Micro-Credit Defaulter Project

Submitted by:

Ganta Ganesh

ACKNOWLEDGMENT

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.

INTRODUCTION

During the last few decades, credit quality emerged as an essential indicator for banks' lending decisions. Numerous elements reflect the borrower's creditworthiness, and the use of credit scoring mechanisms could moderate the estimation of the probability of default (PD) while predicting the individual's payment performance. The existing literature concentrates on understanding why organizations' lending mechanisms are successful at decreasing defaults or addressing the issues from an economic theory perspective. More specifically, it reported that almost one-third of the world's adult population were unbanked, according to a 2017 world bank report. Therefore, they rely on micro-finance institutions' services. Jarrow and Protter (2019) acknowledged the gap in the existing literature on how to determine fair lending rates in micro-finance, which would be granting lending access (or credit) to low-income populations excluded from traditional financial services.

Machine Learning

Machine learning is a capacity of the machines to analyse a set of data and build generic algorithms. There is no need to write codes, just feeding of data is enough it builds its logic based on it.

There are majorly two types of machine learning

- Supervised learning
- Un supervised learning

Supervised Learning

From the above context itself, it is so much evidence that there must be some supervisor as a teacher to carry on the process. Usually in this methodology, the training or the teaching is provided.

For Example, if it is to train to identify various kinds of fruits it must be like the shape of the fruit is round and a cavity is found at the top centre and the colour must be red. Which signifies Apple? If the shape is curved and long enough with the colour of green or yellow then it must be Banana. Now after this training it is given with another set of examples to identify.

Un Supervised Learning

This second type of learning in which there is no supervision or guidance required is called unsupervised learning. Here it can act without any guidance required. For example, if a picture of dogs and cats is given together to analyse, it has no information on Cats or Dogs. But still, it can categorize based on the similarities between them by analysing the patterns, Size, shape, figures, and differences.

Coverage of the Study

This report is restrained about machine learning model building that predicts the loan default status of micro credit loan transaction.

Source of Data

The study is based on secondary data collected through various internet web sites.

Data Analysis

Analysis of data and the information collected from the secondary sources were made keeping the objectives of the study in mind.

Project Definition

To create a model that predicts the loan payment status to micro credit organisation by the customers.

Hardware and Software Requirements

SYSTEM SPECIFICATION

The hardware and the software specifications of the projects are

1) Hardware Requirements

Processor: Intel I 3

Ram: 4 GB

Hard disk Driver: 50 GB

Monitor: 15” Colour monitor

2) Software requirements

OS: Linux/ Windows/ MAC

Language: Python

Libraries: Jupyter notebook, Python, Matplot lib, Pandas, Numpy, Ibmlearn.

PROJECT DESCRIPTION

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.

Idea

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.

Solution

Using classification models, we need to build a model that gives best predictions.

Summary

The ultimate goal of the project is to provide best information on repayment of loan by customer.

Classification

In machine learning, classification refers to a predictive modelling problem where a class label is predicted for a given example of input data.
Classification is a task that requires the use of machine learning algorithms that learn how to assign a class label to examples from the problem domain. An easy-to-understand example is classifying emails as "*spam*" or "*not spam*."

Data Analysis:

We begin data analysis by first getting information about all the variables in our data.

label = Flag indicating whether the user paid back the credit amount within 5 days of issuing the loan {1:success, 0:failure}
msisdn = mobile number of user
aon = age on cellular network in days
daily_decr30 = Daily amount spent from main account, averaged over last 30 days (in Indonesian Rupiah)
daily_decr90 = Daily amount spent from main account, averaged over last 90 days (in Indonesian Rupiah)
rental30 = Average main account balance over last 30 days Unsure of given definition rental90 = Average main account balance over last 90 days Unsure of given definition last_rech_date_ma = Number of days till last recharge of main account
last_rech_date_da = Number of days till last recharge of data account
last_rech_amt_ma = Amount of last recharge of main account (in Indonesian Rupiah)
cnt_ma_rech30 = Number of times main account got recharged in last 30 days
fr_ma_rech30 = Frequency of main account recharged in last 30 days Unsure of given definition sumamnt_ma_rech30 = Total amount of recharge in main account over last 30 days (in Indonesian Rupiah)
medianamnt_ma_rech30 = Median of amount of recharges done in main account over last 30 days at user level (in Indonesian Rupiah)
medianmarechprebal30 = Median of main account balance just before recharge in last 30 days at user level (in Indonesian Rupiah)
cnt_ma_rech90 = Number of times main account got recharged in last 90 days
fr_ma_rech90 = Frequency of main account recharged in last 90 days Unsure of given definition sumamnt_ma_rech90 = Total amount of recharge in main account over last 90 days (in Indonasian Rupiah)
medianamnt_ma_rech90 = Median of amount of recharges done in main account over last 90 days at user level (in Indonasian Rupiah)
medianmarechprebal90 = Median of main account balance just before recharge in last 90 days at user level (in Indonasian Rupiah)

cnt_da_rech30 = Number of times data account got recharged in last 30 days
fr_da_rech30 = Frequency of data account recharged in last 30 days
cnt_da_rech90 = Number of times data account got recharged in last 90 days
fr_da_rech90 = Frequency of data account recharged in last 90 days
cnt_loans30 = Number of loans taken by user in last 30 days
amnt_loans30 = Total amount of loans taken by user in last 30 days
maxamnt_loans30 = maximum amount of loan taken by the user in last 30 days There are only two options: 5 & 10 Rs., for which the user needs to pay back 6 & 12 Rs. respectively medianamnt_loans30 = Median of amounts of loan taken by the user in last 30 days
cnt_loans90 = Number of loans taken by user in last 90 days
amnt_loans90 = Total amount of loans taken by user in last 90 days
maxamnt_loans90 = maximum amount of loan taken by the user in last 90 days
medianamnt_loans90 = Median of amounts of loan taken by the user in last 90 days
payback30 = Average payback time in days over last 30 days
payback90 = Average payback time in days over last 90 days
pcircle = telecom circle
pdate = date

In next step, we dropped columns that contain all unique or mostly same values.

```
micro.drop(['msisdn'],axis=1,inplace=True)
micro.drop(['pcircle'],axis=1,inplace=True)
micro.drop(['pdate'],axis=1,inplace=True)
micro.drop(['Unnamed: 0'],axis=1,inplace=True)
```

We can see that there are no null values in dataset.

```
print(micro.isnull().values.any())
print(micro.isnull().sum())
```

```
False
label                   0
aon                     0
daily_decr30            0
daily_decr90            0
rental30                0
rental90                0
last_rech_date_ma       0
last_rech_date_da       0
last_rech_amt_ma        0
cnt_ma_rech30           0
fr_ma_rech30            0
sumamnt_ma_rech30       0
medianamnt_ma_rech30    0
medianmarechprebal30    0
cnt_ma_rech90           0
fr_ma_rech90            0
sumamnt_ma_rech90       0
medianamnt_ma_rech90    0
medianmarechprebal90    0
cnt_da_rech30           0
fr_da_rech30            0
cnt_da_rech90           0
fr_da_rech90            0
cnt_loans30             0
amnt_loans30            0
maxamnt_loans30         0
medianamnt_loans30      0
cnt_loans90             0
amnt_loans90            0
maxamnt_loans90         0
medianamnt_loans90      0
payback30               0
payback90               0
dtype: int64
```

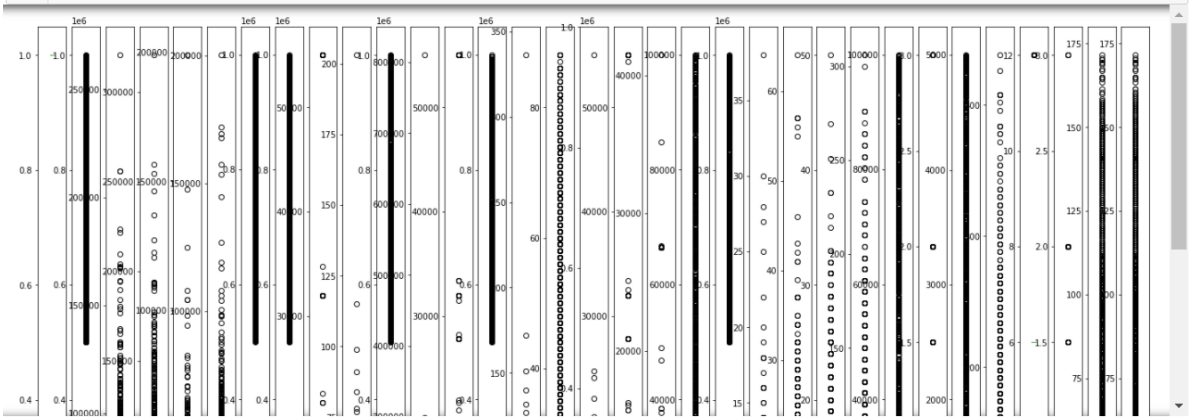
We could see that there is lot of difference between 50% quartile and mean values in many columns which tells us that there is lot of skewness in our data and also there might be outliers in our data.

```
micro.describe()
```

	label	aon	daily_decr30	daily_decr90	rental30	rental90	last_rech_date_ma	last_rech_date_da	last_rech_amt_ma
count	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.000000	209593.00000
mean	0.875177	8112.343445	5381.402289	6082.515068	2692.581910	3483.406534	3755.847800	3712.202921	2064.45279
std	0.330519	75696.082531	9220.623400	10918.812767	4308.586781	5770.461279	53905.892230	53374.833430	2370.78603
min	0.000000	-48.000000	-93.012667	-93.012667	-23737.140000	-24720.580000	-29.000000	-29.000000	0.00000
25%	1.000000	246.000000	42.440000	42.692000	280.420000	300.260000	1.000000	0.000000	770.00000
50%	1.000000	527.000000	1469.175667	1500.000000	1083.570000	1334.000000	3.000000	0.000000	1539.00000
75%	1.000000	982.000000	7244.000000	7802.790000	3356.940000	4201.790000	7.000000	0.000000	2309.00000
max	1.000000	999860.755168	265926.000000	320630.000000	198926.110000	200148.110000	998650.377733	999171.809410	55000.00000

We could see that many outliers in data by looking at boxplot, so we use z score to remove outliers.

```
micro.plot(kind='box',subplots=True,figsize=[24,14])
plt.show()
```



Z_Score:

The z score (also called the standard score) represents the number of standard deviations with which the value of an observation point or data differ than the mean value of what is observed .

This technique consists of subtracting the mean of the column from each value in a column, and then dividing the result by the standard deviation of the column. The formula to achieve this is the following:

$$x_{scaled} = \frac{x - mean}{sd}$$

The result of standardization is that the features will be rescaled so that they'll have the properties of a standard normal distribution, as follows:

- *μ=0*
- *σ=1*

μ is the mean and *σ* is the standard deviation from the mean.

In summary, the z score (also called the **standard score**) represents the number of standard deviations with which the value of an observation point or data differ than the mean value of what is observed.

```
from scipy import stats
z_scores = stats.zscore(micro)
z_scores
```

```
array([[-2.64789583, -0.10357685, -0.25229941, ..., -0.22959366,
         2.9046997 ,  2.39409346],
       [ 0.37765836, -0.09776412,  0.73103667, ..., -0.22959366,
        -0.38562959, -0.41923266],
       [ 0.37765836, -0.10010243, -0.43201111, ..., -0.22959366,
        -0.38562959, -0.41923266],
       ...,
       [ 0.37765836, -0.09378769,  0.70079045, ..., -0.22959366,
         0.06820893, -0.04735622],
       [ 0.37765836, -0.08428915,  0.77075515, ..., -0.22959366,
        -0.38562959,  0.59938541],
       [ 0.37765836, -0.08628398, -0.09674426, ..., -0.22959366,
        -0.38562959, -0.41923266]])
```

```
abs_z_scores = np.abs(z_scores)
micro_new = (abs_z_scores < 3).all(axis=1)
micro_new = micro[micro_new]
```

Feature Scaling

Feature Scaling is a technique to standardize the independent features present in the data in a fixed range. It is performed during the data pre-processing to handle highly varying magnitudes or values or units. If feature scaling is not done, then a machine learning algorithm tends to weigh greater values, higher and consider smaller values as the lower values, regardless of the unit of the values.

Min Max Scaler

In this approach, the data is scaled to a fixed range - usually 0 to 1. The cost of having this bounded range - in contrast to standardization - is that we will end up with smaller standard deviations, which can suppress the effect of outliers.

A Min-Max scaling is typically done via the following equation:

$$X_{sc} = \frac{X - X_{min}}{X_{max} - X_{min}}.$$

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
temp_scaler=scaler.fit_transform(micro_new)
new_micro=pd.DataFrame(temp_scaler,columns=micro_new.columns)
```

Skewness

Skewness is a measure of the symmetry of a distribution. The highest point of a distribution is its mode. The mode marks the response value on the x-axis that occurs with the highest probability. A distribution is skewed if the tail on one side of the mode is fatter or longer than on the other: it is asymmetrical. In an asymmetrical distribution a negative skew indicates that the tail on the left side is longer than on the right side (left-skewed), conversely a positive skew indicates the tail on the right side is longer than on the left (right-skewed). Asymmetric distributions occur when extreme values lead to a distortion of the normal distribution.

```
1 new_micro.skew()
```

```
label                   -2.090315
aon                      0.957902
daily_decr30             1.963747
daily_decr90             2.077637
rental30                 2.194889
rental90                 2.244866
last_rech_date_ma        3.099484
last_rech_date_da       10.384887
last_rech_amt_ma         2.125356
cnt_ma_rech30            1.175157
fr_ma_rech30             2.005139
sumamnt_ma_rech30        1.634226
medianamnt_ma_rech30     2.326312
medianmarechprebal30    10.538891
cnt_ma_rech90            1.321145
fr_ma_rech90             1.985567
sumamnt_ma_rech90        1.707309
medianamnt_ma_rech90     2.373140
medianmarechprebal90     3.692650
cnt_da_rech30           50.760988
fr_da_rech30             0.000000
cnt_da_rech90            6.934340
fr_da_rech90             0.000000
cnt_loans30              1.465414
amnt_loans30             1.441450
maxamnt_loans30         53.470571
medianamnt_loans30       5.355423
cnt_loans90              1.708977
amnt_loans90             1.695156
```

Power Transform:

A power transform will make the probability distribution of a variable more Gaussian. This is often described as removing a skew in the distribution, although more generally is described as stabilizing the variance of the distribution.

```
from sklearn.preprocessing import power_transform
micro_skew = power_transform(micro)
micro_skew = pd.DataFrame(micro_new,columns = micro.columns)
```

Correlation

Correlation explains how one or more variables are related to each other. These variables can be input data features which have been used to forecast our target variable.

Correlation, statistical technique which determines how one variables moves/changes in relation with the other variable. It gives us the idea about the degree of the relationship of the two variables. It's a bi-variate analysis measure which describes the association between different variables. In most of the business it's useful to express one subject in terms of its relationship with others.

```
corr_target = micro.corrwith(micro['label'],axis=0)
corr_target
```

```
label                   1.000000
aon                    -0.003785
daily_decr30            0.168298
daily_decr90            0.166150
rental30                0.058085
rental90                0.075521
last_rech_date_ma       0.003728
last_rech_date_da       0.001711
last_rech_amt_ma        0.131804
cnt_ma_rech30           0.237331
fr_ma_rech30            0.001330
sumamnt_ma_rech30       0.202828
medianamnt_ma_rech30    0.141490
medianmarechprebal30   -0.004829
cnt_ma_rech90           0.236392
fr_ma_rech90            0.084385
sumamnt_ma_rech90       0.205793
medianamnt_ma_rech90    0.120855
medianmarechprebal90    0.039300
cnt_da_rech30           0.003827
fr_da_rech30           -0.000027
cnt_da_rech90           0.002999
fr_da_rech90           -0.005418
cnt_loans30             0.196283
amnt_loans30            0.197272
maxamnt_loans30         0.000248
medianamnt_loans30      0.044589
cnt_loans90             0.004733
amnt_loans90            0.199788
maxamnt_loans90         0.084144
medianamnt_loans90      0.035747
payback30               0.048336
payback90               0.049183
```

Random Oversampling

Imbalanced datasets are those where there is a severe skew in the class distribution, such as 1:100 or 1:1000 examples in the minority class to the majority class.

This bias in the training dataset can influence many machine learning algorithms, leading some to ignore the minority class entirely. This is a problem as it is typically the minority class on which predictions are most important.

One approach to addressing the problem of class imbalance is to randomly resample the training dataset. The two main approaches to randomly resampling an imbalanced dataset are to delete examples from the majority class, called undersampling, and to duplicate examples from the minority class, called oversampling.

Random Oversampling includes selecting random examples from the minority class with replacement and supplementing the training data with multiple copies of this instance, hence it is possible that a single instance may be selected multiple times.

```
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
from collections import Counter
```

```
print(Counter(y))
```

```
Counter({1: 139065, 0: 22400})
```

```
# instantiating the random oversampler
ros = RandomOverSampler()
# resampling X, y
x_ros, y_ros = ros.fit_resample(x, y)

# new class distribution
print(Counter(y_ros))
```

```
Counter({0: 139065, 1: 139065})
```

Model Building

A machine learning model is built by learning and generalizing from training data, then applying that acquired knowledge to new data it has never seen before to make predictions and fulfil its purpose.

```
x_train,x_test,y_train,y_test = train_test_split(x_ros,y_ros,test_size=0.25, random_state=15)
```

Logistic regression

Logistic regression is a statistical analysis method used to predict a data value based on prior observations of a data set. Logistic regression has become an important tool in the discipline of machine learning. The approach allows an 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 activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load (ETL) process in order to stage the information for analysis.

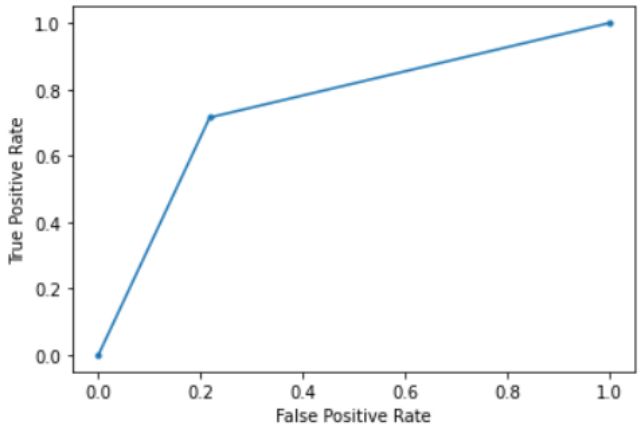
A logistic regression model predicts a dependent data variable by analysing the relationship between one or more existing independent variables. For example, a logistic regression could be used to predict whether a political candidate will win or lose an election or whether a high school student will 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 about earlier outcomes involving the same input criteria, it then scores new cases on their probability of falling into a particular outcome category.

```python
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
lr_predict =lr.predict(x_test)
lr_auc = roc_auc_score(y_test, lr_predict)
print('Logistic: ROC AUC=%.3f' % (lr_auc))
lr_fpr, lr_tpr, _= roc_curve(y_test, lr_predict)
lr_accuracy = accuracy_score(y_test, lr_predict)
print(lr_accuracy)
```

```
Logistic: ROC AUC=0.749
0.7487955359325787
```

```python
plt.plot(lr_fpr, lr_tpr, marker='.', label='Logistic')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```



Gaussian Naive Bayes

Gaussian naive bayes is a variant of 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.

Bayes Theorem

Bayes Theorem can be used to calculate conditional probability. Being a powerful tool in the study of probability, it is also applied in Machine Learning.

The Formula For Bayes' Theorem Is

$$P(A|B) = \frac{P(A \bigcap B)}{P(B)} = \frac{P(A) \cdot P(B|A)}{P(B)}$$

where:

$P(A) =$ The probability of A occurring

$P(B) =$ The probability of B occurring

$P(A|B) =$ The probability of A given B

$P(B|A) =$ The probability of B given A

$P\left(A \bigcap B\right)) =$ The probability of both A and B occurring

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 apply to many real-life situations.

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-

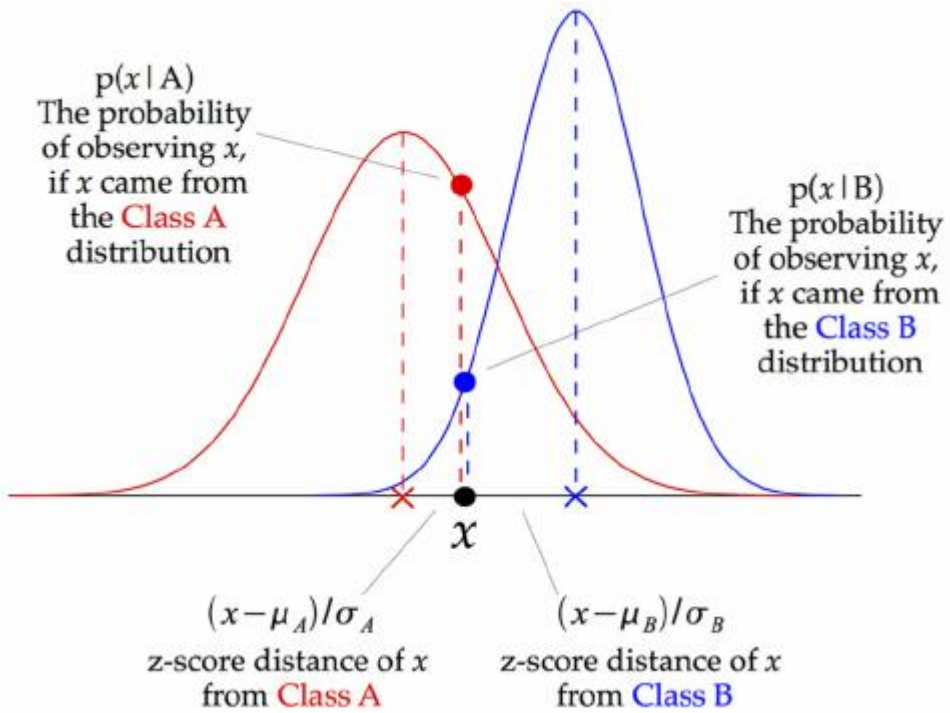
$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

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.



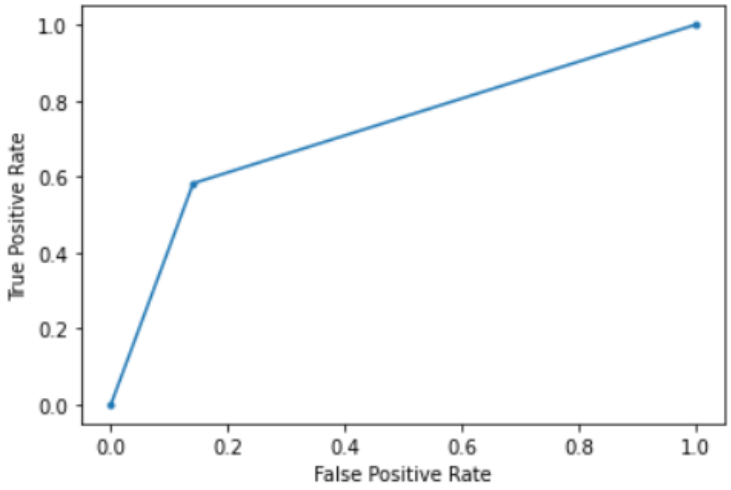
The above illustration indicates how a Gaussian Naive Bayes (GNB) classifier works. At every data point, the z-score distance between that point and each class-mean is calculated, namely the distance from the class mean divided by the standard deviation of that class.

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

```python
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(x_train, y_train)
gnb_predict = gnb.predict(x_test)
gnb_auc = roc_auc_score(y_test, lr_predict)
print('GaussianNB: ROC AUC=%.3f' % (gnb_auc))
gnb_accuracy_score = accuracy_score(y_test, gnb_predict)
gnb_fpr, gnb_tpr, _ = roc_curve(y_test, gnb_predict)
print(gnb_accuracy_score)
```

```
GaussianNB: ROC AUC=0.749
0.7208951145499259
```

```python
plt.plot(gnb_fpr, gnb_tpr, marker='.', label='GaussianNB')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```



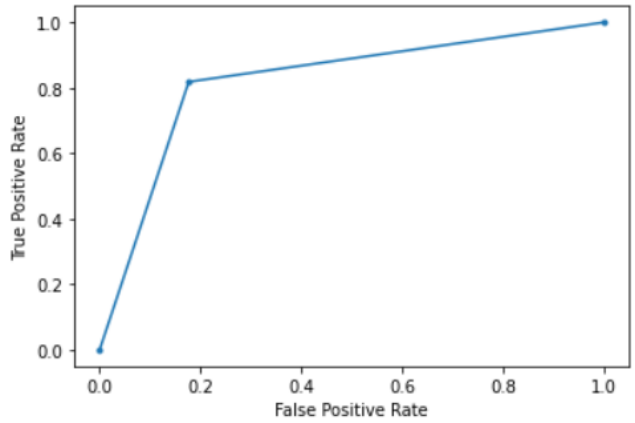
Decision Tree Algorithm

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving **regression and classification problems** too.
The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by **learning simple decision rules** inferred from prior data (training data).
In Decision Trees, for predicting a class label for a record we start from the **root** of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

```python
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(max_depth=10)
dt.fit(x_train,y_train)
dt_predict = dt.predict(x_test)
dt_auc = roc_auc_score(y_test, lr_predict)
print('Decision Tree: ROC AUC=%.3f' % (dt_auc))
dt_accuracy_score = accuracy_score(y_test,dt_predict)
print(dt_accuracy_score)
dt_fpr, dt_tpr, _ = roc_curve(y_test, dt_predict)
```

```
Decision Tree: ROC AUC=0.749
0.8208045100887349
```

```python
plt.plot(dt_fpr, dt_tpr, marker='.', label='Decision Tree Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```



Random Forest

Random Forest is an example of ensemble learning, in which we combine multiple machine learning algorithms to obtain better predictive performance.

Why the name “Random”?

Two key concepts that give it the name random:

1. A random sampling of training data set when building trees.
2. Random subsets of features considered when splitting nodes.

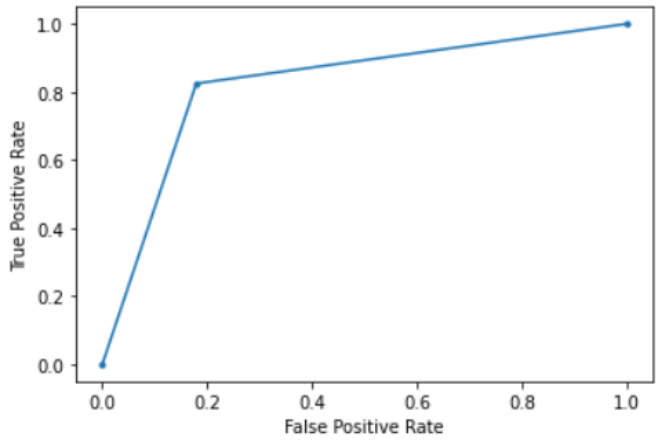
A technique known as bagging is used to create an ensemble of trees where multiple training sets are generated with replacement.

In the bagging technique, a data set is divided into **N** samples using randomized sampling. Then, using a single learning algorithm a model is built on all samples. Later, the resultant predictions are combined using voting or averaging in parallel.

```python
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(max_depth=10)
rf.fit(x_train, y_train)
rf_predict = rf.predict(x_test)
rf_accuracy_score = accuracy_score(y_test,rf_predict)
rf_auc = roc_auc_score(y_test, rf_predict)
print('Random Forest: ROC AUC=%.3f' % (rf_auc))
rf_accuracy_score = accuracy_score(y_test,rf_predict)
print(rf_accuracy_score)
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_predict)
```

```
Random Forest: ROC AUC=0.823
0.8228035608991414
```

```python
plt.plot(rf_fpr, rf_tpr, marker='.', label='Random Forest Classifier')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
```



Cross Validation

The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

```
from sklearn.model_selection import cross_val_score
scr=cross_val_score(lr, x, y, cv=6)
print('Cross validation score of Logistic Regression : ',scr.mean())
```

```
Cross validation score of Logistic Regression :  0.8636670593387823
```

```
scr=cross_val_score(gnb, x, y, cv=6)
print('Cross validation score of Naive Bayes : ',scr.mean())
```

```
Cross validation score of Naive Bayes :  0.6460471378320553
```

```
scr=cross_val_score(dt, x, y, cv=6)
print('Cross validation score of Decision Tree : ',scr.mean())
```

```
Cross validation score of Decision Tree :  0.9018672873472027
```

```
scr=cross_val_score(dt, x, y, cv=6)
print('Cross validation score of Decision Tree : ',scr.mean())
```

```
Cross validation score of Decision Tree :  0.9019044445715777
```

From above cross validation, we can observe that Random Forest Classifier is having least difference between accuracy score and cross validation.

So, Random Forest Classifier with accuracy score of 82.28% is the best model

```
import joblib
joblib.dump(rf,'Micro_Credit.pkl')
```

```
['Micro_Credit.pkl']
```

Deployment

Deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data.

```
import joblib
joblib.dump(lr,'Customer_Retention.pkl')
```

```
['Customer_Retention.pkl']
```

```
import joblib
joblib.dump(rf,'Customer_Retention.pkl')
```

```
['Customer_Retention.pkl']
```

Conclusion:

As part of study, we now understood that how microfinance services providers provide loan at remote locations, and what are the measures they follow to recover loan and also how they help people in remote locations by providing loans like Group Loans, Agricultural Loans, Individual Business Loans and so on and how far they are going to effect the future generation small scale business needs.