Sport Analytic Project

Atieh Khaleghi(atikh281)

4/18/2021

Introduction

Distinguishing features to predict the status of a tennis player is the aim of this study. Among all attributes in the data set[1], the following set of features related to Serve of the player have been considered.

- Winner's serve points
- Winner first serves made
- Winner first serve points won
- Winner second serve points won
- Winner service games won
- Winner's ace count
- Loser's serve points
- Loser first serves made
- Loser first serve points won
- Loser second serve points won
- Loser service games won
- Loser's ace count

Exploring the data, training various Machine learning algorithms and evaluate them to choose the most accurate ones to predict the prospective players will be processed in this project.

Thus the following question will be answered:

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

Background

In many sport matches, interested in predicting the outcomes and watching the games to verify their predictions has been increased among spectators. Traditional approaches include subjective prediction, objective prediction, and simple statistical methods. However, these approaches may not be too reliable in many situations.[3]

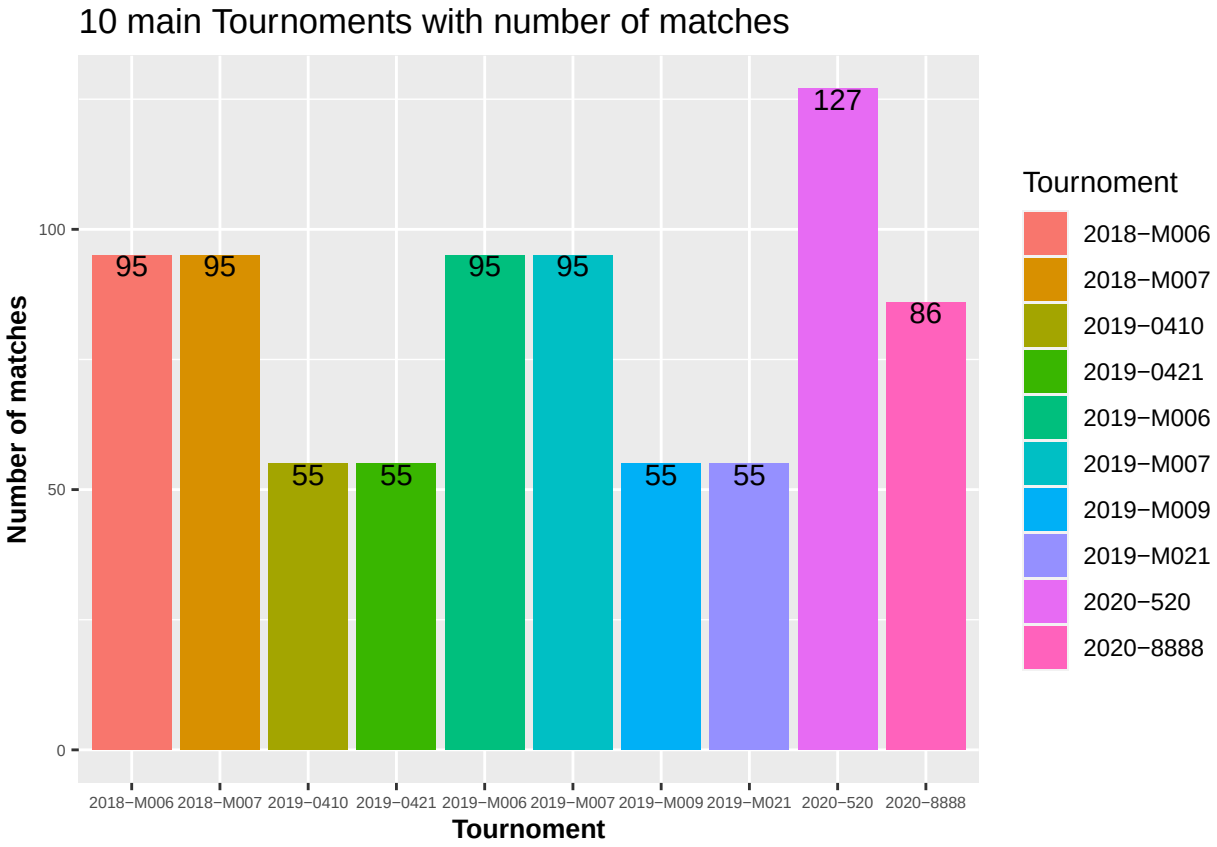
Machine learning (ML) is one of the intelligent techniques that have powerful tools for classification and prediction. One of the expanding areas necessitating good predictive accuracy is sport prediction, due to the large monetary amounts involved in betting. Moreover, coaches, club managers and owners‘ demand has been increased for classification models so that they can distinguish and design better strategies needed to win matches.[2]

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

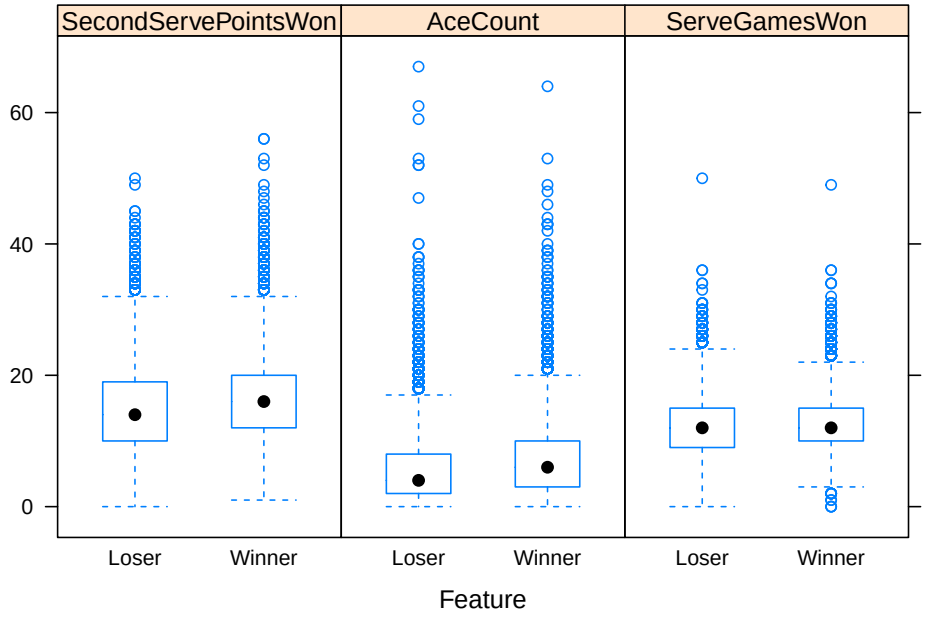


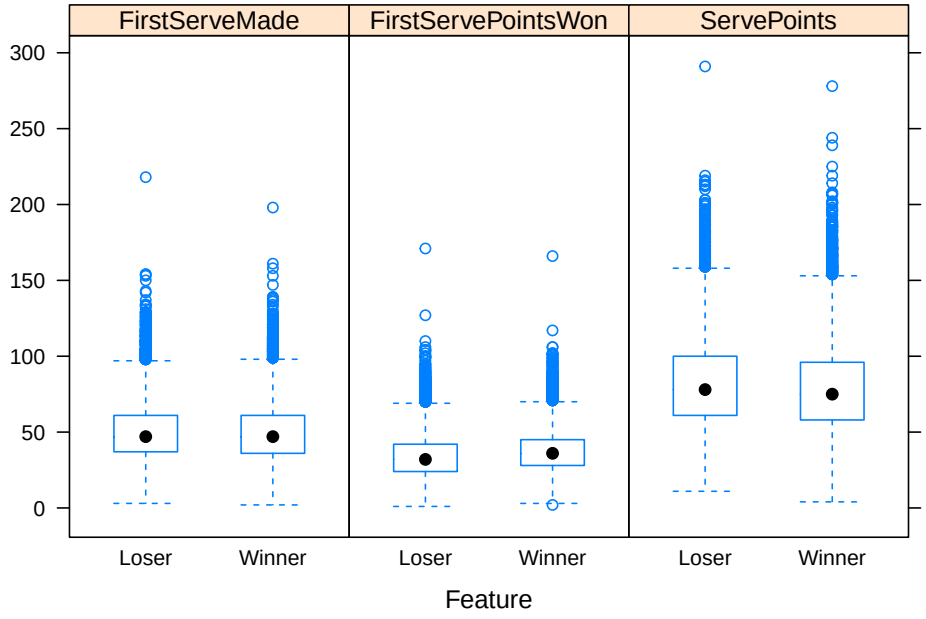
Preparing a new data set for our analyzing:

FirstServeMade	FirstServePointsWon	SecondServePointsWon
Min. : 2.00	Min. : 1.00	Min. : 0.00
1st Qu.: 36.00	1st Qu.: 26.00	1st Qu.:11.00
Median : 47.00	Median : 34.00	Median :15.00
Mean : 50.53	Mean : 36.25	Mean :15.84
3rd Qu.: 61.00	3rd Qu.: 44.00	3rd Qu.:20.00
Max. :218.00	Max. :171.00	Max. :56.00
NA's :352	NA's :352	NA's :352

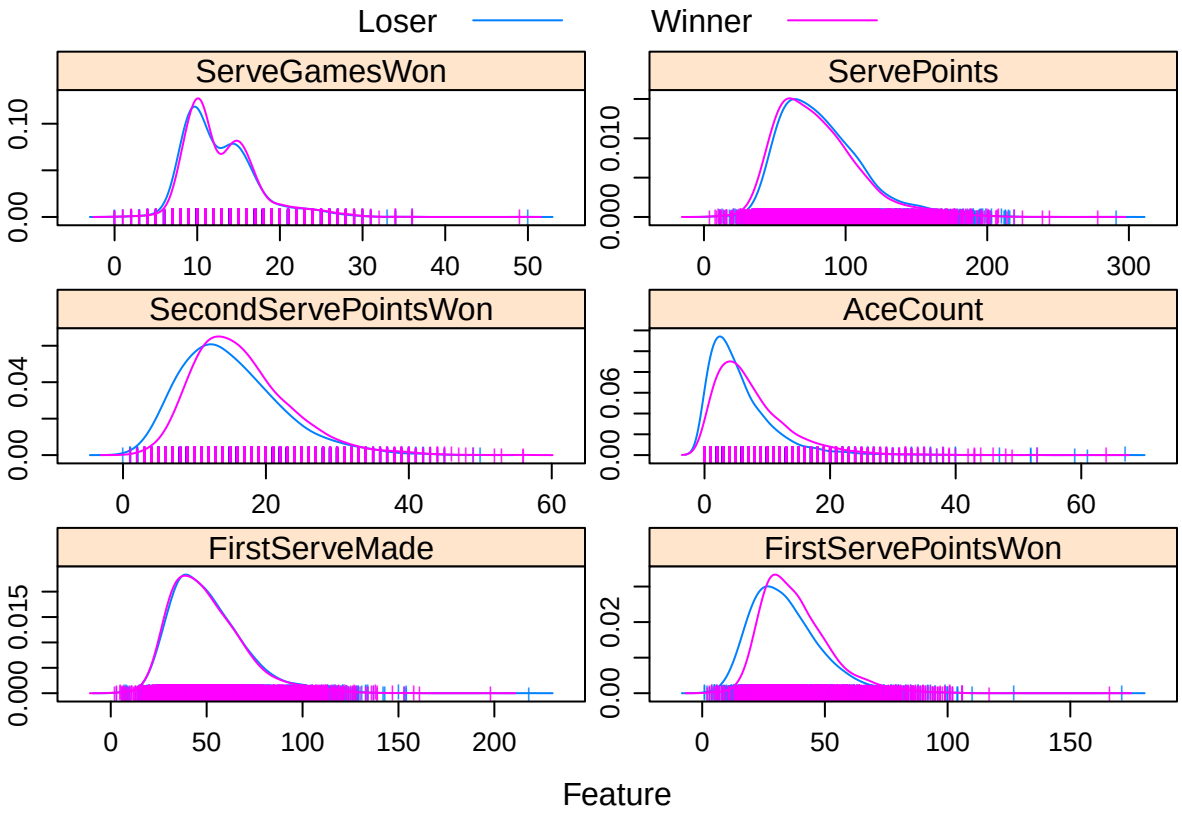
AceCount	ServeGamesWon	ServePoints
Min. : 0.00	Min. : 0.00	Min. : 4.00
1st Qu.: 3.00	1st Qu.:10.00	1st Qu.: 60.00
Median : 5.00	Median :12.00	Median : 77.00
Mean : 6.56	Mean :12.79	Mean : 81.78
3rd Qu.: 9.00	3rd Qu.:15.00	3rd Qu.: 98.00
Max. :67.00	Max. :50.00	Max. :291.00
NA's :352	NA's :352	NA's :352

Box Plot and density plot





Using Boxplot we explore the outliers defined as data points that is located outside the whiskers of the box plot. In our dataset all the predictors have significant outliers.



The density plots show that we have a bimodal distribution for ServeGamesWon feature and positive-skew distribution for ServePoints, AceCount and even FirstServeMade features.

Gaussian-like curves can been seen for all the predictors.

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.

Preparing the new data set for MAchine learning algorithms

- Omitting Nas
- Split dataset into train and test

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.

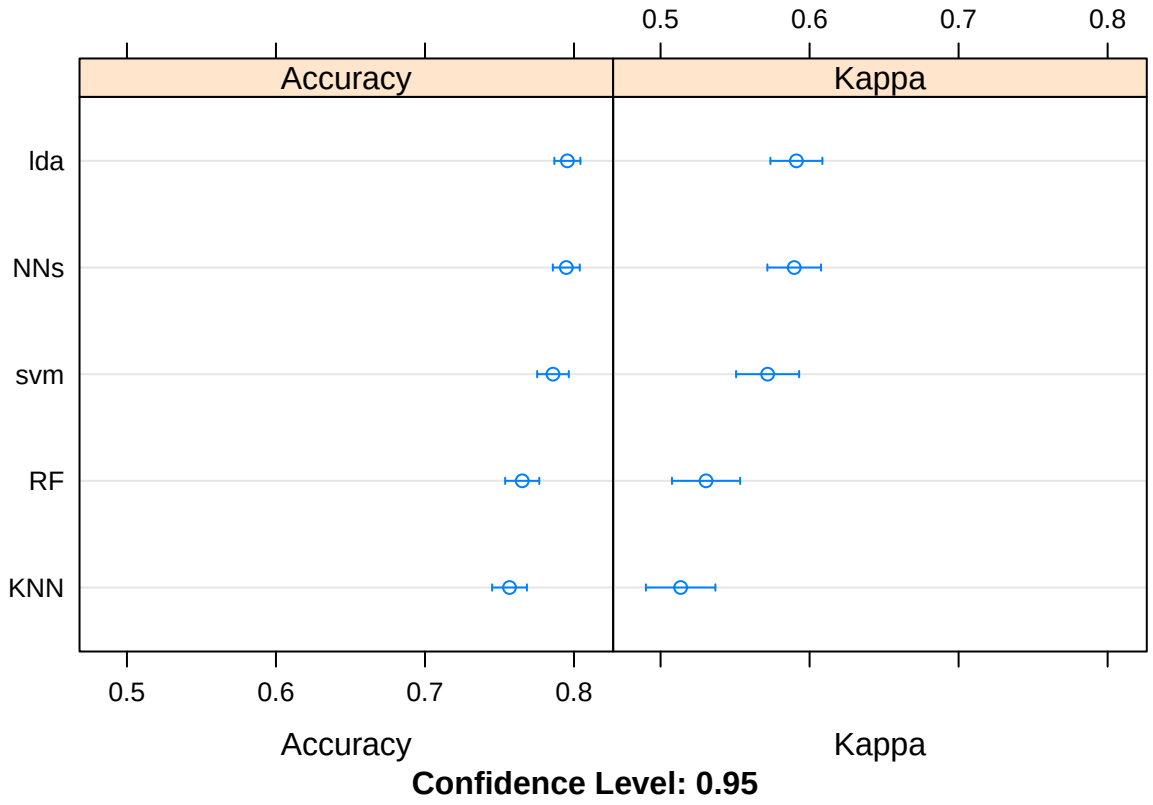
summarize accuracy of models

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

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
lda	0.7782341	0.7831982	0.8002072	0.7955425	0.8056982	0.8080082	0
NNs	0.7710472	0.7862822	0.7946612	0.7948243	0.8061611	0.8090349	0
svm	0.7546201	0.7819156	0.7859343	0.7858910	0.7931211	0.8110883	0
KNN	0.7340862	0.7448665	0.7587269	0.7567270	0.7615503	0.7821172	0
RF	0.7392197	0.7582136	0.7633470	0.7652487	0.7789471	0.7843943	0

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
lda	0.5566402	0.5664278	0.6004985	0.5911488	0.6113857	0.6160391	0
NNs	0.5422836	0.5726141	0.5893502	0.5896991	0.6122878	0.6181213	0
svm	0.5094181	0.5638293	0.5719458	0.5718341	0.5862824	0.6221638	0
KNN	0.4682666	0.4897049	0.5175232	0.5134752	0.5231176	0.5642228	0
RF	0.4785274	0.5164007	0.5267444	0.5305124	0.5579464	0.5687594	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).



Predicting

The NNs and LDA were respectively 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:

Confusion Matrix and Statistics

```
      Reference
```

Prediction Loser Winner Loser 1559 415 Winner 507 1693

```
           Accuracy : 0.7791
             95% CI : (0.7662, 0.7916)
No Information Rate : 0.505
P-Value [Acc > NIR] : < 2.2e-16

              Kappa : 0.558
```

Mcnemar's Test P-Value : 0.002727

```
        Sensitivity : 0.7546
        Specificity : 0.8031
     Pos Pred Value : 0.7898
     Neg Pred Value : 0.7695
         Prevalence : 0.4950
     Detection Rate : 0.3735
```

Detection Prevalence : 0.4729
Balanced Accuracy : 0.7789

```
   'Positive' Class : Loser
```

Confusion matrix for LDA model:

Confusion Matrix and Statistics

```
      Reference
```

Prediction Loser Winner Loser 1552 410 Winner 514 1698

```
           Accuracy : 0.7786
             95% CI : (0.7657, 0.7911)
No Information Rate : 0.505
P-Value [Acc > NIR] : < 2.2e-16

              Kappa : 0.557
```

Mcnemar's Test P-Value : 0.0007029

```
        Sensitivity : 0.7512
        Specificity : 0.8055
     Pos Pred Value : 0.7910
     Neg Pred Value : 0.7676
         Prevalence : 0.4950
     Detection Rate : 0.3718
```

Detection Prevalence : 0.4701
Balanced Accuracy : 0.7784

```
   'Positive' Class : Loser
```

We can see that the accuracies for both models are approximately 78%.

Refrences

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

[2] Bunker, Rory & Thabtah, Fadi. (2017). A Machine Learning Framework for Sport Result Prediction. Applied Computing and Informatics. 15. 10.1016/j.aci.2017.09.005.

[3] Leung, Carson & Joseph, Kyle. (2014). Sports Data Mining: Predicting Results for the College Football Games. Procedia Computer Science. 35. 10.1016/j.procs.2014.08.153.

Appendix

```
library(readxl)
library(dplyr)
library(ggplot2)
library(caret)
library(knitr)
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")+
  theme(axis.text=element_text(size=6),
        axis.title=element_text(size=10,face="bold"))

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"))
set.seed(123456)
new_df <- as.data.frame(new_df[sample(nrow(new_df)),])
kable(summary(new_df[,1:3]))
kable(summary(new_df[,4:6]))

x1 <- new_df[,3:5]
x2 <- new_df[,c(1,2,6)]
y <- new_df[,7]
#par(mfrow=c(2,3))
#  for(i in 1:6) {
#  boxplot(x[,i], main=names(new_df)[i])
#  }
```

```
featurePlot(x=x1, y=y, plot="box",## Add a key at the top
            auto.key = list(columns = 3))
featurePlot(x=x2, y=y, plot="box",## Add a key at the top
            auto.key = list(columns = 3))

featurePlot(x=new_df[,1:6], 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 = "|",
            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)
confusionMatrix(preds, testSet$Type)
# estimate skill of LDA on the Test dataset
preds <- predict(fit.lda, testSet)
confusionMatrix(preds, testSet$Type)
```