Univerzitet u Novom Sadu

Fakultet tehničkih nauka

Dokumentacija za projektni zadatak

Student: Jelena Adamović, SV 6/2021

Predmet: Računarska inteligencija

Broj projektnog zadatka: 1

Tema projektnog zadatka: Analiza preživljavanja na Titaniku: Primena klasifikacionih algoritama na skupu podataka putnika

Opis problema

Projekat "Analiza preživljavanja na Titaniku" ima za cilj da primeni različite klasifikacione algoritme na poznatom skupu podataka putnika sa Titanika kako bi se predvidelo da li bi određeni putnik preživeo nesreću na osnovu određenih karakteristika. Skup podataka sadrži informacije o putnicima, uključujući njihove demografske podatke, informacije o brodu, kao i informacije o kartama.

Titanik je jedan od najpoznatijih brodoloma u istoriji, gde je preko 1500 ljudi izgubilo život. Korišćenje ovog skupa podataka omogućava nam da istražimo koje su karakteristike i faktori imali značajan uticaj na preživljavanje putnika. Ključni faktori koji se analiziraju uključuju starost, pol, klasa karte, broj članova porodice na brodu, cena karte, i mesto ukrcavanja.

Cilj ovog projekta je da se razviju i evaluiraju različiti klasifikacioni modeli koji mogu efikasno predvideti ishod (preživljavanje ili smrt) na osnovu dostupnih podataka. Očekivani rezultati uključuju identifikaciju najvažnijih karakteristika koje utiču na preživljavanje i izbor najboljeg modela za ovu vrstu predikcije.

Ovaj problem spada u domen binarne klasifikacije, gde je ciljna varijabla (preživljavanje) binarna (1 - preživeo, 0 - nije preživeo). Kroz analizu i modelovanje, nadamo se da ćemo dobiti uvid u faktore koji su doprineli preživljavanju i poboljšati naše razumevanje korišćenja klasifikacionih algoritama u praktičnim primenama.

Uvod

Brodska nesreća Titanika, koja se dogodila 15. aprila 1912. godine, ostavila je dubok trag u istoriji zbog ogromnog broja izgubljenih života i veličine tragedije. Na svom prvom putovanju iz Sautemptona u Njujork, Titanik je udario u ledeni breg i potonuo, što je rezultiralo smrću preko 1500 od ukupno 2224 putnika i članova posade. Ova tragedija je postala predmet mnogih istraživanja i analiza, ne samo zbog svog istorijskog značaja već i zbog bogatog skupa podataka o putnicima koji omogućava detaljnu analizu.

Projekat "Analiza preživljavanja na Titaniku" koristi dostupne podatke o putnicima kako bi istražio faktore koji su imali uticaj na preživljavanje. Skup podataka sadrži informacije kao što su starost, pol, klasa karte, broj članova porodice na brodu, cena karte, i mesto ukrcavanja. Analizom ovih podataka možemo otkriti obrasce i faktore koji su povećavali ili smanjivali šanse za preživljavanje.

Primena klasifikacionih algoritama na ovom skupu podataka omogućava nam da izgradimo modele koji mogu predvideti da li bi određeni putnik preživeo na osnovu poznatih karakteristika. Kroz korišćenje različitih algoritama, kao što su logistička regresija, decision trees, random forest, i drugi, možemo proceniti performanse svakog modela i odabrati najefikasniji za ovu vrstu predikcije.

Cilj ovog projekta je da pruži dublje razumevanje faktora koji su uticali na preživljavanje putnika na Titaniku i da pokaže primenu različitih klasifikacionih algoritama u rešavanju praktičnih problema. Na osnovu rezultata ove analize, možemo doneti zaključke o važnosti određenih karakteristika i unaprediti naše pristupe u oblasti mašinskog učenja i analize podataka.

Implementacija

Upoznavanje sa osobinama skupa podataka

Skup podataka koji je korišćen ima ukupno 891 unos. Svaki unos se sastoji iz nekoliko različitih kolona koje nam pružaju informacije neophodne za analizu za svakog putnika.

Skup podataka koji se koristi u projektu "Analiza preživljavanja na Titaniku" pruža detaljne informacije o putnicima broda. Ovaj skup podataka je često korišćen u zajednici za analizu podataka i mašinsko učenje zbog svoje bogatosti i istorijskog značaja. Podaci uključuju različite karakteristike koje se mogu koristiti za izgradnju prediktivnih modela. Evo glavnih osobina skupa podataka:

Osnovne karakteristike korišćenog skupa podataka su:

- **PassengerId:** Jedinstveni identifikator za svakog putnika.
- **Survived:** Ciljna varijabla koja označava da li je putnik preživeo (1) ili nije (0).
- **Pclass:** Klasa u kojoj je putnik putovao (1 = prva klasa, 2 = druga klasa, 3 = treća klasa).
- **Name:** Ime putnika.
- S**ex:** Pol putnika (“male” ili “female”).
- **Age:** Starost putnika u godinama. Neke vrednosti su nepoznate.
- **SibSp:** Broj braće i sestara ili supružnika koji su putovali sa putnikom.
- **Parch:** Broj roditelja ili dece koji su putovali sa putnikom.
- **Ticket:** Broj karte putnika.
- **Fare:** Cena karte putnika.
- **Cabin:** Broj kabine u kojoj je putnik bio smešten. Mnoge vrednosti su nepoznate.
- **Embarked:** Luka ukrcavanja putnika (C = Cherbourg, Q = Queenstown, S = Southampton).

U nastavku sledi analiza osnovnih karakteristika skupa podataka:

Survived: Ova binarna varijabla je ključna za našu analizu jer predstavlja ishod koji želimo da predvidimo. Procentualna raspodela preživelih i nastradalih putnika će nam pružiti prvi uvid u podatke.

Pclass: Klasa karte može biti indikativna za socijalno-ekonomski status putnika, što može imati značajan uticaj na šanse za preživljavanje. Analiza raspodele preživelih u različitim klasama pomoći će nam da identifikujemo ovu vezu.

Sex: Pol putnika je još jedan važan faktor. Istorijski podaci i pravilo "žene i deca prvo" sugerišu da bi pol mogao igrati ključnu ulogu u preživljavanju.

Age: Starost putnika može pružiti uvid u demografske aspekte preživljavanja. Analiza raspodele starosti i njene povezanosti sa ishodom preživljavanja pomoći će nam da razumemo značaj ovog faktora.

SibSp i Parch: Ove dve varijable nam govore o putnikovoj porodici na brodu. Moguće je da putnici koji putuju sa porodicom imaju veće ili manje šanse za preživljavanje, što ćemo istražiti kroz analizu ovih podataka.

Fare: Cena karte može biti povezana sa klasom putovanja, ali može pružiti i dodatne informacije o ekonomskoj situaciji putnika.

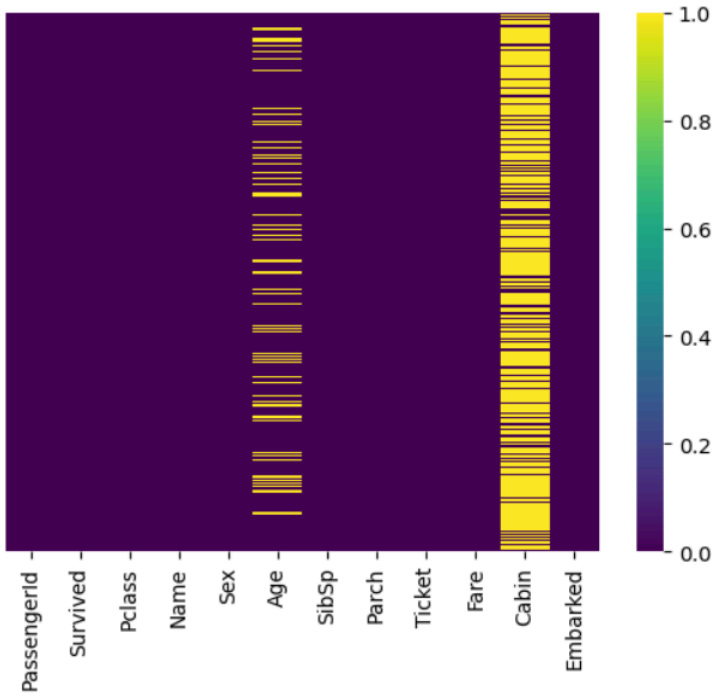
Cabin: Iako mnoge vrednosti nedostaju, analiza dostupnih podataka o kabinama može pružiti uvid u lokaciju putnika na brodu, što može biti značajno u kontekstu nesreće.

Embarked: Luka ukrcavanja može otkriti geografske ili društvene faktore koji utiču na preživljavanje.

Kroz detaljnu analizu ovih karakteristika, možemo dobiti uvid u ključne faktore koji utiču na preživljavanje putnika na Titaniku. Ovi uvidi će nam pomoći u izgradnji i evaluaciji klasifikacionih modela, te u konačnici doprineti boljem razumevanju ovog istorijskog događaja i primeni analitičkih tehnika.

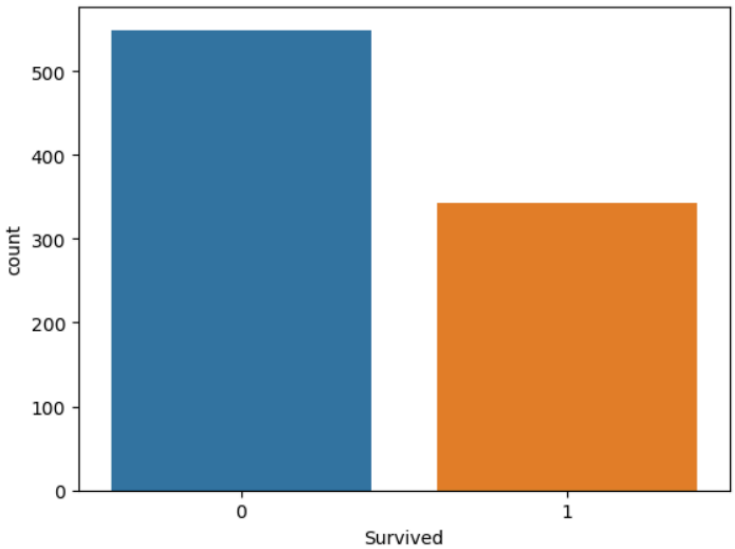
U nastavku ćemo prikazati nekoliko dijagrama koji će slikovito prikazati stanje podataka iz skupa podataka o putnicima na brodu. Za učitavanje podataka korišćena je *pandas* biblioteka, za rad sa numeričkim podacima korišćena je *NumPy* biblioteka, a za prikaz dijagrama korišćene su *matplotlib* i *seaborn* biblioteke.

Prvi dijagram koji ćemo prikazati je *heatmap* skupa podataka, gde se jasno može videti u kojim kolonama imamo nedostajuće podatke. Sa slike se jasno vidi da najviše podataka nedostaje u koloni *Cabin*, a nešto manja količina podataka nedostaje u koloni *Age*.



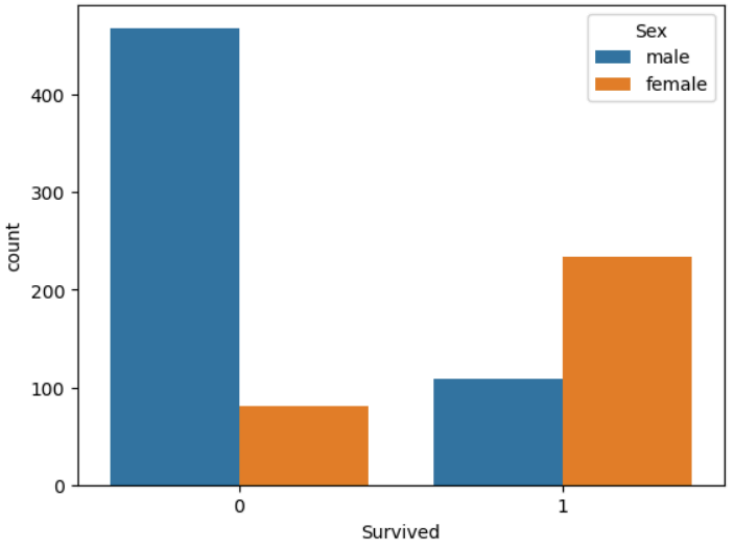
Slika 1. *Heatmap* skupa podataka sa jasno prikazanim nedostajućim vrednostima

Drugi dijagram koji ćemo prikazati predstavlja odnos putnika koji su preživeli i onih koji, nažalost, nisu. Vrednost 1 označava da je putnik preživeo, a vrednost 0 označava da putnik nije preživeo.



Slika 2. Prikaz putnika u zavisnosti od toga da li su preživeli ili ne

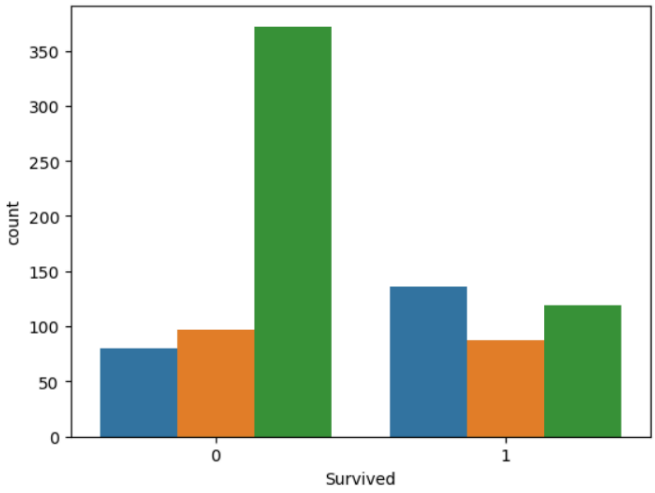
Sledeći dijagram predstavlja prikaz putnika koji su preživeli ili nisu preživeli, u zavisnosti od pola. Informacija o polu putnika mam je bitna zbog toga što se zbog pravila da se spašavaju prvo žene i deca, moglo desiti da je više žena preživelo.



Slika 3. Prikaz putnika koji su preživeli ili ne, u zavisnosti od pola

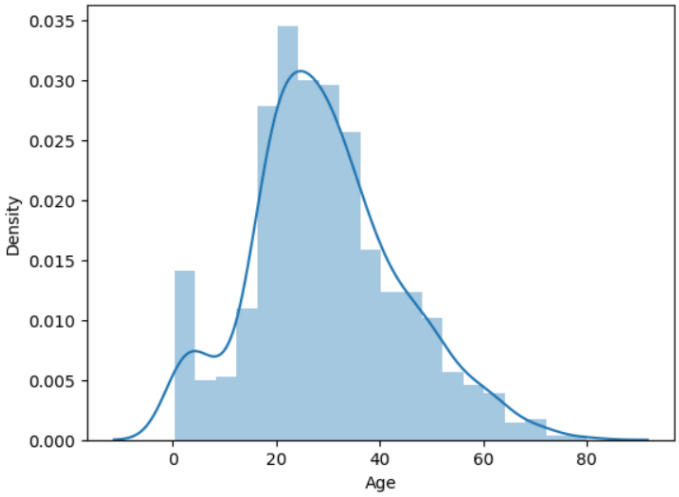
Na dijagramu možemo videti da je veći broj preživelih putnika bio ženskog pola, što može biti uslovljeno poštovanjem pravila o prioritetu putnika za spašavanje (prvo žene i deca).

Na sledećem dijagramu ćemo prikazati broj preživelih putnika u zavisnosti od klase u kojoj su putovali. Klasa u kojoj su putnici putovali je povezana sa lokacijom na brodu gde su njihove kabine bile smeštene, što je imalo uticaj pri akciji spašavanja.

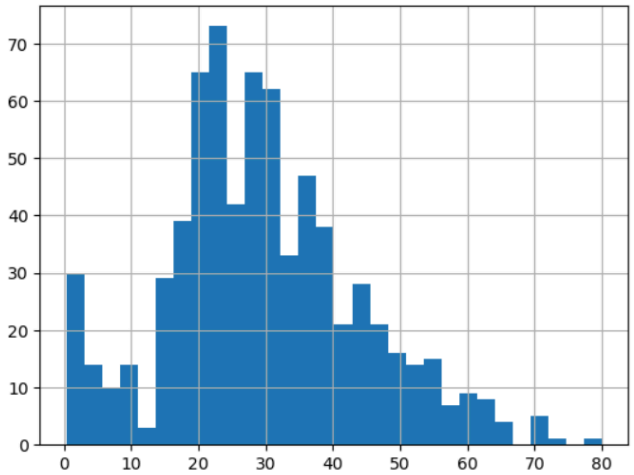


Slika 4. *Countplot* putnika u zavisnosti od klase u kojoj su putovali

Na sledećem dijagramu ćemo prikazati raspodelu putnika po starosti. Na osnovu dijagrama možemo zaključiti da je najveći broj putnika imao oko 25 godina. Starost putnika nam je značajna zbog toga što su deca imala prioritet pri spašavanju, kao i zbog toga što su mlađi putnici fizički sposobniji i to bi moglo da utiče na njihove šanse u preživljavanju.

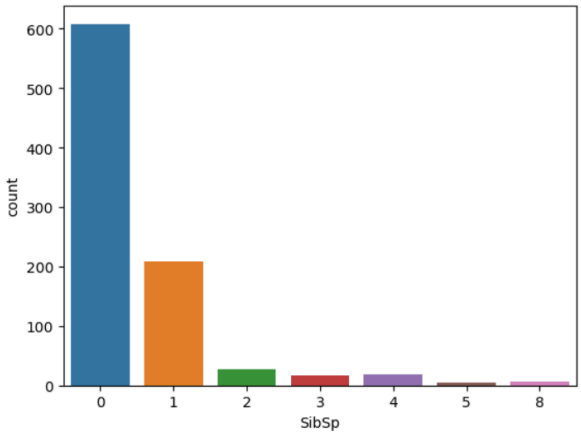


Slika 5. *Distplot* putnika po starosti



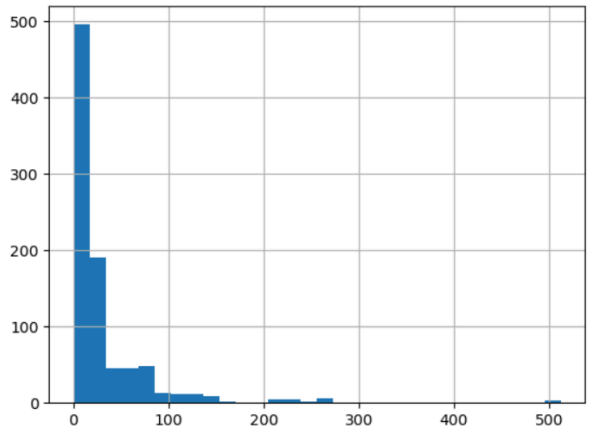
Slika 6. *Histogram* putnika po starosti

U sledećem dijagramu, prikazaćemo stanje putnika na osnovu broja braće, sestara i supružnika koji su putovali sa njima. Ovaj podatak može biti značajan jer su porodice sa malom decom imale prioritet pri spašavanju, ali isto tako u nekim situacijama, porodice nisu želele da se razdvajaju i neki su odlučili da ostanu na brodu uprkos tome što je to značilo da će poginuti. Iz dijagrama zaključujemo da najveći broj putnika nije imao rođake ili supružnike na brodu, a većina onih koji su imali, imali su jednog rođaka/supružnika.



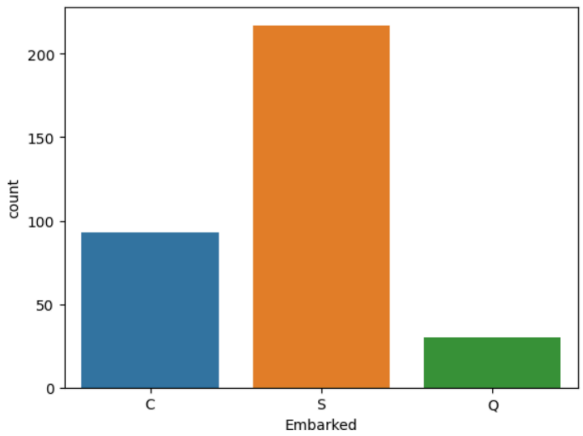
Slika 7. Prikaz putnika na osnovu broja braće, sestara i supružnika koji su putovali sa njima

Na sledećem dijagramu, prikazaćemo putnike u zavisnosti od toga koliko su platili kartu za putovanje. Ova informacija značajna nam je zbog toga što je cena karte blisko povezana sa klasom u kojoj su putnici putovali, što je kasnije moglo da utiče na njihov prioritet pri spašavanju jer su putnici iz viših klasa imali veći prioritet pri spašavanju u odnosu na putnike iz nižih klasa.



Slika 8. Prikaz putnika na osnovu cene karte koju su platili za putovanje

Na sledećem dijagramu, izvršićemo prikaz putnika u zavisnosti od luke u kojoj su se ukrcali na brod. Ovaj podatak je značajan zato što su, na primer, putnici koji su se ukrcali u luci Sautempton imali više vremena da se upoznaju sa brodom i načinima spasavanja, što je moglo uticati na njihovu pripremljenost u akciji spasavanja. Sa dijagrama vidimo da se najveći broj putnika i ukrcao u luci Sautempton.

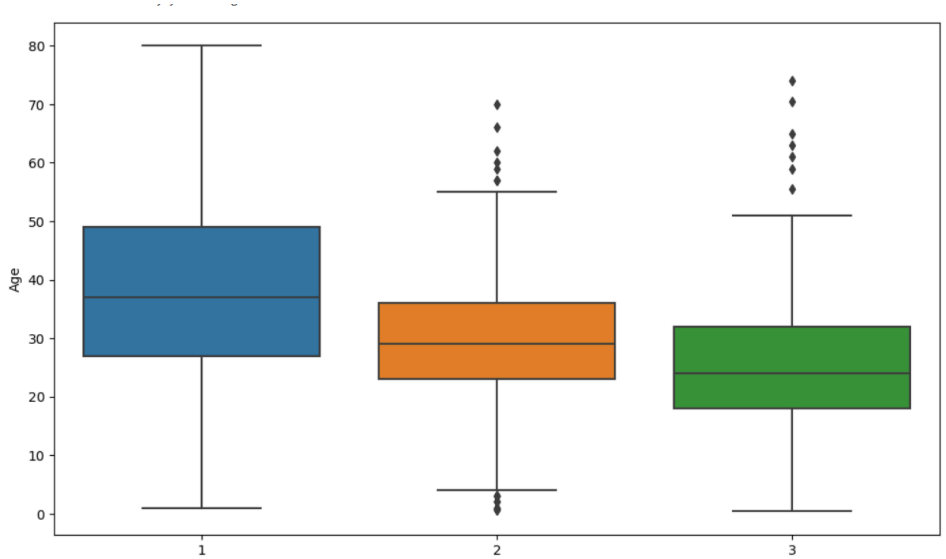


Slika 9. Prikaz putnika na osnovu luke u kojoj su se ukrcali

Priprema podataka za analizu

Kako bi analiza protekla što lakše i efikasnije, neophodno je da podatke pripremimo i dovedemo u stanje koje nam je najpogodnije za analizu. Kao što smo videli na *heatmap*-u podataka, vidimo da u koloni *Cabin* imamo najveći broj podataka koji nedostaju, a nešto manji broj podataka nedostaje u koloni *Age*. Pošto nam nedostajući podaci nisu korisni za analizu, moramo ih se nekako rešiti, odnosno skup podataka moramo dovesti u stanje gde nema nijednog nedostajućeg podatka.

Prvo ćemo rešiti problem nedostajućih vrednosti u koloni *Age*. Pošto sa *heatmap*-a skupa podataka zaključujemo da u koloni *Age* nedostaje nešto manji broj podataka u odnosu na kolonu *Cabin*, a i kolona *Age* nam je bitna za analizu, odlučujemo da podatke u koloni *Age* dopunimo. Za to će nam biti potreban *boxplot* podataka o starosti putnika (kolona *Age*) na osnovu klase u kojoj su putovali (kolona *Pclass*).

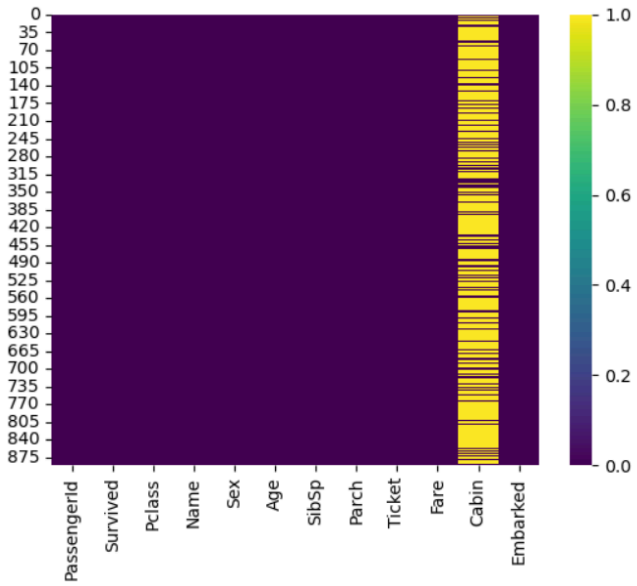


Slika 10. *Boxplot* putnika na osnovu podataka o njihovoj starosti i klasi u kojoj su putovali

Ovde nam je posebno značajna osobina *boxplot*-a da se na njemu jasno može videti prosečna vrednost članova određenog skupa. Koristeći tu osobinu, zaključujemo da je prosečna starost putnika u prvoj klasi bila 37 godina, u drugoj klasi 29 godina, a u trećoj 25 godina. Ovo nas dovodi do činjenice da su stariji putnici u proseku bili boljeg ekonomskog stanja u odnosu na svoje mlađe saputnike.

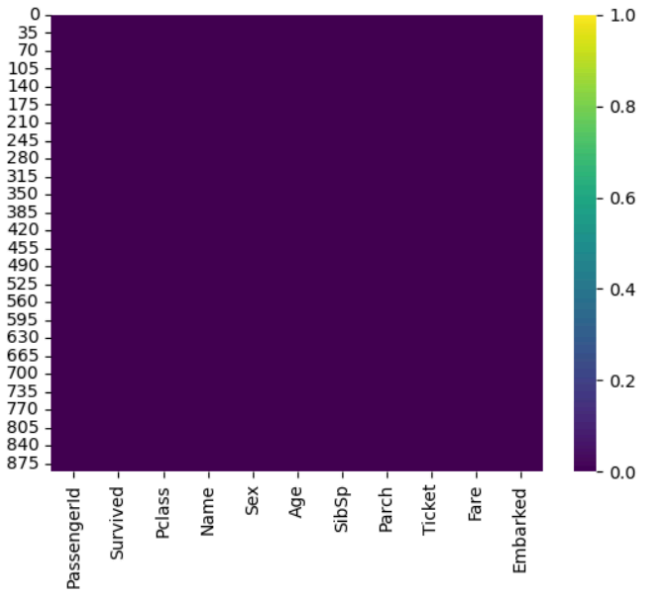
Popunjavanje nedostajućih podataka vršimo na sledeći način: prolazimo kroz listu svih putnika i za svakog od njih uzima njihovu starost, odnosno vrednost koja je smeštena u koloni *Age*, a zatim uzima vrednost klase u kojoj su putovali i koja je smeštena u koloni *Pclass*. Ukoliko se utvrdi da je kod nekog putnika vrednost u koloni *Age* nepostojeća (jednaka je sa *None*), uzimamo vrednost njegove klase i na osnovu toga za starost mu postavljamo prosečnu vrednost godina za njegovu klasu, u suprotnom samo vratimo vrednost koja je smeštena u koloni *Age*.

Znači, ako utvrdimo da putniku nedostaje vrednost u koloni *Age* i putovao je u prvoj klasi, u kolonu *Age* ćemo postaviti vrednost 37, ako je putovao u drugoj klasi 29 godina i ako je putovao u trećoj klasi – 25 godina. Nakon izvršene obrade celokupnog skupa podataka na ovaj način, više nećemo imati nijednu nedostajuću vrednost u koloni *Age*, što se može videti i na *heatmap*-u skupa podataka.



Slika 11. Prikaz *heatmap*-a skupa podataka nakon izvršenog dopunjavanja podataka u koloni *Age*

Međutim, i dalje imamo veliki broj nedostajućih podataka u koloni *Cabin*. Zbog velikog broja podataka koji nedostaju, a samog njihovog oblika, jer su stringovi, bilo bi nam nezgodno da ih dopunimo kao što smo to učinili u koloni *Age*. Takođe, vrednosti u koloni *Cabin* nam nisu preterano bitne za dalju analizu, tako da nam se kao najlogičniji način za rešavanje problema nedostajućih vrednosti u koloni *Cabin* nameće njeno potpuno uklanjanje iz skupa podataka. Nakon obavljene operacije uklanjanja kolone *Cabin* iz skupa podataka, prikaz nedostajućih vrednosti izgledaće ovako:



Slika 12. Prikaz skupa podataka nakon izvršene operacije uklanjanja kolone *Cabin*

Ovde ne završavamo sa pripremom podataka za analizu jer imamo još nekoliko problema koje moramo rešiti pre nego što pređemo na glavni deo naše implementacije, odnosno samu analizu. Pre svega, imena putnika, koja su smeštena u koloni *Name,* nam nisu podaci koji su nam bitni za našu analizu, pa i tu možemo upotrebiti potpuno uklanjanje kolone iz skupa podataka, kao što smo učinili i sa kolonom *Cabin*.

Sličnu situaciju imamo i kod kolone *Ticket* koja predstavlja broj karte putnika, pa ćemo i nju potpuno ukloniti iz skupa podataka.

Nešto drugačiji problem imamo kod kolone *Sex* koja predstavlja pol putnika. Trenutno nam se u ovoj koloni podaci čuvaju u obliku stringova "male" i "female", a kako nam to nije pogodno za analizu jer se stringovi posmatraju kao tip podataka *object*, primenićemo tehniku koja se naziva *one-hot encoding*.

One-hot encoding predstavlja tehniku mapiranja vrednosti tipa *object* na binarne vrednosti koje su nam pogodne za analizu, a u ovom slučaju čemo iskoristiti brojčane vrednosti tipa *Integer*. Naime, pošto u koloni *Sex* imamo samo dve moguće vrednosti – "male" i "female", vrednosti ćemo namapirati na brojeve 0 i 1, gde će 0 zamenjivati sve "male" vrednosti, a 1 sve "female" vrednosti.

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	male	22.0	1	0	7.2500	S
1	2	1	1	female	38.0	1	0	71.2833	C
2	3	1	3	female	26.0	0	0	7.9250	S
3	4	1	1	female	35.0	1	0	53.1000	S
4	5	0	3	male	35.0	0	0	8.0500	S

Slika 13. Prikaz skupa podataka pre izvršenog *one hot encoding*-a za kolonu *Sex*

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	0	22.0	1	0	7.2500	S
1	2	1	1	1	38.0	1	0	71.2833	C
2	3	1	3	1	26.0	0	0	7.9250	S
3	4	1	1	1	35.0	1	0	53.1000	S
4	5	0	3	0	35.0	0	0	8.0500	S

Slika 14. Prikaz skupa podataka nakon izvršenog *one-hot encoding*-a za kolonu *Sex*

Sličan problem imamo i u koloni *Embarked*, gde se stringovima označava luka u kojoj su se putnici ukrcali. Međutim, pošto ovde imamo 3 moguće vrednosti – C=Cherbourg, S=Southampton i Q=Queenstown, ne možemo primeniti one-hot encoding jer ne možemo 3 vrednosti namapirati na neku binarnu vrednost, iskoristićemo tehniku *label-encoding*. Za vršenje *label encoding*-a iskoristićemo funkciju *get_dummies* iz biblioteke *pandas*. Kada izvršimo *label encoding* za kolonu *Embarked*, umesto jedne kolone *Embarked* sa vrednostima tipa *string (object)*, dobićemo tri kolone – *Embarked_C, Embarked_S i Embarked_Q* u kojima su smeštene *boolean* vrednosti. U koloni *Embarked_C* imaćemo *True* vrednost ukoliko se putnik ukrcao u luci *Cherbourg*, a u ostalim slučajevima *False*. Slična situacija je i u koloni *Embarked_S* gde imamo *True* vrednost ako se putnik ukrcao u luci *Southampton*, a u ostalim slučajevima *False* vrednost. Na posletku, u koloni *Embarked_Q* imamo *True* vrednost ako se putnik ukrcao u luci *Queenstown*, a u ostalim slučajevima imamo vrednost *False*. Nakon izvršenog *label encoding*-a za kolonu *Embarked*, skup podataka će izgledati ovako:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked_C	Embarked_Q	Embarked_S
0	1	0	3	0	22.0	1	0	7.2500	False	False	True
1	2	1	1	1	38.0	1	0	71.2833	True	False	False
2	3	1	3	1	26.0	0	0	7.9250	False	False	True
3	4	1	1	1	35.0	1	0	53.1000	False	False	True
4	5	0	3	0	35.0	0	0	8.0500	False	False	True

Slika 15. Prikaz skupa podataka nakon izvršenog *label encoding*-a za kolonu *Embarked*

Poslednja priprema podataka pred analizu je pravljenje kopije skupa podataka i njeno smeštanje u promenljivu *X*, a zatim uklanjanje kolone *Survived* iz promenljive *X* pošto ćemo podatke iz promenljive *X* koristiti za treniranje modela, a model ne sme pri treniranju da vidi stvarne vrednosti ciljne kolone *Survived*.

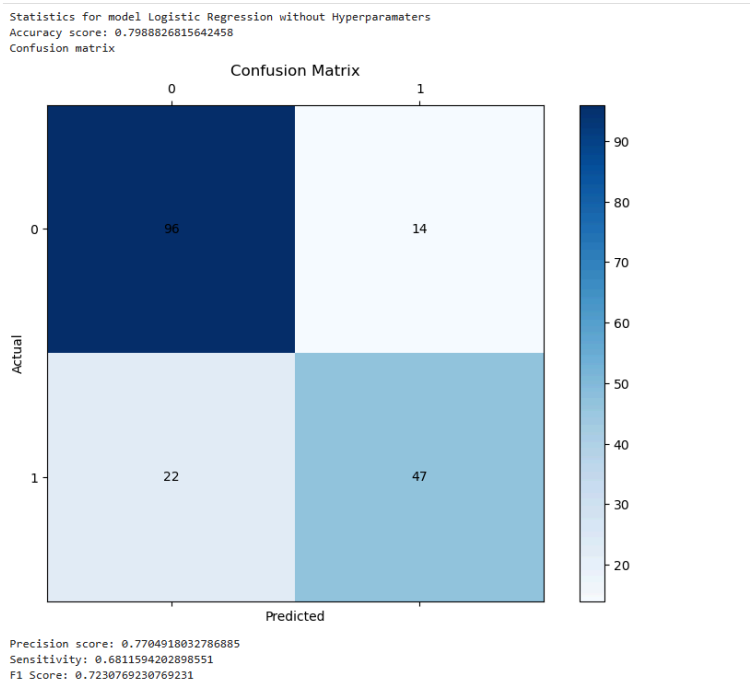
Treniranje modela

Prvo što radimo je skaliranje podataka u promenljivoj *X*. Za tu svrhu iskoristićemo dva različita načina skaliranja – *StandardScaler* i *MinMaxScaler* i kasnije ćemo upoređivati rezultate dobijene korišćenjem ova dva *scaler*-a.

Zatim, sledeći korak je podela skupa podataka na *train* i *test* podatke. Za tu namenu koristimo funkciju *train_test_split* i delimo podatke u razmeri 80% za *train* podatke i 20% za *test* podatke.

U ovoj analizi koristićemo nekoliko različitih modela, a to su *LogisticRegression, NaiveBayes, SVM, RandomForestClassifier* i *KNNClassifier*.

Za svaki od ovih modela vršimo predikcije i izračunavamo matricu konfuzije, *accuracy score, precision score, recall score i F1 score*.



Slika 16. Primer *LogisticRegression* bez hiperparametara

Kod **LogisticRegression** sa *StandardScaler*-om i bez hiperparametara dobijamo sledeće rezultate: accuracy score – 79.88%, precision score – 77.04%, sensitivity – 68.11%, F1 Score – 72.3%.

Kod **LogisticRegression** sa *MinMaxScaler*-om i bez hiperparametara dobijamo sledeće rezultate: accuracy score – 79.88%, precision score – 66.23%, sensitivity – 83.6%, F1 Score – 73.91%.

Kod **NaiveBayes**-a sa *StandardScalerom* bez hiperparametara dobijamo sledeće rezultate: accuracy score – 77.09%, precision score 78%, sensitivity 56.52%, F1 Score – 65.42%.

Kod **SVC-a** je korišćen *linear* kernel, a *random_state* je postavljen na 42. Kod SVC-a sa *StandardScaler*-om dobijamo sledeće rezultate: accuracy score – 78.21%, precision score – 74.19%, sensitivity – 66.67%, F1 Score – 70.22%.

Sledeći model koji je korišćen je **RandomForestClassifier (RFC)**, sa postavljenim parametrima *n_estimators* na vrednost 100 i *random_state* na vrednost 42. Upotrebom RFC-a sa StandardScaler-om

dobijeni su sledeći rezultati: accuracy score – 82.68%, precision score – 82.75%, sensitivity – 69.56%, F1 Score – 75.59%.

Kod **KNeighborsClassifier-a** sa *n_neigbors=5* i *StandardScaler-om* dobijamo sledeće rezultate – accuracy score – 78.21%, precision score – 75.86%, sensitivity – 63.76%, F1 Score – 69.29%.

Kod **NaiveBayes-a** sa *MinMaxScaler*-om dobijeni su sledeći rezultati: accuraccy score – 75.98%, precision score – 78.33%, sensitivity – 61.03%, F1 Score – 68.61%.

Kod **SVM-a** sa linearnim kernelom, *random_state=42* i *MinMaxScaler*-om dobijeni su sledeći rezultati: accuracy score – 76.53, precision score – 78.69%, sensitivity – 62.33%, F1 Score – 69.56%.

Kod **RandomForestClassifier-a (RFC)** sa *n_estimators=100, random_state=42* i *MinMaxScaler*-om dobijeni su sledeći rezultati: accuracy score – 75.98%, precision score – 78.33%, sensitivity – 61.03%, F1 Score – 68.61%.

Kod **KNeighborsClassifier-a** sa *n_neighbors=5* i *MinMaxScaler*-om dobijamo sledeće rezultate – accuracy score – 75.98%, precision score – 78.33%, sensitivity – 61.03% i F1 Score – 68.61%.

Zatim prelazimo na kreiranje i treniranje neuronskih mreža različitih nivoa složenosti, počevši od najjednostavnije ka najkomplikovanjoj. Naravno, i ovde ćemo koristiti i podatke iz *StandardScaler*-a i podatke iz *MinMaxScaler*-a i upoređivati dobijene rezultate. Za pravljenje neuronskih mreža korišćena je biblioteka *Keras* koja je deo *TensorFlow*-a za binarnu klasifikaciju. Koristimo *Sequential* za pravljenje linearnog steka slojeva, *Dense* za kreiranje potpuno povezanih slojeva neurona, *input_shape* za definisanje oblika ulaznog podatka u prvom sloju, ovde koristimo broj ulaznih podataka. Kao aktivacione funkcije koristimo *relu* (*Rectified Linear Unit*) koja se koristi u skrivenim slojejvima za dodavanje nelinearnosti i funkciju *sigmoid* koja se koristi u izlaznom sloju za binarnu klasifikaciju jer kompresuje izlaz u raspon [0, 1]. Kao *optimizer* iskoristićemo *adam* koji se često koristi zbog svojih adaptivnih mogućnosti učenja. Kao funkciju gubitka (*loss*) koristimo *binary_crossentropy* koja se koristi za binarne klasifikacijske probleme, a kao metriku koristimo tačnost (accuracy).

Kod jednostavne troslojne neuronske mreže koristimo *relu* aktivacionu funkciju u prva dva sloja, a u poslednjem funkciju *sigmoid*. Prvi sloj ima 16 jedinica, drugi 8 i poslednji 1. Broj epoha je postavljen na 100, a batch_size na 32. Koristimo podatke iz standard scaler-a i dobijamo sledeće rezultate: accuraccy score – 80.44%, precision score – 84%, sensitivity – 60.87%, F1 Score – 70.58%.

Sledeća neuronska mreža ima iste osobine kao prethodna, jedina razlika je ta što koristimo podatke iz *MinMaxScaler*-a i dobijamo sledeće rezultate: accuracy score – 78.77%, precision score – 66.23%, sensitivity – 80.95% i F1 Score – 72.85%.

Sledeća neuronska mreža je nešto komplikovanija, ima 4 sloja. U prvom sloju imamo 128 jedinica, u drugom 64 jedinice, u trećem 8 jedinica i u četvrtom jednu jedinicu. U prva 3 sloja koristimo *relu* akrivacionu funkciju, a u poslednjem *sigmoid* aktivacionu funkciju. Optimizer ostaje *adam*, kao i broj epoha koji ostaje 100 i batch size koji ostaje 32. Koristimo podatke skalirane *StandardScaler*-om i

dobijamo sledeće rezultate: accuracy score – 79.33%, precision score – 75%, sensitivity – 69.56%, F1 Score – 72.18%.

Naredna neuronska mreža je još komplikovanija, ima 5 slojeva, u prva 4 sloja korišćena je *relu* aktivaciona funkcija, a u poslednjem *sigmoid* funkcija. Prvi sloj ima 256 jedinica, drugi 128, treći 64, četvrti 8 i peti jednu jedinicu. Optimizer je i dalje *adam*, broj epoha je 100, a batch_size je 32. Koristeći podatke iz *StandardScaler*-a dobili smo sledeće rezultate: accuracy score – 78.21%, precision score – 65.15%, sensitivity – 72.88%, F1 Score – 68.8%.

Naredna neuronska mreža je još komplikovanija, ima 7 slojeva, u prvih 6 slojeva je korišćena *relu* aktivaciona funkcija, u poslednjem *sigmoid*. Prvi sloj ima 512 jedinica, poslednji jednu jedinicu. Korišćeni optimizer je *adam*, broj epoha iznosi 100, a batch size 32 i dobijeni rezultati su: accuracy score – 73.18%, precision score – 63.63%, sensitivity – 63.63%, F1 Score – 63.63%.

U narednoj neuronskoj mreži imamo 9 slojeva, gde prvi ima 1024 jedinice, a poslednji jednu. U prvih 8 slojeva koristi se *relu* aktivaciona funkcija, a u poslednjem sigmoid funkcija. Vrednosti optimizera, broja epoha i batch size ostaju iste kao u prethodnim mrežama. Dobijeni rezultati su: accuracy score – 79.33%, precision score – 76.63%, sensitivity – 63.63%, F1 Score – 69.42%.

U narednoj neuronskoj mreži imamo 11 slojeva, gde prvi ima 4096 jedinica, a poslednji jednu jedinicu. U prvih 10 slojeva koristimo *relu* aktivacionu funkciju, a u poslednjem *sigmoid* aktivacionu funkciju. Optimizer, broj epoha i batch size ostaju isti kao u prethodnim mrežama. Dobijeni rezultati su: accuracy score – 76.53%, precision score – 74.24%, sensitivity – 66.21%.

Potraga za najboljim modelom

Za svaki od korišćenih modela (*LogisticRegression, NaiveBayes, SVM, RandomForestClassifier, KNeighborsClassifier* i neuronske mreže) izvršili smo pretragu najboljih parametara koristeći *GridSearchCV* iz biblioteke *scikit-learn*. U nastavku ćemo predstaviti rezultate svake od pretraga i uporediti dobijene rezultate.

Prvi model za koji ćemo uraditi pretragu najboljih hiperparametara upotrebom *GridSearchCV* je **LogisticRegression**. Prvo što radimo je postavljanje odgovarajućih parametara. Penalty – tip regularizacije koji ćemo koristiti i ovde smo odabrali *l1, l2, elasticnet* i *none*. Sledeći parametar je *C* koji predstavlja inverznu regulacijsku snagu, gde manje vrednosti impliciraju jaču regularizaciju – ovde smo odabrali vrednosti 0.001, 0.01, 0.1, 1, 10 i 100. Zatim postavljamo parametar *solver* koji predstavlja algoritme optimizacije za nalaženje koeficijenata i tu smo odabrali sledeće algoritme: *lbfgs, liblinear i saga*. Poslednji parametar koji postavljamo je *max_iter* koji predstavlja maksimalan broj iteracija za konvergenciju i ovde biramo vrednosti 100, 200 i 300. Zatim kreiramo osnovni model logističke regresije. Sledeći korak je inicijalizacija GridSearchCV alata za pretragu mreže hiperparametara. Ovde postavljamo sledeće parametre:

- estimator=logistic_regression – predstavlja model koji koristimo
- Param_grid – definisani parametarski prostor (parametri koje smo gore postavili)

- Cv – korišćenje cross validation za procenu performansi modela (ovde je postavljen na 5, te stoga koristimo 5-fold cross validation)
- Verbose – prikazuje napredak pretrage (postavljen je na 1)
- N_jobs – korišćenje svih dostupnih procesora za ubrzanje pretrage

Zatim fitujemo grid search na podatke (ovde koristimo *train* podatke iz StandardScaler-a). Ovo koristimo da bismo pronašli najbolje parametre koji daju najbolju cross-validation tačnost. Zatim izdvajamo najbolje pronađene parametre i najbolji cross-validation rezultat. Izdvojene najbolje parametre koristimo za kreiranje novog *LogisticRegression* modela i fitovanje tog modela na trening podatke. Zatim koristimo taj model za pravljenje predikcija nad test podacima i na kraju ispisujemo dobijene rezultate. U ovom slučaju dobili smo sledeće rezultate:

- Accuracy score – 79.32%
- Precision score – 77.59%
- Sensitivity – 65.21%
- F1 Score – 70.87%.

Za isti model, al uz upotrebu podataka skaliranih MinMaxScaler-om dobijamo sledeće rezultate:

- Accuracy score – 77.09%
- Precision score – 81.03%
- Sensitivity – 61.03%
- F1 Score – 69.62%.

Sledeći model za koji ćemo izvršiti pretragu mreže hiperparametara je *RandomForestClassifier (RFC)*. Koristimo podatke skalirane *StandardScaler*-om. Postavljamo sledeće parametre:

- N_estimators – broj stabala u šumi. Više stabala može poboljšati performanse modela, ali takođe povećava vreme trajanja treninga. U ovom slučaju testiraćemo 3 različite vrednosti za broj stabala – 100, 200 i 300.
- Max_depth – maksimalna dubina svakog stabla. Kontroliše koliko duboko svako stablo može ići. Ograničavanje dubine može pomoći u sprečavanju prenaučenosti (overfitting). Vrednosti koje smo odabrali su [‘None’, 10, 20, 30], gde None znači da stabla rastu sve dok svi čvorovi ne sadrže manje od min_samples_split uzoraka, dok druge vrednosti ograničavaju dubinu stabala.
- Min_samples_split – minimalan broj uzoraka potreban da se čvor podeli. Veće vrednosti mogu pomoći u sprećavanju prenaučenosti smanjenjem složenosti modela. Ovde smo odabrali 3 vrednosti [2, 5, 10].
- Min_samples_leaf – minimalan broj uzoraka potrebnih da bude list (čvor bez dece). Veće vrednosti mogu sprečiti model da nauči previše specifičnosti iz trening podataka. Ovde smo odabrali vrednosti [1, 2, 4].
- Bootstrap – da li se koristi bootstrap za uzorkovanje prilikom izgradnje stabala. Bootstrap uzorkovanje može pomoći u smanjenju varijanse modela. Ovde smo odabrali vrednosti [True, False], što znači da testiramo i korišćenje bootstrap uzorkovanja i nekorišćenje bootstrap uzorkovanja.

Rezultati koje dobijamo su:

- Accuracy score – 79.88%
- Precision score – 77.04%
- Sensitivity – 68.11%

- F1 Score – 72.3%.

Za isti model, samo pri korišćenju podataka skaliranih *MinMaxScaler*-om, dobijamo sledeće rezultate:

- Accuracy score – 78.77%
- Precision score – 80.95%
- Sensitivity – 66.23%
- F1 Score – 72.85%

Naredni model za koji ćemo izvršiti pretragu mreže hiperparametara je KNeigborsClassifier. Koristimo podatke skalirane *StandardScaler*-om. Definišemo parametarski prostor:

- N_neighbors – broj suseda koji će se koristiti za predikciju. Izabrali smo vrednosti [3, 5, 7, 9, 11] i testiramo različite vrednosti za broj najbližih suseda. Manje vrednosti mogu biti osetljivije na buku, dok veće vrednosti mogu dati glatkije granice odlučivanja.
- Weights – funkcija težina koja će se koristiti:
    - Uniform – svi susedi imaju istu važnost (težinu)
    - Distance – susedi koji su bliži imaju veću težinu, tj. njihovi glasovi se više uvažavaju.
- Metric – mera udaljenosti koja će se koristiti za pronalaženje najbližih suseda:
    - Euclidean – euklidska udaljenost, standardna L2 norma
    - Manhattan – menhetn udaljenost, poznata kao L1 norma
    - Minkowski – minkovskijeva udaljenost, generalizacija euklidske i Menhetn udaljenosti. Za različite vrednosti parametra *p* može se koristiti euklidska (p=2) ili Menhetn (p=1) udaljenost.

Rezultati koje smo dobili su:

- Accuracy score – 78.77%
- Precision score – 80.39%
- Sensitivity – 59.42%
- F1 score – 68.33%.

Za isti model, samo sa upotrebom podataka skaliranih MinMaxScaler-om dobijamo sledeće rezultate:

- Accuracy score – 77.65%
- Precision score – 79.36%
- Sensitivity – 64.93%
- F1 Score – 71.42%.

Naredni model za koji ćemo izvršiti pretragu mreže hiperparametara je NaiveBayes. Koristimo podatke skalirane *StandardScaler*-om. Definišemo parametarski prostor:

- Var_smoothing – Hiperparametar koji dodaje malu količinu varijanse u podatke kako bi se izbeglo deljenje sa nulom. Ovaj parametar omogućuje modelu da bude stabilniji i robusniji, posebno kod malih uzoraka ili uzoraka sa nultom varijansom.
    - Odabrali smo vrednost *np.logspace(0, -9, num=100)* – generiše 100 vrednosti logaritamski raspoređenih između 10^0 (1) i 10^{-9}. Logaritamska raspodela omogućava testiranje vrednosti koje pokrivaju širok raspon skala, od velikih 10^0 (1) do veoma malih 10^{-9}.

Rezultati koje smo dobili su:

- Accuracy score – 74.86%
- Precision score – 80%
- Sensitivity – 46.37%
- F1 score – 58.71%.

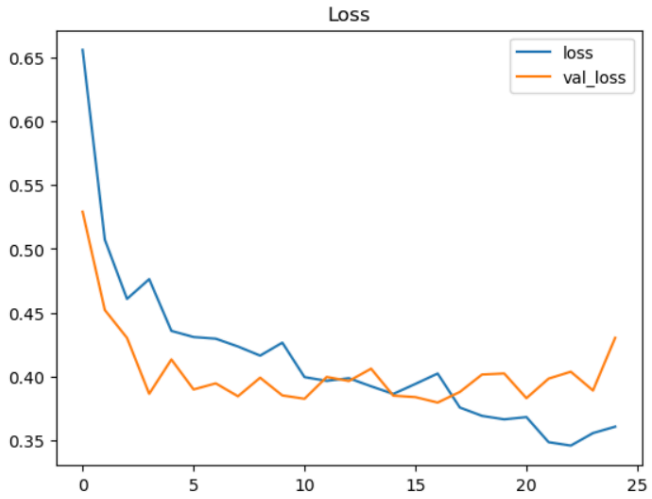
Za isti model, samo sa upotrebom podataka skaliranih MinMaxScaler-om, dobijamo sledeće rezultate:

- Accuracy Score – 77.65%
- Precision score – 81.35%
- Sensitivity – 62.33%
- F1 score – 70.58%.

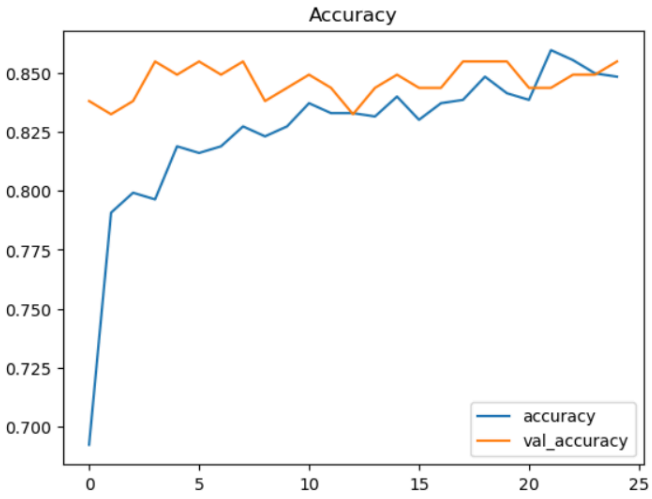
Upotreba PCA za smanjenje dimenzionalnosti podataka

Pored ovih modela, koristili smo i Principal Component Analysis (PCA) kao tehniku za smanjenje dimenzionalnosti podataka. PCA je primenjen kako bi se optimizovala efikasnost modela i poboljšala interpretabilnost, bez značajnog gubitka informacija. PCA omogućava smanjenje broja karakteristika (feature-a) koje se koriste za treniranje modela, što može smanjiti složenost modela i vreme treniranja. PCA pomaže u uklanjanju redundantnih i kolinearnosti karakteristika, što može poboljšati performanse modela.

Prvo smo na primeru jedne neuronske mreže prikazali dijagrame vrednosti funkcije gubitka i tačnosti iz istorije treniranja za svaku epohu. Neuronska mreža koju smo koristili u primeru ima 11 slojeva sa promenljivim brojem slojeva koji nisu u stalnom rastu ili opadanju. Optimizer je i dalje adam, a u privih 10 slojeva koristi se relu aktivaciona funkcija, a u poslednjem sloju sigmoid funkcija. U funkciji *plot_metrics* uzimamo odgovarajuce parametre iz *history*-ja uz pomoć *Keras* biblioteke i iscrtavamo grafike funkcije gubitka (loss) i tačnosti (accuracy). Podaci iz history-ja se odnose na period pre primene PCA.

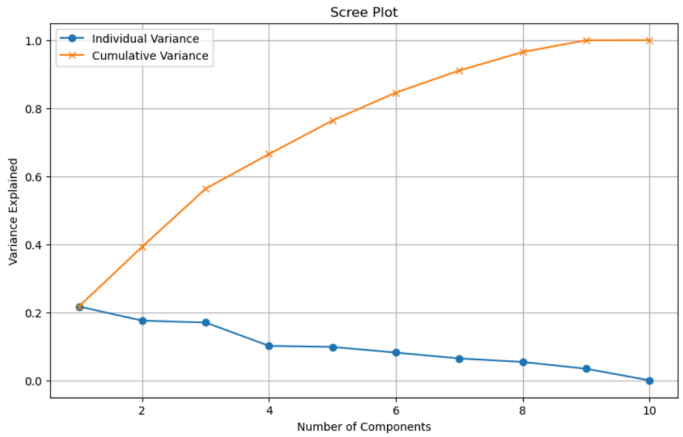


Slika 17. Grafik *loss* funkcije



Slika 18. Grafik tačnosti (accuracy)

Sada koristimo PCA da odredimo optimalan broj komponenti uz upotrebu individualne i kumulativne varijanse i iscrtavamo *Scree* plot. Ovde koristimo podatke dobijene skaliranjem *StandardScaler*-om.



Slika 19. *Scree plot* optimalnog broja komponenti uz upotrebu *StandardScaler*-a

Na osnovu grafika zaključujemo da optimalni broj komponenti iznosi 8. Taj broj ćemo iskoristiti za primenu PCA nad podacima skaliranim *StandardScaler*-om. Zatim ćemo za svaki od modela koji smo i pre koristili upotrebiti podatke dobijene nakon primene PCA i uporediti dobijene rezultate.

Kod jednostavne neuronske mreže sa 3 sloja, dobili smo sledeće rezultate:

- Accuracy score – 79.33%
- Precision score – 58.73%
- Sensitivity – 77.08%
- F1 Score – 66.66%

Kod nešto komplikovanije neuronske mreže sa 5 slojeva i istim aktivacionim funkcijama, dobijeni su sledeći rezultati:

- Accuracy score – 80.45%
- Precision score – 70.89%
- Sensitivity – 82.36%
- F1 Score – 76.19%

Kod NaiveBayes-a sa primenjenim PCA i podacima iz StandardScaler-a, dobili smo sledeće rezultate:

- Accuracy score – 78.77%
- Precision score – 84.74%
- Sensitivity – 63.29%
- F1 Score – 72.46%

Kod RandomForestClassifier-a sa PCA i podacima iz StandardScaler-a, dobijeni su sledeći rezultati:

- Accuracy score – 79.88%
- Precision score – 79.45%
- Sensitivity – 73.41%
- F1 Score – 76.31%

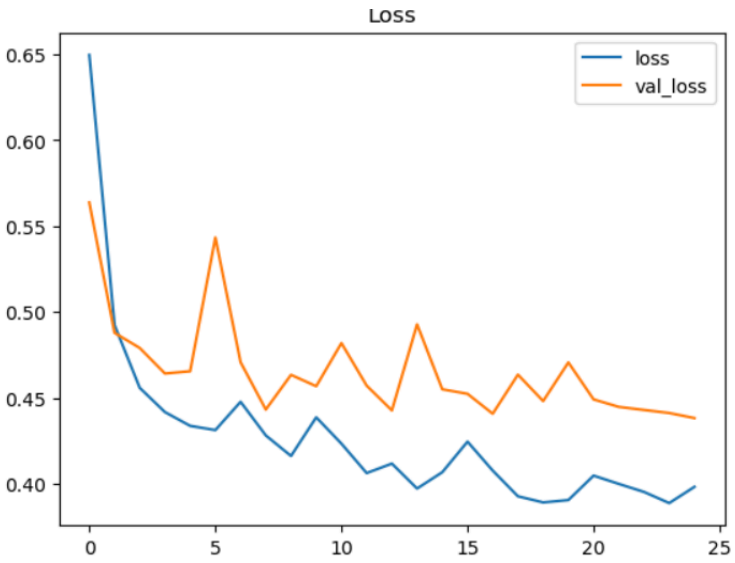
Kod KNeighbourClassifier-a pod istim osobinama, dobili smo sledeće rezultate:

- Accuracy score – 83.24%
- Precision score – 90.16%
- Sensitivity – 69.62%
- F1 Score – 78.57%

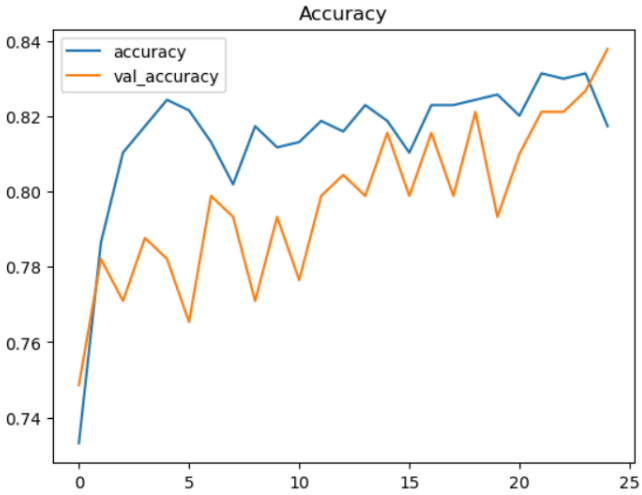
Kod SVM-a pod istim osobinama, dobili smo sledeće rezultate:

- Accuracy score – 76.53%
- Precision score – 79.36%
- Sensitivity – 63.29%
- F1 Score – 70.42%

Sada ćemo isti postupak izvršiti nad podacima dobijenim iz *MinMaxScaler*-a. Prvo iscrtavamo funkciju gubitka i tačnost:

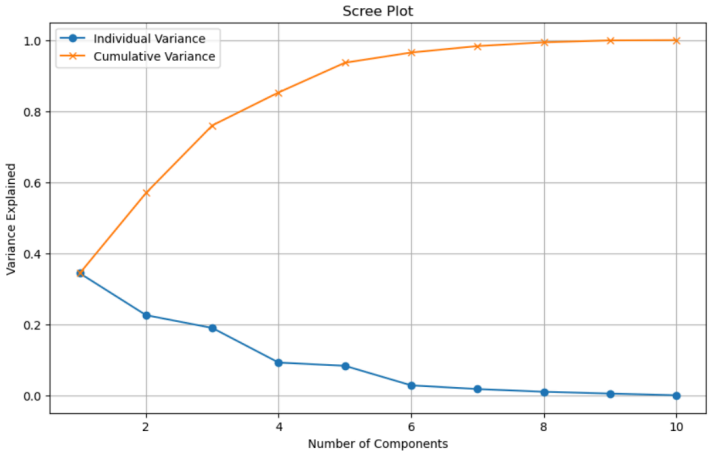


Slika 20. Funkcija gubitka (loss) sa podacima dobijenim iz MinMaxScaler-a



Slika 21. Tačnost (accuracy) sa podacima iz MinMaxScaler-a

Zatim na podatke dobijene iz MinMaxScaler-a primenjujemo PCA, iscrtavamo Scree plot i dobijamo optimalan broj komponenti:



Slika 22. *Scree plot* uz upotrebu podataka dobijenih iz MinMaxScaler-a

U ovom slučaju, zaključujemo da je optimalan broj komponenti jednak 6. Sada ćemo, uz upotrebu optimalnog broja komponenti, PCA primeniti na sve modele koje smo i ranije koristili, uz upotrebu podataka dobijenih iz MinMaxScaler-a i nad kojima je primenjen PCA.

Pod ovim uslovima, kod jednostavne neuronske mreže dobijamo sledeće rezultate:

- Accuracy score – 85.47%
- Precision score – 72.88%
- Sensitivity – 81.13%
- F1 Score – 76.78%

Kod komplikovanije neuronske mreže dobijamo sledeće rezultate:

- Accuracy score – 86.03%
- Precision score – 74.57%
- Sensitivity – 81.48%
- F1 Score – 77.87%

Kod NaiveBayes-a dobijamo sledeće rezultate:

- Accuracy score – 84.35%
- Precision score – 74.6%
- Sensitivity – 79.66%
- F1 Score – 77.05%

Kod SVM-a dobijamo sledeće rezultate:

- Accuracy score – 72.65%
- Precision score – 64.91%
- Sensitivity – 56.06%
- F1 Score – 60.16%

Kod RandomForestClassifier-a dobili smo sledeće rezultate:

- Accuracy score – 81%

- Precision score – 80.77%
- Sensitivity – 63.63%
- F1 Score – 71.18%

Kod KNeighborsClassifier-a dobili smo sledeće rezultate:

- Accuracy score – 79.33%
- Precision score – 78.43%
- Sensitivity – 60.6%
- F1 Score – 68.37%

Zaključak

U ovoj analizi smo primenili nekoliko klasifikacionih algoritama na skupu podataka putnika sa Titanika kako bismo predvideli verovatnoću preživljavanja. Cilj je bio da se identifikuju ključni faktori koji su uticali na preživljavanje putnika i da se razviju precizni modeli za predikciju.

Ključni nalazi:

1. **Istraživačka analiza podataka:**
    - Vizuelizacija podataka pokazala je da su faktori poput pola, starosti, klase karte i veličine porodice imali značajan uticaj na stopu preživljavanja.
    - Na primer, žene i deca su imale veću verovatnoću da prežive, dok su putnici iz prve klase imali značajno veću stopu preživljavanja u poređenju sa putnicima iz druge i treće klase.
2. **Inženjering karakteristika:**
    - Kategorijalne promenljive su enkodirane i numerički podaci su skalirani kako bi se omogućila primena različitih modela.
3. **Primena i evaluacija modela:**
    - Korišćeni su različiti klasifikacioni algoritmi, uključujući logističku regresiju, Random Forest, K-Nearest Neighbors, i Gaussian Naive Bayes.
    - Najbolje performanse su postignute korišćenjem Random Forest Classifier-a, koji je postigao najbolju tačnost i balans između preciznosti i osetljivosti.
    - Parametri modela su optimizovani korišćenjem GridSearchCV, što je dodatno poboljšalo performanse.
4. **Validacija modela:**
    - Performanse modela su evaluirane pomoću metričkih kao što su tačnost, preciznost, osetljivost i F1 skor.
    - Modeli su testirani na validacionom skupu podataka kako bi se osigurala njihova generalizabilnost na nove podatke.

Naša analiza preživljavanja na Titaniku pokazala je da su klasifikacioni algoritmi efikasni u predviđanju verovatnoće preživljavanja putnika na osnovu dostupnih podataka. Random Forest Classifier se izdvojio kao najefikasniji model za ovaj zadatak, zahvaljujući svojoj sposobnosti da se nosi sa kompleksnim relacijama između karakteristika.

Preporuke za budući rad:

- **Dodatni inženjering karakteristika:** Dalja analiza i kreiranje novih karakteristika, kao što su interakcioni termini između postojećih karakteristika, mogu dodatno poboljšati performanse modela.
- **Proširenje skupa podataka:** Integracija dodatnih podataka (npr. društveno-ekonomski status putnika) mogla bi pružiti dublji uvid i poboljšati tačnost predikcija.
- **Korišćenje naprednih modela:** Istraživanje naprednih mašinsko-učenja tehnika kao što su Gradient Boosting ili Deep Learning modeli može dodatno unaprediti performanse.

Na kraju, ova studija ne samo da demonstrira primenu mašinskog učenja u analizi istorijskih događaja, već takođe pruža korisne uvide koji bi mogli biti primenjeni u sličnim domenima za predikciju ishoda na osnovu sličnih skupova podataka.