Predikcija bolesti srca

Vasiljevic Velibor
Mentor: Igor Jovin

Motivacija

- U americi na svakih 37 sekundi 1 osoba umre od srcanih bolesti
- Svake godine 647 000, na svake 4 smrti jedna je od srcanih bolesti
- Svakih 40 sekundi neko ima srcani udar
- 1 u 5 srcanih udara su “tihi”, steta je naneta ali mi to ne osetimo
- Najcesci razlozi: dijabetes, gojaznost, nezdrava ishrana, fizicka neaktivnost, preterana upotreba alkohola

Dataset 1 - Cleveland

age

sex (1 = male; 0 = female)

cp: chest pain type (1: typical angina, 2: atypical angina, 3: non-anginal pain, 4: asymptomatic)

trestbps: resting blood pressure (in mm Hg on admission to the hospital)

mochol: serum cholestoral in mg/dl

years (number of years as a smoker)

fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)

restecg: resting electrocardiographic results (0: normal, 1: having ST-T wave abnormality, 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

thalach: maximum heart rate achieved

exang: exercise induced angina (1: yes, 0: no)

oldpeak = ST depression induced by exercise relative to rest

slope: the slope of the peak exercise ST segment (1: upsloping, 2: flat, 3: downsloping ldv5: height at rest)

ca: number of major vessels (0-3) colored by flourosopy

thal: 3: normal, 6, fixed defect, 7 reversable defect

num: diagnosis of heart disease (0: < 50% diameter narrowing, 1: > 50% diameter narrowing)

Dataset 2 - Framingham

age

male(0 – zensko, 1 - musko)

education

currentSmoker (0 – nepusac, 1 - pusac),

cigsPerDay, BPMeds (lekovi za visok pritisak, 0 – ne uzima, 1 - uzima),

prevalentStroke (da li je osoba do sad imala srcani udar 0/1),

prevalentHyp (da li je osoba do sad imala visoki krvni pritisak 0/1),

diabetes (0 – nema, 1 - ima), totChol (nivo holesterola u krvi mg/dL),

sysBP(systolic? mmHg),

diaBP(diastolic? mmHg),

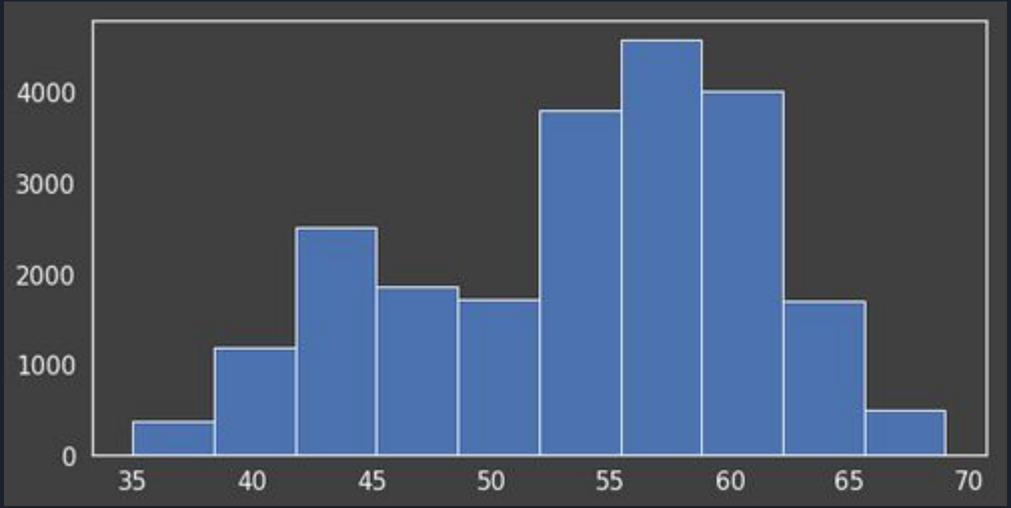
BMI ("Body Mass Index" = tezina/visina),

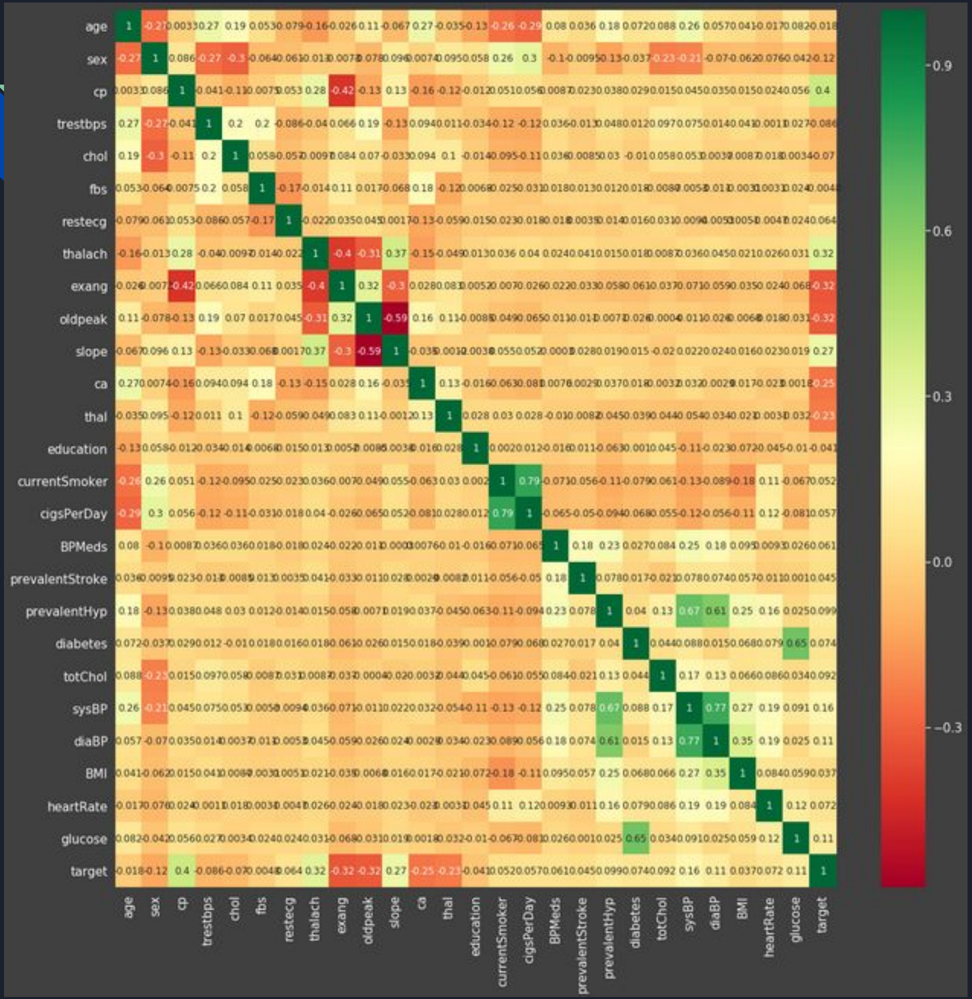
heartRate (beats/min)

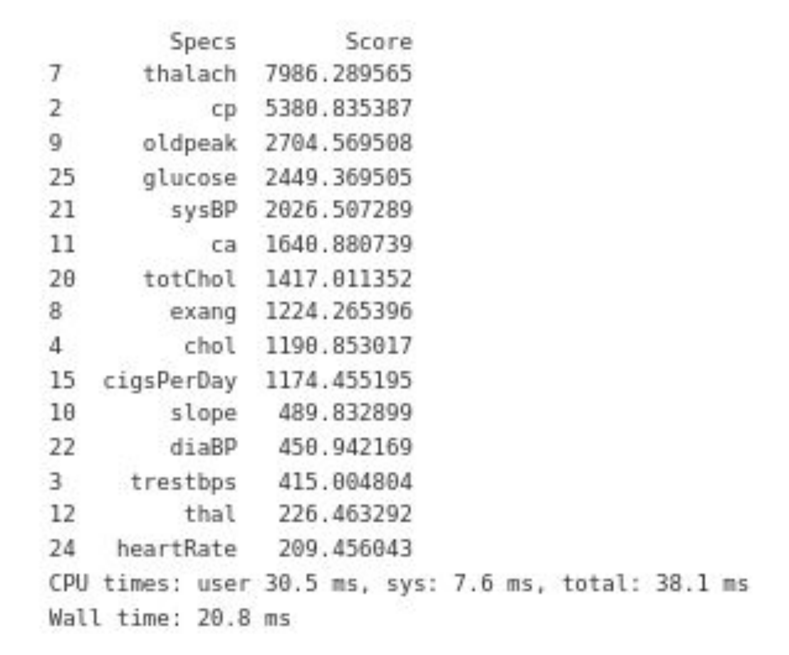
glucose (mg/dL)

TenYearCHD (target)

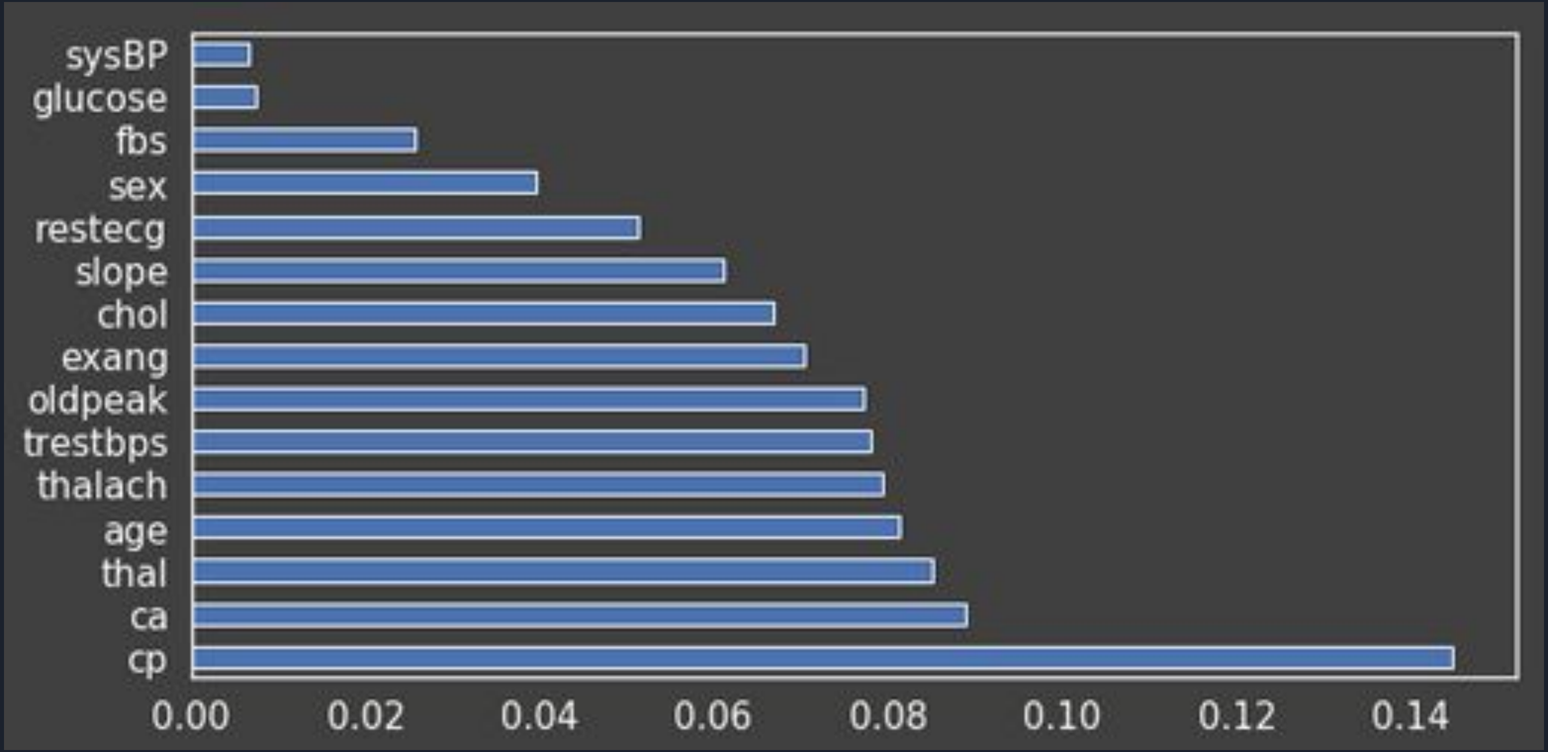
Broj ljudi/godine

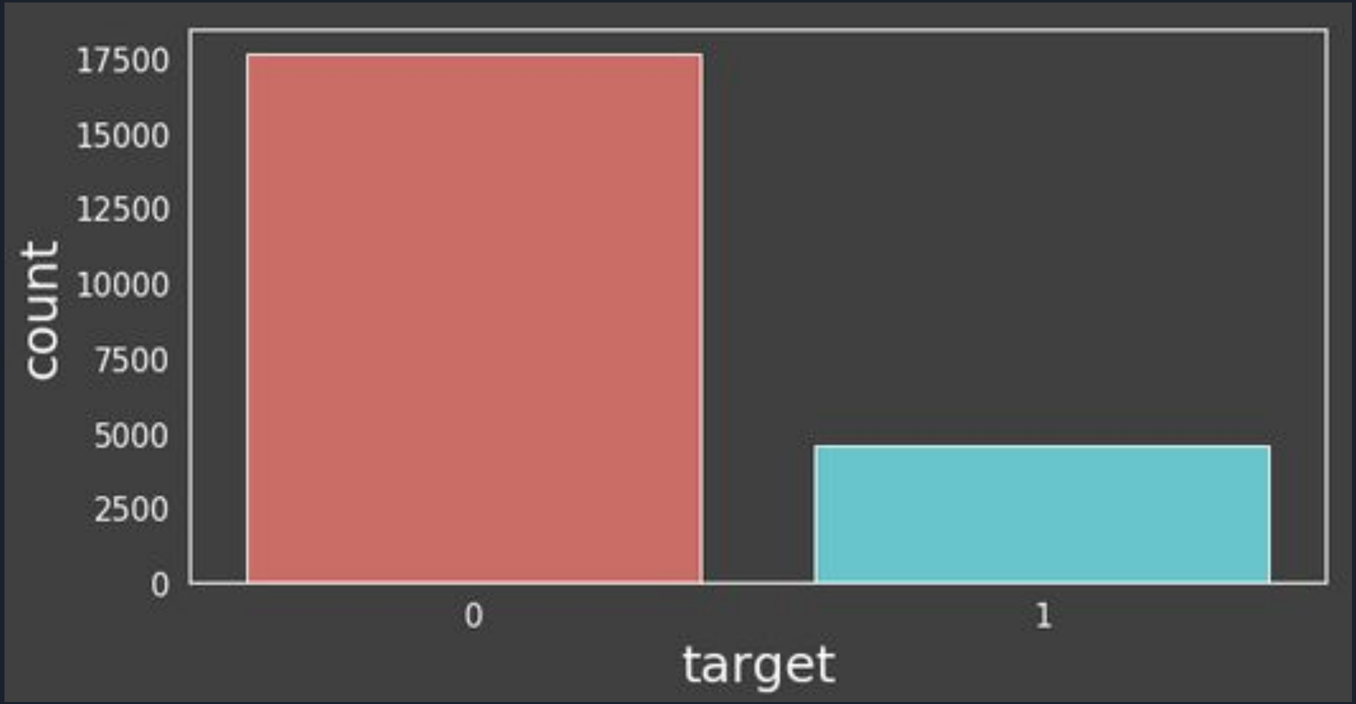




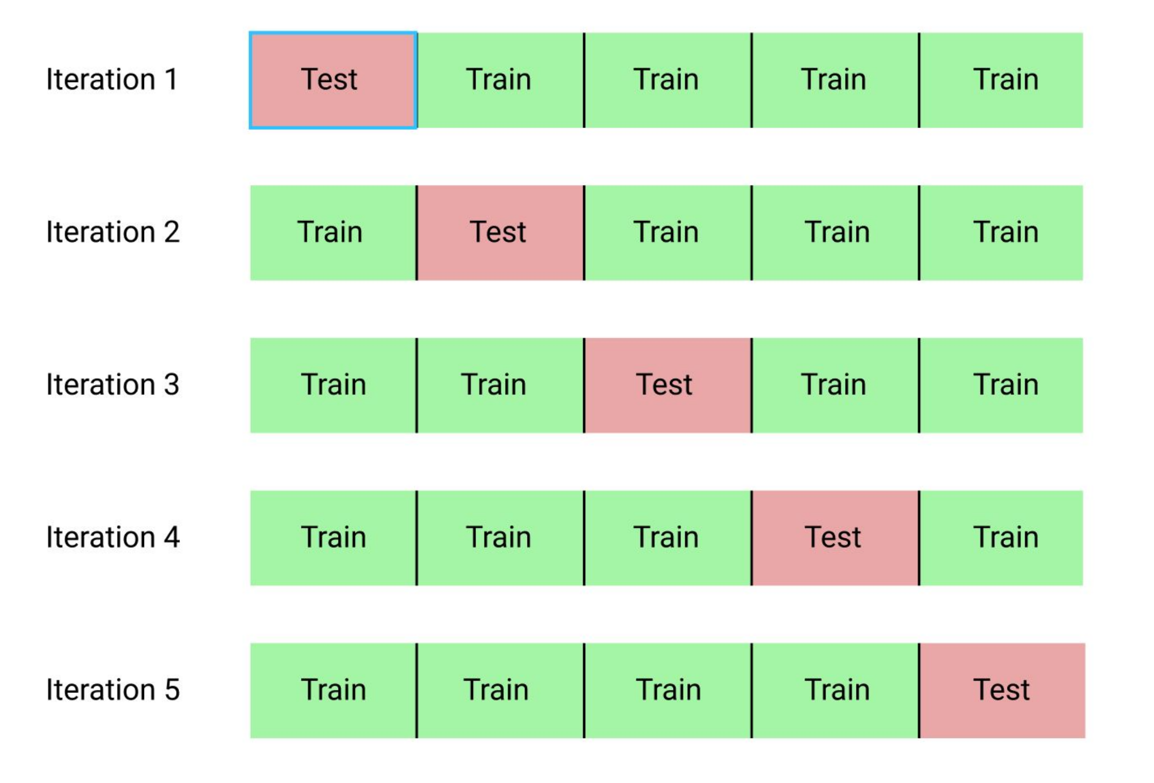


```
              Specs        Score
7           thalach  7986.289565
2                cp  5380.835387
9           oldpeak  2704.569508
25          glucose  2449.369505
21            sysBP  2026.507289
11               ca  1640.880739
20          totChol  1417.011352
8             exang  1224.265396
4              chol  1190.853017
15       cigsPerDay  1174.455195
10            slope   489.832899
22            diaBP   450.942169
3          trestbps   415.004804
12             thal   226.463292
24        heartRate   209.456043
CPU times: user 30.5 ms, sys: 7.6 ms, total: 38.1 ms
Wall time: 20.8 ms
```


sysBP
glucose
fbs
sex
restecg
slope
chol
exang
oldpeak
trestbps
thalach
age
thal
ca
cp
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14

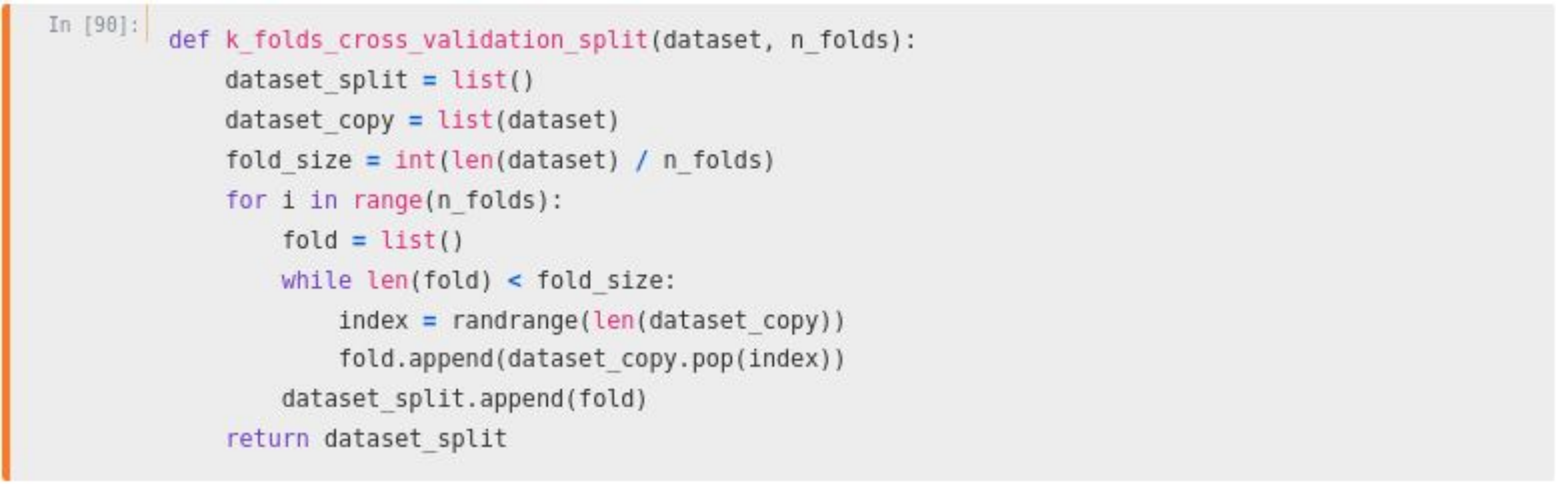

17500
15000
12500
10000
7500
5000
2500
0
count
0
1
target

K folds cross validation split



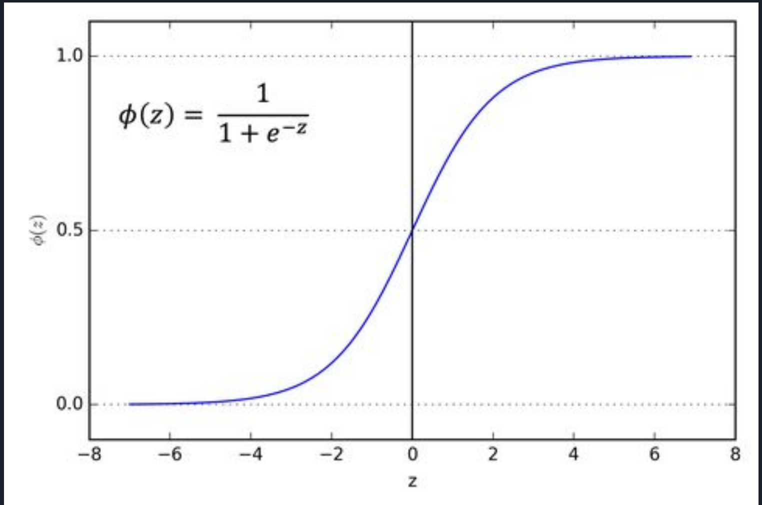
Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

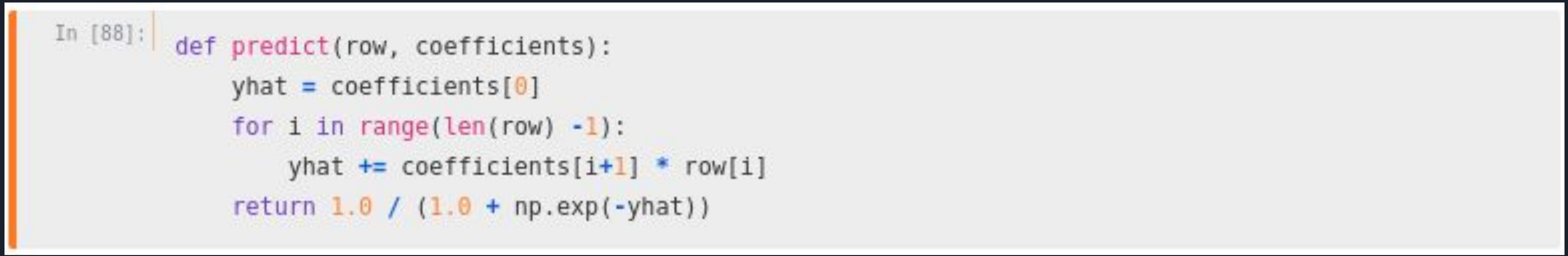
Implementation



```
def k_folds_cross_validation_split(dataset, n_folds):
    dataset_split = list()
    dataset_copy = list(dataset)
    fold_size = int(len(dataset) / n_folds)
    for i in range(n_folds):
        fold = list()
        while len(fold) < fold_size:
            index = randrange(len(dataset_copy))
            fold.append(dataset_copy.pop(index))
        dataset_split.append(fold)
    return dataset_split
```

Sigmoid



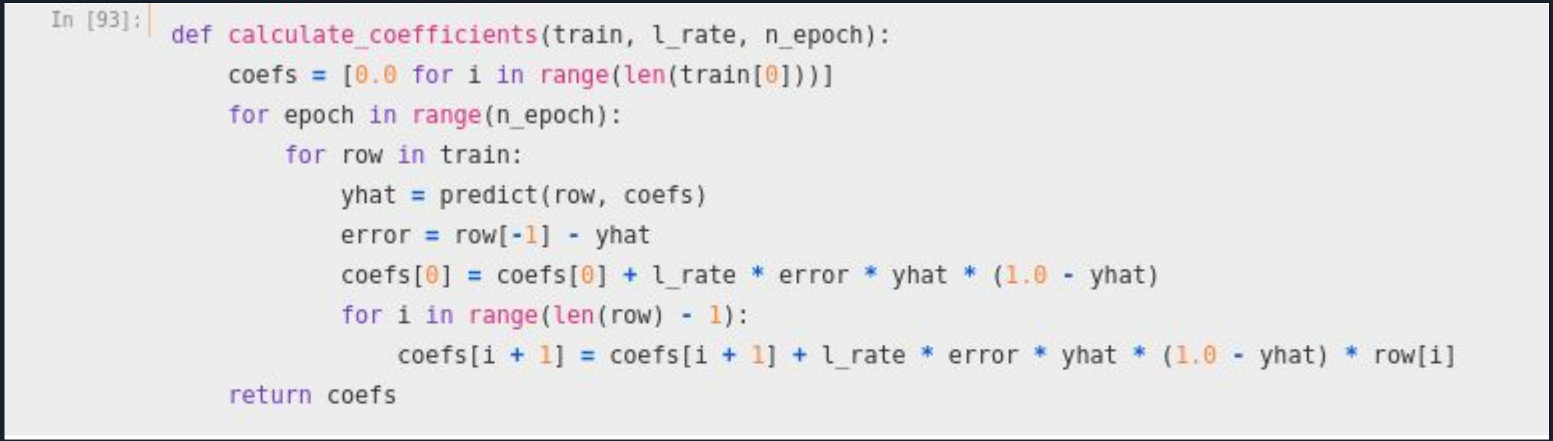


```
def predict(row, coefficients):
    yhat = coefficients[0]
    for i in range(len(row) -1):
        yhat += coefficients[i+1] * row[i]
    return 1.0 / (1.0 + np.exp(-yhat))
```

Koeficijenti

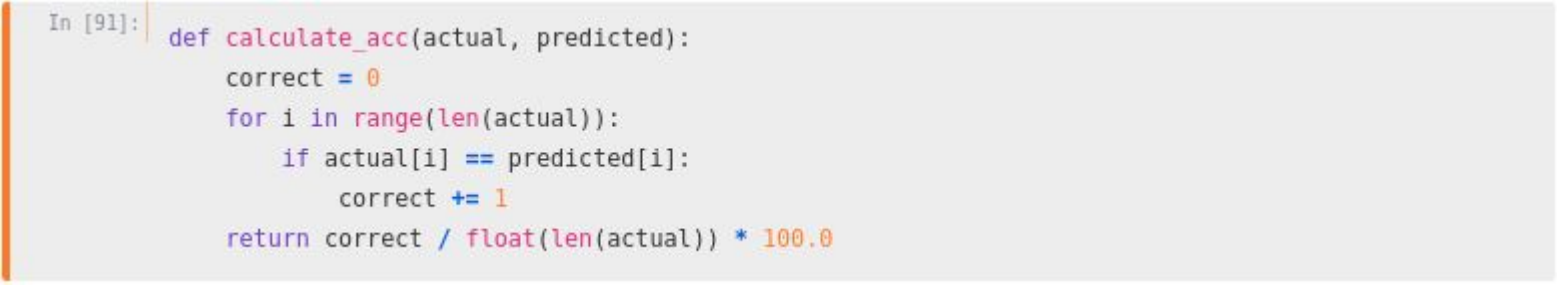
$$b_0^{t+1} = b_0^{t+1} + \alpha * ((y^t - \hat{y}^t) * \hat{y}^t * (1 - \hat{y}^t))$$

$$b_1^{t+1} = b_1^{t+1} + \alpha * ((y^t - \hat{y}^t) * \hat{y}^t * (1 - \hat{y}^t)) * x_1^t$$



```
In [93]: def calculate_coefficients(train, l_rate, n_epoch):
             coefs = [0.0 for i in range(len(train[0]))]
             for epoch in range(n_epoch):
                 for row in train:
                     yhat = predict(row, coefs)
                     error = row[-1] - yhat
                     coefs[0] = coefs[0] + l_rate * error * yhat * (1.0 - yhat)
                     for i in range(len(row) - 1):
                         coefs[i + 1] = coefs[i + 1] + l_rate * error * yhat * (1.0 - yhat) * row[i]
             return coefs
```

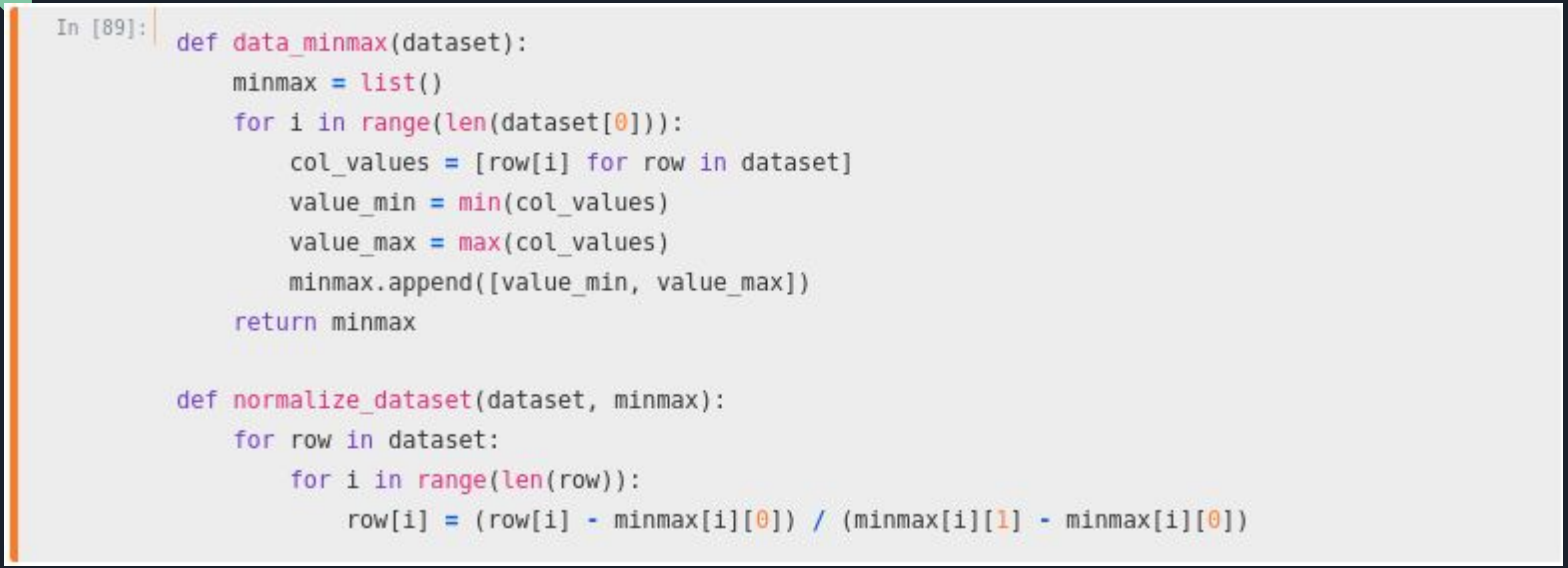
Racunanje tacnosti



In [91]:

```
def calculate_acc(actual, predicted):
    correct = 0
    for i in range(len(actual)):
        if actual[i] == predicted[i]:
            correct += 1
    return correct / float(len(actual)) * 100.0
```

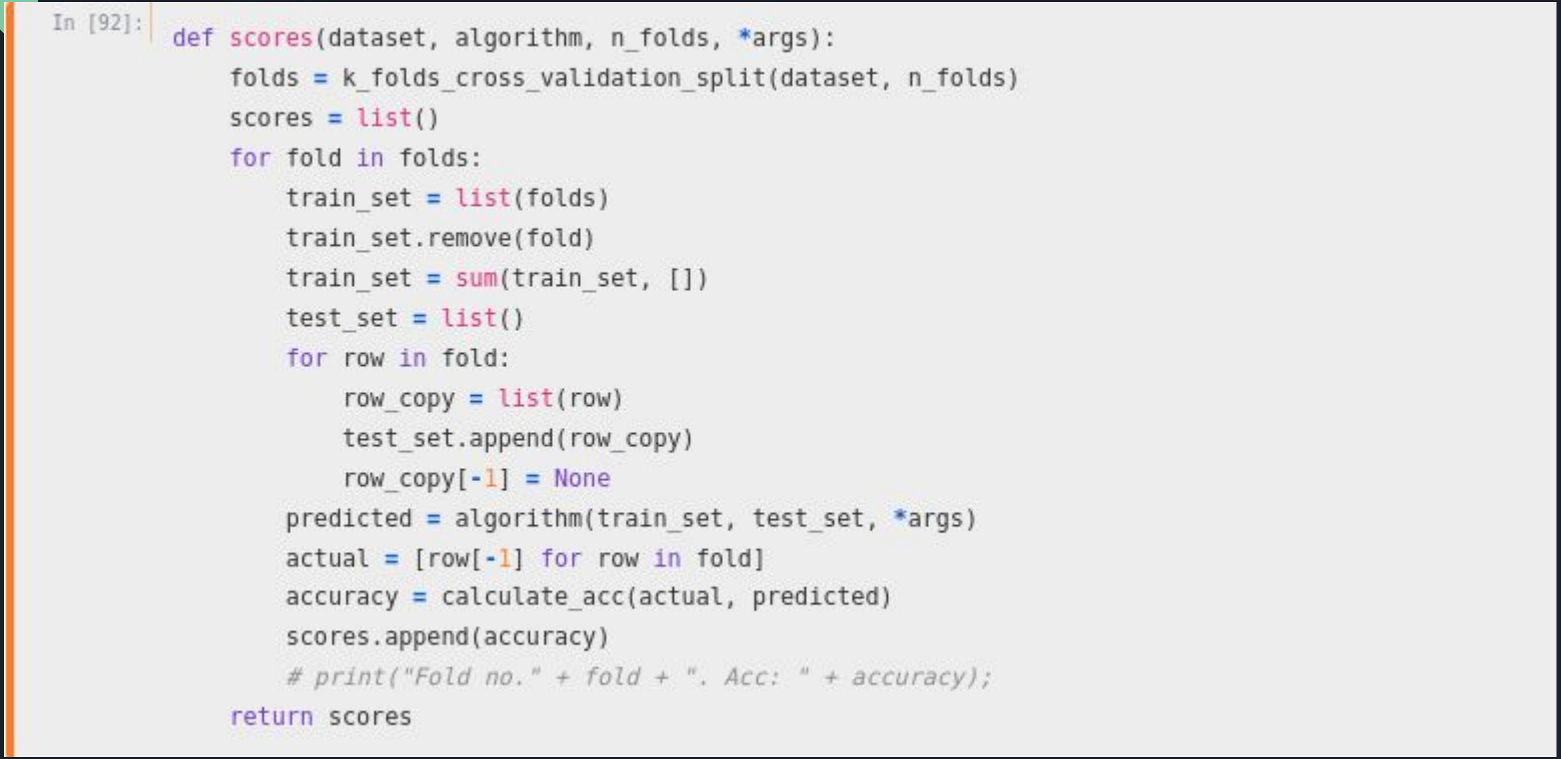
Normalizovanje podataka



```python
def data_minmax(dataset):
    minmax = list()
    for i in range(len(dataset[0])):
        col_values = [row[i] for row in dataset]
        value_min = min(col_values)
        value_max = max(col_values)
        minmax.append([value_min, value_max])
    return minmax

def normalize_dataset(dataset, minmax):
    for row in dataset:
        for i in range(len(row)):
            row[i] = (row[i] - minmax[i][0]) / (minmax[i][1] - minmax[i][0])
```

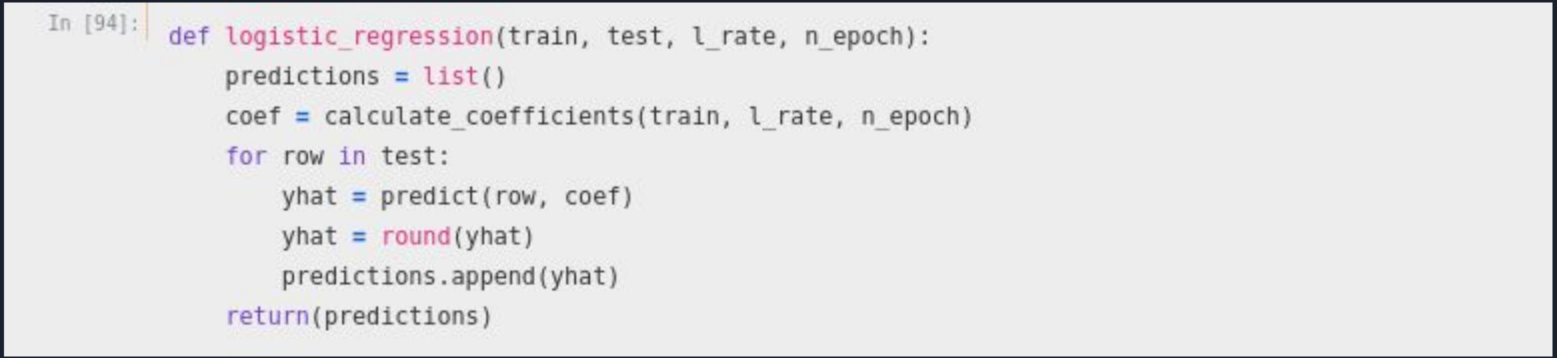
Rezultati



In [92]:

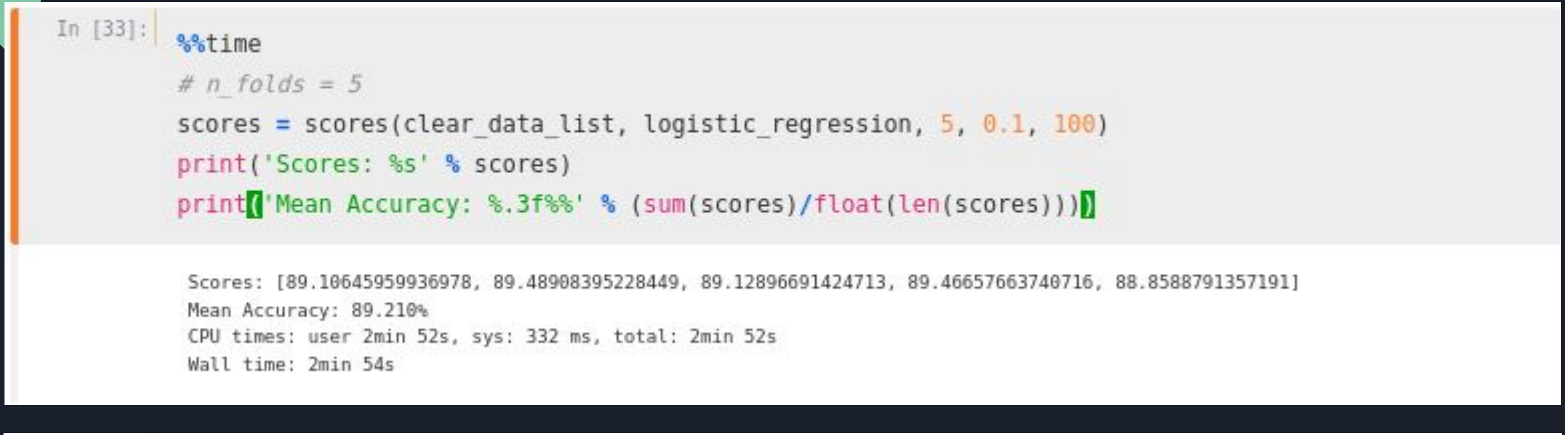
```
def scores(dataset, algorithm, n_folds, *args):
    folds = k_folds_cross_validation_split(dataset, n_folds)
    scores = list()
    for fold in folds:
        train_set = list(folds)
        train_set.remove(fold)
        train_set = sum(train_set, [])
        test_set = list()
        for row in fold:
            row_copy = list(row)
            test_set.append(row_copy)
            row_copy[-1] = None
        predicted = algorithm(train_set, test_set, *args)
        actual = [row[-1] for row in fold]
        accuracy = calculate_acc(actual, predicted)
        scores.append(accuracy)
        # print("Fold no." + fold + ". Acc: " + accuracy);
    return scores
```

Logisticka regresija



```python
def logistic_regression(train, test, l_rate, n_epoch):
    predictions = list()
    coef = calculate_coefficients(train, l_rate, n_epoch)
    for row in test:
        yhat = predict(row, coef)
        yhat = round(yhat)
        predictions.append(yhat)
    return(predictions)
```

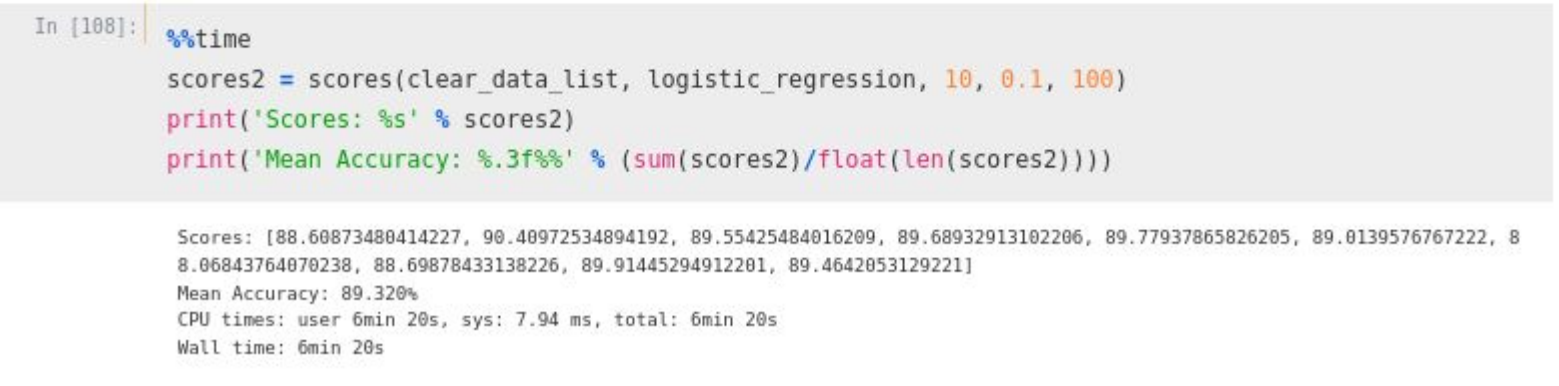
Konacni rezultati



In [33]:

```
%%time
# n_folds = 5
scores = scores(clear_data_list, logistic_regression, 5, 0.1, 100)
print('Scores: %s' % scores)
print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
```

```
Scores: [89.10645959936978, 89.48908395228449, 89.12896691424713, 89.46657663740716, 88.8588791357191]
Mean Accuracy: 89.210%
CPU times: user 2min 52s, sys: 332 ms, total: 2min 52s
Wall time: 2min 54s
```

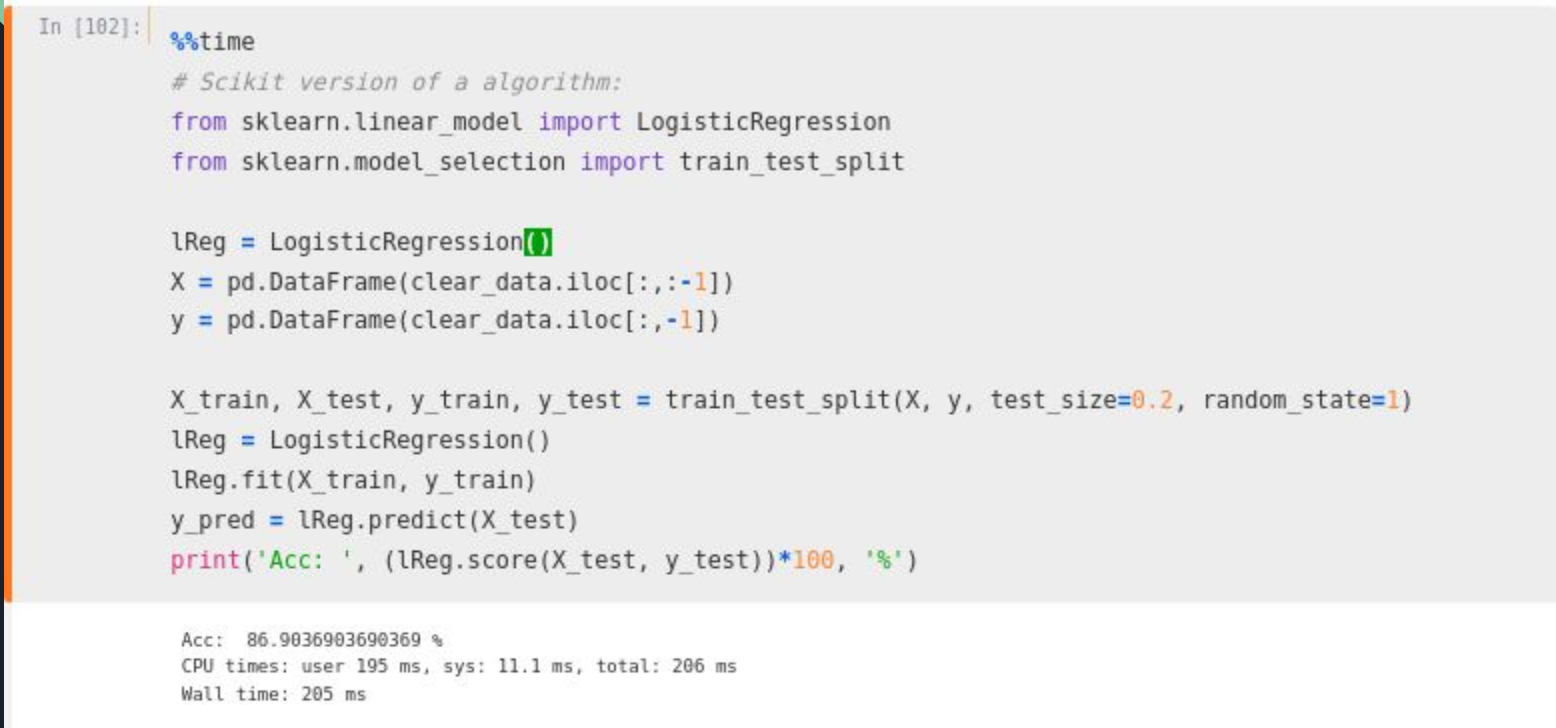


In [108]:

```
%%time
scores2 = scores(clear_data_list, logistic_regression, 10, 0.1, 100)
print('Scores: %s' % scores2)
print('Mean Accuracy: %.3f%%' % (sum(scores2)/float(len(scores2))))
```

```
Scores: [88.60873480414227, 90.40972534894192, 89.55425484016209, 89.68932913102206, 89.77937865826205, 89.0139576767222, 8
8.06843764070238, 88.69878433138226, 89.91445294912201, 89.4642053129221]
Mean Accuracy: 89.320%
CPU times: user 6min 20s, sys: 7.94 ms, total: 6min 20s
Wall time: 6min 20s
```

Poredjenje sa skleanovom gotovom f-jom



In [102]:

```
%%time
# Scikit version of a algorithm:
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

lReg = LogisticRegression()
X = pd.DataFrame(clear_data.iloc[:,:-1])
y = pd.DataFrame(clear_data.iloc[:,-1])

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
lReg = LogisticRegression()
lReg.fit(X_train, y_train)
y_pred = lReg.predict(X_test)
print('Acc: ', (lReg.score(X_test, y_test))*100, '%')
```

```
Acc:  86.9036903690369 %
CPU times: user 195 ms, sys: 11.1 ms, total: 206 ms
Wall time: 205 ms
```

Hvala na paznji!

Reference

1. https://intellipaat.com/blog/what-is-logistic-regression/
2. https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/
3. https://machinelearningmastery.com/gradient-descent-for-machine-learning/
4. https://machinelearningmastery.com/k-fold-cross-validation/
5. https://www.geeksforgeeks.org/cross-validation-machine-learning/
6. https://en.wikipedia.org/wiki/Sigmoid_function
7. https://www.coursera.org/learn/machine-learning
8. http://aima.cs.berkeley.edu/
9. https://machinelearningmastery.com/logistic-regression-for-machine-learning/
10. https://machinelearningmastery.com/logistic-regression

Datasets

1. https://www.kaggle.com/ronitf/heart-disease-uci
2. https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset