**VISOKA ŠKOLA ELEKTROTEHNIKE I RAČUNARSTVA STRUKOVNIH STUDIJA**

**BEOGRAD**



**PROJEKTNI ZADATAK**

**Tema: “Otkrivanje dijabetesa kod žena**

**Pima Indijanskog nasleđa”**

**Predmet: Veštačka inteligencija**

**Datum:** *26.05.2022.*

***Milanče Andrejić NRT-8/20***

**S A D R Ž A J**

1. Uvod.............................................................................. 3

2. Opis problema koji se rešava .......................................4

2.1. Opis skupa podataka ..........................................4

2.2. Cilj projekta i analiza podataka............................5

2.3. Opis skupa podataka ..........................................8

3. Rešenje..........................................................................10

3.1. Izbor modela......................................................10

3.2. Evaluacija modela .............................................12

Prilog (rešenje)...................................................................13

Reference...........................................................................20

1. **Uvod**

Dijabetes je oboljenje koje nastaje usled nedovoljnog lučenja ili delovanja insulina. Insulin je hormon koji se stvara u žlezdi gušterači – pankreasu i odgovoran je za pravilan metabolizam ugljenih hidrata. Cilj je dijagnostikovati u što kraćem periodu kako bi se na vreme krenulo sa odgovarajućom terapijom i regulisanjem ishrane. Rano dijagnostikovanje dijabetesa može sprečiti neke od mnogo ozbiljnijih problema kao što su infarkt miokarda, moždani udar, pogoršanje vida ili smanjenje funkcije bubrega.

Istraživanja su pokazala da od dijabetesa češće oboljevaju žene i da je za njih dijabetes opasniji jer su šanse za infarkt miokarda duplo veće kod žena nego kod muškaraca obolelih od dijabetesa, a dijabetes takođe ima negativnog uticaja na plodnost žena.

U ovom slučaju ćemo raditi analizu podataka i obučavanje modela raditi na skupu podataka kod žena Pima Indijanskog porekla.

Ko su Indijanci Pima?

Pima (ili Akimel O'odham, koji se takođe piše Akimel O'otham, „Ljudi reke“, ranije poznati kao Pima) su grupa Indijanaca koji žive u oblasti koja se sastoji od današnje centralne i južne Arizone. Većinsko stanovništvo od dva preživela plemena Akimel O'odham su smeštena u dva rezervata: Keli Akimel O'otham na indijanskoj zajednici reke Gila i On'k Akimel O'odham na indijanskoj zajednici Salt River Pima-Maricopa.

Za potrebe ovog projekta korišćeni su [*www.kaggle.com*](http://www.kaggle.com)za skup podataka i *jupyter* za realizaciju koda.

**2. Opis problema koji se rešava**

2.1. Opis skupa podataka

Pregnancies - Broj trudnoća

Glucose - Koncentracija glukoze u plazmi

BloodPressure - Izmeren krvni pritisak

SkinThickness - Debljina kože u predelu tricepsa

Insulin - Serum insulina

BMI - Indeks telesne mase

DiabetesPedigreeFunction - Genetska sklonost dijabetesu

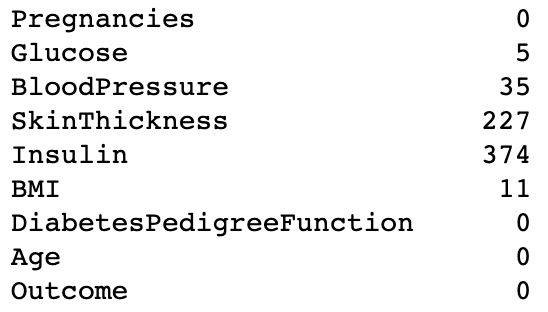
Age - Broj godina osobe

Outcome - konačan ishod 1: dijagnostikovan dijabetes, 0: nije dijagnostikovan

2.2. Cilj projekta i analiza podataka

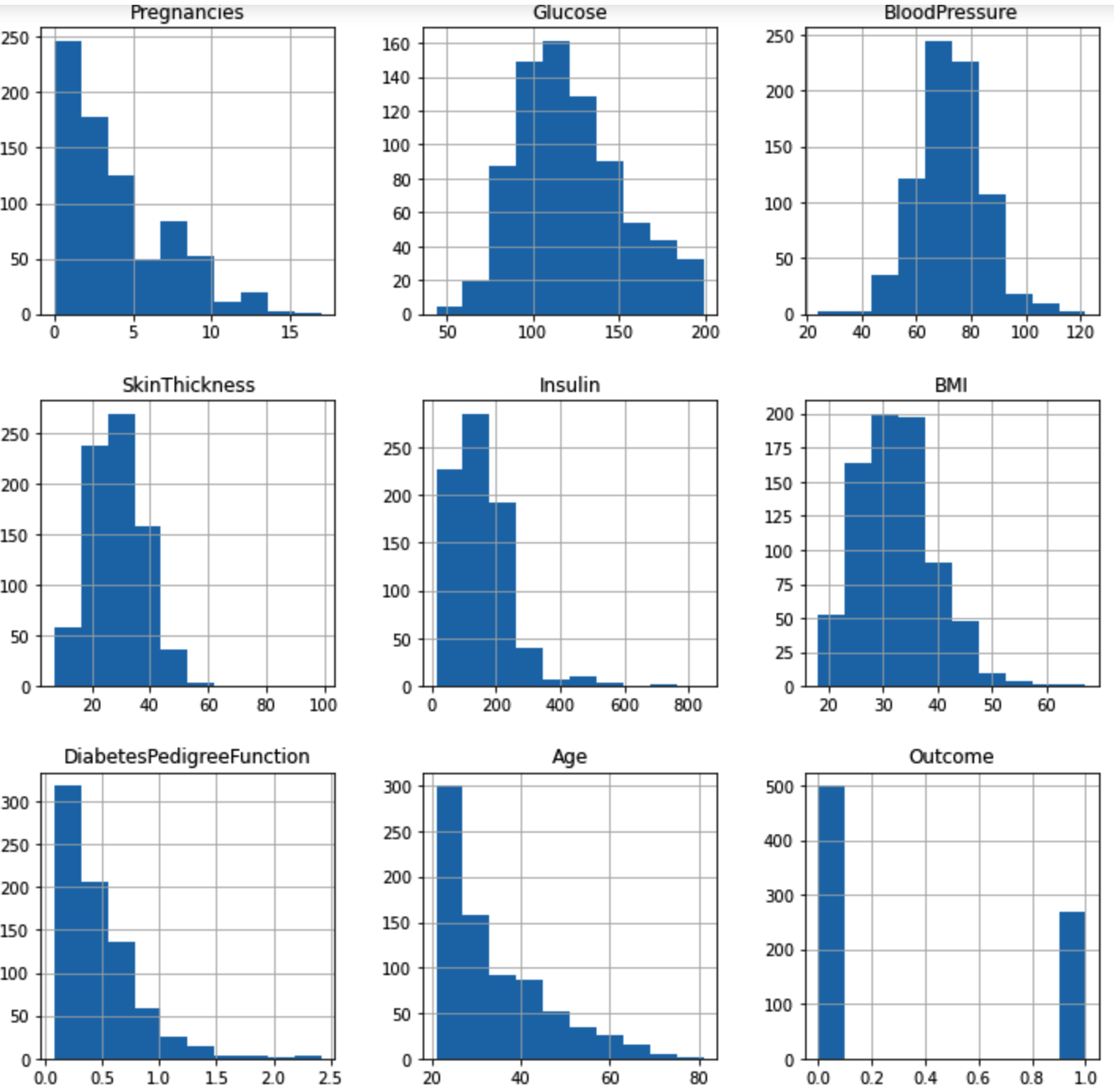
Cilj ovog projekta je da naučimo softver kako da prepozna dijabetes kod žena Pima Indijanskog porekla. Podaci pomoću kojih učimo softver su prikupljeni na *Nacionalnom institutu za dijabetes i digestivne i bubrežne bolesti* u SAD-u .

Pre svega ćemo analizirati prikupljene podatke, odrediti skup za obučavanje i validaciju softvera. Nakon toga ćemo testirati neke od algoritama za klasifikaciju i odabrati najpouzdaniji algoritam za evaluaciju našeg modela softvera. Na kraju ćemo testirati naš softver i izmeriti preciznost modela za odlučivanje.



slika 2.2.1. - Broj NULL vrednosti u datasetu

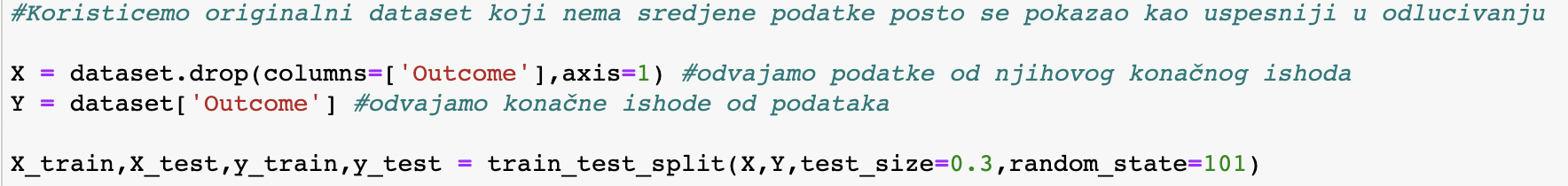
U sledećem delu možemo primetiti veliki broj NULL vrednosti koje ćemo zameniti srednjom vrednošću plus minus standardna devijacija i takav skup podataka smo prikazali na slici 2.2.2.



slika 2.2.2. - Histogram podataka

Možemo primetiti da imamo skoro dvostruko više instanci kod kojih nije dijagnostikovan dijabetes, te ćemo ovu klasu posmatrati kao *Pozitivnu klasu* u matrici konfuzije.

Iako smo sredili podatke u datasetu, koristićemo originalne podatke koji nisu sređeni pošto se takav model pokazao uspešnijim usled unošenja greške prilikom sređivanja dataseta.



slika 2.2.3.

Podelićemo originalni dataset na set za obučavanje modela i set za testiranje modela, tako da set za testiranje bude 30% originalnog dataseta.

2.3. Opis mogućih rešenja

Cilj projekta je da obučimo softver kako da prepozna dijabetes. Možemo koristiti različite algoritme za obučavanje modela ili napraviti veštačku neuralnu mrežu i istrenirati je.

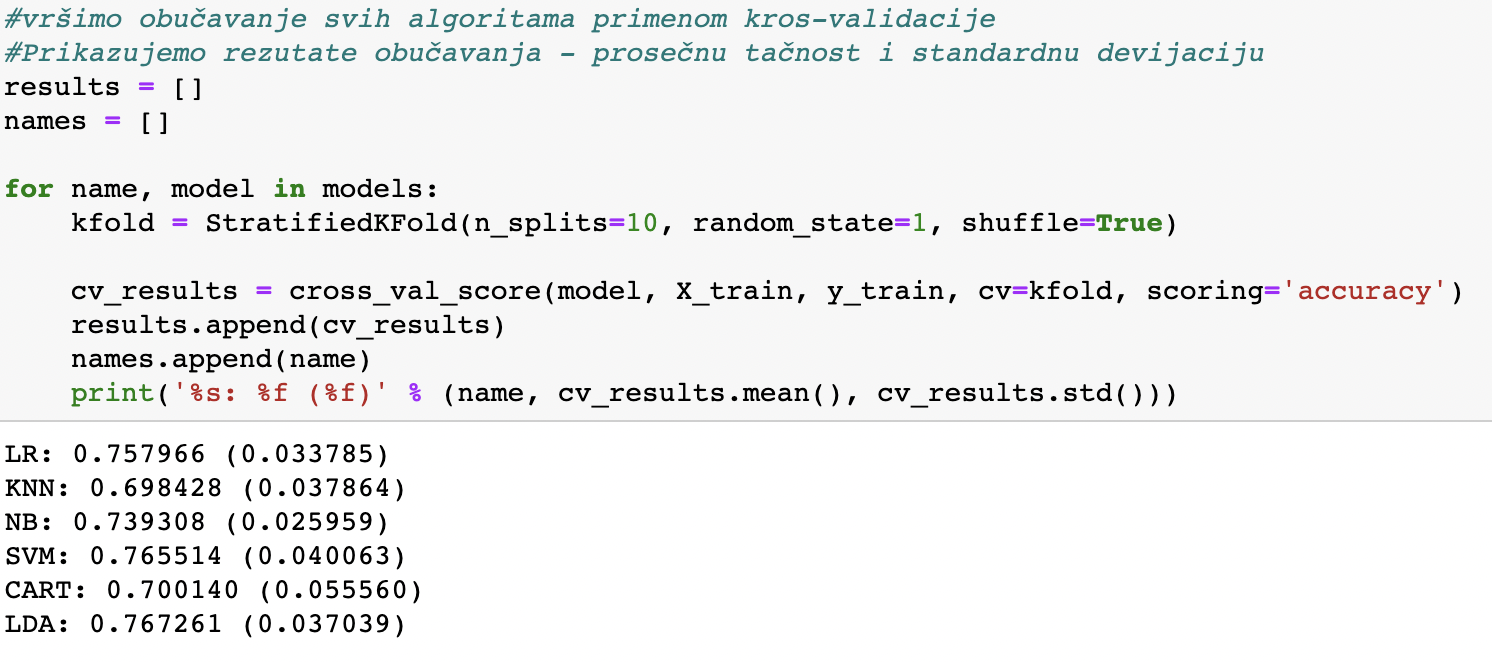
U ovom projektu ćemo koristiti jednostavnije algoritme i testiraćemo algoritme:

* Logistička regresija
* K - najbližih suseda
* Naivni Bajes
* Support Vector Machine
* Stablo odlučivanja
* Linearna diskriminantna analiza



slika 2.3.1. - kreiranje niza modela za obučavanje

Testiraćemo svaki od ovih algoritama metodom kros validacije i odabrati najbolji algoritam.

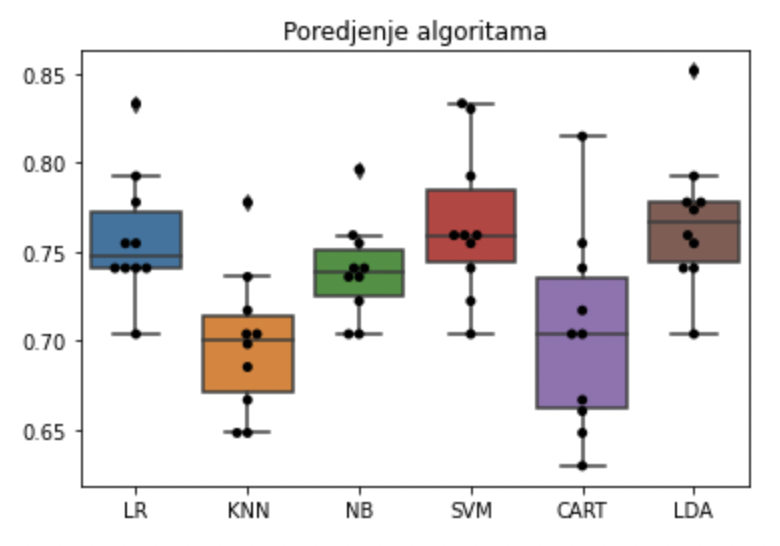


slika 2.3.2. - validacija svih modela i prikaz preciznosti

Sa slike 2.3.2. možemo primetiti najbolje rezultate kod SVM i LDA algoritama.

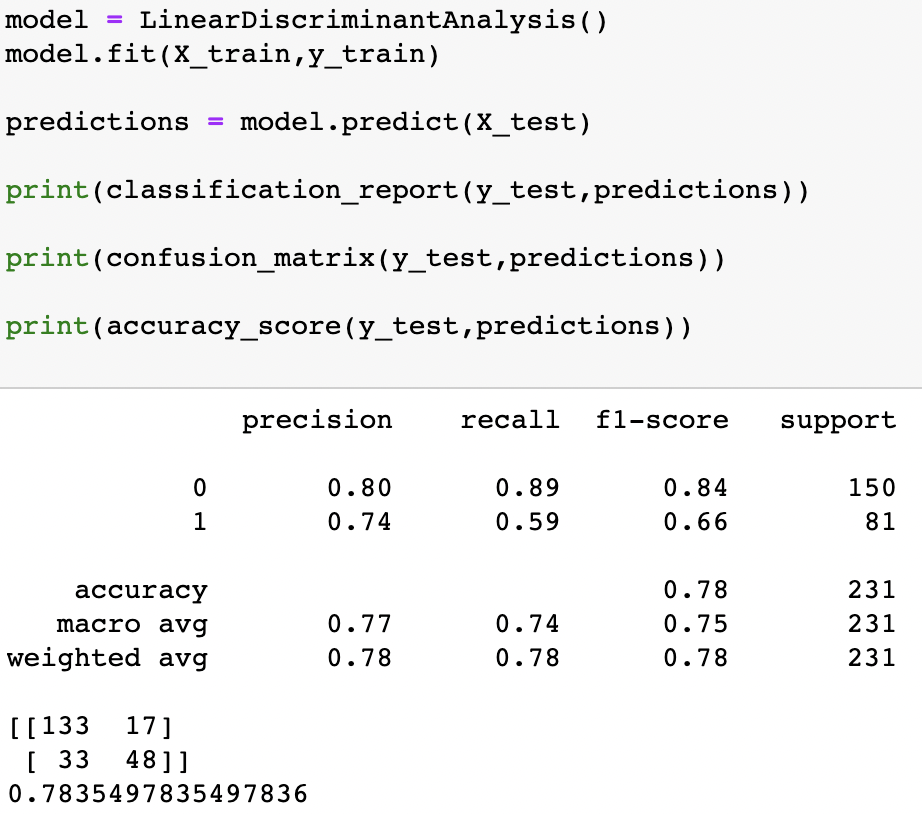
**3. Rešenje**

**3.1. Izbor modela**

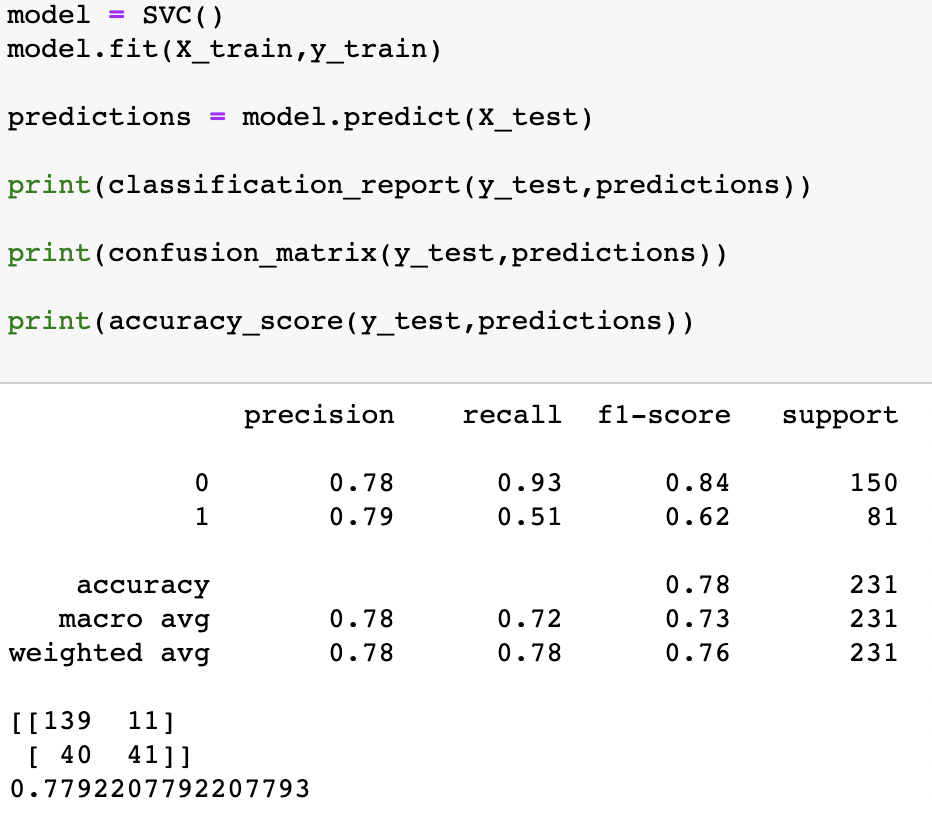
****

slika 3.1.1. - boxplot modela

Na prvi pogled iz boxplota ne možemo odrediti koji algoritam je bolji. SVM i LDA algoritmi su se pokazali kao približno tačni i pouzdani. Zato ćemo testirati oba modela i videti koji će dati bolji rezultat kroz matricu konfuzije.



slika 3.1.2. - obučavanje i testiranje LDA algoritma



slika 3.1.3. - obučavanje i testiranje SVM algoritma

**3.2. Evaluacija modela**

U daljoj evaluaciji možemo primetiti da je LDA algoritam blago pouzdaniji u dijagnostikovanju dijabetesa. Takođe, LDA algoritam se pokazao bržim za obučavanje što predstavlja dodatni razlog da ga odaberemo.

**Prilog (rešenje)**

*#uvoz biblioteka za rad sa podacima i linearnu algebru*

*import pandas as pd*

*import numpy as np*

*#uvoz biblioteka za vizuelizaciju*

*import matplotlib.pyplot as plt*

*import seaborn as sns*

*import mlxtend*

*from mlxtend.plotting import plot\_decision\_regions*

*#uvoz biblioteke za upozorenja*

*import warnings*

*#uvoz biblioteka za pripremu podataka*

*from sklearn.model\_selection import train\_test\_split*

*from sklearn.model\_selection import cross\_val\_score*

*from sklearn.model\_selection import StratifiedKFold*

*from sklearn.impute import SimpleImputer*

*#uvoz biblioteka za testiranje modela*

*from sklearn.metrics import accuracy\_score*

*from sklearn.metrics import confusion\_matrix*

*from sklearn.metrics import classification\_report*

*#uvoz algoritmima za obučavanje modela*

*from sklearn.linear\_model import LogisticRegression*

*from sklearn.svm import SVC*

*from sklearn.neighbors import KNeighborsClassifier*

*from sklearn.naive\_bayes import GaussianNB*

*from sklearn.tree import DecisionTreeClassifier*

*from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis*

*import scipy.stats as ss*

*from scipy import interp*

*from scipy.stats import randint as sp\_randint*

*from scipy.stats import uniform as sp\_uniform*

*#Učitavanje dataseta pomoću pandas biblioteke*

*#prikaz osnovnih podataka o datasetu*

*dataset = pd.read\_csv('diabetes.csv')*

*display(dataset.info(),dataset.head())*

*#Sve nemoguce nule menjamo sa 'NaN'*

*#kreiramo kopiju originalnog dataseta zbog prikaza realnih podataka i pravljenja izmena*

*data\_copy = dataset.copy(deep = True)*

*data\_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = data\_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)*

*print(data\_copy.isnull().sum()) #Prikazujemo broj NaN u datasetu(kopiji)*

*display(data\_copy.head())*

*#Pravimo izmene tako što popunjavamo vrednost NaN za Glucose sa*

*#slučajnim brojevima između srednje vrednosti plus minus standardna devijacija*

*for data in data\_copy:*

*mean = data\_copy["Glucose"].mean()*

*std = data\_copy["Glucose"].std()*

*is\_null = data\_copy["Glucose"].isnull().sum()*

*rand = np.random.randint(mean-std,mean+std,size=is\_null)*

*gluc = data\_copy["Glucose"].copy()*

*gluc[np.isnan(gluc)] = rand*

*data\_copy["Glucose"] = gluc*

*data\_copy.info()*

*#isti postupak ponavljamo i za ostale kolone*

*#Pravimo izmene tako što popunjavamo vrednost NaN za BloodPressure sa*

*#slučajnim brojevima između srednje vrednosti plus minus standardna devijacija*

*for data in data\_copy:*

*mean = data\_copy["BloodPressure"].mean()*

*std = data\_copy["BloodPressure"].std()*

*is\_null = data\_copy["BloodPressure"].isnull().sum()*

*rand = np.random.randint(mean-std,mean+std,size=is\_null)*

*bp = data\_copy["BloodPressure"].copy()*

*bp[np.isnan(bp)] = rand*

*data\_copy["BloodPressure"] = bp*

*data\_copy.info()*

*#Pravimo izmene tako što popunjavamo vrednost NaN za SkinThickness sa*

*#slučajnim brojevima između srednje vrednosti plus minus standardna devijacija*

*for data in data\_copy:*

*mean = data\_copy["SkinThickness"].mean()*

*std = data\_copy["SkinThickness"].std()*

*is\_null = data\_copy["SkinThickness"].isnull().sum()*

*rand = np.random.randint(mean-std,mean+std,size=is\_null)*

*st = data\_copy["SkinThickness"].copy()*

*st[np.isnan(st)] = rand*

*data\_copy["SkinThickness"] = st*

*data\_copy.info()*

*#Pravimo izmene tako što popunjavamo vrednost NaN za SkinThickness sa*

*#slučajnim brojevima između srednje vrednosti plus minus standardna devijacija*

*for data in data\_copy:*

*mean = data\_copy["SkinThickness"].mean()*

*std = data\_copy["SkinThickness"].std()*

*is\_null = data\_copy["SkinThickness"].isnull().sum()*

*rand = np.random.randint(mean-std,mean+std,size=is\_null)*

*st = data\_copy["SkinThickness"].copy()*

*st[np.isnan(st)] = rand*

*data\_copy["SkinThickness"] = st*

*data\_copy.info()*

*#Pravimo izmene tako što popunjavamo vrednost NaN za BMI sa*

*#slučajnim brojevima između srednje vrednosti plus minus standardna devijacija*

*for data in data\_copy:*

*mean = data\_copy["BMI"].mean()*

*std = data\_copy["BMI"].std()*

*is\_null = data\_copy["BMI"].isnull().sum()*

*rand = np.random.randint(mean-std,mean+std,size=is\_null)*

*bmi = data\_copy["BMI"].copy()*

*bmi[np.isnan(bmi)] = rand*

*data\_copy["BMI"] = bmi*

*data\_copy.info()*

*#Prikazujemo korelaciju naših podataka grafički*

*pd.plotting.scatter\_matrix(data\_copy.drop(columns=['Outcome']),figsize=(20,20))*

*plt.show()*

*#Prikazujemo histogram podataka i ishoda*

*data\_copy.hist(figsize=(12,12))*

*plt.show()*

*#vršimo odvajanje dataseta na set podataka za obučavanje i set za testianje*

*#NaN problem*

*npSkup = data\_copy.values*

*imp = SimpleImputer(missing\_values=np.nan, strategy='median')*

*tfSkup = imp.fit\_transform(npSkup)*

*#Koristicemo originalni dataset koji nema sredjene podatke posto se pokazao kao uspesniji u odlucivanju*

*X = dataset.drop(columns=['Outcome'],axis=1) #odvajamo podatke od njihovog konačnog ishoda*

*Y = dataset['Outcome'] #odvajamo konačne ishode od podataka*

*X\_train,X\_test,y\_train,y\_test = train\_test\_split(X,Y,test\_size=0.3,random\_state=101)*

*#dodajemo niz modela koje ćemo testirati*

*models = []*

*#Linearna Regresija*

*models.append(('LR', LogisticRegression(solver='liblinear',multi\_class='ovr',max\_iter=200)))*

*#K Nearest Neighbors*

*models.append(('KNN', KNeighborsClassifier()))*

*#Naive Bayes*

*models.append(('NB', GaussianNB()))*

*#Support Vector Machine*

*models.append(('SVM', SVC(C=0.5, kernel='linear')))*

*#Stablo odlučivanja*

*models.append(('CART', DecisionTreeClassifier()))*

*#Linear Discriminant Analysis*

*models.append(('LDA', LinearDiscriminantAnalysis()))*

*#vršimo obučavanje svih algoritama primenom kros-validacije*

*#Prikazujemo rezutate obučavanja - prosečnu tačnost i standardnu devijaciju*

*results = []*

*names = []*

*for name, model in models:*

*kfold = StratifiedKFold(n\_splits=10, random\_state=1, shuffle=True)*

*cv\_results = cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring='accuracy')*

*results.append(cv\_results)*

*names.append(name)*

*print('%s: %f (%f)' % (name, cv\_results.mean(), cv\_results.std()))*

*warnings.filterwarnings("ignore")*

*#Uporedni prikaz algoritama(box) i prikaz prolazaka kroz krosvalidaciju(dots)*

*ax = sns.boxplot(data=results)*

*ax = sns.swarmplot(data=results, color = "black")*

*plt.xticks(np.arange(0,6),names)*

*plt.title('Poredjenje algoritama')*

*plt.show()*

*#U prethodnom mozemo primetiti da su LDA i SVM algoritam priblizno jednaki u uspesnosti ali iz matrica konfuzije*

*#mozemo zakljuciti da se LDA bolje pokazao u otkrivanju Dijabetesa*

*#Zbog jednostavnosti algoritma i brzine obucavanja izabracemo LDA algoritam za evaluaciju naseg modela*

*model = LinearDiscriminantAnalysis()*

*model.fit(X\_train,y\_train)*

*predictions = model.predict(X\_test)*

*print(classification\_report(y\_test,predictions))*

*print(confusion\_matrix(y\_test,predictions))*

*print(accuracy\_score(y\_test,predictions))*

*model = SVC()*

*model.fit(X\_train,y\_train)*

*predictions = model.predict(X\_test)*

*print(classification\_report(y\_test,predictions))*

*print(confusion\_matrix(y\_test,predictions))*

*print(accuracy\_score(y\_test,predictions))*

**Reference**

<https://www.beo-lab.rs/dijabetes-simptomi-dijagnostika-prevencija-i-kontrola-secerne-bolesti/>

<https://rs.n1info.com/zdravlje/a552292-zasto-je-dijabetes-opasniji-za-zene-nego-za-muskarce/>

<https://www.kaggle.com/datasets/mathchi/diabetes-data-set>

Veštačka inteligencija - priručnik za laboratorijske vežbe, Emilija Kisić