate the data based on the labels or outputs defined (Ray, 2017).

- Some popular kernels are,
    1. Linear Kernel
    2. Polynomial Kernel
    3. Radial Basis Function Kernel
- We have used the ‘rbf’ kernel for hyperparameter optimization in this use case.



Support Vector Machine Model

```
from sklearn.svm import SVC,SVR
model = SVC(kernel="rbf",C=1,probability=True)
classify(model, x, y)
```

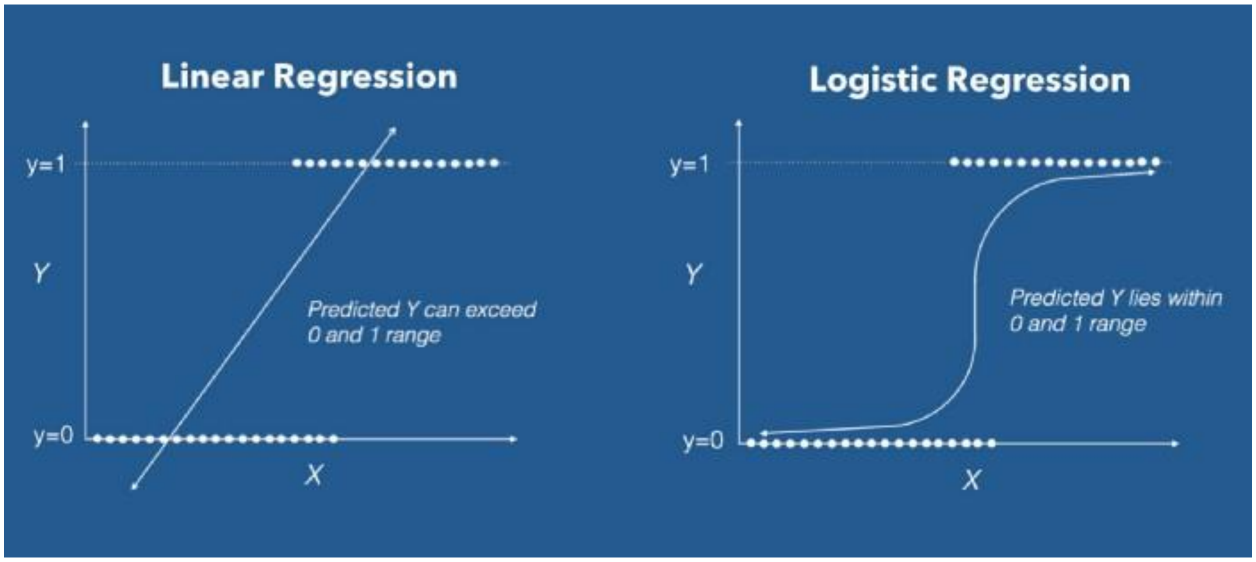
```
Train Accuracy - : 0.670
Test Accuracy - : 0.615
```

```
              precision    recall  f1-score   support

           0       0.61      0.60      0.61      1920
           1       0.62      0.62      0.62      1983

    accuracy                           0.61      3903
   macro avg       0.61      0.61      0.61      3903
weighted avg       0.61      0.61      0.61      3903
```

Logistic Regression

Logistic regression is a Machine Learning classification algorithm that is used to predict the probability of certain classes based on some dependent variables. In short, the logistic regression model computes a sum of the input features and calculates the logistic of the result.

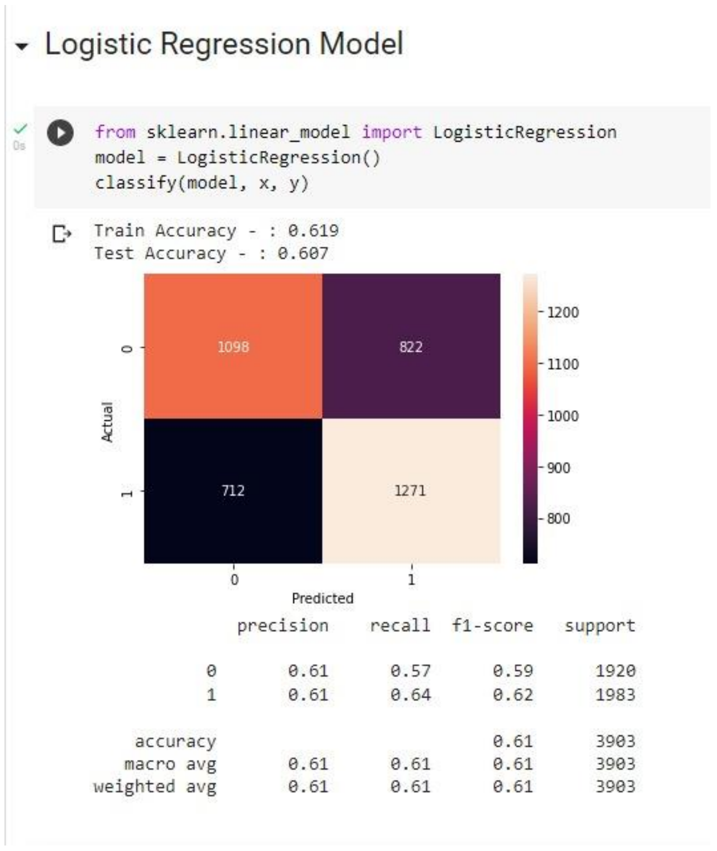
The output of logistic regression is always between (0, and 1), which is suitable for a binary classification task. The higher the value, the higher the probability that the current sample is classified as class=1, and vice versa (Swaminathan, 2018).


Linear Regression
y=1
Y
Predicted Y can exceed 0 and 1 range
y=0
X
Logistic Regression
y=1
Y
Predicted Y lies within 0 and 1 range
y=0
X

We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the '**Sigmoid function**' or also known as the 'logistic function' instead of a linear function.

The equation of the sigmoid function is:

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$



Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model, x, y)
```

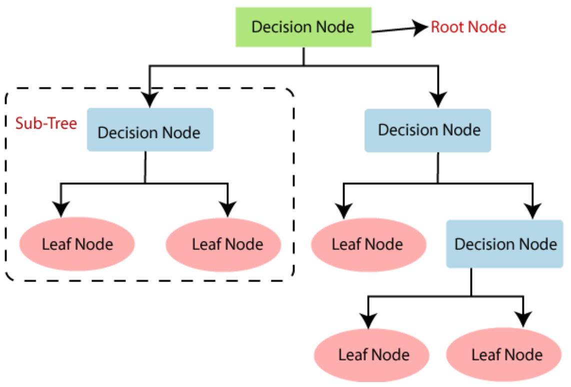
```
Train Accuracy - : 0.619
Test Accuracy - : 0.607
```

```
              precision    recall  f1-score   support

           0       0.61      0.57      0.59      1920
           1       0.61      0.64      0.62      1983

    accuracy                           0.61      3903
   macro avg       0.61      0.61      0.61      3903
weighted avg       0.61      0.61      0.61      3903
```

Decision Tree

Decision Tree is one of the most popular classification models available.

A decision tree is a non-parametric supervised learning method for classification and regression. To create a model that predicts the value of a target variable, the goal is to learn simple decision rules generated from the data attributes (scikit-learn, scikit-learn, n.d.).



The main challenge of the decision tree is to find the best splitting. This process is known as attribute selection. The most popular attribute selection option in this situation is information gain.

Information gain is assessed using the Gini index and entropy. Gini and entropy are both measures of a node's impurity. When a node has one class, it is said to be pure; when it has numerous classes, it is said to be impure.

$$Gini(t) = 1 - \sum_{i=1}^{j} P(i|t)^2$$

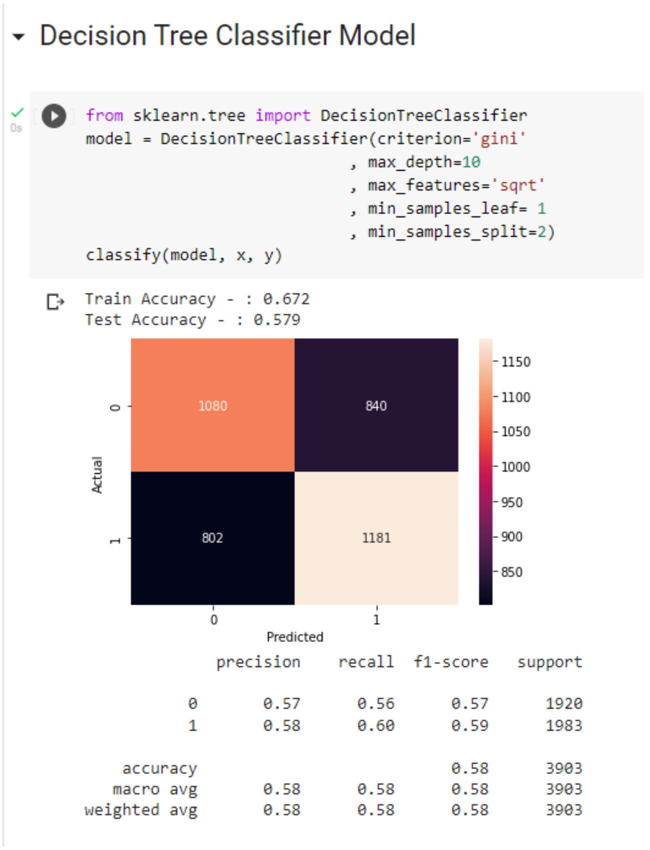
$$E = -\sum_{i=1}^{N} p_i log_2 p_i$$

Where,

The *j* represents the number of classes in the label, and

The *P* represents the ratio of class at the ith node.

Below is the model and accuracy that was returned with the model when the Decision Tree model was built, trained, and tested against the dataset.



Decision Tree Classifier Model

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion='gini'
                               , max_depth=10
                               , max_features='sqrt'
                               , min_samples_leaf= 1
                               , min_samples_split=2)
classify(model, x, y)
```

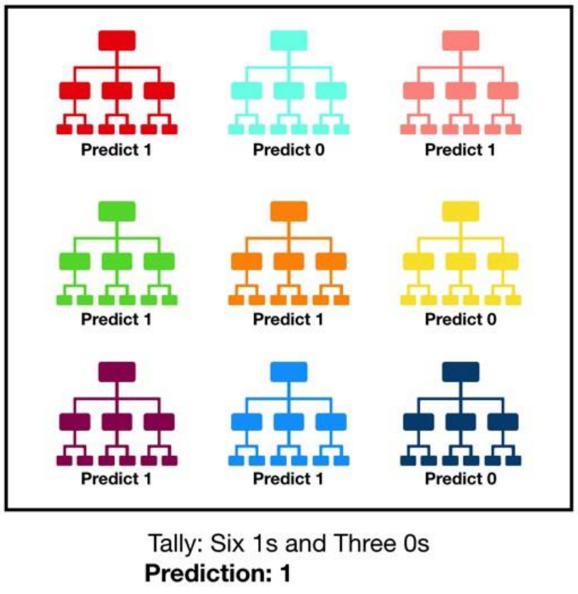
```
Train Accuracy - : 0.672
Test Accuracy - : 0.579
```

```
              precision    recall  f1-score   support

           0       0.57      0.56      0.57      1920
           1       0.58      0.60      0.59      1983

    accuracy                           0.58      3903
   macro avg       0.58      0.58      0.58      3903
weighted avg       0.58      0.58      0.58      3903
```

Random Forest Classifier

A popular algorithm for classification and regression issues is the supervised machine learning technique known as random forest. It creates decision trees from various samples, relying on their majority for categorization and average for regression. One of the key characteristics of the Random Forest Algorithm is its ability to handle data sets with both continuous variables, as in regression, and categorical variables, as in classification. For categorization issues, it performs better (scikit-learn, scikit-learn, n.d.).

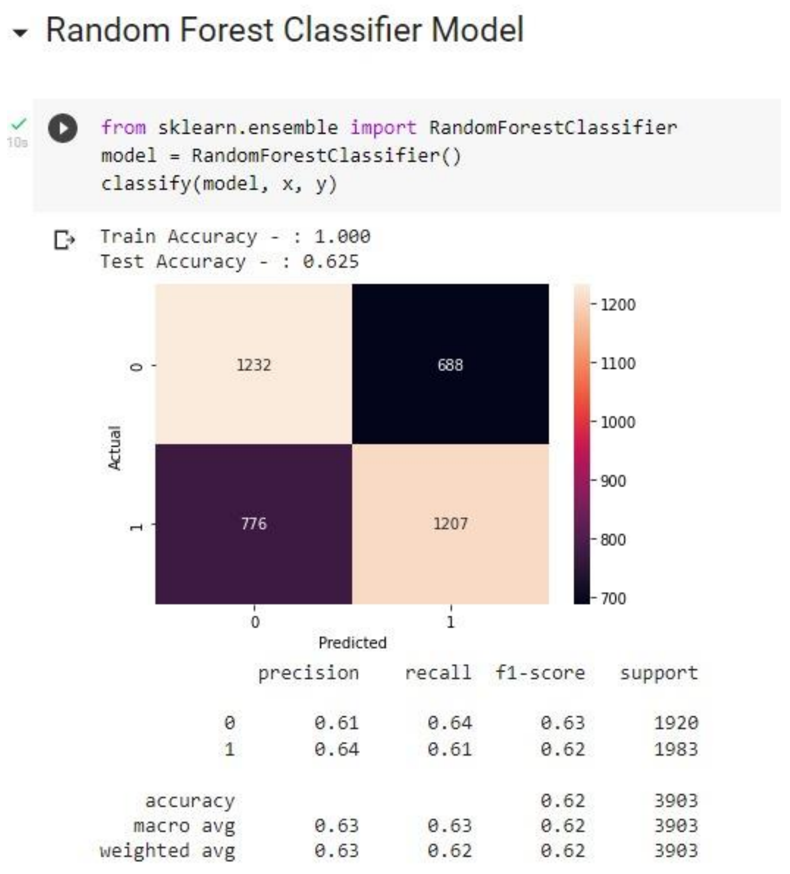
The below picture will provide an understanding of the basic structure of the Random Forest Classifier.



Here, several Decision trees are considered as samples and the predictive result given by each decision tree is considered. Then the predictive results which get the highest number of decision trees are selected. According to the above example, out of the 9 decision trees considered, 6 of them predict as 1 and the other 3 decision trees predict 0. Since 1 is predicted by most of the decision trees, the final prediction is 1.

The below image will provide an instance where we used the Random Forest classifier. It also includes the training and test accuracy of the said classifier in our chosen dataset.

- **NOTE**: Random Forest Classifier model was fit into the dataset chosen, as it was the model which gave the highest accuracy.



Random Forest Classifier Model

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
classify(model, x, y)
```

```
Train Accuracy - : 1.000
Test Accuracy - : 0.625
```

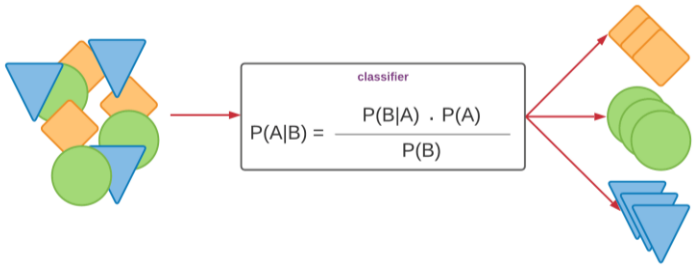
```
              precision    recall  f1-score   support

           0       0.61      0.64      0.63      1920
           1       0.64      0.61      0.62      1983

    accuracy                           0.62      3903
   macro avg       0.63      0.63      0.62      3903
weighted avg       0.63      0.62      0.62      3903
```

Naive Bayes Classifier

Naive Bayes classifiers are a group of classification algorithms based on the Bayes theorem. Instead of being a single technique, it is a family of algorithms, and they are all predicated on the notion that every pair of qualities being classified is independent of the other.

When dealing with continuous data, it's common to consider the assumption that each class's continuous values are distributed according to a normal (or Gaussian) distribution

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

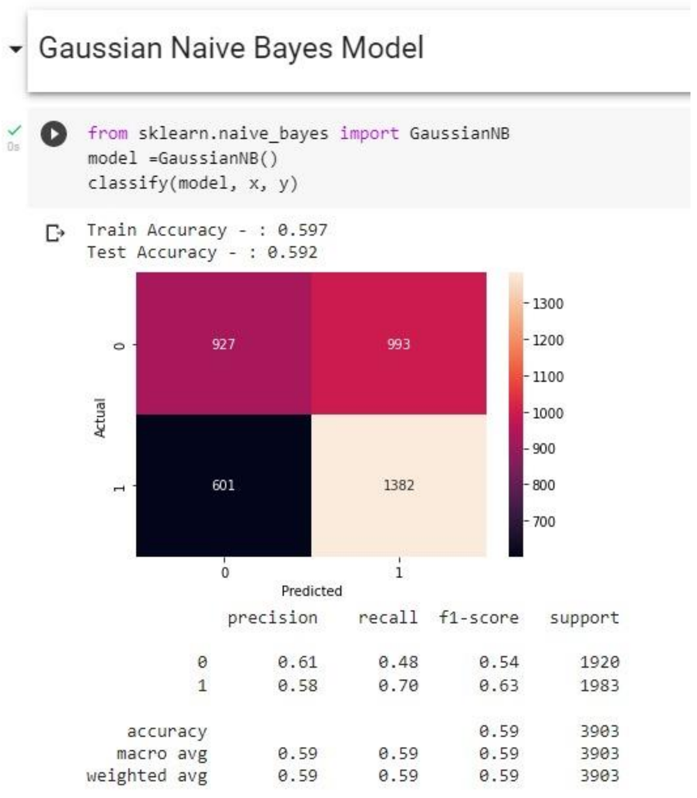


The Gaussian Naive Bayes model allows continuous-valued features and assumes that each of these follows a Gaussian (normal) distribution (Gandhi, 2018).

To build a model we can assume that the data is characterized by a Gaussian distribution with no covariance (independent dimensions) between dimensions is one method for building a straightforward model and finding the mean and standard deviation of the points inside each label, which is all that is required to establish such a distribution, will allow this model to be fit.

We can see that the Gaussian Naïve Bayes has a slightly different approach to building a model when considering other models and this approach might be more efficient than the other approaches.

Below is the model and accuracy that was returned with the model when the Naive Bayes classification model was built, trained, and tested against the dataset.


Gaussian Naive Bayes Model

```
from sklearn.naive_bayes import GaussianNB
model =GaussianNB()
classify(model, x, y)
```

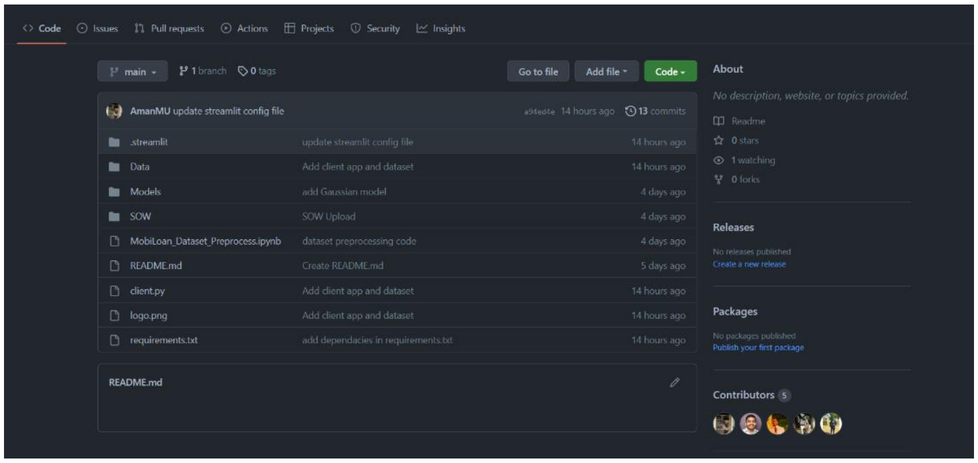
```
Train Accuracy - : 0.597
Test Accuracy - : 0.592
```

```
              precision    recall  f1-score   support

           0       0.61      0.48      0.54      1920
           1       0.58      0.70      0.63      1983

    accuracy                           0.59      3903
   macro avg       0.59      0.59      0.59      3903
weighted avg       0.59      0.59      0.59      3903
```

Deployment of the Implemented Model

Model deployment is the process of putting machine learning models into production. This makes the model's predictions available to users, developers, or systems, so they can make business decisions based on data, and interact with their applications.

In this case, the project was deployed by pushing the model and the necessary metadata into a Git repository and hosting the application via the Streamlit cloud platform.



AmanMU update streamlit config file a94ed4e 14 hours ago 13 commits

.streamlit	update streamlit config file	14 hours ago
Data	Add client app and dataset	14 hours ago
Models	add Gaussian model	4 days ago
SOW	SOW Upload	4 days ago
MobiLoan_Dataset_Preprocess.ipynb	dataset preprocessing code	4 days ago
README.md	Create README.md	5 days ago
client.py	Add client app and dataset	14 hours ago
logo.png	Add client app and dataset	14 hours ago
requirements.txt	add dependacies in requirements.txt	14 hours ago

README.md

About

No description, website, or topics provided.

Readme
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 5

User Interfaces

The user interface (UI) is the point of human-computer interaction and communication in a device.

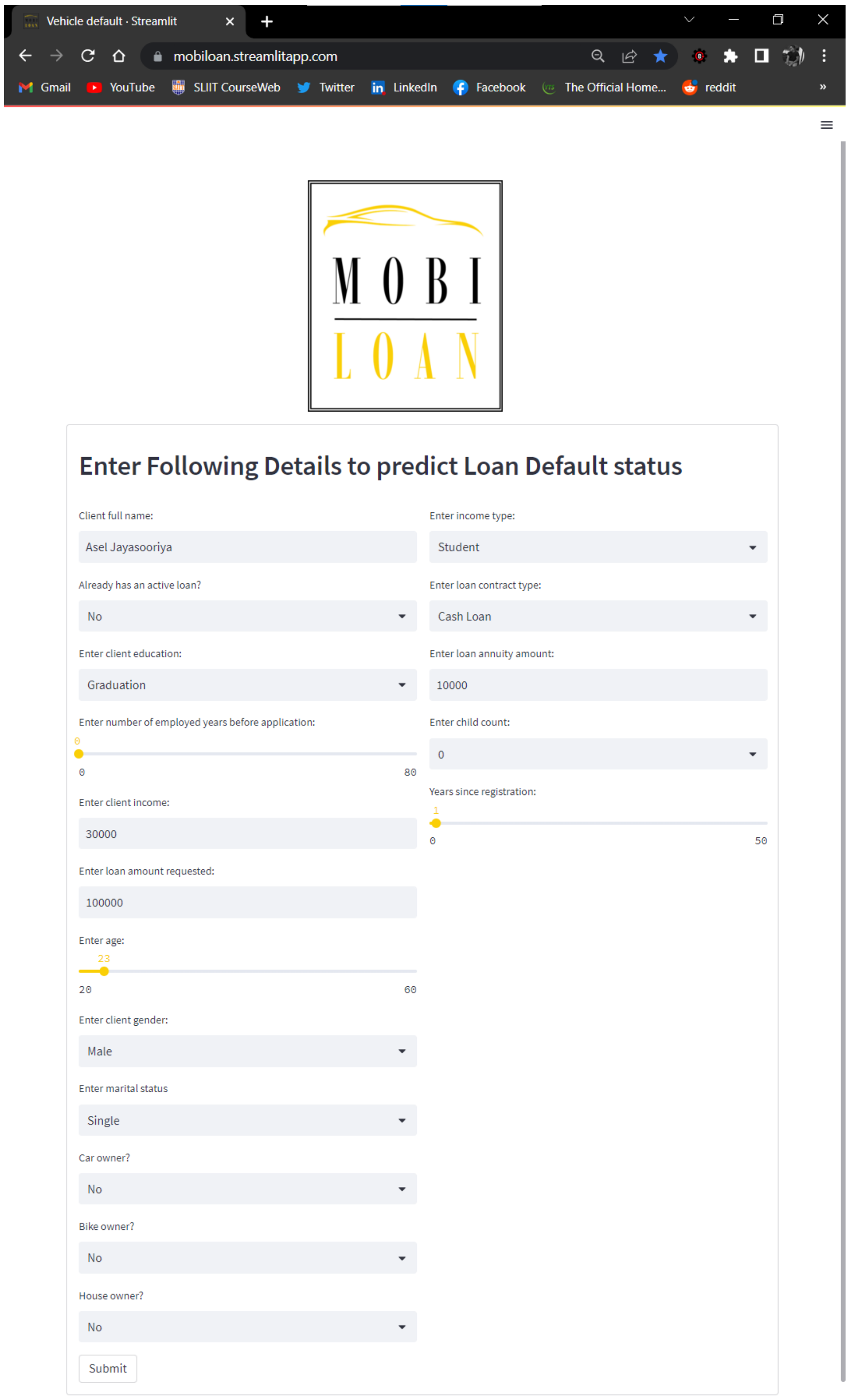
Thus, a very interactive, user-friendly user interface was built for the application. In the user interface build we have chosen the theme of Yellow and White which was continued as the application's theme throughout this project.

The user interface was built with a minimalist theme which requires near to no expertise in IT, hence training the staff would not be an issue for our clients.

The interface obtains the necessary information about the loan request via input fields and feeds them into the trained model. Then a text notification would be displayed which would inform the client prediction and whether to accept or reject the loan request.

All the necessary field validations have been done and testing of the interface has been done for user-friendliness and user experience via a third party.

- The following is a screen capture that represents the UI form that receives the necessary data for the prediction.

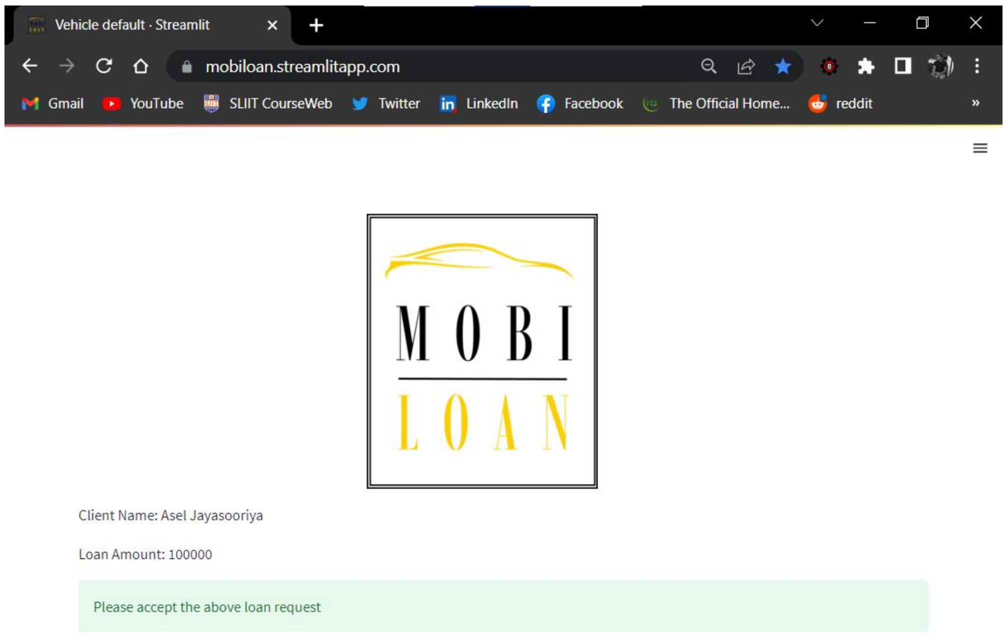


Enter Following Details to predict Loan Default status

Client full name:	Asel Jayasooriya
Already has an active loan?	No
Enter client education:	Graduation
Enter number of employed years before application:	0 (0–80)
Enter client income:	30000
Enter loan amount requested:	100000
Enter age:	23 (20–60)
Enter client gender:	Male
Enter marital status	Single
Car owner?	No
Bike owner?	No
House owner?	No
Enter income type:	Student
Enter loan contract type:	Cash Loan
Enter loan annuity amount:	10000
Enter child count:	0
Years since registration:	1 (0–50)

Submit

- The following is a screen capture that represents the acceptance of a loan request by the model.

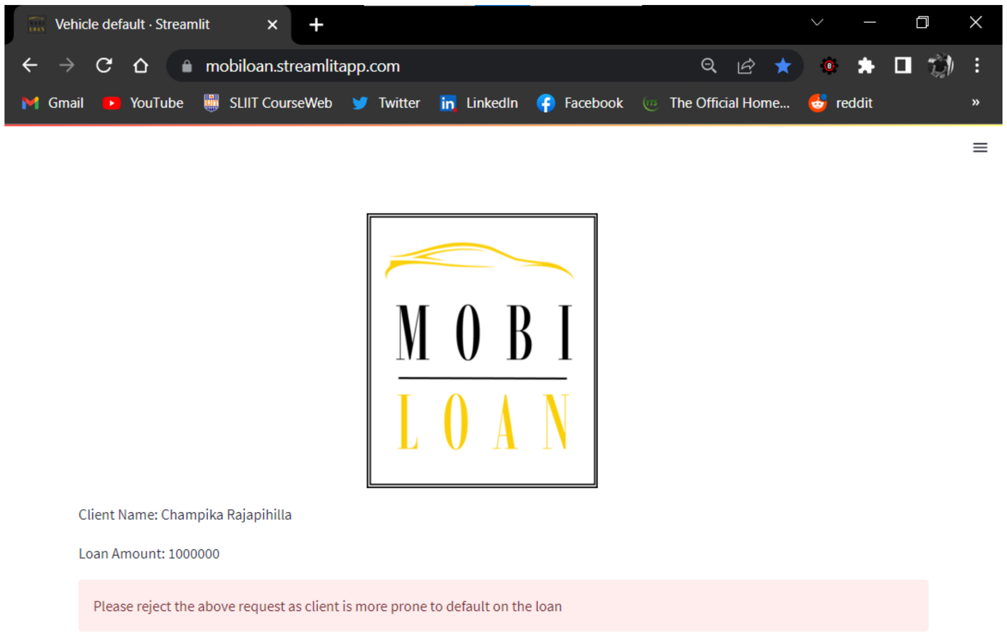


Client Name: Asel Jayasooriya

Loan Amount: 100000

Please accept the above loan request

- The following is a screen capture that represents the rejection of a loan request by the model.



Client Name: Champika Rajapihilla

Loan Amount: 1000000

Please reject the above request as client is more prone to default on the loan

Conclusion

This project was implemented for a Non-Banking Financial Institution that was not monitored by any international or national monetary authorities.

Through this project, we have implemented a predictive model which will give an understanding of whether the loan borrowers will be able to pay back the loan or not. For this task, we chose a dataset that was related to the ‘automobile loan default’ topic.

The most tedious task was to preprocess the data so that it can be fit into a predictive model. Since there were a lot of data with faults, we had to handle some missing values, handle inconsistent data, normalization of data, reduction of data, null value handling, and also had to convert some categorical data into numerical variables as well. Even after doing the said preprocessing, the accuracy of the models was relatively low. So, we had to further enhance the dataset by doing more preprocessing and we used various techniques to balance the dataset.

We picked five classification technique models and implemented and observed the best model where it gave the highest accuracy for the chosen dataset. After doing so, our team observed that the Random Forest Classifier model was the model which gave the best test accuracy for the preprocessed dataset.

After building, training and implementing, and deploying the model with a proper user interface, a user can utilize the model to get predictions on loan requests by entering the necessary values. Thus, a user can get an understanding of whether the customer to who they are lending the loan, will be able to repay the loan amount or not.

This would help the company to target the favorable customer groups (who repay the loan amount) to get profits. This will also help the company to cut losses and would help the company to be more effective and efficient.

References

Brownlee, J. (2020, August 20). *machinelearningmastery*. Retrieved from https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

galaktika-soft. (n.d.). *galaktika*. Retrieved from https://galaktika-soft.com/blog/data-mining-normalization.html

Gandhi, R. (2018, May 5). *towardsdatascience*. Retrieved from https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

geeksforgeeks. (2022, September 27). *geeksforgeeks*. Retrieved from https://www.geeksforgeeks.org/dimensionality-reduction/

Kumar, A. (2022, April 16). *vitalflux*. Retrieved from https://vitalflux.com/correlation-heatmap-with-seaborn-pandas

Ray, S. (2017, September 13). *analyticsvidhya*. Retrieved from https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/

S, R. A. (n.d.). *simplilearn*. Retrieved from www.simplilearn.com: https://www.simplilearn.com/tutorials/python-tutorial/data-visualization-in-python

scikit-learn. (n.d.). *scikit-learn*. Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/tree.html

scikit-learn. (n.d.). *scikit-learn*. Retrieved from scikit-learn.org: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Sucky, R. N. (n.d.). *Medium*. Retrieved from https://towardsdatascience.com/6-tips-for-dealing-with-null-values-e16d1d1a1b33

Swaminathan, S. (2018, March 15). *towardsdatascience*. Retrieved from https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

Wu, Y. (2022, August 24). *kdnuggets*. Retrieved from https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html

Xenonstack. (2018, December 23). *Medium*. Retrieved from https://medium.com/@xenonstack/data-preparation-process-preprocessing-and-data-wrangling-6c4068f2fcd1