Classification of Parkinson's Disease Using Vocal Shimmer, Jitter, and Speech Measurements

Sean Groves and Cody York

Abstract Parkinson's disease (PD) is a progressive neurological disorder that affects movement. Symptoms of PD can include tremor, slowness of movement, stiffness, and impaired balance. There is currently no cure for PD, but early diagnosis and treatment can help to improve quality of life. Machine learning (ML) models can be used to help with the early detection of PD. ML models are trained on data sets that include information about people with PD and people without PD. The models can then be used to predict whether a person is likely to have PD.

In this study, we used four ML models to predict PD: logistic regression, decision tree, random forest, and support vector machine. The data set we used included vocal, shimmer, and jitter data on people with PD and those without. The data set had 23 attributes across 197 instances. We used feature selection to eliminate redundant features from our models.

The results of our study showed that the ML models were able to predict PD with a high degree of accuracy. The random forest model had the highest accuracy, followed by the support vector machine model, the logistic regression model and the decision tree model. These results suggest that ML models can be used to help with the early detection of PD. Further research is needed to validate these results and to develop ML models that can be used in clinical practice.

Introduction

Parkinson's Disease is a neurological disorder effecting more than 6 million people worldwide that impacts individuals motor skills, speech and cognitive abilities.[1] There are currently no tests that are able to use biomarkers to accurately identify and diagnose Parkinson's Disease (PD). Therefore, diagnosis of PD is reliant on clinical observation of the symptoms of PD. Relying on the presence of these symptoms prevents early diagnosis, and costs the effected individuals valuable time in getting treatment to slow the progression of the disease. One option to improve the early diagnosis of PD is to use speech data, which has been found to be one of the earliest detectable PD symptoms.[1] Using the biomedical voice measurements data from [2], we will attempt to use various data analytics methods to classify potential PD cases.

Objectives

- Identify which explanatory variables to use in classifying Parkinson's disease status
- Build a Logistic Regression, Decision Tree, Random Forest, and a Support Vector Machine classification model to classify Parkinson's disease status
- Compare the models to find which is best at classifying Parkinson's disease status

Analysis Plan

Exploratory Data Analysis

Exploring this dataset was accomplished mainly through thorough research of the various features to see what they mean and how they can be used. This gave us a lot of insight into the data as at first we were unsure where to begin with feature selection. We also did basic statistics on the fields to ensure that there weren't outliers and that the data was clean and complete.

Data Analytics Methods

Data Preprocessing was the first step to complete after the exploratory analysis. The data set we used was already very clean and thorough. The preprocessing we did included creating a correlation matrix to find some redundant features in our data set, and removing them from our models. After doing the feature selection, we compared the results to the previous run and determined that the feature selection greatly improved the accuracy of our models.

After using the feature selection we then split the data into a training and a testing set of data. We did 70% for training and the remaining 30% were used for testing. This meant we ended up with 136 records for training and 59 records for testing.

The machine learning models we considered for this analysis were **Logistic Regression**, **Decision Tree**, **Random Forest**, and **Support Vector Machines**. These models are first trained on the training data and then tested against the testing data after the model has been trained. The results that the model gives are then compared to the expected results (whether someone has Parkinson's disease) and rate of miss-classification and confusion matrix are then created. The **rate of miss-classification** is created by subtracting the mean of the amount of correct observations from 1. This gives us a percentage of the time the model made an incorrect prediction. To create the **confusion matrix** we use the **kable()** method to create a table of the amount of observations that have Parkinson's and those that do not for both the testing dataset and the predictions the model gives us.

Data Summary

The dataset we have used for this analysis came form the **UCI Machine Learning Repository** and is referred to as the "Parkinsons Data Set". This dataset was created by Max Little of the University of Oxford in collaboration with the National Centre for Voice and Speech in Denver Colorado. This dataset contains 31 voice recordings of people with Parkinson's disease (PD) and 31 healthy controls. The recordings are in CSV format and each row corresponds to one recording. The main aim of the dataset is to discriminate between PD and healthy controls. [3] This dataset contained 23 attributes across 197 instances.

The features from the dataset we will be looking at include: MDVP.Fhi(Hz), MDVP:Flo(Hz), NHR, RPDE, DFA, spread2 and status.

MDVP.Fhi(Hz) is the maximum vocal fundamental frequency in Hertz and **MDVP:Flo(Hz)** is the minimum vocal fundamental frequency in Hertz. This is calculated using the **Kay Pentax Multidimensional**

Voice Program (MDVP). The fundamental frequency is the lowest frequency of vibration in a sound wave and is what determines the pitch of the sound. [5]

Descriptive Statistics of MDVP.Fhi(Hz):

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   102.1   134.9   175.8   197.1   224.2   592.0
```

Descriptive Statistics of MDVP:Flo(Hz):

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   65.48   84.29  104.31  116.32  140.02  239.17
```

NHR stands for Noise-to-Harmonic Ratio, and it is calculated by dividing the energy of the tonal components in the voice by the energy of the noise components. It is a useful measure for assessing vocal health and can be used to identify changes in vocal quality that may be associated with conditions such as vocal cord nodules, vocal cord paralysis, and Parkinson's disease. [4]

Descriptive Statistics of NHR:

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.000650 0.005925 0.011660 0.024847 0.025640 0.314820
```

RPDE stands for Rescaled Phase Dependent Entropy. It is a measure of the complexity of a time series that is based on the distribution of the phases of the constituent frequencies. It is a useful measures for assessing the complexity of a time series. They can be used to identify changes in the complexity of a time series that may be associated with changes in the underlying system. For example, RPDE has been used to study the changes in the complexity of the voice that are associated with Parkinson's disease. [3]

Descriptive Statistics of RPDE:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2566  0.4213  0.4960  0.4985  0.5876  0.6852
```

DFA stands for Detrended fluctuation analysis (DFA) is a measure of the similarity of noise in speech signals. It is used to identify voice disorders by measuring the scaling exponent of the noise. A higher scaling exponent indicates a more disordered signal, which is associated with voice disorders. [3]

Descriptive Statistics of DFA:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5743  0.6748  0.7223  0.7181  0.7619  0.8253
```

spread2 is a measure of the unevenness of the distribution of fundamental frequencies in a voice signal. It is calculated as the standard deviation of the fundamental frequencies in the signal. A higher spread2 value indicates a more variable fundamental frequency, which is associated with voice disorders. [6]

Descriptive Statistics of spread2:

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.006274 0.174350 0.218885 0.226510 0.279234 0.450493
```

status is a binary attribute that states whether the person has Parkinson's (1) or does not have Parkinson's (0).

Descriptive Statistics of status:

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  1.0000  1.0000  0.7538  1.0000  1.0000
```

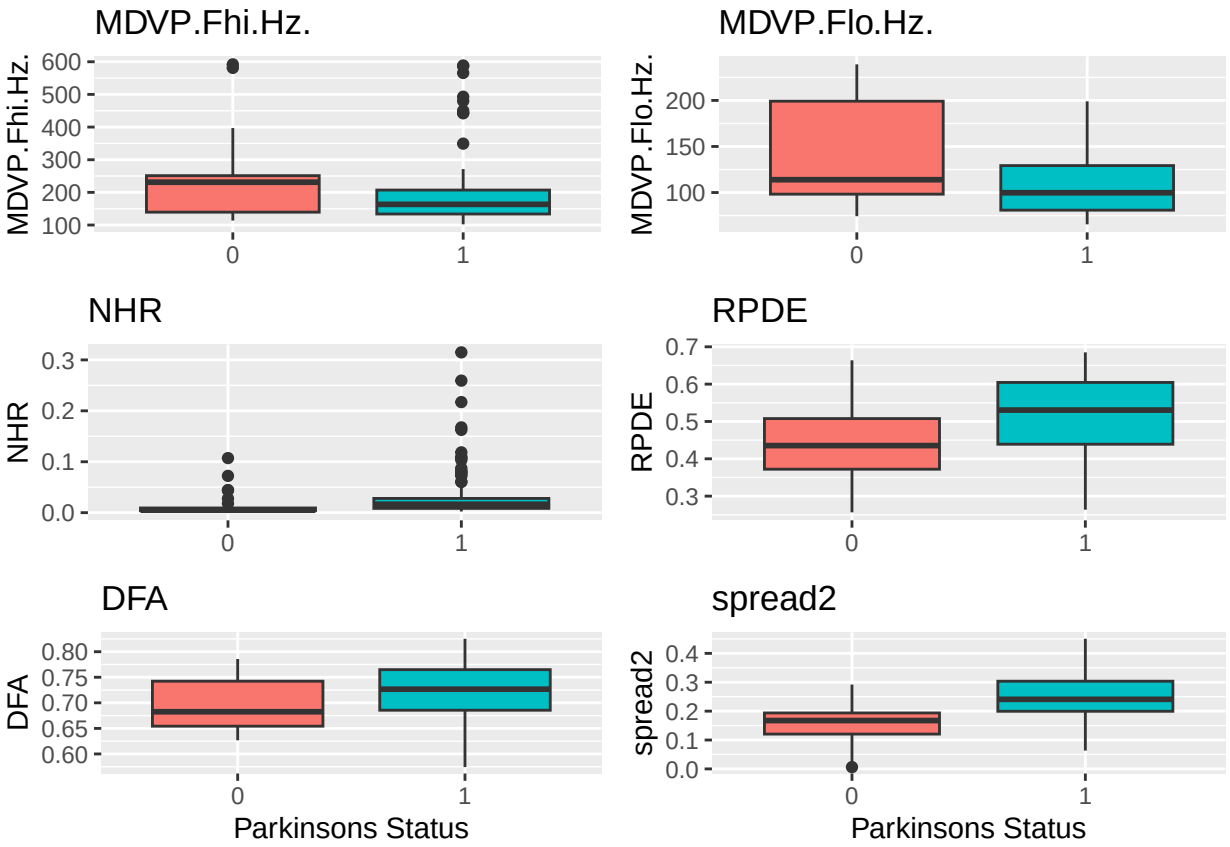


Figure 1: Boxplots of all the features grouped by PD status (1 has PD 0 does not have PD)

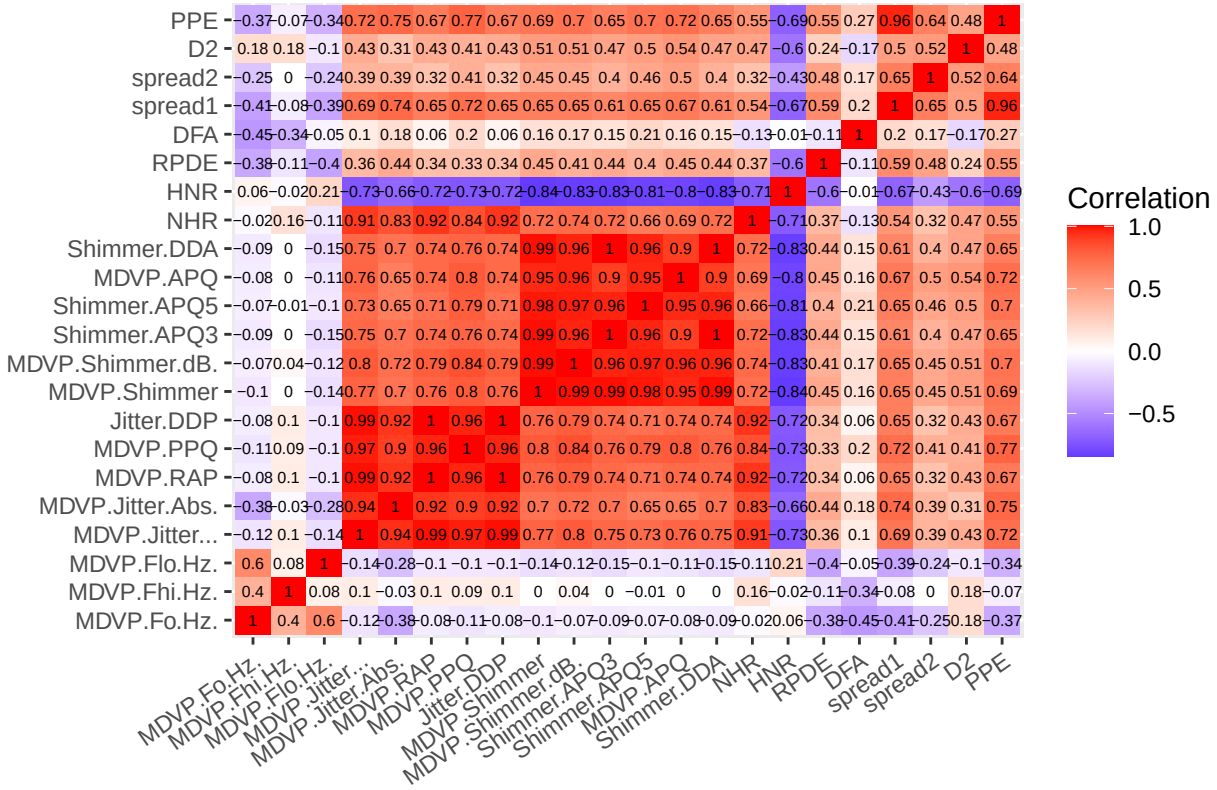
The status of PD does appear to have an impact on the distribution of the features. (Figure 1) This reinforces our selection of features, and leads us to believe they can be used to accurately model the classification of PD.

Data Analysis and Results

Feature Selection

To select the features to be used in the models, we first examined the correlation matrix from all 22 features (Figure 2). Since the features all came from biomedical voice measurements, some of the features were likely to be heavily correlated, and thus redundant. Looking at the correlation heat map in figure 2, there are a large number of the features that are significantly correlated. Using the findCorrelation() function we identified the features with an absolute correlation over 0.5, and excluded them from our feature list. This left us with the 6 features that we used for our models. Getting rid of these redundant features improves the models accuracy, and reduces over-fitting.

Figure 2: Correlation Heat Map



Machine Learning Algorithms

We used Logistic Regression, Decision Tree, Random Forest (RF), and Support Vector Machines (SVM) models to classify our data as having PD or not. The logistic regression model assigns a value of the predicted probability of having PD for each observation in the testing data set. We then assigned all the cases with probability > 0.5 to 1 to represent a prediction of having PD, and all other cases as 0 to represent a prediction of not having PD. The decision tree, RF, and SVM all make predictions of either 0 for not having PD or 1 for having PD, for each case in the testing data set. We then made confusion matrices (Figure 3) for all the models to show the counts of correctly and incorrectly predicted cases of PD and no PD for the testing data sets. The top left corner of each confusion matrix indicates a correctly predicted case of PD, while the bottom right corner indicates a correct predicted case of not having PD. The decision tree had the largest number of false negatives, while the RF had the fewest false negatives, and the most correct PD predictions.

The decision tree had the largest miss-classification rate at 18.6%, the logistic regression model had the next highest at 11.9%, the SVM had the second lowest miss-classification rate at 10.2%, and the RF was the lowest at 6.8%.(Table 1) Thus based on our testing data set, the Random Forest model was the best at classifying cases of PD.

Discussion

As demonstrated above, machine learning algorithms can be used to accurately classify cases of Parkinson's disease using data from voice recordings. From the models we tested, the Random Forest classification was the most accurate(Table 1). Our RF model only produced 1 case of a false negative. Using voice recordings is a non-intrusive method, and could potentially be implemented regularly at check up visits of patients with the highest risk factors (mainly age). Improvements for this model could come from several different options. One such option

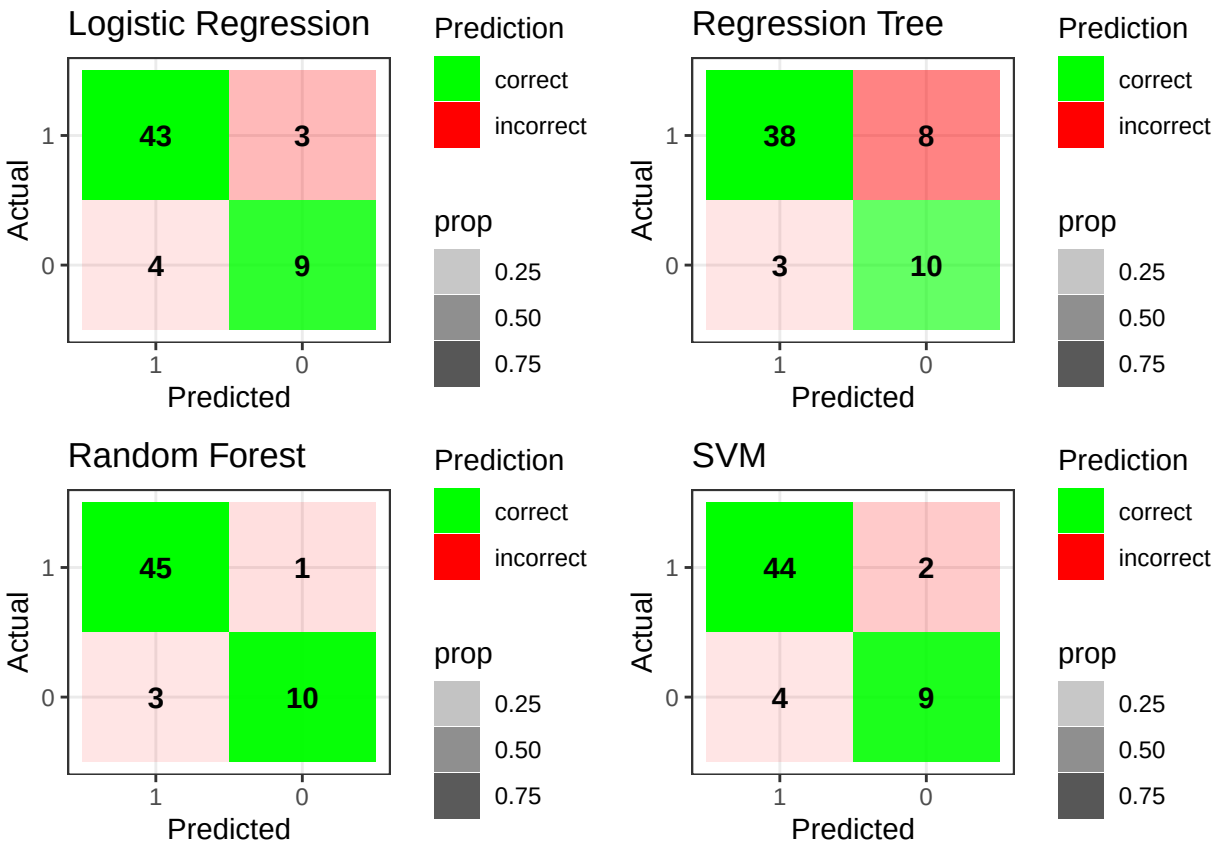


Figure 3: Confusion matrices of the Logistic Regression, Decision Tree, Random Forest, and SVM models.

Green squares are correct predictions and red squares are incorrect.

Table 1: Table 1: The miss-classification rates of the models.

Test	Error Rate
Logistic	0.119
Regression tree	0.186
RF Tree	0.068
SVM	0.102

could be an improved feature selection process. We used an unsupervised method of redundancy reduction using correlation, but there are numerous options for feature selection including supervised methods like

recursive feature elimination, or dimensionality reduction methods like principal component analysis. Future works can explore these methods to find the features that maximize the models performance. Another area for future exploration is using our model in a study to see how accurately we can predicted PD before a patient would otherwise be clinically diagnosed. Speech data has been found to be among the earliest detectable symptoms of PD[1]. A study that tests our model with speech data of people without other PD symptoms, and follows them over a large time (5+ years), could be instrumental in seeing if our model can in fact predict PD before clinical diagnosis.

Conclusion

We have shown that a random forest machine learning algorithm is a viable way to predict Parkinson's disease using voice recording data. Our model was trained on a limited data set containing only 197 attributes, but was still able to predict PD over 93% of the time. Future works can be done to expand the available data sets, and hold a study to implement a model in the clinical setting.

References

- *[1]*: Zhang, H.-H., Yang, L., Liu, Y., Wang, P., Yin, J., Li, Y., Qiu, M., Zhu, X., & Yan, F. (2016, November 16). Classification of parkinson's disease utilizing multi-edit nearest-neighbor and ensemble learning algorithms with speech samples. PubMed Central. Retrieved April 30, 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5112697/
- *[2]*: 'Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection', Little MA, McSharry PE, Roberts SJ, Costello DAE, Moroz IM. BioMedical Engineering OnLine 2007, 6:23 (26 June 2007) https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/
- *[3]*: Max A. Little, Patrick E. McSharry, Eric J. Hunter, Lorraine O. Ramig (2008), 'Suitability of dysphonia measurements for telemonitoring of Parkinson's disease', IEEE Transactions on Biomedical Engineering (to appear).
- *[4]*: Little, Max A. (2019). "A Tutorial on Voice Analysis in R". RPubs. Retrieved from https://rstudio-pubs-static.s3.amazonaws.com/197257_4ecb7314bc1f4619b59ab44a1374ce7f.html
- *[5]*: Wu Y, Chen P, Yao Y, Ye X, Xiao Y, Liao L, Wu M, Chen J. Dysphonic Voice Pattern Analysis of Patients in Parkinson's Disease Using Minimum Interclass Probability Risk Feature Selection and Bagging Ensemble Learning Methods. Comput Math Methods Med. 2017;2017:4201984. doi: 10.1155/2017/4201984. Epub 2017 May 3. PMID: 28553366; PMCID: PMC5434464.
- *[6]*: Vance, Matt. (2019). "A Tutorial on Voice Analysis in R". RPubs. Retrieved from https://rpubs.com/VanceMatt/197257

Appendix

```
# set.seed(123)
# ################################################### using selected features
# test_Park <- parkinsons_data %>%
#   mutate(parkinson = status)%>%
#   dplyr::select(-c(name,status))
#
# ######## find highly correlated features
# cor_matrix2 <-cor(test_Park[,1:22])
# findCorrelation(cor_matrix2, cutoff=0.5)
#
# #Cor mat heat map
```

```
# round_cor_mat <- round(cor_matrix2,2)
# melted_corr_mat <- melt(round_cor_mat)
#
# heat.map <- ggplot(data = melted_corr_mat, aes(x=Var1, y=Var2,fill=value)) +
#   theme(axis.title.y=element_blank(),axis.title.x=element_blank(),
#         axis.text.x = element_text(angle=35, vjust=1, hjust=1)) +
#   scale_fill_gradient2(low = "blue",
#                        mid = "white",
#                        high = "red")+
#   labs(fill="Correlation", title = "Figure 2: Correlation Heat Map")+
#   geom_tile() +
#   geom_text(aes(Var2, Var1, label = value),
#             color = "black", size = 2)
#
# heat.map
#
# #keep only non-highly correlated features
# #MDVP.Fhi.Hz , MDVP.Flo.Hz.,  NHR , RPDE, DFA , spread2, parkinson(status)
# park_subset<-test_Park[,c(2,3,15,17,18,20,23)]
#
# sub_train <- sample(nrow(park_subset), nrow(park_subset) * 0.7)
#
# predict_sub_parkinsons.train <- park_subset[sub_train, ]
#
#
# predict_sub_parkinsons.test <- park_subset[-sub_train, ]
#
#
#
# # glm
# glm.park2 <- glm(parkinson ~ ., family = binomial ,data = predict_sub_parkinsons.train)
# glm.park2
#
# glm.probs2 <- predict(glm.park2,predict_sub_parkinsons.test, type = 'response' )
# glm.pred2=rep(0,nrow(predict_sub_parkinsons.test))
# glm.pred2[glm.probs2 >.5]=1
#
# table(glm.pred2, predict_sub_parkinsons.test$parkinson)
# # rate of misclassification
#
# sub_glm_error_rate<- 1-mean(glm.pred2==predict_sub_parkinsons.test$parkinson)
# paste("The logistic missclasification rate is:", sub_glm_error_rate)
#
#
# # decision tree
#
# sub_park_tree <- tree(as.factor(parkinson) ~., data = predict_sub_parkinsons.train)
# plot(sub_park_tree)
# text(sub_park_tree, pretty = 0)
#
#
# sub_tree.pred <- predict(sub_park_tree,predict_sub_parkinsons.test, type = 'class')
# table(sub_tree.pred,predict_sub_parkinsons.test$parkinson)
```

```
# # rate of misclassification
# sub_tree_error_rate<- (1-mean(sub_tree.pred==predict_sub_parkinsons.test$parkinson))
# paste("The regression tree  missclasification rate is:", sub_tree_error_rate)
#
#
# ## RF tree
#
# sub_rf.parkinsons <- randomForest(as.factor(parkinson) ~ ., data = predict_sub_parkinsons.train, mtry
# sub_rf.parkinsons
# varImpPlot(sub_rf.parkinsons)
#
# #using testing data set
# rf.predict <- predict(sub_rf.parkinsons,predict_sub_parkinsons.test[-7], type = 'response')
# table(rf.predict,predict_sub_parkinsons.test$parkinson)
# rf_error_rate<- (1-mean(rf.predict==predict_sub_parkinsons.test$parkinson))
# rf_error_rate
#
# paste("The random forest tree  missclasification rate is:", sub_rf.parkinsons$err.rate[500,1])
#
#
#
# ### SVM
# sub_tune.out <- tune(svm, as.factor(parkinson) ~., data=predict_sub_parkinsons.train, kernel ="radial
#                  ranges =list(cost=c(0.1 ,1 ,10 ,100 ,1000),
#                               gamma=c(0.5,1,2,3,4) ))
# summary(sub_tune.out)
#
#
# svm.parkinsons_sub <- svm(as.factor(parkinson) ~ .,
#                       data = predict_sub_parkinsons.train,
#                       kernel = "radial",
#                       gamma = 0.5,
#                       cost = 100)
#
# svm.parkinsons.predictions_sub <- predict(svm.parkinsons_sub, predict_sub_parkinsons.test)
# summary(svm.parkinsons.predictions_sub)
#
# table( svm.parkinsons.predictions_sub,predict_sub_parkinsons.test$parkinson)
#
# svm_error_rate<- (1-mean(svm.parkinsons.predictions_sub==predict_sub_parkinsons.test$parkinson))
# paste("The svm  missclasification rate is:", svm_error_rate)
#
#
# ###############error rate table
# model_type <- c('Logistic','Regression tree','RF Tree','SVM')
# error_rate <- c(sub_glm_error_rate,sub_tree_error_rate,rf_error_rate,svm_error_rate)
# error_table <- data.frame(model_type,error_rate)
# #error_table
#
# error.kable <- kable(error_table, caption = "Table 1: The miss-classification rates of the models.",
#       escape = F,
#       digits = 3,
#       longtable = T,
```

```
#       col.names = c("Test", "Error Rate")) %>%
#   kable_styling(latex_options = "striped",full_width = FALSE)
#
# ############################## confusion matrix visualizations
#
# #SVM    #####
# svm_tab <-data.frame(table(svm.parkinsons.predictions_sub,predict_sub_parkinsons.test$parkinson))
#
# plotTable <- svm_tab %>%
#   mutate(Prediction = ifelse(svm_tab$Var2 == svm_tab$svm.parkinsons.predictions_sub, "correct", "inco
#   group_by(svm.parkinsons.predictions_sub) %>%
#   mutate(prop = Freq/sum(Freq))
#
#
# plot1 <- ggplot(data = plotTable, mapping = aes(x = svm.parkinsons.predictions_sub, y = Var2, fill = 
#   geom_tile() +
#   geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
#   scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
#   theme_bw() +
#   xlim(rev(levels(svm_tab$svm.parkinsons.predictions_sub)))+
#   labs(x="Predicted", y='Actual',title = "SVM")
#
#
# #Log Reg #####
# log_tab <-data.frame(table(glm.pred2, predict_sub_parkinsons.test$parkinson))
#
# log_plotTable <- log_tab %>%
#   mutate(Prediction = ifelse(log_tab$Var2 == log_tab$glm.pred2, "correct", "incorrect")) %>%
#   group_by(glm.pred2) %>%
#   mutate(prop = Freq/sum(Freq))
#
#
# plot2 <- ggplot(data = log_plotTable, mapping = aes(x = glm.pred2, y = Var2, fill = Prediction, alpha
#   geom_tile() +
#   geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
#   scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
#   theme_bw() +
#   xlim(rev(levels(log_tab$glm.pred2)))+
#   labs(x="Predicted", y='Actual',title = 'Logistic Regression')
#
#
# #Reg Tree
#
#
# tree_tab <-data.frame(table(sub_tree.pred, predict_sub_parkinsons.test$parkinson))
#
# tree_plotTable <- tree_tab %>%
#   mutate(Prediction = ifelse(tree_tab$Var2 == tree_tab$sub_tree.pred, "correct", "incorrect")) %>%
#   group_by(sub_tree.pred) %>%
#   mutate(prop = Freq/sum(Freq))
#
#
# plot3 <- ggplot(data = tree_plotTable, mapping = aes(x = sub_tree.pred, y = Var2, fill = Prediction, 
```

```
#   geom_tile() +
#   geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
#   scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
#   theme_bw() +
#   xlim(rev(levels(tree_tab$sub_tree.pred)))+
#   labs(x="Predicted", y='Actual', title = 'Regression Tree')
#
# # RF
# rf_tab <-data.frame(sub_rf.parkinsons$confusion) %>%
#   dplyr::select(-class.error)%>%
#   mutate(actual= c(0,1))%>%
#   pivot_longer(cols = c('X0','X1'),
#                names_to = 'rf.pred',
#                values_to = 'Freq')%>%
#   mutate(rf.pred= case_when(rf.pred=="X0"~0,rf.pred=="X1"~1))%>%
#   mutate(rf.pred= factor(rf.pred))%>%
#   mutate(actual= factor(actual))
#
#
# rf_plotTable <- rf_tab %>%
#   mutate(Prediction = ifelse(rf_tab$actual == rf_tab$rf.pred, "correct", "incorrect")) %>%
#   group_by(rf.pred) %>%
#   mutate(prop = Freq/sum(Freq))
#
#
# plot4 <- ggplot(data = rf_plotTable, mapping = aes(x = rf.pred, y = actual, fill = Prediction, alpha =
#   geom_tile() +
#   geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
#   scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
#   theme_bw() +
#   labs(x="Predicted", y='Actual', title = "Random Forest")+
#   xlim(rev(levels(rf_tab$rf.pred)))
#
# #rf with test data
#
#
# rf2_tab <-data.frame(table(rf.predict, predict_sub_parkinsons.test$parkinson))
#
# rf2_plotTable <- rf2_tab %>%
#   mutate(Prediction = ifelse(rf2_tab$Var2 == rf2_tab$rf.predict, "correct", "incorrect")) %>%
#   group_by(rf.predict) %>%
#   mutate(prop = Freq/sum(Freq))
#
#
# plot5 <- ggplot(data = rf2_plotTable, mapping = aes(x = rf.predict, y = Var2, fill = Prediction, alph
#   geom_tile() +
#   geom_text(aes(label = Freq), vjust = .5, fontface  = "bold", alpha = 1) +
#   scale_fill_manual(values = c(correct = "green", incorrect = "red")) +
#   theme_bw() +
#   xlim(rev(levels(rf2_tab$rf.predict)))+
#   labs(x="Predicted", y='Actual', title = 'Random Forest')
#
# grid.arrange(plot2,plot3,plot5,plot1)
```

```
#
# ##### boxplots for EDA
#
# eda.plot1 <- ggplot(park_subset,aes(x=factor(parkinson) , y=MDVP.Fhi.Hz., fill =factor(parkinson)))+
#   geom_boxplot()+
#   theme(axis.title.x=element_blank(),legend.position="none")+
#   labs(title = "MDVP.Fhi.Hz.")
#
# #eda.plot1
#
# eda.plot2 <- ggplot(park_subset,aes(x=factor(parkinson) , y=MDVP.Flo.Hz., fill =factor(parkinson)))+
#   geom_boxplot()+
#   theme(axis.title.x=element_blank(),legend.position="none")+
#   labs(title = "MDVP.Flo.Hz.")
#
# #eda.plot2
#
# eda.plot3 <- ggplot(park_subset,aes(x=factor(parkinson) , y=NHR, fill =factor(parkinson)))+
#   geom_boxplot()+
#   theme(axis.title.x=element_blank(),legend.position="none")+
#   labs(title = "NHR")
#
#
# #eda.plot3
#
# eda.plot4 <- ggplot(park_subset,aes(x=factor(parkinson) , y=RPDE, fill =factor(parkinson)))+
#   geom_boxplot()+
#   theme(axis.title.x=element_blank(),legend.position="none")+
#   labs(title = "RPDE")
#
#
# #eda.plot4
#
# eda.plot5 <- ggplot(park_subset,aes(x=factor(parkinson) , y=DFA, fill =factor(parkinson)))+
#   geom_boxplot()+
#   theme(legend.position="none")+
#   labs(title = "DFA", x ="Parkinsons Status")
#
#
# #eda.plot5
#
# eda.plot6 <- ggplot(park_subset,aes(x=factor(parkinson) , y=spread2, fill =factor(parkinson)))+
#   geom_boxplot()+
#   theme(legend.position="none")+
#   labs(title = "spread2", x ="Parkinsons Status")
#
# #eda.plot6
#
#
# grid.arrange(eda.plot1,eda.plot2,eda.plot3,eda.plot4,eda.plot5,eda.plot6)
```