Santander Customer Transaction Prediction

Akash Ingole

20 July 2020

Table of Contents

Chapter 1 – Introduction

1.1 Introduction

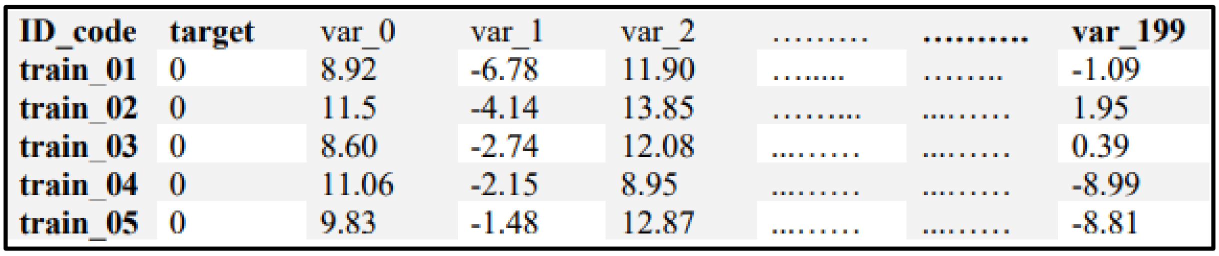
At Santander, our mission is to help people and businesses prosper. We are always looking for ways to help our customers understand their financial health and identify which products and services might help them their monetary goals.

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 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.

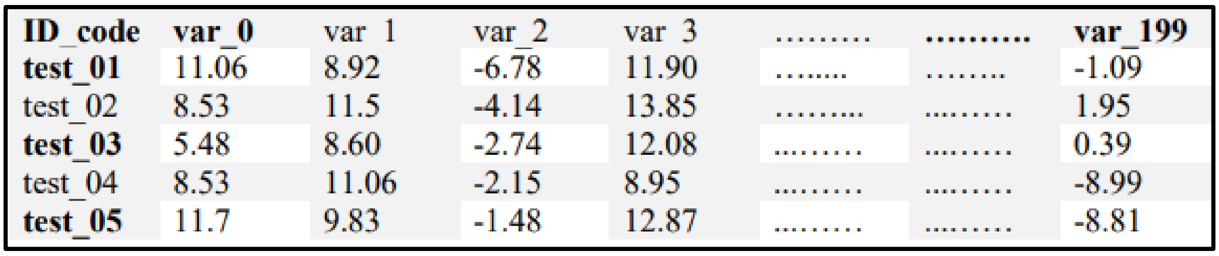
1.2 Data

In this project, our task is to build classification models which will be sued to predict which customers will make a specific transaction in future. Given below is a sample of the Santander Customer Transaction Dataset:



ID_code	**target**	var_0	var_1	var_2	....	**..........**	**var_199**
train_01	0	8.92	-6.78	11.90	....	....	-1.09
train_02	0	11.5	-4.14	13.85	....	....	1.95
train_03	0	8.60	-2.74	12.08	....	....	0.39
train_04	0	11.06	-2.15	8.95	....	....	-8.99
train_05	0	9.83	-1.48	12.87	....	....	-8.81

Figure 1: Test Dataset



ID_code	**var_0**	var_1	var_2	var_3	....	**..........**	**var_199**
test_01	11.06	8.92	-6.78	11.90	....	....	-1.09
test_02	8.53	11.5	-4.14	13.85	....	....	1.95
test_03	5.48	8.60	-2.74	12.08	....	....	0.39
test_04	8.53	11.06	-2.15	8.95	....	....	-8.99
test_05	11.7	9.83	-1.48	12.87	....	....	-8.81

Figure 2: Train Dataset

Chapter 2 – Methodology

2.1 Exploratory Data Analysis (EDA)

Exploratory Data Analysis is a philosophy as to how we dissect the data set, to maximise insights about the data set, what we are looking for and how we interpret it.

Most EDA techniques are graphical in nature, with a few quantitative / statistical techniques. The reason for heavy reliance on graphics is that by its very nature the main role of EDA is to open-mindedly explore, the graphics gives the data scientists the unparalleled power to do so, enticing the data to reveal its structural secrets, and being always ready to gain some new, often unsuspected, insight into the data. In combination with the natural pattern recognition capabilities that we all possess, graphics provides unparalleled power to carry this out.

2.2 Target Variable Class Count (Categorical Data)

The target variable contains binary data, wherein only two classes are present

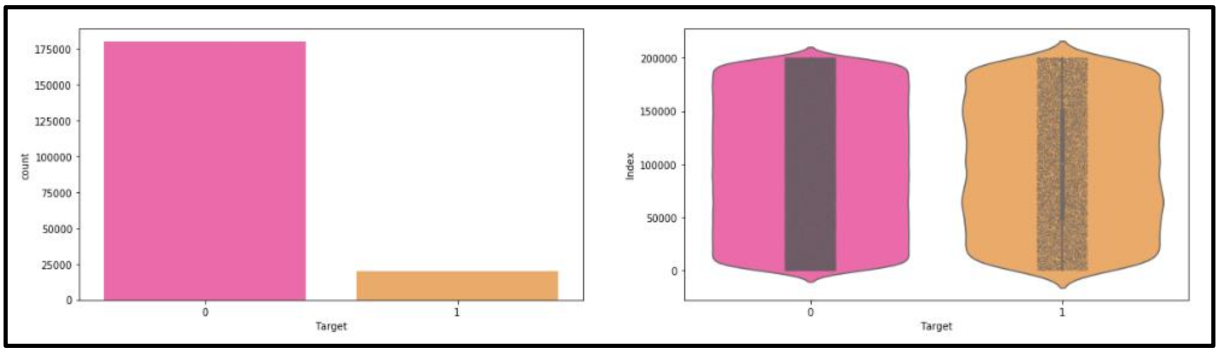


Figure 3: Visualization of target variable using count plot and violin plot.

- We have an unbalanced dataset, where 91 % of data belongs to customers who will not make any transaction, whereas, only 10 % of data belongs to customers who will make the transaction.
- Also, from the violin plots it is evident that, the target variable is more dominated with zeros as compared to ones.
- From jitter plots within violin plots, it can be discerned that target variable appears to be uniformly distributed over the indexes of the dataframe.

2.3 Missing Value Analysis

If the missing values are not handled properly, the data scientist may end up drawing incorrect inferences about the data. If the missing data is less than 5% of the total dataset, then the data scientist may choose to drop the missing / incomplete data. Another way is to impute the missing values using mean, median or using KNN imputation.

There are no missing data in both, training and testing datasets.

2.4 Distribution of Attributes

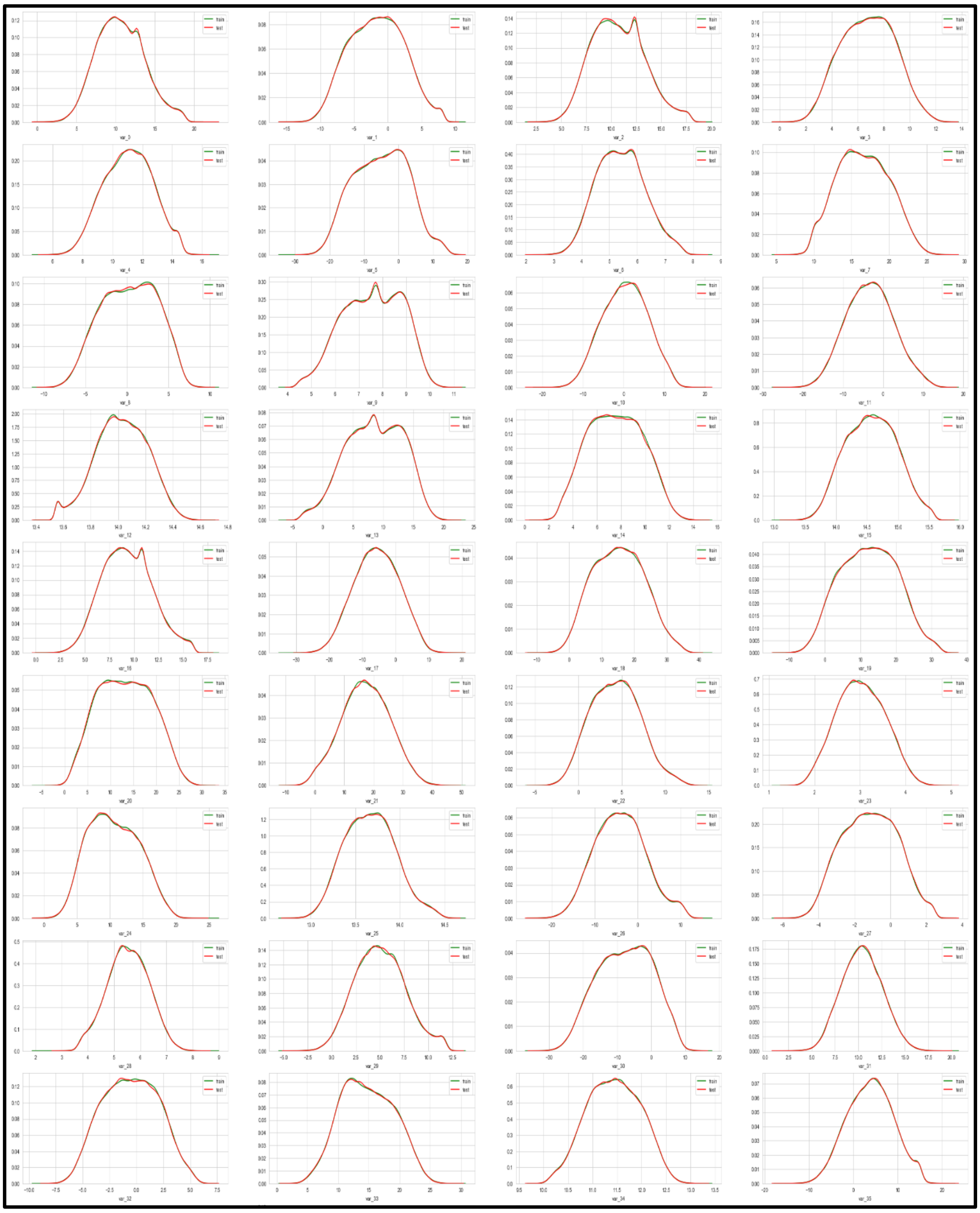


Figure 4: Distribution of first 37 attributes, from var_0 to var_36.

2.4 Outlier Analysis

Outlier analysis is carried out using Chauvenet's criterion. The method works by creating an acceptable band of data around the mean, specifying any values that fall outside that band should be eliminated.

To apply Chauvenet's criterion, we first need to calculate the mean and standard deviation of the data point, and then compute the Z-score.

2.5 Feature Selection

Feature selection is the process where you automatically or manually select those features which contribute most to your prediction variable. Having irrelevant features in your data may decrease the accuracy of the model and make your model learn based on irrelevant features.

Feature selection enables machine learning algorithms to train faster. It reduces the complexity of model and makes it easier to interpret. It improves the accuracy of the model if right subset is chosen. In this project, we are looking at correlation matrix for feature selection.

2.5.1 Correlation Matrix

The correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables.

For correlation distribution plot, it is observed that correlation between both train and test attributes is very small.

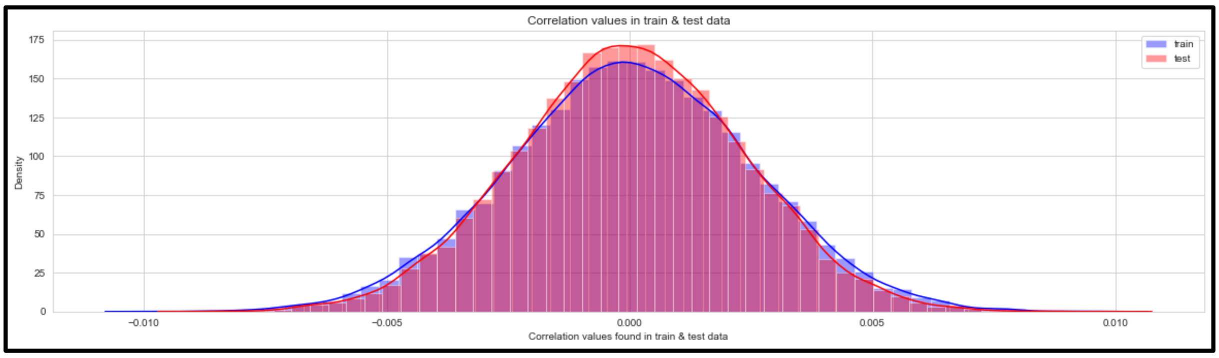


Figure 5: Correlation values found in train and test data.

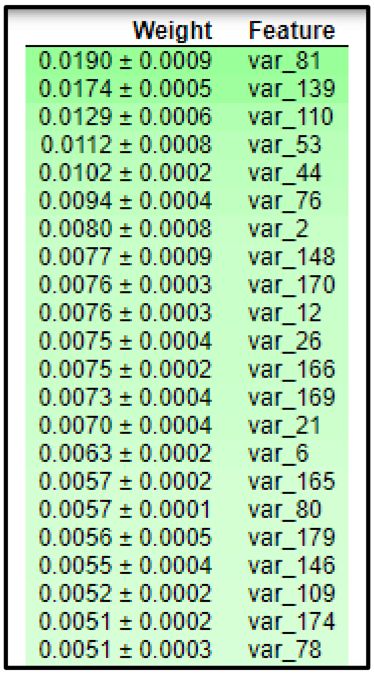
2.6 Feature Engineering

Feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data. We can perform feature engineering using:

1. Permutation Importance
2. Partial Dependence Plots

2.6.1 Permutation Importance

Permutation importance is defined to be the decrease in model score when a single feature value is randomly shuffled. This procedure breaks the relationship between the feature and the target, thus a drop in the model score is indicative of how much the model depends on the feature.



Weight	Feature
0.0190 ± 0.0009	var_81
0.0174 ± 0.0005	var_139
0.0129 ± 0.0006	var_110
0.0112 ± 0.0008	var_53
0.0102 ± 0.0002	var_44
0.0094 ± 0.0004	var_76
0.0080 ± 0.0008	var_2
0.0077 ± 0.0009	var_148
0.0076 ± 0.0003	var_170
0.0076 ± 0.0003	var_12
0.0075 ± 0.0004	var_26
0.0075 ± 0.0002	var_166
0.0073 ± 0.0004	var_169
0.0070 ± 0.0004	var_21
0.0063 ± 0.0002	var_6
0.0057 ± 0.0002	var_165
0.0057 ± 0.0001	var_80
0.0056 ± 0.0005	var_179
0.0055 ± 0.0004	var_146
0.0052 ± 0.0002	var_109
0.0051 ± 0.0002	var_174
0.0051 ± 0.0003	var_78

Figure 6: Feature and their weights calculated using permutation importance.

2.6.2 Partial Dependence Plot

Partial dependence plot gives the graphical depiction of marginal effect of a variable on the class probability or classification. While feature importance shows which features affect the predictions the most, partial dependence plot shows how a feature affects the prediction.

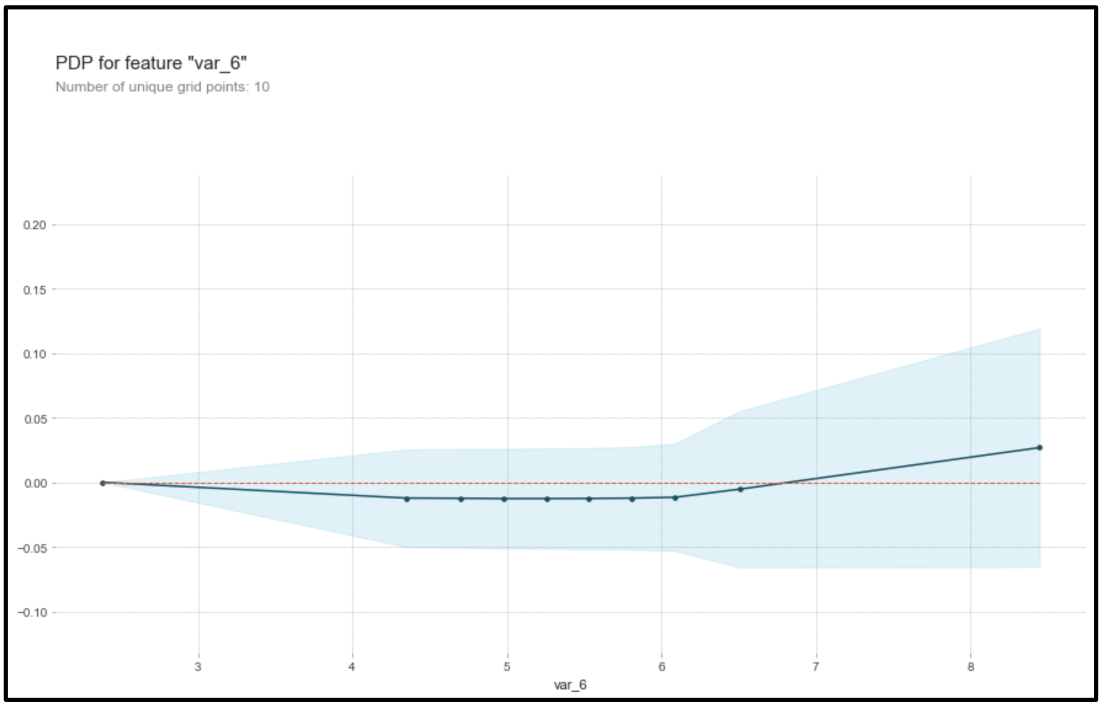


Figure 7: Partial dependence plot for var_6

The blue shaded area indicates the level of confidence for var_6.

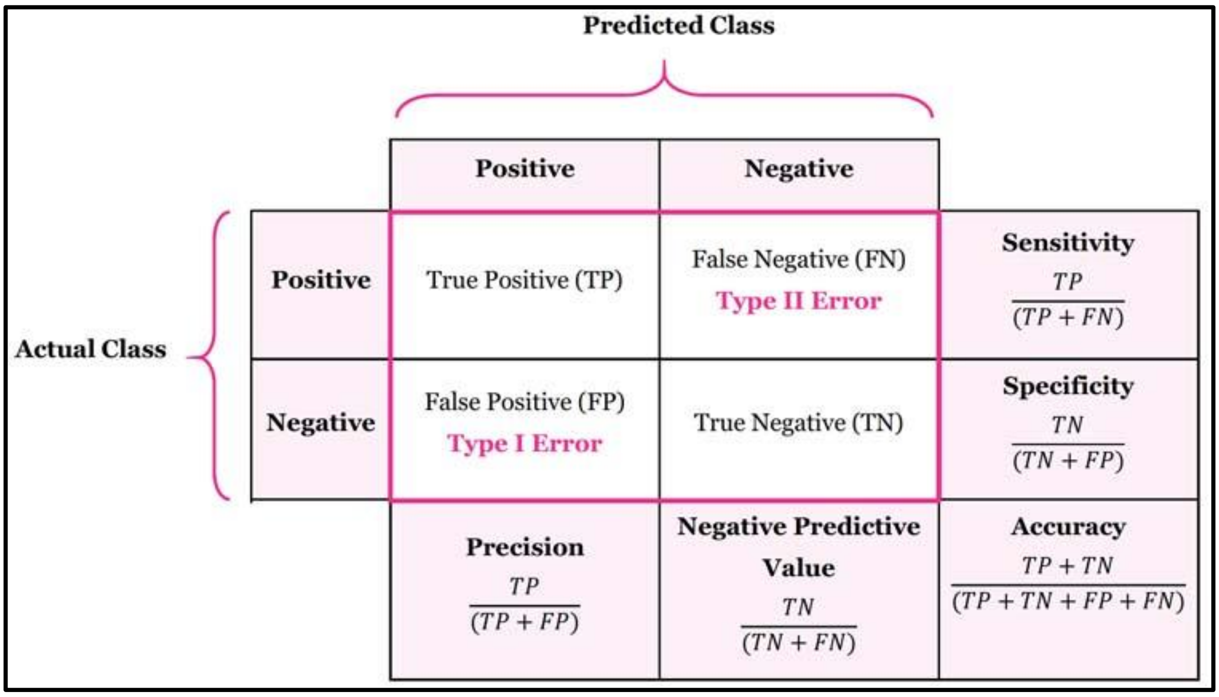
Please note that PDPs assume that target features are independent from the complement features and this assumption is often violated in practice.

2.7 Handling Imbalanced Data

Machine learning algorithms tend to produce unsatisfactory results when faced with unbalanced datasets. The conventional model evaluation methods do not accurately measure model performance when faced with imbalanced datasets.

Standard classifier algorithms like decision trees and logistic regression have bias towards classes with number of instances. They tend to predict only the majority class data and the features of minority class data are treated as noise and are often ignored. Thus, there is a high probability of misclassification of minority class as compared to majority class.

Evaluation of classification algorithm performance is measured by confusion matrix which contains information about actual and predicted classes.



		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	Negative Predictive Value $\frac{TN}{(TN+FN)}$	Accuracy $\frac{TP+TN}{(TP+TN+FP+FN)}$

Figure 8: Confusion Matrix

The main objective of balancing classes is to either increase the frequency of minority class or decrease the frequency of majority class. This is done to obtain approximately same number of instances for both the classes. Let us take a look at few resampling techniques:

2.7.1 Random Under-Sampling

Random under-sampling aims to balance class distribution by randomly eliminating majority class examples. This is done until the majority and minority instances balances out.

- **Advantages:**
  - It can help improve runtime and storage problems by reducing the number of training data samples when training dataset is huge.
- **Disadvantages:**
  - It can discard potentially useful information which could be important for building important classifiers.

    - The sample chosen by random under-sampling may be a biased sample. And it will not be an accurate representative of a population. Thereby, resulting in inaccurate results with the actual test dataset.

2.7.2 Random Over-Sampling

Over-sampling increases the instances of minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.

- **Advantages:**
    - Unlike under-sampling this method leads to no information loss.
    - Outperforms under-sampling.
- **Disadvantages:**
    - It increases the likelihood of overfitting since it replicates the minority class events.

2.7.3 Synthetic Minority Over-Sampling Technique

This technique is used to avoid overfitting which occurs when exact replicas of minority instances are added to the main dataset. A subset of data is taken from the minority class as an example and then new synthetic similar instances are created. These synthetic instances are then added to original dataset. The new dataset is used as a sample to train the classification models.

- **Advantages:**
    - Mitigates the problem of overfitting caused by random over-sampling as synthetic samples are generated rather than replication of instances.
    - No loss of useful information.
- **Disadvantages:**
    - While generating synthetic samples SMOTE does not take into consideration neighbouring examples from other classes. This can result in increase in overlapping of classes and can introduce additional noise.
    - SMOTE is not very effective for high dimensional data.

Chapter 3 – Modelling

3.1 Model Selection

Model selection is the process of choosing between machine learning approaches, e.g. SVM, logistic regression, etc. or choosing between different hyperparameters or sets of features for the same machine learning approaches. The optimal model is the one that fits the data with best values for the evaluation metrics.

3.2 Logistic Regression

Logistic regression is a statistical technique used to predict probability of binary response based on one or more independent variables. It is used when the output variable is “binary” or “dichotomous”. Logistic regression fits a single line to divide the space into two and so performs better than decision tree when data is distributed in a fashion such that it can be linearly classified.

- Assumptions
  - The dependent variable should be dichotomous in nature.
  - There should be no outliers in data.
  - There should be no high correlations (multicollinearity) among the predictors.

The logistic function is given by:

$$f(x) = \frac{L}{1 + e^{-k(x-x0)}}$$

Where:

- ***L =*** curve’s maximum value
- ***k*** = steepness of the curve
- ***x0*** = x value of sigmoid’s midpoint

Sigmoid’s function is a standard logistic function (k = 1, x0 = 0, L = 1)

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

The Sigmoid function has an S shaped curve. It has a finite limit of 0 as x approaches negative infinity and 1 as x approaches positive infinity. Thus, the output of sigmoid function is not only used to classify YES or NO, it can also be used to determine the probability of YES or NO.

3.2 Random Forest

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

Decision trees that are grown very deep tend to learn highly irregular patterns, they overfit the training sets, and i.e. they have low bias and high variance. Random forests are a way of averaging multiple decision trees, trained on different parts of same training sets, with the goal of reducing the variance. This comes at the expense of small increase in bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

3.3 Light GBM (Light Gradient Boosting Machine)

Light GBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher accuracy
- Lower memory usage
- Better accuracy
- Support of parallel and GPU learning
- Capable of learning large-scale data

Benefitting from these advantages Light GBM is being widely used in many winning solutions of machine learning competitions.

Comparison experiments on public datasets show that Light GBM can outperform existing boosting frameworks on both efficiency and accuracy, with significantly lower memory consumption.

Light GBM grows tree vertically while other algorithms grows trees horizontally, meaning that Light GBM grows trees leaf-wise while other algorithms grow level-wise. It will choose tree with max delta loss to grow. When growing the same leaf, leaf-wise algorithm can reduce more loss than level-wise algorithm.

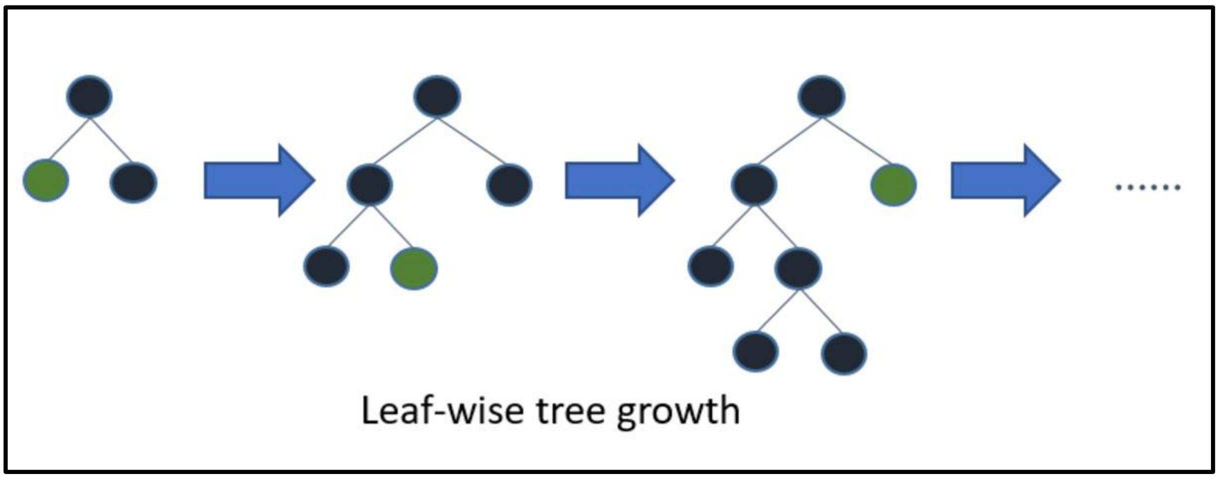


Figure 9: How Light GBM works with leaf-wise growth.

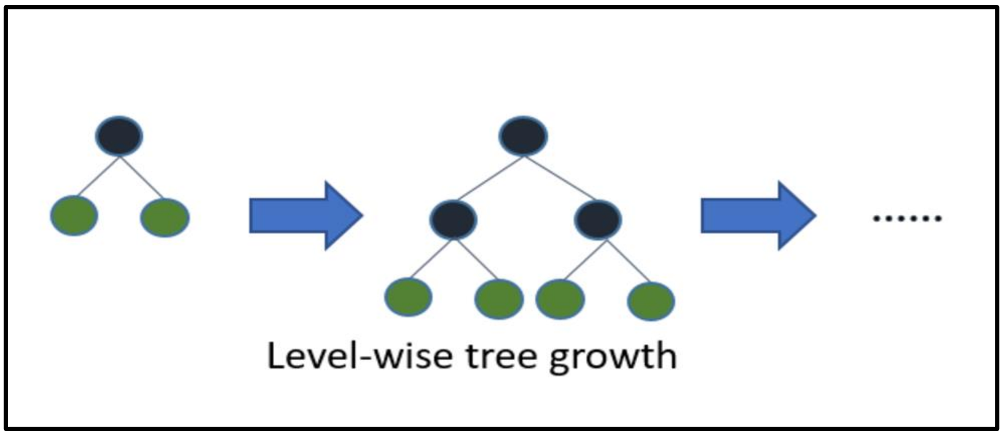


Figure 10: How other boosting frameworks work with level-wise tree growth.

The size of data is increasing day by day and it is becoming difficult for traditional data science algorithms to give faster results. Light GBM is prefixed 'Light' because of its high speed. Light GBM can handle large size of data and takes lower memory to learn.

Another reason why Light GBM is popular because it focuses on accuracy of results. Light GBM also supports GPU learning and thus data scientists are widely using LGBM for data science application development.

However, it is not advisable to use Light GBM on smaller datasets, as it is sensitive to overfitting and can easily overfit small data.

3.3.1 Parameters of Light GBM

1. **`boosting:`** Defines the type of algorithm you want to run, default = `gdbt`
    - `gdbt`: Traditional Gradient Boosting Decision Tree
    - `rf`: Random Forest
    - `dart`: Dropout meets Multiple Additive Regression Trees
    - `goss`: Gradient based One-Side Sampling

    - **`max_depth`***:* It describes the maximum depth of tree. This parameter is used to handle model overfitting. One can lower the `max_depth` if you feel that the model is overfitted

2. **`objective`***:* binary for classification problem.

3. **`metric`***:* Specifies type of measurement for model building.

4. **`num_leaves`***:* Number of leaves in a full tree, default is 31.

5. **`learning_rate`***:* This determines the impact of each tree on the final outcome. GBM works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates. Typical values are 0.1, 0.01, 0.001, etc.

6. **`max_bin`***:* It denotes the maximum number of bins that feature value will bucket in.

7. **bagging_fraction*:*** Specifies the fraction of data to be used for each iteration and is generally used to speed up training and avoid overfitting.

8. **feature_fraction*:*** Used when you are boosting is random forest. A feature_fraction of 0.8 means Light GBM will select 80% of parameters randomly in each iteration for building trees.

9. **early_stopping_round*:*** This parameter helps to speed up the analysis. The model will stop training if one metric of one validation data doesn't improve in last early_stopping_round rounds. This will reduce excessive iterations.

For faster speed:

- Use bagging by setting bagging_fraction and bagging_freq.
- Use feature sub-sampling by setting feature_fraction.
- Use small max_bin.
- Use save_binary to speedup data loading in future learning.
- Use parallel learning.

For better accuracy:

- Use large max_bin (may be slower).
- Use small learning_rate with large num_iterations.
- Use large num_leaves (may cause over-fitting).
- Use bigger training data.
- Try dart.

To deal with overfitting:

- Use small max_bin.
- Use small num_leaves.
- Use min_data_in_leaf and min_sum_hessian_in_leaf.
- Use bagging by set bagging_fraction and bagging_freq.
- Use feature sub-sampling by set feature_fraction.
- Use bigger training data.
- Try lambda_l1, lambda_l2 and min_gain_to_split to regulatization.
- Try max_depth to avoid growing deep tree.

Chapter 4 – Conclusion

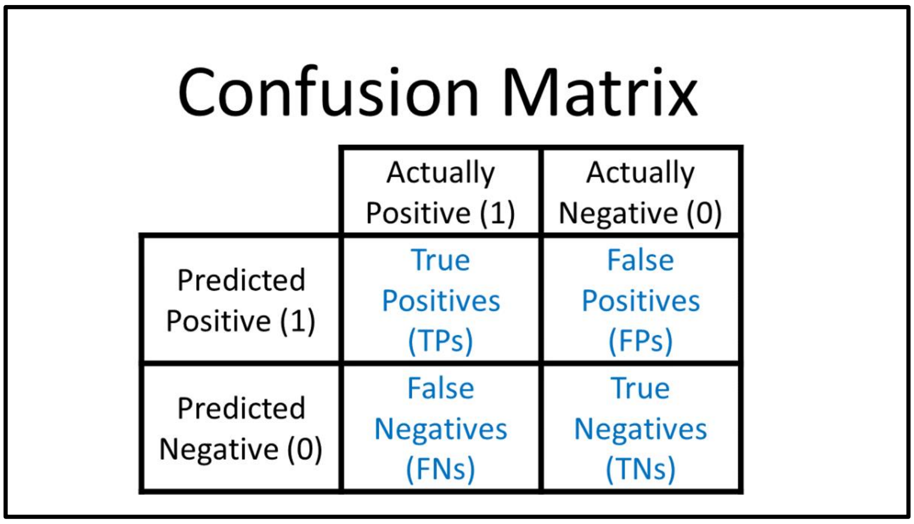
We have three models for predicting the target variable, but we are yet to decide which model is best for our dataset. There are many metrics that are used for model evaluation. And as we have in imbalanced dataset classification accuracy can even be misleading.

For classification problems, the performance metric being used is classification report. But as we have an imbalanced dataset, so ROC AUC score will be used for the evaluation of different models.

4.1 Confusion Matrix

Confusion matrix is a table that is often used to describe the performance of a classification model on a training dataset for which the values are known.

The number of correct predictions and incorrect predictions are summarized with count values and broken by each class.



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Figure 11: Confusion Matrix

4.2 Receiver Operating Characteristics (ROC) Area Under Curve (AUC) Score

ROC AUC curve is a performance metric for classification problems at various threshold settings. ROC is a probability curve and AUC represents degree or measure of classification. It tells how much the model is capable of distinguishing between the classes. Higher the AUC, better is the model at predicting the category.

The ROC curve is plotted with True Positive Rate (TPR) against the False Positive Rate (FPR) where TPR is on Y – axis and FPR is on X – axis.

- Logistic Regression:

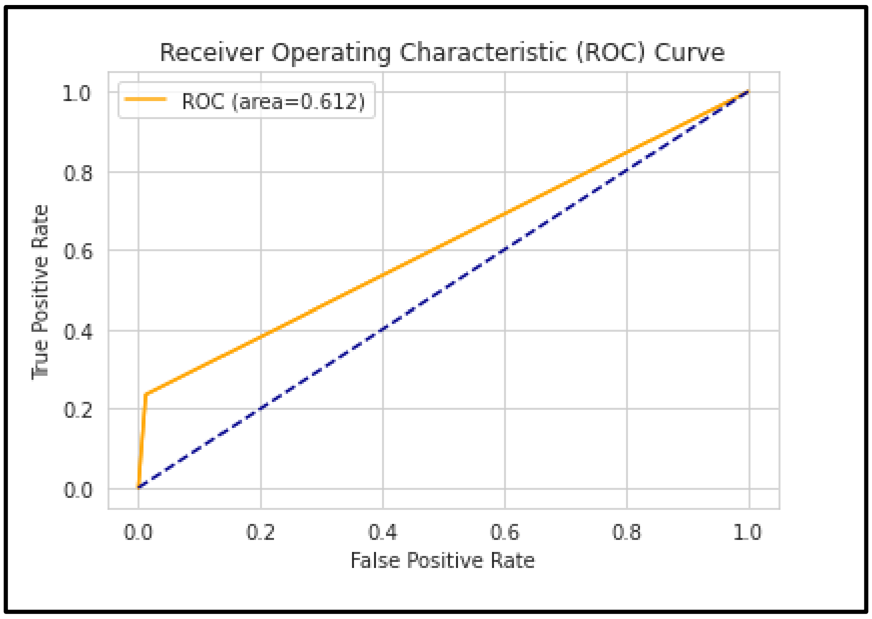


Figure 12: ROC Curve for simple Logistic Regression with AUC 0.60

- SMOTE:

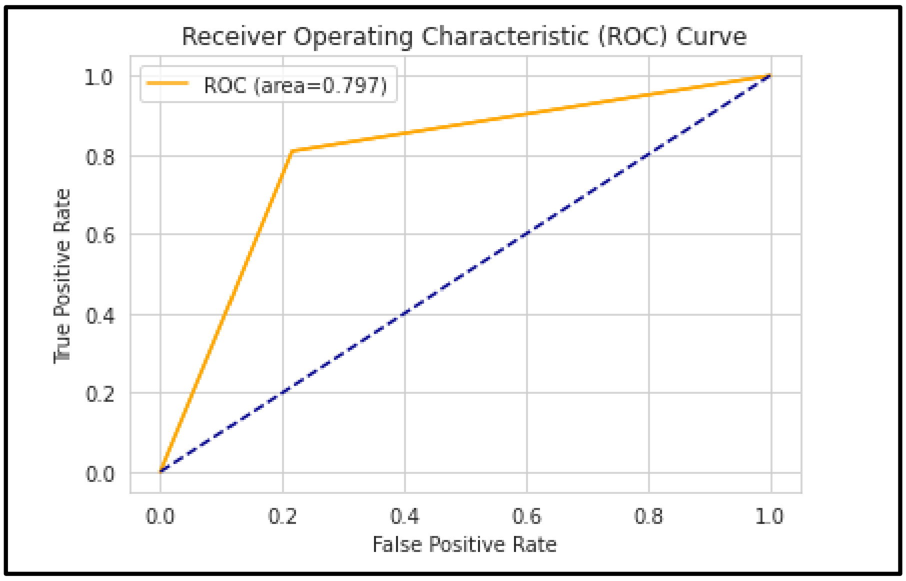


Figure 13: ROC Curve for SMOTE with AUC 0.80

- Light GBM:

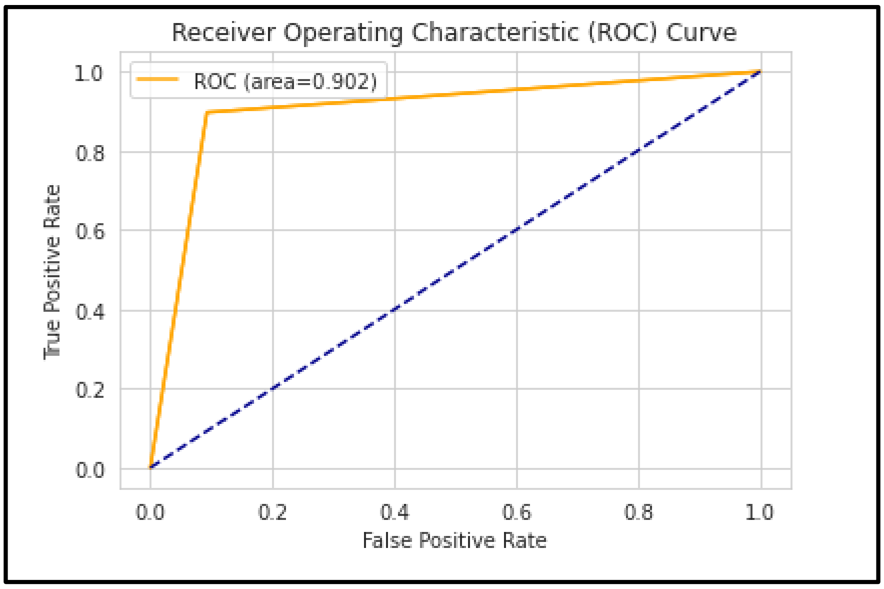


Figure 14: ROC Curve for Light GBM with AUC 0.90

Chapter 5 – Model Selection

After comparing the scores of areas under ROC curves of all the models fitted on an imbalanced dataset, we come to following conclusion:

- Logistic regression does not perform well on imbalanced dataset.
- We balance the imbalanced datasets using resampling techniques like SMOTE in Python and ROSE in R.
- Light GBM performed well on imbalanced dataset.

Thus, Light GBM is the best choice for identifying which customers will make a specific transaction is future, irrespective of the amount of money transacted.