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

Sveučilišni studij

PREDVIĐANJE USPJEHA UČENIKA

Raspoznavanje uzoraka i strojno učenje

Ivan Jakab

Osijek, 2019. godina

1. UVOD

[1]Cilj seminara je istrenirati različite regresijske modele kako bi predvidjeli uspjeh učenika na kraju godine, na osnovu podataka koje su dali. Modeli će biti istrenirani na skupu podataka koji se nalaze na uci, Student Performance Data Set. Podaci su prikupljeni na učenicima u Portugalu te se pratila konačna ocjena učenika na osnovu različitih značajki.

Neke od značajki koje se prate su spol, dob, status roditelja, vrijeme utrošeno na učenje, izlasci, konzumiranje alkohola, izvannastavne aktivnosti itd. Nakon što se model istrenira na tim značajkama, cilj je izraditi jednostavno web sučelje za unos vlastitih podataka, koje će potom dati rezultate koje su predvidjeli različiti modeli.

2. POSTOJUĆA RJEŠENJA I KORIŠTENE BIBLIOTEKE

Teško je naći konkretan projekt koji koristi baš ovaj skup podataka, mada vjerojatno postoji. No, postoji jako puno rješenja sa regresijom općenito.

U pythonu postoji puno biblioteka koji rješavaju probleme regresije. Najpoznatija, koja se koristi u ovome radu je scikit learn. Ona daje izrađene modele (ne trenirane), koji većinom dijele isto sučelje prema van – *fit* metoda za učenje, *predict* metoda za dobivanje rezultata. Osim ovih, nudi još puno metoda za predobradu podataka, evaluaciju modela i slično.

Osim scikit learn, koristi se i biblioteka *pandas*, koja omogućava manipuliranje skupom podataka kroz svoj *DataFrame.* Treba i spomenuti biblioteku *numpy*, koju koriste ostale u pozadini, a i u seminaru se koriste radi lakšeg odrađivanja matematičkih operacija.

[1] [4]

3. IZRADA REGRESIJSKOG DIJELA

3.1. Učitavanje skupa podataka

Za učitavanje i dohvaćanje podataka, izrađen je poseban modul *dataset.py*. Prilikom inicijalizacije, učitava podatke iz .csv datoteke u pandas DataFrame. Nakon učitavanja, čisti nepotpune podatke (one u kojima nisu svi stupci popunjeni). Zatim, odrađuje predobradu koristeći modul koji će kasnije biti obrađen. Konačno, razdvaja podatke na one za trening i za testiranje, te osigurava da svi modeli koriste iste podatke.

Izvorni skup podataka sadrži podatke o 33 značajke za 395 instanci. U projektu se prate sljedeće značajke: starost, spol, adresa, veličina obitelj, status oba roditelja, vrijeme provedeno na put, izvannastavne aktivnosti, učenje, pristup internetu, slobodno vrijeme, konzumacija alkohola i odsutnost s nastave. Predviđa se vrijednost "G3", koja predstavlja uspjeh učenika, a poprima vrijednosti od 1 do 20. Mali podskup prikazan je sljedećom tablicom.

age	famsize	Medu	Fedu	absences	G3
18	GT3	4	4	6	6
17	GT3	1	1	4	6
15	LE3	1	1	10	10
15	GT3	4	2	2	15
16	GT3	3	3	4	10

Tablica 1Prikaz podskupa skupa podataka

Prema van, ovaj modul daje metode za dohvaćanje testnog, trening i cijelog skupa podataka. Kod modula je sljedeći:

```
import pandas
from numpy import random
from kod.preprocess import preprocess

SPLIT_RATE = 0.8

random.seed(12)
dataframe_original = pandas.read_csv('./dataset.csv', sep=r'\s*;\s*',
engine='python')
dataframe_dropped = dataframe_original.dropna()
dataframe = preprocess(dataframe_dropped)

#display info about dataset
print(dataframe_original.head())
print(len(dataframe_original))
dataframe_original.to_clipboard()


# split train and test data
mask = random.rand(len(dataframe)) < SPLIT_RATE
train = dataframe[mask]
test = dataframe[~mask]
```

```
def getFullDataset():
    return dataframe


def getTrainDataset():
    return train


def getTestDataset():
    return test
```

3.2. Predobrada podataka

Nakon učitavanja podataka, slijedi njihova predobrada. U ovu svrhu je također napravljen poseban modul, *preprocess.py*. Ovo je izdvojeno od modula za skup podataka jer će jednaku predobradu morati proći podaci za učenje i testiranje, kao i podaci pristigli preko web servera.

Ovaj modul prema van daje metodu *preprocess*, koji uzima DataFrame podataka te ih obrađuje za ulaz u modele.

Prvi je korak selekcija značajki koje se prate, jer neke nisu primjenjive u ovome slučaju (npr. škola, koja predstavlja neku školu u Portugalu). Ovo ujedno i osigurava da se ne uzmu svi podaci sa web servera, jer nikada ne možemo znati što će klijent poslati.

Zatim, slijedi pretvaranje riječi u brojeve – skup podataka dolazi sa mnogim riječima koje modeli ne mogu razumjeti (npr. *teacher*), pa ih je potrebno prebaciti u brojeve. Ovo je napravljeno dinamičkim preslikavanjem podataka, a konkretna implementacija se može vidjeti u kodu.

Na kraju, slijedi korak skaliranja. Za ovo se koristi MinMaxScaler, koji dolazi sa scikit learn bibliotekom. U ovome koraku će se različite vrijednosti numeričkih podataka koji imaju različite raspone (npr. 1-5 konzumacija alkohola, ali 0-93 za izostanke) skalirati na vrijednosti od 0 do 1. Ovim se korakom znatno povećava efikasnost procjene parametara.

Osim ove metode, modul daje metodu *get_x* i *get_y* (razdvajanje DataFramea na ulaz i izlaz modela) te neke metode koje će biti korisne kada se uključi web server.

```
from sklearn.preprocessing import MinMaxScaler
import pandas

TRACKED_FEATURES =
['age','sex','address','famsize','Pstatus','Medu','Fedu','Mjob','Fjob','traveltime'
,'studytime','activities','higher','internet','romantic','freetime','goout','Dalc',
'Walc','absences']
#sample =
[15,"F","U","LE3","T",3,4,"health","teacher",2,3,"yes","no","yes","yes",1,3,2,4,23]
PREDICT_VALUE = 'G3'
tracked_columns = TRACKED_FEATURES.copy()
tracked_columns.append(PREDICT_VALUE)

strMap = {
    "sex": {'M': 0, 'F': 1},
    "address": {'U': 0, 'R': 1},
    "famsize": {'LE3': 0, 'GT3': 1},
    "Pstatus": {'T': 0, 'A': 1},
    'Mjob': {'teacher': 0, 'health': 1, 'services': 2, 'at_home': 3, 'other': 4},
    'Fjob': {'teacher': 0, 'health': 1, 'services': 2, 'at_home': 3, 'other': 4},
    'activities': {'yes': 1, 'no': 0},
    'higher': {'yes': 1, 'no': 0},
    'internet': {'yes': 1, 'no': 0},
    'romantic': {'yes': 1, 'no': 0},
}


def preprocess(dataframe):
    # select only some columns
    dataframe = dataframe[tracked_columns]

    # convert text to numeric
    for key in strMap:
        dataframe[key] = [strMap[key][item] for item in dataframe[key]]

    # normalize data
    scaler = MinMaxScaler()
    scaled = scaler.fit_transform(dataframe.values)
    dataframe_scaled = pandas.DataFrame(scaled, columns=dataframe.columns.tolist())

    return dataframe_scaled


def get_x(dataframe):
    return dataframe[TRACKED_FEATURES]


def get_y(dataframe):
    return dataframe[PREDICT_VALUE]


def preprocess_and_get_x(dataframe):
    preprocessed = preprocess(dataframe)
    return get_x(preprocessed)


def dataframe_from_features(features):
    new_dataframe = pandas.DataFrame(columns=tracked_columns)
    new_dataframe.loc[0] = features
    return new_dataframe
```

3.3. Treniranje i evaluacija modela

Kao i za ostale stvari, i za ovo je napravljen modul, *models.py*. Kada se modul inicijalizira, prvo inicijalizira sve modele koji se koriste u seminaru. Njih sprema u rječnik, gdje je ključ naziv modela, a vrijednost instanca modela. Linearni modeli koji se koriste jesu Linearna, Bayesova i Lasso regresija. Uz njih koristi se i Random Forest regresija, te na kraju neuronska mreža - Multilayer Perceptron.

[2]Iako svi modeli imaju različit način rada, olakotna okolnost je da svi dijele jednako sučelje prema van. Tako svi modeli na sebi imaju metodu *fit* koja prima listu značajki i listu ciljnih vrijednosti – DataFrame trening podataka na osnovu kojih trenira. Uz ovu, imaju i metodu *predict* koja prima listu značajki, te za rezultat daje predviđene vrijednosti izlaza.

Poslije inicijalizacije, dinamički se prolazi kroz sve modele te se na njima poziva *fit* metoda kako bi se model istrenirao. Nakon treninga, modeli daju svoju predikciju za testni skup podataka. Zatim se računa srednja kvadratna pogreška između predviđenih i pravih rezultata testnog skupa.

Rezultati evaluacije su sljedeći

Naziv modela	MSE test	MSE trening
Linearna regresija	0.0511	0.0429
Bayesova regresija	0.0449	0.0441
Lasso	0.0407	0.0552
Random forest	0.0433	0.0420
MLP	0.0571	0.0225

Tablica 2 Srednja kvadratna pogreška modela

Vidi se da najmanju pogrešku za testni skup ima Lasso regresija, no svi modeli se dobro ponašaju.

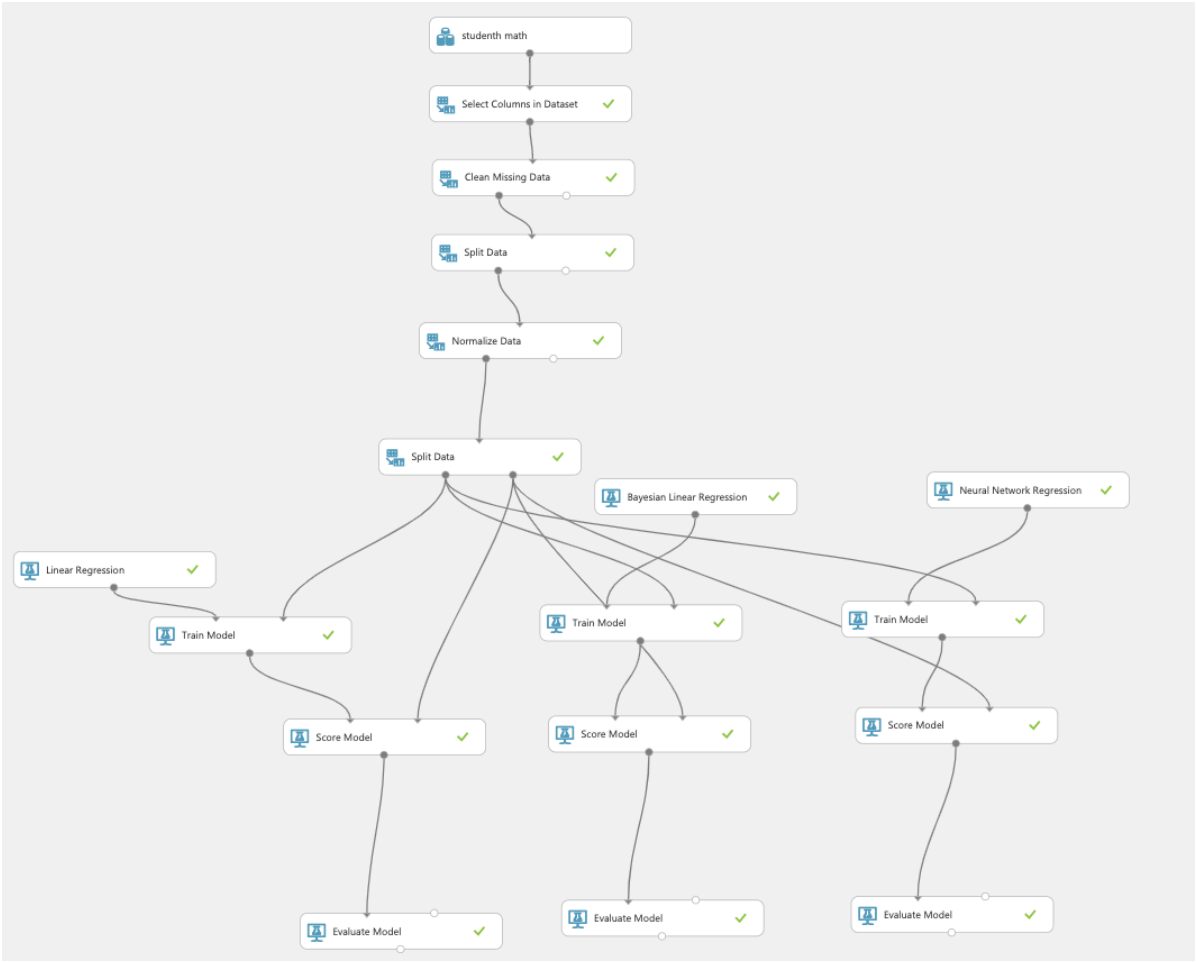
[2] [3]

```
from kod import preprocess, dataset
from sklearn.linear_model import LinearRegression, BayesianRidge, Lasso
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
from sklearn.neural_network import MLPRegressor


data_train = dataset.getTrainDataset()
data_test = dataset.getTestDataset()
X_train = preprocess.get_x(data_train)
X_test = preprocess.get_x(data_test)
Y_train = preprocess.get_y(data_train)
Y_test = preprocess.get_y(data_test)


models = {
    'Linear Regression': LinearRegression(),
    'Bayesian Ridge': BayesianRidge(compute_score=True),
    'Lasso': Lasso(alpha=0.1),
    'Random Forest': RandomForestRegressor(max_depth=2, random_state=0),
    'Multilayer Perceptron': MLPRegressor()
}

for modelName in models:
    # train a model
    models[modelName].fit(X_train, Y_train)

    # predict test data
    Y_test_predicted = models[modelName].predict(X_test)
    MSE = mean_squared_error(Y_test, Y_test_predicted)
    print(f'MSE for {modelName} is {MSE}')
```

Prema van, ovaj modul daje metodu *predict_all*, koja za dani DataFrame predviđa rezultat po svim modelima koji su trenirani, te vraća riječnik u kojem je ključ ime modela, a vrijednost predviđeni rezultat. Prije predviđanja će napraviti istu predobradu podataka kao i prilikom učenja modela.

Osim ove, dodana je i metoda *predict_all_from_features*, koja je korisna za web server – uzet će polje značajki, od njega napraviti DataFrame i pozvati *predict_all*.

```
def predict_all(dataframe, prep):
    result = {}
    X = 0
    if prep:
        X = preprocess.preprocess_and_get_x(dataframe)
    else:
        X = preprocess.get_x(dataframe)
    for modelName in models:
        result[modelName] = models[modelName].predict(X)[0]
    return result


def predict_all_from_features(features, prep):
    dataframe = preprocess.dataframe_from_features(features)
    return predict_all(dataframe, prep)
```



Slika 1 Prikaz cijelog procesa, izrađeno u ml studiju

4. IZRADA WEB APLIKACIJE

4.1. Poslužiteljska strana

[3]Kako bi se otvorio jednostavan web server, koristi se biblioteka flusk. Ona omogućuje jednostavno slušanje na HTTP upite, dobivanje podataka iz upita i slanje odgovora.

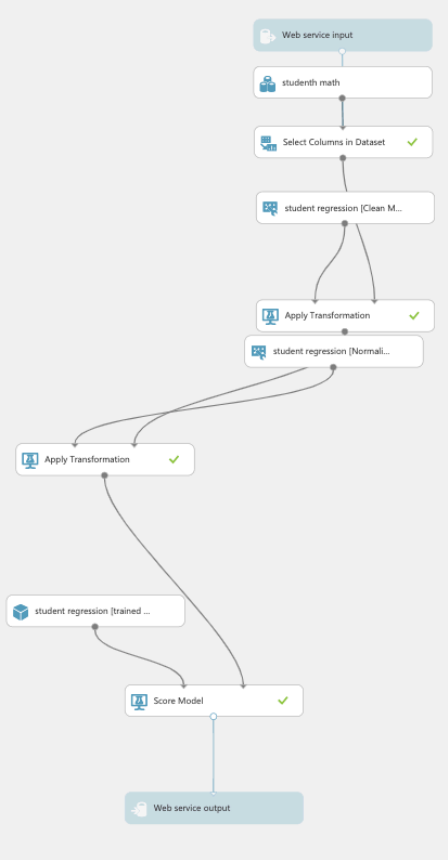
Prvo, radi se ruta koja poslužuje statički direktorij. Jednostavno uzme put sa klijenta i posluži datoteku iz direktorija „web“ koja odgovara putu koji je klijent zatražio. Osim ove, postoji i ruta koja poslužuje *index.html* kada nema zatraženog puta.

Posljednja ruta služi za komunikaciju s modelima. Ona je REST tipa, vraća i prima JSON kroz odgovor umjesto neke datoteke. Tipa je post, a očekuje polje značajki. Uzima te značajke poslane sa klijenta te s njima poziva metodu *predict_all_from_features* iz modula za

[3] [1]

modele opisanu gore, kako bi dobila rezultat predikcije modela. Taj rezultat vraća kroz odgovor u JSON obliku.

```
from flask import request, jsonify, Flask, send_from_directory
from kod import models
from json import loads

app = Flask('student_performance_server')


@app.route('/predict', methods=['POST'])
def predict():
    features = 0
    try:
        features = request.json.get('features')
    except:
        # for some reason, i cant send content type application/json, it is always
reset to text/plain
        features = loads(request.data)["features"]
    modelOutputs = models.predict_all_from_features(features, prep=True)
    return jsonify(modelOutputs), 200


@app.route('/<path:path>', methods=['GET'])
def serve_web(path):
    return send_from_directory('web', path)


@app.route('/', methods=['GET'])
def serve_homepage():
    return send_from_directory('web', 'index.html')



app.run()
```



Slika 2 Prikaz web servisa, izrađeno u ml studiju

4.2. Klijentska strana

Na klijentskoj strani, koriste se standardne tehnologije, HTML, CSS i JavaScript. Od biblioteka koristi se bootstrap za brže stiliziranje i chart.js za prikaz grafova. Sav JavaScript je napisan u *script.js* datoteci, a css u *style.css*

U datoteci *index.html* koristi definiran je markup stranice, te su učitane biblioteke, stil i skripte. Većina markupa definira formu s poljima za unos značajki. Značajke imaju jednaka imena kao u DataFrameu. Kada se forma podnese, okida se upit na gore opisani poslužitelj, te se pomoću biblioteke crta graf koji prikazuje predikciju različitih modela.

```
<div class="row">
    <div class="col">
       <input type="text" class="form-control" placeholder="Izvannastavne
aktivnosti" id="activities" value="yes" required>
    </div>
    <div class="col">
       <input type="text" class="form-control" placeholder="Zelim nastaviti
skolovanje" id="higher" value="no" required>
    </div>
    <div class="col">
       <input type="text" class="form-control" placeholder="Imam pristup internetu"
id="internet" value="yes" required>
    </div>
    <div class="col">
       <input type="text" class="form-control" placeholder="U vezi sam"
```

```
id="romantic" value="no" required>
    </div>
</div>
```

JavaScriptom se sluša na predaju forme. Kada se to dogodi, čitaju se unesene vrijednosti značajki. Od njih se izrađuje polje značajki. S tim poljem se poziva poslužiteljeva ruta za komunikaciju s modelima, a poslužitelj vraća u odgovoru vrijednosti koje su pojedini modeli predvidjeli na gore opisan način.

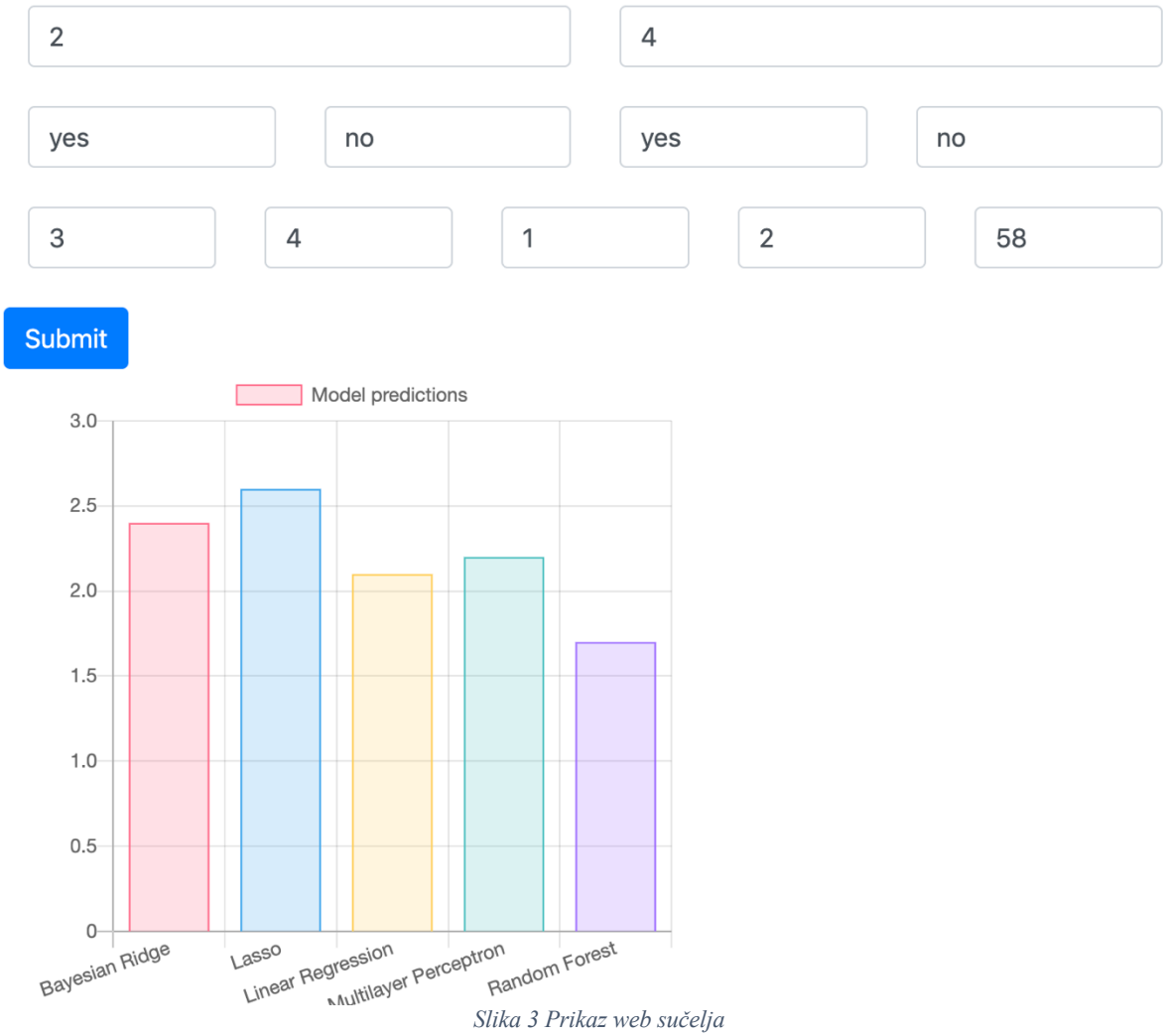
```
mainForm.addEventListener('submit', async function (e) {
    e.preventDefault()
    let features = getFeatures()
    let data = await callApi({features})
    plot(data)
})

async function callApi(data) {
    let res = await fetch('/predict', {
        method: 'POST',
        body: JSON.stringify(data),
        mode: 'no-cors', // no-cors, *cors, same-origin
        headers: {
           'Accept': 'application/json',
            'Content-Type': 'application/json'
        }
    })
    let json = await res.json()
    for(let key of Object.keys(json)) {
        json[key] *= 5
        json[key] = json[key].toFixed(1)
        json[key] = Number(json[key])
    }
    return json
}

function getFeatures() {
    let features = []
    for(let featureName of trackedFeatures) {
        let element = document.getElementById(featureName)
        if(Number(element.value)) features.push(Number(element.value))
        else features.push(element.value)
    }
    features.push(0)
    return features
}
```

[4]Kada su predikcije dostupne, pomoću *chart.js* biblioteke se crta graf predviđenih vrijednosti u obliku bar charta.

[4] [2]

```
function plot(models) {
    var ctx = document.getElementById('myChart').getContext('2d');
    new Chart(ctx, {
        type: 'bar',
        data: {
            labels: Object.keys(models),
            datasets: [{
                label: 'Model predictions',
                data: Object.values(models),
                backgroundColor: [
                    'rgba(255, 99, 132, 0.2)',
                    'rgba(54, 162, 235, 0.2)',
                    'rgba(255, 206, 86, 0.2)',
                    'rgba(75, 192, 192, 0.2)',
                    'rgba(153, 102, 255, 0.2)',
                ],
                borderColor: [
                    'rgba(255, 99, 132, 1)',
                    'rgba(54, 162, 235, 1)',
                    'rgba(255, 206, 86, 1)',
                    'rgba(75, 192, 192, 1)',
                    'rgba(153, 102, 255, 1)',
                ],
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                yAxes: [{
                    ticks: {
                        beginAtZero: true
                    }
                }]
            }
        }
    });
}
```

U samom sučelju, korisnik treba ispuniti formu značajki. Svako polje popunjava vrijednošću značajke koja je predviđena za to polje. Kada je popuni, podnosi formu te mu se crta graf predviđenih vrijednosti za svaki od istreniranih modela.



Slika 3 Prikaz web sučelja

5. ZAKLJUČAK

U radu je prikazan klasični problem regresijske predikcije. Istrenirano je i evaluirano više različitih modela, te je za svaki prikazan kako se ponaša. Modeli su istrenirani sa relativno malom srednjom kvadratnom pogreškom.

Dodatno, izrađen je web servis kao primjer kako modeli mogu jednostavno komunicirati sa vanjskim svijetom. Također je omogućen grafički prikaz i usporedba različitih modela kako bi se uočile razlike na ovome skupu podataka.

6. LITERATURA

[1] "Flask dokumentacija," [Online]. Dostupno na: https://flask.palletsprojects.com/en/1.1.x/#api-reference. [Accessed 2020].

[2] chart.js, "chart.js dokumentacija," [Online]. Dostupno na: https://www.chartjs.org/docs/latest/. [Accessed 2020].

[3] "Scikit learn dokumentacija," [Online]. Dostupno na: https://scikit-learn.org/stable/modules/classes.html. [Accessed 2020].

[4] A. S. P. Cortez, "UCI dataset," [Online]. Dostupno na: https://archive.ics.uci.edu/ml/datasets/student+performance. [Accessed 2020].