Gender Recognition by Voice

Aselac

4/24/2020

Project Overview

In this project we will be using a data set available at kaggle website (https://www.kaggle.com/primaryobjects/voicegender) to predict voice as male or female based on the acoustic properties of voice and speech. Voice samples collected from total of 3168 males and females will be tested under three different prediction models and accuracy of each model will be compared at the end.

Loading libraries and the data file.

```
if (!require(tidyverse)) install.packages('tidyverse')
if (!require(caret)) install.packages('caret')
if (!require(rpart)) install.packages('rpart')
if (!require(knitr)) install.packages('knitr')
if (!require(knitr)) install.packages('matrixStats')
library(tidyverse)
library(caret)
library(rpart)
library(knitr)
library(matrixStats)


data<-read.csv("https://raw.githubusercontent.com/aselac/Voice_Recognition/master/voice.csv")
```

Analysis

We can see that data set has 3168 observations and 21 variables. The last variable gender is a factor with two levels of "male" and "female", which we are going to predict as outcome.

```
colnames(data)[colnames(data) == "label"] <- "gender"
str(data)
```

```
## 'data.frame':    3168 obs. of  21 variables:
##  $ meanfreq: num  0.0598 0.066 0.0773 0.1512 0.1351 ...
##  $ sd      : num  0.0642 0.0673 0.0838 0.0721 0.0791 ...
##  $ median  : num  0.032 0.0402 0.0367 0.158 0.1247 ...
##  $ Q25     : num  0.0151 0.0194 0.0087 0.0966 0.0787 ...
##  $ Q75     : num  0.0902 0.0927 0.1319 0.208 0.206 ...
##  $ IQR     : num  0.0751 0.0733 0.1232 0.1114 0.1273 ...
##  $ skew    : num  12.86 22.42 30.76 1.23 1.1 ...
##  $ kurt    : num  274.4 634.61 1024.93 4.18 4.33 ...
```

```
##  $ sp.ent  : num  0.893 0.892 0.846 0.963 0.972 ...
##  $ sfm     : num  0.492 0.514 0.479 0.727 0.784 ...
##  $ mode    : num  0 0 0 0.0839 0.1043 ...
##  $ centroid: num  0.0598 0.066 0.0773 0.1512 0.1351 ...
##  $ meanfun : num  0.0843 0.1079 0.0987 0.089 0.1064 ...
##  $ minfun  : num  0.0157 0.0158 0.0157 0.0178 0.0169 ...
##  $ maxfun  : num  0.276 0.25 0.271 0.25 0.267 ...
##  $ meandom : num  0.00781 0.00901 0.00799 0.2015 0.71281 ...
##  $ mindom  : num  0.00781 0.00781 0.00781 0.00781 0.00781 ...
##  $ maxdom  : num  0.00781 0.05469 0.01562 0.5625 5.48438 ...
##  $ dfrange : num  0 0.04688 0.00781 0.55469 5.47656 ...
##  $ modindx : num  0 0.0526 0.0465 0.2471 0.2083 ...
##  $ gender  : Factor w/ 2 levels "female","male": 2 2 2 2 2 2 2 2 2 2 ...
```

There are no missing values in the data set.

```
sum(is.na(data))
```

```
## [1] 0
```

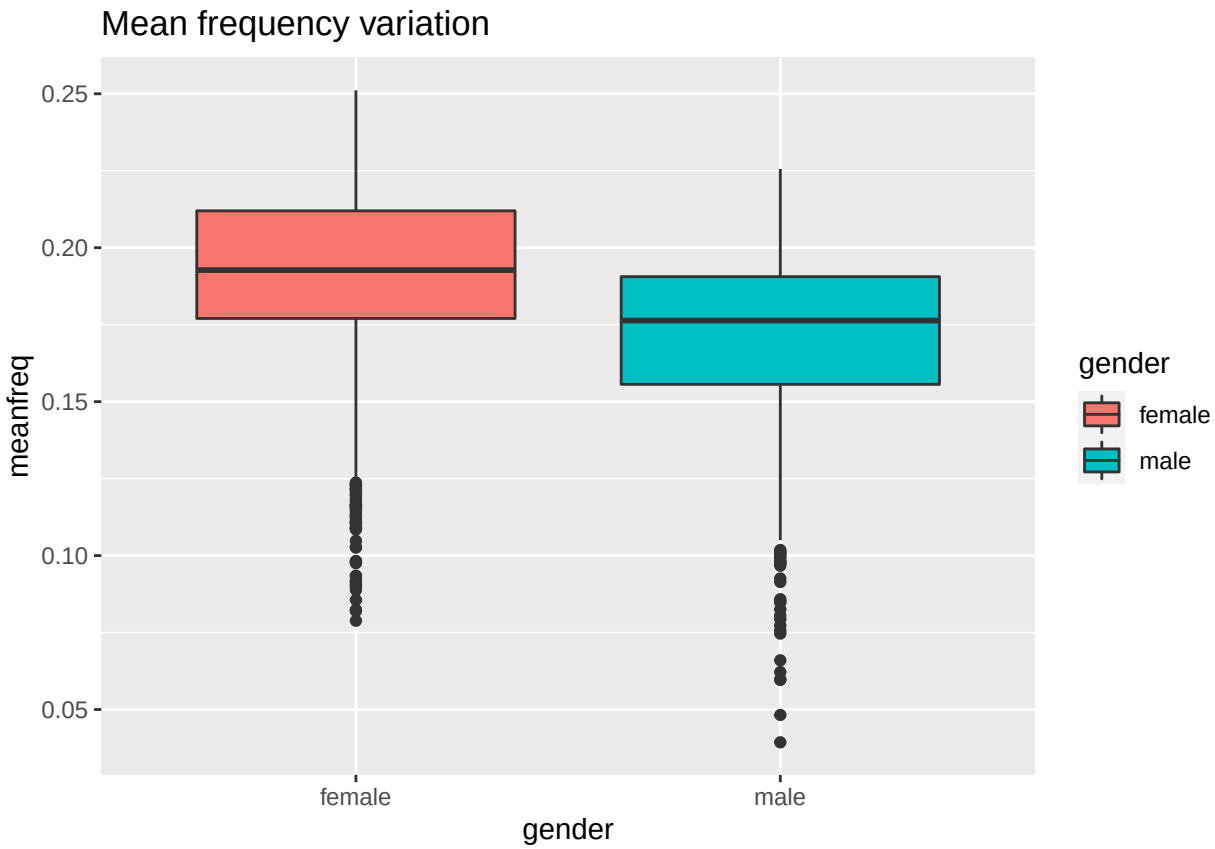
The data set has equal numbers of “males” and “females”

```
data%>%group_by(gender)%>%summarise(count=n())%>%kable
```

gender	count
female	1584
male	1584

For voice recognition mean frequency plays a major role. Variation of mean frequency across genders are shown below.

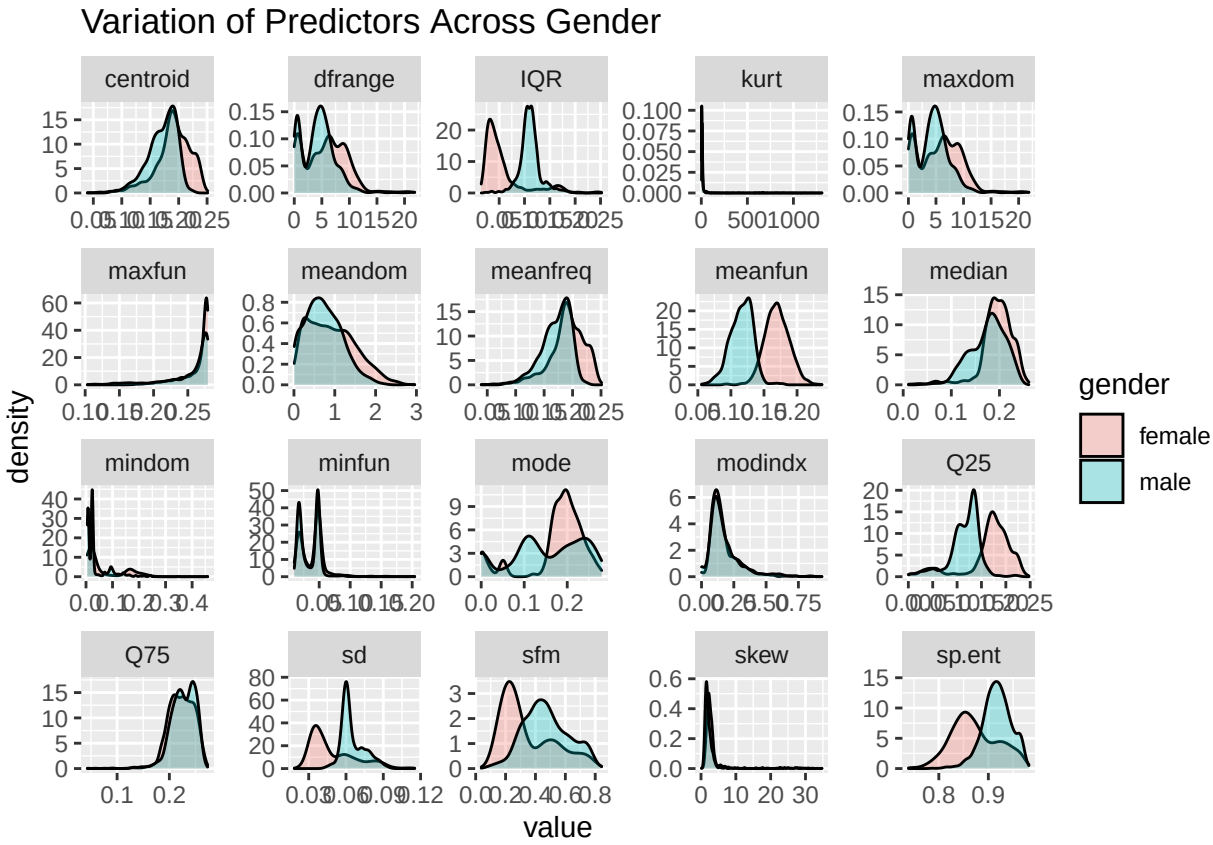
```
data%>%ggplot(aes(gender,meanfreq, fill=gender))+geom_boxplot()+ggtitle("Mean frequency variation")
```



We see an overlap in the plot.

Lets see how 20 predictors are distributed across genders with below graphs.

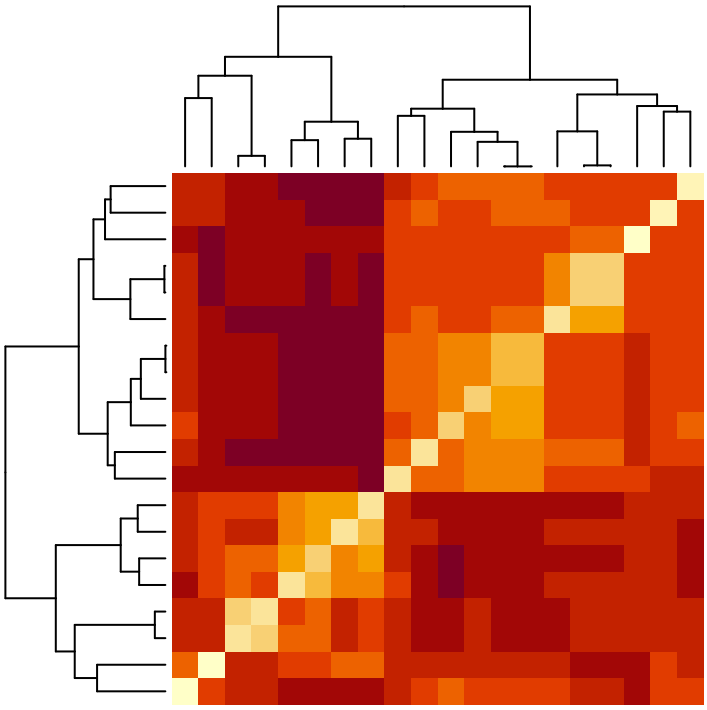
```
data%>%gather(key,value,1:20)%>%ggplot(aes(value,fill=gender))+geom_density(alpha=0.3)+
  facet_wrap(~key,scales = "free")+ggtitle("Variation of Predictors Across Gender")
```



We can see almost 7 predictors are overlapped

Another powerful visualization tool is a heat map which helps to find clusters in the data set. Following code generates heatmap for the predictors.

```
x_centered <- sweep(data[-21], 2, colMeans(data[-21])) #reduce colMeans
x_scaled <- sweep(x_centered, 2, colSds(as.matrix(data[-21])), FUN = "/") # divide by Sd
d_features <- dist(t(x_scaled))
heatmap(as.matrix(d_features), labRow = NA, labCol = NA)
```



We see a correlation in certain predictors. Lets see what predictors are correlated. We can consider values over 0.7 as highly correlated.

```
cor_data<-cor(data[-21])
highly_cor<-colnames(data)[findCorrelation(cor_data, cutoff = 0.7)]
highly_cor
```

```
## [1] "meanfreq" "centroid" "Q25"      "sd"       "median"   "sfm"      "maxdom"
## [8] "dfrange"  "skew"
```

We can see nine predictors are highly corelated. Removing predictors that are highly correlated since highly correlated features can mask the interactions with below code.

```
data_clean<-data[, which(!colnames(data) %in% highly_cor)]
```

Data splitting

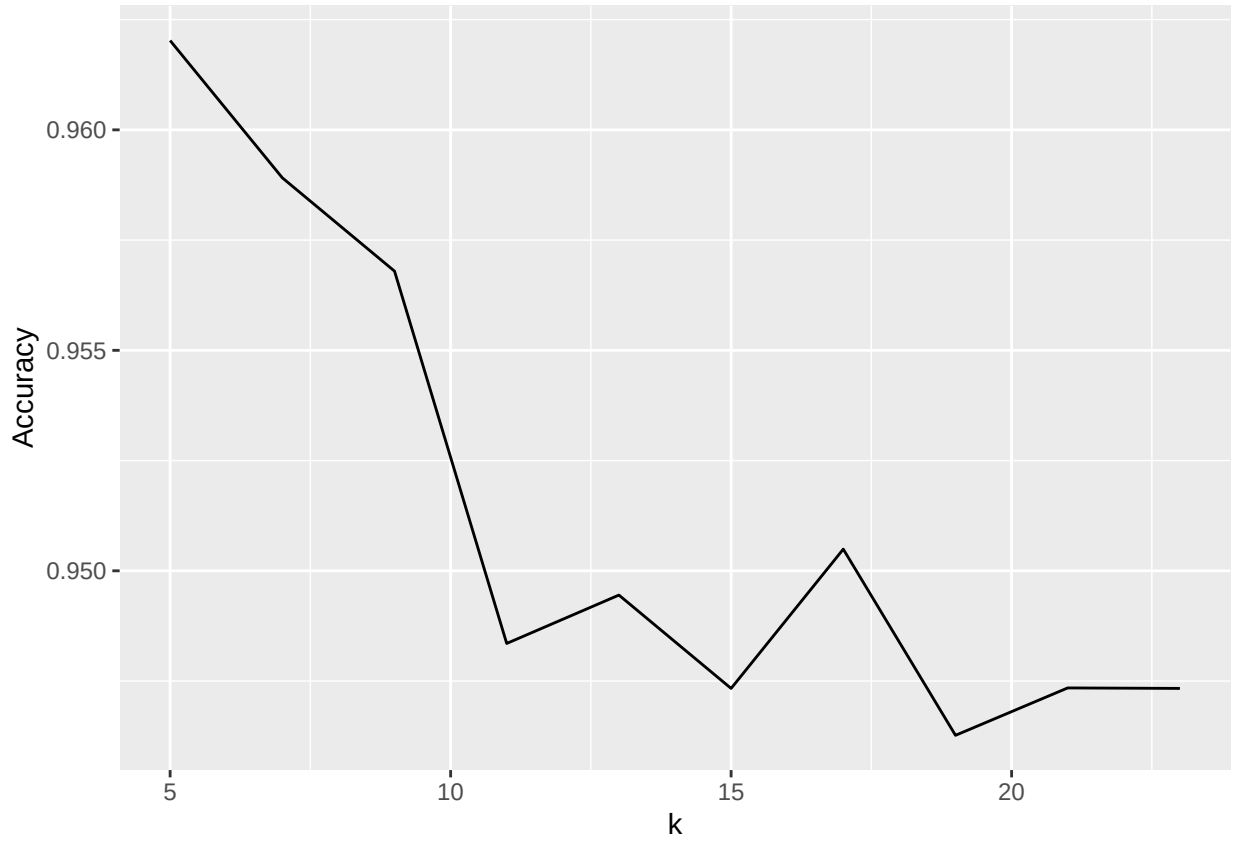
The function createDataPartition of the caret package can be used to create balanced splits of the data. Given the categorical outcome data set will be split into 30% for testing and 70% for training.

```
set.seed(1, sample.kind = "Rounding")
test_index<-createDataPartition(data_clean$gender,  times = 1,p=0.7,list=FALSE)
test_set<-data_clean[test_index,]
train_set<-data_clean[-test_index,]
```

Model 1: K nearest neighbors (KNN)

The KNN algorithm assumes that similar things exist in close proximity. It works well with numerical variables since we are dealing with distances. Lets apply the KNN model and see the accuracy.

```
set.seed(1, sample.kind = "Rounding")
train_knn<-train(gender~., method="knn", preProcess=c("center","scale"),tuneLength=10,
                 trControl=trainControl(method="cv",number=10),data = train_set)
knn_predict<-predict(train_knn,test_set)
knn<-mean(knn_predict==test_set$gender)
train_knn$results%>%ggplot(aes(k,Accuracy))+geom_line()
```




```
knn
```

```
## [1] 0.9688909
```

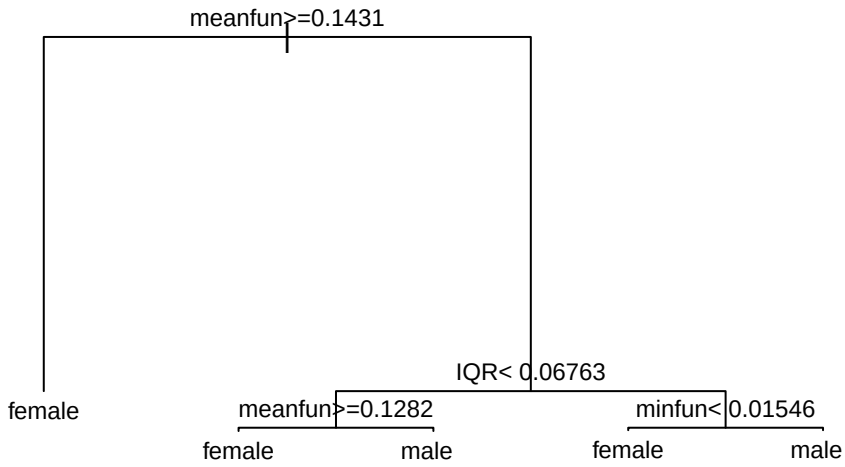
It provides accuracy of 97% with best value with 5 closest neighbors in order to determine the category.

Model 2: rpart

The rpart model is for creating a decision tree. It is useful because the results are very easily interpretable.

```
set.seed(1, sample.kind = "Rounding")
train_rpart<-train(gender~., method="rpart", tuneGrid=data.frame(cp=seq(0,0.05,0.002)),data = train_set)
rpart_predict<-predict(train_rpart,test_set)
rpart<-mean(rpart_predict==test_set$gender)

plot(train_rpart$finalModel, margin = 0.1)
text(train_rpart$finalModel, cex = 0.75)
```




```
rpart
```

```
## [1] 0.9634806
```

The rpart model provides an accuracy of 96%

Model 3: Random Forest

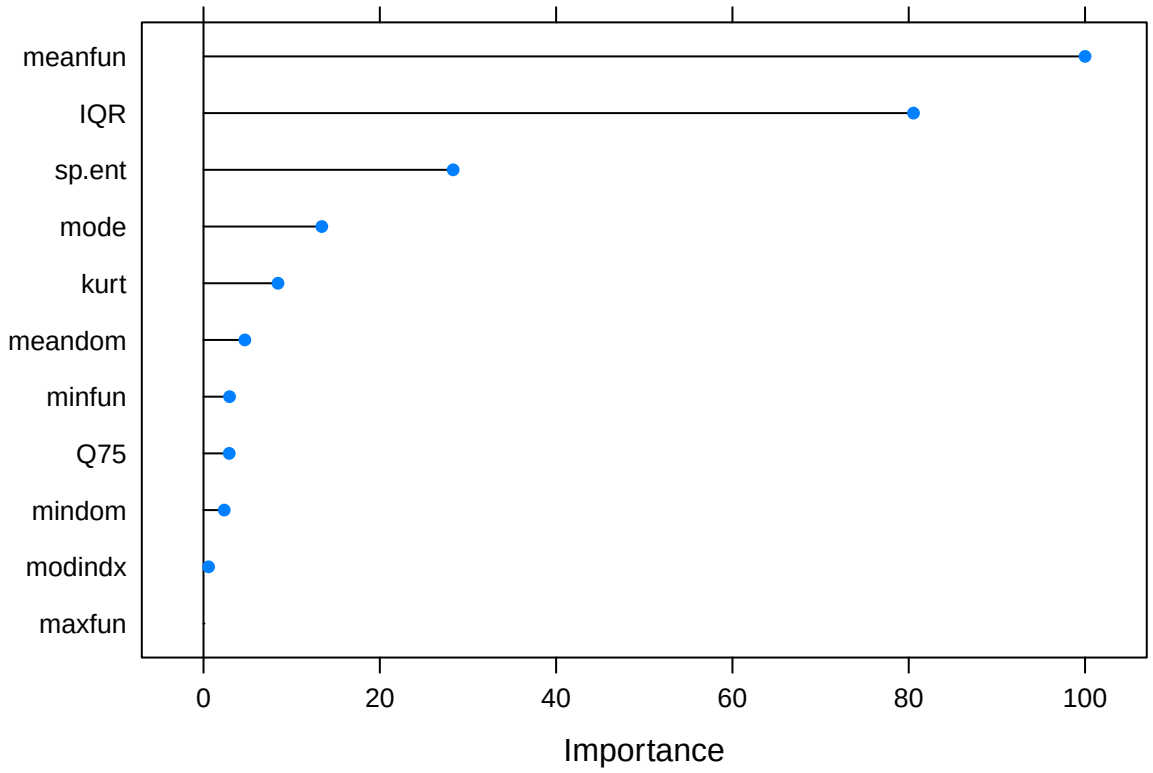
Random forests addresses the shortcomings of decision trees by improving prediction performance and reducing instability by averaging multiple decision trees. Lets apply our data set to this model

```
set.seed(1, sample.kind = "Rounding")
train_rf<-train(gender~.,method = "rf", tuneGrid=data.frame(mtry=seq(1,7)), ntree= 150, data=train_set)
rf_predict<-predict(train_rf,test_set)
rf<-mean(rf_predict==test_set$gender)
rf
```

```
## [1] 0.9733995
```

Random forest model provides accuracy of 97%. Lets look at importance of various predictors to the random forest model

```
plot(varImp(train_rf))
```

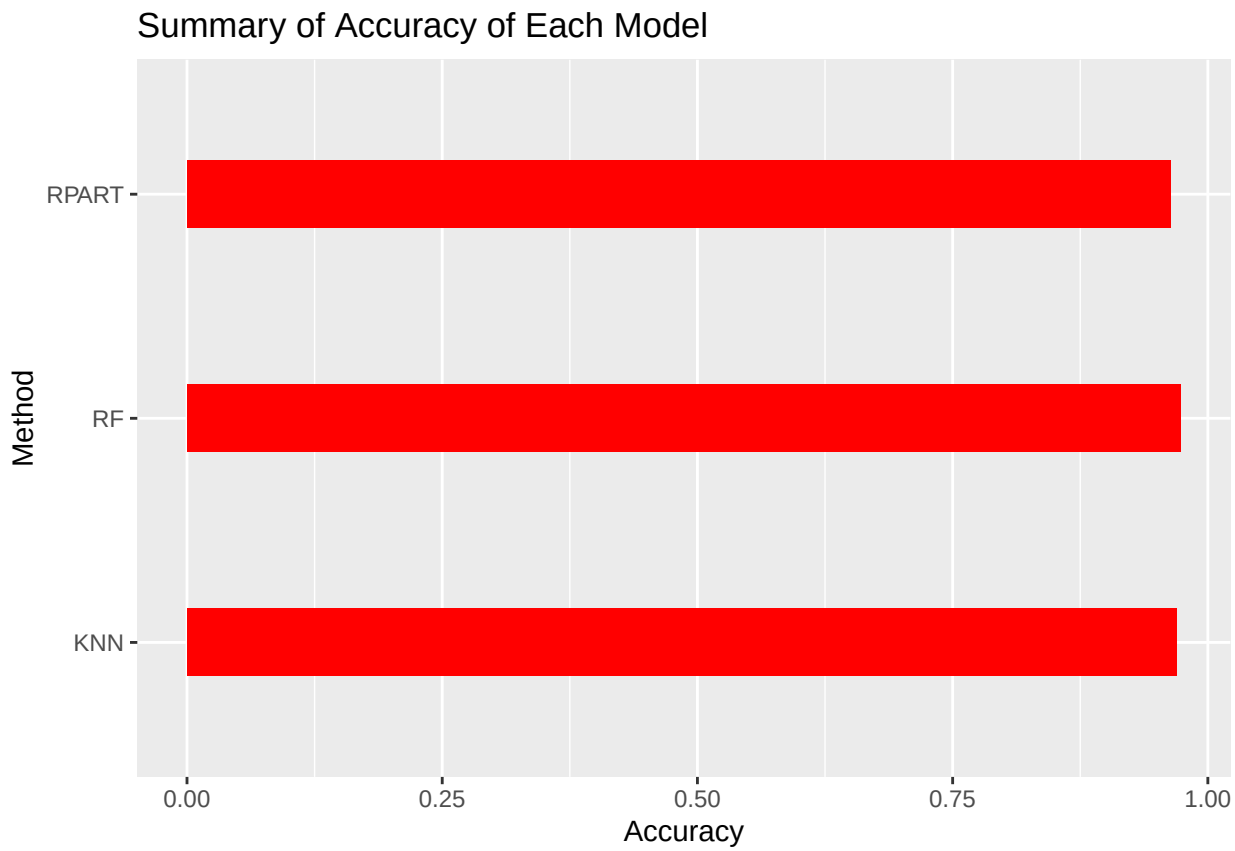


We can see five predictors contribute mainly to the random forest model

Summary

Lets compare the accuracy of different models tested.

```
data.frame(Method=c("KNN","RPART","RF"), Accuracy=c(knn,rpart,rf))%>%ggplot(aes(Method, Accuracy))+
  geom_bar(stat = "identity", fill="red",width = 0.3 )+coord_flip()+ggtitle("Summary of Accuracy of Eac
```



We can see that all three models provide similar results with random forest model slightly showing higher accuracy.

Conclusion

The three different models provided fairly accurate results. There were highly overlapping predictors which were removed prior to the prediction. Further analysis is required to be done to reduce number of predictors and improve the accuracy.

Ensemble methods needs to be tested as future work.

Reference

Irizzary,R.,Introduction to Data Science

https://www.kaggle.com/primaryobjects/voicegender