USE CASE STUDY REPORT

Group No.: Group 23

Student Names: Abhi Vyas, Nupur Dongare

I. Background and Introduction

Nowadays, there are numerous risks related to bank loans both for the banks and the borrowers getting the loans. The risk analysis about bank loans needs understanding about the risk and the risk level. Banks need to analyze their customers for loan eligibility so that they can specifically target those customers.Banks wanted to automate the loan eligibility process (real time) based on customer details such as Gender, Marital Status, Age, Occupation, Income, debts, and others provided in their online application form. As the number of transactions in banking sector is rapidly growing and huge data volumes are available, the customers' behavior can be easily analyzed and the risks around loan can be reduced. So, it is very important to predict the loan type and loan amount based on the banks' data.

Dataset:
About Company
Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan.
Problem Statement:
Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers. Here they have provided a partial data set.
Data source: https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/

Data Structure: Variable	**Description**
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi Urban/ Rural
Loan_Status	Loan approved (Y/N)

II. Data Exploration and Visualization

We removed the existing attributes like Loan_ID which do not contribute to the outcome variable and are present only for the numbering purposes.

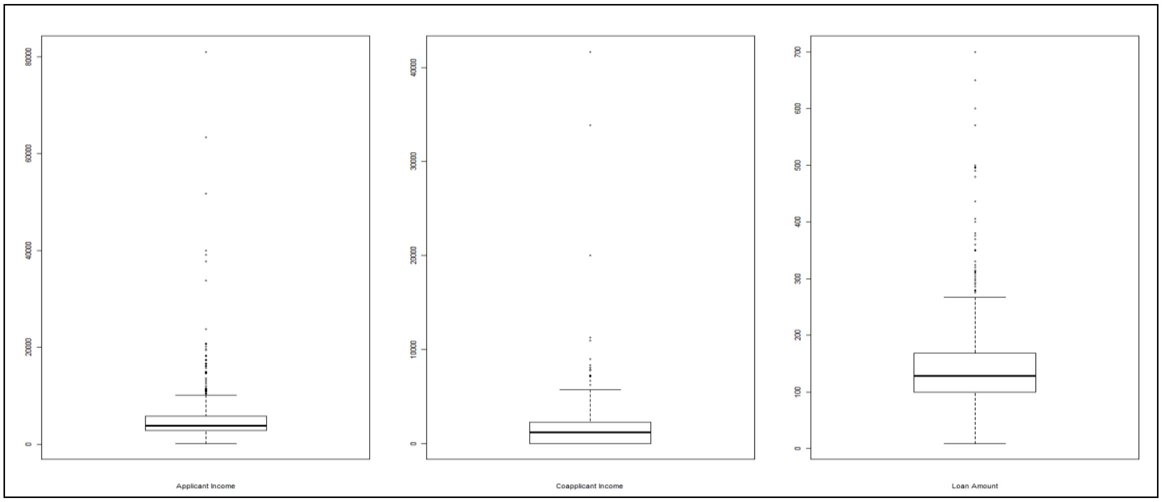
```
> str(traindata)
'data.frame':	614 obs. of  13 variables:
 $ Loan_ID          : Factor w/ 614 levels "LP001002","LP001003",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ Gender           : Factor w/ 3 levels "","Female","Male": 3 3 3 3 3 3 3 3 3 3 ...
 $ Married          : Factor w/ 3 levels "","No","Yes": 2 3 3 3 2 3 3 3 3 3 ...
 $ Dependents       : Factor w/ 5 levels "","0","1","2",..: 2 3 2 2 2 4 2 5 4 3 ...
 $ Education        : Factor w/ 2 levels "Graduate","Not Graduate": 1 1 1 2 1 1 2 1 1 1 ...
 $ Self_Employed    : Factor w/ 3 levels "","No","Yes": 2 2 3 2 2 3 2 2 2 2 ...
 $ ApplicantIncome  : int  5849 4583 3000 2583 6000 5417 2333 3036 4006 12841 ...
 $ CoapplicantIncome: num  0 1508 0 2358 0 ...
 $ LoanAmount       : int  NA 128 66 120 141 267 95 158 168 349 ...
 $ Loan_Amount_Term : int  360 360 360 360 360 360 360 360 360 360 ...
 $ Credit_History   : int  1 1 1 1 1 1 1 0 1 1 ...
 $ Property_Area    : Factor w/ 3 levels "Rural","Semiurban",..: 3 1 3 3 3 3 3 2 3 2 ...
 $ Loan_Status      : Factor w/ 2 levels "N","Y": 2 1 2 2 2 2 2 1 2 1 ...
> traindata$Loan_ID <- NULL
```

From the structure we could see that few variables are not in the proper form (continuous/ categorical) we want.
We converted them into categorical or numerical, as necessary.

Summary of data:

```
> summary(traindata)
    Gender      Married    Dependents        Education     Self_Employed ApplicantIncome CoapplicantIncome   LoanAmount   
       : 13        :  3      : 15    Graduate    :480      : 32      Min.   :  150   Min.   :    0     Min.   :  9.0  
 Female:112    No :213    0  :345    Not Graduate:134   No :500      1st Qu.: 2878   1st Qu.:    0     1st Qu.:100.0  
 Male  :489    Yes:398    1  :102                       Yes: 82      Median : 3812   Median : 1188     Median :128.0  
                          2  :101                                    Mean   : 5403   Mean   : 1621     Mean   :146.4  
                          3+: 51                                     3rd Qu.: 5795   3rd Qu.: 2297     3rd Qu.:168.0  
                                                                     Max.   :81000   Max.   :41667     Max.   :700.0  
                                                                                                       NA's   :22     
 Loan_Amount_Term Credit_History   Property_Area Loan_Status
 360    :512      0    : 89      Rural    :179   N:192      
 180    : 44      1    :475      Semiurban:233   Y:422      
 480    : 15      NA's: 50       Urban    :202              
 300    : 13                                                
 84     :  4                                                
 (Other): 12                                                
 NA's   : 14                                                
> |
```

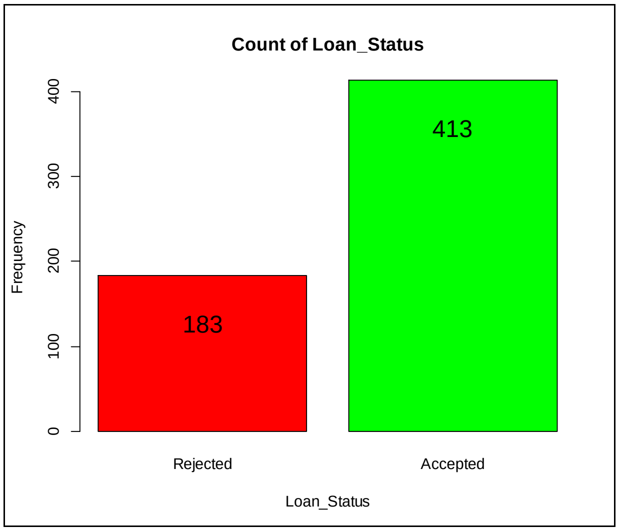
From the summary above, we could see that data is full of missing values and needs cleaning.



From the above boxplots we can see that there are many outliers present. Further, we plan to clean these outliers.
The missing values need to be cleared and scaling/normalizing the data is necessary. We are planning to study the relationship of each variable with every other variable using various data visualization tools like scatterplots, bar plots and boxplots .

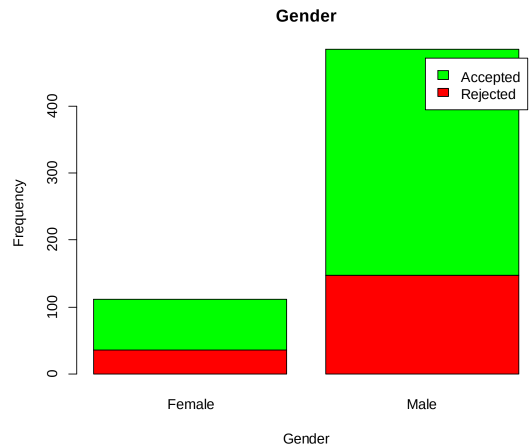
There were about 140 missing values in total, which if we had decided to eliminate completely would have resulted in loss of 140 records out of 614 records i.e. about 23% data would have been lost. Hence we went for filling up the missing values with relevant values. For categorical variables: We used ‘front fill’ function in Python to fill the missing values in categorical variables 'Self_Employed', 'Loan_Amount_Term' and 'Credit_History' Then, after exporting this dataset from Python to R, we filled the missing values for numerical variable ‘LoanAmount’ by the median of the entire dataset. We removed outliers for numerical variables ‘ApplicantIncome’ and ‘CoapplicantIncome’ by comparing the histograms for original and cleaned variables.

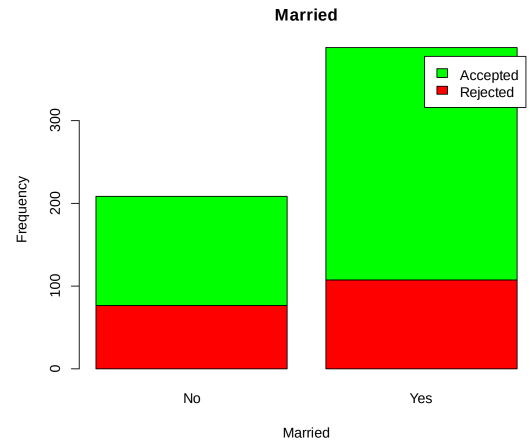
<u>Data Visualization:</u>

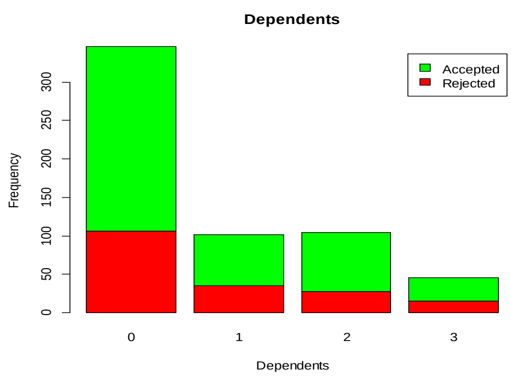


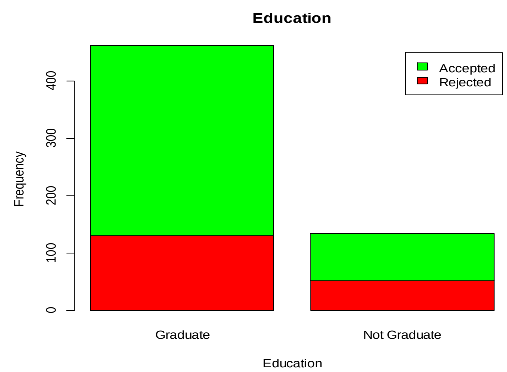
From above plot, we can see that Loan Application was “Accepted” for almost 70% of the cases.

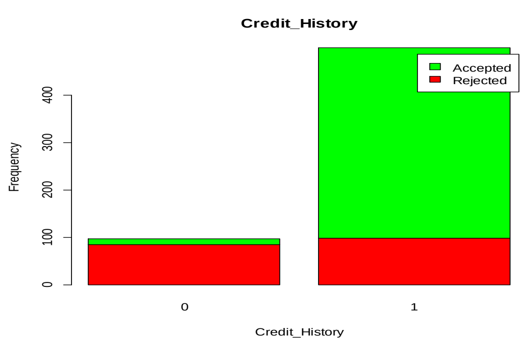
Now, we visualized relationship of our response variable with all our categorical predictors.

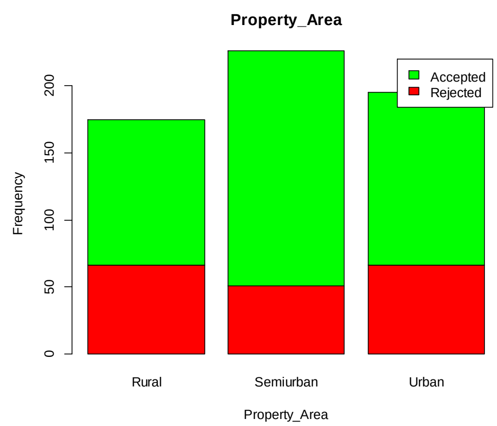


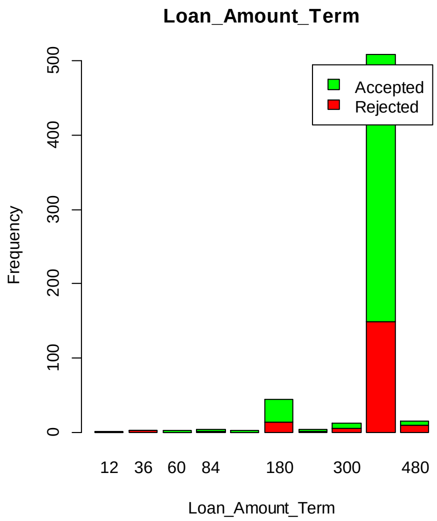


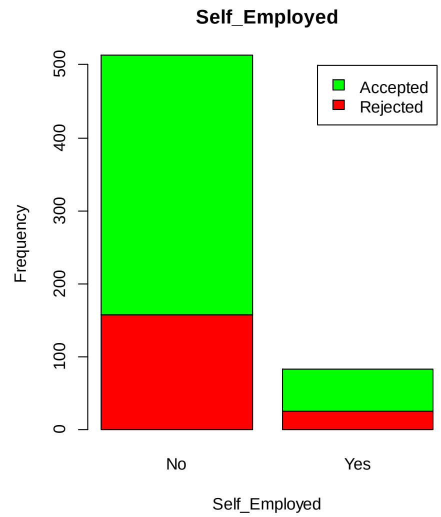




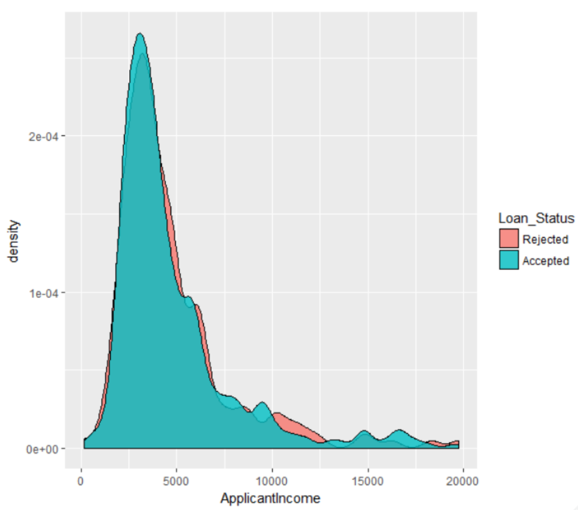


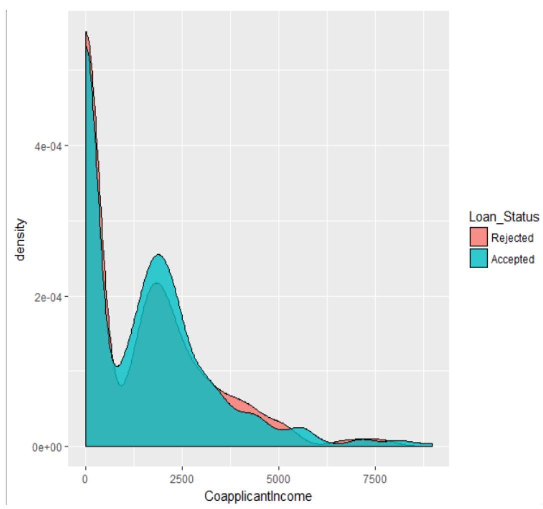


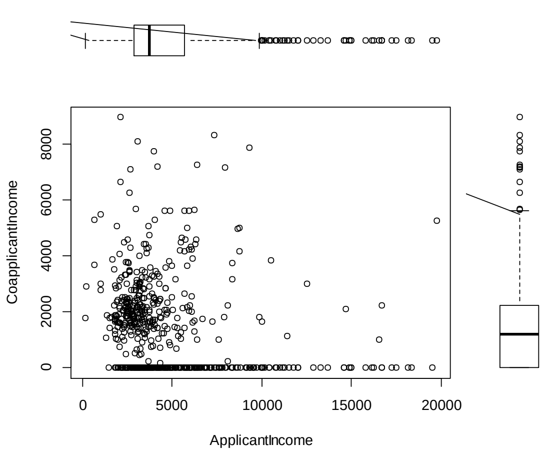


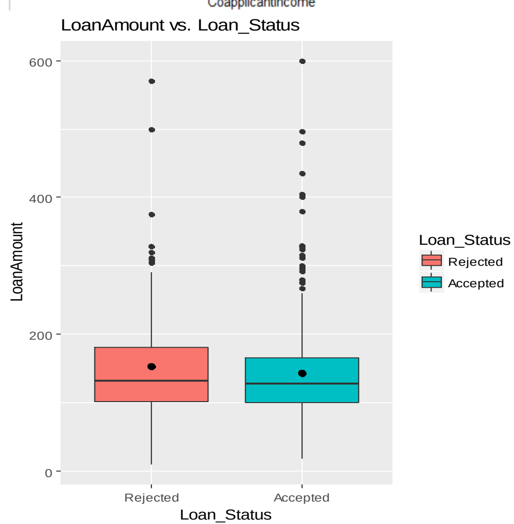


Now, we visualized relationship of our response variable with all our numerical predictors and relationship amongst them.







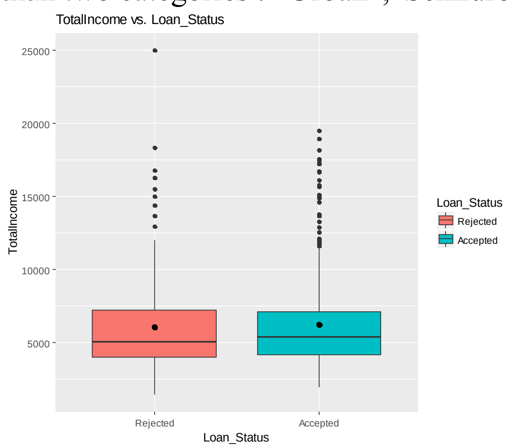


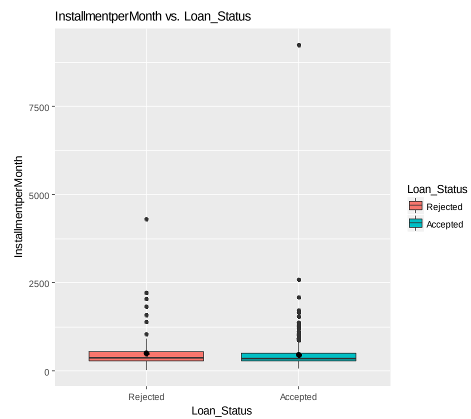
III. Data Preparation and Preprocessing

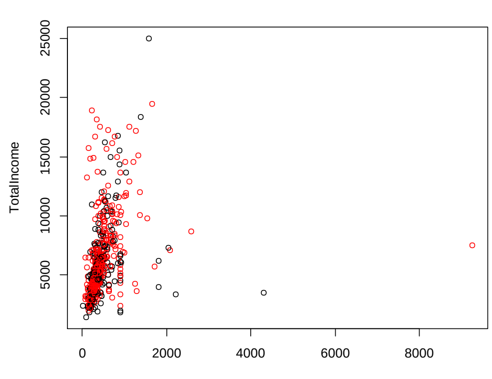
We could see from the above graphs that the variables ‘ApplicantIncome’ and ‘CoapplicantIncome’ can be represented by a combined variable “Totalincome”, which satisfactorily contributes in classification without adding unwanted errors.
We had two variables as ‘LoanAmount’ and ‘Loan_Amount_Term’. It seems irrelevant independently on output variables. So we tried to derive new variable called ‘InstallmentperIncome’ and checked its effect on output variable. To check its effect on output variable we plotted monthly installments against monthly income and check against loan status.

```
#Feature Engineering
Loan_Prediction$TotalIncome<-Loan_Prediction$ApplicantIncome+Loan_Prediction$CoapplicantIncome
Loan_Prediction<-Loan_Prediction[,c(-6,-7)]
ggplot(Loan_Prediction,aes(x=Loan_Status, y=TotalIncome)) + geom_boxplot(aes(fill = Loan_Status)) + stat_summary(f
Loan_Prediction$InstallmentperMonth<-((Loan_Prediction$LoanAmount*1000)/Loan_Prediction$Loan_Amount_Term)
Loan_Prediction<-Loan_Prediction[,c(-6,-7)]
ggplot(Loan_Prediction,aes(x=Loan_Status, y=InstallmentperMonth)) + geom_boxplot(aes(fill = Loan_Status)) + stat_s
plot(Loan_Prediction$InstallmentperMonth, Loan_Prediction$TotalIncome,
          xlab="InstallmentperMonth",
          ylab="TotalIncome",col=Loan_Prediction$Loan_Status,legend=row.names(Loan_Prediction$Loan_Status))
```

It is indicating that as the income increases against monthly installments chances of loan status approval also increases.
Also, created dummy variables for categorical variable ‘Property_Area’ as it has more than two categories : “Urban”,”Semiurban”,”Rural”.







IV. Data Mining Techniques and Implementation

As the goal of our project is to classify the response variable in "Accepted" and "Rejected", we selected below three algorithms which we thought would work the best for our data.

Also, we used one dataset for kNN & Naïve Bayes and different dataset for CART. As categorical predictors converted into binary numerical variables work best with former whereas categorical variables as "factor" class work best with later.

We partitioned our data into Training(50%), Validation(30%) and Test(20%) for all the models built.

1) Naïve Bayes:

   Console:

```
> Loan_Prediction.nb<-naiveBayes(Loan_Status~.,data=train.df)
> Loan_Prediction.nb

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
 Rejected  Accepted
0.3322148 0.6677852

Conditional probabilities:
          Gender
Y               [,1]      [,2]
  Rejected 0.7878788 0.4108907
  Accepted 0.8090452 0.3940448

          Married
Y               [,1]      [,2]
  Rejected 0.5757576 0.4967426
  Accepted 0.6582915 0.4754786

          Dependents
Y               [,1]      [,2]
  Rejected 0.7373737 1.0059604
  Accepted 0.6934673 0.9647879

          Education
Y               [,1]      [,2]
  Rejected 0.6767677 0.4700908
  Accepted 0.7839196 0.4126078

          Self_Employed
Y               [,1]      [,2]
  Rejected 0.1515152 0.3603750
  Accepted 0.1155779 0.3205244

          Credit_History
Y               [,1]      [,2]
  Rejected 0.5151515 0.5023138
  Accepted 0.9698492 0.1714333
```

```
          TotalIncome
Y               [,1]     [,2]
  Rejected 6052.960 3361.410
  Accepted 6350.386 3414.834

          InstallmentperMonth
Y               [,1]     [,2]
  Rejected 488.8047 304.5950
  Accepted 439.5484 278.6606

          Urban
Y                  0         1
  Rejected 0.6363636 0.3636364
  Accepted 0.6834171 0.3165829

          Semiurban
Y                  0         1
  Rejected 0.7474747 0.2525253
  Accepted 0.5678392 0.4321608

          Rural
Y                  0         1
  Rejected 0.6161616 0.3838384
  Accepted 0.7487437 0.2512563

> pred.class.valid <- predict(Loan_Prediction.nb, newdata = valid.df)
> table(pred.class.valid, valid.df$Loan_Status)

pred.class.valid Rejected Accepted
        Rejected       24        7
        Accepted       24      123

> df <- data.frame(actual = valid.df$Loan_Status, predicted = pred.class.valid, pred.pr
> View(df)
```

	actual	predicted	Rejected	Accepted
1	Accepted	Accepted	0.06118607	9.388139e-01
2	Rejected	Rejected	0.99999821	1.790945e-06
3	Accepted	Accepted	0.01578999	9.842100e-01
4	Accepted	Accepted	0.04366362	9.563364e-01
5	Accepted	Accepted	0.12435388	8.756461e-01
6	Accepted	Accepted	0.08943402	9.105660e-01
7	Accepted	Accepted	0.02226676	9.777332e-01
8	Accepted	Accepted	0.15359194	8.464081e-01
9	Rejected	Accepted	0.13467878	8.653212e-01
0	Accepted	Accepted	0.18859229	8.114077e-01
1	Rejected	Accepted	0.08872808	9.112719e-01
2	Accepted	Accepted	0.08573807	9.142619e-01
3	Rejected	Accepted	0.16263642	8.373636e-01
4	Accepted	Accepted	0.03045117	9.695488e-01
5	Accepted	Accepted	0.06607171	9.339283e-01
6	Rejected	Rejected	0.99999912	8.795846e-07

Naïve bayes gives accuracy of: 82.58% for validation dataset.

2) KNN:

Console:

```
> #knn
> library(class)
> Actualtrainclass<-train.df$Loan_Status
> Actualvalidclass<-valid.df$Loan_Status
> knn1<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=1)
> # confusionMatrix(knn1, valid.df$Loan_Status)
> table(knn1,valid.df$Loan_Status)

knn1        Rejected Accepted
  Rejected        19       49
  Accepted        29       81
> knn3<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=3)
> # confusionMatrix(knn3, valid.df$Loan_Status)
> table(knn3,valid.df$Loan_Status)

knn3        Rejected Accepted
  Rejected        12       30
  Accepted        36      100
> knn5<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=5)
> # confusionMatrix(knn5, valid.df$Loan_Status)
> table(knn5,valid.df$Loan_Status)

knn5        Rejected Accepted
  Rejected         7       19
  Accepted        41      111
> knn7<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=7)
> # confusionMatrix(knn7, valid.df$Loan_Status)
> table(knn7,valid.df$Loan_Status)

knn7        Rejected Accepted
  Rejected         4       15
  Accepted        44      115
> knn9<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=9)
> # confusionMatrix(knn9, valid.df$Loan_Status)
> table(knn9,valid.df$Loan_Status)

knn9        Rejected Accepted
  Rejected         3       10
  Accepted        45      120
> knn11<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=11)
> # confusionMatrix(knn11, valid.df$Loan_Status)
> table(knn11,valid.df$Loan_Status)

knn11       Rejected Accepted
  Rejected         4        8
  Accepted        44      122
> knn13<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=13)
> # confusionMatrix(knn13, valid.df$Loan_Status)
> table(knn13,valid.df$Loan_Status)

knn13       Rejected Accepted
  Rejected         3        9
  Accepted        45      121
```

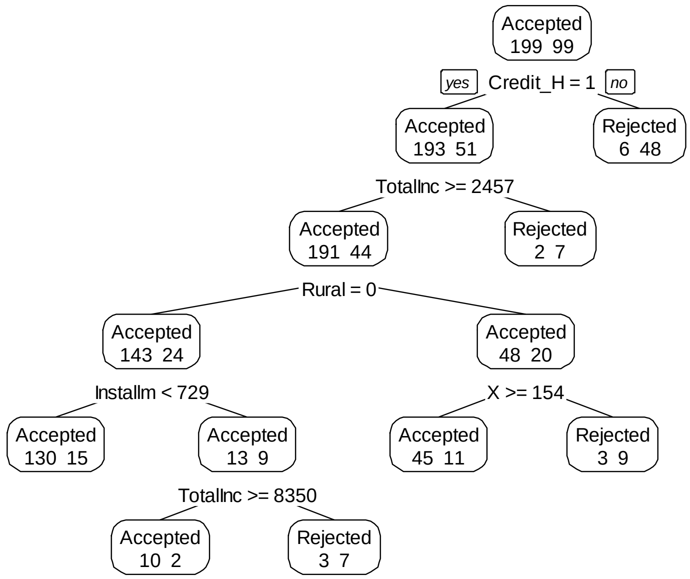
We found that kNN=11 gives best accuracy of 70.78 % and will be selected for our analysis.

3) CART:

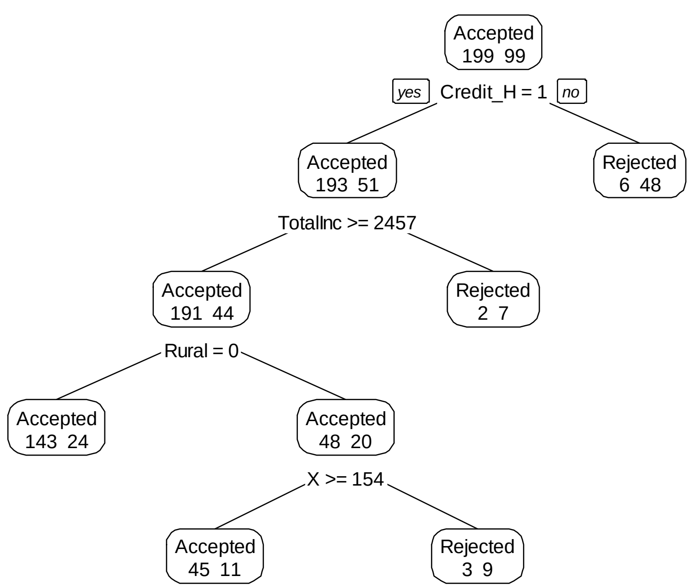
We built a tree with lowest cp and then pruned it with the best cp and ran the model on validation dataset and test dataset.

Console:

```
> #Begin with small cp
> CT1<-rpart(Loan_Status~.,data=train.df,method="class",control =
rpart.control(cp = 0.0001))
> CT1
n= 298

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 298 99 Accepted (0.6677852 0.3322148)
   2) Credit_History=1 244 51 Accepted (0.7909836 0.2090164)
     4) TotalIncome>=2457 235 44 Accepted (0.8127660 0.1872340)
       8) Rural=0 167 24 Accepted (0.8562874 0.1437126)
        16) InstallmentperMonth< 729.1667 145 15 Accepted (0.8965517
0.1034483) *
        17) InstallmentperMonth>=729.1667 22  9 Accepted (0.5909091
0.4090909)
          34) TotalIncome>=8350 12  2 Accepted (0.8333333 0.1666667) *
          35) TotalIncome< 8350 10  3 Rejected (0.3000000 0.7000000) *
       9) Rural=1 68 20 Accepted (0.7058824 0.2941176)
        18) X>=154 56 11 Accepted (0.8035714 0.1964286) *
        19) X< 154 12  3 Rejected (0.2500000 0.7500000) *
     5) TotalIncome< 2457 9  2 Rejected (0.2222222 0.7777778) *
   3) Credit_History=0 54  6 Rejected (0.1111111 0.8888889) *
> printcp(CT1)

Classification tree:
rpart(formula = Loan_Status ~ ., data = train.df, method = "class",
    control = rpart.control(cp = 1e-04))

Variables actually used in tree construction:
[1] Credit_History      InstallmentperMonth Rural
TotalIncome         X

Root node error: 99/298 = 0.33221

n= 298

        CP nsplit rel error  xerror     xstd
1 0.424242      0   1.00000 1.00000 0.082130
2 0.050505      1   0.57576 0.57576 0.068581
3 0.030303      2   0.52525 0.58586 0.069036
4 0.020202      4   0.46465 0.55556 0.067646
5 0.000100      6   0.42424 0.60606 0.069923
> prp(CT1, type = 2, extra = 1, split.font = 1, varlen = -8)
```



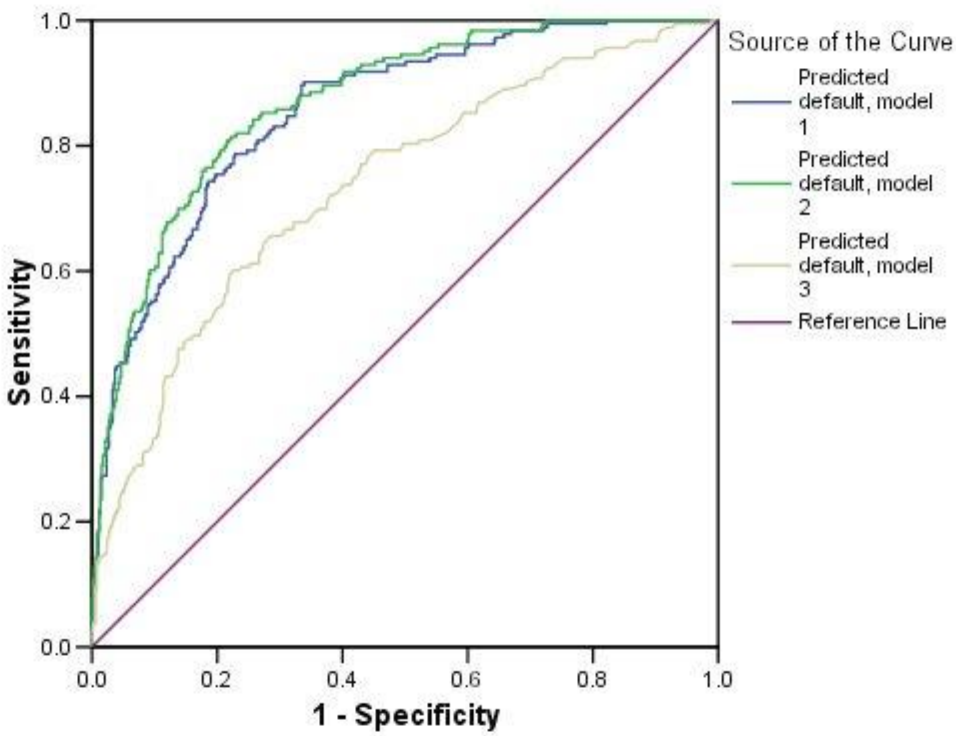

```
> # Step2: Pick the tree size that minimizes misclassification rate
(i.e. prediction error).
> # Prediction error rate in training data = Root node error * rel
error * 100%
> # Prediction error rate in cross-validation = Root node error *
xerror * 100%
> # Hence we want the cp value (with a simpler tree) that minimizes the
xerror.


> bestcp <- CT1$cptable[which.min(CT1$cptable[,"xerror"]),"CP"]
> # Step3: Prune the tree using the best cp.
> tree.pruned <- prune(CT1, cp = bestcp)
> prp(tree.pruned, type = 2, extra = 1, split.font = 1, varlen = -8)
```



```
> #confusion matrix (training data)
> conf.matrix.train <- table(train.df$Loan_Status,
predict(tree.pruned,type="class"))
> rownames(conf.matrix.train) <- paste("Actual",
rownames(conf.matrix.train), sep = ":")
> colnames(conf.matrix.train) <- paste("Pred",
colnames(conf.matrix.train), sep = ":")
> print(conf.matrix.train)

                  Pred:Accepted Pred:Rejected
  Actual:Accepted           188            11
  Actual:Rejected            35            64
> #Work on validation dataset
> pred.valid<-predict(tree.pruned,valid.df,type="class")
> conf.matrix.valid <- table(pred.valid,valid.df$Loan_Status)
> rownames(conf.matrix.valid) <- paste("Actual",
rownames(conf.matrix.valid), sep = ":")
> colnames(conf.matrix.valid) <- paste("Pred",
colnames(conf.matrix.valid), sep = ":")
> print(conf.matrix.valid)

pred.valid        Pred:Accepted Pred:Rejected
  Actual:Accepted           123            23
  Actual:Rejected             7            25
> #Work on test dataset
> pred.test<-predict(tree.pruned,test.df,type="class")
> conf.matrix.test <- table(pred.test,test.df$Loan_Status)
> rownames(conf.matrix.test) <- paste("Actual",
rownames(conf.matrix.test), sep = ":")
> colnames(conf.matrix.test) <- paste("Pred",
colnames(conf.matrix.test), sep = ":")
> print(conf.matrix.test)

pred.test         Pred:Accepted Pred:Rejected
  Actual:Accepted            78            20
  Actual:Rejected             6            16
```

CART gives accuracy of 84.56% for training dataset, 83.14% for validation dataset and 78.33% for test dataset

V. Performance Evaluation



Diagonal segments are produced by ties.

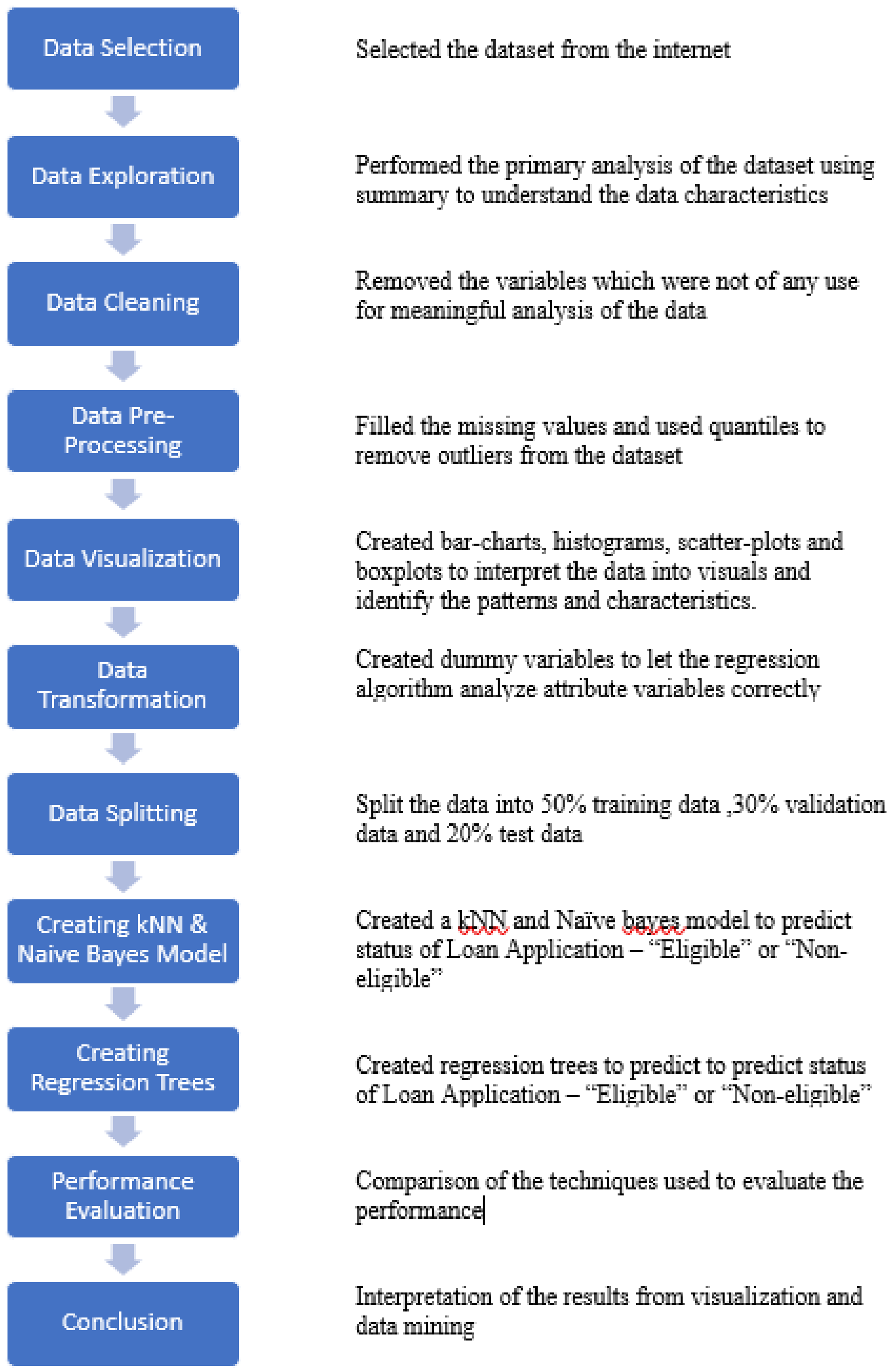
In above chart, model 1 is Naïve Bayes, model 2 is CART, model 3 is kNN.

VI. Discussion and Recommendation

As our task was to classify the application as “Accepted” or “Rejected”, we decided to take ‘Accuracy’ as deciding metric as it is necessary to accurately classify the eligibility as a result of automating the process. We did a lot of work in cleaning and preprocessing of data as well as in refining the datasets for every model. That has made our results much more reliable. Also, as CART gave best accuracy for our data, we ran the tree on test data and error came out to be 78.33%.

The potential improvement could be of trying the results using different cut-off values. Also, here we could not implement “confusionMatrix” function from package “caret” as it kept crashing whenever we tried installing it, tried on several computers.

VII. Summary



Appendix: R Code for use case study

```
# install.packages(c("dplyr","knitr"))
```

```
#
install.packages(c("ggplot2","ggpubr","formattable","gtools","scales","corrplot","caret","
pROC","ROCR","tidyr","parallel","parallelMap","DT","DMwR","mlr"))
# install.packages(c("caretEnsemble"))
# library(dplyr)
# library(knitr)
# library(ggplot2)
# library(ggpubr)
# library(formattable)
# library(gtools)
# library(scales)
# library(corrplot)
# library(caret)
# library(pROC)
# library(ROCR)
# library(tidyr)
# library(parallel)
# library(parallelMap)
# library(mlr)
# library(caretEnsemble)
# library(DT)
# library(DMwR)
# library(RColorBrewer)
setwd("C:/Users/nupur/OneDrive/Documents/Data Mining/use case study")
Loan_Prediction<-read.csv("training_data_cleaned.csv")
View(Loan_Prediction)
Loan_Prediction$Credit_History<-factor(Loan_Prediction$Credit_History, levels =
c(0,1))
Loan_Prediction$Loan_Status = as.numeric(Loan_Prediction$Loan_Status)-1
Loan_Prediction$Loan_Status<-factor(Loan_Prediction$Loan_Status, levels = c(0,1),
labels = c("Rejected", "Accepted"))
Loan_Prediction$Dependents<-factor(Loan_Prediction$Dependents, levels = c(0,1,2,3))
#Barplot for count of Loan_Status
freqLS<-table(Loan_Prediction$Loan_Status)
countls<-barplot(freqLS,main="Count of Loan_Status",
        xlab="Loan_Status", ylab="Frequency",
        col=c("red", "green"))
text(x = countls, y = freqLS, label = freqLS, pos = 3,offset=-3, cex = 1.6, col = "black")
#barplot for count of LS vs gender
freqgen<-table(Loan_Prediction$Loan_Status,Loan_Prediction$Gender)
countgen<-barplot(freqgen,main="Gender",
                 xlab="Gender", ylab="Frequency",
                 col=c("red", "green"),legend = row.names(freqgen))

#barplot for count of LS vs married
freqmar<-table(Loan_Prediction$Loan_Status,Loan_Prediction$Married)
countmar<-barplot(freqmar,main="Married",
```

```
            xlab="Married", ylab="Frequency",
            col=c("red", "green"),legend = row.names(freqmar))

#barplot for count of LS vs Dependents
freqdep<-table(Loan_Prediction$Loan_Status,Loan_Prediction$Dependents)
countdep<-barplot(freqdep,main="Dependents",
            xlab="Dependents", ylab="Frequency",
            col=c("red", "green"),legend = row.names(freqdep))

#barplot for count of LS vs Education
freqedu<-table(Loan_Prediction$Loan_Status,Loan_Prediction$Education)
countedu<-barplot(freqedu,main="Education",
            xlab="Education", ylab="Frequency",
            col=c("red", "green"),legend = row.names(freqedu))

#barplot for count of LS vs Self_Employed
freqse<-table(Loan_Prediction$Loan_Status,Loan_Prediction$Self_Employed)
countse<-barplot(freqse,main="Self_Employed",
            xlab="Self_Employed", ylab="Frequency",
            col=c("red", "green"),legend = row.names(freqse))

#barplot for count of LS vs Credit_History
freqch<-table(Loan_Prediction$Loan_Status,Loan_Prediction$Credit_History)
countch<-barplot(freqch,main="Credit_History",
            xlab="Credit_History", ylab="Frequency",
            col=c("red", "green"),legend = row.names(freqch))

#barplot for count of LS vs Property_Area
freqpa<-table(Loan_Prediction$Loan_Status,Loan_Prediction$Property_Area)
countpa<-barplot(freqpa,main="Property_Area",
            xlab="Property_Area", ylab="Frequency",
            col=c("red", "green"),legend = row.names(freqpa))

#

Factor_summary.DF = data.frame(Variable = character(), Test_statistic = numeric(),
p_value = numeric(), cramers_v = numeric(), Dependency = character())

factor_cols = (Loan_Prediction[, sapply(Loan_Prediction, is.factor)] %>% names())[-12]


for(i in factor_cols){
  temp.DF = Loan_Prediction[,c(i,"Loan_Status")]%>% na.omit()
  test_value = chisq.test(temp.DF[,1], temp.DF[,2], correct = F)


  k = temp.DF[,1] %>% unique() %>% length()
```

```
  r = temp.DF[,2] %>% unique %>% length()



  cbind("Variable" = i, "Test_statistic" = test_value$statistic %>% round(4), p_value =
test_value$p.value %>% round(4),
      Dependency = ifelse(test_value$p.value >= 0.05, "Independent", "Dependent"))
%>% as.data.frame() -> result.DF

  Factor_summary.DF = rbind(Factor_summary.DF, result.DF)
}

rownames(Factor_summary.DF) <- NULL

Factor_summary.DF %>% mutate(Dependency = ifelse(Dependency == 'Dependent',
                                  color_tile("white", "orange")(Dependency),
                                  cell_spec(
                                    Dependency, "html", color = "white", bold = T,
                                    background = spec_color(3, end = 0.9, option = "A",
direction = -1))
))
Factor_summary.DF
#
ggplot(Loan_Prediction,aes(x=Loan_Status, y=ApplicantIncome)) +
geom_boxplot(aes(fill = Loan_Status)) + stat_summary(fun.y = mean, geom="point",
size=2) + xlab('Loan_Status') + ylab('ApplicantIncome') +  ggtitle('ApplicantIncome vs.
Loan_Status')
ggplot(Loan_Prediction,aes(x=Loan_Status, y=CoapplicantIncome)) +
geom_boxplot(aes(fill = Loan_Status)) + stat_summary(fun.y = mean, geom="point",
size=2) + xlab('Loan_Status') + ylab('CoapplicantIncome') +  ggtitle('CoapplicantIncome
vs. Loan_Status')
opar <- par(no.readonly=TRUE)
par(fig=c(0, 0.8, 0, 0.8))
plot(Loan_Prediction$ApplicantIncome, Loan_Prediction$CoapplicantIncome,
    xlab="ApplicantIncome",
    ylab="CoapplicantIncome")

par(fig=c(0, 0.8, 0.55, 1), new=TRUE)
boxplot(Loan_Prediction$ApplicantIncome, horizontal=TRUE, axes=FALSE)
par(fig=c(0.65, 1, 0, 0.8), new=TRUE)
boxplot(Loan_Prediction$CoapplicantIncome, axes=FALSE)
par(opar)
d<-density(Loan_Prediction$ApplicantIncome)
plot(d)
d<-density(Loan_Prediction$CoapplicantIncome)
d<-density(Loan_Prediction$ApplicantIncome)
d2<-density(Loan_Prediction$CoapplicantIncome)
```

```
plot(d2)
# h<-hist(Loan_Prediction$ApplicantIncome)
# h<-hist(log(Loan_Prediction$ApplicantIncome))
# h2<-hist(Loan_Prediction$CoapplicantIncome)
# h2<-hist(log(Loan_Prediction$CoapplicantIncome))
qplot(ApplicantIncome,data=Loan_Prediction,geom="density",adjust=0.7,size=I(0.7),fill
=Loan_Status,alpha=I(0.8))
qplot(CoapplicantIncome,data=Loan_Prediction,geom="density",adjust=0.7,size=I(0.7),f
ill=Loan_Status,alpha=I(0.8))
#Feature Engineering
Loan_Prediction$TotalIncome<-
Loan_Prediction$ApplicantIncome+Loan_Prediction$CoapplicantIncome
Loan_Prediction<-Loan_Prediction[,c(-6,-7)]
ggplot(Loan_Prediction,aes(x=Loan_Status, y=TotalIncome)) + geom_boxplot(aes(fill =
Loan_Status)) + stat_summary(fun.y = mean, geom="point", size=2) +
xlab('Loan_Status') + ylab('TotalIncome') +  ggtitle('TotalIncome vs. Loan_Status')
Loan_Prediction$InstallmentperMonth<-
((Loan_Prediction$LoanAmount*1000)/Loan_Prediction$Loan_Amount_Term)
Loan_Prediction<-Loan_Prediction[,c(-6,-7)]
ggplot(Loan_Prediction,aes(x=Loan_Status, y=InstallmentperMonth)) +
geom_boxplot(aes(fill = Loan_Status)) + stat_summary(fun.y = mean, geom="point",
size=2) + xlab('Loan_Status') + ylab('InstallmentperMonth') +
ggtitle('InstallmentperMonth vs. Loan_Status')
plot(Loan_Prediction$InstallmentperMonth, Loan_Prediction$TotalIncome,
        xlab="InstallmentperMonth",

ylab="TotalIncome",col=Loan_Prediction$Loan_Status,legend=row.names(Loan_Predict
ion$Loan_Status))
#creating dummy variables
Loan_Prediction$Urban<-0
Loan_Prediction$Semiurban<-0
Loan_Prediction$Rural<-0
for (i in 1:nrow(Loan_Prediction)){
  if(Loan_Prediction$Property_Area[i]=="Urban"){
    Loan_Prediction$Urban[i]<-1
  }
  if(Loan_Prediction$Property_Area[i]=="Semiurban"){
    Loan_Prediction$Semiurban[i]<-1
  }
  if(Loan_Prediction$Property_Area[i]=="Rural"){
    Loan_Prediction$Rural[i]<-1
  }
}

Loan_Prediction<-Loan_Prediction[,-7]
Loan_Prediction$Urban<-factor(Loan_Prediction$Urban,levels = c(0,1))
Loan_Prediction$Semiurban<-factor(Loan_Prediction$Semiurban,levels = c(0,1))
```

```
Loan_Prediction$Rural<-factor(Loan_Prediction$Rural,levels = c(0,1))

write.csv(Loan_Prediction,"Loan_Prediction_basic.csv")  ##Dataset for CART
##Below is the dataset for knn & NB
Loan_Prediction<-Loan_Prediction
Loan_Prediction$Gender<-ifelse(Loan_Prediction$Gender=="Male",1,0)
Loan_Prediction$Married<-ifelse(Loan_Prediction$Married=="Yes",1,0)
Loan_Prediction$Education<-ifelse(Loan_Prediction$Education=="Graduate",1,0)
Loan_Prediction$Self_Employed<-ifelse(Loan_Prediction$Self_Employed=="Yes",1,0)
Loan_Prediction$Credit_History<-as.numeric(Loan_Prediction$Credit_History)-1
Loan_Prediction$Dependents<-as.numeric(Loan_Prediction$Dependents)-1
View(Loan_Prediction)
#Normalizing data
scaled_LP<-scale(Loan_Prediction[,-9])
#partitioning into train & validation data
set.seed(1)
train.index<-sample(c(1:dim(Loan_Prediction)[1]),0.5*dim(Loan_Prediction)[1])
train.df<-Loan_Prediction[train.index,]
rem.df<-Loan_Prediction[-train.index,]
valid.index<-sample(c(1:dim(rem.df)[1]),0.3*dim(Loan_Prediction)[1])
valid.df<-rem.df[valid.index,]
test.df<-rem.df[-valid.index,]
#naiveBayes
install.packages("e1071")
library(e1071)
Loan_Prediction.nb<-naiveBayes(Loan_Status~.,data=train.df)
Loan_Prediction.nb
pred.prob.valid <- predict(Loan_Prediction.nb, newdata = valid.df, type = "raw")
pred.class.valid <- predict(Loan_Prediction.nb, newdata = valid.df,type = "class")
pred.prob.valid
pred.class.valid
df <- data.frame(actual = valid.df$Loan_Status, predicted = pred.class.valid,
pred.prob.valid)
View(df)
table(pred.class.valid, valid.df$Loan_Status)
# pred.test <- predict(Loan_Prediction.nb, newdata = test.df)
# table(pred.test, test.df$Loan_Status,positive = "Accepted")
#knn
install.packages("class")
library(class)
Actualtrainclass<-train.df$Loan_Status
Actualvalidclass<-valid.df$Loan_Status
knn1<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=1)
# confusionMatrix(knn1, valid.df$Loan_Status)
table(knn1,valid.df$Loan_Status)
knn3<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=3)
# confusionMatrix(knn3, valid.df$Loan_Status)
```

```
table(knn3,valid.df$Loan_Status)
knn5<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=5)
# confusionMatrix(knn5, valid.df$Loan_Status)
table(knn5,valid.df$Loan_Status)
knn7<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=7)
# confusionMatrix(knn7, valid.df$Loan_Status)
table(knn7,valid.df$Loan_Status)
knn9<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=9)
# confusionMatrix(knn9, valid.df$Loan_Status)
table(knn9,valid.df$Loan_Status)
knn11<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=11)
# confusionMatrix(knn11, valid.df$Loan_Status)
table(knn11,valid.df$Loan_Status)
knn13<-knn(train=train.df[,-7],test=valid.df[,-7],cl=Actualtrainclass,k=13)
# confusionMatrix(knn13, valid.df$Loan_Status)
table(knn13,valid.df$Loan_Status)


install.packages("rpart")
library(rpart)
install.packages("rpart.plot")
library(rpart.plot)
# install.packages("caret")
# library(caret)
setwd("C:/Users/nupur/OneDrive/Documents/Data Mining/use case study")
Loan_Prediction<-read.csv("Loan_Prediction_basic.csv")
Loan_Prediction$Credit_History<-factor(Loan_Prediction$Credit_History, levels =
c(0,1))
Loan_Prediction$Dependents<-factor(Loan_Prediction$Dependents, levels = c(0,1,2,3))
Loan_Prediction$Urban<-factor(Loan_Prediction$Urban,levels = c(0,1))
Loan_Prediction$Semiurban<-factor(Loan_Prediction$Semiurban,levels = c(0,1))
Loan_Prediction$Rural<-factor(Loan_Prediction$Rural,levels = c(0,1))
#Partitioning
set.seed(1)
train.index<-sample(c(1:dim(Loan_Prediction)[1]),0.5*dim(Loan_Prediction)[1])
train.df<-Loan_Prediction[train.index,]
rem.df<-Loan_Prediction[-train.index,]
valid.index<-sample(c(1:dim(rem.df)[1]),0.3*dim(Loan_Prediction)[1])
valid.df<-rem.df[valid.index,]
test.df<-rem.df[-valid.index,]
#Begin with small cp
CT1<-rpart(Loan_Status~.,data=train.df,method="class",control = rpart.control(cp =
0.0001))
CT1
printcp(CT1)
prp(CT1, type = 2, extra = 1, split.font = 1, varlen = -8)
summary(CT1)
```

```
# Step2: Pick the tree size that minimizes misclassification rate (i.e. prediction error).
# Prediction error rate in training data = Root node error * rel error * 100%
# Prediction error rate in cross-validation = Root node error * xerror * 100%
# Hence we want the cp value (with a simpler tree) that minimizes the xerror.
bestcp <- CT1$cptable[which.min(CT1$cptable[,"xerror"]),"CP"]
# Step3: Prune the tree using the best cp.
tree.pruned <- prune(CT1, cp = bestcp)
prp(tree.pruned, type = 2, extra = 1, split.font = 1, varlen = -8)
#display pruned tree
plot(tree.pruned)
text(tree.pruned, cex = 0.8, use.n = TRUE, xpd = TRUE)
#confusion matrix (training data)
conf.matrix <- table(train.df$Loan_Status, predict(tree.pruned,type="class"))
rownames(conf.matrix) <- paste("Actual", rownames(conf.matrix), sep = ":")
colnames(conf.matrix) <- paste("Pred", colnames(conf.matrix), sep = ":")
print(conf.matrix)
#Work on validation dataset
pred.valid<-predict(tree.pruned,valid.df,type="class")
conf.matrix.valid <- table(pred.valid,valid.df$Loan_Status)
rownames(conf.matrix.valid) <- paste("Actual", rownames(conf.matrix.valid), sep = ":")
colnames(conf.matrix.valid) <- paste("Pred", colnames(conf.matrix.valid), sep = ":")
print(conf.matrix.valid)
#Work on test dataset
pred.test<-predict(tree.pruned,test.df,type="class")
conf.matrix.test <- table(pred.test,test.df$Loan_Status)
rownames(conf.matrix.test) <- paste("Actual", rownames(conf.matrix.test), sep = ":")
colnames(conf.matrix.test) <- paste("Pred", colnames(conf.matrix.test), sep = ":")
print(conf.matrix.test)
```