

**MICRO-CREDIT DEFAULTER MODEL**

**Submitted By**

**K. Naveen Varma**

**Micro-Credit Defaulter Model**

**Problem Statement:**

A Microfinance Institution (MFI) is an organization that offers financial services to low income populations. MFS becomes very useful when targeting especially the unbanked poor families living in remote areas with not much sources of income. The Microfinance services (MFS) provided by MFI are Group Loans, Agricultural Loans, Individual Business Loans and so on.

Many microfinance institutions (MFI), experts and donors are supporting the idea of using mobile financial services (MFS) which they feel are more convenient and efficient, and cost saving, than the traditional high-touch model used since long for the purpose of delivering microfinance services. Though, the MFI industry is primarily focusing on low income families and are very useful in such areas, the implementation of MFS has been uneven with both significant challenges and successes.

Today, microfinance is widely accepted as a poverty-reduction tool, representing $70 billion in outstanding loans and a global outreach of 200 million clients.

We are working with one such client that is in Telecom Industry. They are a fixed wireless telecommunications network provider. They have launched various products and have developed its business and organization based on the budget operator model, offering better products at Lower Prices to all value conscious customers through a strategy of disruptive innovation that focuses on the subscriber.

They understand the importance of communication and how it affects a person’s life, thus, focusing on providing their services and products to low income families and poor customers that can help them in the need of hour.

They are collaborating with an MFI to provide micro-credit on mobile balances to be paid back in 5 days. The Consumer is believed to be defaulter if he deviates from the path of paying back the loaned amount within the time duration of 5 days. For the loan amount of 5 (in Indonesian Rupiah), payback amount should be 6 (in Indonesian Rupiah), while, for the loan amount of 10 (in Indonesian Rupiah), the payback amount should be 12 (in Indonesian Rupiah).

The sample data is provided to us from our client database. It is hereby given to you for this exercise. In order to improve the selection of customers for the credit, the client wants some predictions that could help them in further investment and improvement in selection of customers.

**Business Goal:**

Build a model which can be used to predict in terms of a probability for each loan transaction, whether the customer will be paying back the loaned amount within 5 days of insurance of loan. In this case, Label ‘1’ indicates that the loan has been payed i.e. Non- defaulter, while, Label ‘0’ indicates that the loan has not been payed i.e. defaulter.

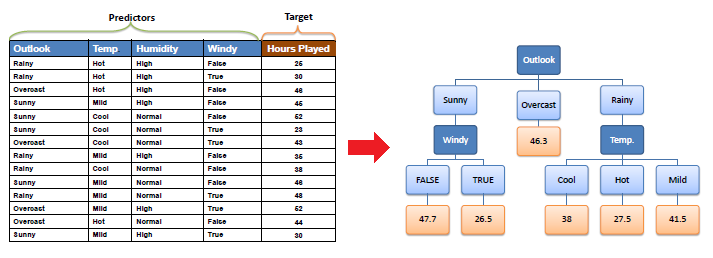
**Conceptual Background of the Domain Problem:**

**Logistic Regression:** Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a [data set](https://whatis.techtarget.com/definition/data-set). Logistic regression has become an important tool in the discipline of [machine learning](https://searchenterpriseai.techtarget.com/definition/machine-learning-ML). The approach allows an [algorithm](https://whatis.techtarget.com/definition/algorithm) being used in a machine learning application to classify incoming data based on historical data. As more relevant data comes in, the algorithm should get better at predicting classifications within data sets. Logistic regression can also play a role in [data preparation](https://searchbusinessanalytics.techtarget.com/definition/data-preparation) activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load ([ETL](https://searchdatamanagement.techtarget.com/definition/Extract-Load-Transform-ELT)) process in order to stage the information for analysis.

A logistic regression model predicts a [dependent data variable](https://whatis.techtarget.com/definition/dependent-variable) by analyzing the relationship between one or more existing independent variables. For example, a logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will be admitted to a particular college.

The resulting analytical model can take into consideration multiple input criteria. In the case of college acceptance, the model could consider factors such as the student’s grade point average, SAT score and number of extracurricular activities. Based on [historical data](https://whatis.techtarget.com/definition/historical-data) about earlier outcomes involving the same input criteria, it then scores new cases on their probability of falling into a particular outcome category.

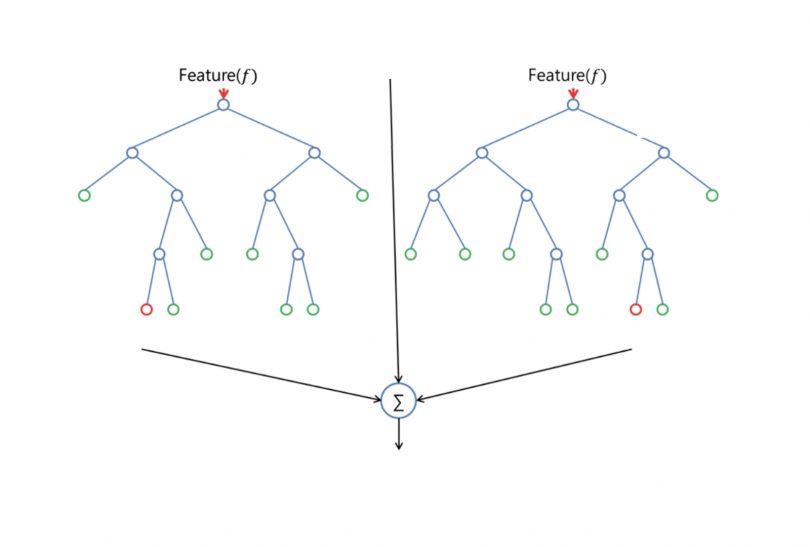
**Decision Tree Claasifier:** Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested. Leaf node (e.g., Hours Played) 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.



**Decision Tree Algorithm:** The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with *Standard Deviation* *Reduction*.

**Standard Deviation:** A decision tree is built top-down from a root node and involves partitioning the data into subsets that contain instances with similar values (homogenous). We use standard deviation to calculate the homogeneity of a numerical sample. If the numerical sample is completely homogeneous its standard deviation is zero.

**Random Forest Regression:** Random forest is a [supervised learning algorithm](https://builtin.com/data-science/supervised-learning-python). The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the [bagging method](https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies) is that a combination of learning models increases the overall result.One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning. Below you can see how a random forest would look like with two trees:

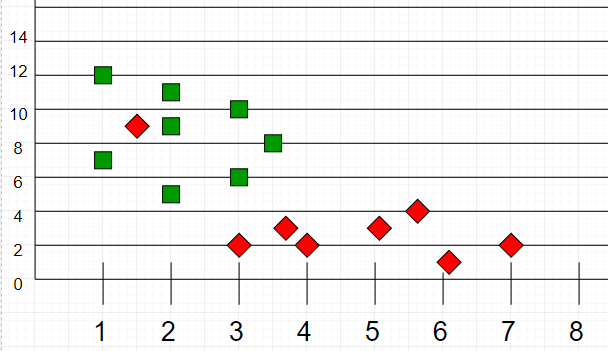


Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's regressor.

Random forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

Therefore, in random forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node. You can even make trees more random by additionally using random thresholds for each feature rather than searching for the best possible thresholds (like a normal decision tree does).

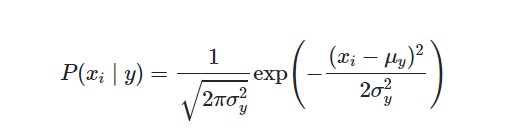
**KNeighbour Classifer:** K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.  
It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as [GMM](https://en.wikipedia.org/wiki/Mixture_model), which assume a Gaussian distribution of the given data).  
We are given some prior data (also called training data), which classifies coordinates.



**GaussianNB Classifier:** [Gaussian Naive Bayes](https://iq.opengenus.org/gaussian-naive-bayes/) is a variant of [Naive Bayes](https://iq.opengenus.org/text-classification-naive-bayes/) that follows Gaussian normal distribution and supports continuous data.**Naive Bayes** are a group of supervised machine learning classification algorithms based on the **Bayes theorem**. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high. Complex classification problems can also be implemented by using Naive Bayes Classifier. Naive Bayes Classifiers are based on the Bayes Theorem. One assumption taken is the strong independence assumptions between the features. These classifiers assume that the value of a particular feature is independent of the value of any other feature. In a supervised learning situation, Naive Bayes Classifiers are trained very efficiently. Naive Bayed classifiers need a small training data to estimate the parameters needed for classification. Naive Bayes Classifiers have simple design and implementation and they can applied to many real life situations.

**Gaussian Naive Bayes**

When working with continuous data, an assumption often taken is that the continuous values associated with each class are distributed according to a normal (or Gaussian) distribution. The likelihood of the features is assumed to be-

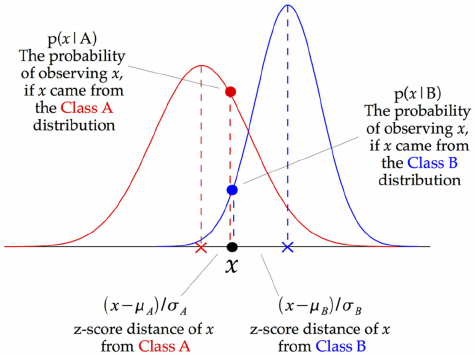


Sometimes assume variance

* is independent of Y (i.e., σi),
* or independent of Xi (i.e., σk)
* or both (i.e., σ)

Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian (normal) distribution.

An approach to create a simple model is to assume that the data is described by a Gaussian distribution with no co-variance (independent dimensions) between dimensions. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution.



The above illustration indicates how a Gaussian Naive Bayes (GNB) classifier works. At every data point, the z-score distance between that point and each class-mean is calculated, namely the distance from the class mean divided by the standard deviation of that class.

Thus, we see that the Gaussian Naive Bayes has a slightly different approach and can be used efficiently.

**Gradient Boosting Classifier:**

Gradient boosting is a technique attracting attention for its prediction speed and accuracy, especially with large and complex data. Don't just take my word for it, the chart below shows the rapid growth of Google searches for xgboost (the most popular gradient boosting R package). From data science competitions to machine learning solutions for business, gradient boosting has produced best-in-class results. In this blog post I describe what is gradient boosting and how to use gradient boosting.

**Ensembles and boosting**

Machine learning models can be fitted to data individually, or combined in an *ensemble*. An ensemble is a combination of simple individual models that together create a more powerful new model.

Machine learning boosting is a method for creating an ensemble. It starts by fitting an initial model (e.g. a tree or linear regression) to the data. Then a second model is built that focuses on accurately predicting the cases where the first model performs poorly. The combination of these two models is expected to be better than either model alone. Then you repeat this process of boosting many times.  Each successive model attempts to correct for the shortcomings of the combined boosted ensemble of all previous models.

**Gradient boosting explained**

Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error. How are the targets calculated? The target outcome for each case in the data depends on how much changing that case's prediction impacts the overall prediction error:

* If a small change in the prediction for a case causes a large drop in error, then next target outcome of the case is a high value. Predictions from the new model that are close to its targets will reduce the error.
* If a small change in the prediction for a case causes no change in error, then next target outcome of the case is zero. Changing this prediction does not decrease the error.

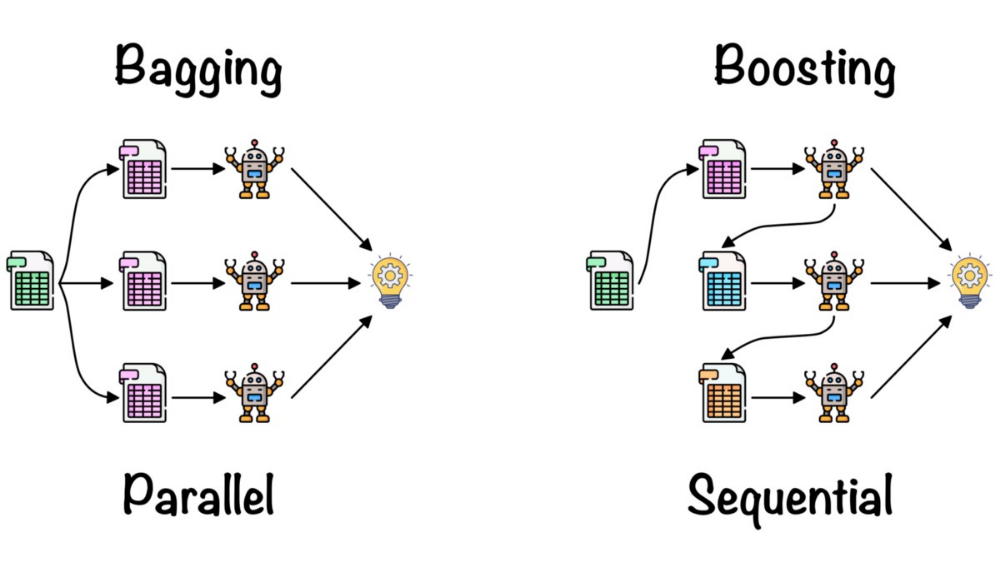
The name *gradient boosting*arises because target outcomes for each case are set based on the gradient of the error with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case.

Random forest is a ***Supervised Machine Learning Algorithm*** that is ***used widely in Classification and Regression problems***. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.One of the most important features of the Random Forest Algorithm is that it can handle the data set containing ***continuous variables*** as in the case of regression and ***categorical variables*** as in the case of classification. It performs better results for classification problems.

Before understanding the working of the random forest we must look into the ensemble technique. ***Ensemble***simplymeans combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.

*Ensemble uses two types of methods*:1. **Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example,  Random Forest.

2. **Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example,  ADA BOOST, XG BOOST



As mentioned earlier, Random forest works on the Bagging principle. Now let’s dive in and understand bagging in detail.

#### **Bagging**

Bagging, also known as ***Bootstrap Aggregation*** is the ensemble technique used by random forest.Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as ***row sampling***. This step of row sampling with replacement is called***bootstrap***. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as ***aggregation***.

**Steps involved in random forest algorithm:**

Step 1: In Random forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

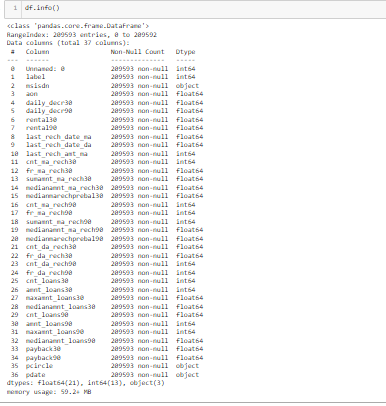
Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on ***Majority Voting or Averaging***for Classification and regression respectively.

**XGBooster:** [XGBoost](https://xgboost.ai/)is a decision-tree-based ensemble Machine Learning algorithm that uses a [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting) framework. In prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class right now.

1. A wide range of applications: Can be used to solve regression, classification, ranking, and user-defined prediction problems.
2. Portability: Runs smoothly on Windows, Linux, and OS X.
3. Languages: Supports all major programming languages including C++, Python, R, Java, Scala, and Julia.
4. Cloud Integration: Supports AWS, Azure, and Yarn clusters and works well with Flink, Spark, and other ecosystems.

**Data Analysis:** The Dataset Contains a Data of 209593 entries each having 37 variables, in which some are numerical Data and some are Categorical Data



**Exploratory Data Analysis:**

In EDA we need to Pre-process the Data and Visualization:

Steps include in Pre-Processing Data are

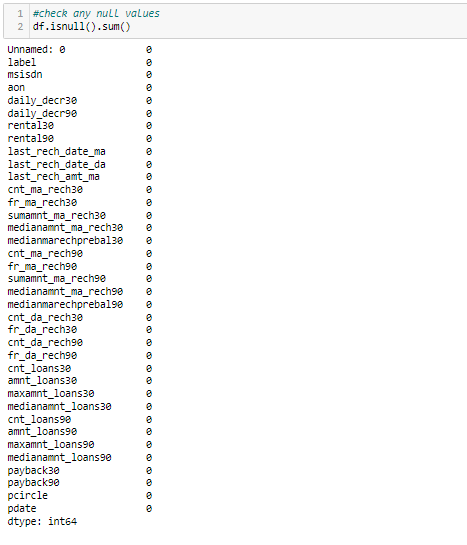
1)**Data Cleaning**: - Removing Outliers, Skewness and imputing Missing Values.

2)**Data Transformation**: - like Normalization by applying normalization we can improve the accuracy and efficiency of the models. And also reduce the errors.

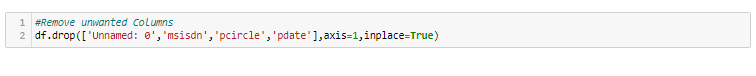
3)**Data Reduction**: By Reducing the no of features by Feature Selection Process, PCA And VIF

**1.Data Cleaning:** As a Part of EDA we need to do Data cleaning so firstly we need to check any null values in our data, From the below image shows we don’t have any null values, so no need to impute any data

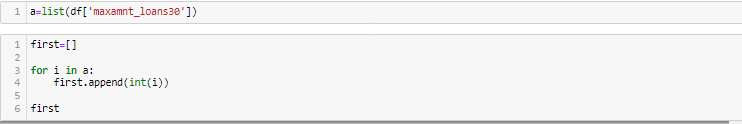
First of All Need to check the data is having any missing values

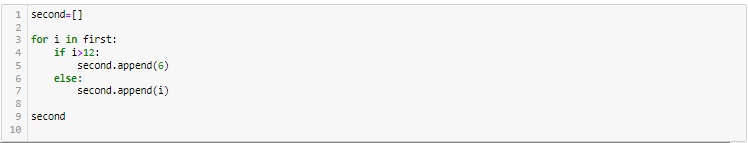


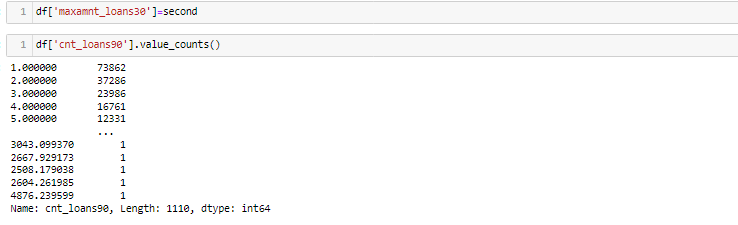
As per the above formula my data-set is clean and from the Data I removed some columns which are not useful for my model

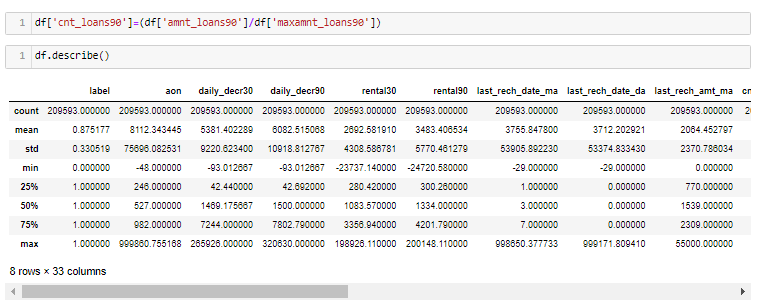


Replace some unwanted amount into a particular amount to make it understandable



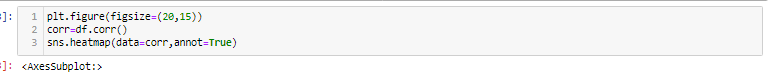


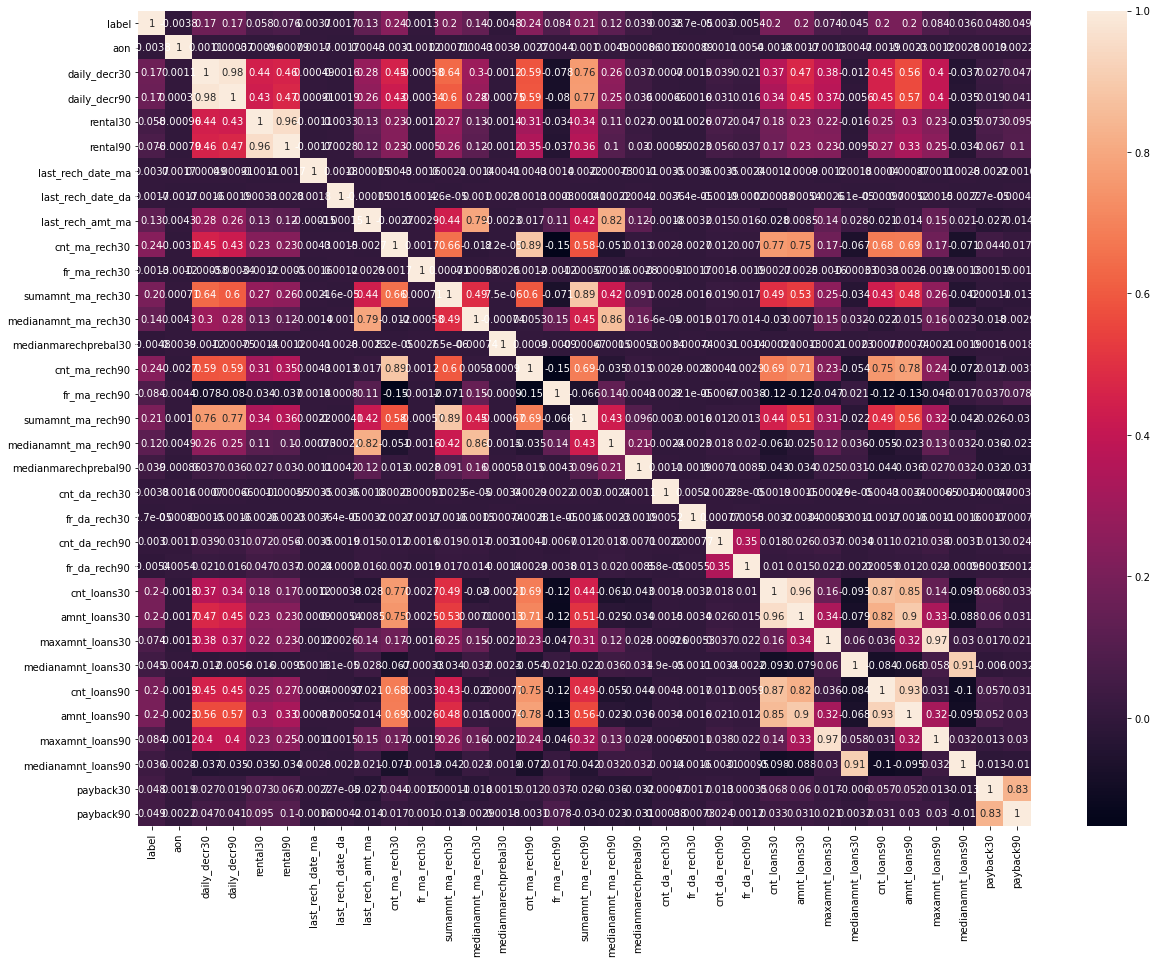




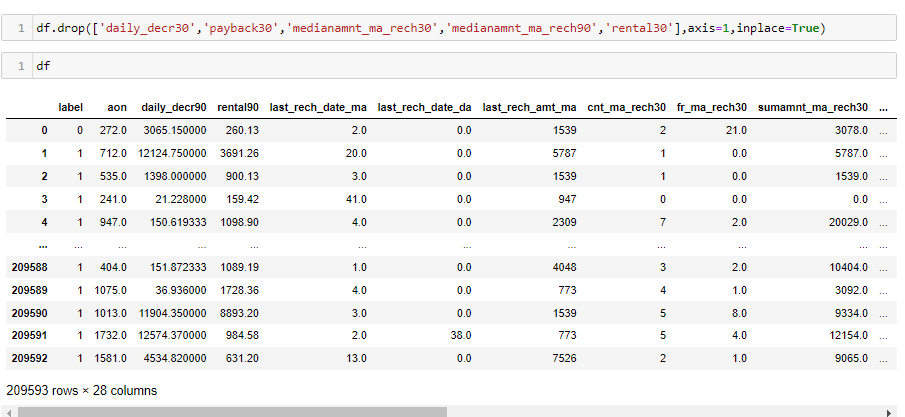
From the above describe function we know each variable percentiles, from observing we understand 75% to 100% there is a lot of difference so there are so many outliers.

By checking correlation we understood little bit more about variables using heatmap function

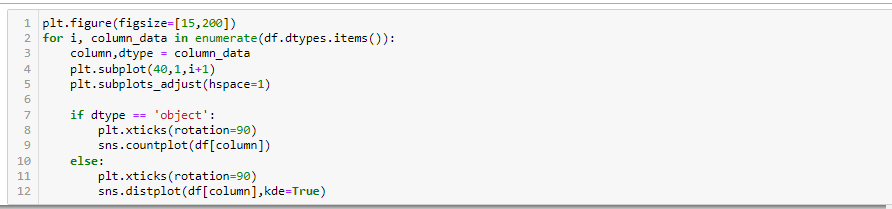




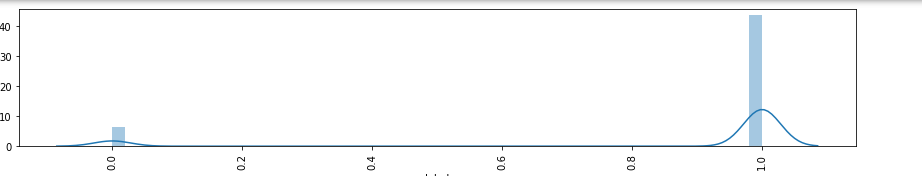
From the above heat map, we could remove the variables which are highly corelated with each other

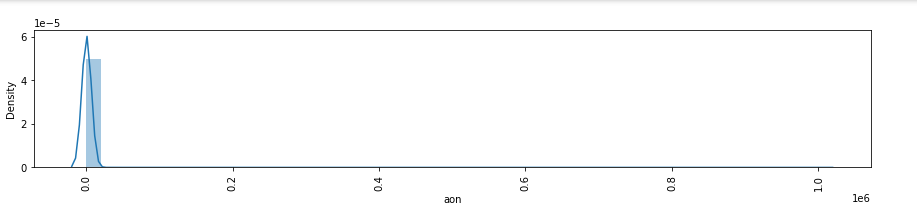


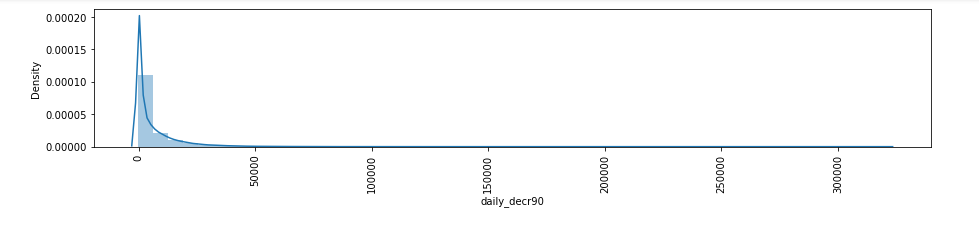
**VISUALIZATION:**

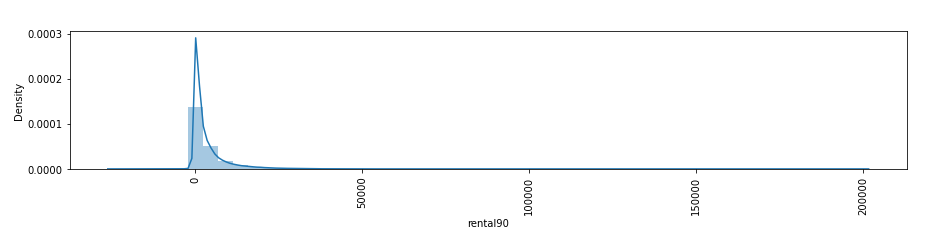
****

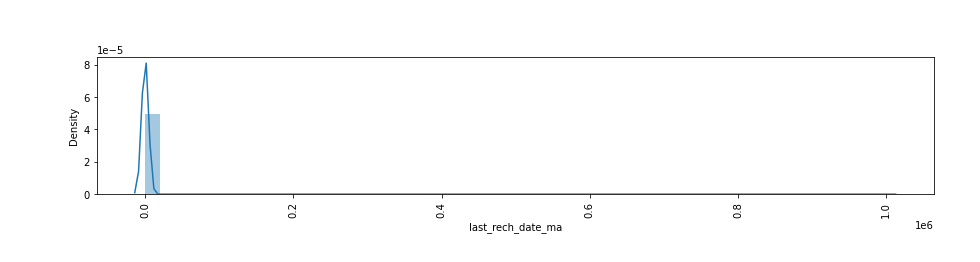
From the above Formula we can visualize all the numerical data and categorical data graphs

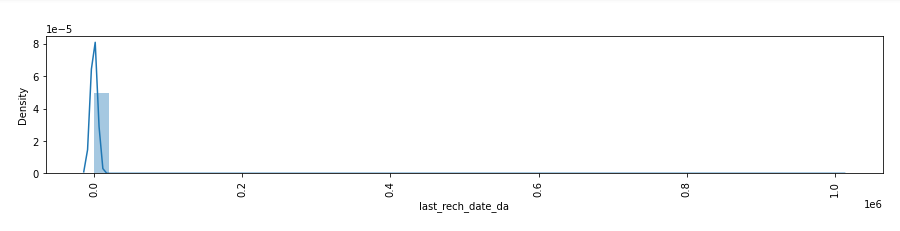


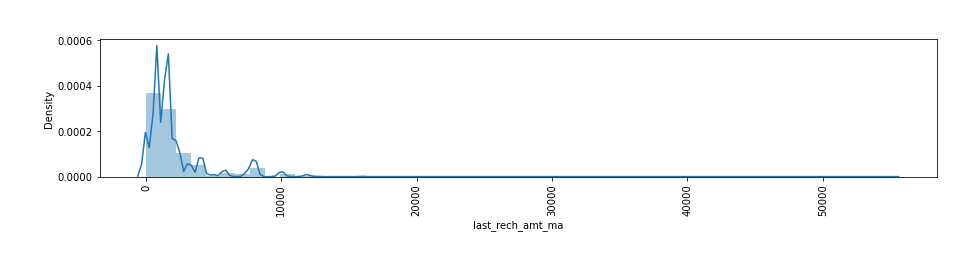


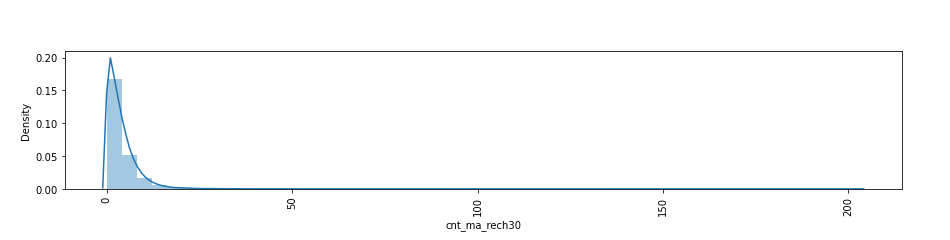


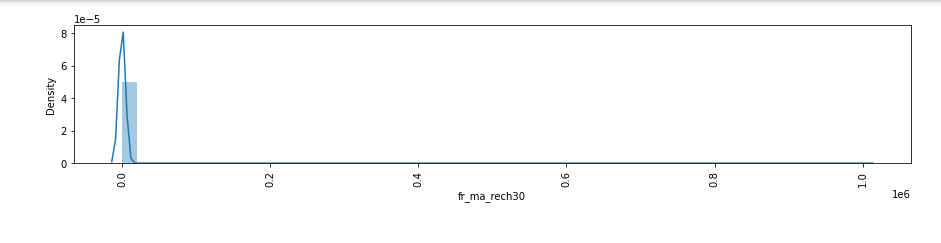
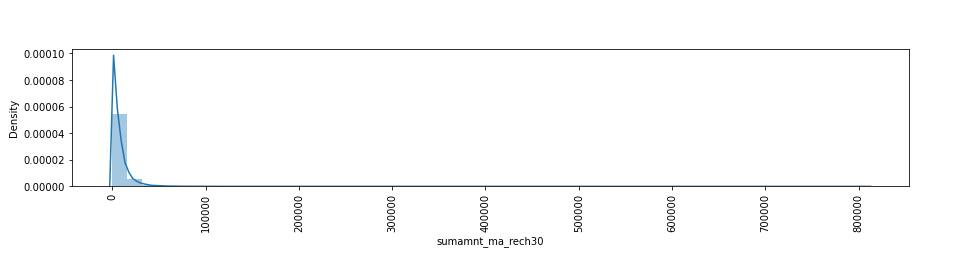






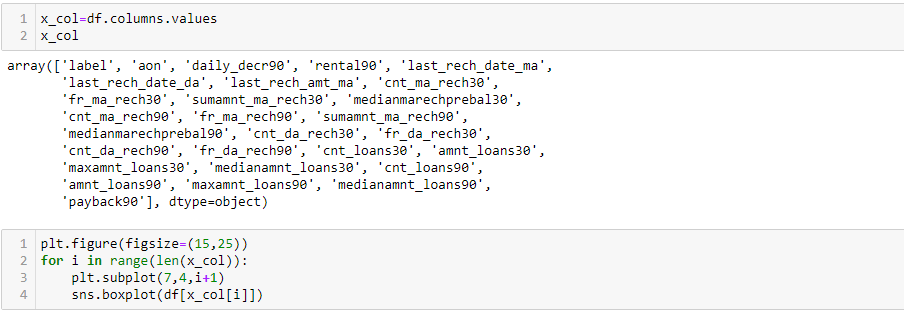


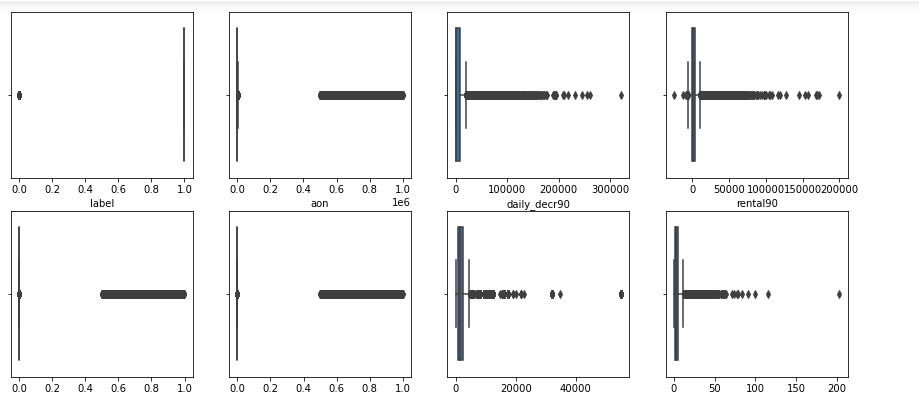


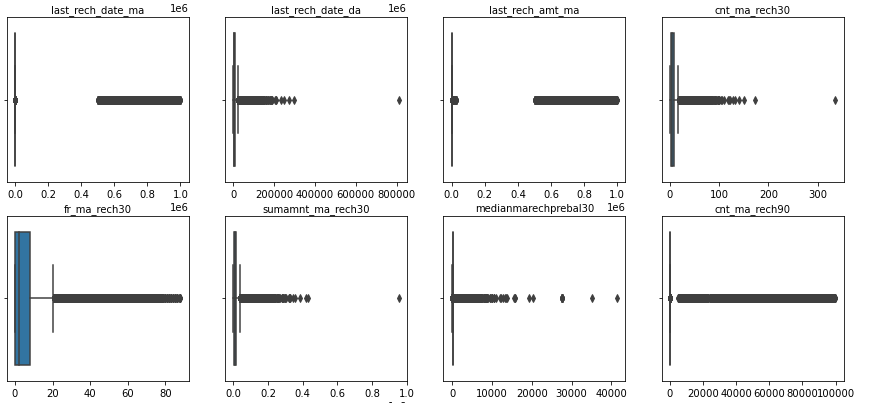
 

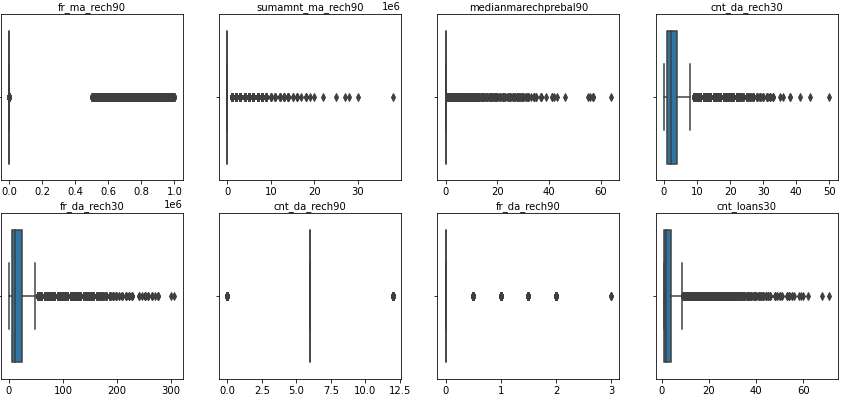
By Observing Above Graphs we understood our data is skewed like left skewed and right skewed.

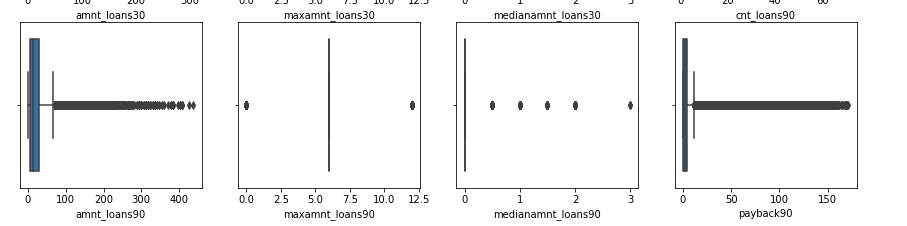
**Checking Outliers:**

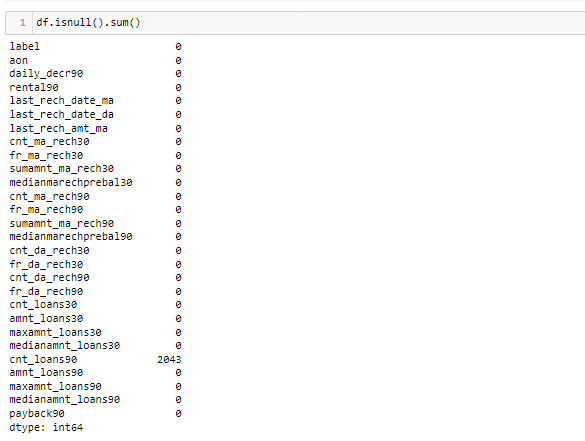
****

****

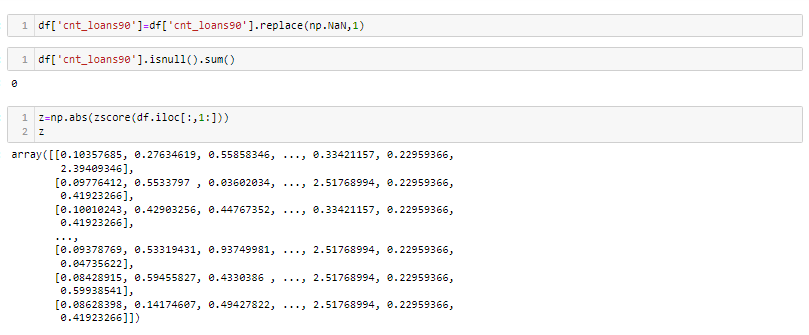
****

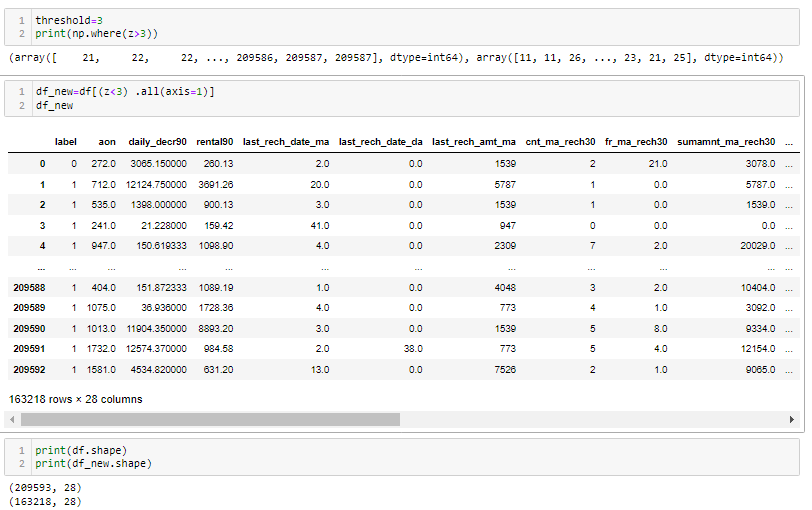
****

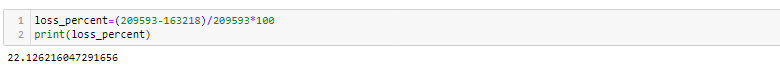
****

****

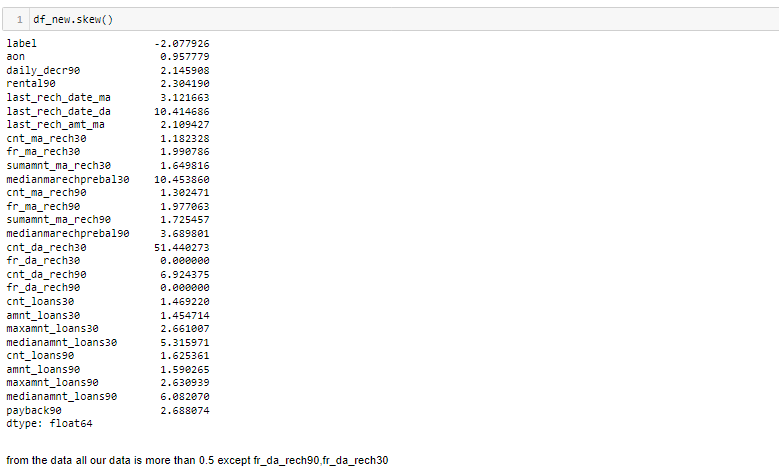
There are some null values in one of the column, so need to replace the values.

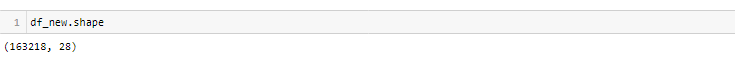




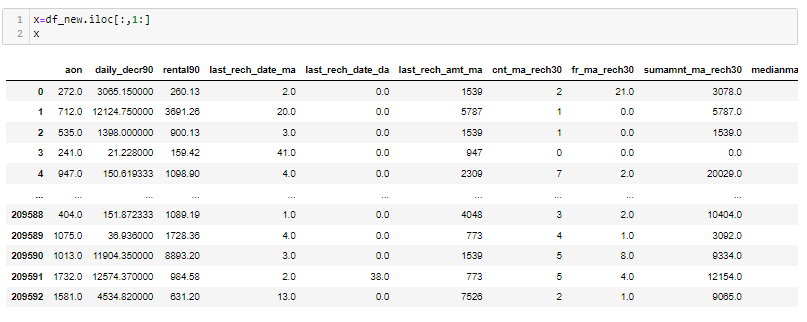


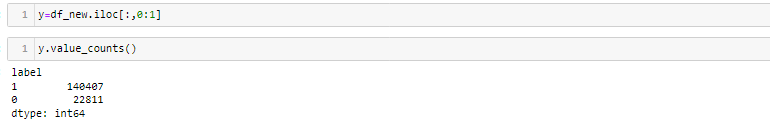
As per the new data almost 22% of the data is removed as outliers.





As per our dataset almost 28 columns are there so we need to use VIF or PCA Techniques to reduce the columns.





As our Dependent Variable is Im Balanced so we need to use smote technique,to make our model as accurate.

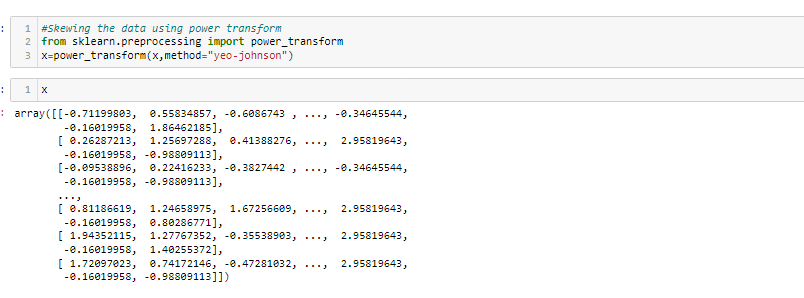
**Skewness of Data:**

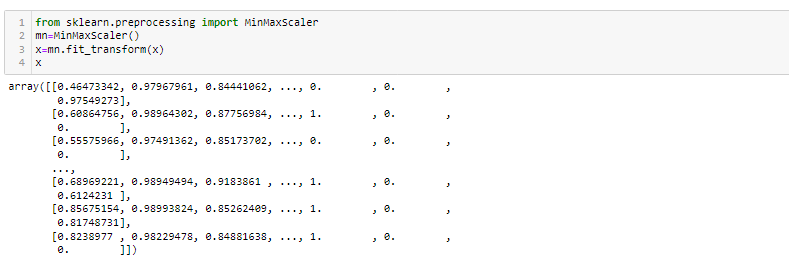
As of our numeric data is skewed we need to do normalization before go for training and testing for that need to check the skewness of data if our data is Greater than 0.5% in both positive and negative sides ,then need to do power transformation and do scaling

Scaling are of two types:

1.**Standard Scaler:** Standard scalar standardizes features of the data set by scaling to unit variance and removing the mean (optionally) using column summary statistics on the samples in the training set.

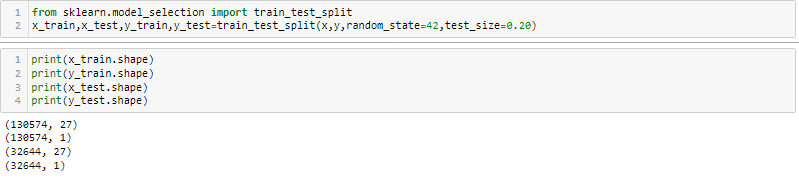
**MIN-MAX Scaler:** MinMaxScaler. For each value in a feature, MinMaxScaler subtracts the minimum value in the feature and then divides by the range. The range is the difference between the original maximum and original minimum. MinMaxScaler preserves the shape of the original distribution.





**Splitting Data Into train\_test\_split: -**

This function is in sklearn. Model selection splitting the data array into two arrays. Train and Test with this function we don’t need to splitting train and test manually.by default it make random partition and we can also set the random state.it gives four o/p like x\_train, x\_test, y\_train, y\_test.

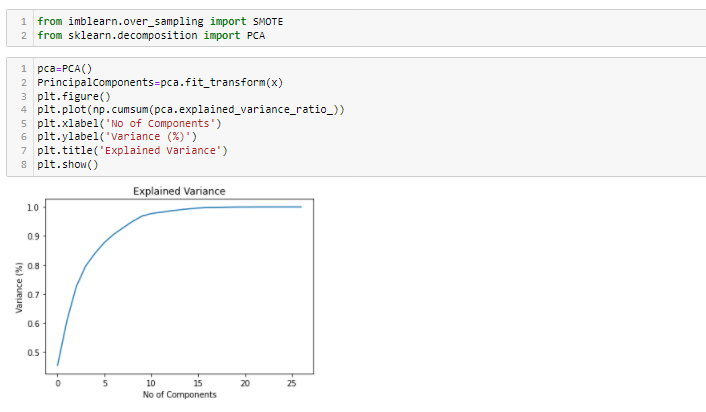


**Principal Component Analysis(PCA):**

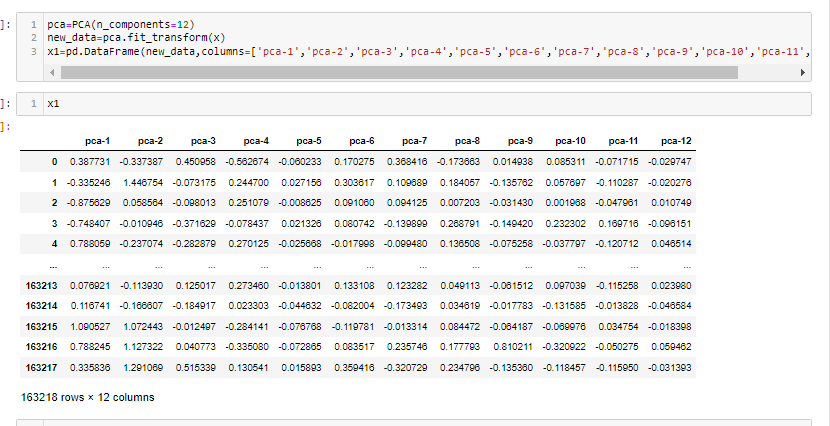
 Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance.

Large datasets are increasingly widespread in many disciplines. In order to interpret such datasets, methods are required to drastically reduce their dimensionality in an interpretable way, such that most of the information in the data is preserved. Many techniques have been developed for this purpose, but principal component analysis (PCA) is one of the oldest and most widely used. Its idea is simple—reduce the dimensionality of a dataset, while preserving as much ‘variability’ (i.e. statistical information) as possible. Although it is used, and has sometimes been reinvented, in many different disciplines it is, at heart, a statistical technique and hence much of its development has been by statisticians. This means that ‘preserving as much variability as possible’ translates into finding new variables that are linear functions of those in the original dataset, that successively maximize variance and that are uncorrelated with each other. Finding such new variables, the principal components (PCs), reduces to solving an eigenvalue/eigenvector problem. The earliest literature on PCA dates from Pearson and Hotelling but it was not until electronic computers became widely available decades later that it was computationally feasible to use it on datasets that were not trivially small. Since then its use has burgeoned and a large number of variants have been developed in many different disciplines. Substantial books have been written on the subject and there are even whole books on variants of PCA for special types of data the formal definition of PCA will be given, in a standard context, together with a derivation showing that it can be obtained as the solution to an eigenproblem or, alternatively, from the singular value decomposition (SVD) of the (centred) data matrix. PCA can be based on either the covariance matrix or the correlation matrix. The choice between these analyses will be discussed. In either case, the new variables (the PCs) depend on the dataset, rather than being pre-defined basis functions, and so are adaptive in the broad sense. The main uses of PCA are descriptive, rather than inferential; an example will illustrate this.

Although for inferential purposes a multivariate normal (Gaussian) distribution of the dataset is usually assumed, PCA as a descriptive tool needs no distributional assumptions and, as such, is very much an adaptive exploratory method which can be used on numerical data of various types. Indeed, many adaptations of the basic methodology for different data types and structures have been developed, two of which will be described in Some techniques give simplified versions of PCs, in order to aid interpretation. Two of these are briefly described in, which also includes an example of PCA, together with a simplified version, in atmospheric science, illustrating the adaptive potential of PCA in a specific context. Section discusses one of the extensions of PCA that has been most active in recent years, namely robust PCA (RPCA). The explosion in very large datasets in areas such as image analysis or the analysis of Web data has brought about important methodological advances in data analysis which often find their roots in PCA. Each of gives references to recent work. Some concluding remarks, emphasizing the breadth of application of PCA and its numerous adaptations, are made in.

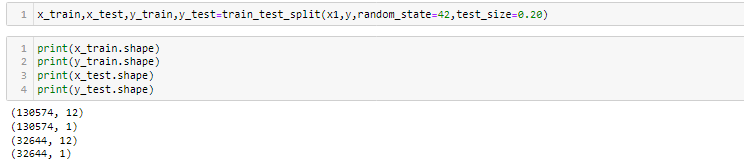
****

**From the Graph, its clearly understands that instead of using 27 variables, we can use 12 to make the model is accurate.**

****

As we do splitting the data for training and testing the data then now we need to modeling

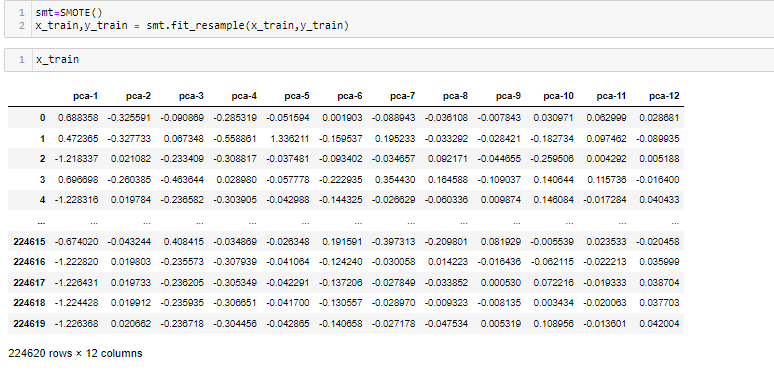
Try Different Models….

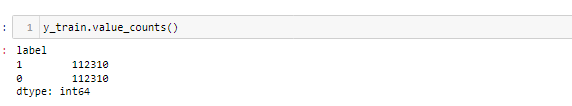


After Doing splitting we have to balanced our data.it can be by SMOTE or oversampling methods. Like Up Sampling, down sampling.

**Up sampling:** -This method used to modify the unequal data into the balanced data by increases the minor class or rare class. Advantage of this method is to no loss of information but from that model can be in overfitting.

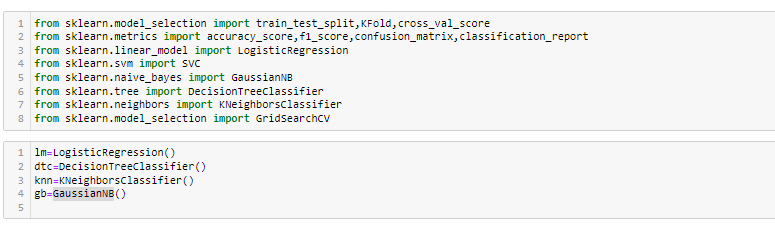
**Down Sampling:** like the Up sampling its also balanced data but by reducing the size of the class which is high

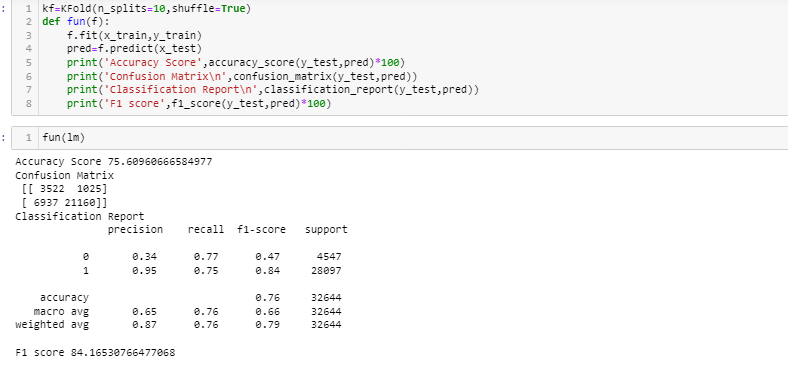




**But if we balanced our data before train test split means we balanced from our whole data set or form x. it means at that time our test data is leak. We have to isolate our test data. Here you expose it.so our f1 or recall or precision will be good. so, our model will already know which is positive or negative. And I can also say because of that there is bias or model Overfitting.to prevent this We balanced our data**

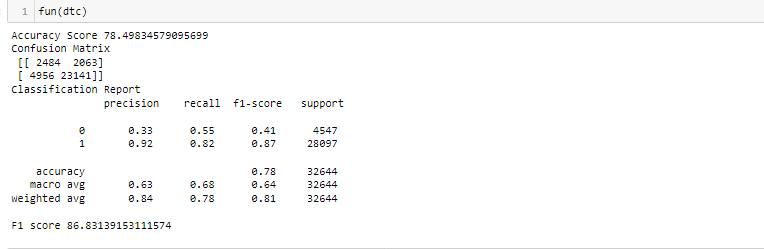
1)**Logistic Regression:** -logistic regression is the supervised machine learning problem which is used for the classification problem and used to predict the probability of the classification.it is widely used for the binary classification problem. It is one od the simplest methos of ML

****

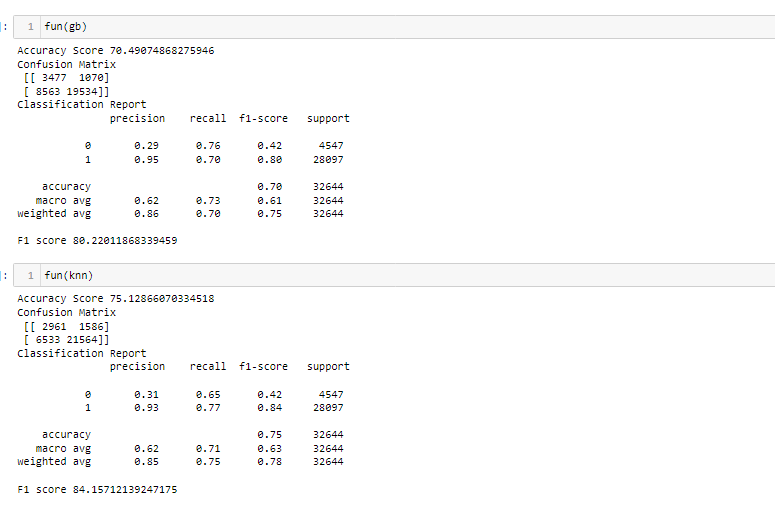
****

Here my Logistic Regression is giving an accuracy score of 75.6% and F1-Score :84%

2) **Decision Tree Classifier:** DTC can be used by both classification and regression both. But mostly it’s used for the classification problem. Its structure is tree based. Where internal nodes represents the features of dataset and branches represents the decision rules and each leaf nodes represents the outcomes.



Here my Decision Tree Classifier is giving an accuracy score of 78.49% and F1-Score :86.83%.



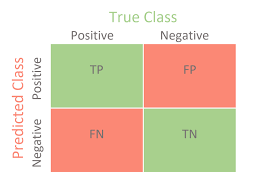
Here my GaussianNB Classifier is giving an accuracy score of 70.49% and F1-Score :80.22%.

Here my KNN Classifier is giving an accuracy score of 75.12% and F1-Score :84.15%.

**Now lets see about each one in our output clearly:**

**Confusion Matrix:** It is the table that is used to describe the performance of classification model on set of tests data.by using different parameters.

We get the best score in Random Forest Classifier as Accuracy score is 82.85%,And model predicts 3892 as True Positive,462 False Positive , 455 False Negative and 538 True Negative

****

**Now lets understand the Recall Precision and f1-score**

**Accuracy:** it can be defined as the ratio of total number of correct classifications divided by total number of classifications.

Accuracy=(TP+TN)/(TP+FP+TN+FN)

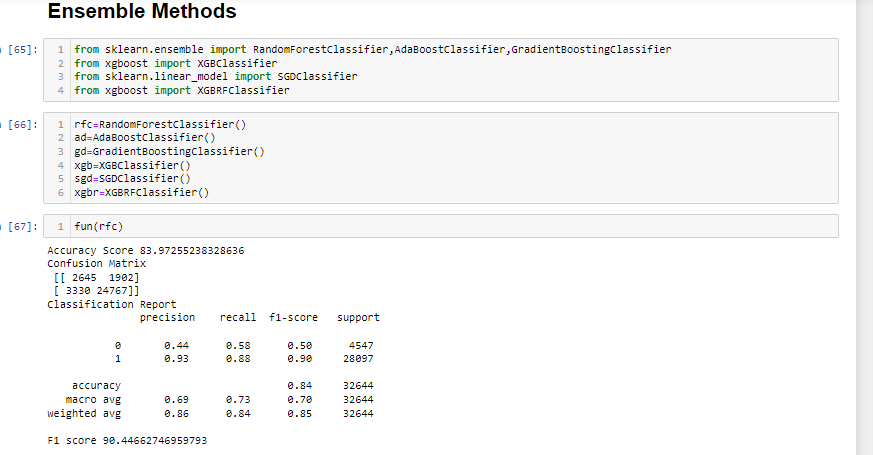
**Precision**: It is measure of all the positive predictions how many of them actually positive. Precision=TP/(TP+FP)

**F1-Score:** It give the combine result of Recall and Precision

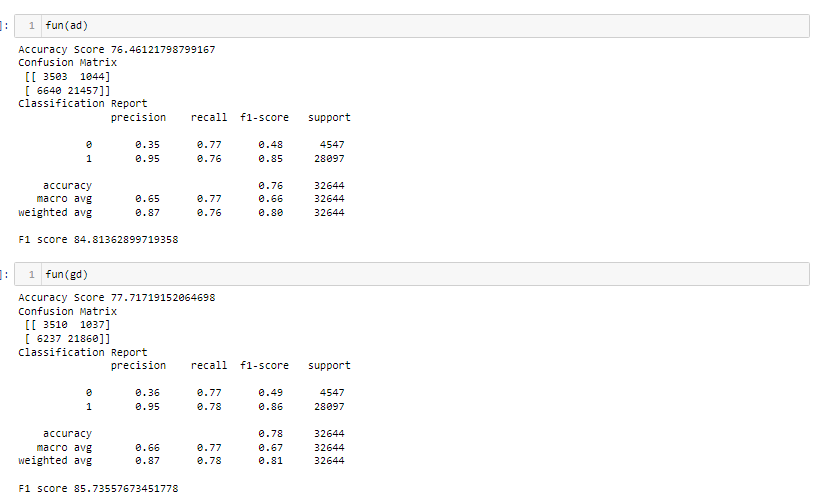
F1-score=2\*(Precision\*Recall)/ (Precision + Recall)

**Random Forest Classifier:** Random forest is a [supervised learning algorithm](https://builtin.com/data-science/supervised-learning-python). The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the [bagging method](https://builtin.com/data-science/tour-top-10-algorithms-machine-learning-newbies) is that a combination of learning models increases the overall result.

Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, there's no need to combine a decision tree with a bagging classifier because you can easily use the classifier-class of random forest. With random forest, you can also deal with regression tasks by using the algorithm's .

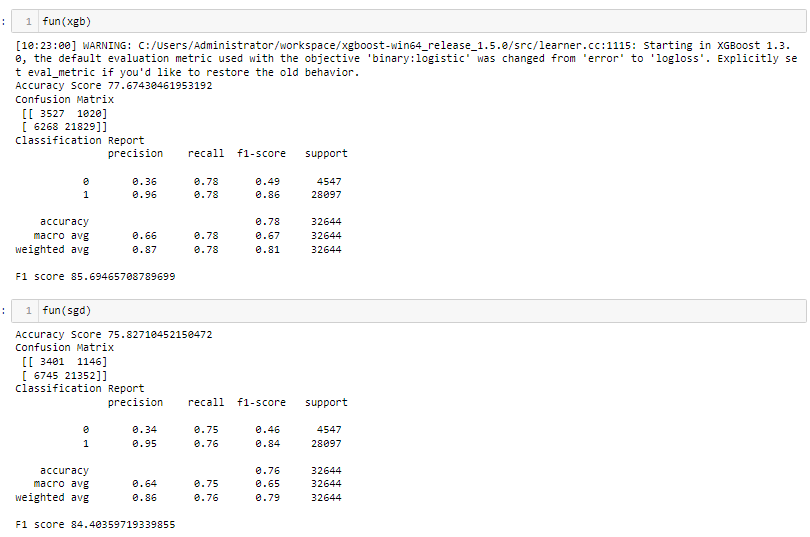


Here my Forest Classifier is giving an accuracy score of 83.97% and F1-Score :90.44%.



Here my ADA Booster Classifier is giving an accuracy score of 76.46% and F1-Score :84.81%.

Here my Gradient Classifier is giving an accuracy score of 77.71% and F1-Score :85.73%.



Here my XGB Booster Classifier is giving an accuracy score of 77.67% and F1-Score :85.69%.

Here my Strochastic Gradient Classifier is giving an accuracy score of 75.82% and F1-Score :84.40%.

**Concluding Remarks:** - From this model we can predict the defaulters in micro finance and predicts how it affects with each variable.

We used different classifiers like Logistic Regression, Decision tree Classifiers and Ada boosting classifiers, KNN Classifier, Gradient Classifier, Random Forest Classifier. And also used the data Balanced process and also hyper parameter tunning for improving score.

We get good score in Random Classifier we got accuracy of 83.97% on training data,F1 score is 90% in random forest classifier. the model performance is excellent.

We further proceed to test the object that we saved using pickle, and create a data frame of predicted values –



