

**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

¹Cilj 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 jednostavni 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 P. Cortez and A. Silva. Using Data Mining to Predict Secondary School Student Performance. In A. Brito and J. Teixeira Eds., Proceedings of 5th FUTURE BUSINESS TECHNOLOGY Conference (FUBUTEC 2008) pp. 5-12, Porto, Portugal, April, 2008, EUROSIS, ISBN 978-9077381-39-7.

3. IZRADA REGRESIJSKOG DIJELA

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 gdje nisu svi stupci popunjeni). Zatim, odrađuje predobradu koristeći model koji će kasnije biti obrađen. Konačno, razdvaja podatke na one za trening i za testiranje, te osigurava da svi modeli koriste iste podatke.

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

```
1 import pandas
2 from numpy import random
3 from preprocess import preprocess
4
5 SPLIT_RATE = 0.8
6
7 random.seed(12)
8 dataframe_original = pandas.read_csv('./dataset.csv', sep=r'\s*;\s*', engine='python')
9 dataframe_dropped = dataframe_original.dropna()
10 dataframe = preprocess(dataframe_dropped)
11
12 # split train and test data
13 mask = random.rand(len(dataframe)) < SPLIT_RATE
14 train = dataframe[mask]
15 test = dataframe[~mask]
16
17
18 def getFullDataset():
19     return dataframe
20
21
22 def getTrainDataset():
23     return train
24
25
26 def getTestDataset():
27     return test
```

Slika 1 Dataset modul

2. Predobrada podataka

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

Ovaj modul prema van daje metodu *preprocess*, koji uzima dataframe podaka 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 mapiranjem podataka, 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 pojačava preciznost istreniranih modela.

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', 'stud
#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},
}
```

Slika 2 Modul za predobradu interno

```

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

```

Slika 3 Modul za preobradu sučelje prema van

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 mapu, 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.

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

```
MSE for Linear Regression is 0.05112690521172566  
MSE for Bayesian Ridge is 0.04495712531573687  
MSE for Lasso is 0.0407203372856914  
MSE for Random Forest is 0.043366417085643115  
MSE for Multilayer Perceptron is 0.05710171728382834
```

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

```

import dataset
import preprocess
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}')

```

Slika 4 Modul za modele interno

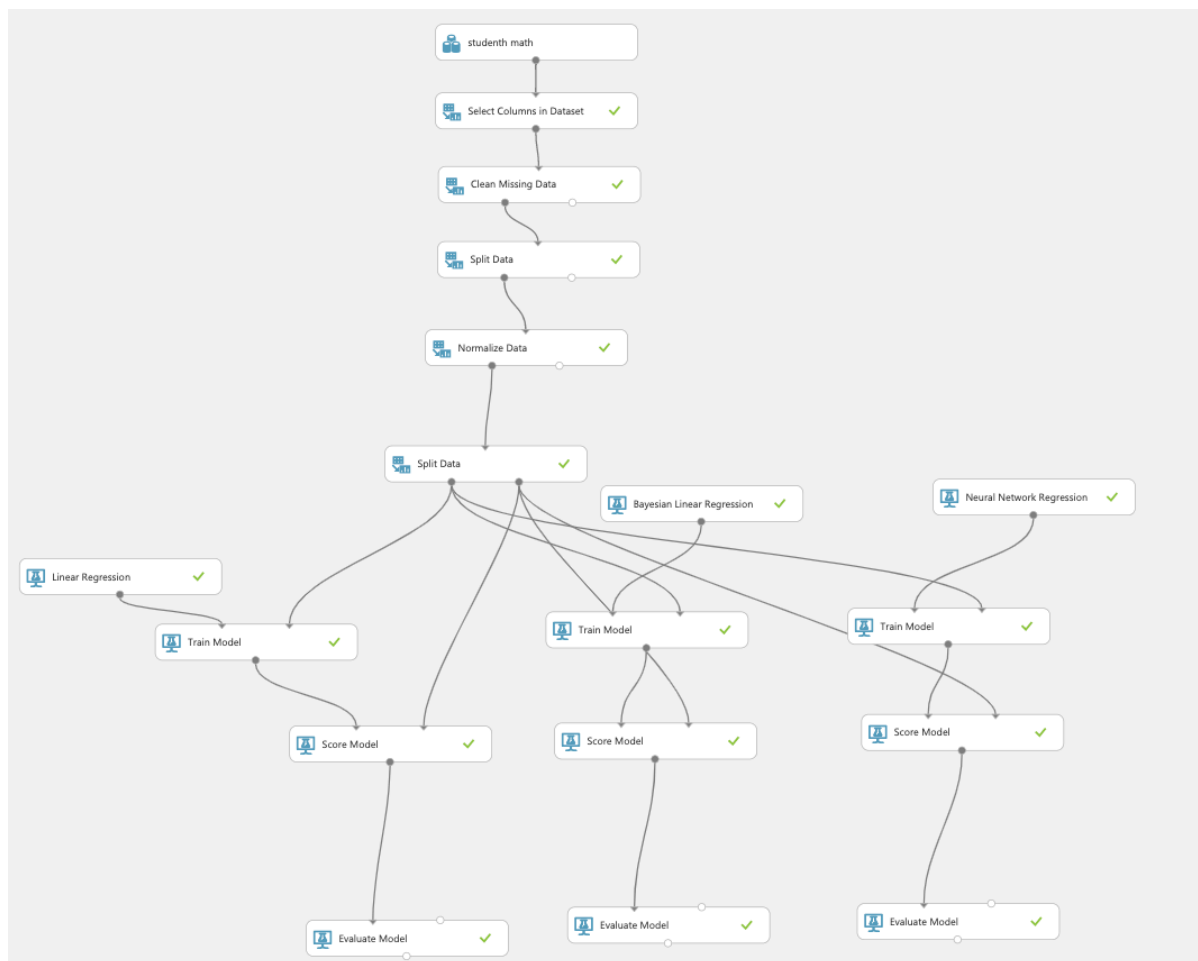
Prema van, ovaj modul daje metodu *predict_all*, koja za dani dataframe predviđa rezultat po svim modelima koji su trenirani, te vraća mapu u kojoj je ključ ime modela, a vrijednost predviđeni rezultat. Prije predviđanja će napraviti istu predobradu podataka kao i prilikom učenja modela.

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 5 Modul za modele, sučelje prema van



Slika 6 Prikaz cijelog procesa, izrađeno u ml studiu

4. IZRADA WEB APLIKACIJE

1. Poslužiteljska strana

Kako bi se otvorio jednostavan web server, koristi se biblioteka flask. On 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 indeks.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 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
import models
from json import loads

app = Flask('student_performance_server')

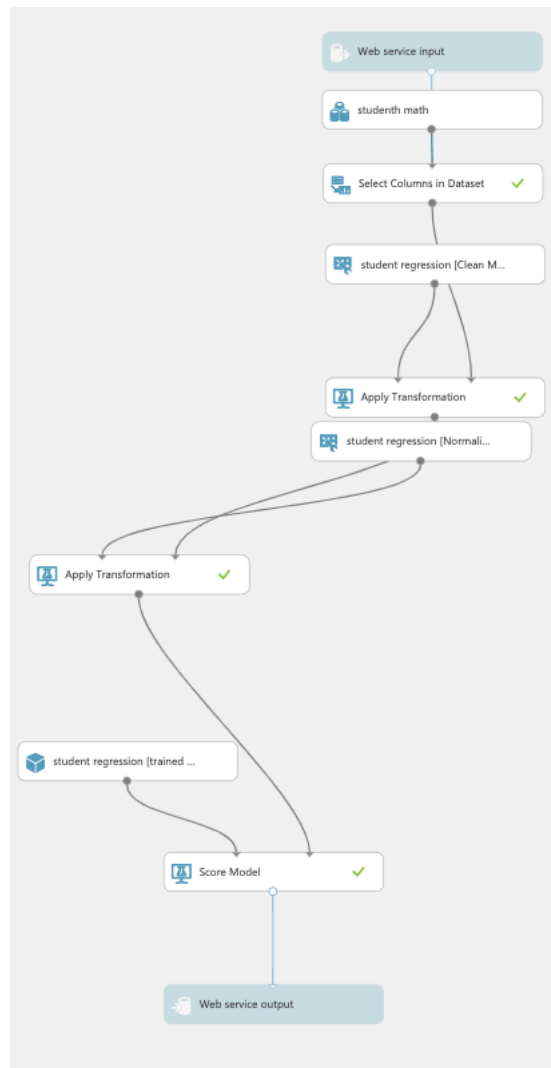
@app.route('/predict', methods=['POST'])
def predict():
    features = 0
    try:
        features = request.json.get('features')
    except:
        # for some reason, i cant send content type application/json, it is always reset to text/plain
        features = loads(request.data)["features"]
    modelOutputs = 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 7 Kod poslužitelja



Slika 8 Prikaz web servisa, izrađeno u ml studiu

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 iscrtava 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>

```

Slika 9 Primjer markupa za značajke

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
}

```

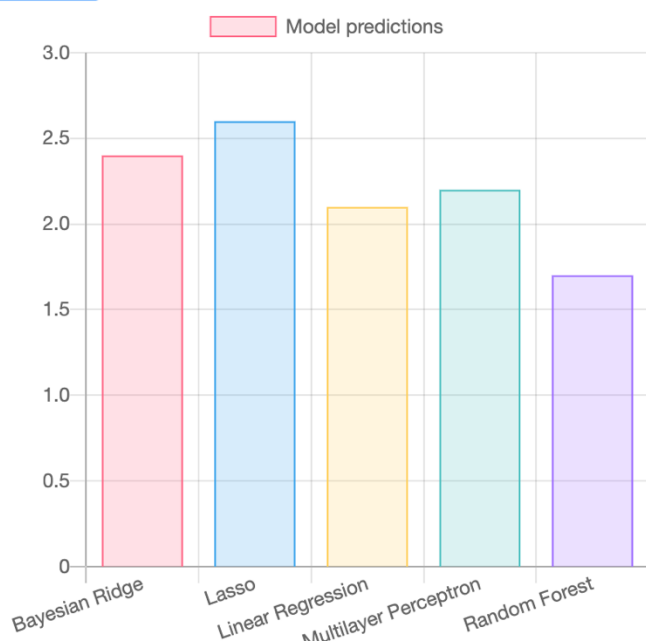
Slika 10 Slanje upita na poslužitelj

Kada su predikcije dostupne, pomoću *chart.js* biblioteke se iscrtava graf predviđenih vrijednosti u obliku bar charta.

```
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  
                    }  
                }]  
            }  
        }  
    });  
}
```

Slika 11 Iscrtavanje grafa

Submit



Slika 12 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.