LOAN PAID-OFF STATUS

-BY MACHINE LEARNING USING PYTHON

➢ Created by

SUMANTA GARAI
DEPARTMENT OF INFORMATION TECHNOLOGY
KALYANI GOVERNMENT ENGINEERING COLLEGE

ANIK DAKUA
DEPARTMENT OF INFORMATION TECHNOLOGY
KALYANI GOVERNMENT ENGINEERING COLLEGE

SUVAJIT ADAK
DEPARTMENT OF INFORMATION TECHNOLOGY
KALYANI GOVERNMENT ENGINEERING COLLEGE

SUJOY KUMAR HALDAR
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KALYANI GOVERNMENT ENGINEERING COLLEGE

DEVSMITA MUKHERJEE
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KALYANI GOVERNMENT ENGINEERING COLLEGE

Table Of Contents

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty Mr. Titas Roy Chowdhury for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

SUMANTA GARAI
ROLL- 10200217021
REG.NO.-
181020120030
OF 2018-19

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty Mr. Titas Roy Chowdhury for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

ANIK DAKUA
ROLL- 10200217068
REG.NO.-
171020110064 of
2017-18

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty Mr. Titas Roy Chowdhury for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

SUVAJIT ADAK
ROLL- 10200217021
REG.NO.-
171020110111 of
2017-18

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty Mr. Titas Roy Chowdhury for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

SUJOY KUMAR HALDAR
ROLL- 10200117022
REG.NO.-
1710200110057 of
2017-18

Acknowledgement

I take this opportunity to express my profound gratitude and deep regards to my faculty Mr. Titas Roy Chowdhury for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and guidance given time to time shall carry me a long way in the journey of life on which I am about to embark.

I am obliged to my project team members for the valuable information provided by them in their respective fields. I am grateful for their cooperation during the period of my assignment.

DEVSMITA MUKHERJEE
ROLL- 10200118012
REG.NO.- 181020120003 of 2018-19

Project Objective:

- **Describing Problem**

Simply put, predicting loan paid off status is the ability to predict whether or not a person who has taken a loan and is repaying it back in instalments will be able to keep on paying his instalments until he has paid back the sum of money with interest in due time.

In lay man's language it is the ability to predict whether a person who has taken a loan will be able to pay back his loan or not, depending on various parameters. A few of these parameters include current loan amount, credit score, monthly debt, annual income of the person.

Various banks are trying to predict this result for a variety of reasons, as predicting this will help the banks to either cancel out bad loans or adjust their interest rates accordingly or adopt some various other techniques in order to prevent the people from defaulting on their loan payments or assign equivalent collateral to cut their losses. The ability to predict if a person will fully pay his loan is of global importance and will be greatly beneficial to all the banks and investment firms throughout the world.

➢ Causes of loan default:

There are a multitude of issues that can lead a person to default on his loan. Some of the most common reasons include credit score, a high instalment amount, low annual income etc. There are various reasons for an individual to default on his/her loan they can be personal or financial, here in this project we aim to predict this based on data that can be recorded which encompasses only the financial aspects of this problem. We will be using various financial data to predict the causes of instalment default and see if we can identify any pattern between the various data which will help us in the prediction. A lower credit score and a lower annual income of a individual also contribute towards a person who is not able to pay back his loan. Moreover a individual's liability can also be affected by a sudden change in his financial assets or a spike in draining of his financial resources.

Data Description:

- **Software Used:** Python using Spyder/Anaconda
- **Column Description:**

Column Name	Categorical/ Continuous	Data Types	Null Value
Loan ID	Non categorical	Object	0
Customer ID	Non categorical	Object	0
Loan Status	Categorical	Object	0
Current Loan Amount	Continuous	Float	0
Term	Categorical	Object	0
Credit Score	Continuous	Float	1
Annual Income	Continuous	Float	1
Years in Current Job	Non continuous	Object	1
Home Ownership	Categorical	Object	0
Purpose	Categorical	Object	1
Monthly Debt	Continuous	Float	1
Year of Credit History	Continuous	Float	1
Months Since		Float	1

Number of Open Account	Continuous	Float	1
Number of Credit Problem	Continuous	Float	1
Current Credit Balance	Continuous	Float	1
Maximum Open Credit	Continuous	Float	1
Bankruptcies	Continuous	Float	1
Tax Lines	Continuous	Float	1

Procedure

- We analized the data frame.
- Dropped the columns which have high multi co-linearity and low contribution towards ‘label’.
- Applied different classification models and noted down the scores for each models.
- Selected the model with the best scores.

Linear Regression Model

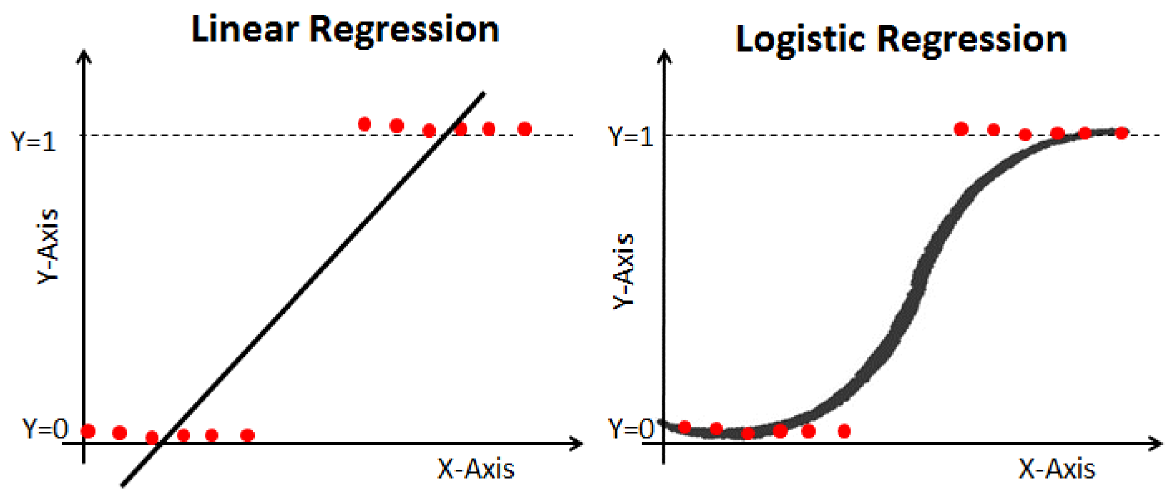
In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression. This term is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted, rather than a single scalar variable.

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Linear regression was the first type of regression analysis to be studied rigorously, and to be used extensively in practical applications. This is because models which depend linearly on their unknown parameters are easier to fit than models which are non-linearly related to their parameters and because the statistical properties of the resulting estimators are easier to determine.

Linear Regression Vs. Logistic Regression

Linear regression gives you a continuous output, but logistic regression provides a constant output. An example of the continuous output is house price and stock price. Example's of the discrete output is predicting whether a patient has cancer or not, predicting whether the customer will churn. Linear regression is estimated using Ordinary Least Squares (OLS) while logistic regression is estimated using Maximum Likelihood Estimation (MLE) approach.

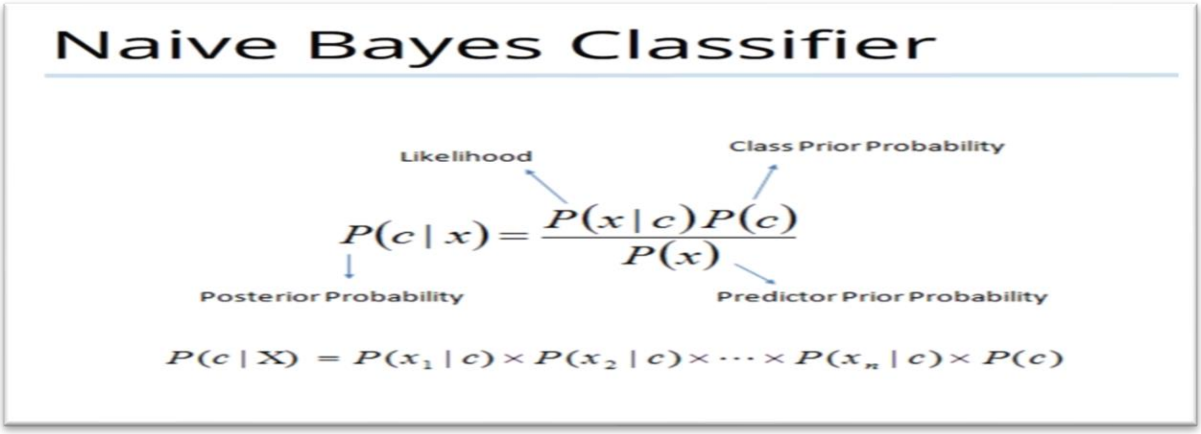


Naive Bayes Model (NBM)

In machine learning, naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models.

Naïve Bayes has been studied extensively since the 1960s. It was introduced (though not under that name) into the text retrieval community in the early 1960s, and remains a popular (baseline) method for text categorization, the problem of judging documents as belonging to one category or the other (document categorization)(such as spam or legitimate, sports or politics, etc.) with word frequencies as the features. With appropriate pre-processing, it is competitive in this domain with more advanced methods including support vector machines. It also finds application in automatic medical diagnosis.

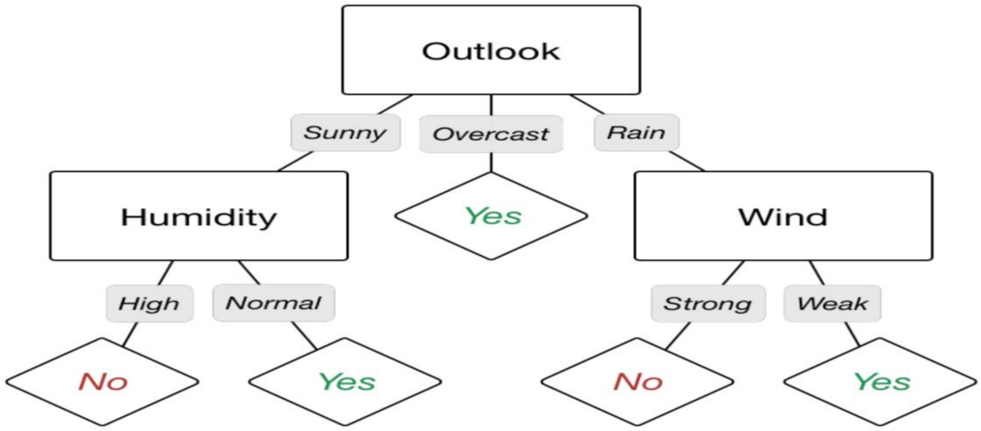
Naïve Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.



Decision Tree

Decision tree learning is one of the predictive modeling approaches used in statistics, data mining and machine learning. It uses a decision tree (as a predictive model) to go from observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves). Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. In data mining, a decision tree describes data (but the resulting classification tree can be an input for decision making). This page deals with decision trees in data mining.



How does the Decision Tree algorithm work?

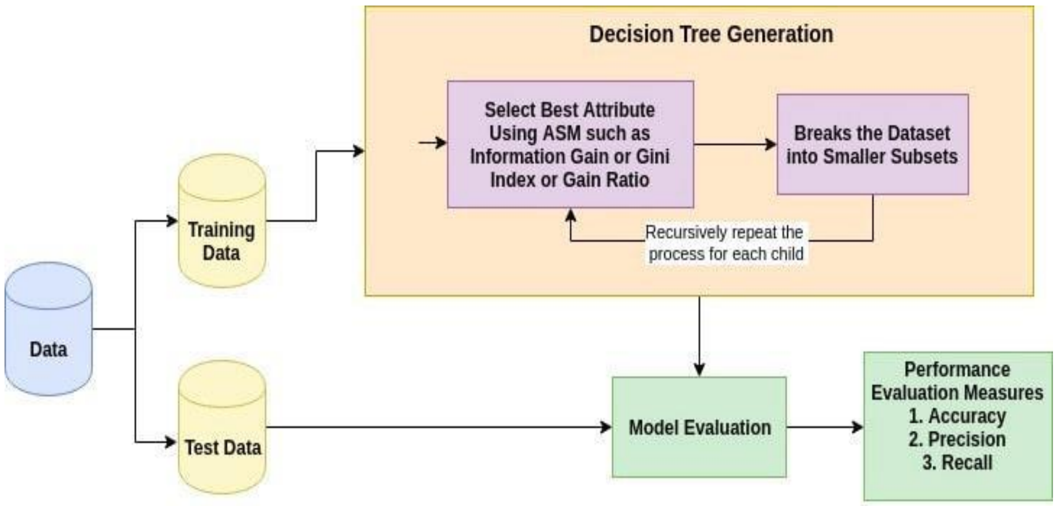
The basic idea behind any decision tree algorithm is as follows:

1. Select the best attribute using Attribute Selection Measures(ASM) to split the records.

2. Make that attribute a decision node and breaks the dataset into smaller subsets.

3. Starts tree building by repeating this process recursively for each child until one of the condition will match:

   - All the tuples belong to the same attribute value.
   - There are no more remaining attributes.
   - There are no more instances.



K-Nearest Neighbors

In pattern recognition, the *k*-nearest neighbors algorithm (*k*-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the *k* closest training examples in the feature space. The output depends on whether *k*-NN is used for classification or regression:

- In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive integer, typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

- In *k-NN regression*, the output is the property value for the object. This value is the average of the values of *k* nearest neighbors.

k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until function evaluation.

Both for classification and regression, a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of $1/d$, where d is the distance to the neighbor.[2]

The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

Algorithm:

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the k training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance). In the context of gene expression microarray data, for example, *k*-NN has been employed with correlation coefficients, such as Pearson and Spearman, as a metric.[3] Often, the classification accuracy of *k*-NN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighbourhood components analysis.

A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the *k* nearest neighbors due to their large number.[4] One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its *k* nearest neighbors. The class (or value, in regression problems) of each of the *k* nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example, in a self-organizing map (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. *K*-NN can then be applied to the SOM.

KNN Regression :

In K-NN regression, the k-NN algorithm is used for estimating continuous variables. One such algorithm uses a weighted average of the k nearest neighbors, weighted by the inverse of their distance. This algorithm works as follows:

1. Compute the Euclidean or Mahalanobis distancefrom the query example to the labeled examples.

2. Order the labeled examples by increasing distance.

3. Find a heuristically optimal number k of nearest neighbors, based on RMSE. This is done using cross validation.

4. Calculate an inverse distance weighted average with the k-nearest multivariate neighbors.

Random Forest Model:

Let's understand the algorithm in layman's terms. Suppose you want to go on a trip and you would like to travel to a place which you will enjoy. So what do you do to find a place that you will like? You can search online, read reviews on travel blogs and portals, or you can also ask your friends. Let's suppose you have decided to ask your friends, and talked with them about their past travel experience to various places. You will get some recommendations from every friend. Now you have to make a list of those recommended places. Then, you ask them to vote (or select one best place for the trip) from the list of recommended places you made. The place with the highest number of votes will be your final choice for the trip. In the above decision process, there are two parts. First, asking your friends about their individual travel experience and getting one recommendation out of multiple places they have visited. This part is like using the decision tree algorithm. Here, each friend makes a selection of the places he or she has visited so far. The second part, after collecting all the recommendations, is the voting procedure for selecting the best place in the list of recommendations. This whole process of getting recommendations from friends and voting on them to find the best place is known as the random forests algorithm. It technically is an ensemble method (based on the divide-and-conquer approach) of decision trees generated on a randomly split dataset. This collection of decision tree classifiers is also known as the forest. The individual decision trees are generated using an attribute selection indicator such as information gain, gain ratio, and Gini index for each attribute. Each tree depends on an independent random sample. In a classification problem, each tree votes and the most popular class is chosen as the final result. In the case of regression, the average of all the tree outputs is considered as the final result. It is simpler and more powerful compared to the other non-linear classification algorithms.

Random Forests vs Decision Trees

- Random forests is a set of multiple decision trees.
- Deep decision trees may suffer from overfitting, but random forests prevents overfitting by creating trees on random subsets.
- Decision trees are computationally faster.
- Random forests is difficult to interpret, while a decision tree is easily interpretable and can be converted to rules.

Result From All Model

- **Linear Regression Model**

Initial Data

Model Name	Accuracy	Precision	Recall	AUC
LR-Train	0.818735	0.973275	0.194005	0.596236
LR-Test	0.820037	0.965922	0.194189	0.596120

Final Data

Model Name	Accuracy	Precision	Recall	AUC
LR Train	0.821534	0.991140	0.203021	0.601249
LR Test	0.822809	0.985092	0.202929	0.601028

- **Naïve Bayes Model**

Initial Data

Model Name	Accuracy	Precision	Recall	AUC

Naïve Bayes Train	0.346456	0.253987	0.993795	0.577007
Naïve Bayes Test	0.342454	0.251075	0.993149	0.575260

Final Data

Model Name	Accuracy	Precision	Recall	AUC
Naïve Bayes Train	0.361262	0.256697	0.980623	0.581848
Naïve Bayes Test	0.356892	0.253457	0.978738	0.579377

➨ Decesion Tree Model

Initial Data

Model Name	Accuracy	Precision	Recall	AUC
D T - Train	1.000000	1.000000	1.000000	1.000000
DT- Test	0.754748	0.444553	.430900	0.638881

Final Data

Model Name	Accuracy	Precision	Recall	AUC
D T- Train	1.000000	1.000000	1.000000	1.000000
Decession Tree Test	0.757363	0.450549	0.435861	0.642336

➨ K-Nearest Neighbour Model

Initial Data

Model Name	Accuracy	Precision	Recall	AUC
KNN Train	**0.809069**	**0.654863**	**0.307458**	**0.630420**
KNN Test	**0.741512**	**0.319018**	**0.147413**	**0.528955**

Final Data

Model Name	Accuracy	Precision	Recall	AUC
KNN Train	**0.84397**	**0.805381**	**0.397787**	**0.685066**
KNN Test	**0.798640**	**0.593659**	**0.287503**	**0.615765**

Final Accepted Model

Naïve Bayes Model

Initial Data

Model Name	**Accuracy**	**Precision**	**Recall**	**AUC**
Naïve Bayes-Train	0.346456	0.253987	0.993795	0.577007
Naïve Bayes -Test	0.342454	0.251075	0.993149	0.575260

Final Data

Model Name	**Accuracy**	**Precision**	**Recall**	**AUC**
Naïve Bayes -Train	0.361262	0.256697	0.980623	0.581848

Naïve Bayes -Test	0.356892	0.253457	0.978738	0.579377

Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import linear_model
from sklearn import model_selection
from sklearn.model_selection import GridSearchCV
from sklearn import preprocessing
from sklearn import metrics
from sklearn import feature_selection
from sklearn import naive_bayes
from sklearn import neighbors
from sklearn import tree
```

```
from sklearn import ensemble
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble.partial_dependence import
plot_partial_dependence
from sklearn.ensemble.partial_dependence import
partial_dependence
from sklearn.metrics import log_loss
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix,accuracy_score
from sklearn.feature_selection import chi2

def modelstats1(Xtrain,Xtest,ytrain,ytest):
    stats=[]
    modelnames=["LR","DecisionTree","KNN","NB"]
    models=list()
    models.append(linear_model.LogisticRegression())
    models.append(tree.DecisionTreeClassifier())
    models.append(neighbors.KNeighborsClassifier())
    models.append(naive_bayes.GaussianNB())
    for name,model in zip(modelnames,models):
        if name=="KNN":
            k=[l for l in range(5,17,2)]
            grid={"n_neighbors":k}
            grid_obj =
model_selection.GridSearchCV(estimator=model,param_grid=gri
d,scoring="f1")
```

```
            grid_fit =grid_obj.fit(Xtrain,ytrain)
            model = grid_fit.best_estimator_
            model.fit(Xtrain,ytrain)

name=name+"("+str(grid_fit.best_params_["n_neighbors"])+")"
            print(grid_fit.best_params_)
        else:
            model.fit(Xtrain,ytrain)
        trainprediction=model.predict(Xtrain)
        testprediction=model.predict(Xtest)
        scores=list()
        scores.append(name+"-train")

scores.append(metrics.accuracy_score(ytrain,trainprediction))

scores.append(metrics.precision_score(ytrain,trainprediction))
        scores.append(metrics.recall_score(ytrain,trainprediction))

scores.append(metrics.roc_auc_score(ytrain,trainprediction))
        stats.append(scores)
        scores=list()
        scores.append(name+"-test")
        scores.append(metrics.accuracy_score(ytest,testprediction))
        scores.append(metrics.precision_score(ytest,testprediction))
        scores.append(metrics.recall_score(ytest,testprediction))
        scores.append(metrics.roc_auc_score(ytest,testprediction))
        stats.append(scores)
```

```
colnames=["MODELNAME","ACCURACY","PRECISION","RECALL
","AUC"]
   return pd.DataFrame(stats,columns=colnames)

df=pd.read_csv("D:\\vt\\credit_train.csv")
dforig=pd.read_csv("D:\\vt\\credit_train.csv")

df
df.info()
df.shape
df.isnull().sum()
df.isnull().sum()/df.shape[0]

df.apply(lambda x: sum(x.isnull()))
df

df["Years in current job"].isnull().sum()
df_new = df[pd.notnull(df['Years in current job'])]
df_new.apply(lambda x: sum(x.isnull()))

df_new['Credit Score'].fillna(value=df_new['Credit
Score'].mean(),inplace=True)
df1=df_new.round(decimals=0)

df1['Annual Income'].fillna(value=df1['Annual
Income'].mean(),inplace=True)
df2=df1.round(decimals=2)
```

```
df2['Months since last delinquent'].fillna(value=df2['Months since last delinquent'].mean(),inplace=True)
df3=df2.round(decimals=0)

df3.dropna(inplace=True)
df3.apply(lambda x: sum(x.isnull()))
df3.shape

hist = df3['Current Loan Amount'].hist(color='red')
plt.title("Histogram for Current Loan Amount")
plt.show()

sns.distplot(df3['Monthly Debt'], kde=True, color='blue', bins=10)

sns.distplot(df3['Annual Income'], kde=False, color='blue', bins=10)

sns.distplot(df3['Years of Credit History'], kde=True, color='green', bins=10)

df3['Home Ownership'] = df3['Home Ownership'].map({'Home Mortgage':0,'Own Home':1,'Rent':2,'HaveMortgage':3})
df3['Term'] = df3['Term'].map({'Short Term':0,'Long Term':1})
df3['Loan Status'] = df3['Loan Status'].map({'Fully Paid':0,'Charged Off':1})
df3.shape
```

```
sns.boxplot(y='Current Loan Amount',data=df3)


q=df3['Current Loan Amount'].quantile(0.99)
df4=df3[df3['Current Loan Amount']<q]
df4.shape
print(df4.columns)


df3.drop(["Loan ID","Customer ID"],axis=1,inplace=True)



d1={ label:i  for i,label in enumerate(df3["Years in current job"].unique())} # dictionary
df3["Years in current job"].replace(d1,inplace=True)


d2={ label:i  for i,label in enumerate(df3["Purpose"].unique())} # dictionary
df3["Purpose"].replace(d2,inplace=True)


X=df3.drop("Loan Status",axis=1)
y=df3["Loan Status"]
scaler=StandardScaler()
scaled_df=scaler.fit_transform(df3[["Current Loan Amount","Credit Score","Annual Income","Monthly Debt","Years of Credit History","Number of Open Accounts",
            "Current Credit Balance","Maximum Open Credit","Months since last delinquent"]])
```

```
scaled_df = pd.DataFrame(scaled_df, columns=["Current Loan Amount","Credit Score","Annual Income","Monthly Debt","Years of Credit History","Number of Open Accounts",
        "Current Credit Balance","Maximum Open Credit","Months since last delinquent"])

notscaled=df3.drop(["Current Loan Amount","Credit Score","Annual Income","Monthly Debt","Years of Credit History","Number of Open Accounts",
        "Current Credit Balance","Maximum Open Credit","Months since last delinquent"],axis=1)
allcol=notscaled.copy()
for col in scaled_df:
   allcol[col]=scaled_df[col].values

X=allcol.drop("Loan Status",axis=1)
y=allcol["Loan Status"]

Xtrain,Xtest,ytrain,ytest = train_test_split(X,y,test_size=0.2, random_state=66)
modelstats1(Xtrain,Xtest,ytrain,ytest)

df3.drop("Number of Credit Problems",axis=1,inplace=True)

# No improvement
X=df3.drop(["Loan Status","Bankruptcies"],axis=1)
y=df3["Loan Status"]

chi_sq=chi2(X,y)
```

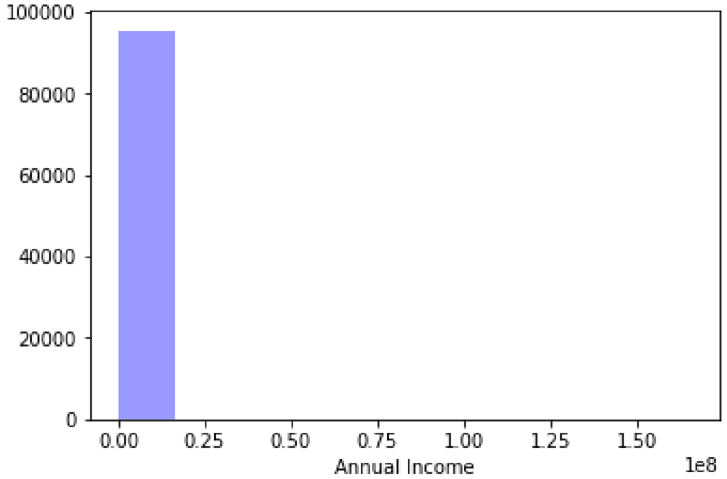
```
chi_sq

p_values=pd.Series(chi_sq[1],index=X.columns)
p_values.sort_values(ascending=False,inplace=True)
p_values[:].plot

df3.drop("Purpose",axis=1,inplace=True)
modelstats1(Xtrain,Xtest,ytrain,ytest)

from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100, bootstrap = True,
                    max_features = 'sqrt')


X=df3.drop("Loan Status",axis=1)
y=df3["Loan Status"]


Xtrain,Xtest,ytrain,ytest = train_test_split(X,y,test_size=0.2,
random_state=67)
modelstats1(Xtrain,Xtest,ytrain,ytest)

clf.fit(Xtrain,ytrain)

ypred=clf.predict(Xtest)
```

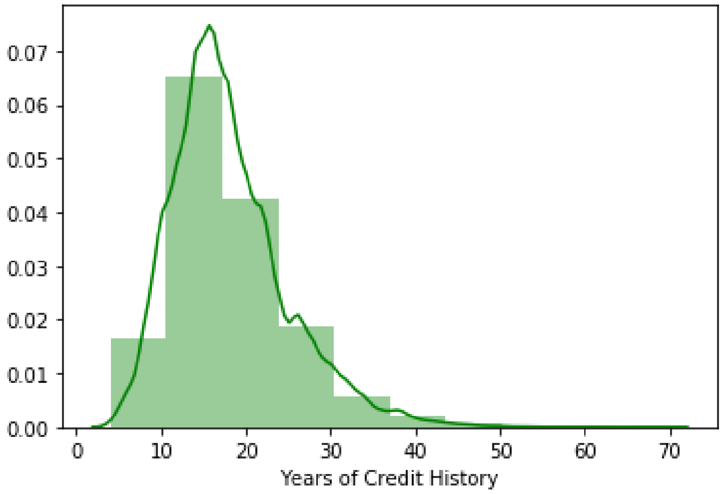
```
from sklearn import metrics
print("Recall:",metrics.recall_score(ytest, ypred))



X=df3.drop("Loan Status",axis=1)
y=df3["Loan Status"]

chi_sq=chi2(X,y)
chi_sq

p_values=pd.Series(chi_sq[1],index=X.columns)
p_values.sort_values(ascending=False,inplace=True)

df3.drop("Bankruptcies",axis=1,inplace=True)

df3.drop("Monthly Debt",axis=1,inplace=True)

corr=df3.corr()
Xtrain,Xtest,ytrain,ytest = train_test_split(X,y,test_size=0.2,
random_state=68)
modelstats1(Xtrain,Xtest,ytrain,ytest)
```

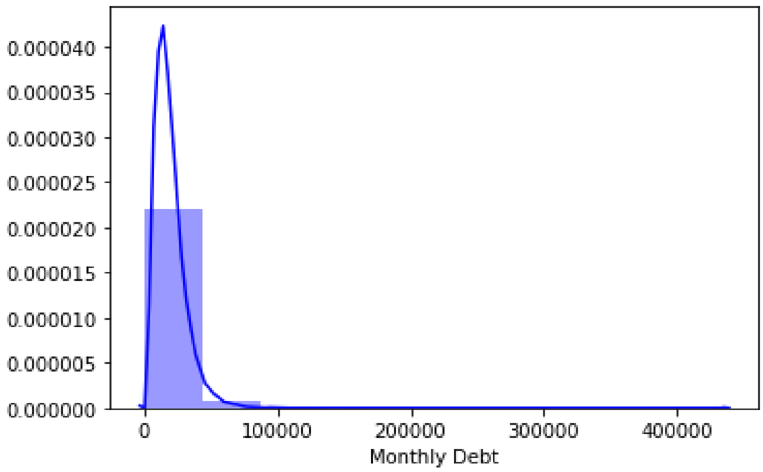
Conclusion and Future Works:

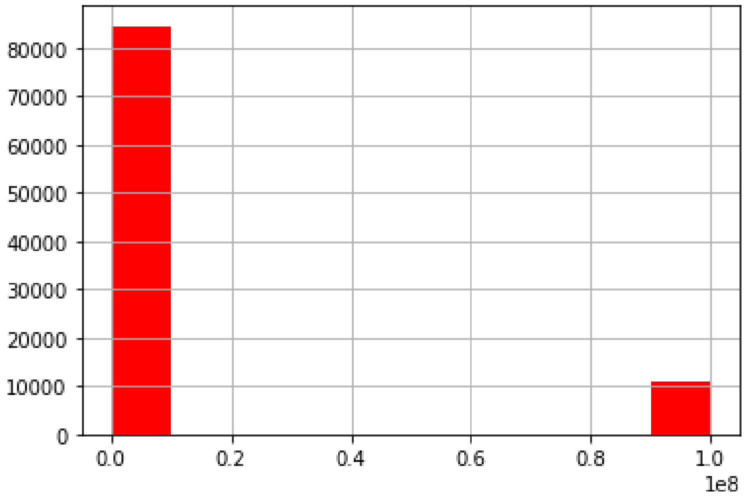
- In conclusion, the Naïve Bayes method demonstrated better recall score and accuracy for predicting the instalment defaulters. The results obtained after optimization of parameters for train model has proven that by using recursive feature elimination with cross validation with model specific estimators gives a better result. The result obtained has addressed the impact of class imbalance problem which makes it difficult for classifier to make prediction.
- The training set that consists of 100514 instances has only 2.1262% of defaultee while the remaining 97.8738% is people who have paid their loans. Thus, this requires further investigation theoretically and experimentally by considering several pertinent issues. One of the approaches to improve the model is to divide the dataset into training and testing models during data discretization process; also, the number of positives to negatives. Lastly, this technique can also be improved by adopting different machine learning algorithm such as support vector machine, decision tree as well as the Bayesian network that allows learning of non-linear data sample.

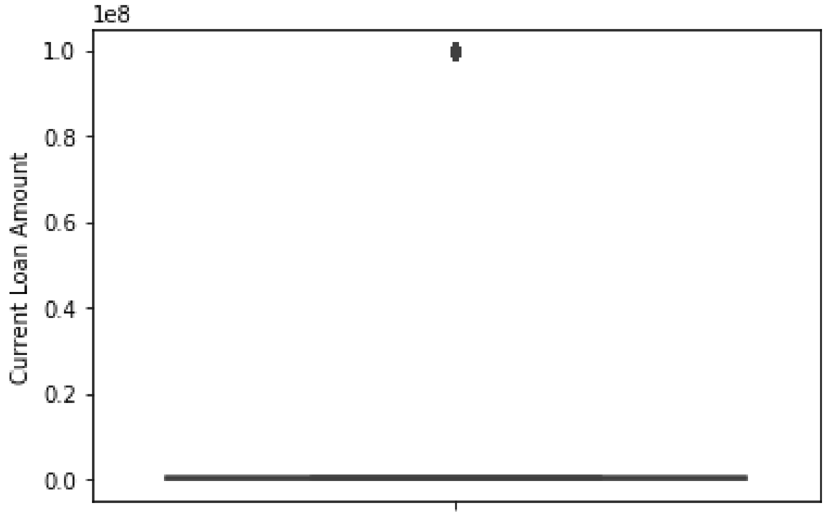
Related Graphs :








80000
70000
60000
50000
40000
30000
20000
10000
0
0.0
0.2
0.4
0.6
0.8
1.0
1e8


1e8
1.0
0.8
0.6
0.4
0.2
0.0
Current Loan Amount

Certificate

This is to certify that Mr., SUMANTA GARAI, KALYANI GOVERNMENT ENGINEERING COLLEGE, registration number: 181020120030 of 2018-19, has successfully completed a project on 'Loan Paid-Off Status' using "Machine learning using Python" under the guidance of Mr. Titas Roy Chowdhury.

(**Mr. Titas Roy Chowdhury**)

Globsyn Finishing School

Certificate

This is to certify that Mr., ANIK DAKUA , KALYANI GOVERNMENT ENGINEERING COLLEGE, registration number: 171020110064 of 2017-18,has successfully completed a project on ‘Loan Paid-Off Status’ using “Machine learning using Python” under the guidance of Mr. Titas Roy Chowdhury.

(**Mr. Titas Roy Chowdhury**)

Globsyn Finishing School

Certificate

This is to certify that Mr., SUVAJIT ADAK , KALYANI GOVERNMENT ENGINEERING COLLEGE, registration number: 171020110111 of 2017-18,has successfully completed a project on 'Loan Paid-Off Status' using "Machine learning using Python" under the guidance of Mr. Titas Roy Chowdhury.

____

(**Mr. Titas Roy Chowdhury**)

Globsyn Finishing School

Certificate

This is to certify that Mr., SUJOY KUMAR HALDER , KALYANI GOVERNMENT ENGINEERING COLLEGE, registration number: 171020011 0057 of 2017-18,has successfully completed a project on 'Loan Paid-Off Status' using "Machine learning using Python" under the guidance of Mr. Titas Roy Chowdhury.

____

(**Mr. Titas Roy Chowdhury**)

Globsyn Finishing School

Certificate

This is to certify that Mr., DEVSMITA MUKHERJEE , KALYANI GOVERNMENT ENGINEERING COLLEGE, registration number: 181020120003 of 2018-19,has successfully completed a project on ‘Loan Paid-Off Status’ using “Machine learning using Python” under the guidance of Mr. Titas Roy Chowdhury.

____

(**Mr. Titas Roy Chowdhury**)

Globsyn Finishing School

THANK YOU