parkinson\_final\_project.R

soumya

2023-07-27

#Importing the required library  
library(readxl)  
  
#importing the dataset  
parkinson<- read\_excel("C:/Users/soumy/ONeDrive/Desktop/parkinson.xlsx")  
  
#creating a copy of the dataset  
dataset<- parkinson  
  
print(head(dataset))

## # A tibble: 6 × 24  
## name `MDVP:Fo(Hz)` `MDVP:Fhi(Hz)` `MDVP:Flo(Hz)` `MDVP:Jitter(%)`  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 phon\_R01\_S01\_1 120. 157. 75.0 0.00784  
## 2 phon\_R01\_S01\_2 122. 149. 114. 0.00968  
## 3 phon\_R01\_S01\_3 117. 131. 112. 0.0105   
## 4 phon\_R01\_S01\_4 117. 138. 111. 0.00997  
## 5 phon\_R01\_S01\_5 116. 142. 111. 0.0128   
## 6 phon\_R01\_S01\_6 121. 131. 114. 0.00968  
## # ℹ 19 more variables: `MDVP:Jitter(Abs)` <dbl>, `MDVP:RAP` <dbl>,  
## # `MDVP:PPQ` <dbl>, `Jitter:DDP` <dbl>, `MDVP:Shimmer` <dbl>,  
## # `MDVP:Shimmer(dB)` <dbl>, `Shimmer:APQ3` <dbl>, `Shimmer:APQ5` <dbl>,  
## # `MDVP:APQ` <dbl>, `Shimmer:DDA` <dbl>, NHR <dbl>, HNR <dbl>, status <dbl>,  
## # RPDE <dbl>, DFA <dbl>, spread1 <dbl>, spread2 <dbl>, D2 <dbl>, PPE <dbl>

print(tail(dataset))

## # A tibble: 6 × 24  
## name `MDVP:Fo(Hz)` `MDVP:Fhi(Hz)` `MDVP:Flo(Hz)` `MDVP:Jitter(%)`  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 phon\_R01\_S50\_1 202. 263. 78.2 0.00694  
## 2 phon\_R01\_S50\_2 174. 231. 94.3 0.00459  
## 3 phon\_R01\_S50\_3 210. 253. 89.5 0.00564  
## 4 phon\_R01\_S50\_4 175. 240. 74.3 0.0136   
## 5 phon\_R01\_S50\_5 199. 397. 74.9 0.0074   
## 6 phon\_R01\_S50\_6 214. 260. 78.0 0.00567  
## # ℹ 19 more variables: `MDVP:Jitter(Abs)` <dbl>, `MDVP:RAP` <dbl>,  
## # `MDVP:PPQ` <dbl>, `Jitter:DDP` <dbl>, `MDVP:Shimmer` <dbl>,  
## # `MDVP:Shimmer(dB)` <dbl>, `Shimmer:APQ3` <dbl>, `Shimmer:APQ5` <dbl>,  
## # `MDVP:APQ` <dbl>, `Shimmer:DDA` <dbl>, NHR <dbl>, HNR <dbl>, status <dbl>,  
## # RPDE <dbl>, DFA <dbl>, spread1 <dbl>, spread2 <dbl>, D2 <dbl>, PPE <dbl>

#counting no. of rows  
nrow(dataset)

## [1] 195

#counting no. of cols  
ncol(dataset)

## [1] 24

#shape of dataset  
dim(dataset)

## [1] 195 24

#columns of dataset  
colnames(dataset)

## [1] "name" "MDVP:Fo(Hz)" "MDVP:Fhi(Hz)" "MDVP:Flo(Hz)"   
## [5] "MDVP:Jitter(%)" "MDVP:Jitter(Abs)" "MDVP:RAP" "MDVP:PPQ"   
## [9] "Jitter:DDP" "MDVP:Shimmer" "MDVP:Shimmer(dB)" "Shimmer:APQ3"   
## [13] "Shimmer:APQ5" "MDVP:APQ" "Shimmer:DDA" "NHR"   
## [17] "HNR" "status" "RPDE" "DFA"   
## [21] "spread1" "spread2" "D2" "PPE"

#numeric columns of dataset  
colnames(dataset[,sapply(dataset,is.numeric)])

## [1] "MDVP:Fo(Hz)" "MDVP:Fhi(Hz)" "MDVP:Flo(Hz)" "MDVP:Jitter(%)"   
## [5] "MDVP:Jitter(Abs)" "MDVP:RAP" "MDVP:PPQ" "Jitter:DDP"   
## [9] "MDVP:Shimmer" "MDVP:Shimmer(dB)" "Shimmer:APQ3" "Shimmer:APQ5"   
## [13] "MDVP:APQ" "Shimmer:DDA" "NHR" "HNR"   
## [17] "status" "RPDE" "DFA" "spread1"   
## [21] "spread2" "D2" "PPE"

#printing status of every patient  
status\_val<-dataset$status  
print(status\_val)

## [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1  
## [38] 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1  
## [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0  
## [186] 0 0 0 0 0 0 0 0 0 0

sum(status\_val==0)

## [1] 48

sum(status\_val==1)

## [1] 147

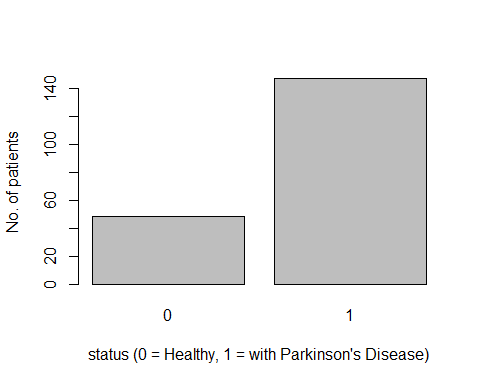
#searcing unique and null values  
  
unique\_values<-unique(dataset)  
  
null\_values<-is.na(dataset)  
  
print(unique\_values)

## # A tibble: 195 × 24  
## name `MDVP:Fo(Hz)` `MDVP:Fhi(Hz)` `MDVP:Flo(Hz)` `MDVP:Jitter(%)`  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 phon\_R01\_S01\_1 120. 157. 75.0 0.00784  
## 2 phon\_R01\_S01\_2 122. 149. 114. 0.00968  
## 3 phon\_R01\_S01\_3 117. 131. 112. 0.0105   
## 4 phon\_R01\_S01\_4 117. 138. 111. 0.00997  
## 5 phon\_R01\_S01\_5 116. 142. 111. 0.0128   
## 6 phon\_R01\_S01\_6 121. 131. 114. 0.00968  
## 7 phon\_R01\_S02\_1 120. 137. 115. 0.00333  
## 8 phon\_R01\_S02\_2 107. 114. 104. 0.0029   
## 9 phon\_R01\_S02\_3 95.7 132. 91.8 0.00551  
## 10 phon\_R01\_S02\_4 95.1 120. 91.2 0.00532  
## # ℹ 185 more rows  
## # ℹ 19 more variables: `MDVP:Jitter(Abs)` <dbl>, `MDVP:RAP` <dbl>,  
## # `MDVP:PPQ` <dbl>, `Jitter:DDP` <dbl>, `MDVP:Shimmer` <dbl>,  
## # `MDVP:Shimmer(dB)` <dbl>, `Shimmer:APQ3` <dbl>, `Shimmer:APQ5` <dbl>,  
## # `MDVP:APQ` <dbl>, `Shimmer:DDA` <dbl>, NHR <dbl>, HNR <dbl>, status <dbl>,  
## # RPDE <dbl>, DFA <dbl>, spread1 <dbl>, spread2 <dbl>, D2 <dbl>, PPE <dbl>

print(sum(null\_values))

## [1] 0

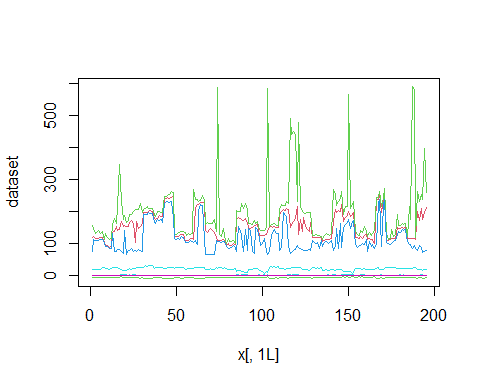
# Plotting the ratio of healthy people to people with Parkinson's  
barplot(table(dataset$status), xlab = "status (0 = Healthy, 1 = with Parkinson's Disease)", ylab = "No. of patients")



matplot(y=dataset,type='l',lty=1)

## Warning in xy.coords(x, y, xlabel, ylabel, log = log, recycle = TRUE): NAs  
## introduced by coercion

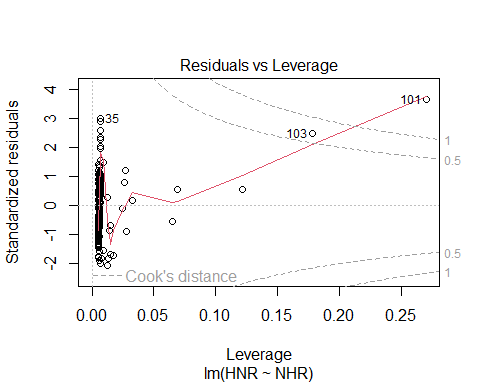
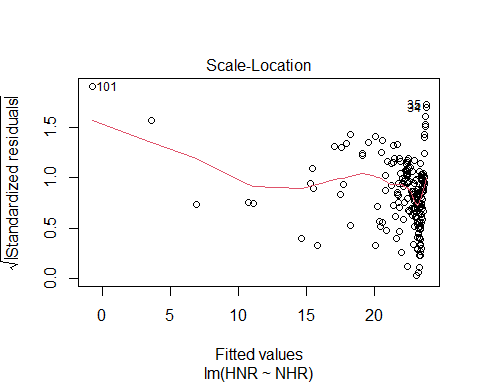
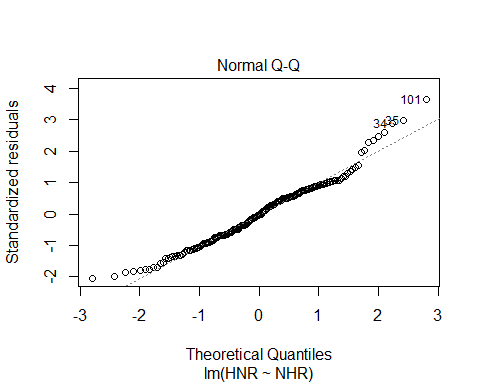
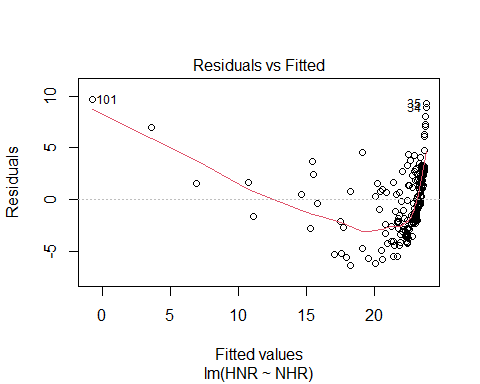
## Warning in xy.coords(x, y, xlabel, ylabel, log): NAs introduced by coercion



fit<-lm(HNR~NHR,data=dataset)  
summary(fit)

##   
## Call:  
## lm(formula = HNR ~ NHR, data = dataset)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -6.3151 -2.1989 -0.0422 2.0399 9.6540   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 23.8288 0.2613 91.19 <2e-16 \*\*\*  
## NHR -78.1899 5.5179 -14.17 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 3.106 on 193 degrees of freedom  
## Multiple R-squared: 0.5099, Adjusted R-squared: 0.5074   
## F-statistic: 200.8 on 1 and 193 DF, p-value: < 2.2e-16

plot(fit)



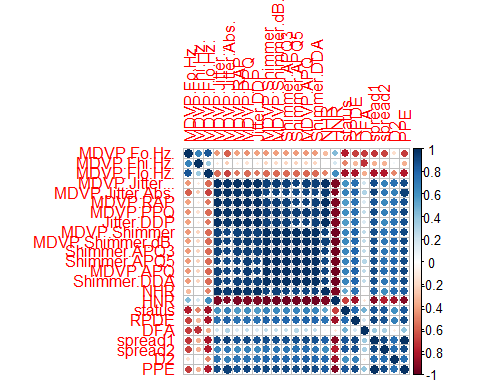
#removing name column  
dataset=subset(dataset,select=-c(name))  
colnames(dataset)

## [1] "MDVP:Fo(Hz)" "MDVP:Fhi(Hz)" "MDVP:Flo(Hz)" "MDVP:Jitter(%)"   
## [5] "MDVP:Jitter(Abs)" "MDVP:RAP" "MDVP:PPQ" "Jitter:DDP"   
## [9] "MDVP:Shimmer" "MDVP:Shimmer(dB)" "Shimmer:APQ3" "Shimmer:APQ5"   
## [13] "MDVP:APQ" "Shimmer:DDA" "NHR" "HNR"   
## [17] "status" "RPDE" "DFA" "spread1"   
## [21] "spread2" "D2" "PPE"

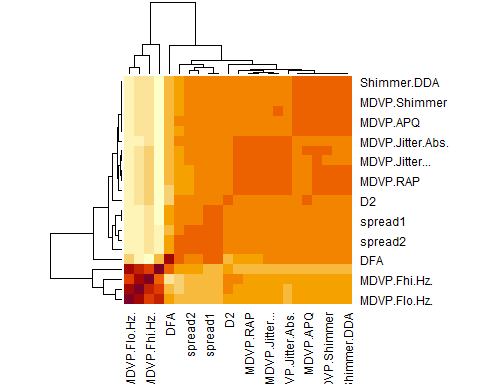
#Importing the required library  
library(corrplot)

## corrplot 0.92 loaded

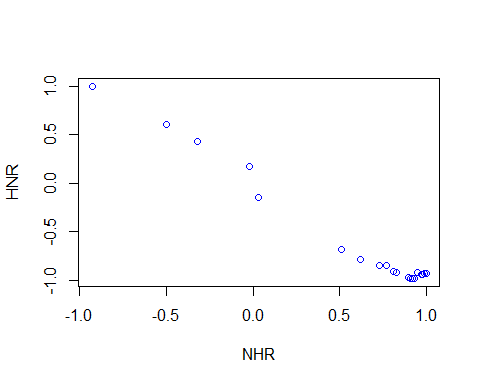
#corrplot(dataset,method='pearson')  
M<-transform(dataset,status=as.numeric(status))  
  
#plotting using pearson method  
  
cor\_data<-cor(M,method=c("pearson"))  
cor\_matrix<-round(cor(cor\_data),2)  
corrplot::corrplot(cor\_matrix, method = "circle")



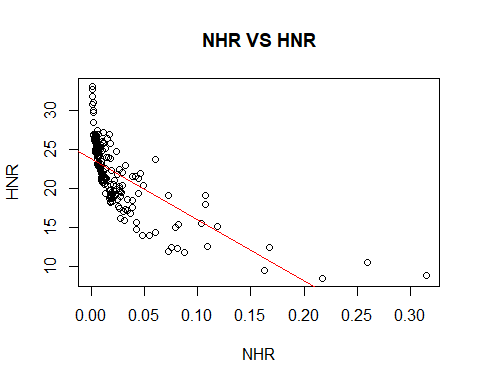
#heatmap of pearson correlation  
heatmap(cor\_matrix)



#regression line and scatterplot  
plot(HNR~NHR, data=cor\_matrix,col='blue')



plot(dataset$NHR,dataset$HNR,main="NHR VS HNR", xlab="NHR",ylab="HNR")  
abline(lm(HNR~NHR,data=dataset),col='red')



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* TRAINING THE MODEL \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
#Importing the required libraries  
library(caTools)  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

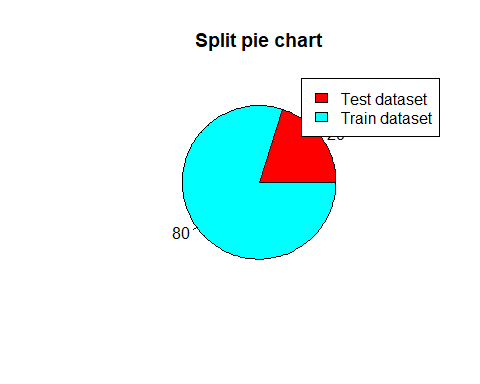
library(e1071)  
  
X <- dataset[, !(names(dataset) %in% c("name", "status"))]  
# Extracting the "status" column as the target variable  
Y <- parkinson$status  
  
# Setting the random seed for reproducibility  
set.seed(2)  
  
# Creating training indices using 80% of the data  
train\_indices <- createDataPartition(Y, times = 1, p = 0.8, list = FALSE)  
  
# Subsetting the data using the training indices  
X\_train <- X[train\_indices, ]  
Y\_train <- Y[train\_indices]  
  
# Subsetting the data using the non-training indices for testing  
X\_test <- X[-train\_indices, ]  
Y\_test <- Y[-train\_indices]  
  
# Check the shape of train\_data  
dim(X\_train)

## [1] 156 22

# Check the shape of test\_data  
dim(X\_test)

## [1] 39 22

#plotting a pie chart of the train and test data  
x<-c(20,80)  
labels<-c("Test dataset","Train dataset")  
piepercent<-round(100\*x/sum(x),1)  
pie(x, labels = piepercent, main = "Split pie chart",col=rainbow(length(x)))  
legend("topright", c("Test dataset","Train dataset"),fill=rainbow(length(x)))



#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PRE-PROCESSING THE DATA \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
  
# Applying feature scaling to the training data  
scaler <- preProcess(X\_train, method = c("center", "scale"))  
  
# Applying feature scaling to the train and test data using the same scaler  
X\_train <- predict(scaler, X\_train)  
X\_test <- predict(scaler, X\_test)  
  
# Converting the training labels to factors  
Y\_train <- as.numeric(as.character(Y\_train))  
  
#Importing the required libraries  
library(ggplot2)  
  
# Creating the SVM model with linear kernel  
model <- svm(x = as.matrix(X\_train), y = Y\_train, kernel = "linear", cache = TRUE)  
  
# Function to extract coefficients and create a hyperplane equation  
get\_hyperplane <- function(model) {  
 coef <- t(model$coefs) %\*% model$SV  
 w <- coef[, -1]  
 b <- coef[, 1]  
 return(list(w = w, b = b))  
}  
  
# Extract hyperplane coefficients  
hyperplane <- get\_hyperplane(model)  
  
# Function to calculate y-coordinate from x-coordinate and hyperplane equation  
calculate\_hyperplane\_y <- function(x, w, b) {  
 return((-b - w[1] \* x) / w[2])  
}  
  
# Subsetting the dataset, removing the "name" and "status" columns  
X <- parkinson[, !(names(parkinson) %in% c("name", "status"))]  
  
# Extracting the "status" column as the target variable  
Y <- parkinson$status  
  
# Check the structure of the dataset  
str(X)

## tibble [195 × 22] (S3: tbl\_df/tbl/data.frame)  
## $ MDVP:Fo(Hz) : num [1:195] 120 122 117 117 116 ...  
## $ MDVP:Fhi(Hz) : num [1:195] 157 149 131 138 142 ...  
## $ MDVP:Flo(Hz) : num [1:195] 75 114 112 111 111 ...  
## $ MDVP:Jitter(%) : num [1:195] 0.00784 0.00968 0.0105 0.00997 0.01284 ...  
## $ MDVP:Jitter(Abs): num [1:195] 0.00007 0.00008 0.00009 0.00009 0.00011 0.00008 0.00003 0.00003 0.00006 0.00006 ...  
## $ MDVP:RAP : num [1:195] 0.0037 0.00465 0.00544 0.00502 0.00655 0.00463 0.00155 0.00144 0.00293 0.00268 ...  
## $ MDVP:PPQ : num [1:195] 0.00554 0.00696 0.00781 0.00698 0.00908 0.0075 0.00202 0.00182 0.00332 0.00332 ...  
## $ Jitter:DDP : num [1:195] 0.0111 0.0139 0.0163 0.015 0.0197 ...  
## $ MDVP:Shimmer : num [1:195] 0.0437 0.0613 0.0523 0.0549 0.0643 ...  
## $ MDVP:Shimmer(dB): num [1:195] 0.426 0.626 0.482 0.517 0.584 0.456 0.14 0.134 0.191 0.255 ...  
## $ Shimmer:APQ3 : num [1:195] 0.0218 0.0313 0.0276 0.0292 0.0349 ...  
## $ Shimmer:APQ5 : num [1:195] 0.0313 0.0452 0.0386 0.0401 0.0483 ...  
## $ MDVP:APQ : num [1:195] 0.0297 0.0437 0.0359 0.0377 0.0447 ...  
## $ Shimmer:DDA : num [1:195] 0.0654 0.094 0.0827 0.0877 0.1047 ...  
## $ NHR : num [1:195] 0.0221 0.0193 0.0131 0.0135 0.0177 ...  
## $ HNR : num [1:195] 21 19.1 20.7 20.6 19.6 ...  
## $ RPDE : num [1:195] 0.415 0.458 0.43 0.435 0.417 ...  
## $ DFA : num [1:195] 0.815 0.82 0.825 0.819 0.823 ...  
## $ spread1 : num [1:195] -4.81 -4.08 -4.44 -4.12 -3.75 ...  
## $ spread2 : num [1:195] 0.266 0.336 0.311 0.334 0.235 ...  
## $ D2 : num [1:195] 2.3 2.49 2.34 2.41 2.33 ...  
## $ PPE : num [1:195] 0.285 0.369 0.333 0.369 0.41 ...

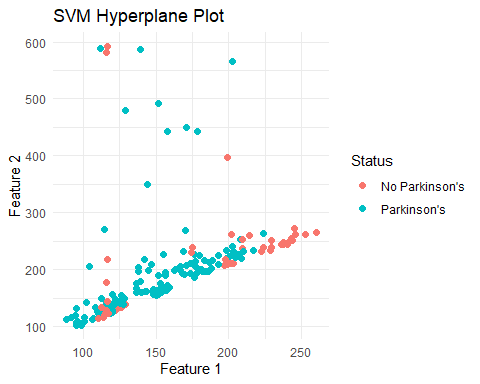
# Set the column names of X to X1 and X2  
colnames(X) <- c("X1", "X2")  
  
# Combine X and Y to create the plot\_data data frame  
plot\_data <- cbind(X, Y)  
  
# Check the structure of the plot\_data data frame  
str(plot\_data)

## 'data.frame': 195 obs. of 23 variables:  
## $ X1: num 120 122 117 117 116 ...  
## $ X2: num 157 149 131 138 142 ...  
## $ NA: num 75 114 112 111 111 ...  
## $ NA: num 0.00784 0.00968 0.0105 0.00997 0.01284 ...  
## $ NA: num 0.00007 0.00008 0.00009 0.00009 0.00011 0.00008 0.00003 0.00003 0.00006 0.00006 ...  
## $ NA: num 0.0037 0.00465 0.00544 0.00502 0.00655 0.00463 0.00155 0.00144 0.00293 0.00268 ...  
## $ NA: num 0.00554 0.00696 0.00781 0.00698 0.00908 0.0075 0.00202 0.00182 0.00332 0.00332 ...  
## $ NA: num 0.0111 0.0139 0.0163 0.015 0.0197 ...  
## $ NA: num 0.0437 0.0613 0.0523 0.0549 0.0643 ...  
## $ NA: num 0.426 0.626 0.482 0.517 0.584 0.456 0.14 0.134 0.191 0.255 ...  
## $ NA: num 0.0218 0.0313 0.0276 0.0292 0.0349 ...  
## $ NA: num 0.0313 0.0452 0.0386 0.0401 0.0483 ...  
## $ NA: num 0.0297 0.0437 0.0359 0.0377 0.0447 ...  
## $ NA: num 0.0654 0.094 0.0827 0.0877 0.1047 ...  
## $ NA: num 0.0221 0.0193 0.0131 0.0135 0.0177 ...  
## $ NA: num 21 19.1 20.7 20.6 19.6 ...  
## $ NA: num 0.415 0.458 0.43 0.435 0.417 ...  
## $ NA: num 0.815 0.82 0.825 0.819 0.823 ...  
## $ NA: num -4.81 -4.08 -4.44 -4.12 -3.75 ...  
## $ NA: num 0.266 0.336 0.311 0.334 0.235 ...  
## $ NA: num 2.3 2.49 2.34 2.41 2.33 ...  
## $ NA: num 0.285 0.369 0.333 0.369 0.41 ...  
## $ Y : num 1 1 1 1 1 1 1 1 1 1 ...

# Check for missing values in the dataset  
sum(is.na(plot\_data))

## [1] 0

# Create the SVM hyperplane plot  
ggplot(plot\_data, aes(x = X1, y = X2, color = factor(Y))) +  
 geom\_point(size = 2) +  
 geom\_abline(intercept = hyperplane$b, slope = -hyperplane$w[1] / hyperplane$w[2], linetype = "dashed") +  
 coord\_cartesian(xlim = range(X$X1), ylim = range(X$X2)) +  
 scale\_color\_discrete(name = "Status", labels = c("0" = "No Parkinson's", "1" = "Parkinson's")) +  
 labs(title = "SVM Hyperplane Plot", x = "Feature 1", y = "Feature 2") +  
 theme\_minimal()



# Making predictions on the training data  
X\_train\_prediction <- predict(model, X\_train)  
  
# Calculating the accuracy of the training predictions  
training\_data\_accuracy <- sum(X\_train\_prediction == Y\_train) / length(Y\_train)  
  
# Printing the performance of the training data  
print(paste("R2 score of training data:", training\_data\_accuracy))

## [1] "R2 score of training data: 0.891025641025641"

# Subsetting the test data to include only the columns present in the training data  
X\_test <- X\_test[, colnames(X\_train)]  
  
# Converting the test labels to factors  
Y\_test <- as.factor(as.character(Y\_test))  
  
# Making predictions on the test data  
X\_test\_prediction <- predict(model, X\_test)  
  
# Calculating the performance of the test predictions  
test\_data\_accuracy <- sum(X\_test\_prediction == Y\_test) / length(Y\_test)  
  
# Printing the accuracy of the test data  
print(paste("R2 score of testing data:", test\_data\_accuracy))

## [1] "R2 score of testing data: 0.923076923076923"

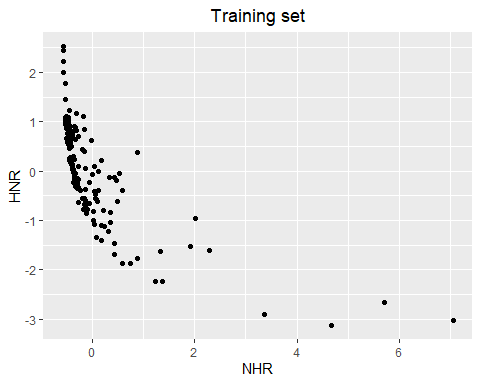
# Preparing the input data for prediction  
input\_data <- c(199.228,209.512,192.091,0.00241,0.00005,0.00134,0.00138,0.00402,  
 0.01015,0.089,0.00504,0.00641,0.00762,0.01513,0.00167,30.940,0.432439,0.742055,-7.682587,0.173319,2.103106,0.068501)  
  
#input\_data=X\_test[1,]  
input\_data\_reshaped <- matrix(input\_data, nrow = 1)  
input\_data\_df <- as.data.frame(input\_data\_reshaped)  
colnames(input\_data\_df) <- colnames(X\_train)  
  
# Scaling the input data using the same scaler  
std\_data <- predict(scaler, input\_data\_df)  
  
# Making predictions on the input data  
prediction <- predict(model, std\_data)  
  
# Printing the prediction result  
print(prediction)

## 1   
## -0.2115037

# Printing the interpretation based on the prediction  
if (prediction == "0") {  
 print("The Person does not have Parkinson's Disease")  
} else {  
 print("The Person has Parkinson's Disease")  
}

## [1] "The Person has Parkinson's Disease"

#Importing the required library  
library(ggplot2)  
  
ggplot()+geom\_point(data=X\_train,aes(NHR,HNR))+ggtitle("Training set") +  
 theme(plot.title=element\_text(hjust=0.5))



ggplot()+geom\_point(data=X\_test,aes(NHR,HNR))+ggtitle("Testing set") +  
 theme(plot.title=element\_text(hjust=0.5))

