GLOSSARY

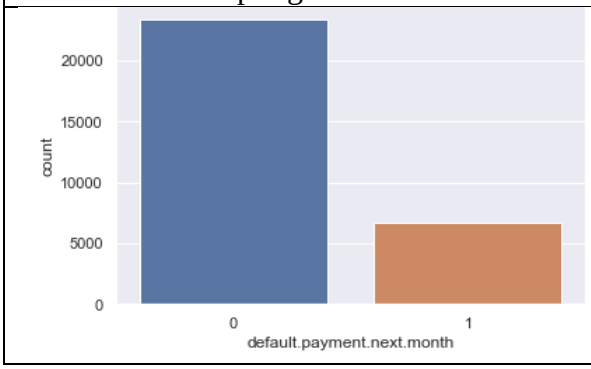
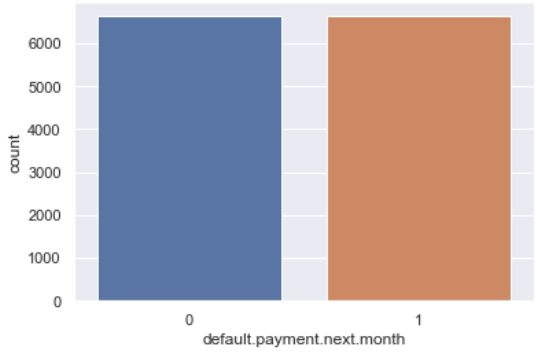
- **Binary classification problem**: Classifying the elements of a set into two groups.
- **Features**: Predictors or the independent values of the dataset.
- **Z-Score**: Measure of how many standard deviations below or above the mean.
- **Variance Inflation Factor**: Measure of the amount of Multicollinearity.
- **Multicollinearity**: Two or more features in a multiple regression model that are highly linearly related.
- **Sigmoid Function**: A Mathematical Function.
- **High-Dimensional**: Dataset where number of features can exceed the number of observations.
- **Over Fitting Issues**: A condition where a statistical model begins to describe the random error.
- **Credit Score**: A measure of an individual's ability to pay back the borrowed amount
- **Accuracy**: Total number of correct predictions in the total number of predictions.
- **Precision**: Ratio between the True Positives and all the Positives.
- **Recall**: Measure of our model correctly identifying True Positives.
- **F1-score**: Harmonic mean of the Precision and Recall.
- **ROC**: Summarises the trade-off between the true positive rate and false positive rate.
- **Lambda**: Regularization rate.
- **Priors**: Probability of an event.
- **Distributions**: Distribution provides a parameterized mathematical function that can be used to calculate the probability for any individual observation from the sample.
- **Hyperparameters**: Hyperparameters are parameters whose values are set prior to the start of the learning process.
- **Normal Distribution** : Gaussian distribution.
- **Empirical Values**: The class prior probabilities are the class relative frequencies in Y.
- **Undersampling**: Technique used to adjust the class distribution of a data set.
- **False Positive**: Outcome where the model incorrectly predicts the positive class.
- **Feature Engineering**: Process of extracting features can be used to improve the performance of machine learning algorithms.
- **SMOTE**: (Synthetic Minority Over-sampling Technique) A type of over-sampling procedure that is used to correct the imbalances in the groups.
- **Ensemble**: Technique that combines several base models in order to produce one optimal predictive model.

Intermediate results

1) **Dataset Imbalance:**

- The Dataset was imbalanced with more zeros as compared to ones in the target output as shown in the figure below:

Figure- The Dataset before and after Undersampling

Before undersampling	After undersampling
	

- In attempting to overcome it, I tried undersampling, but it was a failed attempt as the output seemed more biased towards zeros as output and Recall for the models reduced. The results were as below:

Table- Output of the Models after Undersampling

Models	Accuracy	Precision	Recall	F1 Score
Logistic Regression	68.4078	68.4621	68.58	68.5229
Naive Bayes	61.6022	61.3473	63.6225	62.4642

- Hence I decided against an undersampling of the data.

2) Logistic Regression

a) Using 'fitclinear' to find the best value of lambda and Regularization method(code as below):

```
Mld=fitclinear(xtrain,ytrain, 'Learner', 'logistic', 'OptimizeHyperparameters',...
    {'Lambda', 'Regularization'},'HyperparameterOptimizationOptions',...
    struct('Kfold', 10, 'AcquisitionFunctionName','expected-improvement-plus'));
%result -the model gave lambda value (0.0014031) and  Regularization-ridge
```

Table- Result of the model with lambda value=0.0014031 and Regularization='ridge'

Models	Accuracy	Precision	Recall	F1 Score
LR	77.3556	0.5000	0	0

- The results were unexpected and negative in this case.

b)Using 'templateLinear' and 'fitcecoc' to find the best values for learning rate and Regularization method (code as below):

```
Learning_rate=[0.0001,0.0003,0.001,0.003,0.01,0.03,0.1,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0];
Regularisation={'lasso','ridge'};
for i=1:length(Learning_rate)
    for j=1:length(Regularisation)
        rng default
        t=templateLinear('Learner','logistic','LearnRate',Learning_rate(i),...
        'Regularization',char(Regularisation(j)));
        Mdl=fitcecoc(x,y,'Learners',t,'CrossVal','on');
        klosses(i,j)=kfoldLoss(Mdl);

    end
end
klosses;
```

- The results however were unfruitful in determining the best learning rate and regularisation method.

3) Naïve Bayes

- Code shown as below:

```
rng('default')
Mdl_1 = fitcnb(xtrain,ytrain,'ClassNames',{'0','1'},'OptimizeHyperparameters','all',...
    'HyperparameterOptimizationOptions',struct('AcquisitionFunctionName',...
    'expected-improvement-plus'));
```

- The output was as below:

DistributionNames	Width	Kernel
kernel	7564.3	triangle

- The Model output for the above hyperparameters was as follows:

Table- Model output when DistributionNames=’kernel’ , Kernel=’triangle’,Width=’7564.3’

Models	Accuracy	Precision	Recall	F1 Score
NB	77.8967	66.6667	0.1507	0.3007

- The recall was very low so the model could not be used.

Description of main implementation choices

1) Logistic Regression

The use of Lasso Regularization shrinks the estimated coefficients of features to zero. We use lambda as our hyperparameter for penalising the features. In our MATLAB code we iterate through 50 values of lambda to find the best coefficients.We perform feature selection by identifying the non-zero coefficients.The 5 features selected in our case are LIMIT_BAL,PAY_0, PAY_2, PAY_3, PAY_4.

- LIMIT_BAL**:** Amount of given credit in NT dollars **(**it includes both the individual consumer credit and his/her family (supplementary) credit

History of past payment. tracking the past monthly payment records (from April to September,2005) as follows:

- PAY_0-the repayment status in September,2005(-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, … 8=payment delay for eight months, 9=payment delay for nine months and above)
- PAY_2-the repayment status in August,2005
- PAY_3- the repayment status in July,2005
- PAY_4- the repayment status in June,2005

the above mentioned features are important in detecting the credit card defaulters according to the results obtained by the model.

2) **Naïve Bayes**

- Intermediate results for Naïve Bayes as below:

```
%Using for-loops to find the best hyperparameter
DistributionName = {'Normal','kernel',};
prior={'empirical','uniform'};
for i=1:length(DistributionName)
    for j=1:length(prior)
        rng default
        Mdl = fitcnb(xtrain,ytrain,'ClassNames',{'0','1'},'Prior',char(prior(j)),...
                    'DistributionNames',char(DistributionName(i)));
        losses(i,j)=resubLoss(Mdl);

    end
end
losses;
```

- The losses observed were:

```
losses =

    0.3904    0.3897
    0.4307    0.4291
```

- The loss for losses for prior ='empirical' and Distribution ='Normal' was the lowest hence those hyperparameter values were used it in the model.