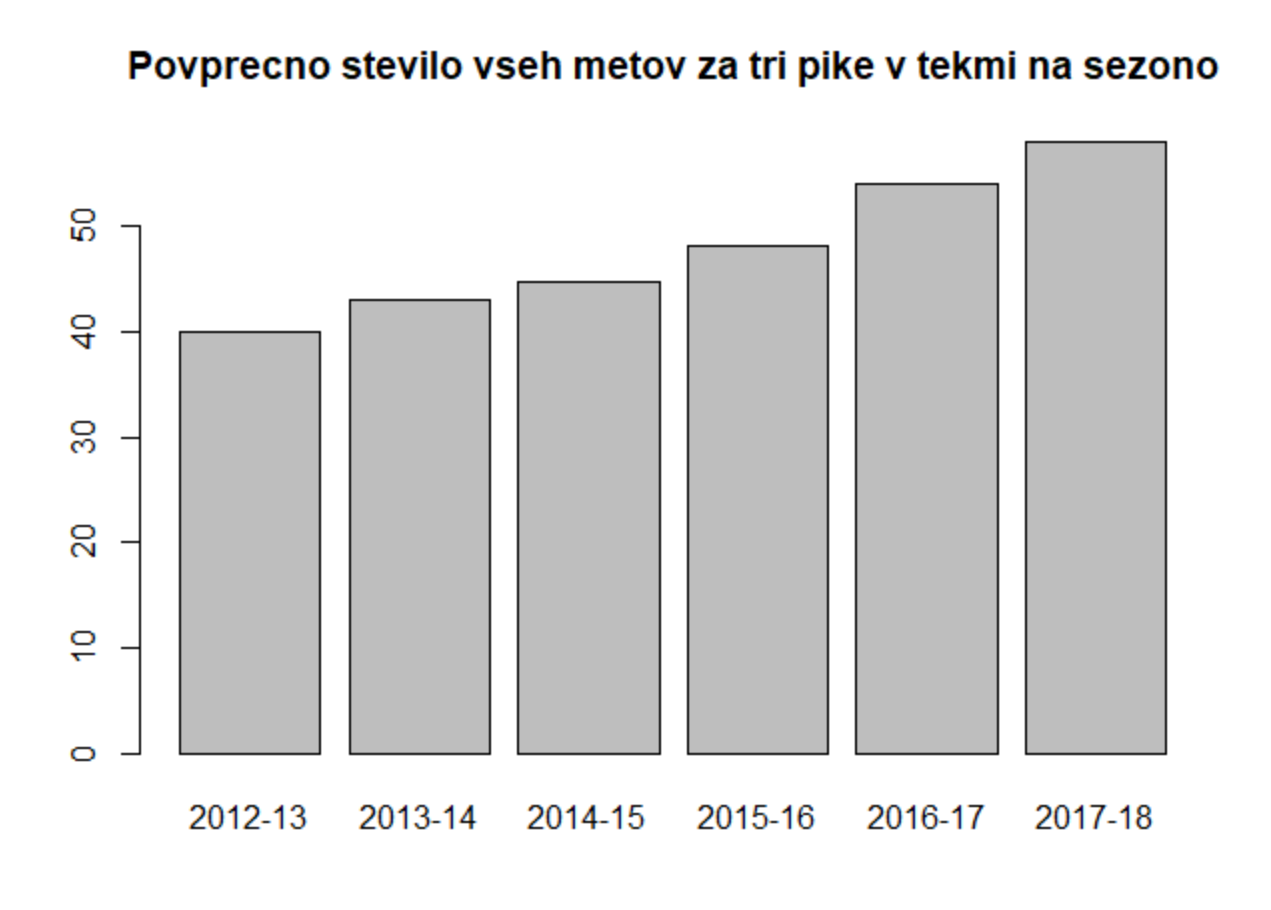
# Seminarska naloga UI 1

## Vizualizacija



SEASONS = unique(md$gmSeason);

averageThrees <- vector()

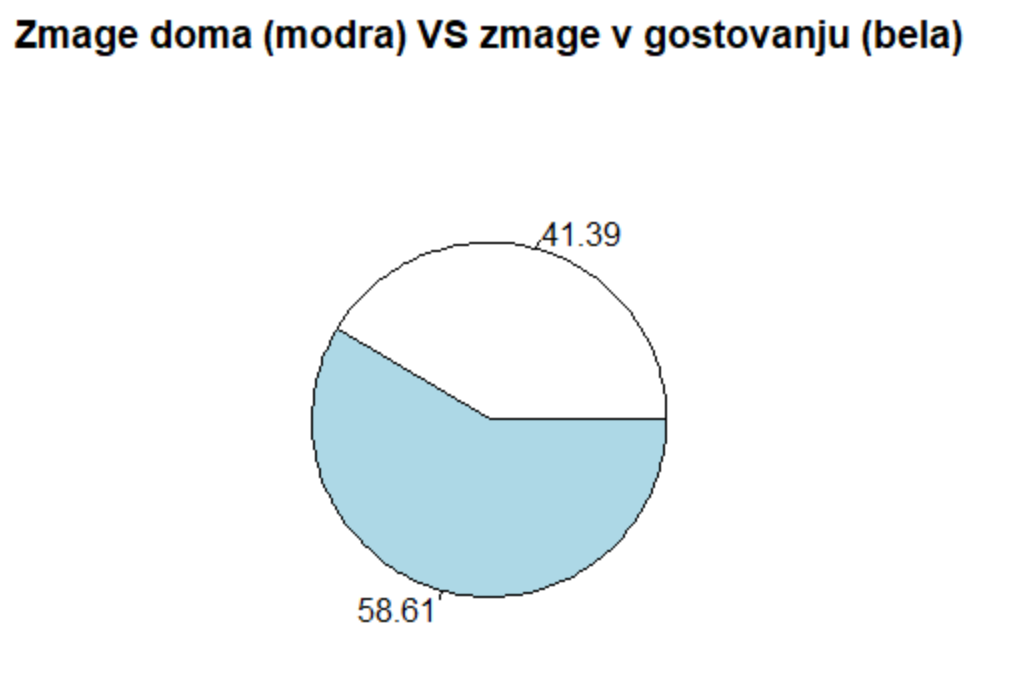
for(season in SEASONS) {

    seasonGames <- md$gmSeason == season;

    averageThrees <- c(averageThrees, mean(md$home3PA[seasonGames] + md$away3PA[seasonGames]));

}

barplot(averageThrees, names=SEASONS, main="Povprecno stevilo vseh metov za tri pike v tekmi na sezono");

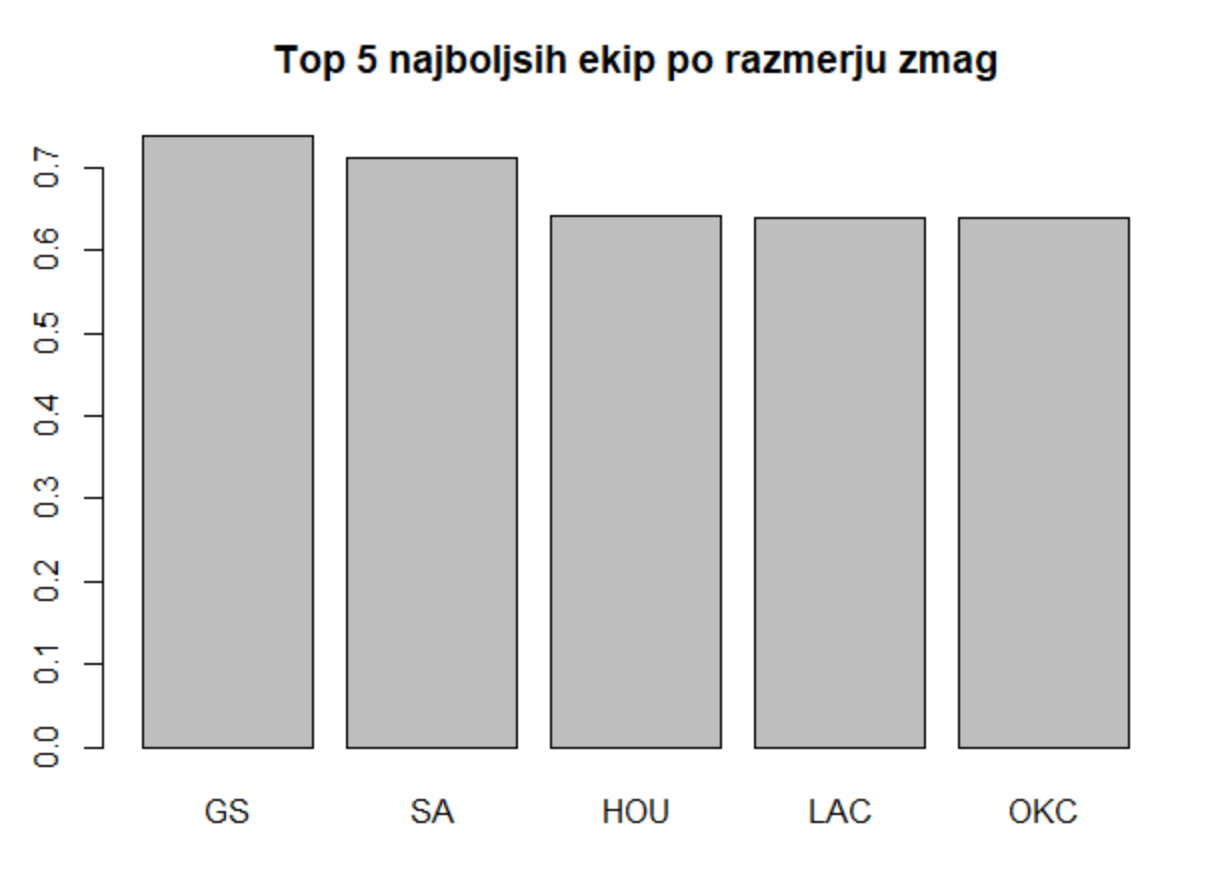


homeWins = table(md$homePTS > md$awayPTS);

homeWinPercentage <- homeWins / sum(homeWins)

pie(homeWins, labels=round(homeWinPercentage \* 100, 2),

    main="Zmage doma (modra) VS zmage v gostovanju (bela)")



winRatioByTeam = list();

for (team in TEAMS) {

    homeGames <- md$homeAbbr == team;

    awayGames <- md$awayAbbr == team;

    homeWins <- md$homePTS[homeGames] > md$awayPTS[homeGames];

    awayWins <- md$homePTS[awayGames] < md$awayPTS[awayGames];

    totalWins  <- sum(homeWins) + sum(awayWins);

    totalGames <- sum(homeGames) + sum(awayGames);

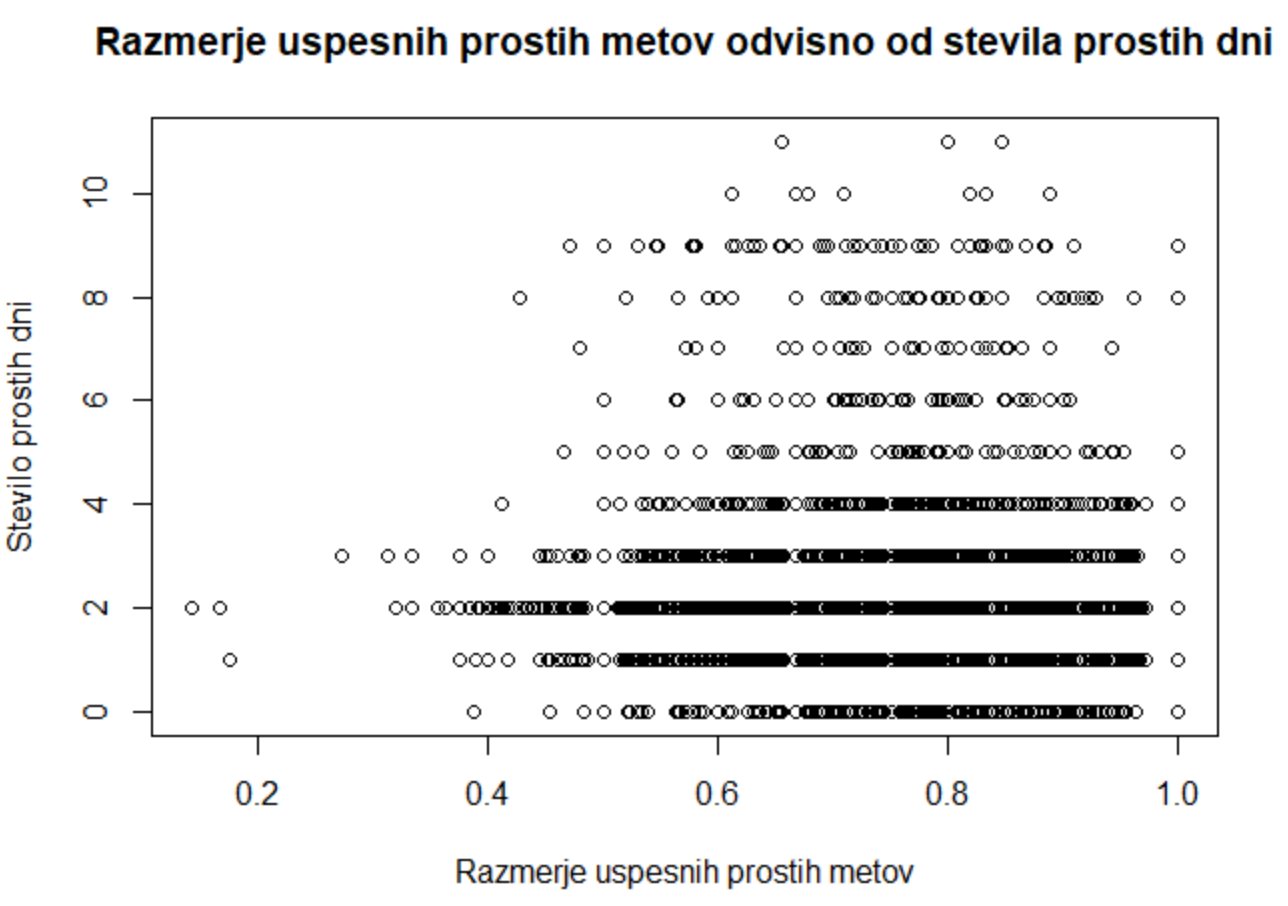
    ratio <- totalWins / totalGames;

    winRatioByTeam[[team]] <- ratio;

}

winRatioByTeam <- sort(unlist(winRatioByTeam), decreasing=TRUE);

barplot(winRatioByTeam[1:5], names=names(winRatioByTeam)[1:5], main="Top 5 najboljsih ekip po razmerju zmag");



homeFreeShotSuccessRatio = md$homeFTM / md$homeFTA;

awayFreeShotSuccessRatio = md$awayFTM / md$awayFTA;

totalFreeShotSuccessRatio = append(homeFreeShotSuccessRatio, awayFreeShotSuccessRatio);

daysOff = append(md$homeDayOff, md$awayDayOff);

plot(totalFreeShotSuccessRatio,daysOff, main="Razmerje uspesnih prostih metov odvisno od stevila prostih dni", xlab="Razmerje uspesnih prostih metov", ylab="Stevilo prostih dni");

## Strukturiranje podatkov

Podatke iger sva predstavila kot statistiko iz prejsnjih iger ekip. Ker prve igre ekip v podatkovni množici nimajo predhodnih iger, iz katerih bi lahko izracunali statistiko, sva te vrstice izpustila in jih uporabila le za racunanje statistike ekipe sledecih iger.

Za predstavitev ekipe sva izbrala atribute:

* **Home/AwayPTS** - Povprecje tock na igro
* **Home/AwayORBR** – delez skokov v napadu izmed vseh skokov
* **Home/AwayDRBR** – delez skokov v obrambi izmed vseh skokov
* **Home/AwayFGR** – delez uspesnih metov iz igre
* **Home/Away3PTR**  delez uspesnih metov za tri tocke
* **Home/Away2PTR** – delez uspesnih metov za dve tocki
* **Home/AwayFTR** – delez uspesnih prostih metov
* **Home/AwayWins –** delez zmag iz vseh preteklih iger
* **pastMactchScoreDifference** – skupna razlika tock med ekipama izmed preteklih iger

Atributa, ki ju napovedujemo:

* **isHomeWinner** – ali je v trenutni tekmi zmagala domaca ekipa
* **scoreDifference**  - razlika tock v trenutni tekmi (ce je zmagovalec **home** je pozitivna)

Atribute sva ocenila z metrikami **GainRatio, Gini, Relief in MSEofMean.**

Vse metrike so originalne atribute iz zacetka naloge ocenile zelo nizko.

**GainRatio** je najbolje ocenill atributa **awayDRBR** in **awayORBR**, **awayPTS** in **pastMatchesScoreDifference**.

**Gini** je najbolje ocenil atribute **homeWins, awayWins**  in  **patsMatchesScoreDifference**

**Relief** je najbolje ocenil **awayPTS, awayFTR** in **homeWins.**

**MseOfMean** je najbolje ocenil **homeWins, awayWins** in **homeFGR.**

## Klasifikacija

Za podatke za trening sva uporabila prvih 70% kronolosko urejenih podatkov, za testiranje pa preostalih 30%.

Odlocitveno drevo

Drevo sva implementirala s knjiznico rpart. S **CP** parametrom sva dolocilanivo kompleksnosti odlocitvenega drevesa. Sprva je CP parameter nastavljen na 0 (uporablja privzeto vrednost), ko pa je drevo zgrajeno pa printcp evaluira kvaliteto drevesa pri razlicnih vrednostih CP parametra. S funkcijo **prune** pa porezemo drevo, da ima model najmanjso napako (ravnovesje med natancnostjo modela in napako)

dt <- rpart(isHomeWinner ~ . - scoreDifference, data=train, cp=0)

rpart.plot(dt)

# rpart med gradnjo drevesa interno ocenjuje njegovo kvaliteto

printcp(dt)

tab <- printcp(dt)

# izberemo vrednost parametra cp, ki ustreza minimalni napaki internega presnega preverjanja

row <- which.min(tab[,"xerror"])

th <- mean(c(tab[row, "CP"], tab[row-1, "CP"]))

# porezemo drevo z izbrano nastavitvijo

dt <- prune(dt, cp=th)

rpart.plot(dt)

**Classification Accuracy: 59.4%**

**Sensitivity: 80.4%**

**Specificity: 30%**

**Precision: 61.6%**

**Brier Score: 0.485**

Naivni bayesov klasifikator

Naivni bayesov klasifikator predvideva, da so vsi atributi neodvisni drug od drugega.

Model sva zgradila s pomocjo knjiznice **e1071.**

library(e1071)

nb <- naiveBayes(isHomeWinner ~ . - scoreDifference, data = train)

predicted <- predict(nb, test, type="class")

**Classification Accuracy: 59%**

**Sensitivity: 57.6%**

**Specificity: 61%**

**Precision: 67.3%**

**Brier Score: 0.60**

Random Forest

Random Forest sva implementirala s knjiznico CORElearn. Pri random forest sva dobila presenetljive rezultate, saj je model po vseh metrikah perfekten. Ker veva, da rezultata tekem vnaprej ni mogoce 100% predvidevati, lahko ovrzeva hipotezo, da sva sestavila perfekten model. Mozno je, da je prislo do *overfitinga*,pojav pri katerem se model prilagodi podatkom ucenje tako natancno, da ne more generalizirati znanja na nove primere. Vendar pa ne znava ugotoviti kako bi prislo do tega, saj sva podatke za trening in testiranje strogo locila, podobnost med trening pa test podatki pa ni velika.

rf <- CoreModel(isHomeWinner ~ . - scoreDifference, data = train, model="rf")

predicted <- predict(rf, test, type="class")

## Regresija

Ker so atributi porazdeljeni priblizno normalno, atributov nisva transformirala.

Trivialni model

Je model, ki ki ne upošteva nobenih prediktorjev in za vsak primer v podatkovnem nizu predvideva enako vrednost za ciljno spremenljivko, v najinem primeru povprecke attributa scoreDifference. Ta model se uporablja za za osnovno primerjavo za druge modele. Ce drugi modeli ne presegajo trivialnega modela, nam to pove, da prediktorji, ki jih uporabljajo ne prispevajo k izboljsanju napovedi.

meanVal <- mean(train$scoreDifference);

predTrivial <- rep(meanVal, nrow(test))

**Mae: 1.12**

**Mse: 1.51**

Linearna regresija

Za linearno regresijo sva uporabila *leave-one-out cross-validation*: za vsako vrstico sva naredila nov podatkovno mnozico, ki ne vkljucuje te vrstice. Nato sva na novi podatkovni mnozici naredila linearno regresijo in dobila model. Dobljeni model sva uporabila za napovedovanje ciljne spremenljivke trenutne vrstice.

predicted <- vector()

for (i in 1:nrow(structuredData))

{

    model <- lm(scoreDifference ~ . - isHomeWinner, structuredData[-i,])

    predicted[i] <- predict(model, structuredData[i,])

}

**MAE: 10**

**MSE: 163**

**RMAE: 0.93**

**RMSE: 0.90**

K-najblizjih sosedov

KNN za vsak nov primer, ki ga napovedujemo, poisce *5* najbolj podobnih primerov iz podatkovne mnozice. Napoved ciljne spremenljivke za nov primer doloci kot povprecje napovedi ciljne spremenljivke za 5 najbolj podobnih primerov.

knn.model <- kknn(scoreDifference ~ . - isHomeWinner, train, test, k = 5)

predicted <- fitted(knn.model)

**MAE: 11.9**

**MSE: 227**

**RMAE: 1.1**

**RMSE: 1.22**

## Evalvacija modelov

## Modeli za klasifikacijo:

* odlocitveno drevo:

**Classification Accuracy: 59.4%**

**Sensitivity: 80.4%**

**Specificity: 30%**

**Precision: 61.6%**

**Brier Score: 0.485**

Odloctveno drevo je dobro napovedalo tekme, ko so zmagali domačini (**Sensitivity** – 80%), a zato veliko slabše, ko so izgubile (**specificity** – samo 30%). To si lahko razlozimo z dejstvom, da dejansko domace ekipe zmagujejo veckrat, torej ze v splosnem imamo vecjo moznost ugotovitve, ce napovemo zmago domacih.

* Naivni bayesov klasifikator:

**Classification Accuracy: 59%**

**Sensitivity: 57.6%**

**Specificity: 61%**

**Precision: 67.3%**

**Brier Score: 0.60**

Naivni Bayesov klasifikator je bolj uravnotezeno dobro napovedal, torej ni toliko laznih pozitivnih rezultate (domace zmage). Ima boljsi Brier Score in Preciznos, vendar ni veliko boljsi od odlocitvenega drevesa.

Oba modela nista dobra, ampak pri sportu na rezultat vpliva veliko vec spremenljivk kot jih imamo v tej mnozici, zato bi za izboljsanje modelov bilo smiselno uporabiti vec razlicnih podatkov.

### Modeli za regresijo:

* Trivialni model:

**Mae: 1.12**

**Mse: 1.51**

Trivialni model je uporabljen za primerjavo z drugimi modeli, da vidimo, ce sploh pripomorejo k izboljsanju napovedi.

* Linearna regresija:

**MAE: 10**

**MSE: 163**

**RMAE: 3.16**

**RMSE: 12.77**

Nizja MAE in MSE pomenita boljsi model. Tukaj je MAE in MSE bistveno vecja kot pri trivialnem modelu, zato je ta model neuporaben. Tukaj sva zracunala se RMAE in RMSE, ki sta zgolj korena prvih dveh.

* K-najblizjih sosedov:

**MAE: 11.9**

**MSE: 227**

**RMAE: 3.45**

**RMSE: 15.1**

Tudi pr item modelu vidimo, da nam ne pomaga.