Predviđanje performansi interakcije pokaznom napravom

Dudić, Mia

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

PREDVIĐANJE PERFORMANSI INTERAKCIJE POKAZNOM NAPRAVOM

Rijeka, rujan 2023.

Mia Dudić

0069073925

SVEUČILIŠTE U RIJECI

TEHNIČKI FAKULTET

Diplomski sveučilišni studij računarstva

Diplomski rad

PREDVIĐANJE PERFORMANSI INTERAKCIJE POKAZNOM NAPRAVOM

Mentor: doc. dr. sc. Goran Mauša

Komentor: izv. prof. dr. sc. Sandi Ljubić

Rijeka, rujan 2023.

Mia Dudić

0069073925

IZJAVA

Izjavljujem da sam samostalno, pod vodstvom doc. dr. sc. Gorana Mauše i izv. prof. dr. sc. Sandija Ljubića, primjenjujući znanja stečena tijekom studija, te koristeći navedenu literaturu izradila diplomski rad naslova „Predviđanje performansi interakcije pokaznom napravom“.

Rijeka, rujan 2023.

_____

Mia Dudić

ZAHVALA

Želim se zahvaliti svojem malom krugu ljudi, koji su uvijek bili tu. Zahvala i kolegama s fakulteta, s kojima sam dijelila svaki korak ovog iskustva, a koji su usput postali i prijatelji. Zahvala i mentorima, profesorima i ostalim zaposlenicima fakulteta na prenesenom znanju. I posebna zahvala ide mojoj obitelji na mogućnostima koje su mi pružene i potpori na svakom koraku.

Sadržaj

1. UVOD

Danas je tehnologija nezaobilazna stavka ljudskih života i ostavlja nemjerljivi učinak – od socijalnog aspekta pa sve do posla. Bilo da je riječ o poslovanju, fakultetu ili privatnim potrebama, gotovo i da nema sfere u kojoj ne koristimo neki oblik tehnologije. Uglavnom je to za obavljanje poslovnih ili studentskih obaveza pri čemu koristimo interaktivni uređaj kako bismo komunicirali sa stolnim računalom ili laptopom. Prosječni će korisnik koristiti miš, dok je u slučaju prijenosnih računala zastupljeno podjednako korištenje miša ili *touchpada* samog laptopa. Upravo ovdje započinje ideja projekta – osmisliti testiranje ljudskih performansi prilikom rješavanja programskih zadataka odabranim interaktivnim uređajem te za dobivene rezultate napraviti model predviđanja. Miš i *touchpad* pokazni su uređaji gdje oboje postižu isti cilj – pokazivanje, odabir i pomicanje kursora. Međutim, razlikuju se u tome kako izgledaju i djeluju kako bi postigli svoju funkciju. Miš se pomiče po ravnoj površini za što je potreban cijeli dlan, dok se samo prstom pomiče preko *touchpada* kako bi se napravio pomak kursora. Uobičajeni *touchpad* koristi kapacitivnu tehnologiju za otkrivanje kretanja prsta, dok je miš optički za otkrivanje kretanja po površini. Ideja je testirati korisnike prilikom izvršavanja određenih zadataka kako bi se vidjela razlika između različitih tipova i karakteristika ljudi. S obzirom da su u testiranje uključeni ljudi u razmaku od 15 do 40 godina te su svi oni iz različitih područja struke, bilo je zanimljivo pratiti kako se mogu snaći u situaciji rješavanja tunelskih zadataka i koliko im je vremena za to potrebno. Također, općenito većina korisnika pokazni uređaj koristi duži vremenski period ali s prekidima te je uglavnom to mali pokret koji ne zahtjeva posebnu pozornost na koordinaciju pokreta. Isto tako, prilikom korištenja pokaznog uređaja na dnevnoj bazi, korisnici nisu ograničeni vremenski, niti im brzina korištenja uređaja ovisi o poslu. Zato je ideja bila staviti korisnike u drugačiju situaciju – gdje kontinuirano koriste pokazni uređaj na duži vremenski period i duže radnje koje je potrebno njime napraviti te im se pri tome mjeri vrijeme koje im je bilo potrebno za rješavanje zadatka.

Cilj je projekta napraviti model predviđanja brzine izvršavanja tunelskih zadataka. I prije nastajanja same ideje ovog projekta, želja je bila imati vlastite podatke koje se može koristiti za strojno učenje. Pa kada je nastala ideja testiranja pokaznih uređaja prilikom rješavanja tunelskih zadataka, slijedno je razmišljanje bilo da se dobiveni podaci obrade i pripreme za potrebe razvoja modela predviđanja. Potrebno je bilo testirati više različitih algoritama kako bi se utvrdilo koji od njih daje najbolje rezultate. S obzirom da se radi o problemu regresije jer je ciljana varijabla

kontinuirana, izrada modela predviđanja započeta je s osnovnim i najpoznatijim algortimom regresije – linearnom regresijom. Potom su testirane regularizacije linearne regresije: *Ridge* i *Lasso* regresija. Testirana je i *Bayes* linearna regresija, *Gaussianov* proces, algoritam slučajne šume, algoritam stabla odluke, metoda najbližih susjeda te polinomna regresija.

Kako bi se započeo praktični dio ovog rada, prvotno je osmišljena i implementirana aplikacija koja pruža mogućnost automatiziranog testiranja korisnika u korištenju pokaznog uređaja. Testiranje se zasniva na rješavanju tunelskih zadataka – niz tunela kroz koje je potrebno proći mišem, ostavljajući trajektoriju, prilikom čega se mjeri vrijeme potrebno da se prođe kroz jedan tunel. Tuneli su definirani kao dvije krivulje i kroz 15 različitih slučajeva mijenja se njihova širina i broj amplituda. S obzirom da se prilikom testiranja za interaktivni uređaj koristio samo miš, postavljeni su dodatni uvjeti testiranja kako bi se utvrdile performanse korisnika prilikom rješavanja zadataka – boja programa, postojanje idealne linije i glazba. Rješavanjem zadataka dobiveno je 120 rezultata po osobi koji su analizirani i obrađeni za daljnje korake ovog projekta. Izrađena je baza podataka svih korisnika te je ista dalje korištena u strojnom učenju kao temelj za model predviđanja. Kako je korišteno više različitih algoritama za rješavanje problema regresije, napravljena je njihova je usporedba, a pažnja je stavljena na model koji je dao najbolje rezultate i čije su predviđene vrijednosti bile najbliže stvarnim vrijednostima.

2. METODOLOGIJA

Prilikom testiranja, mjeri se vrijeme koje je kontinuirana varijabla. A definiranje kontinuirane varijable kao ciljane, odnosno varijable koja se u konačnici predviđa, svrstava ovaj problem u problem regresije za čije se rješavanje i model predviđanja koriste regresijski algoritmi. Ideja regresijskih algoritama jest predviđanje ciljanje kontinuirane varijable (u ovom slučaju brzine/vremena) na temelju nezavisnih varijabli (u ovom slučaju to su spol, godine, iskustvo, orijentacija ruke, krivulja, *level*, boja, idealna putanja, glazba, naočale, posao, doba dana, daltonizam). Regresija radi na način da traži najbolju odgovarajuću liniju koja će što točnije predvidjeti ciljanu varijablu. Ostvaruje se veza između nezavisnih varijabli i ciljane zavisne varijable te bi ona trebala biti linearne prirode.

Kod problema regresije ciljana varijabla je realan broj koji predstavlja izlaznu vrijednost. Na primjeru baze podataka napravljene u ovom projektu gdje su prikupljene osobne informacije svakog korisnika, informacije i uvjeti vezani za testiranje te je dobivena izlazna vrijednost vrijeme, možemo koristiti te podatke kako bismo dobili predviđene vrijednosti. Tako možemo na temelju godina ili težine krivulja predvidjeti brzinu korisnika potrebnu da riješi određeni zadatak. Primjer je dan tablicom 2.1:

Tablica 2.1. Primjer problema regresije

GODINE	KRIVULJA	VRIJEME
26	1	2840
26	1	4104
26	1	3848
31	2	3173
31	2	3674
31	2	5448
37	3	5369
37	3	8158
37	3	12707
33	4	7993
33	4	9920
33	4	15915
27	5	18939
27	5	23329
27	5	25918

Problem regresije uvijek izgleda slično kao u tablici 2.1. – postoje nezavisne varijable i zavisna varijabla koja se pokušava predvidjeti na temelju danih nezavisnih varijabli.

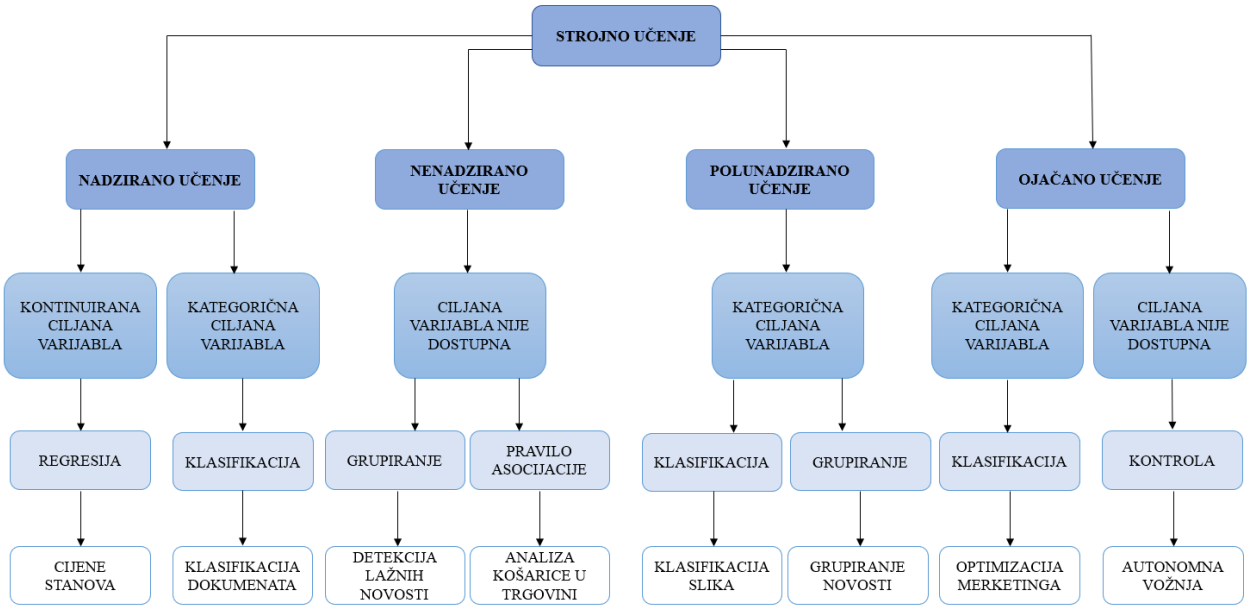
Općenito, problem regresije može se riješiti primjenom algoritama strojnog učenja. Strojno učenje je dio umjetne inteligencije koji se fokusira na korištenje dostupnih podataka i pripadajućih algoritama kako bi se oponašalo učenje i postepeno povećavala točnost modela. Algoritmi strojnog učenja koriste dostupne podatke kako bi se predvidjele, klasificirale ili kategorizirale vrijednosti. Upravo je danas strojno učenje jako široko i utjecajno područje, a njegova je primjena od velike važnosti. Unutar poslovanja omogućuje uvid u trendove i obrasce ponašanja te uzorke poslovanja. Isto tako, primjenu strojnog učenja možemo vidjeti kod autonomne vožnje ili ono bliskije većini ljudi – Netflixov sustav preporuke. [1] [2]

Strojno učenje karakterizira se ovisno o učenju algoritma s težnjom za što većom točnosti. Razlikujemo četiri pristupa [1]:

- Nadzirano učenje – algoritmu se daju definirani podaci i specificirane nezavisne i zavisne varijable. Postoji jasna slika veza između dostupnih varijabli gdje je točno određeno što je ulazna varijabla, a što izlaz,
- Nenadzirano učenje – algoritmi se treniraju na neoznačenim podacima tako što pregledavaju podatke i traže značajnu međusobnu vezu,
- Polunadzirano učenje – kombinacija prethodna dva pristupa. Algoritam dobije označene podatke, ali može samostalno proizvoljno istražiti dobivene podatke i razviti vlastito razumijevanje istih,
- Ojačano učenje – algoritam je programiran da odradi određen zadatak, a potom dobije pozitivan ili negativan odgovor u smislu potkrepljivanja.

Pristup koji će se odabrati ovisi direktno o vrsti podataka s kojom se radi i koje se želi predvidjeti.

Na slici 2.1. dan detaljni vizualni prikaz raspodjele vrsta strojnog učenja i primjera njihove karakteristične primjene:



Slika 2.1. Podjela strojnog učenja

S raspodjele na slici 2.1. vidimo četiri definirana pristupa strojnom učenju. Svaki od njih koristi algoritme preferirane ovisno o dostupnim podacima i njihovoj vrsti.

Prednosti strojnog učenja vrlo su očite – samostalno učenje ovisno o analizi podataka i prepoznavanja ponašanja što u konačnici dovodi do pronalaska rješenja. Smanjuje se potreba za manualnim kalibracijama, a mogućnost analize velikih volumena podataka dovodi do visoke efikasnosti samog poslovanja što je od iznimne važnosti u industriji gdje je vrijeme od osobite važnosti. S druge strane, najveća je mana cijena – potrebna tehnologija i ljudi iziskuju znatne troškove. Pogreške su sastavni dio svakog projekta, ali u ovom konceptu mogu rezultirati visokim troškovima financija i vremena. Također, uz cijenu je vrijeme ključna komponenta svakog projekta. Za svako različito područje upotrebe strojnog učenja, potreban je sustav koji će biti specijaliziran za specifične potrebe. Specijalizacija sustava na toj je razini skup proces koji je znatno vremenski zahtjevan. [3]

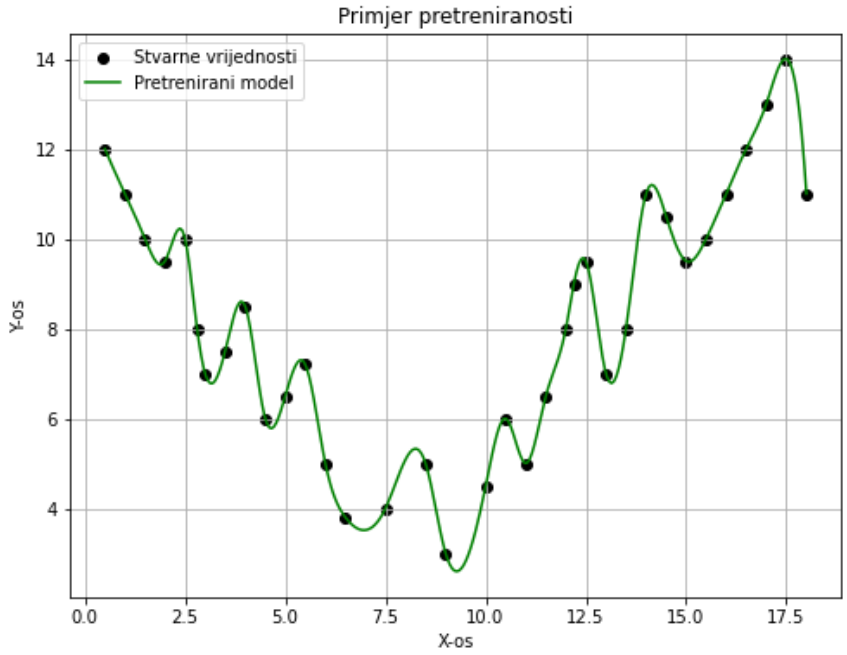
Za ovaj je projekt odgovarajući pristup strojnog učenja nadzirano učenje jer se koristi baza podataka gdje su jasno definirane ulazne varijable i jedna izlazna ciljana varijabla. Ulazne su varijable: spol, godine, iskustvo, orijentacija ruke, krivulja, *level*, boja, idealna putanja, glazba, naočale, posao, doba dana, daltonizam, a varijabla koja se predviđa je ciljana varijabla i u ovom je slučaju to vrijeme potrebno za rješavanja zadatka Također, varijabla koja se predviđa je kontinuiranog karaktera što definira problem regresije. Proces je započet učitavanjem datoteke s odgovarajućim setom podataka za korištenje na kojima se onda provode određeni algoritmi. Učitavanje se provodi uz korištenje knjižnice *pandas*. Nakon uspješnog učitavanja datoteke podaci su podijeljeni na ulazne nezavisne varijable te jednu zavisnu ciljanu varijablu. Za algoritme je napravljena unakrsna validacija kako bi se u konačnici dobili bolji rezultati. Za svaki algoritam napravljena je instanca objekta na kojemu se provodi *fit()* funkcija za treniranje modela i zatim se funkcijom *predict()* ostvaruju predviđene vrijednosti ciljane varijable. Koristeći knjižnicu *matplotlib.pyplot* grafički su prikazani dobiveni rezultati.

2.1. Regresija

Regresijska analiza je temeljni koncept u području strojnog učenja. Spada pod učenje pod nadzorom gdje se algoritam trenira i s ulaznim parametrima i s ciljanom varijablom. To je proces estimacije veze između zavisne varijable i nezavisnih varijabli. Koristeći regresiju cilj je postaviti funkciju na dostupnim podacima i pokušati predvidjeti ponašanje podataka u budućnosti. [4] Kako bi se vrednovala predviđanja moraju se u obzir uzeti sljedeće stavke:

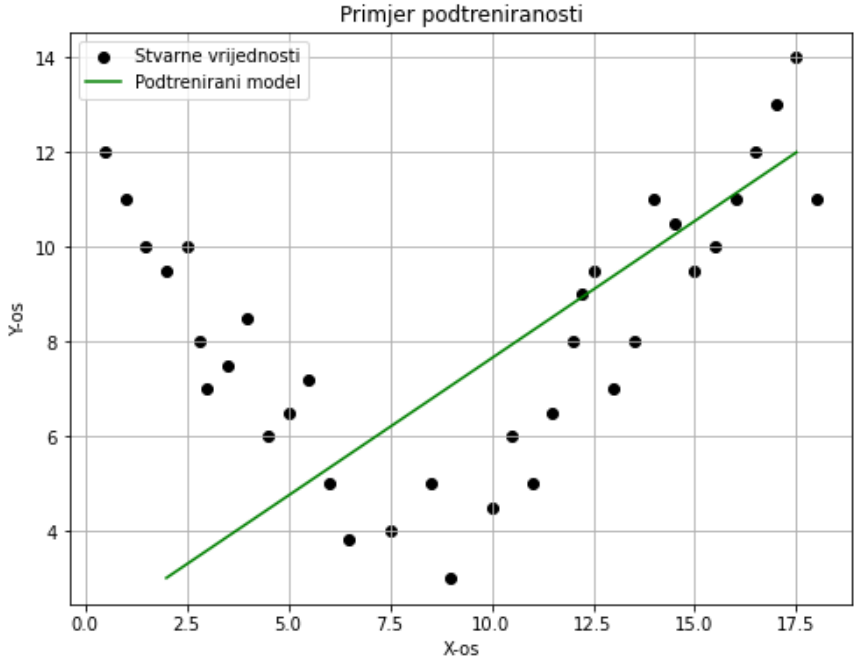
- Varijanca – iznos za koji se procjena ciljane funkcije mijenja ako su korišteni drugačiji podaci za treniranje modela. Model s visokom varijancom pažnju stavlja na podatke za treniranje i ne generalizira na podacima s kojima se susreće prvi put. Kada se koristi drugačiji skup podataka ciljana funkcija mora ostati stabilna s malim odstupanjima jer bi za bilo koju vrstu podataka model trebao biti generaliziran. Kako bismo izbjegli lažna predviđanja moramo osigurati da je varijanca niska. Visoku varijancu možemo prepoznati tako što se u fazi treniranja ponaša vrlo dobro, ali u fazi testiranja daje velike pogreške, [5] [6]
- Pristranost – razlika između prosječnog predviđanja modela i stvarne vrijednosti koju pokušavamo predvidjeti. Model s visokom pristranosti posvećuje jako malo pažnje podacima za treniranje i previše pojednostavljuje model što u konačnici uvijek dovodi do visokih pogrešaka. Konstanto uči krivo jer ne uzima u obzir sve informacije u skupu podataka. Kako bi model bio što točniji pristranost mora biti niska, [5] [6]
- Točnost – udio predviđanja koje je model točno pogodio,
- Pogreška – razlika između stvarne i predviđene vrijednosti.

Kako bi model bio idealan, trebao bi imati nisku varijancu, nisku pristranost i nisku pogrešku. Kako bi se ovo postiglo potrebno je podijeliti skup podataka na treniranje i testiranje. Model će se učiti na podacima za treniranje te će se performanse pokazati i evaluirati na podacima za testiranje. Ako model pamti ili oponaša podatke za treniranje, dobit ćemo pogrešna predviđanja podataka na kojima se model nije trenirao. U tom slučaju, krivulja bi prolazila kroz sve točke i takav se slučaj naziva pretreniranost koji je prikazan na slici 2.2. Uzrok tome je visoka varijanca. [5]



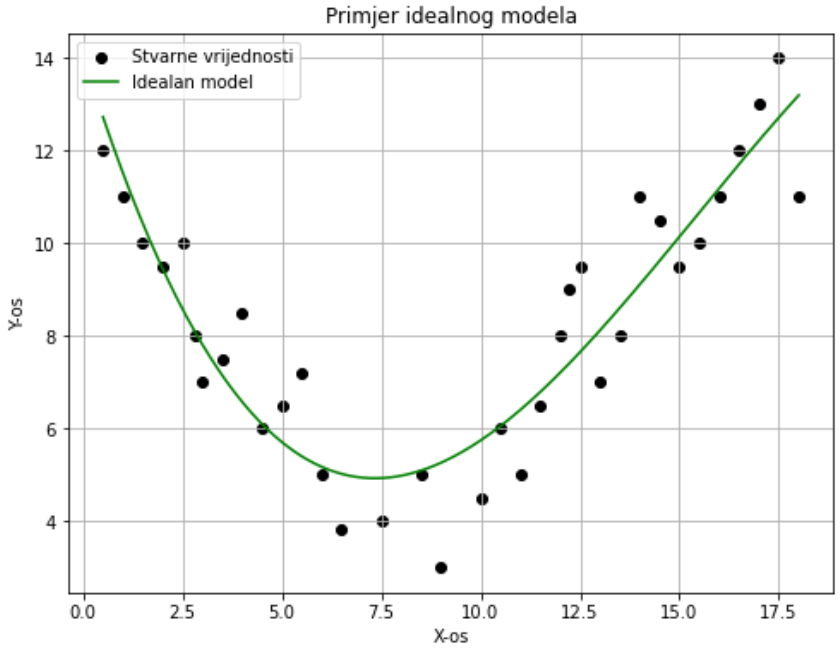
Slika 2.2. Pretreniranost modela predviđanja

U suprotnom slučaju dolazi do podtreniranosti – ako se model pokazuje dobrim na podacima za treniranje, ali s niskom točnosti na podacima za testiranje. Uzrok tome je visoka pristranost. Primjer je prikazan na slici 2.3.



Slika 2.3. Podtreniranost modela predviđanja

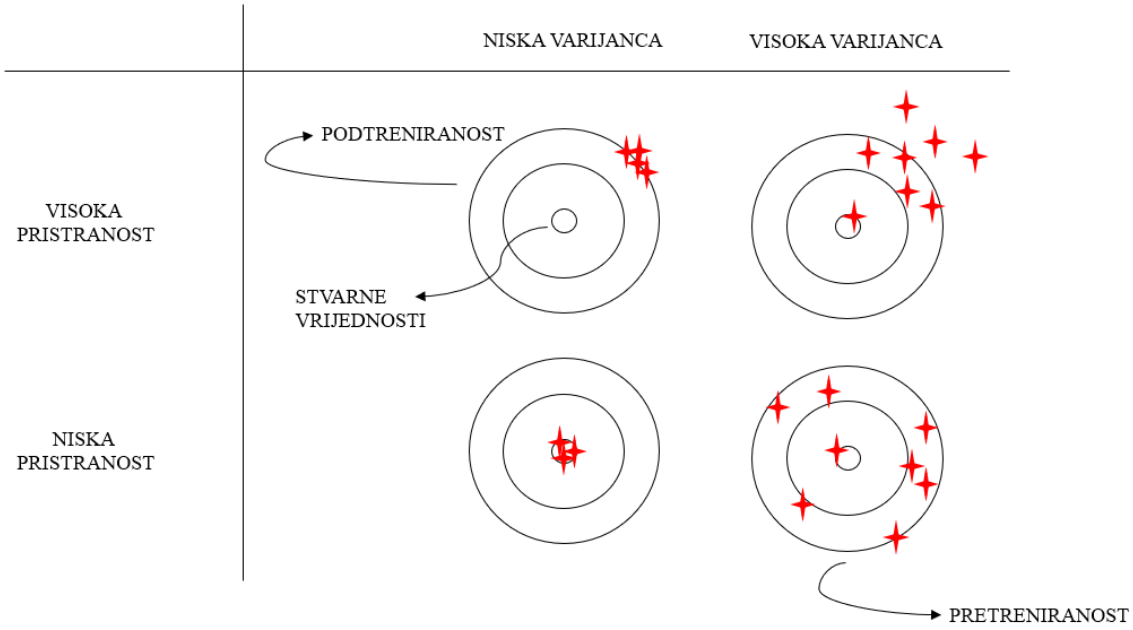
U idealnom slučaju regresijska bi krivulja napravljena na podacima trebala izgledati kao na slici 2.4.:



Slika 2.4. Idealan model

Što znači da su i varijanca i pristranost niske te da je model rezultirao dobrim predviđanjima s visokom točnosti, a malim brojem pogrešaka.

Na slici 2.5. prikazana je veza između varijance i pristranosti te kako utječu na konačan model.



Slika 2.5. Veza pristranosti i varijance

Na dijagramu je centrom prikaz model koji idealno predviđa točne vrijednosti. Kako se odmičemo od sredine, predviđanja postaju sve lošija. Ukoliko je niska varijanca, a visoka pristranost, model rezultira s pogrešnim predviđanjima i podtreniranosti. Ukoliko su i varijanca i pristranost niski, model će predviđati idealno. Ako je visoka varijanca i visoka pristranost, model će dati pogrešna predviđanja. Ako je visoka varijanca i niska pristranost, model će dati pogrešna predviđanja i rezultirati s pretreniranosti. [7]

Kako bi se riješio problem podtreniranosti, neke od mogućnosti su [7]:

- Povećati složenost modela,
- Povećati broj parametara (inženjering značajki),
- Uklanjanje šuma iz skupa podataka,
- Produljiti trajanje treninga.

Za rješavanje problema pretreniranosti moguće je [7]:

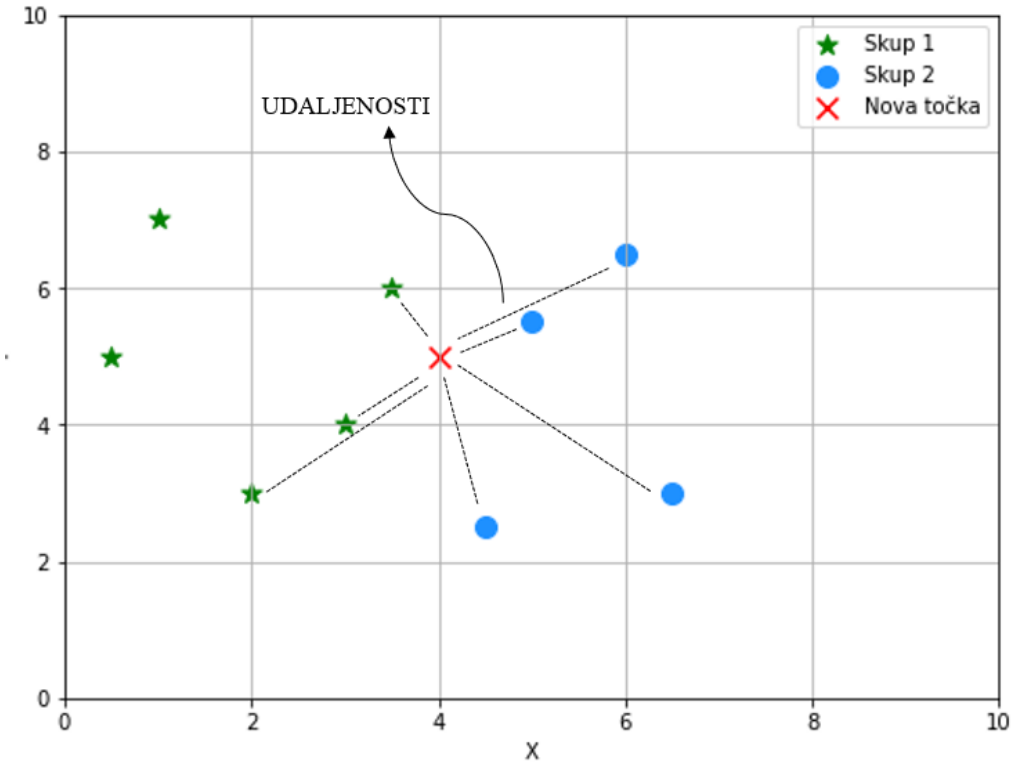
- Povećati broj podataka za testiranje,
- Smanjiti složenost modela,
- Korisititi regularizaciju (Ridge i Lasso regresija).

2.2. Algoritmi regresije

U nastavku je dana raspodjela algoritama koji su korišteni u razvijanju modela predviđanja. Testirano je desetak algoritama, ali su detaljno obrađeni troje od njih kojima su dobiveni najbolji rezultati.

2.2.1. Metoda najbližih susjeda

Metoda najbližih susjeda (eng. *K-Nearest Neighbors,* kNN) je algoritam koji se temelji na ideji da su slični objekti uvijek u neposrednoj blizini i računa njihovu međusobnu udaljenost kao na slici 2.6. Odabire specificirani broj primjera k najbližih upitu i izračunava prosjek oznaka.



Slika 2.6. kNN algoritam

Sa grafa je vidljivo kako su u većini slučajeva, slični podaci međusobno bliski, odnosno da se nalaze u međusobnoj blizini. kNN algoritam se bazira na činjenici da je ova pretpostavka točna i na računanju udaljenosti između točaka na grafu. [8] Koraci za provođenje kNN algoritma dani su u nastavku [8]:

1. Učitavanje baze podataka na kojoj će se provoditi model predviđanja
2. Inicijalizirati K na proizvoljan broj susjeda
3. Za svaku točku u skupu podataka

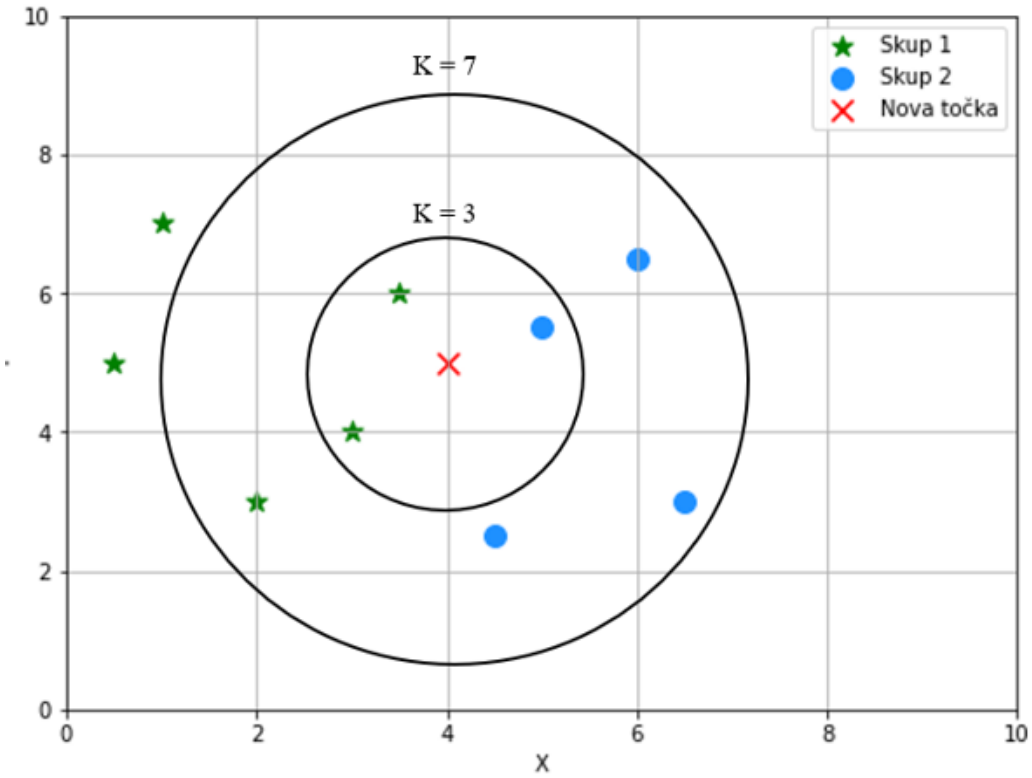
3.2. Izračunati udaljenost između točke u redu čekanja i trenutnog primjera

3.3. Dodati udaljenost i indeks točke u listu

4. Sortirati uređenu listu od manje prema većoj – prema udaljenostima
5. Odabrati prvih K ulaza iz liste (K broj najbližih točaka)
6. Dohvatiti vrijednosti za odabrane K ulaze
7. Vratiti srednju vrijednost K oznaka kao predviđanje

Kako bi se odabrala odgovarajuća vrijednost za K potrebno je izvršiti algoritam nekoliko puta s različitim vrijednostima i odabrati K koji reducira broj pogrešaka dok se održava mogućnost algoritma da točno predviđa vrijednosti za podatke s kojima se još nije susreo. Potrebno je uzeti u obzir da kako se vrijednosti za K smanjuju prema 1, predviđanja postaju manje stabilna. Kako se vrijednost K povećava, predviđanja će biti stabilnija. Međutim, ako se počne povećavati broj pogrešaka, vrijednost za K definirana je previsoko. Također, vrijednost za K se uobičajeno postavlja na neparan broj. [8]

Na slici 2.7. dan je prikaz kNN algoritma ako se za K uzima 3 ili ako se za K uzima vrijednost 7. U prvom slučaju, kada je K = 3, nova točka pripast će skupu 1. U drugom slučaju, kada je vrijednost za K = 7, nova točka pripast će skupu 2. U slučaju regresije, za slučaj kada je K = 3 računat će se prosjek vrijednosti triju najbližih točaka. Kada je K = 7, princip je isti samo se računa prosjek vrijednosti sedam najbližih susjednih točaka.

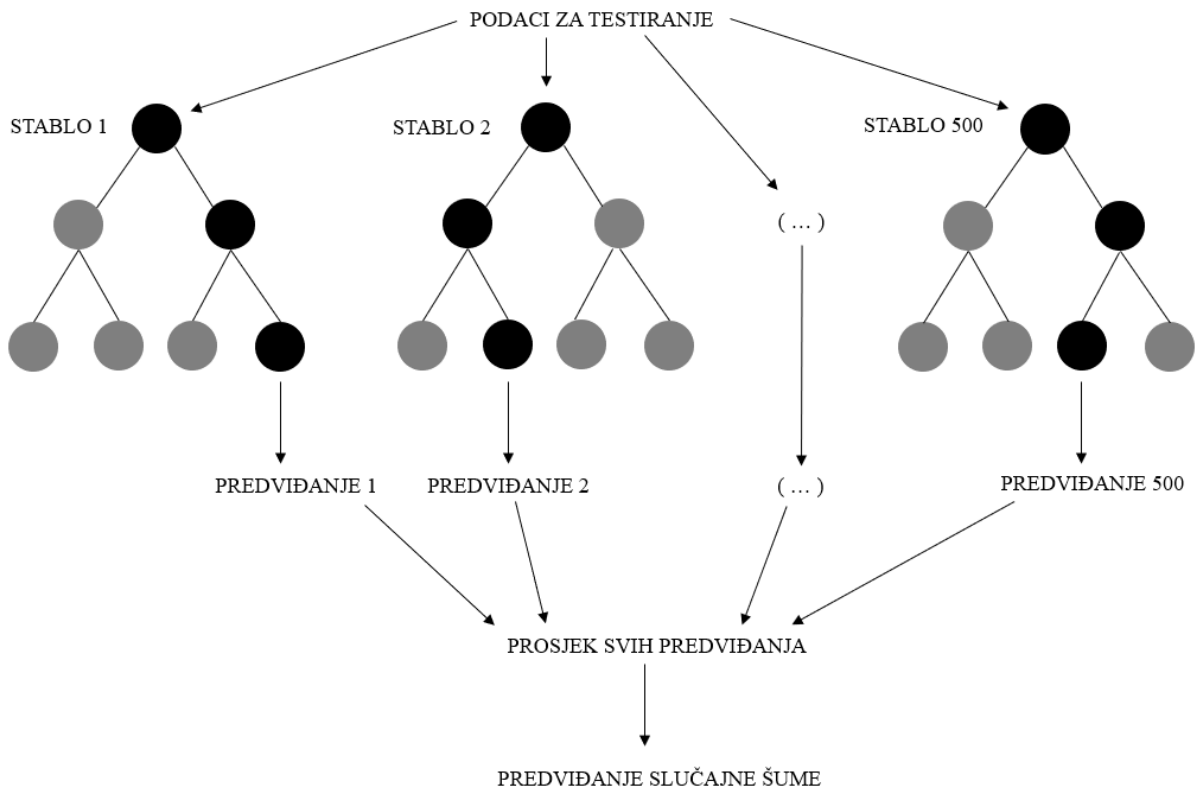


Slika 2.7. Primjer za K=3 i K=7

2.2.2. Slučajna šuma

Algoritam slučajne šume (eng. *Random Forest,* RF) je algoritam koji se može koristiti za rješavanje problema klasifikacije i regresije. Zbog toga se karakterizira kao jedan od prilagodljivih algoritama u strojnom učenju. Jednostavno radi sa složenim bazama podataka i može smanjiti ili ublažiti pretreniranost, što ga čini korisnim alatom za široko područje modeliranja predviđanja. [9]

Slučajna šuma je algoritam koji za rješavanje problema regresije koristi metodu skupnog učenja. Koristi se više pojedinačnih stabala odluke gdje svako stablo daje svoje predviđanje. Ovisno o odabranom broju treniranih stabala dobit će se jednako toliko predviđanja, a srednja vrijednost svih predviđanja definira se kao konačno predviđanje algoritma slučajne šume. [10] Struktura algoritma prikazana je na slici 2.8.:



Slika 2.8. Algoritam slučajne šume

Sa slike 2.8. vidimo da je napravljeno više stabala odluke koji rade paralelno, ali bez međusobne interakcije. Na originalnom skupu podataka, odnosno na setu za treniranje, radi se uzorkovanje s ponavljanjem kako bi se stvorili različiti podskupovi za svako stablo. Svaki podskup sadrži nasumično odabrane primjere iz izvornog skupa podataka, pri čemu se pojedini primjeri mogu ponavljati u istom podskupu, dok drugi možda neće biti uključeni. Veličina napravljenog podskupa obično je jednaka broju primjera u originalnom skupu podataka. Kako se stvaraju različiti podskupovi svako će stablo biti različito jer će svako stablo koristiti drugačiji skup podataka za donošenje odluka. Također, moguće je ograničiti dubinu stabala kako bi se spriječila pretreniranost modela.

Točni koraci algoritma slučajne šume raspisani su u nastavku [11]:

1. Odabire se nasumičan podskup iz skupa podataka ili seta za treniranje
2. Izrađuje se stablo odluke za svaki podskup podataka za treniranje
3. Svako stablo odluke generira vlastito predviđanje

4. Računa se srednja vrijednost predviđanja svih stabala odluke
5. Odabire se konačno predviđanje koji predstavlja finalni rezultat

Prednosti algoritma slučajne šume su [12]:

- Jednostavna upotrebljivost,
- Manja osjetljivost na podatke za treniranje u odnosu na stablo odluke,
- Veća točnost u odnosu na stablo odluke,
- Učinkovitost u rukovanju većim bazama podataka,
- Može se nositi s nedostajućim podacima, ekstremima i šumovima u setu.

Na lošiju stranu naginje činjenica da algoritam može biti teško interpretirati. Također, potrebno je šire poznavanje domene kako bi se odabrali prikladni parametri algoritma. Za zahtjevnije baze podataka, kao što su slike i videa strojno učenje može biti potencijalno skupo. Potrebne su veoma dobre performanse računala kako bi se algoritam mogao uopće izvršiti i dati rezultate. [12]

2.2.3. Polinomna regresija

S obzirom da se algoritam linearne regresije u ovom slučaju nije pokazao dobrim rješenjem, testirana je polinomna regresija. Linearna regresija funkcionira kada su podaci na kojima se radi linearni. Ako su podaci nelinearne prirode, koristi se linearna regresija višeg stupnja kako bi se lakše opisali podaci i njihova međusobna veza. Ovisno o stupnju linearne regresije razlikujemo različite tipove funkcije [13]:

- Linearna regresija = 1. stupanj,
- Kvadratna regresija = 2. stupanj,
- Kubična regresija = 3.stupanj,
- Kvartna regresija = 4. stupanje,
- Kvintička regresija = 5. stupanj.

Stupanj regresije može varirati do n-tog broja polinoma, odnosno do stupnja funkcije koji će najbolje opisati podatke na kojima se radi. [13] U ovom je slučaju korištena polinomna regresija četvrtog stupnja u kombinaciji s unakrsnom validacijom. Četvrti stupanj se pokazao kao funkcija

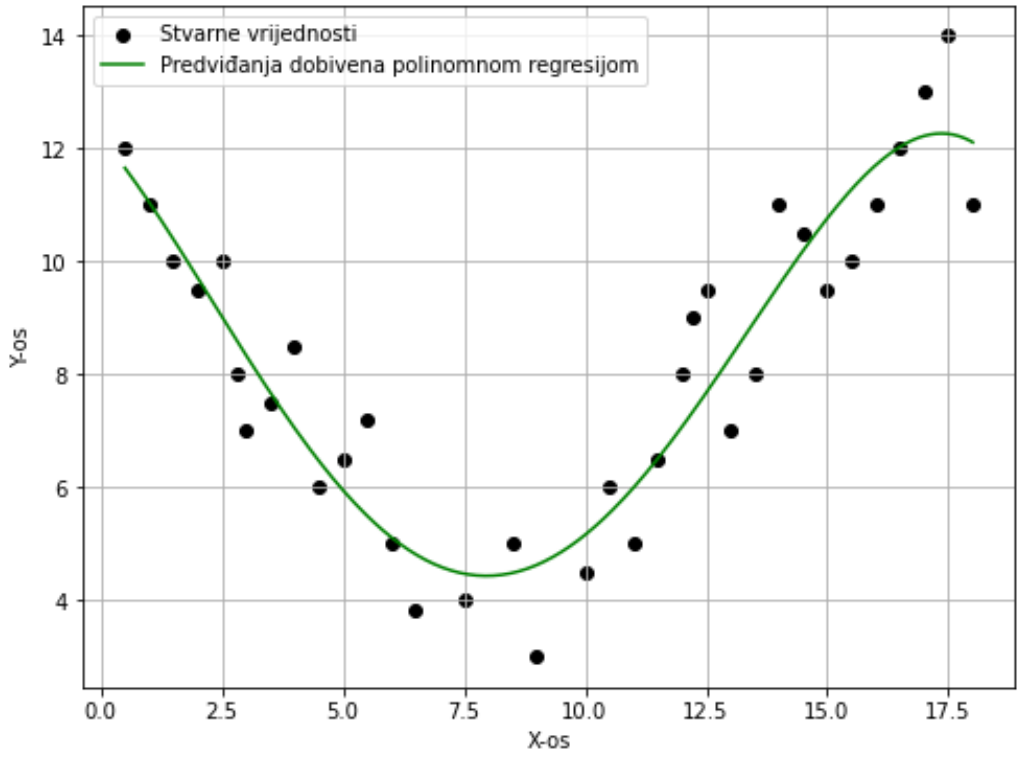
koja je dala model najboljih rezultata. Matematička formula koja opisuje ovu funkciju dana je izrazom (2.1):

$$f(x) = c_0 + c_1 x + c_2 x^2 \cdots c_n x^n \qquad (2.1)$$

gdje je:

n	stupanj polinoma
c	set koeficijenata.

Polinomna regresija dovoljno je fleksibilna da se može prilagoditi širokom rasponu podataka te nudi najbolju aproksimaciju veze između zavisnih i nezavisnih varijabli. Mana ovog algoritma su ekstremi, odnosno, polinomna regresija je jako osjetljiva na ekstreme u bazi podataka. [13] Na slici 2.9. dan je prikaz modela napravljenog polinomnom regresijom.



Slika 2.9. Primjer predviđanja polinomnom regresijom

2.3. Metrike vrednovanja

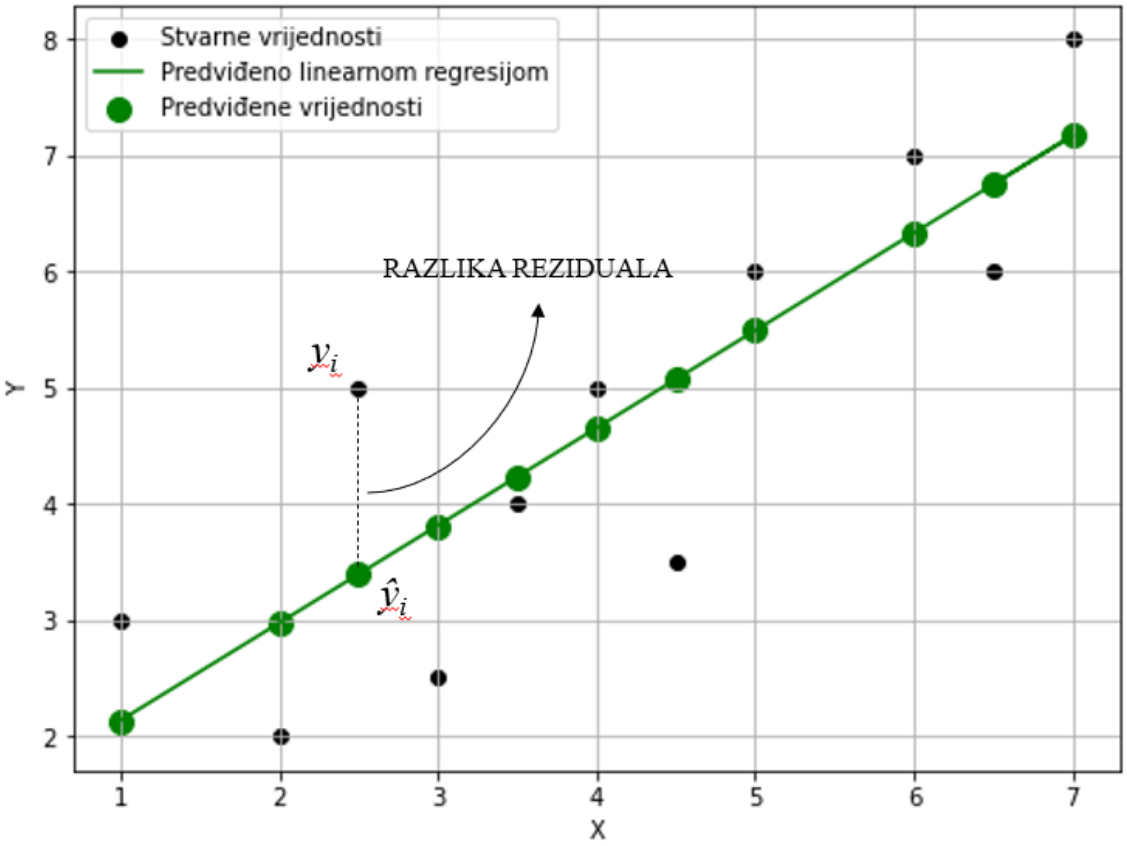
Kako bismo bili sigurni da model radi dobro i da daje dobre rezultate, koriste se metrike vrednovanja. S obzirom da za model nije moguće ili nije idealno da daje točna predviđanja kontinuiranih varijabli, već daje samo predviđanja koja su veća ili manja od stvarnih vrijednosti, jedini način da se utvrdi točnost modela jest preko reziduala. Oni su razlika između stvarnih i predviđenih vrijednosti – kao udaljenost. Što je njegova vrijednost bliža nuli, model je dao bolja predviđanja. [14] Razlika se računa prema formuli (2.2):

$$e_i = y_i - \hat{y}_i \qquad (2.2)$$

gdje je:

- e_i razlika reziduala
- y_i stvarna vrijednost
- $\hat{y}_i$ predviđena vrijednost

Ako je primjer stvarne vrijednosti u bazi podataka 5, predviđena vrijednost je 3.4, razlika reziduala biti će 1.6. Na slici 2.10. crnim su točkama označene stvarne vrijednosti, a zelenom je linijom označen model predviđanja dobiven linearnom regresijom. Zelene točke duž pravca linearne regresije označavaju predviđene vrijednosti.



Slika 2.10. Razlika reziduala

Najznačajnije metrike za vrednovanje uključuju:

- R^2 rezultat (eng. *R-Squared Score,* R^2) - statistička metrika koja nam govori koliko dobro model radi sva svoja predviđanja na skali od 0 do 1. Izrazom (2.3) dan izračun metrike:

$$R2 = 1 - (RSS / TSS) \qquad (2.3)$$

gdje je:

R2	koeficijent determinacije
RSS	suma kvadrata reziduala
TSS	ukupna suma kvadrata

Iz prethodne je formule potrebno izračunati sumu kvadrata reziduala RSS i ukupnu sumu kvadrata TSS. *RSS* se dobije po formuli (2.4):

$$RSS = \sum (yi - \hat{y}_i)^2 \qquad (2.4)$$

TSS se dobije po formuli (2.5):

$$TSS = \sum (y_i - \bar{y})^2 \qquad (2.5)$$

gdje je:

$\bar{y}$ srednja vrijednost varijable

R^2 rezultat koristi se kada želimo točnost modela na skali postotka. Kako bi se metrika implementirala, koristi se knjižnica *Scikit-Learn*, a kod je*: metrics.r2_score(yTest, yPredict)* što nam daje rezultat u decimalnoj vrijednosti, od 0 do 1 te se interpretira kao postotak množenjem sa 100. [14]

- Srednje apsolutno dostupanje (eng. *Mean Absolute Error,* MAE) – definira se kao zbroj svih reziduala podijeljenih s ukupnim brojem točaka u skupu podataka. Predstavlja apsolutnu prosječnu udaljenost modela predviđanja i računa se po formuli (2.6):

$$MAE = 1/N * \sum | yi - \hat{y}_i | \qquad (2.6)$$

gdje je:

N ukupan broj rezultata u bazi

Koristi se apsolutna vrijednost s ciljem da se negativni reziduali pretvore u pozitivne kako ne bi poništili druge pozitivne reziduale. Inače, negativni reziduali su mogući kao rezultat predviđene vrijednosti veće od stvarne vrijednosti. Koristi se kada za model želimo znati koliko su predviđanja blizu stvarnim vrijednostima u prosjeku. Niske vrijednosti ukazuju na točnija predviđanja, dok više vrijednosti ukazuju na suprotno. Unutar programa kod kojim ćemo dobiti rezultat za MAE je *metrics.mean_absolute_error(yTest, yPredict).* Funkcija prima dvije vrijednosti: stvarnu i predviđenu vrijednost. Konačna vrijednost rezultira između 0 i beskonačno. Shodno objašnjenu iznad, vrijednosti koje konvergiraju prema 0 označavaju model s boljim predviđanjima. [14]

- Kvadratni korijen srednjeg apsolutnog odstupanja (eng. *Root Mean Squared Error,* RMSE) – kvadratni korijen prosjeka kvadratne udaljenosti (razlike stvarne i predviđene vrijednosti) izražava se formulom (2.7):

$$RMSE = \sqrt{[(1/N) * \sum (y_i - \hat{y}_i)^2]} \qquad (2.7)$$

Definira se kao kvadratni korijen zbroja kvadrata udaljenosti podijeljenih s ukupnim brojem točaka u skupu podataka. Temeljno je metrika slična MAE s manjom razlikom: koristi se za određivanje postojanja velikih pogrešaka ili udaljenosti koje bi mogle biti rezultat precjenjivanja (model je predvidio vrijednosti značajno veće od stvarnih) ili podcjenjivanja (model je predvidio vrijednosti značajno manje od stvarnih vrijednosti). Implementacija je malo drugačija jer prethodno korištena knjižnica nema spomenutu metriku, ali kako uključuje metodu srednjeg kvadratnog odstupanja (eng. *Mean Squared Error*, MSE), dobijemo vrijednost za istu, a potom se kvadratni korijen od MSE referira se kao RMSE. Prema tome, MSE dobijemo po izračunu: *metrics.mean_squared_error(yTest, yPredict)* gdje se opet traže dvije vrijednosti – stvarna i predviđena. Potom uz pomoć knjižnice *numpy* iz dobivene se vrijednosti traži korijen koji nam daje konačni rezultat. Jednako kao i MAE, dobit će se izračun blizine predviđenih i stvarnih vrijednosti u prosjeku, ali će isto tako ukazati i na efekt velikih pogrešaka. S obzirom da će reziduali biti prikazani kao kvadrat, model će rezultirati velikim pogreškama. Također može poprimati vrijednosti od 0 do beskonačno, ali što je vrijednost manja, znači da je model dao bolja previđanja. Veća vrijednost RMSE od MAE ukazuje na velike pogreške u skupu podataka. [14]

2.4. Unakrsna validacija

Kako bi model mogao naučiti obrasce unutar podataka i izolirati šumove, potrebno je raditi unakrsnu validaciju. U suštini, ova se tehnika koristi za procjenu performansi modela i generalizaciju na novim podacima. Na taj će se način smanjiti rizik od pretreniranosti ili podtreniranosti. Radi se podjela dostupnih podataka na određeni broj podskupova, od čega se jedan podskup koristi za testiranje, a treniranje se vrši na preostalim podskupovima. Proces se ponavlja, koristeći svaki put drugi podskup za testiranje. Konačno, uzima se prosjek rezultata iz svakog koraka validacije kako bi se razvile robusnije procjene modela predviđanja. Kako bi se napravila unakrsna validacija, potrebno je pratiti sljedeće korake [15]:

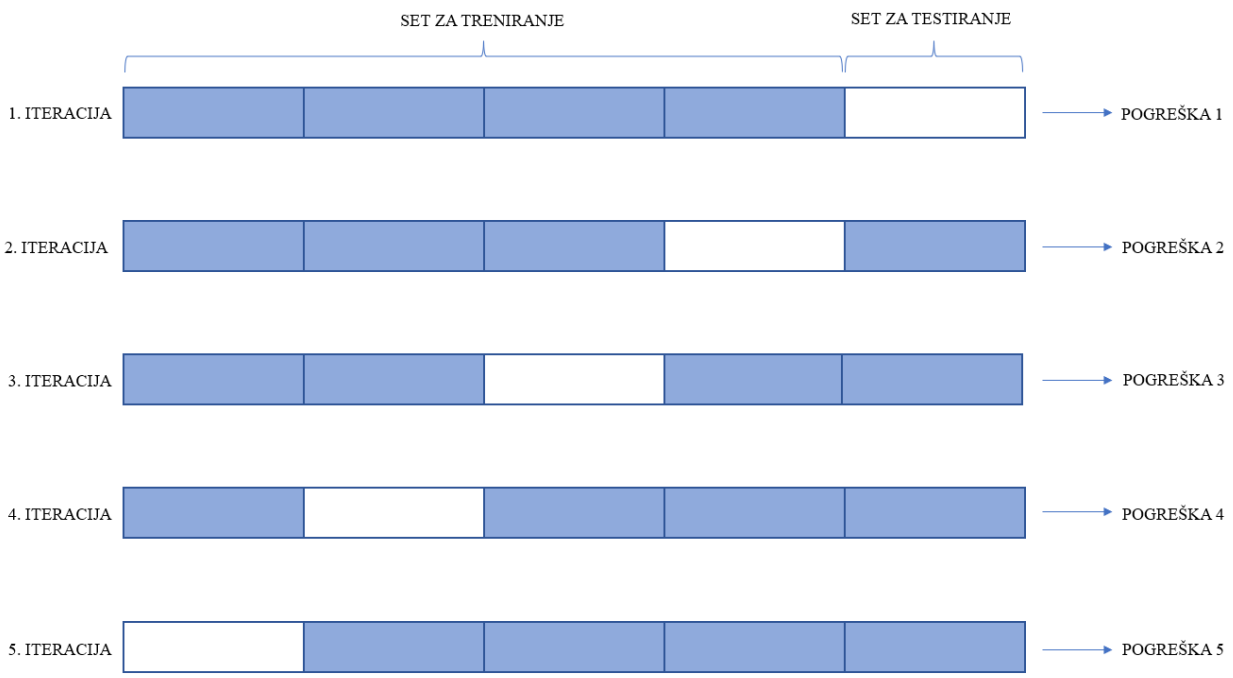
- Rezervira se jedan podskup za validaciju,
- Ostatak podataka koristi se za treniranje modela,

- Model se testira na podacima koji su bili rezervirani.

Ovo su koraci koji se općenito prate prilikom upotrebe bilo koje metode unakrsne validacije, a neke od metoda su:

- Unakrsna validacija,*Holdout* metoda
- Unakrsna validacija u K preklopa,
- Unakrsna validacija stratificirana metoda u K preklopa,
- Unakrsna validacija *Leave One Out* metoda.

U ovom je projektu korištena unakrsna validacija u K preklopa koja podatke dijeli u K broj podskupova. Temelji se na prvoj metodi, odnosno na *Holdout* metodi koja se onda ponavlja za definirani K. Odvaja se dio podataka koji model koristi za razvijanje predviđanja nakon što se trenirao na ostatku podataka. Međutim, kod bazične *Holdout* metode problem je što micanje dijela podataka za validaciju potencijalno može stvarati problem za podtreniranost. Odnosno, reduciranjem podataka za treniranje, pojavljuje se rizik da se izostave bitni uzorci ponašanja u skupu podataka, što će u konačnici povećati pogrešku koja je inducirana s pristranosti. Tu se koristi unakrsna validacija u K preklopa koja ostavlja dovoljno podataka za treniranje, ali i dovoljno podataka za validaciju. Kako je spomenuto ranije, definira se podjela ovisno o broju K i svaki se put jedan od K podskup koristi kao set za testiranje, a preostalih K-1 podskupova zajedno klasificiraju se za treniranje. Na ovaj način svaka točka skupa podataka će točno jednom biti u validacijskom setu, a u setu za treniranje K-1 puta. Ovako se značajno smanjuje pristranost jer se većina podataka koristi za treniranje, ali se smanjuje i varijanca jer se isto većina podataka koristi u validacijskom setu. [16]



Slika 2.11. Unakrsna validacija

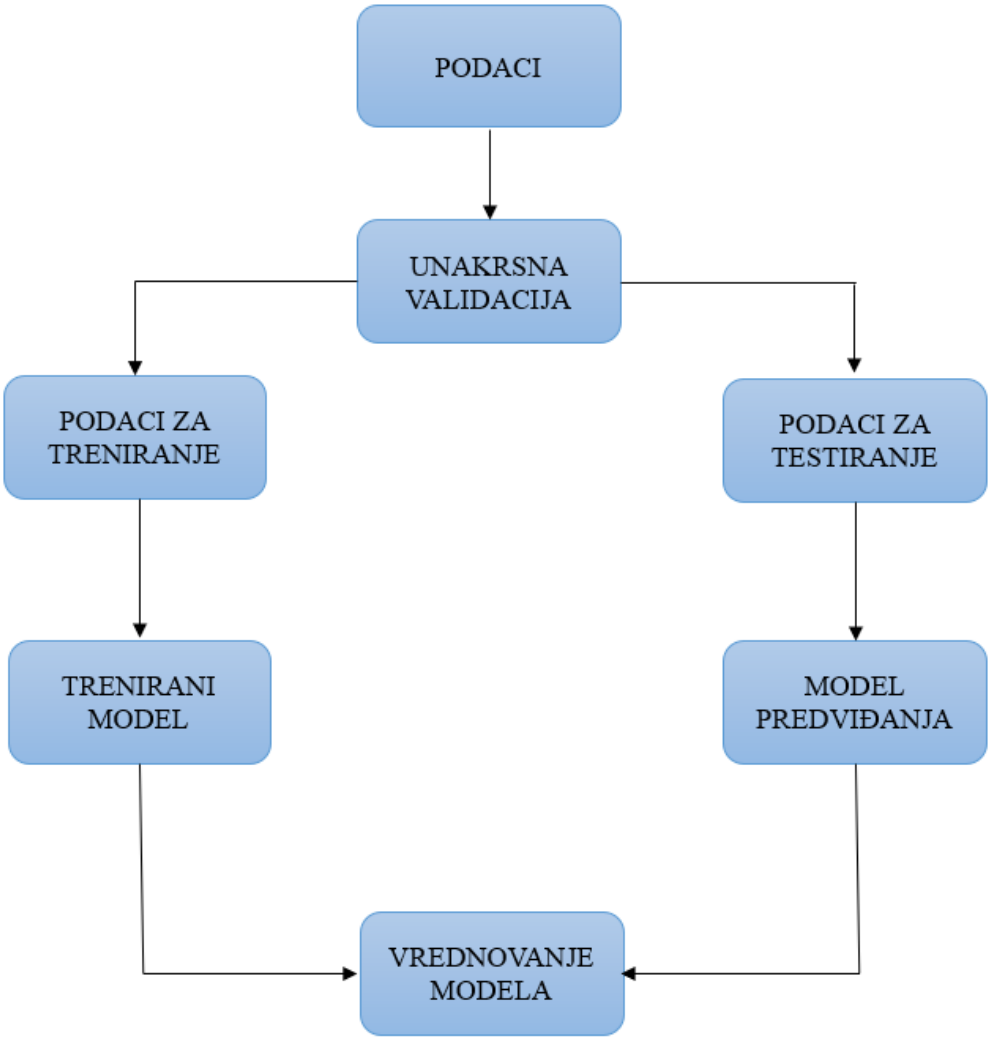
Na slici 2.11. dan je primjer unakrsne validacije u K preklopa kada je K = 5. Prilikom svake iteracije generirat će se pogreška. Prosjek svih pogrešaka uzima se kao rezultat. Izračun je dan formulom (2.8):

$$Pogreška = (1/n) * \sum pogreška_i \quad (2.8)$$

gdje je:

n broj iteracija

Na slici 2.12. dan je vizualni opis razvoja modela prilikom korištenja unakrsne validacije. Na cijeloj bazi podataka koristi se unakrsna validacija koja ovisno o definiranom broju K ponavlja podjelu na skup za treniranje i testiranje. U svakoj iteraciji model uči iz skupa podataka za treniranje, a predviđanja postavlja na skupu podataka za testiranje. Po razvoju modela, koriste se metrike vrednovanja kako bi se prikazale njegove performanse.



Slika 2.12. Razvoj modela predviđanja unakrsnom validacijom

3. FAZA TESTIRANJA

Faza testiranja i prikupljanja podataka za daljnju obradu trajala je nekoliko mjeseci. U eksperimentu su prikupljeni podaci od 50 različitih korisnika koji su sudjelovali u svakom dijelu ispitivanja. Istraženo je vrijeme potrebno za izvršavanje zadataka, a zadatak je definiran kao prolazak kroz tunel različitih konfiguracija. Konfiguracija tunela kombinira *level* testiranja koji je definiran kao lagan, srednji ili težak te predefinirane sinusoidne krivulje koje se razlikuju u amplitudama, odnosno „broju zavoja“. Testirano je interaktivno sučelje miša s kojim je bilo potrebno izvršiti definirane zadatke. Od korisnika se zahtijevalo da odabranim interaktivnim uređajem prođe kroz tunel što brže i što točnije je moguće. Svakom je korisniku na jednom primjeru pokazano kako zadatak izgleda i objašnjeno što se treba napraviti. Uvjeti testiranja isti su za svakog korisnika. Svaki je korisnik testirao na istom računalu s istim mišem.

3.1. Aplikacija i uvjeti testiranja

Aplikacija je trebala biti što jednostavnija kako bi se korisnik prilikom testiranja mogao usredotočiti samo na rješavanje zadataka. Također, sučelje je aplikacije napravljeno jednostavno i intuitivno jer sama aplikacija nije bila cilj ovog projekta već sredstvo da se prikupe podaci koji bi se mogli koristiti za izradu modela predviđanja. Tako su i zahtjevi same aplikacije definirani:

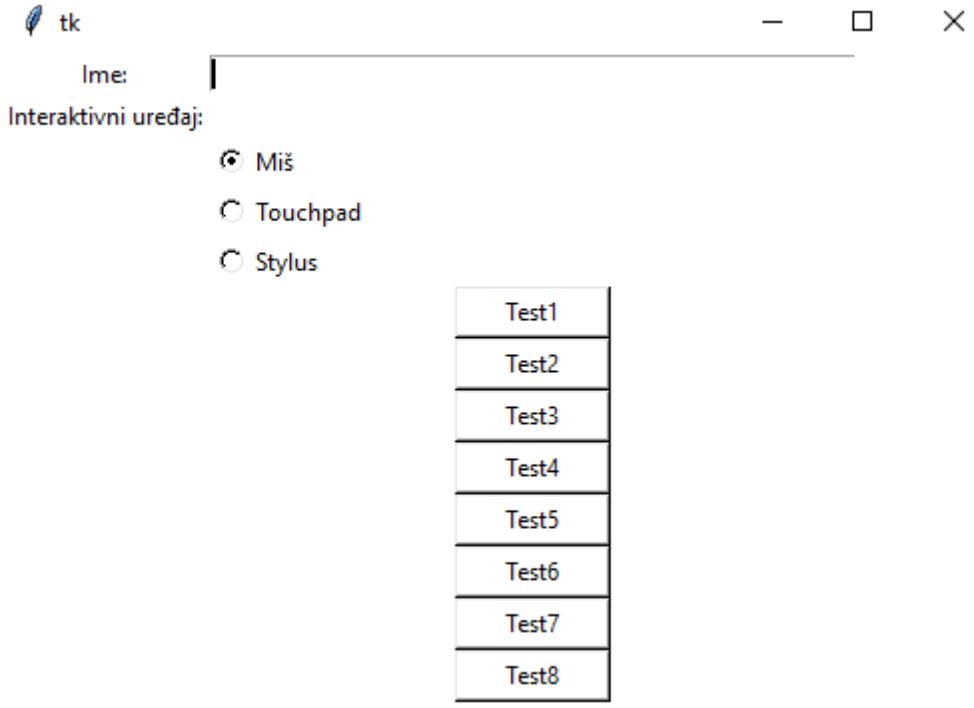
- Jednostavno sučelje samo s poljima i gumbima koji su nužni za funkcioniranje aplikacije
- Mogućnost prikupljanja informacija s korisničkog sučelja i zapisivanje unutar datoteke
- Oblik datoteke mora biti dokument formata Excel
- Automatsko generiranje tunela po testnom slučaju
- Slučajni redoslijed tunela po testnom slučaju
- Vidljivo označene granice tunela
- Vidljivo označen početak i kraj tunela
- Ponavljanje zadatka ukoliko se izađe van granica tunela
- Ponavljanje zadatka ako se otpusti tipka miša prije kraja tunela
- Izračun vremena potrebnog za rješavanje jednog zadatka
- Zapis vremena u datoteku formata Excel

- Mogućnost promjene boje programa
- Izračun i iscrtavanje idealne linije
- Mogućnost sviranja glazbe

Zahtjevi su implementirani koristeći programski jezik Python koji se pokazao kao dobro rješenje za izradu jednostavnog grafičkog sučelja, za analizu podataka, ali i za izradu modela predviđanja. Pri tome je jednostavan i ima široku dostupnost knjižnica koje se mogu koristiti. Za izradu grafičkog sučelja korišten je *Tkinter* grafički alat, dok se podaci prikupljeni prilikom testiranja spremaju pomoću knjižnice *Openpyxl* u dokument formata Excel. Izrada vizualnog okruženja za provođenje testnih slučajeva i prikaz grafičkih elemenata napravljena je s knjižnicom *Pygame*, uglavnom korištenoj u razvoju računalnih igara. Za matematičke funkcije i obrade te računanje vremena korištene su knjižnice *time* i *math*.

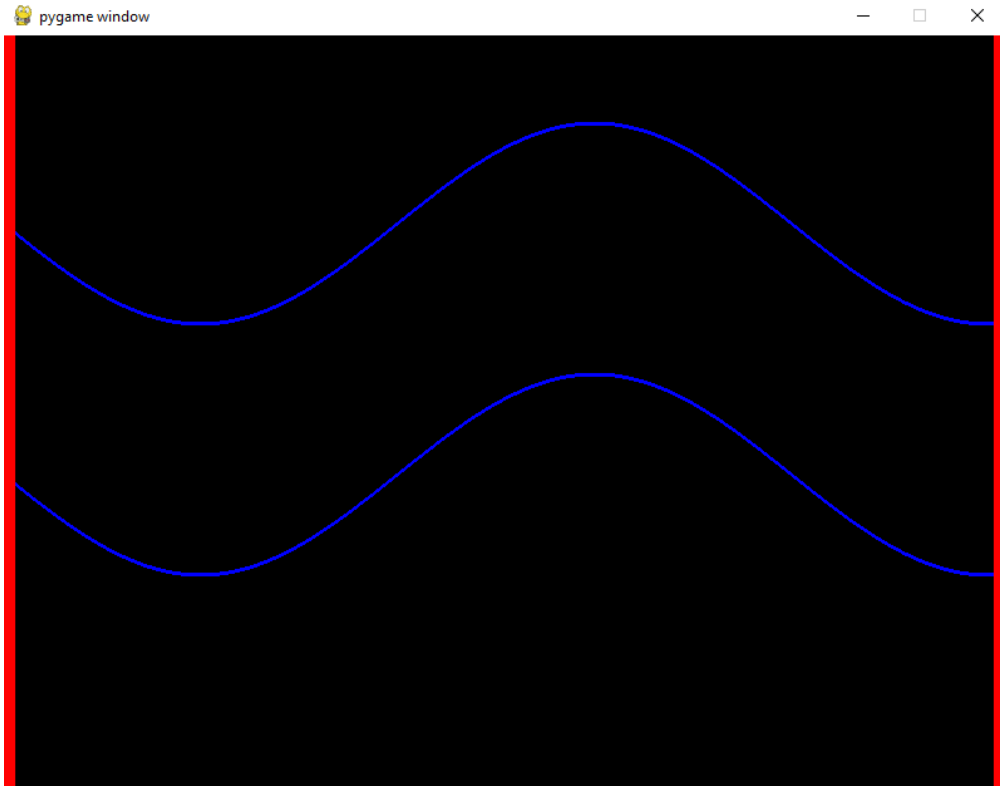
U ovom istraživanju mjerila se zavisna varijabla vremena. Vrijeme se mjeri od pritiska lijeve tipke miša do njenog puštanja po završetku prolaska kroz jedan tunel. Mjereno je u milisekundama što je kasnije preračunato u sekunde radi lakšeg rada s podacima. S obzirom da korisnik može unutar samo par sekundi proći kroz tunel, postoji mogućnost da se u tom kratkom vremenu neće zabilježiti svi pikseli trajektorije miša. Navedeni je problem česta pojava u računalnoj grafici. Isti se slučaj pojavio i na samom početku ovog projekta dok je aplikacija bila u fazi razvoja. Ako bi se kroz tunel prolazilo dovoljno sporo da se zabilježe svi (ili bar veći dio) piksela, vrijeme potrebno za rješavanje zadataka naraslo bi i preko 30 sekundi za zadatak za koji inače treba 5 sekundi. Ako bi se kroz tunel prošlo brže, jednostavno ne bi bilo moguće zabilježiti sve piksele u odnosu na brzinu korisnika. Kako bi se zabilježila cijela trajektorija miša napravljena je interpolacija točaka koristeći *Bresenham's* algoritam. Na taj se način interpolira razlika između trenutnog položaja miša i zadnje zabilježene pozicije miša. Na ovaj način, neovisno koliko se brzo napravi pokret mišem, trajektorija korisnika ostaje u potpunosti zabilježena. Umjesto oslanjanja na diskretna ažuriranja položaja miša, interpolacija osigurava da se zabilježe svi pikseli duž segmenta linije posljednje i trenutne pozicije miša. Ovim načinom iscrtava se linija koja je glađa i prilagođenija trajektoriji miša pri velikim brzinama. [17]

Kada se aplikacija pokrene otvara se grafičko sučelje koje sadrži: polje za upis imena korisnika, odabir interaktivnog uređaja kojim će se rješavati zadaci te odabir testnog slučaja kao što je prikazano na slici 3.1.



Slika 3.1. Izgled grafičkog sučelja

Kada se popune traženi podaci i odabere testni slučaj otvara se novi prozor sa zadatkom koji je potrebno riješiti. Izgled jednog od zadataka prikazan je na slici 3.2.

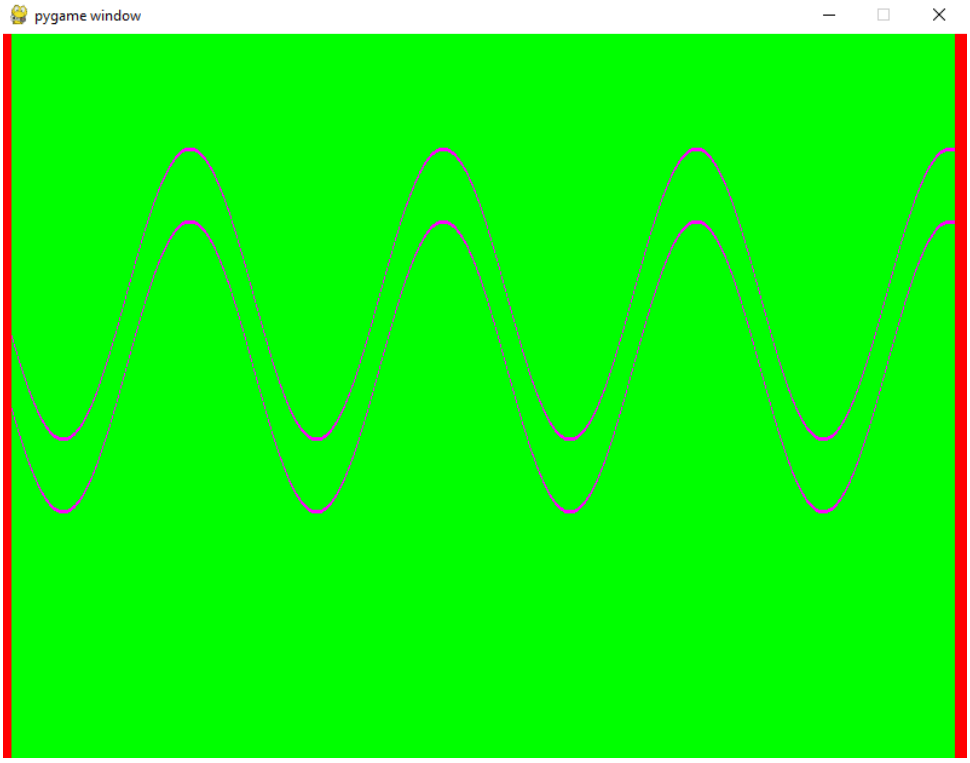


Slika 3.2. Primjer zadatka za rješavanje

Testnih slučajeva je osam od kojih svaki ima 15 zasebnih zadataka. Jedan zadatak sadrži tunel predefinirane težine. Težina se odnosi na širinu tunela definiranu u pikselima te na broj zavoja tunela, odnosno amplitude same krivulje. *Levela* težine je 3 – lagan, srednji i težak odnosno širok tunel, srednji i uzak. Krivulja je pet gdje se mijenja broj amplituda. Kombinacijom dvaju prethodnih parametara dobijemo 15 tunela, odnosno zadataka koje je potrebno riješiti. Kroz preostalih sedam testnih slučajeva ponavljaju se isti tuneli kojima je programski definiran slučajni odabir. Ono što razlikuje sve testne slučajeve je kombinacija preostalih tri uvjeta, a to su boja programa, idealna putanja i glazba. Svaki od tri uvjeti poprima dvije moguće vrijednosti te time dobijemo 8 kombinacija uvjeta, odnosno osam testnih slučajeva.

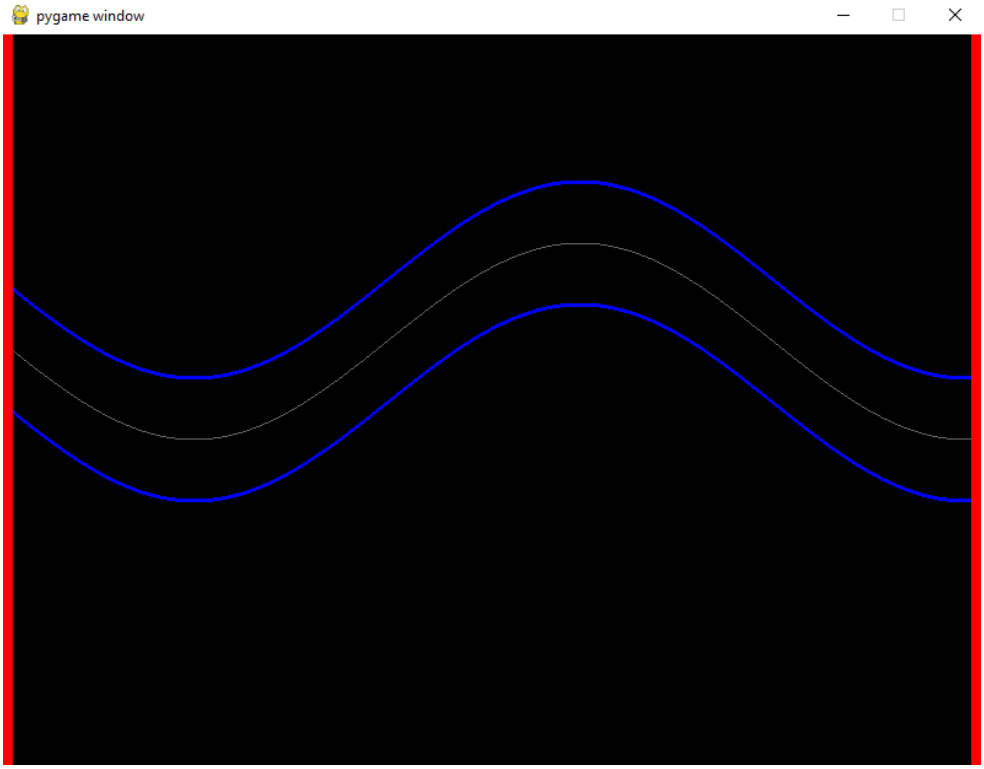
- Boja – uvjet je definiran kao normalna boja i kričava. Ideja je da se testira ako boja programa može utjecati na korisnika kako bi se sporije ili brže prolazilo kroz sam tunel. Uvjet gdje je boja definirana kao normalna ima crnu pozadinu, plave granice tunela, a linija kojom se prolazi je crvene boje. Uvjet gdje je boja definirana kao kričava, pozadina je

zelene boje, granice tunela su roze, a linija kojom se prolazi kroz tunel je žute boje. Primjer je vidljiv na slici 3.3.



Slika 3.3. Uvjet boje

- Idealna linija – odnosi se na uvjet iscrtane idealne linije koja označava putanju kojom bi se trebalo proći kroz tunel kako bi pogreška bila jednaka nuli. Idealna linija se nalazi točno na polovici granica tunela i prati njegove amplitude. Iscrtana je svjetlo sivom bojom kao što je prikazano na slici 3.4. Ideja ovog uvjeta je u jednostavnosti prolaska kroz tunel kada je predefinirana putanja prolaska. Uvjet isto poprima dvije vrijednosti: ili je idealna linija iscrtana ili ne.



Slika 3.4. Uvjet idealne linije

- Glazba – uvjet glazbe postavljen je da u pola slučajeva glazba postoji u pola ne. Korištena je pjesma koja je besplatno preuzeta s Interneta zbog *copyrighta*.

Tablični prikaz kombinacija uvjeta po testnim slučajevima dan je u tablici 3.1.:

Tablica 3.1. Kombinacije uvjeta

TESTNI SLUČAJ	BOJA	IDEALNA PUTANJA	GLAZBA
Test1	Normalno	Ne	Ne
Test2	Normalno	Ne	Da
Test3	Normalno	Da	Ne
Test4	Normalno	Da	Da
Test5	Kričavo	Ne	Ne
Test5	Kričavo	Ne	Da
Test7	Kričavo	Da	Ne
Test8	Kričavo	Da	Da

Svaki testni slučaj sastoji se od 15 zadataka koji su kombinacija težine tunela i broja zavoja, odnosno pripadajuće krivulje. Kako je težina tunela definirana u tri *levela* – lagano, srednje i teško, a broj zavoja je definiran u pet različitih krivulja, nastaje kombinacija od 15 zadataka po testnom slučaju. Kombinacija težine tunela i krivulja prikazana je tablicom 3.2.

Tablica 3.2. Kombinacija težine tunela i krivulja

TESTNI SLUČAJ	TEŽINA TUNELA	KRIVULJA
Test1	Lagano	Krivulja 1
Test1	Srednje	Krivulja 1
Test1	Teško	Krivulja 1
Test1	Lagano	Krivulja 2
Test1	Srednje	Krivulja 2
Test1	Teško	Krivulja 2
Test1	Lagano	Krivulja 3
Test1	Srednje	Krivulja 3
Test1	Teško	Krivulja 3
Test1	Lagano	Krivulja 4
Test1	Srednje	Krivulja 4
Test1	Teško	Krivulja 4
Test1	Lagano	Krivulja 5
Test1	Srednje	Krivulja 5
Test1	Teško	Krivulja 5

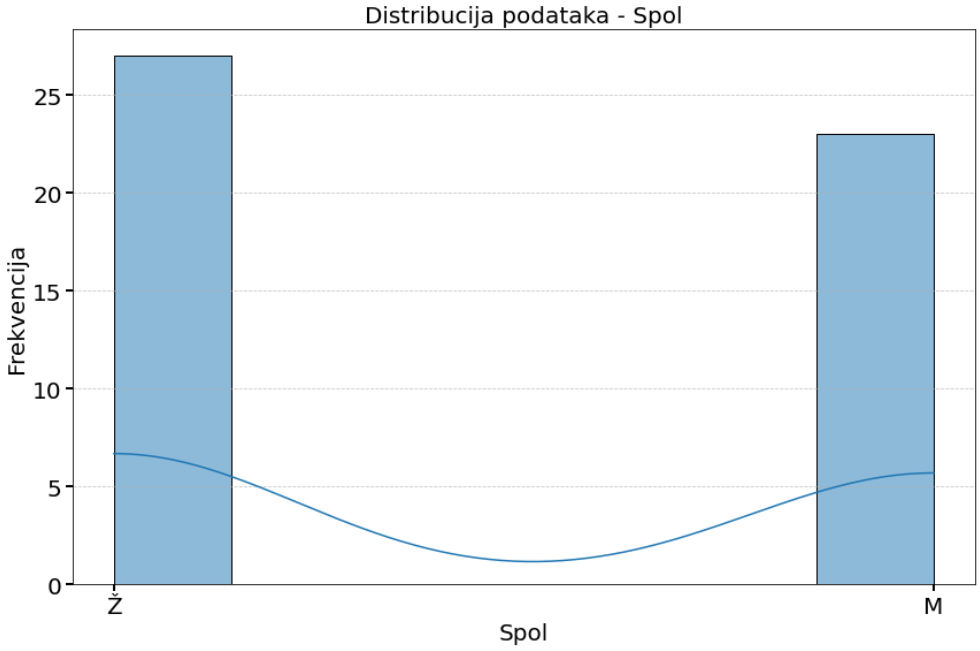
Za svaki testni slučaj ponavlja se prethodna kombinacija uvjeta težine *levela* i krivulje. To je baza ovog testiranja. Što znači da će i testni slučajevi od Test2 do Test8 imati istu kombinaciju uvjeta težine tunela i krivulje kao i Test1. Kako bi se proširilo područje testiranja i kako bi se testiralo pod više različitih uvjeta, uvedeni su uvjeti definirani u tablici 3.1. Tako se baza testiranja proširila na 8 različitih kombinacija uvjeta boje, idealne linije i glazbe. 15 zadataka po testnom slučaju rješava se u 8 definiranih uvjeta što kulminira sa 120 rezultata po testnom ispitaniku.

3.2. Oprema i programska podrška

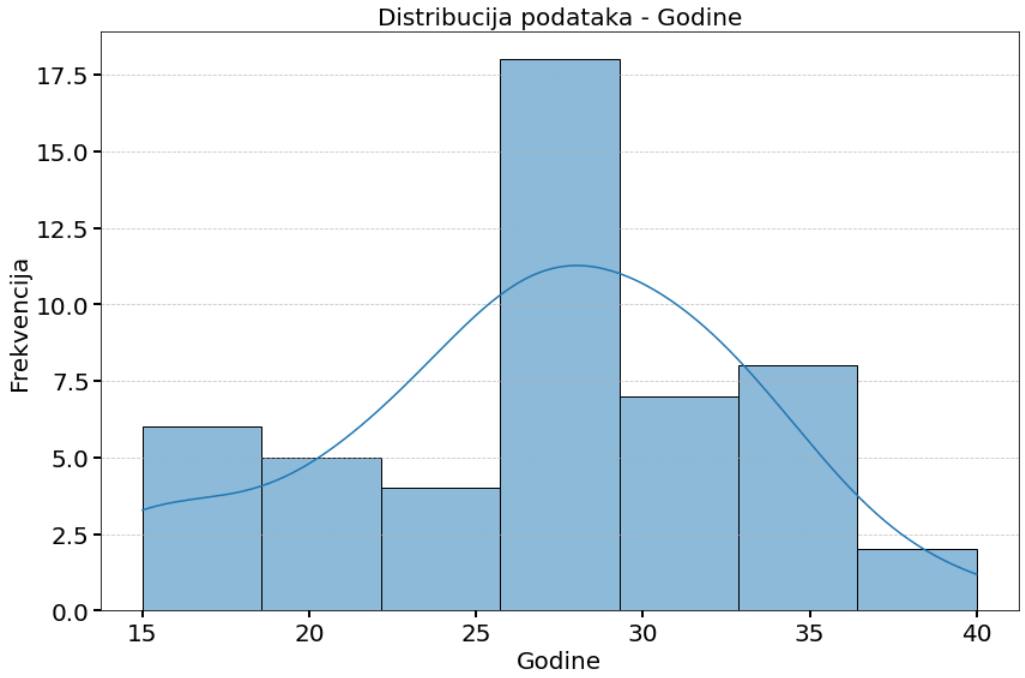
Za izvođenje eksperimenta potrebno je prijenosno računalo i miš. U ovom je slučaju korišten laptop HP 250 G7 Notebook te miš Logitech Pebble m350. Od programske podrške, potrebno je imati instaliran Python te knjižnice za isti – *math, time, openpyxl, pygame.* Python se pokazao kao validan izbor programskog jezika u strojnom učenju. Ima širok spektar knjižnica i alata koji olakšavaju rad u području strojnog učenja i implementaciji složenih modela. Pored toga, Python je jedan od programskih jezika koji ima čitljivu i intuitivnu sintaksu što ga čini iznimno privlačnim kako za početnike, tako i za stručnjake ovog područja. Upravo iz navedenih razloga, široke podrške i dostupnosti informacija, Python je odabran kao programski jezik za rad na ovom projektu.

3.3. Korisnici

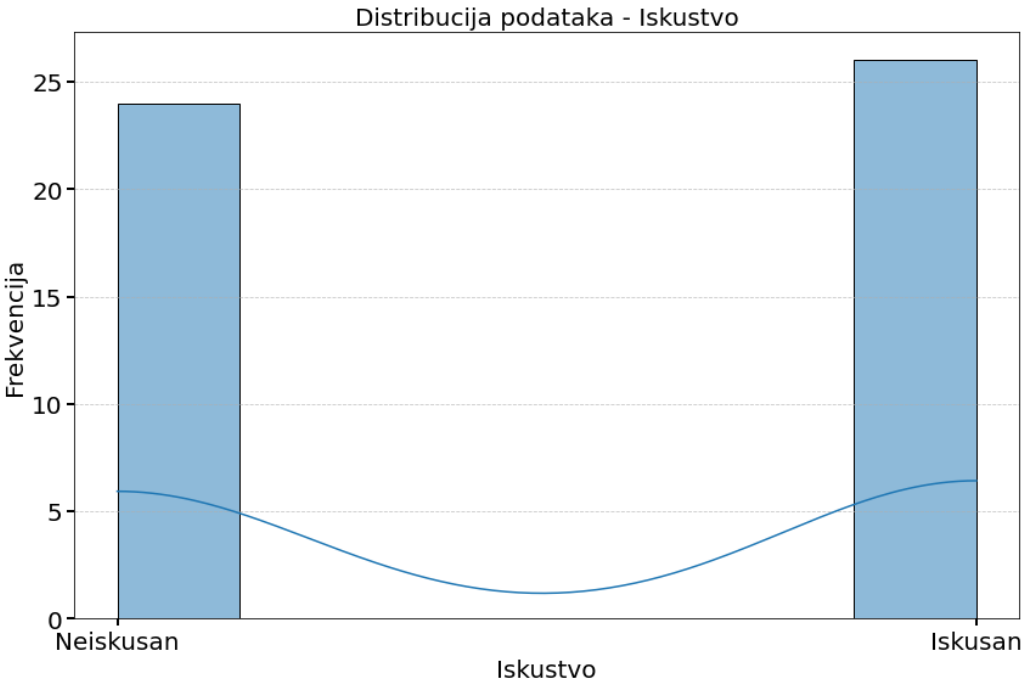
Za korisnike je napravljena distribucija podataka kako bi se za svaki od parametara vidjela podjela, odnosno kako su podaci raspoređeni za tu varijablu. Na slikama od 3.5. do slike 3.12. dan je grafički prikaz podjele podataka za spol, godine, iskustvo, orijentaciju ruke, posao, doba dana, naočale i daltonizam.



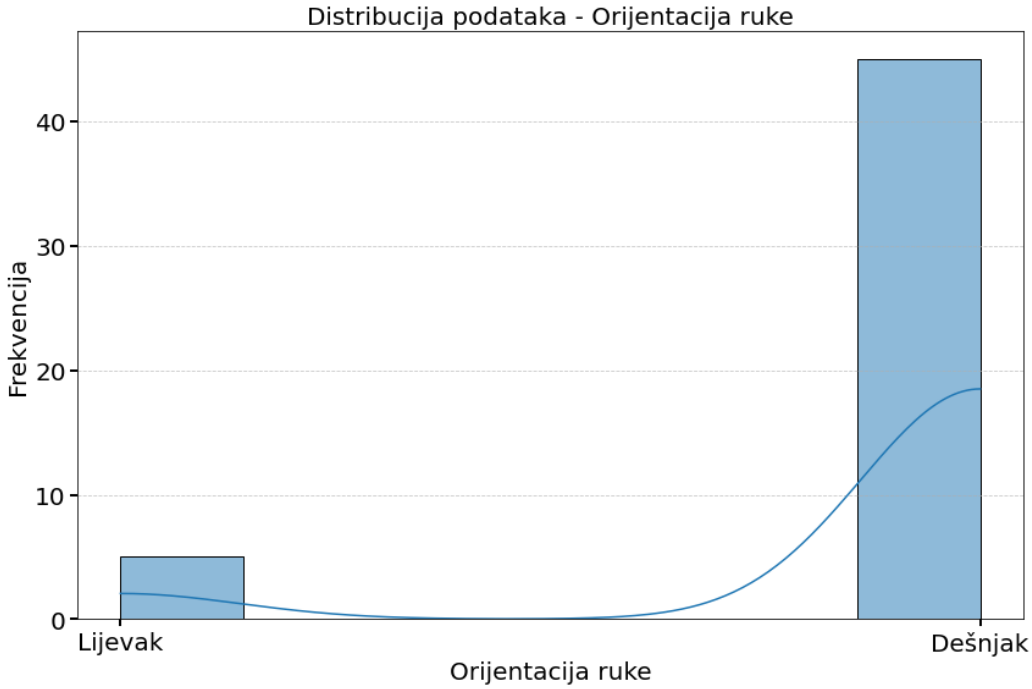
Slika 3.5. Distribucija podataka za spol ispitanika



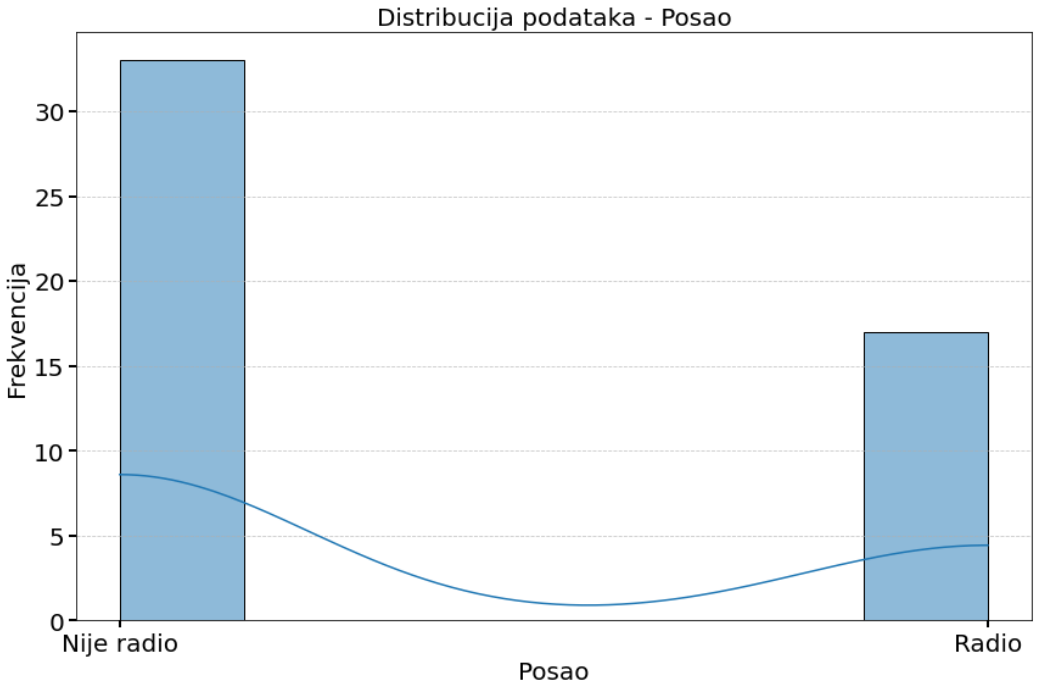
Slika 3.6. Distribucija podataka za godine ispitanika



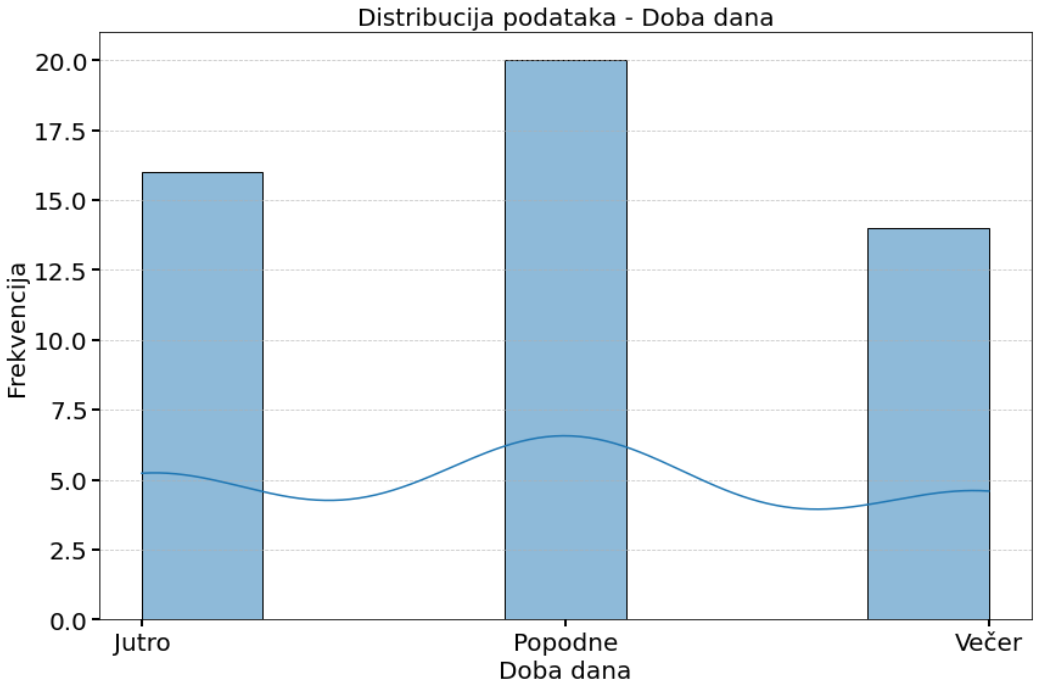
Slika 3.7. Distribucija podataka za iskustvo ispitanika



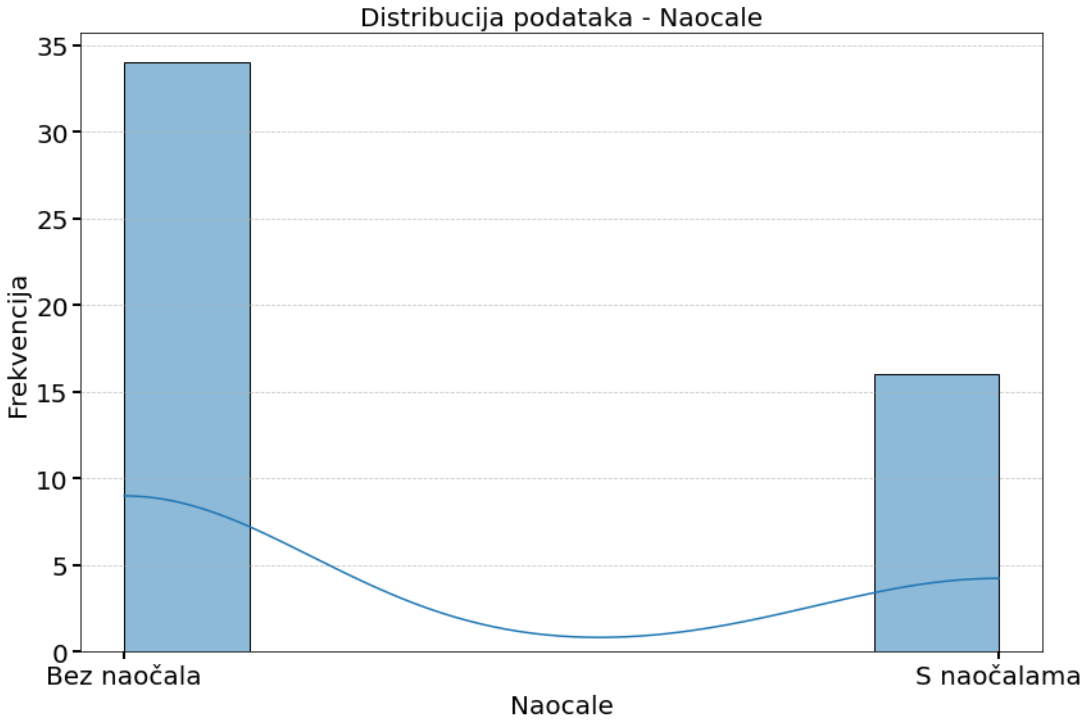
Slika 3.8. Distribucija podataka za orijentaciju ruke ispitanika



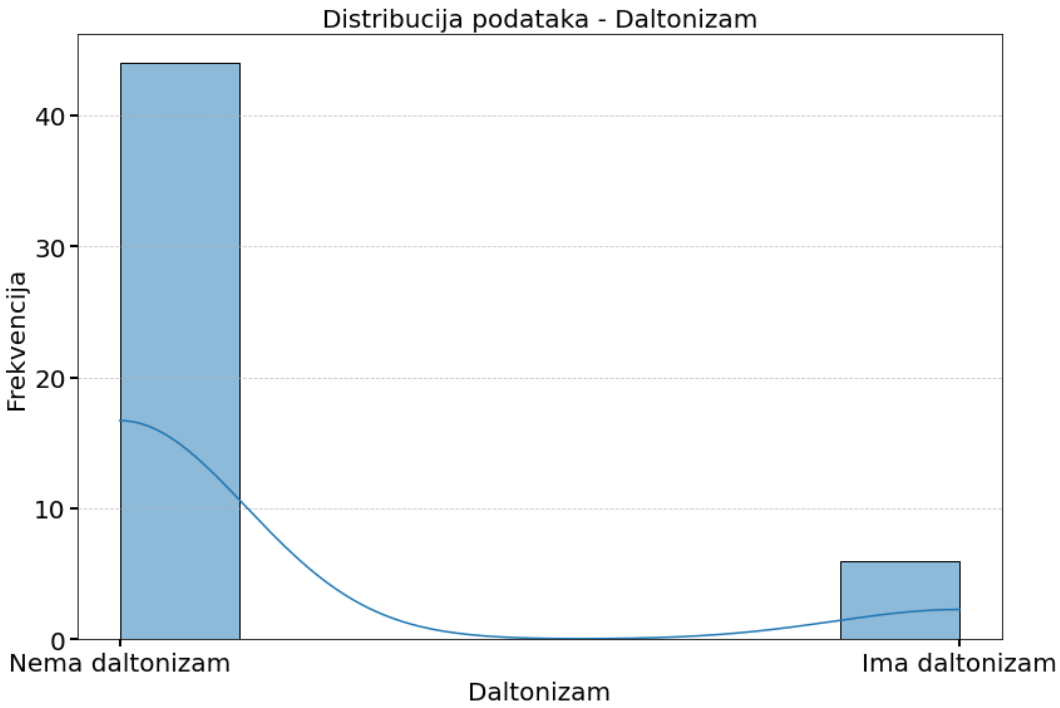
Slika 3.9. Distribucija podataka za posao ispitanika



Slika 3.10. Distribucija podataka za doba dana



Slika 3.11. Distribucija podataka za nošenja naočala kod korisnika



Slika 3.12. Distribucija podataka za daltonizam ispitanika

U testiranju je sudjelovalo 50 korisnika, od čega je 27 žena i 23 muškaraca, a raspon godina je od 15 do 40. Najveća je zastupljenost osoba od 25 godina. Iskustvo je definirano kao prethodno napredno korištenje miša ili kao izvanredno snalaženje u korištenju istog. To bi obuhvaćalo zanimanja iz područja informatičke struke, modeliranja, 3D crtanja i slično. S obzirom da prolazak mišem kroz tunel zahtjeva dobru koordinaciju pokreta, kao iskustvo tretiralo se i igranje video igara. Prema tome je 24 ljudi vođeno kao neiskusno, dok je preostalih 26 iskusan korisnik u korištenju miša kao pokaznog uređaja. Petero je osoba lijevak, od toga četiri žene i jedan muškarac, a svi preostali su dešnjaci. Svi korisnici koriste računalo na dnevnoj bazi, bilo zbog posla, škole/fakulteta ili hobija/zabave. Posao je definiran kao vrijednost ako je osoba taj dan prije testiranja radila. Sa grafa vidimo da 33 korisnika na dan testiranja nije obavljalo posao, što podupire činjenicu da je većina testiranja odrađena preko vikenda, a i neki od korinika niti nemaju posao. Doba dana je isto uzeto kao parametar testiranja te poprima tri vrijednosti – jutro, popodne i večer, ovisno u kojem je dijelu dana osoba odradila testiranje. Najviše je zastupljeno popodne kada je testiralo 20 korisnika, dok su jutro i večer podjednako zastupljeni - 16 korisnika za jutro, a 14 za večer. Za parametar testiranja uzeto je i ako korisnik ima problema s vidom, odnosno ako

nosi naočale ili ima daltonizam. Od toga 16 korisnika nosi naočale, a 6 ih ima daltonizam pri čemu su svi muškarci s tim problemom.

3.3.1. Percepcija problematike od strane korisnika

Tijekom i nakon testiranja, korisnici su dali svoje utiske vezano za problematiku testiranja. Većina korisnika požalila se na osjetljivost miša, teško pomicanje po podlozi jer je potreban lagan i manje zamjetan pokret dok je to teško izvedivo s mišem. Kao problem nameću se i trzaji ruke, znojne ruke, drhtanje dlanova. Također, zamijećena je ograničenost prostora ili loša koordiniranost pokreta iako je podloga na kojoj se radilo bila veličine 40x60 centimetara. Zbog loše koordiniranosti korisnik se dovede do kraja podloge i više nema prostora za pomicati miš te se mora vraćati na sam početak. Ako se korisnik uredno vrati na početak nakon svakog tunela, problem je smanjen, međutim, postojali su slučajevi kada korisnik ne prati situaciju, pa se dogodi da u tijeku testiranja određenog tunela više nema dovoljno mjesta za pokret. To dovodi do pogreške u testiranju te potrebe ponavljanja testiranja određenog tunela. Još jedan veći problem je ukočenost ruke. Prilikom testiranja korištene su pauze kako bi se testni ispitanici odmorili i kako umor ne bi utjecao na izvršavanje zadataka. Međutim, 120 tunela je svejedno za većinu korisnika predstavljalo problem jer su se tijekom testiranja i po završetku žalili na bolan i ukočen dlan i podlakticu.

3.4. Koraci testiranja

Aplikacija se pokreće iz naredbenog retka (*cmd*), navigirajući do datoteke gdje se ista nalazi i pokretanjem pomoću naredbe *python* login.py. Otvara se grafičko sučelje na kojemu se odabire određeni Test ovisno o predefiniranom redoslijedu testiranja za svakog korisnika. Prethodno je napravljena skripta za generiranje nasumičnih testnih slučajeva te je za svakog korisnika predefiniran nasumični redoslijed koji je bilo potrebno slijediti prilikom izvršavanja zadataka. Kako je već spomenuto, *levela* težine su tri, a krivulja je pet. Međusobnom kombinacijom istih dobijemo 15 tunela po testnom slučaju. Testnih slučajeva je osam što nam daje sveukupno 120 rezultata po korisniku. Na odabir testnog slučaja otvara se prozor unutar kojeg je potrebno izvršiti zadatak. Vrijeme mjerenja počinje u trenutku pritiska tipke miša, a završava

kada se tipka pusti. Po završetku jednog tunela, program automatski prelazi na drugi tunel. Prilikom prolaska kroz tunel potrebno je zadovoljiti dva uvjeta:

- Ne smije se izaći van granica tunela jer se zadatak prekida i potrebno ga je ponoviti,
- Ne smije se pustiti miš prije kraja tunela (crveno označena linija/granica) jer se zadatak prekida i potrebno ga je ponoviti.

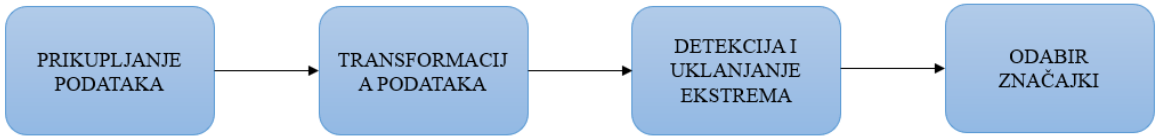
Po završetku jednog testnog slučaja, potrebno je odabrati sljedeći testni slučaj kako je predefinirano u uputama za tog korisnika. Testiranje je završeno kada za svakog korisnika postoji svih 120 zapisa (osam testnih slučajeva po 15 krivulja).

4. OBRADA PODATAKA

Obrada podataka i odabir varijabli su temeljni koraci u strojnom učenju koji se moraju napraviti kako bi se poboljšala kvaliteta podataka i same performanse modela. Korake možemo podijeliti na [18]:

- Prikupljanje podataka – ovisno o načinu prikupljanja podataka, odnosno kako sam projekt dozvoljava, slaže se skup podataka. U ovom je slučaju napravljeno testiranje sa 50 korisnika te su na taj način prikupljeni svi potrebni podaci. Dio podataka prikupljen je programski prilikom rješavanja zadataka, a drugi dio podataka prikupljen je kroz razgovor s korisnikom. Izolirane su sve subjektivne procjene te su korišteni samo utemeljeni podaci. Prilikom rješavanja zadataka, za svakog se korisnika kreirao njegov dokument formata *Excel* u koji su se automatski zapisivali njegovi rezultati, odnosno vrijeme potrebno da riješi određeni zadatak. Svi zasebni dokumenti formata *Excel* sklopljeni su u jednu *csv* bazu podataka koja se dodatno dalje obrađivala,
- Dopuna nedostajućih podataka – standardni je korak provjeriti ako postoje prazna polja u bazi podataka, odnosno vrijednosti koje su ništavne. Takve je vrijednosti potrebno identificirati te nadomjestiti ako je moguće. Mogu se koristiti jednostavnije metode poput medijana ili kompliciranije tehnike poput interpolacije podataka. U ovom slučaju, nije se koristilo ništa od navedenog jer nedostajuće vrijednosti nisu postojale,
- Čišćenje podataka – brisanje duplikata kako bi se spriječili pristrani rezultati, ispravljanje netočnosti ili pogreška prilikom izrade baze podataka,
- Transformacija podataka – svi podaci bi trebali biti na zajedničkoj ljestvici, što zahtjeva normaliziranje i standardiziranje podataka. U ovom se slučaju varijabla vrijeme računala u milisekundama te je kasnije preračunata u sekunde radi jednostavnosti i lakšeg prikaza. Za ostale varijable napravljene su klase jer sve varijable osim varijable godina, poprimaju jednu od 2-5 mogućih vrijednosti,
- Otkrivanje i rukovanje ekstremnim vrijednostima – identificiranje ekstrema koristeći statističke metode. Odluka o rukovanju ekstremnim vrijednostima donosi se ovisno o njihovom utjecaju na model i konačne rezultate – mogu se ukloniti, transformirati ili ostaviti,
- Izračun novih značajki – ako je moguće od postojećih značajki izraditi nove kako bi se prikupilo više informacija,

- Odabir značajki - Prije izrade modela predviđanja bitno je definirati koje su nezavisne varijable od veće važnosti kako bi se potencijalno kreirao model veće točnosti. Očekivano je da ne utječu sve nezavisne varijable jednako i da će neke biti manje, a neke od veće važnosti za ciljanu varijablu koja se predviđa.



Slika 4.1. Redoslijed obrade podataka

Slika 4.1. prikazuje korake koji su bili potrebni za ovaj projekt. Istraženi su svi prethodno opisani koraci, međutim, u slučaju ovog projekta neke od koraka nije bilo potrebno napraviti. U nastavku su detaljno opisani koraci koji su napravljeni te koji je njihov rezultat.

4.1. Transformacija podataka

Kako bi se razvio model što većih točnosti i radi jednostavnosti korištenja podataka i rada s istima, potrebno je svesti korištene varijable na neku zajedničku ljestvicu. U ovom je slučaju varijabla godine jedina koja poprima stvarnu vrijednost, vrijeme je preračunato u sekunde, a ostale su ostale vrijednosti transformirane u klase. Transformacije podataka prikazane su tablicom 4.1.:

Tablica 4.1. Transformacija podataka

PARAMETAR	VRIJEDNOST	TRANSFORMACIJA VRIJEDNOSTI
SPOL	Muško	0
	Žensko	1
ISKUSTVO	Ne	0
	Da	1
ORIJENTACIJA RUKE	Lijevak	0
	Dešnjak	1
KRIVULJA	Krivulja 1	1
	Krivulja 2	2
	Krivulja 3	3
	Krivulja 4	4
	Krivulja 5	5
LEVEL	Lagano	1
	Srednje	2
	Teško	3
BOJA	Normalno	0
	Kričavo	1
IDEALNA PUTANJA	Ne postoji	0
	Postoji	1
GLAZBA	Ne postoji	0
	Postoji	1
POSAO	Nije radio	0
	Radio	1
DOBA DANA	Jutro	1
	Popodne	2
	Večer	3
NAOČALE	Ne nosi	0
	Nosi	1
DALTONIZAM	Nema	0
	Ima	1

4.2. Detekcija ekstrema

Detekcija ekstrema unutar skupa podataka korak je koji može značajno utjecati na analizu i na sam model koji će se kreirati. Odnosi se na detekciju podataka koji značajno odskaču od ostalih točaka u skupu podataka. Ukoliko se ekstremi ne detektiraju i ne uklone mogu rezultirati modelom koji će se ili previše dobro ili premalo prilagođavati, a niti jedna od tih opcija ne bi smjela biti slučaj. Kako bi se ovo spriječilo koriste se neke od metoda za otkrivanje ekstrema, kao: *Z*-rezultat *(eng. Z-score),* Interkvartilni raspon *(eng. Interquartile Range,* IQR), Lokalni faktor ekstrema (eng. *Local Outlier Factor*, LOF). Odabrane su metode interkvartilnog raspona i z-rezultat jer su se prilikom istraživanja pokazale kao najučestaliji odabir za detektiranje ekstrema i njihovo uklanjanje. S time da se Z-rezultat metoda adaptirala za upotrebu na podacima čija distribucija ne prati normalnu raspodjelu. S obje su metode uklonjeni ekstremi te je napravljena baza podataka bez istih. Ovisno o metodi, ekstremi su detektirani drugačije i time su isti uklonjeni, što znači da su u konačnici dobivene dvije različite baze podataka bez ekstrema.

Princip rada Z-*Score* metode temelji se na mjerenju broja standardnih odstupanja podatkovne točke udaljenje od srednje vrijednosti. Točke koje imaju Z-rezultat veći od definiranog praga tretiraju se kao ekstremi te su posljedično uklonjeni iz baze podataka. [19] Osnovna Z-*Score* metoda računa se prema formuli (4.1):

$$z = (x-\mu)/\sigma \qquad (4.1)$$

gdje je:

- x podatkovna točka
- μ srednja vrijednost
- σ standardna devijacija populacije

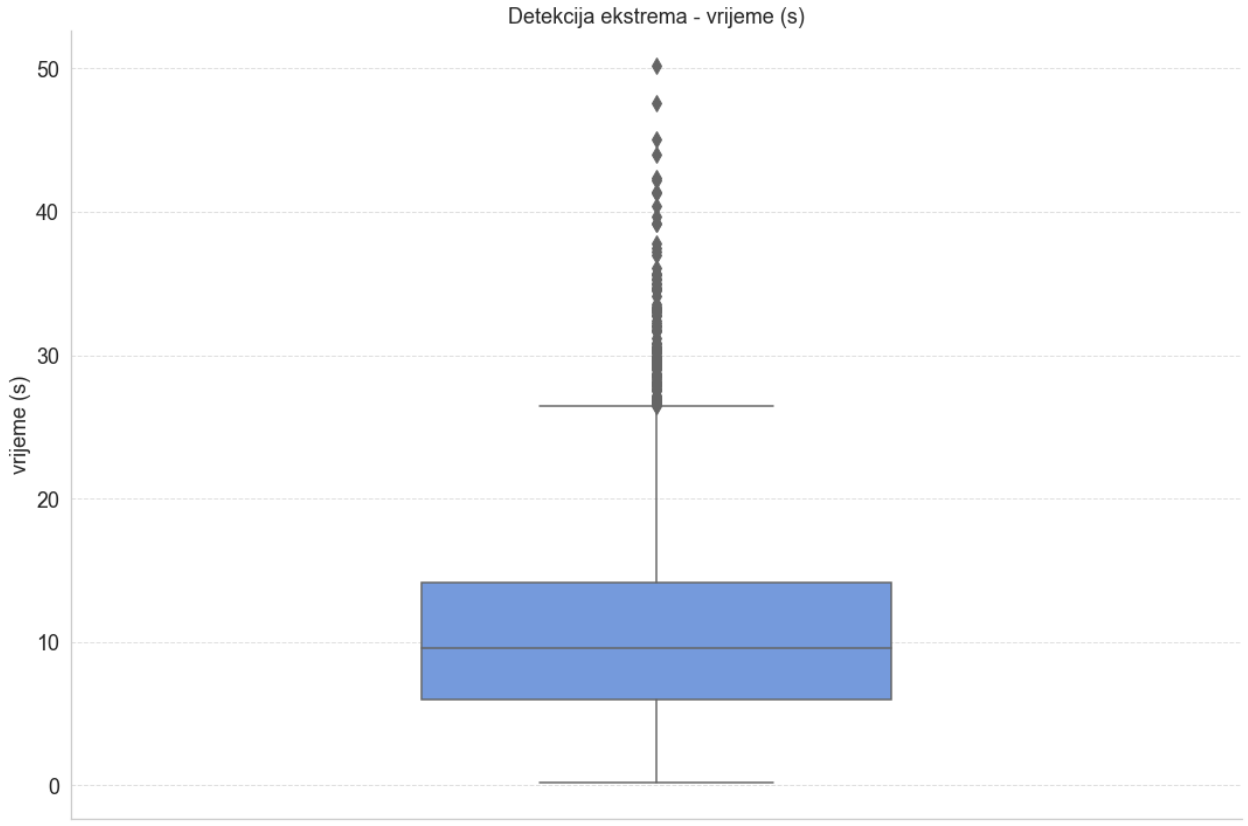
Vrijednost Z-rezultata indicira za podatkovnu točku koliko je standardnih odstupanja udaljena od srednje vrijednosti. [20]

IQR metoda je izbor za podatke koji ne odgovaraju normalnoj distribuciji. Predstavlja razliku između prvog i četvrtog kvartala distribucije, odnosno, mjeru širine distribucije s obzirom da je to dio koji sadržava pola podataka. Točka će se identificirati kao ekstrem ukoliko zadovoljava jedan od sljedećih uvjeta:

- Veće od trećeg kvartala + 1.5 IQR,
- Manje od prvog kvartala – 1.5 IQR.

S obzirom da detekcija ekstrema koja leži na mjeri koja može ovisiti o samim ekstremima, kao što je primjer srednje vrijednosti, uvodi se koeficijent IQR koji iznosi 1.5 čime se smanjuje ranjivost kvartala u prisutnosti ekstrema. [21]

Sa slike 4.2. vidimo distribuciju podataka i detektirane ekstreme. Pravokutnik plave boje označava podatke između prve četvrtine i treće četvrtine podataka. Linija na sredini pravokutnika je podjela podataka na 50%. Vodoravne linije van pravokutnika označavaju minimum i maksimum baze podataka, a točke koje se nalaze ispod minimuma ili iznad maksimuma označavaju potencijalne ekstreme. [22]



Slika 4.2. Detektirani ekstremi

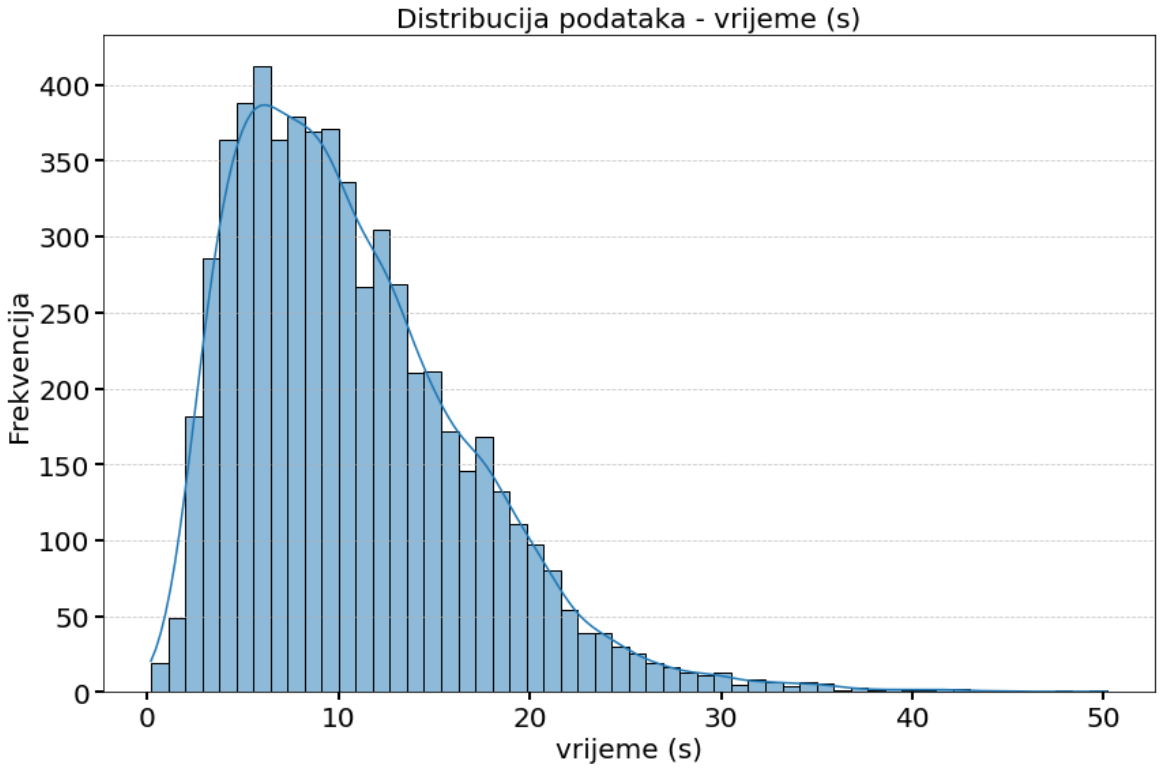
S obzirom da je prikupljanje podataka provedeno u kontroliranim uvjetima, dobivena je baza podataka koja nije imala nedostajuće ili ništavne vrijednosti. Ekstremi koji su identificirani na slici

4.2. rezultat su prirodne varijacije korisnika, a ne pogreška. S obzirom da korisnici variraju u rasponu godina, kao i u iskustvu te koordinaciji pokreta, dobiveni su rezultati isto tako varijacija njihovih sposobnosti. Prema tome, zaključujem da su detektirani ekstremi prirodni (istiniti) ekstremi te su iz tog razloga ostavljeni u bazi podataka na kojoj je daljnje rađen model predviđanja. Informativno su testirani algoritmi i na bazi bez ekstrema te su rezultirali lošijim modelom predviđanja te ti rezultati nisu opisani u nastavku ovog rada.

4.3. Distribucija podataka

Distribucija se odnosi na raspodjelu vrijednosti u skupu podataka. Definira se kao funkcija koja specificira sve moguće vrijednosti za varijablu te kvantificira relativnu učestalost, odnosno, vjerojatnost učestalosti pojavljivanja. Analiza distribucije široka je u statistici kako bi se organizirali sirovi podaci i prikazali grafovima radi lakšeg razumijevanja podataka i njihove daljnje upotrebljivosti. [23]

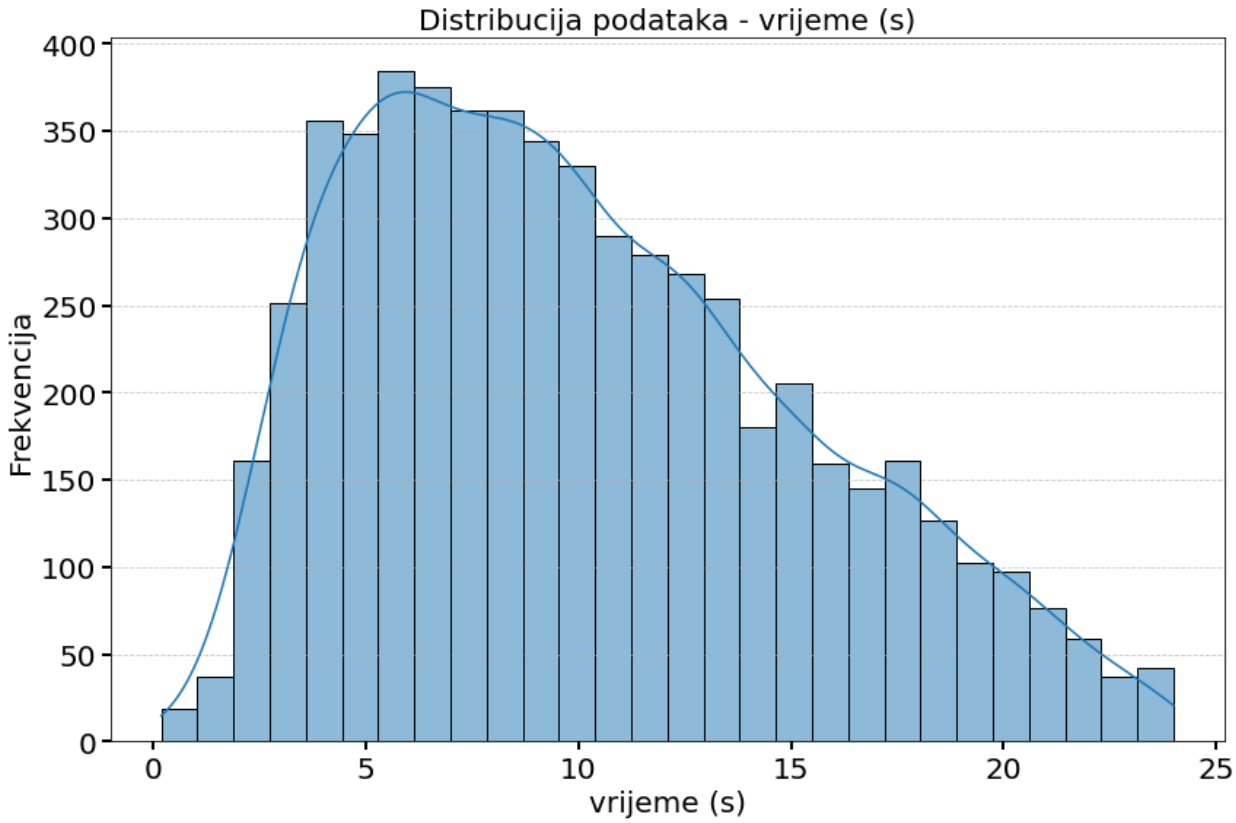
U nastavku, na slici 4.3, dan je prikaz distribucije podataka za varijablu vrijeme:



Slika 4.3. Distribucija podataka za ciljanu varijablu vrijeme

Sa slike 4.3. možemo uočiti kako vrijeme poprima vrijednosti u intervalu od 0 do 50, s time da su najzastupljenije vrijednosti do 30 sekundi. Frekvencija označava učestalost pojavljivanja određenog vremenskog perioda.

Distribucija varijable vrijeme nakon detekcije i brisanja ekstrema prikazana je na slici 4.4.:



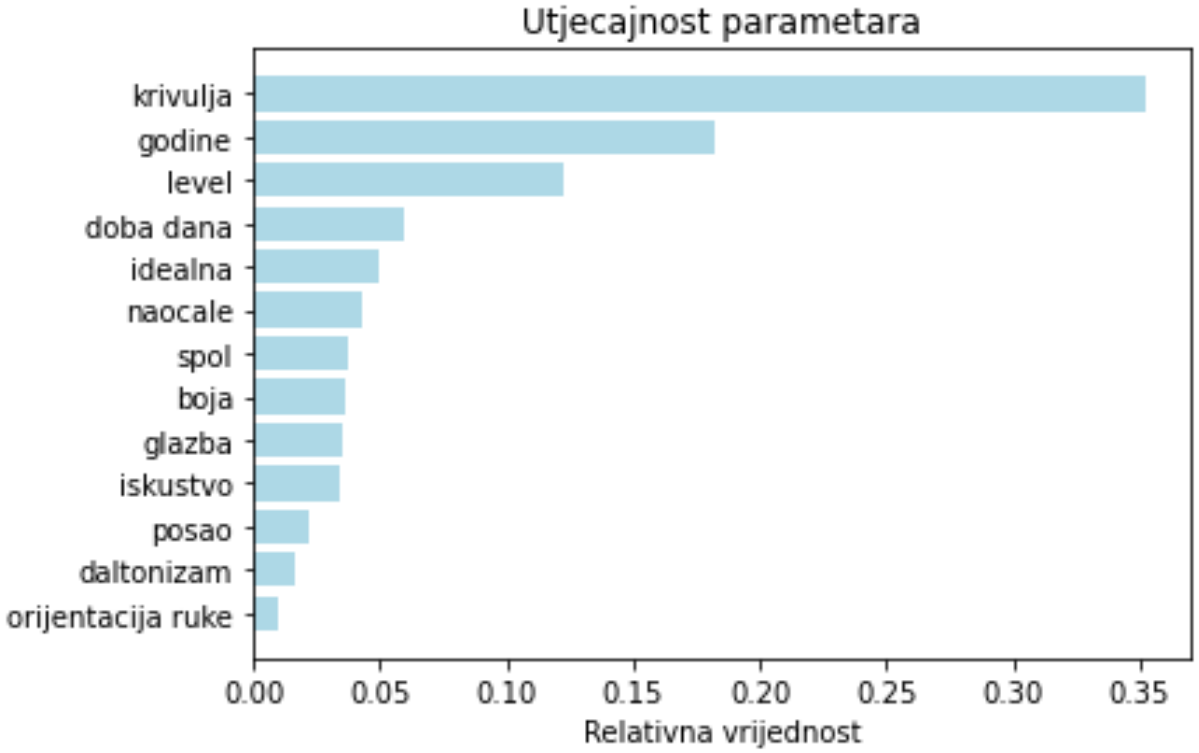
Slika 4.4. Distribucija podataka za varijablu vrijeme nakon uklanjanja ekstrema

Ako usporedimo grafove, jasno se vidi kako vrijeme sada poprima vrijednosti do 25, a da su sve vrijednosti iznad toga uklonjene iz baze podataka. Ovaj je graf dan kao prikaz vizualnog stanja baze podataka za varijablu vremena nakon uklanjanja ekstrema. S obzirom da su opisane metode uklanjana ekstrema i iste su provedene, dan je i graf distribucije vremena kao rezultat. Međutim, kako je prethodno opisano, ekstremi su ostavljeni u bazi podataka i tretirani su kao valjane vrijednosti u nastavku projekta.

4.4. Utjecajnost parametara

Utjecajnost parametara računa se kako bi se dobio rezultat za svaki parametar koji prikazuje njegovu važnost za model. Što je rezultat veći, veći će biti utjecaj tog parametra na model koji će se koristiti za predviđanje ciljane varijable. [24]

Ova tehnika omogućuje bolje razumijevanje podataka i njihovih međusobnih veza s ciljanom varijablom. Također, lako je prepoznati koji su parametri od manjkave važnosti za model. S obzirom da će se neki parametri moći odbaciti, odnosno ne koristiti za daljnji model, sam model postaje jednostavniji jer se smanjuje njegova dimenzionalnost i brže će se učiti što će u konačnici povećati performanse modela.



Slika 4.5. Utjecajnost parametara

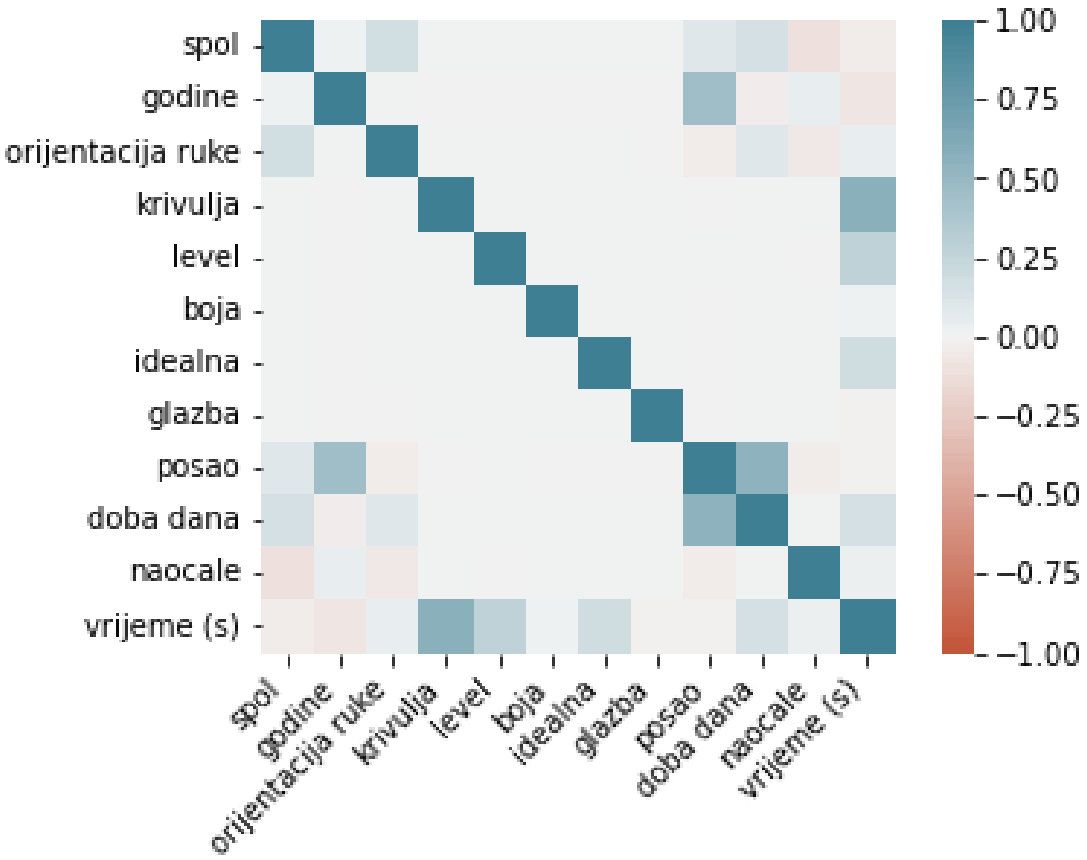
Sa slike 4.5. vidljivo je da najveći utjecaj ima parametar krivulja, a potom godine i *level*. Najmanji utjecaj ima orijentacija ruke, daltonizam i posao.

Prilikom izgradnje modela, korišteni su svi parametri, ali su isto tako testirane i određene kombinacije parametara koje su dale najbolje rezultate. Kombinacije parametara su napravljene

prema njihovom značaju na sam model. Tako je napravljena jedna baza podataka koja je uključivala sve parametre, a potom druga baza koje je sadržavala samo sljedeće parametre: krivulja, godine, *level*, doba dana, idealna putanja, i naočale. Preostali parametri izbačeni su iz ovog skupa podataka zbog svoje male utjecajnosti. Svi su korišteni algoritmi testirani na oba seta podataka, međutim kada su uklonjeni parametri manje utjecajnosti, konačni model nije rezultirao većom točnosti. Štoviše, rezultati su bili gotovo jednaki ili za 2-3% lošiji. Upravo se iz tog razloga drugi set podataka odbacio te su prikazani rezultati na cijeloj bazi podataka, odnosno uključujući sve parametre.

4.5. Korelacija parametara

U strojnom učenju od iznimne je važnosti definirati međusobno djelovanje parametara i njihovo doprinošenje modelu. Jedna od tehnika je korelacija koja se odnosi na mjeru linearnog odnosa između različitih parametara koji se promatraju. Ako dvije varijable međusobno imaju visoku korelaciju znači da se njihove vrijednosti mijenjaju u sličnom omjeru, odnosno, kako se povećava vrijednost jedne varijable, raste i druga te obratno. Korelacija se izražava koeficijentom korelacije koji poprima vrijednosti od -1 do 1, odnosno potpuna negativna korelacija i potpuna pozitivna korelacija. [25]



Slika 4.6. Korelacija parametara

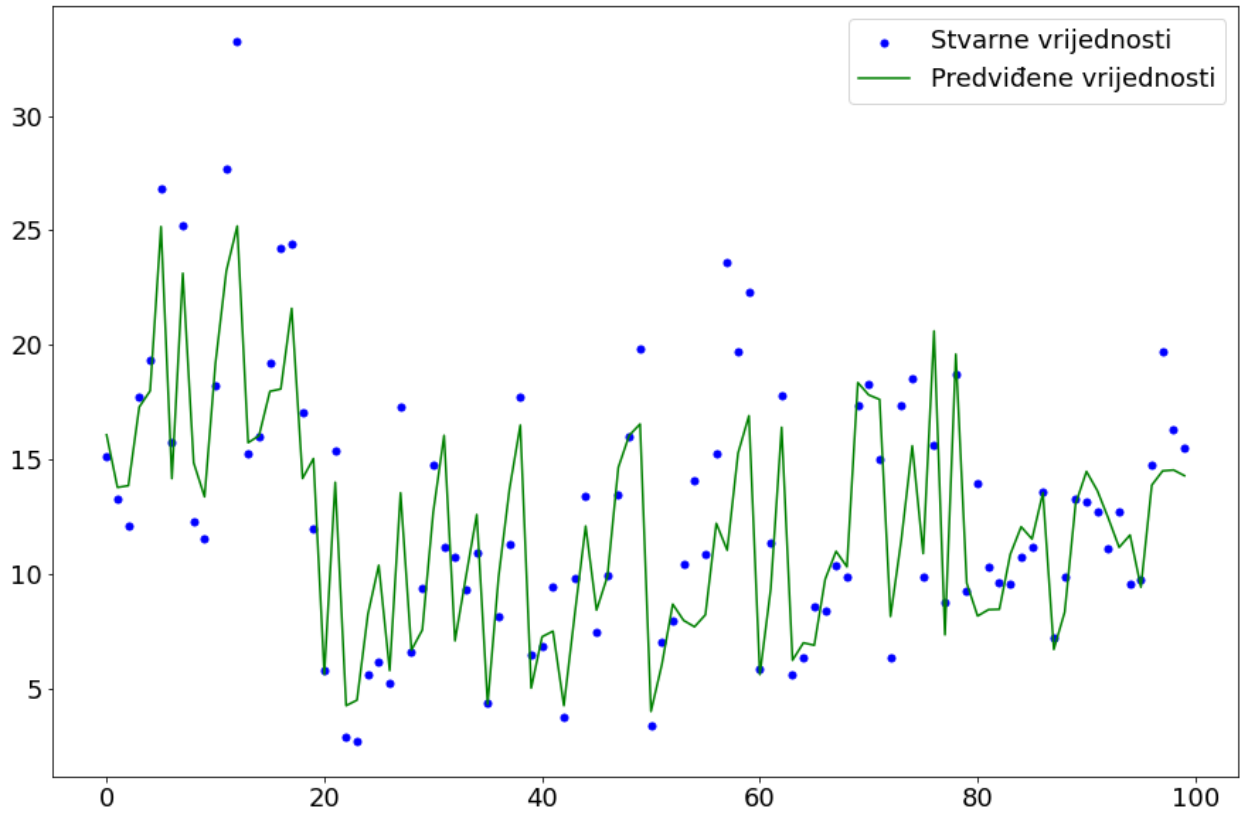
Sa slike 4.6. vidimo da vrijeme rješavanja zadatka jako korelira s krivuljom, a manje s *levelom*, uvjetom idealne linije i dobom dana u kojem je provedeno testiranje. Što znači da krivulja i *level*, odnosno težina tunela značajnije utječu na vrijeme rješavanja zadataka, isto kao i idealna linija, dok preostale varijable imaju manji utjecaj na vrijeme. Prema tome, kako se mijenja težina tunela ili postojanje idealne linije, tako se mijenja i vrijeme. Ako je vrijeme sporo, tunel je kompliciraniji, ako postoji idealna linija, vrijeme rješavanja zadataka biti će brže te obratno.

5. REZULTATI

Prilikom odabira algoritama za izradu modela prethodno je definirana vrsta problema te su potom za taj problem istraženi algoritmi koji se koriste i koji bi potencijalno mogli dati dobre rezultate. Algoritmi koji su istraženi su: linearna regresija, polinomna regresija, *Bayes*, *Gaussian*, slučajna šuma, stablo odluke, metoda najbližih susjeda, *Ridge* regresija i *Lasso* regresija. U nastavku su obrađeni i prikazani rezultati za algoritme koji su dali najbolje rješenje – metoda najbližih susjeda, algoritam slučajne šume i polinomna regresija. Prilikom odabira algoritama isti su testirani u više različitih slučajeva i s različitim parametrima, a prikazana je ona kombinacija koja je u konačnici dala najbolje rezultate. Također, u obzir su osim točnosti uzete i druge metrike vrednovanja koje su prethodno opisane. S obzirom da su algoritmi korišteni zajedno s unakrsnom validacijom, sve metrike vrednovanje prikazane su kao srednja vrijednost zbroja određene metrike ovisno o definiranom broju iteracija.

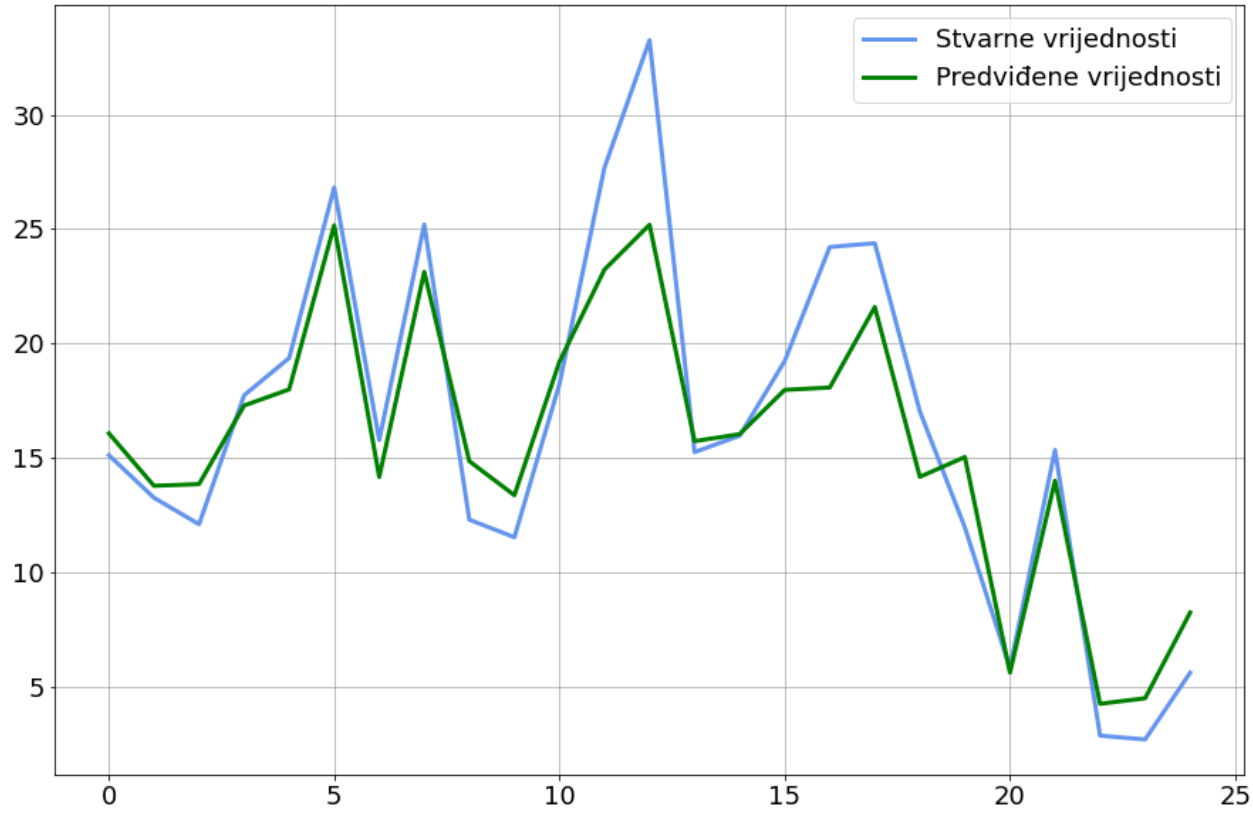
5.1. Usporedba rezultata

Na slici 5.1. dan je grafički prikaz stvarnih vrijednosti i vrijednosti koje su predviđene kNN algoritmom. Plavim su točkama označene stvarne vrijednosti prikupljene testiranjem, a zelenom je linijom prikazana prilagodba modela stvarnim vrijednostima. Na grafu je prikazano predviđanje za prvih 100 točaka zbog lakše čitljivosti grafa. U ovom se slučaju za bazu podataka uzeo cijeli skup podataka odnosno svi parametri koji su prikupljeni tijekom testiranja – spol, godine, orijentacija ruke, iskustvo, krivulja, *level*, boja, idealna, glazba, posao, doba dana, naočale i daltonizam. Koeficijent K unakrsne validacije definiran je na 7 preklopa, a broj susjeda je definiran isto na 7.



Slika 5.1. Model predviđanja kNN algoritma

Na slici 5.2. prikazan je linijski graf stvarnih vrijednosti – označeno plavom bojom i linijski prikaz predviđenih vrijednosti – označeno zelenom bojom. Sa grafa se točno može vidjeti kako se model dobiven kNN algoritmom prilagođava stvarnim vrijednostima. Prikaz je napravljen na prvih 25 vrijednosti te se točno može vidjeti odstupanje predviđene vrijednosti od stvarne.



Slika 5.2. Linijski graf predviđanja kNN algoritma

U tablici 5.1. dan je prikaz metrika vrednovanja koje su izračunate kao srednja vrijednost od ukupnog broja iteracija za svaku metriku.

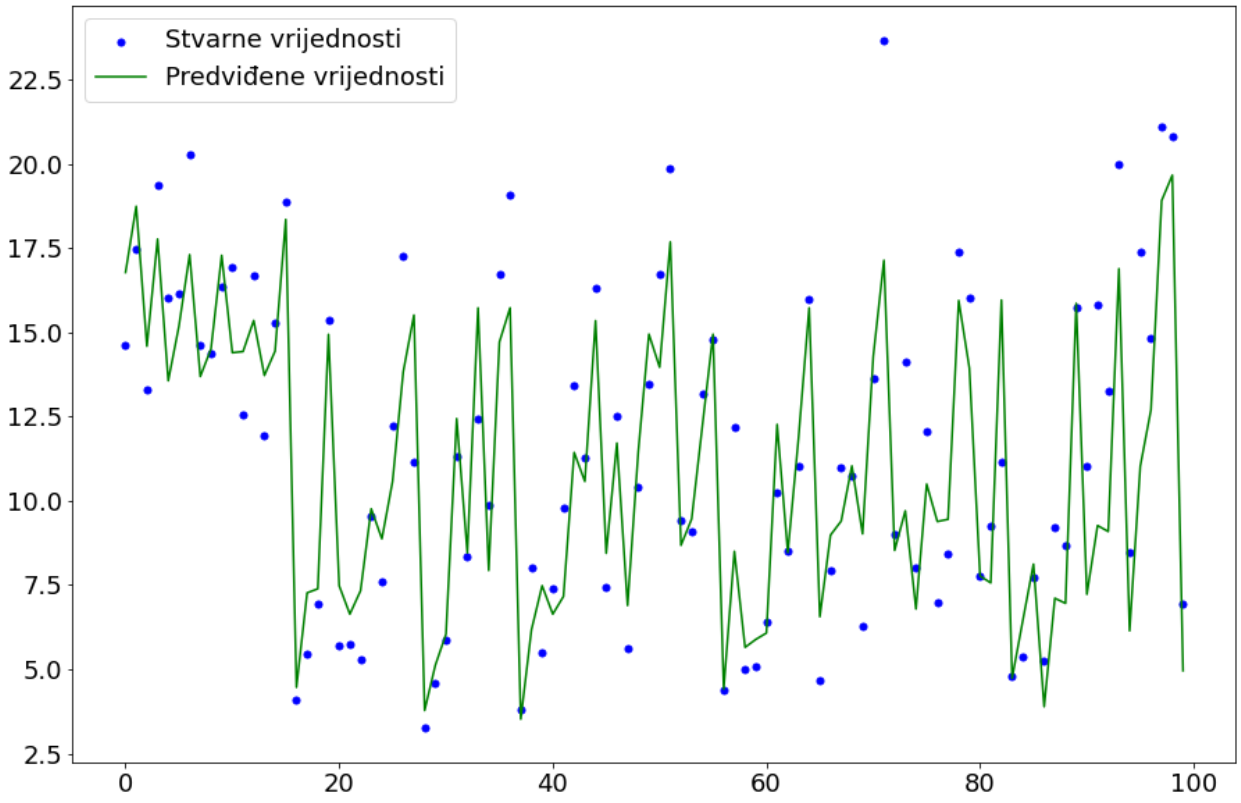
Tablica 5.1. Metrike vrednovanja za kNN

R2	0.798
MAE	1.869
MSE	7.709
RMSE	2.775

Iz tablice uočavamo da je točnost modela dobivenog metodom najbližih susjeda rezultirala s 79.8% točnosti, preostale metrike poprimaju dovoljno niske vrijednosti da se može zaključiti kako su pogreške u predviđanjima niske.

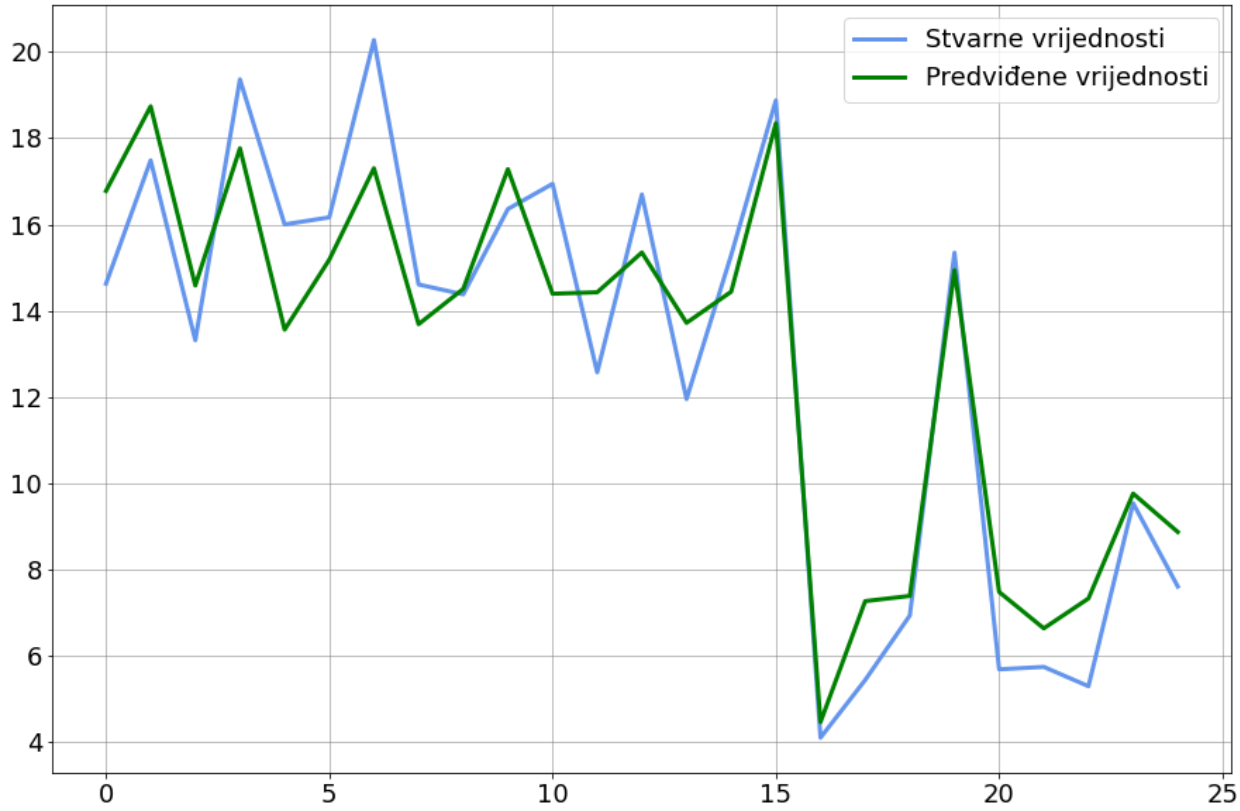
Sljedeće je testiran algoritam slučajne šume te je prilikom izrade modela i testiranja algoritma napravljena faza testiranja parametara samog algoritma kako bi se utvrdilo pod kojim se uvjetima dobiju najbolja predviđanja i rezultati. S obzirom da algoritam slučajne šume kreira više stabala odluke prilikom razvoja modela, zaseban algoritam stabla odluke je izbačen iz testiranja. Algoritam slučajne šume je stabilniji i daje rezultate veće točnosti, dok algoritam stabla odluke podliježe lošijim rezultatima ili prenaučenosti modela. Broj stabala odluke definiran je na 50, a maksimalna dubina stabla na 100. Koeficijent unakrsne validacije postavljen je na 5. Kao i kod kNN metode, za bazu podataka koristili su se svi parametri – spol, godine, orijentacija ruke, iskustvo, krivulja, *level*, boja, idealna, glazba, posao, doba dana, naočale i daltonizam.

U nastavku su slikom 5.3. dani rezultati dobiveni algoritmom slučajne šume. Kao i u prethodnom slučaju, na grafu su plavim točkama označene stvarne vrijednosti originalnih podataka, a zelena linija označava prilagodbu modela stvarnim vrijednostima, odnosno predviđanja stvarnih vrijednosti.



Slika 5.3. Model predviđanja algoritmom slučajne šume

U nastavku je dana slika 5.4. koja prikazuje linijski graf za oba slučaja – stvarne i predviđene vrijednosti. Možemo uočiti da je model dao predviđanja koja su blizu stvarnih vrijednosti. Čak i kada postoje neka veća odstupanja, vidljivo je da model daje predviđanja što je bliže moguće toj vrijednosti. Graf je definiran da prikazuje prvih 25 vrijednosti kako bi točke bile što jasnije vidljive.



Slika 5.4. Linijski graf predviđanja slučajne šume

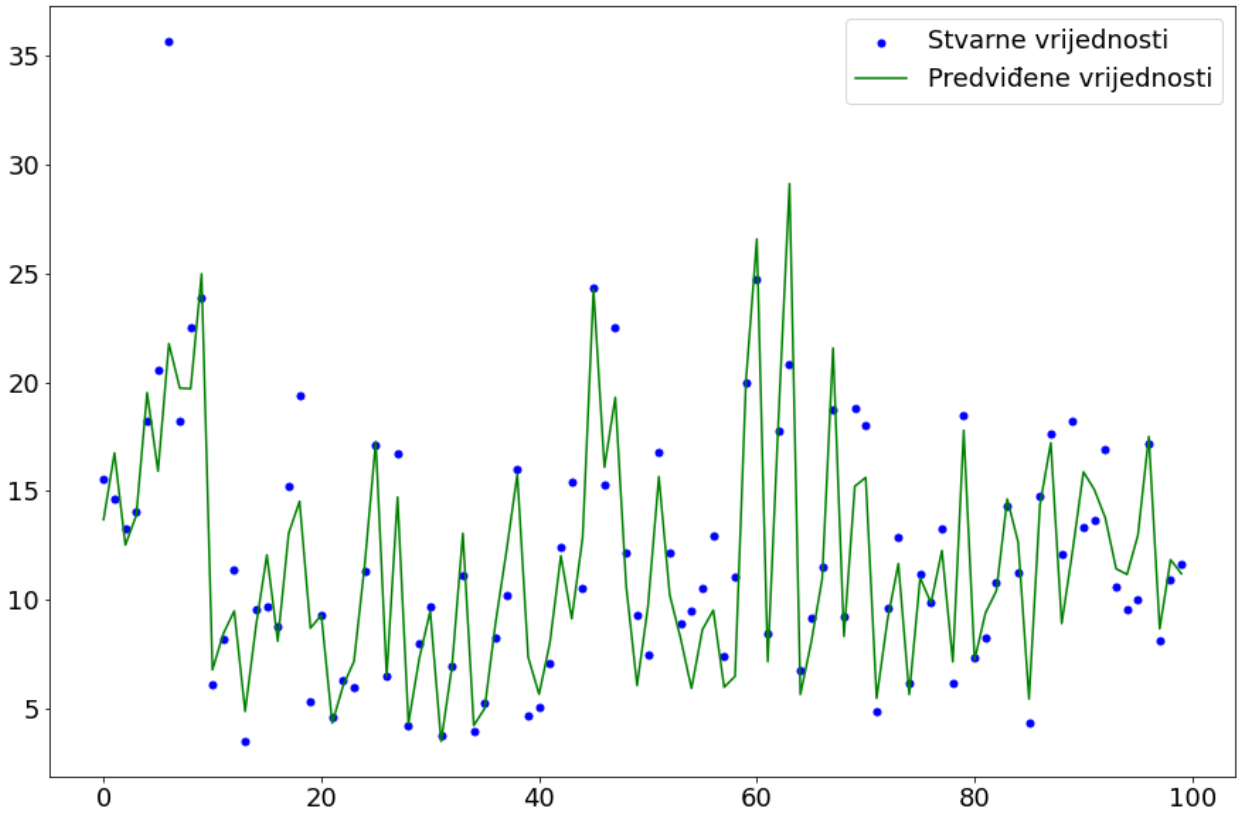
Rezultati metrika vrednovanja za algoritam slučajne šume dani su u tablici 5.2. Kao i u prethodnom slučaju prikazane metrike su srednje vrijednosti svake metrike ovisno o definiranom broju iteracija unakrsne validacije.

Tablica 5.2. Metrike vrednovanja za slučajnu šumu

R2	0.771
MAE	1.851
MSE	6.194
RMSE	2.489

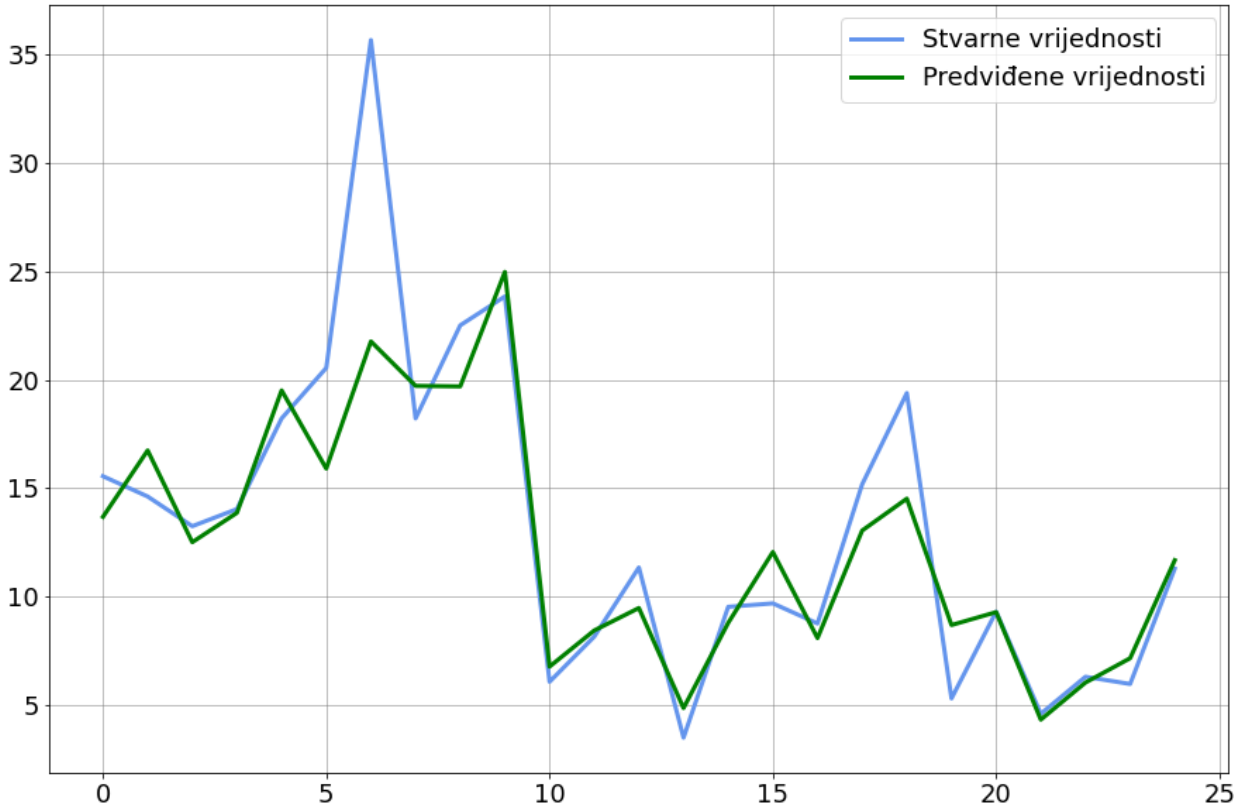
Tablica nam daje rezultate za algoritam slučajne šume od 77.1% točnosti, što se i dalje klasificira dobrim rezultatom. Preostalim metrikama konvergira se prema nižim vrijednostima što ukazuje na male pogreške modela prilikom predviđanja.

Polinomna regresija odabrana je kao algoritam nakon što linearna regresija i regularizacija iste nisu dale zadovoljavajuće rezultate. U nastavku su rada dani grafovi koji prikazuju kako se polinomna regresija pokazala kao rješenje u slučaju predviđanja vremena potrebnog za rješavanje tunelskih zadataka. Korištena je baza podataka koja uključuje sve dostupne parametre te je na cijelom skupu korištena unakrsna validacija. Koeficijent je definiran na K = 7 jer su njime dobiveni najbolji rezultati. Za stupanj polinomne regresije napravljeno je podešavanje parametara te je kao rezultat dobiveno najbolje rješenje sa stupnjem 4, odnosno kvartnom regresijom. Na slici 5.5 prikazan je model predviđanja dobiven polinomnom regresijom četvrtog stupnja.



Slika 5.5. Model predviđanja polinomne regresije

Plavim su točkama označene stvarne vrijednosti baze podataka na kojima je napravljeno testiranje, a zelenom su bojom označene predviđene vrijednosti polinomnom regresijom. Radi boljeg razumijevanja dan je graf prikazan na slici 5.6. na kojima su linijski označene stvarne i previđene vrijednosti pa se lakše može vidjeti prilagodba modela dobivenog polinomnom regresijom. Također je broj točaka smanjen na 25 kako bi se jasnije vidjele stvarne i predviđene vrijednosti.



Slika 5.6. Linijski graf predviđanja polinomne regresije

Plavom su linijom označene stvarne vrijednosti, a zelenom predviđene vrijednosti dobivene polinomnom regresijom. Uočavamo da su predviđene vrijednosti veoma blizu stvarnim vrijednostima.

Za polinomnu regresiju četvrtog stupnja metrike vrednovanja dane su u tablici 5.6. Kao i za slučaj kNN metode i algoritma slučajne šume, metrike prikazane u tablici su srednja vrijednost svake metrike po iteraciji unakrsne validacije.

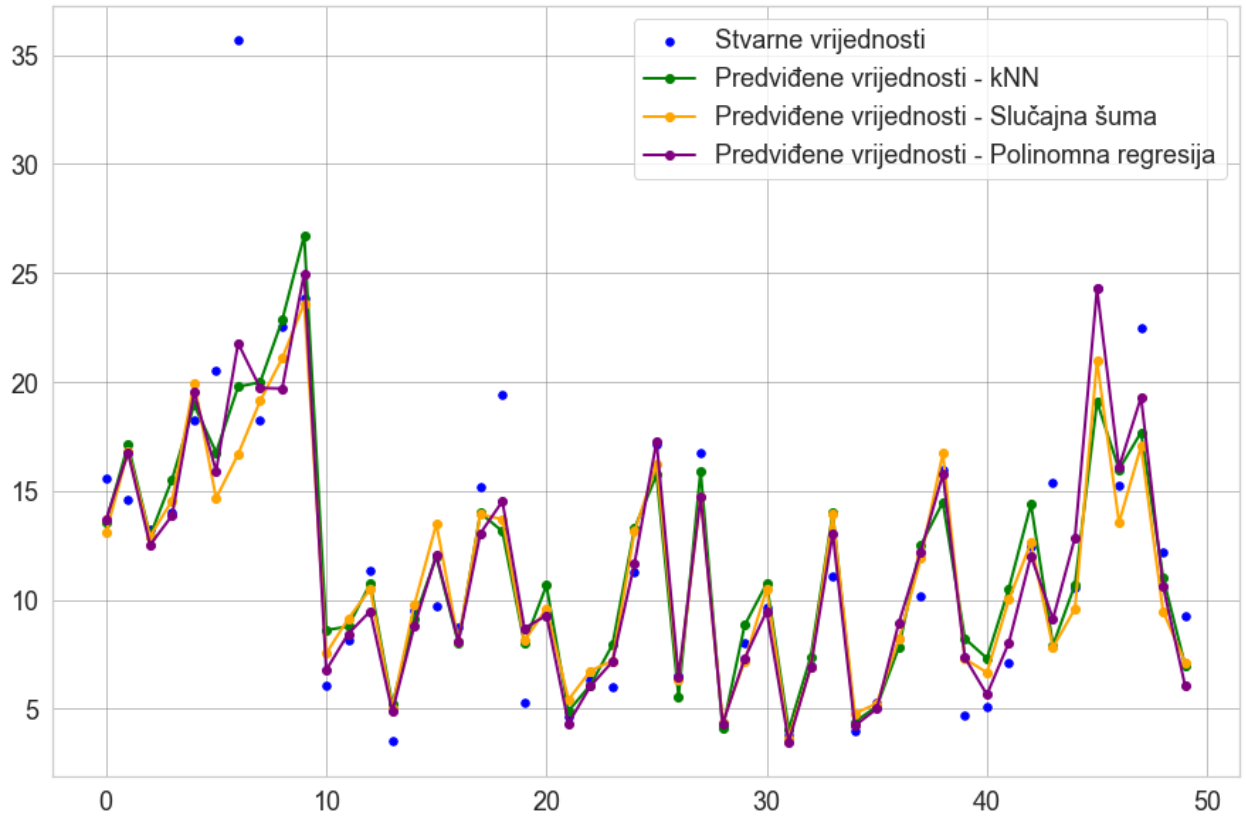
Tablica 5.3. Metrike vrednovanja za polinomnu regresiju

R2	0.824
MAE	1.802
MSE	6.714
RMSE	2.588

Iz tablice vidimo kako je polinomna regresija četvrtog stupnja dala točnost modela predviđanja od 82.4% točnosti. MAE, MSE i RMSE poprimaju niske vrijednosti pa zaključujemo da model rezultira s manjim pogreškama.

5.2. Diskusija

Na slici 6.1. dan je prikaz rezultata sva tri testirana algoritma. Kako bi se na jednom grafu prikazali svi dobiveni rezultati, definiran je isti parametar K = 7 unakrsne validacije za svaki algoritam. Plavim su točkama označene stvarne vrijednosti, zelena je linija za rezultate dobivene kNN algoritmom, narančastom je linijom prikazan model dobiven slučajnom šumom, a ljubičastom su linijom prikazani rezultati dobiveni polinomnom regresijom. Graf je postavljen da prikazuje prvih pedeset vrijednosti radi lakše čitljivosti. Vidljivo je da se sva tri algoritma podjednako dobro prilagođavaju stvarnim vrijednosti te da se u određenim točkama gotovo podudaraju sa stvarnim vrijednostima.



Slika 5.7. Usporedba rezultata

U nastavku su tablicom 6.2. prikazani rezultati metrika vrednovanja. Korištene su sve prethodno opisane metrike – MAE, MSE, RMSE i R2. Metrike koje su prikazane u tablici označavaju srednju vrijednost rezultata metrika za broj provedenih iteracija.

Tablica 5.1. Rezultati metrika vrednovanja za korištene algoritme

	kNN	SLUČAJNA ŠUMA	POLINOMNA REGRESIJA
R2	0.798	0.783	0.824
MAE	1.887	2.018	1.802
MSE	7.691	8.251	6.714
RMSE	2.768	2.870	2.588

Iz tablice uočavamo da kNN i slučajna šuma rezultiraju podjednakom točnosti. Štoviše, razlika u točnosti je približno 0.015%, dok je algoritam polinomne regresije dao najveću točnost od 82.4%. Od ostalih metrika, kNN je također rezultirao boljim rezultatima u odnosu na slučajnu šumu, ali je polinomna regresija za svaku metriku ipak dala najbolji produkt. Za MAE za kNN algoritam dobivamo vrijednost od 1.887, za slučajnu šumu ta je vrijednost 2.018, a za polinomnu je regresiju vrijednost 1.802. S obzirom da MAE označava bolje rezultate što je njegova vrijednost bliža 0, zaključujemo da je u ovoj usporedbi bolji rezultat za polinomnu regresiju u odnosu na kNN i slučajnu šumu. Za MSE niže vrijednosti ukazuju točnijim predviđanjima pa s obzirom da se za kNN poprima vrijednost od 7.691, slučajna šuma rezultira s 8.251, a polinomna regresija ima rezultat od 6.714 uočavamo da su rezultati opet bolji za algoritam polinomne regresije. Zadnja je metrika vrednovanja RMSE koja za kNN poprima vrijednost 2.768, za slučajnu šumu 2.870, a za polinomnu regresiju 2.588. Kao i za MAE, ova metrika definirana bolji model ako je njena vrijednost bliža nuli. Pravilo možemo primijeniti na rezultate ovih modela gdje utvrđujemo da je polinomna regresija opet dala bolje rezultate u odnosu na kNN i algoritam slučajne šume.

6. ZAKLJUČAK

U ovom je radu napravljena aplikacija kojom su testirane performanse korisnika prilikom rješavanja zadataka. Zadaci su definirani kao tunel kroz koji je potrebno proći odabranim pokaznim uređajem što je točnije moguće u što kraćem vremenu. Kroz tunel je potrebno proći po sredini prateći njegove zavoje. Jedan testni slučaj ima 15 zadataka koji su kombinacija tri težine tunela i pet različitih krivulja. Definirano je 8 testnih slučajeva koji su rezultat triju kombinacije uvjeta boje, idealne linije i glazbe. U konačnici je dobiveno 120 zadataka koje je potrebno riješiti. Prikupljeni su rezultati od 50 korisnika te je sve skupa rezultiralo sa 6000 rezultata. Podaci su obrađeni te je napravljena baza podataka koja je dalje korištena prilikom razvoja modela predviđanja. Obrada podataka uključivala je izračun utjecajnosti parametara kako bismo što točnije znali koja je ulazna varijabla najutjecajnija, a čija je korisnost gotovo zanemariva. Dobivenim je rezultatima utvrđeno da su najutjecajniji parametri krivulja, godine i level, a najmanji utjecaj ima orijentacija ruke, daltonizam i posao. Također, napravljena je i korelacija svih varijabli kako bi se mogla definirati veza između korištenih parametara. Utvrđeno je da vrijeme izvršavanja zadatka najjaču korelaciju ima s krivuljom, a malo slabije korelira s *levelom,* uvjetom idealne linije i dobom dana u kojem je provedeno testiranje. Što znači da te varijable značajnije utječu na vrijeme rješavanja zadataka, dok preostale imaju manji utjecaj na vrijeme. Na skupu podataka detektirane su ekstremne vrijednosti, ali je daljnjim istraživanjem ustanovljeno da su otkrivene ekstremne vrijednosti rezultat prirodne varijacije korisnika, a ne pogreške te je odlučeno da se ekstremi ne uklanjaju iz baze podataka. Nadalje, obrada podataka uključivala je i transformaciju podataka te prikaz distribucije podataka kako bi podaci bili što razumljiviji.

Testirano devet algoritama od kojih su prikazani rezultati troje od njih koji su dali najbolje rezultate – metoda najbližih susjeda, slučajna šuma i polinomna regresija. Algoritmi su korišteni u kombinaciji s unakrsnom validacijom koja je napravljena na cijeloj bazi podataka. Broj iteracija unakrsne validacije testiran je i definiran na pet ili sedam iteracija ovisno koji je uvjet dao bolje rezultate i za koji algoritam. Prilikom svake iteracije izračunate su metrike vrednovanja te je njihov prosjek uzet kao konačan rezultat. Svaki je model predviđanja opisan vizualno i numerički kako bi se dobio što jasniji prikaz konačnih rezultata. U svakom slučaju, model se vrlo dobro prilagodio stvarnim vrijednostima. Kao najbolji model pokazao se onaj dobiven algoritmom polinomne regresije, koji je rezultirao s 82.4% . Nešto lošiji rezultati dobiveni su kNN algoritmom, 79.8%, a

najlošije se pokazao algoritam slučajne šume čiji je model rezultirao s 77.1% točnosti. Također je napravljena usporedba svih triju algoritama prema istim parametrima kako bi se na jednom grafu s istim vrijednostima mogla prikazati razlika između dobivenih modela. Polinomna je regresija rezultirala najboljim modelom predviđanja, prema svim metrikama vrednovanja, potom je metoda najbližih susjeda, a model dobiven slučajnom šumom rezultirao je najslabijim učinkom. Visoka točnost modela smatra se poželjnim ishodom jer označava model koji dobro radi. Vrijednosti preostalih metrika poprimaju niske vrijednosti, što znači da su pogreške u predviđanjima niske. Dobiveni modeli jasno daju predviđanja performansi interakcije pokaznim uređajem. Općenito se dobiveni rezultati mogu koristiti dalje za analizu i kreiranje izvješća, za usporedbu s drugim interaktivnim uređajima (npr. *touchpad)* i provođenje statističkih testova kako bi se utvrdilo ako su razlike u vremenu između interaktivnih uređaja statistički značajne. Svaki se model predviđanja lako može primijeniti na neki drugi problem i drugi set podataka. Potrebno je samo definirati druge ulazne varijable, kao i kontinuiranu ciljanu varijablu koja će se predviđati. Model može biti dobar temelj, odnosno početak za upotrebu u nekim drugim slučajevima i problemima regresije. Ovdje je dan prikaz rješavanja zadataka, koji više nalikuju računalnoj igri, međutim, model predviđanja lako se može prenamijeniti na neki drugi problem računalnih zadataka. Dobivenim je predviđanjima moguće prioritizirati zadatke s obzirom na vrijeme koje im je potrebno da se odrade i moguće je bolje organizirati vlastito vrijeme te bolje planirati rješavanje zadataka.

7. LITERATURA

[1] Buns, E. "What Is Machine Machine Learning and Why Is it Important?", s Interneta, https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML, 31. Kolovoza 2023.

[2] IBM, "What is Machine Learning?", s Interneta, https://www.ibm.com/topics/machine-learning, 31. Kolovoza 2023.

[3] W. A. Labs, "Advantages and Disadvantages of Machine Learning in 2023", s Interneta, https://www.westagilelabs.com/blog/pros-and-cons-of-implementing-machine-learning-in-your-projects/, 31. Kolovoza 2023.

[4] A. Anwar, "A Beginner's Guide to Regression Analysis in Machine Learning", s Interneta, https://towardsdatascience.com/a-beginners-guide-to-regression-analysis-in-machine-learning-8a828b491bbf, 14. Kolovoza 2023.

[5] V. Kurama, "Regression in Machine Learning: What It Is and Examples od Different Models", s Interneta, https://builtin.com/data-science/regression-machine-learning, 14. Kolovoza 2023.

[6] S. Singh, "Understanding the Bias-Variance Tradeoff", s Interneta, https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229, 14. Kolovoza 2023.

[7] dewangNautiyal, "ML | Underfitting and Overfitting", s Interneta, https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/, 15. Kolovoza 2023.

[8] O. Harrison, "Machine Learning Basics with the K-Nearest Neighbors Algorithm", s Interneta, https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761, 15. Kolovoza 2023.

[9] S. E. R, "Understand Random Forest Algorithms With Examples", s Interneta, https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/, 22. Kolovoza 2023.

[10] Chaya, "Random Forest Regression", s Interneta, https://levelup.gitconnected.com/random-forest-regression-209c0f354c84, 22. Kolovoza 2023.

[11] Simplilearn, "Random Forest Algorithm", s Interneta, https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm, 22. Kolovoza 2023.

[12] A. Dutta, "Random Forest Regression in Python", s Interneta, https://www.geeksforgeeks.org/random-forest-regression-in-python/, 22. Kolovoza 2023

[13] Simplilearn, "What Is Python Polynomial Regression In Machine Learning?", s Interneta, https://www.simplilearn.com/what-is-python-polynomial-regression-in-machine-learning-article, 22. Kolovoza 2023.

[14] I. A. Ogunbiyi, "Top Evaluation Metrics for Regression Problems in Machine Learning", s Interneta, fhttps://www.freecodecamp.org/news/evaluation-metrics-for-regression-problems-machine-learn, 2. Srpanja 2023.

[15] A. S. 44, "Cross Validation in Machine Learning", s Interneta, https://www.geeksforgeeks.org/cross-validation-machine-learning/, 30. Kolovoza 2023.

[16] P. Gupta, "Cross-Validation in Machine Learning", s Interneta, https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f, 30. Kolovoz 2023.

[17] A. Ihalapathirana, "Bresenham's Line Drawing Algorithm", s Interneta, https://medium.com/geekculture/bresenhams-line-drawing-algorithm-2e0e953901b3, 10. Kolovoza 2023.

[18] G. Lawton, "Data Preprocessing: Definition, Key Steps and Concepts", s Interneta, https://www.techtarget.com/searchdatamanagement/definition/data-preprocessing, 8. Kolovoza 2023.

[19] Yennhi95zz, "The Importance of Outlier Detection in Machine Learning: Methods and Implementation in Python", s Interneta, https://medium.com/@yennhi95zz/the-importance-of-outlier-detection-in-machine-learning-methods-and-implementation-in-python-125e3d5ada7d, 11. Kolovoza 2023.

[20] S. Mcleod, "Z-Score: Definition, Formula, Calculation & Interpretation", s Interneta, https://www.simplypsychology.org/z-score.html#:~:text=The%20formula%20for%20calculating%20a,Figure%202, 11. Kolovoza 2023.

[21] G. Malato, "Outlier identification using Interquartile Range", s Interneta, https://towardsdatascience.com/outlier-identification-using-interquartile-range-74f5de12932a, 30. Kolovoza 2023.

[22] towsifahmedlabib, "What is Box plot and the condition of outliers?", s Interneta, https://www.geeksforgeeks.org/what-is-box-plot-and-the-condition-of-outliers/#article-meta-div, 11. Kolovoza 2023.

[23] T. Hessing, "Data Distributions", s Interneta, https://sixsigmastudyguide.com/data-distributions/, 9. Kolovoza 2023.

[24] T. Shin, "Understanding Feature Importance and How to Implement it in Python", s Interneta, https://towardsdatascience.com/understanding-feature-importance-and-how-to-implement-it-in-python-ff0287b20285#:~:text=Feature%20Importance%20refers%20to%20techniques,to%20predict%20a%20certain%20variable., 10. Kolovoza 2023.

[25] ProjectPro, "How to drop out highly correlated features in Python?" , s Interneta, https://www.projectpro.io/recipes/drop-out-highly-correlated-features-in-python, 7. Kolovoza 2023.

8. SAŽETAK

U ovome je radu napravljena desktop aplikacija za automatizirano testiranje korisnika prilikom korištenja pokaznog uređaja za rješavanje tunelskih zadataka. Prilikom testiranja mjerila se brzina rješavanja pojedinog zadatka. Zadatak se manifestira kao tunel kroz koji je potrebno proći odabranim interaktivnim uređajem. Konfiguracija tunela mijenja se u 15 različitih izvedba gdje se kombiniraju uvjeti težine tunela i različitih krivulja. Testiranje se sastojalo od 8 testnih slučajeva od kojih svaki sadrži 15 pojedinih zadataka. Za testne slučajeve postavljeni su uvjeti boje programa, idealne linije i glazbe. Također, prikupljene su dodatne osobne informacije za svakog korisnika. Napravljeno je testiranje na 50 korisnika te je napravljena baza podataka koja se dalje koristila za obradu i razvoj modela predviđanja. Za model predviđanja korišteni su algoritmi za rješavanje problema regresije te su njihovi rezultati dani u grafičkom i opisnom obliku. Korištena je unakrsna validacija kako bi se smanjio problem podtreniranosti i pretreniranosti modela. Za svaki su algoritam korištene metrike vrednovanja kako bi se definirala točnost modela. Algoritmi i njihovi rezultati potom su uspoređeni kako bi se točno prikazalo kojim je algoritmom dobiveno najbolje rješenje.

ABSTRACT

In this project, a desktop application was developed for automated user testing when using an interactive device for solving tunnel tasks. During testing, the speed of solving individual tasks was measured. The task is manifested as a tunnel that needs to be navigated through using an interactive device. The tunnel configuration varies in 15 different setups, combining tunnel difficulty conditions and different curves. Testing consisted of 8 test cases, each containing 15 individual tasks. Conditions for program color, ideal line, and music were set for the test cases. Additionally, personal information was collected for each user. Testing was conducted with 50 users, and a database was created, which was further used for data processing and model development. Regression algorithms were used for prediction modeling, and their results were presented in graphical and descriptive forms. Cross-validation was employed to mitigate the issues of underfitting and overfitting. Evaluation metrics were used for each algorithm to define the accuracy of the model. The algorithms and their results were then compared to accurately depict which algorithm produced the best solution.