Project Report
Bank Loan Default

Submitted By -
Nitisha Yadav

Contents

A problem statement in Data Science can be solved by following the below steps:

1. Define Problem Statement/ Business Requirement
2. Data Collection
3. Data Pre-processing
4. Data Exploration & Analysis
5. Data Modelling
6. Model Evaluation

1. **Problem Statement:** The given problem is about a bank statement records of loan defaults in which each applicant is rated as default or not default on the basis of some information. The given problem statement is a supervised binary classification problem as the target variable is categorical which has only two values 0 and 1 for not default and default respectively.

2. **Data Collection:** To get the best results it is very important to understand the data. Here data consists of 850 observations and 9 variables. The different variables of the data are:
   - age - age of each customer
   - education – education categories
   - employment – employment status
   - address – geographic area being
   - income – gross income of each customer
   - creddebt-debt-to-credit ratio is a measurement of how much you owe to your creditors as a percentage of your available credit
   - debtinc – individual debt payment to his/her gross income
   - Othdebt – other debts
   - Default-contains values 0 and 1

The data consists of 8 independent variables and 1 dependent variable

Independent Variables:

1. Age
2. Education
3. Employment
4. Address
5. Income
6. Creddebt
7. Debtinc
8. Othdebt

Dependent Variable:

1. Default

	age	ed	employ	address	income	debtinc	creddebt	othdebt	default
0	41	3	17	12	176	9.3	11.359392	5.008608	1.0
1	27	1	10	6	31	17.3	1.362202	4.000798	0.0
2	40	1	15	14	55	5.5	0.856075	2.168925	0.0
3	41	1	15	14	120	2.9	2.658720	0.821280	0.0
4	24	2	2	0	28	17.3	1.787436	3.056564	1.0
5	41	2	5	5	25	10.2	0.392700	2.157300	0.0
6	39	1	20	9	67	30.6	3.833874	16.668126	0.0
7	43	1	12	11	38	3.6	0.128592	1.239408	0.0
8	24	1	3	4	19	24.4	1.358348	3.277652	1.0
9	36	1	0	13	25	19.7	2.777700	2.147300	0.0
10	27	1	0	1	16	1.7	0.182512	0.089488	0.0
11	25	1	4	0	23	5.2	0.252356	0.943644	0.0
12	52	1	24	14	64	10.0	3.929600	2.470400	0.0
13	37	1	6	9	29	16.3	1.715901	3.011099	0.0
14	48	1	22	15	100	9.1	3.703700	5.396300	0.0
15	36	2	9	6	49	8.6	0.817516	3.396484	1.0
16	36	2	13	6	41	16.4	2.918216	3.805784	1.0

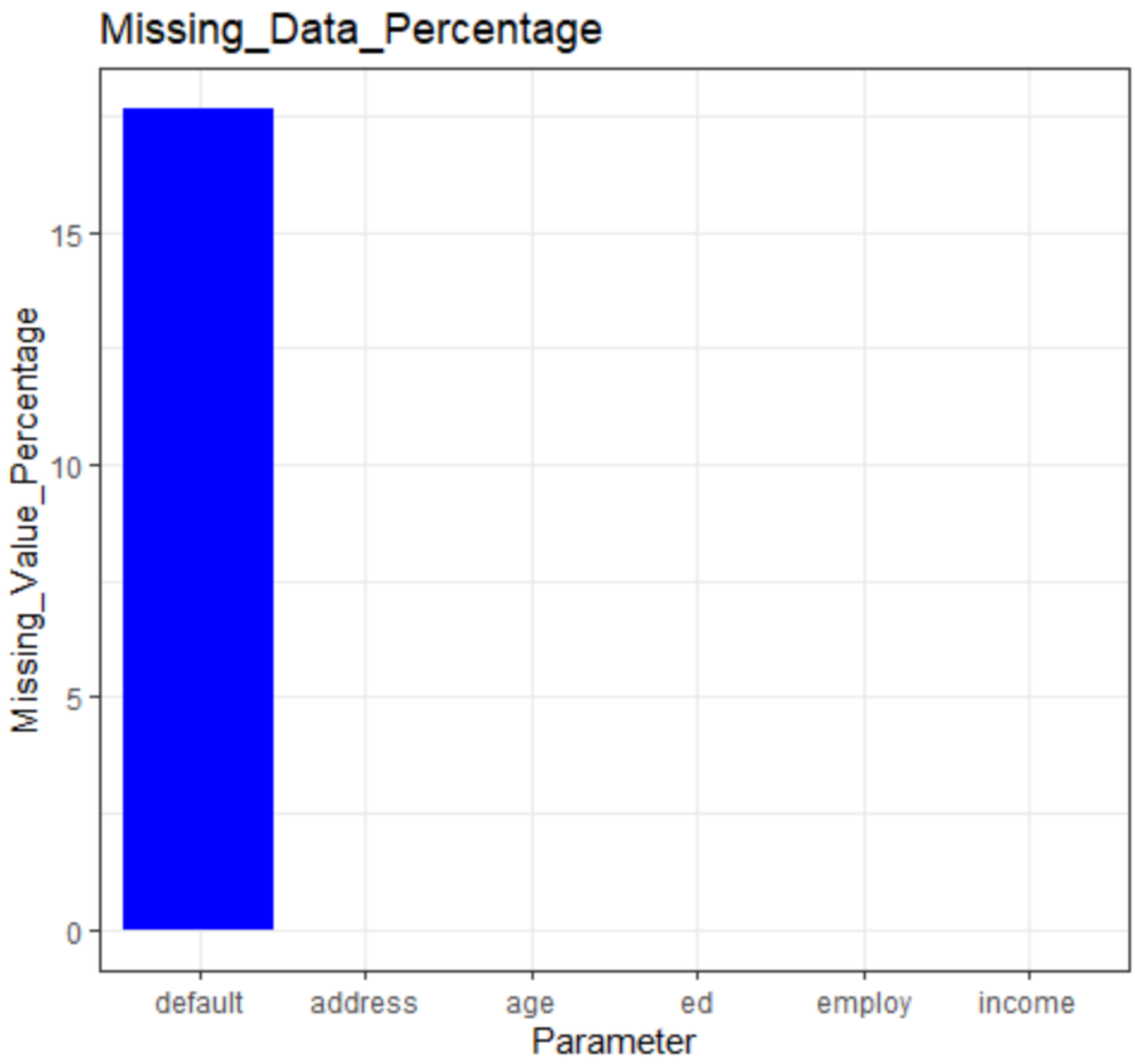
3. **Data Pre-processing** - This stage includes removing NA values, getting rid of redundant variables and any inconsistencies in the data. It is a data mining process that involves transformation of incomplete, inconsistent data which will be useful for building our model well. The data which we collect contains errors so we also remove them in this process.

   a. **Removing Variables -** In this process the Variables are removed which do not contribute much in the model development.
      - Education variable is removed as it do not provide much information for the loan status as being default or not
      - Address variable is also removed as it also don't carry much information it just gives the geographical area information.

   b. **Converting Data Types -** Converting the data type of variables which are not appropriate.
      - Converting the data type of variable default to categorical as it contains only two categories 0 and 1.

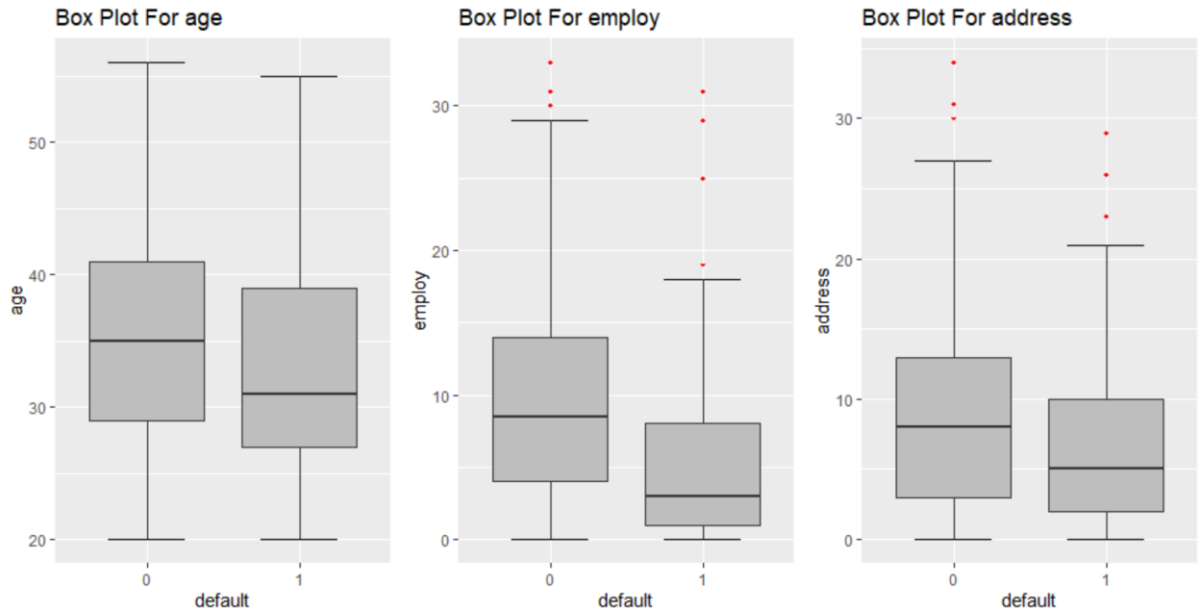
   c. **Missing value analysis** - Missing value means there are some observation in the data which are incomplete which affects the accuracy of the model.It

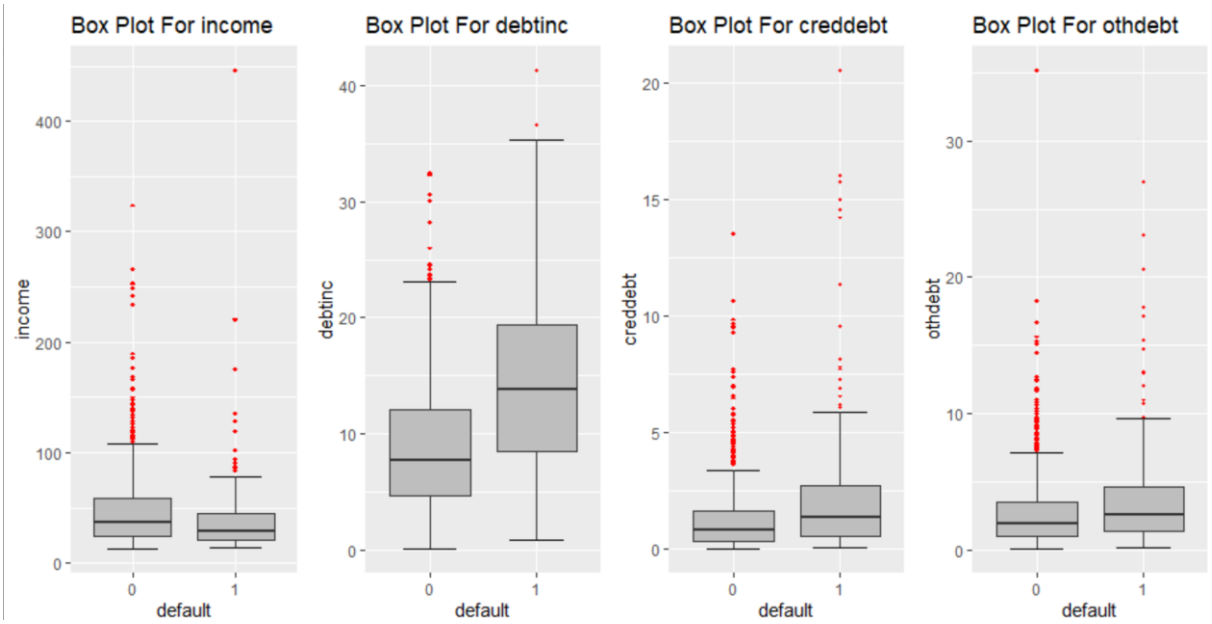
occurs in the data due to manual errors , wrong input, incomplete observations etc.There were various missing values in the dataset which are converted to NA and then its value is imputed using mean ,mode and knn imputations method.



In this dataset only default variable contains missing values which is a categorical variable .So in this the missing value is imputed using mode method

d.)Outlier Analysis - Outlier are the observations which are inconsistent with the rest of the dataset.It is the observations that deviate away from the other observations. It is due to poor data quality or contamination,low quality measurements,malfunctioning equipment,manual error.Outlier causes an error in predicting the target variable.



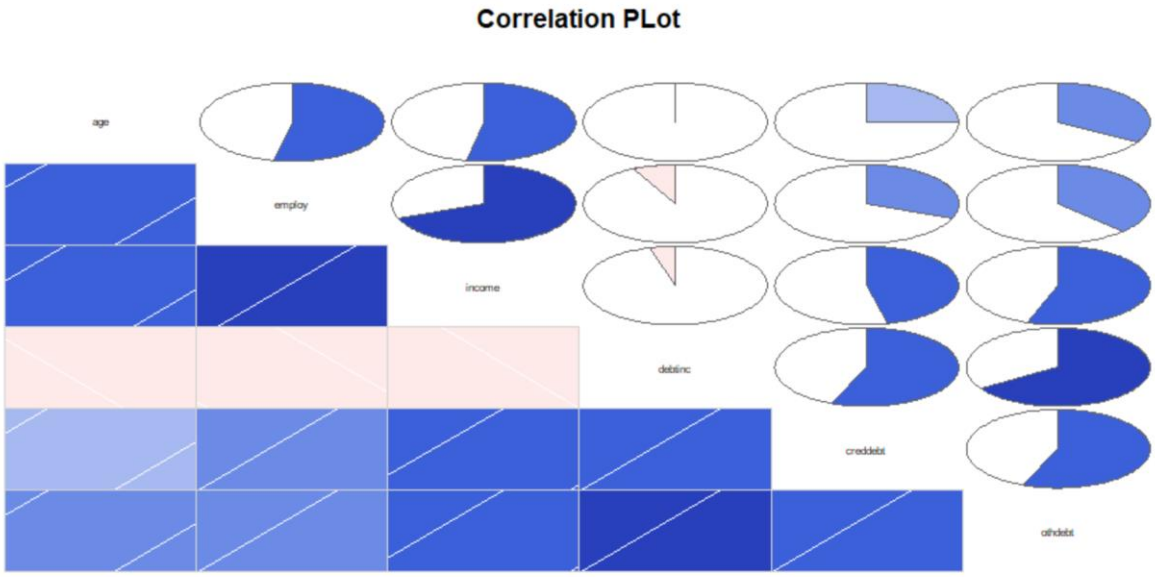


In this dataset there are outliers present in every variable which is converted to NA's and then the value is imputed using KNN imputation method.

e.)Feature Selection :-

Correlation Analysis -:

Correlation Analysis is used to define the relation between the variables.It tells the association between two continuous variables.Value of correlation analysis ranges from -1 to +1. It measures the direction and strength of the linear relationship between two quantitative variables.



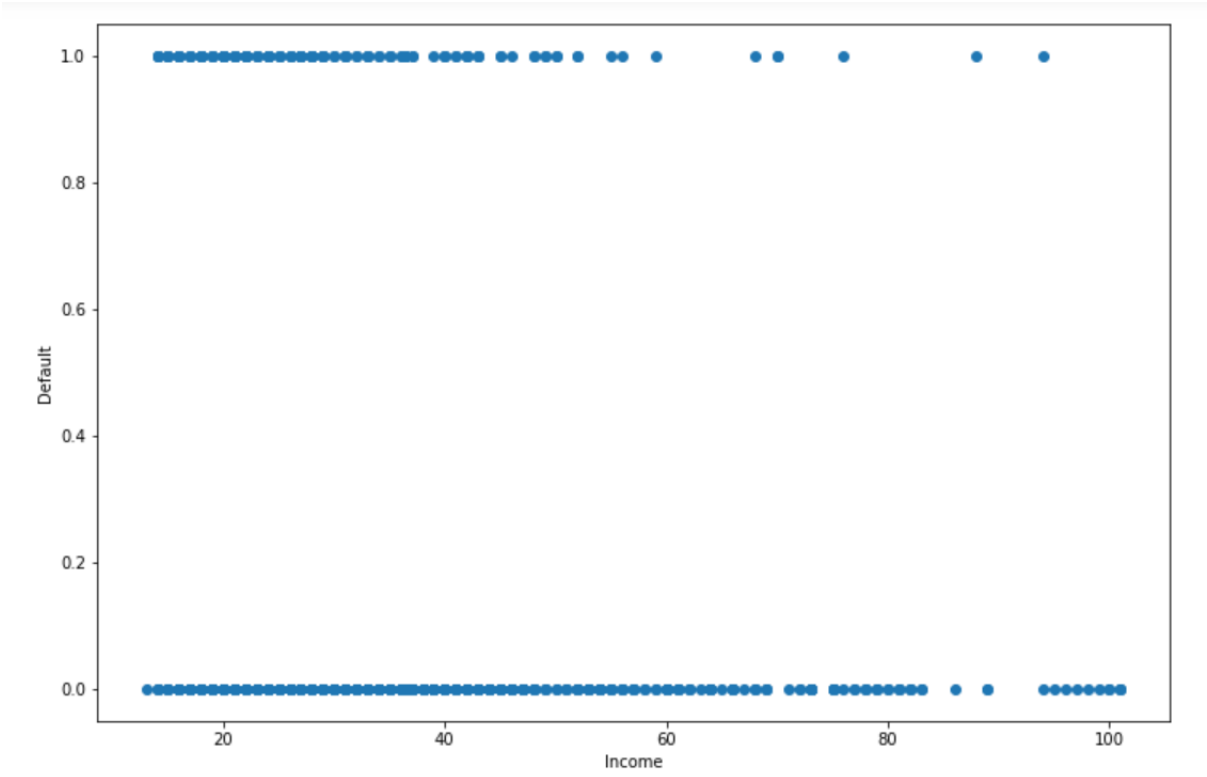
	age	employ	income	debtinc	creddebt	othdebt
age	1.000000	0.470075	0.489578	-0.063373	0.173010	0.208095
employ	0.470075	1.000000	0.650012	-0.190891	0.177285	0.200550
income	0.489578	0.650012	1.000000	-0.207573	0.312356	0.344615
debtinc	-0.063373	-0.190891	-0.207573	1.000000	0.489359	0.533810
creddebt	0.173010	0.177285	0.312356	0.489359	1.000000	0.432846
othdebt	0.208095	0.200550	0.344615	0.533810	0.432846	1.000000

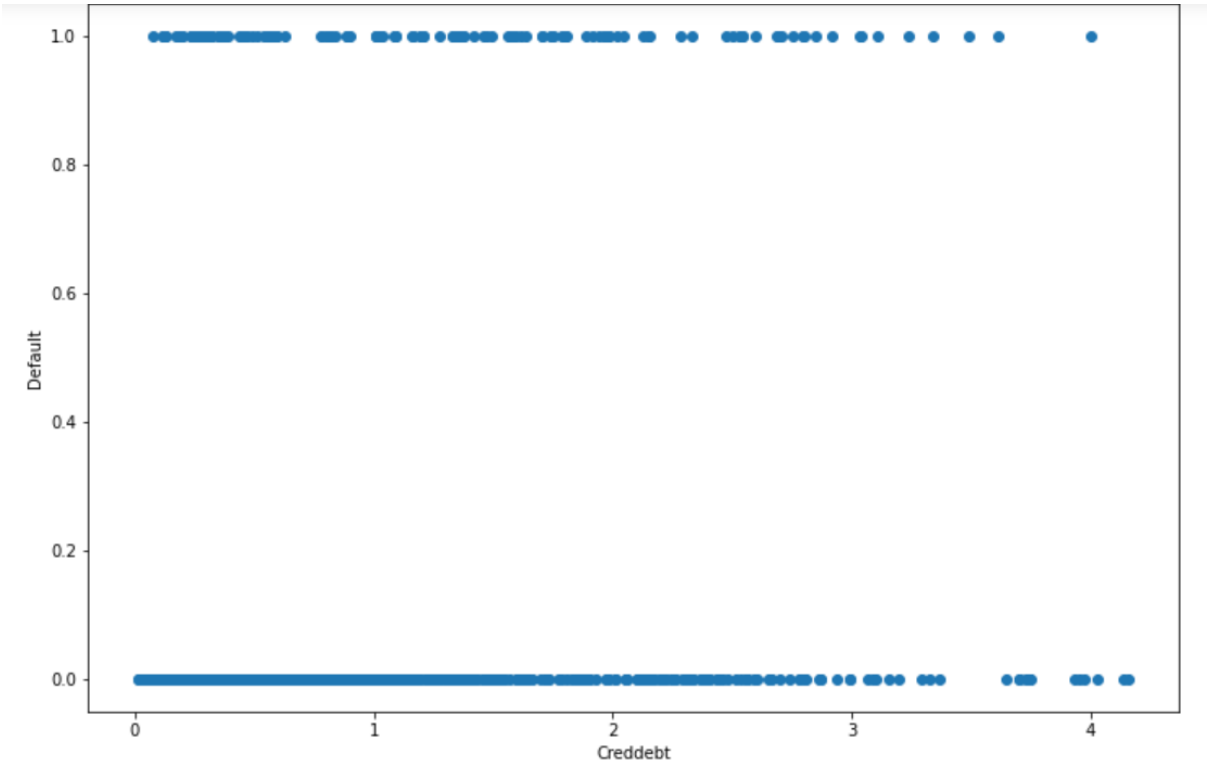
From the above it is clear that variables are positively correlated.

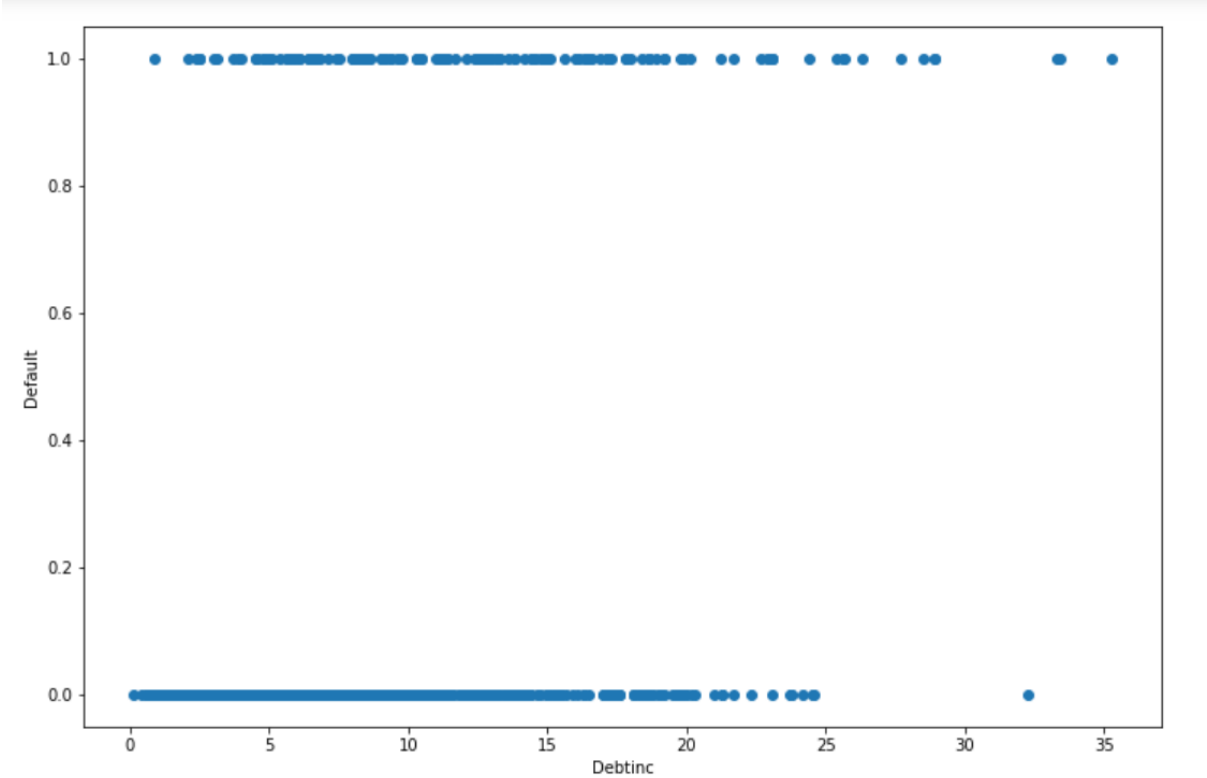
4. Data Exploration and Analysis :-

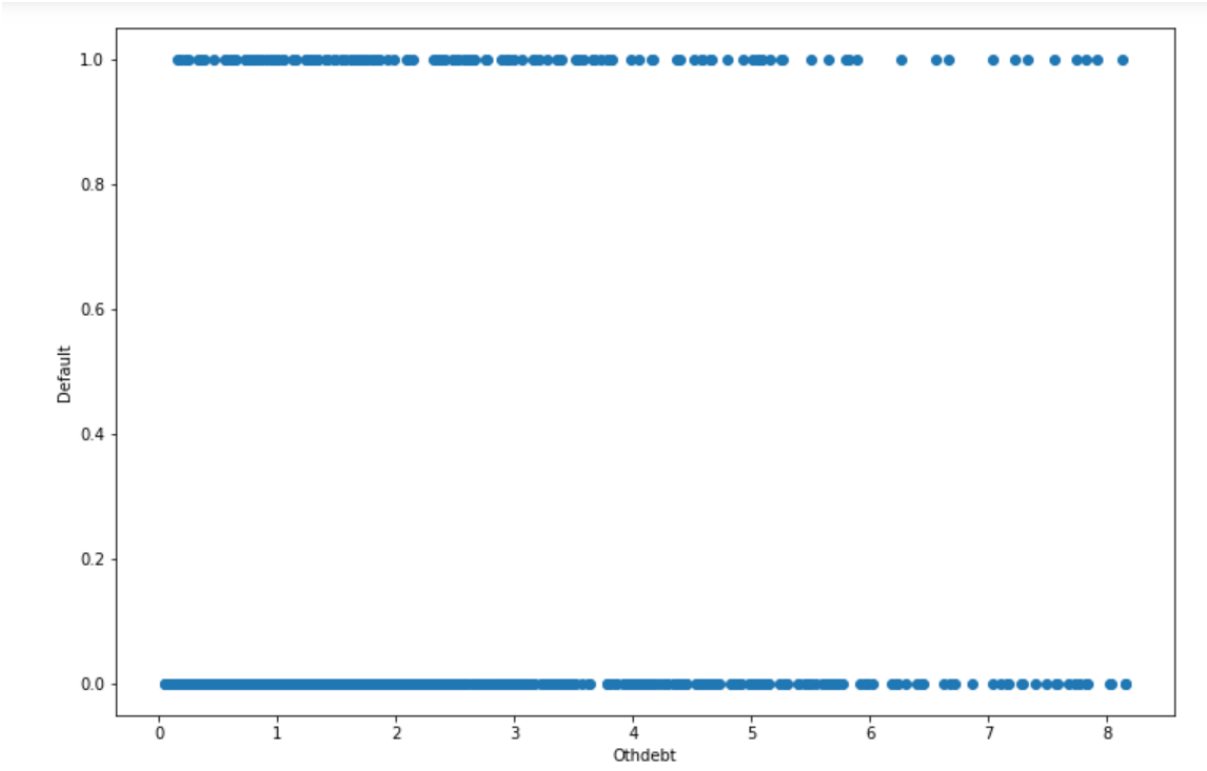
The dataset of bank loan dataset is pre processed using various methods and is now ready for model development.
Visualizations below shows relationship between variables

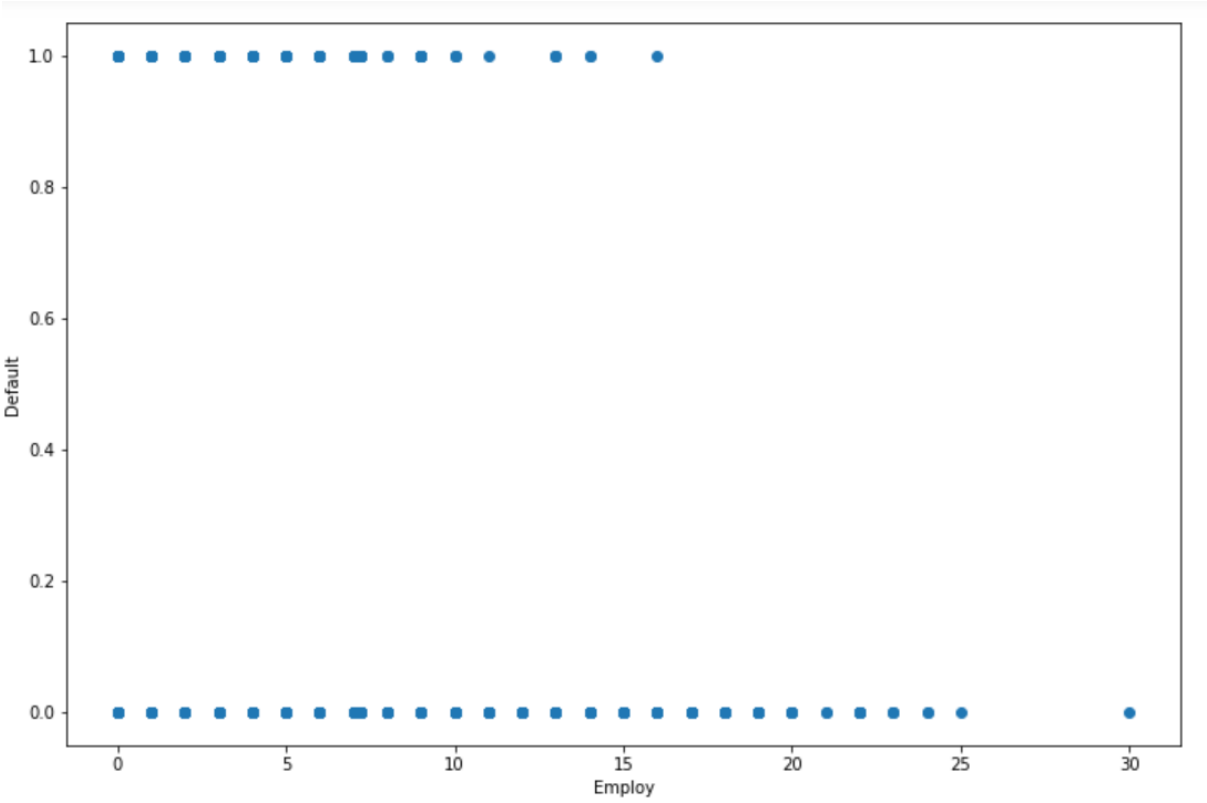

default
age
1.0
0.8
0.6
0.4
0.2
0.0
20
25
30
35
40
45
50
55


Default
Income
1.0
0.8
0.6
0.4
0.2
0.0
20
40
60
80
100


1.0
0.8
0.6
0.4
0.2
0.0
Default
0
1
2
3
4
Creddebt


1.0
0.8
0.6
0.4
0.2
0.0
Default
0
5
10
15
20
25
30
35
Debtinc


1.0
0.8
0.6
0.4
0.2
0.0
Default
0
1
2
3
4
5
6
7
8
Othdebt


1.0
0.8
0.6
0.4
0.2
0.0
Default
0
5
10
15
20
25
30
Employ

5.)**Model Development :-**

After preparing the data such as cleaning and analysis the next step is to develop a model on the processed data. As the target variable is categorical that is why it comes under Supervised classification problem statement. The algorithms which is used for Classification problem solving are Decision Trees algorithm, Random Forest ,Logistic Regression ,Naïve Bayes and KNN Imputation.

4.1. Decision Trees Algorithm -

Decision tree is a predictive model based on a branching series of boolean tests.

Decision Tree in R:

```
library(C50)
c50_model=C5.0(default ~.,train, trials=100, rules=TRUE)
summary(c50_model)
c50_predictions=predict(c50_model,test[,-9],type ="class")
```

Decision Tree in Python:

```
#Decision Tree
cif=tree.DecisionTreeClassifier(criterion='entropy').fit(x_train,y_train)
```

```
cif
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
y_predict=cif.predict(x_test)
```

5.2)Random Forest Model -

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

Random Forest In R :

```
library(randomForest)
rf_model=randomForest(default ~., train, importance = TRUE,
ntree=100)
library(inTrees)
tree_list=RF2List(rf_model)
exec=extractRules(tree_list,train[,-7])
exec[1:2,]
readableRules=presentRules(exec,colnames(train))
readableRules[1:2,]
rulemetric=getRuleMetric(exec,train[,-7],train$default)
rulemetric[1:2,]
RF_Predictions = predict(rf_model, test[,-7])
print(length(RF_Predictions))
```

Random Forest In Python :

```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
```

```
RF_model = RandomForestClassifier(n_estimators = 100).fit(x_train,y_train)
```

```
RF_model
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
RF_Predictions = RF_model.predict(x_test)
```

5.3)Logistic Regression Model –

The logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Logistic regression models the probabilities for classification problems with two possible outcomes. It's an extension of the linear regression model for classification problems.

Logistic Regression In R :

```
library(glmnet)
logit_model=glm(default ~.,data = train,family = "binomial")
summary(logit_model)
logit_predictions=predict(logit_model,newdata = test,type =
"response")
logit_predictions=ifelse(logit_predictions > 0.5,1,0)
```

Logistic Regression In Python:

```
logit=LogisticRegression(solver='liblinear').fit(x_train,y_train)
```

```
logit
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
logit_predict=logit.predict(x_test)
```

KNN Model –

KNN stands for K- nearest neighbour.KNN is simple algorithm that stores all available cases and classifies new cases based on a similarity measure.
It is a supervised lazy learning algorithm and can be used for both classification and regression.

KNN Imputation in R

```
library(class)
KNN_Predictions = knn(train[,1:7], test[, 1:7], train$default, k = 1)
```

KNN Imputation in Python

```
#KNN algorithm
from sklearn.neighbors import KNeighborsClassifier
```

```
KNN_model = KNeighborsClassifier(n_neighbors = 1).fit(x_train,y_train)
```

```
KNN_model
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=1, p=2,
                     weights='uniform')
```

```
KNN_Predictions = KNN_model.predict(x_test)
```

Naïve Bayes Model –

Naïve Bayes in R:

```
library(e1071)
NB_model=naiveBayes(default ~.,data=train)
NB_predictions=predict(NB_model,test[,1:6],type='class')
conf_matrix=table(observed=test[,7],predicted=NB_predictions)
confusionMatrix(conf_matrix)
```

Naïve Bayes in Python:

```
#NaivE Bayes
from sklearn.naive_bayes import GaussianNB
```

```
NB_model=GaussianNB().fit(x_train,y_train)
```

```
NB_model
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
NB_predict=NB_model.predict(x_test)
```

5. **Model Evaluation and Deployment :-**

Model is evaluated using the error metric Accuracy ,Recall,Error Rate and False Negative Rate . With the help of these metric model is selected on the basis of high accuracy low False Negative Rate(FNR),less error rate and more Recall.

Error metric is calculated with the help of confusion matrix of each model developed.

Accuracy=(TP+TN) *100/(TP+TN+FP+FN)
FNR =FN *100/(FN+TP)
Recall =(TP*100)/(TP+FN)
Error Rate=1 - Accuracy

Error Metric In R:

MODEL NAME	ACCURACY	FNR	RECALL	ERROR
Decision Tree	81%	15%	98%	19%
Random Forest	84%	3%	96%	15%
Logistic Regression	85%	3%	96%	16%
KNN	73%	18%	81%	27%
Naïve Bayes	79%	9%	90%	21%

Error Metric In Python:

MODEL NAME	ACCURACY	FNR	RECALL	ERROR
Decision Tree	74%	13%	96%	26%
Random Forest	78%	3%	96%	21%
Logistic Regression	77%	3%	96%	22%
KNN	75%	9%	90%	24%
Naïve Bayes	80%	5%	94%	20%

According to the predictions and analysis done on different models and also considering the error metric for comparison the best model for prediction among Decision Trees , Logistic Regression, Random Forest ,Naïve Bayes and KNN Imputation is **Logistic Regression.**

So the model which fits best to our dataset to predict the loan default status of bank customers more accurately is the **Logistic Regression.**

HyperParameter Tuning:

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm.
A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned. A hyperparameter **is** a parameter that is set before the learning process begins. These parameters are tunable and can directly affect how well a model trains.

```
parameters={'penalty':['l1','l2'],'C':[0.1,0.4,0.8,1,2,5]}
```

```
grid_search=GridSearchCV(LogisticRegression(solver='liblinear'),parameters,cv=3,return_train_score=True )
```

```
grid_search.fit(x_train,y_train)
```

```
GridSearchCV(cv=3, error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None, dual=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_ratio=None,
                                          max_iter=100, multi_class='auto',
                                          n_jobs=None, penalty='l2',
                                          random_state=None, solver='liblinear',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'C': [0.1, 0.4, 0.8, 1, 2, 5],
                         'penalty': ['l1', 'l2']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring=None, verbose=0)
```

```
grid_search.best_params_
```

```
{'C': 0.4, 'penalty': 'l2'}
```

```
#Optimizing Model
logistics_model=LogisticRegression(solver='liblinear',penalty=grid_search.best_params_['penalty'],
                C=grid_search.best_params_['C']).fit(x_train,y_train)
```

```
logistics_model
```

```
LogisticRegression(C=0.4, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
logit_predict=logistics_model.predict(x_test)
```

```
CM5=pd.crosstab(y_test,logit_predict)
TN5=CM5.iloc[1,1]
FN5=CM5.iloc[0,1]
TP5=CM5.iloc[0,0]
FP5=CM5.iloc[1,0]
#Accuracy
accuracy_score(y_test,logit_predict)*100
```

```
77.64705882352942
```

```
(FN5*100)/(FN5+TP5)
```

```
3.1496062992125986
```

```
#Recall
(TP5*100)/(TP5+FN5)
```

```
96.85039370078741
```

```
#Accuracy=77        FNR=3.14    Recall=96     Error Rate=22%
```