

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU  
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH  
TEHNOLOGIJA U OSIJEKU

PREDDIPLOMSKI SVEUČILIŠNI STUDIJ



## **Predviđanje cijene mobitela na osnovu unesenih specifikacija**

Seminarski rad

Osijek, 2023.

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU  
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH  
TEHNOLOGIJA U OSIJEKU

PREDDIPLOMSKI SVEUČILIŠNI STUDIJ



**Predviđanje cijene mobitela na osnovu unesenih specifikacija**

Seminarski rad

Predmet:      Računarstvo usluga i analiza podataka

Tim:

1. Marko Ćosić – voditelj tima
2. Matija Barić
3. Benjamin Varvodić

Osijek, 2023.

# Tablica sadržaja

1. Uvod .....	1
2. Korišteni podaci .....	2
2.1. Kaggle, dataset .....	2
2.3. Gotovi model.....	5
3. Tehnologije.....	11
3.1. HTML.....	11
3.2. CSS .....	12
3.3. Javascript.....	13
3.4. TypeScript .....	15
3.5. Node.js.....	15
3.5.1. Node Package Manager (NPM) .....	16
3.6. Angular.....	17
4. Izrada web aplikacije .....	19
4.1 Izrada novog projekta.....	19
4.2 Angular elementi unutar projekta.....	20
4.2.1 Angular komponente .....	20
4.2.2 Angular usmjeravanje .....	20
4.3 Konačan izgled web aplikacije .....	21
4.3.1 Testiranje rada web aplikacije.....	24
5. Zaključak .....	29
Literatura .....	30

## 1. Uvod

Danas je sve veća prisutnost mobitela na tržištu. Usporedimo li cijene i specifikacije mobitela danas i prije nekoliko godina, može se zaključiti kako su specifikacije mobitela iz godine u godinu sve bolje. No, to za sobom povlači neke nedostatke, kao što su povećana cijena proizvodnje, a samim time i povećana cijena konačnog proizvoda. Danas su na tržištu mobitela brojne tvrtke koje nude puno mobitela u različitim cjenovnim razredima. Upravo se u opisanome nalazi i cilj ovog seminarskog rada, a to je pomoći korisniku da na osnovu unesenih specifikacija mobitela i korisnikove procjene cijene dobije povratnu informaciju o tome koji cjenovni razred mobitela u pitanju. U ovome je seminarskom radu izrađen model pomoću već definiranog skupa podataka (engl. *dataset*), istreniran model te spojen s odgovarajućom web aplikacijom koja je izrađena korištenjem Angular.js okvira (engl. *framework*).

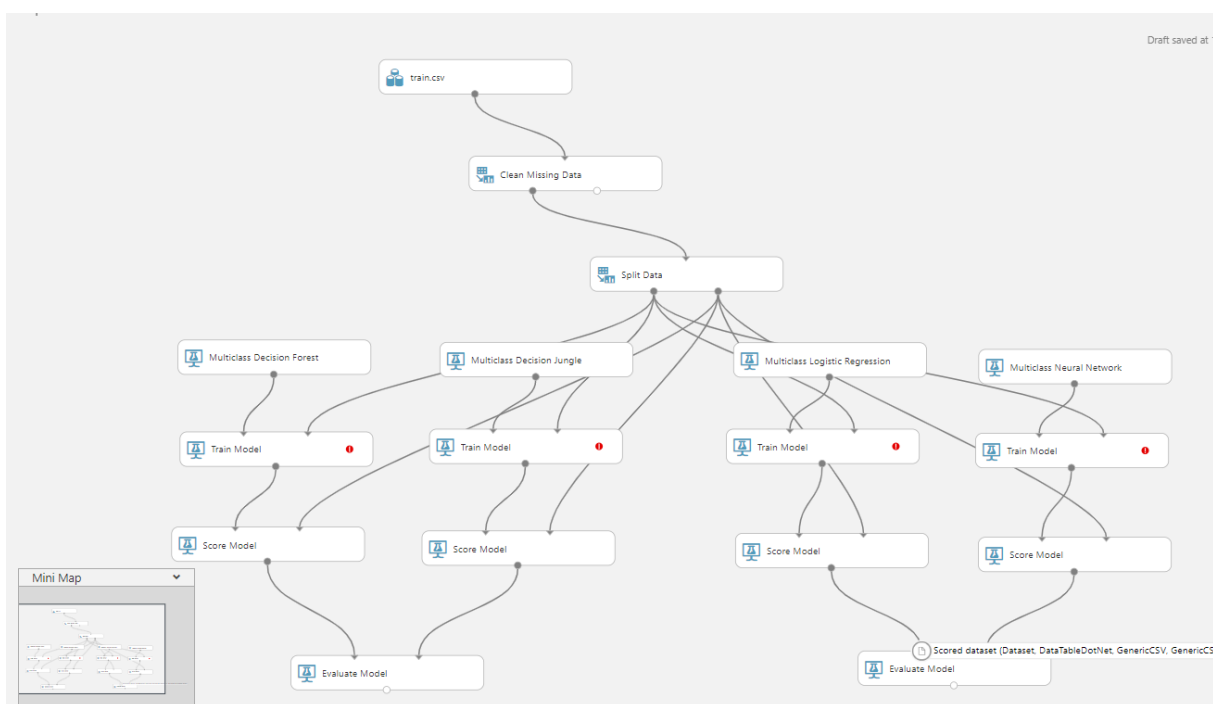
## 2. Korišteni podaci

### 2.1. Kaggle, dataset

Za izradu ovog projekta, korišten je dataset naziva „**Mobile Price Classification**“, autora Abhishek Sharma. Sastoji se od slijedećih podataka: ID, kapacitet baterije u mAh, bluetooth, brzina mikroprocesora, dual-sim podrška, megapiksela prednje kamere, 4G podrška, pohrana u Gb, debljina mobitela, težina mobitela, megapiksela stražnje kamere, rezolucija zaslona u 2 stavke (visina piksela x širina piksela), količina RAM-a u Mb, dimenzije zaslona u 2 stavke (visina x širina), talk time baterije, 3G podrška, touchscreen podrška, Wi-Fi podrška.

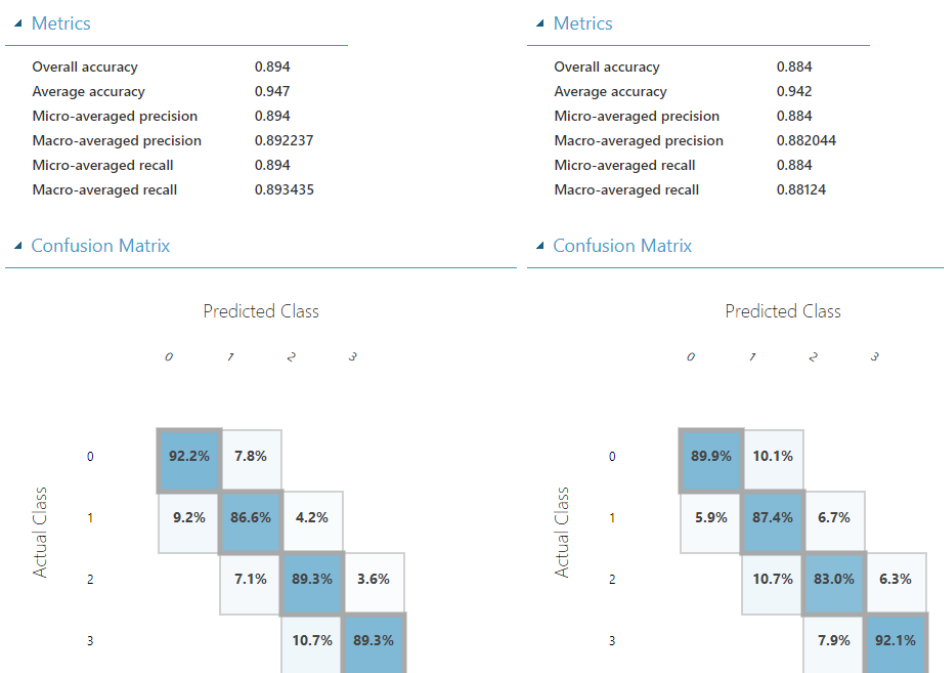
### 2.2. Treniranje i priprema modela

Kako bi napravili kvalitetan model, potrebno je testirati nekoliko metoda, odnosno algoritama klasifikacije. Usporedbom rezultata više algoritama potrebno je odabrati algoritam koji ima najbolju preciznost pri klasifikaciji. Pri testiranju korišteni su slijedeći algoritmi: Multiclass Decision Forest, Multiclass Decision Jungle, Multiclass Decision Regression i Multiclass Neural Network. Na slijedećoj slici će biti prikazan način spajanja algoritama i train modela za navedene algoritme.

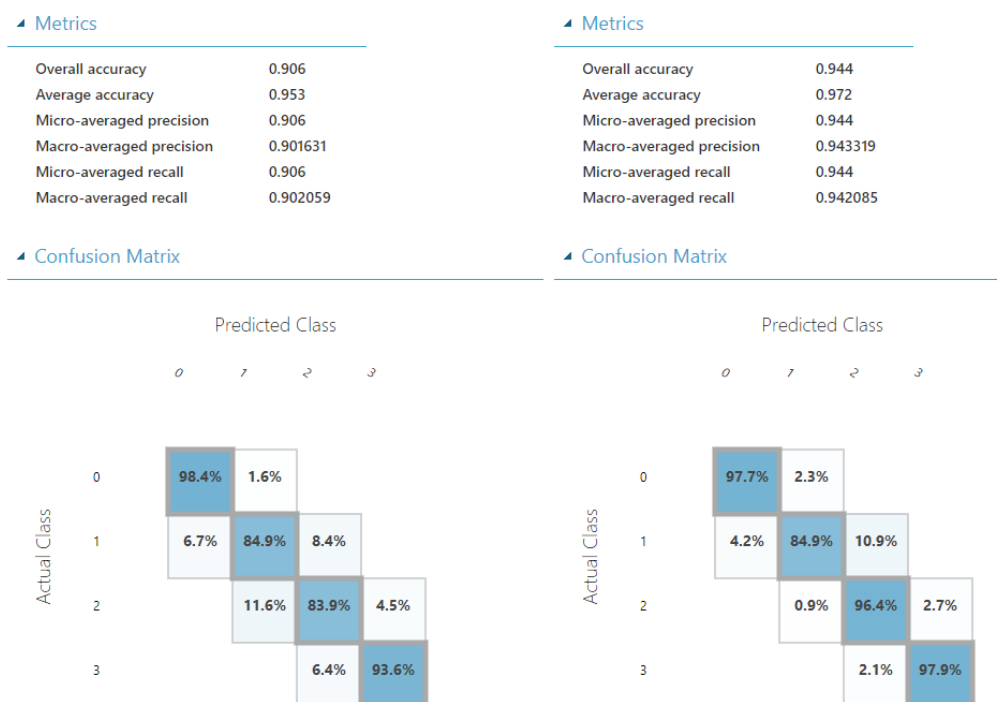


Slika 2.2.1. Algoritmi klasifikacije i model

Kako bi odabrali najbolji train model, potrebno je usporediti rezultate svakog od njih. Za početak su postavljene početne vrijednosti za svaki algoritam. Rezultati njih su prikazani na sljedećim slikama.



**Slika 2.2.2.** Algoritmi Multiclass Decision Forest i Multiclass Decision Jungle

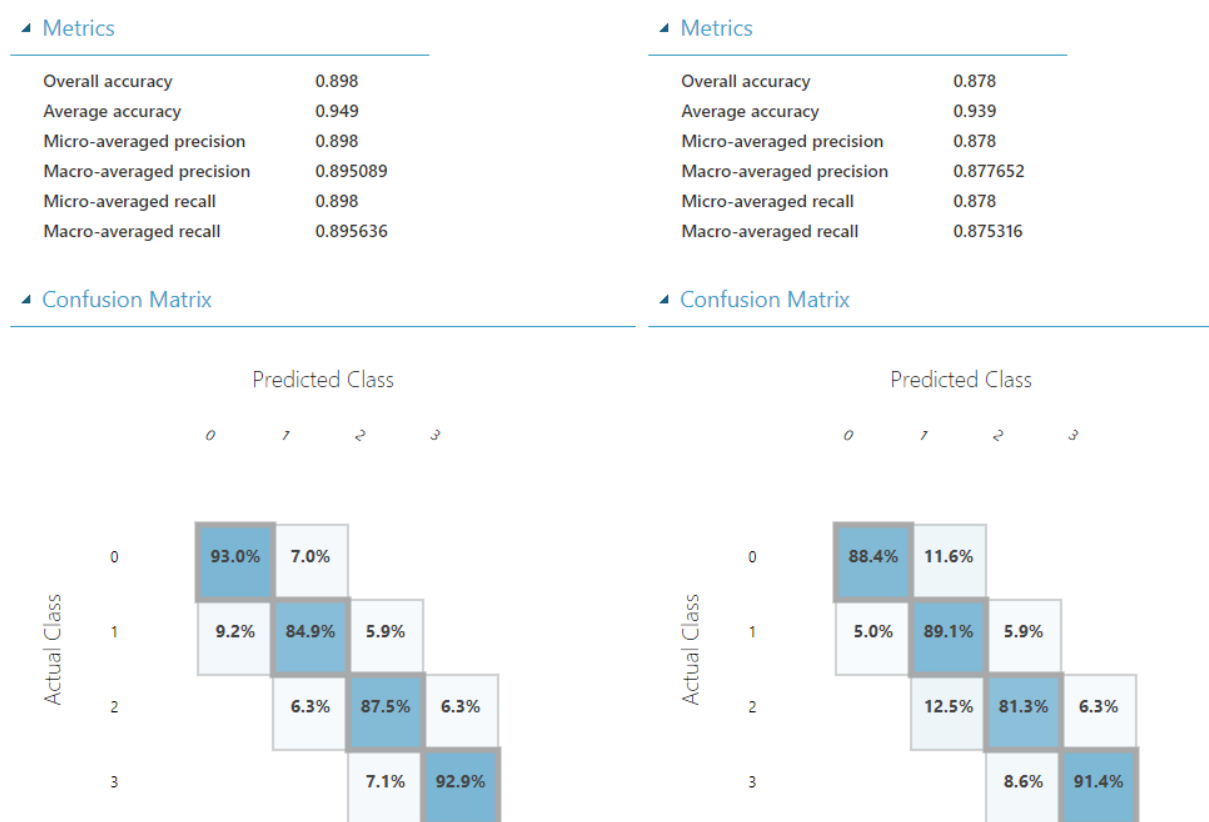


**Slika 2.2.3.** Algoritmi Multiclass Decision Regression i Multiclass Neural Network

Usporedbom train modela sa početnim vrijednostima, najbolji se pokazao algoritam Multiclass Neural Network. Neki od parametara za usporedbu su Overall accuracy, Average accuracy, Micro-averaged accuracy i drugi. Prema svim parametrima izabrani algoritam se pokazao najbolji kada su zadani početni parametri. Drugi po preciznosti bi bio algoritam Multiclass Decision Regression. Iz evaluate modela se može vidjeti kako je Average accuracy 0.972 te Overall accuracy 0.944.

Slijedeći korak je promjena parametara. Za algoritam Multiclass Decision Forest, Resampling method je promijenjena sa Bagging na Replicate te je Number of decision trees povećan sa 8 na 10. Za algoritam Multiclass Decision Jungle, Resampling method je promijenjena sa Bagging na Replicate te je Number of decision DAGs povećan sa 8 na 10. Za algoritam Multiclass Logistic Regression, Memory size for L-BFGS je promijenjen sa 20 na 24. Za algoritam Multiclass Neural Network, Number of hidden nodes je promijenjen sa 100 na 110, The learning rate je promijenjen sa 0.1 na 0.2 te je Number of learning iterations promijenjen sa 0.1 na 0.2. Svi parametri su promijenjeni nasumično.

Slijedi prikaz rezultata nakon promijene parametara.

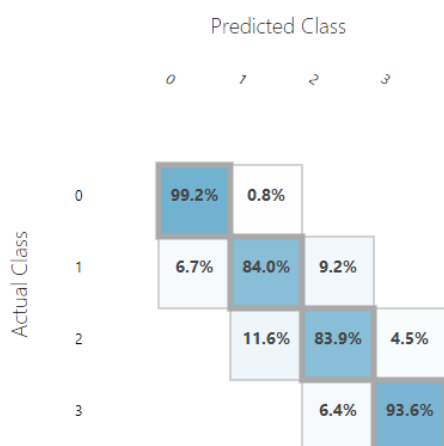


**Slika 2.2.4.** Algoritmi Multiclass Decision Forest i Multiclass Decision Jungle

#### Metrics

Overall accuracy	0.906
Average accuracy	0.953
Micro-averaged precision	0.906
Macro-averaged precision	0.901542
Micro-averaged recall	0.906
Macro-averaged recall	0.901896

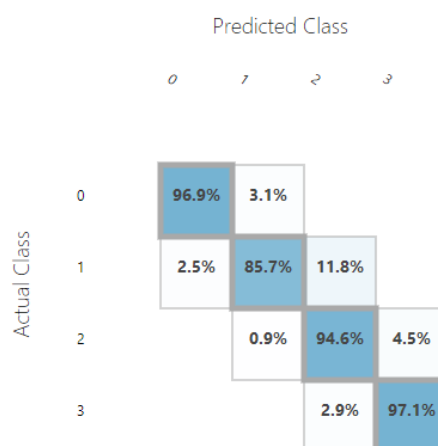
#### Confusion Matrix



#### Metrics

Overall accuracy	0.938
Average accuracy	0.969
Micro-averaged precision	0.938
Macro-averaged precision	0.937303
Micro-averaged recall	0.938
Macro-averaged recall	0.935998

#### Confusion Matrix



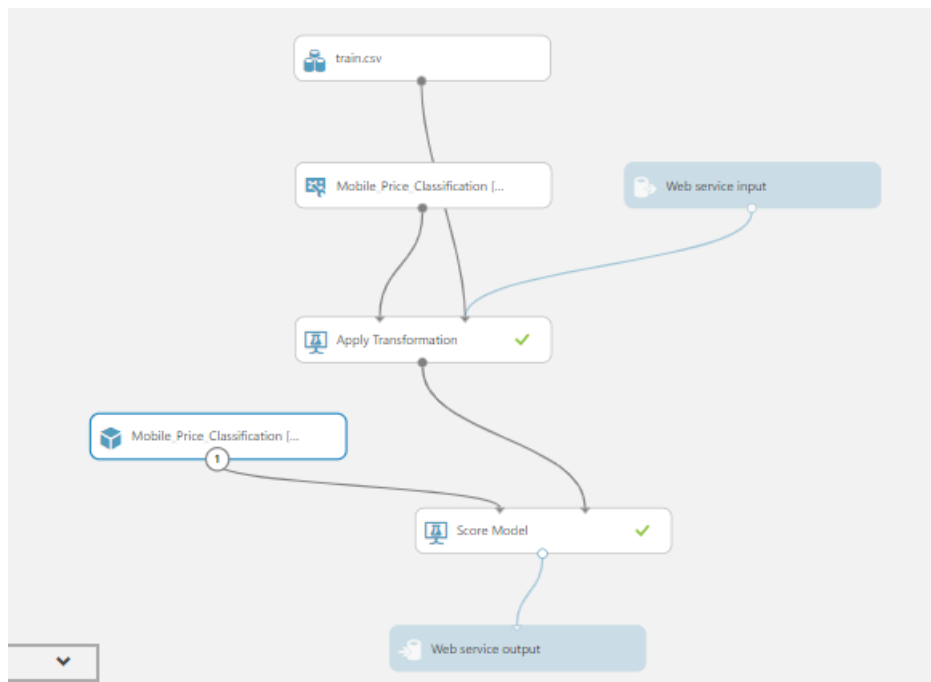
**Slika 2.2.5.** Algoritmi Multiclass Decision Regression i Multiclass Neural Network

Nakon promjene parametara, vidljivo je da su rezultati nekih algoritama bolji, kao na primjer Multiclass Decision Forest, a nekima su rezultati lošiji, na primjer Multiclass Neural Network. Bez obzira što su rezultati za Multiclass Neural Network algoritam lošiji u odnosu na početne parametre, ovaj algoritam i dalje ima najbolje rezultate u odnosu na druge algoritme. Rezultat toga je ponovni odabir ovoga algoritma za daljni nastavak izrade web api-a, ali sa početnim parametrima iz razloga što su bili bolji rezultati.

## 2.3. Gotovi model

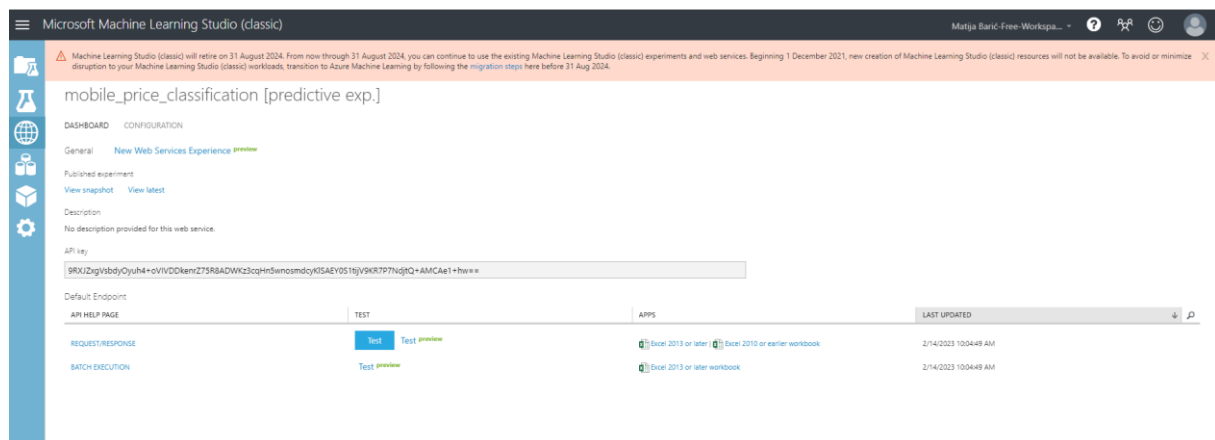
Kada je odabran algoritam, sljedeći korak je odabrati željeni train model te pokrenuti izradu predictive web service-a. Dobiveni rezultat će biti prikazan na sljedećoj slici.





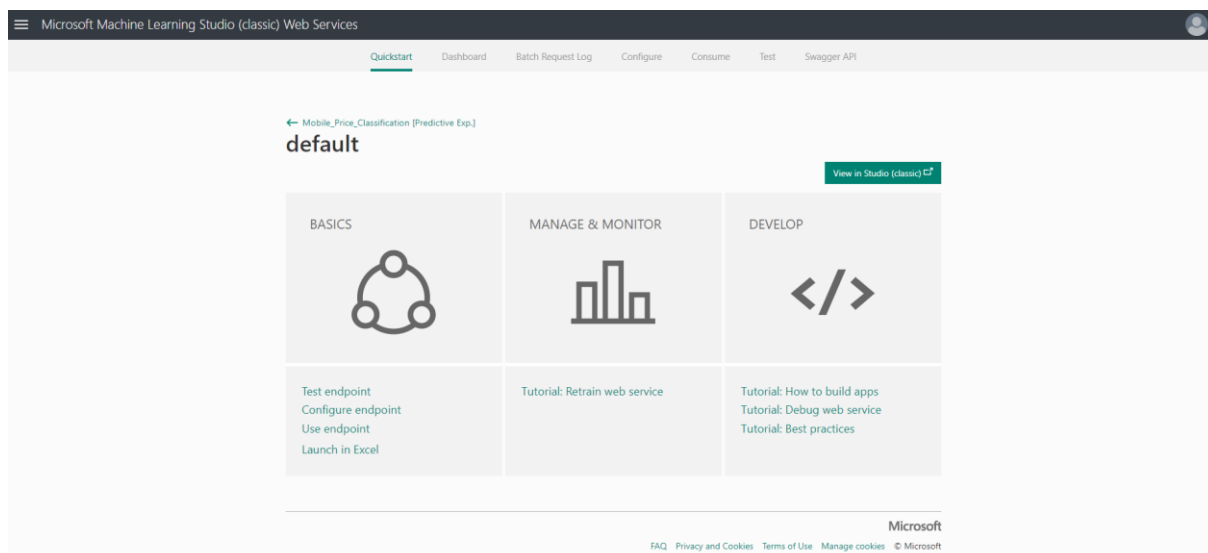
**Slika 2.3.1.** Izrada prediktivnog web servisa

Nakon izrade Predictive experiment-a potrebno je pokrenuti naredbu „deploy web service“. Rezultat naredbe je prikazan na sljedećoj slici.



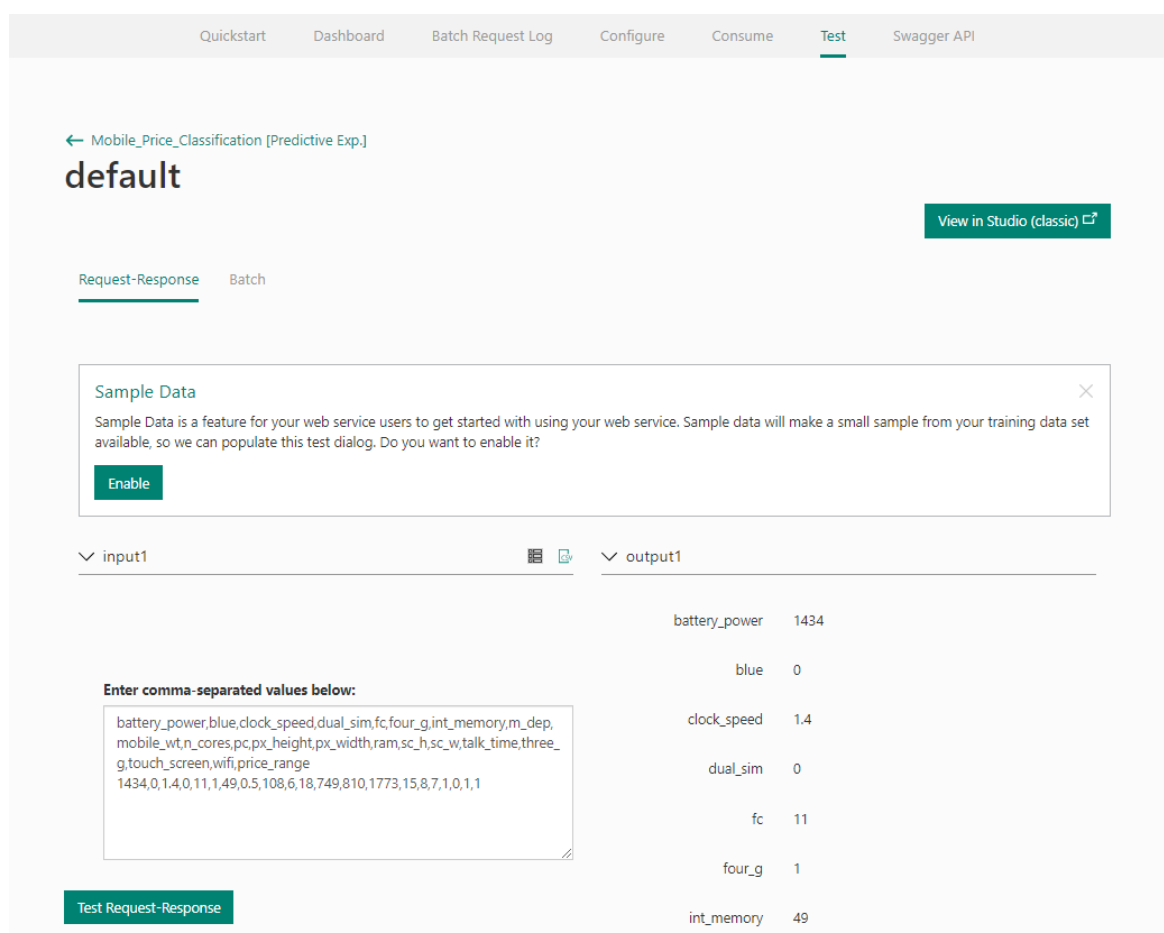
**Slika 2.3.2.** Izgled prozora nakon pokretanja opcije „Deploy Web Service“

U ovome prozoru koji je otvoren se može pronaći API key koji je potreban za daljnu izradu web aplikacije. Pritiskom na link „New Web Services Experience“ otvara se novi prozor u pregledniku te njegov izgled je prikazan na sljedećoj slici.



**Slika 2.3.3.** Izgled prozora New Web Services Experience

U ovome prozoru klikom na Test u navigacijskoj traci možemo testirati naš model. Primjer testiranja prikazan je na sljedećoj slici. Za testiranje će biti korišteno nekoliko različitih primjera s različitim očekivanim rezultatom.



**Slika 2.3.4.** Prvi primjer testiranja modela

Rezultat testiranja za prvi set parametara je prikazan na sljedećoj slici.

Scored Probabilities for Class "0"	1.66839765824989E-06
Scored Probabilities for Class "1"	0.992210149765015
Scored Probabilities for Class "2"	0.000732862448785454
Scored Probabilities for Class "3"	1.19969737269465E-11
Scored Labels	1

**Slika 2.3.5.** Rezultat testiranja modela

Sample Data

Sample Data is a feature for your web service users to get started with using your web service. Sample data will make a small sample from your training data set available, so we can populate this test dialog. Do you want to enable it?

Enable

input1

output1

Enter comma-separated values below:

battery\_power,blue,clock\_speed,dual\_sim,fc,four\_g,int\_memory,m\_dep, mobile\_wt,n\_cores,pc,px\_height,px\_width,ram,sc\_h,sc\_w,talk\_time,three\_g,touch\_screen,wifi,price\_range  
1718,0,2,4,0,1,0,47,1,156,2,3,1283,1374,3873,14,2,10,0,0,0,1

Test Request-Response

battery\_power1718

blue0

clock\_speed2.4

dual\_sim0

fc1

four\_g0

int\_memory47

**Slika 2.3.6.** Drugi primjer testiranja modela

Rezultat testiranja za drugi set parametara je prikazan na sljedećoj slici.

Scored Probabilities for Class "0"	7.14305229660266E-27
Scored Probabilities for Class "1"	2.12495422596339E-24
Scored Probabilities for Class "2"	2.94710787329677E-07
Scored Probabilities for Class "3"	1
Scored Labels	3

**Slika 2.3.7.** Rezultat testiranja modela

Sample Data

Sample Data is a feature for your web service users to get started with using your web service. Sample data will make a small sample from your training data set available, so we can populate this test dialog. Do you want to enable it?

Enable

input1

Enter comma-separated values below:

battery\_power,blue,clock\_speed,dual\_sim,fc,four\_g,int\_memory,m\_dep,
mobile\_wt,n\_cores,pc,px\_height,px\_width,ram,sc\_h,sc\_w,talk\_time,three\_
g,touch\_screen,wifi,price\_range
1520,0,0.5,0,1,0,25,0.5,171,3,20,52,1009,651,6,0,5,1,0,1,2

output1

battery\_power1520

blue0

clock\_speed0.5

dual\_sim0

fc1

four\_g0

int\_memory25

Test Request-Response

**Slika 2.3.8.** Treći primjer testiranja modela

Rezultat testiranja za treći set parametara je prikazan na sljedećoj slici.

Scored Probabilities for Class "0"	0.999997675418854
Scored Probabilities for Class "1"	2.67255290964385E-05
Scored Probabilities for Class "2"	1.32081765172518E-29
Scored Probabilities for Class "3"	9.26513920932176E-33
Scored Labels	0

**Slika 2.3.9.** Rezultat testiranja modela

U rezultatima testa najprije su ispisane vjerojatnosti za svaku klasu koja je moguća, te se na kraju ispisuje rezultat predviđanja.

### 3. Tehnologije

U ovome će poglavlju biti predstavljene tehnologije koje su korištene za izradu seminarskog djela projekta.

#### 3.1. HTML

Osnovni jezik koji se koristi za izradu web aplikacija je HTML (engl. *HyperText Markup Language*), predstavlja prezentacijski jezik za izradu web aplikacija i koristi se za postavljanje sadržaja i stvaranje svih hiperveza hipertekst dokumenta. Predstavlja jezik koji se lako upotrebljava, koji se lako uči, zbog čega je opće prihvaćen i popularan.

Prikaz hipertekst dokumenta omogućuje web preglednik. Temeljna zadaća koju je HTML jezik morao ostvariti je uputiti web preglednik za pravilno prikazivanje hipertekstnog dokumenta. Pri tome treba obratiti pažnju da navedeni dokument izgleda isto bez obzira o kojem web pregledniku, operacijskom sustavu i računalu je riječ.

HTML nije programski jezik, niti su ljudi koji ga koriste programeri. HTML ne može obaviti čak niti najjednostavnije matematičke operacije zbrajanja ili oduzimanja dvaju cijelih brojeva. Služi samo za opis hipertekst dokumenta. Aktualna inačica HTML-a koja se danas koristi je HTML5.

Svaki se HTML dokument sastoji od nekoliko osnovnih građevnih blokova – HTML oznaka (engl. *tag*). Isto tako, svaki element može imati i atribut pomoću kojih se definiraju svojstva elementa. Na početku HTML dokumenta preporuča se postavljanje `<!DOCTYPE>` elementa kojim se označava DTD (engl. *Document Type Declaration*), čime se definira točna inačica standarda korištena za izradu HTML dokumenta. Nakon `<!DOCTYPE>` elementa, pomoću `<html>` elementa označava se početak HTML dokumenta. Unutar `<html>` elementa nalaze se i `<head>` element te `<body>` element. `<head>` element služi za postavljanje zaglavlja HTML dokumenta u kojemu se najčešće specificiraju jezične značajke HTML dokumenta kao i naslov (engl. *title*) stranice. Korištenjem određenih HTML elemenata u zaglavlju, dodaju se i stilska obilježja stranice, bila ona dodana kao referenca na vanjsku CSS datoteku ili direktno ugrađena (engl. *embedded*). Vrlo često unutar zaglavlja još se definiraju i skripte kreirane u JavaScript jeziku. Unutar `<body>` elementa kreira se sadržaj HTML dokumenta, točnije stranice koju reprezentira.

Svaka HTML oznaka (koja u paru kreira HTML element) počinje znakom `<` (manje od), a završava znakom `>` (više od). Zatvarajuća HTML oznaka kreira se slično kao otvarajuća, ali se

nakon znaka < (manje od) dodaje kosa crta (engl. *slash*). Primjerice, <primjer\_oznake> </primjer\_oznake>.

U HTML-u moguće je koristiti i komentare. Otvaraju se s <!--, a zatvaraju s -->.

Naslovi u HTML dokumentu oblikuju se radi uočljivosti i kako bi bili jedinstveni za cijelu web aplikaciju. Postoji šest veličina naslova. Početna oznaka najvećeg naslova je <h1>, a završna </h1>. Najmanji naslov počinje s <h6>, a završava s </h6>.

Osnovno postavljanje teksta unutar HTML dokumenta moguće je ostvariti postavljanjem određenih oznaka na početku teksta koji se želi oblikovati, te postavljanjem završne oznake na kraju teksta. Primjerice:

<p></p> predstavlja oznaku za odlomak (engl. *paragraph*)

<b></b> predstavlja oznaku za podebljani tekst (engl. *bold*)

<u></u> predstavlja oznaku za podcrtani tekst (engl. *underlined*)

<i></i> predstavlja oznaku za nakrivljeni tekst (engl. *italic*).

Slike je moguće dodati oznakom <img>. Budući da navedena oznaka zahtijeva obilježje *src* (engl. *source*), tada se navedena oznaka promatra kao cjelovita i nema završnu oznaku, moguće je i dodati *alt* oznaku, koja može prikazati opis slike u slučaju njena neispravnog učitavanja.

Meta oznake dijelovi su HTML-a u stanici koje koriste tražilice da bi zapisale informacije o stranici. Ove oznake sadrže ključne riječi, naziv stranice, informacije o vlasništvu, opis i slično. Oni su među mnogim stvarima koje ispituju tražilice kada *gledaju* stranicu. Iako nije nužno, vrlo ih je korisno upotrebljavati.

## 3.2. CSS

CSS (engl. *Cascading Style Sheets*) je stilski jezik koji se koristi za opisivanje prezentacije dokumenta napisanog pomoću HTML jezika. S razvojem weba, prvotno su u HTML dodavani elementi za definiciju prezentacije, ali je vrlo brzo primijećena potreba za stilskim jezikom koji će HTML osloboditi potrebe prikazivanja sadržaja, što je i prvenstvena namjena HTML-a, kao i njegova oblikovanja, čemu danas i služi CSS. Drugim riječima, stil definira kako prikazati HTML elemente. Pomoću CSS-a se uređuje i sam izgled i raspored stranice. Aktualna inačica CSS-a koja se danas koristi je CSS3.

Stilska lista (engl. *style sheet*) u CSS-u sastoji se od nekoliko pravila. Svako se pravilo sastoji od selektora i deklaracijskog bloka.

Selektor (engl. *selector*) je dio označnog jezika (engl. *markup*) na koji se primjenjuje stil. Selektor može biti:

- svi elementi istog tipa, primjerice svi h2 elementi
- elementi određenog id ili class atributa, pri čemu id predstavlja jedinstven element
- class može obuhvaćati više od jednog elementa elementi u odnosu na druge elemente u DOM-u (*Document Object Modelu*)

Pseudoklase su klase koje omogućuju opisivanje informacija koje nisu dostupne u DOM-u poput *:hover* koji identificira sadržaj samo ako korisnik drži pokazivač nad sadržajem.

Deklaracijski blok predstavlja vitičaste zagrade unutar kojih se nalaze deklaracije. Svaka se deklaracija sastoji od svojstva, dvotočke (:) i vrijednosti. Između svake dvije uzastopne deklaracije mora se nalaziti točka zarez (;). Vrijednosti mogu biti ključne riječi poput *center* (sredina) i *inherit* (naslijedi), brojučane vrijednosti poput 100 (debljina fonta), 200px (200 piksela), 50vw (50% širine viewpota) ili 80% (80% širine prozora). Vrijednosti boja mogu biti ključne riječi, primjerice *red* za crveno, heksadecimalne vrijednosti, primjerice *#FF0000*, RGB vrijednosti od 0 do 255, primjerice *rgb(255, 0, 0)*, RGBA vrijednosti koje uključuju i aplha prozirnost, primjerice *rgba(255, 0, 0, 0.8)*.

CSS se može pisati unutar same HTML stranice na dva načina: 1) kao stilove u zaglavlju HTML dokumenta, između `<style>` i `</style>` elementa, 2) unutar samih HTML oznaka, primjerice `<p style="color: magenta">Primjer teskta</p>`.

Osim toga, CSS je moguće definirati u posebnom dokumentu i rabiti pomoću poziva: `<link rel="stylesheet" href="primjer.css">`.

### 3.3. Javascript

JavaScript, ili skraćeno JS, programski je jezik i jedna od temeljnih tehnologija World Wide Web-a uz HTML i CSS. Uzimajući u obzir 2022. godinu, 98% web aplikacija koriste JavaScript na klijentskoj strani za ponašanje web aplikacija, često inkorporirajući druge biblioteke. Svi veći web pretraživači imaju JavaScript *engine* koji izvodi kod na klijentskim uređajima. Uključujemo ga u web aplikaciju da bi je učinili dinamičnijom.

Omogućuje definiranje ponašanja za prethodno napisane HTML elemente stranice.



Objektno je zasnovan skriptni jezik jer programer ne definira samo tip podataka, nego definira i vrstu funkcija koje se primjenjuju na strukture podataka. Na ovaj način struktura podataka postaje objekt koji uključuje i funkcije i podatke. Osim toga, programeri mogu kreirati i odnose između jednog i drugog objekta. JavaScript je skriptni jezik jer se sastoji od niza naredbi koje se izvode u *interpreteru*, a da se prethodno ne kompajlira sadržaj. Odnosno, ne prevodi se u strojni jezik iz kojega nikada nećemo saznati originalni jezik, nego se naredbe izravno čitaju iz koda. Zbog ove se karakteristike JavaScript izvršava na strani korisnika, točnije na uređaju na kojem je pokrenut sadržaj s JavaScriptom.

Prilagođen je ECMAScript standardu, a ima dinamičko tipkanje, objektnu orijentaciju na prototipu i prvoklasne funkcije. Također, podržava više paradigmi upravljanih događajima, te funkcionalne i imperativne stilove programiranja. Ima sučelja za programiranje aplikacija (engl. *API – Application Programming Interface*) za rad s tekstom, datumima, standardnim strukturama podataka i DOM-om (engl. *DOM – Document Object Model*). ECMAScript standard ne uključuje nikakav ulaz/izlaz (engl. *I/O*) podataka, kao što su umrežavanje, pohrana ili grafički sadržaji. U praksi, web preglednik ili drugi *runtime* sustav pruža JavaScriptu API-je za I/O. JavaScript *engine* ispočetka su samo korišteni u web preglednicima, a sada su ključne komponente nekih servera i mnoštva aplikacija. Najpopularniji *runtime* sustav za ovu upotrebu je Node.js.

Česta greška koja se može čuti jest da su Java i JavaScript ista stvar. Iako imaju slično ime, sintaksu i standardne biblioteke, Java i JavaScript nisu isti i uvelike se razlikuju po dizajnu.

U nastavku su navedene karakteristike JavaScripta:

**skriptni jezik** – kao što je već spomenuto, JavaScript je lagani skriptni jezik napravljen za izvršavanje na strani klijenta u web pregledniku. Budući da nije dizajniran kao jezik opće namjene i posebno je dizajniran za web aplikacije, skup biblioteka također je usmjeren primarno ka web aplikacijama.

**baziran na interpreteru** – JavaScript interpreterski je jezik, a ne kompajlerski, stoga je srodniji jezicima kao što su Ruby ili Python. Web pretraživač interpretira izvorni kod JavaScripta, liniju po liniju i izvodi ga. S druge strane, kompajlirani jezik mora biti kompajliran u izvršni byte-kod, kao što je primjerice slučaj kod Jave i C++.

**lagan za upotrebu** – JavaScript nije kompajlirani jezik pa ne prevodi unaprijed u byte-kod. Kako god, prati kompilacijsku paradigmu koja se zove *just-in-time (JIT)*, što znači da se

pretvori u bajt-kod prije samog pokretanja, a to ga čini lakim za upotrebu i ne tako memorijski zahtjevnim. Zbog ove karakteristike i slabiji uređaji mogu pokrenuti JavaScript.

**osjetljivost na velika i mala slova** – JavaScript vrlo je osjetljiv na velika i mala slova. Sve ključne riječi, varijable, nazivi funkcija i drugi identifikatori mogu i moraju pratiti samo dosljedno pisanje velikih i malih slova. Primjerice, ako imamo `var hitCounter = 5` i `var hitcounter = 5`, varijable `hitCounter` i `hitcounter` su dvije potpuno različite varijable zbog razlike u nazivima. Također, važno je napomenuti da su i ključne riječi poput `var` isto osjetljive na velika i mala slova

### 3.4. TypeScript

TypeScript je besplatni programski jezik otvorenog koda kojeg je razvio i koji održava Microsoft. Strogi je sintaktički nadskup JavaScripta i omogućuje dodatno statičko pisanje u jeziku. Dizajniran je za razvoj velikih aplikacija i prevodi se u JavaScript. Budući da je nadskup JavaScripta, postojeći JavaScript programi također su važeći TypeScript programi.

Može se koristiti za razvoj JavaScript aplikacija kako za izvođenje na klijentskoj strani, tako i za izvođenje na poslužiteljskoj strani (kao što je slučaj s Node.js-om ili Denom). Dostupno je više opcija za transpilaciju. Može se koristiti zadani kompajler TypeScripta ili se može pozvati Babel kompajler za pretvaranje TypeScripta u JavaScript.

Podržava definiranje datoteka koje mogu sadržavati informacije o vrsti postojećih JavaScript biblioteka, slično kao što C++ datoteke zaglavlja mogu opisati strukturu postojećih objektnih datoteka. To omogućuje drugim programima da koriste vrijednosti definirane u datotekama kao da su statički upisani TypeScript entiteti. Postoje datoteke zaglavlja trećih strana za popularne biblioteke kao što su jQuery, MongoDB i D3.js. Dostupna su i TypeScript zaglavlja za osnovne module Node.js-a, što omogućuje razvoj Node.js programa unutar TypeScript-a.

Sam kompajler TypeScripta napisan je u TypeScriptu i prevodi se u JavaScript. Licenciran je pod licencom *Apache 2.0*. TypeScript je uključen kao prvoklasni programski jezik u Microsoft Visual Studio 2013, u ažuriranju broj 2 i kasnijim, zajedno s programskim jezikom C# i drugim Microsoft programskim jezicima. Službeno proširenje također omogućuje Visual Studio 2012 da podržava TypeScript.

### 3.5. Node.js

Node.js je backend okruženje JavaScripta, otvorenog koda, koje se izvodi na V8 *engine*-u, te izvodi JavaScript kod izvan web pretraživača. Kreiran je za izvođenje skalabilnih web

aplikacija. Node.js omogućuje programerima korištenje JavaScripta za pisanje alata komadne linije i za skriptiranje serverske strane, odnosno za izvođenje skripti na serverskoj strani za kreiranje sadržaja dinamične web aplikacije prije nego što je stranica poslana na klijentski web pretraživač.

Zastupa *JavaScript bilo gdje* paradigmu, objedinjujući razvoj web aplikacija oko jednog programskog jezika, umjesto korištenja različitih programskih jezika za skripte na serverskoj strani i skripte na klijentskoj strani.

Posjeduje arhitekturu vođenu događajima sposobnu za asinkroni ulaz/izlaz (engl. *I/O*). Ovi izbori dizajna ciljaju na optimizaciju propusnosti i skalabilnosti u web aplikacijama s brojnim I/O operacijama, kao i za web aplikacije u stvarnom vremenu, primjerice programi za komunikaciju u stvarnom vremenu i igre preglednika.

Node.js omogućuje kreiranje web servera i mrežnih alata koristeći JavaScript i kolekciju modula koji upravljaju različitim ključnim funkcionalnostima. Moduli su dostupni za datotečni I/O sustav, mrežu (DNS, HTTP, TCP, TLS/SSL, ili UDP), binarne sustave, kriptografske funkcije, te druge funkcionalnosti. Moduli Node.js-a koriste API dizajniran da smanji kompleksnost pisanja serverskih aplikacija. JavaScript jedini je jezik kojeg Node.js podržava nativno, ali postoje i mnogi *compile-to-JS* jezici. Kao rezultat toga, Node.js aplikacije mogu biti pisane u jezicima kao što su CoffeeScript, Dart, TypeScript, ClojureScript i slično.

Node.js primarno se koristi za izgradnju mrežnih programa kao što su web serveri. Najvažnija razlika između Node.js-a i PHP-a je ta što većina funkcija u PHP-u blokira do završetka (naredbe se izvršavaju tek nakon završetka prethodnih naredbi), dok Node.js funkcije ne blokiraju (naredbe se izvršavaju istovremeno ili čak paralelno i koriste povratne pozive za signaliziranje završetka ili neuspjeha).

Službeno je podržan na Linux, macOS i Windows operacijskim sustavima.

### **3.5.1. Node Package Manager (NPM)**

Node Package Manager, ili skraćeno NPM, je upravitelj paketa za programski jezik JavaScript. NPM je zadani upravitelj paketa za JavaScript *runtime* okruženje Node.js-a. Sastoji se od komandne linije klijenta, koja se još naziva NPM, i online baze podataka javnih i plaćenih privatnih paketa, koje se nazivaju NPM registar. Registru se pristupa putem klijenta, a dostupni se paketi mogu pregledavati i pretraživati putem NPM web stranice.

NPM je uključen kao preporučena karakteristika u Node.js instalaciji. Sastoji se od komandne linije klijenta koja komunicira s udaljenim registrom. Dopušta korisnicima korištenje i distribuciju JavaScript modula koji su dostupni u registru. Paketi u registru su u formatu CommonJS i uključuju datoteku meta podataka u JSON formatu. Više od 1.3 milijuna paketa dostupni su u glavnom NPM registru.

Registar nema postupak provjere za podnošenje, što znači da paketi koji se tamo nalaze mogu potencijalno biti lošije kvalitete, nesigurni ili zlonamjerni. Umjesto toga, NPM se oslanja na izvješća korisnika kako bi uklonio pakete ukoliko krše pravila ako su niske kvalitete, nesigurni, ili pak zlonamjerni. NPM izlaže statistiku koja uključuje broj preuzimanja i broj zavisnih paketa kako bi pomogao programerima u procjeni kvalitete paketa.

NPM može upravljati paketima koji su lokalne ovisnosti određenog projekta, kao i globalno instaliranim JavaScript alatima. Kada se koristi kao upravitelj ovisnosti za lokalni projekt, NPM može instalirati u jednoj naredbi sve ovisnosti projekta kroz datoteku *package.json*.

U *package.json* datoteci svaka ovisnost može specificirati niz odgovarajućih verzija koristeći semantičku shemu verzioniranja, što omogućuje programerima da automatski ažuriraju svoje pakete dok istovremeno izbjegavaju neželjene promjene koje bi mogle izazvati pogreške.

### 3.6. Angular

Angular je okvir (engl. *framework*) otvorenog koda baziran na TypeScriptu koji se primjenjuje u web aplikacijama, a omogućuje kreiranje reaktivnih aplikacija na jednoj stranici (engl. *SPA – Single-Page-Application*) [12]. Vođen je od strane Angular tima u Google-u, te zajednicom pojedinaca i korporacija. Angular je potpuno ponovljena verzija istoga tima koji je kreirao AngularJS.

Neke od značajki Angulara:

- povezivanje podataka
- vlastiti sustav za kreiranje i upravljanje komponentama
- podrška za upravljanje i validaciju formi
- korištenje direktiva

Dopušta brzo i kvalitetno razvijanje klijentskog dijela aplikacije, odnosno dijela kojeg korisnik vidi. Angular pojednostavljuje razvoj aplikacija tako što daje jednu razinu apstrakcije programeru, ali to može utjecati na fleksibilnost aplikacije.

Angular tim najavio je promjenu kojom je pojasnio stvari, pri čemu se *AngularJS* (*Angular1*) odnosi na prvu verziju popularnog *framework*-a, a samo *Angular* na drugu verziju i sve verzije nakon nje. Do sada smo se susreli s dvanaest različitih verzija, od kojih je posljednja verzija 14, puštena u rad u lipnju 2022. godine.

Razlog zašto imamo toliko puno verzija je taj što Angular tim u rad pušta novu verziju svakih šest mjeseci. To ne znači da svaka nova verzija ima velike razlike u odnosu na prethodnu, nego da se Angular tim dosljedno drži svojeg rasporeda objavljivanja novih verzija.

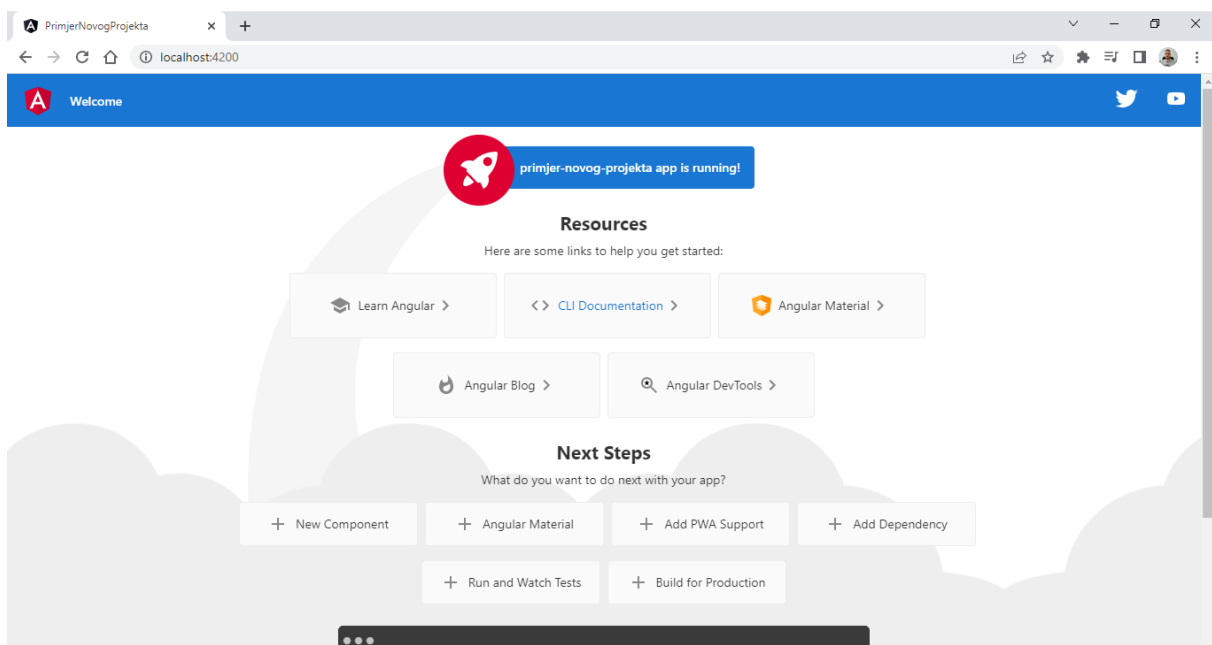
Svaka je nova verzija kompatibilna s prethodnom (engl. *backward compatible*), primjerice verzija Angular11 kompatibilna je s verzijom Angular10 i slično. Najveća razlika je između verzije 1 i 2, stoga se verzija Angular1 naziva *AngularJS*, a verzija Angular2 i svaka nakon nje, jednostavno *Angular*.

## 4. Izrada web aplikacije

U ovom će poglavlju biti objašnjena izrada Angular web aplikacije za predviđanje cijene mobitela na osnovu unesenih parametara. Korisniku će biti omogućeno da izradi novu konfiguraciju mobitela, ili da podnese (engl. *submit*) ispunjenu konfiguraciju. Nakon toga, korisniku će biti prikazan raspon cijene unutar kojega se nalazi mobitel s odabranim specifikacijama.

### 4.1 Izrada novog projekta

Potrebno je kreirati folder u koji ćemo smjestiti sve vezano za aplikaciju. Nakon toga, potrebno je otvoriti terminal putem Visual Studio Code uređivača. U terminalu je potrebno pozvati naredbu `ng new ime-projekta`, što je u našem slučaju `ng new RUAP`. Nakon izvršavanja naredbe potrebno se pozicionirati unutar foldera u kojemu je projekt, pomoću naredbe `cd`, točnije `cd RUAP`. Posljednji korak, potrebno je pozvati naredbu `ng serve` koja pokreće našu web aplikaciju na web lokaciji <http://localhost:4200/>. Svaki put kada napravimo neku promjenu unutar projekta i spremimo, naredba `ng serve` automatski će primijeniti te promjene na web aplikaciji. Treba imati na umu da jednom kada ju pokrenemo, naredba `ng serve` mora ostati pokrenuta dokle god radimo na projektu i želimo imati ovakav dinamički prikaz promjena na web aplikaciji.



Slika 4.1.1. Početni izgled web aplikacije

## 4.2 Angular elementi unutar projekta

U ovome će poglavlju biti prikazani Angular elementi koji su kreirani i koji se koriste unutar projekta. Ranije u ovome radu su isti elementi objašnjeni teorijski, a sada će biti prikazan način na koji se kreiraju i kako izgledaju unutar projekta. Potrebno je napomenuti kako se neće koristiti svi opisani Angular elementi, budući da je riječ o jednostavnijoj web aplikaciji. Stoga će biti korišteni samo neki elementi.

### 4.2.1 Angular komponente

Angular komponente moguće je kreirati na dva načina; prvi je način *ručnim* kreiranjem i on zahtjeva više posla. Primjerice, desnim klikom unutar *app* foldera možemo odabrati kreiranje novog foldera (engl. *New Folder*) ili nove datoteke (engl. *New File*). Nakon odabira, možemo dodati novu datoteku. Također, moramo se pobrinuti da u *app.module.ts* datoteci dodamo novu komponentu koju smo stvorili unutar *NgModule* deklaracija i ažuriramo popis *import* komponenti na vrhu datoteke. Drugi način je da unosimo naredbe unutar terminala. Primjerice za kreiranje nove komponente unijet ćemo naredbu *ng generate component naziv-komponente* ili skraćeno *ng g c naziv-komponente*. Ovaj način jednostavniji je za korištenje, ali i bolji je odabir jer smo tako sigurni da nismo zaboravili niti jedan uvoz (engl. *import*) komponenti unutar *app.module.ts* datoteke, što moramo napraviti u prvoj opciji. Svi se uvozi komponenti obavljaju automatski.

Kao primjer kreiranja nove Angular komponente bit će korištena druga opcija.

```
PS C:\Users\Marko\Stranica\notus> ng generate component nova-komponenta
CREATE src/app/nova-komponenta/nova-komponenta.component.html (30 bytes)
CREATE src/app/nova-komponenta/nova-komponenta.component.spec.ts (656 bytes)
CREATE src/app/nova-komponenta/nova-komponenta.component.ts (310 bytes)
CREATE src/app/nova-komponenta/nova-komponenta.component.css (0 bytes)
UPDATE src/app/app.module.ts (3540 bytes)
```

Slika 4.2.1.1. Kreiranje nove Angular komponente

### 4.2.2 Angular usmjeravanje

Angular usmjeravanje (engl. *routing*) kreirano je na slijedeći način: Desni klik na željeni folder gdje želimo dodati datoteku, odabir opcije nova datoteka (engl. *New File*) i imenovanje *naziv.module.ts*. Odabran je naziv *app-routing.module.ts*, a u nastavku je prikazan dio koda navedene datoteke.

```

1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { ClassificationComponent } from '../classification/classification.component';
4
5 const routes: Routes = [
6   { path: '', redirectTo: '/classification', pathMatch: 'full' },
7   { path: 'classification', component: ClassificationComponent },
8 ];
9
10 @NgModule({
11   imports: [RouterModule.forRoot(routes)],
12   exports: [RouterModule]
13 })
14 export class AppRoutingModule { }
15

```

Slika 4.2.2.1. Kod *app-routing.module.ts* datoteke

### 4.3 Konačan izgled web aplikacije

Nakon kreiranja potrebnih komponenti i pravilno odrađenog usmjeravanja, web aplikacija poprimila je svoj konačan izgled. U nastavku se može vidjeti izgled aplikacije.

Mobile price classification

Please, enter mobile phone specifications and submit the form.

Battery power  
mAh

Bluetooth  
▼

Clock speed  
GHz

Dual Sim  
▼

Front camera  
megapixels

4G  
▼

Internal memory  
GB

Depth  
centimeters

Slika 4.3.1. Konačan izgled web aplikacije – prvi dio



Depth  
centimeters

Weight  
grams

Number of cores  
▼

Primary camera  
megapixels

Height  
pixels

Width  
pixels

RAM  
MB

Screen height  
centimeters

Screen width  
centimeters

Talk time

**Slika 4.3.2.** Konačan izgled web aplikacije – drugi dio

Screen width  
centimeters

Talk time  
hours

3G  
▼

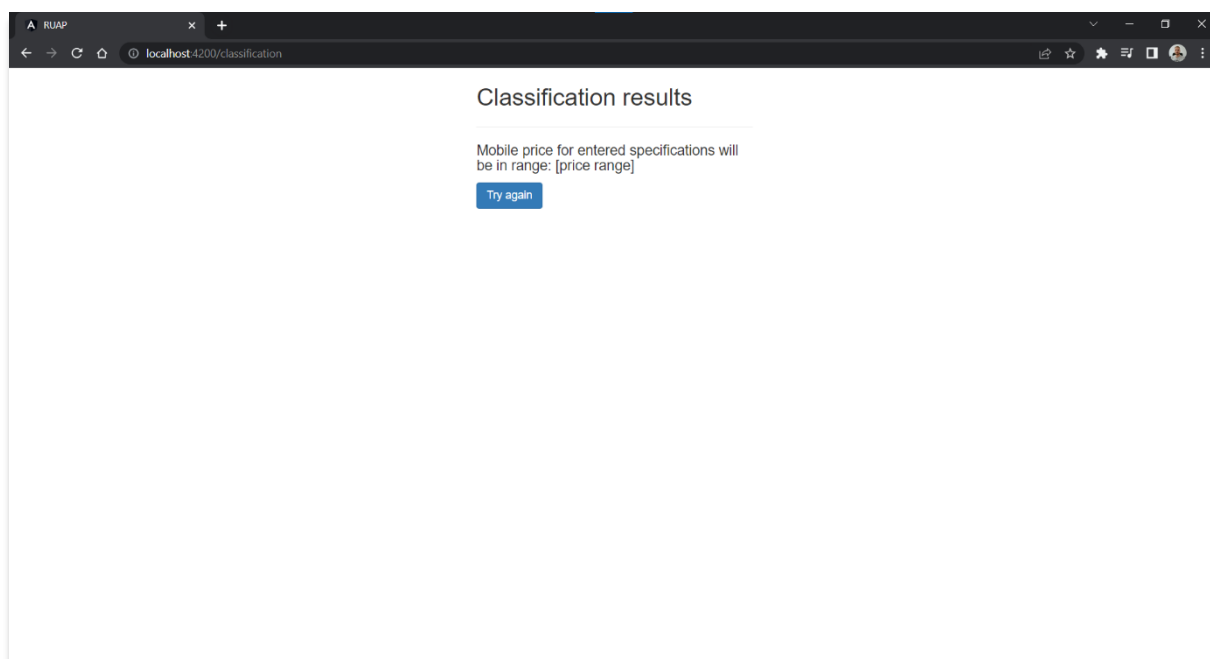
Touch screen  
▼

WiFi  
▼

Price range  
Last but not least, enter the price range you expect the mobile phone with given specifications will be in.  
Reminder:  
0 - low cost  
1 - medium cost  
2 - high cost  
3 - very high cost  
▼

New configuration Submit

**Slika 4.3.3.** Konačan izgled web aplikacije – treći dio



**Slika 4.3.4.** Konačan izgled web aplikacije – četvrti dio

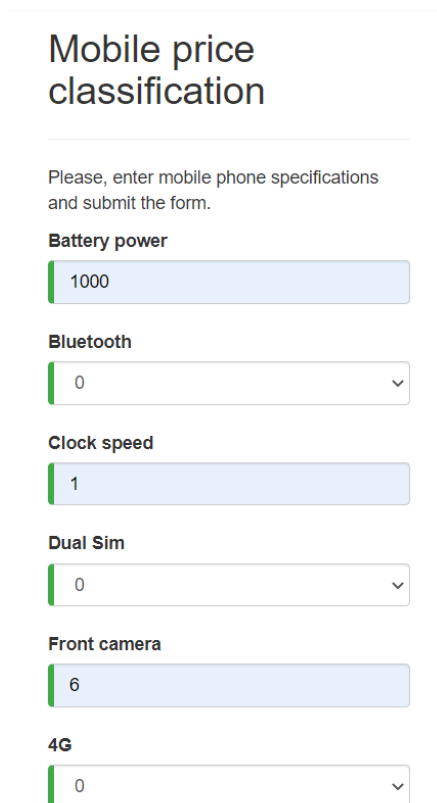
A form titled 'Mobile price classification'. Below the title is a message: 'Please, enter mobile phone specifications and submit the form.' There are three input fields: 'Battery power' with a unit 'mAh', 'Bluetooth' with a dropdown menu showing '1', and 'Clock speed' with a unit 'GHz'. A red error message 'Please, fill in this form.' is displayed below the 'Battery power' field.

**Slika 4.3.5.** Primjer validacije podataka prilikom postavljanja specifikacija mobitela

### 4.3.1 Testiranje rada web aplikacije

U nastavku će biti prikazani primjeri testiranja rada web aplikacije. Vrijednosti parametara, kao i odgovor (engl. *response*) koji se dobije s API-ja bit će prikazani na slikama u nastavku.

Prvi primjer:



**Mobile price classification**

Please, enter mobile phone specifications and submit the form.

**Battery power**  
1000

**Bluetooth**  
0

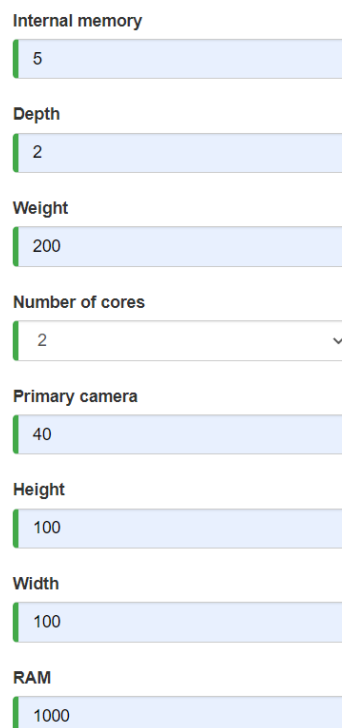
**Clock speed**  
1

**Dual Sim**  
0

**Front camera**  
6

**4G**  
0

Slika 4.3.1.1 Unošenje specifikacija mobitela – prvi dio



**Internal memory**  
5

**Depth**  
2

**Weight**  
200

**Number of cores**  
2

**Primary camera**  
40

**Height**  
100

**Width**  
100

**RAM**  
1000

Slika 4.3.1.2 Unošenje specifikacija mobitela – drugi dio

Screen height

12

Screen width

3

Talk time

4

3G

0

Touch screen

0

WiFi

0

**Slika 4.3.1.3** Unošenje specifikacija mobitela – treći dio

Kao posljednji unos u web aplikaciji, korisnik treba odabrati cjenovni rang u kojem očekuje da će mobitel s unesenim specifikacijama biti.

**Price range**

Last but not least, enter the price range you expect the mobile phone with given specifications will be in.

Reminder:

0 - low cost

1 - medium cost

2 - high cost

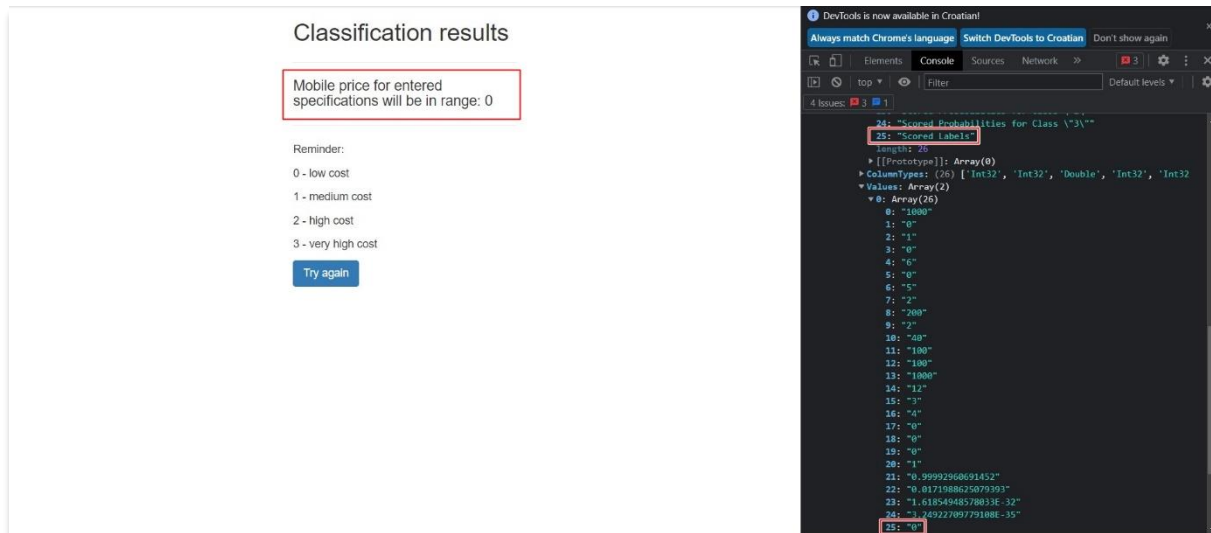
3 - very high cost

1

New configuration Submit

**Slika 4.3.1.4** Unošenje specifikacija mobitela – četvrti dio

Na kraju se može vidjeti da je korisnik krivo predvidio cijenu mobitela za unesene parametre. Korisnik je predvidio cjenovni rang 1, odnosno srednju skupocu, a odgovor (engl. *response*) s API-ja je vratio da će mobitel s unesenim specifikacijama biti u cjenovnom rang 0, odnosno niske skupoce.



Slika 4.3.1.5 Odgovor (engl. *response*) s API-ja

Drugi primjer:

## Mobile price classification

Please, enter mobile phone specifications and submit the form.

### Battery power

### Bluetooth

### Clock speed

### Dual Sim

### Front camera

### 4G

Slika 4.3.1.6 Unošenje specifikacija mobitela – prvi dio

---

**Internal memory**

**Depth**

**Weight**

**Number of cores**

**Primary camera**

**Height**

**Width**

**RAM**

**Slika 4.3.1.7** Unošenje specifikacija mobitela – drugi dio

**Screen height**

**Screen width**

**Talk time**

**3G**

**Touch screen**

**WiFi**

**Slika 4.3.1.8** Unošenje specifikacija mobitela – treći dio

Kao posljednji unos u web aplikaciji, korisnik treba odabrati cjenovni rang u kojem očekuje da će mobitel s unesenim specifikacijama biti.

### Price range

Last but not least, enter the price range you expect the mobile phone with given specifications will be in.

Reminder:

- 0 - low cost
- 1 - medium cost
- 2 - high cost
- 3 - very high cost

2

New configuration

Submit

Slika 4.3.1.9 Unošenje specifikacija mobitela – četvrti dio

Kao i u prethodnom primjeru, može se vidjeti da je korisnik krivo predvidio cijenu mobitela za unesene parametre. Korisnik je predvidio cjenovni rang 2, odnosno visoku skupoću, a odgovor (engl. *response*) s API-ja je vratio da će mobitel s unesenim specifikacijama biti u cjenovnom rang 3, odnosno vrlo visoke skupoće.

### Classification results

Mobile price for entered specifications will be in range: 3

Reminder:

- 0 - low cost
- 1 - medium cost
- 2 - high cost
- 3 - very high cost

Try again

DevTools is now available in Croatian

Always match Chrome's language | Switch DevTools to Croatian | Don't show again

Elements | Console | Sources | Network

7 Issues

25: "Scored Labels"

[[Prototype]]: Array(0)

ColumnTypes: (26) ["Int32", "Int32", "Double", "Int32", "Int32"]

Values: Array(26)

0: "3000"

1: "1"

2: "2"

3: "1"

4: "12"

5: "1"

6: "512"

7: "2"

8: "200"

9: "6"

10: "40"

11: "200"

12: "200"

13: "8000"

14: "12"

15: "3"

16: "4"

17: "1"

18: "1"

19: "1"

20: "2"

21: "8.39040031857758E-23"

22: "3.66174576988039E-28"

23: "1.3554505951729E-08"

24: "1"

25: "3"

length: 26

Slika 4.3.1.10 Odgovor (engl. *response*) s API-ja

## **5. Zaključak**

Tržište mobitela iz dana u dan se sve više mijenja. Specifikacije mobitela se poboljšavaju, ali isto tako raste i cijena njihove izrade i cijena konačnog proizvoda. Uz pomoć ovog seminarskog rada uočena je prednost korištenja modela strojnog učenja te njihova primjena u web, računalnim ili mobilnim aplikacijama. Rješenje seminarskog rada ostvareno je korištenjem HTML, CSS, Bootstrap i TypeScript tehnologija, što je objedinjeno u Angular frameworku. Priprema i treniranje modela odrađeno uz pomoć Azure Machine Learning alata.



## Literatura

- [1] *Angular*, dostupno na <https://angular.io/>
- [2] *Bootstrap*, dostupno na <https://getbootstrap.com/>
- [3] *CSS*, dostupno na <https://www.w3.org/Style/CSS/Overview.en.html>
- [4] *HTML*, dostupno na <https://html.com/>
- [5] *JavaScript*, dostupno na <https://www.javascript.com/>
- [6] *Kaggle*, dostupno na <https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?fbclid=IwAR3S1jVhoKbsmneUj891rw1xQIDYkAaZfdJK16VADknilzEIMwcTLjxbhVE>
- [7] *Node.js*, dostupno na <https://nodejs.org/en/about/>
- [8] *TypeScript*, dostupno na <https://www.typescriptlang.org/>