Sport Analytic Project

Atieh Khaleghi(atikh281)

4/18/2021

In this study we will answer the following questions:

- How can we predict a new player will be a winner or loser regarding the result of his/her serve results?
- How can we predict a new player will be a winner or loser regarding the result of his/her ranking?

Data Recourse: https://www.kaggle.com/taylorbrownlow/atpwta-tennis-data?select=KaggleMatches.csv

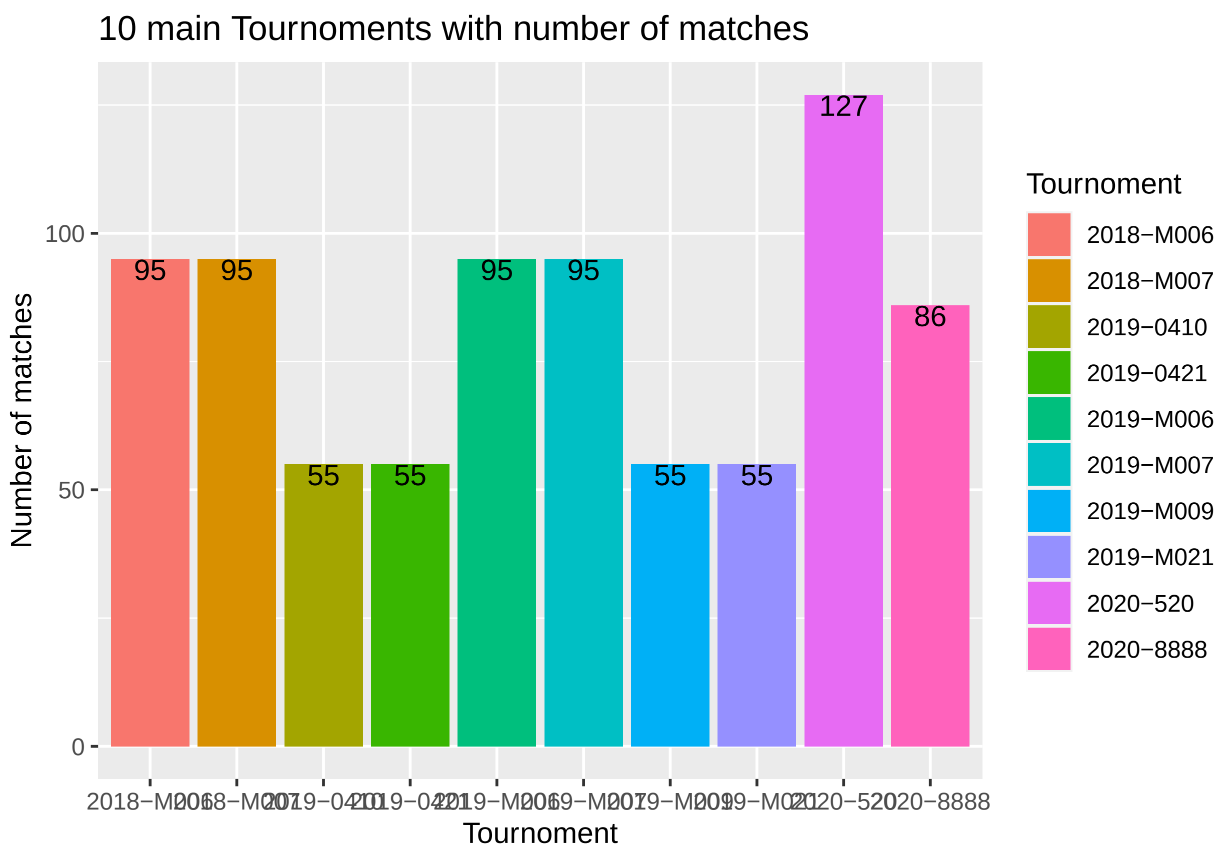
```
df <- read_excel("~./DataTennis.xlsx")
```

Visualizing

Different tournaments: We have 328 unique tournaments in our dataset.

The following is the table for the 20 tournaments which had the most number of matches.

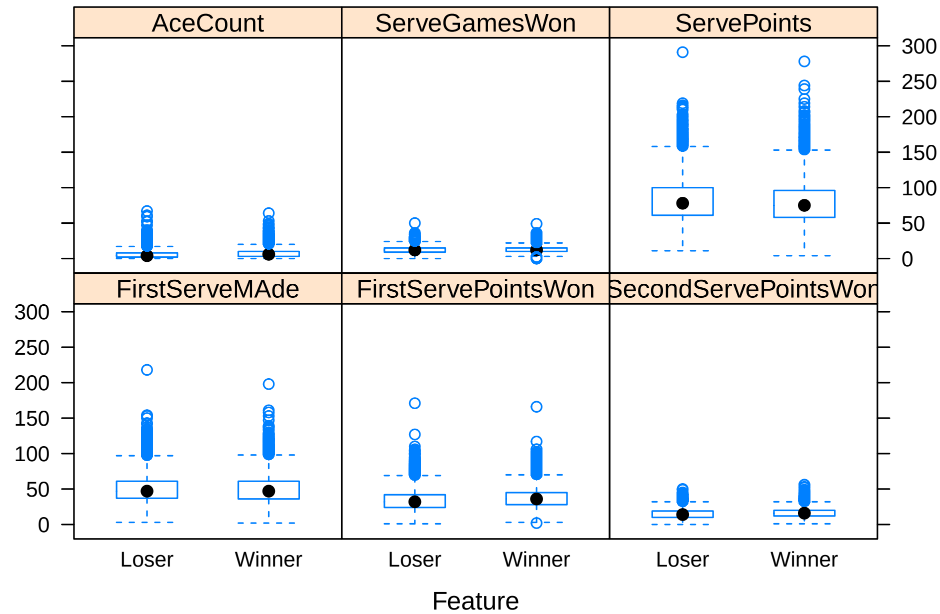
Tournoment	Number of matches
2019-580	127
2019-520	127
2019-540	127
2019-560	127
2018-580	127
2018-520	127
2018-540	127
2018-560	127
2020-580	127
2020-560	127
2020-520	127
2019-M006	95
2019-M007	95
2018-M006	95
2018-M007	95
2020-8888	86
2019-0410	55
2019-M021	55
2019-M009	55
2019-0421	55

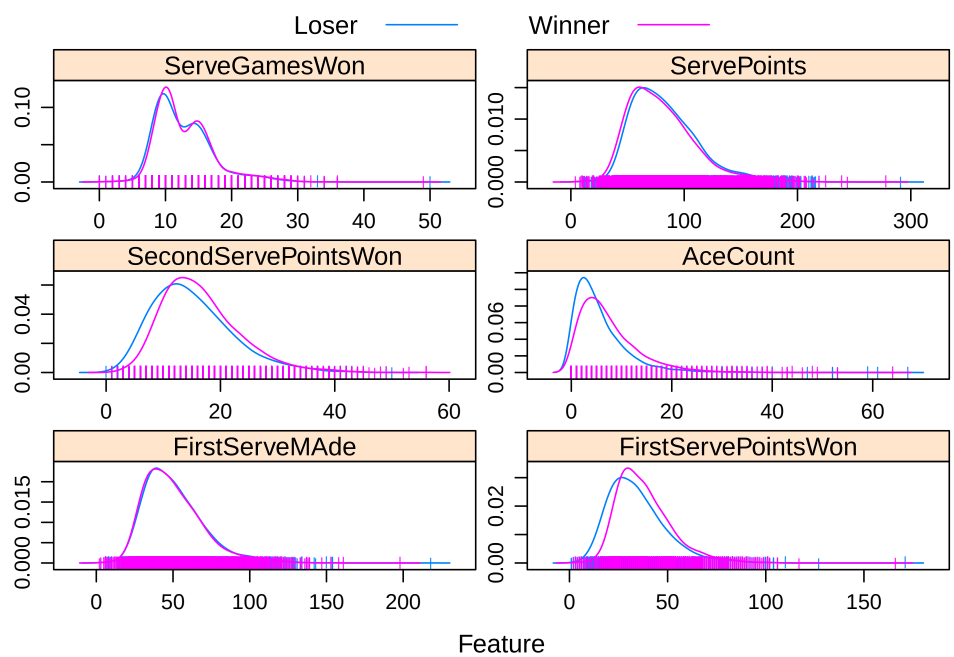


Preparing a new dataset for our analysing:

```
winner_df <- as.data.frame(cbind(df$w_1stIn,df$w_1stWon,
                                 df$w_2ndWon,df$w_ace,df$w_SvGms,
                                 df$w_svpt,"Type"=1))
loser_df <- as.data.frame(cbind(df$l_1stIn,df$l_1stWon,df$l_2ndWon,
                                df$l_ace,df$l_SvGms,df$l_svpt,"Type"=0))
new_df <- as.data.frame(rbind(winner_df,loser_df))
colnames(new_df)[1:6] <- c("FirstServeMAde","FirstServePointsWon",
                      "SecondServePointsWon","AceCount",
                      "ServeGamesWon","ServePoints")
new_df$Type <- as.factor(ifelse(new_df$Type==1,"Winner","Loser"))
```

Box Plot and density plot





Selecting a proper model

Since we do not know which algorithm is better for our data set we will evaluate 5 different algorithms:

- Linear Discriminant Analysis (LDA)
- k-Nearest Neighbors (KNN).
- Support Vector Machines (SVM) with a linear kernel.

- Random Forest (RF)
- Neural Networks(NNs)

We will 10-fold crossvalidation to estimate accuracy.

This will split our dataset into 10 parts, train in 9 and test on 1 and release for all combinations of train-test splits. We will also repeat the process 3 times for each algorithm with different splits of the data into 10 groups, in an effort to get a more accurate estimate.

Run algorithms using 10-fold cross validation

We are using the metric of "Accuracy" to evaluate models. This is a ratio of the number of correctly predicted instances in divided by the total number of instances in the dataset multiplied by 100 to give a percentage (e.g. 95% accurate). We will be using the metric variable when we run build and evaluate each model next.

Omitting Nas

```
set.seed(12356)
n <- dim(new_df)[1]
id <- sample(1:n)
N_df <- new_df[id,]
N_df <- as.data.frame(na.omit(N_df))
```

Split dataset into train and test

```
n <- dim(N_df)[1]
id <- sample(1:n,0.7*n)
trainSet <- N_df[id,]
testSet <- N_df[-id,]
```

```
#linear algorithms
set.seed(12356)
fit.lda <- caret::train(Type~., data=trainSet,method="lda",
                        metric="Accuracy",
                        trControl=trainControl(method="cv", number=10))

#KNN
set.seed(12356)
fit.knn <- train(Type~., data=trainSet, method="knn",
                 metric="Accuracy",
                 trControl=trainControl(method="cv", number=10))

# c) advanced algorithms
# SVM
set.seed(12356)
fit.svm <- train(Type~., data=trainSet, method="svmRadial",
                 metric="Accuracy",
                 trControl=trainControl(method="cv", number=10))
```

```r
# RF
set.seed(12356)
fit.rf <- train(Type~., data=trainSet, method="rf",
                metric="Accuracy",
                trControl=trainControl(method="cv", number=10))


# NNs
set.seed(12356)
fit.nns <- train(Type~., data=trainSet, method="nnet",
                 metric="Accuracy",
                 trControl=trainControl(method="cv", number=10))
```

We now have 5 models and accuracy estimations for each. We need to compare the models to each other and select the most accurate.

We can report on the accuracy of each model by first creating a list of the created models and using the summary function.

summarize accuracy of models

We can see the accuracy of each classifier and also other metrics like Kappa:

```r
results <- resamples(list(lda=fit.lda, NNs=fit.nns,
                          svm=fit.svm, KNN=fit.knn,
                          RF=fit.rf))
resTemp <- summary(results)
knitr::kable(resTemp$statistics$Accuracy)
```

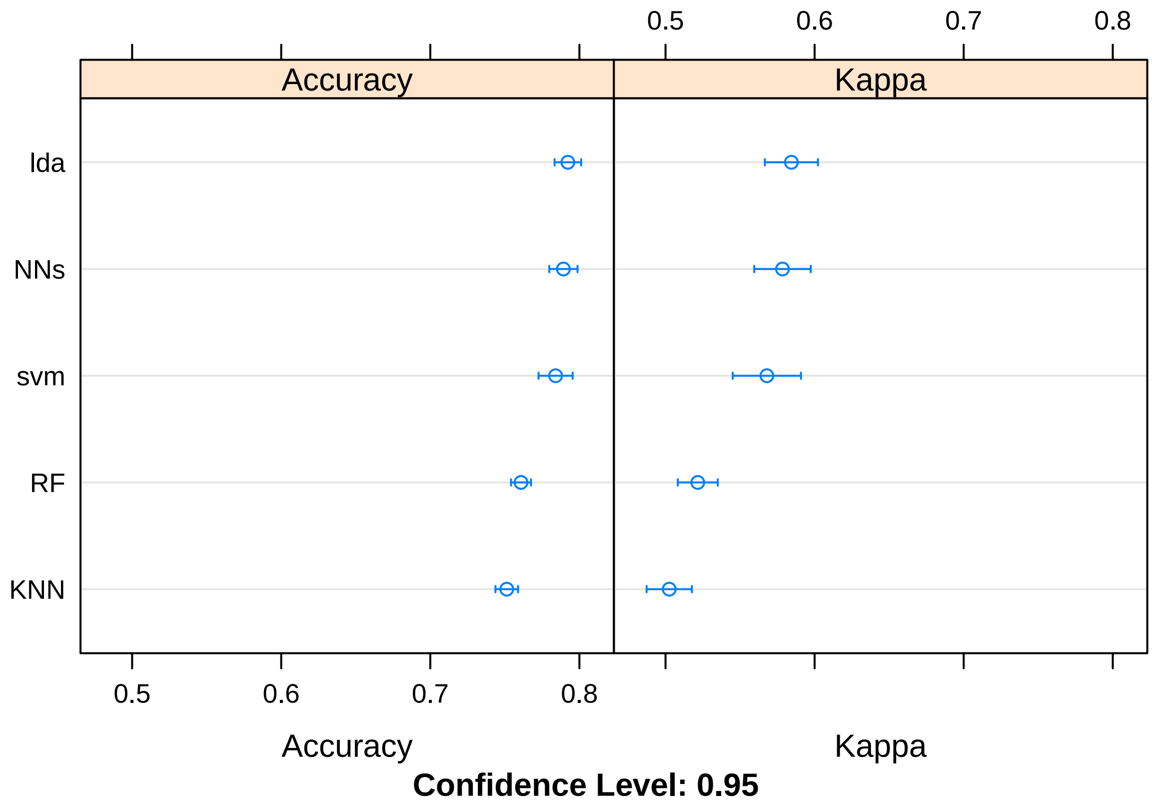
	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
lda	0.7751540	0.7848527	0.7926078	0.7922575	0.8006686	0.8141684	0
NNs	0.7628337	0.7871636	0.7920945	0.7892800	0.7976893	0.8028747	0
svm	0.7615622	0.7720739	0.7868502	0.7840396	0.7936345	0.8121150	0
KNN	0.7351129	0.7446043	0.7530801	0.7512824	0.7597536	0.7659138	0
RF	0.7433265	0.7535317	0.7627123	0.7608340	0.7689938	0.7710472	0

```r
knitr::kable(resTemp$statistics$Kappa)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
lda	0.5501525	0.5696323	0.5851141	0.5844386	0.6012745	0.6282270	0
NNs	0.5255673	0.5742231	0.5841417	0.5784524	0.5951849	0.6057495	0
svm	0.5229868	0.5439853	0.5736225	0.5679763	0.5872272	0.6241254	0
KNN	0.4700292	0.4891471	0.5061541	0.5025047	0.5194180	0.5317959	0
RF	0.4865144	0.5070531	0.5253252	0.5216194	0.5379740	0.5420056	0

We can also create a plot of the model evaluation results and compare the spread and the mean accuracy of each model. There is a population of accuracy measures for each algorithm because each algorithm was evaluated 10 times (10 fold cross validation).

```r
dotplot(results)
```



Make Predictions

The NNs and LDA were respictively the most accurate models. Now we want to get an idea of the accuracy of the NNs model and LDA model on our test set.

Confusion matrix for Neural Networks model:

```r
# estimate skill of NNs on the Test dataset
preds <- predict(fit.nns, testSet)
confusionMatrix(preds, testSet$Type)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Loser Winner
##     Loser   1647    433
##     Winner   459   1635
##
##                Accuracy : 0.7863
##                  95% CI : (0.7735, 0.7986)
##     No Information Rate : 0.5046
##     P-Value [Acc > NIR] : <2e-16
```

```
##
##                   Kappa : 0.5726
##
##  Mcnemar's Test P-Value : 0.4026
##
##             Sensitivity : 0.7821
##             Specificity : 0.7906
##          Pos Pred Value : 0.7918
##          Neg Pred Value : 0.7808
##              Prevalence : 0.5046
##          Detection Rate : 0.3946
##    Detection Prevalence : 0.4983
##       Balanced Accuracy : 0.7863
##
##        'Positive' Class : Loser
##
```

Confusion matrix for LDA model:

```
# estimate skill of LDA on the Test dataset
preds <- predict(fit.lda, testSet)
confusionMatrix(preds, testSet$Type)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Loser Winner
##     Loser   1608    402
##     Winner   498   1666
##
##                Accuracy : 0.7844
##                  95% CI : (0.7716, 0.7968)
##     No Information Rate : 0.5046
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5689
##
##  Mcnemar's Test P-Value : 0.001542
##
##             Sensitivity : 0.7635
##             Specificity : 0.8056
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.7699
##              Prevalence : 0.5046
##          Detection Rate : 0.3852
##    Detection Prevalence : 0.4816
##       Balanced Accuracy : 0.7846
##
##        'Positive' Class : Loser
##
```

We can see that the accuracy is 79%. Fairly we have an accurate and a reliably accurate model with NNs.

Appendix

```
library(readxl)
library(dplyr)
library(ggplot2)
library(caret)
knitr::opts_chunk$set(echo = TRUE)
df <- read_excel("~./DataTennis.xlsx")

uniqList <- unique(df$tourney_id)
res <- c()
for (i in 1:length(uniqList)) {
  res[i] <- length(which(df$tourney_id==uniqList[i]))
}

BiggestMatch <- order(res,decreasing = TRUE)
inds <- BiggestMatch[1:20]
new_tr <- as.data.frame(cbind("Tournoment"=uniqList[inds],"Number of matches"=res[inds]))
new_tr$`Number of matches` <- as.numeric(new_tr$`Number of matches`)
knitr::kable(new_tr)

ggplot(new_tr[11:20,], aes(Tournoment,`Number of matches`,fill=Tournoment))+
  geom_bar(stat = 'identity') +
  geom_text(aes(label = `Number of matches`), position = position_dodge(width = 1), vjust = 1)+
  labs(title = "10 main Tournoments with number of matches")

winner_df <- as.data.frame(cbind(df$w_1stIn,df$w_1stWon,
                                 df$w_2ndWon,df$w_ace,df$w_SvGms,
                                 df$w_svpt,"Type"=1))
loser_df <- as.data.frame(cbind(df$l_1stIn,df$l_1stWon,df$l_2ndWon,
                                df$l_ace,df$l_SvGms,df$l_svpt,"Type"=0))
new_df <- as.data.frame(rbind(winner_df,loser_df))
colnames(new_df)[1:6] <- c("FirstServeMAde","FirstServePointsWon",
                      "SecondServePointsWon","AceCount",
                      "ServeGamesWon","ServePoints")
new_df$Type <- as.factor(ifelse(new_df$Type==1,"Winner","Loser"))

x <- new_df[,1:6]
y <- new_df[,7]
#par(mfrow=c(2,3))
#  for(i in 1:6) {
#  boxplot(x[,i], main=names(new_df)[i])
#  }

featurePlot(x=x, y=y, plot="box",## Add a key at the top
            auto.key = list(columns = 3))
featurePlot(x=x, y=y,plot = "density",
            ## Pass in options to xyplot() to
            ## make it prettier
            scales = list(x = list(relation="free"),
                          y = list(relation="free")),
            adjust = 1.5,
            pch = "|",
```

```r
           layout = c(2, 3),
           auto.key = list(columns = 2))

set.seed(12356)
n <- dim(new_df)[1]
id <- sample(1:n)
N_df <- new_df[id,]
N_df <- as.data.frame(na.omit(N_df))
n <- dim(N_df)[1]
id <- sample(1:n,0.7*n)
trainSet <- N_df[id,]
testSet <- N_df[-id,]
#linear algorithms
set.seed(12356)
fit.lda <- caret::train(Type~., data=trainSet,method="lda",
                        metric="Accuracy",
                        trControl=trainControl(method="cv", number=10))

#KNN
set.seed(12356)
fit.knn <- train(Type~., data=trainSet, method="knn",
                 metric="Accuracy",
                 trControl=trainControl(method="cv", number=10))

# c) advanced algorithms
# SVM
set.seed(12356)
fit.svm <- train(Type~., data=trainSet, method="svmRadial",
                 metric="Accuracy",
                 trControl=trainControl(method="cv", number=10))

# RF
set.seed(12356)
fit.rf <- train(Type~., data=trainSet, method="rf",
                metric="Accuracy",
                trControl=trainControl(method="cv", number=10))


# NNs
set.seed(12356)
fit.nns <- train(Type~., data=trainSet, method="nnet",
                 metric="Accuracy",
                 trControl=trainControl(method="cv", number=10))

results <- resamples(list(lda=fit.lda, NNs=fit.nns,
                          svm=fit.svm, KNN=fit.knn,
                          RF=fit.rf))
resTemp <- summary(results)
knitr::kable(resTemp$statistics$Accuracy)
knitr::kable(resTemp$statistics$Kappa)
dotplot(results)
# estimate skill of NNs on the Test dataset
preds <- predict(fit.nns, testSet)
```

```r
confusionMatrix(preds, testSet$Type)
# estimate skill of LDA on the Test dataset
preds <- predict(fit.lda, testSet)
confusionMatrix(preds, testSet$Type)
```