**Credit Card Default Prediction**

PRERNA RAI PRITHA GHOSH ZEHRA AHMED

## I. Introduction

A credit card default occurs when a borrower is unable to make timely payments, misses a payment or when payment is stopped or avoided. Risk management and mitigation is one of the most important aspects for any financial institution. Identifying customers with a higher risk of default can prevent institutions from issuing credit cards to unqualifying candidates. Classification of customers on the basis of their ability to pay off their credit card statements can help financial institutions save large amounts of money and prevent avoidable losses. To address this, the aim of this paper is to develop several classification models to predict credit card defaulters by using machine learning algorithms and study the performance of these models.

In addition to creating machine learning models and measuring their performances, we also aim to understand how the probability of default payment varies by categories amongst different demographic variables. We are also interested in understanding which variables are the strongest predictions of default payment.

The dataset used for this study is available on the UCI Machine Learning Repository. It consists of payment data from a bank in Taiwan from 2005. In the early 2000s, credit card issuers in Taiwan faced a lot of credit card debt crisis with peaking delinquency rates. In order to increase market shares, banks in Taiwan over-issued cash and credit cards to unqualified applicants.

## II. Related Work

1. ***"A Deep Neural Network (DNN) based classification model in application to loan default prediction" by: Bayraci, Selçuk and Orkun Susuz***

   This article applied a neural network model to two distinct real-world credit datasets (loan performance data and loan application data) containing characteristics of the loan clients in a medium-sized Turkish commercial bank. Their results found that the accuracy of the neural network model increases with the size of the dataset, implying that the neural network models may yield better results than regression-based models in terms of balanced accuracy, Type I error and Type II error metrics. On the other hand, the DNN model underperformed against the logistic regression and SVM models in the smaller dataset. Thus, linear classification models might be preferred in small sample datasets due to simplicity of their implementation. Furthermore, it is also known that deep learning algorithms are hard to implement and require a rigorous process of hyper-parameter tuning. Thus, deep learning based classification models are not always the solution, especially for datasets which have relatively small dimensions.

2. ***"An Investigation of Credit Card Default Prediction in the Imbalanced Datasets" : by Alam, Talha Mahboob, et al.***

   This article seeks to classify 3 credit card default datasets from Taiwan, South Germany, and Belgium. To deal with imbalanced datasets, techniques such as oversampling and undersampling are employed, along with min - max data normalization and PCA data transformation in addition to various machine learning models. The results on imbalanced datasets show the accuracy of 66.9% on Taiwan clients credit dataset, 70.7% on South German clients credit dataset, and 65% on Belgium clients credit dataset. Conversely, the results using the proposed methods significantly improved the accuracy to 89% on Taiwan clients credit dataset, 84.6% on South German clients credit dataset, and 87.1% on Belgium clients credit dataset. The results show that the performance of classifiers is better on the balanced dataset as compared to the imbalanced dataset. It is also observed that the performance of data oversampling techniques are better than undersampling techniques.

3. ***"Loan Default Prediction on Large Imbalanced Data Using Random Forests" by: Zhou, Lifeng and Hong, Wang***

   This article deals with a large dataset of loan default predictions in which the data is highly skewed. The authors implement an improved random forests approach which employs weighted majority votes in tree aggregation. The weights assigned to each tree in the forest are based on OOB(out-of-bag) errors. Also, due to the fact that random forests can be paralleled in nature, the foreach package is employed in the statistical software R to make random forest parallelable and greatly reduce the learning time. In order to make predictions, they proposed two ways: one is cost sensitive learning which incorporates class weights into the random forests classifier, and the other is by using over-sampling methods with the minority class and/or downsampling with the majority one to balance the original data. Experiments show that the weighted majority approach in tree aggregation improves random forest performance in terms of overall accuracy and balanced accuracy and also outweighs many classifiers such as SVM, KNN and C4.5. It was also proven that sample size-tuning balanced random forest and SMOTE-based balanced approach get similar results in dealing

with the imbalanced problem. Results showed that random forests with parallelism can greatly reduce the learning time of random forests and this technique should be considered as a standard practice on learning large datasets.

## III. Exploratory Data Analysis

### A. Data Acquisition

This dataset has been acquired from UCI Machine Learning Repository. It consists of payment data from 2005 from a bank in Taiwan. The dataset has 24 features, with a mix of numerical as well as categorical features. There are a total of 30,000 instances, out of which approximately 6,600 are defaulters. The response variable for our classification models is a binary variable, with an outcome of 1 in case of defaulters and 0 for non-defaulters.

### B. Data Exploration & Cleaning

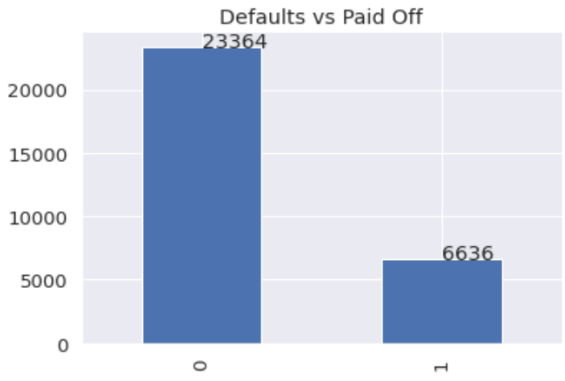
We begin our study with exploring the individual features in the dataset in order to understand the trends in the data, if any and to understand the behavior of the individual variables.

#### 1) Missing Values Analysis

We have verified that our dataset does not consist of any missing values.

#### 2) Distribution of Target Variable

The summary statistics of our target variable show that the dataset is skewed and imbalanced. Out of 30,000 observations, only 22.12% belong to the class of interest.



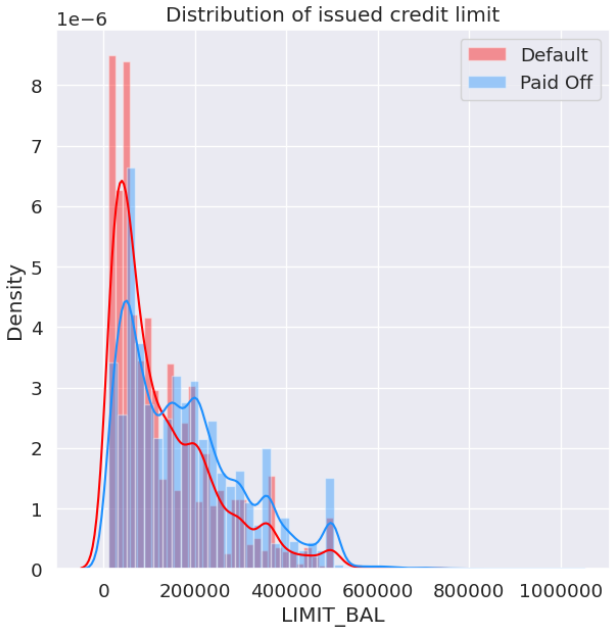

**Graph 1: Target Variable Distribution**

#### 3) Distribution of Issued Credit Limit

The summary statistics of the issued credit limit field are shown below.

```
count      30000.000000
mean      167484.322667
std       129747.661567
min        10000.000000
25%        50000.000000
50%       140000.000000
75%       240000.000000
max      1000000.000000
Name: LIMIT_BAL, dtype: float64
```

**Graph 2 : Summary Statistics of Credit Limit**




**Graph 3: Distribution of issued credit limit**

We can see from the distribution that most of the defaulters have an issued credit limit of less than 100k, which is less than the average credit limit.

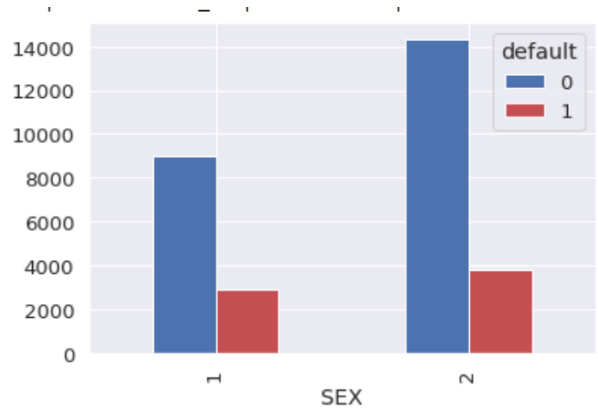
We have also performed the t-test for difference in means. The t-test quantifies the difference between the arithmetic means of the two samples. A p-value larger than a chosen threshold (in our case 5%), indicates that our observation is not so unlikely to have occurred by chance. Therefore, we do not reject the null hypothesis of equal population means. If the p-value is smaller than our threshold, then we have evidence against the null hypothesis of equal population means.

```
P-value is :
1.3022439532597397e-157
```

In this case, the p-value is less than the alpha value of 0.05. Therefore, we can reject the null hypothesis and conclude that this variable is a significant predictor of default.

#### 4) Distribution of Sex

We can see from the distribution below that there are more females (SEX=2) than males (SEX=1) in our dataset. The proportion of defaulters is more in males than in females.



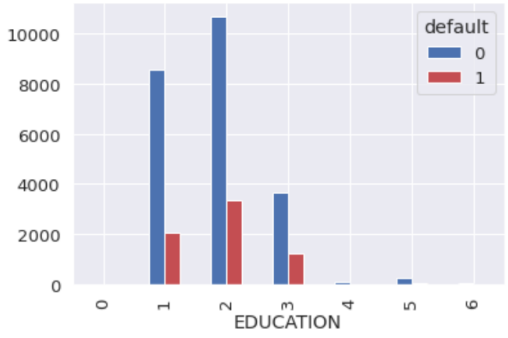

**Graph 4: Distribution of sex and default**

We have also performed the chi-square test of independence, which is used as a measure of independence of different categories of a population.

```
p value:
4.944678999412044e-12
```

Since the p-value is less than the alpha value of 0.05, we can reject the null hypothesis and conclude that this variable is a significant predictor of default.

#### 5) Distribution of Education




**Graph 5: Distribution of education and default**

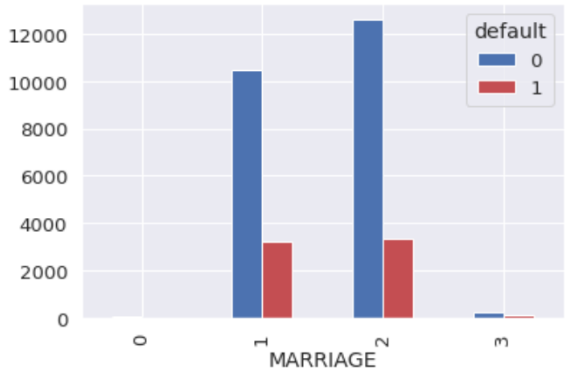
Most of the clients in the dataset have a university level education (EDUCATION=2), followed by graduate school level (EDUCATION=1) and then high school level (EDUCATION=3).

In order to make analysis easier, we have grouped EDUCATION=0,4,5,6 into EDUCATION=4 (Others).

```
p-value:
1.4950645648106153e-34
```

We have then used the chi-square test for independence to conclude that education level and default have a significant relationship, based on the p-value.

#### 6) Distribution of Marriage



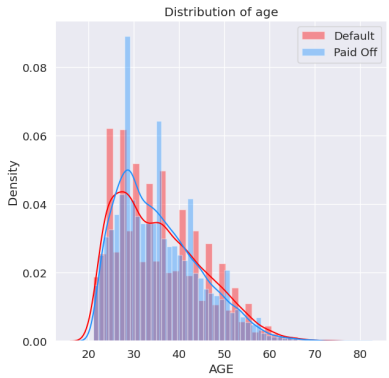

**Graph 6:Distribution of marriage and default**

The dataset is mainly divided into two groups - Married (MARRIAGE=1) or Single (MARRIAGE=2). We have grouped the other values in this column into the singles group.

We have also performed the chi-square test and concluded that there exists a significant relationship between marriage and default.

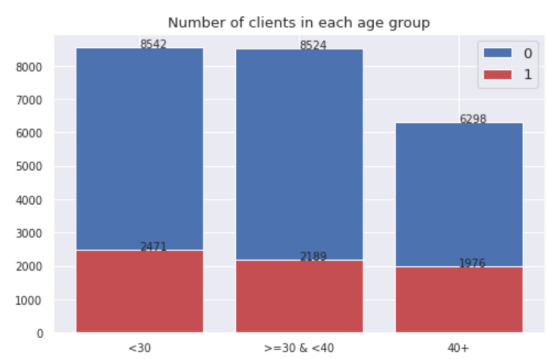
```
P-value:
2.7017310027696824e-07
```

### 7) Distribution of Age




**Graph 7 : Distribution of age**

We have grouped the data into three groups in order to treat age as a categorical variable rather than a continuous variable.



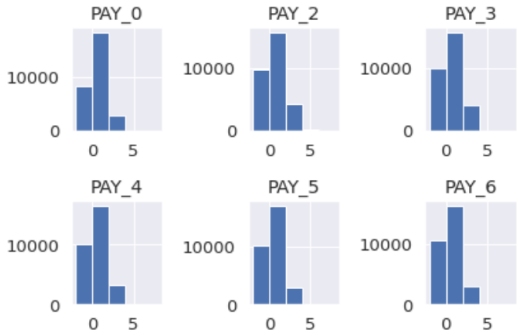

**Graph 8: Distribution of age group and default**

```
P-value:
6.021593890462918e-08
```

We then performed the chi-square test and determined that age and default have a significant relationship.

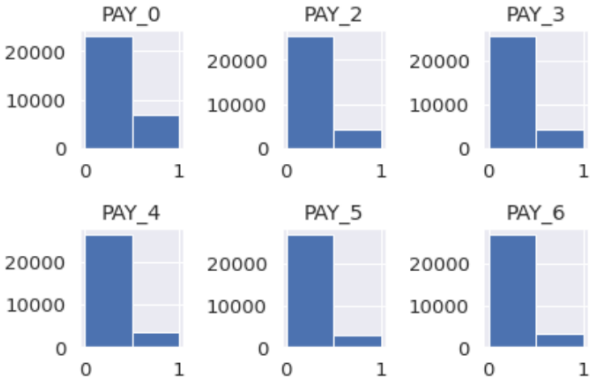
### 8) Distribution of PAY_0 to PAY_6

These columns indicate if the credit card payment was made on time or whether there was a delay. Values less than 0 indicate that payments were made on time, while values more than 0 indicate that there were delays.




**Graph 9: Distribution of PAY_0 to PAY_6**

We have grouped the values in these columns to be binary - zero if payments were made on time and 1 for late payments.




**Graph 10: Distribution after processing the data**

After the grouping, we can see that the proportion of payments made on time is much higher than the proportion of delays.

### 9) Correlation between the payment variables

We have plotted the correlation matrix for the different payment related variables.

| | PAY_0 | PAY_2 | PAY_3 | PAY_4 | PAY_5 | PAY_6 |
|---|---|---|---|---|---|---|
| PAY_0 | 1.000000 | 0.668176 | 0.430527 | 0.371030 | 0.348587 | 0.309867 |
| PAY_2 | 0.668176 | 1.000000 | 0.625247 | 0.478977 | 0.443047 | 0.397391 |
| PAY_3 | 0.430527 | 0.625247 | 1.000000 | 0.625660 | 0.482382 | 0.434247 |
| PAY_4 | 0.371030 | 0.478977 | 0.625660 | 1.000000 | 0.662637 | 0.496443 |
| PAY_5 | 0.348587 | 0.443047 | 0.482382 | 0.662637 | 1.000000 | 0.662683 |
| PAY_6 | 0.309867 | 0.397391 | 0.434247 | 0.496443 | 0.662683 | 1.000000 |

| | BILL_AMT1 | BILL_AMT2 | BILL_AMT3 | BILL_AMT4 | BILL_AMT5 | BILL_AMT6 |
|---|---|---|---|---|---|---|
| BILL_AMT1 | 1.000000 | 0.951484 | 0.892279 | 0.860272 | 0.829779 | 0.802650 |
| BILL_AMT2 | 0.951484 | 1.000000 | 0.928326 | 0.892482 | 0.859778 | 0.831594 |
| BILL_AMT3 | 0.892279 | 0.928326 | 1.000000 | 0.923969 | 0.883910 | 0.853320 |
| BILL_AMT4 | 0.860272 | 0.892482 | 0.923969 | 1.000000 | 0.940134 | 0.900941 |
| BILL_AMT5 | 0.829779 | 0.859778 | 0.883910 | 0.940134 | 1.000000 | 0.946197 |
| BILL_AMT6 | 0.802650 | 0.831594 | 0.853320 | 0.900941 | 0.946197 | 1.000000 |

| | PAY_AMT1 | PAY_AMT2 | PAY_AMT3 | PAY_AMT4 | PAY_AMT5 | PAY_AMT6 |
|---|---|---|---|---|---|---|
| PAY_AMT1 | 1.000000 | 0.285576 | 0.252191 | 0.199558 | 0.148459 | 0.185735 |
| PAY_AMT2 | 0.285576 | 1.000000 | 0.244770 | 0.180107 | 0.180908 | 0.157634 |
| PAY_AMT3 | 0.252191 | 0.244770 | 1.000000 | 0.216325 | 0.159214 | 0.162740 |
| PAY_AMT4 | 0.199558 | 0.180107 | 0.216325 | 1.000000 | 0.151830 | 0.157834 |
| PAY_AMT5 | 0.148459 | 0.180908 | 0.159214 | 0.151830 | 1.000000 | 0.154896 |
| PAY_AMT6 | 0.185735 | 0.157634 | 0.162740 | 0.157834 | 0.154896 | 1.000000 |

**Graph 11: Correlation between variables**

We can see that several fields are highly correlated and should not be used directly for modeling otherwise they would lead to biased results. We have to do some feature engineering and derive new columns based on the data in order to use these variables effectively.

### C. Feature Engineering

In this section, we derive some new variables based on the dataset.

#### 1) Number of months of payment delays

We use the PAY_0 to PAY_6 columns, which indicate whether or not the credit card payment was made on time or whether there was a delay in payment. We used this to make a new variable denoting how many times a customer has delayed payments.

```
count    30000.000000
mean         0.834200
std          1.554303
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max          6.000000
```

**Graph 12: Summary Statistics**

The summary statistics of this variable indicate that on an average, there is no payment delay. However, there are instances where there have been payment delays for every month.

We have also performed the t-test for difference in means to verify that there is a significant relationship between the number of payment delays and defaults.

#### 2) Average Credit Utilization

Since the BILL_AMT1 to BILL_AMT6 fields, which denote the amount of credit card bill a customer has every month are highly correlated, we decided to make a feature called Average Credit Utilization, which is the average of bill amount by the available credit limit each month.

```
count    30000.000000
mean         0.373048
std          0.351890
min         -0.232590
25%          0.029997
50%          0.284834
75%          0.687929
max          5.364308
Name: avg_credit_utilization, dtype: float64
```

**Graph 13: Summary Statistics**

The summary statistics of this variable indicate that on an average, the credit utilization is ~37%.

We have also performed the t-test for difference in means to verify that there is a significant relationship between the average credit utilization and default.

```
P-value is :
1.3022439532597397e-157
```

#### 3) Paid Off Percentage

We used the PAY_AMT1 to PAY_AMT6 and BILL_AMT1 to BILL_AMT6 variables to compute the Paid Off percentage. This is a ratio of the total amount of credit card bill paid off and the total amount of expenses incurred on the credit card.

```
count    30000.000000
mean         0.380941
```

**Graph 14: Summary Statistics**

The average percentage of credit card bills paid off is ~38%.

## IV. DATA CLEANING

Before using our data to build prediction models, we have to perform some data cleaning steps. Standardization of a dataset is a common requirement for many machine learning estimators, which may behave badly if the individual features do not more or less look like standard normally distributed data.

We have normalized the numeric features using the StandardScaler() function from the sklearn package. It standardizes the features by removing the mean and scaling to unit variance.

The standard score of a sample `x` is calculated as:

$$z = (x - u) / s$$

where u is the mean of the training samples and s is the standard deviation.

We have grouped the AGE variable into bins instead of treating it as a continuous variable. We have also grouped some of the values in the EDUCATION and MARRIAGE columns, as described in the previous section. We have also removed the highly correlated variables, instead using some new derived variables for our prediction models. We have manipulated the PAY_0 to PAY_6 columns and then derived a new column - Number of months of payment delays. We have also removed the Bill Amounts and Paid Amounts and instead used the Average Credit Utilization and Paid Off Percentage.

We have also performed one-hot encoding on the categorical variables in order to convert them into numerical before using them in our prediction models.

## V. Model Introduction

### 1) K-Nearest Neighbors Model

The K-nearest neighbors algorithm is a supervised machine learning algorithm often used in classification settings. It works on the simple assumption that similar things are always in close proximity. Therefore, it classifies an unknown item by looking at "k" of its already classified, nearest neighbors by calculating the majority votes from the nearest neighbors that have similar attributes.

For distance metrics, we use the Euclidean distance.

$$d(x, x') = \sqrt{(x_1 - x_1')^2 + \ldots + (x_n - x_n')^2}$$

The input x gets assigned to the class with the largest probability.

$$P(y = j|X = x) = \frac{1}{K} \sum_{i \in \mathcal{A}} I(y^{(i)} = j)$$

It is a lazy learning algorithm because it does not have a training phase, but rather memorizes the training dataset. All classifications are delayed till computation. It is a non-parametric method because irrespective of the size of the dataset, the only unknown parameter is k.

We chose to implement KNN as our first prediction model because it is easy to explain and simple to understand, yet extremely powerful. It does not require any assumptions about the data distribution, unlike several other algorithms.

### 2) Logistic Regression Model

Logistic Regression is a classification algorithm used to assign observations to a discrete set of classes. It transforms its output using the logistic sigmoid function to return a probability value.

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

Formula of a sigmoid function | Image: Analytics India Magazine

The hypothesis of logistic regression is as follows -

$Z = \beta_0 + \beta_1 X$

$h\Theta(x) = \text{sigmoid}(Z)$

i.e. $h\Theta(x) = 1/(1 + e^{-(\beta_0 + \beta_1 X)})$

$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

The cost function of logistic regression can be represented as -

$$J(\theta) = -\frac{1}{m} \sum \left[ y^{(i)} \log(h\theta(x(i))) + \left(1 - y^{(i)}\right) \log(1 - h\theta(x(i))) \right]$$

The cost function is optimized by Gradient Descent. In order to minimize the cost function, we have to run the gradient descent function on each parameter.

$$\theta j := \theta j - \alpha \frac{\partial}{\partial \theta j} J(\theta)$$

Want $\min_\theta J(\theta)$:

Repeat {

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all $\theta_j$)

}

### 3) Support Vector Machine Model

The objective of the support vector algorithm is to find a hyper-plane in n-dimensional space (n=number of features) that distinctly classifies the data points. Our objective is to find the plane that has the maximum margin, that is, that maximum distance between data points of both classes. Maximizing the margin provides some reinforcement so that future data points can be classified with more confidence.

The loss function that helps maximize the margin is hinge loss.

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases} \qquad c(x, y, f(x)) = (1 - y * f(x))_+$$

Hinge loss function (function on left can be represented as a function on the right)

We also add a regularization parameter to the cost function to balance the margin maximization and loss. Now, our cost function looks like this -

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Loss function for SVM

Next, we find the gradients by taking partial derivatives with respect to the weights.

$$\frac{\delta}{\delta w_k} \lambda \parallel w \parallel^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} \left(1 - y_i \langle x_i, w \rangle\right)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Gradients

When there is no misclassification, we only have to update the gradient from the regularization parameter -

$$w = w - \alpha \cdot (2\lambda w)$$

Gradient Update — No misclassification

In case of misclassifications, we perform the gradient update by including the loss along with the regularization parameter -

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

Gradient Update — Misclassification

### 4) Naive Bayes Model

The Naive Bayes algorithm's underlying assumption is that the predictors are independent of each other. This is a big assumption because in real-world scenarios, there is often at least some correlation between variables. Despite this, the Naive-Bayes algorithm has been proven to perform really well in classification problems. Simultaneously, it is a fast algorithm since it scales easily to include many predictors without having to handle multidimensional correlations.

$$P(C_k \mid x) = \frac{P(C_k) * P(x \mid C_k)}{P(x)}$$

where:

- **P(C|x)** is the posterior probability of class C (target viable) given the predictor x (attribute / independent variable);
- **P(C)** is the prior probability of class C;
- **P(x|C)** is the likelihood, which is the probability of the predictor x given class C;
- **P(x)** is the prior probability of the predictor x;
- **Little k** is just the notation to distinguish between different classes as you would have at least 2 separate classes in the classification scenario (e.g., spam / not-spam, red ball / black ball).

Gaussian Naive Bayes is an adaptation of Naive Bayes for continuous attributes. When dealing with continuous data, a typical assumption is that each class's continuous values are distributed according to a normal distribution.

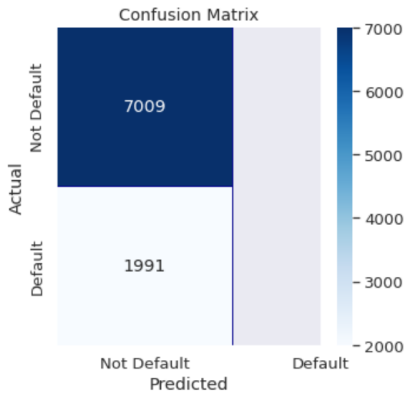
While we can use frequencies to calculate the probability of occurrence for categorical variables, we cannot use the same approach for continuous variables. Instead, we first need to calculate the mean and variance for x in each class and then calculate P(x|C) using the following formula:

$$P(x_i \mid C_k) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i - \mu_k)^2}{2\sigma_k^2}}$$

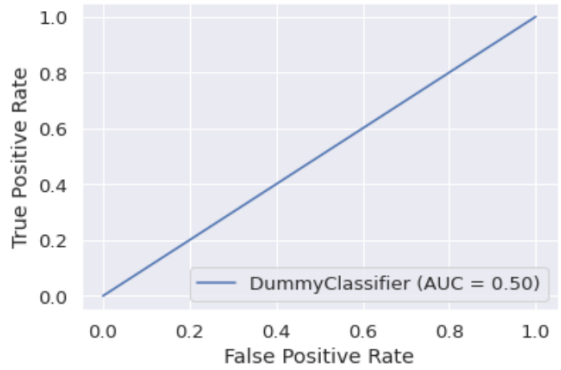
## VI. Experimental Results

**Baseline Model Results and Discussion:**

Before building our main prediction model, we first built a baseline model so that we have a point of comparison. For our problem, our baseline model predicts the class value most common in our dataset, which is "Not Default". Since our model has 77.87% observations of the majority class, that is what our accuracy score is on the test data and the AUC value is 0.5.



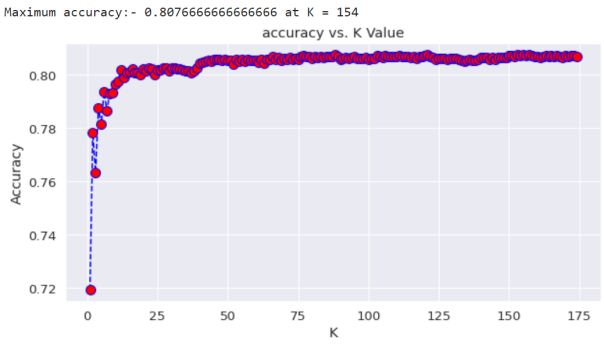

**Graph 15: Confusion Matrix - Baseline Model**




**Graph 16: ROC curve - Baseline Model**

**KNN Model Results and Discussion:**

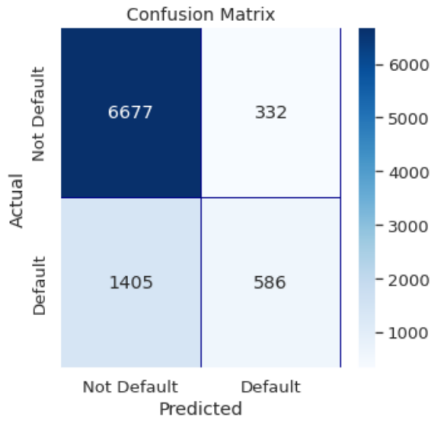
In order to find the optimal k-value, we chose a range of values between 1 and 175 (square root of the number of observations) and plotted the error rate against each value of k. The optimal k value was computed to be k=154 and the accuracy was 80.77%, which is approximately 3% higher than the baseline model.




**Graph 17: Finding the optimal k-value**

We then performed 10-fold cross validation, which gave us a chance to test the model on multiple splits so that we can get a better idea of how well the model performs on test data. The mean accuracy after performing cross validation was 80.27%, which is approximately 3% higher than the baseline model accuracy.

```
[0.79516667 0.80083333 0.79866667 0.80783333 0.811     ]
cv_scores mean:0.8027000000000001
```



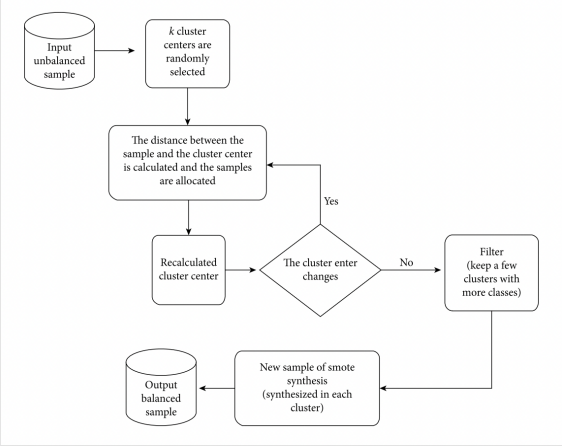

**Graph 18: Confusion Matrix - KNN**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.96 | 0.89 | 7009 |
| 1 | 0.65 | 0.27 | 0.38 | 1991 |
| accuracy | | | 0.81 | 9000 |
| macro avg | 0.74 | 0.61 | 0.63 | 9000 |
| weighted avg | 0.79 | 0.81 | 0.77 | 9000 |

**Graph 19: Classification Report - KNN**

However, we can see that the model did not perform very well in classifying defaults since it is a minority class. For our problem, the prediction of the minority class (defaults) is more important.

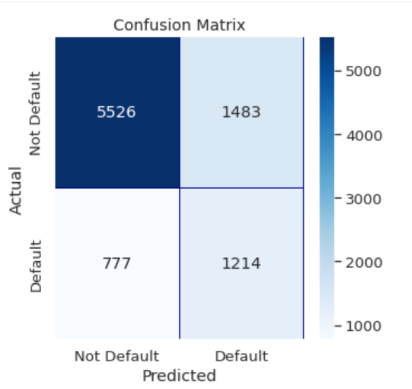
One way to tackle the problem of imbalance in the dataset is by oversampling the examples in the minority class. The most simple technique to do that is by simply duplicating examples from the minority class in the training dataset prior to fitting the model. This can balance the class distribution but does not provide any additional information to the model.




**Graph 20: k-means SMOTE algorithm flowchart**

A better way is to synthesize new examples from the minority class. The most widely used approach to do this is by applying the Synthetic Minority Oversampling Technique (SMOTE) on the training set. This technique first selects a minority class instance x at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors y at random and connecting x and y to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances x and y.

We now see the prediction on the test data set -

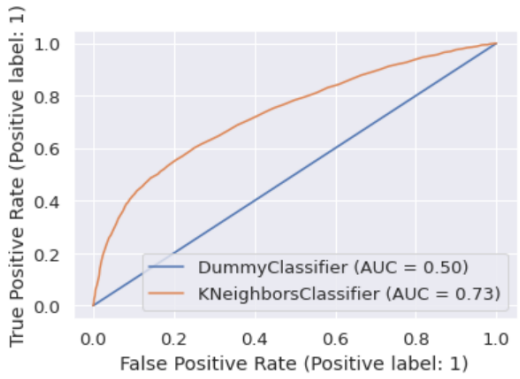



**Graph 21:Confusion Matrix-KNN- after SMOTE**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.86 | 0.80 | 0.83 | 7009 |
| 1 | 0.44 | 0.55 | 0.49 | 1991 |
| accuracy | | | 0.75 | 9000 |
| macro avg | 0.65 | 0.68 | 0.66 | 9000 |
| weighted avg | 0.77 | 0.75 | 0.76 | 9000 |

**Graph 22:Classification Report - KNN - after SMOTE**

We can see that by applying the SMOTE technique, the model is able to classify the defaults much better.

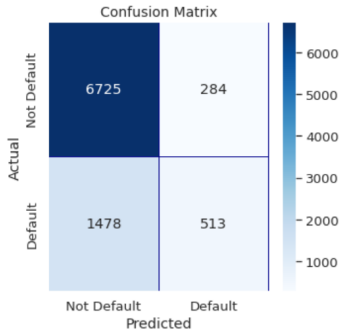



**Graph 23: ROC Curve - KNN after SMOTE**

The AUC value of the KNN model is 0.73 on the test set, which is a vast improvement over the performance of the baseline model.

**<u>Logistic Regression Model Results and Discussion:</u>**

We built two logistic regression models, model 1 - without SMOTE sampling and model 2 - with SMOTE sampling. We experimented with both L1 and L2 regularization and found the results to be slightly better with L2. We split the training set into train and validation sets (75:25) in order to find the best C-value , which for model 1 was 0.001 with an accuracy of 77.04%. The C-value here represents the amount of penalty. Here is the confusion matrix below generated on the test data -

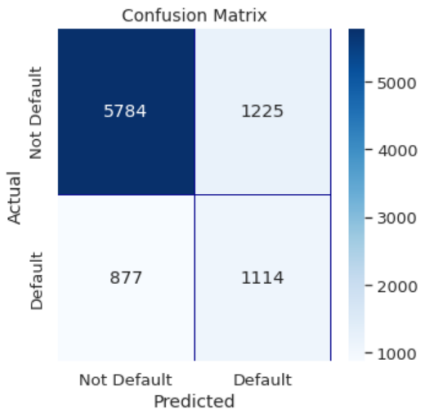



**Graph 24: Confusion Matrix -Logistic Regression**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.83 | 0.85 | 7009 |
| 1 | 0.48 | 0.56 | 0.51 | 1991 |
| accuracy | | | 0.77 | 9000 |
| macro avg | 0.67 | 0.69 | 0.68 | 9000 |
| weighted avg | 0.78 | 0.77 | 0.77 | 9000 |

**Graph 25: Classification Report - Logistic Regression**

For the model 2 with SMOTE oversampling on the train data, the best C-value was also 0.001 and we achieved an accuracy of 76.87% on the test set. Here is the confusion matrix below -

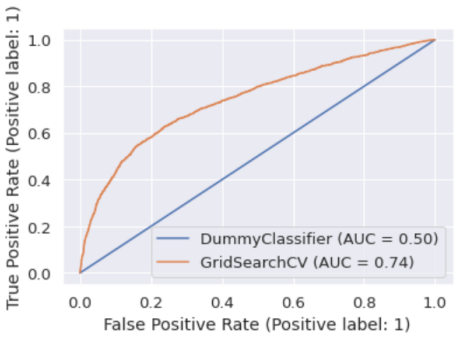



**Graph 26: Confusion Matrix - Logistic Regression - after SMOTE**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.87 | 0.83 | 0.85 | 7009 |
| 1 | 0.48 | 0.56 | 0.51 | 1991 |
| accuracy | | | 0.77 | 9000 |
| macro avg | 0.67 | 0.69 | 0.68 | 9000 |
| weighted avg | 0.78 | 0.77 | 0.77 | 9000 |

**Graph 27: Classification Report - Logistic Regression after SMOTE**

We can see that the results from model 1 and model 2 are almost identical in terms of F1 score and accuracy, however the model with SMOTE sampling classifies defaults more accurately, although it also increases the number of false positives.




**Graph 28: ROC Curve - Logistic Regression after SMOTE**

The AUC value of the Logistic Regression model is 0.74, which is an improvement over both the baseline model as well as the KNN model. The F1 score has also improved from the best KNN model.

**SVM Model Results and Discussion:**

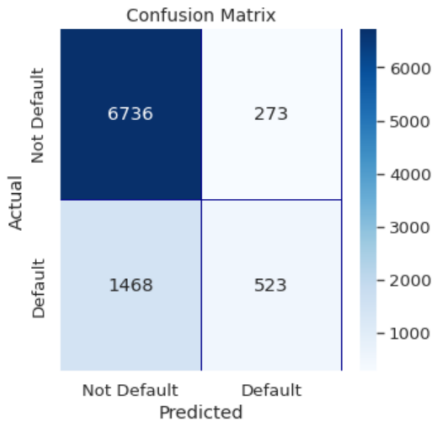
In order to find the best value for the regularization parameter, we split our training data into train and validation sets (75:25). We then run the model for different values of C with a linear kernel and check the classification error on the validation dataset.

```
Overall classification error for 0.125 = 0.19733333333333336
Overall classification error for 0.25 = 0.19866666666666666
Overall classification error for 0.5 = 0.19733333333333336
Overall classification error for 1 = 0.19771428571428573
Overall classification error for 2 = 0.19733333333333336
Overall classification error for 4 = 0.19485714285714284
Overall classification error for 8 = 0.1980952380952381
Overall classification error for 16 = 0.1980952380952381
Overall classification error for 32 = 0.1980952380952381
Overall classification error for 64 = 0.19866666666666666
Overall classification error for 128 = 0.19904761904761903
Overall classification error for 256 = 0.1994285714285714
Overall classification error for 512 = 0.19904761904761903
```

**Graph 29: Determining value of C using Cross Validation**

The misclassification error is least for C=4.

Next, we ran the model on the entire training data for this value of C, with kernel='linear'. We then tested out the model on the test data and we can see the results in the confusion matrix below -



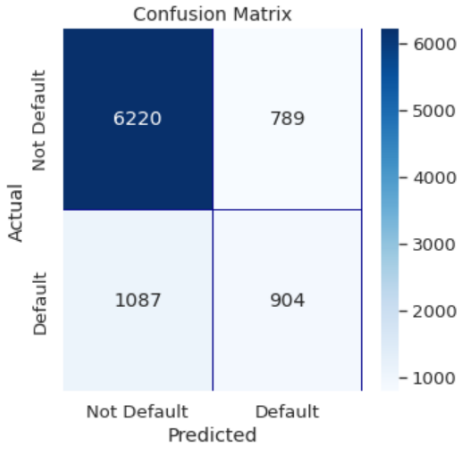

**Graph 30: Confusion Matrix - SVM - Linear Kernel**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.82 | 0.96 | 0.89 | 7009 |
| 1 | 0.66 | 0.26 | 0.38 | 1991 |
| accuracy | | | 0.81 | 9000 |
| macro avg | 0.74 | 0.61 | 0.63 | 9000 |
| weighted avg | 0.78 | 0.81 | 0.77 | 9000 |

**Graph 31: Classification Report - SVM with Linear Kernel**

We can see that this model does not classify defaults well.

Therefore, we ran the model again with the same parameters, after applying SMOTE sampling on the train data. The confusion matrix for the unseen test data is below -




**Graph 32: Confusion Matrix - SVM with Linear Kernel - after SMOTE**

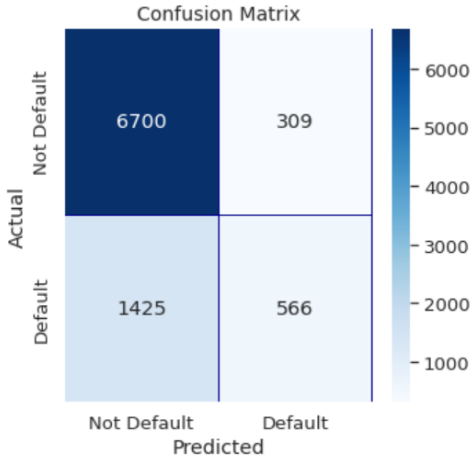
We can see from the confusion matrix generated on the test data that the performance of the model at classifying defaults has improved from before. However, there is room for more improvement since more than half the defaults are being classified as non-defaults.

Next, we tried the model with kernel='rbf'. We have again divided the training set into train and validation (75:25) and tried out different values of C and gamma - where the C parameter trades off the correct classification of training examples against maximization of the decision function's margin and the gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.

```
Overall classification error for C = 0.125 and gamma =  10 = 0.2211428571428572
Overall classification error for C = 0.125 and gamma =  100 = 0.2211428571428572
Overall classification error for C = 0.25 and gamma =  0.01 = 0.20304761904761903
Overall classification error for C = 0.25 and gamma =  0.1 = 0.19904761904761903
Overall classification error for C = 0.25 and gamma =  1 = 0.20438095238095233
Overall classification error for C = 0.25 and gamma =  10 = 0.22076190476190471
Overall classification error for C = 0.25 and gamma =  100 = 0.2211428571428572
Overall classification error for C = 0.5 and gamma =  0.01 = 0.20323809523809522
Overall classification error for C = 0.5 and gamma =  0.1 = 0.19866666666666666
Overall classification error for C = 0.5 and gamma =  1 = 0.200952380952381
Overall classification error for C = 0.5 and gamma =  10 = 0.2209523809523809
Overall classification error for C = 0.5 and gamma =  100 = 0.22190476190476194
Overall classification error for C = 1 and gamma =  0.01 = 0.20323809523809522
Overall classification error for C = 1 and gamma =  0.1 = 0.1982857142857143
Overall classification error for C = 1 and gamma =  1 = 0.20304761904761903
Overall classification error for C = 1 and gamma =  10 = 0.21904761904761905
Overall classification error for C = 1 and gamma =  100 = 0.2251428571428571
Overall classification error for C = 2 and gamma =  0.01 = 0.20323809523809522
Overall classification error for C = 2 and gamma =  0.1 = 0.19923809523809521
```

**Graph 33: Determining value of C using Cross Validation**

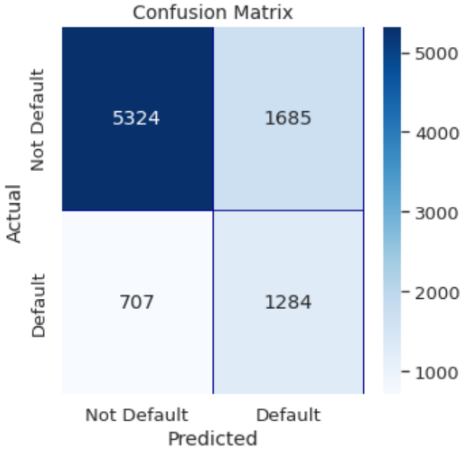
The best parameters are C=1 and gamma=0.1. We then run the model with these parameters on the entire training data. The results on the test set are displayed below -




**Graph 34: Confusion Matrix - SVM with RBF Kernel**

We can see that SVM using the rbf kernel did not perform significantly better than SVM using linear kernel.

We now ran the model again using the same parameters, after applying SMOTE sampling on the train set. The results on the unseen test data are documented below -

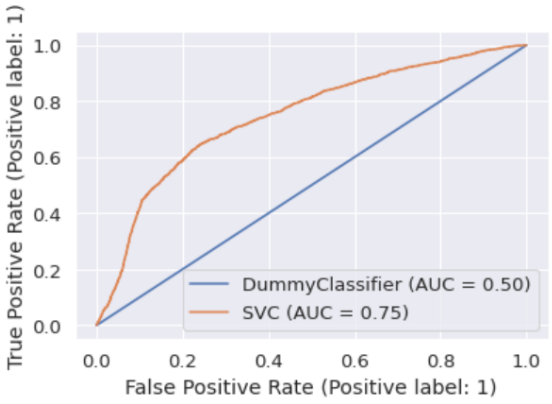



**Graph 35: Confusion Matrix - SVM with RBF Kernel - after SMOTE**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.88 | 0.76 | 0.82 | 7009 |
| 1 | 0.43 | 0.64 | 0.52 | 1991 |
| accuracy | | | 0.73 | 9000 |
| macro avg | 0.66 | 0.70 | 0.67 | 9000 |
| weighted avg | 0.78 | 0.73 | 0.75 | 9000 |

**Graph 36: Classification Report - SVM with Radial Kernel - after SMOTE**

This model is a significant improvement over the others and does a better job at classifying defaults.



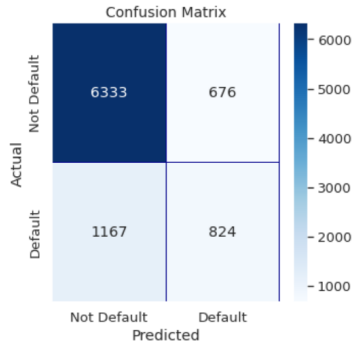

**Graph 37: ROC Curve - SVM with Radial Kernel after SMOTE**

The AUC value of the best SVM model is 0.75 on the test data, which is better than the KNN and Logistic Regression models implemented previously. There is also an improvement in the F1 score compared to the other models.

**Naive Bayes Model Results and Discussion:**

We built the first model without using any oversampling/undersampling techniques and without using smoothing.

We can find the results on the test set below -




**Graph 38: Confusion Matrix - Naive Bayes**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.90 | 0.87 | 7009 |
| 1 | 0.55 | 0.41 | 0.47 | 1991 |
| accuracy | | | 0.80 | 9000 |
| macro avg | 0.70 | 0.66 | 0.67 | 9000 |
| weighted avg | 0.78 | 0.80 | 0.78 | 9000 |

**Graph 39: Classification Report - Naive Bayes**

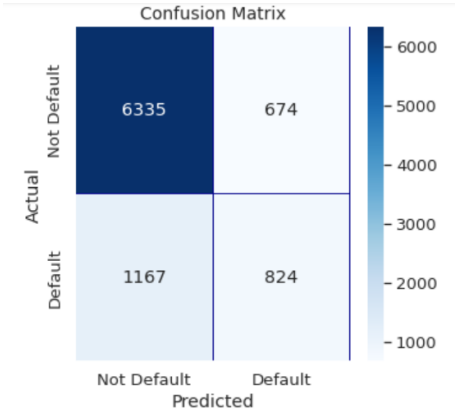
We then built another model applying **Laplace Smoothing**. Laplace Smoothing is introduced to solve the problem of zero probability.

We split the training data into train and validation splits (75:25) and run the model for different values of lambda (the smoothing parameter) -

```
Score for 0.0001 = 0.796
Score for 0.001 = 0.8011428571428572
Score for 0.01 = 0.8051428571428572
Score for 0.1 = 0.8055238095238095
Score for 1 = 0.7961904761904762
Score for 10 = 0.7788571428571428
Score for 100 = 0.7788571428571428
Score for 1000 = 0.7788571428571428
```

**Graph 40: Determining the lambda value using Cross Validation**

The best accuracy was obtained for lambda=0.1. Therefore, we run our model on the entire training data with this value in the smoothing parameter. After training the model, we tested the model on the test data and generated the results shown below -



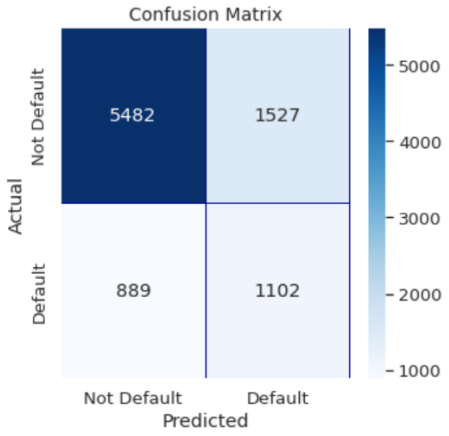

**Graph 41: Confusion Matrix - Naive Bayes with smoothing**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.90 | 0.87 | 7009 |
| 1 | 0.55 | 0.41 | 0.47 | 1991 |
| accuracy | | | 0.80 | 9000 |
| macro avg | 0.70 | 0.66 | 0.67 | 9000 |
| weighted avg | 0.78 | 0.80 | 0.78 | 9000 |

**Graph 42: Classification Report - Naive Bayes with smoothing**

However, we can see that applying smoothing did not make any difference on the performance of our model.

We then oversampled the training data and built another model.

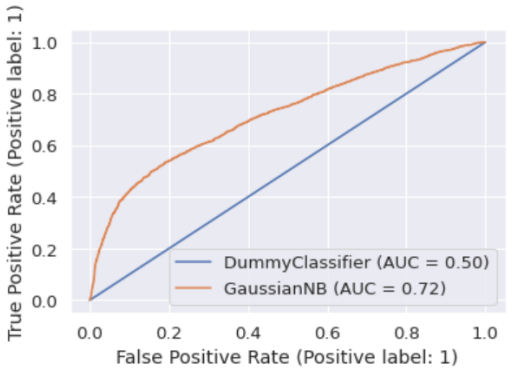



**Graph 43: Confusion Matrix - Naive Bayes - after SMOTE**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.86 | 0.78 | 0.82 | 7009 |
| 1 | 0.42 | 0.55 | 0.48 | 1991 |
| accuracy | | | 0.73 | 9000 |
| macro avg | 0.64 | 0.67 | 0.65 | 9000 |
| weighted avg | 0.76 | 0.73 | 0.74 | 9000 |

**Graph 44: Classification Report - Naive Bayes after SMOTE**

We can see from the metrics above generated on the test dataset that this model does a better job at classifying defaults than the previous models. However, it also increases the number of false positives.



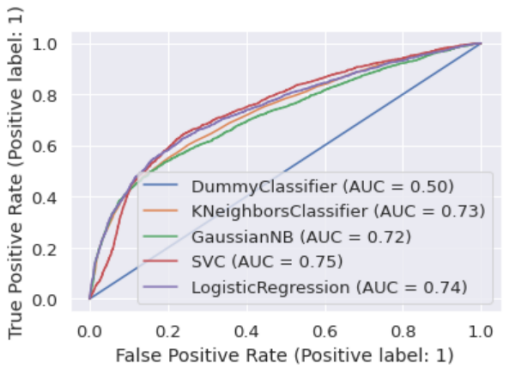

**Graph 45: ROC Curve - Naive Bayes - after SMOTE**

The AUC value of the Naive Bayes model is 0.72 on the test data, which is not better than any of the algorithms we have implemented.

We then applied 10-fold cross validation and found that the mean cross-validation score is 0.66, while the score from our model is 0.68. Therefore, we can conclude that the model is independent of the particular folds we used for training.

**Model Comparison:**

We have implemented four algorithms - KNN, Logistic Regression, SVM and Naive Bayes.




**Graph 46: ROC Curve - Model Comparison**

In terms of AUC, SVM performed the best (AUC=0.75), followed by Logistic Regression, KNN and Naive Bayes. The evaluation metrics on the test dataset have been compiled below -

| Model | F1-score | AUC |
|---|---|---|
| SVM - RBF Kernel - SMOTE | 0.52 | 0.75 |
| Logistic Regression -SMOTE | 0.51 | 0.74 |
| SVM - Linear Kernel - SMOTE | 0.49 | 0.75 |
| KNN - SMOTE | 0.49 | 0.73 |
| Naïve Bayes - Laplace Smoothing - SMOTE | 0.48 | 0.72 |
| Naïve Bayes | 0.47 | 0.72 |
| Naïve Bayes - Laplace Smoothing | 0.47 | 0.72 |
| SVM - RBF Kernel | 0.39 | 0.69 |
| SVM - Linear Kernel | 0.38 | 0.74 |
| KNN | 0.38 | 0.73 |
| Logistic Regression | 0.37 | 0.74 |

**Graph 47: Evaluation Metrics - Model Comparison**

**Variable Importance**

| features | coeff | \|coeff\| |
|---|---|---|
| payment_delay_months | 0.6913 | 0.6913 |
| LIMIT_BAL | (0.2302) | 0.2302 |
| Single | (0.0922) | 0.0922 |
| Female | (0.0888) | 0.0888 |
| Others | (0.0874) | 0.0874 |
| >=30 & <40 | (0.0624) | 0.0624 |
| <30 | (0.0538) | 0.0538 |
| 40+ | 0.0489 | 0.0489 |
| Male | 0.0411 | 0.0411 |
| Married | 0.0341 | 0.0341 |
| paid_off | (0.0142) | 0.0142 |
| University | (0.0098) | 0.0098 |
| GraduateSchool | 0.0082 | 0.0082 |
| HighSchool | (0.0058) | 0.0058 |
| avg_credit_utilization | 0.0050 | 0.0050 |

The most important variables generated by the best model are summarized above.

**Number of months of payment delays:** This is the most important predictor of default. Customers who have a greater number of delayed payments in the past are more likely to default.

**Credit Limit Balance:** This variable is negatively correlated to default, which means that the higher the Credit Limit Balance of a customer, the lower is the risk of default.

**Marital Status:** Single customers are less likely to default compared to married customers.

**Sex:** Female customers are less likely to default compared to male customers.

**Age**:Customers above 40+ are more likely to default.

**Paid-Off Percentage:** This variable is negatively correlated to default, which means that customers who pay off a higher percentage of their bill are less likely to default.

**Education:** Clients who have a Graduate School level of education are more likely to default.

**Average Credit Utilization:** As the percentage of credit utilization increases, the risk of default tends to increase.

## VII. Conclusion

**Key Insights:**

- Since our dataset is unbalanced with ~22% of observations belonging to the class of interest, we have to use oversampling techniques for our model to perform well.
- SVM handles outliers better as it derives maximum margin solution.
- A properly tuned rbf kernel performs slightly better than a linear kernel. However, they require more resources to train, with little improvement in predictive powers, therefore they may not be suitable for large datasets with large numbers of features.
- Naive Bayes does not perform very well for numeric features.

**Future Implications:**

- Optimize the models to maximize profit by changing the classification thresholds – with lower thresholds, the model allows more credit cards to be issued whereas with higher thresholds, the model is more conservative and will not issue credit cards unless there is a high probability that the customer will be able to repay the bill.
- We would like to experiment with other sampling techniques to understand if there is a difference in performance.

- We would like to experiment with other standardization techniques to see if it improves the performance of the model.
- We would like to incorporate other algorithms not covered in this class, like random forests. Based on the literature review above, random forest models can handle imbalances well by employing weighted majority votes in tree aggregation. Random forests can also be used to find variable importance so that we could see which variables contribute most to predicting default.
- Lastly, we would also like to explore unsupervised learning algorithms and analyze their performance as well.

## VIII. References


1) Alam, Talha Mahboob, et al. "An investigation of credit card default prediction in the imbalanced datasets." *IEEE Access* 8 (2020): 201173-201198.
2) Bayraci, Selçuk, and Orkun Susuz. "A Deep Neural Network (DNN) based classification model in application to loan default prediction." *Theoretical and Applied Economics* 22.4 (621), Winter (2019): 75-84.
3) Gandhi, Rohith. "Support Vector Machine — Introduction to Machine Learning Algorithms | by Rohith Gandhi." *Towards Data Science*, 7 June 2018, https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. Accessed 30 November 2021.
4) Harrison, Onel. "Machine Learning Basics with the K-Nearest Neighbors Algorithm." *Towards Data Science*, 10 September 2018, https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. Accessed 30 November 2021.
5) Li, Susan. "Building A Logistic Regression in Python, Step by Step." *Towards Data Science*, 28 September 2017, https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8. Accessed 30 November 2021.
6) Prabhakaran, Selva. "How Naive Bayes Algorithm Works? (with example and full code) | ML+." *Machine Learning Plus*, 4 November 2018, https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/. Accessed 30 November 2021.
7) Srivastava, Tavish. "K Nearest Neighbor | KNN Algorithm | KNN in Python & R." *Analytics Vidhya*, 26 March 2018, https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/. Accessed 30 November 2021.
8) Ying Chen, Ruirui Zhang, "Research on Credit Card Default Prediction Based on k-Means SMOTE and BP Neural Network", *Complexity*, vol. 2021. vol. 2021, Article ID 6618841.
9) Yeh, I. C., & Lien, C. H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2), 2473-2480.
10) Zhou, Lifeng, and Hong Wang. "Loan default prediction on large imbalanced data using random forests." *TELKOMNIKA Indonesian Journal of Electrical Engineering* 10.6 (2012): 1519-1525.