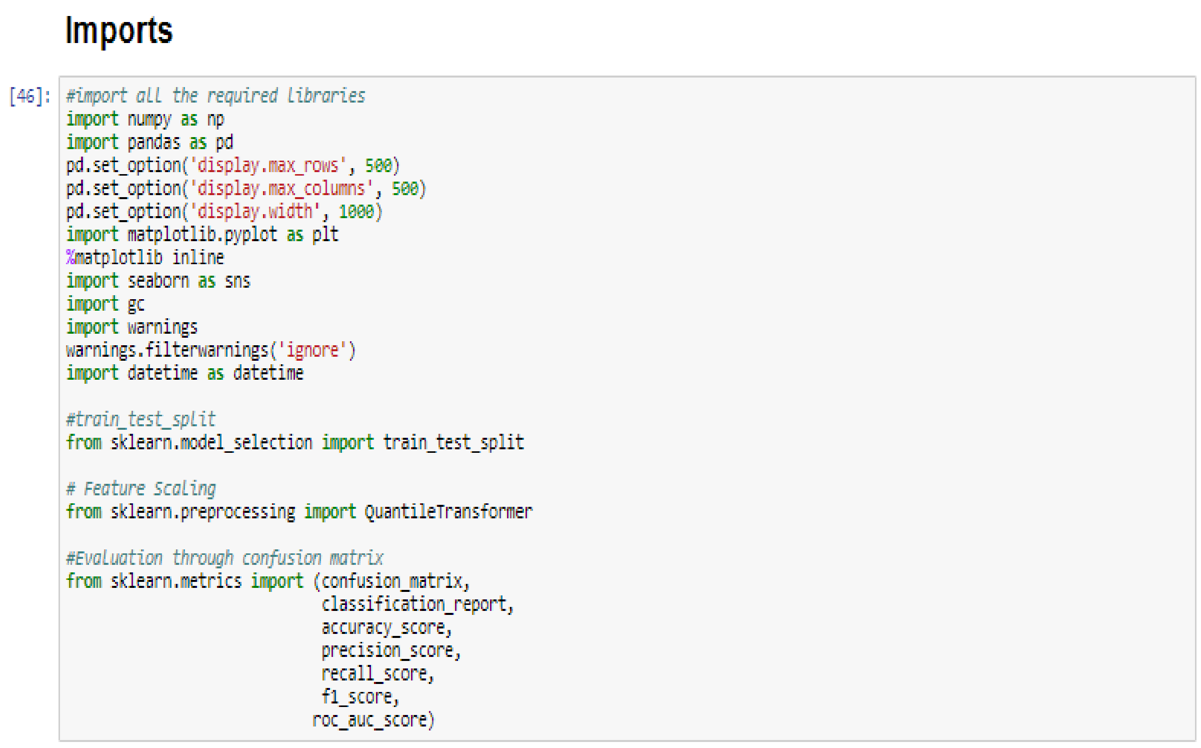
APPROACH

1. ANALYSING THE PROBLEM STATEMENT

My first and foremost step was to analyse the problem statement. I wanted to know what the problem is demanding and what the nature of my target variable is. After analysing I came to the conclusion that the problem statement requires a supervised classification approach.

2. IMPORTING THE BASIC LIBRARIES

At the beginning, I imported all the necessary libraries.



Imports

```
[46]: #import all the required libraries
import numpy as np
import pandas as pd
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import gc
import warnings
warnings.filterwarnings('ignore')
import datetime as datetime

#train_test_split
from sklearn.model_selection import train_test_split

# Feature Scaling
from sklearn.preprocessing import QuantileTransformer

#Evaluation through confusion matrix
from sklearn.metrics import (confusion_matrix,
                             classification_report,
                             accuracy_score,
                             precision_score,
                             recall_score,
                             f1_score,
                            roc_auc_score)
```

3. EXPLORATORY DATA ANALYSIS

This step is one of the most crucial and important step to understand the pattern of our data. It includes checking the dimension of the data, checking the numeric details of the data, checking for missing values, feature importance, correlation of several fields and lastly the imbalanced target.

4. DATA PRE-PROCESSING

The data pre-processing step included the following steps:-

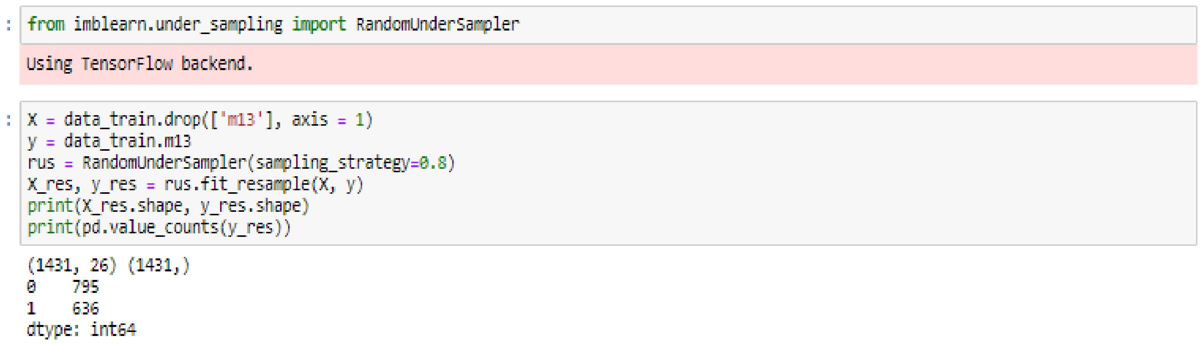
a) I dropped the loan_id column.
b) I format the first_payment_date column of test data to month-year format.
c) I imputed the first_payment_date and origination_date by calculating the days since epoch.

d) The columns source and loan_purpose clearly showed values depicting frequency. Thus, I decided to frequency encode those fields.

e) The column financial_institution showed least importance, so I decided to drop it.

5. DATA SPLIT AND FEATURE ENGINEERING

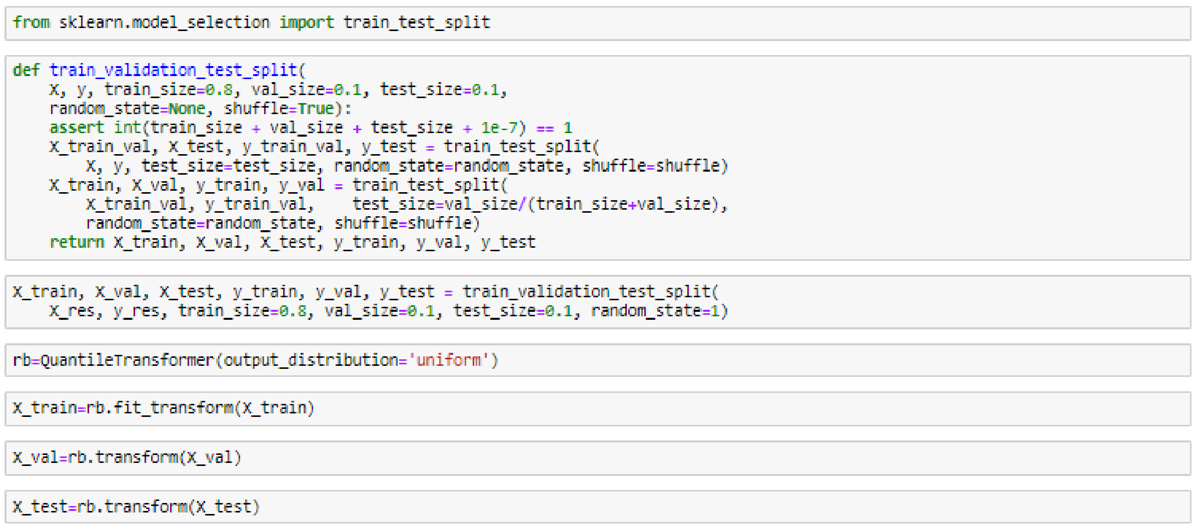
After analysis in the EDA, it was clearly found that it is a case of imbalanced classification so sampling of our data was required before we split it. The ratio of majority class to minority class was 99.45% to 0.55%, so I decided to under sample the majority class as 99.5% of majority class even after under sampling would contain all the important underlying data patterns.



```
: from imblearn.under_sampling import RandomUnderSampler

  Using TensorFlow backend.

: X = data_train.drop(['m13'], axis = 1)
  y = data_train.m13
  rus = RandomUnderSampler(sampling_strategy=0.8)
  X_res, y_res = rus.fit_resample(X, y)
  print(X_res.shape, y_res.shape)
  print(pd.value_counts(y_res))

  (1431, 26) (1431,)
  0    795
  1    636
  dtype: int64
```

Now let’s split the dataset into 3 parts — training, validation, and test datasets. The validation dataset we can use again and again with different models. Once we think we’ve got the best model, we will use our testing dataset.

The reason we do it this way is that our model should not just give us good results with a part of the training dataset, but should also provide good results with data that we have never seen before. This is what will happen in real life. By keeping aside the test dataset to be used only once, we force ourselves to not overfit on the validation dataset. This is what Kaggle competitions do as well.The trainsize/valsize/testsize show the fraction of the total dataset that should be reserved for training/validation/testing.



```
from sklearn.model_selection import train_test_split

def train_validation_test_split(
    X, y, train_size=0.8, val_size=0.1, test_size=0.1,
    random_state=None, shuffle=True):
    assert int(train_size + val_size + test_size + 1e-7) == 1
    X_train_val, X_test, y_train_val, y_test = train_test_split(
        X, y, test_size=test_size, random_state=random_state, shuffle=shuffle)
    X_train, X_val, y_train, y_val = train_test_split(
        X_train_val, y_train_val,    test_size=val_size/(train_size+val_size),
        random_state=random_state, shuffle=shuffle)
    return X_train, X_val, X_test, y_train, y_val, y_test

X_train, X_val, X_test, y_train, y_val, y_test = train_validation_test_split(
    X_res, y_res, train_size=0.8, val_size=0.1, test_size=0.1, random_state=1)

rb=QuantileTransformer(output_distribution='uniform')

X_train=rb.fit_transform(X_train)

X_val=rb.transform(X_val)

X_test=rb.transform(X_test)
```

6. FEATURE SCALING

Feature scaling is done so that the variance of all the features are in the same range. I have used QuantileTransformer(output_distribution='uniform') for scaling.

```
X_train, X_val, X_test, y_train, y_val, y_test = train_validation_test_split(
    X_res, y_res, train_size=0.8, val_size=0.1, test_size=0.1, random_state=1)
```

```
rb=QuantileTransformer(output_distribution='uniform')
```

```
X_train=rb.fit_transform(X_train)
```

```
X_val=rb.transform(X_val)
```

```
X_test=rb.transform(X_test)
```

7. MODEL BUILDING AND MODEL CHOICE EXPLANATION

- I tried different ML models like Random Forests, Logistic Regression, Gradient boosting model, Decision Trees, SVM and KNN. All the models were not giving me desired results. Some models were underfitting like Random Forest and Logistic Regression and some models were overfitting like Decision Trees.
- Then I tried using all the open sourced powerful models like XgBoost, CatBoost and LightGBM.
- Finally, I found XGBoost giving me close to desired results. So, I stuck to this model.
- First, I used the baseline XGBoost model and then hyper-parametrised it with different parameters to get the optimum results.

BaseLine XGBoost Model

```
from xgboost import XGBClassifier
```

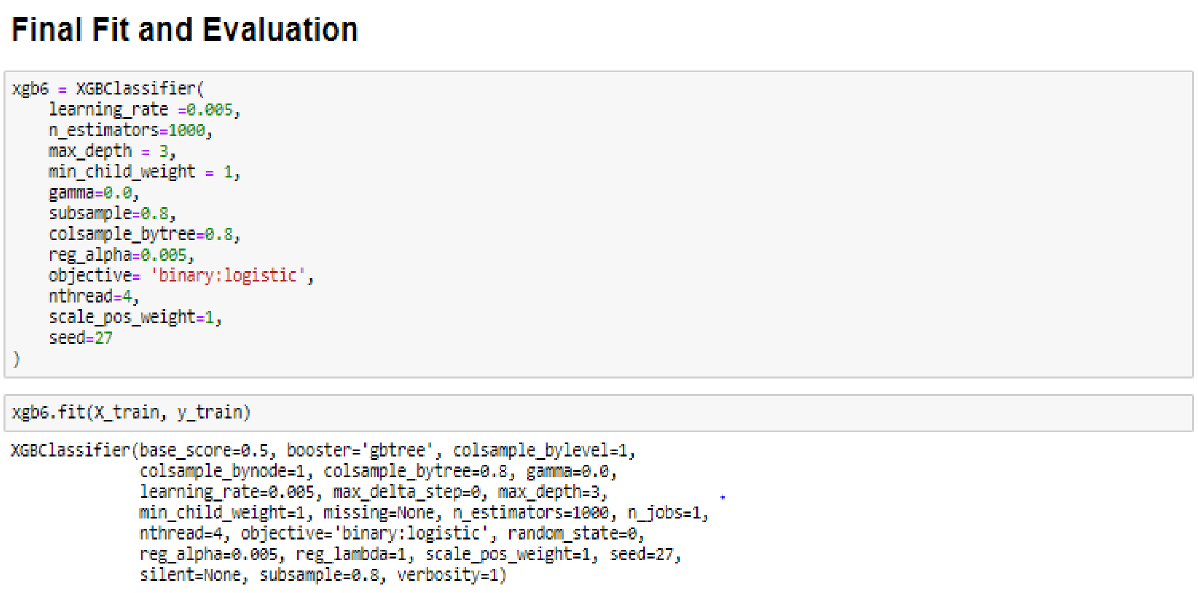
```
class_weight = {0: 4, 1: 5}
model = XGBClassifier(class_weight=class_weight)
model.fit(X_train, y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight={0: 4, 1: 5},
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

```
Results of various models on validation set.
```

```
y_pred = model.predict(X_val)
print(classification_report(y_val, y_pred))
print('accuracy', accuracy_score(y_val, y_pred))
roc_auc_score(y_val, y_pred)
```

```
              precision    recall  f1-score   support

           0       0.81      0.90      0.85        84
           1       0.84      0.69      0.76        59

    accuracy                           0.82       143
   macro avg       0.82      0.80      0.81       143
weighted avg       0.82      0.82      0.81       143

accuracy 0.8181818181818182

0.7998385794995965
```



Final Fit and Evaluation

```
xgb6 = XGBClassifier(
    learning_rate =0.005,
    n_estimators=1000,
    max_depth = 3,
    min_child_weight = 1,
    gamma=0.0,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_alpha=0.005,
    objective= 'binary:logistic',
    nthread=4,
    scale_pos_weight=1,
    seed=27
)
```

```
xgb6.fit(X_train, y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=0.8, gamma=0.0,
              learning_rate=0.005, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=1000, n_jobs=1,
              nthread=4, objective='binary:logistic', random_state=0,
              reg_alpha=0.005, reg_lambda=1, scale_pos_weight=1, seed=27,
              silent=None, subsample=0.8, verbosity=1)
```

The most important features are loan_term and insurance type.

8. KEY TAKEWAYS FROM THE CHALLENGE

- Always visualize the dataset to get the insights before applying any ML model to it.
- Try different models. Never stick to one model.
- For classification problems, examine and understand the ratio of the majority classes and the minority classes.