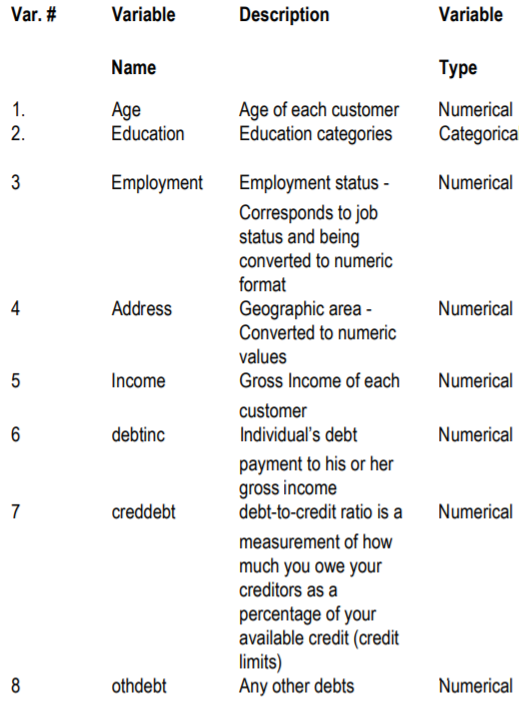
Project-1 Report

* Gaurav Khurana

**Problem Statement –**

The loan default dataset has 8 variables and 850 records, each record being loan default status for each customer. Each Applicant was rated as “Defaulted” or “Not-Defaulted”. New applicants for loan application can also be evaluated on these 8 predictor variables and classified as a default or non-default based on predictor variables.

**Variables Description –**



**Exploratory Data Analysis –**

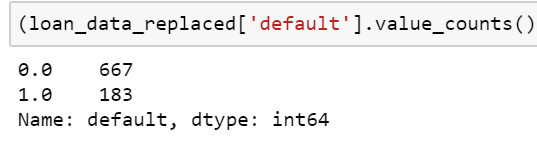
1. Missing Value Analysis



Upon carrying out the missing data analysis it was found that out of the given 8 variables (7 independent variables and 1 dependent variable) only the dependent variable “default” had missing values.

To deal with the missing “default” values two approaches were used:

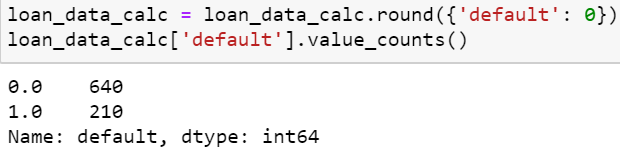
1. All the 150 missing values were replaced by 0



On doing this, it was observed the values were unbalanced, indicating that the dataset was more biased towards 0 i.e. defaulted. Therefore, another approach needed to be taken.

2. Calculating the values using KNN estimations

KNN was used in order to calculate the missing values. The calculated values were then rounded off to the nearest one’s place (n >= 0.5 =1 and n < 0.5 =0)

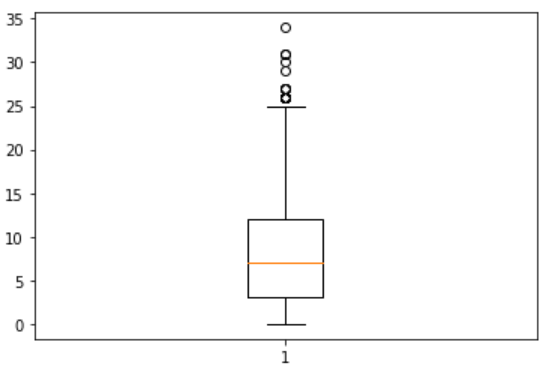


It can be observed that the ratio of 1 to 0 was more balanced than the previous approach.

Thus, these values were chosen to further carry out model development.

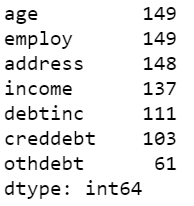
1. Outlier Analysis

Outlier Analysis was done by using box-plot to check whether the given data had any outliers. One such example was the box-plot plotted for the “address” field.



As it can be observed that there are notable number of outliers present in that particular field. Similarly, outlier analysis for all the variables were carried out.

The results were as follows:



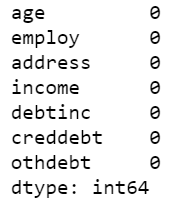
The method used to find out all these outliers used was to calculate the 25th and the 75th percentile of all the above mention variables. Then, to calculate the Inter Quartile Range (IQR) for each variable which is nothing but the difference between the 25th and the 75th percentile.

Next, the minimum and the maximum values were calculated. The values that were either higher than the maximum or lower than the minimum value were found out and were replaced by NaN.

Which resulted in the above values of the above table.

(It can be observed that “ed” and “default” variables were not involved in outlier analysis as they are categorical variables.)

Since, the number of outliers were not high enough thus, the values could not be simply dropped or ignored and hence, KNN (Key Nearest Neighbour) was used in order calculate the NaN values and thus replacing those values.

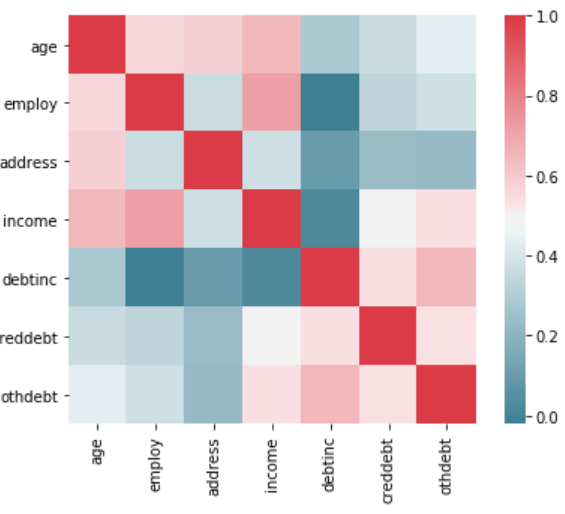


1. Feature Extraction

There are various methods which can be used to find out the correlation or dependence amongst the variables. The method used for this particular data set was correlation matrix, since majority of the variables are continuous variables.

(Chi squared test can only be applied to categorical variables)

The following correlation matrix was obtained:

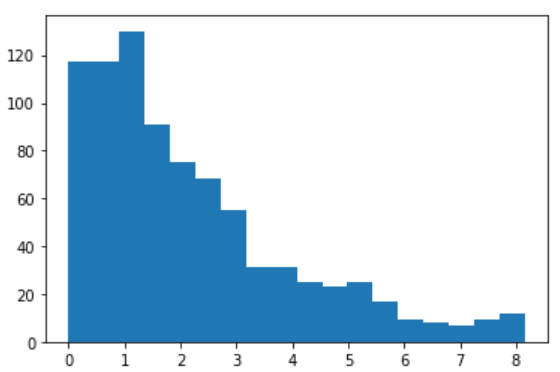


On careful observation it was noticed that none of the variables were either highly positively or negatively co-related to each other. Therefore, no variable needed to be dropped to carry out rest of the analysis.

1. Feature Scaling

Feature Scaling is a process that is applied on independent variables in order to bring the data points within a range so that it speeds up computation as many algorithms use distance between two points for their computation.

Histogram plot was used in order to see whether the data was normally distributed or not. For example the histogram plot for ‘otherdebt’ is as follows:



Similarly, histograms were plot for all continuous variables in order to see if the data is normally distributed or not.

Had the data been normally distributed we could’ve used the normalisation technique to scale the given data set. But since, the data is not normally distributed the normalization technique has been used.

Xn = (Xi – Xmin)/(Xmax – Xmin)

**Model Development –**

As part of the evaluation metrics, simple to more complex algorithms were implemented to find out which algorithms were suitable for this particular data set.

5 algorithms were implemented and their suitability was evaluated using a confusion matrix. The 5 algorithms used were:

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. KNN classifier
5. Naïve Bayes

**Confusion Matrix:**

it is a performance measurement for machine learning classification problem where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.



It is extremely useful for measuring Recall, Precision, Specificity, Accuracy and most importantly AUC-ROC Curve.

**True Positive:**

Interpretation: You predicted positive and it’s true.

**True Negative:**

Interpretation: You predicted negative and it’s true.

**False Positive: (Type 1 Error)**

Interpretation: You predicted positive and it’s false.

**False Negative: (Type 2 Error)**

Interpretation: You predicted negative and it’s false.

The evaluation metrics considered in each of the machine algorithms are

1. Accuracy

Accuracy= ((TP+TN)\*100)/(TN+TP+FN+FP)

1. False Negative Rate (Recall)

FNR= (FN\*100)/(FN+TP)

1. Logistic Regression

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary values (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

y = e^(b0 + b1\*x) / (1 + e^(b0 + b1\*x))

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

1. Decision Tree

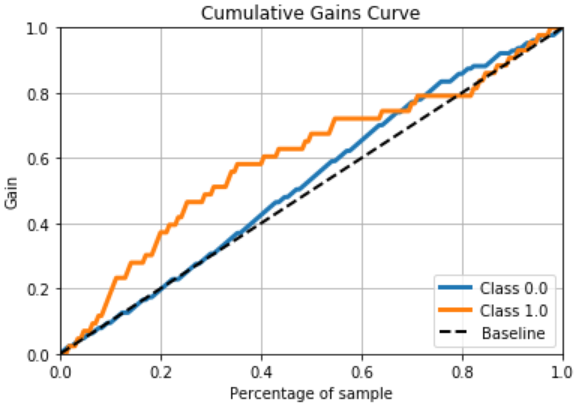
**Decision Trees (DTs)** are a non-parametric supervised learning method used for [classification](https://scikit-learn.org/stable/modules/tree.html#tree-classification) and [regression](https://scikit-learn.org/stable/modules/tree.html#tree-regression). The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

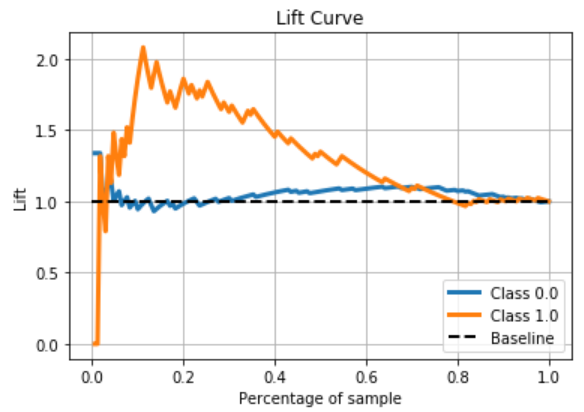
Some advantages of decision trees are:

* Simple to understand and to interpret. Trees can be visualised.
* Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
* The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
* Able to handle both numerical and categorical data. Other techniques are usually specialised in analysing datasets that have only one type of variable.

The disadvantages of decision trees include:

* Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
* Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
* The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree.





Gain and Lift charts for Decision Tree Model

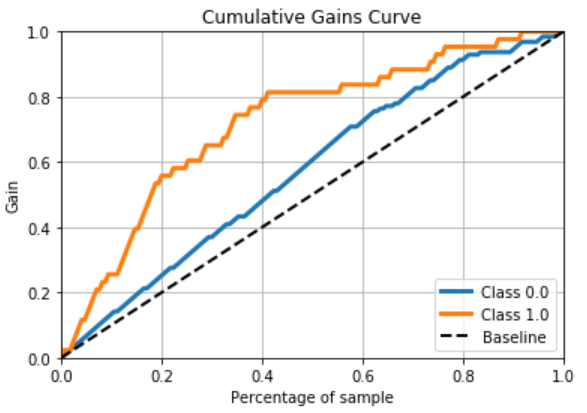
1. Random Forest

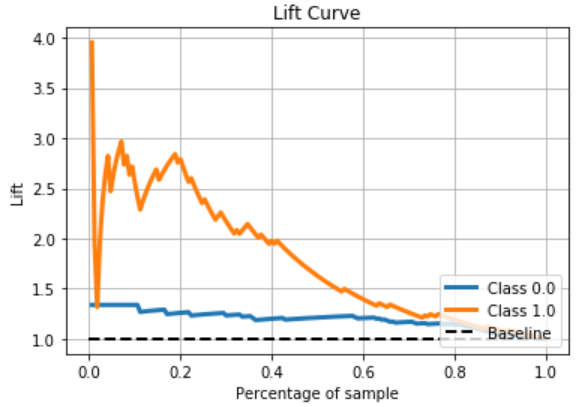
Random forest, like its name implies, consists of a large number of individual decision trees that operate as an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning). Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction.

The fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds.

***A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models.***

**The reason for this wonderful effect is that the trees protect each other from their individual errors** (as long as they don’t constantly all err in the same direction). While some trees may be wrong, many other trees will be right, so as a group the trees are able to move in the correct direction.





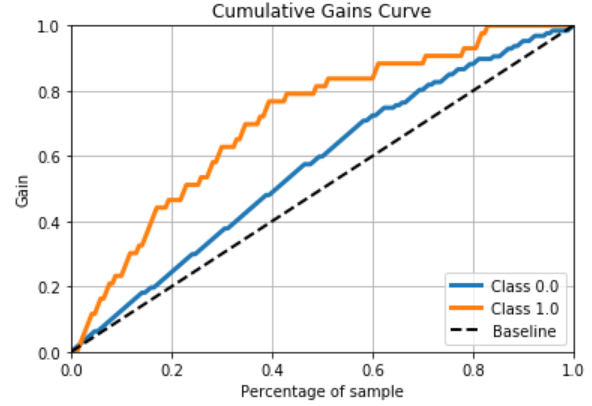
Gain and Lift charts for Random Forest Model

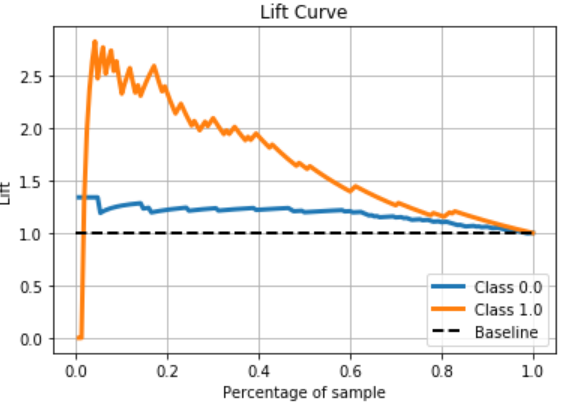
1. KNN

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other.

The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.

There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice.



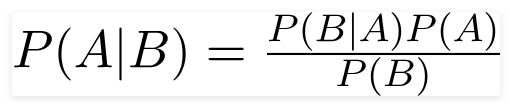


Gain and Lift charts for KNN(n=4) Model

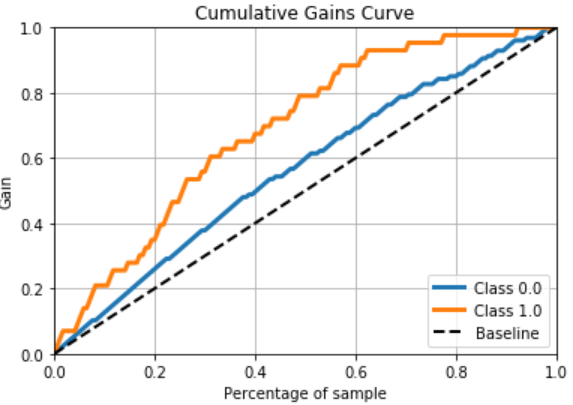
1. Naïve Bayes

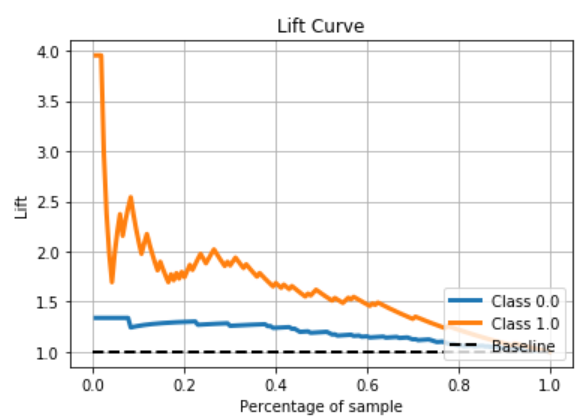
A Naive Bayes classifier is a probabilistic machine learning model that’s used for classification task. The crux of the classifier is based on the Bayes theorem.

**Bayes Theorem:**



Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naïve





Gain and Lift charts for Naïve Bayes Model

**Summary Table for Python:**

(The results may vary if the train test split is run again as that would change how the data has been split into train and test data sets, thus these results are for a particular configuration of train and test data.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr No | Machine Learning Algorithm | Accuracy (%) | False Negative Rate (%) | Review |
| 1. | Logistic Regression | 87.80 | 28.6 | The logistic regression model used on this particular set of data yields very good accuracy that means it predicts 8.7 times out of 10 correctly and has a relatively acceptable false negative rate of 28.6 percent that means it predicts every 2.86 times out of 10 to be false when it was in fact true. |
| 2. | Decision Tree | 74.11 | 51.06 | The decision tree model used on this particular set of data yields an alright accuracy that means it predicts 7.4 times out of 10 correctly and has a high false negative rate of 51.06 percent that means it predicts every 5.1 times out of 10 to be false when it was in fact true, therefore using this algorithm would not be recommended. |
| 3. | Random Forest | 81.17 | 31.03 | The Random Forest model used on this particular set of data yields good accuracy that means it predicts 8.1 times out of 10 correctly and has a relatively acceptable false negative rate of 31.03 percent that means it predicts every 3.1 times out of 10 to be false when it was in fact true, therefore on the basis of those factors this model can be used for this particular data set. |
| 4. | KNN (neighbours=4) | 88.82 | 20 | The KNN model used on this particular set of data yields the best accuracy amongst all the models used, that means it predicts 8.8 times out of 10 correctly and has the lowest false negative rate of 20 percent that means it predicts every 2 times out of 10 to be false when it was in fact true, therefore on the basis of those factors this model can be used for this particular data set. |
| 5. | Naïve Bayes | 71.17 | 56.52 | The Naïve Bayes used on this particular set of data yields moderate accuracy that means it predicts 7.1 times out of 10 correctly and has a very poor false negative rate of 56.52 percent that means it predicts every 5.6 times out of 10 to be false when it was in fact true, therefore on the basis of those factors this model is not recommended for this particular data set. |

**Summary Table for R:**

(The results may vary if the train test split is run again as that would change how the data has been split into train and test data sets, thus these results are for a particular configuration of train and test data.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr No | Machine Learning Algorithm | Accuracy (%) | False Negative Rate (%) | Review |
| 1. | Logistic Regression | 68.23 | 25.65 | The Logistic regression model used on this particular set of data yields moderate-low accuracy that means it predicts 6.8 times out of 10 correctly and has a low false negative rate of 25.65 percent that means it predicts every 2.5 times out of 10 to be false when it was in fact true, therefore on the basis of those factors this model can be used for this particular data set. |
| 2. | Decision Tree | 78.23 | 19.86 | The Decision Tree model used on this particular set of data yields a good accuracy that means it predicts 7.8 times out of 10 correctly and has a low false negative rate of 19.86 percent that means it predicts every 1.9 times out of 10 to be false when it was in fact true, therefore on the basis of those factors this model can be used for this particular data set. |
| 3. | Random Forest | 77.06 | 19.72 | The Random Forest model used on this particular set of data yields a good accuracy that means it predicts 7.7 times out of 10 correctly and has a low false negative rate of 19.72 percent that means it predicts every 1.9 times out of 10 to be false when it was in fact true, therefore on the basis of those factors this model can be used for this particular data set. |
| 4. | KNN (k=5) | 74.12 | 3.90 | The KNN model used on this particular set of data yields a moderate accuracy that means it predicts 7.4 times out of 10 correctly and has the lowest false negative rate of 3.9 percent that means it predicts every .3 times out of 10 to be false when it was in fact true, therefore on the basis of those factors this model would be the second most recommended for this particular data set. |
| 5. | Naïve Bayes | 77.65 | 11.86 | The Naïve Bayes model used on this particular set of data yields a moderate-high accuracy that means it predicts 7.7 times out of 10 correctly and has the second lowest false negative rate of 11.86 percent that means it predicts every 1.1 times out of 10 to be false when it was in fact true, therefore on the basis of those factors this model would be the most recommended for this particular data set. |

How to Run the code:

For Python:

* 1. Ensure that Jupyter notebooks have been installed on the system along with anaconda distribution.
  2. Download and save the file in any location on the system.
  3. Open the command prompt and change the current directory location to where the notebook has been saved on the system.



* 1. After changing the location, type “jupyter notebook” in the command prompt.



* 1. After a while, the jupter notebook should open up.



* 1. Click on “Project1.ipynb” file.
  2. When the file opens, on the menu bar at the top, click on “Cell” and the “Run All”.
  3. This should run all the cell, the file includes pre-processing of the data with the help of charts and statistics, then runs the chosen models and gives out the accuracy and false negative rates along with the Gain and Lift charts.

For R:

1. Ensure R-studios has been downloaded and updated on the system.
2. Download and save the entire folder to any location on the system.
3. Open R-Studios, “open file”, select the destination of where the file has been saved, find the file and click on open.

(In the second line of code where setwd() is mentioned, enter the path at which the entire file was downloaded and kept for eg “setwd("C:/Users/Gaurav/Desktop/Project 1")” )

1. To run the entire code, select “Code”, then select “Run Region” and then “Run All”.
2. All the variables will then be run and stored in the R environment along with all the model results such as accuracy and false negative rate.