Project Report

on

Credit Default Classification



Submitted in partial fulfillment for the award of

Post Graduate Diploma in Big Data Analytics (PG-DBDA) from C-DAC ACTS (Pune)

### Guided by:

## Mr. Abhay Dandekar

#### Presented by:

**Mr. Himanshu Jagtap Prn: 52**

**Ms. Shruti Padole Prn: 38**

**Mr. Parth Salvi Prn: 33**

**Centre of Development of Advanced Computing (C-DAC), Pune**

**ACKNOWLEDMENT**

This project “**Credit Default Classification**” was a great learning experience for us and we are submitting this work to Advanced Computing Training School (CDAC ACTS).

We all are very glad to mention the name of *Mr Abhay Dandekar* for his valuable guidance to work on this project. His guidance and support helped us overcome various obstacles and intricacies during the course of project work.

We are highly grateful to Ms. Risha P.R. (Manager (ACTS training Centre), C-DAC, for her guidance and support whenever necessary while doing this course Post Graduate Diploma in *Big Data Analytics (PG-DBDA)* through C-DAC ACTS, Pune.

Our most heartfelt thanks go to *Ms. Namrata Ailawar* (Course Coordinator, PG-*DBDA*) who gave all the required support and kind coordination to provide all the necessities like required hardware, internet facility and extra Lab hours to complete the project and throughout the course up to the last day here in C-DAC ACTS, Pune.

**From:**

**Himanshu Jagtap (170840125052)**

**Shruti Padole (170840125038)**

**Parth Salvi (170840125033)**

**TABLE OF CONTENTS**

[**Introduction**](#_1xdztgy3q7f0) **4**

[**Architecture**](#_fqxcfrfk12im) **5**

[**Data Sources**](#_z1325q9p8pij) **6**

[**Hardware requirements**](#_54jsilpf29lx) **7**

[**Software requirements**](#_p22xujhmgnph) **8**

[**Testing**](#_wt468r4nou0b) **9**

[**Performance numbers**](#_5r17en9o911q) **10**

[**Conclusion**](#_mabkxs2oen3v) **11**

# Introduction

1. ***PROBLEM STATEMENT***

The goal of our project is to build a model that borrowers can use to help make the best financial decisions. This project makes use of defaulter classification algorithms, which calculates the probability of default, often used by banks to determine whether or not a loan should be granted.

1. ***SCOPE OF YOUR PROBLEM***

Lenders, such as [banks](https://en.wikipedia.org/wiki/Bank) and credit card companies, need to evaluate the potential risk posed by lending money to consumers and to mitigate losses due to [bad debt](https://en.wikipedia.org/wiki/Bad_debt). This is where machine learning comes into use, in order to classify customers as potential defaulters (someone who does not repay their debts). These algorithms use personal information of customers like their age, monthly income, other debts, number of dependent family members, etc. and evaluate the correlation between them to predict the probability that the customer will repay the debt.

1. ***REQUIREMENT***

All banks need some kind of software or algorithm to estimate the risk in lending money to customers. It’s very common for people to take loans and not repay it. Thus, to keep themselves from incurring such losses, banks use various machine learning algorithms to determine whether a customer should be granted loan or not.

Defaulter classification is not limited to just banks. Other organizations, such as mobile phone companies, insurance companies, landlords, and government departments employ the same techniques. Digital finance companies such as online lenders also use alternative data sources to calculate the creditworthiness of borrowers.

1. **PROJECT ABSTRACT**

A **credit risk** is the risk of [default](https://en.wikipedia.org/wiki/Default_(finance)) on a debt that may arise from a borrower failing to make required payments. Through our project, we aim to classify a borrower as a defaulter or non-defaulter by applying machine learning algorithms on credit-related behavior of the borrower. This data includes details like age of the borrower, their debt ratio, number of dependent family members, monthly income, revolving utilization of unsecured lines, number of open credit lines and loans, number real estate loans or lines, etc.

We have applied several algorithms in order to determine the one that gives maximum **AUC** (area under the curve). **AUC** is a metric to measure the performance of an algorithm. The greater the **AUC**, the more efficient the algorithm.

The algorithms we used are:

* Logistic Regression
* Naïve Bayes Classifier
* Random Forrest
* Gradient Boosting
* XGBoost (Extreme Gradient Boosting)
* Neural Networks

After, comparing the values of **AUC** for all the above-mentioned algorithms, we concluded that Neural Networks was the best model.

1. ***WHY IT CANNOT BE ACHIEVED TRIVIALLY***

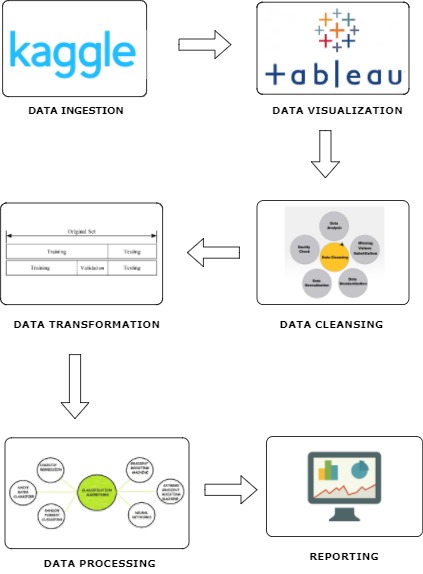
Credit risk analysis involves access to sensitive information like monthly income, the spending history of an individual, their current debts, details about their assets which in the wrong hands, would do serious damage. The data gathered is large in volume, sometimes not even uniform, and at times, partly ambiguous.

The raw data needs to be cleansed of null values in such a way that it doesn’t affect the trend in the data.

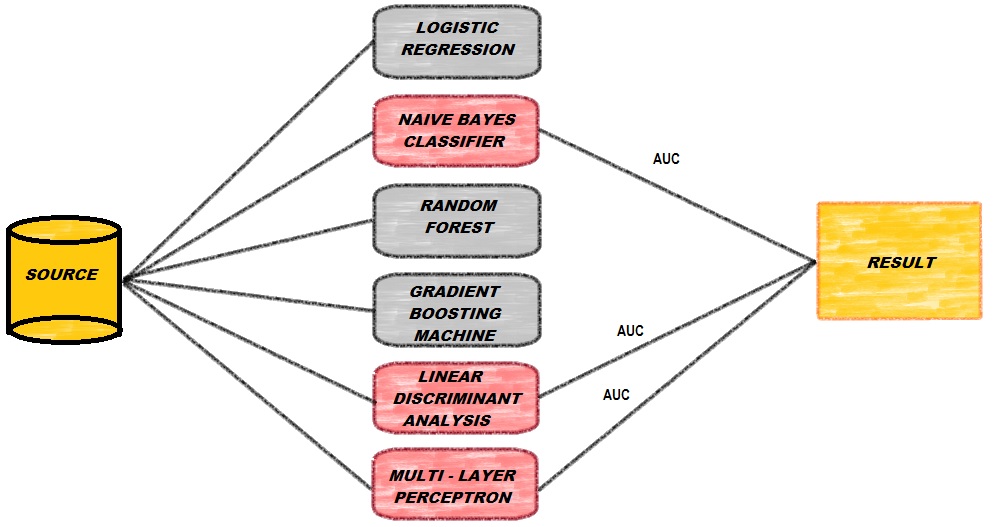
Furthermore, finding the apt model for prediction is no trivial job either. We tried several models, and through trial and error, were finally able to determine the best working model.

# Architecture

1. ***DATA FLOW DIAGRAM***



1. ***DESIGN DIAGRAM***



1. ***DESCRIPTION OF EACH COMPONENT USED IN THE ARCHITECTURE***
2. **Data Ingestion:**

Data ingestion is the process of obtaining and importing data for immediate use or storage in a database. We used dataset from a competition named ‘Give me some credit’ which was hosted in 2011.

1. **Data Visualization:**

Data visualization is the presentation of data in a pictorial or graphical format. It enables decision makers to see analytics presented visually, so they can grasp difficult concepts or identify new patterns. With interactive visualization, you can take the concept a step further by using technology to drill down into charts and graphs for more detail, interactively changing what data you see and how it’s processed.

1. **Data Transformation:**

Data transformation is the process of converting data or information from one format to another, usually from the format of a source system into the required format of a new destination system.

1. **Data Cleansing:**

Data cleansing or data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.

1. **Data Processing:**

Data processing is the conversion of data into usable and desired form. This conversion or “processing” is carried out using a predefined sequence of operations either manually or automatically. Most of the data processing is done by using computers and thus done automatically.

1. **Reporting:**

Data reporting is the process of collecting and submitting data which gives rise to accurate analyses of the facts on the ground; inaccurate data reporting can lead to vastly uninformed decisions based on erroneous evidence. When data is not reported, the problem is known as underreporting; the opposite problem leads to false positives.

**Hardware requirements**

Based upon the software products used, we have documented the pertinent hardware requirements.

**Rstudio** -

* An Intel-compatible platform running Windows 2000, XP/2003/Vista/7/8/2012 Server/8.1/10.
* Latest version of R statistical programming tool installed on the respective system.
* Depending upon the computations, minimum of 1GB of RAM.
* The administrative privileges are required to install and run R-Studio utilities under Windows 2000/XP/2003/Vista/7/8/2012 Server/8.1/10.

**Python -**

* Processors: Intel Atom® processor or Intel® Core™ i3 processor
* Disk space: 1 GB
* Operating systems: Windows\* 7 or later, macOS, and Linux
* Python\* versions: 2.7.X, 3.6.X
* Included development tools: conda\*, conda-env, Jupyter Notebook\* (IPython)
* Compatible tools: Microsoft Visual Studio\*, PyCharm\*
* Included Python packages: NumPy, SciPy, scikit-learn\*, pandas, Matplotlib, Numba\*, Intel® Threading Building Blocks, pyDAAL, Jupyter, mpi4py, PIP\*, and others

**Jupyter Notebook** –

* Per user: 1GB RAM + 1GB of disk + .5 CPU core.
* Overhead: 2-4GB or 10% system overhead (whatever is larger), .5 CPU cores, 10GB disk space

**Tableau** –

* 64-bit processor.
* 8 physical cores, 2.0 GHz or higher CPU.
* 32 GB system memory.
* 50 GB minimum free disk space.

**Apache Spark** –

* As Spark ecosystem reads input from external storage system, the Hadoop File System (HDFS) or Hbase can be used.
* For managing cluster services such as Hadoop YARN or Apache Mesos can be used.
* In general, Spark can run well with anywhere from 8 GB to hundreds of gigabytes of memory per machine. In all cases, we recommend allocating only at most 75% of the memory for Spark; leave the rest for the operating system and buffer cache.
* Using a 10 Gigabit or higher network is the best way to make these applications faster.
* Spark scales well to tens of CPU cores per machine because it performes minimal sharing between threads. You should likely provision at least 8-16 cores per machine.

# Software requirements

The major requirements of our project in terms of software were:

1.**RStudio** – RStudio is a free and open-source integrated development environment for R, a programming language for statistical computing and graphics. Used for data cleaning, feature selection, application of machine learning algorithms and predictions.

2. **Python** - Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python's popularity for data science is largely due to the strength of its core libraries (NumPy, SciPy, pandas, matplotlib, seaborn), high productivity for prototyping and building small and reusable systems, and its strength as a general purpose programming language.

3.**Jupyter Notebook** - Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text Segregation or creating bins in data, application of machine learning algorithms and predictions.

4.**Tableau** – Tableau is a software for creating data visualizations and deriving insights from it. We used Tableau for analizing inherent trends and data visualization.

5**. Apache** **Spark** – An open source clustering platform which can be used for programming entire clusters with implicit data parallelism and fault tolerance. We used Apache Spark for improving scalability and performance of our applied algorithms.

# 

# Data Sources

The dataset that we selected was a part of the Kaggle competition “Give Me Some Credit”. The competition went live in the year of 2012. Almost 920 teams competed for the top prize (which was $5000).

1. SCHEMA – Dataset had 12 features and a target variable(‘SeriousDlqin2yrs’). Features included in the set were,

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Variable Name** | **Description** | **Type** |
| **SeriousDlqin2yrs** | **Person experienced 90 days past due delinquency or worse** | **Y/N** |
| RevolvingUtilizationOfUnsecuredLines | Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits | percentage |
| age | Age of borrower in years | integer |
| NumberOfTime30-59DaysPastDueNotWorse | Number of times borrower has been 30-59 days past due but no worse in the last 2 years. | integer |
| DebtRatio | Monthly debt payments, alimony, living costs divided by monthy gross income | percentage |
| MonthlyIncome | Monthly income | real |
| NumberOfOpenCreditLinesAndLoans | Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards) | integer |
| NumberOfTimes90DaysLate | Number of times borrower has been 90 days or more past due. | integer |
| NumberRealEstateLoansOrLines | Number of mortgage and real estate loans including home equity lines of credit | integer |
| NumberOfTime60-89DaysPastDueNotWorse | Number of times borrower has been 60-89 days past due but no worse in the last 2 years. | integer |
| NumberOfDependents | Number of dependents in family excluding themselves (spouse, children etc.) | integer |

**Data Transformations:**

Total no. of records: **1,50,000**

1) **Imputing missing values:**

Two columns contain NaN's, MonthlyIncome and NumberofDependents.

**MonthlyIncome** : 29,731

**NumberofDependent**s : 3,924

a) Monthly Income imputation: We calculated the means of Monthly Income based on Age bins of 10 years. This is assumed to be a practical alternative for missing values in income column.

b) Number of dependents was imputed by using the Mode of the column, which is 0.

2) **Standardization:**

Standardize features by removing the mean and scaling to unit variance

Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set.

Mean and standard deviation are then stored to be used on later data using the transform method.

For instance, many elements used in the objective function of a learning algorithm (such as the RBF kernel of Support Vector Machines or the L1 and L2 regularizers of linear models) assume that all features are centered around 0 and have variance in the same order.

If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected.

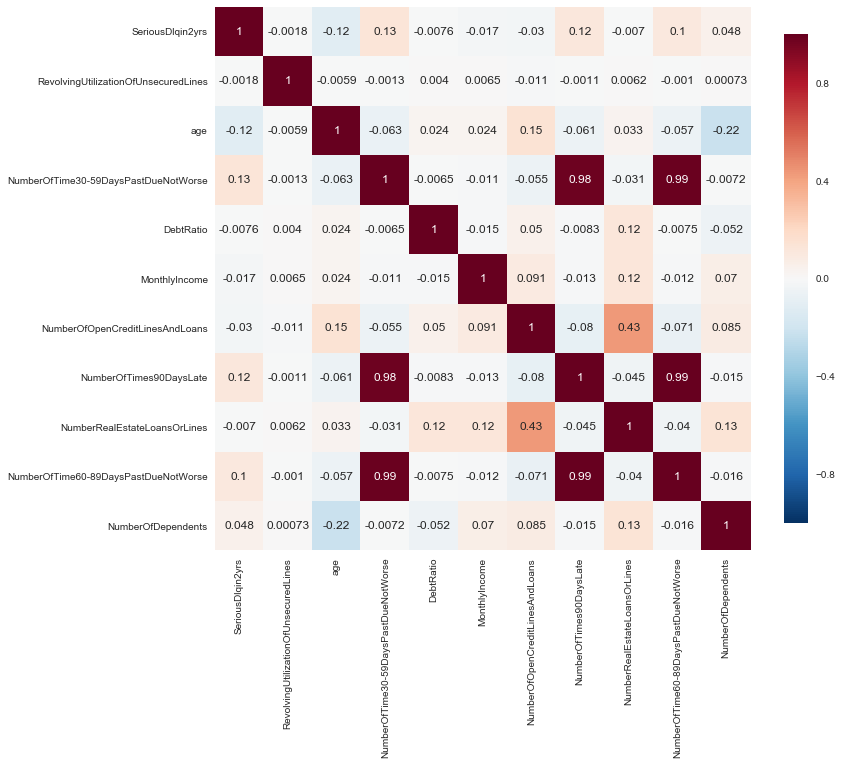
3) **Discretization:**

Binning or discretization is the process of transforming numerical variables into categorical counterparts

ex: to bin values for Age into categories such as 20-29,30-39. Supervised binning methods transform numerical variables into categorical counterparts and refer to the target(class)

information when selecting discretization cut points. Ex: Entropy-based binning is an example of a supervised binning methods. The goal of discretization is to reduce the number of values a continuous variable assumes by grouping them into a number, b, of interval or bins.

We divided each column into bins based on criteria such that each column contains more than 500 records labeled '1'.



**Algorithms:**

1. Logistic Regression
2. Random Forest
3. Naïve Bayes
4. Linear Discriminant Analysis
5. Gradient Boosting Machines
6. Neural Networks
7. **Logistic regression** is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a *logit transformation* of the probability of presence of the characteristic of interest:

Logistic regression equation

where p is the probability of presence of the characteristic of interest. The logit transformation is defined as the logged odds:

Odds=p/(1-p)

and

Logit(p)=ln(p/(1-p))

Rather than choosing parameters that minimize the sum of squared errors (like in ordinary regression), estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values.

1. **Random forests** or **random decision forests** are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

### Features of Random Forests

* It is unexcelled in accuracy among current algorithms.
* It runs efficiently on large data bases.
* It can handle thousands of input variables without variable deletion.
* It gives estimates of what variables are important in the classification.
* It generates an internal unbiased estimate of the generalization error as the forest building progresses.
* It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
* It has methods for balancing error in class population unbalanced data sets.
* Generated forests can be saved for future use on other data.
* Prototypes are computed that give information about the relation between the variables and the classification.
* It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
* The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
* It offers an experimental method for detecting variable interaction.

1. **Naive Bayes** methods are a set of supervised learning algorithms based on applying Bayes’ theorem with the “naive” assumption of independence between every pair of features. Given a class variable y and a dependent feature vector x_1through x_n, Bayes’ theorem states the following relationship:

P(y \mid x_1, \dots, x_n) = \frac{P(y) P(x_1, \dots x_n \mid y)}
                                 {P(x_1, \dots, x_n)}

Using the naive independence assumption that

P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),

for all i, this relationship is simplified to

P(y \mid x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}
                                 {P(x_1, \dots, x_n)}

1. **Linear Discriminant Analysis (LDA)**

Logistic regression is a simple and powerful linear classification algorithm. It also has limitations that suggest at the need for alternate linear classification algorithms.

Two-Class Problems. Logistic regression is intended for two-class or binary classification problems. It can be extended for multi-class classification, but is rarely used for this purpose.

Unstable With Well Separated Classes. Logistic regression can become unstable when the classes are well separated.

Unstable With Few Examples. Logistic regression can become unstable when there are few examples from which to estimate the parameters.

Linear Discriminant Analysis does address each of these points and is the go-to linear method for multi-class classification problems. Even with binary-classification problems, it is a good idea to try both logistic regression and linear discriminant analysis.

The model uses Bayes Theorem to estimate the probabilities. Briefly Bayes’ Theorem can be used to estimate the probability of the output class (k) given the input (x) using the probability of each class and the probability of the data belonging to each class:

**P(Y=x|X=x) = (PIk \* fk(x)) / sum(PIl \* fl(x))**

Where PIk refers to the base probability of each class (k) observed in your training data

1. **Gradient boosting** is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

Algorithm:

1. Set *f* ˆ(*x*) = 0 and *ri* = *yi* for all *i* in the training set.

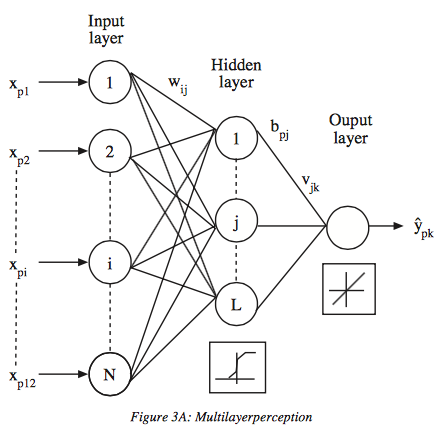
2. For *b* = 1*,* 2*,… B*, repeat:

(a) Fit a tree *f* ˆ*b* with *d* splits (*d* + 1 terminal nodes) to the training  
 data (*X, r*).  
 (b) Update *f* ˆ by adding in a shrunken version of the new tree:

*f* ˆ(*x*) *← f* ˆ(*x*) + *λ f* ˆ*b*(*x*)  
(c) Update the residuals,  
 *ri ← ri – λ f* ˆ*b*(*xi*)*.*

3. Output the boosted model,  
 *f* ˆ(*x*) = *∑* *λ f* ˆ*b*(*x*)*.*

1. A **feedforward artificial neural network (**ANN) model, also known as deep neural network (DNN) or multi-layer perceptron (MLP). A **multilayer perceptron** (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.



# Testing

Testing was primarily done on 2 datasets.

1**)Cross Validation dataset** – Created from inherent training data set provided. Training dataset was divided into sets, in the ratio of 2:1. On the larger set, all the feature cleaning and learning was done. On the smaller set, we tested how well our trained model predicted the required classes and we estimated model properties (mean error for numeric predictors, classification errors for classifiers, recall and precision for IR-models etc.) which were further used for tuning.

2)**Kaggle Test dataset** – This dataset was provided on Kaggle platform. Generally used for actual implementation of the model, this data set is used as benchmark for testing the accuracy of created models. It generally showcases a real world scenario. Also this is final stage of machine learning after which developed models can’t be tuned.

# Performance numbers

**Approach 1:**

We tracked a metric called AUC (Area Under Curve) for quantifying the performance of different algorithms. We had a normal training data set and a synthetically engineered dataset. We created a validation set from the training dataset. Of all the algorithms applied on normal dataset, when tested on validation set, we achieved a highest AUC of .678 using Neural Net. Our performance considerably increased on synthetically engineered dataset, which was created using a technique called ‘**Smote**’. When tested on validation set using algorithms fitted on smote dataset, we achieved an AUC of **0.734**, which was obtained using Naïve Bayes algorithm. After considering this figures we used algorithms used on smote dataset, for predicting on test dataset provided on Kaggle platform. By using best classifier trained on this data we achieved AUC of **0.81** on the test dataset.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Misclassification** | | **AUC** | |
|  | **Norm.** | **SMOTE** | **Norm.** | **SMOTE** |
| **Logistic Regression** | 0.069 | 0.154 | 0.5066 | 0.661 |
| **Naïve Bayes** | 0.0615 | **0.1253** | 0.678 | **0.7338** |
| **Random Forest** | 0.0692 | 0.1145 | 0.6129 | 0.6371 |
| **Gradient Boosting Machines** | 0.0605 | 0.0844 | 0.5226 | 0.6188 |
| **XGBoost** | 0.187 | 0.0612 | 0.6457 | 0.5386 |
| **MLP Classifier** | 0.179 | **0.2184** | 0.6766 | **0.7155** |

**Approach 2 (Discretization):**

In this approach we discretized all columns based on the frequency of labels. Keeping a threshold of at least 500 defaulters in a column we created dummy variables out of continuous data type. We applied the above mentioned models (except Gradient boosting) on the engineered dataset. By doing so we achieved the following output.

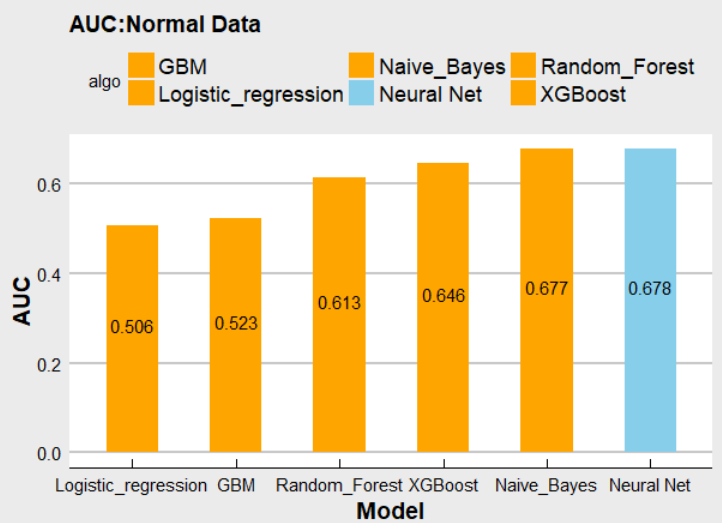
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Models | **AUC** | **Precision**  0 1 | | **Recall**  0 1 | | **F1-score**  0 1 | |
| Logistic Regression (L2) | 0.859 | 0.95 | 0.57 | 0.99 | 0.20 | 0.97 | 0.30 |
| Random Forest | 0.84 | 0.93 | 0.0 | 1.0 | 0.0 | 0.97 | 0.0 |
| Bernoulli Naïve Bayes | 0.846 | 0.96 | 0.35 | 0.93 | 0.51 | 0.95 | 0.42 |
| Gaussian Naïve Bayes | 0.836 | 0.97 | 0.32 | 0.92 | 0.54 | 0.94 | 0.41 |
| **Linear Discriminant Cl.** | **0.8579** | **0.96** | **0.46** | **0.97** | **0.38** | **0.96** | **0.42** |
| Quadratic Discriminant Cl. | 0.838 | 0.97 | 0.34 | 0.93 | 0.54 | 0.94 | 0.41 |
| MLP Classifier | 0.858 | 0.94 | 0.55 | 0.99 | 0.18 | 0.97 | 0.27 |
| K Nearest Classifier | 0.693 | 0.94 | 0.46 | 0.99 | 0.12 | 0.96 | 0.19 |
| **Voter Classifier** | **0.859** | **0.96** | **0.43** | **0.96** | **0.43** | **0.96** | **0.43** |

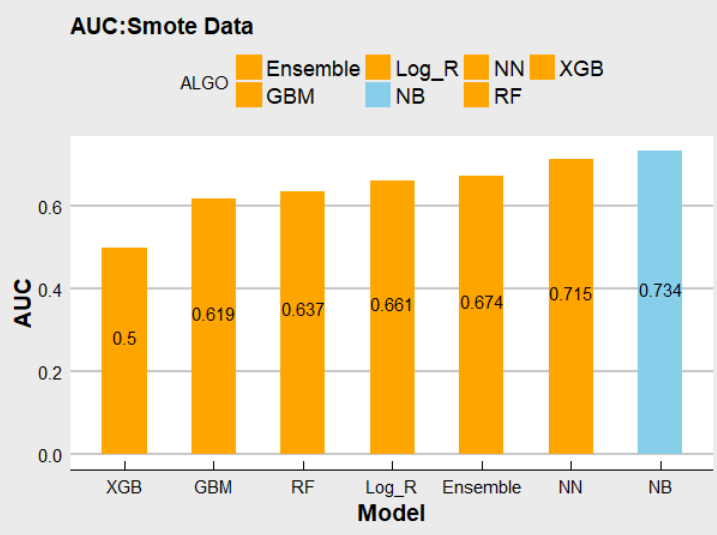
**Precision /Positive predictive value = TP / ( TP + FP)**

**Recall / true positive rate / Sensitivity = TP / (TP + FN)**

**F1- score = 2 \* (precision \* recall) / (precision + recall)**

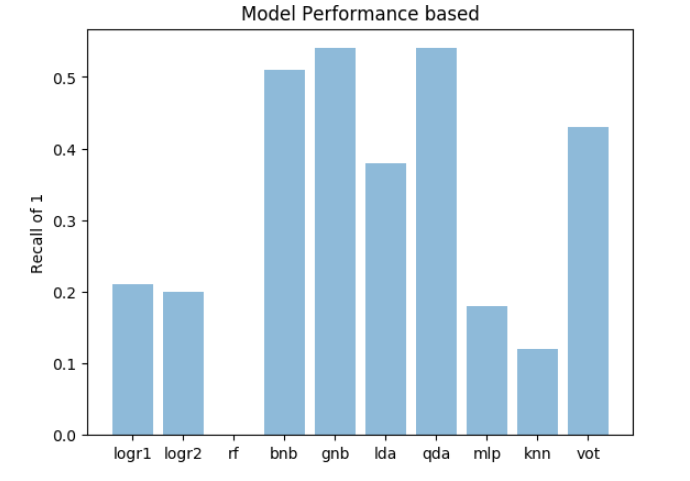
**Approach 1:**

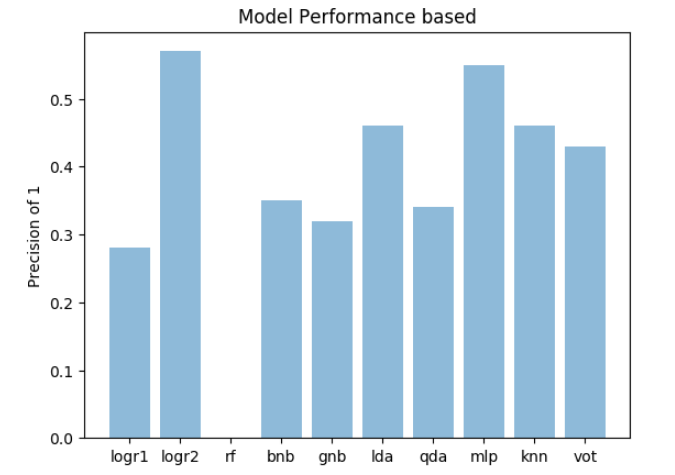




# 

**Approach 2:**





**Implementation in Spark:**

MLlib is Spark’s machine learning (ML) library. Its goal is to make practical machine learning scalable and easy. At a high level, it provides tools such as:

* ML Algorithms: common learning algorithms such as classification, regression, clustering, and collaborative filtering
* Featurization: feature extraction, transformation, dimensionality reduction, and selection
* Pipelines: tools for constructing, evaluating, and tuning ML Pipelines
* Persistence: saving and load algorithms, models, and Pipelines
* Utilities: linear algebra, statistics, data handling, etc.

**Environment Setting for Spark:**

Pypark installation in Windows

1. Download and install Gnu on windows (GOW).

Basically, GOW allows you to use linux commands on windows. In this install, we will need curl, gzip, tar which GOW provides.

2) Download and install Anaconda

3) Install apache pyspark

a) Choose a Spark release

b) Choose a package type

c) Choose a download type: (Direct Download)

d) Download spark

4) Move the file to where you want to unzip it.

mkdir C:\spark

5) Unzip the file. Use the bolded commands below

6) gzip -d spark-2.1.0-bin-hadoop2.7.tgz

tar xvf spark-2.1.0-bin-hadoop2.7.tar

7) Download winutils.exe into your spark-2.1.0-bin-hadoop2.7\bin

curl -k -L -o winutils.exe

<https://github.com/steveloughran/winutils/blob/master/hadoop-2.6.0/bin/winutils.exe?raw=true>

8) Make sure you have Java 7+ installed on your machine.

9) Next, we will edit our environmental variables so we can open a spark notebook in any directory.

setx SPARK\_HOME C:\spark\spark-2.1.0-bin-hadoop2.7

setx HADOOP\_HOME C:\spark\spark-2.1.0-bin-hadoop2.7

setx PYSPARK\_DRIVER\_PYTHON ipython/jupyter

setx PYSPARK\_DRIVER\_PYTHON\_OPTS notebook

Add ;C:\spark\spark-2.1.0-bin-hadoop2.7\bin to your path.

***To start the master and worker on windows***

Go to %SPARK\_HOME%\bin folder in a command prompt

Run spark-class org.apache.spark.deploy.master.Master to run the master. This will give you a URL of the form spark://ip:port

Run spark-class org.apache.spark.deploy.worker.Worker spark://ip:port to run the worker. Make sure you use the URL you obtained in step 2.

Run spark-shell --master spark://ip:port to connect an application to the newly created cluster.

Run pyspark.

Run all above commands in separate shells.

**Preprocessing in PySpark :**

## **StringIndexer :**

StringIndexer encodes a string column of labels to a column of label indices. The indices are in [0, numLabels), ordered by label frequencies, so the most frequent label gets index 0. The unseen labels will be put at index numLabels if user chooses to keep them. If the input column is numeric, we cast it to string and index the string values.

**VectorAssembler:**

VectorAssembler is a transformer that combines a given list of columns into a single vector column. It is useful for combining raw features and features generated by different feature transformers into a single feature vector, in order to train ML models like logistic regression and decision trees. VectorAssembler accepts the following input column types: all numeric types, boolean type, and vector type.

**LabeledPoint:**

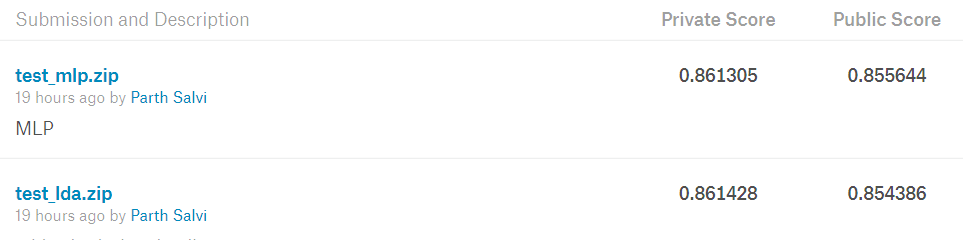
A labeled point is a local vector, either dense or sparse, associated with a label/response. In MLlib, labeled points are used in supervised learning algorithms. We use a double to store a label, so we can use labeled points in both regression and classification.

**Algorithms implemented in PYSPARK:**

* 1. **Logistic Regression**
  2. **RandomForest**
  3. **NaiveBayes Classifier**
  4. **MLP Classifier**

All the above algorithms show similar performance on the underlying data. Hence, spark can be used when the size of the data is large.

By using the **2nd** approach, we achieved a Kaggle private score of **0.86148** and a public score of 0.85.



# **Conclusion:**

The metric that we are using to track our accuracy is Area Under Curve(AUC). We achieved an AUC of ***0.81*** on test data set provided on Kaggle platform using the 1st approach.

We achieved an AUC of ***0.861*** using 2nd approach. The dataset was part of Kaggle competition “Give me some credit”. The team that won the competition had an AUC of **0.869**.

Considering performance and scalability, we used **Spark** as a platform to implement the same algorithms that we used in R and Python.