*Week 5 -Project-Bank Loan Case Study report*

Submitted by:

PRANAV.S

Objective: -

To analyze and identify the potential customers who have higher probability of purchasing the loan.

**Table of content**

[Introduction 3](#_Toc120298809)

[1. Understanding Data 4](#_Toc120298810)

[2. Data Wrangling 5](#_Toc120298811)

[3. Logistic Regression 6](#_Toc120298812)

[4. k-Nearest Neighbors 9](#_Toc120298813)

[5. Decision Tree 11](#_Toc120298814)

[6. Random Forest 12](#_Toc120298815)

[7. Conclusion 14](#_Toc120298816)

Executive summary

Thera Bank is a bank with a sizable customer base. Most of these clients are liability customers (depositors), and their deposits range in amount. Asset customers make up a very tiny portion of the bank's customer base, and the bank is keen to swiftly grow this group to increase loan activity and, as a result, increase interest revenue. The management is especially interested in finding ways to turn its liability consumers into personal loan customers (while retaining them as depositors). The bank executed a campaign for liability customers last year that had a successful conversion rate of over 9%. The retail marketing department has been inspired by this to create campaigns that utilize better target marketing to increase.

In this project a model has been developed to identify the potential borrowers who are most likely to avail the loan.

Dataset contains of 14 attributes and 5000 records.

The model is built using different machine learning models in python like logistic regression, KNN model, decision tree and random forest.

# Introduction

The aim of this project is to pinpoint the clients from the previous campaign who are utilizing personal loans as well as those who are far more likely avail personal loan . As a result, the bank will be able to target these clients more effectively, which will lower campaign costs and boost the bank's success rate.

The data set given by the bank to create the required model contains the data on 5000 customers who were targeted in the previous campaign. This data contains customer information like, their age, income, education, etc. along with the relationship of customers with the bank like mortgage, security accounts, etc.

The first step in the project is to understand the data and then cleaning the data to make is more precise. And next to check for relationships between the variables.

The independent variables are being separated and are stored in a form of x- array and the dependent variables from the data set and are stored into y- array.

After separating the targeted data from the bank data set, the data is prepared for Logistic Regression by training and testing it and since logistic regression cannot be worked on data frames it was needed to create separate arrays.

Once the training and testing of the dependent array and independent array is done, the logistic regression is performed to predict the personal loan affinity -removing the binned columns considering only 11 columns which are numeric. In this step model accuracy score is calculated(predicted).

Using other classification models too like KNN, Decision tree, Random Forest the prediction of the customers buys personal is done.

column discriptions:

* ID: Customer ID
* Age: Customer's age in completed years
* Experience:  #years of professional experience
* Income:  Annual income of the customer ($000)
* ZIPCode: Home Address ZIP code.
* Family: Family size of the customer
* CCAvg: Avg. spending on credit cards per month ($000)
* Education: Education Level.1: Undergrad; 2: Graduate; 3: Advanced/Professional
* Mortgage:  Value of house mortgage if any. ($000)

# Understanding the Data

The first step to start with any data set is to understand the data. By understanding the data, it means to know all the details of the dataset provides, like number of columns, column names, what does they describe along with the datatype of each column. the statistical distribution of each numerical column.

It is also important of check for the null values as well as any negative values.

In this data set, after reading the data set in python,

**df.describe()** is used to describe the dataset, giving all the statistical analysis like (count, mean, standard deviation, minimum, 25%, 50%, 75% and maximum) of the numerical columns in the dataset.

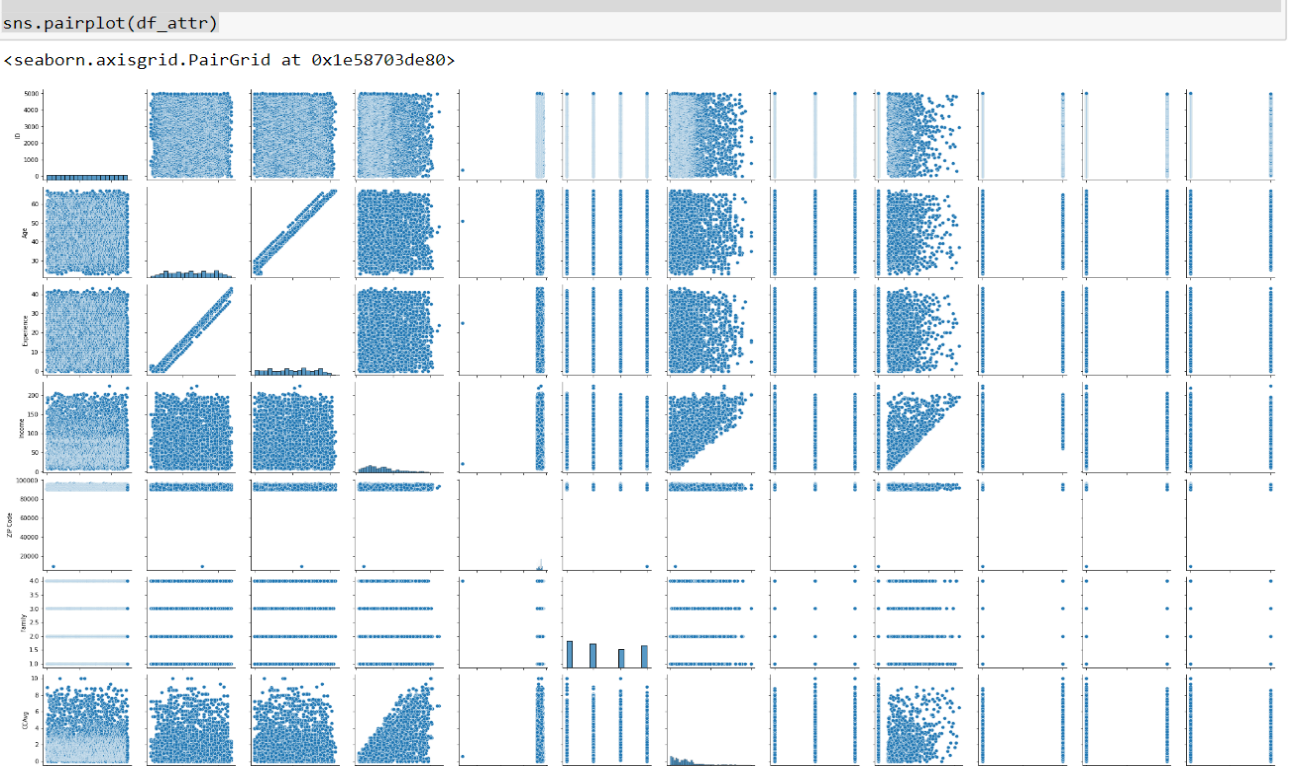
As there are negative numbers in experience! maybe typing error. Convert to non-negative using .abs function

**df['Experience'] = df['Experience'].abs()**

Next using seaborn library, statistical plots are plotted to give a comparison of all the attributes of the data frame visually to check for the relationship than can be exploited**.**

**import seaborn as sns**

**df\_attr = df.iloc[:,0:12]**

**sns.pairplot(df\_attr)**

# Data Wrangling

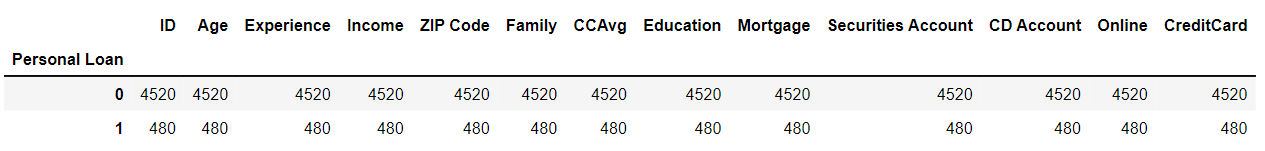
To make complicated data sets more accessible and understandable, data wrangling is the act of cleaning up errors and merging different complex data sets. Large amounts of data need to be stored and organized for analysis because the amount of data and data sources available today are expanding quickly.

Data wrangling, also referred to as data munging, is the act of rearranging, changing, and mapping data from one "raw" form to another to increase its value and usability for a range of downstream uses, including analytics.

To give datatype of all the columns of the given data frame using function: **df.dtypes**

The data set is skewed in terms of target column. There are far few records in class 1 i.e. people who took the personal loan last time. But that is the class of interest. We must identify potential customers and do not want any potential customer to be missed. So to get the desired count of the customers who have availed personal loan groupby function is uses.

**df.groupby(["Personal Loan"]).count()**

****

# Logistic Regression

Classification issues are resolved via logistic regression. In contrast to linear regression, which predicts a continuous outcome, it achieves this by forecasting categorical outcomes.

The simplest scenario, known as a binomial case, has two outcomes: one customer purchases a personal loan, and the other consumer does not purchase a personal loan.

Here, we shall forecast a binomial variable using fundamental logistic regression. Thus, there are only two events that could occur.

As a linear classifier, logistic regression uses the function

also known as the logit. The regression coefficients, often known as projected weights or just coefficients, are estimated by the variables b0, b1,..., and br.

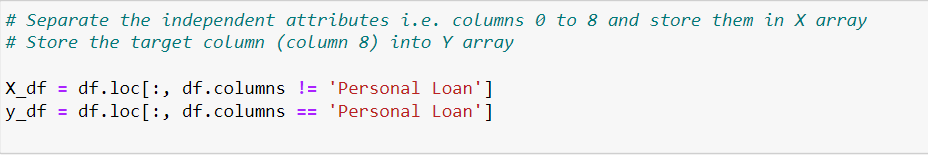
The sigmoid function of f(x) is the logistic regression function

As a result, it frequently falls between 0 and 1. The output of the function p(x) is frequently understood as the projected probability that x is equal to 1. As a result, the likelihood that the output is zero is 1 p(x).

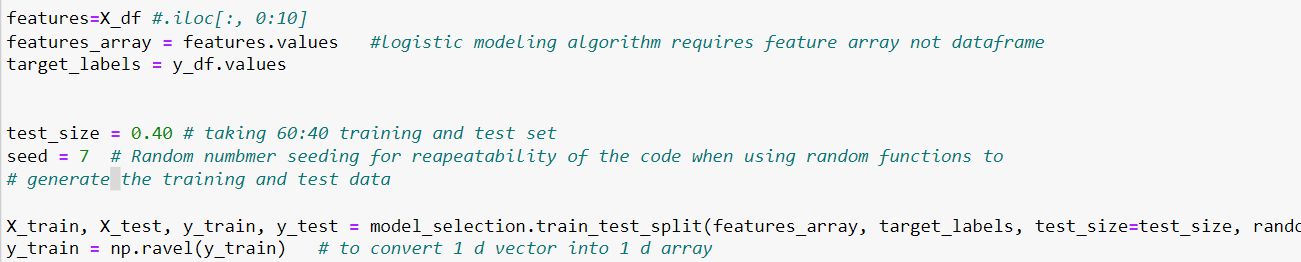
The best projected weights are determined via logistic regression in such a way that the function p(x) is as near to all actual answerss possible, where n is the number of observations. Model training or fitting is the process of selecting the optimal weights based on the available observations.

In Python we have modules that will do the work for us. Start by importing the NumPy module then sorting independent and dependent variables.

Separate the independent attributes i.e. columns 0 to 8 and store them in X array. Store the target column (column 8) into Y array for training and testing the data.

****

Create the training and test data set in the ratio of 60:40 respectively. Prepare data for logistic regression

****

We will use a method from the sklearn module, so we will have to import that module as well:

**from sklearn import linear\_model**

From the sklearn module we will use the LogisticRegression() method to create a logistic regression object.

**sklearn.linear\_model import LogisticRegression**

This object has a method called fit() that takes the independent and dependent values as parameters and fills the regression object with data that describes the relationship:

The precision and recall for class 1 are low

**Precision**: Within a given set of positively labelled results, the fractions that were

,

this must be calculated for each class i.e., 0 and 1 and should be high for the class less represented, class 1 in our example

**Recall:** Given a set of positively labelled results, the fraction of all positives that were

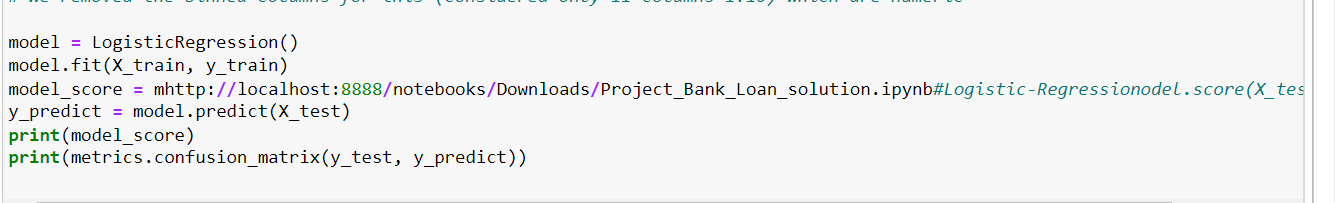
**Accuracy**:

But this measure can be dominated by larger class. Suppose 10, 90 and 80 of 90 is correctly predicted while only 2 of 0 is predicted correctly. Accuracy is 80+2 / 100 i.e., 82%

F is harmonic mean of precision and recall given by When B is set to 1, we get .

**Code:**

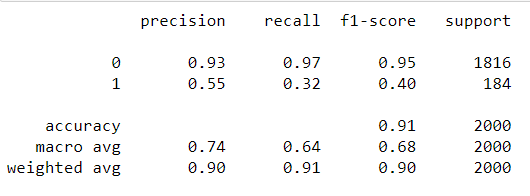
Let us first try logistic regression to predict the personal loan affinity - We removed the binned columns for this (considered only 11 columns 1:10) which are numeric

****

The accuracy score of .954 looks impressive but do not forget, it is unreliable as it is a score at model level. Let us look at class level, especially the class 1. summarize the fit of the model

**print(metrics.classification\_report(y\_test, y\_predict))**

Result:

****

# k-Nearest Neighbors

To address the categorization model difficulties, this algorithm is utilized. The K-nearest neighbor technique, also known as K-NN, essentially draws an illogical boundary to categories the data. The programmed will attempt to anticipate additional data points to the nearest boundary line as they are received.

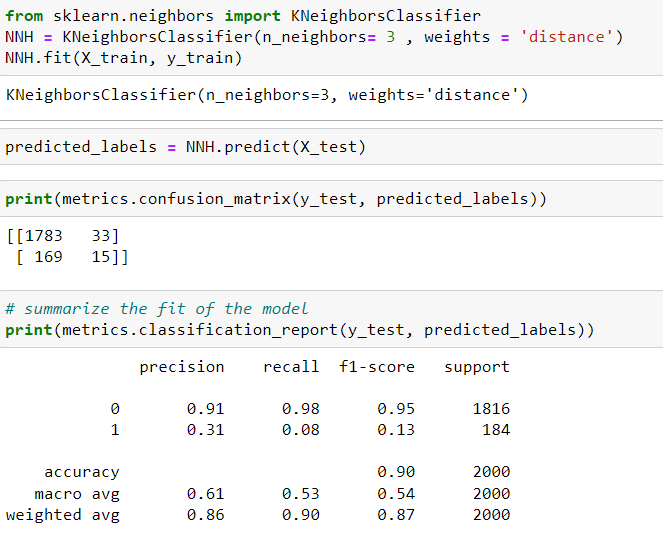
Larger k values result in smoother curves of separation, which lead to fewer complex models. Smaller k values, on the other hand, tend to overfit the data and produce complex models.

To prevent overfitting and underfitting of the dataset, it is crucial to use the proper k-value when analyzing the data.

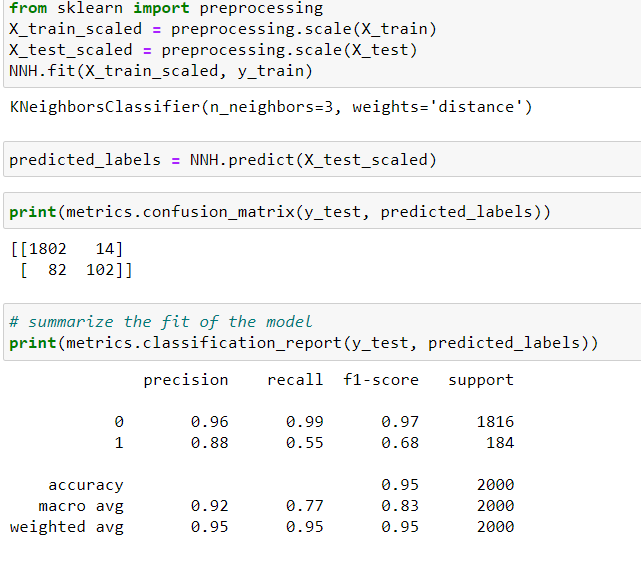
We fit the past data (or train the model) using the k-nearest neighbor approach, and then project the future.

Steps performed while working on KNN algorithm:

1. The k-nearest neighbor algorithm is imported from the scikit-learn package.
2. Create feature and target variables.
3. Split data into training and test data.
4. Generate a k-NN model using neighbors value.
5. Train or fit the data into the model.
6. Predict the future.



Recall (true positives / (true positives + false negatives)) for class 1 is the least. That because majority of data points belong to class 0 and in KNN, probability of finding data point from class 0 closer to a test point than a data point from class 1 is high. let us check the effect of scaling (convert all dimensions to z scores).



Scaled KNN algorithm gives the best result for class 1 till now. Let us check decision tree

# Decision Tree

An internal node represents a feature (or attribute), a branch represents a decision rule, and each leaf node indicates the conclusion in a decision tree, which resembles a flowchart. The root node in a decision tree is the first node from the top. It gains the ability to divide data according to attribute values.

A class called DecisionTreeClassifier may do multi-class classification on a dataset.

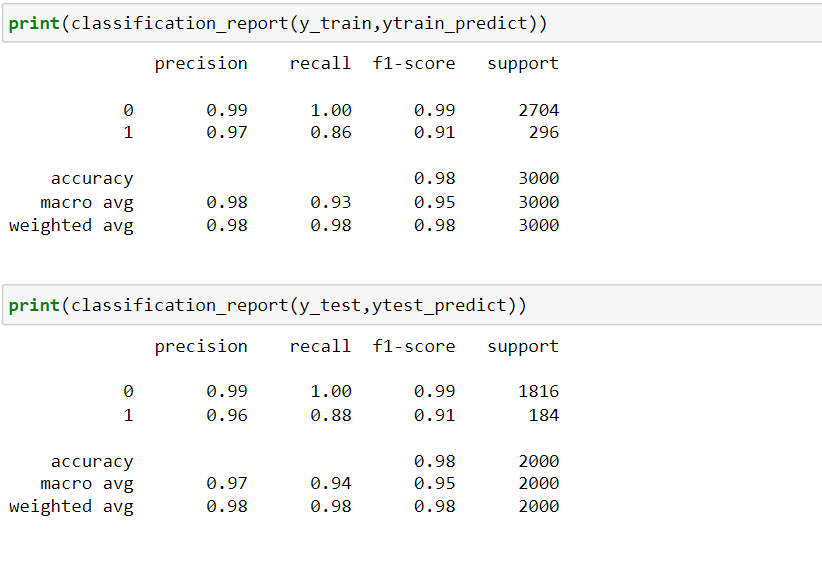
The DecisionTreeClassifier, like other classifiers, accepts two arrays as input: an array X, sparse or dense, of form (n samples, n features), storing the training samples, and an array Y, of integer values, shape (n samples), holding the class labels for the training data.

After being fitted, the model then is used to predict the class of samples.

In this model, next we have imported the classification report to give the accurate classification of the test and train model, from sklearn.matrix.

This classification report is between y-train and y-train predicted, and y-test and y-test predicted.

Result of classification report:



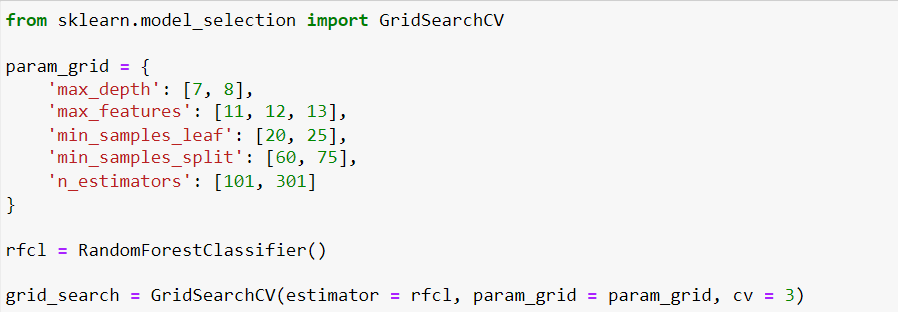
# Random Forest

A decision tree ensemble method called random forest is created using a dataset that has been randomly divided and is based on the divide-and-conquer strategy. The forest is a collective term for these decision tree classifiers. An attribute selection indicator, such as information gain, gain ratio, and Gini-index for each attribute, is used to create the individual decision trees. Every tree relies on a distinct random sample. Each tree casts a vote in a classification problem, and the class with the most votes wins out in the end. The outcome in a regression scenario is taken to be the average of every tree output. When compared to other non-linear classification algorithms, it is both easier and more effective.

It works in four steps:

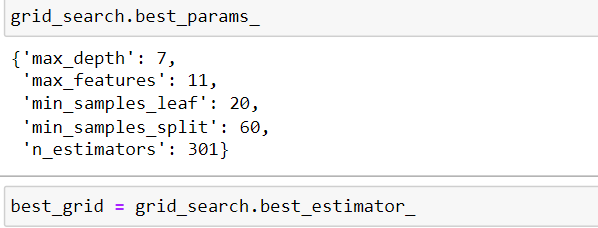
1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

First, separate the columns into dependent and independent variables (or features and labels). Then split those variables into a training and test set.

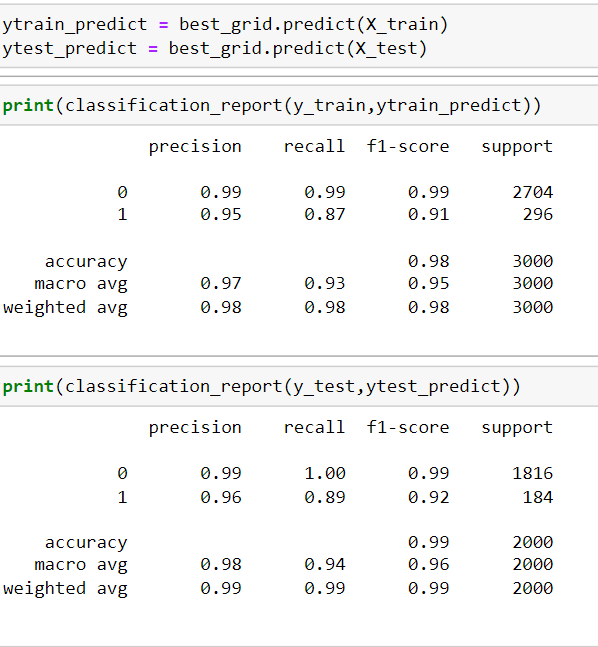




After splitting, train the model on the training set and perform predictions on the test set.



After training, check the accuracy using actual and predicted values.



# Conclusion

After performing different machine learning algorithm to create a confusion matrix for the given banking data set to predict the potential customers who would buy the personal loan, it can be clearly stated the best model to work on is the random tree model.

1. Below are some points that explain why we should use the Random Forest algorithm:
2. In comparison to other algorithms, it requires less training time.
3. Even with the enormous dataset, it operates effectively and predicts the outcome with a high degree of accuracy.
4. When a significant amount of the data is absent, accuracy can still be maintained.
5. we have implemented robusting model decision tree (where we don't have any pre-processing like no scaling)
6. have implemented final model as random forest with the best hyperparameters we fit a model where are accuracy is comparable with 98%
7. finally we got the recall of 89% compare with base model 56% of false negative are reduced (where the random forest is good for prediction)
8. so, the best model is the random forest to predict