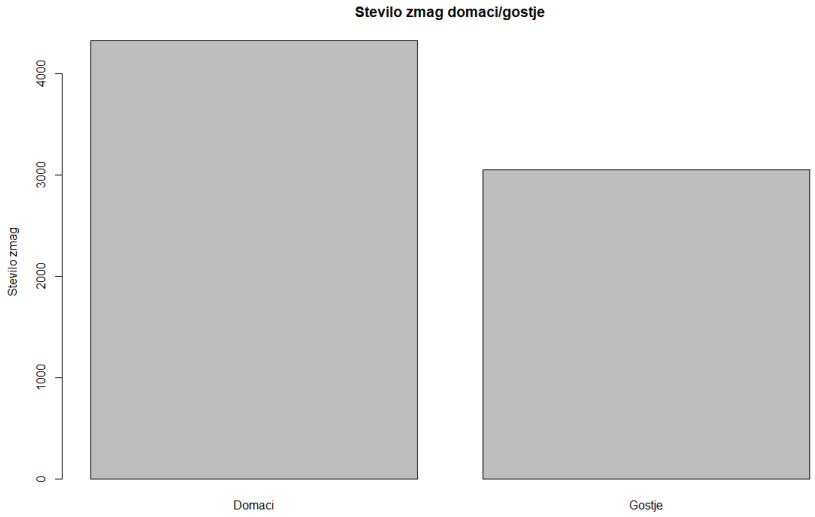
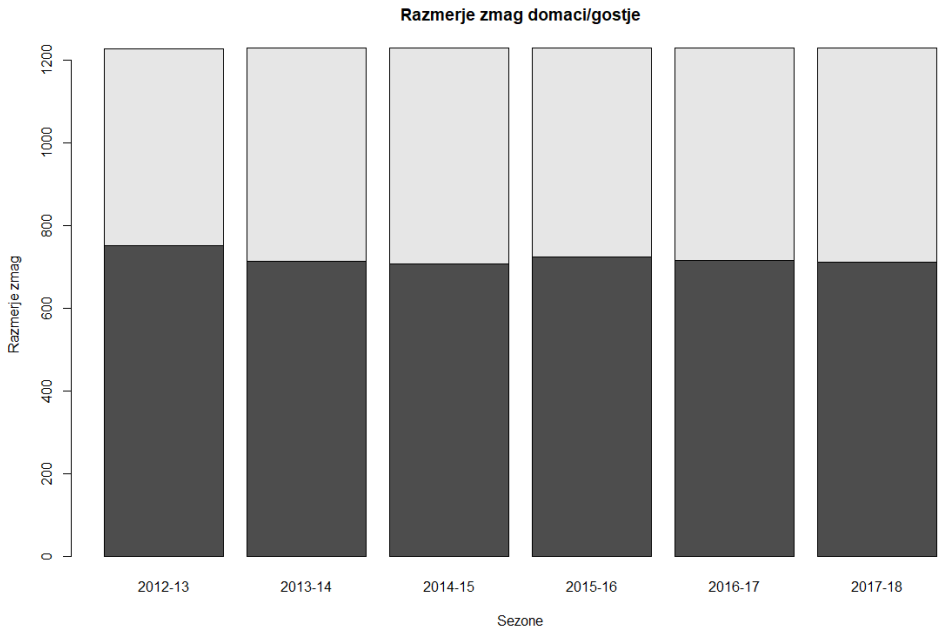
1. Seminarska naloga

UMETNA INTELIGENCA

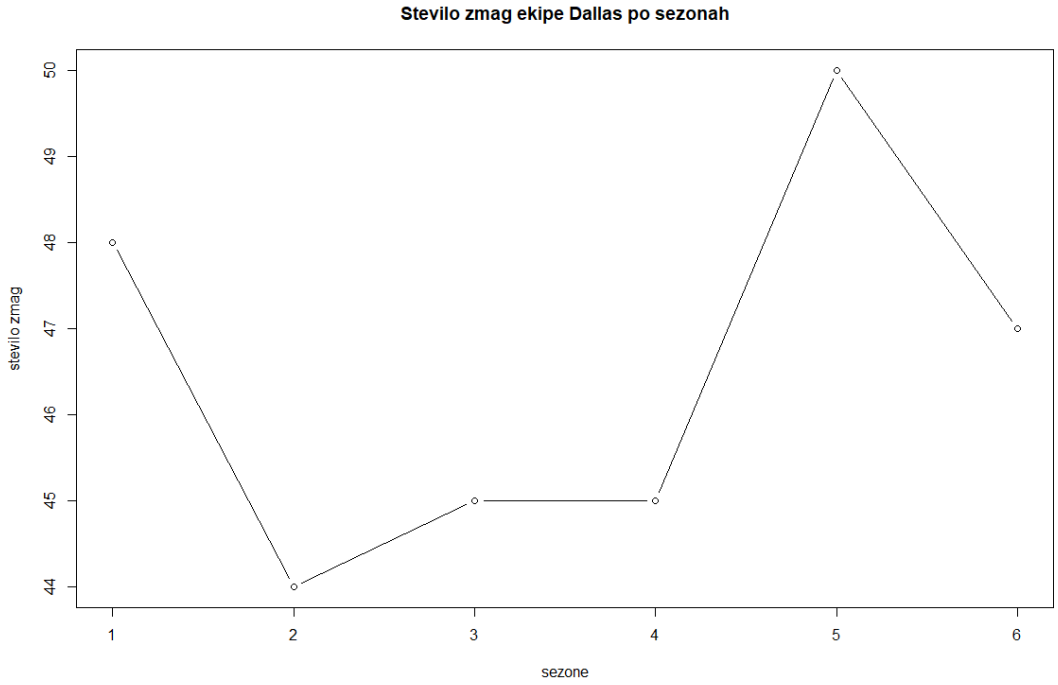
Avtor: Matic Knez

1. Vizualizacija podatkov

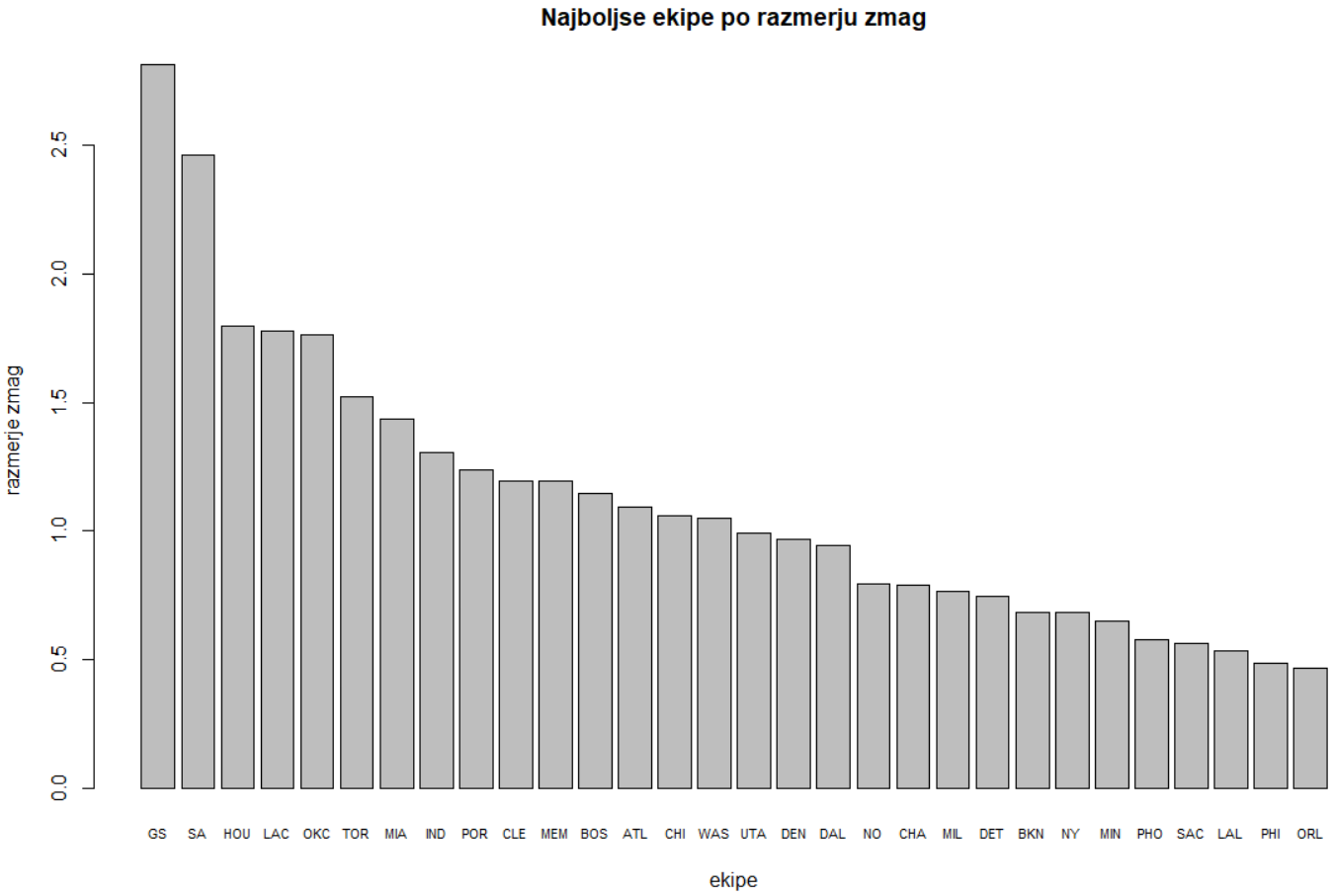




Zgornja grafa prikazujeta razmerje zmag med domačini in gosti. Opazimo, da je zmaga domačinov dosti bolj verjetna.



Graf prikazuje število zmag ekipe Dallas po sezonah od 2012-13 do 2017-18. Vidimo, da smo med



letom 2013 do 2016 igrali nekoliko slabše, vrh pa so imeli leta 2017.

Graf prikazuje porazdelitev najboljših ekip po razmerje zmag in porazov.

2. Priprava podatkov

Naprej sem atribute vseh metov in uspešnih metov zamenjal z atributi **razmerjem uspešnih metov**.

- Dodal (FGR, 2PR, 3PR, FTR) – home/away
- Odstranil (FGA, 2PA, 3PA, FTA, FGM, 2PM, 3PM, FTM) – home/away

Nato sem dodal atribut **razmerje uspešnosti napada**. Izračunal sem ga iz razmerje **števila uspešnih zadetkov** in **vseh napadov**. Kot napad sem štel **zadetek** ali **zgrešen met**, **izgubljena žoga** ali **ukradena žoga**. Vse napade sem izračunal iz števila vseh metov + izgubljene žoge + ukradene žoge.

- Dodal (AttR) – home/away
- Odstranil (TO, STL) – home/away

Dodal sem se atribut **forme** ekipe, ki nam pove v kasni formi je ekipa po razmerje zmag zadnjih 5 odigranih iger.

- Dodal (form) - home/away

Za napovedujem zmagovalca po končani X četrtini sem dodal atribut razlike rezultata po končani četrtini. Razliko rezultata izračunamo z razliko točk obeh skupin po X četrtini.

- Dodal (diffScore)

Za klasifikacijo sem dodal ciljno spremenljivko (win) – zmaga domačih.

Za regresijo sem dodal ciljno spremenljivko (diff)– razlika rezultata.

Vsaka tekma v učni množici predstavlja povprečje atributov zadnjih 5 iger te ekipe.

```
# dodamo razmerja zadetih (namesto vseh in uspesnih)
# FG - ratio

df['awayFGR'] = df$awayFGM / df$awayFGA
df['homeFGR'] = df$homeFGM / df$homeFGA

df['away2PR'] = df$away2PM / df$away2PA
df['home2PR'] = df$home2PM / df$home2PA

df['away3PR'] = df$away3PM / df$away3PA
df['home3PR'] = df$home3PM / df$home3PA

df['awayFTR'] = df$awayFTM / df$awayFTA
df['homeFTR'] = df$homeFTM / df$homeFTA
summary(df)

napadi_h = df$homeFGA + df$homeTO + df$homeSTL
#print(df$homeFGM / napadi_h)
df["homeAttR"] = df$homeFGM / napadi_h

napadi_a = df$awayFGA + df$awayTO + df$awaySTL
#print(df$awayFGM / napadi)
df["awayAttR"] = df$awayFGM / napadi_a

# razmerje uspesnih napadov
# st zadetkov / vseh napadov
# vseh napadov = st. vseh metov + izgubjene zoge + ukradene zoge
```

Dodano sem se odstranil atribute, ki menim, da nimajo pomena, za predstavitev ekipe.

- Ime ekipe (Addr) – home/away
- Sezona ekipe (GmSeason) – home/away
- Čas tekme (gmDate) – home/away
- Število doseženih točk (PTS, PTS1, PTS2, PTS3, PTS4) – home/away

Atribute, ki so mi ostali sem ocenil z:

- Gini index
- Gain ratio
- ReleifF

```
sort(attrEval(win ~ ., train, "Gini", binaryEvaluation=T), decreasing = T)
sort(attrEval(win ~ ., train, "GainRatio", binaryEvaluation=T), decreasing = T)
sort(attrEval(win ~ ., train, "ReliefFequalK", binaryEvaluation=T), decreasing = T)
```

Nato sem se dodano odstranil najslabše ocenjene atribute:

- Odstranil (DayOff, TRB) – home/away

```
> head(sort(attrEval(win ~ ., train, "Gini", binaryEvaluation=T), decreasing = T), 5)
 diffScore    home2PR    homeBLK    awayBLK   awayForm
0.13818919 0.02261434 0.01385681 0.01379345 0.01268793
> head(sort(attrEval(win ~ ., train, "GainRatio", binaryEvaluation=T), decreasing = T), 5)
diffScore   homeDRB   homeORB   awayORB   away3PR
0.3655222 0.2192381 0.2037964 0.2037964 0.1089813
> head(sort(attrEval(win ~ ., train, "ReliefFequalK", binaryEvaluation=T), decreasing = T), 5)
 diffScore    home2PR    homeDRB    homeFGR    homeFTR
0.36957929 0.08317152 0.06504854 0.05501618 0.05210356
>
```

Ce v ocenjevanje vključimo se atribut razlike rezultatov po končani četrtini (diffScore). Opazimo, da je krepko na 1. mestu. Pomeni, da si bomo s tem atributom najbolj pomagali pri klasifikaciji.

3. Klasifikacija

Kot učno množico sem vzel tekme 4ih sezon, ekipe za katero napovedujemo.

(sezone 2012-13, 2013-14, 2014-15, 2015-16) – učna množica

(sezona 2016-17) – testna množica

Napoved za tekmo **DAL (home)** – **CHI (away)**

Vrstice učne množice predstavljajo tekme, kjer:

- Domači (homeAbbr) == **DAL** ali
- Gostje (awayAbbr) == **CHI**

V posamezni sezoni je vsaka ekipa igrala ~ 40 zato imamo v učni množici posamezne sezone ~ 80 tekem (40 domači + 40 gosti – presek). Ko seštejemo vse 4 sezone skupaj imamo približno 360 tekem (vrstic) v učni množici.

Za gradnjo učne množice sem napisal metode:

- makeDf (data set z vsemi atributi)
- makeTeamMean (izračuna povprečne vrednosti zadnjih 5 odigranih iger, odstrani nepotrebne atribute)
- makeSeasonMean (zgradi data set z vsemi odigranimi igrami domače in gostujoče ekipe v podani sezoni)

Napovedoval sem izid igre za različne pare ekip v sezoni 5 (2016-17). Za ciljno spremenljivko sem uporabil logični atribut (win)

- TRUE – zmaga domače ekipe
- FALSE – zmaga gostujoče ekipe

Napovedal sem rezultat za 200 ekip. (n=200)

Metode klasifikacije in ocenjevanje modelov se nahajajo v datoteki classification.R

Metode za napovedovanje in izračun natančnosti napovedovanja so v prediction.R

Dodatne funkcije za ocenjevanje in testiranje modelov v functions.R

Za napovedovanje sem uporabil naslednje modele:

- Odločitveno drevo (tree)
- Naključni gozd (random forest)
- K – najbližjih sosedov (knn) k=10
- Naivni Bayes

```
> vecinski = max(table(observed)) / sum(table(observed))
> rf = mean(observed == predrf)
> tr = mean(observed == predtr)
> knn = mean(observed == predkn)
> print(paste("Vecinski: ", vecinski))
[1] "Vecinski:  0.565"
> print(paste("Drevo: ", tr))
[1] "Drevo:  0.535"
> print(paste("knn: ", knn))
[1] "knn:  0.56"
> print(paste("Random forest: ", rf))
[1] "Random forest:  0.585"
>
```

Kot osnovo imamo večinski razred, ki ima doseže točnost **0.56**

Najboljšo natančnost je dosegel model naključnega gozda (random forest), s točnostjo **0.58**

Naključni gozd ima malo večjo natančnost kot večinski razred, pomeni, da je uporaben.

Medtem, ko sta modela najbližjih sosedov (knn) in odločitveno drevo neuporabna, ker imata manjšo točnost, kot večinski klasifikator.

```
Levels: FALSE TRUE
> vecinski = max(table(observed)) / sum(table(observed))
> rf = mean(observed == predrf)
> tr = mean(observed == predtr)
> knn = mean(observed == predkn)
> print(paste("Vecinski: ", vecinski))
[1] "Vecinski:  0.545"
> print(paste("Drevo: ", tr))
[1] "Drevo:  0.76"
> print(paste("knn: ", knn))
[1] "knn:  0.7"
> print(paste("Random forest: ", rf))
[1] "Random forest:  0.795"
>
```

V tem primeru smo upoštevali **dodaten atribut** (diffScore), ki nam pove razliko v točkah po odigrani 3/4 igre. Vidimo da je natančnost pri vseh modelih razen večinskega klasifikatorja seveda zelo narasla.

Tudi v tem primeru je zmagal klasifikator naključnega gozda, ki je od večinskega klasifikatorja boljši za kar 25%.

4. Regresija

Podobno kot pri klasifikaciji, sem tudi pri regresiji zgradil učno množico po istem principu. Z razliko v ciljem atributu. Namesto zmage (win) sem uporabil razliko rezultata (diff).

Za napovedovanje sem uporabil naslednje modele:

- Odločitveno drevo (tree)
- Naključni gozd (random forest)
- K – najbližjih sosedov (knn) k=10
- SVM

Za ocenjevanje modelov sem uporabil:

- mae (srednja absolutna napaka)
- rmae (relativna srednja napaka)

```
> rmaerf = round(sum(abs(observed - predrf)) / sum(abs(observed - traindiff)), d
> rmaetr = round(sum(abs(observed - predtr)) / sum(abs(observed - traindiff)), d
> rmaeknn = round(sum(abs(observed - predkn)) / sum(abs(observed - traindiff)), 
> rf = mean(abs(observed - predrf))
> tr = mean(abs(observed - predtr))
> knn = mean(abs(observed - predkn))
> print(paste("Drevo: (mae)", round(tr, digits=3), "(rmae) ", rmaetr))
[1] "Drevo: (mae) 12.065 (rmae)  1.179"
> print(paste("knn: (mae)", round(knn, digits=3), "(rmae) ", rmaeknn))
[1] "knn: (mae) 10.545 (rmae)  1.031"
> print(paste("Random forest: (mae)", round(rf, digits=3), "(rmae) ", rmaerf))
[1] "Random forest: (mae) 10.117 (rmae)  0.989"
>
```

Opazimo, da se je ponovno najbolje odrezal model naključnega gozda z najmanjšo relativno napako 0.99, sledi mu knn z napako 1.03 in najslabše odločitveno drevo z napako 1.18.

```
> rmaerf = round(sum(abs(observed - predrf)) / sum(abs(observed - traindiff)), 
> rmaetr = round(sum(abs(observed - predtr)) / sum(abs(observed - traindiff)), 
> rmaeknn = round(sum(abs(observed - predkn)) / sum(abs(observed - traindiff)),
> rf = mean(abs(observed - predrf))
> tr = mean(abs(observed - predtr))
> knn = mean(abs(observed - predkn))
> print(paste("Drevo: (mae)", round(tr, digits=3), "(rmae) ", rmaetr))
[1] "Drevo: (mae) 7.101 (rmae)  0.686"
> print(paste("knn: (mae)", round(knn, digits=3), "(rmae) ", rmaeknn))
[1] "knn: (mae) 8.156 (rmae)  0.788"
> print(paste("Random forest: (mae)", round(rf, digits=3), "(rmae) ", rmaerf))
[1] "Random forest: (mae) 6.729 (rmae)  0.65"
>
```

Podobno kot pri klasifikaciji, v primeru, da uporabimo atribut razlike v rezultatu po 3/4 igre dobimo veliko boljše rezultate. Naključni gozd z najmanjšo napako 0.65, nato odločitveno drevo z napako 0.68 in na koncu knn z napako 0.78.