**Marketing Campaign for Banking Products**

**A PROJECT REPORT**

**Submitted by**

**Sumit Sanjay Sagave**

**In Machine Learning**

**Of**

**Final Year**

**In**

**Internship Studio**

**From**

A picture containing kitchenware, building

Description generated with high confidence

**SCHOOL OF ELECTRICAL ENGINEERING**

**DR VISHWANATH KARAD MIT WORLD PEACE UNIVERSITY**

**SEPTEMBER 2020**

This project report is of different classification models like Logistic Regression, kNN, Naïve Bayes, SVM (Support Vector Machines), Decision Tree Classifier, with side by side comparison of these in python. It takes through a bank dataset with all data cleaning like removing outliers, dropping non-value-adding columns, etc. and the detailed analysis (both included: Univariate and Bivariate) to get the sense of the dataset effortlessly. It just expects to know the basics of the python.

**Problem Statement:**

The file given below contains data on 5000 customers. The data include customer information (age, income, etc.), the customer's relationship with the bank and their customer response to the last personal loan campaign (Personal Loan). Among these 5000 customers, only 480 (= 9.6%) accepted the personal loan that was offered to them in the earlier campaign.

**Context:**

This case is about a bank whose management wants to explore ways of converting its liability customers to personal loan customers (while retaining them as depositors). A campaign that the bank ran last year for liability customers showed a healthy conversion rate of over 9% success. This has encouraged the retail marketing department to devise campaigns with better target marketing to increase the success ratio with a minimal budget.

**Objective:**

The classification goal is to predict the likelihood of a liability customer buying personal loans.

**Hypothesis Generation:**

This is a very important stage in any data science/machine learning pipeline. It involves understanding the problem in detail by brainstorming as many factors as possible which can impact the outcome. It is done by understanding the problem statement thoroughly and before looking at the data.

Below are some of the factors which I think can affect the likelihood of a liability customer buying personal loans (dependent variable for this loan prediction problem):

**Salary:** Salary can be one of the major dependent variables as customers with high salaries are less feasible to buy personal loans while customers with medium or low salaries are more feasible for buying personal loans.

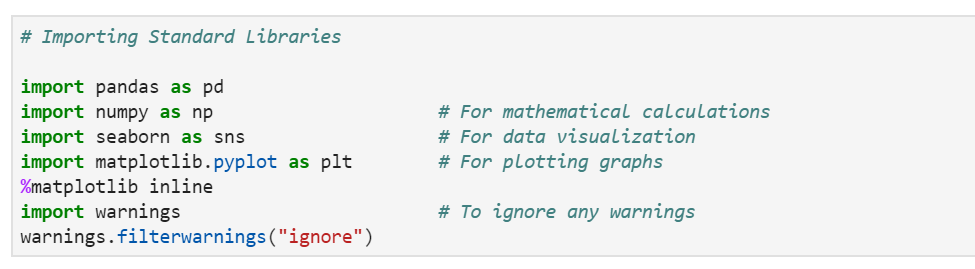
**The number of family members:** More the number of earning family members, less probability of buying personal loans.

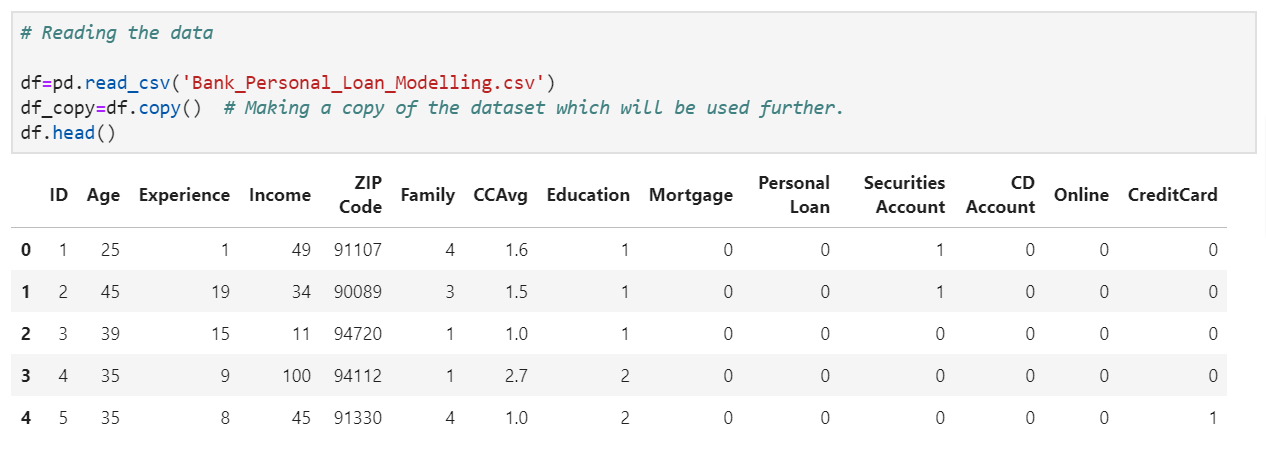
**Age:** Customers with probably the age of 30–50 will buy personal loans.

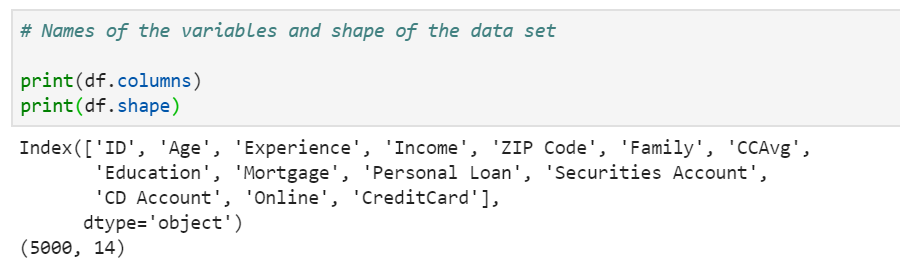
**Education of the customer:** The customer is a graduate or under-graduate can affect the buying probability, people who are graduated or Advanced Professionals are more viable to buy personal loans from a bank rather than people who are under-graduated.

There can be more variables possible, just brainstorm some more if possible and start with the python code:

We will start with importing basic libraries, reading the data into a data frame and figure out the basic shape of our data set:







We have 13 independent variables and 1 dependent variable i.e. ‘Personal Loan’ in the data set. Also, we got 5000 rows which can be split into test & train datasets.

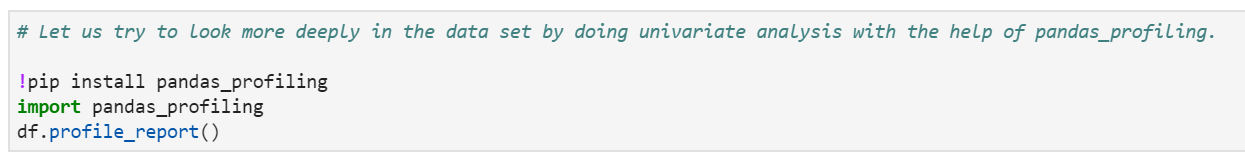
**Description of the variables:**

1. ID: Customer ID
2. Age: Customer’s age in completed years
3. Experience: #years of professional experience
4. Income: Annual income of the customer ($000)
5. ZIP Code: Home Address ZIP code.
6. Family: the Family size of the customer
7. CCAvg: Avg. spending on credit cards per month ($000)
8. Education: Education Level. 1: Undergrad; 2: Graduate;3: Advanced/Professional
9. Mortgage: Value of house mortgage if any. ($000)
10. **Personal Loan**: Did this customer accept the personal loan offered in the last campaign?
11. Securities Account: Does the customer have a securities account with the bank?
12. CD Account: Does the customer have a certificate of deposit (CD) account with the bank?
13. Online: Do customers use internet banking facilities?
14. Credit card: Does the customer use a credit card issued by UniversalBank?

**Univariate Analysis:**

Let us try to look more deeply into the data set by doing univariate analysis with the help of pandas\_profiling.

Note: pandas\_profiling: it generates profile reports from a pandas DataFrame. The pandas df.describe() function is great but a little basic for serious exploratory data analysis. [pandas\_profiling](https://pandas-profiling.github.io/pandas-profiling/docs/#pandas_profiling) extends the pandas DataFrame with df.profile\_report() for quick data analysis.



​**Points observed by profile report & univariate analysis:**

* The data set got 0 missing cells.
* It got **7 numeric variables**: ‘Age’, ‘CC\_Avg’, ‘ID’, ‘Income’, ‘Mortgage’, ‘Zip\_Code’, ‘Experience’
* It got **2 categorical variables**: ‘Education’, ‘Family’
* It got **5 Boolean variables**: ‘CD\_Account’, ‘Credit\_Card’, ‘Online’, ‘Personal\_Loan’, ‘Securities Account’
* Personal Loan is highly correlated with Income, average spending on Credit cards, mortgage & if the customer has a certificate of deposit (CD) account with the bank.
* Also, Experience is highly correlated with Age (ρ = 0.994214857)

**Categorical:**

* 42% of the candidates are graduated, while 30% are professional and 28% are Undergraduate.
* Around 29% of the customer’s family size is 1.

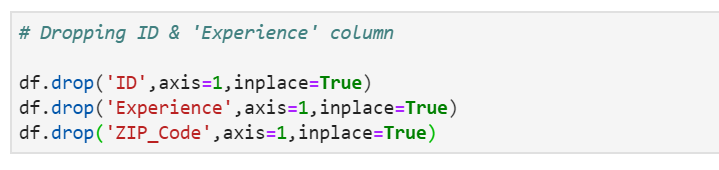
**Boolean:**

* 94% of the customer doesn’t have a certificate of deposit (CD) account with the bank.
* Around 71% of the customer doesn’t use a credit card issued by UniversalBank.
* Around 60% of customers use internet banking facilities.
* Around 90% of the customer doesn’t accept the personal loan offered in the last campaign.
* Around 90% of the customer doesn’t have securities account with the bank.

**Numeric:**

* The mean age of the customers is 45 with a standard deviation of 11.5. Also, we had estimated the average age in hypothesis testing between 30–50. The curve is slightly negatively skewed (Skewness = -0.02934068151) hence the curve is fairly symmetrical
* The mean of Avg. spending on credit cards per month is 1.93 with a standard deviation of 1.75. The curve is highly positive skewed (Skewness = 1.598443337)
* The mean annual income of the customer is 73.77 with a standard deviation of 46. The curve is moderately positive skewed (Skewness = 0.8413386073)
* The mean value of house mortgage is 56.5 with a standard deviation of 101.71. The curve is highly positive skewed (Skewness = 2.104002319) and there are a lot of outlier’s present (Kurtosis = 4.756796669)

Also, no need for ID, ZIP\_Code & Experience columns for further analysis since ID and ZIP\_Code are just numbers of series & Experience is highly correlated with Age.



**Bivariate Analysis:**

Let’s see some of the hypotheses that we generated earlier:

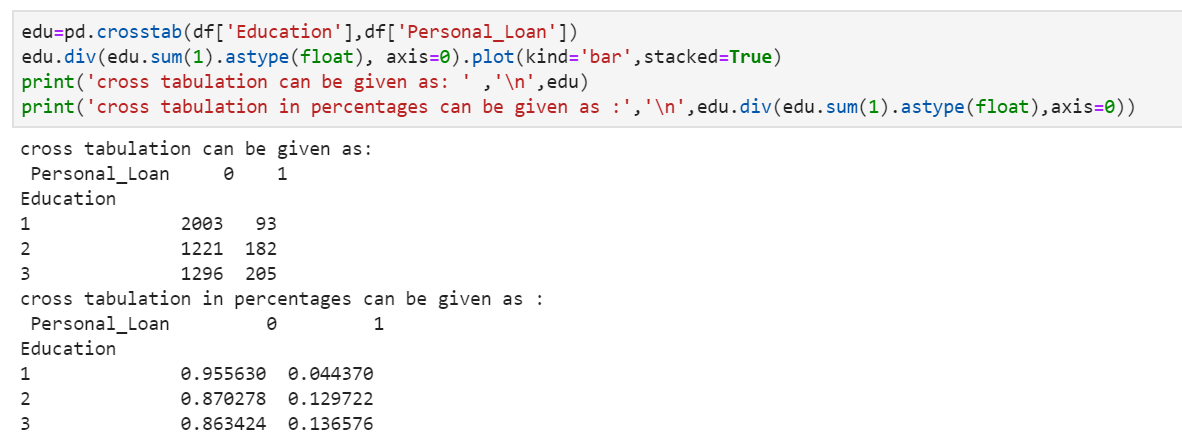
* high salaries are less feasible to buy personal loans while customers with medium or low salaries are more feasible for buying personal loans.
* More the number of earning family members, less probability of buying personal loans.
* Customers with probably the age of 30–50 will buy personal loans.
* The customer is a graduate or under-graduate can affect the buying probability, people who are graduated or Advanced Professionals are more viable to buy personal loans from a bank rather than people who are under-graduated.

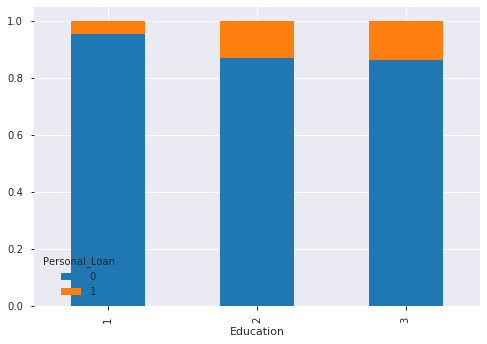
Let’s try to test the above-mentioned hypotheses using bivariate analysis

After looking at every variable individually in univariate analysis, we will now explore them again concerning the target variable.

**Categorical Independent Variable vs Target Variable:**

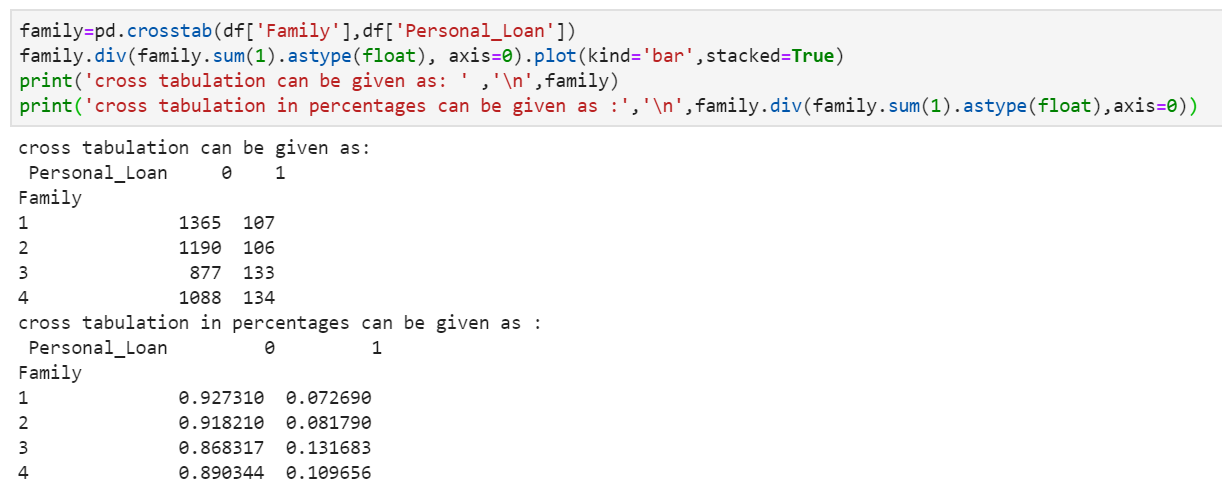
First, we will find the relation between the target variable and categorical independent variables. Let us look at the stacked bar plot now between the personal loan buyers and their education level which will give us the proportion of purchased and not purchased loans. For the plot, we will be using the crosstab function of pandas which computes a simple cross-tabulation of two (or more) factors. We will also use pd.crosstab.div() function to convert it into percentages as shown below.

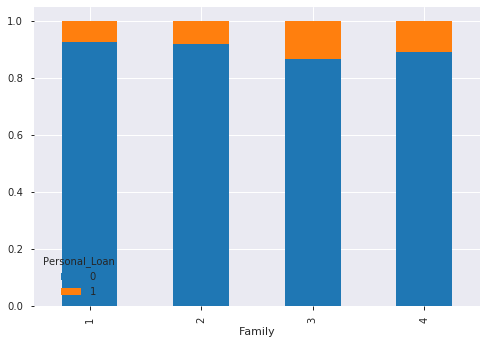




From the above plots, we can infer that customers who are more educated have a higher probability of buying personal loans. Hence our hypothesis is true.

Let us look now at the stacked bar plot between the personal loan buyers and their family size with the same code as above just replacing the categorical variables:

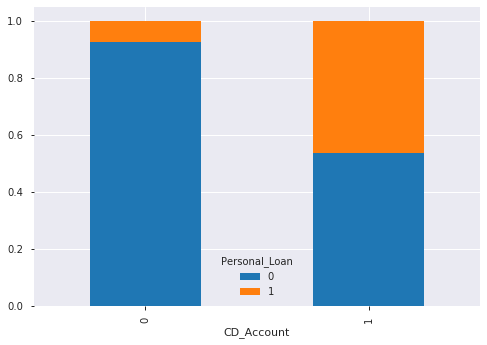




The number of family members not significantly affect probability. Hence it contradicts our hypothesis that the number of family members will affect the probability.

**Boolean Independent Variable vs Target Variable:**

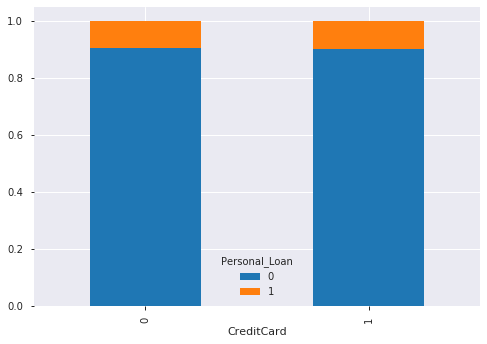
Let us now look at the Boolean variables (CD\_Account, Credit\_Card, Online, Securities Account) vs Target variable (Personal\_Loan). We will be using the same crosstab function and stacked bar plot to compare. First, let us compare the personal loan buyers who have have a certificate of deposit (CD) account with the bank or not:



The customer who has a certificate of deposit (CD) account with the bank seems to buy personal loans from the bank.

Let us now compare between the personal loan buyers who use or doesn’t use a credit card issued by UniversalBank:

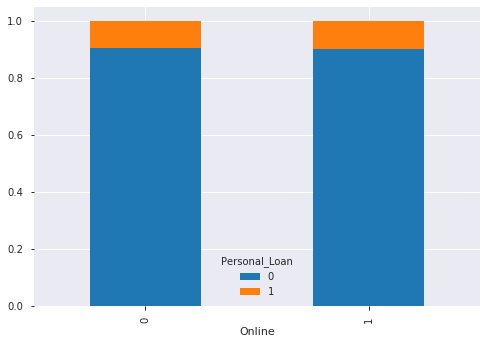




The customer who uses or doesn’t use a credit card issued by UniversalBank doesn’t seem to affect the probability of buying a personal loan.

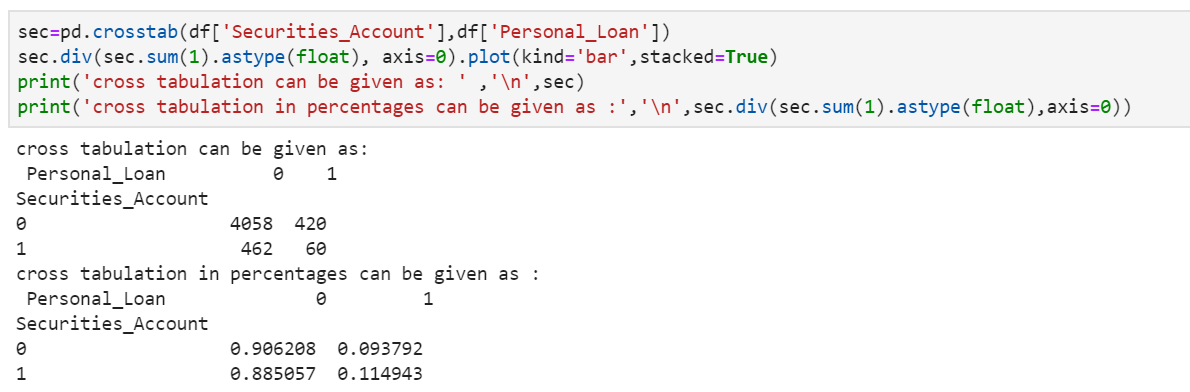
Let us now compare the personal loan buyer’s customer who uses or doesn’t use internet banking facilities:

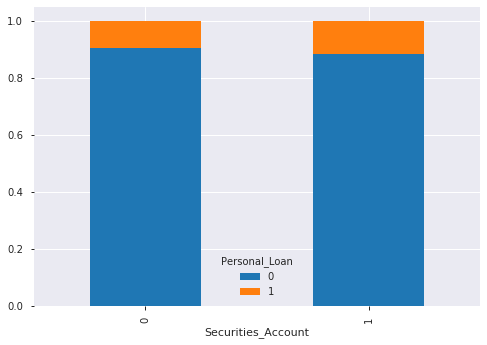




The customer who uses or doesn’t use internet banking facilities seems to not affect the probability of buying personal loans.

Let us now compare between the personal loan buyer’s customer who has or doesn’t have a securities account with the bank:

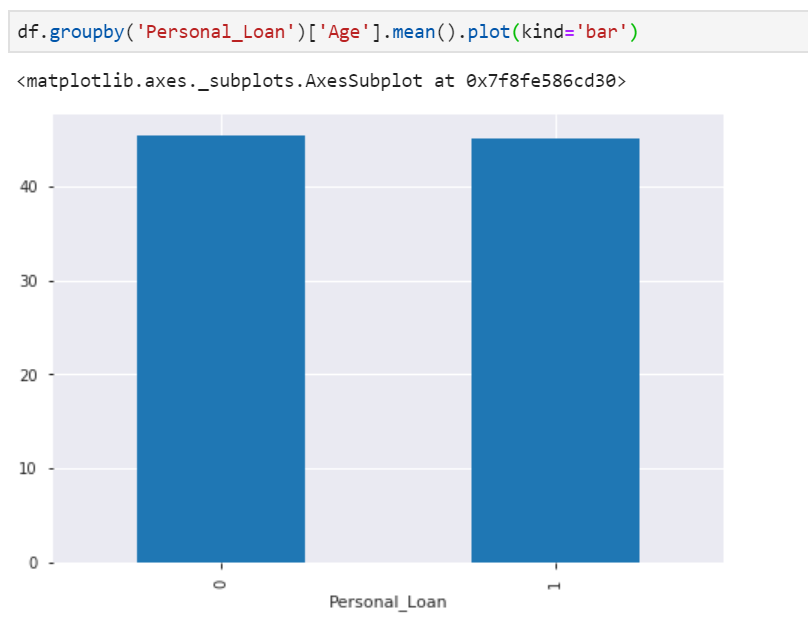




The customers who have or don’t have securities account with the bank do not affect the probability of buying a personal loan.

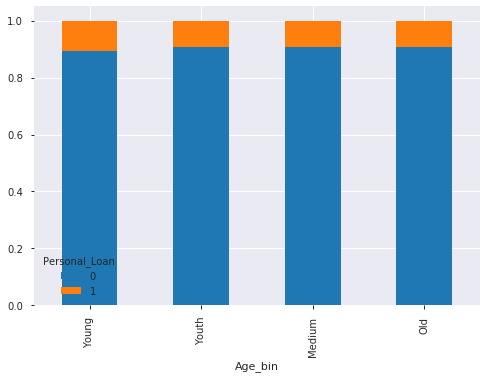
**Numerical Independent Variable vs Target Variable:**

Let us now look at the Numerical variables (Age, CC\_Avg, Income, Mortgage, Experience) vs Target variable (Personal\_Loan). We will try to find the mean of the numeric independent variable for which the customers buy the personal loan vs the mean of the numeric variables who don’t. Here we had used a DataFrame.groupby() function which makes involves some combination of splitting the object, applying a function, and combining the results. This can be used to group large amounts of data and compute operations on these groups.



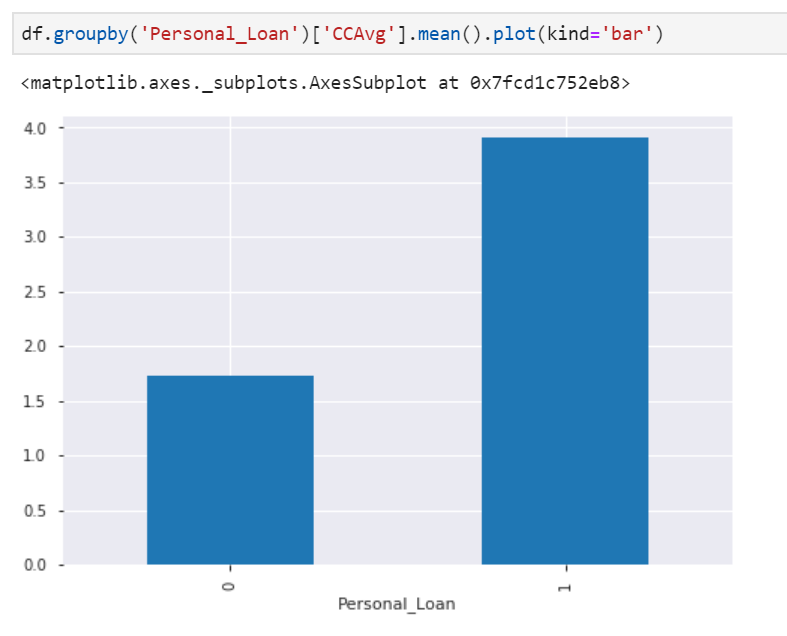
Here the y-axis represents the mean applicant age. We don’t see any change in the mean age. So, let’s make bins for the applicant age variable based on the values in it and analyze the corresponding loan status for each bin. To make the bins, we had used pandas.cut() which is used to segment and sort data values into bins. This function is also useful for going from a continuous variable to a categorical variable. And then we use the same pandas.crosstab() & DataFrame.div() function to plot the stacked bar graph which can be shown as:





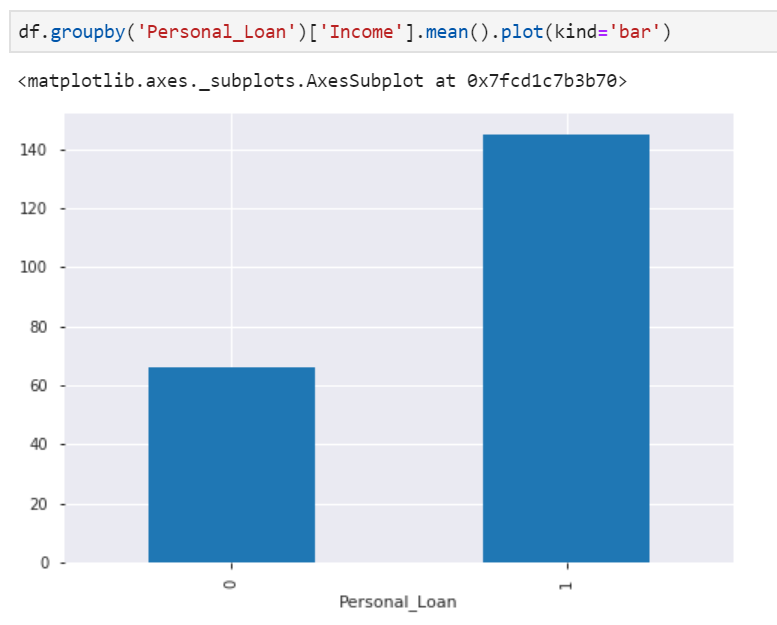
It can be inferred that the Applicant age does not affect the chances of buying the personal loan which contradicts our hypothesis in which we assumed that the applicant age is a major factor for buying the loan.

Let us now look at the personal loan buyer's average spending on credit cards per month. It can be done with a simple group by function as used above:

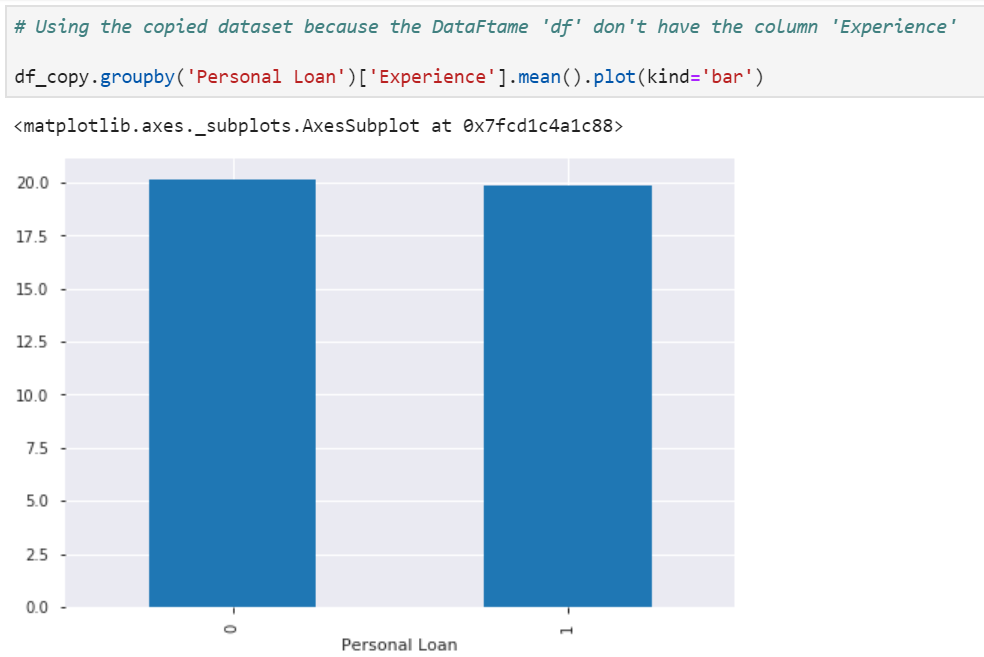


Here the y-axis represents the mean applicant spending on credit cards per month. It can be seen that applicants who spend more on credit cards are more viable to buy personal loans.

Let us look now how the income of the customer affect the possibility of a liability customer. We will be again using groupby() and mean() function:



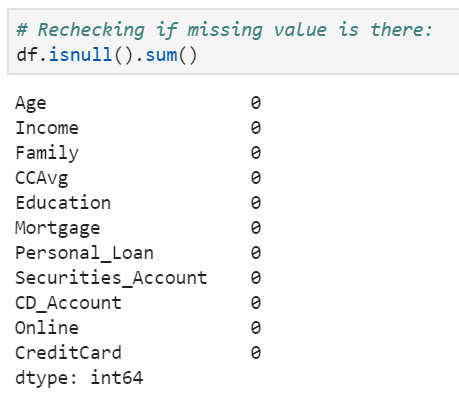
Here the y-axis represents the mean annual income of the customer. It can be seen that the customers with high incomes are more feasible to buy the personal loan which contradicts our hypothesis that high salaries are less feasible to buy personal loans while customers with medium or low salaries are more feasible for buying personal loans.



You can expect this without even plotting this if you remember that the ‘Experience’ is highly correlated with ‘Age’(ρ = 0.994214857).

**Missing Value and Outlier Treatment:**

After exploring all the variables in our data, we can now impute the missing values and treat the outliers because missing data and outliers can harm the model performance. We are clear with our data set that it does not have any missing values, but it got many outliers as we had seen in univariate analysis with the help of Kurtosis.



We are pretty much sure now that there are no missing values but let’s start with Outlier’s Treatment.

**Outlier Treatment:**

As we saw earlier in univariate analysis, Mortgage contains outliers, so we must treat them as the presence of outliers affects the distribution of the data. Let’s examine what can happen to a data set with outliers. For the sample data set:

1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4

We find the following: mean, median, mode, and standard deviation

Mean = 2.58

Median = 2.5

Mode = 2

Standard Deviation = 1.08

If we add an outlier to the data set:

1, 1, 2, 2, 2, 2, 3, 3, 3, 4, 4, 400

The new values of our statistics are:

Mean = 35.38

Median = 2.5

Mode = 2

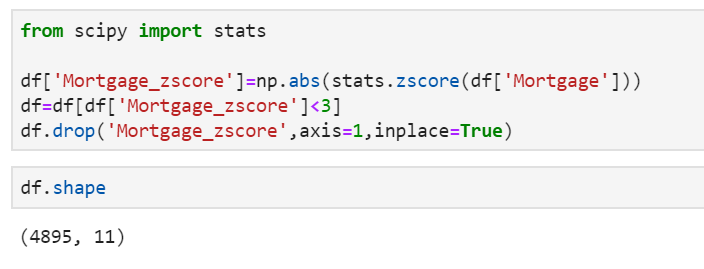
Standard Deviation = 114.74

Having outliers often has a significant effect on the mean and standard deviation and, hence affecting the distribution. We must take steps to remove outliers from our data sets.

Due to these outliers’ bulk of the data in the Mortgage is at the left and the right tail is longer. This is called right skewness. One way to remove the skewness is by doing the z-score.

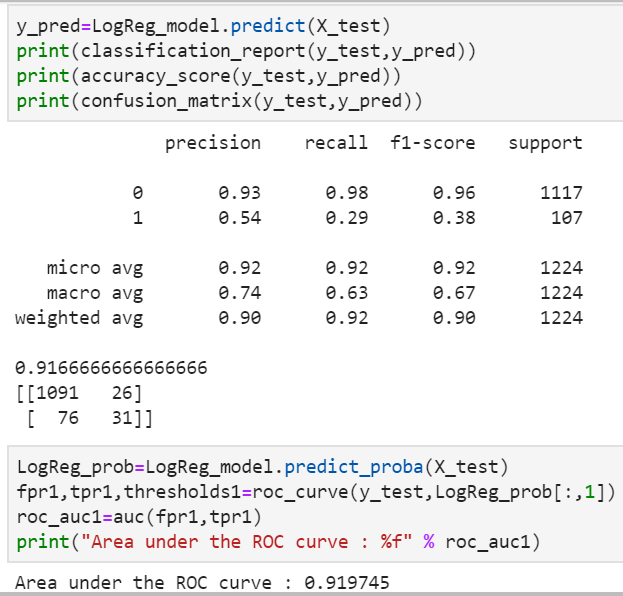
· Wikipedia Definition: The Z-score is the signed number of standard deviations by which the value of an observation or data point is above the mean value of what is being observed or measured.

We can import z-score from the stats library of scipy.



Here I had chosen those rows only whose z\_score is less than 3, it can vary accordingly. Here we had dropped more than 100+ rows which contain outliers and now we can start with the model building.





**Model Building: Logistic Regression:**

Let us make our first model predict the target variable. We will start with Logistic Regression which is used for predicting binary outcome.

* Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables.
* Logistic regression is an estimation of the Logit function. Logit function is simply a log of odds in favor of the event.
* This function creates an S-shaped curve with the probability estimate, which is very similar to the required stepwise function

We will use scikit-learn (sklearn) for making different models which is an open-source library for Python. It is one of the most efficient tools which contains many inbuilt functions that can be used for modeling in Python.

Sklearn requires the target variable in a separate dataset. So, we will drop our target variable from the dataset and save it in another dataset with the help of train\_test\_split function in the model\_selection library of sklearn.

**Model Evaluation Criteria Explained:**

And here we are with a classification report which consists of a precision, recall, f1-score & support, Also the accuracy\_score and a 2\*2 confusion matrix. How to determine if our model has done well…? Well first have a look at the accuracy, 92% accuracy is not a small thing, but we know from the data that the number of buyer’s percentage to the non-buyer percentage is very less. Hence accuracy didn’t play a big role in determining how our model performed.

We will look upon the error types now:

* Type I error - Null hypothesis but predicted an alternate hypothesis by the model
* Type II error - Alternate hypothesis but predicted null hypothesis by the model

In our case, the null hypothesis presents the non-buyer case, while the alternate hypothesis is the buy one.

We must concentrate upon or reduce type II error here since we are interested in the customers who had bought personal loans, but our model predicted them to be a non-buyer.

Eventually, we can concentrate upon our confusion matrix and look for the False Negatives which in this case is 76, less the number of False Negatives, wiser our model will be or we can directly look upon the recall for ‘1’ which in this case is 29%. So, in this case, out of the total number of customers who bought personal loans our model is only able to pick 29% of customers of them to be correctly predicted.

From now onwards, we will print all three, but we can directly compare the recall for ‘1’ to compare different models.

Also, Accuracy (not similar to accuracy\_score) is measured by the area under the ROC curve. An area of 1 represents a perfect test; an area of .5 represents a worthless test. A rough guide for classifying the accuracy of a diagnostic test is the traditional academic point system:

.90–1 = excellent (A) .80-.90 = good (B) .70-.80 = fair © .60-.70 = poor (D) .50-.60 = fail (F)

We will also compare the area under the roc curve to determine how our model performs. In this case, the area comes out to be around 92% which is good.

**STANDARDIZATION:**

Here your data Z is rescaled such that μ = 0 and 𝛔 = 1, and is done through this formula:

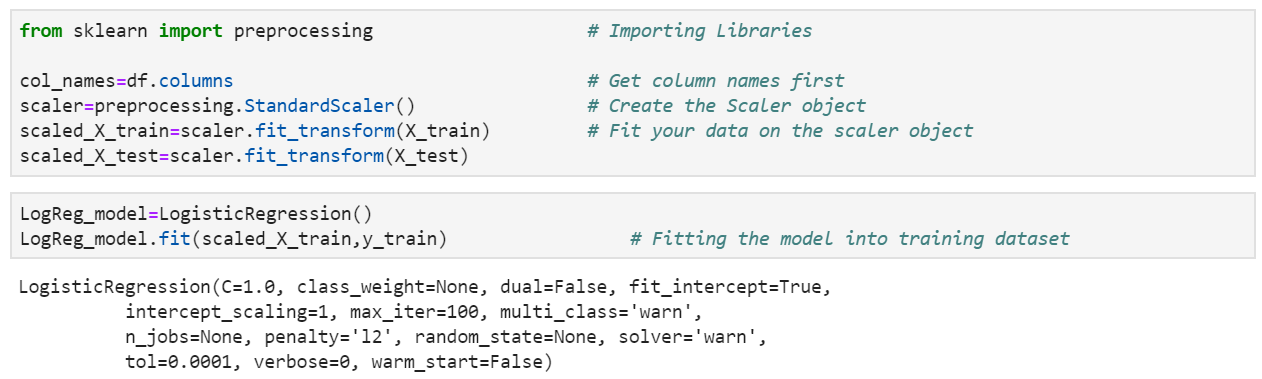
z = x-μ/𝛔

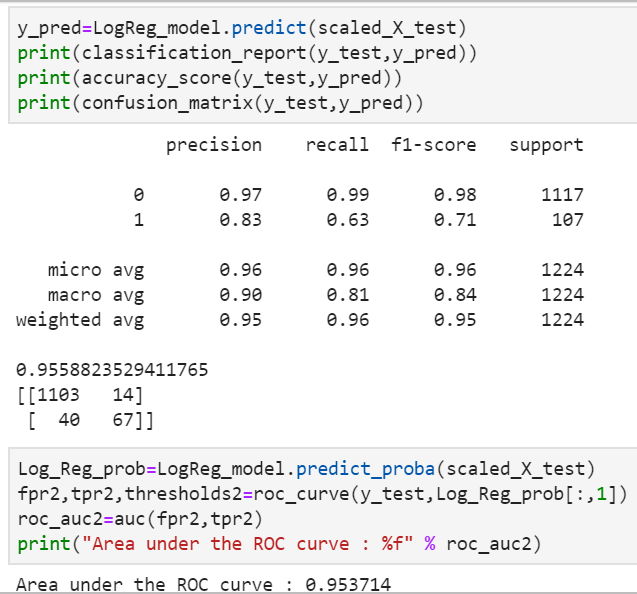
**Why would we do this?**

Compare features that have different units or scales.

Consider our data above, we have different scales and units. We can start to compare these features and use them in our models once we have standardized them.

Later, when you’re running models (logistic regression, SVMs, perceptron’s, neural networks, etc.) the estimated weights will update similarly rather than at different rates during the build process. This will give you more accurate results when the data has been first standardized.





We get an astonishing 4% increase in accuracy and see the difference between evaluation metrics with standardization of the data. As mentioned before, accuracy alone can’t define my model how well it predicted so we will play with recall now.

We get a recall value of 63%, which means our model did much better in predicting True Positives.

Also, the area under the curve is around 95%, much higher than previously.

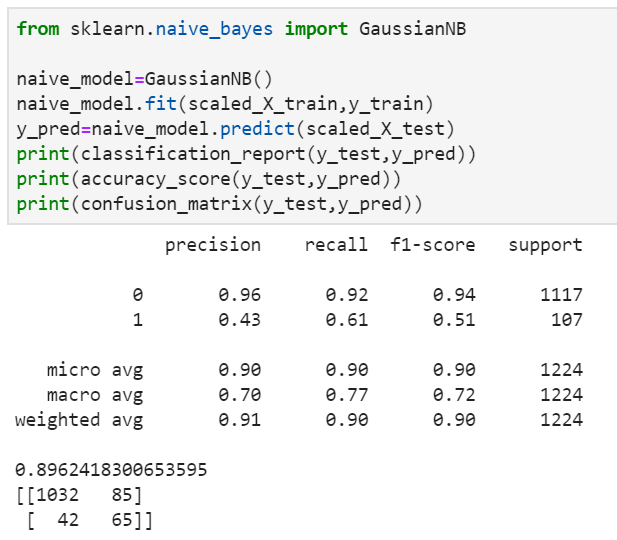
Further, we will analyze other models with only scaled data.

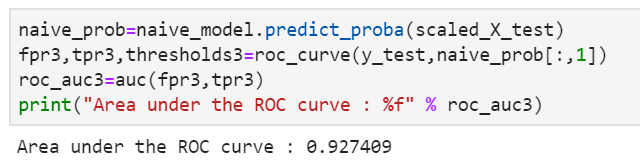
**Naive Bayes:**

What is Bayes Theorem?

* Bayes theorem uses the conditional probability of an event. Events should be mutually exclusive like throwing a dice.
* Bayes Theorem assumes predictors or input features are independent of each other.
* Bayesian probability relates to the degree of belief. It gives the likelihood of an event to occur. It does this with prior knowledge of the condition related to the event

Let’s implement it in python which is as simple as importing the package and fitting the dataset.





We got an accuracy score of around 90% with a recall value of 61% which is much less as compared to the Logistic Regression.

Also, the area under the curve is around 93%, less than the logistic regression one.

Hence Naive Bayes terms out to be not a good classifier with this particular dataset.

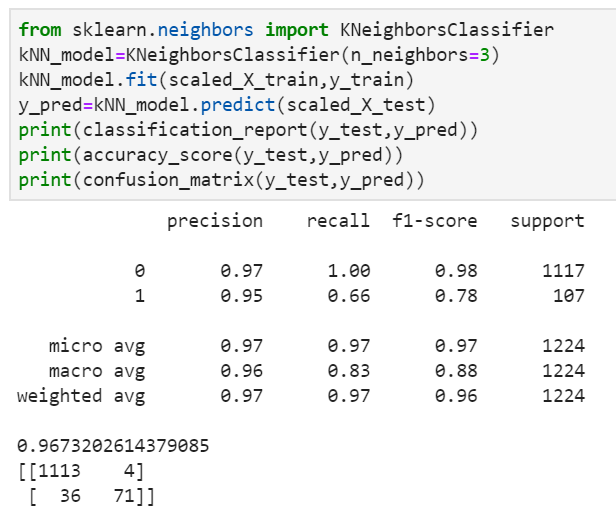
**kNN:**

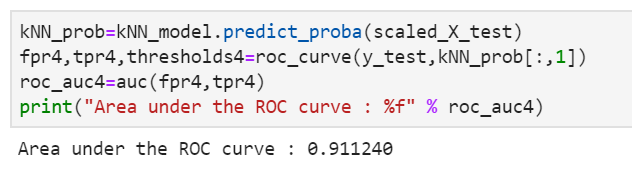
kNN is a very popular algorithm, it is one of the top 10 AI algorithms. Its popularity springs from the fact that it is very easy to understand and interpret yet many times its accuracy is comparable or even better than other, more complicated algorithms.

kNN is a supervised algorithm, it is non-parametric and lazy (instance-based).

It is lazy, because it does not explicitly learn the model, but it saves all the training data and uses the whole training set for classification or prediction. This contrasts with other techniques like SVM, where you can discard all non-support vectors without any problem.

This means that the training process is very fast, it just saves all the values from the data set. The real problem is the huge memory consumption (because we must store all the data) and time complexity at the testing time (since classifying a given observation requires a rundown of the whole data set). But in general, it’s a very useful algorithm in case of small data sets (or if you have lots of time and memory) or for educational purposes.





We import the kNN Classifier from sklearn. This takes multiple parameters. The most important parameters are:

* n\_neighbors: the value of k, the number of a neighbor’s considered
* weights: if you want to use weighted attributes, here you can configure the weights. This takes values like the uniform, distance (inverse distance to the new point), or callable which should be defined by the user. The default value is uniform.
* algorithm: if you want a different representation of the data, here you can use values like ball\_tree, kd\_tree, or brute, default is auto which tries to automatically select the best representation for the current data set.
* metric: the distance metric (Euclidean, Manhattan, etc), default is Euclidean.

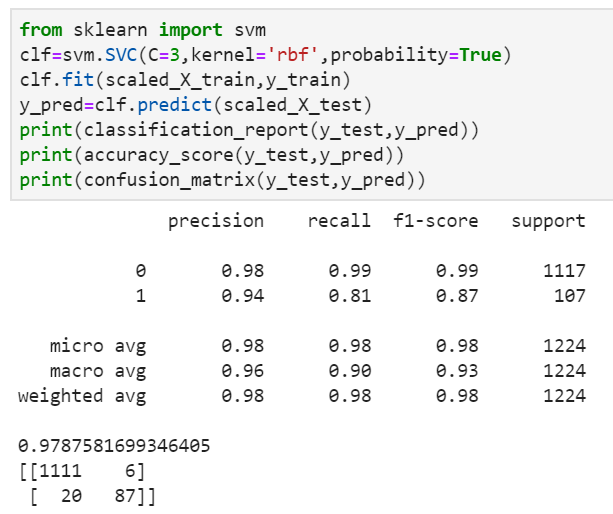
We leave all the default parameters, but for n\_neighbors we had use 3 (the default is 5).

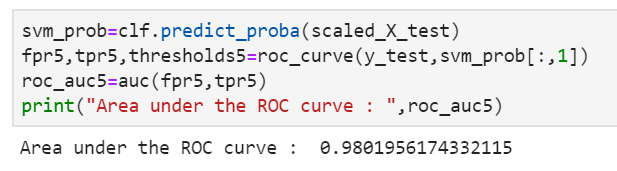
And here we are with around 97% accuracy in determining if a customer will buy the personal loan or not. Also, the recall value is 66% is much better than logistic regression and Naive Bayes algorithms. Also, the area under the curve is fairly good.

**SVM (Support Vector Machine):**

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two-dimensional space, this hyperplane is a line dividing a plane into two parts wherein each class lay on either side.

Let us see classification between a buyer and a non-buyer.





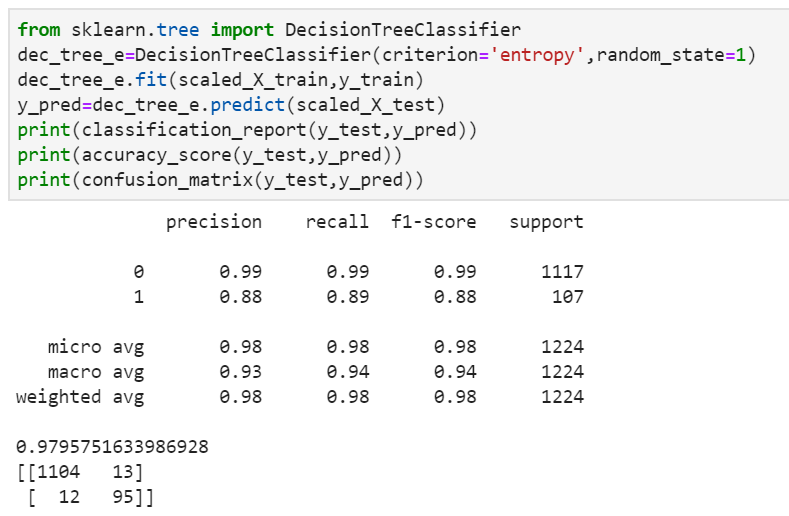
We got a 98% accuracy score with 81% recall value, also the area under the curve is about 98%.

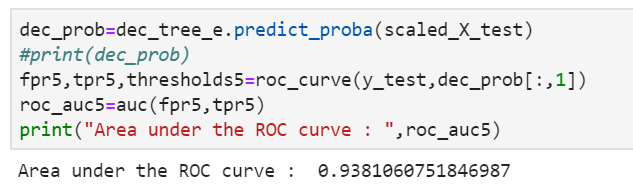
SVM can be deployed to predict the likelihood of a liability customer buying personal loans but we will look further if trees can predict better or not.

**Decision Trees:**

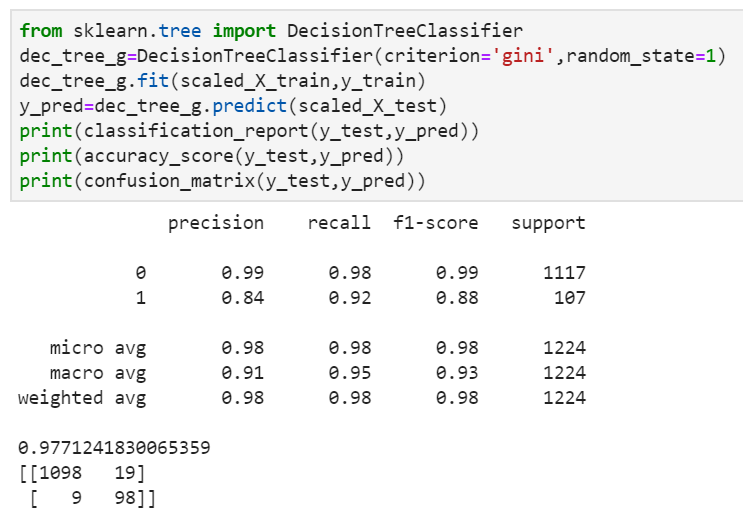
A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

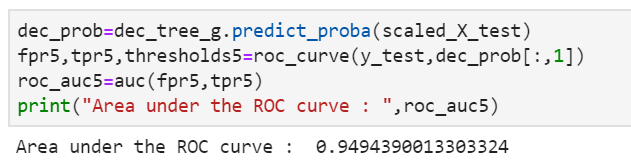
Let us see if the decision tree with criterion as entropy can nail it down to higher recall value or with criterion as gini. We will also show you the part of bagging and boosting later on.



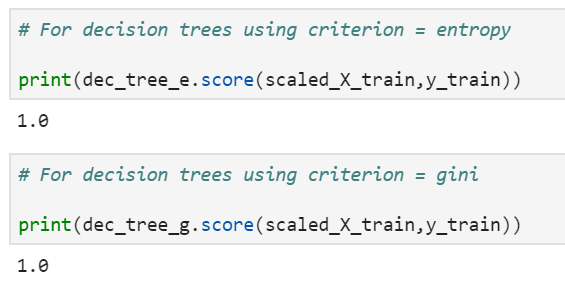


We got a 98% accuracy score while 89% recall value but if you look at the AUC for decision trees, you will realize it is much smaller than SVM AUC. Something is fuzzy here (as accuracy and recall value are very high but the area under the curve is low), I will explain it down the line but first, let’s look at the decision tree with criterion ‘gini’ :

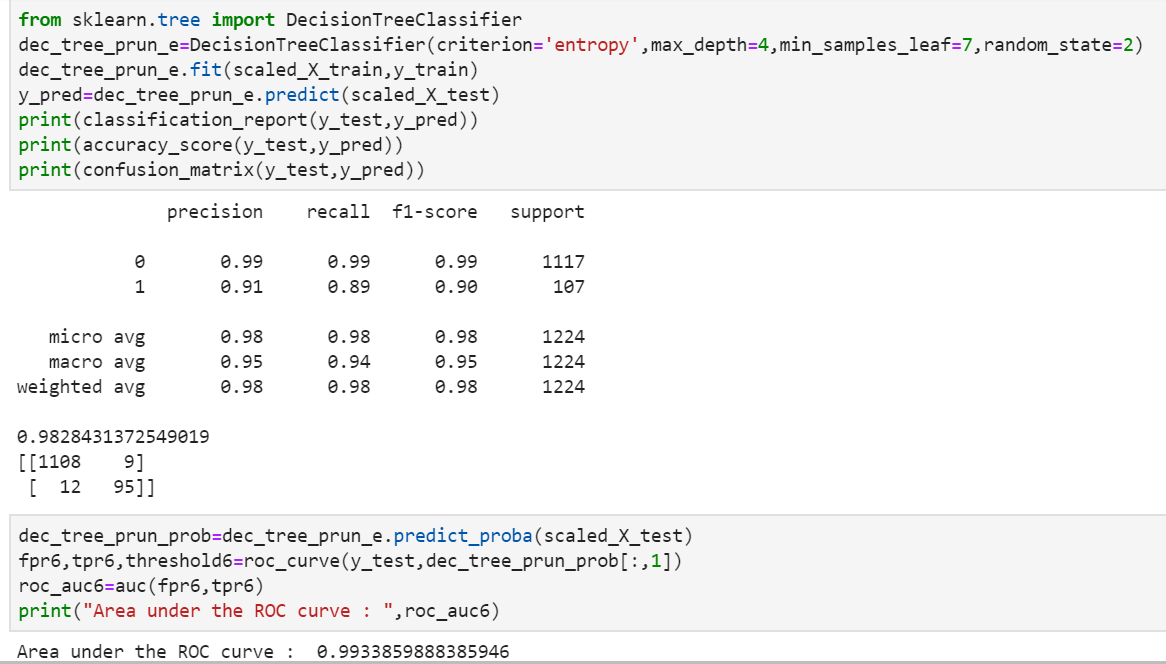




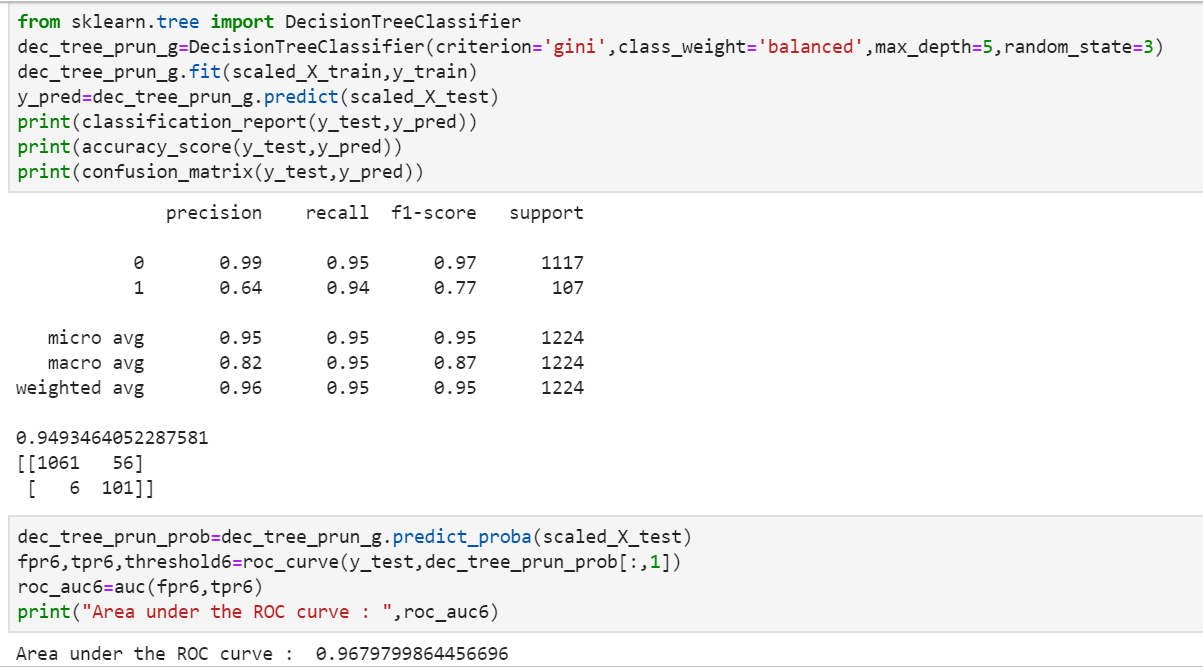
We got a 98% accuracy score while a 92% recall value but the area under the ROC curve is much smaller in this case too. The reason being the **‘**overfitting**’**of the data. Let us check the accuracy score for training as well as test data.



The decision tree's biggest disadvantage is it overfits the data. To overcome this, we use hyper-parameters to cutting down its branches so that it generalizes well. Let’s have a look at that:



Difference: We got 98% accuracy with 89% recall value, also the AUC is 99% which is fairly good. Let us check the same for the criterion ‘Gini’ then we can conclude to the results of decision trees.



94% recall value with a 95% accuracy level, also AUC is approximately 97% which is fairly good.

**Combine Model Predictions into Ensemble Predictions:**

The three most popular methods for combining the predictions from different models are:

1) Bagging- Building multiple models (typically of the same type) from different subsamples of the training dataset.

2) Boosting- Building multiple models (typically of the same type) each of which learns to fix the prediction errors of a prior model in the chain.

3) Voting - Building multiple models (typically of differing types) and simple statistics (like calculating the mean) are used to combine predictions.

This post will not explain each of these methods just give a brief introduction.

It assumes you are generally familiar with machine learning algorithms and ensemble methods and that you are looking for information on how to create ensembles in Python.

**Bagging Algorithms:**

Bootstrap Aggregation or bagging involves taking multiple samples from your training dataset (with replacement) and training a model for each sample. The final output prediction is averaged across the predictions of all of the sub-models.

The three bagging models covered in this section are as follows:

1) Bagged Decision Trees

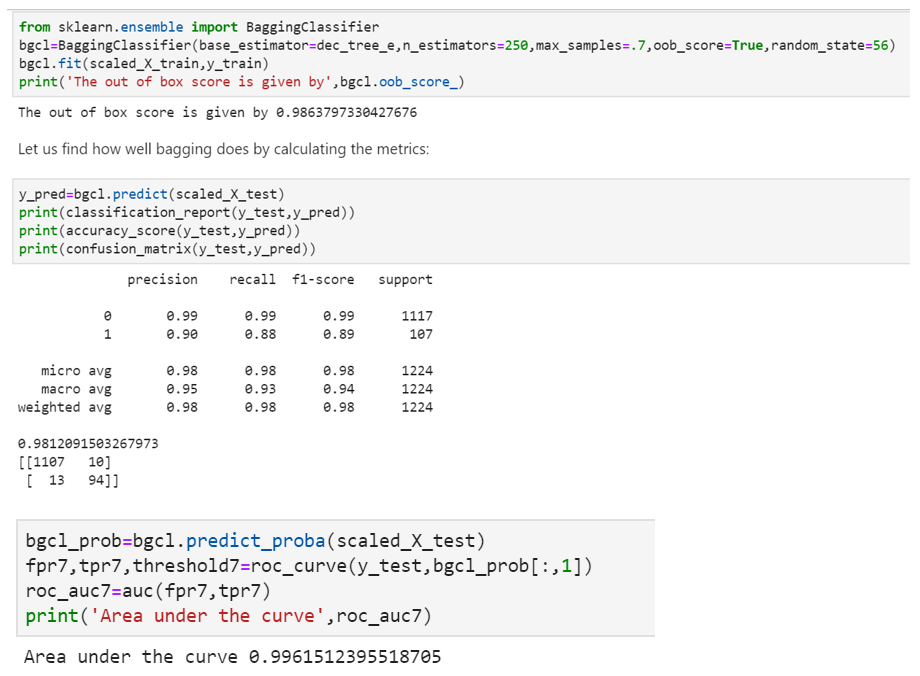
2) Random Forest

3) Extra Trees

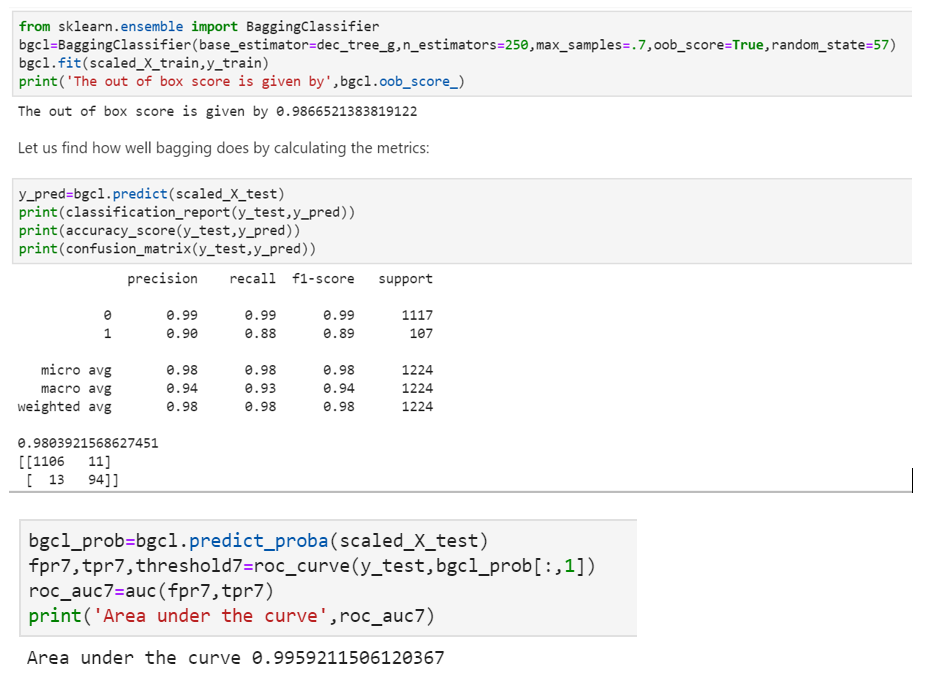
**Bagged Decision Trees:**

Bagging performs best with algorithms that have high variance. A popular example is decision trees, often constructed without pruning.

In the example below, we will be using the Bagging Classifier with the Classification and Regression Trees algorithm (Decision Tree Classifier). A total of 250 trees are created.



We got 98% accuracy with 88% recall value which is smaller than Decision Trees but we got a 99.6% area under the curve. Also, here another parameter called out of bag score which is 98.63% which specifies the number of correctly predicted rows from the out of bag sample. Previously if we look, while calling the Bagging Classifier, I specified base estimator to be decision tree using the criterion of entropy, let us check for Gini also:



Similar to the entropy criterion.

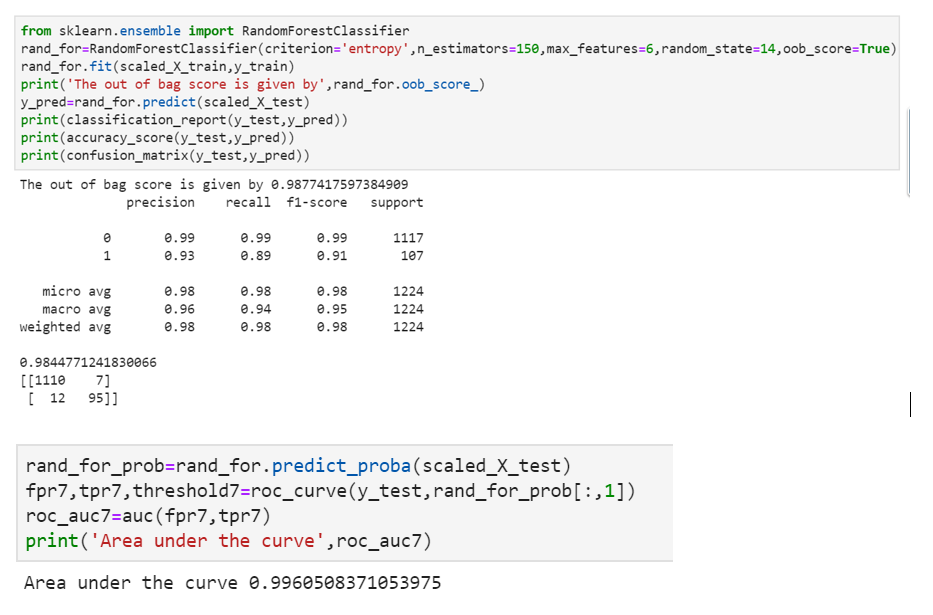
**Random Forest:**

Random forest is an extension of bagged decision trees.

Samples of the training dataset are taken with replacement, but the trees are constructed in a way that reduces the correlation between individual classifiers. Specifically, rather than greedily choosing the best split point in the construction of the tree, only a random subset of features is considered for each split.

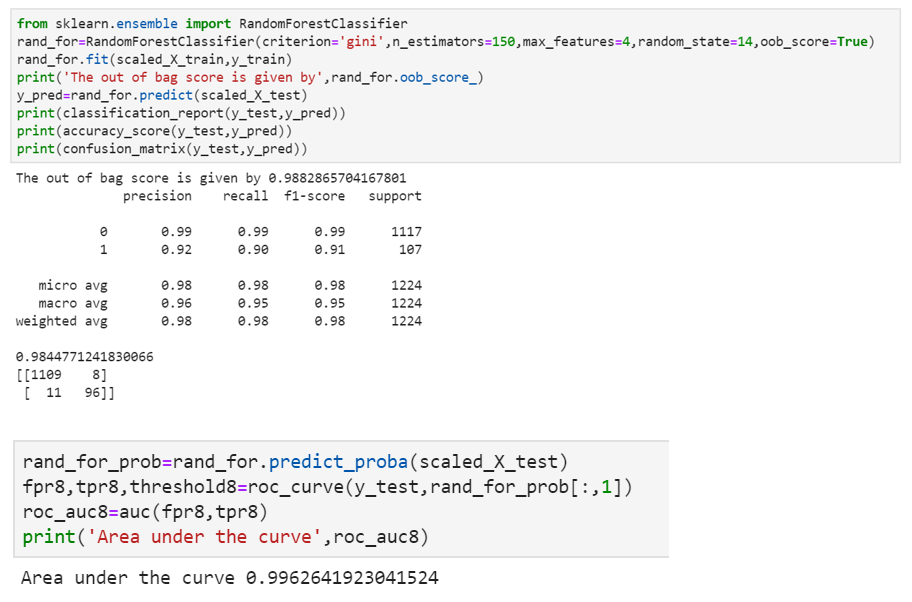
We can construct a Random Forest model for classification using the Random Forest Classifier class.

The example below provides an example of Random Forest for classification with 150 trees and split points chosen from a random selection of 6 features.



We got 98% accuracy with 89% recall value and an astonishing 99.6% area under the curve. Also, oob score counts to around 99%.

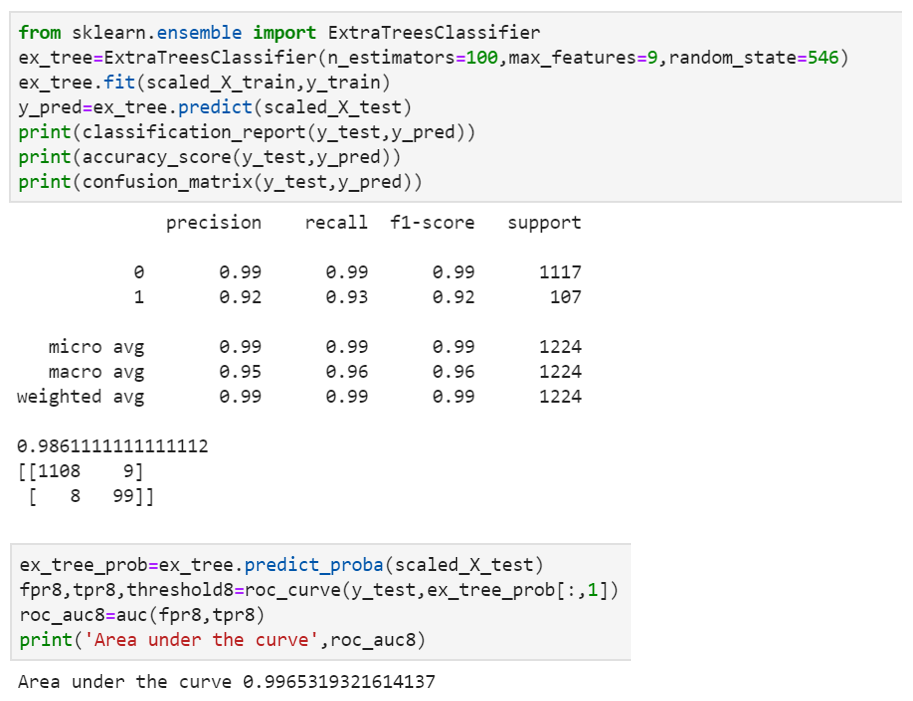
Let’s try with the criterion ‘gini’:



We got 98% accuracy with 90% recall value and an astonishing 99.6% area under the curve. Also, oob score counts to around 99%.

**Extra Trees:**

Extra Trees are another modification of bagging where random trees are constructed from samples of the training dataset. We can construct an Extra Trees model for classification using the ExtraTreesClassifier class. The example below provides a demonstration of extra trees with the number of trees set to 100 and splits chosen from 7 random features.



We got an astonishing 93% recall value with 98.6% of accuracy and area under the curve being 99.65%

Let us look for boosting techniques:

**Types of Boosting Algorithms:**

The underlying engine used for boosting algorithms can be anything. It can be a decision stamp, margin-maximizing classification algorithm, etc. There are many boosting algorithms which use other types of an engine such as:

1) AdaBoost (Adaptive Boosting)

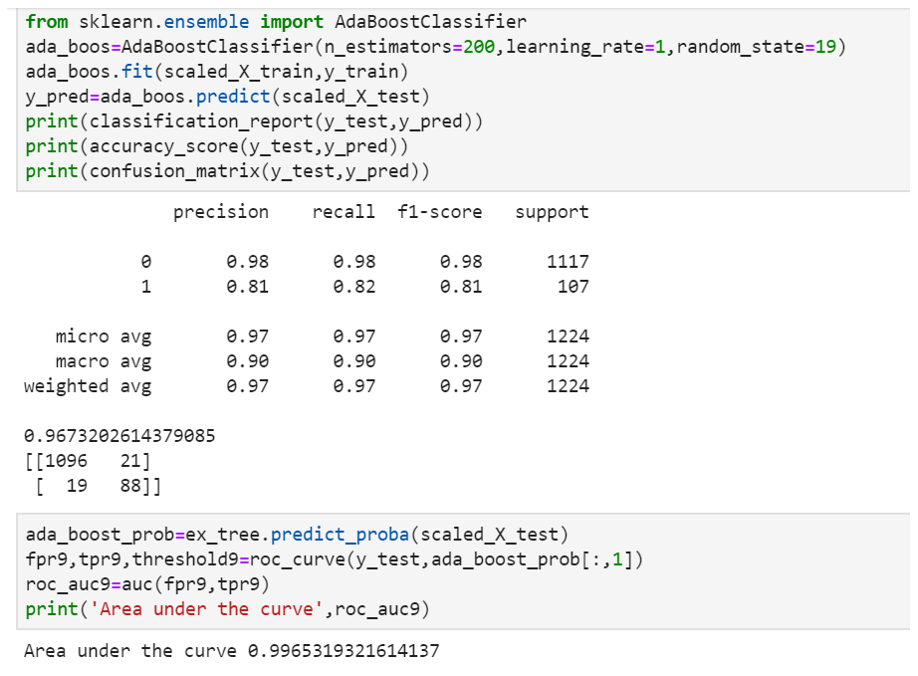
2)Gradient Tree Boosting

3)XGBoost (eXtreme Gradient Boosting)

**Boosting Algorithm: AdaBoost**

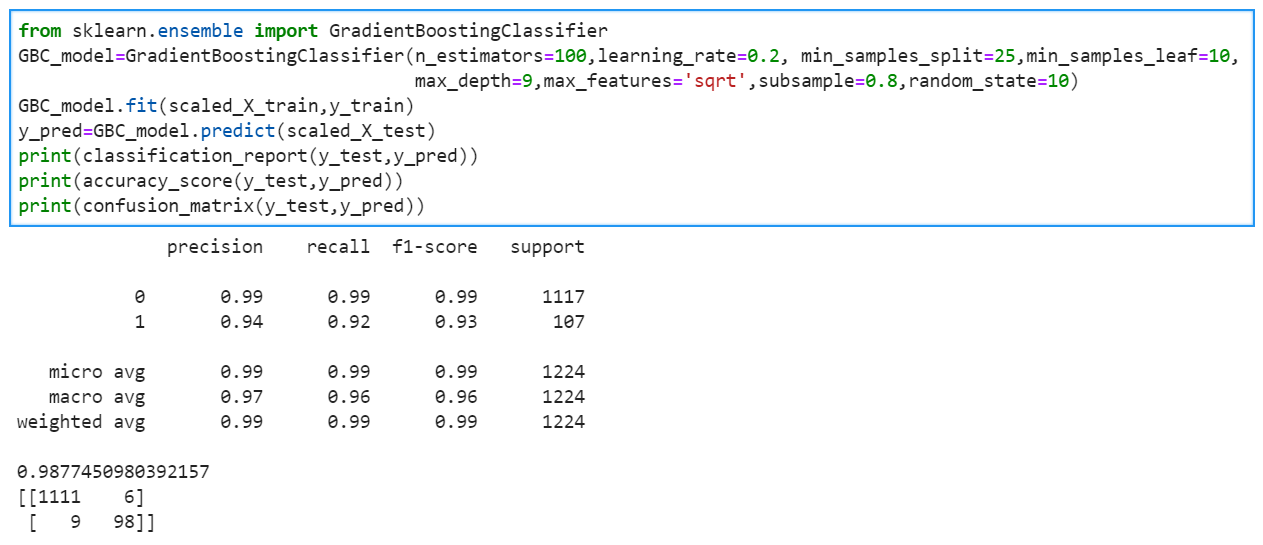
It works on a similar method as discussed above. It fits a sequence of weak learners on different weighted training data. It starts by predicting the original data set and gives equal weight to each observation. If the prediction is incorrect using the first learner, then it gives higher weight to observation which has been predicted incorrectly. Being an iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy.

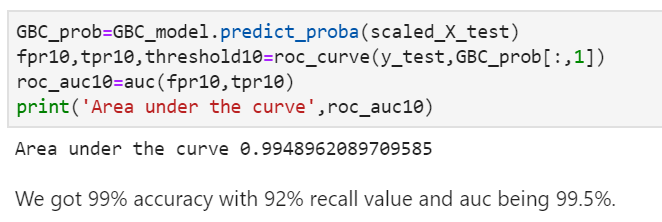
Mostly, we use decision stamps with AdaBoost. But we can use any machine learning algorithms as a base learner if it accepts weight on training data set. We can use AdaBoost algorithms for both classification and regression problems.



**Boosting Algorithm: Gradient Boosting**

In gradient boosting, it trains many models sequentially. Each new model gradually minimizes the loss function (y = ax + b + e, ‘e’ needs special attention as it is an error term) of the whole system using the Gradient Descent method. The learning procedure consecutively fit new models to provide a more accurate estimate of the response variable.

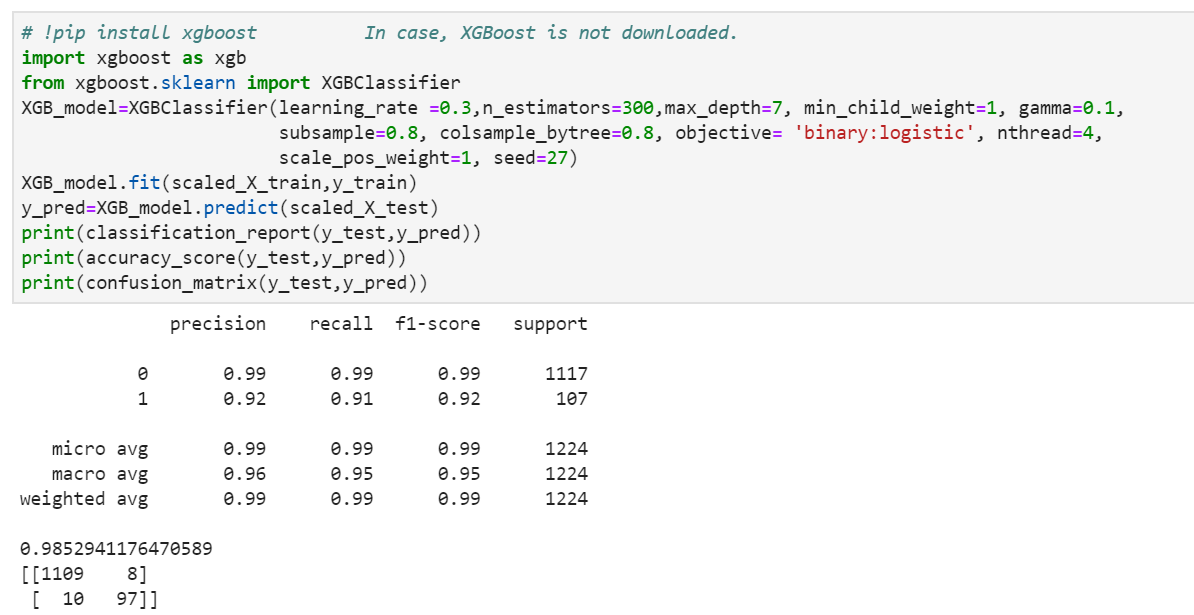


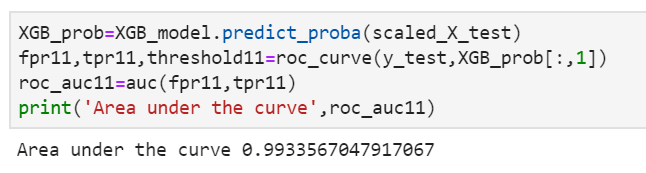


**Boosting Algorithm: eXtreme Gradient Boosting (XGBoost)**

If things don’t go in predictive modeling, we can use XGboost. XGBoost algorithm has become the ultimate weapon of many data scientists. It’s a highly sophisticated algorithm, powerful enough to deal with all sorts of irregularities of data.

Building a model using XGBoost is easy, but improving the model using XGBoost is difficult. This algorithm uses multiple parameters. To improve the model, parameter tuning is a must.





We got a 91% recall value with 98.5% accuracy while the AUC is 99.33%. While we can play with the hyperparameters to check if we can get more recall value either we can use a grid search or random search.