Poročilo

Najprej nastavimo delovno mapo okolja, ter naložimo podatke iz datoteke regular.txt v tabelo »teams«. Ko pregledamo atribute vidimo, da se hranijo iste vrste podatkov za obe ekipi (skupne točke, uspešni meti, št ukradenih žog, …), torej osnovno statistiko tekme, ter datum in pa sezono. Za prvi model si dajmo izbrati vse atribute, za sodelovanje v zgradbi modela, atribut SEASON pa uporabimo za ločitev na učno in testno množico, zato tega atributa ne uporabimo v modelu (odstranimo ga iz učne in testne množice). Za določitev zmagovalca pa dejmo ustvariti nov atribut WIN, ki bo glede na število skupnih točk med ekipama določil zmagovalko torej bi lahko imela vrednost »HOME« ali pa »AWAY«.

Načrt je pa takšen: najprej bomo za model uporabili vse atribute in izvedli več algoritmov, ter izračunali in primerjali njihovo klasifikacijsko točnost. Za drugi model, bomo uporabili nove atribute in odstranili tiste, ki mislimo, da ne vplivajo toliko na končen rezultat (v dobrem smislu) in kombinirali algoritme, ter izračunali in primerjali njihovo klasifikacijsko točnost.
Pri zadnjem modelu bomo pa uporabili glasovanje za ocenjevanje atributov ter primerjali model, ki nam ga glasovanje priporoča, ter poskusili izbrati samo nekatere pomembnejše atribute in nato primerjali rezultate.

Prvi model

Prvega modela se torej lotimo kar na enostaven način, uporabimo vse atribute, razen SEASON, saj smo ga uporabili, za delitev množice.

Večinski klasifikator

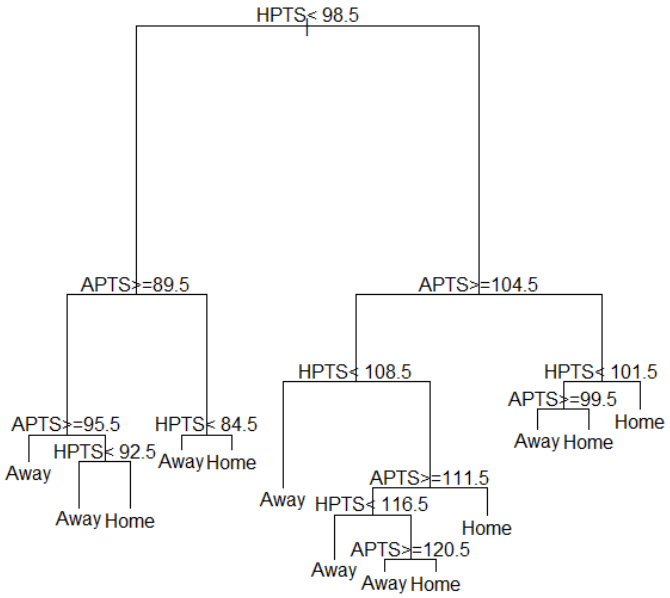
Testno množico smo napolnili s podatki o tekmah v zadnji sezoni, učno pa z vsemi prej. Sedaj izračunamo večinski razred, ter točnost večinskega klasifikatorja.

```
> setwd("E:/UI/Seminarska")
> teams <- read.table(file = 'regular.txt', header = T, sep = ',')
> WIN <- vector()
> h <- teams$HPTS
> a <- teams$APTS
> WIN[h < a] <- 'Away'
> WIN[h > a] <- 'Home'
> teams$WIN <- as.factor(WIN)
> learn <- teams[teams$SEASON != '2016-17',]
> test <- teams[teams$SEASON == '2016-17',]
> learn$SEASON <- NULL
> test$SEASON <- NULL
> nrow(learn)
[1] 2460
> nrow(test)
[1] 1230
>
> majority.class <- names(which.max(table(learn$WIN)))
> sum(test$WIN == majority.class) / length(test$WIN)
[1] 0.5837398
>
```

Dobili smo klasifikacijsko točnost 58%.

Odločitveno drevo

Uporabili bomo knjižnico rpart, zgradili model, izrisali drevo ter nato izračunali klasifikacijsko točnost modela. Zato potrebujemo prave vrednosti testnih primerov (hranimo jih v »observed«) ter napovedane vrednosti modela (hranimo jih v »predicted«), nato zgradimo tabelo napačnih klasifikacij.




```
> t <- table(observed, predicted)
> t
          predicted
observed Away Home
    Away  506    6
    Home   37  681
```

```
> CA(observed, predicted)
[1] 0.9650407
```

Vidimo lahko, kako nam je zgradilo drevo, tabelo pravilnih in napačnih predikcij ter klasifikacijsko točnost 96%.

Naivni Bayes

Uporabili bomo knjižnico e1071 ter uporabili funkcijo naiveBayes, ocenili klasifikacijsko točnost ter napake s pomočjo prečnega preverjanja.

```
> library(e1071)
>
> nb <- naiveBayes(WIN ~ ., data = learn)
> predicted <- predict(nb, test, type="class")
> CA(observed, predicted)
[1] 0.9056911
> predMat <- predict(nb, test, type = "raw")
> errorest(WIN~., data=learn, model = naiveBayes, predict = mypredict.generic
+ )

Call:
errorest.data.frame(formula = WIN ~ ., data = learn, model = naiveBayes,
    predict = mypredict.generic)

         10-fold cross-validation estimator of misclassification error

Misclassification error:  0.0858
```

Vidimo lahko, da je klasifikacijska točnost 91% in temu primerno izračun napake s pomočjo prečnega preverjanja 9 %.

Random forest

```
> library(randomForest)
randomForest 4.6-14
Type rfNews() to see new features/changes/bug fixes.
>
> rf <- randomForest(WIN ~ ., data = learn)
> predicted <- predict(rf, test, type="class")
> CA(observed, predicted)
[1] 0.9252033
>
> predMat <- predict(rf, test, type = "prob")
>
> mypredict.rf <- function(object, newdata){predict(object, newdata, type = "class")}
> errorest(WIN~., data=learn, model = randomForest, predict = mypredict.generic)

Call:
errorest.data.frame(formula = WIN ~ ., data = learn, model = randomForest,
    predict = mypredict.generic)

         10-fold cross-validation estimator of misclassification error

Misclassification error:  0.0577
```

Vidimo lahko, da je klasifikacijska točnost 93% in izračun napake s pomočjo prečnega preverjanja 6 %.

Drugi model

Pri drugem modelu bomo kombinirali 3 algoritme in sicer odločitveno drevo, naivni bayes ter k-najbližji sosed, pri tem pa bomo uporabili CORElearn knjižnico. Torej najprej razdelimo množico (70% učna, 30% testna), nato izvedemo glasovanje, uteženo glasovanje, bagging ter boosting. Med atribute dodamo še razmerje uspešnih/zgrešenih metov ter razmerje ukradenih/izgubljenih žog.

Glasovanje

Najprej ustvarimo 3 modele (Naivni Bayes, K-najbližji sosed in odločitveno drevo), ocenimo njihovo klasifikacijsko točnost ter nato zaženemo glasovanje in spet ocenimo klasifikacijsko točnost. Vidimo lahko, da se je najboljše odrezalo odločitveno drevo.

```
> predNB <- predict(modelNB, test, type="class")
> caNB <- CA(test$WIN, predNB)
> caNB
[1] 0.8988257
> predKNN <- predict(modelKNN, test, type="class")
> caKNN <- CA(test$WIN, predKNN)
> caKNN
[1] 0.8139115
> predDT <- predict(modelDT, test, type = "class")
> caDT <- CA(test$WIN, predDT)
> caDT
[1] 0.9765131
>
```

```
> predicted <- voting(pred)
> CA(test$WIN, predicted)
[1] 0.9421861
> 
```

Uteženo glasovanje

Spet ustvarimo tri modele, ter seštejemo napovedane vrednosti s strani le teh, ter izberemo razred z največjo verjetnostjo.

```
> predicted <- levels(learn$WIN)[apply(pred.prob, 1, which.max)]
> 
> CA(test$WIN, predicted)
[1] 0.9719964
> 
```

Bagging in Boosting

Pomagamo si s pomočjo knjižice ipred ter izvedemo funkcijo bagging in boosting ter ocenimo klasifikacijsko točnost obeh.

```
> library(ipred)
> 
> bag <- bagging(WIN ~ ., learn, nbagg=15)
> bag.pred <- predict(bag, test, type="class")
> CA(test$WIN, bag.pred)
[1] 0.9810298
> 
```

```
> predicted <- predictions$class
> CA(test$WIN, predicted)
[1] 0.9855465
> 
```

Vidimo lahko, da sta se odrezali zelo podobno.

Tretji model

Za zadnji model bomo opravili ocenjevanje atributov, pogledali kateri se najbolj ponavljajo na začetku, ter sestavili model z »najpomembnejšimi« atributi, s pomočjo wrapperja.

```
> sort(attrEval(WIN ~ ., teams, "InfGain"), decreasing = TRUE)
        APTS         HPTS         HDRB         HOME         HAST
1.261967e-01 1.167319e-01 7.566286e-02 7.339082e-02 6.483613e-02 5.362
        A3PM         H3PM         A2PM         H2PM         HBLK
3.034515e-02 2.894716e-02 2.374093e-02 2.266793e-02 1.705374e-02 1.423
        ABLK         AFTA         ATOV         HFTM         HTOV
1.182967e-02 9.230541e-03 9.014455e-03 7.810650e-03 7.769743e-03 7.549
         APF         AORB         DATE         H2PA         A3PA
4.021300e-03 2.947133e-03 1.629458e-03 1.185965e-03 1.164478e-03 1.061
> sort(attrEval(WIN ~ ., teams, "Gini"), decreasing = TRUE)
        APTS         HPTS         HDRB         HOME         HAST
8.244901e-02 7.736202e-02 5.047244e-02 4.684587e-02 4.256107e-02 3.552
        A3PM         H3PM         A2PM         H2PM         HBLK
2.056362e-02 1.915065e-02 1.593645e-02 1.499224e-02 1.134383e-02 9.477
        ABLK         AFTA         ATOV         HTOV         HFTM
7.965021e-03 6.306853e-03 6.146485e-03 5.296520e-03 5.256700e-03 5.041
        HFTA         AORB         DATE         H2PA         A3PA
2.376269e-03 2.001426e-03 1.107482e-03 8.056866e-04 7.729053e-04 7.180
>
> sort(attrEval(WIN ~ ., teams, "GainRatio"), decreasing = TRUE)
        HPTS         APTS         AFTM          HPF         A2PM
1.657826e-01 1.301724e-01 1.247196e-01 1.203492e-01 1.203492e-01 1.203
        AFTA          APF         HAST         ADRB         ATOV
1.076810e-01 1.025325e-01 8.335568e-02 8.179212e-02 7.838520e-02 7.719
        H3PA         HFTM         H2PA         HFTA         HTOV
7.116814e-02 7.116814e-02 6.347659e-02 6.347659e-02 5.928419e-02 4.956
        A3PA         HBLK         ASTL         HOME         ABLK
2.480539e-02 1.774564e-02 1.551908e-02 1.495669e-02 1.394680e-02 1.092
>
> sort(attrEval(WIN ~ ., teams, "ReliefFequalK"), decreasing = TRUE)
         APTS          HPTS          HDRB          A3PM          A2PM
 0.2378274058  0.2352478878  0.0717986022  0.0651219512  0.0594811290
         ADRB          AAST          AFTM          HOME          AWAY
 0.0493261992  0.0433196463  0.0171738604  0.0167750678  0.0133875339
         H2PA          HFTM          HTOV          A3PA          HSTL
 0.0069706610  0.0043270099  0.0028883536  0.0024077838  0.0009291521
         HFTA          APF          HBLK          HORB          ASTL
-0.0065016330 -0.0066023035 -0.0069527251 -0.0110006254 -0.0125835592
> sort(attrEval(WIN ~ ., teams, "MDL"), decreasing = TRUE)
         APTS          HPTS          HDRB          HAST          ADRB
 1.248685e-01  1.154221e-01  7.437453e-02  6.354133e-02  5.170227e-02
         AWAY          H3PM          A2PM          H2PM          HBLK
 2.885065e-02  2.767961e-02  2.247647e-02  2.140892e-02  1.579650e-02
         ABLK          AFTA          ATOV          HTOV          HFTM
 1.057239e-02  8.089120e-03  7.840865e-03  6.598802e-03  6.554256e-03
          APF          AORB          DATE          H3PA          HORB
 3.085691e-03  1.761683e-03  4.966548e-04  3.576088e-04  2.481698e-04
>
```

Vidimo lahko, da so ponavljajoče na »vrhu« atributi APTS, HPTS, HDRB, AFTM, HPF, AORB, … Iz njih lahko sedaj poskusimo sestaviti model in izračunati klasifikacijsko točnost.

```
> res <- errorest(WIN ~ APTS + HPTS + HDRB + AFTM + HPF + AORB, data = test,
> 1-res$error
[1] 0.9756098
>
```

Seveda, pa lahko s pomočjo wrapperja dobimo predlog najboljšega modela.

```
best model: estimated accuracy =  0.9846432 , selected feature subset =  WIN ~ HPTS + APTS + HTOV + H2PA + HBLK + AORB
```

Matic Isovski, 2.12.2018, Koper