Analysis using Gaussian Mixture Model

A001 Ved Deshpande
A002 Maithili Mithbavkar
A003 Shivprasad Choudhary
A004 Lakshmi Prasanna
A005 Pradnya Jagtap

Nilkamal School of Mathematics, Applied Statistics and Analytics

Contents

1 Introduction

In the modern times, statistics has become an essential field of study. The diverse application of statistics in field of biology, physics, commerce which includes business, banking institutions and many more has been very essential for the constant development of these fields. While looking at the applications of statistics in the field of finance, this project brings an insightful analysis on a given banking data-set to forecast the probability of default using Gaussian Mixture Model. Further using these probabilities we have also calculated expected loss and profit.

2 Problem Statement

Given a particular data-set, calculate the probability of loan default for individual customer using Gaussian Mixture Model and calculate the expected loss and gain by using those probabilities. Finding out the Factor importance and comparing our model with Support Vector Mechine

3 Theory

The Gaussian Mixture Model (GMM) stands as a fundamental and versatile statistical technique with a widespread application in the domain of finance. Its adaptability extends to various financial tasks, including risk assessment, fraud detection, credit scoring, and market risk management. Within the context of speaker recognition, GMMs play a pivotal role during training, generating prototypes to represent the feature space as a mixture of Gaussian distributions [4]. GMMs, renowned for their utility in clustering applications, provide a probabilistic association between data samples and clusters, unlike the binary classification approach of k-means clustering [1].

In the realm of consumer loan management, GMMs serve as a powerful tool for estimating the probability of default for individual customers. These models group customers into clusters based on feature similarities, enabling the assessment of bankruptcy probabilities within each cluster. This approach, outlined by Hamidreza Arian in "Forecasting Probability of Default for Consumer Loan Management with Gaussian Mixture Models" (2020), relies on GMMs' ability to discern the ratio of good and bad customers within each cluster [1]. Notably, GMMs employ the Expectation-Maximization (EM) algorithm to determine both the optimal number of clusters and the parameters of each cluster [2].

Expanding beyond finance, GMMs have also found applicability in stock market prediction and data generation. These models offer distinct advantages, particularly in modeling stock returns. With a sufficient number of components and careful fitting to training data using the Expectation–Maximization algorithm, GMMs can project future outcomes under various scenarios. Importantly, GMMs do not impose normality assumptions on return distributions when the number of components exceeds two [3].

In various domains, especially those involving data from multiple populations, Gaussian mixtures have been extensively employed as models. Speaker recognition systems, a prominent example, often utilize GMMs due to their capacity to represent diverse feature distributions. GMMs excel in shaping smooth estimations for arbitrarily shaped densities, with each component density potentially representing a distinct hidden class. This adaptability and ability to represent feature distributions have solidified GMMs' role in biometric systems, as detailed in Singh and colleagues' work on "Gaussian Mixture Model: A Modeling Technique for Speaker Recognition and its Component" [4].

Moreover, GMMs have demonstrated their utility in modeling wind parameters. The GMM's flexibility shines as it accurately estimates the 50-year wind parameter contour and aligns well with established standards like IEC 61400-1. This flexibility extends across various sectors of wind measurement data, showing promise in modeling the joint distribution of wind parameters. Notably, GMMs exhibit lower estimation error for marginal and conditional distributions compared to copula methods, as highlighted in Zhang and colleagues' research on "Gaussian mixture model for extreme wind turbulence estimation" [5].

In summary, Gaussian Mixture Models have proven to be versatile and powerful tools in various fields, including finance, speaker recognition, stock market prediction, and wind parameter modeling. Their ability to adapt to complex data distributions, estimate probabilities, and model diverse populations makes them a valuable asset in a wide range of applications. These models, often parameterized using the Expectation-Maximization algorithm, have significantly contributed to the advancement of research and practical solutions in these domains.

4 Formulas

4.1 Probability of Default

This equation gives a probability distribution of Gaussian Mixture Model(GMM), weighted summation of finite set of normal distributions.

$$p(x|\theta) = \sum_{i=1}^{N_c} \omega_i \phi(x|\mu_i, \Sigma_i)$$

where, x is the customers features data, Nc is the number of clusters, ω_i is the cluster weight,ϕ is the multivariate Gaussian/normal probability distribution,μ is the mean vector and the $\sum$ is the covariance matrix.

$$\phi(x|\mu_i, \Sigma_i) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp\{-\frac{1}{2}(x-\mu_i)^{'} \Sigma_i^{-1}(x-\mu_i)\} \tag{1}$$

4.2 Expected Loss

To calculate expected error we have made certain assumptions

- Recover rate is 0.5 for all customers
- Customers are randomly given a certain amount from normal distribution with mean 1000 and sd 100

$$Expectedloss = \sum PD_i * EAD_i * RR_i$$

Where, PD_i is the probability of default of individual customer, RR_i is the recovery rate, and EAD_i is the loan amount given to an individual(also known as Exposure at default)

4.3 Profit

To calculate profit we have made certain assumptions

- Recover rate is 0.5 for all customers
- The amount to be paid back is calculated using Simple Interest(for simplicity)

5 Algorithm

- Installing necessary packages in R
- Data cleaning and oversampling using SMOTE
- To find the optimum number of clusters we have used AIC and BIC method.
- Define a function for calculating probability of default with the z values obtained using Mclust package
- calculate the probability of default
- Using those calculations formulate Expected loss and make a function for the same
- Now make a function for profit calculation for each intent and then calculate profits
- Give the necessary visualization
- Find out the Important factor using Random Forest and show Variable importance chart
- Do the similar analysis using SVM and find the accuracy

6 Plots and observations

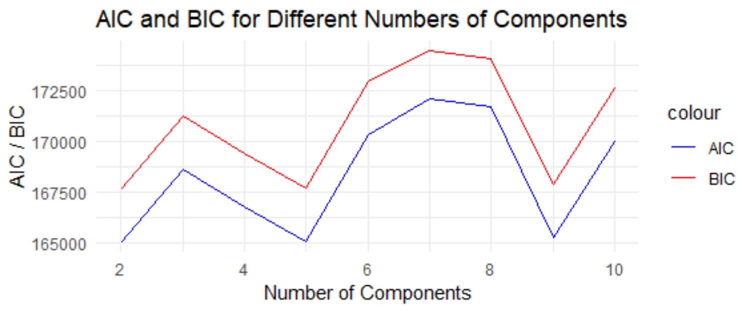

Figure 1: Optimum cluster plot

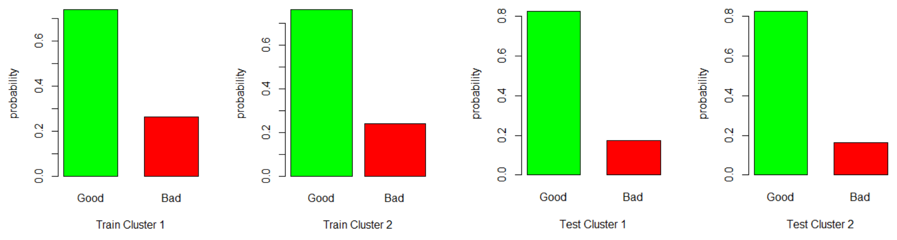

Figure 2: Cluster Probability Plot

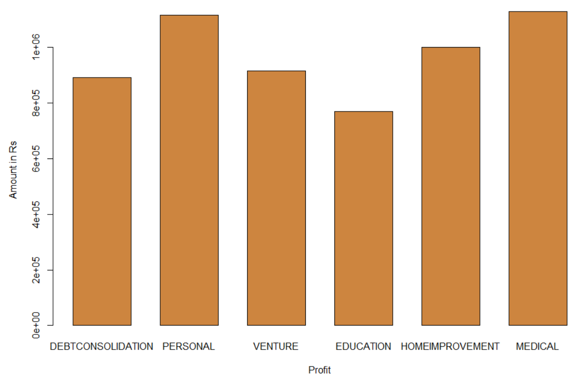

Figure 3: Profit Plot(Intent Wise)

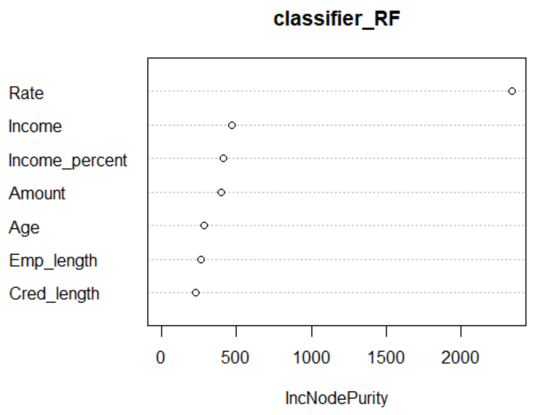

Figure 4: Factor Plot

7 Tables

Expected Loss for Train Dataset

Total Capital	Expected Loss	Actual Loss	Error percentage
2,53,22,603	8,51,298	7,97,711	6.294799%

Expected Loss for Test Dataset

Total Capital	Expected Loss	Actual Loss	Error percentage
97,66,162	1,47,147	1,37,915	6.274146%

8 Result and Conclusion

We successfully calculated the expected loss and the profits using the probability of default given by the Gaussian Mixture Model and we further calculated the accuracy of our model which was around 73% . The data was highly imbalanced due to which the accuracy might have taken a hit. The similar analysis was conducted using Support Vector Mechine and we found out that it was more accurate than our Gaussian Mixture model with an accuracy of 82.19%.

Looking at the business aspect, we should get more insights on the data to predict the default more precisely and choose not to give loan to such defaulters. We can draw a threshold probability line below which if a customer lies, will not be given a loan. We can see that the factors affecting the most among them is Rate of interest, so we can come up with a solution to find optimal interest rates by which even the banks won't lose their profits and at the same time the customer would be able to repay the loan amount.

9 References

- Hamidreza Arian, S. M. (2019, August). A Novel Classification Approach for Credit Scoring.
- Hamidreza Arian, S. M. (2020, october). Forecasting Probability of Default for Consumer Loan Management with Gaussian Mixture Models.
- Seyfi, S. M. (2021). Portfolio Value-at-Risk and expected-shortfall using an efficient simulation approach based on Gaussian Mixture Model. Mathematical and computer simulations, 1056-1079.
- Singh, N. . (2014). Gaussian Mixture Model: A Modeling Technique for Speaker Recognition and its Component.
- Zhang, X. . (2022). Gaussian mixture model for extreme wind turbulence estimation.

Appendix

Data Cleaning

```
#importing the data
Data=read.csv("D:\\MSc(ASA)\\Sem1\\prOJECT\\Data Set\\credit_risk.csv")
summary(Data)
View(Data)

#changing the missing value rows with mean
library(dplyr)
which(is.na(Data))
mean.data=Data %>% group_by(Data$Intent) %>%
  summarise(Mean=mean(Data$Rate, na.rm = T))
Emp=Data %>%
  group_by(Home, Class) %>%
  summarise(Max=max(Data$Emp_length, na.rm=T),.groups = "keep")
#changing the na column with the mean
nrow(Data)
Data$Rate=ifelse(is.na(Data$Rate),11,Data$Rate)
Data$Emp_length=ifelse(is.na(Data$Emp_length)&(Data$Class==0),54,
                      ifelse(is.na(Data$Emp_length)
                             &(Data$Class==1),46.25,Data$Emp_length))

#checking for Na values
which(is.na(Data))


#looking at the frequency of default and not default/good or
#bad customer
table(Data$Default)

#Converting the target column default into Good or Bad
Data$Default=ifelse(Data$Default=="Y","Bad","Good")

#removing the character columns
Data=subset(Data,select=-c(Home,Intent))


#Scaling the income and amount column
Vec1=((Data$Amount)-min(Data$Amount))/(max(Data$Amount)-min(Data$Amount))
Vec2=((Data$Income)-min(Data$Income))/(max(Data$Income)-min(Data$Income))
Vec3=((Data$Percent_income)-min(Data$Percent_income))/
  (max(Data$Percent_income)-min(Data$Percent_income))

Data=cbind(Data,Vec1,Vec2,Vec3)
```

```
View(Data)
Data=Data[,-c(1,3,5,7,8)]
Data=Data[,c(1,2,3,5,6,7,8,4)]


#Changing the column names to scaled amount and scaled income
colnames(Data)[c(5,6,7)]=c("Amount","Income","Income_percent")
summary(Data)

#there is a problem with the age column we have to fix it
View(table(Data$Age))
which(Data$Age>100)
Data$Age[c(82,184,576,748,32298)]=c(44,44,23,23,44)
Data$Age=round(Data$Age,0)
Data$Emp_length=round(Data$Emp_length,0)

#our target column is default column and rest are features

#now we will split the dagta into train and test
x=sort(sample(nrow(Data),nrow(Data)*0.7))
Train_Data=Data[x,]
Test_Data=Data[-x,]

#we have to see the ratio of default to non default
library(imbalance)
nrow(subset(Data,Data$Default=="Good"))/nrow(Data)
imbalanceRatio(Data,classAttr = "Default")

#this means that we have to bring a
#balance ratio of 35-75 and therefore we have to oversample the data
Train_Data=oversample(Train_Data,ratio=0.35,method="SMOTE", classAttr = "Default
imbalanceRatio(Train_Data,classAttr = "Default")
table(Train_Data$Default)

#now our data is fairly balanced and we can finally use

#Train data cleaning
Train_Data$Age=round(Train_Data$Age,0)
Train_Data$Emp_length=round(Train_Data$Emp_length,0)
View(Train_Data)
Train_Data$Age=round(Train_Data$Age,0)
Train_Data$Emp_length=round(Train_Data$Emp_length,0)
Train_Data$Emp_length=ifelse(Train_Data$Emp_length==1,
    mean(Train_Data$Emp_length),Train_Data$Emp_length)
```

Model and Expected loss and Profits

```
set.seed(50)
library(mclust)
library(ggplot2)

K=seq(2,10,1)
AIC=numeric()
BIC=numeric()
dummy_data=Train_Data
View(dummy_data[,c(1:8)])

for(i in K){
  kmeans_model=Mclust(dummy_data[,1:7],centers=i)
  AIC=c(AIC,AIC(kmeans_model))
  BIC=c(BIC,BIC(kmeans_model))
}
unique(dummy_data)

df=data.frame(Components = K, AIC = AIC, BIC = BIC)
# Find the optimal number of components based on minimum AIC and BIC

optimal_components_aic=K[which.min(AIC)];optimal_components_aic
optimal_components_bic=K[which.min(BIC)];optimal_components_bic
View(df)



# making the Mclust
Gmm_model=Mclust(dummy_data[,c(1:7)],2)
table(dummy_data$Default,Gmm_model$classification)
summary(Gmm_model)


?density

#to find the maximum of a row
Probability_Dataframe$Max=pmax(Probability_Dataframe$V1,
                               Probability_Dataframe$V2)
View(Probability_Dataframe)
#making an empty column to get the max value column
Column_for_cluster=numeric()
Probability_Dataframe=Probability_Dataframe[,-4]
Column_for_cluster=
  ifelse(Probability_Dataframe$V1==Probability_Dataframe$Max,
                        append(Column_for_cluster,1),
                        ifelse
        (Probability_Dataframe$V2==Probability_Dataframe$Max,
          append(Column_for_cluster,2),0))


#Creating a dataframe column for it and binding it with the original dataframe
data1=data.frame(Column1=Column_for_cluster)
Probability_Dataframe=cbind(Probability_Dataframe,data1$Column1)
View(Probability_Dataframe)
View(dummy_data)
#adding the cluster column to dummy data column
dummy_data=cbind(dummy_data,data1$Column1)
colnames(dummy_data)[10]="Cluster Number"

#Dividing the dataframe according to the clusters
C_1=subset(dummy_data,dummy_data$`Cluster Number`==1)
C_2=subset(dummy_data,dummy_data$`Cluster Number`==2)
View(C_1)
#Looking at the default and non-default
table(C_1$Default)
table(C_2$Default)

#Calculating probability
Probability_good=function(x,y){
  Probability=x/y
}
P1=Probability_good(length(which(C_1$Default=="Good")),nrow(C_1));P1
P2=Probability_good(length(which(C_2$Default=="Good")),nrow(C_2));P2

#calculating Ratio
R1=length(which(C_1$Default=="Good"))/length(which(C_1$Default=="Bad"))
R2=length(which(C_2$Default=="Good"))/length(which(C_2$Default=="Bad"))

#Calculating Weight
W1=nrow(C_1)/nrow(dummy_data)
W2=nrow(C_2)/nrow(dummy_data)
sum(W1,W2)

#to check the test data
Dummy_test=Test_Data
predictions=predict.Mclust(Gmm_model,newdata = Test_Data[,1:7]);
predictions$classification
Test_Dataframe=data.frame(predictions$z)
View(Test_Dataframe)
predictions$z

Test_Dataframe$Max=pmax(Test_Dataframe$X1,Test_Dataframe$X2)
```

```
#making an empty column to get the max value column
Column_for_cluster=numeric()

Column_for_cluster=ifelse(Test_Dataframe$X1==Test_Dataframe$Max,
                          append(Column_for_cluster,1),
                          ifelse(Test_Dataframe$X2==Test_Dataframe$Max,
                                 append(Column_for_cluster,2),0))

#Creating a dataframe column for it and binding it with the original dataframe
data1.0=data.frame(Column1=Column_for_cluster)
Test_Dataframe=cbind(Test_Dataframe,data1.0$Column1)

#adding the cluster column to dummy data column
Dummy_test=cbind(Dummy_test,data1.0$Column1)
colnames(Dummy_test)[10]="Cluster Number"
Dummy_test=Dummy_test[,-10]
View(Dummy_test)

#Dividing the dataframe according to the clusters
T_1=subset(Dummy_test,Dummy_test$`Cluster Number`==1)
T_2=subset(Dummy_test,Dummy_test$`Cluster Number`==2)
View(T_2)

#Looking at the default and non-default
table(T_1$Default)
table(T_2$Default)

PT1=Probability_good(length(which(T_1$Default=="Good")),nrow(T_1))
PT2=Probability_good(length(which(T_2$Default=="Good")),nrow(T_2))

#calculating Ratio
RT1=length(which(T_1$Default=="Good"))/length(which(T_1$Default=="Bad"))
RT2=length(which(T_2$Default=="Good"))/length(which(T_2$Default=="Bad"))

#Calculating Weight
WT1=nrow(T_1)/nrow(Dummy_test)
WT2=nrow(T_2)/nrow(Dummy_test)
sum(W1,W2)

t1=as.data.frame(table(C_1$Default))
t2=as.data.frame(table(C_2$Default))

#calculating Probabilities based on test dataset
PT1=Probability_good(length(which(T_1$Default=="Good")),nrow(T_1))
PT2=Probability_good(length(which(T_2$Default=="Good")),nrow(T_2))
```

```
#calculating Ratio
RT1=length(which(T_1$Default=="Good"))/length(which(T_1$Default=="Bad"))
RT2=length(which(T_2$Default=="Good"))/length(which(T_2$Default=="Bad"))

#Calculating Weight
WT1=nrow(T_1)/nrow(Dummy_test)
WT2=nrow(T_2)/nrow(Dummy_test)
sum(W1,W2)

t1=as.data.frame(table(C_1$Default))
t2=as.data.frame(table(C_2$Default))

Cluster1_PD=data.frame(Category=c("Good","Bad",
                                  "Good","Bad","Good",
                                  "Bad","Good","Bad"),
                       Probability_of_default=c(P1,1-P1,
                                                P2,1-P2,PT1,
                                                1-PT1,PT1,1-PT2))

library(dplyr)
Master_table=cbind(t1,t2$Freq,t3$Freq,t4$Freq);Master_table



ggplot(Master_table,aes(x=factor(Var1),y=Freq))+
  geom_col(color='black',fill='cyan3')+
  xlab('Response')


C_1D=subset(C_1,C_1$Default=="Bad")
C_2D=subset(C_2,C_2$Default=="Bad")
View(C_1D)
T_1D=subset(T_1,T_1$Default=="Bad")
T_2D=subset(T_2,T_2$Default=="Bad")


Eloss=function(P1,Amount){
  loss=(1-P1)*Amount*(1-0.5)
  return(loss)
}

#for Train
C_1D$ElossR=NA
C_2D$ElossR=NA

set.seed(1)
```

```
for(i in 1:nrow(C_1D)){
  C_1D$ElossR[i]=Eloss(P1,rnorm(1,1000,100))
}
set.seed(1)
for(i in 1:nrow(C_2D)){
  C_2D$ElossR[i]=Eloss(P2,rnorm(1,1000,100))
}
e1=sum(C_1D$ElossR)
e2=sum(C_2D$ElossR)
Total_exploss=sum(C_1D$ElossR)+sum(C_2D$ElossR)
Total_earning=nrow(C_1)*1000-Total_exploss
View(C_2D)

#For test
T_1D$ExplossR=NA
T_2D$ExplossR=NA

View(T_1D)
set.seed(1)
for(i in 1:nrow(T_1D)){
  T_1D$ExplossR[i]=Eloss(PT1,rnorm(1,1000,100))
}
set.seed(1)
for(i in 1:nrow(T_2D)){
  T_2D$ExplossR[i]=Eloss(PT2,rnorm(1,1000,100))
}

et1=sum(T_1D$ElossR)
et2=sum(T_2D$ElossR)

View(T_1D)

##Random forest for factors
# Loading package
library(caTools)
library(randomForest)

# Fitting Random Forest to the train dataset
View(Train_Data)
set.seed(120)  # Setting seed
trainmodified=Train_Data

trainmodified$Default=ifelse(trainmodified$Default=="Good",0,1)
classifier_RF = randomForest(x = trainmodified[-8],
                             y = trainmodified$Default,
                             ntree = 1000)
```

```
varImpPlot(classifier_RF)

# Predicting the Test set results
y_pred = predict(classifier_RF , newdata = Train_Data[-10])


?predict()
classifier_RF = randomForest(factor(Default~.),Train_Data)
classifier_RF$mtry



#Visualization Part
#plotting the AIC and BIC Values for optimum number of clusters
ggplot(df,  aes(x = Components)) +
  geom_line(aes(y = AIC, color = "AIC")) +
  geom_line(aes(y = BIC, color = "BIC")) +
  labs(title = "AIC and BIC for Different Numbers of Components",
       x = "Number of Components",
       y = "AIC / BIC") +
  scale_color_manual(values = c("AIC" = "blue", "BIC" = "red")) +
  theme_minimal()

par(mfrow=c(1,2))
barplot(Cluster1_PD[1:2,2],space=0.5,width=1,
        xlab="Train Cluster 1",ylab="probability",
        names.arg =c("Good","Bad"),col =c("green","red") )
barplot(Cluster1_PD[3:4,2],space=0.5,
        xlab="Train Cluster 2",ylab="probability",
        names.arg =c("Good","Bad"),col =c("green","red"))
barplot(Cluster1_PD[5:6,2],space=0.5,width=0.1,
        xlab="Test Cluster 1",ylab="probability",
        names.arg =c("Good","Bad"),col =c("green","red"))
barplot(Cluster1_PD[7:8,2],space=0.5,
        width=0.1,xlab="Test Cluster 2",ylab="probability",
        names.arg =c("Good","Bad"),col =c("green","red"))


#Expected Loss from main data and model data
probtrain=subset(Train_Data,Train_Data$Default=="Bad")
probtest=subset(Test_Data,Test_Data$Default=="Bad")

View(probtrain)
View(probtest)
prob=nrow(probtrain)/nrow(Train_Data)
prob1=nrow(probtest)/nrow(Test_Data)
```

```
set.seed(1)
ExplossTrain1=Eloss(1-prob,rnorm(1,1000,100))*nrow(probtrain)
set.seed(1)
ExplossTest1=Eloss(1-prob1,rnorm(1,1000,100))*nrow(probtest)

e1=sum(C_1D$ElossR)+sum(C_2D$ElossR)
et1=sum(T_1D$ExplossR)+sum(T_2D$ExplossR)

T_1D$ExplossR=NA
T_2D$ExplossR=NA

View(T_1D)
set.seed(1)
for(i in 1:nrow(T_1D)){
  T_1D$ExplossR[i]=Eloss(PT1,rnorm(1,1000,100))
}
set.seed(1)
for(i in 1:nrow(T_2D)){
  T_2D$ExplossR[i]=Eloss(PT2,rnorm(1,1000,100))
}


set.seed(1)
sum=NA
for(i in 1:nrow(T_2)){
  a=rnorm(1,1000,100)
  sum[i]=a
}

nrow(Train_Data)

sum(sum)
Loss_Dataframe=data.frame(Exp_lossTrain=e1,
                          ExplossTest=et1,
                          ActualLossTrain=ExplossTrain1,
                          ActualLossTest=ExplossTest1,
                          ErrorPercTrain=(e1-ExplossTrain1)/(e1)*100,
                          ErrorPercTest=(et1-ExplossTest1)/(et1)*100)


#Model Accuracy
install.packages("caret")
library("caret")

Train_Data$Default_Probability=NA
```

```
for(i in 1:nrow(Train_Data)){
  Train_Data$Default_Probability[i]=sum(Gmm_model$z[i,]*c(1-P1,1-P2))
}

Test_Data=Test_Data[,-9]

for(i in 1:nrow(Test_Data)){
  Test_Data$Default_Probability[i]=sum(GMM_test$z[i,]*c(1-PT1,1-PT2))
}

confusionMatrix(as.factor(ifelse(predictions$classification==1,
                                 'Good','Bad')),
                as.factor(Test_Data$Default))
ifelse(predictions$classification==1,"Good","Bad")
table(ifelse(predictions$classification==1,"Good","Bad"))
table(Test_Data$Default)

#importing the data
Credit_score=read.csv("D:\\MSc(ASA)\\Sem1\\prOJECT\\Data Set\\
                      credit_risk.csv")
View(Cred_Score)

#changing the missing value rows with mean
library(dplyr)
mean.data=Credit_score %>% group_by(Credit_score$Intent) %>%
  summarise(Mean=mean(Credit_score$Rate, na.rm = T))
Emp=Credit_score %>% group_by(Home,Class) %>%
  summarise(Max=max(Credit_score$Emp_length, na.rm=T),.groups = "keep")
#changing the na column with the mean
nrow(Data)
Credit_score$Rate=ifelse(is.na(Credit_score$Rate),11,Credit_score$Rate)
Credit_score$Emp_length=ifelse(is.na(Credit_score$Emp_length)&
                               (Credit_score$Class==0),54,
                       ifelse(is.na(Credit_score$Emp_length)&
                              (Credit_score$Class==1),46.25,
                             Credit_score$Emp_length))

summary(Credit_score$Cred_length)


#Joining it with the main table
New_Cred=Credit_score

######Intent 1
Intent1=subset(New_Cred,New_Cred$Intent=="DEBTCONSOLIDATION")
```

```
Intent2=subset(New_Cred,New_Cred$Intent=="PERSONAL")
Intent3=subset(New_Cred,New_Cred$Intent=="VENTURE")
Intent4=subset(New_Cred,New_Cred$Intent=="EDUCATION")
Intent5=subset(New_Cred,New_Cred$Intent=="HOMEIMPROVEMENT")
Intent6=subset(New_Cred,New_Cred$Intent=="MEDICAL")

par(mfrow=c(2,3))

ABC=function(X){
  X_P=length(which(X$Default=="Y"))/nrow(X)
  X$SI=NA
  X$Expected_return=NA
  for(i in 1:nrow(X)){
    S.I=(X$Amount[i]*X$Rate[i]*X$Cred_length[i]/(12*100))
    X$SI[i]=(X$Amount[i]+S.I)
  }

  for(i in 1:nrow(X)){
    ExpRet=(X$SI[i]*X_P*0.8)+(X$SI[i]*(1-X_P))
    X$Expected_return[i]=ExpRet
  }
  X$Profit=X$Expected_return-X$Amount
  Profit6=sum(X$Profit)
  hist(X$Profit,main = paste("Histogram of",X$Intent[1]),xlab="Profit")
  return(X)

}
Intent1=ABC(Intent1)
Intent2=ABC(Intent2)
Intent3=ABC(Intent3)
Intent4=ABC(Intent4)
Intent5=ABC(Intent5)
Intent6=ABC(Intent6)
par(mfrow=c(1,1))

profitDf=data.frame(Category=c("DEBTCONSOLIDATION","PERSONAL",
                               "VENTURE",
                               "EDUCATION",
                               "HOMEIMPROVEMENT",
                               "MEDICAL"),
                    Profits=c(sum(Intent1$Profit),
                              sum(Intent2$Profit),
                              sum(Intent3$Profit),
                              sum(Intent4$Profit),
                              sum(Intent5$Profit),
                              sum(Intent6$Profit)))
```

```
Category=data.frame(Category=c("DEBTCONSOLIDATION"=859886.5,
                               "PERSONAL"=1148443.9,
                               "VENTURE"=1148443.9,
                               "EDUCATION"=841113.4,
                               "HOMEIMPROVEMENT"=867121.0,
                               "MEDICAL"=1145454.6))
barplot(profitDf[,2],
        ylimit=c(0,10*10^8),
        col = c("red","blue","green","orange","violet","purple"),
        xlab = "Profit",
        ylab="Amount in Rs",
        names.arg = c("DEBTCONSOLIDATION",
                      "PERSONAL","VENTURE","EDUCATION",
                      "HOMEIMPROVEMENT","MEDICAL"),space=0.5)
sum(Intent6$Profit)-sum(Intent6$Profit)


Cor=cor(Train_Data[,1:7],method = "kendall")


library(corrplot)
corrplot(Cor)



profitDf=data.frame(Category=c("DEBTCONSOLIDATION",
                               "PERSONAL","VENTURE",
                               "EDUCATION","HOMEIMPROVEMENT"
                               ,"MEDICAL"),
                    Profits=c(sum(Intent1$Profit),
                              sum(Intent2$Profit),
                              sum(Intent3$Profit),
                              sum(Intent4$Profit),
                              sum(Intent5$Profit),
                              sum(Intent6$Profit)))
Category=data.frame(Category=c("DEBTCONSOLIDATION"=859886.5,
                               "PERSONAL"=1148443.9,
                               "VENTURE"=1148443.9,
                               "EDUCATION"=841113.4,
                               "HOMEIMPROVEMENT"=867121.0,
                               "MEDICAL"=1145454.6))
barplot(profitDf[,2],width=0.6,
        ylimit=c(0,10*10^8),
        col = c("tan3"),
        xlab = "Profit",
        ylab="Amount in Rs",
        names.arg = c("DEBTCONSOLIDATION",
                      "PERSONAL","VENTURE",
                      "EDUCATION","HOMEIMPROVEMENT"
                      ,"MEDICAL"),
        space=0.5)
```