**Visualisation, clustering and classification of**

**Raipur Road Accident Data**

**For Year 2019 & 2020 by using R**

**#Import library**

library(lubridate)

library(scales)

library(ggplot2)

library(dplyr)

require(ggplot2)

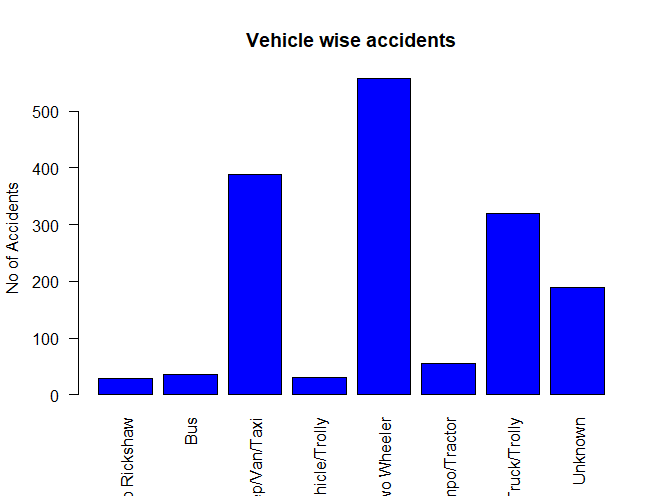
**#Load data**

ds <- read.csv(file.choose())# **Select file Raipur\_Accident\_Data\_Visua.csv**

count\_Vh <- table(ds$Vehicle)

**#Plot bar chart according to vehicle wise**

barplot(count\_Vh, main="Vehicle wise accidents", las=2, ylab = "No of Accidents",col="blue")



**#National, State highway or Other road type Road Accident**

count\_RT <- table(ds$Road.Type)

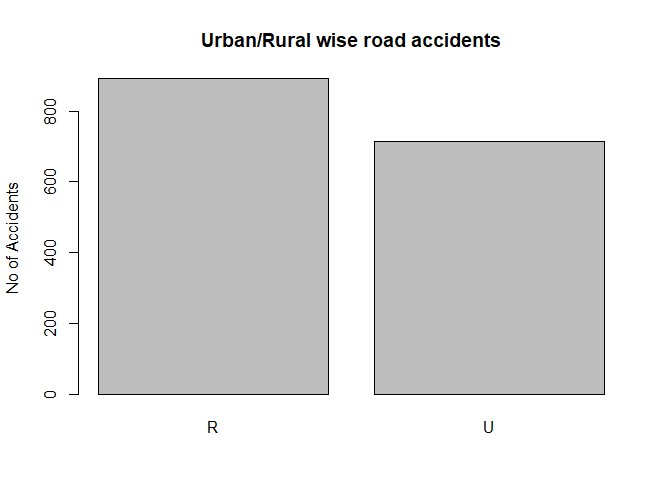
barplot(count\_RT, main="Road wise accidents", ylab = "No of Accidents",col="lightblue")



**#Urban/Rural wise Road Accident**

count\_Place <- table(ds$Place)

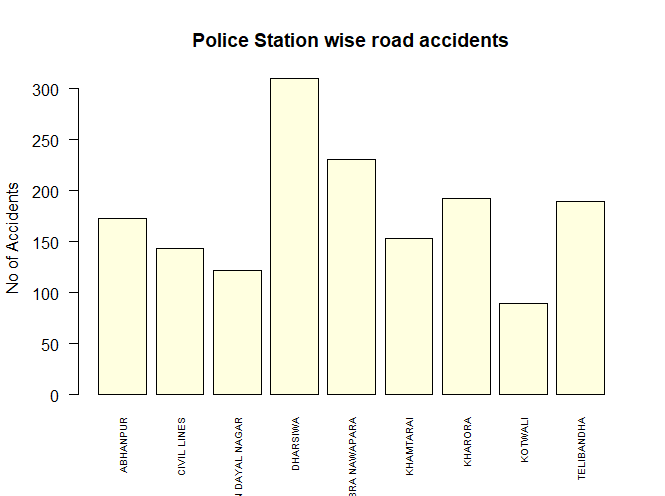
barplot(count\_Place, main="Urban/Rural wise road accidents", ylab = "No of Accidents",col="gray")



**#Police Station wise Road Accident**

count\_PS <- table(ds$ï..Police\_Station)

barplot(count\_PS, main="Police Station wise road accidents",las=2, ylab = "No of Accidents",col="lightyellow",cex.names = 0.6)



**# Data manipulation**

ds$Date = as.POSIXct(ds$Date, format="%m/%d/%Y")

ds$Year = year(ds$Date)

ds$Month = month(ds$Date)

ds$Day = day(ds$Date)

ds$Time = as.POSIXct(ds$Time, format="%H:%M")

ds$Hour = hour(ds$Time)

ds$Minute = minute(ds$Time)

**# Time series**

daily = group\_by(ds, Date)

day\_counts = summarise(daily, count = n())

library(ggplot2)

**# average counts per hour**

daily\_group = group\_by(ds, Month, Day, Hour)

day\_hour\_counts = summarise(daily\_group, count = n())

hour\_group = group\_by(day\_hour\_counts, Hour)

hour\_avg\_counts = summarise(hour\_group, count = mean(count))

**# time series: average counts by time of day**

ggplot(hour\_avg\_counts, aes(x = Hour, y = count)) + geom\_point(colour = "red") +

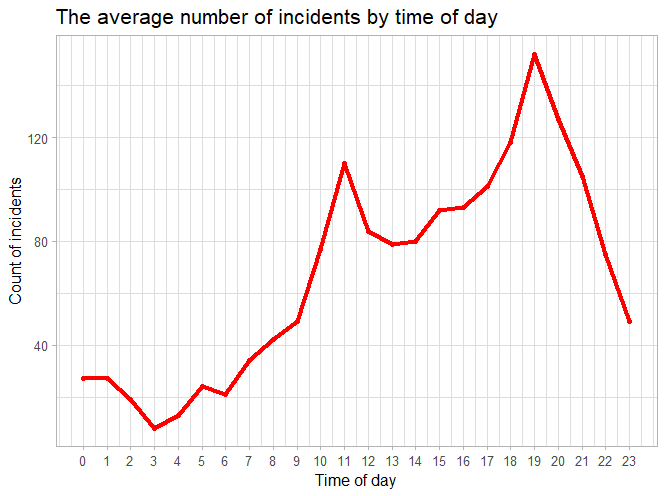
geom\_line(colour = "red", size = 1.5) +

theme\_light(base\_size = 12) + xlab("Time of day") + ylab("Count of incidents") +

scale\_x\_continuous(breaks=c(0:23)) +

ggtitle("The average number of incidents by time of day") +

theme(plot.title = element\_text(size = 16))



**# Histogram: average counts by time of day**

ggplot(hour\_avg\_counts, aes(x = Hour, y = count)) +

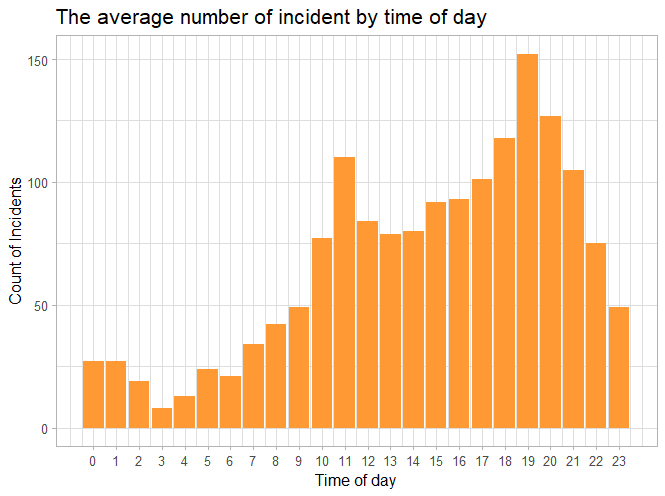
geom\_bar(position = "dodge", stat = "identity", fill = "#FF9933") +

theme\_light(base\_size = 12) + labs(x = "Time of day", y = "Count of Incidents") +

scale\_x\_continuous(breaks=c(0:23)) +

ggtitle("The average number of incident by time of day") +

theme(plot.title = element\_text(size = 16))



hourly\_group = group\_by(ds, Road.Type, Month, Day, Hour)

Location\_day\_hour\_counts = summarise(hourly\_group, count = n())

Location\_hourly\_group = group\_by(Location\_day\_hour\_counts, Road.Type, Hour)

Location\_hour\_avg\_counts = summarise(Location\_hourly\_group, count = mean(count))

**#Accident in NH, SH and other road type by time**

ggplot(Location\_hour\_avg\_counts, aes(x = Hour, y = Road.Type)) +

geom\_tile(aes(fill = count)) +

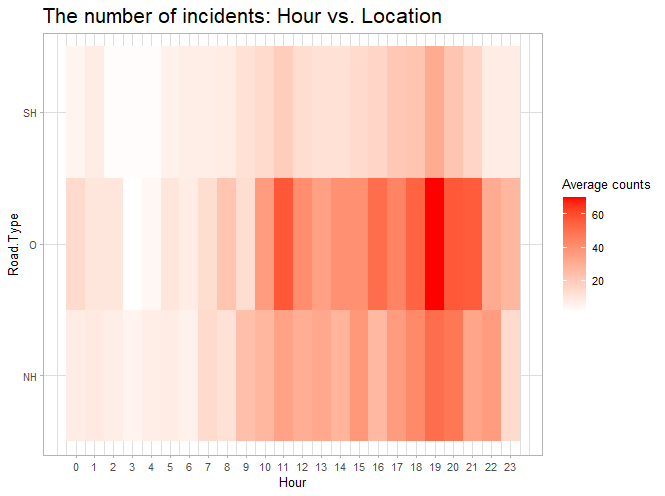
scale\_fill\_gradient(name = "Average counts", low = "white", high = "red") +

scale\_x\_continuous(breaks=c(0:23)) +

theme(axis.title.y = element\_blank()) + theme\_light(base\_size = 10) +

theme(plot.title = element\_text(size=16)) +

ggtitle("The number of incidents: Hour vs. Location")



hourly\_group = group\_by(ds, ï..Police\_Station, Month, Day, Hour)

Police\_Station\_day\_hour\_counts = summarise(hourly\_group, count = n())

Police\_Station\_hourly\_group = group\_by(Police\_Station\_day\_hour\_counts, ï..Police\_Station, Hour)

Police\_Station\_hour\_avg\_counts = summarise(Police\_Station\_hourly\_group, count = mean(count))

**#Police Station wise number of accident by Time**

ggplot(Police\_Station\_hour\_avg\_counts, aes(x = Hour, y = ï..Police\_Station)) +

geom\_tile(aes(fill = count)) +

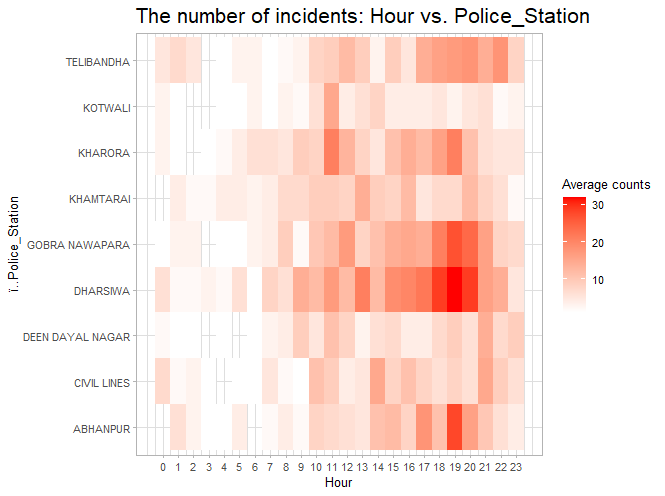
scale\_fill\_gradient(name = "Average counts", low = "white", high = "red") +

scale\_x\_continuous(breaks=c(0:23)) +

theme(axis.title.y = element\_blank()) + theme\_light(base\_size = 10) +

theme(plot.title = element\_text(size = 16)) +

ggtitle("The number of incidents: Hour vs. Police\_Station")



Location\_group = group\_by(ds, Month, Day, ï..Police\_Station, Road.Type)

day\_Police\_Station\_Location\_counts = summarise(Location\_group, count = n())

Police\_Station\_Location\_group = group\_by(day\_Police\_Station\_Location\_counts,ï..Police\_Station, Road.Type)

Police\_Station\_Location\_avg\_counts = summarise(Police\_Station\_Location\_group, count = mean(count))

**#Police Station and Road wise accident**

ggplot(Police\_Station\_Location\_avg\_counts, aes(x = ï..Police\_Station, y =Road.Type)) +

geom\_tile(aes(fill = count)) +

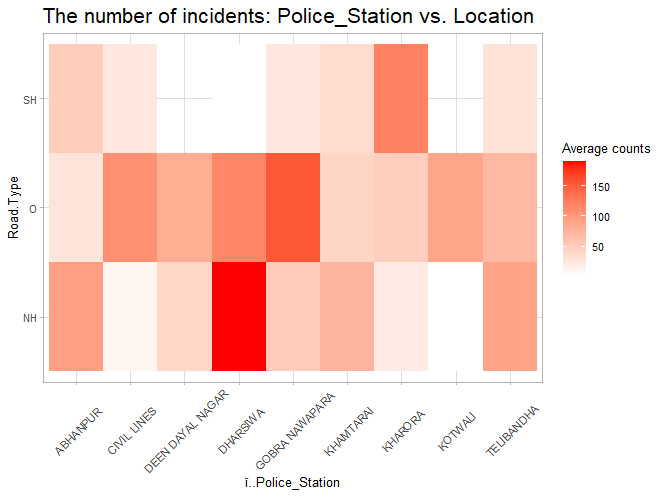
scale\_fill\_gradient(name="Average counts", low="white", high="red") +

theme(axis.title.y = element\_blank()) + theme\_light(base\_size = 10) +

theme(plot.title = element\_text(size = 16)) +

ggtitle("The number of incidents: Police\_Station vs. Location") +

theme(axis.text.x = element\_text(angle = 45,size = 8, vjust = 0.5))

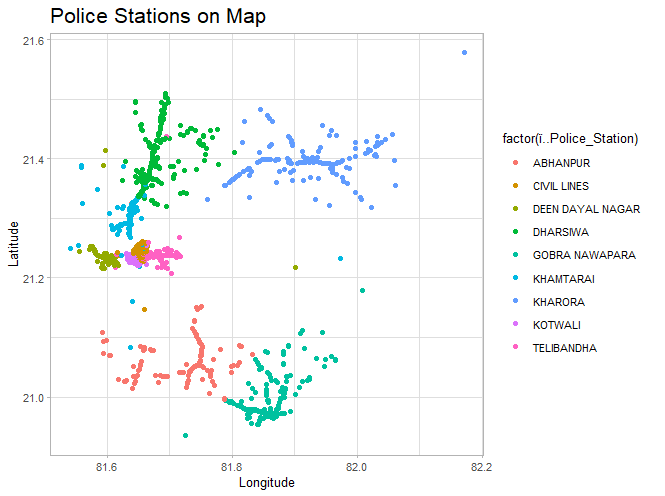


**# scatter plot**

ggplot(ds, aes(x = Longitude, y = Latitude)) + geom\_point(aes(colour = factor(ï..Police\_Station)), size = 1.25) +

theme\_light(base\_size = 10) + xlab("Longitude") + ylab("Latitude") +

ggtitle("Police Stations on Map") + theme(plot.title=element\_text(size = 16))

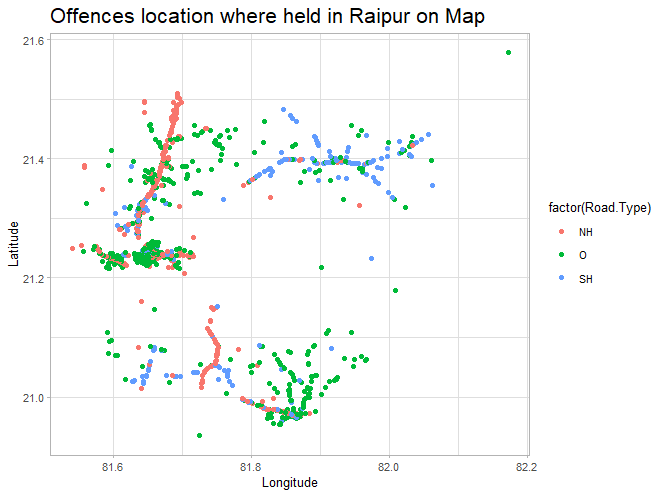


**# Accident location where held in Raipur on Map**

ggplot(ds, aes(x = Longitude, y = Latitude)) + geom\_point(aes(colour = factor(Road.Type)), size = 1.25) +

theme\_light(base\_size = 10) + xlab("Longitude") + ylab("Latitude") +

ggtitle("Accident location where held in Raipur on Map") + theme(plot.title=element\_text(size = 16))



################### **VISUALIZATION ---MAPPING**----########################

**# insert map layers to base map**

library(leaflet)

library(readr)

m <- leaflet(ds) %>%

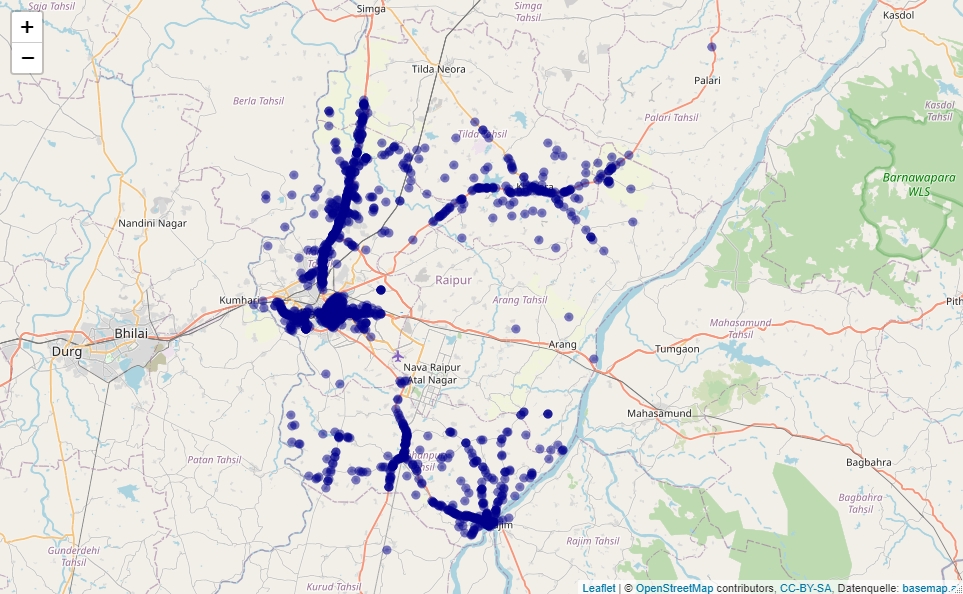
addTiles() %>% # Add default Open Street Map map tiles

addCircleMarkers(radius = 2, color = "darkblue",

lng = ~Longitude, lat = ~Latitude, popup= ~paste('<b>', `ï..Police\_Station`)) %>%

addProviderTiles(providers$BasemapAT.grau)

m

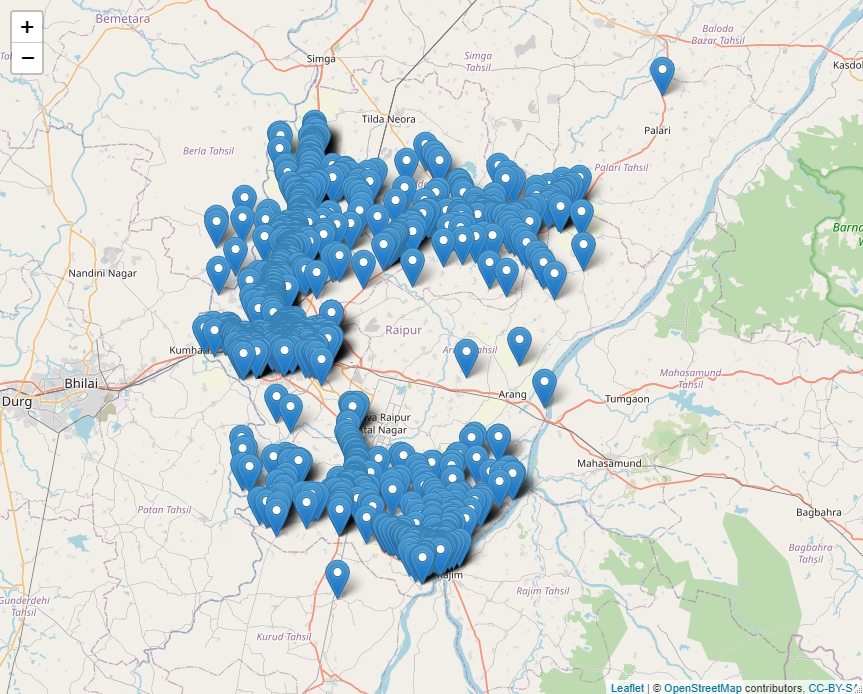


m <- leaflet(ds) %>%

addTiles() %>% # Add default Open Street Map map tiles

addMarkers(lng = ~Longitude, lat = ~Latitude, popup= ~paste('<b>', `ï..Police\_Station`))

m



###########**Split data**######################

library(lubridate)

library(scales)

library(ggplot2)

library(dplyr)

require(ggplot2)

library(leaflet)

library(caTools)

library(rpart.plot)

library(randomForest)

library(rpart.plot)

library(caret)

library(readr)

library(urca)

split <- sample.split(ds, SplitRatio = 0.70)# **we spliting the data in train test from 70% and 30%.**

split

train <- subset(ds, split == "TRUE")

test <- subset(ds, split == "FALSE")

str(train)# check the structure of train

str(test)# check the structure of test

#################### **Logistic Regression** #######################

**#Train model with logistic regression using glm function for ACT\_279.304.A**

logit\_model <- glm(ACT\_279.304.A~Latitude+Longitude, data = train, family = "binomial")

logit\_model

summary(logit\_model)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.7181 -0.6411 -0.5838 -0.5203 2.2333

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -349.9019 54.8619 -6.378 1.80e-10 \*\*\*

Latitude 3.1739 0.5291 5.998 1.99e-09 \*\*\*

Longitude 3.4395 0.6527 5.270 1.37e-07 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1128.1 on 1088 degrees of freedom

Residual deviance: 1065.8 on 1086 degrees of freedom

AIC: 1071.8

Number of Fisher Scoring iterations: 4

**#Predict test data based on trained model**

fitted\_result <- predict(logit\_model, test, type = "response")

fitted\_result

fitted\_result <- ifelse(fitted\_result> 0.5,1,0)

fitted\_result #Predict Result

test$ACT\_279.304.A #Actual Result

**#Evaluate Model Accuracy using with Confusion Matrix**

table(test$ACT\_279.304.A, fitted\_result)

MissclassError <- mean(fitted\_result != test$ACT\_279.304.A)

print(paste('Accuracy = ', 1-MissclassError))

**fitted\_result**

0 1

0 399 8

1 105 5

**"LRM accuracy” = 0.781431334622824**

#################### **Decision Tree** #######################

**#Train model using rpart for Decision Tree**

**#predict on test data**

fitted\_value <- predict(DTModel, newdata = test, type = "class")

**#Evaluate the model accuarcy**

table(test$ACT\_279.304.A, fitted\_value)

misClassError <- mean(fitted\_value != test$

**fitted\_value**

0 1

0 385 22

1. 94 16

print(paste('Accuracy =', 1-misClassError))

**"DTM accuracy = 0.774390243902439"**

#################### **Random Forest** #######################

library(stats)

library(randomForest)

**#Inspect base data**

View(ds)

**#Variable selection**

str(ds)

**# Splitting Data in Training and Testing**

index = sample(2,nrow(ds), replace = TRUE, prob=c(0.7,0.3))

**#Training Data**

Training = ds[index==1,]

**#Testing Data**

Testing = ds[index==2,]

**# Random Forest model**

RF = randomForest(ACT\_279.304.A~.,data = Training)

**# Evaluating Model Accuracy**

ACT\_279.304.A\_Pred = predict(RF, Testing)

Testing$ACT\_279.304.A\_Pred = ACT\_279.304.A\_Pred

View(Testing)

**# Confusion Matrix**

CFM = table(Testing$ACT\_279.304.A, Testing$ACT\_279.304.A\_Pred)

CFM

**0 1**

**0** 396 0

**1** 6 95

Mod\_Accu = sum(diag(CFM)/sum(CFM))

Mod\_Accu

**RFM accuracy - 0.9879276**