Dokumentacija Zadaće 2 iz predmeta Mašinsko učenje

Beglerović Vedad Beglerović Vildana Buturović Lejla

Elektrotehnički fakultet Sarajevo
9. januar 2022.

Sadržaj

1 Zadatak 1

1.1 Predprocesiranje podataka

Za izradu Zadaće 2 koristit ćemo podatke koje smo dobili kao rezultat transformacija iz zadatka 1 Zadaće 1 sa određenim izmjenama. U nastavku ćemo navesti transformacije koje su urađene nad originalnim podacima:

- odbacivanje atributa DailyCharges zbog visokog stepena korelacije sa atributom MonthlyCharges
- odbacivanje atributa TotalCharges zbog viskog stepena korelacije sa atributom tenure
- odbacivanje atributa MultipleLines i StreamingMovies zbog viskog stepena korelacije sa atributima PhoneService i InternetService, respektivno
- popunjavanje nedostajućih vrijednosti brojčanih atributa sa medianom (nedostajuće vrijednosti atributa tenure smo popunili medianom za različite vrijednosti kategoričke varijable Contract)
- popunjavanje nedostajućih vrijednosti kategoričkih varijabli na osnovu vrijednosti drugih varijabli sa kojim su povezane (npr. PhoneService treba imati vrijednost "No" ukoliko atribut MultipleLines ima vrijednost "No phone service" i sl.)
- popunjavanje nedostajućih vrijednosti kategoričkih varijabli koji nemaju logičku povezanost sa drugim varijablama na način da se naizmjenično popunjavaju sve moguće vrijednosti koje atribut može imati (Gender i Dependents) ili da se nedostajuće vrijednosti popune vrijednošću koju ima većina instanci za taj atribut (preostali atributi)
- odbacivanje outlier-a (instanca koja ima vrijednost atributa Dependents "Maybe", instanca koja ima vrijednost atributa PaymentsMethod "abcd", instanca koja ima negativnu vrijednost za MonthlyCharges, instance koje imaju vrijednost 0 za tenure atribut)
- odbacivanje instanci koje nemaju definisanu vrijednost za labelu klase

Napomena: Razlika u odnosu na dobijene predprocesirane podatke iz Zadatka 1 Zadaće 1 je ta što smo vratili kolonu InternetService, jer smo ustanovili da se na taj način dobijaju bolji rezultati i što smo nedostajuće vrijednosti za varijablu tenure popunili na osnovu mediana za različite vrijednosti kategoričke varijable Contract. Također, nismo uradili min-max normalizaciju, već smo ponovo u okviru ove zadaće pri kreiranju modela isprobali i decimalno skaliranje i z-score normalizaciju, te izvršili onu koja daje bolje rezultate za taj model.

Nakon što smo učitali procesirani skup podataka iz Zadaće 1, uradili smo faktorizaciju kategoričkih varijabli, te podijelili dataset na trening i testni podskup holdout metodom sa nasumičnim redovima podataka. S obzirom na veličinu dateset-a, odlučili smo da se u trening podskupu nalazi 85%, a u testnom 15% instanci skupa podataka.

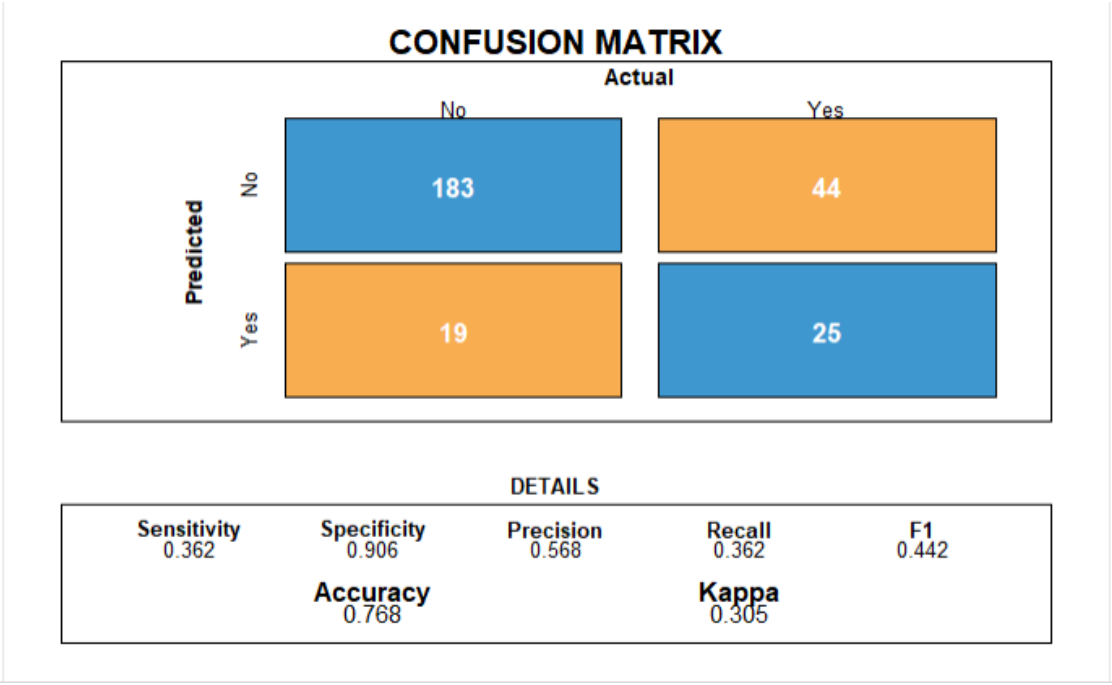
1.2 KNN model predikcije

Budući da KNN model predikcije nije pogodan za rad sa kategoričkim varijablama, sve kategoričke varijable, osim labele klase, smo pretvorili u numeričke pomoću as.numeric() funkcije.

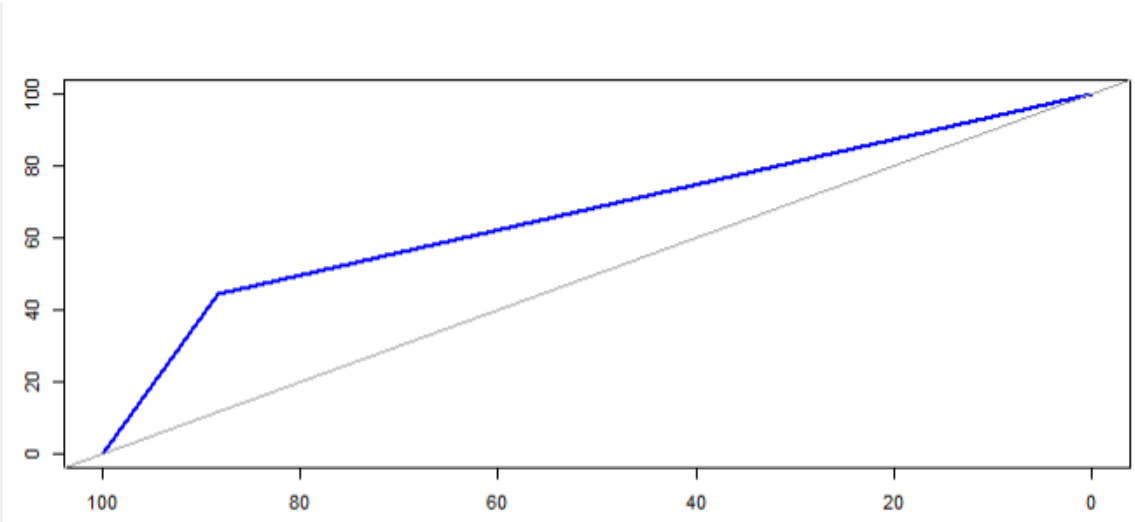
Pri izgradnji osnovnog KNN modela predikcije, vrijednost parametra k smo postavili na 15, jer se radi sa dataset-om koji nije pretjerano mali, ali isto tako, u poređenju sa nekim poznatijim dataset-ovima nije ni približno velik kao oni.

```
library(class)
library(caret)
predictions <- knn(train = subset(podaci_train_KNN, select = -c(Churn)), test = subset(
    podaci_test_KNN, select = -c(Churn)),
cl = podaci_train_KNN$Churn, k = 15)
```

Zatim smo uradili evaluaciju KNN modela pomoću konfuzijske matrice i ROC krive. Više puta smo pokrenuli kod za kreiranje modela i crtanje konfuzijske matrice (jer izgrađeni model dosta ovisi od podjele na trening i testni skup) te smo ustanovili da je prosječna tačnost koju smo dobijali oko 0.75, dok je kappa statistika oko 0.33. Rezultati dobijeni prilikom jednog od pokretanja su prikazani u nastavku:



Slika 1: Konfuzijska matrica za osnovni KNN model



Slika 2: ROC kriva za osnovni KNN model

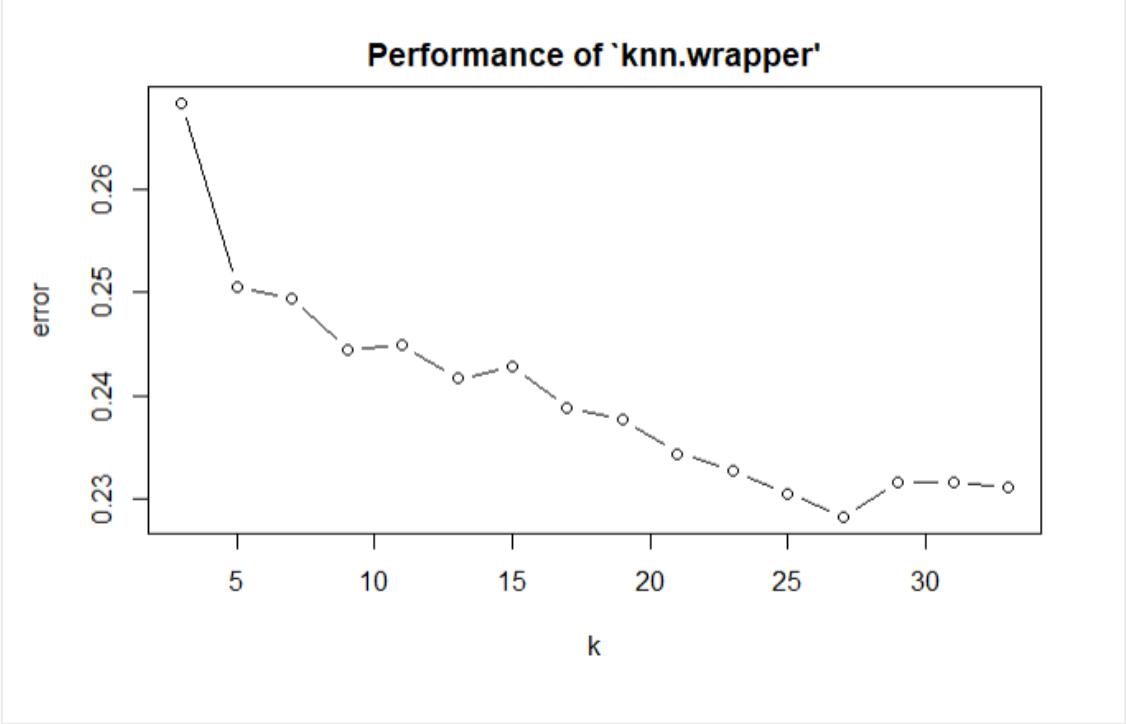
Sa slike vidimo da kreirani model ima puno veću specifičnost nego senzitivnost, što znači da bolje klasifikuje negativne instance nego pozitivne. Lošijoj senzitvnosti može doprinijeti i to što u skupu podataka imamo više negativnih nego pozitvnih instanci, odnosno dataset nije balansiran. Kako bi bili sigurniji u dobijene rezultate, uradili smo i k-fold validaciju. Dobijeni rezultati su prikazani u nastavku i oni potvrđuju naše pretpostavke:

```
10-fold validacija
Najveća tačnost: 0.7944444 , fold: 2, najveća kappa: 0.4397712 , fold: 5
Najmanja tačnost: 0.6888889 , fold: 6, najmanja kappa: 0.2022941 , fold: 9
Srednja tačnost: 0.7544475, srednja kappa: 0.3152902

5-fold validacija
Najveća tačnost: 0.7728532 , fold: 5, najveća kappa: 0.3394683 , fold: 1
Najmanja tačnost: 0.7361111 , fold: 3, najmanja kappa: 0.2473592 , fold: 3
Srednja tačnost: 0.7538704, srednja kappa: 0.3097538

Najbolja vrijednost k: 31
Najveća tačnost: 0.769957
```

Slika 3: Rezultati k-fold validacije za KNN model za k = 15

Zatim smo uradili tuning parametra k kako bi ustanovili za koju vrijednost parametra k model daje najveću tačnost. Za k smo naveli sve neparne brojeve u rasponu od 3 do 33. Parne brojeve nismo uzimali s obzirom da imamo dvije moguće vrijednosti za labelu klase, pa su zbog toga veće šanse da KNN model dođe u situaciju da ne zna koju klasu da pridruži instanci, jer podjednak broj susjeda pripada objema klasama. U tom slučaju se nasumično odabire klasa koja će se dodijeliti i samim tim efikasnost KNN modela slabi. Postupak smo ponovili više puta. Za najbolju vrijednost parametra k smo dobijali vrijednosti u rasponu od 25 do 31 (oboje uključeno), a najbolja tačnost za sve te vrijednosti k je bila približno ista (oko 77%). Dakle, sa povećanjem paramtera k greška se smanjivala, ali ne pretjerano značajno za veće vrijednosti k. Razlog ovome je što su instance koje pripadaju istoj klasi blizu jedna drugoj, ali puno veći uticaj ima to što je dataset nebalansiran, odnosno puno je više instanci sa negativnom labelom, nego sa pozitivnom, pa je bolje što se više susjeda uzima u obzir. Rezultati dobijeni prilikom

jednog pokretanja su prikazani u nastavku:

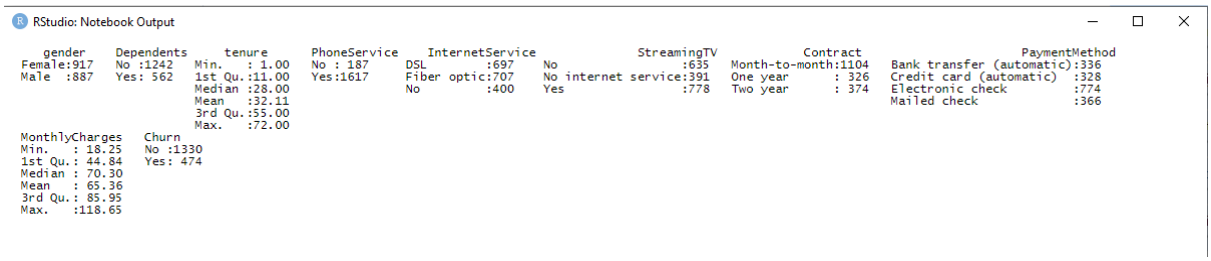


Slika 4: Vizualizacija greške klasifikacije pri tuningu parametra k

```
Najbolja vrijednost k: 27
Najveća tačnost: 0.7716268
```

Slika 5: Informacije o najboljem KNN modelu

KNN modelom predikcije i nije dobijena pretjerano velika tačnost. Vidimo da i sa povećanjem parametra k nećemo dobiti veću tačnost, tako da bi razlog ne tako velike tačnosti trebao biti u samim podacima nad kojima model radi. Pokušat ćemo dodatnim transformacijama nad podacima povećati tačnost. S obzirom da KNN model predikcije ne daje dobre rezultate ukoliko numerički atributi nemaju približno isti opseg vrijednosti (u tom slučaju zanemaruje se uticaj onih atributa koji imaju manji opseg), izvršili smo metodu summary nad našim podacima kako bi provjerili opsege numeričkih atributa tenure i MonthlyCharges.



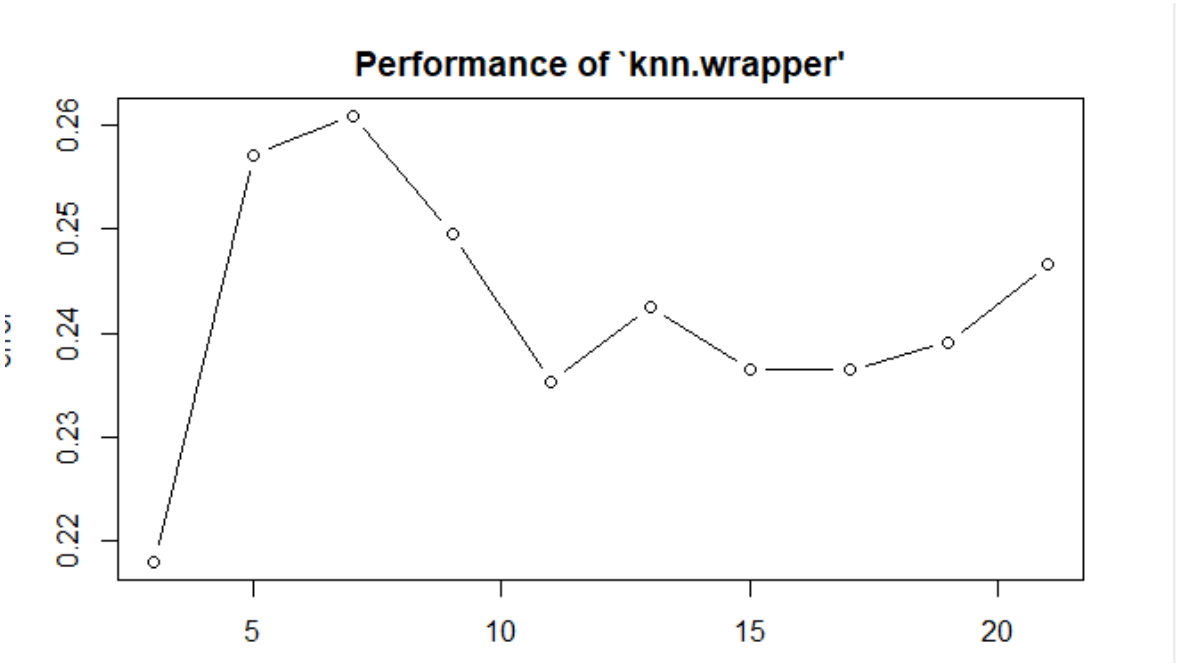
Slika 6: Prikaz informacija o setu podataka korištenjem summary() funkcije

Vidmo da je opseg vrijednosti koje može imati atribut tenure od 1 do 72 uključeno, a MonthlyCharges 18.25 do 118.65, te smo zbog toga uradili normalizaciju numeričkih podataka. Provjerili smo za koju normalizaciju

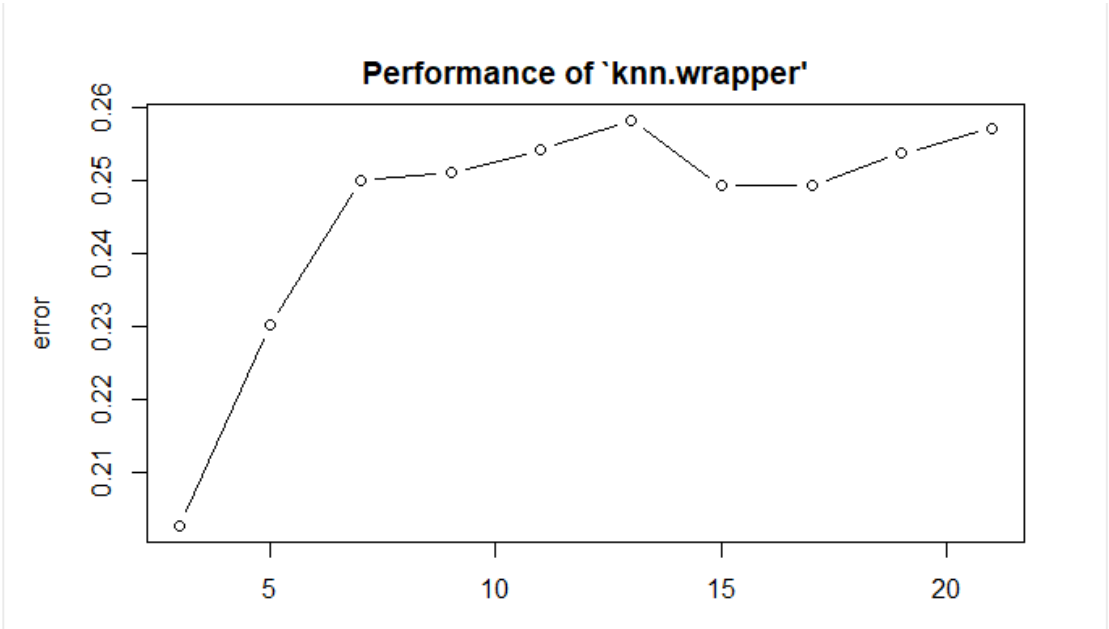
dobijamo bolje vrijednosti (min-max normalizacija ili decimalno skaliranje). Min-max normalizacijom smo sveli vrijednosti na opseg od 0 do 1. Normalizacije smo izvršili nad oba numerička atributa.

KNN model daje puno bolje rezultate ukoliko radi nad balansiranim setom podataka. S obzirom da u našem datesetu imamo 1330 instanci sa negativnom labelom klase i 474 instance sa pozitivnom labelom klase uradili smo balansiranje dataseta. Koristili smo oversampling metodu, jer nismo željeli da izgubimo podatke, s obzirom da naš dataset svakako nije pretjerano velik. Nakon balansiranja, u datasetu imamo po 1330 pozitivnih i negativnih instanci.

Ponovno smo uradili tuning hiperparametra k. Rezultati su prikazani u nastavku:



Slika 7: Vizualizacija greške klasifikacije pri tuningu parametra k nakon min-max normalizacije i balansiranja dataseta



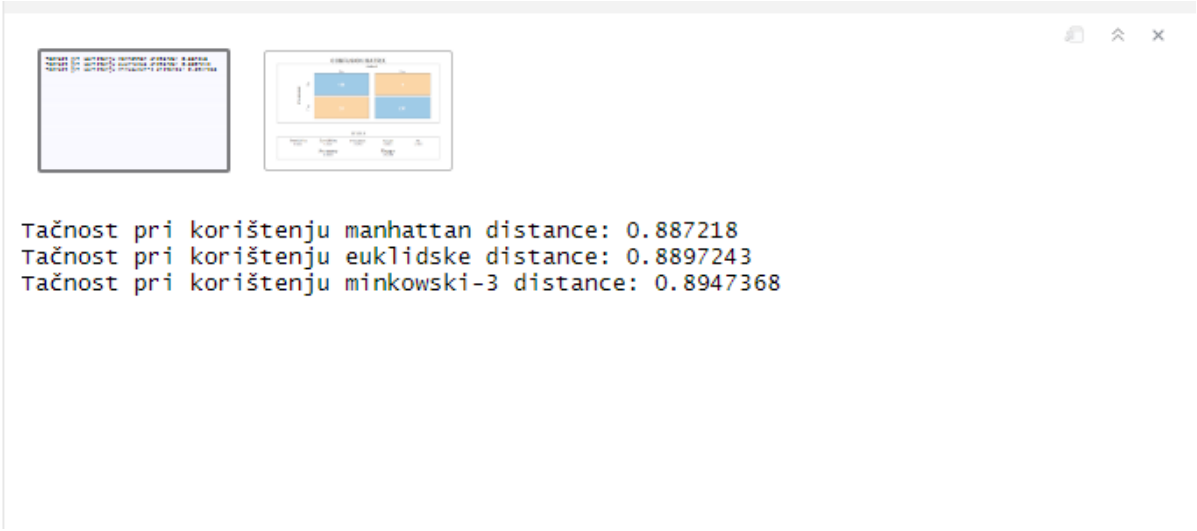
Slika 8: Vizualizacija greške klasifikacije pri tuningu parametra k nakon decimalnog skaliranja i balansiranja dataseta

```
Nakon min-max normalizacije i balansiranja podataka:
Najbolja vrijednost k: 3
Najveća tačnost: 0.7819549

Nakon decimalnog skaliranja i balansiranja podataka:
Najbolja vrijednost k: 3
Najveća tačnost: 0.7973684
```

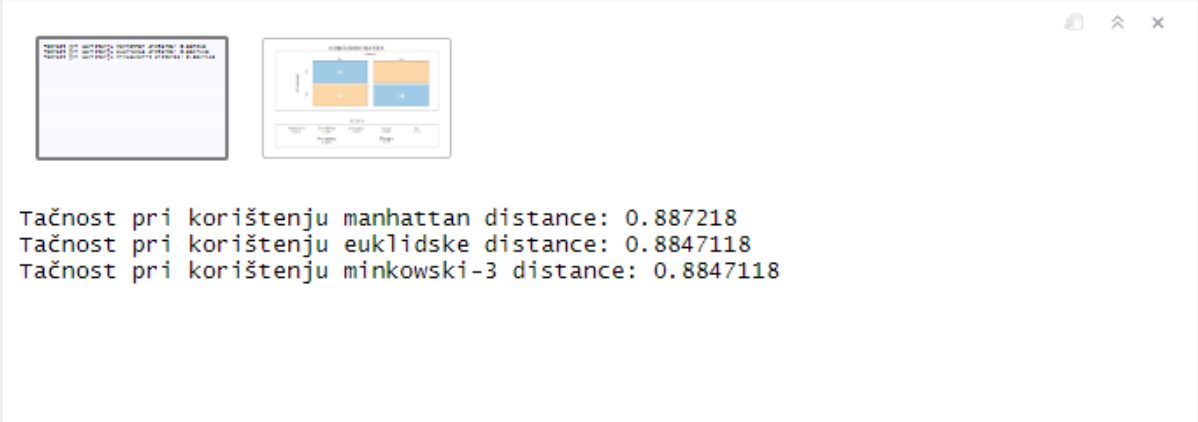
Slika 9: Informacije o najboljem KNN modelu

Vidimo da smo veću tačnost dobili kada smo KNN model primijenili nad podacima nad kojima smo izvršili decimalno skaliranje i balansiranje, nego kada smo primijenili nad podacima nad kojima smo izvršili min-max normalizaciju i balansiranje, ali u oba slučaja smo dobili veću tačnost nego kada podaci nisu bili balansirani i kada nije bila izvršena normalizacija nad numeričkim atributima. Najveću tačnost u oba slučaja smo dobili za k = 3.

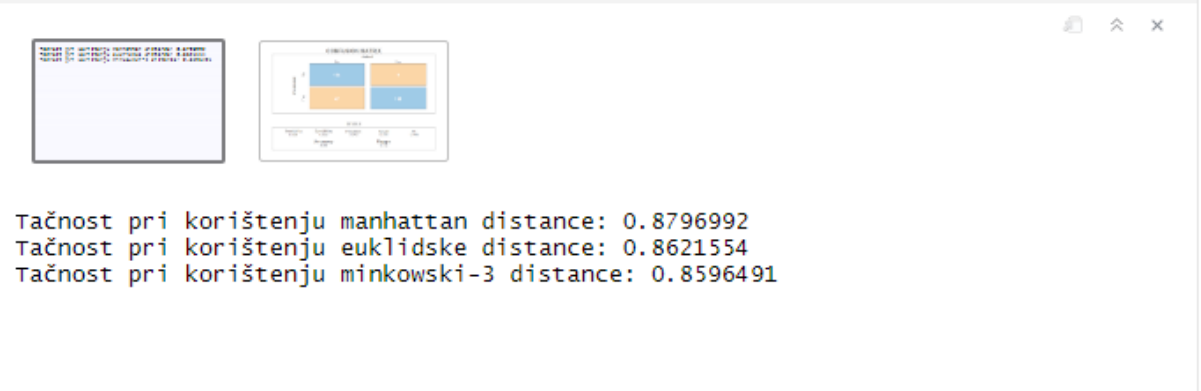
KNN model predikcije smo trenirali i testirali primjenom Manhattan i Minkowski distance (dosad je korištena Euklidska). S obzirom da rezultati koje dobijemo ovim modelima dosta ovise od podjele skupa podataka na trening i testni podskup, odnosno od toga koje distance ulaze u trening, a koje u testni, više puta smo izvršili treniranje i testiranje KNN modela, a rezultati neki od njih su prikazani u nastavku:



Slika 10: Različite metrike distance

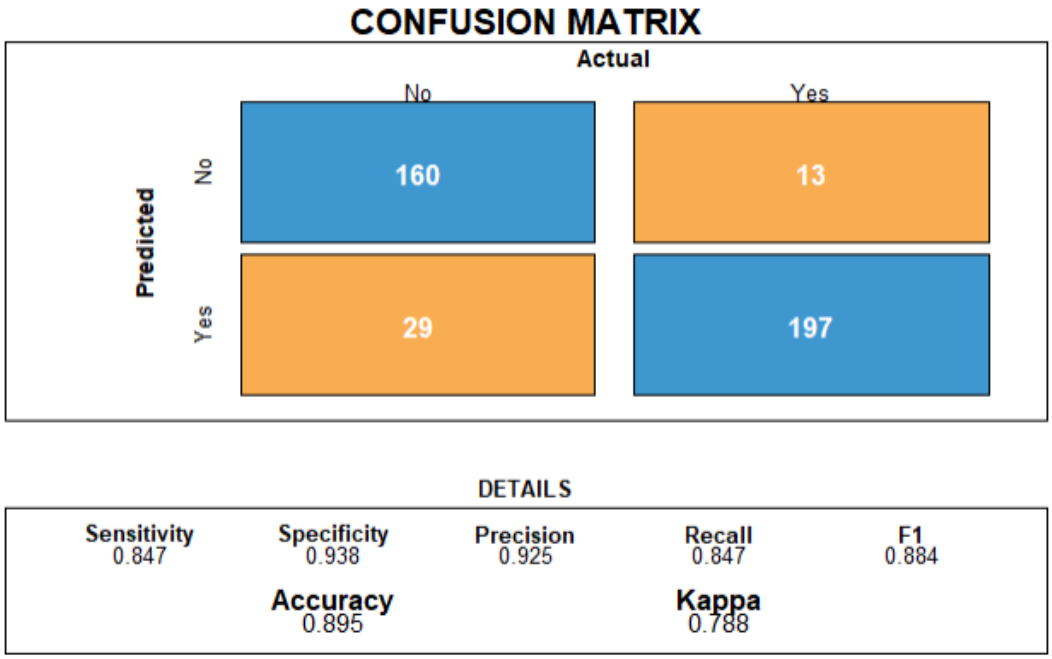


Slika 11: Različite metrike distance

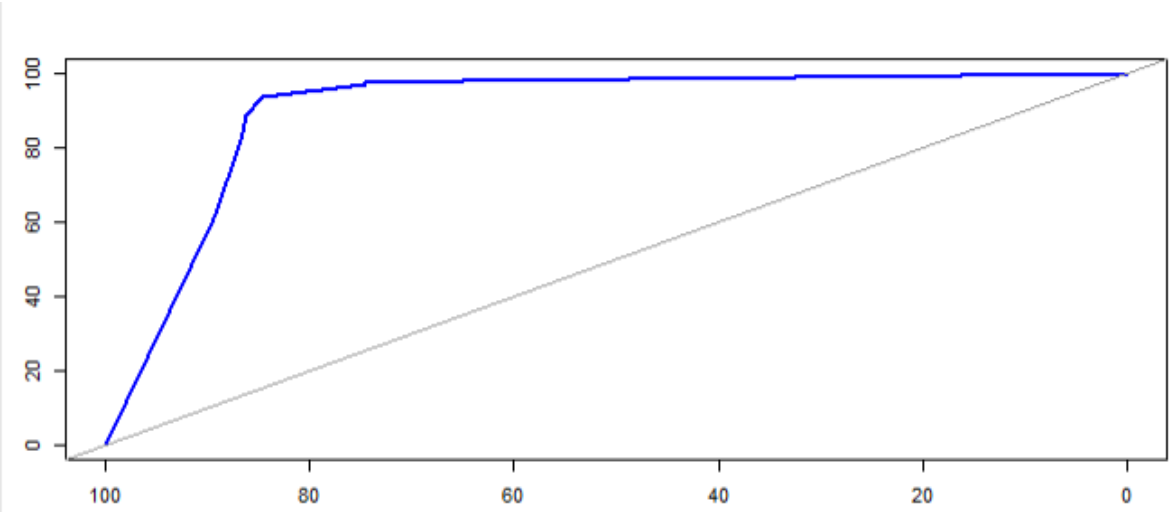


Slika 12: Različite metrike distance

Vidimo da za različite podjele dataseta na trening i testni, KNN model i za različite metrike distance daje najbolje rezultate. Primjetno je da su sada podaci značajno bolji nego na samom početku. U nastavku je prikazana konfuzijska matrica i ROC kriva za KNN model koji koristi Minkowski distancu koja je dobijena u jednom od pokretanja, jer je za nju dobijena najveća tačnost (u jednom pokretanju 0.90) .



Slika 13: Konfuzijska matrica za KNN model (Minkowski metrika distance)



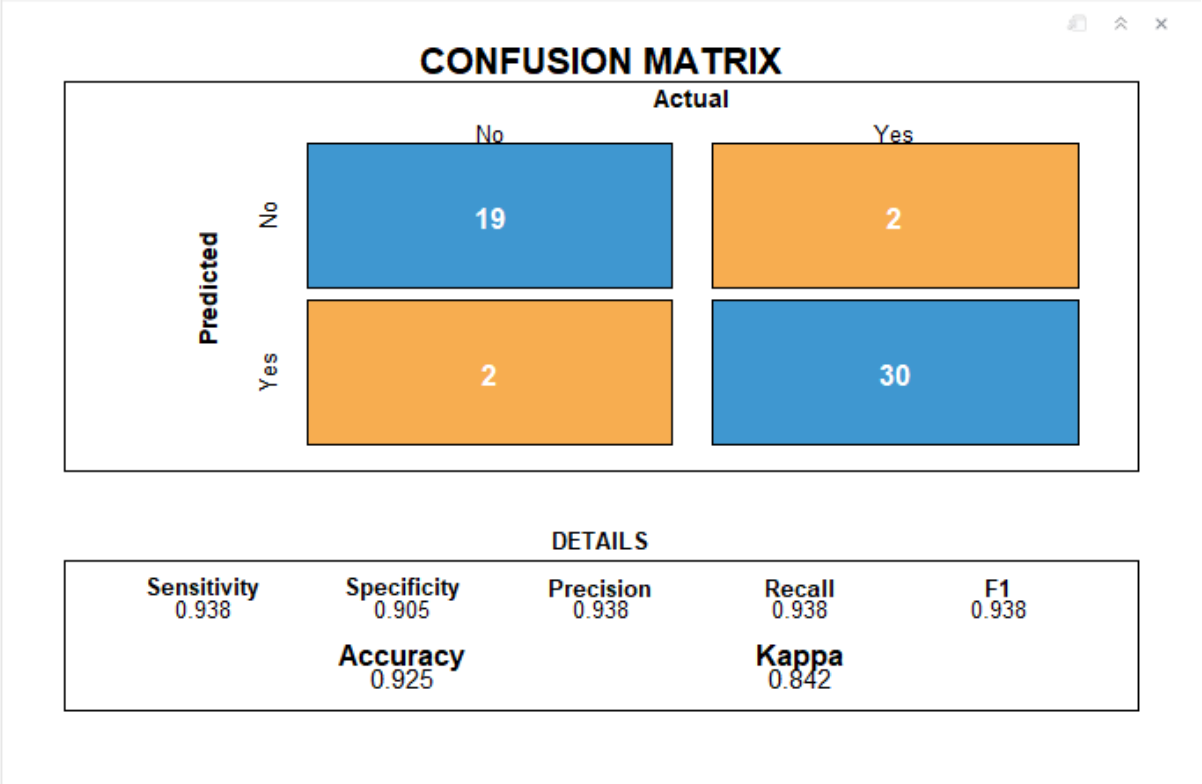
Slika 14: ROC kriva za KNN model (Minkowski metrika distance)

Uradili smo i k-fold validaciju za KNN model koji koristi Minkowski metriku kako bi bili sigurniji u dobijene rezultate.

```
10-fold validacija
Najveća tačnost: 0.8909774 , fold: 10, najveća kappa: 0.7819549 , fold: 10
Najmanja tačnost: 0.8458647 , fold: 9, najmanja kappa: 0.6934967 , fold: 9
Srednja tačnost: 0.8684211, srednja kappa: 0.7365126

5-fold validacija
Najveća tačnost: 0.8815789 , fold: 4, najveća kappa: 0.7627958 , fold: 4
Najmanja tačnost: 0.8327068 , fold: 2, najmanja kappa: 0.6655837 , fold: 2
Srednja tačnost: 0.8541353, srednja kappa: 0.7081528
```

Slika 15: Rezultati k-fold validacije za KNN model (Minkowski metrika distance)

Kreirali smo i vlastitu funkciju ansambl (bagging) na osnovu k-fold bootstrapping funkcije iz laboratorijske vježbe 5. Parametar k predstavlja broj modela koji će biti kreirani. U testnom skupu će biti B elemenata iz prosljeđenog skupa podataka. B smo odredili na isti način kao i u primjeru sa Lab. vježbe 5. Testni skup se kreira na samom početku kako bi bio isti za sve modele, dok se trening skup određuje u for petlji za svaki model posebno. Trening instance se biraju nasumično iz prosljeđenog skupa. Nakon što smo uradili treniranje i testiranje svakog modela, rezultate predikcije smo sačuvali u listu lista_predikcija. Zatim smo za svaku instancu iz testnog skupa provjerili koliko modela je tu instancu klasifikovalo kao Yes, a koliko kao No. Ukoliko je više modela instancu klasifikovalo kao Yes, instanci je dodjeljena labela Yes, u suprotnom joj je dodjeljena labela No. U nastavku je prikazana konfuzijska matrica za model dobijen ansambl tehnikama. Vrijednost parametra k smo postavili na 50 (kao što je urađeno u primjerima sa Lab. vježbe 5 za bagging, AdaBoost i RandomForest metode).



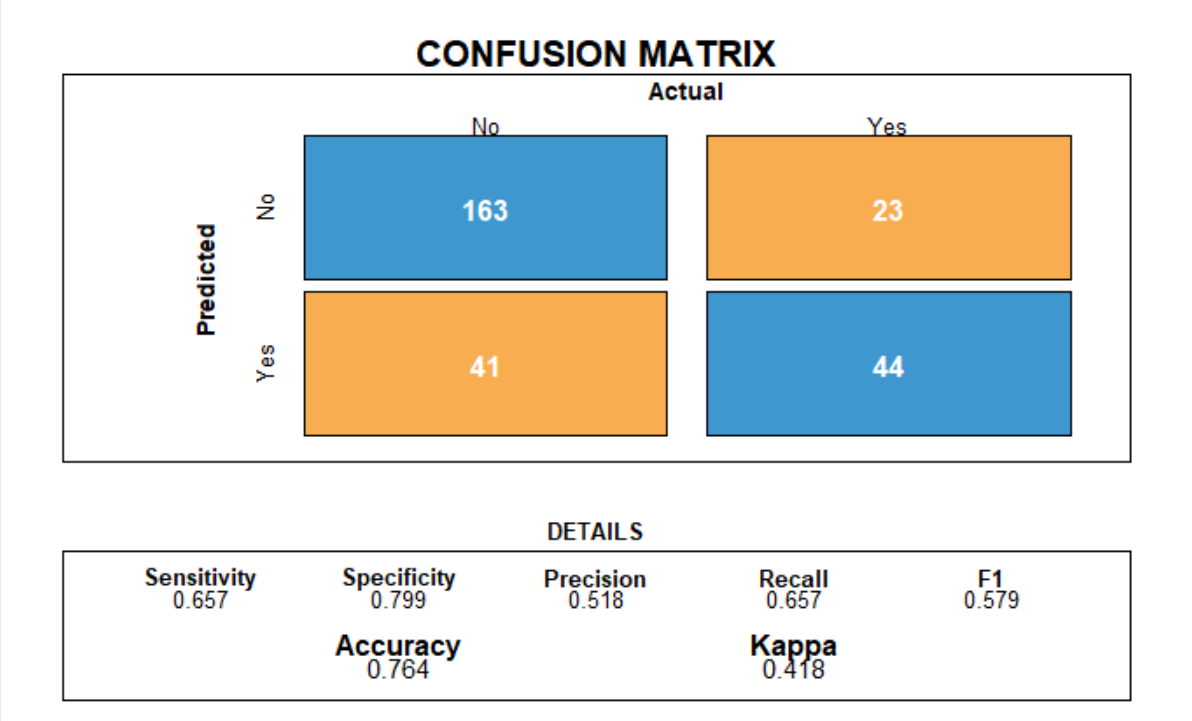
Slika 16: Konfuzijska matrica za KNN model (ansanbl)

1.3 Naivni Bayes

Za kreiranje naivnog Bayesovog modela predikcije koristili smo funkciju naiveBayes iz biblioteke e1071. Ovoj metodi se kao parametar prosljeđuje formula koja određuje labelu klase i trening podskup. Testiranje se radi metodom predict, kao i za sve dosadašnje modele. Prvi parametar je kreirani model, a drugi testni podskup bez labele klase.

Što se tiče pripreme podataka za ovaj model, uradili smo podjelu skupa na trening i testni holdout metodom sa nasumičnim redovima podataka (85% trening, 15% testni), te izvršili faktorizaciju kategoričkih varijabli. Ponovno smo više puta pokretali kreiranje modela za različitu podjelu skupa na trening i testni. U nekim slučajevima smo dobili da je tačnost oko 73%, dok smo u pojedinim pokretanjima dobili i tačnost od 77%.

Dobijeni rezultati prilikom jednog od pokretanja su prikazani u nastavku:



Slika 17: Konfuzijska matrica za naivni Bayesov model)

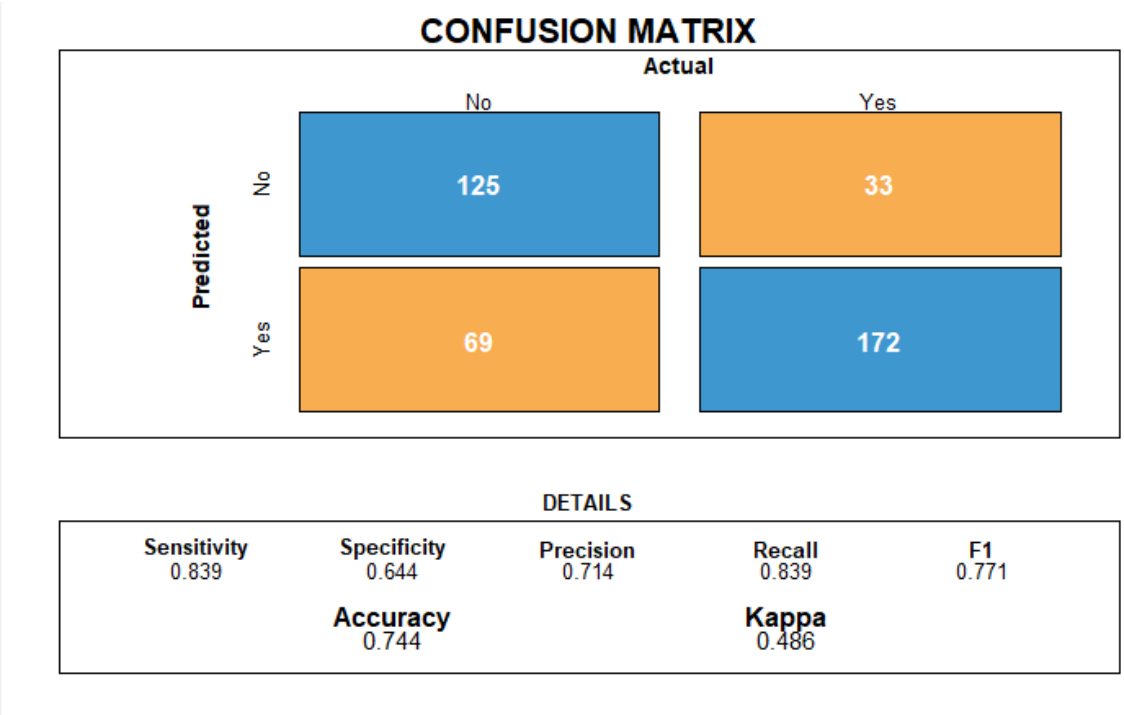
Uradili smo i k-fold validaciju kako bi dobili precinzije vrijednosti:

```
10-fold validacija
Najveća tačnost: 0.7955801 , fold: 10, najveća kappa: 0.5301271 , fold: 7
Najmanja tačnost: 0.7 , fold: 3, najmanja kappa: 0.3013226 , fold: 3
Srednja tačnost: 0.7560681, srednja kappa: 0.4302427

5-fold validacija
Najveća tačnost: 0.7811634 , fold: 2, najveća kappa: 0.4861349 , fold: 2
Najmanja tačnost: 0.7174515 , fold: 1, najmanja kappa: 0.3711268 , fold: 5
Srednja tačnost: 0.7555525, srednja kappa: 0.4296689
```

Slika 18: Rezultati k-fold cross validacije za naivni Bayesov model predickije

S obzirom da smo naivni Bayesov model predikcije primijenili nad nebalansiranim skupom podataka, uradili smo balansiranje oversample metodom, tako da u datasetu bude po 1330 instanci obje klase.

Ponovili smo postupak i dobili sljedeće rezultate:

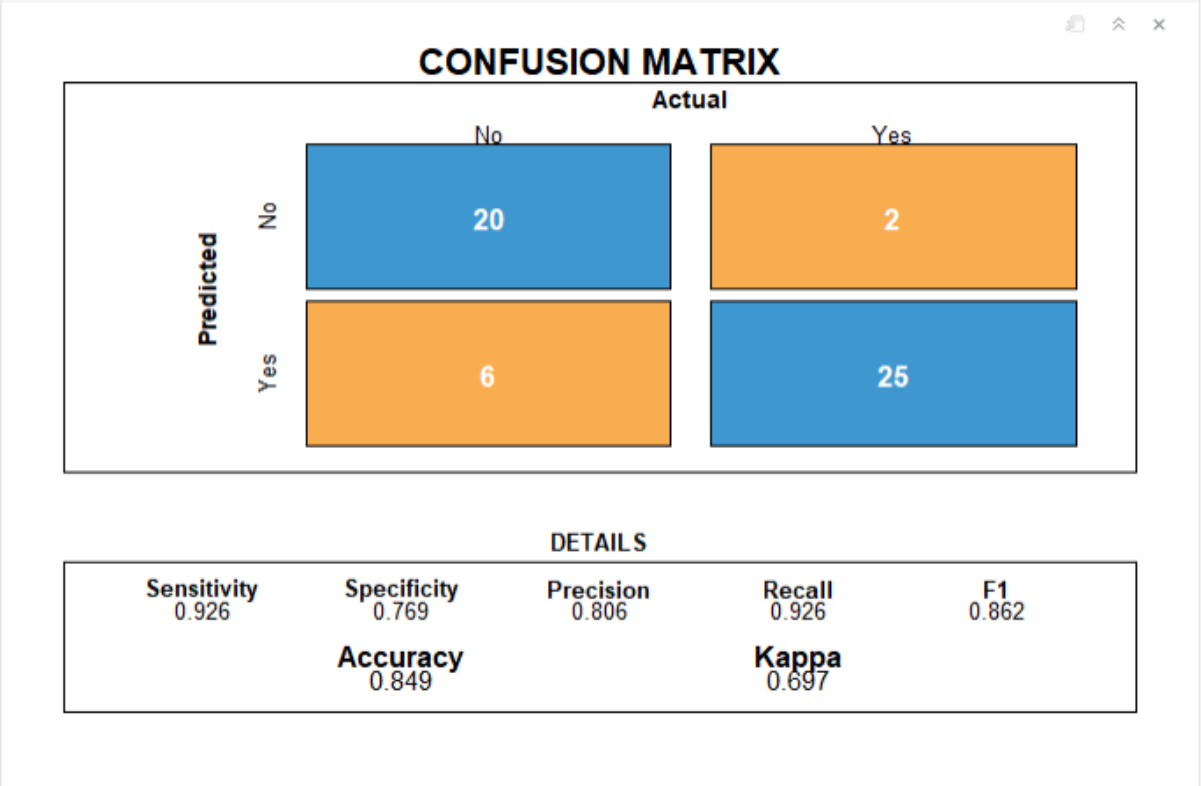


Slika 19: Konfuzijska matrica za naivni Bayesov model nakon balansiranja)

```
10-fold validacija
Najveća tačnost: 0.8045113 , fold: 2, najveća kappa: 0.6086464 , fold: 2
Najmanja tačnost: 0.7142857 , fold: 8, najmanja kappa: 0.4278275 , fold: 8
Srednja tačnost: 0.7443609, srednja kappa: 0.4882267

5-fold validacija
Najveća tačnost: 0.768797 , fold: 4, najveća kappa: 0.5351104 , fold: 4
Najmanja tačnost: 0.7199248 , fold: 3, najmanja kappa: 0.4381822 , fold: 3
Srednja tačnost: 0.7466165, srednja kappa: 0.4926011
```

Slika 20: Rezultati k-fold cross validacije za naivni Bayesov model predickije nakon balansiranja

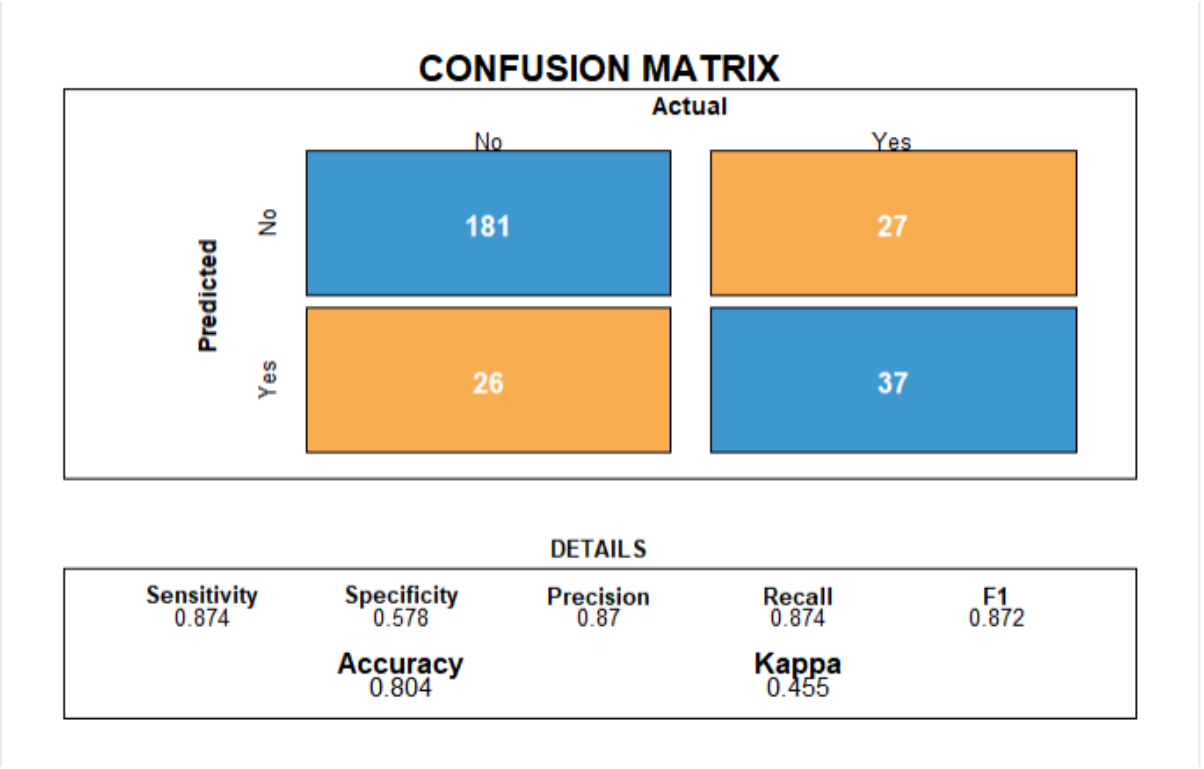
Vidimo da su se tačnost i specifičnost pogoršale nakon balansiranja, dok su se preostali parametri poboljšali. Tačnost i senzitivnost smo uspjeli poboljšati primjenom bagging ansambl tehnike kao i u prvom zadataku. Broj modela je 50.



Slika 21: Naivni Bayesov model (bagging ansambl tehnika)

1.4 Model logističke regresije

U okviru pripreme podataka za ovaj model uradili smo faktorizaciju kategoričkih varijabli i podjelili dataset na trening i testni holdout metodom sa nasumičnim redovima podataka. Model smo kreirali uz pomoć glm funkcije. Vrijednost parametra family smo postavili na binomial(link = "logit"), jer se radi o binarnoj klasifikaciji. Da bi predikcije bile u opsegu od [0, 1] postavili smo vrijednost parametra type na response. Ručno smo pretvorili dobijenu vjerovatnoću u labele klasa. Kada smo primijenili ovaj model na nebalansiranim podacima dobili smo sljedeće rezultate:



Slika 22: Konfuzijska matrica za model logističke regresije bez normalizacije i balansiranja podataka

Dodatno smo uradili k-fold validaciju da bi bili sigurniji u dobijene rezultate:

```
Accuracy: 0.80442810-fold validacija
Najveća tačnost: 0.8277778 , fold: 6, najveća kappa: 0.515625 , fold: 6
Najmanja tačnost: 0.7458564 , fold: 7, najmanja kappa: 0.2382867 , fold: 10
Srednja tačnost: 0.7805249, srednja kappa: 0.3943137

5-fold validacija
Najveća tačnost: 0.8060942 , fold: 4, najveća kappa: 0.4614237 , fold: 4
Najmanja tačnost: 0.7590028 , fold: 2, najmanja kappa: 0.3299416 , fold: 1
Srednja tačnost: 0.7782718, srednja kappa: 0.3924367
```

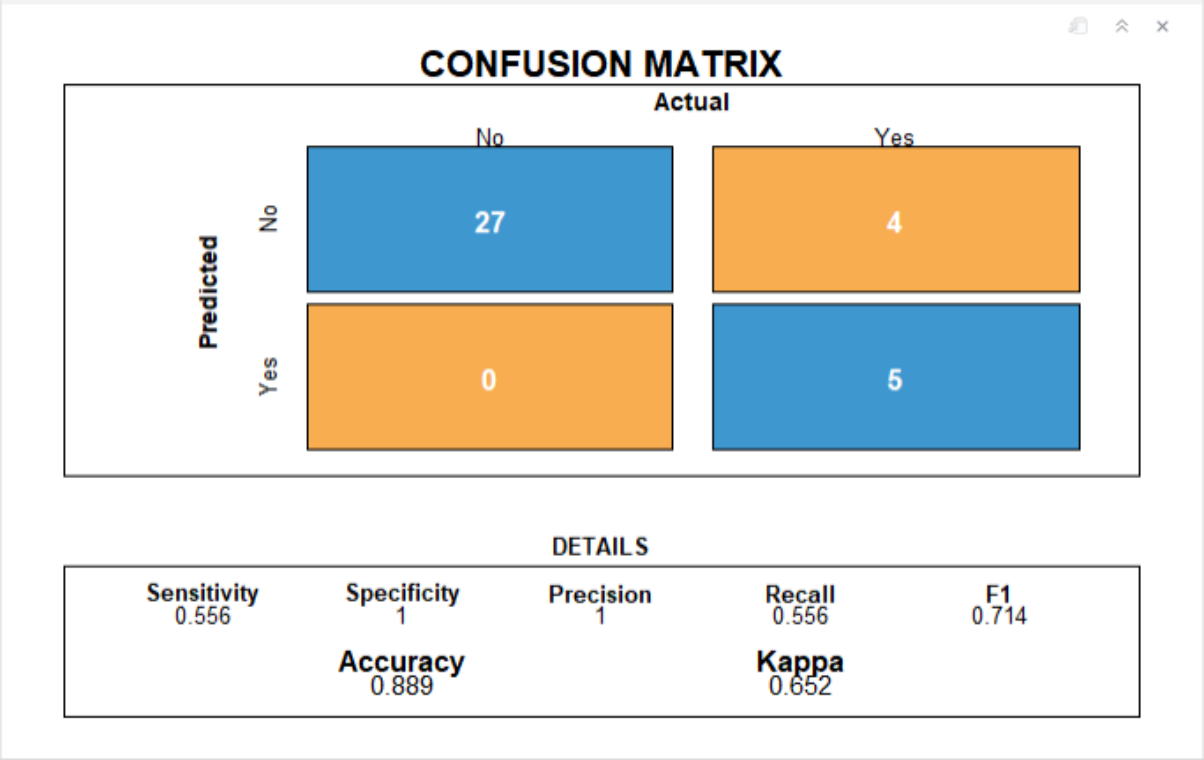
Slika 23: K-fold cross validacija (nebalansirani podaci)

Zatim smo pokušali normalizacijom i balansiranjem dobiti bolje rezultate. Primijenili smo decimalno skaliranje nad svim nenumeričkim podacima. Kappa se povećala za oko 10%, dok se tačnost smanjila za 2-3%.

```
10-fold validacija
Najveća tačnost: 0.7894737 , fold: 9, najveća kappa: 0.5702002 , fold: 9
Najmanja tačnost: 0.7293233 , fold: 3, najmanja kappa: 0.4586466 , fold: 3
Srednja tačnost: 0.7522556, srednja kappa: 0.5040843

5-fold validacija
Najveća tačnost: 0.7744361 , fold: 3, najveća kappa: 0.5465136 , fold: 3
Najmanja tačnost: 0.7293233 , fold: 4, najmanja kappa: 0.4595403 , fold: 4
Srednja tačnost: 0.7469925, srednja kappa: 0.4940627
```

Slika 24: K-fold cross validacija (balansirani podaci)

Rezultate smo pokušali unaprijediti korištenjem bagging ansambl metode, kao i za prethodne modele. Broj modela je n = 50.

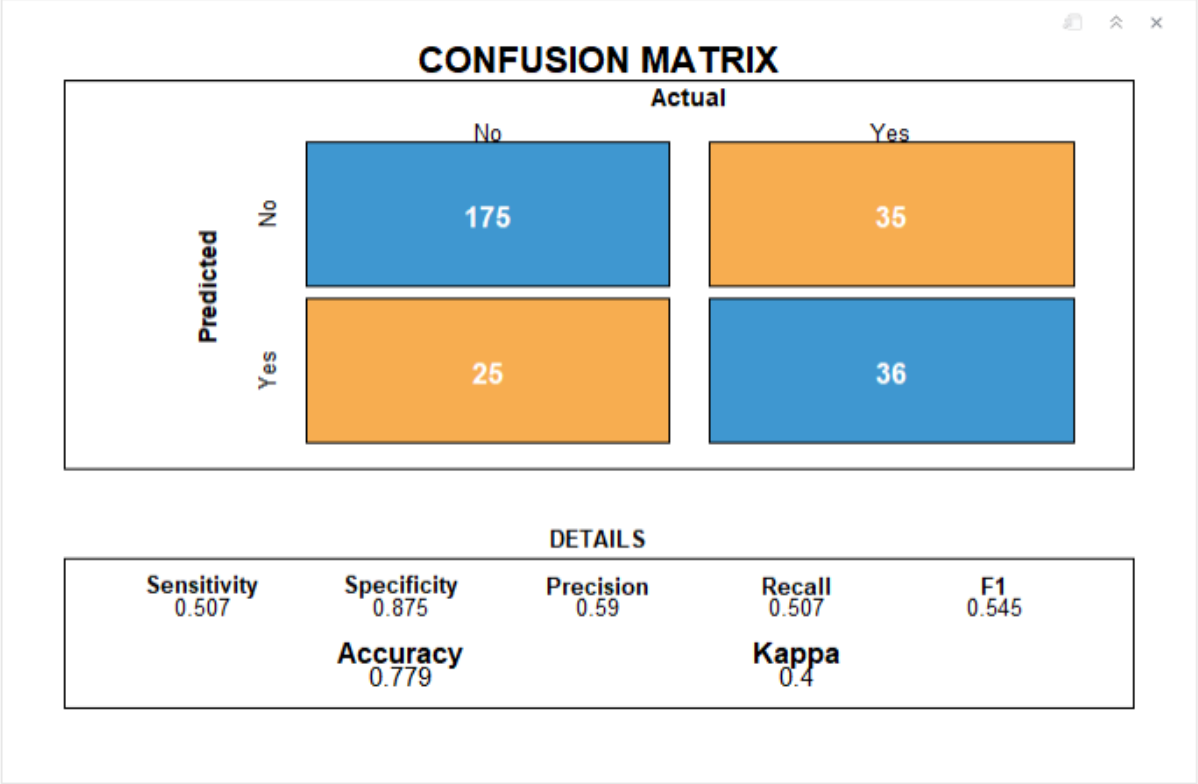


Slika 25: Model logističke regresije bagging ansambl tehnika

1.5 SVM model klasifikacije

Za podjelu dataseta na setove za treniranje i testiranje koristili smo ponovo holdout metodu sa nasumičnim redovima podataka tako da se 85% podataka nalazi u trening, a 15% u testnom skupu.

Za kreiranje SVM algoritma za klasifikaciju koristili smo funkciju svm iz biblioteke e1701. Korištena je linearna kernel funkcija. Kada smo primijenili algoritam nad podacima (bez balansiranja i normalizacije) dobili smo sljedeće rezultate (kod smo pokrenuli više puta, u jednom slučaju smo dobili i tačnost od 80%, dok smo najčešće oko 75%):



Slika 26: Konfuzijska matrica za SVM algoritam bez normalizacije i balansiranja podataka

Tačnost smo pokušali povećati normalizacijom numeričkih podataka. Provjerili smo kakve rezultate dobijemo sa z-score, decimalnim skaliranjem i min-max normalizacijom. Pošto smo u različitim pokretanjima dobijali da različiti modeli odnosno modeli primjenjeni nad podacima nad kojima je urađena različita normalizacija nad numeričkim podacima, odlučili smo da uradimo k-fold cross validaciju.

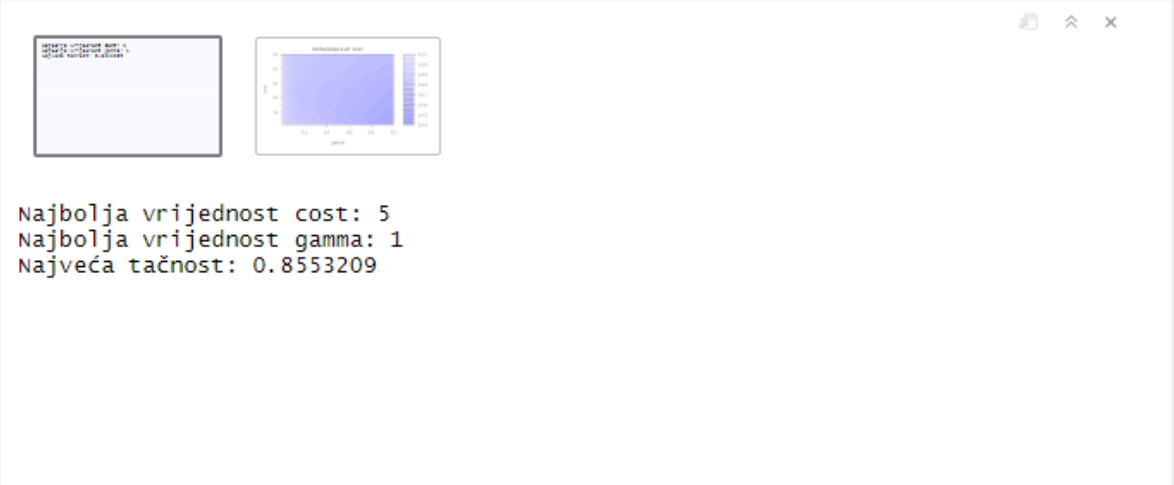
```
Decimalno skaliranje:
10-fold validacija
Najveća tačnost: 0.8176796 , fold: 10, najveća kappa: 0.4615869 , fold: 8
Najmanja tačnost: 0.7444444 , fold: 2, najmanja kappa: 0.32616 , fold: 4
Srednja tačnost: 0.7799171, srednja kappa: 0.395541

5-fold validacija
Najveća tačnost: 0.7916667 , fold: 3, najveća kappa: 0.4471744 , fold: 3
Najmanja tačnost: 0.7534626 , fold: 5, najmanja kappa: 0.3158948 , fold: 5
Srednja tačnost: 0.7749538, srednja kappa: 0.3810112

Min-max normalizacija:
10-fold validacija
Najveća tačnost: 0.8342541 , fold: 10, najveća kappa: 0.5207414 , fold: 10
Najmanja tačnost: 0.7222222 , fold: 8, najmanja kappa: 0.270428 , fold: 8
Srednja tačnost: 0.7726857, srednja kappa: 0.3756392

5-fold validacija
Najveća tačnost: 0.8033241 , fold: 5, najveća kappa: 0.4760308 , fold: 5
Najmanja tačnost: 0.7506925 , fold: 4, najmanja kappa: 0.295136 , fold: 4
Srednja tačnost: 0.7766005, srednja kappa: 0.3853854

Z-score normalizacija:
10-fold validacija
Najveća tačnost: 0.8111111 , fold: 3, najveća kappa: 0.4572708 , fold: 8
Najmanja tačnost: 0.7292818 , fold: 7, najmanja kappa: 0.2723767 , fold: 7
Srednja tačnost: 0.7749969, srednja kappa: 0.3834906

5-fold validacija
Najveća tačnost: 0.8033241 , fold: 4, najveća kappa: 0.4444107 , fold: 4
Najmanja tačnost: 0.7451524 , fold: 1, najmanja kappa: 0.3401912 , fold: 2
Srednja tačnost: 0.7760449, srednja kappa: 0.3822673
```

Slika 27: Rezultati k-fold cross validacije za različite normalizacije

U većini slučajeva najveća tačnost se dobije za decimalno skaliranje, koja je svakako bolja nego bez normalizacije, pa smo to primijenili i nad originalnim podacima. Također, smo balansirali dataset (po 1330 instanci obje klase).

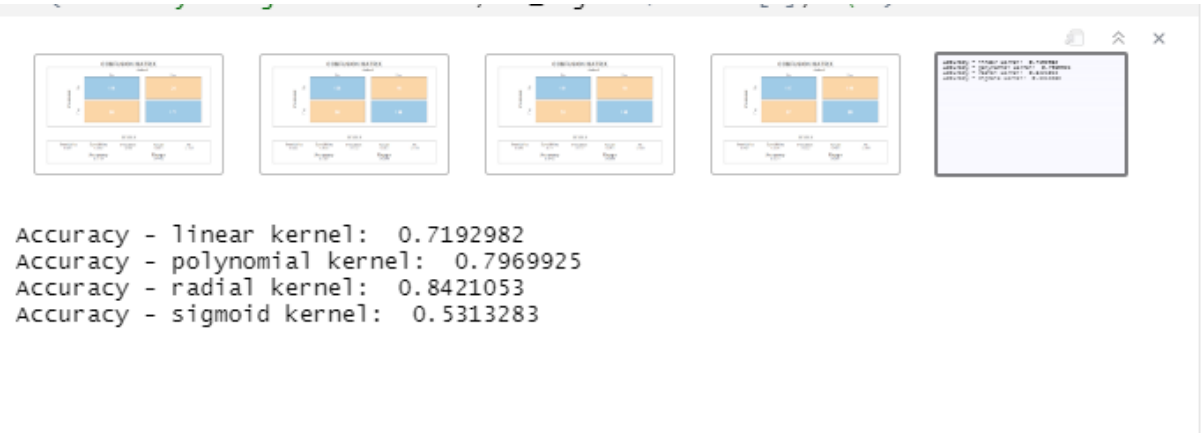
Kako bismo uradili tuning parametara, sve nenumeričke varijable smo pretvorili u numeričke pomoću as.numeric() funkcije.

Za vrijednost parametra cost smo uzeli sljedeće vrijednosti 1, 5, 10, 25, 50, a za gamma 0.001, 0.01,0.1, 1. Nakon tuninga dobili smo da su najbolje vrijednost za cost je 5 i gamma = 1.



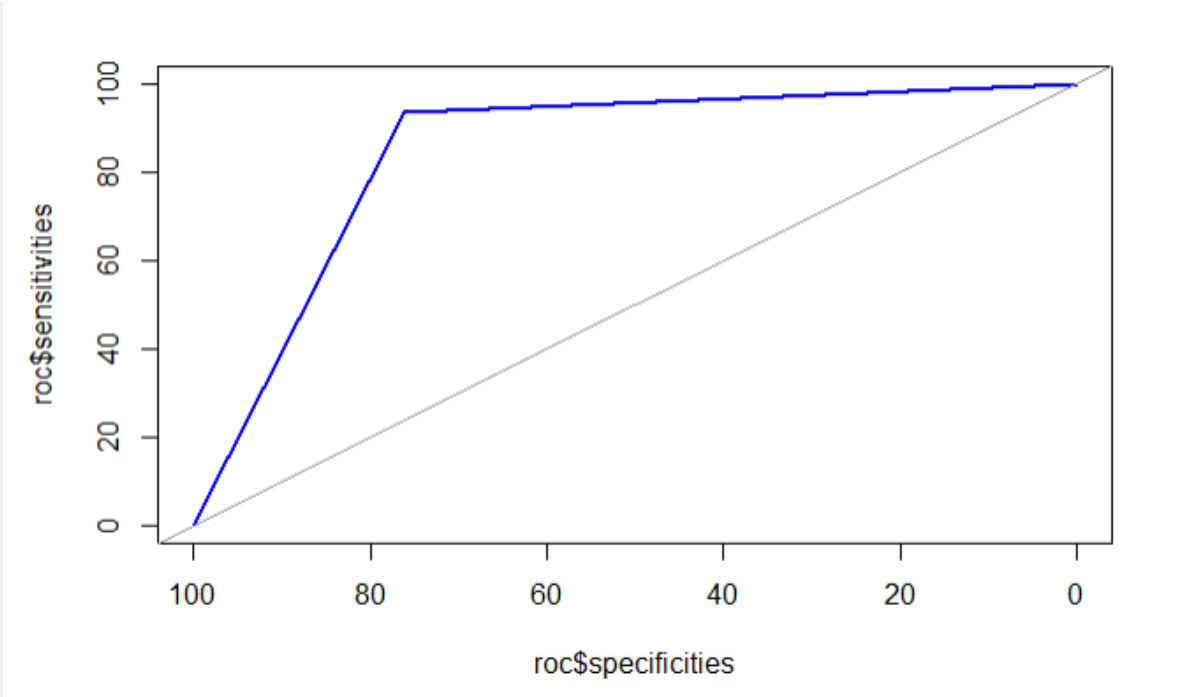
Slika 28: Tuning parametara za SVM algoritam

Zatim smo kreirali SVM klasifikator sa različitim vrijednostima kernel parametra. Za sva 4 modela (linearni, polinomijalni, radijalni i sigmoid) smo naveli iste vrijednosti za cost i gamma parametar i to one vrijednosti koje smo dobili kao najbolje kada smo uradili tuning parametara. Rezultati su prikazni ispod:



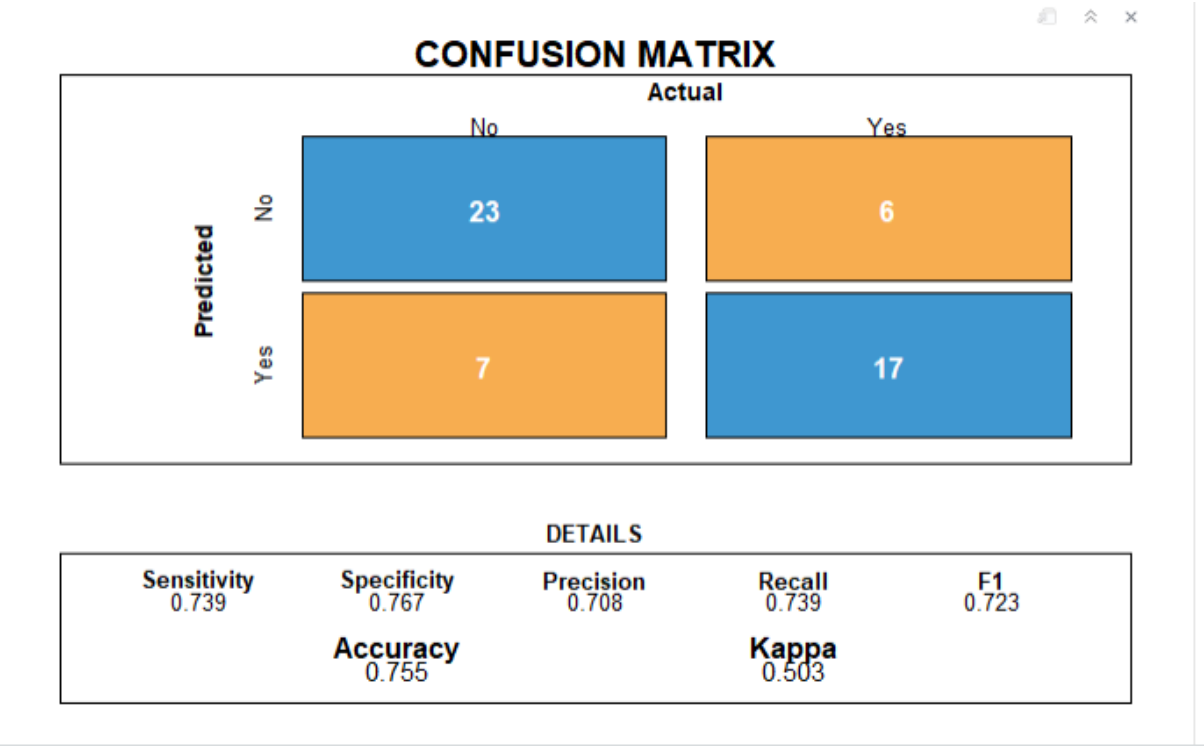
Slika 29: SVM klasifikatori sa različitim kernel funkcijama

Vidimo da se najveća tačnost dobila za SVM klasifikator koji koristi radijalnu kernel funkciju.

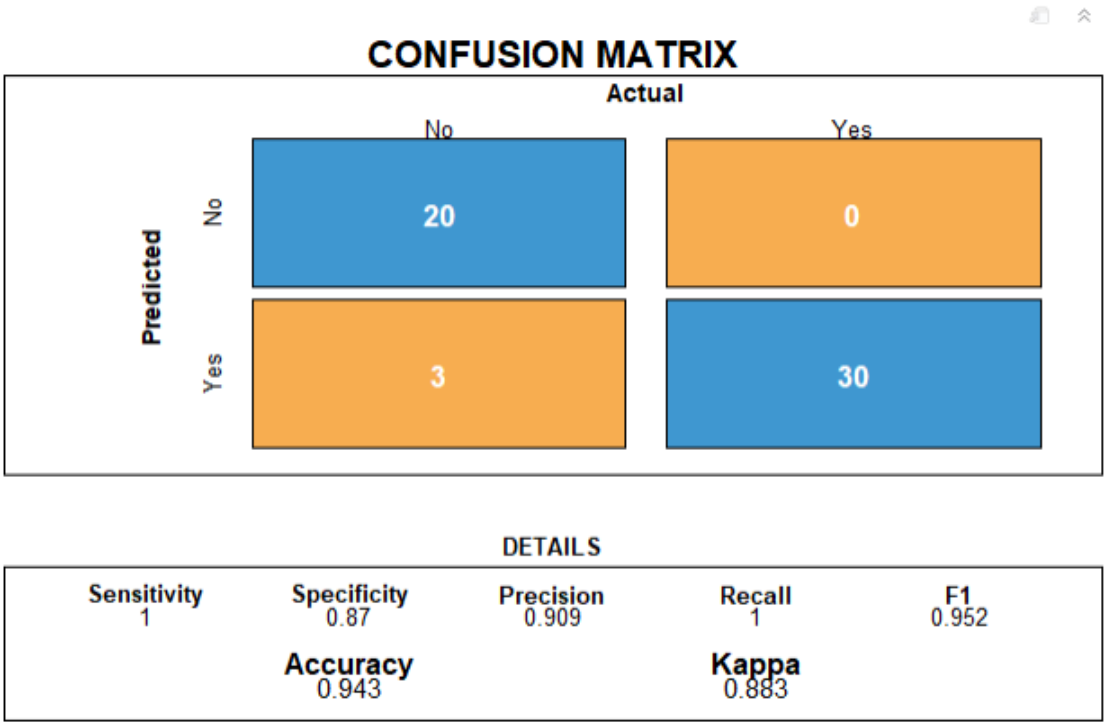


Slika 30: ROC kriva za SVM klasifikator sa radialnim kernelom

Bagging metodu smo iskoristili kako bismo dodatno poboljšali rezultate. Metodu smo primijenili nad SVM modelom sa lineranim i radijalnim kernelom. Rezultati su prikazani u nastavku.



Slika 31: Bagging metoda za SVM sa linearnim kernelom



Slika 32: Bagging metoda za SVM sa radijalnim kernelom

1.6 Neuralne mreže

Učitani dataset je već preprocesiran i ne posjeduje NA vrijednosti. Neuralne mreže ne mogu raditi sa kategoričkim atributima pa ih je potrebno pretvoriti u numeričke. Labela klase mora biti označena isključivo sa 0 ili 1, a vrijednosti ostalih atributa potrebno je svesti na opseg od 0 do 1, odnosno izvršiti normalizaciju. Nakon toga vrši se podjela dataseta na trening i testni skup u omjeru 80% - 20%. Izuzevši učitavanje i faktorizaciju, navedeno je prikazano u narednom isječku koda:

```
podaci$Churn[podaci$Churn == "Yes"] <- 1
podaci$Churn[podaci$Churn == "No"] <- 0
podaci$Churn <- as.numeric(podaci$Churn)
#normalizacija svih atributa osim labele klase u opseg 0-1
library(class)
library(caret)

preObj <- preProcess(subset(podaci, select = -c(Churn)),
method=c("range"), rangebounds = c(0, 1))
class <- podaci$Churn
podaci <- predict(preObj, subset(podaci, select = -c(Churn)))
podaci$Churn <- class

#podjela na trening i testne podatke
#korisitimo seed 5*
set.seed(5)
rows <- sample(nrow(podaci))
podaci <- podaci[rows, ]
```

```
end <- length(podaci$Churn)
n <- as.integer(0.8 * end)
podaci_train <- podaci[1 : n, ]
podaci_test <- podaci[(n + 1) : end, ]

podaci_train_pocetni <- podaci_train
podaci_test_pocetni <- podaci_train
```

Modeli prikazani u nastavku bit će izgrađeni u tri varijante:

1. Nad trening podacima bez sampleinga (podaci_train)
2. Nad oversampleovanim trening podacima (oversample_train)
3. Nad undersampleovanim trening podacima (undersample_train)

Oversampleovanje je urađeno metodom "both", odnosno neke instance su se umnožile, a neke obrisale:

```
library(ROSE)
oversample_train <- ovun.sample(Churn ~ ., data = podaci_train, method = "both",
N = 1610)$data
```

Omjer instanci prije i nakon oversampleinga:

```
Broj instanci koje imaju Yes vrijednost u trening skupu:  474
Broj instanci koje imaju No vrijednost u trening skupu:  1330Loaded ROSE 0.0-4


Broj instanci koje imaju Yes vrijednost u trening skupu nakon oversamplinga:  827
Broj instanci koje imaju No vrijednost u trening skupu nakon oversamplinga:  783
```

Slika 33: Omjer instanci prije i nakon oversamplinga

Analogno programskom isječku iznad, koristeći metodu "under" dobijaju se sljedeći rezultati:

```
Broj instanci koje imaju Yes vrijednost u trening skupu:  474
Broj instanci koje imaju No vrijednost u trening skupu:  1330
Broj instanci koje imaju Yes vrijednost u trening skupu nakon undersamplinga:  393
Broj instanci koje imaju No vrijednost u trening skupu nakon undersamplinga:  457
```

Slika 34: Omjer instanci prije i nakon undersamplinga

Nazivi modela nad oversampleovanim podacima imaju prefiks "O", a nad undersapleovanim podacima "U".

*** S obzirom da generalno ima mnogo modela nećemo vršiti prikaz modela treniranih nad sampleovanim trening skupovima. U podsekciji "Rezultati" osvrnut ćemo se na sveukupne rezultate svih modela. Također, tuning hiperparametara kreiranih neuralnih mreža i korištenje ensamble metoda, kao i k-fold validacija nije bilo moguće izvršiti jer je pri svakom pokušaju algoritam divergirao. ***

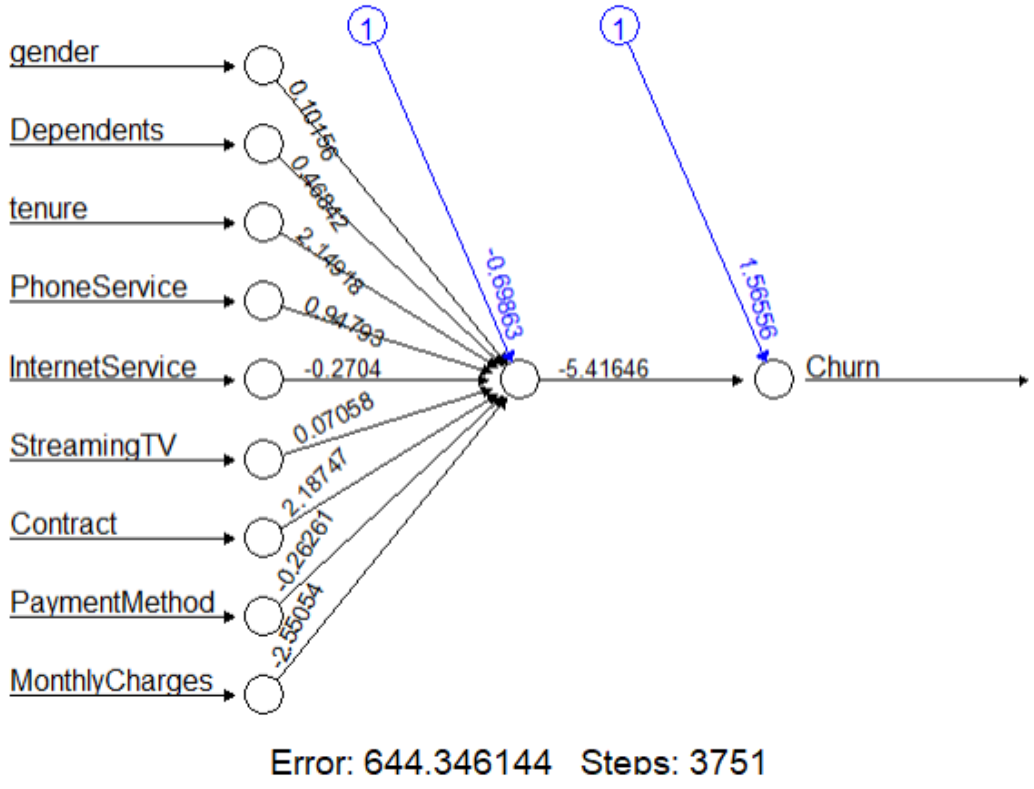
1.6.1 Jednostavni perceptron sa jednim slojem i jednim neuronom

Parametar linear.output postavljen je na vrijednost FALSE, s obzirom da je u pitanju binarna klasifikacija. Za funkciju gubitka koristi se cross-entropy i to vrijedi za sve modele u nastavku.

Jednostavni perceptron definisan je sljedećim kodom:

```
#Jednostavni perceptron sa jednim slojem i jednim neuronom
#za seed = 5 dobije se 644,346 i 3751 koraka

#treniranje u jednoj epohi
set.seed(5)
podaci_train1 <- podaci_train_pocetni
model_one <- neuralnet(formula = Churn ~ .,
 data = podaci_train1,
 linear.output = FALSE,
 err.fct = "ce",
 hidden = 1,
 lifesign = "full")
plot(model_one)
```

Dobijamo sljedeći prikaz neuralne mreže:



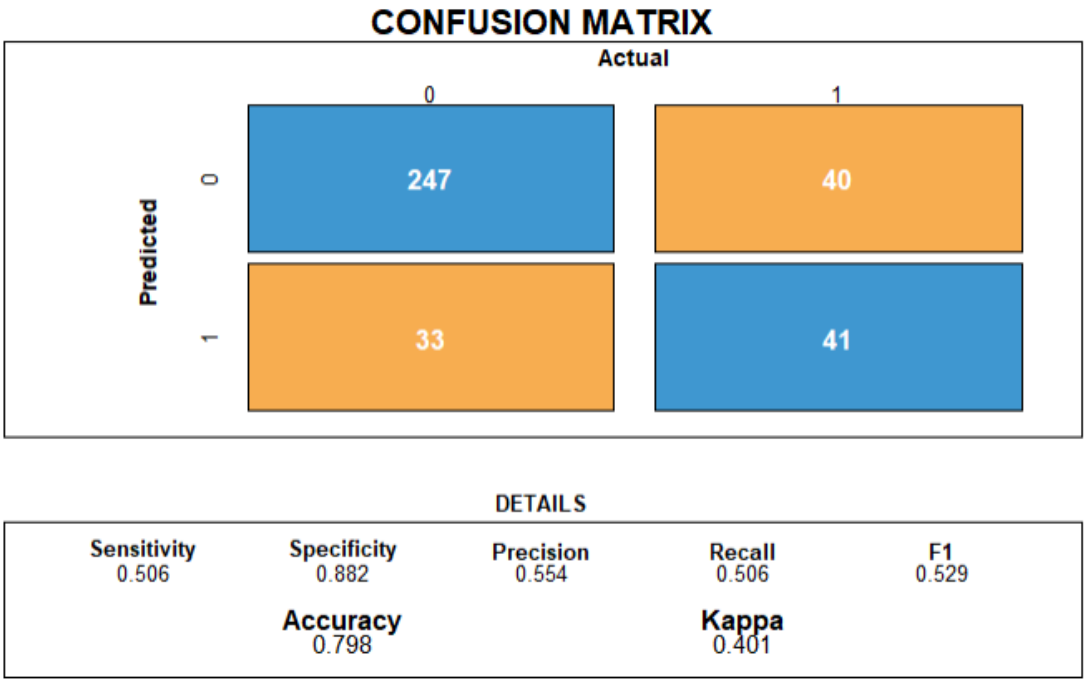
Slika 35: Jednostavni perceptron sa jednim slojem i jednim neuronom

Tačnost nad trening skupom podataka služi za usporedbu performansi različitih neuralnih mreža, ali ne daje nikakve detaljnije informacije o generalizaciji samih modela. Prema tome, visoka tačnost ne znači da će nad

testnim ili realnim podacima neuralna mreža imati očekivanu tačnost.

Jedan neuron, jedna epoha - Train accuracy: 0.7650728

Konfuzijska matrica:



Slika 36: Jednostavni perceptron sa jednim slojem i jednim neuronom

1.6.2 Jednoslojna mreža trenirana u više epoha

Napravljena su dva modela - jedan je treniran u 5, a drugi u 7 epoha. Model sa 5 epoha definisan je na sljedeći način:

```
#treniranje u 5 epoha sa jednim slojem i jednim neuronom
#644,345   4027
set.seed(5)
podaci_train2 <- podaci_test_pocetni
model_one_rep_5 <- neuralnet(formula = Churn ~ .,
 data = podaci_train2,
 linear.output = FALSE,
 err.fct = "ce",
 hidden = 1,
 rep = 5,
 lifesign = "minimal")
plot(model_one_rep_5, rep = "best")
```

Prikaz svih epoha treniranja:

```
hidden: 1    thresh: 0.01    rep: 1/5    steps:    3751 error: 644.34614        time: 2.49
secs
hidden: 1    thresh: 0.01    rep: 2/5    steps:    4193 error: 644.34558        time: 2.1 secs
hidden: 1    thresh: 0.01    rep: 3/5    steps:    4170 error: 644.34553        time: 2.1 secs
hidden: 1    thresh: 0.01    rep: 4/5    steps:    4221 error: 644.34565        time: 2.1 secs
hidden: 1    thresh: 0.01    rep: 5/5    steps:    4027 error: 644.34553        time: 2.05
secs
```

Slika 37: Proces treniranja jednoslojne neuralne mreže sa jednim neuronom u 5 epoha

Najmanja greška i najmanje koraka je u epohi 5, stoga se postavlja pitanje da li se greška može još smanjiti u više epoha (npr. 7):

```
hidden: 1    thresh: 0.01    rep: 1/7    steps:    3751 error: 644.34614        time:
2.33 secs
hidden: 1    thresh: 0.01    rep: 2/7    steps:    4193 error: 644.34558        time:
2.34 secs
hidden: 1    thresh: 0.01    rep: 3/7    steps:    4170 error: 644.34553        time:
2.35 secs
hidden: 1    thresh: 0.01    rep: 4/7    steps:    4221 error: 644.34565        time:
2.17 secs
hidden: 1    thresh: 0.01    rep: 5/7    steps:    4027 error: 644.34553        time:
2.26 secs
hidden: 1    thresh: 0.01    rep: 6/7    steps:    3603 error: 644.3457         time:
1.83 secs
hidden: 1    thresh: 0.01    rep: 7/7    steps:    4351 error: 644.34559        time:
2.23 secs
```

Slika 38: Proces treniranja jednoslojne neuralne mreže sa jednim neuronom u 7 epoha

Vidimo da je najmanja greška ostala u petoj epohi, što znači da neuralnu mrežu sa ovakvom arhitekturom nad našim setom podataka ne treba trenirati više od 5 epoha jer se greška povećava.

Za tačnost nad trening podacima dobijamo sljedeće rezultate:

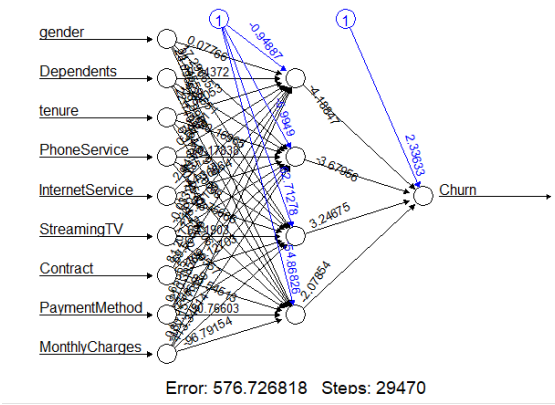
Jedan neuron, pet epoha - Train accuracy: 0.7657658 Jedan neuron, sedam epoha - Train accuracy: 0.7657658

Tačnost u odnosu na jednu epohu razlikuje se tek u četvrtoj decimali, dakle, poboljšanje je zanemarivo. Konfuzijska matrica koja se dobije je identična matrici za jednu epohu.

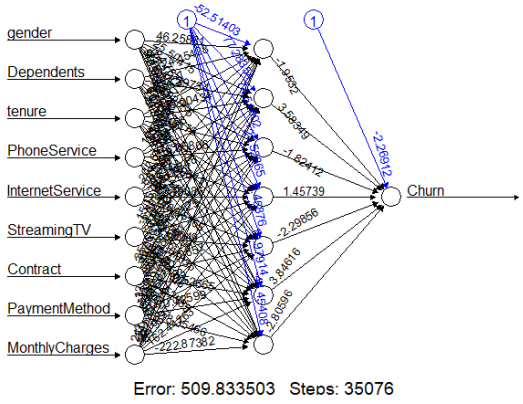
1.6.3 Jednoslojne mreže sa više neurona

Definisali smo četiri mreže sa jednim slojem koje broje 4, 7, 8 i 12 neurona.

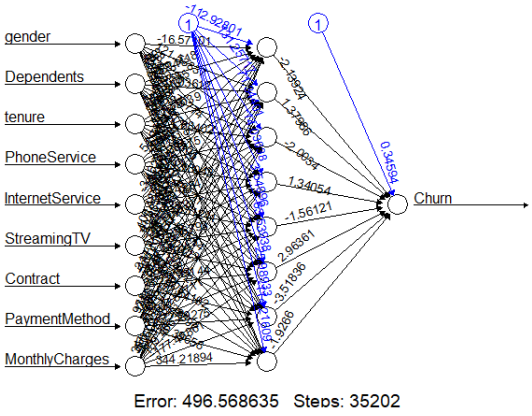
```r
#Perceptron sa jednim slojem i vi e neurona
#12 neurona i 1 epoha
#434,48112
set.seed(5)
model_multiple_12 <- neuralnet(formula = Churn ~ .,
 data = podaci_train,
 linear.output = FALSE,
 err.fct = "ce",
 hidden = 12,
 stepmax = 200000,
 lifesign = "minimal")
plot(model_multiple_12, rep = "best")
```



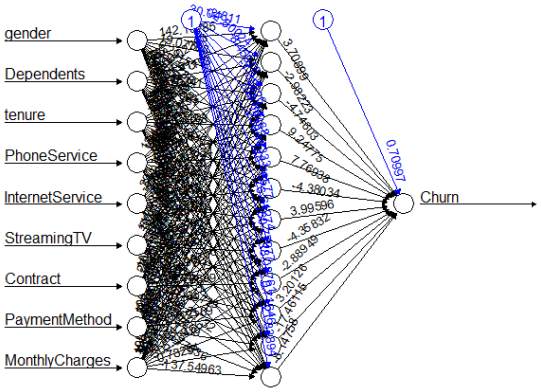
(a) 4 neurona



(b) 7 neurona



(c) 8 neurona



(d) 12 neurona

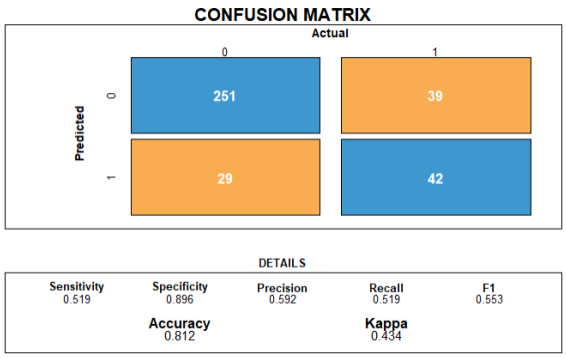
Slika 39: Prikaz jednoslojnih neuralnih mreža

Na slikama iznad možemo uočiti kako se povećanjem broja neurona u sloju greška smanjuje, ali se zato broj koraka odnosno vrijeme treniranja modela povećava. U konačnici, za 12 neurona broj koraka greška i vrijeme su: steps: 48112 error: 434.8827 time: 1.56 mins. Greška je znatno manja nego kod jednoslojne mreže sa jednim neuronom.

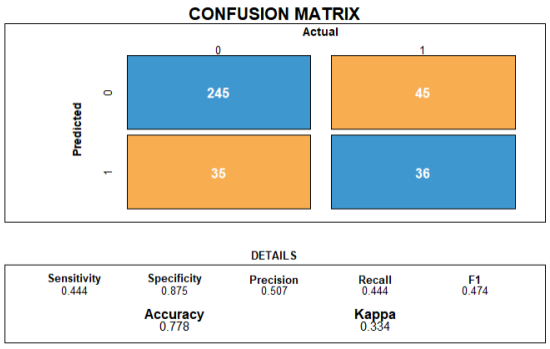
Za tačnost nad trening podacima dobijamo sljedeće rezultate:

Cetiri neurona, jedna epoha - Train accuracy: 0.7934858
Sedam neurona, jedna epoha - Train accuracy: 0.8364518
Osam neurona, jedna epoha - Train accuracy: 0.8371448
Dvanaest neurona, jedna epoha - Train accuracy: 0.8454608

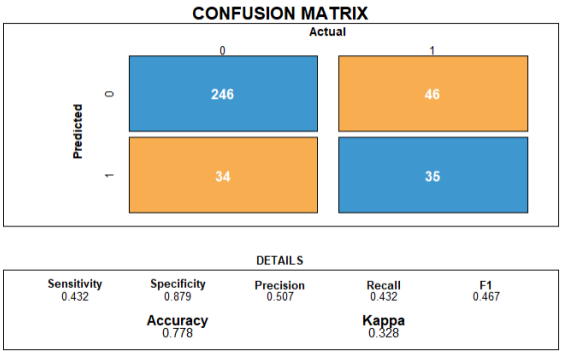
Povećanje tačnosti može biti jedan od indikatora da je najbolja mreža sa 12 neurona.
Dobijamo sljedeće konfuzijske matrice:



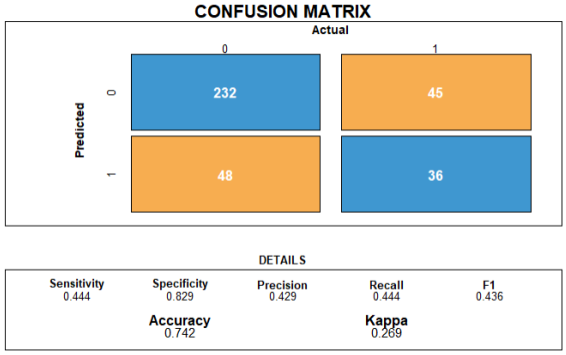
(a) 4 neurona



(b) 7 neurona



(c) 8 neurona



(d) 12 neurona

Slika 40: Konfuzijske matrice jednoslojnih neuralnih mreža

Možemo zaključiti da nad testnim skupom podataka najbolju tačnost ima jednoslojna mreža sa 4 neurona.

Napravljena su i dva modela koja se treniraju u više epoha: model sa 4 neurona i 3 epohe i model sa 12 neurona i 4 epohe. Tačnost nad skupom za treniranje kod modela sa 12 neurona i 4 epohe (0.8482328) u odnosu na model s jednom epohom (0.8454608) razlikuje se u trećoj decimali.

Rezultati su sljedeći:

Cetiri neurona, jedna epoha - Test accuracy: 0.8116343
Cetiri neurona, tri epohe - Test accuracy: 0.8116343
Dvanaest neurona, jedna epoha - Test accuracy: 0.7423823
Dvanaest neurona, cetiri epohe - Test accuracy: 0.7423823

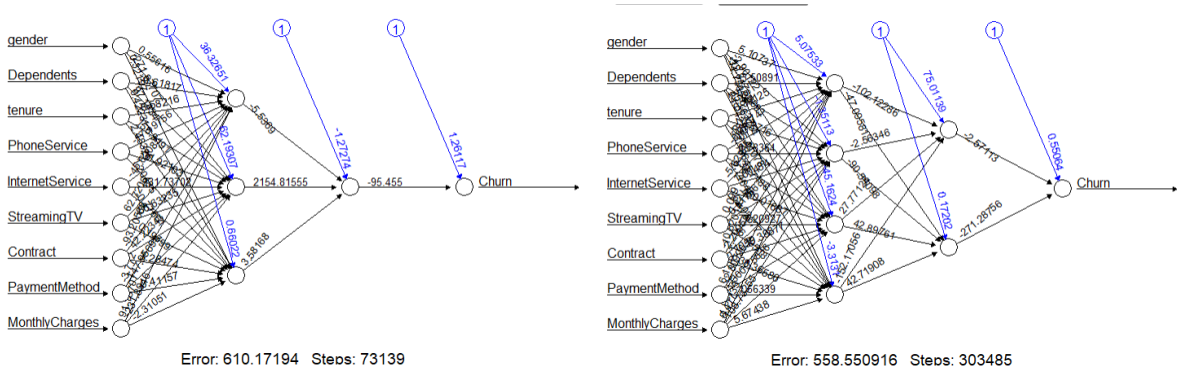
Odnosno, modeli sa jednom i više epoha u slučaju našeg dataseta na isti način klasificiraju testne instance, što nas upućuje na to da je bilo dovoljno model trenirati u jednoj epohi.

1.6.4 Višeslojne mreže

Definisane su dvije višeslojne mreže: prva mreža ima dva skrivena sloja sa po tri i jednim neuronom. Druga mreža ima također dva skrivena sloja sa četiri i dva neurona. Ovakve mreže definisane su na sljedeći način:

```
#vi eslojne neuralne mre e
# dva sloja sa po 4 i 2 neurona i jednom epohom
#558.55092     303485
set.seed(5)
model_multiple_layers_4_2 <- neuralnet(formula = Churn ~ .,
 data = podaci_train,
 linear.output = FALSE,
 err.fct = "ce",
 hidden = c(4,2),
 stepmax = 700000,
 lifesign = "minimal")
```

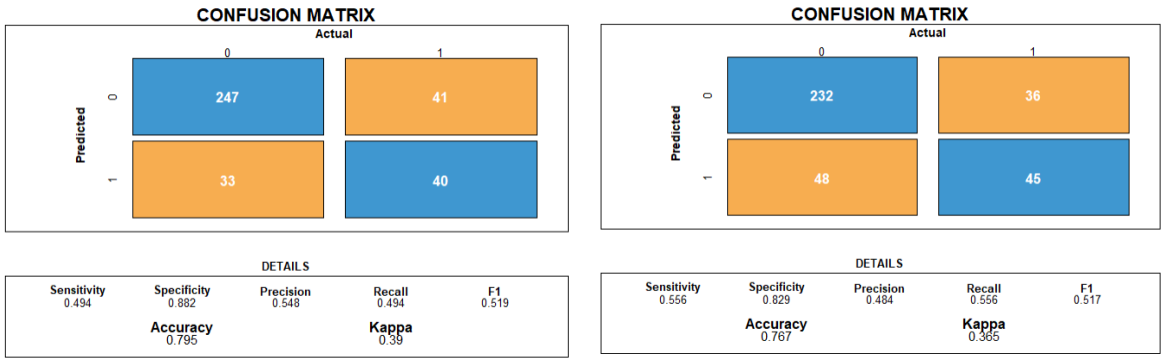
Ovdje je parametar stepmax postavljen na 700000 jer bi u suprotnom došlo do prekoračenja maksimalnog broja koraka koji iznosi 100000 ukoliko se eksplicitno ne navede.



(a) 3 i 1 neuron

(b) 4 i 2 neurona

Slika 41: Prikaz višeslojnih neuralnih mreža



(a) 3 i 1 neuron

(b) 4 i 2 neurona

Slika 42: Konfuzijske matrice višeslojnih neuralnih mreža

1.6.5 Rezultati

Rezultati tačnosti nad trening skupom (20%) od dataseta prikazani su u nastavku:

Ukupni rezultati:
Jedan neuron, jedna epoha -
Train accuracy: 0.7650728
Test accuracy: 0.7977839

Jedan neuron, pet epoha -
Train accuracy: 0.7657658
Test accuracy: 0.7977839

Jedan neuron, sedam epoha -
Train accuracy: 0.7657658
Test accuracy: 0.7977839

Cetiri neurona, jedna epoha -
Train accuracy: 0.7934858
Test accuracy: 0.8116343

Cetiri neurona, tri epohe -
Train accuracy: 0.8101178
Test accuracy: 0.8116343

Dva sloja (4,2) -
Train accuracy: 0.8121968
Test accuracy: 0.767313

Dva sloja (3,1) -
Train accuracy: 0.7713098
Test accuracy: 0.7950139

Sedam neurona, jedna epoha -
Train accuracy: 0.8364518
Test accuracy: 0.7783934

Osam neurona, jedna epoha -
Train accuracy: 0.8371448
Test accuracy: 0.7783934

Dvanaest neurona, jedna epoha -
Train accuracy: 0.8454608
Test accuracy: 0.7423823

Dvanaest neurona, cetiri epohe -
Train accuracy: 0.8482328
Test accuracy: 0.7423823

Nakon oversampleinga:

Jedan neuron, jedna epoha -
Train accuracy: 0.7677019
Test accuracy: 0.6786704

Jedan neuron, pet epoha -
Train accuracy: 0.7677019
Test accuracy: 0.6786704

Cetiri neurona, jedna epoha -
Train accuracy: 0.810559
Test accuracy: 0.6952909

Dvanaest neurona, jedna epoha -
Train accuracy: 0.8987578
Test accuracy: 0.6842105

Nakon undersampleinga:

Jedan neuron, jedna epoha -
Train accuracy: 0.7576471
Test accuracy: 0.7229917

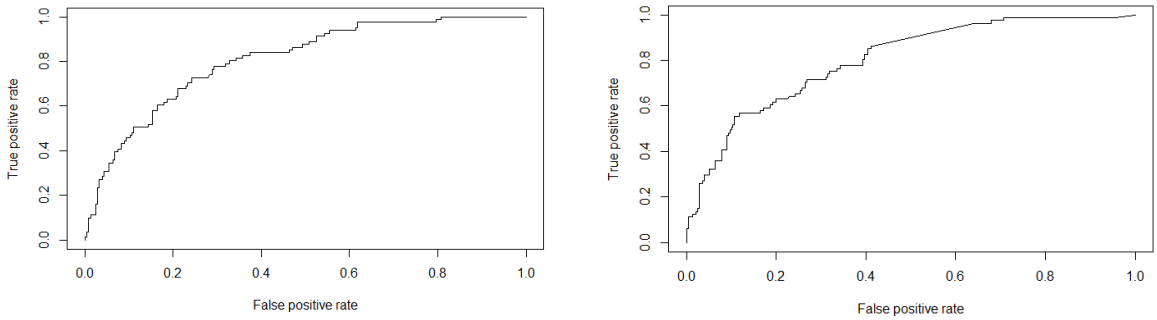
Jedan neuron, pet epoha -
Train accuracy: 0.7552941
Test accuracy: 0.7229917

Cetiri neurona, jedna epoha -
Train accuracy: 0.8176471
Test accuracy: 0.7119114

Dvanaest neurona, jedna epoha -
Train accuracy: 0.9023529
Test accuracy: 0.700831

U našem slučaju balansiranje podataka nije dalo bolje rezultate. Mogući razlozi za to su loša podjela trening i testnog skupa ili to što je balansiranje urađeno samo nad trening skupom. Željeli smo da naš **testni skup bude isti za svaki model kako bismo ih mogli porediti** i kako bismo smanjili mogućnost da se neka (ili više njih) pravilno klasificirana instanca pojavljuje više puta i u testnom skupu što bi nam dalo lažni privid da klasifikator ima visoku tačnost, dok u stvarnosti to ne bi bilo tako. Nakon oversamplinga zapravo se znatno smanjio broj instanci koje imaju labelu "No", a skoro pa duplo uvećao broj instanci koje imaju labelu "Yes". Kod undersampleinga broj instanci koje imaju labelu "No" je znatno smanjen što znači gubitak informacija.

Nakon što uporedimo sve ROC krive, zaključujemo da su dva najbolja klasifikatora jednoslojna mreža sa jednim neuronom i jednoslojna mreža sa 4 neurona. ROC krive su prikazane na sljedećoj slici:



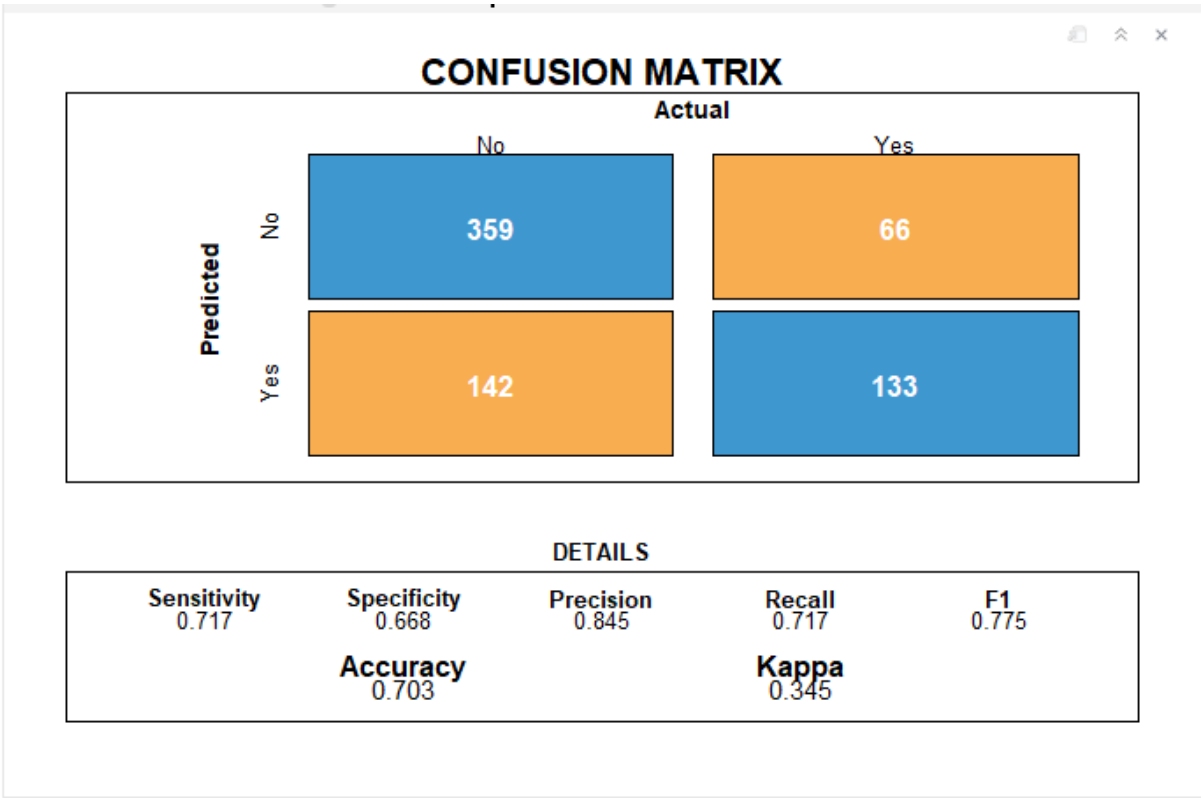
(a) Jednoslojna mreža s jednim neuronom

(b) Jednoslojna mreža s četiri neurona

Slika 43: ROC krive najboljih klasifikatora

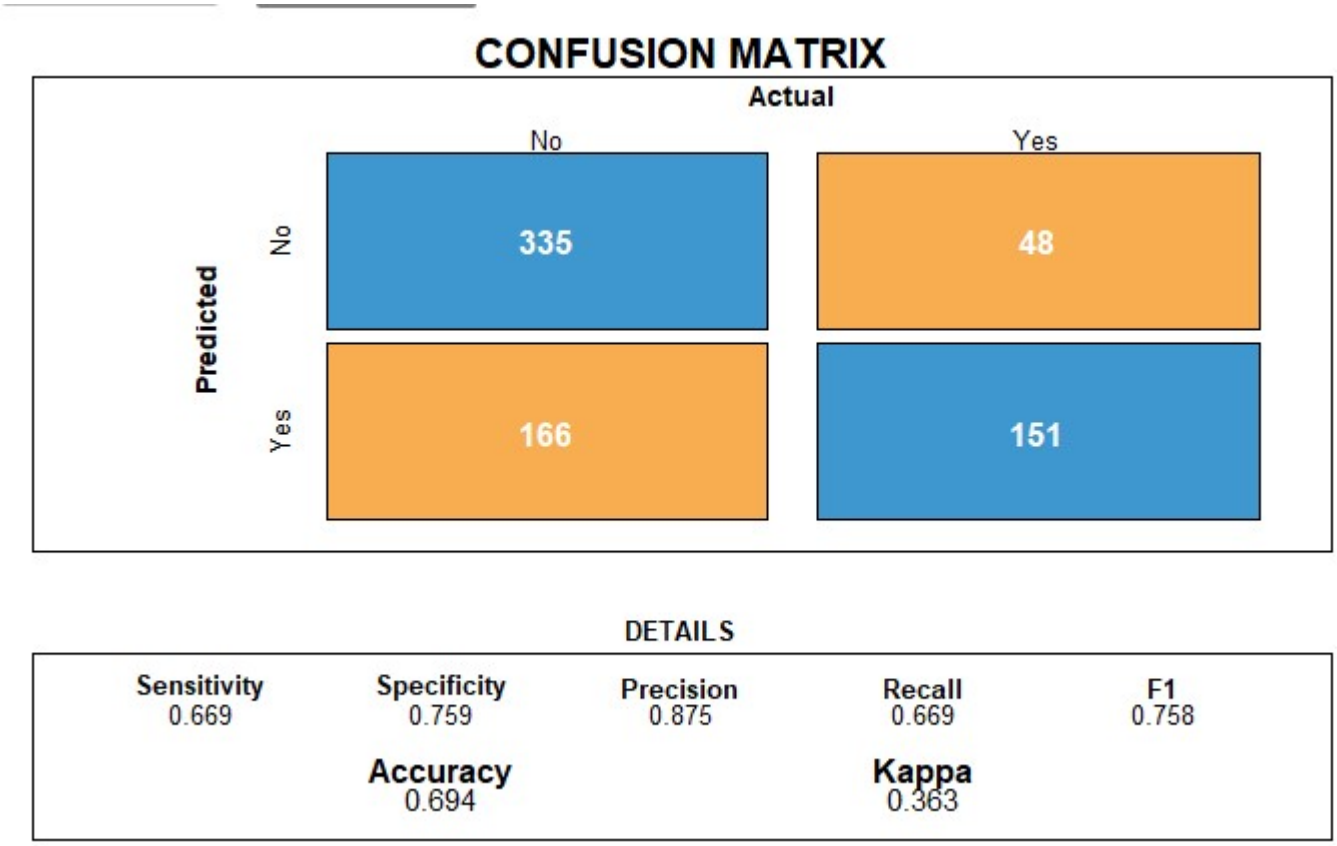
1.7 b)

Najbolje rezultate u Zadatku 1.a smo dobili za SVM klasifikator sa radijalnim kernelom. Kada smo ovaj klasifikator primijenili nad testnim skupom dobili smo sljedeće rezultate:



Slika 44: SVM klasifikator sa radijalnim kernelom nad testnim skupom

Najbolje rezultate u prvoj zadaći nam je dao bagging model. Međutim nad testnim skupom bolje rezultate nam je dao C5.0 pa smo prikazali njegove rezultate.



Slika 45: Primjena C5.0 modela nad testnim skupom

Poređenjem konfuzijskih matrica SVM klasifikatora sa radijalnim kernelom i matrice C5.0 modela možemo zaključiti da imaju veoma slične metrike. Razlikuju se po tome što je kod klasifikatora c5.0 specifičnost veća nego kod SVM klasifikatora, a senzitivnost manja. U našem slučaju specifičnost se odnosi na labelu "Yes", odnosno od svih instanci koje su zaista označene sa "Yes" koliko njih je predviđeno kao takve. Dakle, C5.0 bolje otkriva kada će korisnik otkazati pretplatu nego li je to slučaj kod SVM. S druge strane, SVM bolje otkriva da li će korisnik zaista ostati pretplatnik usluga. Po našem mišljenju za ovaj konkretni problem otkazivanja pretplate bitnije je da klasifikator bolje otkriva korisnike koji nastoje otkazati pretplatu jer na taj način pružatelj usluga može preduhitriti korisnika da ne otkaže pretplatu, ponuditi mu neke beneficije ili bolji paket usluga. Iz početnog dataseta može se zaključiti da je mnogo više osoba koje ne nastoje otkazati pretplatu, stoga ukoliko i dođe to pogrešne pretpostavke da će neko otkazati pretplatu i ponudi mu se jeftini paket usluga ili više beneficija, to ne može biti tako često i ne može štetiti pružatelju usluga, stoga je za ovaj slučaj bolji klasifikator C5.0. Oba klasifikatora imaju skoro istu tačnost (0,703 za SVM i 0,694 za C5.0).

•KNN model predikcije:

Prednost ovog modela je što je intuitivan, može se koristiti za višeklasnu klasifikaciju i regresiju i što je dovoljan samo jedan hiperparametar k. Mana je što nema baš najbolje performanse nad nebalansiranim setom i što ne može raditi sa NA vrijednostima i što ne može direktno raditi sa kategoričkim vrijednostima već se moraju pretvoriti u numeričke.

• naivni Bayesov model predikcije:

Prednost ovog modela je jer je brz i jer može raditi sa kategoričkim atributima. Mana je što svaki broj tretira kao posebnu kategoriju i to nije pogodno za numeričke atribute.

• model logističke regresije:

Treniranje nije vremenski i kompjuterski zahtjevno. Koristi se za binarnu klasifikaciju, ali se može proširiti na višeklasni problem. Ukoliko su atributi linearno nezavisni, ima izuzetno dobre performanse, ali to dovodi do opasnosti za overfittingom ukoliko je dataset veliki. Nelinearni problemi ne mogu biti klasificirani ovim klasifikatorom. Osjetljiv je na outliere.

• SVM model klasifikacije:

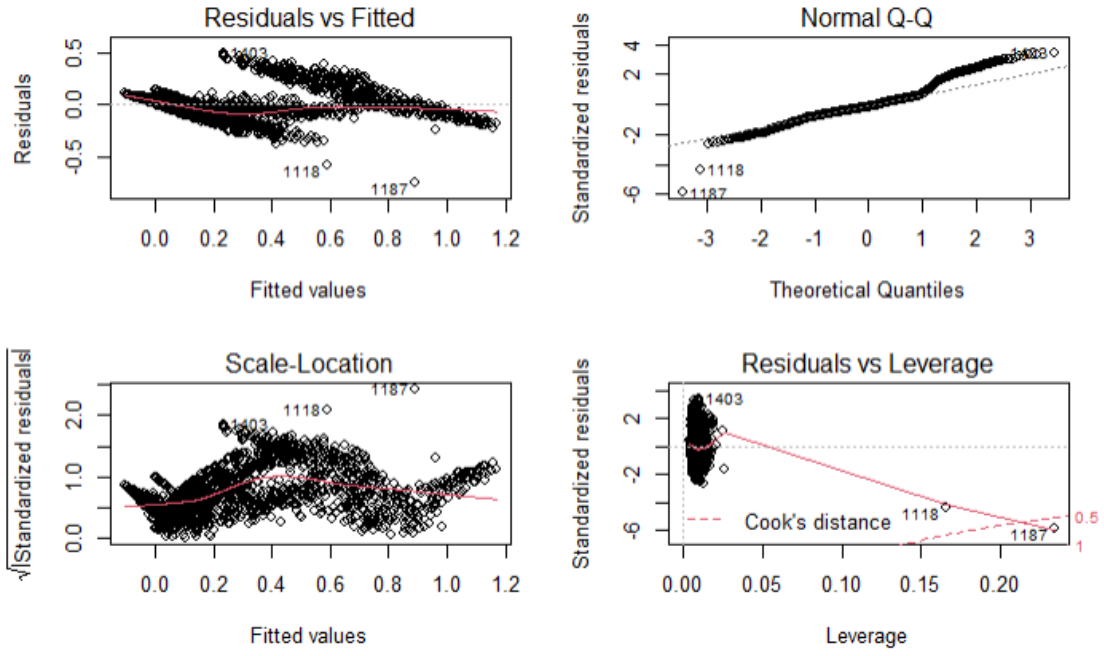
Može se koristiti i za klasifikaciju i za regresiju, može se birati između više kernela. Sa ispravnim parametrima može brzo izvršiti treniranje i nad velikim skupom podataka, ali je izbor pravih parametara težak proces. Radi dobro ukoliko su klase jasno razdvojene, a ukoliko nisu zakazuje. Za razliku od stabla odlučivanja teško ga je interpretirati, s obzirom da se ne oslanja na vjerovatnoću.

• model klasifikacije koji koristi neuralne mreže:

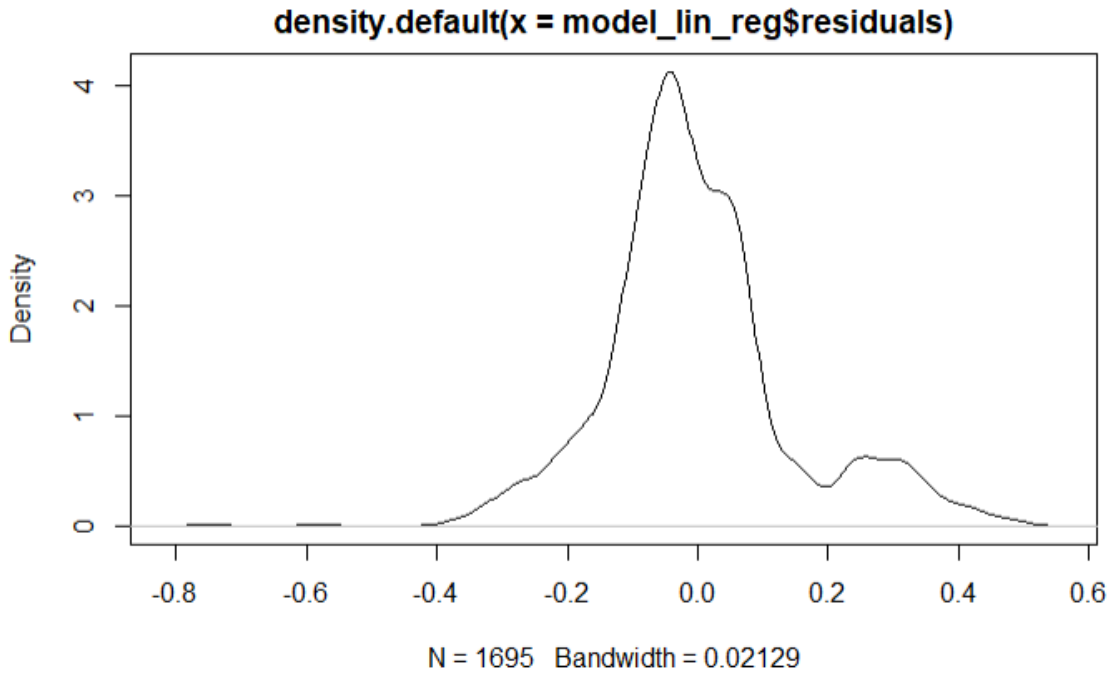
Mana ovog modela je što je treniranje sporo i može da divergira i što može raditi samo sa brojčanim atributima, odnosno mora se vršiti pretvaranje. Može doći do overfittinga ako se izaberu loši hiperparametri. Prednost je što se može koristiti za klasifikaciju i regresiju i što su predviđanja veoma brza.

2 Zadatak 2

Prikaz rezultata za model višestruke linearne regresije.



Slika 46: Rezultati za model visestruke linearne regresije



Slika 47: Rezultati za model visestruke linearne regresije

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.75073 -0.08091 -0.01987  0.05934  0.49423

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      2.898e-02  4.455e-02   0.651   0.5154
X                7.902e-06  6.149e-06   1.285   0.1989
gender           2.970e-03  7.091e-03   0.419   0.6753
Dependents      -6.477e-04  7.827e-03  -0.083   0.9341
tenure           3.201e-03  2.496e-04  12.821   <2e-16 ***
PhoneService     1.856e-02  1.375e-02   1.350   0.1773
MultipleLines    2.071e-03  4.305e-03   0.481   0.6305
InternetService -9.484e-03  6.362e-03  -1.491   0.1362
StreamingTV      1.942e-03  4.541e-03   0.428   0.6689
StreamingMovies  9.000e-03  4.485e-03   2.007   0.0449 *
Contract        -1.124e-02  5.557e-03  -2.022   0.0433 *
PaymentMethod    4.072e-03  3.737e-03   1.090   0.2761
MonthlyCharges   1.424e-03  1.023e-04  13.921   <2e-16 ***
TotalCharges     6.325e-05  2.982e-06  21.208   <2e-16 ***
DailyCharges    -4.320e-02  4.823e-03  -8.958   <2e-16 ***
Churn           -1.365e-02  9.231e-03  -1.478   0.1395
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1456 on 1679 degrees of freedom
Multiple R-squared:  0.8035,    Adjusted R-squared:  0.8018
F-statistic: 457.8 on 15 and 1679 DF,  p-value: < 2.2e-16
```

Slika 48: Rezultati za model visestruke linearne regresije

MAE, RMSE i R2 metrike za model višestruke linearne regresije.

```
[1] 0.1162275
[1] 0.1583798
[1] 0.7165888
```

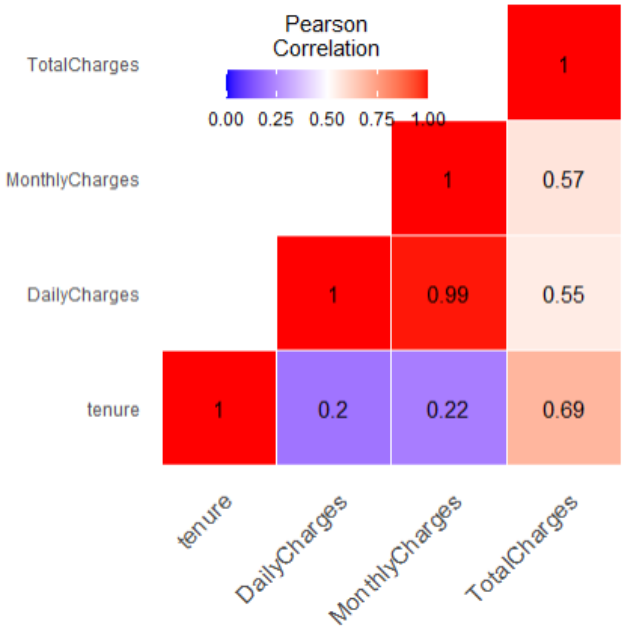
Slika 49: MAE, RMSE, R2 metrike

Ukoliko postoji kolinearnost između ulaznih varijabli, model linearne regresije neće davati ispravne rezultate. Kako bi se odredilo da li postoji kolinearnost ulaznih varijabli, neophodno je izračunati VIF koeficijent (Variance InflationFactor) za izvršene predikcije. Za ovaj model VIF koeficijent je 3.52844. Ova vrijednost koeficijenta pokazuje da postoji srednja kolinearnost varijabli.

```
VIF: 3.528442
```

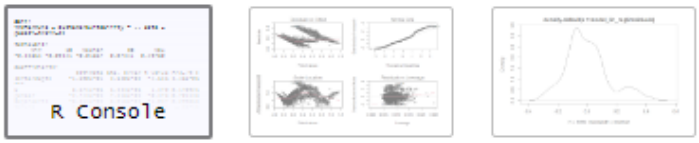
Slika 50: VIF koeficijent

Kako bi smanjili kolinearnost, potrebno je izbaciti varijable koje imaju visok stepen korelacije. Iskoristit ćemo Pearsonov koficjent za određivanje stepena korelacije.



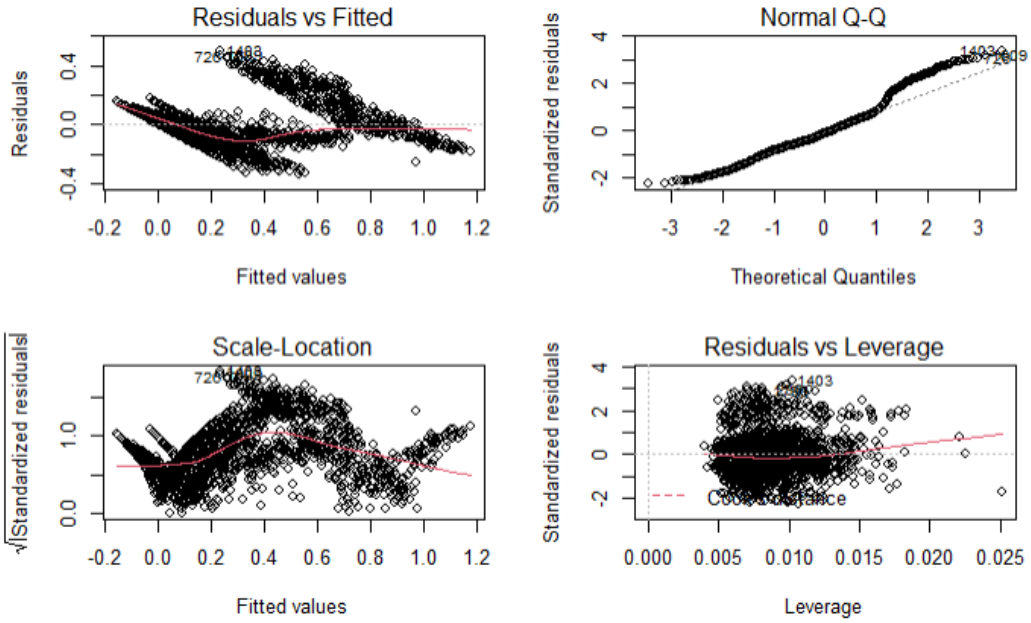
Slika 51: Pearsonov koeficijent

Na osnovu Pearsonovg koeficijenta zaključujemo da su DailyCharges i MonthlyCharges u snažnoj korelaciji, pa smo izbacili DailyCharges. Na sljedecim slikama su prikazani rezultati nakon što smo izbacili DailyCharges.

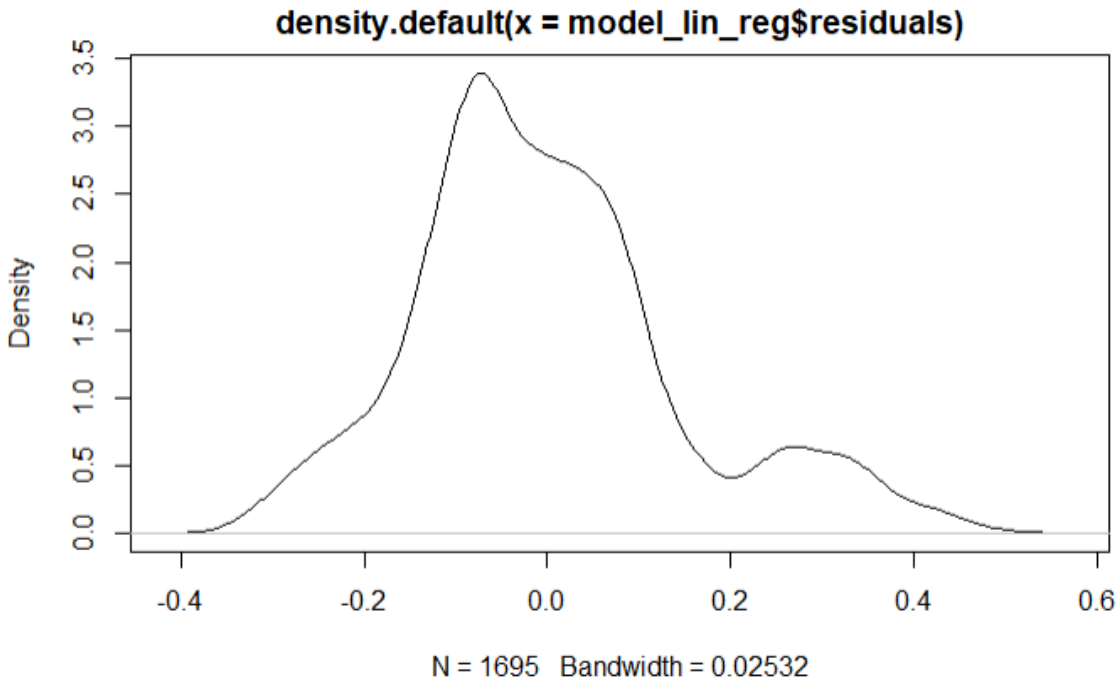



```
-0.33865 -0.09341 -0.01867  0.07341  0.49780

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    -1.690e-01  3.630e-02  -4.656 3.48e-06 ***
X               8.675e-06  6.333e-06   1.370 0.170926
gender         -2.743e-03  7.258e-03  -0.378 0.705538
Dependents     -8.315e-03  8.018e-03  -1.037 0.299868
tenure          3.384e-03  2.609e-04  12.971  < 2e-16 ***
PhoneService   -3.888e-03  1.421e-02  -0.274 0.784464
MultipleLines   1.811e-03  4.335e-03   0.418 0.676181
InternetService 2.134e-02  5.670e-03   3.764 0.000173 ***
StreamingTV     7.473e-03  4.623e-03   1.617 0.106170
StreamingMovies 1.149e-02  4.503e-03   2.551 0.010832 *
Contract       -1.283e-02  5.711e-03  -2.247 0.024742 *
PaymentMethod   3.329e-03  3.826e-03   0.870 0.384361
MonthlyCharges  5.377e-04  2.111e-05  25.472  < 2e-16 ***
TotalCharges    6.330e-05  3.029e-06  20.902  < 2e-16 ***
Churn          -2.600e-02  9.430e-03  -2.757 0.005893 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1492 on 1680 degrees of freedom
Multiple R-squared:  0.7977,    Adjusted R-squared:  0.796
F-statistic: 473.1 on 14 and 1680 DF,  p-value: < 2.2e-16

VIF: 3.059151  [1] 0.1236491
[1] 0.1577073
[1] 0.6731119
```

Slika 52: Rezultati za model visestruke linearne regresije nakon unaprjeđenja



Slika 53: Rezultati za model visestruke linearne regresije nakon unaprjeđenja



Slika 54: Rezultati za model visestruke linearne regresije nakon unaprjeđenja

MAE, RMSE i R2 metrike za ensamble model.

```
0.1812532  0.2178873  0.6060953
```

Slika 55