# 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

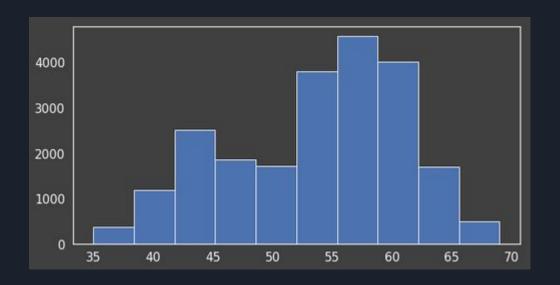
thal: 3: normal, 6, fixed defect, 7 reversable defect

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

#### Dataset 2 - Framingham

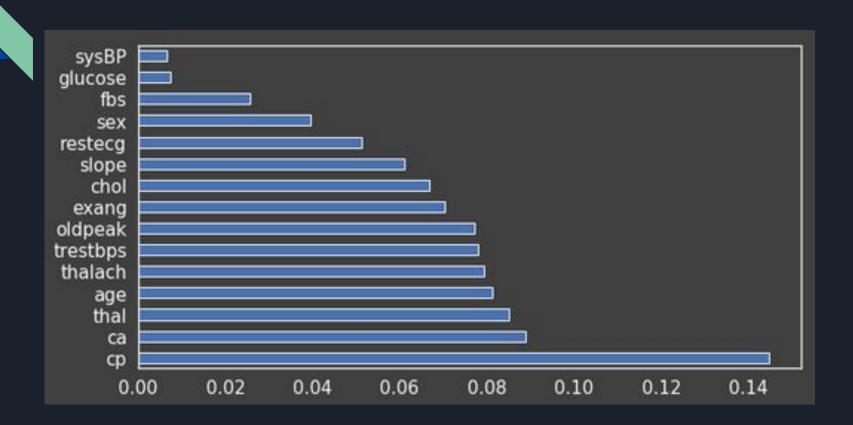
```
age
male(0 - zensko, 1 - musko)
education
currentSmoker (0 - nepusac, 1 - pusac),
cigsPerDay, BPMeds (lekovi za visok pritisak, 0 – ne uzima, 1 - uzima),
prevalentStroke (da li je osoba do sad imala srcani udar 0/1),
prevalentHyp (da li je osoba do sad imala visoki krvni pritisak 0/1),
diabetes (0 – 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)
```

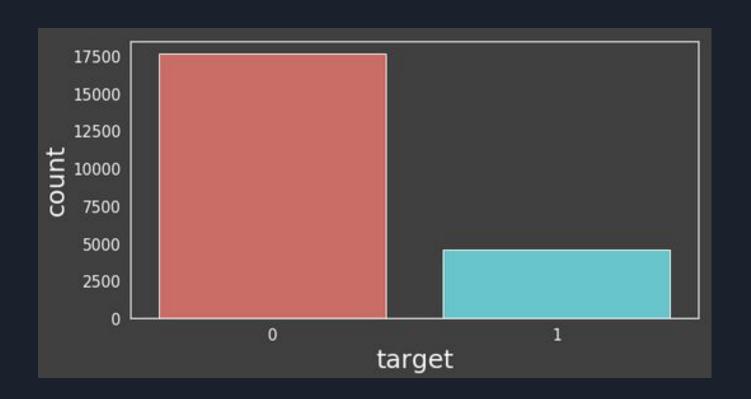
# Broj ljudi/godine



```
00033027 0.19 0.0530 0790 160.0260 11-0.067 0.27-0.0350 13 0 70 0 PC 0.08 0.036 0.18 0.0720 088 0.26 0.0570 0410.0170 0820 018
                             086 0 27 03 0 0640 0630 0130 00740 0760 0940 00740 0950 058 0 26 0 3 0 10 0 00950 13 0 0 37 0 23 0 21 0 0 7 0 0620 0760 042 0 12
                                 0.041-0.130 00730 053 0.28 0.42 0.13 0.13 0.16 0.12-0.0120.0510.0540.00870.0230.0380.0290.0150.0450.0350.0150.0240.056 0.4
                                     02 02 0.086-0.040.066.019 0.150.0940.0110.0340.12 0.120.0360.0130.0480.0120.0970.0750.0140.0430.00130.0270.086
       trestbps
                                       1 00580 0570 00970 084 0.07-0.0330 094 0.1 -0.0140 095-0.110 0360 00850 03 -0.010 0580 0570 00370 00870 0180 00340 07
             fbs: 0.0530.066.0075.0.2 0.058 1 0.17-0.0140.11 0.0170.068.018 0.120.0068.0250.0310.0180.0130.0120.0180.0080.0050.0110.00310.0310.0240.004
        restect 0 0790 0610 0530 0860 057-0.17 1 0 0220 0350 0430 00170 13-0 0590 0150 0230 0180 0130 0140 0160 0310 0090 0030 0050 00430 0240 064
                   0.160.0130.28 -0.040.00970.0140.02 1 -0.4 -0.31 0.37 -0.15-0.0490.0130.036 0.04 0.0240.0410.0150.0180.00870.0360.0450.0210.0260.031 0.32
                   0.026.007 0.42 0.0660.084 0.11 0.035 0.4 1 0.32 0.3 0.0280.0830.00520.0070.0260.0220.0330.0580.0610.0370.0710.0590.0350.0240.068
                   011 0 0780 13 0 19 0 07 0 017 0 045 0 11 0 32 1 0 59 0 16 0 110 0080 0490 0650 0130 010 0070 0260 0040 0130 020 0060 0150 031 06
                    0670 096 0.13 0 130.0330.068 00170 37 015 0.59 1 0.035.0010.0038 0550 0520 000 B.0280 0190 015 0.02 0.020 0220 0240 0160 0230 019 0.27
                   0.270.00740.160.0940.094.0.18 0.13 0.15 0.028.0.16 0.035 1 0.13 0.0160.0630.088.0076.00290.0370.0180.00320.0320.0029.0170.028.0018
                   0.0350.095-0.120.011-0.1-0.12-0.0590.0490.083.0.11-0.00120.13-0.140.028.0.03.0.028-0.030.00820.0450.0390.0440.0540.0340.0210.00310.032
                   0 130 0580 0120 0340 014 00680 0150 0130 005# 0085 003#0.0160 028 1 0 0020 0120 0160 0110 0630 0010 045-0 110 0230 0720 045-0 01-0 041
                      0.26 0.051 0.12 0.095 0.025 0.025 0.030 0.070 0.490 0.55 0.063 0.03 0.002 1 0.79 0.0710 0.56 0.11 0.079 0.061 0.13 0.089 0.18 0.11 0.067 0.052
currentSmoker
                     0 3 0 0 5 6 0 12 0 11 0 0 3 10 0 18 0 0 4 0 0 2 6 0 0 5 2 0 5 2 0 0 8 10 0 2 8 0 0 1 2 0 7 2 1 0 0 5 5 0 0 9 4 0 0 6 8 0 0 5 5 0 1 2 0 0 5 6 0 1 1 0 1 2 0 0 8 10 0 5
    cigsPerDay
                    08 -0 10 00870 0360 0360 0180 0180 0240 0220 0130 000300760 01-0 0160 0710 065 1 0.18 0.23 0.0270 084 0.25 0.18 0.095 0095 0095 0060 061
                    0360 0098 0230 01-0 0088 0130 00350 0410 0330 0110 0260 0026 0088 0110 056-0 05 0 18 1 1 1 0 0780 0170 0230 0780 0740 0570 0110 0010 045
                   018 -0 13 0 038 0 048 0 03 0 012 0 0140 0150 05@ 007 0 0190 0370 0450 0630 11-0 094 0 23 0 078 1 0 0 0 4 0 13 007 0 61 0 25 0 16 0 0250 099
                   0.088 0 > 0.0150.0970.0580.00810.0310.00870.0370.00040.020.00320.0440.045-0.0610.0550.0840.021.0.13.0.044
                        057-0 070 0350 0140 00370 0130 0058 0450 0590 0260 0240 00280 0340 0230 0890 0560 18 0 074 061 0 015 0 13 0 77
                 0.041-0.0620.0150.0430.0080.0030.00530.0210.0330.0068.0160.0170.0210.0720.18-0.110.0950.057.0.25.0.0680.066.0.27.0.35
     heartRate: 0.0170.0760.0240.00180.00380.0048.0260.0240.0180.0230.0230.0038.0039.0045.011.0120.00980.011.016.0.0790.086.0.19.0.19.0.084 1 0.12.0.072
        ciurose 3 0820 0420 0560 0270 00340 0240 0240 0240 0310 0680 0310 0140 00180 0320 01-0 0670 0810 0260 0010 025 065 0 0340 0910 0250 059 0 12
          target 0 018 0 12 0 4 0 086 0 070 004 8 064 0 32 0 52 0 52 0 37 0 7 0 7 0 0 40 0520 0570 0610 0450 0990 0740 092 0 16 0 11 0 0370 072 0 11
```

```
Specs
                      Score
                7986.289565
       thalach
                5380.835387
                2704.569508
       oldpeak
       glucose
                2449.369505
         sysBP
                2026.507289
                1640.880739
                1417.011352
       totChol
                1224.265396
         exang
                1190.853017
          chol
    cigsPerDay
               1174.455195
                 489.832899
         slope
         diaBP
                 450.942169
     trestbps
                 415.004804
                 226,463292
          thal
     heartRate
                 209.456043
CPU times: user 30.5 ms, sys: 7.6 ms, total: 38.1 ms
Wall time: 20.8 ms
```





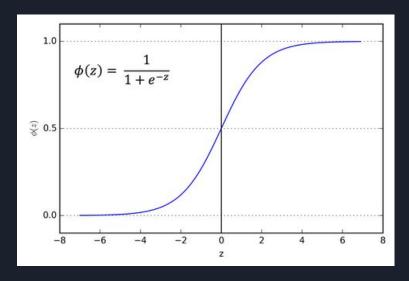
# K folds cross validation split

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

```
In [90]: 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</pre>
```

## Sigmoid



## Koeficijenti

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

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

## Racunanje tacnosti

## Normalizovanje podataka

```
In [89]:
        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

```
In [94]: 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 \text{ 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)
         v pred = lReq.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. <a href="https://intellipaat.com/blog/what-is-logistic-regression/">https://intellipaat.com/blog/what-is-logistic-regression/</a>
- 2. https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd/
- 3. <a href="https://machinelearningmastery.com/gradient-descent-for-machine-learning/">https://machinelearningmastery.com/gradient-descent-for-machine-learning/</a>
- 4. <a href="https://machinelearningmastery.com/k-fold-cross-validation/">https://machinelearningmastery.com/k-fold-cross-validation/</a>
- 5. <a href="https://www.geeksforgeeks.org/cross-validation-machine-learning/">https://www.geeksforgeeks.org/cross-validation-machine-learning/</a>
- 6. <a href="https://en.wikipedia.org/wiki/Sigmoid\_function">https://en.wikipedia.org/wiki/Sigmoid\_function</a>
- 7. <a href="https://www.coursera.org/learn/machine-learning">https://www.coursera.org/learn/machine-learning</a>
- 8. <a href="http://aima.cs.berkeley.edu/">http://aima.cs.berkeley.edu/</a>
- 9. <a href="https://machinelearningmastery.com/logistic-regression-for-machine-learning/">https://machinelearningmastery.com/logistic-regression-for-machine-learning/</a>
- 10. <a href="https://machinelearningmastery.com/logistic-regression">https://machinelearningmastery.com/logistic-regression</a>

#### Datasets

- 1. <a href="https://www.kaggle.com/ronitf/heart-disease-uci">https://www.kaggle.com/ronitf/heart-disease-uci</a>
- 2. <a href="https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset">https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset</a>