Univerzitet u Sarajevu
Elektrotehnički fakultet
Mašinsko učenje

IZGRADNJA MODELA KLASIFIKACIJE I VIŠESTRUKA LINEARNA REGRESIJA

IZVJEŠTAJ ZA ZADAĆU 2

Studenti: Amina Alagić, Nejra Rovčanin, Ema Rudalija
Datum: 9.1.2022.
Predmetni profesor: Prof.dr. Dženana Đonko
Predmetni asistent: Ehlimana Krupalija

S A D R Ž A J

1. PREPROCESIRANJE PODATAKA

Za potrebe Zadaće 2 nije korišteno preprocesiranje podataka koje je urađeno u sklopu Zadaće 1. Preprocesiranje u Zadaći 1 je imalo određene nedostatke koji nas nisu doveli do zadovoljavajućih rezultata kada je u pitanju treniranje i testiranje klasifikatora. Najveći propust je bio popunjavanje NA vrijednosti medijan-vrijednostima kada je mogao biti iskorišten i kategorički medijan.

Zbog ovog, ali i nekih drugih propusta u preprocesiranju, na ovom mjestu će biti objašnjen način na koji smo ovaj put uradili ovaj zadatak. Nakon učitavanja podataka, izbačene su kolone *DailyCharges* i *TotalCharges* jer iste imaju visok stepen korelacije sa kolonom *MonthlyCharges*. Od kategoričkih kolona su izbačene *MultipleLines* i *StreamingMovies* koje su u korelaciji sa *PhoneService* i *StreamingTV* respektivno.

Kada je u pitanju popunjavanje NA vrijednosti, nedostajuće vrijednosti kategoričkih varijabli su popunjene najfrekventnijom kategorijom, a nedostajuće vrijednosti numeričkih varijabli su popunjene tako da su:

- Numeričke vrijednosti kolone *tenure* popunjene na osnovu kategoričke varijable *Contract*
- Numeričke vrijednosti kolone *MonthlyCharges* popunjene na osnovu kategoričke varijable *InternetService*

U ovakvom skupu podataka su bila prisutna tri ouliera u kolonama *Dependents* (vrijednost „*Maybe*“), *PaymentMethod* (vrijednost „*abcd*“) i *MonthlyCharges* (negativna vrijednost) i sva tri su izbačena.

Nakon svih izmjena u skupu podataka su ostale dvije numeričke kolone. Nad obje kolone je urađeno decimalno skaliranje i tako skalirane vrijednosti su korištene u klasifikatorima koji će biti opisani u nastavku.

Na ovom mjestu će biti objašnjen način na koji su podaci balansirani. Obzirom da su instance da labelom „Yes“ u našem data setu malobrojne u odnosu na instance sa labelom „No“, podaci su balansirani tehnikom oversample tako da je ukupni broj instanci postao 2300 od čega je korišteno 1332 primjerka sa labelom „No“ i 968 sa labelom „Yes“.

2. IZGRADNJA MODELA KLASIFIKACIJE

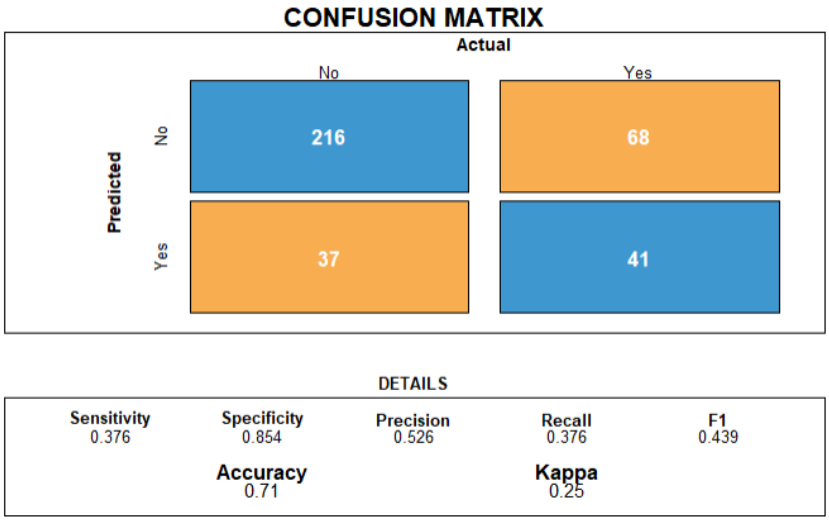
U nastavku ćemo pokazati kako smo nad ovako preprocesiranim podacima primijenili sljedeće klasifikatore:

- KNN-model predikcije
- Naive Bayes model predikcije
- Model logističke regresije
- SVM-model predikcije
- Model klasifikacije koji koristi neuralne mreže

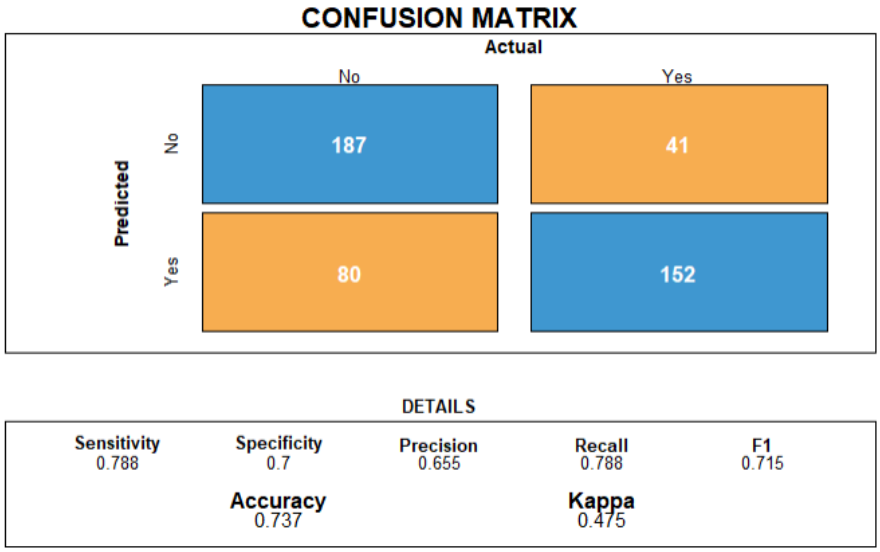
2.1 KNN-MODEL PREDIKCIJE

Kako bismo izgradili KNN model predikcije, prvi korak nakon preprocesiranja podataka je bio pretvaranje kategoričkih vrijednosti u numeričke. Obzirom da funkcija *knn* iz biblioteke class u R-studiu ne podržava treniranje modela ukoliko postoje kategoričke varijable, izvršeno je pretvaranje u numeričke korištenjem funkcije *as.numeric* nad faktor varijablama za kolone *gender*, *Dependents*, *PhoneService*, *InternetService*, *StreamingTV*, *Contract* i *PaymentMethod*.

U idućem koraku izvršena je podjela podataka na trening i testni skup. Trening skup je činilo 80% podataka, a za podjelu je korištena holdout-metoda. Primjena KNN algoritma, odnosno *knn* funkcije nad ovako podijeljenim podacima je urađena na način kako je pokazano na Labaratorijskoj vježbi 6, a za parametar k je izabrana vrijednost 15. To znači da će prilikom testiranja svake instance naš algoritam porediti instancu sa najbližih 15 susjeda. Pomenuta udaljenost se u prvom treniranju računa uz pomoć Euklidske distance. Evaluacija tačnosti je analizirana korištenjem konfuzijske matrice iz koje se vidi da tačnost iznosi 71%. Upotrebom k-fold validacije došli smo do srednje tačnosti od oko 75% i srednje kappe od oko 0.28 za podjelu podataka na 15 foldova.

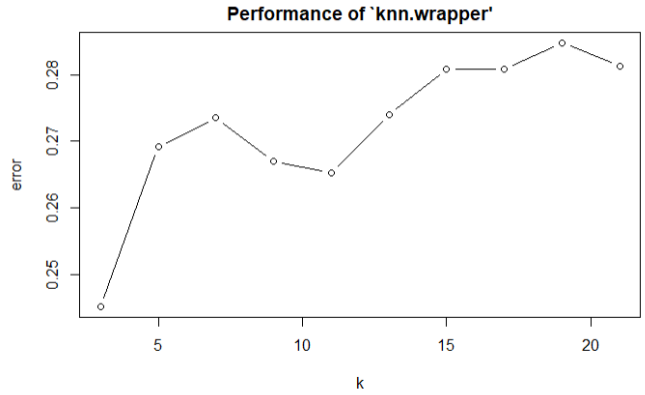


Kao što je već obrađeno u Zadaći 1, ovaj set podataka je nebalansiran i kao takav znatno utiče na metrike kao što su osjetljivost, kappa i F1, konkretno u konfuzijskoj matrici iznad. Da bismo poboljšali ove metrike, urađen je oversample podataka, tako da se ukupni broj instanci povećao, a broj instanci sa labelom "Yes" i broj instanci sa labelom "No" se izjednačio.
Nakon što smo ponovili sve korake koji su opisani iznad, ali ovaj put za balansiranje podatke, dobili smo sljedeće rezultate.

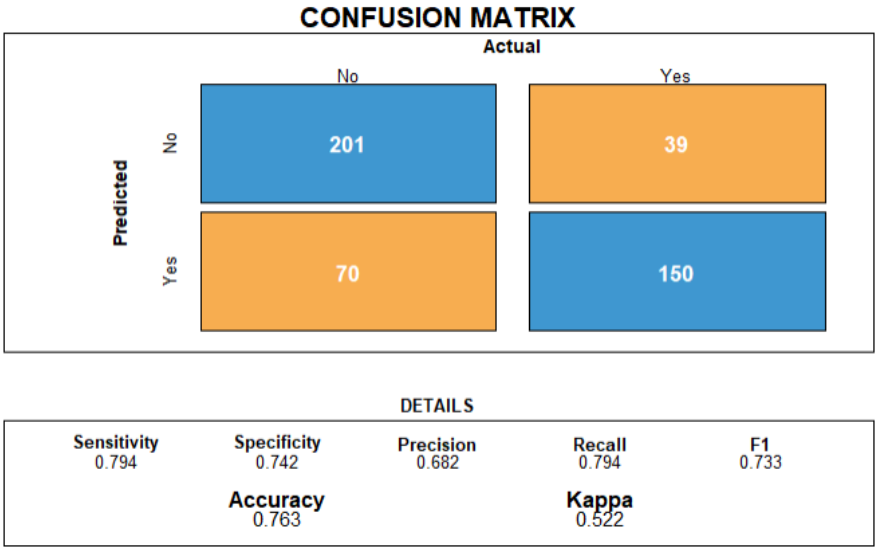


Vidimo da je u ovom koraku došlo do poboljšanja problematičnih metrika koje su prethodno navedene, ali tačnost sada iznosi 73.7%, što je lošije čak i od srednje tačnosti modela koji je radio sa nebalansiranim podacima. Ponovno je izvršena 15-fold validacija nakon koje smo dobili srednju tačnost od oko 72.6% i srednju kappu od oko 0.44.

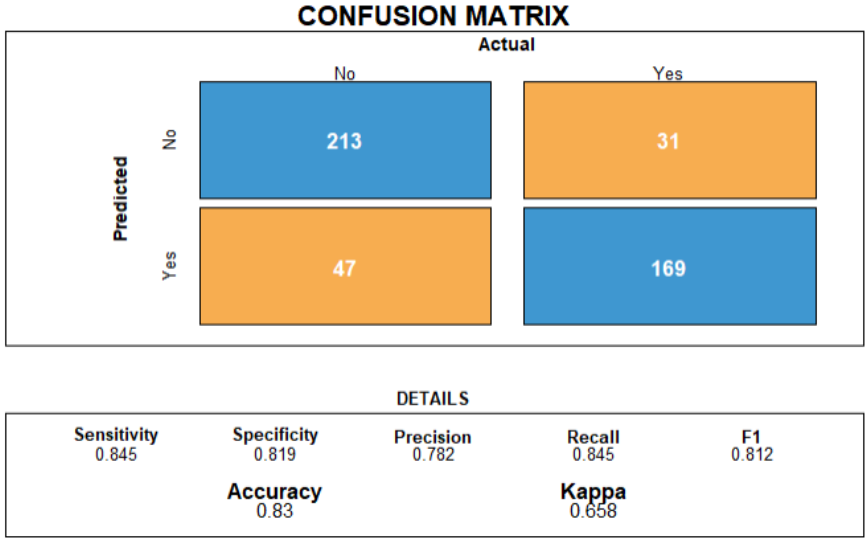
Da bismo dodatno poboljšali tačnost KNN modela izvršen je i tuning parametara, konkretno ispitivanje koja od proslijeđenih vrijednosti parametra k daje najbolje rezultate. U funkciju *tune.knn* atribut k je postavljen na vektor vrijednosti c(3, 5, 7, 9, 11, 13, 15, 17, 19, 21). Rezultat tuninga parametara dao je grafik koji je prikazan ispod.



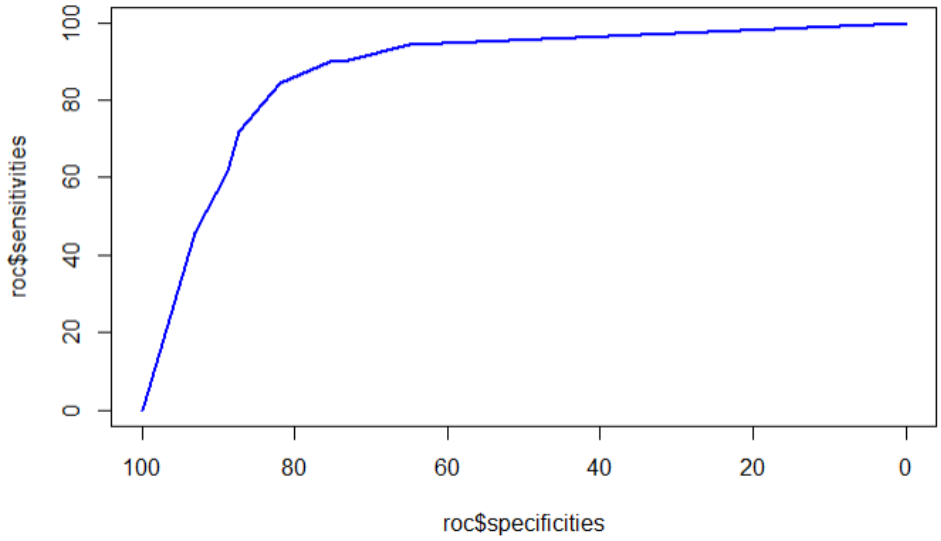
Sa ovog grafika možemo zaključiti da greška modela raste što je broj susjeda veći. Iz ovog razloga je za dalje treniranje modela korišteno k = 3, odnosno 3 susjeda. Ponovnim treniranjem modela dobili smo sljedeće rezultate.



Vidimo da je tuning parametra k znatno poboljšao sve metrike. Kao što je pomenuto na početku, za mjerenje udaljenosti između podataka korištena je Euklidska distanca. U narednom koraku smo probali treniranje klasifikatora korištenjem i drugih distanci kao što su Manhattan i Minkowski distanca pri čemu je Minkowski distanca dala neznatno bolje rezultate od ostalih.

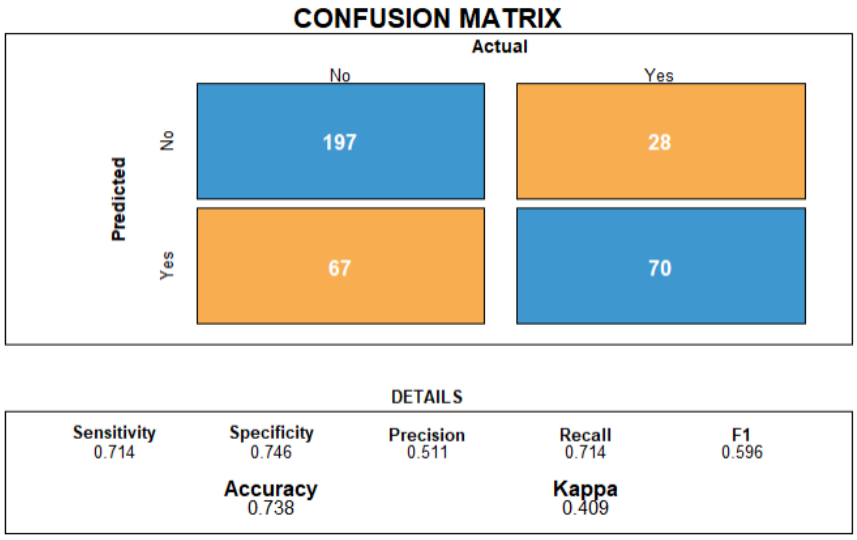


Vidimo da je upotreba *kknn* funkcije iz istoimene biblioteke donijela dosta bolje rezultate. Tačnost modela sada iznosi čak 83%, a ostale metrike su prikazane na konfuzijskoj matrici za Minkowski distancu. Također, evaluacija modela je izvršena i korištenjem ROC-krive.



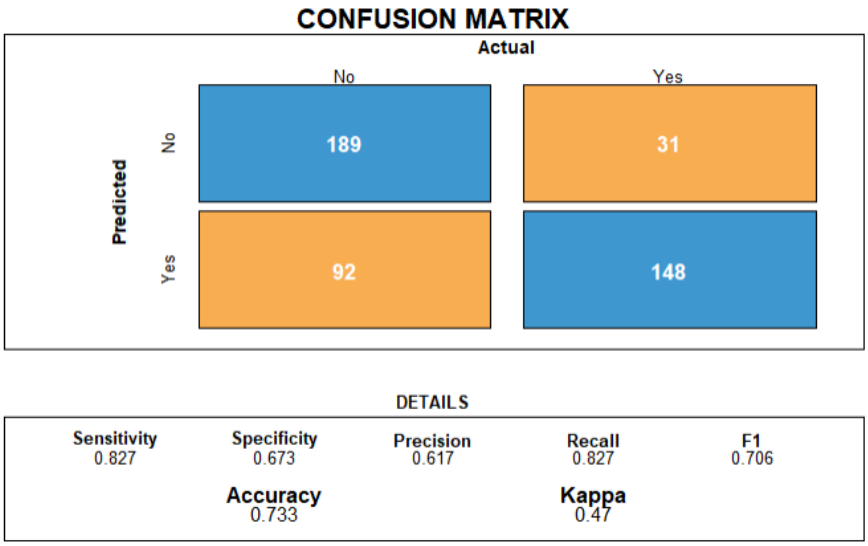
2.2 NAIVNI BAYESOV MODEL PREDIKCIJE

Izvršili smo podjelu podataka na trening i testni skup. Kao i u prethodnom slučaju, trening skup je činilo 80% podataka, a za podjelu je opet korištena holdout-metoda. Ovakva podjela podataka kao i korištenje holdout-metode je prisutno u svim modelima koji slijede. Za treniranje Bayesovog modela predikcije koristili smo funkciju *naiveBayes* iz biblioteke *e1071* kojoj smo proslijedili trening skup podataka. Ovako istreniran model je zatim proslijeđen funkciji predict na način kako je to urađeno u svim ostalim modelima i prethodnoj zadaći.



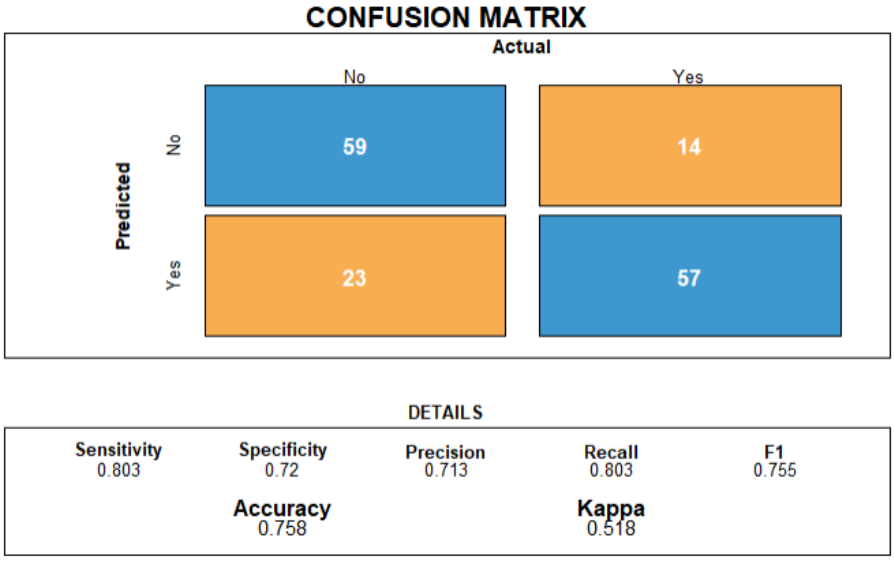
Vidimo da se dobija tačnost od 73.8%, osjetljivost 0.71, specifičnost 0.74, kappa statistika 0.4. Nakon izvršenja 15-fold validacije dobili smo da je srednja tačnost oko 75%, a srednja kappa oko 0.4.

Ponovili smo navedeni postupak za balansirane podatke i dobili sljedeće rezultate.



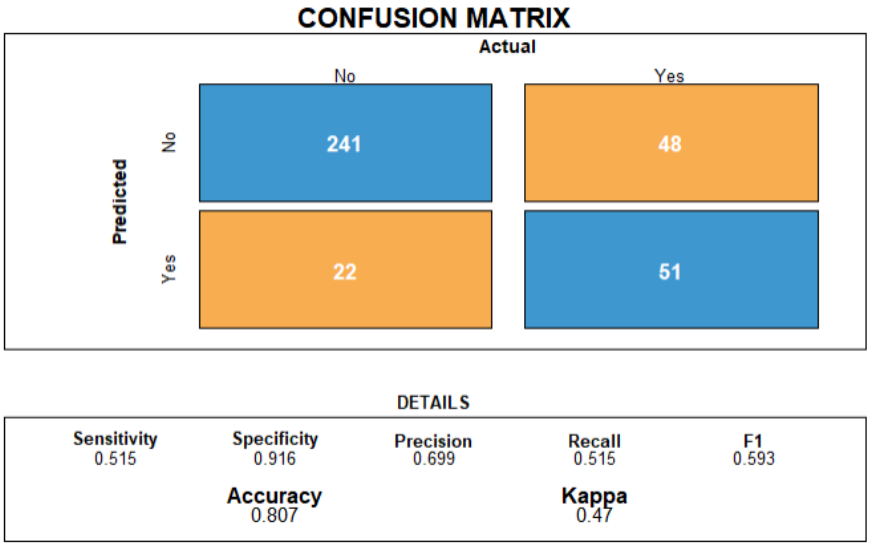
Nakon 15-fold validacije srednja tačnost ostaje ista kao i za nebalansirane podatke, a srednja kappa se povećala na 0.48. Metrike osjetljivost i F1 su se poboljšale, dok se specifičnost pogoršala.

Korištenjem ensamble tehnika za naivni Bayesov model dobili smo nešto bolje rezultate. U zadaći je ručno napravljena funkcija *ensemble* koja koristi bagging-pristup modelu, a način njene izgradnje je dat u uputstvu na c2.

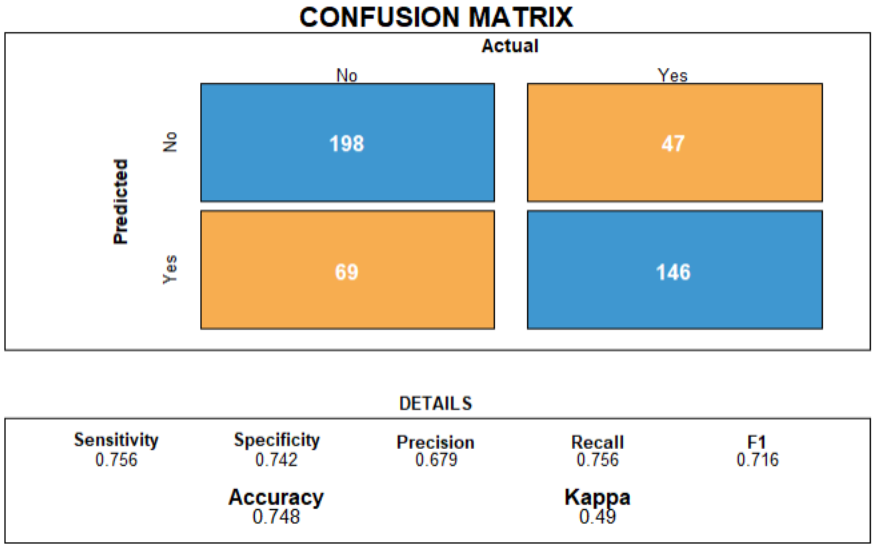


2.3 MODEL LOGISTIČKE REGRESIJE

Treniranje modela je izvršeno korištenjem funkcije *glm* kojoj su proslijeđeni trening podaci te je specificirana vrsta funkcije za logističku regresiju (*logit*). S obzirom da se kao rezultat funkcije predict za model logističke regresije dobiju vrijednosti između 0 i 1 (vjerovatnoće), izvršili smo ručno pretvaranje tih vjerovatnoća u labele klasa. Performanse modela za nebalansirane podatke su prikazane na sljedećoj slici.

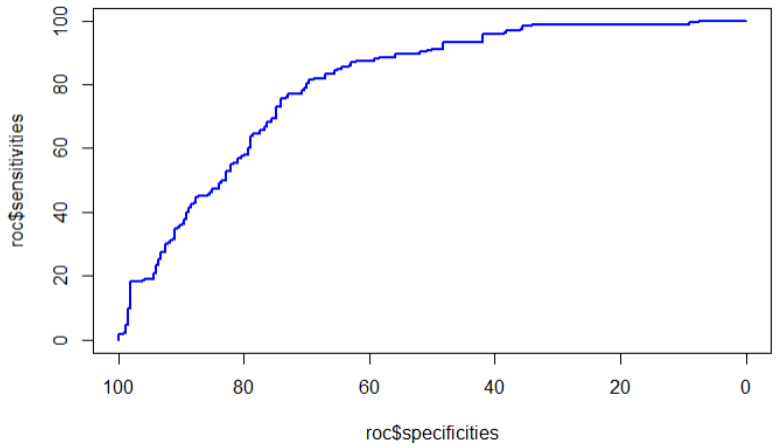


Za ovaj holdout dobijena je tačnost od 80.7%, dok srednja tačnost iznosi oko 77%, a srednja kappa oko 0.38.
Rezultati za balansirane podatke su na sljedećoj slici.

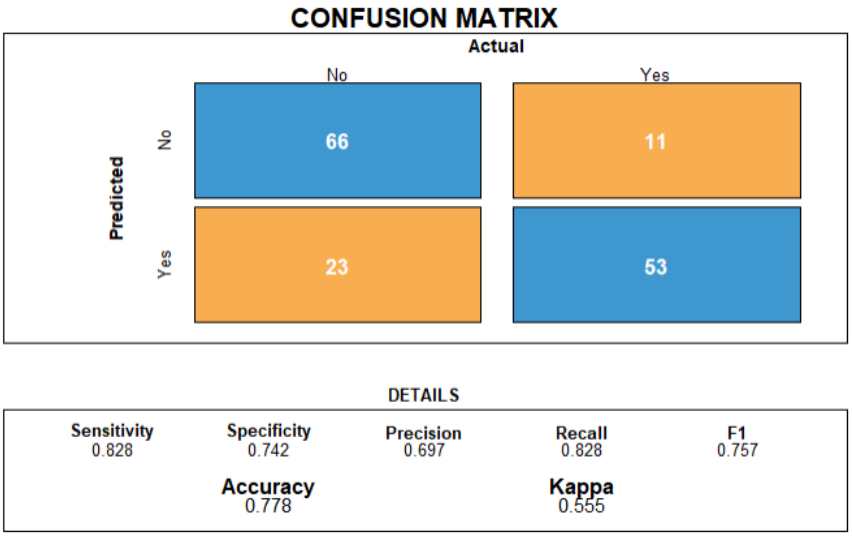


Srednja tačnost se nakon balansiranja smanjila na 75%, a srednja kappa se povećala na 0.49.

ROC kriva za ovako istreniran model logističke regresije je prikazana u nastavku.

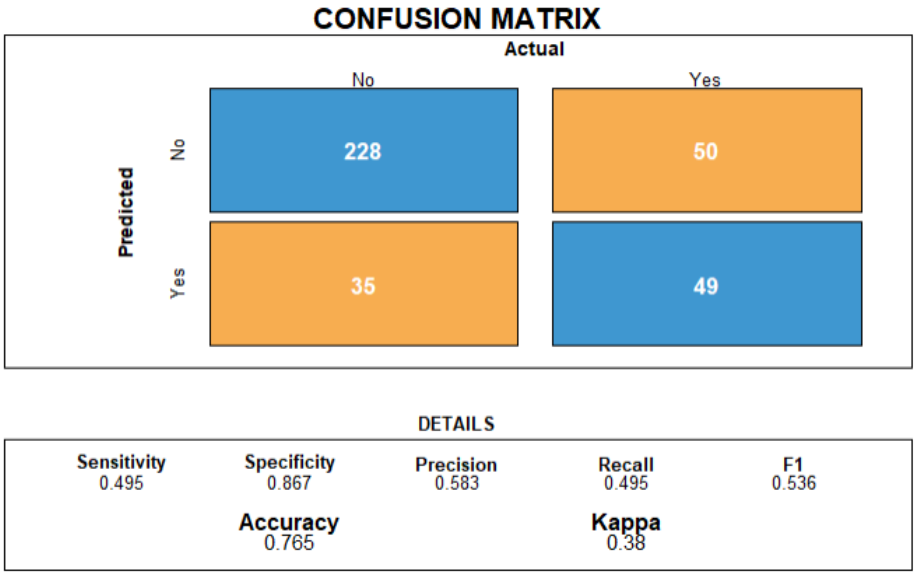


Kao i u prethodnim modelima, i za slučaj modela logističke regresije smo koristili *ensamble*-funkciju nad balansiranim podacima. Dobili smo sljedeće rezultate.



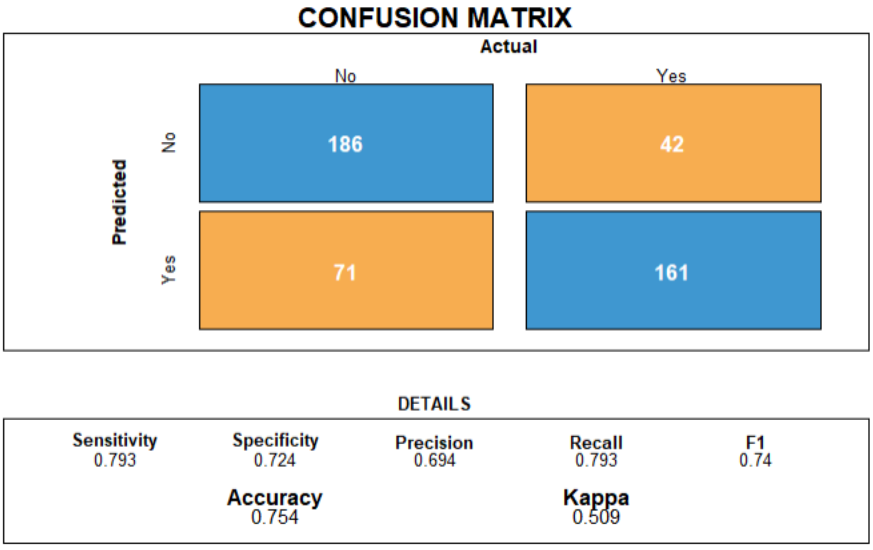
2.4 SVM-MODEL KLASIFIKACIJE

Kada je riječ o SVM modelu klasifikacije, nakon podjele podataka je primijenjena *svm* funkcija kojoj je proslijeđen trening-skup, a kao funkcija kernela je korištena linearna funkcija. Nakon ovako istreniranog modela I izvršenih predikcija dobiveni su sljedeći rezultati.



Da bismo dobili bolji uvid u tačnost modela prilikom treniranja više foldova, primijenili smo 15-fold validaciju na način na koji je to urađeno i za slučaj prethodnih modela. Rezultat 15-fold validacije je dao srednju tačnost od oko 77%, a srednja kappa je iznosila oko 0.38. Budući da se radi o najjednostavnijem SVM modelu koji koristi linearnu kernel funkciju možemo reći da su ovi rezultati zadovoljavajući.

Prije tuninga hiperparametara podaci su izbalansirani korištenjem oversamplinga. Nakon toga je skup ponovno podijeljen na trening i testni skup, a zatim opet korištena funkcija *svm* kojoj smo proslijedili linearni kernel, kao i u prethodnom slučaju. Kada je riječ o tačnosti, možemo reći da se ona nije znatno promijenila u odnosu na slučaj nebalansiranih podataka (konkretno za ovaj fold).



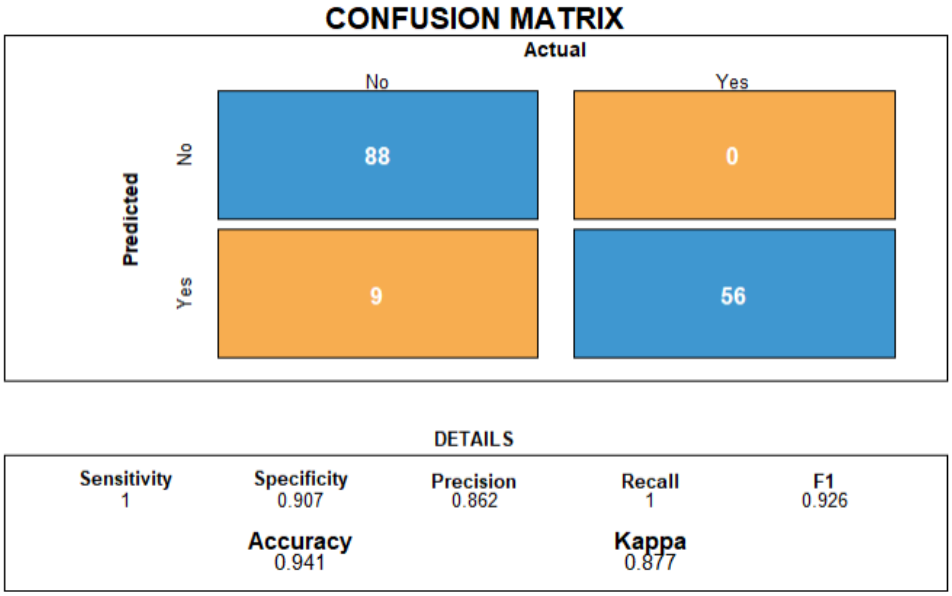
Srednja tačnost dobivena 15-fold validacijom je dala čak lošije rezultate (75%) nego u prethodnom slučaju. Kappa, F1 mjera i osjetljivost su značajno poboljšani. Srednja kappa za balansirane podatke iznosi oko 0.5.

Za potrebe tuninga hiperparametara korištena je funkcija *tune.svm* koja se nalazi u biblioteci *tune*. Korištenjem ove funkcije izvršili smo tuning parametara *cost* i *gamma* koji će nam biti potrebni za treniranje modela koji koriste neki drugi kernel osim linearnog. Obzirom da svm.tune ne podržava rad sa kategoričkim varijablama, iste su pretvorene u numeričke, a zatim su dobiveni sljedeći rezultati. Parametar *cost* je od vrijednosti vektora c(1, 5, 10, 25, 50) je dao najbolje rezultate za posljednji slučaj, odnosno cost = 50. Od mogućih vrijednosti *gamma* parametra u vektoru c(0.001, 0.01, 0.1, 1), najbolji rezultat je dala vrijednost 1.

Kao što je ranije rečeno, *cost* i *gamma* će nam biti potrebni kao dodatni parametri funkcije *svm*. Koristeći vrijednosti koje su dale najbolji rezultat prilikom tuninga parametara (50 i 1), kreirali smo još četiri SVM modela predikcije koji podržavaju linearni, polinomijalni, radijalni i sigmoidni kernel respektivno. Očekivano, radijalni kerner je imao najbolje performanse.

Kernel	Tačnost	F1	Kappa	Osjetljivost	Specifičnost
Linearni	72.6%	0.69	0.45	0.75	0.71
Polinomijalni	76.7%	0.74	0.53	0.81	0.74
Radijalni	81.3%	0.79	0.62	0.87	0.78
Sigmoidni	58.5%	0.5	0.14	0.52	0.63

Budući da je radijalni kernel dao najbolje rezultate, upravo nad njegovim modelom je primijenjena *ensamble*-funkcija. Rezultati pokazuju da primjena ensamble tehnika nad svm modelom koji koristi radijalni kernel značajno poboljšava sve performanse modela.

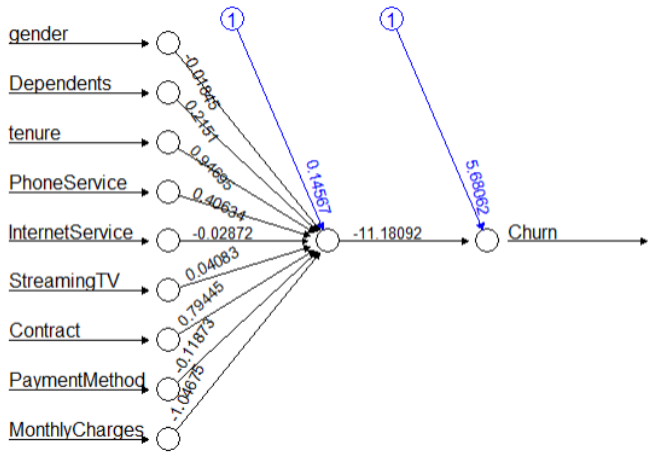


2.5 NEURALNE MREŽE

Za potrebe modela koji koristi neuralne mreže ponovo smo učitali set podataka te izvršili iste korake kao što je opisano na početku, osim u slučaju skaliranja podataka. Zbog osobine neuralnih mreža da rade isključivo sa numeričkim podacima najprije su sve kategoričke varijable pretvorene u numeričke. Nakon toga su tako pretvoreni podaci skalirani min-max normalizacijom. U nastavku će biti opisane 3 arhitekture neuralnih mreža koje su istrenirane za naš set podataka: perceptron sa jednim slojem i jednim neuronom, perceptron sa jednim slojem i četiri neurona i perceptron sa dva sloja i jednim neuronom.

2.5.1 PERCEPTRON SA JEDNIM SLOJEM I JEDNIM NEURONOM

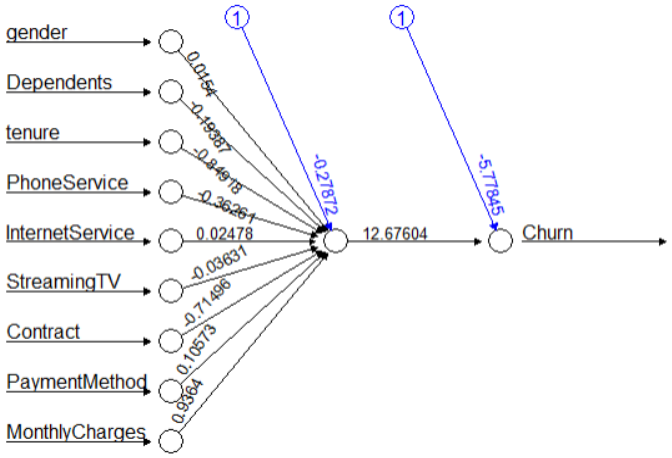
Perceptron sa jednim slojem i jednim neuronom je najprije istreniran kroz jednu epohu, a zatim kroz 5 epoha. Treniranje modela kroz jednu epohu izvršeno je u 43 501 koraka, potrebno vrijeme izvršenja je bilo oko 15 sekundi, a greška je iznosila 641.49. Prikaz perceptrona sa jednim slojem i jednim neuronom se može vidjeti na sljedećoj slici.



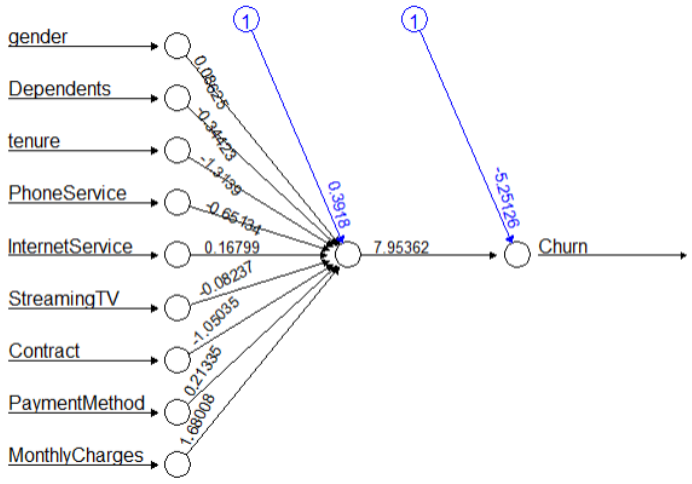
Treniranje istog modela kroz 5 epoha donijelo je slične rezultate kada je u pitanju greška i vrijeme izvršenja.

```
hidden: 1    thresh: 0.01    rep: 1/5    steps:   65048 error: 641.4843        time: 21.48 secs
hidden: 1    thresh: 0.01    rep: 2/5    steps:   40523 error: 641.47614       time: 13.35 secs
hidden: 1    thresh: 0.01    rep: 3/5    steps:   53776 error: 641.48667       time: 17.71 secs
hidden: 1    thresh: 0.01    rep: 4/5    steps:   57938 error: 641.48486       time: 19.11 secs
hidden: 1    thresh: 0.01    rep: 5/5    steps:   57291 error: 641.48634       time: 19.21 secs
```

Prikaz najbolje epohe za ovu arhitekturu je dat na slici ispod.



Ovaj postupak je ponovljen za slučaj balansiranih podataka. Treniranje jedne epohe je izvršeno u 10 160 koraka, potrebno vrijeme izvršenja je bilo oko 3.5 sekundi, a greška je iznosila 634.32. Ovako istrenirani perceptron je prikazan u nastavku.



Error: 634.324113 Steps: 10167

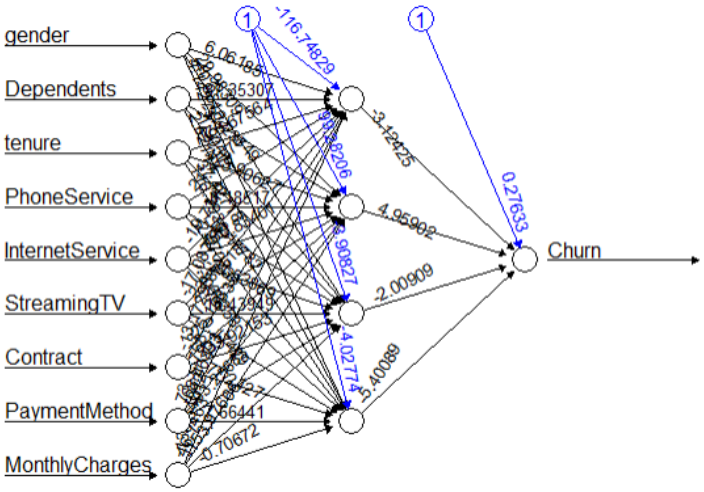
Treniranje kroz više epoha za ovaj slučaj nije donijelo znatno drugačije rezultate u odnosu na slučaj nebalansiranih podataka.

2.5.2 PERCEPTRON SA JEDNIM SLOJEM I VIŠE NEURONA

Za arhitekturu perceptrona sa jednim slojem i više neurona korištena su 4 neurona. Kao i perceptron sa jednim slojem i jednim neuronom i ova arhitektura istrenirana je kroz jednu i kroz 5 epoha. Broj koraka jedne epohe je iznosio 13 493, vrijeme izvršenja je bilo oko 9 sekundi, a greška je iznosila 584.92.

Prilikom treniranja ovog modela kroz 5 epoha primijetili smo smanjenje greške na 551.5 u prvoj epohi izvršenja. Prikaz treniranja epoha i izgleda ove arhitekture u epohi u kojoj je ostvarena najmanja greška mogu se vidjeti na narednim slikama.

```
hidden: 4    thresh: 0.01    rep: 1/5    steps:   24875 error: 551.49478       time: 16.22 secs
hidden: 4    thresh: 0.01    rep: 2/5    steps:   12521 error: 570.16062       time: 7.95 secs
hidden: 4    thresh: 0.01    rep: 3/5    steps:   14407 error: 583.35095       time: 9.25 secs
hidden: 4    thresh: 0.01    rep: 4/5    steps:   59039 error: 577.13103       time: 37.83 secs
hidden: 4    thresh: 0.01    rep: 5/5    steps:   11669 error: 578.76081       time: 7.39 secs
```

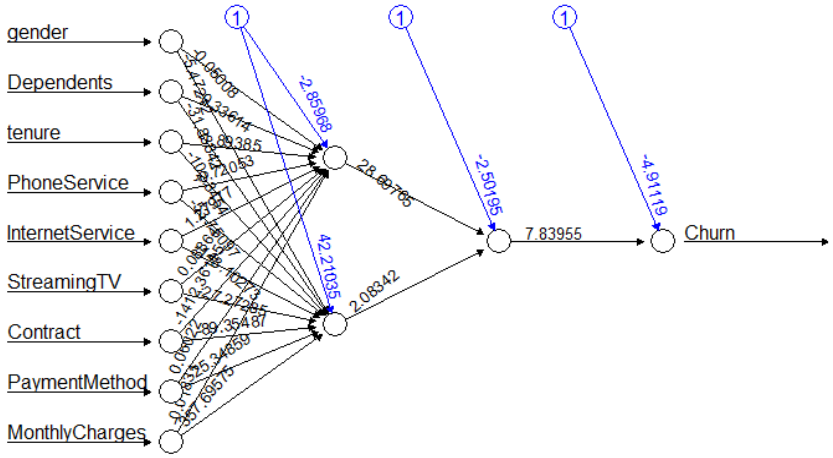


Error: 551.494783 Steps: 24875

Nakon balansiranja podataka dobili smo slične rezultate kada je u pitanju vrijeme izvršenja, broj koraka i veličina greške.

2.5.3 PERCEPTRON SA VIŠE SLOJEVA I JEDNIM NEURONOM

Perceptron sa više slojeva i jednim neuronom istreniran je nakon 31 883 koraka. Vrijeme izvršenja je bilo 17.5 sekundi, a greška je iznosila 612.5. Obzirom da je ovakav perceptron vrlo kompleksan i da je za izvršenje samo jedne epohe bilo potrebno više od 17 sekundi, za ovu arhitekturu nije realizovano treniranje kroz više od jedne epohe. U nastavku se nalazi izgled ove arhitekture primijenjene nad našim skupom podataka.

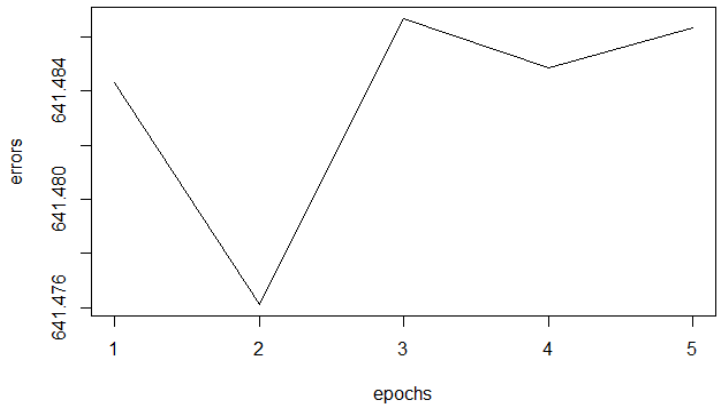


Error: 612.507226 Steps: 31883

Nakon balansiranja podataka dobili smo slične rezultate kada je u pitanju vrijeme izvršenja, broj koraka i veličina greške.

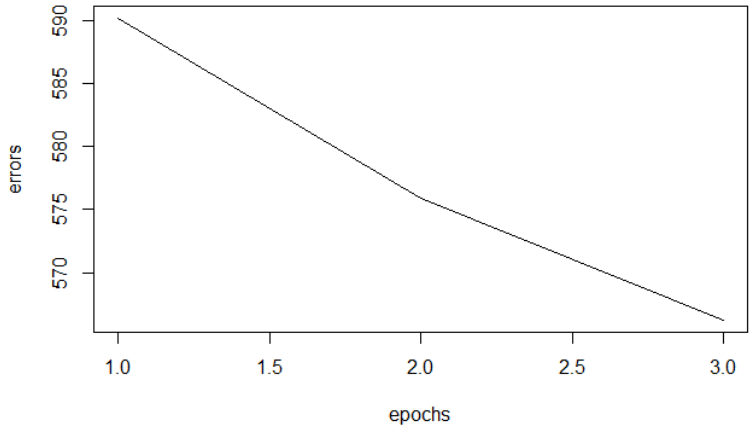
2.5.4 EVALUACIJA VRIJEDNOSTI FUNKCIJE GUBITKA I EVALUACIJA TAČNOSTI

Prilikom evaluacije vrijednosti funkcije gubitka sumirane su sve vrijednosti greške za pojedinačne istrenirane arhitekture i došli smo do zaključka da je perceptron sa jednim slojem i 4 neurona kroz 5 epoha dao najmanju vrijednost funkcije gubitka. U ovom koraku smo analizirali ispravnost odabranog broja epoha. Kao što se može vidjeti na slici ispod perceptron sa jednim slojem i jednim neuronom dolazi do overfittinga nakon druge epohe treniranja.



Kada je riječ o perceptronu sa jednim slojem i 4 neurona, on doživljava overfitting nakon 3 epohe. Da bismo izbjegli overfitting, postavili smo vrijednost rep unutar funkcije neuralnet na rep = 2 za slučaj perceptrona sa jednim slojem i jednim neuronom i rep = 3 za slučaj perceptrona sa jednim slojem i četiri neurona.

Ovakav pristup je značajno popravio vrijednost greške kroz epohe u slučaju perceptrona sa jednim slojem i više neurona. Na sljedećem grafiku vidimo kako je greška u konstantnom padu i kako neće doći do overfittinga.



Nakon izvršene evaluacija tačnosti nad trening-skupom podataka dobili smo sljedeće performanse za sve urađene modele.

Model	Tačnost	F1	Kappa	Osjetljivost	Specifičnost
model_one	77.8%	0.85	0.37	0.89	0.45
model_one_rep	77.8%	0.85	0.37	0.89	0.45
model_multiple	78.5%	0.87	0.27	0.97	0.24
model_multiple_rep	80.8%	0.87	0.45	0.9	0.5

model_multiple_layers	79%	0.86	0.45	0.86	0.58

Nakon izvršene evaluacija tačnosti nad testnim skupom podataka dobili smo sljedeće performanse za sve urađene modele.

Model	Tačnost	F1	Kappa	Osjetljivost	Specifičnost
model_one	75.4%	0.83	0.34	0.89	0.42
model_one_rep	75.4%	0.83	0.34	0.89	0.42
model_multiple	73.5%	0.84	0.17	0.97	0.15
model_multiple_rep	74%	0.8	0.4	0.78	0.64
model_multiple_layers	76%	0.83	0.4	0.85	0.53

Kao što je primjetno iz prethodne dvije tabele, metrike specifičnost i kappa imaju dosta niske vrijednosti. Sljedeći rezultati pokazuju da je rad sa balansiranim podacima popravio performanse ovih metrika. U nastavku su prikazane performanse svih arhitektura prvo za trening-skup, a zatim za testni skup podataka.

Model	Tačnost	F1	Kappa	Osjetljivost	Specifičnost
model_one	74.5%	0.77	0.48	0.74	0.75
model_one_rep	74.5%	0.77	0.48	0.74	0.75
model_multiple	76.8%	0.79	0.53	0.74	0.8
model_multiple_rep	78.3%	0.8	0.56	0.76	0.8
model_multiple_layers	75.4%	0.77	0.5	0.72	0.8

Model	Tačnost	F1	Kappa	Osjetljivost	Specifičnost
model_one	75.7%	0.78	0.51	0.77	0.74
model_one_rep	75.7%	0.78	0.51	0.77	0.74
model_multiple	75.7%	0.77	0.51	0.75	0.76
model_multiple_rep	75.4%	0.78	0.5	0.79	0.7

model_multiple_layers	74.8%	0.76	0.5	0.72	0.77

Konačno, urađen je i tuning hiperparametara uz pomoć funkcije *tune.nnet* iz biblioteke *nnet*. Funkciji su proslijeđene 4 veličine epoha: 1, 2, 5 i 10. Najbolju tačnost je dalo 10 epoha.

```
Najbolja vrijednost size: 10
Najbolja vrijednost decay: 0.1
Najveća tačnost: 0.8345818
```

Svih pet modela ima slične performanse.

2.6 TESTIRANJE NAJBOLJEG MODELA

Prije nego što testiramo model koji je dao najbolje rezultate, kratko ćemo uporediti sve modele sa stanovišta dobivenih performansi.

KNN-model perdikcije je postigao dosta dobre performanse nakon balansiranja podataka i izvršenog tuninga hiperparametara. Međutim, poznato je da se modeli koji se baziraju na računanju distance između podataka mogu izvršavati vrlo nepredvidivo kada se primijene na skup podataka koji sadrži kategoričke varijable, bilo da je riječ o skupovima koje sadrže isključivo kategoričke varijable ili o skupovima koje sadrže i kategoričke i numeričke varijable. Skup nad kojim je primijenjen naš KNN-model je imao 7 kategoričkih i 2 numeričke kolone. Pretvaranje kategoričkih u numeričke vrijednosti je bio neophodan korak. Takvo pretvaranje može uzrokovati gubitak informacija. Slučajevi pretvaranja koji ne dovode do gubitka informacija bi mogle biti binarne kategoričke varijable koje se mogu lako preslikati u 0 ili 1 ili pak varijable sa više kategorija koje imaju prirodni redoslijed (npr. kategorija zastupljenosti sa vrijednostima "*low*", "*intermediate*", "*high*" bi se mogla pretvoriti u numerički tip tako da vrijednost "*low*" postane 1, "*intermediate*" postane 2, a "*high*" postane 3). U našem skupu podataka ipak postoje neki atributi koji imaju više od dvije kategorije, a koje pritom nemaju prirodni redoslijed. Na primjer atribut *Contract* koji predstavlja vrstu ugovora sadrži vrijednosti "*Month-to-month*", "*One year*" i "*Two year*". Očigledno je da ne postoji pravilo koja od varijabli treba da ima najmanju, a koja najveću vrijednost prilikom konverzije. Jedan od načina koji se primjenjuje za ovakve attribute je korištenje one-hot kodiranja.

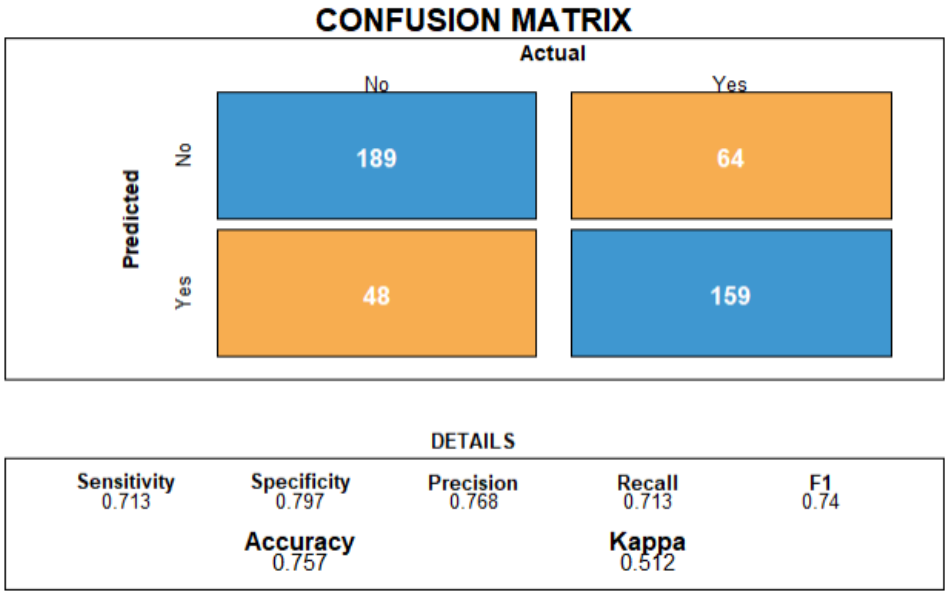
Naivni Bayesov klasifikator je najjednostavniji algoritam koji koristi računanje vjerovatnoća i koji se može primijeniti nad bilo kojim skupom podataka. Za razliku od KNN I SVM modela, za Bayesov model nije izvršen tuning hiperparametara.

Budući da tih hiperparametara nema te da su balansiranje podataka i ensamble-tehnike sve što smo mogli primijeniti da poboljšamo model, možemo reći da je on dao koliko-toliko dobre rezultate. Bayesov klasifikator radi tako da računa pojedinačne vjerovatnoće svih kategorija određenog atributa, a onda koristi te vrijednosti za računanje vjerovatnoće pripadnosti nove instance. Ono što je moglo predstavljati problem u našem skupu podataka je prisustvo dvije numeričke kolone za koje je klasifikator morao računati različite vjerovatnoće kao da se radi o posebnim kategorijama. Drugi problem koji negativno utiče na tačnost klasifikatora je problem nulte frekvencije. Naime, ako u testnom skupu podataka klasifikator naiđe na kategoriju koja se ne nalazi u trening-skupu, takvoj kategoriji će biti dodijeljena vjerovatnoća o i samim tim će model biti spriječen da ostvari dovoljno dobru klasifikaciju te instance.

Neuralne mreže, slično kao i KNN-klasifikator rade isključivo sa numeričkim podacima. Pretvaranje kategoričkih varijabli (koje čine većinu skupa podataka) koje nemaju prirodni redoslijed u numeričke može izazvati nepredvidivo ponašanje klasifikatora kada je u pitanju računanje težina kod neuralnih mreža. Drugi problem koji se dešavao je da neke arhitekture poput perceptrona sa više slojeva uopće ne konvergiraju za neki određeni holdout. Od svih modela, treniranje neuralnih mreža je uzelo najviše vremena, ali i računarskih resursa.

SVM-model klasifikacije je model koji je dao najbolje rezultate za naš data set. Vidjeli smo da je radijalni kernel najviše odgovarao datom skupu podataka te da je bagging nad ovakvim klasifikatorom bio pun pogodak. Zbog toga je upravo ovaj model izabran za testiranje nad testnim skupom podataka. Model logističke regresije, kao što je naglašeno na predavanjima i vježbama, daje slične performanse kao SVM-model. Primijenjen nad našim data setom, model logističke regresije je rezultatski zaista pratio SVM-model. Međutim, budući da SVM-model ima mogućnost poboljšanja kroz tuning hiperparametara i promjenu funkcije kernela, polako je bivao sve bolji i bolji od modela logističke regresije.

Na slici ispod su prikazani rezultati SVM-modela testiranog nad skupom customer_data_test.csv.



3. VIŠESTRUKA LINEARNA REGRESIJA

3.1 IZGRADNJA MODELA VIŠESTRUKE LINEARNE REGRESIJE

U okviru ovog zadatka ponovo je učitan set podataka, izvršeno preprocesiranje kao i ranije, a zatim popunjena kolona *CustomerSuitability* na način koji je opisan u postavci zadaće. S obzirom da linearna regresija koristi samo numeričke varijable, svi kategorički atributi pretvoreni su u numeričke. Dodatno su izbačene kolone *gender* i *Dependents*. Da bismo pojednostavili model, u koloni *StreamingTV* vrijednosti "*No internet service*" i "*No*" su spojene u jednu vrijednost "*No*". Na isti način su u koloni *InternetService* vrijednosti "*DSL*" i "*Fiber optic*" spojene u jednu vrijednost "*Yes*". Slično su modificirane i kolone *MultipleLines* i *StreamingMovies*. Podaci su skalirani decimal scale normalizacijom. Izvršena je podjela podataka na trening i testni skup pri čemu trening skup čini 80% podataka, a za podjelu je korištena holdout-metoda.

Model višestruke linearne regresije je izgrađen pomoću funkcije *lm*. Za početak je za formulu regresije postavljeno *CustomerSuitability* ~ . što znači da se za predikciju vrijednosti varijable koriste sve varijable prisutne u setu podataka. Rezultat poziva *summary* funkcije može se vidjeti na sljedećoj slici.

```
Call:
lm(formula = CustomerSuitability ~ ., data = podaci_regresija_train)

Residuals:
     Min       1Q   Median       3Q      Max
-0.45830 -0.14108 -0.01263  0.12497  0.74916

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     -0.149286   0.050214  -2.973    0.003 **
tenure           0.738355   0.030011  24.603  < 2e-16 ***
PhoneService     0.026391   0.020601   1.281    0.200
InternetService -0.106782   0.021168  -5.045 5.13e-07 ***
StreamingTV     -0.087573   0.012474  -7.020 3.41e-12 ***
Contract        -0.012313   0.008463  -1.455    0.146
PaymentMethod    0.006423   0.005246   1.224    0.221
MonthlyCharges   7.467814   0.322356  23.166  < 2e-16 ***
Churn           -0.107018   0.012767  -8.383  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1891 on 1435 degrees of freedom
Multiple R-squared:  0.7373,    Adjusted R-squared:  0.7358
F-statistic: 503.4 on 8 and 1435 DF,  p-value: < 2.2e-16
```

Za evaluaciju nad testnim skupom podataka korištene su metrike MAE, RMSE i R2. Njihove vrijednosti su na sljedećoj slici.

```
MAE: 0.1530734
RMSE: 0.1933405
R2: 0.7354983
```

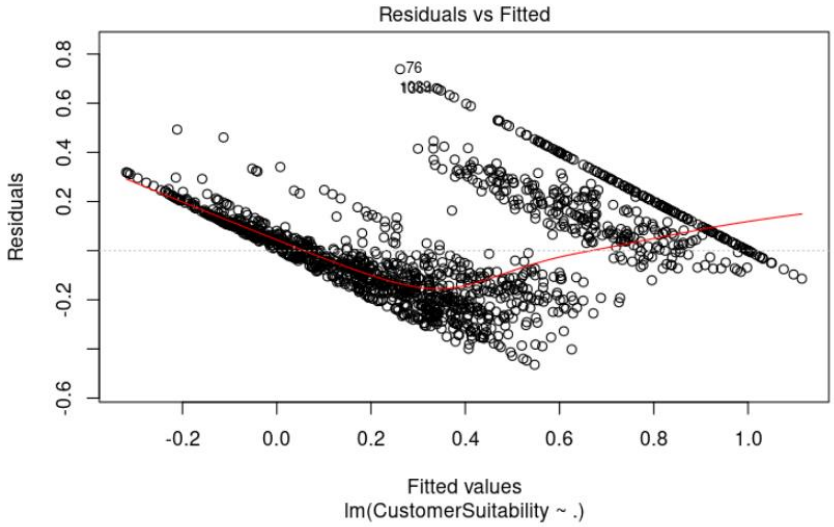
Metrike MAE i RMSE nam govore koliko su u prosjeku predviđene vrijednosti udaljene od stvarnih, a razlika je u načinu računanja ovih metrika. R2 može imati vrijdnost između 0 i 1 i govori nam koliko dobro ulazne varijable opisuju izlaznu varijablu. Poželjno je da ova metrika bude što veća. Konkretno u našem slučaju, metrika MAE nam govori da se predikcije u prosjeku razlikuju za oko 0.15 u odnosu na stvarne vrijednosti, a metrika RMSE da se razlikuju za oko 0.19.

Primijenjen je i ensamble bagging model sa formulom *CustomerSuitability* ~ . i prethodno navedenim preprocesiranjem podataka. Izgradnja ovog modela opisana je u jednom od narednih poglavlja. Na sljedećoj slici su metrike koje se dobijaju primjenom ovog modela.

```
MAE: 0.151739
RMSE: 0.187051
R2: 0.7325282
```

3.1.1 EVALUACIJA PODOBNOSTI POČETNOG MODELA LINEARNE REGRESIJE

Kako bismo ispitali da li važi pretpostavka linearnosti rezidualnih vrijednosti, iscrtan je grafik *Residuals vs Fitted*. Da bi pretpostavka bila ispunjena, crvena linija na ovom grafiku bi trebala biti horizontalna i u blizini nule, što kod nas nije slučaj. Dakle, pretpostavka linearnosti nije ispunjena.



Kako bismo utvrdili da li postoji kolinearnost ulaznih varijabli izračunat je VIF-koeficijent pomoću funkcije *vif* iz biblioteke *car*.

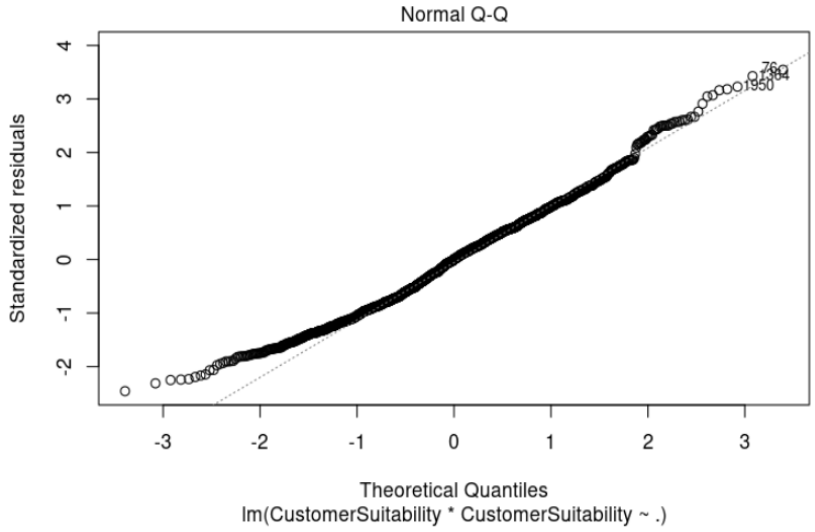
tenure	PhoneService	InternetService	StreamingTV	Contract	PaymentMethod	MonthlyCharges	Churn
2.099869	1.384044	2.686451	1.521056	1.862726	1.121111	3.624217	1.282259

Vidimo da su sve vrijednosti VIF-koeficijenta između 1 i 5 što znači da nema varijabli sa visokom korelacijom. Ovo je i očekivano s obzirom da su na početku, u fazi preprocesiranja, izbačene varijable koje su u korelaciji sa nekim drugim varijablama.

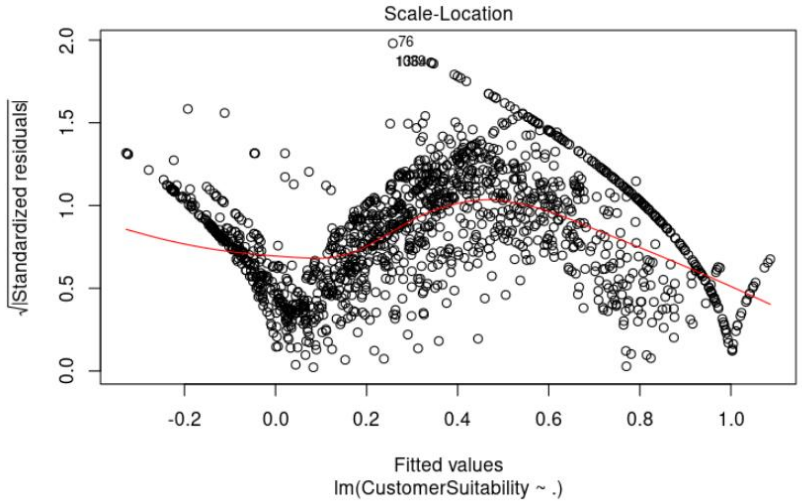
Kako bismo provjerili postoji li autokorelacija rezidualnih vrijednosti, izvršen je Shapiro-Wilk test. S obzirom da je dobijena vrijednost p=3.86e-09 manja od 0.05 to nedvosmisleno znači da postoji autokorelacija odnosno da rezidualne vrijednosti nemaju normalnu distribuciju.

```
Rezultat Shapiro-Wilks testa: 0.9887079
P-vrijednost: 3.858863e-09
```

Također, da reziduali nemaju normalnu distribuciju može se vidjeti i sa Q-Q-plota. Na ovom grafiku tačke reziduala bi trebale biti što bliže pravoj dijagonalnoj liniji.



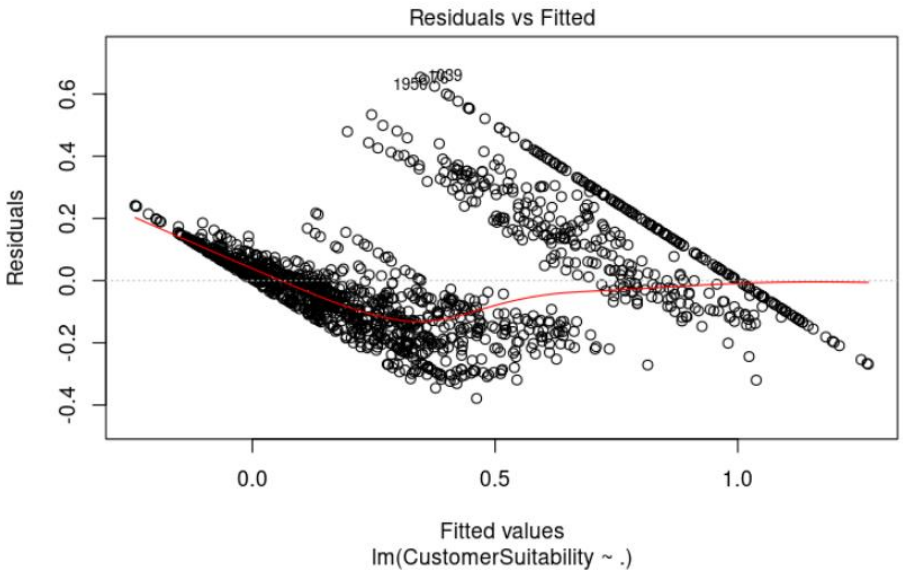
Kako bismo ispitali konstantnost varijanse rezidualnih vrijdnosti odnosno postojanje heteroskedastičnosti rezidualnih vrijednosti korišten je Scale-Location grafik. Da bi se moglo zaključiti da je varijansa konstantna, na ovom grafiku crvena linija bi trebala biti horizontalna i tačke bi trebale biti ravnomjerno raspoređene. Kod nas ovo nije slučaj pa zaključujemo da postoje heteroskedastične vrijednosti i da varijansa nije konstantna.



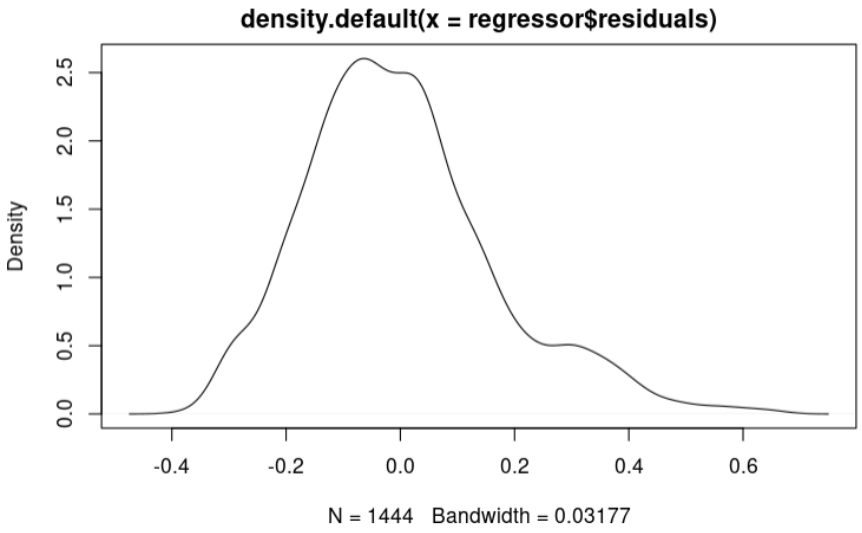
Što se tiče detekcije nepodobnih izlaznih vrijednosti (outliera), pomoću funkcije *rstudent*() smo pronašli sve reziduale koji imaju apsolutnu varijansu veću od 3 i takve podatke odbacili iz seta podataka prilikom izgradnje poboljšanog modela.

3.2 UNAPRJEĐENJE IZGRAĐENOG MODELA

S obzirom da pretpostavka linearnosti nije ispunjena, pokušali smo primijeniti različite nelinearne transformacije kako na izlazni tako i na ulazne atribute, međutim dobijali smo uglavnom lošije rezultate. Primijetili smo da linearnu vezu imaju logaritamske transformacije varijable *CustomerSuitability* i *TotalCharges* za podatke kod kojih je *CustomerSuitability* u određenom opsegu, međutim primjena tih transformacija nije poboljšala regresiju za sve podatke. Na osnovu oznaka za svaki od atributa iz funkcije *summary* pokušali smo poboljšati model korištenjem samo određenih atributa sa oznakom ***. Pokušali smo koristiti i min-max normalizaciju umjesto decimal scale, ali ni to nije dovelo do promjene rezultata. Na kraju smo najbolje rezultate dobili kada smo za ulazne varijable uzeli sve atribute osim *DailyCharges*, *MonthlyCharges*, *Dependents* i *gender*, bez ikakvih nelinearnih transformacija. Za ovakav model ostala je zadovoljena pretpostavka da ne postoji korelacija između ulaznih varijabli. Što se tiče plota Residuals vs Fitted, odnosno linearnosti rezidualnih vrijednosti, došlo je do određenog poboljšanja što se može vidjeti na sljedećoj slici.



Shapiro-Wilk test je ponovo pokazao da rezidualne vrijednosti ne formiraju normalnu distribuciju, a na sljedećoj slici je konačni grafik gustoće vjerovatnoće rezidualnih vrijednosti.



Vrijednosti metrika MAE, RMSE i R2 koje smo na kraju dobili primjenom ensamble bagging modela su na sljedećoj slici.

```
MAE: 0.1227325
RMSE: 0.1583695
R2: 0.817274
```

Ovdje je bilo očekivano da će se R2 poboljšati jer je poznato da R2 ima veću vrijednost kada imamo više nezavisnih varijabli. Međutim, odlučili smo da ovaj model bude naš finalni model jer su se poboljšale i metrike MAE i RMSE koje su pogodnije za poređenje tačnosti modela linearne regresije.

3.3 IZGRADNJA ENSAMBLE BAGGING MODELA

Prethodno su već prikazani rezultati dobijeni primjenom ensamble bagging-modela, a u nastavku će biti opisan način na koji je ovaj model izgrađen. U funkciji *bagging_regresija(podaci, k)* na početku se biraju podaci koji će predstavljati testni podskup. Zatim se u for petlji k puta vrši slučajni izbor trening-skupa i treniranje modela višestruke linearne regresije nad tim skupom. U istoj for-petlji se određuju i predikcije koje će dati svaki od modela za podatke iz testnog skupa. Te predikcije se čuvaju u listi. Na kraju se za svaku instancu testnog skupa računa srednja vrijednost varijable *CustomerSuitability* od vrijednosti koje su dali različiti modeli. Ove predikcije se upoređuju sa stvarnim vrijdnostima i ispisuju se metrike MAE, RMSE i R2.