PROJECT SANTANDER

By Pavithra Mamallan

14th July 2019

SYNOPSIS

Chapter 1

1. Introduction

1.1.Problem Statement

At Santander, mission is to help people and businesses prosper. We are always looking for ways to help our customers understand their financial health and identify which products and services might help them achieve their monetary goals. Our data science team is continually challenging our machine learning algorithms, working with the global data science community to make sure we can more accurately identify new ways to solve our most common challenge, binary classification problems such as: is a customer satisfied? Will a customer buy this product? Can a customer pay this loan?

In this challenge, we need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted.

Data

The dataset "train" contains 2,00,000 observations and 202 variables. The dataset contains numeric feature variables, the binary target column, and a string ID_code column. The task is to predict the value of target column in the test set.

Output Variable

- **Target** – Whether the Customer will make a specific transaction in the future irrespective of the transaction amount. It is a numeric variable with either 0 or 1.

Features

- **ID_Code –** It contains the code ID for each individual customer. It is a character variable.
- **Var_0 to Var_199 -** These are numeric values contributing to the target variable.

Chapter 2

2. Pre-processing techniques

2.1.Outlier Analysis

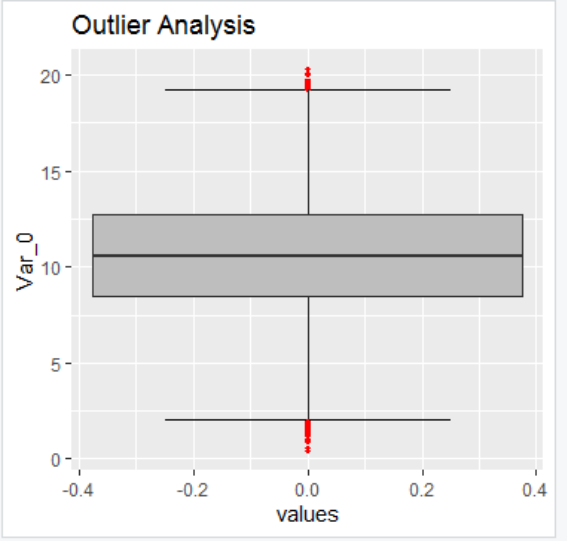
Outliers are extreme values that deviate from other observations on data; they may indicate variability in a measurement, experimental errors or a novelty. In other words, an outlier is an observation that diverges from an overall pattern on a sample. Outliers can be detected using one of the four major techniques.

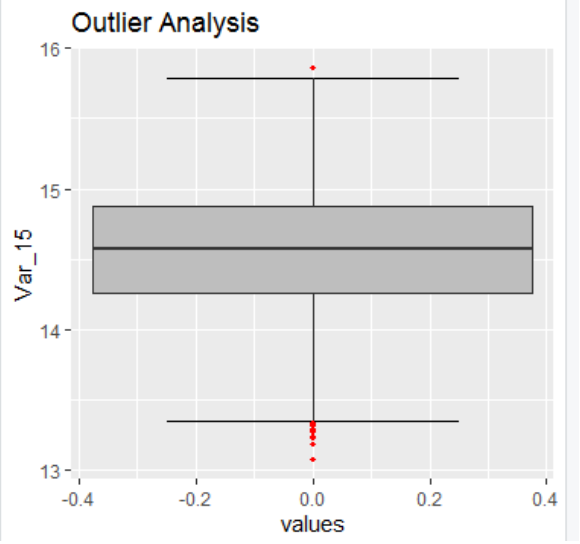
- **Graphical Plot – Box plot**
- **Statistical Technique – Grubb's test**
- **R package – Outlier**
- **Experiment**

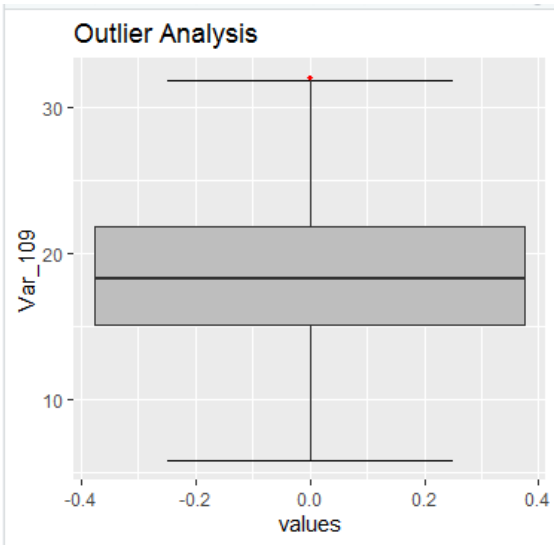
Most of the times, we use only Boxplot method which is most suitable method for all kinds of data. If we consider Grubb's test, which is the statistical technique for detecting the outliers, it is limited only to the normally distributed data. We cannot expect the normally distributed data all the time. Often the data will be skewed. Hence, it is not possible to use Grubb's test for all data. R package, "Outlier" works on the mean concept. It calculates the mean of the variable and then detects which values are falling very far from the mean. This may give some wrong answers and also it takes so much time.

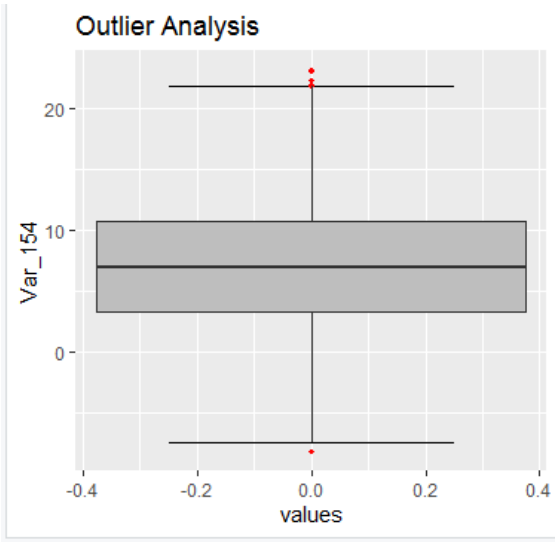
A box plot is a highly visually effective way of viewing a clear summary of one or more sets of data. It is particularly useful for quick **summarizing and comparison** of different sets of variables. At a glance, a box plot allows a graphical display of the distribution of results and provides indications of symmetry within the data.

Below are some of the graphs which show the presence of outlier in the data.









The plots clearly show that the variables contain outliers. The numeric variables are selected and plotted using box plot. The outliers are not removed in this case because they are not extreme values and fall under normal range. These outliers may contribute important information to the model for prediction. Hence, these are not removed.

2.2.Feature Selection and Dimensionality Reduction

Feature Selection is one of the core concepts in machine learning which hugely impacts the performance of the model. The data features that are used to train the machine learning models have a huge influence on the performance. Irrelevant or partially relevant features can negatively impact model performance.

In this dataset, ID_Code is a feature which does not contribute much for model prediction and learning. Hence the variable is dropped from the dataset. It is a character datatype variable which takes up huge amount of space and also requires large computational time for the model.

PCA

Principal Component Analysis, or PCA, is a dimensionality-reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process.

So to sum up, the idea of PCA is simple — reduces the number of variables of a data set, while preserving as much information as possible.

Summary of first 25 variables is listed below after performing PCA anlaysis.

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6
Standard deviation	**1.10572**	**1.02982**	**1.02943**	**1.02925**	**1.02839**	**1.02788**
Proportion of Variance	**0.00611**	**0.00530**	**0.00530**	**0.00530**	**0.00529**	**0.00528**
Cumulative Proportion	**0.00611**	**0.01142**	**0.01671**	**0.02201**	**0.02730**	**0.03258**

	PC7	PC8	PC9	PC10	PC11
Standard deviation	**1.02757**	**1.02706**	**1.02622**	**1.02595**	**1.02524**
Proportion of Variance	**0.00528**	**0.00527**	**0.00527**	**0.00526**	**0.00526**
Cumulative Proportion	**0.03786**	**0.04314**	**0.04840**	**0.05366**	**0.05892**

	PC12	PC13	PC14	PC15	PC16	PC17
Standard deviation	**1.02469**	**1.02431**	**1.02341**	**1.02331**	**1.02323**	**1.02247**
Proportion of Variance	**0.00525**	**0.00525**	**0.00524**	**0.00524**	**0.00524**	**0.00523**
Cumulative Proportion	**0.06417**	**0.06942**	**0.07465**	**0.07989**	**0.08512**	**0.09035**

	PC18	PC19	PC20	PC21	PC22
Standard deviation	**1.02214**	**1.02169**	**1.02159**	**1.02127**	**1.02080**
Proportion of Variance	**0.00522**	**0.00522**	**0.00522**	**0.00521**	**0.00521**
Cumulative Proportion	**0.09557**	**0.10079**	**0.10601**	**0.11123**	**0.11644**

	PC23	PC24
Standard deviation	**1.02052**	**1.0200**
Proportion of Variance	**0.00521**	**0.0052**
Cumulative Proportion	**0.12164**	**0.1268**

It is used **when we need to tackle the curse of dimensionality** among data with linear relationships, i.e. where having too many dimensions (features) in your data causes noise and difficulties. Almost all the variables contribute equally to the model and hence there is no multicollinearity in the variables. It is not necessary to reduce the variables Var_0 to Var_199.

2.3.Feature Scaling

Owing to the mere greater numeric range, the impact on response variables by the feature having greater numeric range could be more than the one having less numeric range, and this could, in turn, **impact prediction accuracy**. **The objective is to improve predictive accuracy** and not allow a particular feature impact the prediction due to large numeric value range. Thus, we may need to normalize or scale values under different features such that they fall under common range.

Data normalization is the process of rescaling one or more attributes to the range of 0 to 1. This means that the largest value for each attribute is 1 and the smallest value is 0. Normalization is a good technique to use when you do not know the distribution of your data or when you know the distribution is not Gaussian (a bell curve). **Normalization** usually

means to scale a variable to have values between 0 and 1. The formula used for this is as follows.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

2.4.Sampling

Sampling is the selection of a subset (a statistical sample) of individuals from within a statistical population to estimate characteristics of the whole population.

- **Population parameter**. A population parameter is the true value of a population attribute.
- **Sample statistic**. A sample statistic is an estimate, based on sample data, of a population parameter.

The quality of a sample statistic (i.e., accuracy, precision, representativeness) is strongly affected by the way that sample observations are chosen; that is., by the sampling method. Some of the Sampling methods are

- Simple random sampling
- Stratified sampling
- Systematic sampling

For this dataset, simple random sampling is used. The dataset is divided into train and test data. 80% of the data is separated for training the data and the remaining 20% is for testing the data.

The process of training an ML model involves providing an ML algorithm (that is, the learning algorithm) with training data to learn from. The term ML model refers to the model artefact that is created by the training process. The training data must contain the correct answer, which is known as a target or target attribute. The learning algorithm finds **patterns in the training data** that map the input data attributes to the target (the answer that you want to predict), and it outputs an ML model that captures these patterns.

'CreateDataPartition' is a function that is used in R for sampling.

Chapter 3

3. Modeling

In the previous sections we have done all the pre-processing steps in the dataset to develop the model. Now, as our problem statement is to predict the target, whether the customer will make the specific transaction in the future or not. The target variable is **categorical** in nature and so we build models for **Classification analysis**. Always, we have move from simple to complex. Hence, the first model that we are going to build is Logistic Regression. And then we move on to complex algorithms, XG Boost, Naïve Bayes, KNN classifier, Decision Tree and Random Forest.

3.1.Logistic Regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. At the center of the logistic regression analysis is the task estimating the log odds of an event. Mathematically, logistic regression estimates a multiple linear regression function defined as:

$$\log(P(y=1) / 1-P(y=1))=\log(P(y=1) / P(y=0))=\beta_0+\beta_1 x_1+\ldots+\beta_p x_p$$

First few rows of the Summary of the model is

Call:
glm(formula = target ~ ., family = "binomial", data = train)

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.5985	-0.3998	-0.2320	-0.1231	3.7877

Coefficients:

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-5.345971	0.464605	-11.506	< 2e-16 ***
var_0	1.083047	0.062352	17.370	< 2e-16 ***
var_1	0.985834	0.060612	16.265	< 2e-16 ***
var_2	1.170267	0.062170	18.824	< 2e-16 ***
var_3	0.225976	0.062810	3.598	0.000321 ***
var_4	0.267622	0.069010	3.878	0.000105 ***
var_5	0.753894	0.061219	12.315	< 2e-16 ***
var_6	1.620805	0.067435	24.035	< 2e-16 ***
var_7	-0.028072	0.063356	-0.443	0.657706
var_8	0.402191	0.060274	6.673	2.51e-11 ***
var_9	-0.782492	0.055871	-14.005	< 2e-16 ***
var_10	-0.001831	0.069494	-0.026	0.978981
var_11	0.578474	0.069989	8.265	< 2e-16 ***

var_12	**-1.437814**	**0.061140**	**-23.517**	**< 2e-16 *****
var_13	**-1.143851**	**0.058678**	**-19.494**	**< 2e-16 *****
var_14	**-0.121581**	**0.060078**	**-2.024**	**0.042998 ***
var_15	**0.341406**	**0.065500**	**5.212**	**1.87e-07 *****
var_16	**0.174222**	**0.065339**	**2.666**	**0.007666 ****
var_17	**0.005504**	**0.075716**	**0.073**	**0.942052**
var_18	**0.875680**	**0.064251**	**13.629**	**< 2e-16 *****
var_19	**0.169875**	**0.057432**	**2.958**	**0.003098 ****
var_20	**-0.440363**	**0.060554**	**-7.272**	**3.54e-13 *****
var_21	**-1.415263**	**0.069776**	**-20.283**	**< 2e-16 *****
var_22	**1.422415**	**0.066876**	**21.269**	**< 2e-16 *****
var_23	**-0.597454**	**0.067468**	**-8.855**	**< 2e-16 *****
var_24	**0.658030**	**0.066804**	**9.850**	**< 2e-16 *****

Signif. codes: 0 '*' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1**

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 104312 on 159999 degrees of freedom
Residual deviance: 73747 on 159799 degrees of freedom
AIC: 74149

Number of Fisher Scoring iterations: 6

The difference between Null Deviance and Residual Deviance of this model is 30,565 which are high. It means that the model is a good fit.

3.2.XG Boost Classifier

XGBoost is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

XGBoost is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models. The tree ensemble model includes functions as parameters and cannot be optimized using traditional optimization methods in **Euclidean space**.

The model summary is as follows.

summary(xgb1)

	Length	**Class**	**Mode**
handle	**1**	**xgb.Booster.handle**	**externalptr**
raw	**452769**	**-none-**	**raw**

niter **1** **-none-** **numeric**

evaluation_log **3** **data.table** **list**

call **9** **-none-** **call**

params **11** **-none-** **list**

callbacks **2** **-none-** **list**

feature_names **200** **-none-** **character**

nfeatures **1** **-none-** **numeric**

In XGB, hyper-parameter optimization (i.e., tuning) aims at searching for the hyper parameter values that minimizes the objective function. RS is one of the hyper parameter optimization. It means the hyper-parameters are randomly picked from the pre-defined searching domain uniformly and the searching does not depend on the previous boosting result.

3.3.Naïve Bayes Classifier

The Naive Bayes Classifier technique is based on the so-called **Bayesian theorem** and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods. Naive Bayes classifiers can handle an arbitrary number of independent variables whether continuous or categorical. Given a set of variables, X = {x1,x2,x...,xd}, we want to construct the posterior probability for the event Cj among a set of possible outcomes C = {c1,c2,c...,cd}. In a more familiar language, X is the predictors and C is the set of categorical levels present in the dependent variable. Using Bayes' rule:

$$p(C_j \mid x_1, x_2, \ldots, x_d) \propto p(x_1, x_2, \ldots, x_d \mid C_j) p(C_j)$$

where p(Cj | x1,x2,x...,xd) is the posterior probability of class membership, i.e., the probability that X belongs to Cj. Since Naive Bayes assumes that the **conditional probabilities** of the independent variables are statistically independent we can decompose the likelihood to a product of terms:

$$p(X \mid C_j) \propto \prod_{k=1}^{d} p(x_k \mid C_j)$$

3.4.Decision Tree Regression

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision**

nodes and **leaf nodes**. A decision node has two or more branches, each representing values for the attribute tested. Leaf node represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.

'C5.0' is the function used in R for performing Decision Tree for Classification. '**Tree.DecisionTreeClassifier'** is the function used in Python.

3.5.Random Forest

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an **ensemble**. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction. A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. The **low correlation** between models is the key. Just like how investments with low correlations (like stocks and bonds) come together to form a portfolio that is greater than the sum of its parts, uncorrelated models can produce ensemble predictions that are more accurate than any of the individual predictions. The reason for this wonderful effect is that the trees protect each other from their individual errors.

'Randomforest()' is the function that is used for performing Random Forest classification. The number of trees is set to 500, which is the default in R.

Chapter 4

4. Error Metrics

Predictive Modeling works on constructive feedback principle. You build a model. Get feedback from metrics, make improvements and continue until you achieve a desirable accuracy. Evaluation metrics explain the **performance of a model**. An important aspect of evaluation metrics is their capability to discriminate among model results. Simply, building a predictive model is not your motive. But, creating and selecting a model which gives high accuracy on out of sample data. Hence, it is crucial to check accuracy of the model prior to computing predicted values.

There are several ways to evaluate the model. In this project, I have used Confusion Matrix, Accuracy, Precision, False Negative Rate, AUC values.

4.1.Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. The confusion matrix itself is relatively simple to understand, but the related terminology can be confusing.

log_Predictions1

	0	1
0	**35494**	**2986**
1	**473**	**1047**

XG_Predictions

	0	1
0	**35510**	**457**
1	**3085**	**948**

NB_Predictions

	0	1
0	**35421**	**558**
1	**2517**	**1504**

C50_Predictions

	0	1
0	**40859**	**4108**
1	**4087**	**946**

RF_Predictions

	0	1
0	**44962**	**5**
1	**5003**	**30**

4.2.Accuracy

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

Results:

Models	Accuracy
Logistic Regression	91.40%
XG Boost	**91.50%**
Naïve Bayes	92.31%
Decision Tree	89.74%
Random Forest	89.93%

Almost all the models are performing well with respect to Accuracy.

4.3.Precision

Precision attempts to answer the following question: "What proportion of positive identifications was actually correct?"

Precision is defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Models	Precision
Logistic Regression	92.24%
XG Boost	**98.73%**
Naïve Bayes	70.96%
Decision Tree	18.71%
Random Forest	85.71%

From all the models, the XG Boost and Logistic Regression models have the highest Precision.

4.4.False Negative Rate

The **false negative rate** is the proportion of positives which yield **negative** test outcomes with the test, i.e., the conditional probability of a **negative** test result given that the condition being looked for is present.

False negative rate (β) = type II error
= 1 − sensitivity
= FN / (TP + FN)

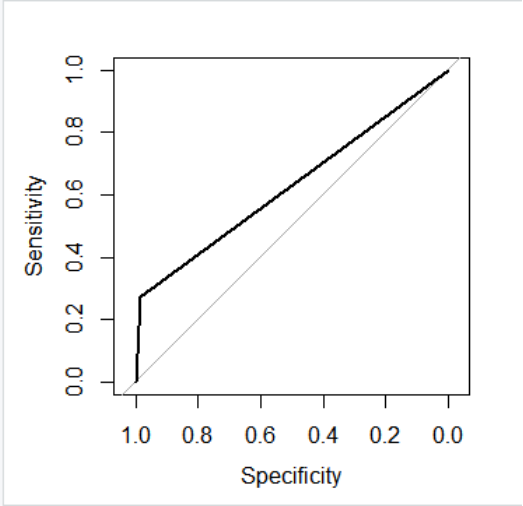
Models	FNR
Logistic Regression	72.54%
XG Boost	**27.97%**
Naïve Bayes	62.59%
Decision Tree	83.86%
Random Forest	99.88%

From the False Negative Rates, the XG Boost model is selected as the best fit since it contains lowest FNR. This means that the model is reliable to a greater extent than other models.

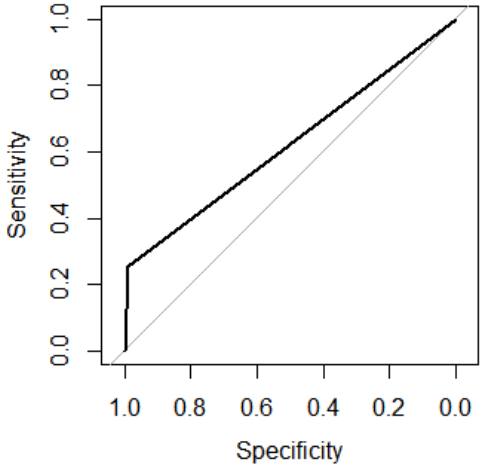
4.5.AUC ROC values

When we need to check or visualize the performance of the multi - class classification problem, we use AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve. It is one of the most important evaluation metrics for checking any classification model's performance. It is also written as AUROC (Area Under the Receiver Operating Characteristics). ROC is a probability curve and AUC represents degree or measure of separability. It tells how much model is capable of distinguishing between classes. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

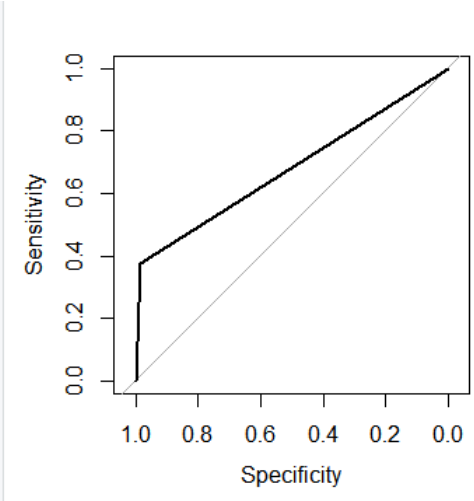
Logistic Regression



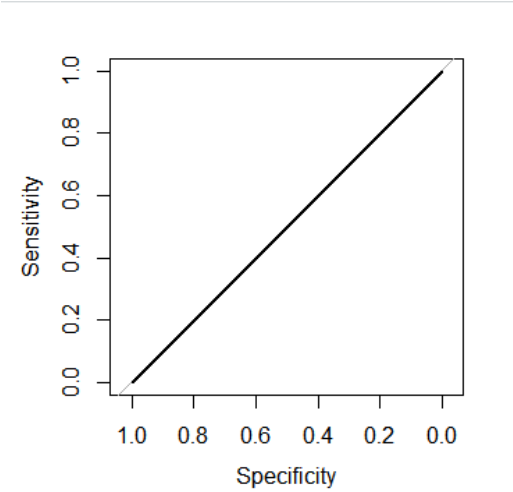
XG Boost



Naïve Bayes



Random Forest



Models	AUC Score
Logistic Regression	0.63
XG Boost	**0.82**
Naïve Bayes	0.67
Decision Tree	0.53
Random Forest	0.50

Using AUC Score, we can get into the conclusion to choose XG Boost as best model fit.

Chapter 5

5. Conclusion

From all the Error metrics, it is arrived at a conclusion that XG Boost performs better for this dataset when compared with all other classification models. It has the highest Accuracy and Precision. It has the lowest False Negative Rate. It has the highest Area under the curve score.

Accuracy = 91.5%`

FNR = 27.97%

AUC value = 0.821

The test dataset is predicted using the XG Boost model classifier. The first few observations of the predicted dataset is shown below.

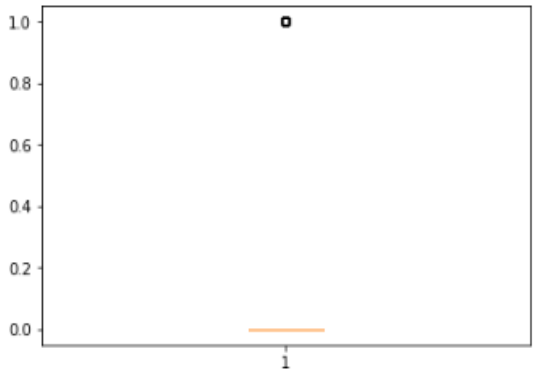
```
In [48]: data_test.head(20)
```

Out[48]:

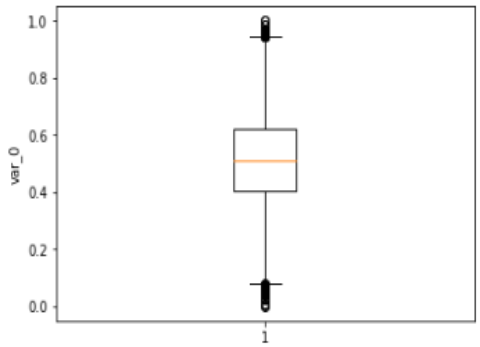
r_4	var_5	var_6	var_7	var_8	var_9	...	var_191	var_192	var_193	var_194	var_195	var_196	var_197	var_198	var_199	Predictions
576	0.563885	0.596918	0.555992	0.616146	0.675517	...	0.745496	0.168422	0.477313	0.233148	0.780223	0.613081	0.745210	0.445426	0.456290	0
353	0.527653	0.624899	0.572118	0.282491	0.255995	...	0.587858	0.377335	0.741635	0.333622	0.569846	0.417179	0.606236	0.628698	0.274352	0
313	0.834554	0.440118	0.643958	0.585037	0.606615	...	0.700735	0.468998	0.467990	0.193832	0.744399	0.228884	0.146114	0.667101	0.241644	0
535	0.687271	0.447462	0.657790	0.679433	0.475497	...	0.600497	0.440591	0.516235	0.312664	0.854102	0.599396	0.501114	0.322375	0.523223	0
955	0.426075	0.762902	0.216621	0.659735	0.429034	...	0.605501	0.407583	0.509229	0.547880	0.489135	0.294091	0.183191	0.367939	0.449389	0
318	0.424047	0.333596	0.400791	0.603165	0.449817	...	0.498068	0.257009	0.698596	0.299542	0.285155	0.265224	0.596459	0.527638	0.746406	0
420	0.456672	0.320501	0.326284	0.288235	0.306909	...	0.484906	0.655567	0.697573	0.437285	0.522496	0.726220	0.488241	0.595839	0.662177	0
035	0.830993	0.615041	0.537400	0.262426	0.476458	...	0.357069	0.376453	0.444558	0.316833	0.329777	0.685614	0.706806	0.845884	0.596541	0
571	0.311108	0.348696	0.392868	0.639314	0.440543	...	0.537684	0.512141	0.391690	0.394915	0.532287	0.418933	0.614310	0.521314	0.536292	0
508	0.376831	0.202823	0.645194	0.435481	0.369333	...	0.704855	0.372963	0.557091	0.310245	0.583613	0.318082	0.558318	0.462585	0.724462	0
306	0.682682	0.542548	0.439136	0.650266	0.696419	...	0.481010	0.345667	0.319421	0.746857	0.611771	0.623806	0.649586	0.277132	0.548651	0
712	0.483929	0.547986	0.432165	0.498667	0.499164	...	0.338301	0.280912	0.667631	0.488500	0.529950	0.205023	0.502096	0.425375	0.476660	0
951	0.049531	0.373458	0.418728	0.669235	0.492419	...	0.355317	0.469747	0.369972	0.442634	0.437905	0.685976	0.456962	0.518567	0.376751	0
363	0.728192	0.449943	0.794496	0.690421	0.238347	...	0.510529	0.579980	0.527730	0.391710	0.562793	0.394426	0.633504	0.701920	0.675043	0
535	0.437283	0.422619	0.413861	0.540321	0.092126	...	0.287107	0.532730	0.426686	0.508419	0.429266	0.813323	0.333776	0.552545	0.572160	0
798	0.650501	0.463252	0.342737	0.377856	0.769448	...	0.431908	0.722076	0.479076	0.723921	0.420775	0.757716	0.673938	0.546706	0.675984	0
972	0.129869	0.903140	0.582893	0.531254	0.532817	...	0.434690	0.410272	0.561501	0.577748	0.636228	0.725353	0.519308	0.731197	0.555895	0
464	0.535107	0.399484	0.637386	0.597544	0.555316	...	0.529657	0.334641	0.489257	0.652141	0.494158	0.741071	0.514526	0.825407	0.551334	0

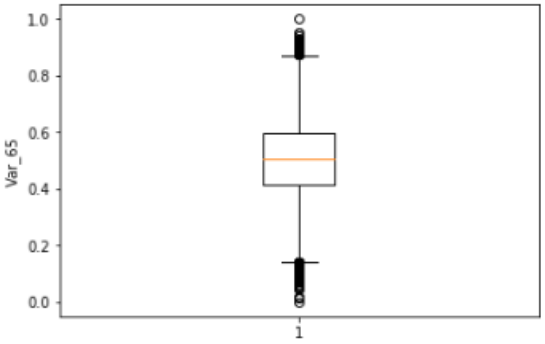
6. Appendix : Graphs

Distribution of the Target variable in the dataset

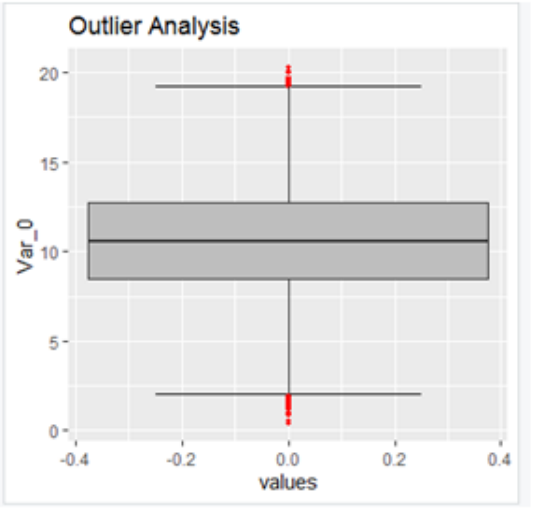


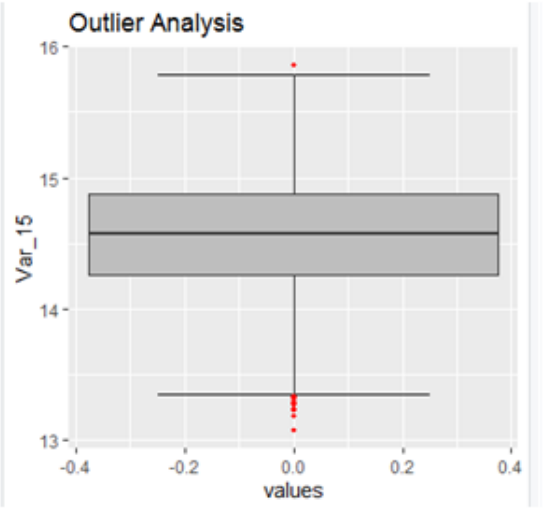
Outlier Analysis of variables ‘var_0’ and ‘var_65



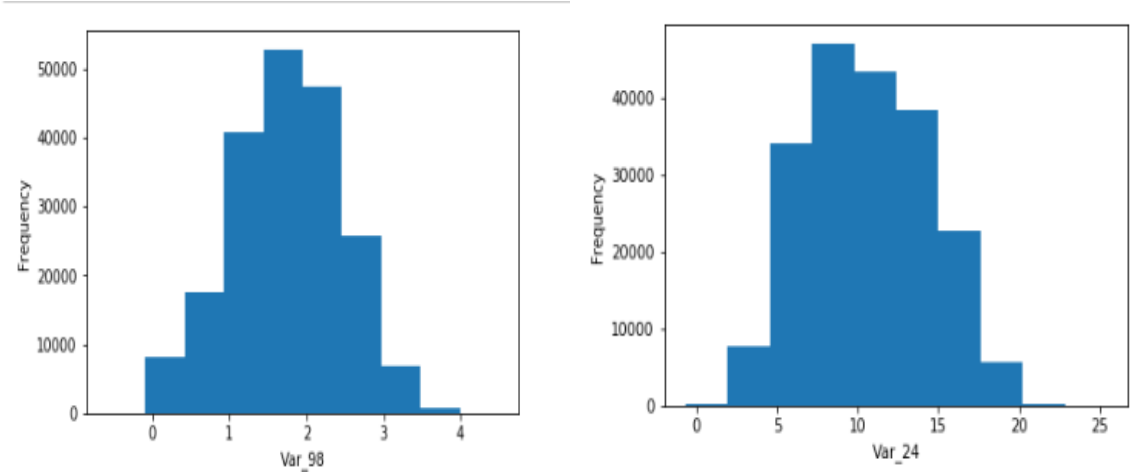


Outlier Analysis of ‘var_0’ and ‘var_15’

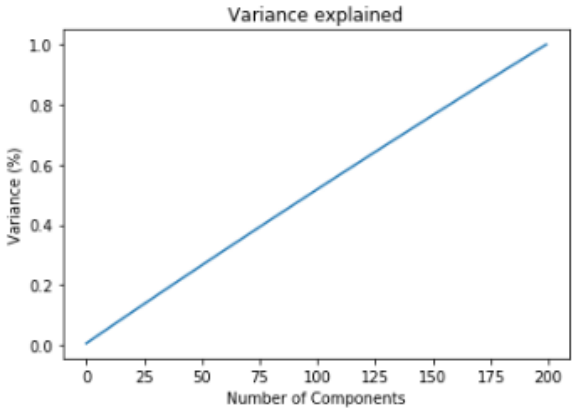




Frequency Histograms of 'var_98' and 'var_24'



Variance explanation of the components in PCA anlysis



Graph between 'First component' and 'Second component' in PCA