Credit Card Default Detection

1st Yash Admuthe
Department of Mathematics
Stevens Institute of Technology
Hoboken, United States
yadmuthe@stevens.edu

2nd Nikhil Chouhan
Department of Mathematics
Stevens Institute of Technology
Hoboken, United States
nchouhan@stevens.edu

3rd Jerry Veembukattu Thomas
Department of Mathematics
Stevens Institute of Technology
Hoboken, United States
jveembuk@stevens.edu

***Abstract*—Credit cards allow users to pay for services and then pay the money later to the card issuer along with some payment fees. But there are often situations where customers are not able to pay back the money or misses the payment deadline which are called credit defaults. They can have a negative impact not only on the customer but also with the financial institutions who have issued the credit cards, causing them loss. The aim of the project is to help the financial institutions in studying the trends of customers from their data such as income, average spending and to predict whether the customers will default in future so that they can know before it happens and hence minimize the loss incurred due to credit defaults. We will be using Logistic Regression, K Nearest Neighbour, Random Forest, Artificial Neural Network, boosting algorithms - AdaBoost and XGBoost to build the model.**

I. Introduction

The purpose of this project is to train a model on the dataset generated by a financial institution. The dataset is about their existing customers with the details including history of whether they have defaulted or not. This model is then tested on the new customers who have joined and are applying for credit cards to determine whether they will default or not to decide on issuance of credit cards. This is achieved by developing prediction model using Machine Learning Algorithms with Python as the preferred language. Basically, three algorithms – Logistic Regression, K Nearest Neighbour, Random Forest, Artificial Neural Network, boosting algorithms - AdaBoost and XGBoost are considered and compared to obtain the best model. Also, as the data obtained is imbalanced as one class has fewer examples when compared to the other, undersampling is also performed to improve its accuracy.

II. Related Work

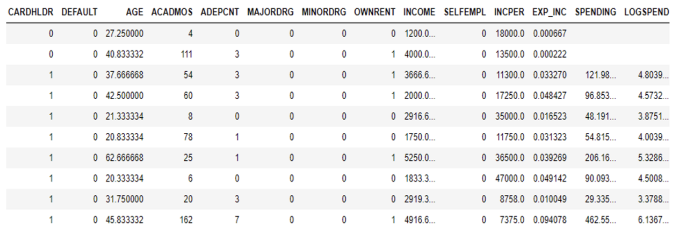
A study was done and published in IEEE regarding considering an imbalanced dataset to help commercial banks, financial organizations, loan institutes, and other decision-makers predict the credit card defaulters earlier. An imbalanced dataset was used as there can be situations where data is more for one class than the other. Real life datasets were collected from banks in Belgium, South Germany and Taiwan. It was shown that they were able to achieve higher accuracy when using GBDT algorithm and have improved the performance of the model using SMOTE based oversampling methods.

III. Our Solution

We are considering an imbalanced dataset to mimic the real life case where there can be situations where most of the data can lie in one class and only a fraction lies in the other class. We will be training the model using machine learning algorithms after performing undersampling so as to handle imbalanced dataset yet provide a higher accuracy. The features considered for training will be age, whether the person is employed or not, and if yes whether the person is self employed, number of dependents, yearly income and average credit card spending.

A. Description of Dataset

The dataset consists of 13,444 observations with 14 variables. It includes CARDHLDR (It is a dummy variable, which state 1 if application of credit card is accepted, 0 if not), AGE (age of consumers in years plus twelfths of a year), ACADMOS (state the number of months living at current address), ADEPCNT (the number of dependents of consumers), MAJORDRG (number of major derogatory reports), MINORDRG (Number of minor derogatory reports), OWNRENT (which consumers owns or rent a house, 1 if owns their home, 0 if rent), INCOME (monthly income of clients (divided by 10,000), SELFEMPL (state that the consumer is self-employed or not - 1 if self-employed, 0 if not), INCPER (state the Income divided by number of dependents), EXPINC (it is a Ratio of monthly credit card expenditure to yearly income), SPENDING (Average monthly credit card expenditure only for CARDHOLDER = 1), LOGSPEND (it is the logarithmic value of spending), DEFAULT (defaulted(1) or not(0) – the target variable(observed when CARDHLDR = 1, 10,499 observations) , if the there is a default on credit card or not).



CARDHLDR	DEFAULT	AGE	ACADMOS	ADEPCNT	MAJORDRG	MINORDRG	OWNRENT	INCOME	SELFEMPL	INCPER	EXP_INC	SPENDING	LOGSPEND
0	0	27.250000	4	0	0	0	0	1200.0...	0	18000.0	0.000667		
0	0	40.833332	111	3	0	0	1	4000.0...	0	13500.0	0.000222		
1	0	37.666668	54	3	0	0	1	3666.6...	0	11300.0	0.033270	121.98...	4.8039...
1	0	42.500000	60	3	0	0	1	2000.0...	0	17250.0	0.048427	96.853...	4.5732...
1	0	21.333334	8	0	0	0	0	2916.6...	0	35000.0	0.016523	48.191...	3.8751...
1	0	20.833334	78	1	0	0	0	1750.0...	0	11750.0	0.031323	54.815...	4.0039...
1	0	62.666668	25	1	0	0	1	5250.0...	0	36500.0	0.039269	206.16...	5.3286...
1	0	20.333334	6	0	0	0	0	1833.3...	0	47000.0	0.049142	90.093...	4.5008...
1	0	31.750000	20	3	0	0	0	2919.3...	0	8758.0	0.010049	29.335...	3.3788...
1	0	45.833332	162	7	0	0	1	4916.6...	0	7375.0	0.094078	462.55...	6.1367...

Fig. III.1. Sample of the dataset

The source of this dataset is obtained from Kaggle. As this dataset has 14 features and there were some null value entries within the dataset, they have been removed and cleaned from the main dataset as they belong to the class which has higher number of samples and hence does not have impact if removed.

B. Machine Learning Algorithms

We are using following six algorithms:

1) Logistic Regression
2) K Nearest Neighbor (KNN)
3) Random Forest
4) Artificial Neural Network (ANN)
5) AdaBoost
6) XGBoost

At first, we implemented using Logistic Regression method. Logistic Regression is commonly used for classification as it outputs a value corresponding to probability of belonging to a given class and is ideal for this project as it works best when there is a binary output. Our data set has two target classes 'defaulted' and 'not defaulted' which we have encoded with 1 and 0 in the default column.

K Nearest Neighbor (KNN) algorithm is a supervised algorithms. Based on the most relevant features we will be using KNN to predict whether there is a default on credit card or not. This will be working according to the nearest neighbor.

Random Forest classification algorithm will be used. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. By developing multiple decision trees and considering the majority of the output values form these decision trees we will be able to correctly predict whether there is a default on credit card or not.

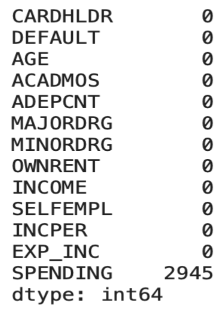
Artificial Neural Network (ANN) is a Machine Learning model inspired by the network of neurons found in a human brain. They consists of artifical neurons which receives multiple inputs and provide a single output which can then be shared to other neurons in the network. The outputs of the final set of neurons provides the output for the problem for which it was used. They are powerful and versatile and handle large and highly complex tasks. It is implemented in the project using Keras, a high-level Deep Learning API with the help of TensorFlow library.

Boosting which were called hypothesis boosting refers to the Ensemble method where a strong learner is obtained by combining several weak learners. The idea commonly follwed by most of the boosting methods is to train predictors sequentially where each will correct its predecessor. Several boosting algorithms are available nowadays, AdaBoost and Gradient Boosting being the popular ones.

AdaBoost, short for Adaptive Boosting pays more attention to the training instances where the predecessor has underfitted which makes the new predictors to focus more on hard cases. This is done by increasing the weight of misclassified training instances for the new predictors. Gradient Boosting also sequentially adds the predictors to the ensemble. But unlike AdaBoost, Gradient Boosting tries to fit the new predictor to the residual errors made by the previous predictor. XGBoost which stands for Extreme Gradient Boosting is an optimized implementation of Gradient Boosting and aims to be fast, scalable and portable.

C. Implementation Details

The project aims at gaining insight and making predictions by training model using machine learning algorithms - Logistic Regression, K - Nearest Neighbour, Random Forest, ANN, AdaBoost and XGBoost. Before starting with training the model, data preprocessing has to be done. At first, we have to verify if there are missing values present in the dataset. Analyzing the dataset, we get the data regarding missing values as shown in the figure.



```
CARDHLDR       0
DEFAULT        0
AGE            0
ACADMOS        0
ADEPCNT        0
MAJORDRG       0
MINORDRG       0
OWNRENT        0
INCOME         0
SELFEMPL       0
INCPER         0
EXP_INC        0
SPENDING    2945
dtype: int64
```

Fig. III.2. Fields with count of missing values

It could be observed that the attribute SPENDING has 2945 missing values and since we do not want to miss data by deleting rows, we have to fill the missing data with values. To decide on which method to follow for filling the missing values, the distribution of spending is checked. As the feature is found to be left skewed as shown in figure below, we will be using median to fill the missing values and in cases of irrelevant data, they are dropped from the dataset.

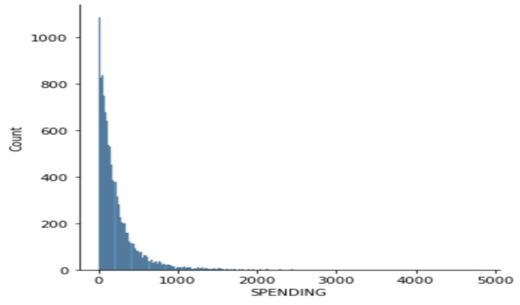


Fig. III.3. Distribution of data in Spending attribute

For some features, the datatypes had to be changed so that they can be used for training the model. As missing values in SPENDING column was updated, existing LOGSPEND column had to be dropped. After dropping, log operation is performed on the updated SPENDING column and the result is stored within the same column. After this step, graph was plotted to understand the distribution of data among the two classes - 0 for not defaulted and 1 for defaulted and is given in the below figure.

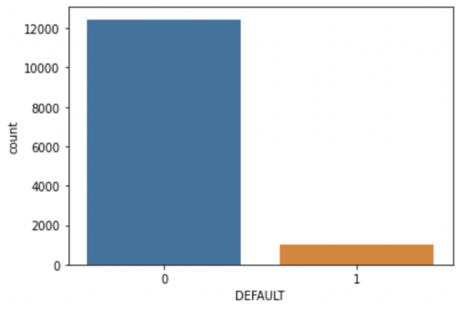


Fig. III.4. Count of Defaulters in dataset

Also, further analysis is done to see the relationship between different attributes and the target attribute and some important graphs are added below. In the graph given below, we could see the distribution of people as self employed or not along with data of people who have defaulted in both categories. We can see the number of defaulters are higher in the case when they are not self employed. Even though the data collected is mostly from people who are not self employed, we cannot have reject any assumption that whether a person is self employed or not, do not have any relation on being a defaulter.

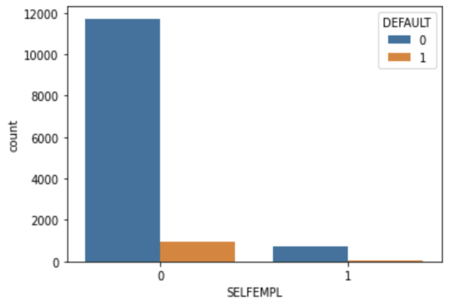


Fig. III.5. Employment Type vs Defaulters

Coming to the part of income of people, we are dividing them into four groups. As from the plot given below, we could see that the number of defaulters are more in the category of people who have an income of up to 2500. Number of defaulters is also significant in the next group of people who have income in the range 2500 to 5000 while it is less noticeable in graph for the people who are earning more than 5000.

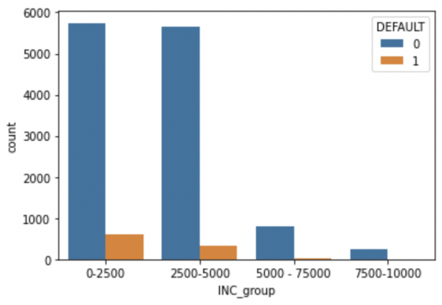


Fig. III.6. Income vs Defaulters

When coming to the age, we are again dividing the age in multiples of 25 to see the distribution of data. From the graph given below, we could see that most of the data is collected from people whose age lies in the range 25 to 50 while less data is taken from people who are 75 years or older. It could also be observed, that number of defaulters are significant in group where people are 50 years old or less.

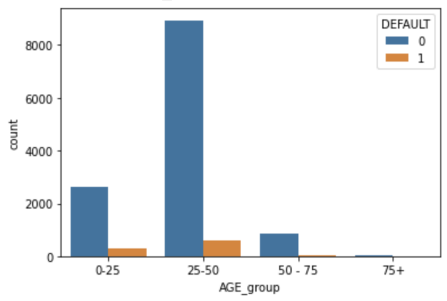
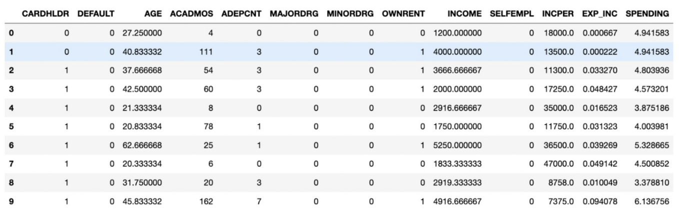


Fig. III.7. Age vs Defaulters

We understood that the data is imbalanced as the target class has uneven distribution. This results in the classfier being biased towards the prediction and we will not get an ideal classifier. To handle this, we performed undersampling. Undersampling is a resampling method where all the data in the minority class are kept and size of the majority class is decreased. The main advantage of undersampling is that we can correct imbalanced data to reduce the risk of our analysis or machine learning algorithm skewing toward the majority.



	CARDHLDR	DEFAULT	AGE	ACADMOS	ADEPCNT	MAJORDRG	MINORDRG	OWNRENT	INCOME	SELFEMPL	INCPER	EXP_INC	SPENDING
0	0	0	27.250000	4	0	0	0	0	1200.000000	0	18000.0	0.000667	4.941583
1	0	0	40.833332	111	3	0	0	1	4000.000000	0	13500.0	0.000222	4.941583
2	1	0	37.666668	54	3	0	0	1	3666.666667	0	11300.0	0.033270	4.803936
3	1	0	42.500000	60	3	0	0	1	2000.000000	0	17250.0	0.048427	4.573201
4	1	0	21.333334	8	0	0	0	0	2916.666667	0	35000.0	0.016523	3.875186
5	1	0	20.833334	78	1	0	0	0	1750.000000	0	11750.0	0.031323	4.003981
6	1	0	62.666668	25	1	0	0	1	5250.000000	0	36500.0	0.039269	5.328665
7	1	0	20.333334	6	0	0	0	0	1833.333333	0	47000.0	0.049142	4.500852
8	1	0	31.750000	20	3	0	0	0	2919.333333	0	8758.0	0.010049	3.378810
9	1	0	45.833332	162	7	0	0	1	4916.666667	0	7375.0	0.094078	6.136756

Fig. III.8. Dataset after Data Preprocessing

It could also be observed in the dataset that the features of the input data have different ranges and some are larger than others. This can cause issues for machine learning models, especially for those models that use gradient descent as optimization technique. Hence, these models need the features to be scaled. Here, we have used standardization, a scaling technique where the values are centered around mean with a unit standard deviation. Standardization is used instead of normalization as it is not affected by outliers.

$$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$$

Fig. III.9. Formula for Standardization

Logistic Regression estimates the likelihood of an instance belonging to a given class. The logistic regression model computes the weighted sum of all the input features and passes the value through a logistic function called sigmoid function that outputs a number between 0 and 1 and estimated probability is calculated. If the estimated probability is more than 50%, the model predicts that the instance belongs to that class which is called the positive class, but if it is less than 50%, it does not and is called the negative class. In the project, the two classes means whether the person has defaulted or not. When we train the model by setting parameters solver as saga which defines the algorithm to use, penalty as l2, maximum iteration whcih defines number of iterations to converge as 50, the accuracy obtained is 84%.

K-Nearest Neighbour(KNN) is a supervised machine learning algorithm. By computing the distance between the test data and all of the training points, KNN tries to predict the proper class for the test data. Then choose the K number of points that are the most similar to the test data. The KNN method analyzes the likelihood of test data belonging to each of the 'K' training data classes, and the class with the greatest probability is chosen.

For the implementation, we are using KNeighborsClassififer class of the popular Python Machine Learning library scikit-learn. The initial accuracy obtained after training the model is 80%. Later, parameters of the class is modified which includes number of neighbors, algorithm to use for computing the nearest neighbors and power parameter which chooses either Euclidean distance or Minkowski distance for Minkowski metric. Choosing number of neighbors as 3 and brute force search as the algorithm, we were able to improve the accuracy of the model from 80% to 82.5%.

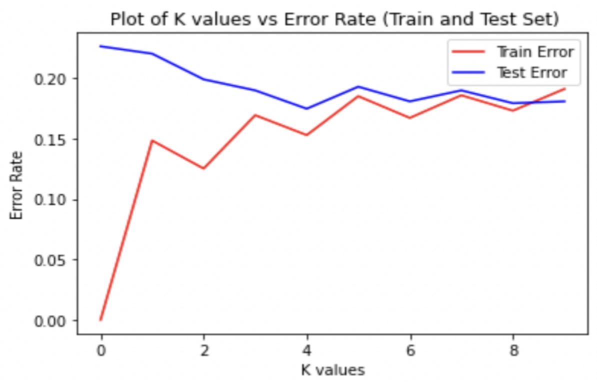


Fig. III.10. Variation in Train and Test error for different values of K

Of all the parameters available for trying out different combinations to improve accuracy, number of nearest neighbours denoted as K in the algorithm plays crucial role. Selecting lower values of K can lead to model overfitting the data or high variance which leads to lower training error but higher test error. Selecting higher values causes bias or underfitting which again leads to lower training error but higher test error. Value of K is considered optimal when test error stabilizes and test error is the lowest. This could be seen from the graph given above. Based on this, we have selected 3 as the optimal value.

Random Forest is an ensemble of Decision Trees, generally trained via the bagging method. Random Forest algorithm introduces extra randomness when growing trees. The prediction obtained at the end of training is more than that of the decision trees. It is because the output chosen by the majority of the decision trees becomes the final output of the random forest algorithm.

We will be using RandomForestClassifier class of scikit-learn library for training the model. Initially after training, we get an accuracy of 84%. Some of the parameters available for improving the accuracy of the model are the number of trees (*n_estimators*), which function to use for measuring quality of the split (*criterion*), maximum depth of the tree (*max_depth*), number of features to consider (*max_features*). By setting *max_depth* as 5, *n_estimators* as 700 and *criterion* as entropy, we were able to improve the accuracy of the model to almost 86%.

Artificial Neural Network are the subset of machine learning and are the heart of deep learning algorithms. They consists of a set of nodes which includes input layer, one or more hidden layers, and an output layer. Each neuron is connected to each other and have some weights. If the output of the node is greater than its specified threshold value, then the node is activated and this node then transfers data to the next level of layers. ANNs rely on training data to learn and improve their accuracy over time.

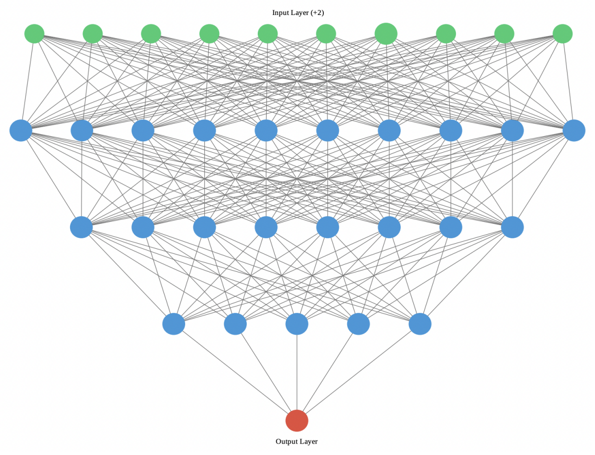


Fig. III.11. Model of the Implemented ANN

TensorFlow and Keras are the libraries we have used for implementing ANN in the project. TensorFlow is an end to end open source platform for machine learning while Keras provides Python interface for artificial neural networks.In our implementation, we have four layers which consist of an input layer, two hidden layers and one output layer. In the input layer, it accepts 12 inputs and has 10 nodes and the activation function used is Rectified Linear Unit (*ReLU*). In the second layer, there are eight nodes and again uses ReLU activation function.In the third layer, we are stepping down the nodes to 1 with ReLU as activation function. In the output layer, we have one node as we have only two classes to predict - either defaulter or not. We are using sigmoid function as the activation function in output node.

While implementing, we have are using Adam algorithm as the optimizer which is a stochastic gradient descent method used for optimization and is proven to be computationally efficient. The metric function is to defined which function to use to judge the model performance and we are using accuracy. We have also used BinaryCrossentropy class for loss function which computes cross-entropy loss between true and predicted labels. Another paramter is specifying the number of epochs. Epoch mean a single cycle where model goes through all the training set. If the number of epochs is higher than optimum value, it can lead to overfitting. The graph given below denotes how accuracy of train and test set varies over epochs. The accuracy obtained after 50 epochs is 87%.

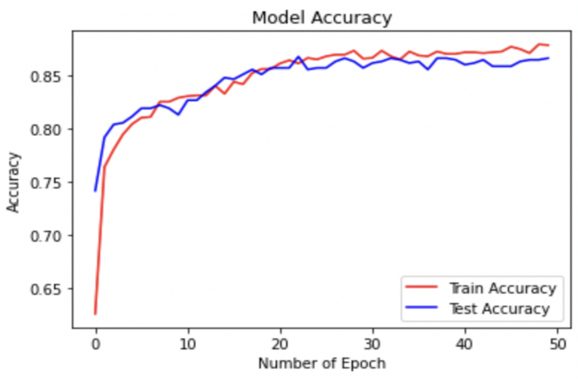


Fig. III.12. Variation in Train and Test error for different epochs

We have also implemented two boosting algorithms in our problem - Adaptive Boosting (AdaBoost) and XGBoost. As most of the machine learning models gives focus on having higher prediction by a single model, boosting algorithms tries to improve the prediction rate by training a sequence of weak models where each model compensate the weakness of its predecessor model.

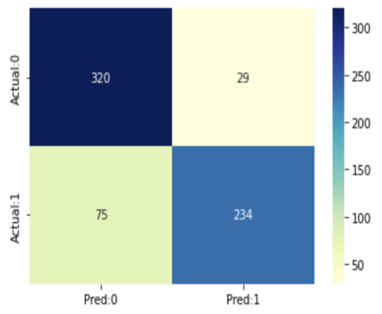
The first boosting algorithm we have implemented is AdaBoost. AdaBoost is an ensemble method which was developed for increasing the efficiency of binary classifiers and hence is an ideal algorithm for our problem. We are again using scikit-learn library to train the model using its AdaBoostClassifier class. It implements using AdaBoost-SAMME.R algorithm. SAMME which stands for Stage wise Additive Modeling, is a multi-class boosting and puts more weight on the misclassified data points than AdaBoost alone. SAMME.R provides probability estimates to update additive model and hence has shown faster convergence with lower testing error. We are also setting the maximum number at which boosting should be terminated (*n_estimators*) to 1250 and learning rate as 0.45, the model has an accuracy of 87%.

The second boosting algorithm used is XGBoost. It is an optimized model of Gradient Boosting decision tree algorithm providing speed and performance. XGBoost provides a parallel tree boosting also known as GBDT for solving problems in a fast and accurate way. For implementation, we are using XGBoost library from where we are importing the class XGBClassifier. At first we are using GridSearchCV class of scikit-learn to find out the best estimators for the model based on XGBoost algorithm. There we obtain that, we get the best prediction when we keep the number of estimators (*n_estimators*) as 140 and number of parallel threads to use (*nthread*) as 4. After fitting the model with these estimators and training the model, we could get an accuracy of almost 88%.

IV. Comparison

After implementation, accuracy of the models trained under all six algorithms are compared. It could be observed that XGBoost has the highest accuracy of 88%. Models of both AdaBoost and Artificial Neural Network have the secong highest accuracy with 87%, slighlty below that of XGBoost model. Still, it could be confirmed that models trained under ANN and both the boosting algorithms have a higher prediction on the imbalanced dataset. While Random Forest algorithm showed an accuracy of 86.5% performing as good as the other mentioned algorithms, Logistic Regression and KNN performed less, having prediction rate of 84% and 82% respectively.

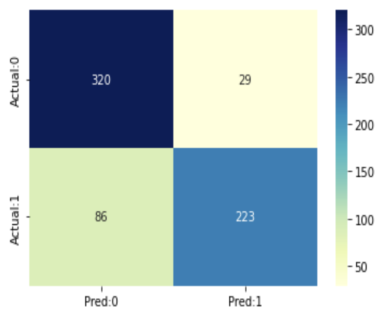
In the case of logistic regression, as shown in below figure, we could see that the model has a accuracy of 75.7% in correctly predicting the people who will be defaulting and an accuracy of 91.69% in predicting who will not be defaulting. This makes up to the total accuracy of 84% when the model uses logistic regression algorithm as discussed earlier.



```
          Pred:0  Pred:1
Actual:0     320      29
Actual:1      75     234
Percent of defaulters correctly predicted:  75.72815533980582
Percent of non-defaulters correctly predicted:  91.69054441260745
Accuracy of the test set:  0.8419452887537994
```

Fig. IV.1. Analysis of Logistic Regression

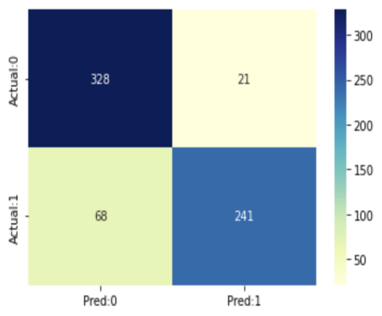
When coming to K-Nearest Neighbor, we can see from the figure given below that the model trained has an accuracy of 72% in predicting people who will be defaulting while in the case of people who will not be defaulting, the accuracy shown is 91.69%. Note that out of the six algorithms, KNN model has the lower accuracy rate in predicting people who will be defaulting. The given accuracy combines to give the total accuracy of 82.5% when the model is trained using K-Nearest Neighbor algorithm.



```
          Pred:0  Pred:1
Actual:0     320      29
Actual:1      86     223
Percent of defaulters correctly predicted:  72.168284789644
Percent of non-defaulters correctly predicted:  91.69054441260745
Accuracy of the test set:  0.8252279635258358
```

Fig. IV.2. Analysis of K-Nearest Neighbor

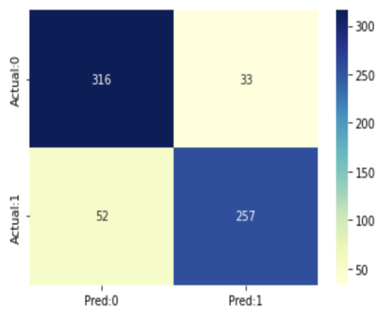
In the case of Random Forest algorithm, the model is able to predict with an accuracy of almost 78% the people who will be defaulting based on their data while the accuracy of predicting people who will not be defaulting is almost 94%. This is illustrated in the figure given below. Hence, the accuracy of the model trained using Random Forest algorithm is almost 86.5%.



```
          Pred:0  Pred:1
Actual:0     328      21
Actual:1      68     241
Percent of defaulters correctly predicted:  77.99352750809061
Percent of non-defaulters correctly predicted:  93.98280802292264
Accuracy of the test set:  0.8647416413373861
```

Fig. IV.3. Analysis of Random Forest

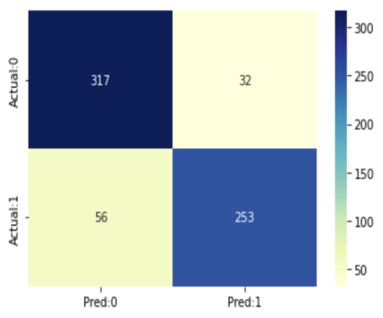
When coming to Artificial Neural Network, we could see from the figure given that the model has a higher accuracy rate of 83% in predicting people who will be defaulting based on their data. Also, note that the model has slightly less accuracy when compared to previous models in predicting people who will not be defaulting, which is 90.5%. Combined together, the model trained under ANN has an accuracy of 87%, making it one of the best algorithm to use falling just behind XGBoost algorithm.



```
          Pred:0  Pred:1
Actual:0     316      33
Actual:1      52     257
Percent of defaulters correctly predicted:  83.1715210355987
Percent of non-defaulters correctly predicted:  90.54441260744986
Accuracy of the test set:  0.8708206686930091
```

Fig. IV.4. Analysis of Artificial Neural Network

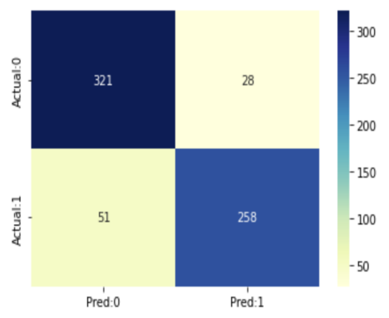
Boosting Algorithms were introduced to improve the accuracy of existing machine learning algorithms and hence we can expect the accuracy to be more when compared to algorithms mentioned before. From the figure, we could see that the model is showing an accuracy of almost 81.9% in predicting people who will be defaulting and has an accuracy of almost 90.8% when predicting people who will not be defaulting. Collectively taking, the model trained under AdaBoost algorithm shows an accuracy of almost 86.6%.



```
          Pred:0  Pred:1
Actual:0     317      32
Actual:1      56     253
Percent of defaulters correctly predicted:  81.87702265372168
Percent of non-defaulters correctly predicted:  90.83094555873924
Accuracy of the test set:  0.8662613981762918
```

Fig. IV.5. Analysis of AdaBoost

XGBoost is one of the most commonly used boosting algorithms as it has proven to be more accurate in predicting. This could be observed in the project as the model has an accuracy of almost 83.5% in predicting people who will be defaulting. When considering the case of people who will not be defaulting, the model is able to display an accuracy of almost 92%. At the end, the model shows the best accuracy when compared to all the other five algorithms by having an accuracy of almost 88%.



```
              Pred:0  Pred:1
Actual:0         321      28
Actual:1          51     258
Percent of defaulters correctly predicted:  83.49514563106796
Percent of non-defaulters correctly predicted:  91.97707736389685
Accuracy of the test set:  0.8799392097264438
```

Fig. IV.6. Analysis of XGBoost

The graph provided below gives a comparison of all the six machine learning algorithms used in the project with respect to their accuracy arranged according to their accuracy in descending order.

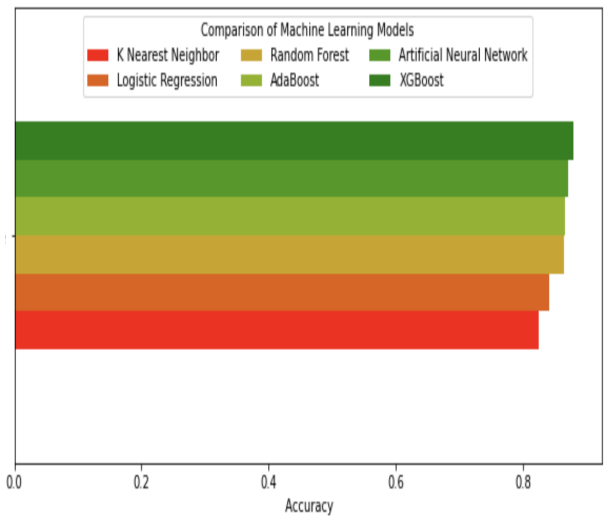


Fig. IV.7. Comparison of Machine Learning Models

V. Future Directions

The data we have used for the problem is an imbalanaced dataset where the number of defaulters is less when compared to number of people who have not defaulted their credit. In future, we would like to collect more data so that we have a balanced dataset which can further improve the accuracy. Also, we will plan to study on attributes which were not considered here, which can have an impact on whether the person defaults or not in future. This will help in improving the performance of the model which in turn provides more accuracy than the currently implemented model.

VI. Conclusion

After implementing all the six algorithms and then comparing each other based on their accuracy, we could see that XGBoost and ANN performed better followed by AdaBoost. In general, we can say that models trained based on both the boosting algorithms showed higher accuracy when compared to other machine learning models such as Logistic Regression and K-Nearest Neighbor.

References

https://ieeexplore.ieee.org/document/9239944
https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/
https://scikit-learn.org/stable/supervised_learning.html#supervised-learning
https://xgboost.readthedocs.io/en/stable/
https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe
https://www.sciencedirect.com/topics/earth-and-planetary-sciences/artificial-neural-network