

A PROJECT REPORT ON

“Loan Default Prediction”

*A project report submitted in fulfillment for the Diploma Degree in AI & ML Under
Applied Roots with University of Hyderabad*



Project submitted by

Shreyaskar Nath Tripathi

Under the Guidance of

Mentor: **Aniket Vishnu**

Enr ID : **40AIML434-21/2**

Approved by: Mentor: Aniket Vishnu



University of Hyderabad

Declaration of Authorship

We hereby declare that this thesis titled" Loan Default Prediction" and the work presented by the undersigned candidate, as part of Diploma Degree in AI & ML.

All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Shreyaskar Nath Tripathi

Thesis Title: Loan Default Prediction

CERTIFICATE OF RECOMMENDATION

We hereby recommend that the thesis entitled “**Loan Default Prediction**” prepared under my supervision and guidance by **Shreyaskar Nath Tripathi** be accepted in fulfilment of the requirement for awarding the degree of Diploma in AI & ML Under applied roots with University of Hyderabad. The project, in our opinion, is worthy for its acceptance.

Mentor: **Aniket Vishnu**

Under Applied roots with



University of Hyderabad

ACKNOWLEDGEMENT

Every project big or small is successful largely due to the effort of a number of wonderful people who have always given their valuable advice or lent a helping hand. I sincerely appreciate the inspiration; support and guidance of all those people who have been instrumental in making this project a success. I, Shreyaskar Nath Tripathi student of applied roots, is extremely grateful to mentors for the confidence bestowed in me and entrusting my project entitled “Loan Default Prediction” with special reference.

At this juncture, I express my sincere thanks to Mentor: Aniket Vishnu of applied roots for making the resources available at right time and providing valuable insights leading to the successful completion of our project who even assisted me in completing the project.

Name: Shreyaskar Nath Tripathi

Contents

1. Introduction

2. Literature Review

3. Data Description.

4. Approach, Methodology and Results

Step 1: Requirement Analysis

Step 2: Initial Processing

Step 3: Checking the Imbalance

Step 4: Standardize Value

Step 5: Key metrics (KPI) to optimize

Step 6: Treating Imbalanced data

Step 7: Data Splitting

Step 8: Predictive Modelling

Step 9: Comparing performance of model

Step 10: Choosing a Final Model

5. Discussion of Results.

6. Conclusion.

7. References

8. Deployment

Abstract:

Lending loans can be a difficult work for banks and their management, and so it becomes very important to automate the process of loan lending. Thus, it is very much necessary to predict whether a loan will be returned or not. We are here trying to create a prediction model that will perform loan default prediction by applying machine models, including **logistic regression**, and **random forest** to classify the applicants with and without loan default from a group of predicting variables, and evaluate their performance. Our main aim was to predict whether or not the borrower would default, and we aimed to achieve best possible results.

1. Introduction:

The major financial institutions do not typically service well the needs of lower income families. Lending to low-income or first-time borrowers is challenging when there is little or no credit history. This creates a segment of 'under-banked' individuals struggling to access formal, safe, and fairly priced credit. For established banks, this represents a profit growth opportunity, while for new entrants offering financial services, an attractive underserved market. However, to be successful in this segment, a financial institution must find non-traditional and creative ways to identify and mitigate default risk.

The work that follows explores a dataset from application of bank. The dataset contains details of loan applicants at the time of application plus a target variable indicating whether the applicant ultimately **defaulted (target = 1) or not (target = 0)**. We look for relationships between the characteristics of a loan applicant and their risk of default. For example, are people employed in a certain industry more likely to default, does length of employment, family status or size and location of residence impact default risk? Lastly, through the use of machine learning algorithms, we attempt to predict the likelihood of default for a new loan applicant.

Exploratory analysis uses the Python data manipulation and visualization modules **NumPy**, **pandas** and **matplotlib**. Predictive modelling uses the Python machine learning module **scikit-learn**. The prediction task is a binary classification problem – likely to default: yes, or no? **The models investigated were “Logistic Regression”, “Random ForestAdaBoost”, and “Naive Bayes”**

Problem Statement:

In this project we need to predict whether to sanction or reject the loan. Using the data given, we must ensure that honest clients are not rejected from giving the loans by building a machine learning model.

This is an important problem to solve, as the loan companies grant a loan after an intensive process of verification and validation. However, they still don't have assurance if the applicant is able to repay the loan with no difficulties and they do follow a manual procedure to determine whether a loan can be approved for an applicant based on results, manual procedures were effective but normally time consuming if the data size is high and deciding would take a long time.

In this Project, we'll build a predictive model to predict whether to sanction or reject the loan, which can be used to make decision quick and effective even if the data size is huge.

Business/Real-world impact of solving this problem:

Loans are the core business of banking system, in today's world, obtaining loans from financial institutions has become a very common phenomenon. Every day many people apply for loans, for a variety of purposes. But not all the applicants are reliable, and not everyone can be approved, in this case there goes an intensive process of verification and validation of applicant's information as the success or failure of organization largely depends on the industry's ability to evaluate credit risk (i.e.) deciding whether the borrower is defaulter or non-defaulter, so it is a challenging task for any organization or bank.

2. Literature Review

A Comparison of Machine Learning Algorithms for the Prediction of Past Due Service in Commercial Credit [1]

In this the authors aim to predict commercial entities that will fall behind on their non-financial payment obligations (i.e., payments to suppliers, service providers, etc.). A large real-world dataset from Equifax was used containing 36 datasets each with over 11 million observations and 305 attributes. A binary target variable was created indicating whether the commercial entity was behind on payments or not. Significant

data cleaning, missing value imputation and dimensionality reduction were performed to reduce the dataset to 16 independent variables. Three algorithms were tested: Logistic Regression, Decision Trees and Neural Networks. Accuracy, KS-statistic and ROC AUC were used to evaluate the models. Neural Networks produced the highest accuracy (at 93%) and Decision Trees had the highest KSstatistic (at 0.70). Both Neural Networks and Decision Trees outperformed Logistic Regression when using the ROC AUC measure. However, the actual numerical results were not shown in the paper.

Machine Learning Application in Online Lending Risk Prediction [2]

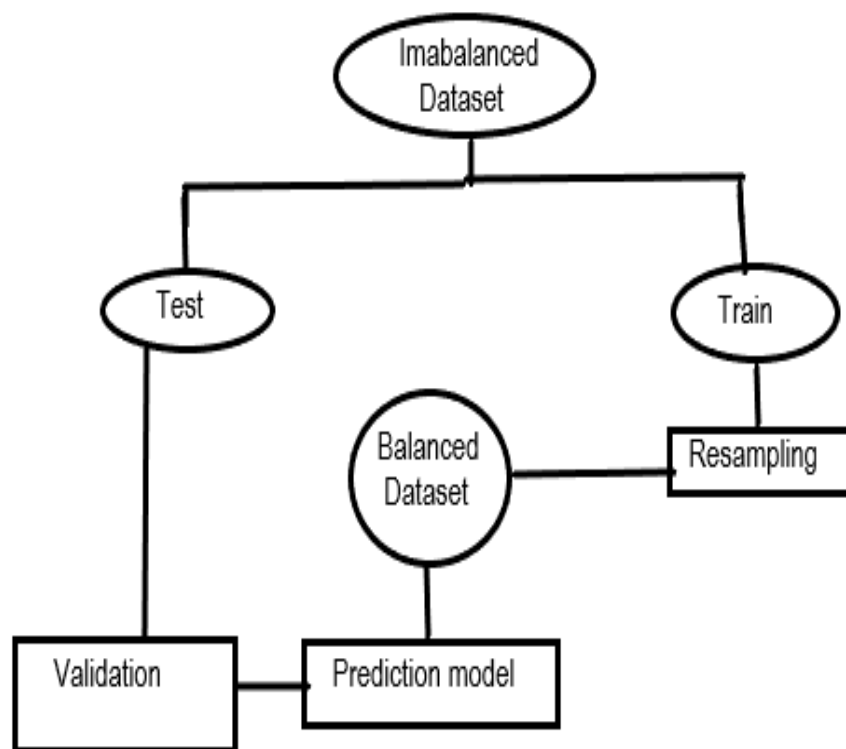
In this the author aims to predict defaulting loan applicants based on a combination of three datasets from an online Chinese lender, online phone records and third-party credit rating agencies. After amalgamation and removing observations with missing data, the dataset had over 8990 attributes and over 301,000 observations. A Random Forest model and an XGBoost model were trained and tested. The parameters of the Random Forest model were optimized using a precision-recall curve methodology, while the parameters of the XGBoost model were optimized using the ROC AUC measure. The prediction results of each model were compared using the KS-statistic. XGBoost outperformed Random Forest with a KS statistic of 0.72 versus 0.65.

Credit Default Mining Using Combined Machine Learning and Heuristic Approach [3]

In this the authors propose a two-step approach for predicting credit card default that combines machine learning with a custom rules-based algorithm (termed the “Heuristic Approach”). Machine learning is used to calculate a probability of default using static historical data, while the custom algorithm is used to calculate a probability of default based on “live” transaction data. The final probability of default is a linear combination of the two previously calculated probabilities. The dataset used appears to be the same **Taiwan credit card dataset** from [1]. Several algorithms were investigated for the machine learning step; the authors chose “**Extremely Random Trees**” (ER Trees) based

on accuracy, precision, recall and Fscore measures. Results of the final Heuristic Approach were compared to pure machine learning using ER Trees and the best results achieved in previous research on the Taiwan dataset. The authors claim the Heuristic Approach achieved accuracy of 93.1% while using ER Trees only had accuracy of 95.8%. Recall was 92.1% for the Heuristic Approach and 85.9% for ER Trees. The authors further claim that both approaches outperformed existing research on the Taiwan dataset for both accuracy and recall measures.

Research Methodology Model Implementation



3. Data Description:

Bank Application data:

'application_data.csv' this data set contains all the information of the client at the time of application. The data is about whether a client has payment difficulties.

In application dataset Out of 122 columns, 30% are categorical columns and 70% columns are numerical.

Source of the dataset: <https://www.kaggle.com/bhargavrk/bank-loan-application-data>

The provided dataset corresponds to all applicant's bank loan application data. The dataset has 307511 observations and 122 features.

Data Acquisition:

Open-Source data - The dataset for this project is retrieved from Kaggle, the home of Data Science.

Source: Kaggle

Link: <https://www.kaggle.com/bhargavrk/bank-loan-application-data>

- Data Size: 166.13 MB
- Data Shape: 307511 rows and 122 columns
- No challenges in processing the data, used pandas to read the data

We can see there are three formats of data types in given features.

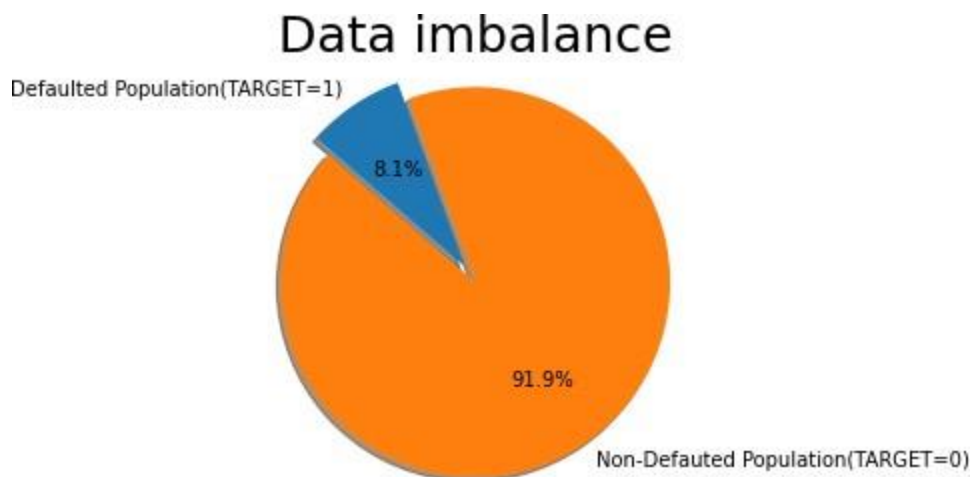
- float64
- int64
- object

Tools (Pandas, SQL, Spark etc) that you will use to process this data

Here, we will be using Python along with the below-listed libraries

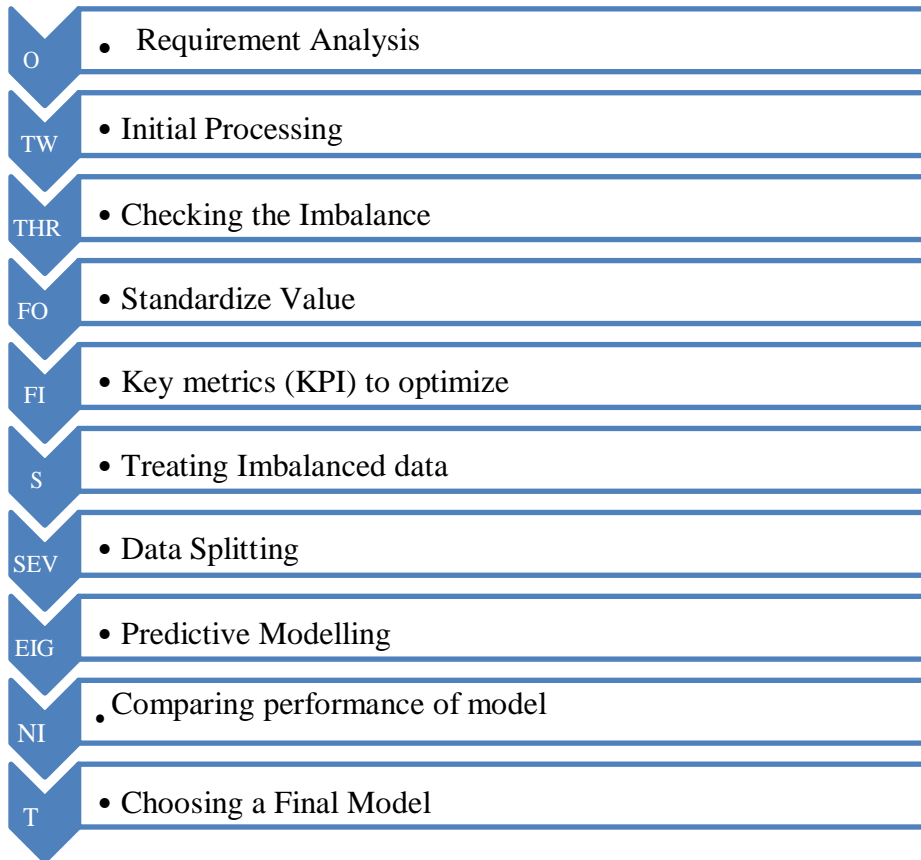
- **Pandas,**
- **NumPy,**
- **matplotlib,**
- **seaborn**
- **Python 3.8, Jupyter Notebook,**
- **Scikit-learn, Scipy**

Class Imbalance:



The class imbalance problems are common in classification problems. Class imbalance occurs when the number of instances in one class is different than another class as show in the above image. Here we can see that 8.1% of Target class=1 and 91.9% of Target class=0. In such cases, classifiers tend to be biased towards the majority class, while the minority class is ignored, many algorithms designed to address imbalanced data classification problems. In this case we going to use resampling & SMOTE are the most important strategies for solving this issue.

4. Approach, Methodology and Results



Step 1: Requirement Analysis

Requirement Analysis

1. Functional Requirements

• Purpose

To predict the loan default. The system tells whether there are chances for loan default or not.

- **Inputs**

Details of customer are provided as the input to the system. The parameters such as income, occupation type, Amount credit etc. are provided to the system as input in csv format.

- **Output**

The system predicts the chances of loan default. It tells whether there are chances for loan default or not.

- **Usability**

It provides ease of use. System is very user-friendly. It is less complicated to work

2. Hardware Requirements

- Computer system
- 16, 32 GB Ram
- Intel i3 and above processor
- Server Overhead
- Port 8000
- Internet connectivity

3. Software Requirements

- Notepad++ or any Code Editor
- Anacoda Navigator , Jupyter Notebook
- Streamlit

- MS-Excel
- Python 3
- Python libraries like pandas, NumPy etc.
- scikit-learn
- Client environment may be Windows or Linux
- Web Browser

Step 2: Initial Processing

During initial processing, we took a first look at the data, checked dimensions, counted missing values, removed some extreme 'outliers', cleaned up inconsistent values and deleted some low variance attributes.

This raw data must be organized before being fetched to the model, this is because, all machine learning algorithm are based on statistics and mathematics. This data contains string values that are not understood by the machine. Also, highly unstable values and empty cells will affect the outcome of the prediction very largely. So, to prepare this data, several data cleaning and preparation techniques as well as algorithms must be used. Treatment of missing values, deletion of empty columns and conversion string values to mathematical values must be done. Also, highly correlated values must be removed, and alignment of test and train data must be done, so that they don't cause problem.

1) Loading the Data

The data files were loaded into Python pandas dataframes:

File Name	Dataframe Name
application_data.csv	Bank_application

2) Missing Values

I have gone through some of the following steps:

- Imputing the missing values with mean/median of the column values for numerical columns and imputing with mode values for categorical columns.
- Dropping the missing value columns, if these values are more than some threshold (you can set any specific threshold value that you want)

Missing value imputation	Handling
columns with missing value < 40 %	Impute with median
AMT_REQ_CREDIT_BUREAU_HOUR',	
'AMT_REQ_CREDIT_BUREAU_DAY',	
'AMT_REQ_CREDIT_BUREAU_WEEK',	
'AMT_REQ_CREDIT_BUREAU_MON',	
'AMT_REQ_CREDIT_BUREAU_QRT',	
'AMT_REQ_CREDIT_BUREAU_YEAR',	
'OBS_30_CNT_SOCIAL_CIRCLE',	
'DEF_30_CNT_SOCIAL_CIRCLE',	
'OBS_60_CNT_SOCIAL_CIRCLE',	
'DEF_60_CNT_SOCIAL_CIRCLE	
CNT_FAM_MEMBERS there are two blank values	Impute with median which is 2
AMT_GOODS_PRICE has standard deviation is quite	Impute with median
AMT_ANNUITY has standard deviation is quite high	Impute with median
OCCUPATION_TYPE', which has high null percentage	Impute with a new category as assigning it to any existing category might influence the analysis
NAME_TYPE_SUITE	categorical feature imputing with mode

- Find the statistical significance of the data and understanding them.
- Replacing some missing values with a valid new value for some important columns, which you think should not be dropped
- Deriving a new column from the existing columns which can help in finding some necessary feature that can be informative.
- Dropping the column that have single value in the column.
- Remove some unwanted columns which do not specify a proper meaning to the data



3) Data Types

The data type of each attribute across the three dataframes was checked and found to be appropriate.

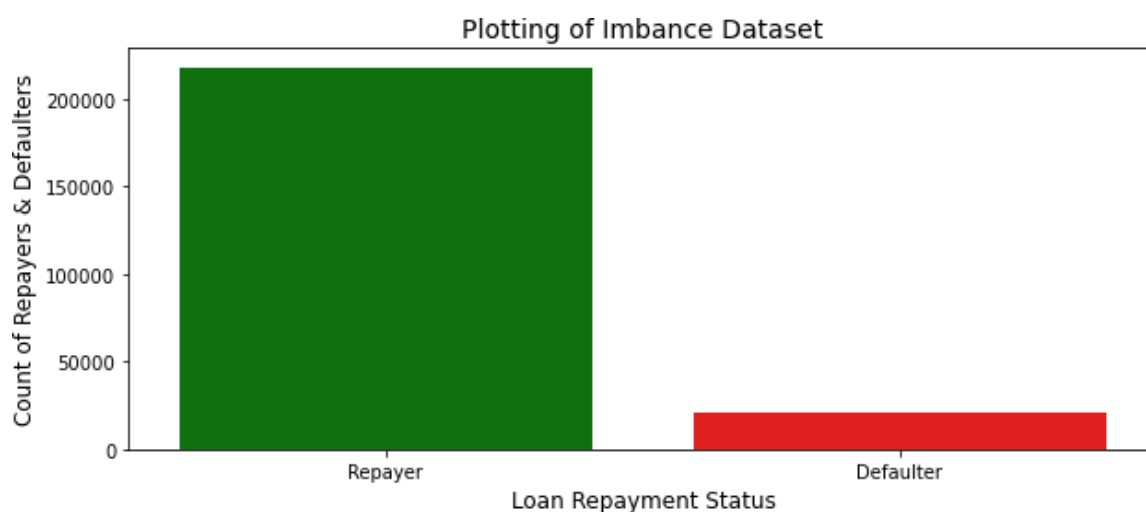
Step 3: Checking Balance or Imbalanced (based on Target Variable)

Data – Imbalance

In the application dataset data corresponding to the clients with TARGET=1 is much lesser (8%) than compared to the data with TARGET=0 (92%).

TARGET	Entries - Count	Entries - %
0	2,82,686	91.93% 
1	24,825	8.07% 
	307511	100%

There is a data imbalance with respect to the TARGET variable in the ratio 92% (0) and 8% (1). This means the data corresponding to the clients with payment difficulties is very less compared to the clients without payment difficulties.



Step 4: Standardize Values

It is a part of Data Exploration, Pre-Processing, and Feature Engineering done in our dataset before passing to the model

1. Conversion of negative to positive values

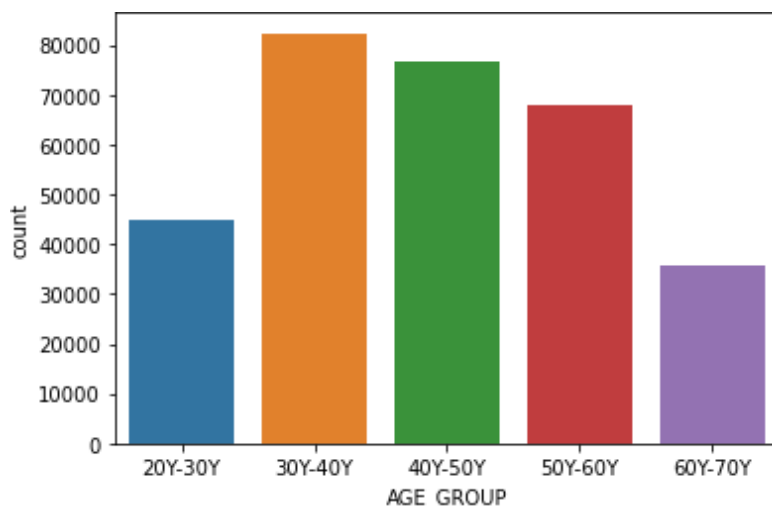
"DAYS_BIRTH", "DAYS_EMPLOYED", "DAYS_REGISTRATION" and "DAYS_ID_PUBLISH"

Since Days birth, Days Employed can't be negative, so need to convert them into positive in application Dataset. Similarly, this will be applicable for above columns

with a new category as assigning it to any existing category might influence the analysis

2. Converting days birth into year

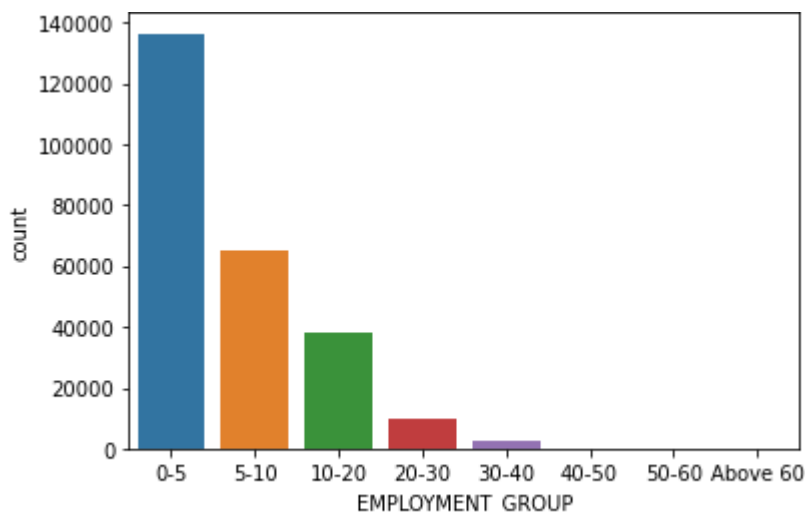
Binning the birth date into new column Age group



Inference: we conclude that more than 50% person whose takes the loan is above the 40 age and 32% loan applicant age is 50 above

3. Converting days employed into year

Binning the DAYS_EMPLOYED into new column EMPLOYMENT_GROUP



Inference: More than 55% of the loan applicants have 0-5 years' work experience and almost 80% of them have less than 10 years' work experience

For both clients with or without Payment difficulties, the peak number of applicants are working with the current organization for less than 5 years of employment and drops considerably after that.

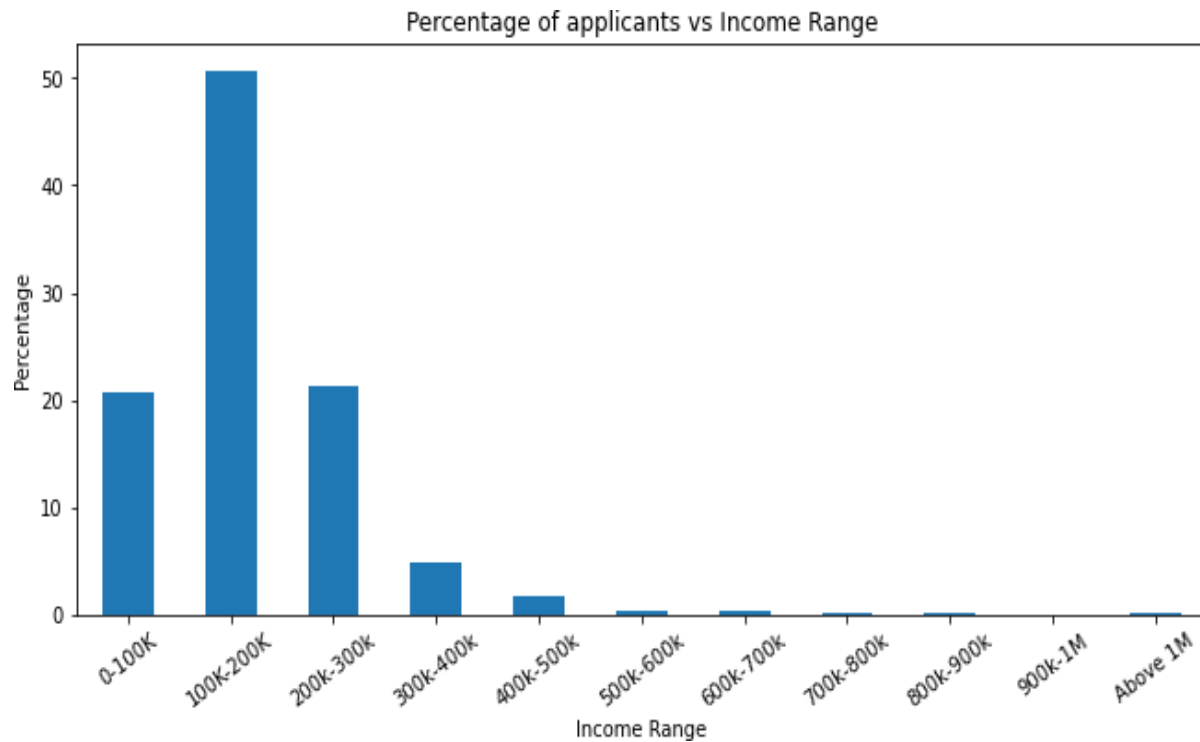
4. **NAME_EDUCATION_TYPE** Changing the name of the value to 'Secondary' from 'Secondary / secondary special'

5. NAME_FAMILY_STATUS

Changing the name of the value to 'Single' from 'Single / not married'

6. Making the columns values readable

Create AMT_INCOME_TOTAL_RANGE column by AMT_INCOME_TOTAL columns



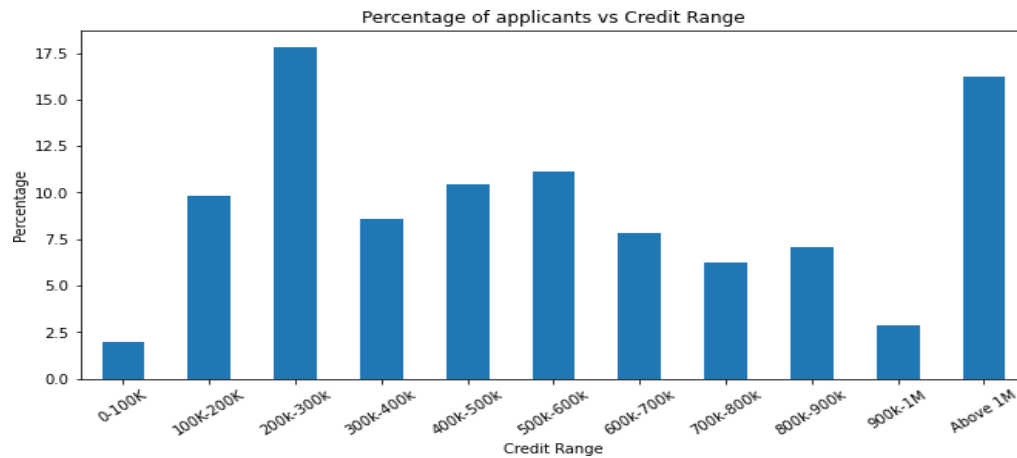
Insight:

50% applicants have income in the range of 100k to 200k

Almost 91% applicants have income in the range of 0 to 300K

Less than 10% applicants have income more than 300K

Create AMT_CREDIT_RANGE column by AMT_CREDIT columns



Insight:

=>Around 16% of clients took loan above 1M

=>Almost 18% of clients took loan between 200k-300k

7. Converting the feature values with two unique values into categorical feature

Replacing the below column values 0, 1 with values 'N', 'Y'

REG_REGION_NOT_LIVE_REGION

REG_REGION_NOT_WORK_REGION

LIVE_REGION_NOT_WORK_REGION

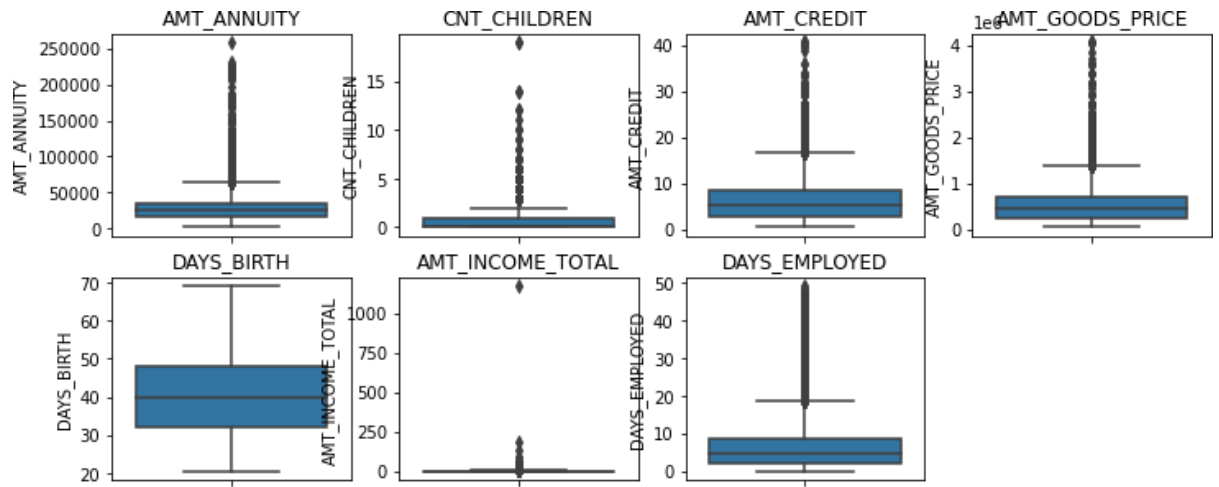
REG_REGION_NOT_LIVE_REGION

REG_CITY_NOT_LIVE_CITY

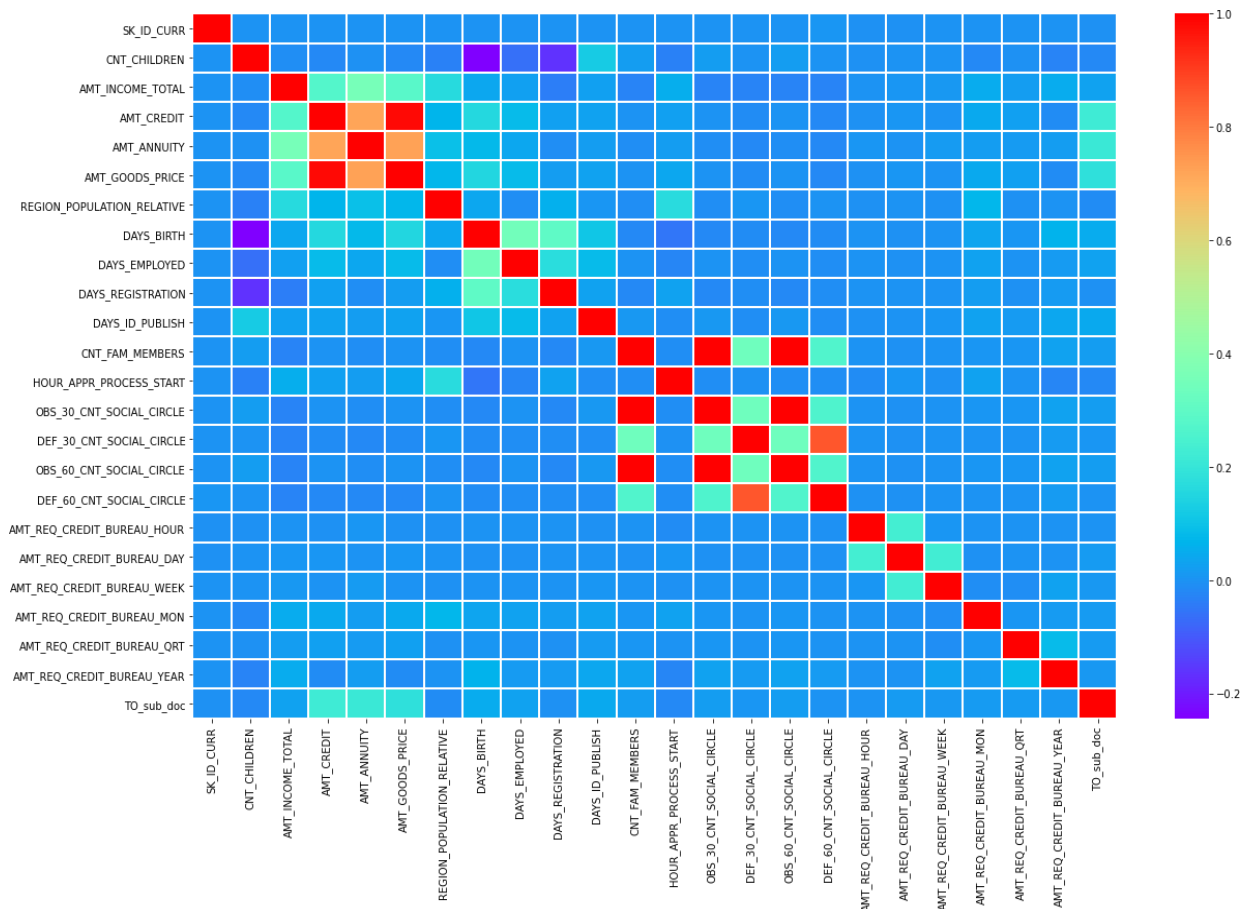
REG_CITY_NOT_WORK_CITY

LIVE_CITY_NOT_WORK_CITY

8. Outlier Detection

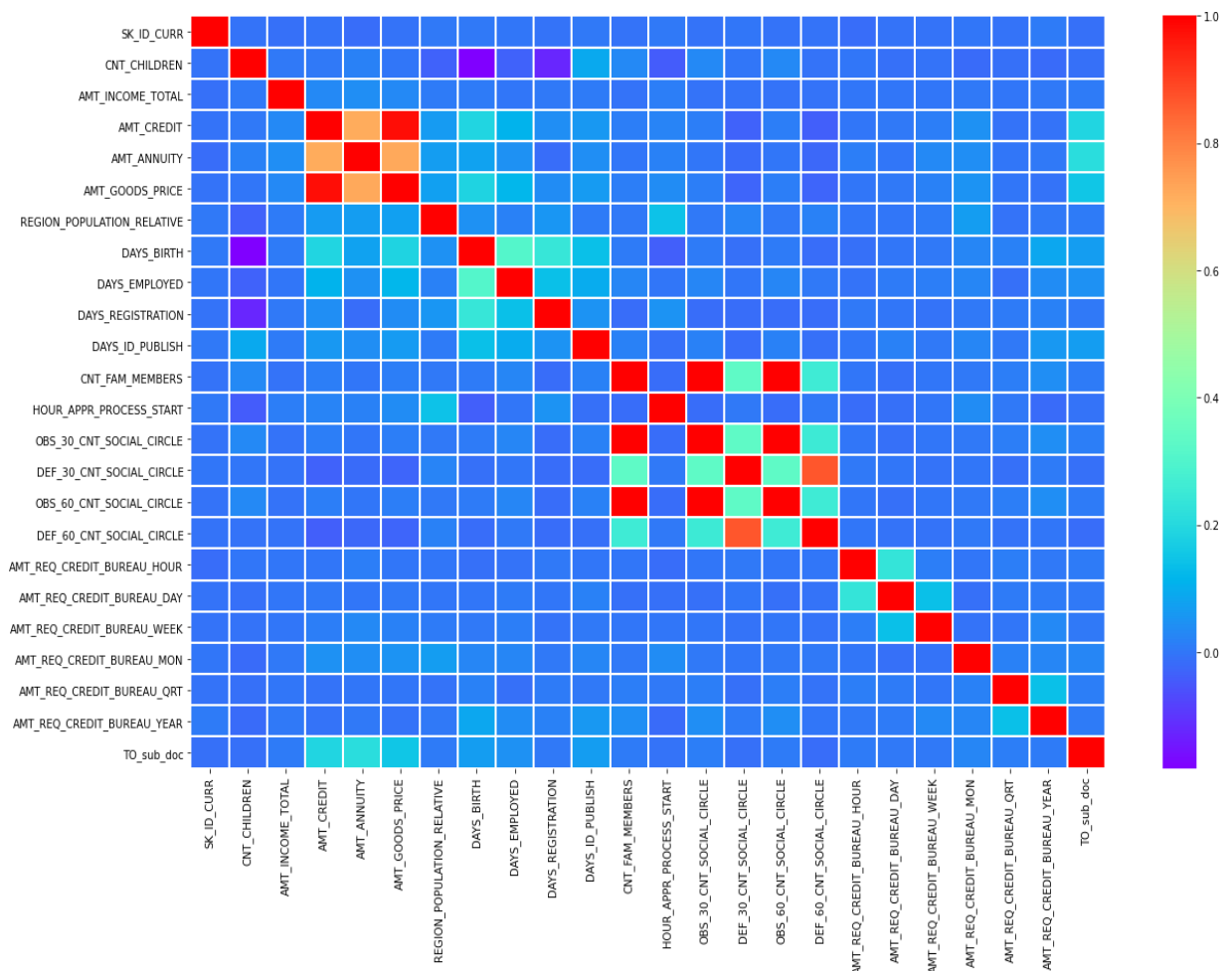


9. Checking Correlation for numerical Values - Target - 0 - Repayer in Application Dataset



DAYS_EMPLOYED and DAYS_BIRTH are also correlated with each other

Correlation for numerical Values - Target - 1- Defaulter in Application Dataset



Inference

AMT_GOODS_PRICE and AMT_CREDIT are highly correlated

CNT_FAM_MEMBERS and CNT_CHILDREN are moderately correlated

AMT_GOODS_PRICE and AMT_ANNUITY are moderately correlated

AMT_ANNUITY and AMT_CREDIT are also correlated with each other

10. Converting categorical values to numerical using Label Encoder and one-hot Encoding:

Label Encoding:

Label encoding does convert the values of the variables from human-readable form to numeric form which is machine understandable. Label encoding converts categorical values to numerical values. It is part of scikit library in python. Label Encoding is of 2 types: (1) Integer Encoding and (2) One-Hot Encoding. We perform “One-Hot Encoding” on our data.

One-Hot Encoding

One-Hot encoding is part of label encoding where the categorical values are converted into numerical values without making any hierarchy. This makes it easier for the machine because hierarchy can affect the model predictions. One-hot encoding takes a column which has categorical data, which has been label encoded, and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on columns and values. For example, we have a column named color that has many values but only 3 unique values, Red, Green, and Blue. We'll get three new columns, one for each color — Red, Green, and Blue. For rows which have the first column value

as Red, the 'Red' column will have a '1' and the other two columns will have '0's. Similarly, for rows which have the first column value as Green, the 'Green' column will have a '1' and the other two columns will have '0's.

color	color_red	color_blue	color_green
red	1	0	0
green	0	0	1
blue	0	1	0
red	1	0	0

One-Hot encoding and Label encoding is performed on our model to convert the categorical values to binary columns. The implementation is done using the **LabelEncoder()** function and **get_dummies()** functions.

```
from sklearn import preprocessing

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

one_hot_encoded_data = pd.get_dummies(categ)
onehot_cols=one_hot_encoded_data.columns
print(onehot_cols)

print(len(onehot_cols))
```

Screenshot 1. (One-Hot Encoding implementation)

Here, we are using Normalization (Min-Max) which will normalized the values of these data in range of 0 to 1, applying this Normalization to both categorical values dataframe and numerical values dataframe so that all my data will be in single normalized formed.

```
#normalizing
normalized_df=(X_train_res-X_train_res.min())/(X_train_res.max()-X_train_res.min())
normalized_df
```

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	REGION_POPULATION_RELATIVE	DAYS_BIRTH	DAYS_EMPL
0	0.333333	0.000742	0.033306	0.076743	0.041379	0.134897	0.126193	0.0
1	0.000000	0.000742	0.191157	0.228655	0.210345	0.158035	0.089024	0.0
2	0.000000	0.001704	0.645364	0.364865	0.831034	0.392880	0.336101	0.2
3	0.000000	0.000935	0.174786	0.127073	0.227586	0.392880	0.220132	0.0
4	0.000000	0.001127	0.064429	0.117783	0.072414	0.158035	0.750099	0.2
...
348345	0.000000	0.000537	0.053780	0.088875	0.072414	0.178817	0.130395	0.0
348346	0.333333	0.000742	0.388088	0.193414	0.393103	0.278181	0.351842	0.0
348347	0.000000	0.001523	0.349572	0.242783	0.451724	0.319123	0.710422	0.0
348348	0.000000	0.000730	0.170146	0.127105	0.141379	0.237154	0.268860	0.0
348349	0.000000	0.000488	0.155510	0.112556	0.151724	0.308552	0.576065	0.2

Screenshot 2. (Normalization (Min-Max))

Step 5: Key metric (KPI) to optimize

Metric is a function that is used to judge the performance of the model. It is a parameter on which the performance of our model depends. This problem comes under binary classification as this problem here is to build a model to predict whether to sanction or reject the loan for the applicant. The commonly used Performance Metrics for Classification Machine Learning Problems are:

- ❖ Accuracy
- ❖ Confusion Matrix
- ❖ Precision,
- ❖ Recall, and F1 score
- ❖ ROC AUC
- ❖ Log-loss
- ❖ G-mean

In this Project we going to use F1 score to evaluate the performance of the algorithm, we often use **F1 score** when the classes are imbalanced and there is a serious downside to predicting false negatives. This Metric is based on the confusion matrix. A confusion

matrix is a summary of prediction results on a classification problem which consists of four parameters: true positive, false positive, true negative and false negative,

F1 score is the harmonic mean of precision and recall. It is given as,

F1 – Score

It is defined as the harmonic mean of precision and recall.

It is used when False Negative and False positive is more important than True Positive and True Negative

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Recall

It is the ratio of correctly predicted positive class out of total actual positive class. Also referred to as 'Sensitivity'.

When the cost of false negative is high and the cost of false negative is low, it is better to use recall as an evaluation metric.

$$Recall = \frac{Predicted\ Positive}{Actual\ Positive} \quad OR \quad \frac{TP}{TP + FN}$$

Precision

It is defined as the ratio of relevant instances made out of all retrieved instances. When the cost of a false positive is very high and the cost of a false negative is low, it is better to use recall as an evaluation metric.

$$Precision = \frac{Predicted\ Positive}{Actual\ Results} \quad OR \quad \frac{TP}{TP + FP}$$

Pros and cons of the metrics (F1 Score)

Pros:

- Considers how the data is distributed. Useful when you have data with imbalance classes.

Cons:

- Less interpretable. Precision and recall are more interpretable than f1-score, since the F1 score is a blend of the precision and recall of the model, which makes it a bit harder to interpret.

We also be using confusion matrix for error analysis and Receiver Operating Characteristic (ROC) curve to visualize our model perform over graph

Confusion Matrix

A confusion matrix or error matrix is a table that is often used to describe the performance of a classification model on a set of data for which the values are known. It allows the visualization of the performance of an algorithm. It allows very easy identification of confusion among classes.

Many performance measures are identified from the confusion matrix.

The number of correct and incorrect predictions are easily visualized through a confusion matrix. It shows how the model is confused by the predictions and the actual values. It also gives the type of error given by the model.

The various terms of a confusion matrix are:

- (1) **True Positive (TP)**: The results under TP show the number of predicted positive results that are actually positive in the validation data.
- (2) **True Negative (TN)**: The results under TN show the number of predicted negative results that are actually negative in the validation data.
- (3) **False Positive (FP)**: The results under FP show the number of predicted positive results that are actually negative in the validation data.

- (4) **False Negative (FN)**: The results under TP show the number of predicted negative results that are actually positive in the validation data.

		prediction outcome		
		p	n	total
actual value	p'	True Positive	False Negative	P'
	n'	False Positive	True Negative	N'
total		P	N	

Receiver Operating Characteristic (ROC)

Receiver Operating Characteristic (ROC) curve is nothing but a way of presenting the rate of FP against the rate of TP over a graph. Each point in the ROC curve represents sensitivity pair for a particular decision threshold.

The better the curve is the better are the prediction results. The area of the curve is the **ROC Score**. More is the area the better trained model it is. If the curve is a straight line, it means that the model is not trained at all.

We calculate the roc score and make a roc curve after getting the confusion matrix. We use function **roc_auc_score()** function of the sklearn.metrics library. Using this score, we plot a graph using “**seaborn**” library. We plot the graph using the **plot()** function of the seaborn library.

We calculate the roc score and plot the curve for both the development and validation process.

Code: Implement this metric from scratch:

```
y_actualvalue= [0,0,0,1,1,1]
y_predictedvalue = [0,1,1,1,1,1]
n = len(y)
```

```
class Metrics:
```

```
    true_postive = 0
    true_negative = 0
    false_postive = 0
    false_negative = 0
    precision = 0
    recall = 0
```

#Confusion matrix implementation

```
def confusion_matrix(self):
```

```
    for i in range(n):
```

```
        if y_actualvalue[i]==1 and y_predictedvalue[i]==1:
            self.true_postive += 1
```

```
        if y_actualvalue[i]==0 and y_predictedvalue[i]==0:
            self.true_negative += 1
```

```
        if y_actualvalue[i]==0 and y_predictedvalue[i]==1:
            self.false_postive += 1
```

```
        if y_actualvalue [i]==1 and y_predictedvalue[i]==0:
            self.false_negative += 1
```

```
    return self.true_postive, self.true_negative, self.false_postive, self.false_negative
```

#Precision and recall implementation

```
def precision_recall(self):
```

```
    self.precision = self.true_postive/(self.true_postive+self.false_postive)
```

```
    self.recall = self.true_postive/(self.true_postive+self.false_negative)
```

```
    print('Precision : ',self.precision, '\nRecall : ',self.recall)
```

#F1_score implementation

```
def f1_score(self):
```

```
    f1 = 2*(self.precision*self.recall)/(self.precision+self.recall)
```

```
    print('F1 Score : ',f1)
```

```
model_performace= Metrics()
```

```
print(model_performace.confusion_matrix())
```

```
model_performace.precision_recall()
```

```
model_performace.f1_score()
```

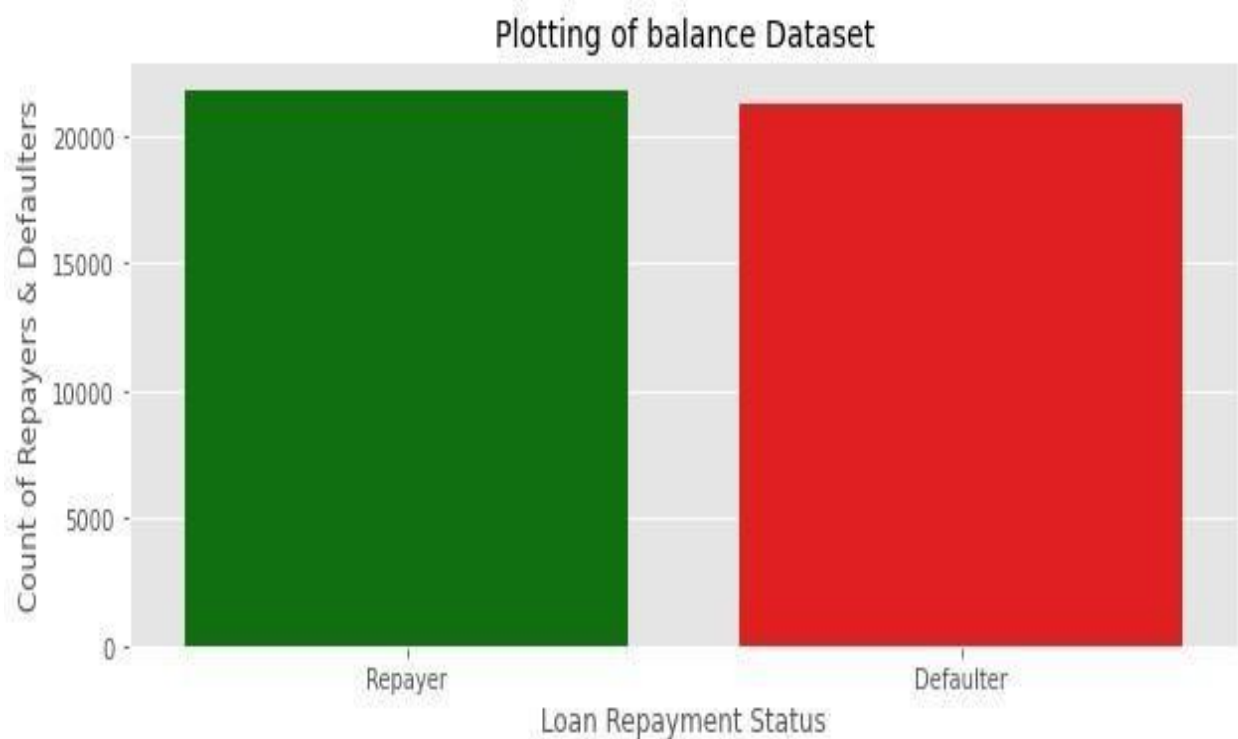
Step 6: Treating Imbalanced data

I will follow the Data Level Approach to treat the Imbalanced Data by using Re-Sampling technique (Random Under-Sampling) and SMOTE Oversampling technique (Over-Sampling)

Random Under-Sampling –

It aims to balance class distribution by randomly eliminating majority class examples. This is done until the majority and minority class instances are balanced out.

After treating the imbalance dataset using Re-Sampling technique (Random Under Sampling)



Over-Sampling Using SMOTE:

Here we oversampled Train and Test data separately to avoid data leakage

Plotting train dataset after Oversampling technique (Over-Sampling)

After treating the imbalance dataset using SMOTE Oversampling technique
(Oversampling)

```
import imblearn
from imblearn.over_sampling import SMOTE |
oversample = SMOTE()
X_train_res, y_train_res = oversample.fit_resample(X_train_1, y_train_1)
X_test_res, y_test_res = oversample.fit_resample(X_test_1, y_test_1)

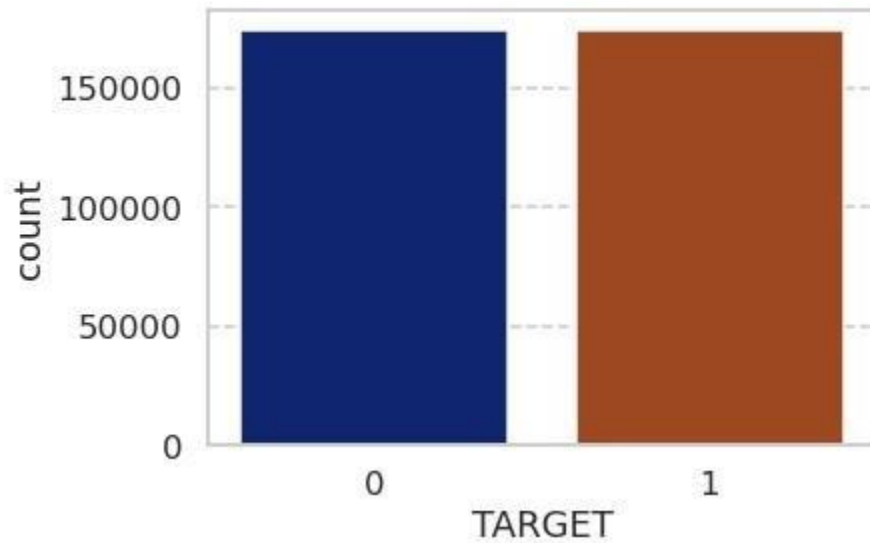
##print("Before sample Train %s" format(y_train_1.shape))
#print("After sample Train %s"% Counter(y_train_res))

#print("Before sample Test %s"% Counter(y_test_1))
#print("After sample Test %s"%Counter(y_test_res))
```

Screenshot 3. (Treating Imbalance)

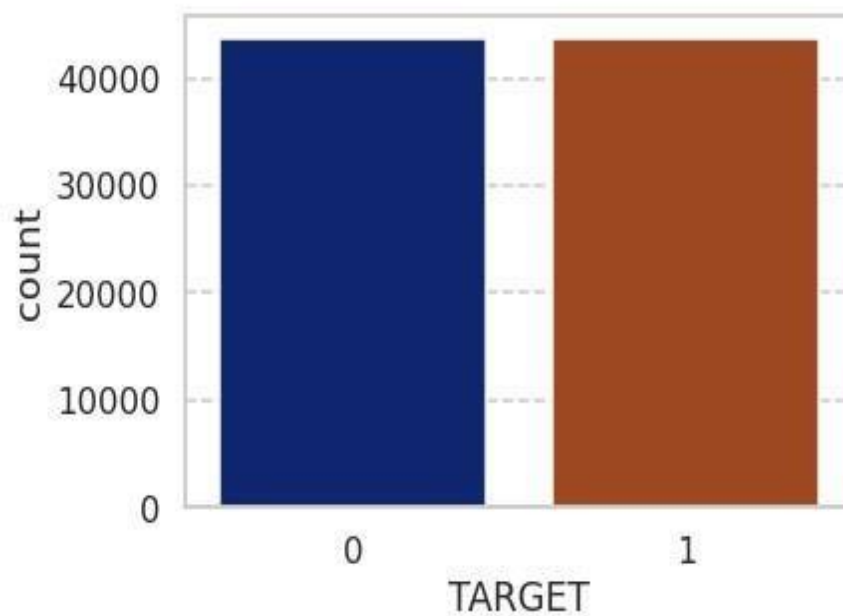
Plotting train dataset after Oversampling technique (Over-Sampling)

<AxesSubplot:xlabel='TARGET', ylabel='count'>



Plotting test dataset after Oversampling technique (Over-Sampling)

<AxesSubplot:xlabel='TARGET', ylabel='count'>



Step 7: Data splitting

The model prediction can only be verified if we check if our results are accurate or not. This can be done by performing validation on the training data itself. So, the train data set must be split so that one part of it can be used to train the model while the other part is used for validation.

Here we used Simple Hold-Out Strategy (splitting data into Train and Test (with 80-20))

There are many methods for splitting the data and doing model validation like

- Simple Hold-Out Strategy (splitting data into Train and Test (with either 80-20 or 70-30 split))
- Three-way-hold-out strategy (splitting data into Training, Validation and Testing (50-20-30 split))
- Cross Validation (K-Fold cross-validation or RepeatedKfold etc)
- Leave-One-Out
- Grid Search etc.

We have split our train data into a 8:2 ratio where 80% of the data will be used for training the model and 20% will be used for validating the results. We will achieve this using **train_test_split()** from the sklearn.model_selection library. We give the test size as 0.2.

```
from sklearn.model_selection import train_test_split

X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X_up, Y_up, test_size = 0.2, random_state = 0)

# describes info about train and test set
print("Number transactions X_train dataset: ", X_train_1.shape)
print("Number transactions y_train dataset: ", y_train_1.shape)
print("Number transactions X_test dataset: ", X_test_1.shape)
print("Number transactions y_test dataset: ", y_test_1.shape)
```

Screenshot 4. (Data Splitting)

Step 8: Predictive Modelling

We will be training our model with both under sampled data and oversampled data

Classification Models

The major aim of this project is to predict which of the customers will have their loan paid or not. Therefore, this is a supervised classification problem to be trained with algorithms like:

The Problems are solved using below techniques:

1. Logistic Regression
2. Random Forest
3. Naive Base
4. Ada Boost

Logistic Regression

Logistic regression is a standard industry algorithm that is commonly used in practice because of its simplicity and balanced error distribution. It is a binary classification technique that generates one of two variables as its result, e.g., reject, or approve borrowers. The logistic regression formula is shown below:

$$\ln\left(\frac{F(x)}{1-F(x)}\right) = \beta_0 + \sum_{i=1}^n \beta_i X_i$$

Random Forest - a decision tree-based ensemble method

It is mostly used in areas such as classification, regression analysis etc. At the training time RF algorithm creates many decision trees. RF is a supervised learning approach which need a test data for the model for training. It creates random forests for the problem set and then find the solution using these random forests

Naive Bayes:

Naive Bayes on the other hand uses probabilistic approach and considers the features as conditionally independent of each other.

AdaBoost – a boosting algorithm using decision trees

Fitting LR model into data set:

Model is fit into the data set using the **LogisticRegresion()** function of the **sklearn.linear_model** package, the **RandomForestClassifier()** function of the **sklearn.ensemble** package, **GaussianNB()** function of the **sklearn.naive_bayes** package and **AdaBoostClassifier(n_estimators=50)** function of the **sklearn.ensemble** package. The result is stored in a variable

Once the function has been implemented. The model is fit using the **fit()** function. After fitting the model, we start predicting the results using **predict()** function. The **predict()** function will start predicting the probability of 1's and 0's in the data set and then verification will be done on the basis of confusion matrix.

Step 9: Comparing performance of model

Model performance in Under sampled data and over sampled data

1. Logistic Regression:

```
Accuracy 0.6029730566738929
      precision    recall  f1-score   support

     0         0.61      0.60      0.60      3242
     1         0.60      0.61      0.60      3216

 accuracy
macro avg         0.60      0.60      0.60      6458
weighted avg         0.60      0.60      0.60      6458
```

Screenshot 5. (Under sampled data result)

```
Accuracy 0.5723771580345286
      precision    recall  f1-score   support

     0         0.56      0.67      0.61    43674
     1         0.59      0.48      0.53    43674

 accuracy
macro avg         0.58      0.57      0.57    87348
weighted avg         0.58      0.57      0.57    87348
```

Screenshot 6. (Over sampled data result)

=>Based on Accuracy we can see that logistic regression model works best on under sampled data.

2. Random Forest

```
# Assess the performance of this model on the test set

pred = randomForest.predict(x_test)
rfPredictproba = randomForest.predict_proba(x_test)[: ,1] #for ROC curve

randomforest_accuracy = accuracy_score(y_test, pred)
print("Accuracy",randomforest_accuracy)
print(classification_report(y_test,pred))
```

Accuracy	0.6217095075874884				
	precision	recall	f1-score	support	
0	0.62	0.64	0.63	6551	
1	0.62	0.60	0.61	6365	
accuracy			0.62	12916	
macro avg	0.62	0.62	0.62	12916	
weighted avg	0.62	0.62	0.62	12916	

Screenshot 7. (Under sampled data result)

```
# Assess the performance of this model on the test set

R_pred = rfc.predict(X_test_res)
rfPredictproba = rfc.predict_proba(X_test_res)[: ,1] #for ROC curve

randomforest_accuracy = accuracy_score(y_test_res, R_pred)
print("Accuracy",randomforest_accuracy)
print(classification_report(y_test_res,R_pred))
```

Accuracy	0.9491001511196593				
	precision	recall	f1-score	support	
0	0.91	1.00	0.95	43674	
1	1.00	0.90	0.95	43674	
accuracy			0.95	87348	
macro avg	0.95	0.95	0.95	87348	
weighted avg	0.95	0.95	0.95	87348	

Screenshot 8. (Over sampled data result)

=>Based on Accuracy we can see that random forest model works best on over sampled data.

3. Naive Bayes

```
# performance of this model on the validation set
pred = gnb.predict(x_valid)

gnb_accuracy = accuracy_score(y_valid, pred)
print("Accuracy", gnb_accuracy)
print(classification_report(y_valid, pred))
```

Accuracy 0.5470733973366367

	precision	recall	f1-score	support
0	0.53	0.85	0.65	3242
1	0.61	0.24	0.35	3216
accuracy			0.55	6458
macro avg	0.57	0.55	0.50	6458
weighted avg	0.57	0.55	0.50	6458

Screenshot 9. (Under sampled data result)

```
# performance of this model on the test set
pred = gnb.predict(X_test_res)
gnbPredictproba = gnb.predict_proba(X_test_res)[: ,1]

gnb_accuracy = accuracy_score(y_test_res, pred)
print("Accuracy", gnb_accuracy)
print(classification_report(y_test_res, pred))
```

Accuracy 0.5956862206347026

	precision	recall	f1-score	support
0	0.62	0.50	0.55	43674
1	0.58	0.69	0.63	43674
accuracy			0.60	87348
macro avg	0.60	0.60	0.59	87348
weighted avg	0.60	0.60	0.59	87348

Screenshot 10. (Over sampled data result)

=>Based on Accuracy we can see that random forest model works best on over sampled data.

4. ADA Boost

```
: # Assess the performance of this model on the test set
pred = adab.predict(x_test)
adabPredictproba = adab.predict_proba(x_test)[: ,1]

adab_accuracy = accuracy_score(y_test, pred)
print("Accuracy",adab_accuracy)
print(classification_report(y_test,pred))
```

```
Accuracy 0.6232579746051409
              precision    recall  f1-score   support

      0         0.63       0.62       0.63       6551
      1         0.62       0.62       0.62       6365

   accuracy                   0.62       12916
  macro avg         0.62       0.62       0.62       12916
 weighted avg         0.62       0.62       0.62       12916
```

Screenshot 11. (Under sampled data result)

```
# Assess the performance of this model on the test set
pred = adab.predict(X_test_res)
adabPredictproba = adab.predict_proba(X_test_res)[: ,1]

adab_accuracy = accuracy_score(y_test_res, pred)
print("Accuracy",adab_accuracy)
print(classification_report(y_test_res,pred))
```

```
Accuracy 0.9418532765489765
              precision    recall  f1-score   support

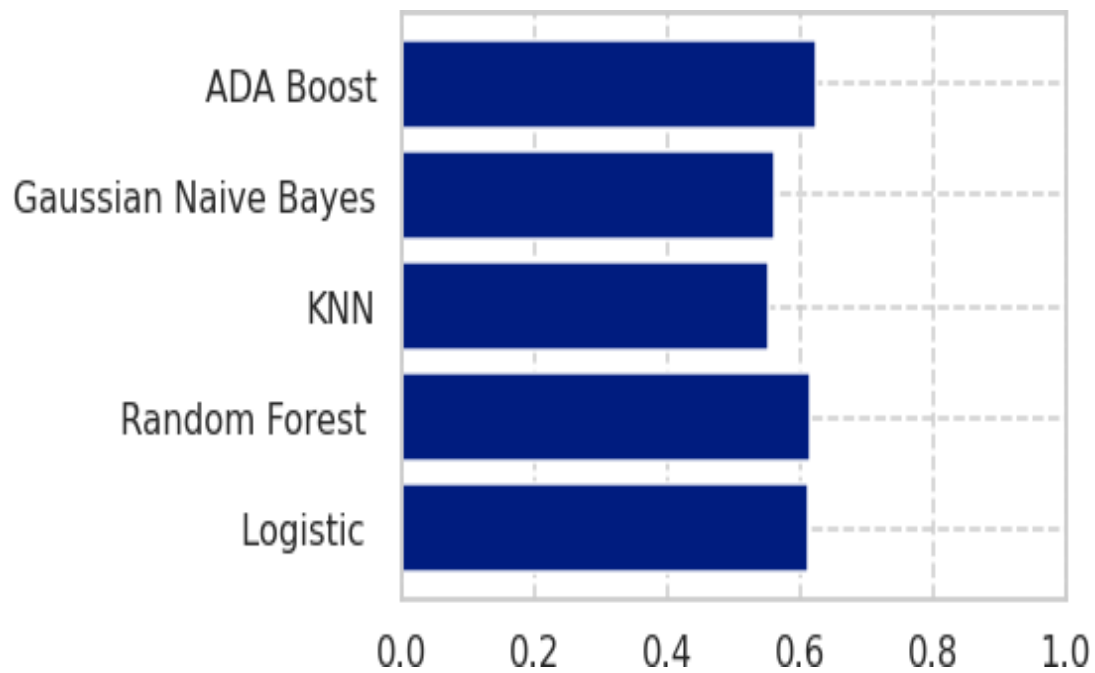
      0         0.90       1.00       0.94      43674
      1         0.99       0.89       0.94      43674

   accuracy                   0.94      87348
  macro avg         0.95       0.94       0.94      87348
 weighted avg         0.95       0.94       0.94      87348
```

Screenshot 12. (Over sampled data result)

=>Based on Accuracy we can see that ADA Boost model works best on over sampled data.

Compare Model Cross Val Scores Result on under-sampled data

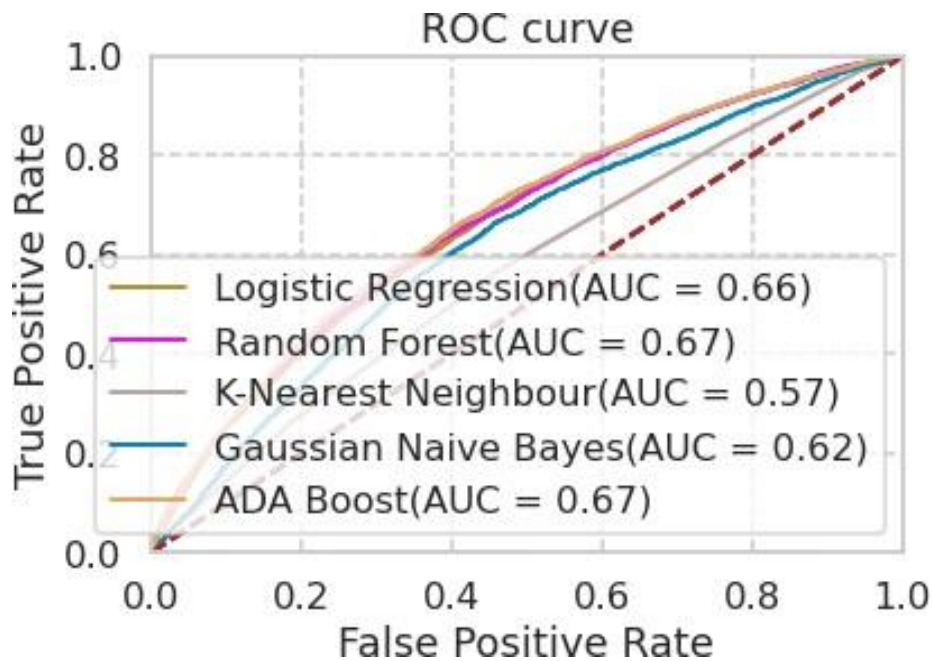


From the above comparison we can say that ADA Boost and Random Forest works best on under-sampled data

Step 10: Choosing a Final Model

Let's Visualizations of Results and Evaluations using ROC Curve to decide the best model

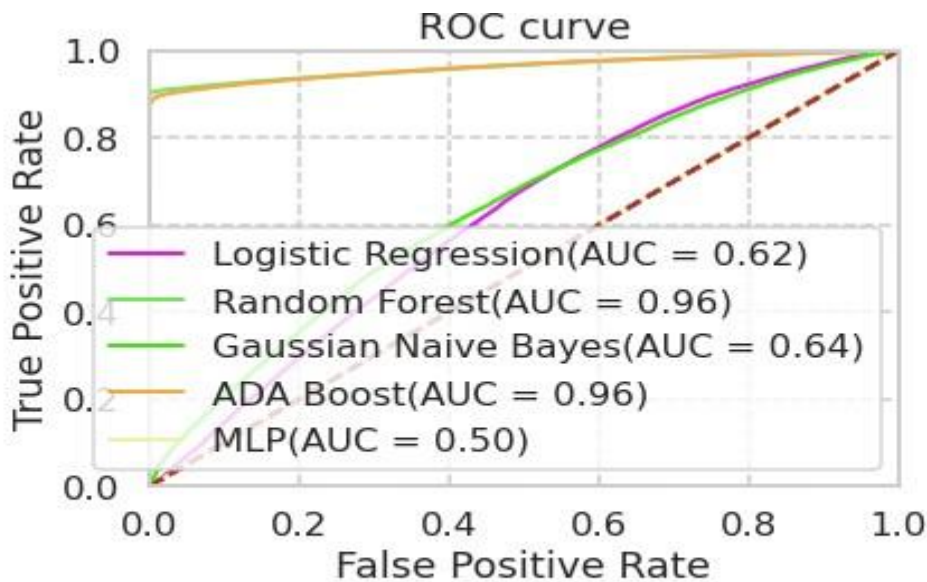
Model performance in Under sampled data



Observations from ROC

- The ADA Boost and Random Forest model gives the best same results as compared to the other models with Logistic Regression pretty close to them.

Model performance in Oversampled data



<Figure size 432x432 with 0 Axes>

Observation: The AdaBoost Classifier and Random Forest model gives the best same results as compared to the other models

5. Discussion of Results

The F1-Score was used to determine a best performing model as it reflects performance in both Recall and Precision. Following hypothesis testing, the F1-Scores from AdaBoost and Random Forest were found to be significantly higher than the F1-Scores from other models.

The Random Forest model continued to produce consistent results (although very slightly worse) when tested on the unseen test set. This indicates that the model did not suffer from overfitting to the training data or data leakage and should maintain consistent performance on further unseen data.

In this project I will be choosing the Random Forest model trained on over-sampled data as my best performance model to deploy.

6. Conclusion:

We have successfully predicted the “Default Cases”, We have achieved an accuracy of about 95% and a roc score of about 96%, Thus, it is concluded that we get better results using Random Forest model trained on over-sampled data.

Link for blog:

<https://medium.com/@aishwaryapachaiyappan/loan-default-prediction-4a713aac14ad>

7. References:

- [1] Liu, M.A., 2018. A Comparison of Machine Learning Algorithms for Prediction of Past Due Service in Commercial Credit. Grey Literature for Ph.D. Candidates, Kennesaw State University.
- [2] Yu, X., 2017. Machine learning application in online lending risk prediction. arXiv preprint arXiv:1707.04831.
- [3] Islam, S.R., Eberle, W. and Ghafoor, S.K., 2018. Credit Default Mining Using Combined Machine Learning and Heuristic Approach. arXiv preprint arXiv:1807.01176
- [4] <https://medium.com/@fenjiro/data-mining-for-banking-loan-approval-use-case-e7c2bc3ece3>
- [5] <https://www.ijert.org/predict-loan-approval-in-banking-system-machine-learning-approach-for-cooperative-banks-loan-approval>

8. Deployment and Productionization

The final phase of this project is Deployment, here we are deploying the whole machine learning pipeline into a production system, into a real-time scenario.

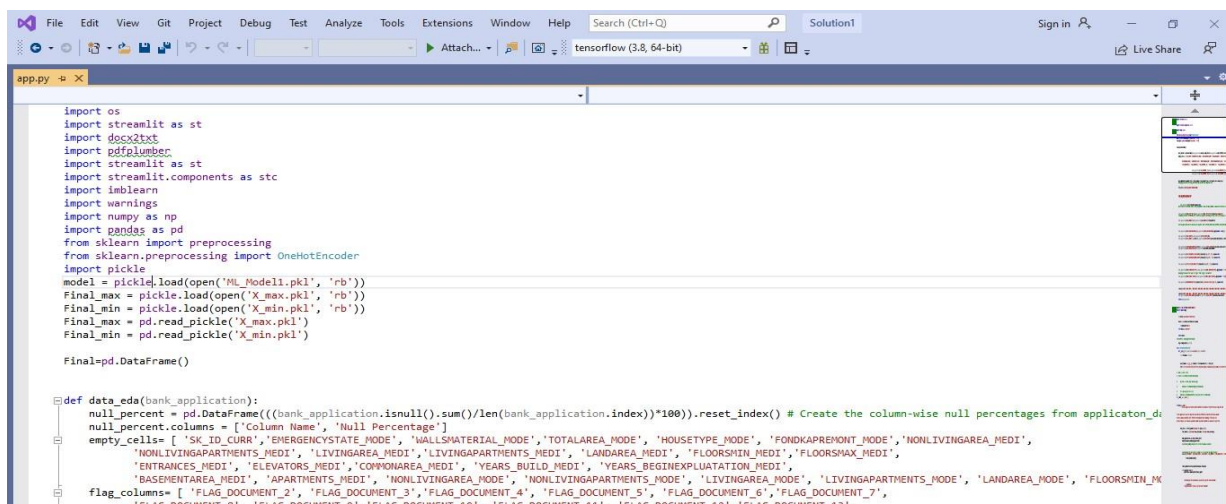
In this final stage we need to deploy this machine learning pipeline to put of research available to end users for use. The model will be deployed in real world scenario where it takes continuous raw input from real world and predict the output.

Logistic Regression, Random Forest, Naive Bayes and ADA Boost were trained using the trained dataset, it was found that , Random Forest gives the best performance.

App Building:

In this step a web application developed using Streamlit. It provides a functionality to the user for uploading the input in csv format.

As part of user interface, I will be developing a Streamlit app. Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. Streamlit allows you to write an app the same way you write a python code. The file will be stored in .py format.



```
import os
import streamlit as st
import docx2txt
import pdfplumber
import streamlit.components as stc
import imblearn
import warnings
import numpy as np
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
import pickle

model = pickle.load(open('ML_Model1.pkl', 'rb'))
Final_max = pickle.load(open('X_max.pkl', 'rb'))
Final_min = pickle.load(open('X_min.pkl', 'rb'))
Final_max = pd.read_pickle('X_max.pkl')
Final_min = pd.read_pickle('X_min.pkl')

Final=pd.DataFrame()

def data_eda(bank_application):
    null_percent = pd.DataFrame(((bank_application.isnull().sum()/len(bank_application.index))*100).reset_index()) # Create the column-wise null percentages from application_data
    null_percent.columns = ['Column Name', 'Null Percentage']
    empty_cells= [ 'SK_ID_CURR','EMERGENCYSTATE_MODE', 'WALLSMATERIAL_MODE', 'TOTALAREA_MODE', 'HOUSETYPE_MODE', 'FONDKAPREMONT_MODE', 'NONLIVINGAREA_MEDI',
                    'NONLIVINGPARTMENTS_MEDI', 'LIVINGAREA_MEDI', 'LIVINGPARTMENTS_MEDI', 'LANDAREA_MEDI', 'FLOORSNIN_MEDI', 'FLOORSMAX_MEDI',
                    'ENTRANCES_MEDI', 'ELEVATORS_MEDI', 'COMMONAREA_MEDI', 'YEARS_BUILD_MEDI', 'YEARS_BEGINEXPLOITATION_MEDI',
                    'BASEMENTAREA_MEDI', 'APARTMENTS_MEDI', 'NONLIVINGAREA_MODE', 'NONLIVINGPARTMENTS_MODE', 'LIVINGAREA_MODE', 'LIVINGPARTMENTS_MODE', 'LANDAREA_MODE', 'FLOORSNIN_MX',
                    'FLAG_DOCUMENT_2', 'FLAG_DOCUMENT_3', 'FLAG_DOCUMENT_4', 'FLAG_DOCUMENT_5', 'FLAG_DOCUMENT_6', 'FLAG_DOCUMENT_7',
                    'FLAG_DOCUMENT_8', 'FLAG_DOCUMENT_9', 'FLAG_DOCUMENT_10', 'FLAG_DOCUMENT_11', 'FLAG_DOCUMENT_12', 'FLAG_DOCUMENT_13' ]
```

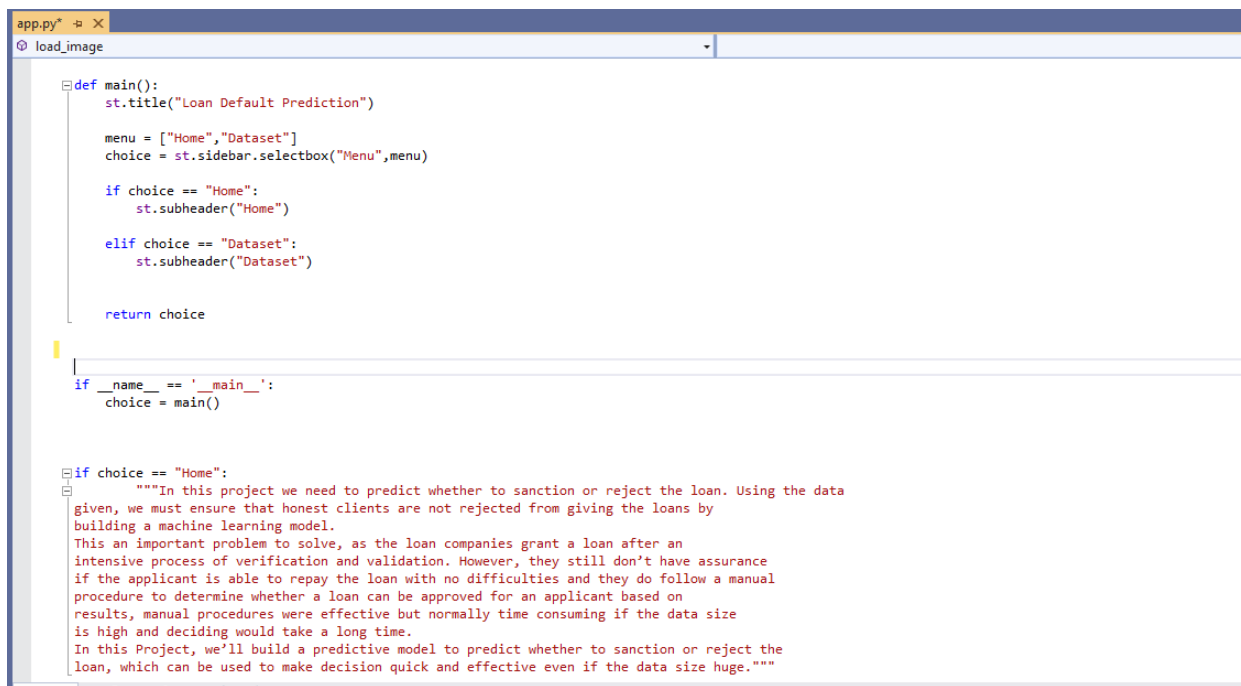
We can our web app using streamlit in local with the following command:



```
Anaconda Prompt (Anaconda3)
(base) C:\Users\Administrator>cd C:\deploy
(base) C:\deploy>streamlit run app.py
```

Functioning of Streamlit app:

=>We are creating a web app with Title as Loan default Prediction, with menus as Home and Dataset which is shown below



```
app.py
load_image

def main():
    st.title("Loan Default Prediction")

    menu = ["Home", "Dataset"]
    choice = st.sidebar.selectbox("Menu", menu)

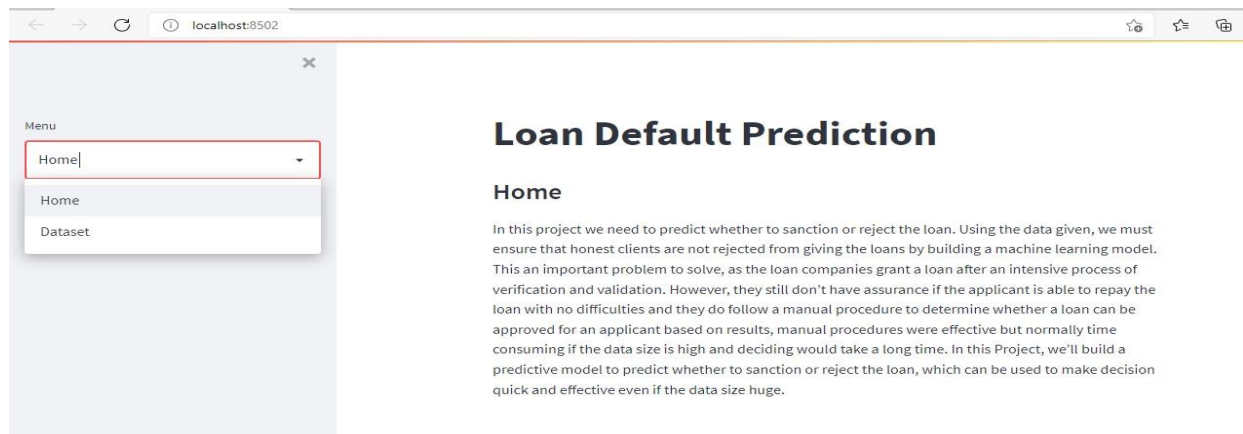
    if choice == "Home":
        st.subheader("Home")

    elif choice == "Dataset":
        st.subheader("Dataset")

    return choice

if __name__ == '__main__':
    choice = main()

if choice == "Home":
    """In this project we need to predict whether to sanction or reject the loan. Using the data
    given, we must ensure that honest clients are not rejected from giving the loans by
    building a machine learning model.
    This an important problem to solve, as the loan companies grant a loan after an
    intensive process of verification and validation. However, they still don't have assurance
    if the applicant is able to repay the loan with no difficulties and they do follow a manual
    procedure to determine whether a loan can be approved for an applicant based on
    results, manual procedures were effective but normally time consuming if the data size
    is high and deciding would take a long time.
    In this Project, we'll build a predictive model to predict whether to sanction or reject the
    loan, which can be used to make decision quick and effective even if the data size huge."""
```

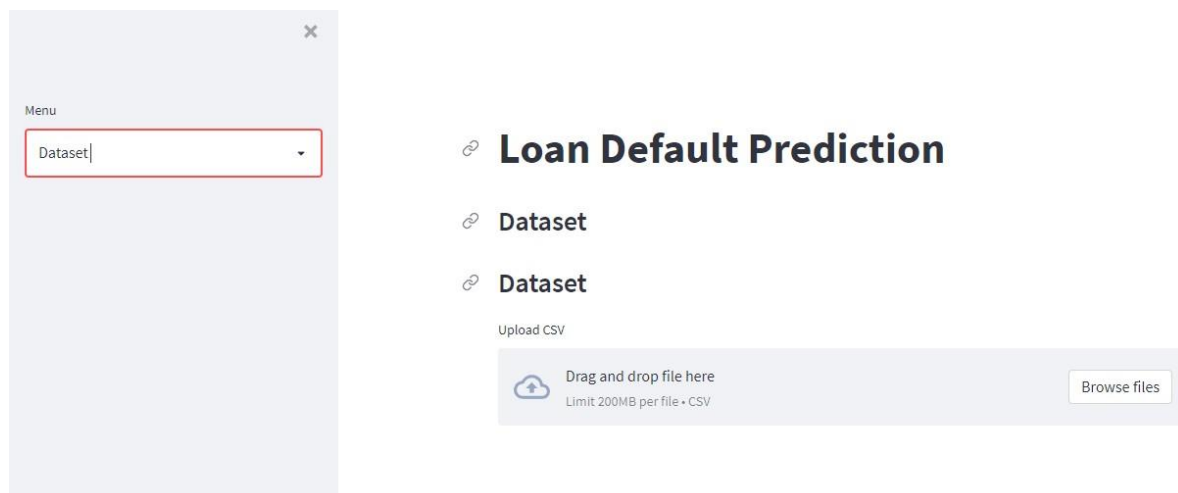


Here we will provide our input in the form .csv will be given from web interface,

```
elif choice == "Dataset":
    st.subheader("Dataset")
    data_file = st.file_uploader("Upload CSV", type=["csv"])
    if data_file is not None:
        file_details = {"filename": data_file.name, "filetype": data_file.type,
                        "filesize": data_file.size}

        st.write(file_details)
        bank_application = pd.read_csv(data_file)
        st.dataframe(bank_application)
        Input_data = data_eda(bank_application)

        # Generating dummy variables for rest of the nominal variables
```



then the input will be cleaned and then featured for prediction

Menu

Dataset

Loan Default Prediction

Dataset

Upload CSV

Drag and drop file here
Limit 200MB per file • CSV

Browse files

input.csv 2.5KB

×

```
{
  "filename": "input.csv"
  "filetype": "application/vnd.ms-excel"
  "filesize": 2563
}
```

	SK_ID_CURR	NAME_CONTRACT_T...	CODE_GEN...	FLAG_OWN_...	FLAG_OWN_REAL...	CH
0	100112	Cash loans	M	Y	Y	

Submit

Once we click on the submit button, our model will be called, and we get the model prediction for the input data as output

Dataset

Upload CSV

Drag and drop file here
Limit 200MB per file • CSV

Browse files

input.csv 2.5KB

×

```
{
  "filename": "input.csv"
  "filetype": "application/vnd.ms-excel"
  "filesize": 2563
}
```

	SK_ID_CURR	NAME_CONTRACT_T...	CODE_GEN...	FLAG_OWN_...	FLAG_OWN_REAL...	CH
0	100112	Cash loans	M	Y	Y	

Submit

Congratulations!! you will get the loan from Bank

Deployment:

I am using Git repository to code base. Attaching the link for further distribution and enhancement

Github : https://github.com/shrey01/loan_default_prediction

Thank you