**Practical 1**

1. Implement Decision tree classification techniques

**Code:**

library(party)

print(head(readingSkills))

library(party)

str(iris)

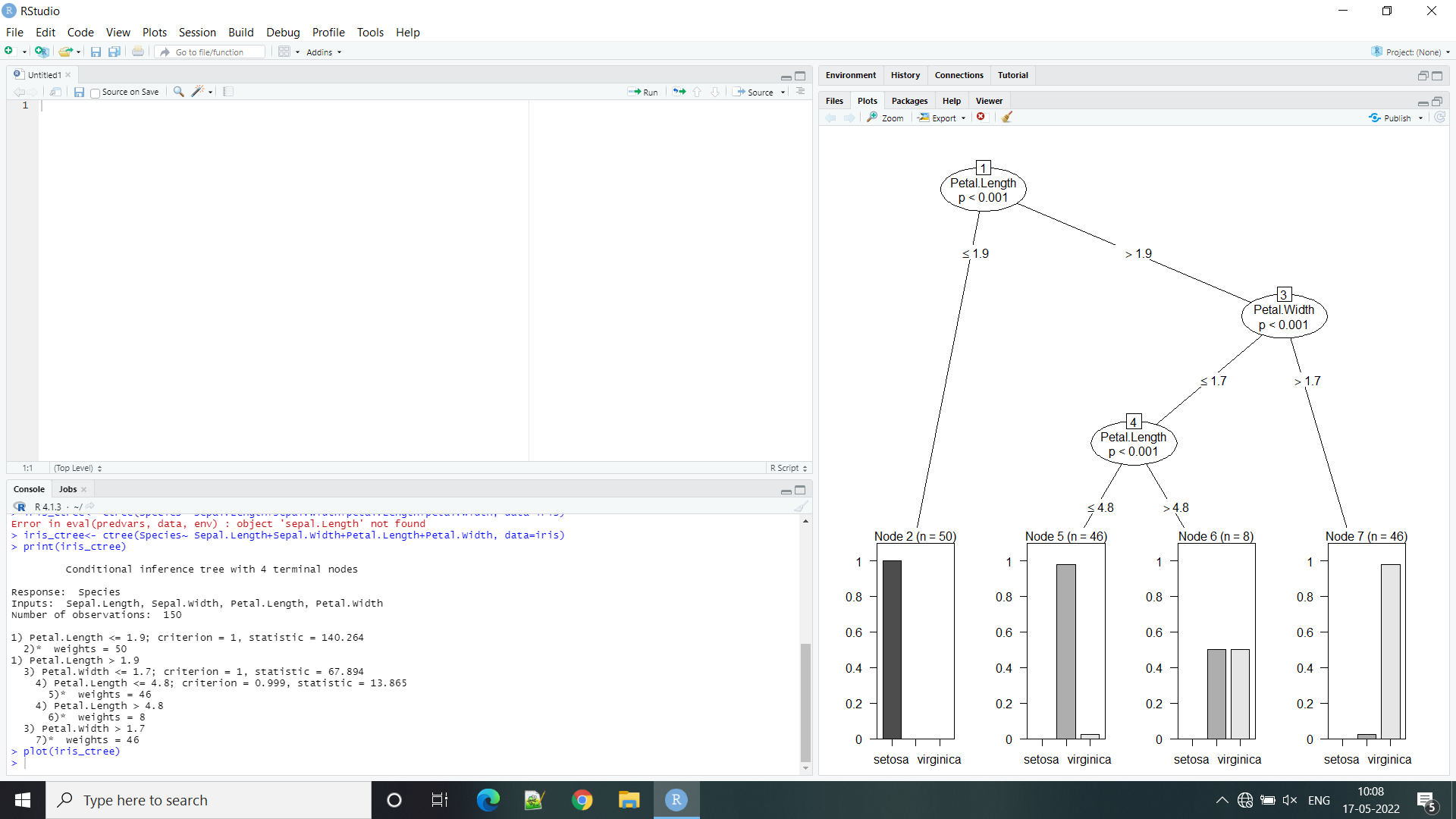
iris\_ctree<- ctree(Species~Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, data = iris)

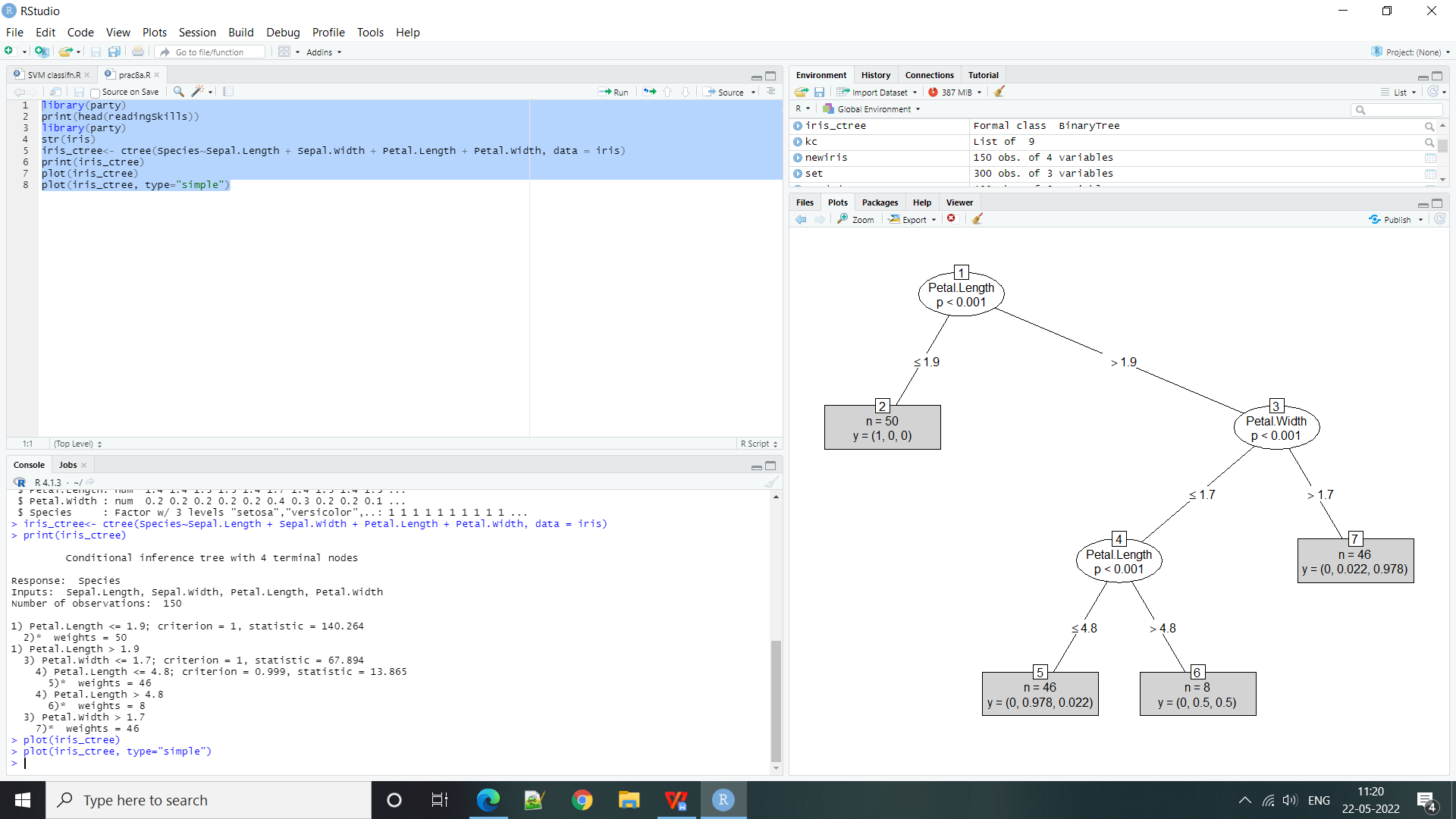
print(iris\_ctree)

plot(iris\_ctree)

plot(iris\_ctree, type="simple")

**Output**





**Practical 2**

**Implement SVM classification techniques**

**Code:**

getwd()

ds=read.csv("social.csv", TRUE, ",")

ds

ds = ds[3:5]

ds

library(caTools)

set.seed(123)

split =sample.split(ds$Purchased, SplitRatio = 0.75)

training\_set = subset(ds, split == TRUE)

test\_set = subset(ds, split == FALSE)

ds

test\_set[-3] = scale(test\_set[-3])

training\_set[-3]= scale(training\_set[-3])

test\_set[-3]=scale(test\_set[-3])

test\_set[-3]

library(e1071)

classifier = svm(formula = Purchased~., data =training\_set, type = 'C-classification', kernel = 'linear')

classifier

y\_pred = predict(classifier, newdata = test\_set[-3])

y\_pred

cm = table(test\_set[, 3], y\_pred)

cm

set = training\_set

X1= seq(min(set[, 1]) -1, max(set[, 1])+1, by= 0.01)

X2= seq(min(set[, 2]) -1, max(set[, 2])+1, by= 0.01)

grid\_set = expand.grid(X1, X2)

colnames(grid\_set) = c('Age', 'EstimatedSalary')

y\_grid = predict(classifier,newdata= grid\_set)

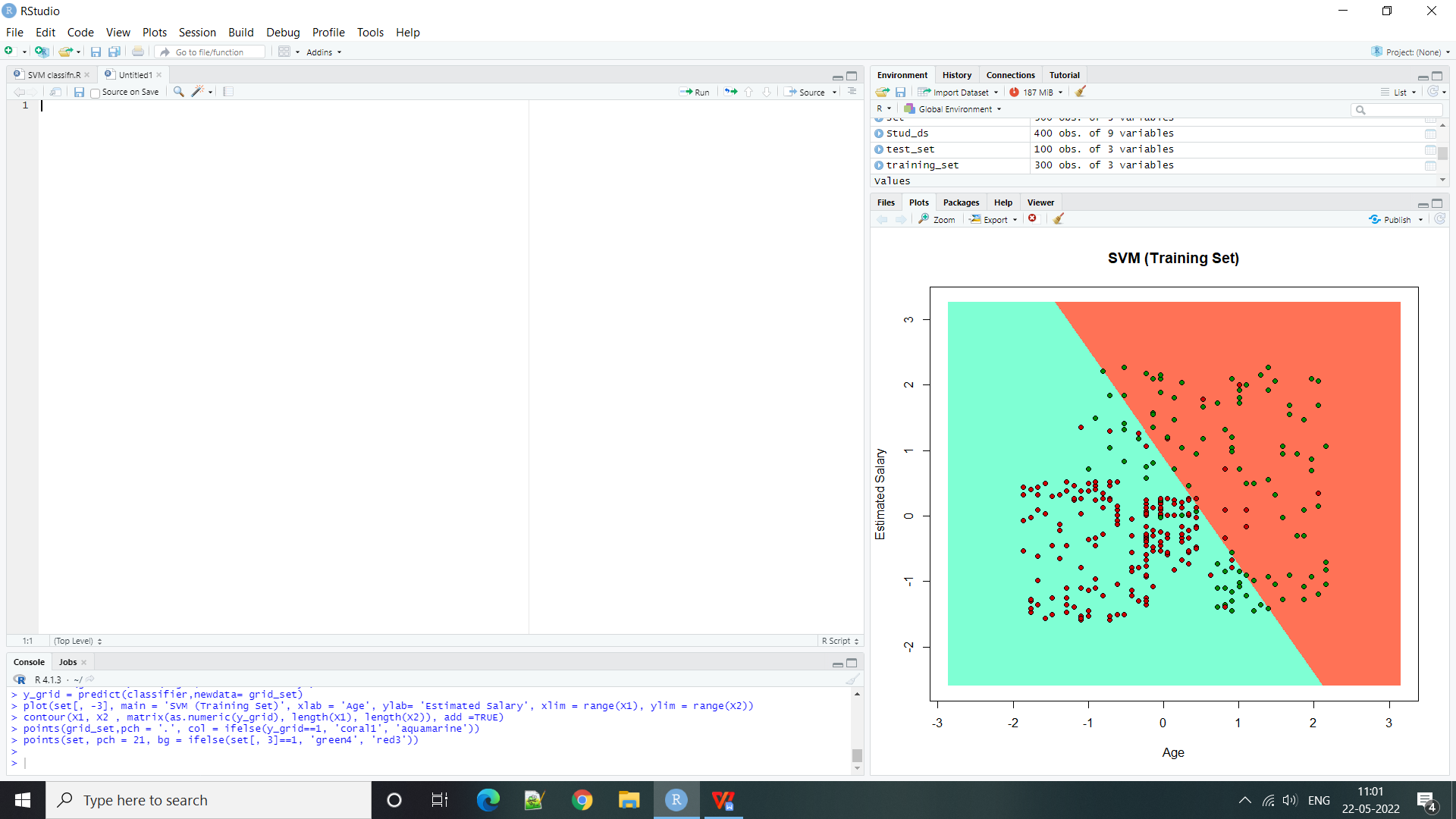
plot(set[, -3], main = 'SVM (Training Set)', xlab = 'Age', ylab= 'Estimated Salary', xlim = range(X1), ylim = range(X2))

contour(X1, X2 , matrix(as.numeric(y\_grid), length(X1), length(X2)), add =TRUE)

points(grid\_set,pch = '.', col = ifelse(y\_grid==1, 'coral1', 'aquamarine'))

points(set, pch = 21, bg = ifelse(set[, 3]==1, 'green4', 'red3'))

**Output:**



**Practical 3**

**REGRESSION MODEL Import a data from web storage. Name the dataset and now do Logistic Regression to find out relation between variables that are affecting the admission of a student in an institute based on his or her GRE score, GPA obtained and rank of the student. Also check the model is fit or not. require (foreign), require(MASS).**

**Code:**

library(MASS)

library(tidyr)

library(dplyr)

library(corrplot)

library(ggplot2)

library(ROCR)

library(tidyverse)

df<- read.csv("D:/M.Sc.IT part-I/Big data analytics/Admission\_Predict.csv")

dff<-df

length(unique(df$Serial.No.))==nrow(df)

sum(is.na(df))

summary(df)

df$Serial.No.=NULL

summary(df$GRE.Score)

df$Serial.No.=NULL

summary(df$GRE.Score)

quantile(df$GRE.Score,seq(0,1,0.01))

q1<-quantile(df$GRE.Score,c(0.25))

q3<-quantile(df$GRE.Score,c(0.75))

IQR<- q3-q1

upper\_range<-q3+1.5\*IQR

lower\_range<-q1-1.5\*IQR

nrow(df[df$GRE.Score > upper\_range,])

nrow(df[df$GRE.Score < lower\_range,])

summary(df$TOEFL.Score)

quantile(df$TOEFL.Score,seq(0,1,0.01))

q1<-quantile(df$TOEFL.Score,c(0.25))

q3<-quantile(df$TOEFL.Score,c(0.75))

IQR<- q3-q1

upper\_range<-q3+1.5\*IQR

lower\_range<-q1-1.5\*IQR

nrow(df[df$TOEFL.Score > upper\_range,])

nrow(df[df$TOEFL.Score < lower\_range,])

summary(df$CGPA)

quantile(df$CGPA.Score,seq(0,1,0.01))

q1<-quantile(df$CGPA.Score,c(0.25))

q3<-quantile(df$CGPA.Score,c(0.75))

IQR<- q3-q1

upper\_range<-q3+1.5\*IQR

lower\_range<-q1-1.5\*IQR

nrow(df[df$CGPA.Score > upper\_range,])

nrow(df[df$CGPA.Score < lower\_range,])

df$CGPA[which(df$CGPA < lower\_range)] <- lower\_range

summary(factor(df$Rank))

df$Rank <- as.factor(df$Rank)

summary(factor(df$SOP))

df$SOP <- as.factor(df$SOP)

summary(factor(df$LOR))

df$LOR <- as.factor(df$LOR)

summary(factor(df$Research))

df$Research <- as.factor(df$Research)

table(df$Chance.of.Admit > 0.5) # False = 35, True=204

df$get\_admission = as.factor(ifelse(df$Chance.of.Admit > 0.72,1,0))

table(df$Chance.of.Admit>0.72)

df\_Numeric\_Variable <- select\_if(df, is.numeric)

df\_Numeric\_Variable

corr <- cor(df\_Numeric\_Variable)

corrplot(corr,method = "number",type = "full")

ggplot(df, aes(x=Rank, y = Chance.of.Admit))+ geom\_boxplot(outlier.colour = "red")

ggplot(df, aes(x=SOP, y = Chance.of.Admit))+ geom\_boxplot(outlier.colour = "red")

ggplot(df, aes(x=LOR, y = Chance.of.Admit))+ geom\_boxplot(outlier.colour = "red")

ggplot(df, aes(x=Research, y = Chance.of.Admit))+ geom\_boxplot(outlier.colour = "red")

length(levels(df$Rank))

dummy\_Rank <- data.frame(model.matrix( ~Rank, data = df))

dummy\_Rank <- dummy\_Rank[,-1]

length(dummy\_Rank)

df\_1 <- cbind(select(df, -'Rank'), dummy\_Rank)

ncol(df\_1)

dummy\_SOP <- data.frame(model.matrix( ~SOP, data = df))

dummy\_SOP <- dummy\_SOP[,-1]

length(dummy\_SOP)

df\_2 <- cbind(select(df\_1, -'SOP'), dummy\_SOP)

ncol(df\_2)

dummy\_LOR <- data.frame(model.matrix( ~LOR, data = df))

dummy\_LOR <- dummy\_LOR[,-1]

length(dummy\_LOR)

df\_3 <- cbind(select(df\_2, -'LOR'), dummy\_LOR)

ncol(df\_3)

df\_3$Chance.of.Admit = NULL

df\_3

set.seed(1000)

indx<- sample(1:nrow(df\_3), 0.7\*nrow(df\_3))

train <- df\_3[indx,]

test <-df\_3[-indx,]

model1 <- glm(get\_admission ~ ., data = train, family = "binomial")

summary(model1) #AIC = 186.62, Null deviance = 387.65

model1.0 <- glm( formula = get\_admission ~ SOP1.5+SOP2+SOP3.5+SOP4.5+LOR2+LOR2.5+LOR4+LOR4.5+Rank2+Rank3+Rank5,family = "binomial", data = train)

summary(model1.0)

model1.1 <- glm( formula = get\_admission ~ TOEFL.Score + CGPA + Research + SOP1.5+SOP2+SOP3.5+SOP4.5+LOR2+LOR2.5+LOR4+LOR4.5+Rank2+Rank3+Rank5,family = "binomial", data = train)

summary(model1.1)

model1.2 <- glm( formula = get\_admission ~ GRE.Score + CGPA + Research + SOP1.5+SOP2+SOP3.5+SOP4.5+LOR2+LOR2.5+LOR4+LOR4.5+Rank2+Rank3+Rank5,family = "binomial", data = train)

summary(model1.2)

model1.3 <- glm( formula = get\_admission ~ GRE.Score + CGPA + Research +SOP2+SOP3.5+SOP4.5+LOR2+LOR2.5+LOR4+Rank2+Rank3,family = "binomial", data = train)

summary(model1.3)

model1.4 <- glm( formula = get\_admission ~ GRE.Score + CGPA + Research+SOP3.5+SOP4+LOR4+LOR4.5+Rank2,family = "binomial", data = train)

summary(model1.4)

model1.5 <- glm( formula = get\_admission ~ GRE.Score + CGPA + Research+SOP3+SOP4+LOR3+LOR4.5+Rank2,family = "binomial", data = train)

summary(model1.5)

model1.6 <- glm( formula = get\_admission ~ GRE.Score + CGPA + Research+SOP2+SOP3+SOP4+LOR3+LOR4.5+Rank2,family = "binomial", data = train)

summary(model1.6)

model1.7 <- glm(formula = get\_addmission~GRE.Score + CGPA + Research + LOR4+LOR5.5+Rank+ SOP2+ SOP4+LOR4, family = "binomial", data="train")

summary(model1.7)

model1.8 <- glm(formula = get\_addmission~GRE.Score + CGPA + Research + LOR4+LOR5.5+Rank+ SOP2+ SOP4+LOR4, family = "binomial", data="train")

summary(model1.8)

model1.9 <- glm( formula = get\_admission ~ GRE.Score + CGPA + Research + LOR1.5+SOP4.5+Rank2 + LOR3.5 + LOR2.5 + LOR4,family = "binomial", data = train)

summary(model1.9)

model1.10 <- glm( formula = get\_admission ~ GRE.Score + Research + CGPA + Rank2 + SOP2 + SOP4 + LOR4,family = "binomial", data = train)

summary(model1.10)

predictTrain = predict(model1.8, type="response")

summary(predictTrain)

table(train$get\_admission, predictTrain > 0.5)

table(train$get\_admission, predictTrain > 0.4)

pred1 <- prediction(predictTrain,train$get\_admission)

roc.perf <- performance(pred1, measure = "tpr", x.measure = "fpr")

plot(roc.perf,colorize=TRUE)

predictTest = predict(model1.10, type = "response", newdata = test)

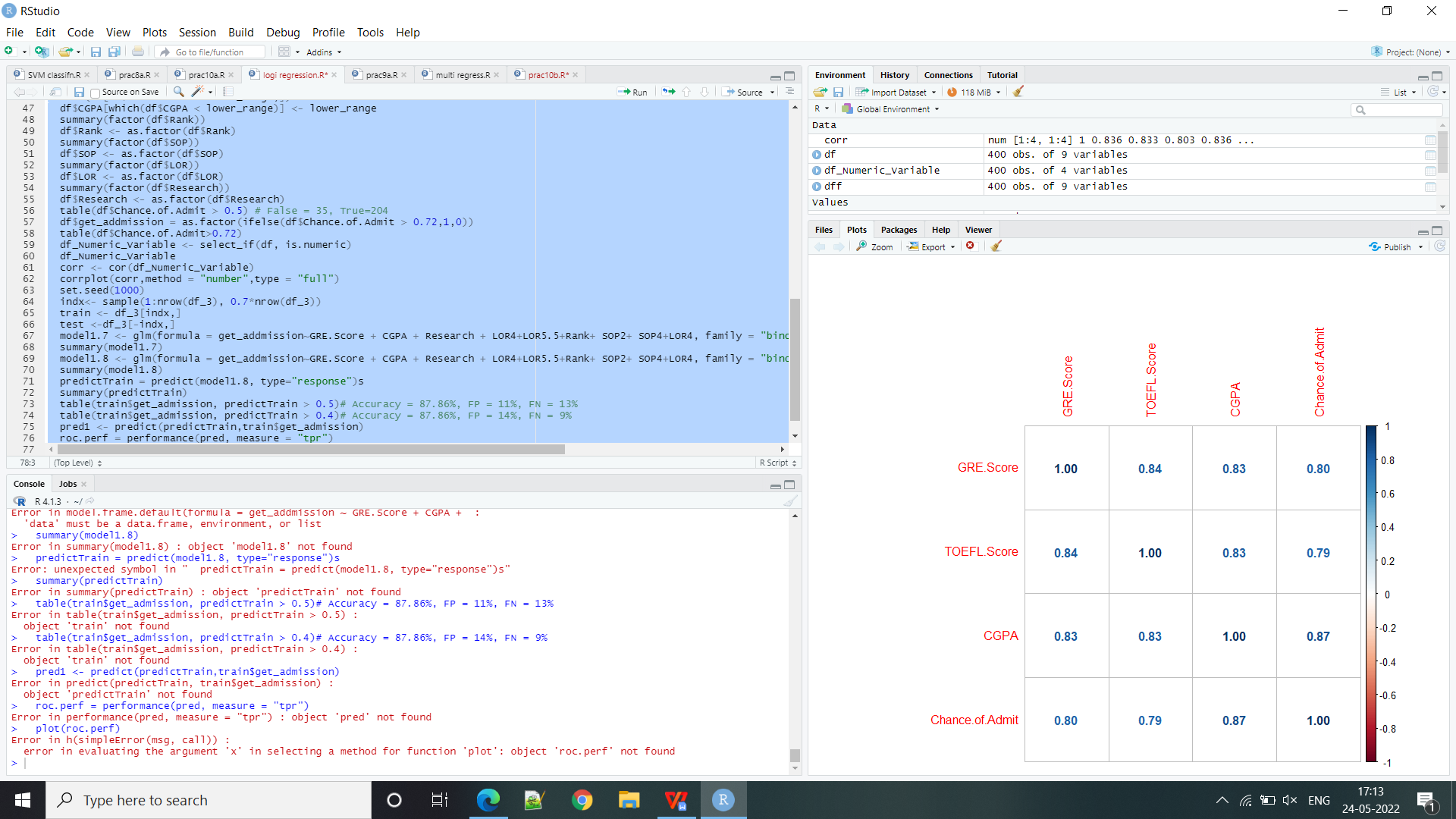
table(test$get\_admission,predictTest >= 0.5)

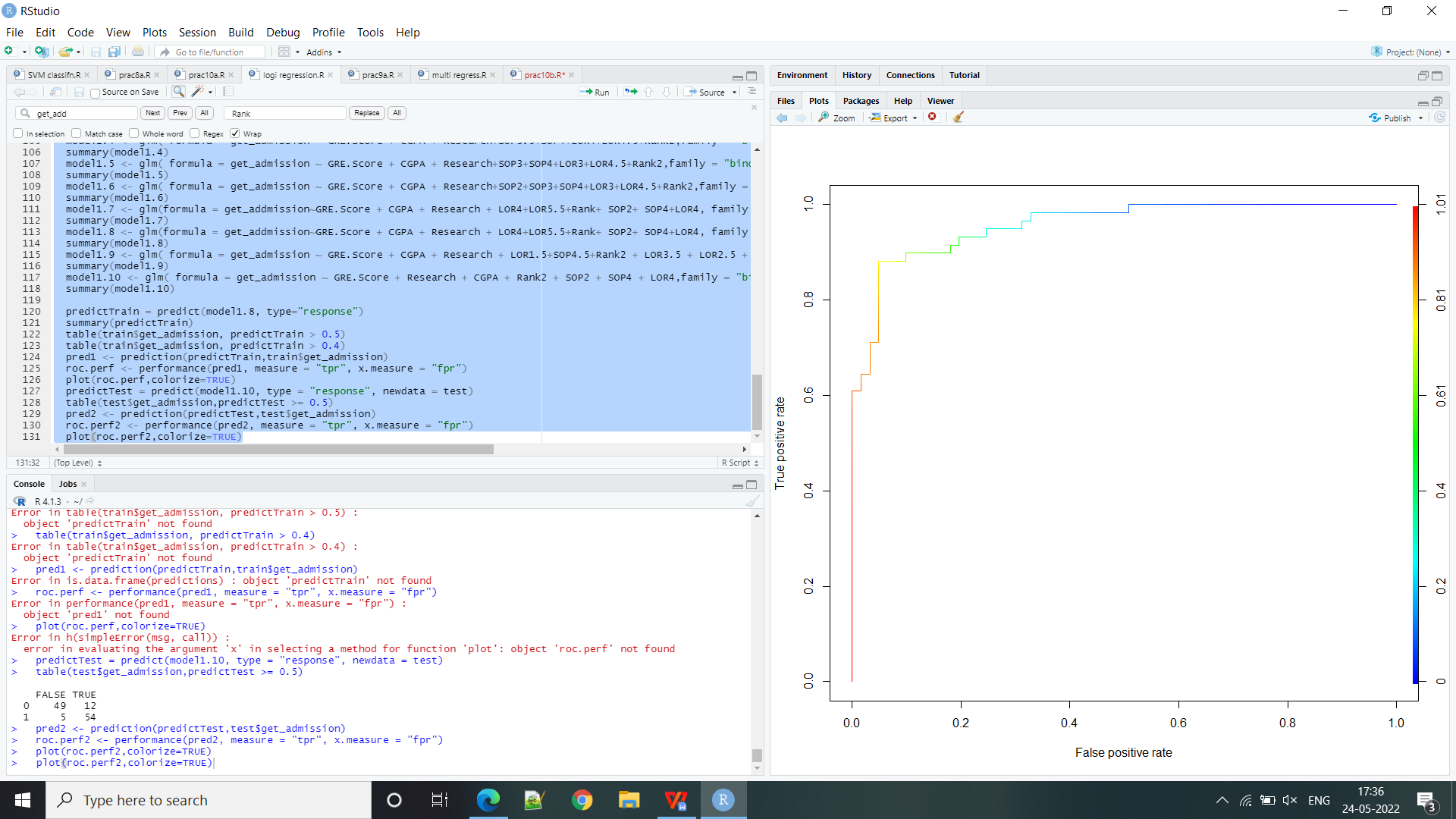
pred2 <- prediction(predictTest,test$get\_admission)

roc.perf2 <- performance(pred2, measure = "tpr", x.measure = "fpr")

plot(roc.perf2,colorize=TRUE)

**Output:**





**Practical 4**

**MULTIPLE REGRESSION MODEL Apply multiple regressions, if data have a continuous independent variable. Apply on above dataset.**

**Code:**

library(GGally)

library(ggplot2)

library(readr)

library(dplyr)

library(funModeling)

library(lmtest)

library(car)

library(MLmetrics)

admission <- read.csv("D:/M.Sc.IT part-I/Big data analytics/Admission\_Predict.csv")

head(str(admission))

admission %>%

is.na() %>%

colSums(is.na(admission))

admission\_new <- admission %>%

select(-Serial.No.)

admission\_new

admission\_new$Research <- as.factor(admission\_new$Research)

admission\_new$Rank <- as.factor(admission\_new$Rank)

head(str(admission\_new))

ggcorr(admission\_new, label = T)

ggplot(data = admission\_new, aes(x = GRE.Score, y = Chance.of.Admit)) +

geom\_point() +

geom\_smooth(method = lm, level = 0.95)

ggplot(data = admission\_new, aes(x = TOEFL.Score, y = Chance.of.Admit)) +

geom\_point() +

geom\_smooth(method = lm, level = 0.95)

ggplot(data = admission\_new, aes(x = SOP, y = Chance.of.Admit)) +

geom\_point() +

geom\_smooth(method = lm, level = 0.95)

ggplot(data = admission\_new, aes(x = LOR, y = Chance.of.Admit)) +

geom\_point() +

geom\_smooth(method = lm, level = 0.95)

ggplot(data = admission\_new, aes(x = CGPA, y = Chance.of.Admit)) +

geom\_point() +

geom\_smooth(method = lm, level = 0.95)

plot\_num(admission\_new)

set.seed(417)

for\_train <- sample(nrow(admission\_new), nrow(admission\_new)\*0.8)

admission.train <- admission\_new[for\_train, ]

admission.test <- admission\_new[-for\_train, ]

model\_admission.none <- lm(formula = Chance.of.Admit ~ 1,data = admission.train)

model\_admission.none

model\_admission.all <- lm(formula = Chance.of.Admit ~.,data = admission.train)

model\_admission.all

summary(model\_admission.all)

step(object = model\_admission.all,direction = backward, trace = 0)

model\_admnew <- lm(formula = Chance.of.Admit ~ GRE.Score + TOEFL.Score + LOR +

CGPA + Research, data = admission.train)

summary(model\_admnew)

hist(model\_admnew$residuals)

shapiro.test(x = model\_admnew$residuals)

plot(model\_admnew$fitted.values, model\_admnew$residuals)

abline(h = 0, col = red)

bptest(model\_admnew)

vif(model\_admnew)

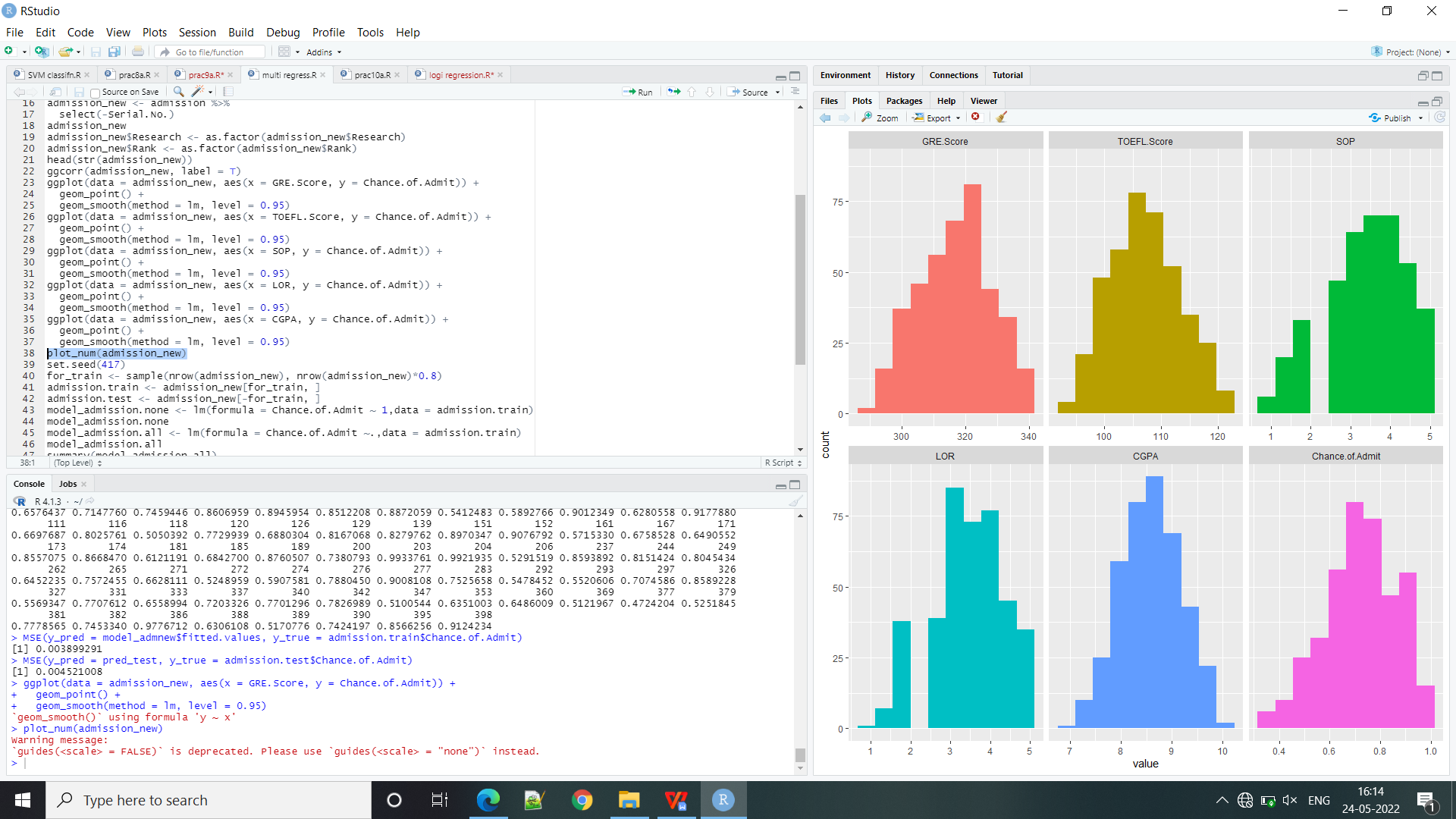
pred\_test <- predict(object = model\_admnew, newdata = admission.test)

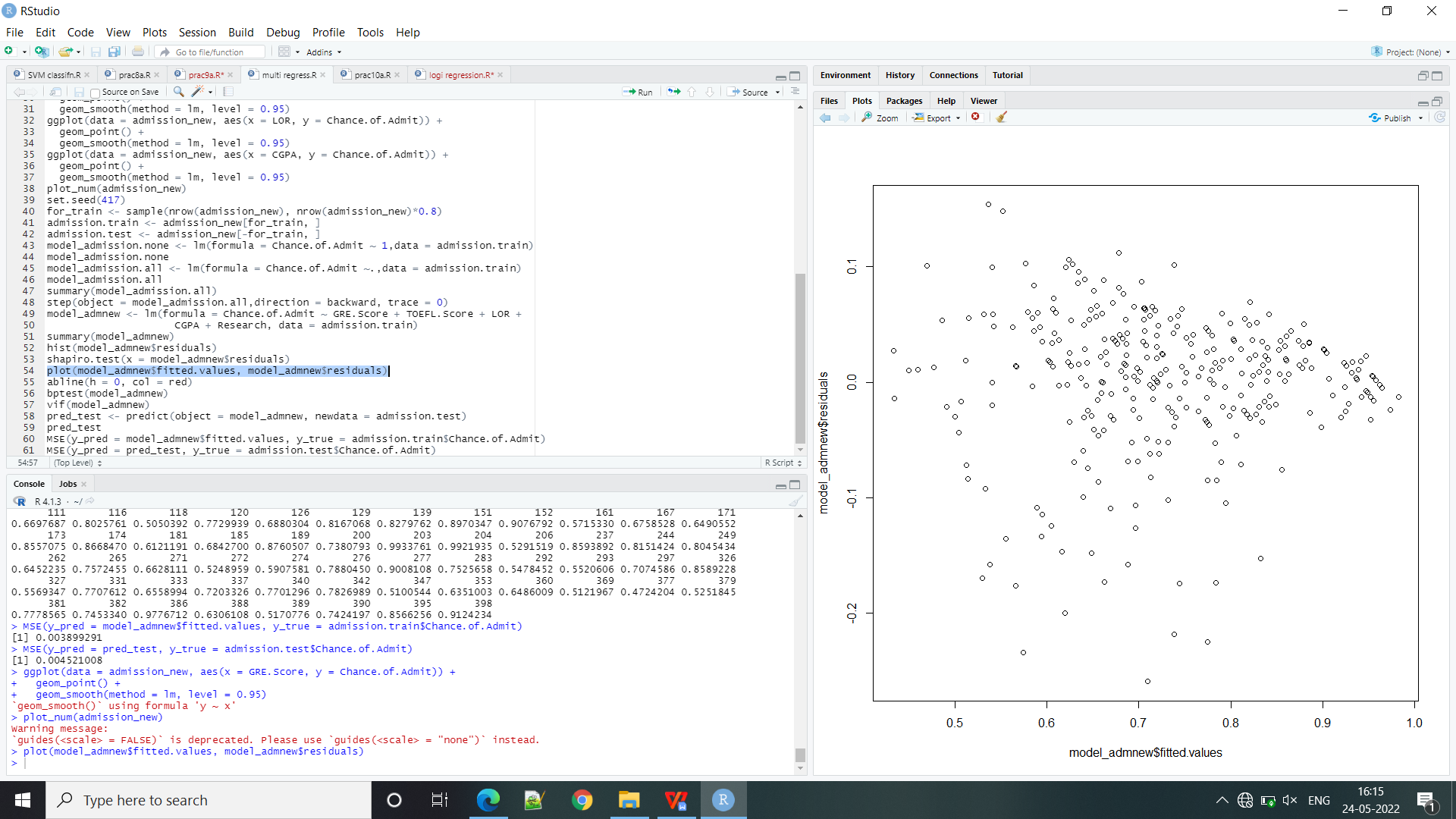
pred\_test

MSE(y\_pred = model\_admnew$fitted.values, y\_true = admission.train$Chance.of.Admit)

MSE(y\_pred = pred\_test, y\_true = admission.test$Chance.of.Admit)

**Output:**





**Practical 5**

**CLASSIFICATION MODEL a. Install relevant package for classification. Choose classifier for classification problem. c. Evaluate the performance of classifier.**

**Code:**

data(iris)

str(iris)

library(e1071)

library(caTools)

library(caret)

split<- sample.split(iris, SplitRatio = 0.7)

train\_cl<- subset(iris, split=="TRUE")

test\_cl<- subset(iris, split=="FALSE")

train\_scale<- scale(train\_cl[,1:4])

test\_scale<- scale(test\_cl[,1:4])

set.seed(120)

classifier\_cl<- naiveBayes(Species~., data= train\_cl)

classifier\_cl

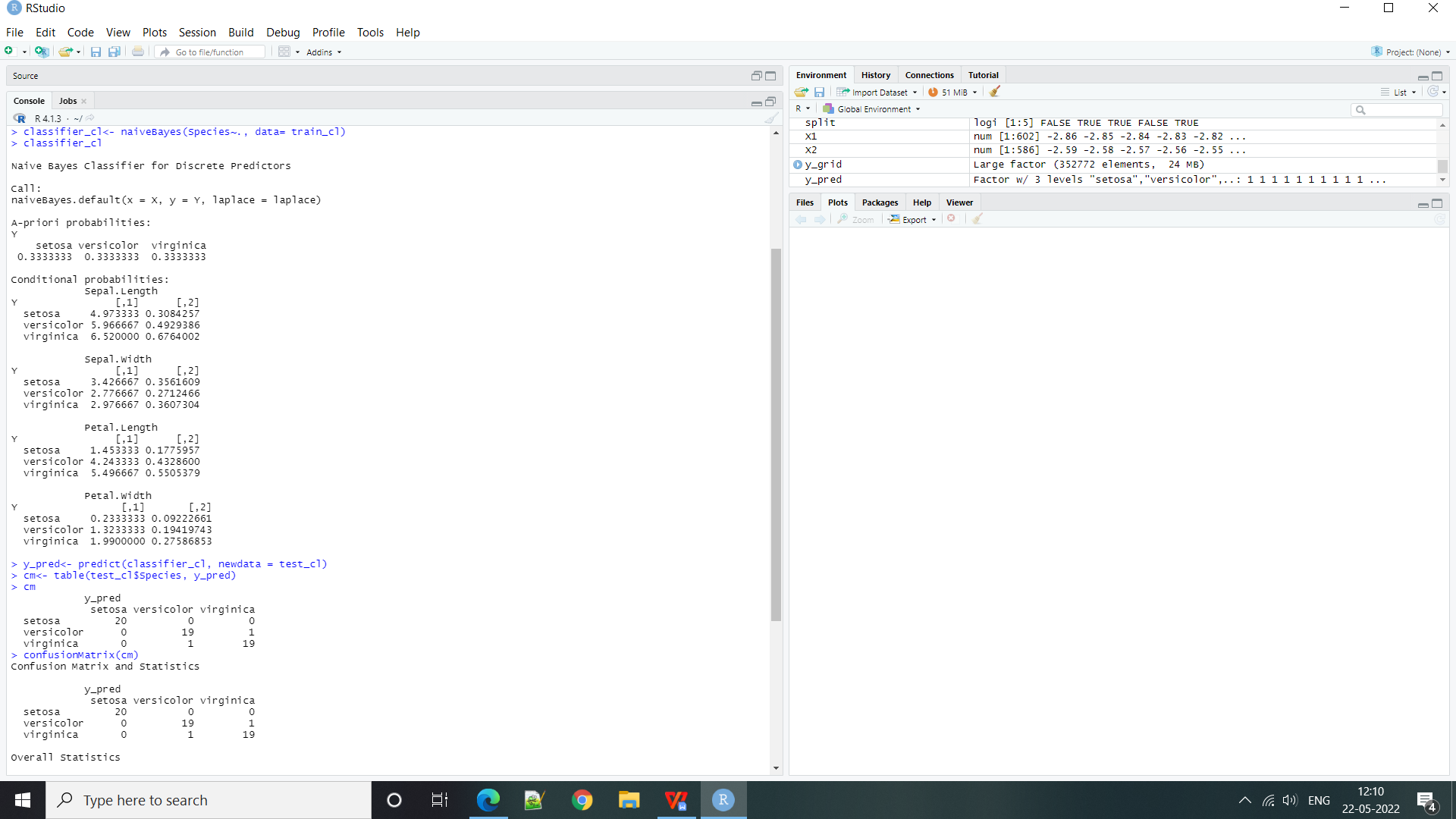
y\_pred<- predict(classifier\_cl, newdata = test\_cl)

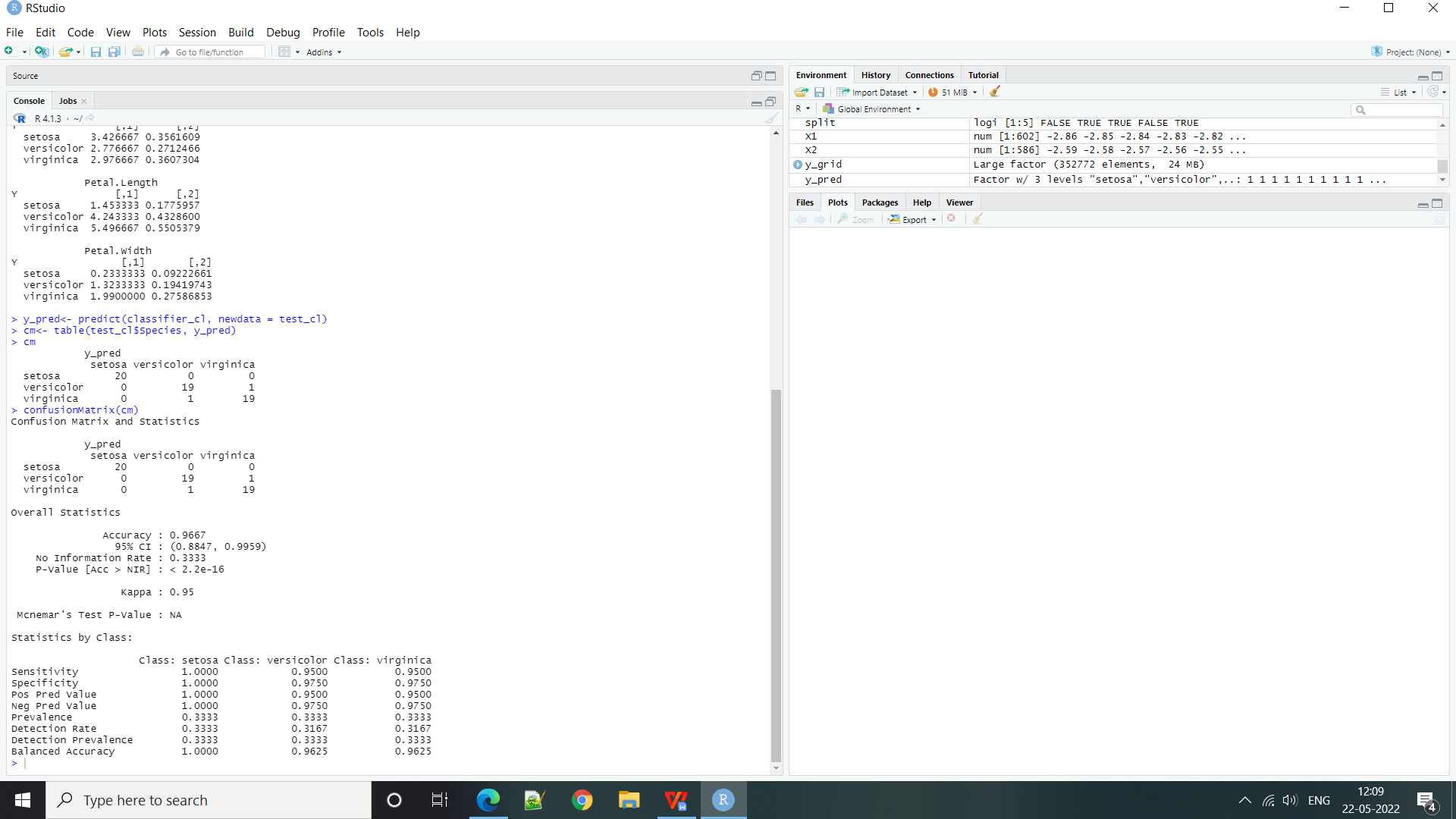
cm<- table(test\_cl$Species, y\_pred)

cm

confusionMatrix(cm)

**Output:**

;



**Practical 6**

**CLUSTERING MODEL a. Clustering algorithms for unsupervised classification. b. Plot the cluster data using R visualizations.**

**Code:**

newiris<- iris

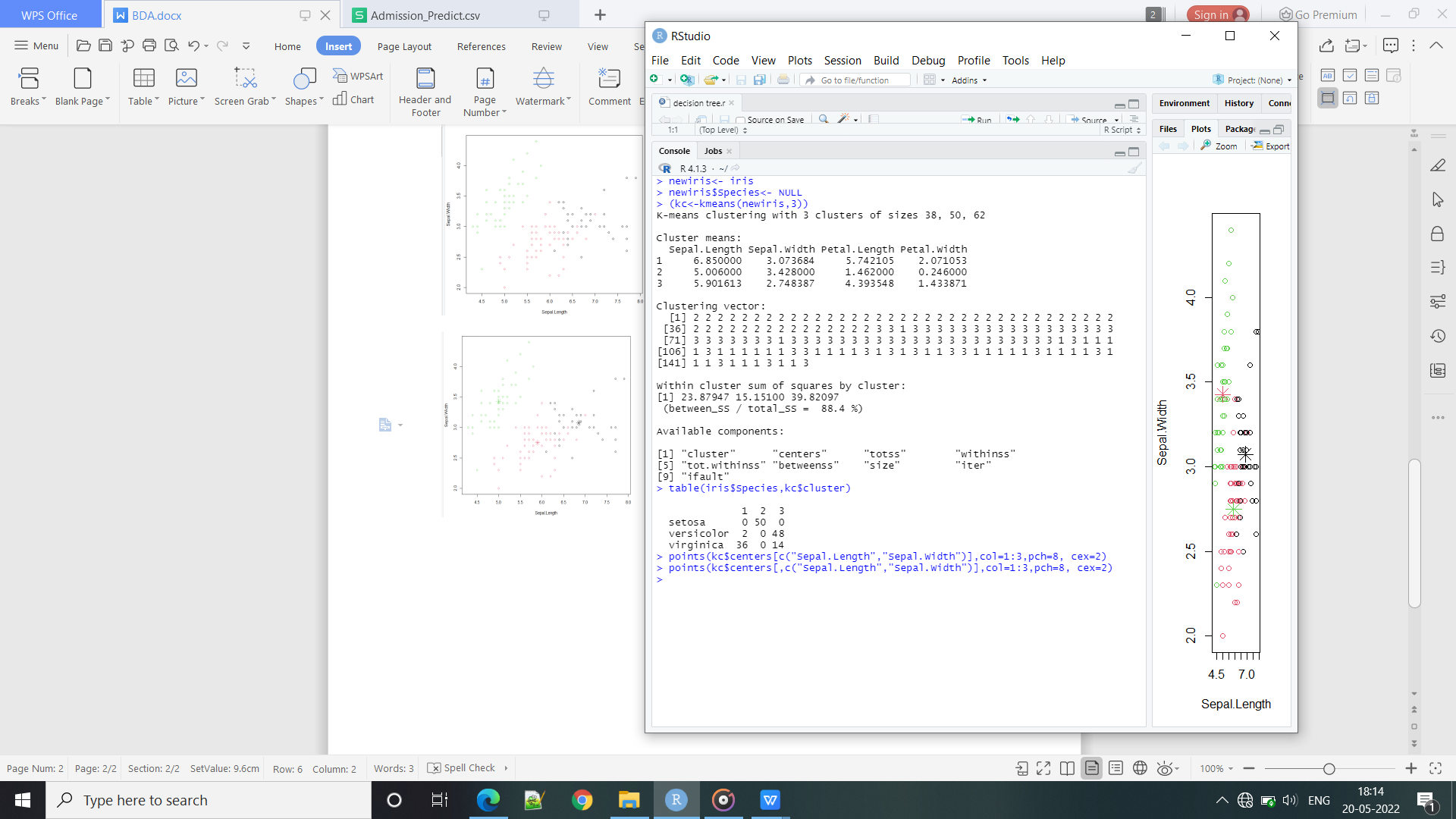
newiris$species<-NULL

(kc<-kmeans(newiris,3))

table(iris$Species,kc$cluster)

points((kc$centers[c("Sepal.Length", "Sepal.width")],col=1:3, pch=8, cex=2))

points((kc$centers[,c("Sepal.Length", "Sepal.width")],col=1:3, pch=8, cex=2))



**Output:**

