Logistička regresija

Najkorišćenija biblioteka u oblasti mašinskog učenja je scikit-learn, koja sadrži mnoštvo funkcija za rad sa različitim metodama mašinskog učenja.

In [4]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

U primeru koji sledi model logističke regresije biće iskorišćen za klasifikaciji tumora na maligne i benigne. Koristićemo Viskonsis skup podataka.

Sledeći kod je većinski preuzet od asistenata Anđelke Zečević i Milana Čugurovića sa časova vežbi iz Mašinskog učenja.

In [5]:
```
# !pip install sklearn
from sklearn import linear_model
from sklearn import model_selection
from sklearn import metrics
from sklearn import preprocessing
from sklearn import datasets
```

In [6]:
```
data = datasets.load_breast_cancer()
```

Svaka instanca skupa opisana je sa 30 različitih atributa.

In [7]:
```
data.feature_names
```

Out[7]:
```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

Prebacujemo podatke u oblik dataframe takav da možemo da koristimo funkcije iz biblioteke pandas.

In [9]:
```
X = pd.DataFrame(data.data, columns = data.feature_names)
```

In [10]:
```
X.shape
```

Out[10]:
```
(569, 30)
```

In [11]:
```
data.target_names
```

Out[11]:
```
array(['malignant', 'benign'], dtype='<U9')
```

In [13]:
```
y = data.target
```

In [14]:
```
print('Broj benignih instanci: ')
np.sum(y==1)
```

```
Broj benignih instanci:
```

Out[14]:
```
357
```

In [15]:
```
print('Broj malignih instanci: ')
np.sum(y==0)
```

```
Broj malignih instanci:
```

Out[15]:
```
212
```

Prilikom podele skupa podataka na skupove za treniranje i testiranje, vodićemo računa o stratifikaciji (parametar stratify). Stratifikacija je način podele podataka kojim se čuva distribucija klasa. Podatke delimo u odnosu 2:1.

In [17]:
```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, te
random_state = 7, stratify = y)
```

Provera stratifikovanosti:

In [19]:
```
m_train = np.sum(y_train == 0)
b_train = np.sum(y_train == 1)
print('Benigni:', b_train, 'Maligni:', m_train)
```

```
Benigni: 239 Maligni: 142
```

In [20]:
```
m_test = np.sum(y_test == 0)
b_test = np.sum(y_test == 1)
print('Benigni:', b_test, 'Maligni:', m_test)
```

```
Benigni: 118 Maligni: 70
```

Standardizacija podataka

In [21]:
```
scaler = preprocessing.StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Pravljenje modela

In [22]:
```
model = linear_model.LogisticRegression(solver='lbfgs')
```

In [23]:
```
model.fit(X_train, y_train)
```

Out[23]:
```
LogisticRegression()
```

Rezultujući parametri se mogu pročitati kroz intercept_ i coef_ svojstva.

In [24]:
```
model.intercept_
```

Out[24]:
```
array([0.33070776])
```

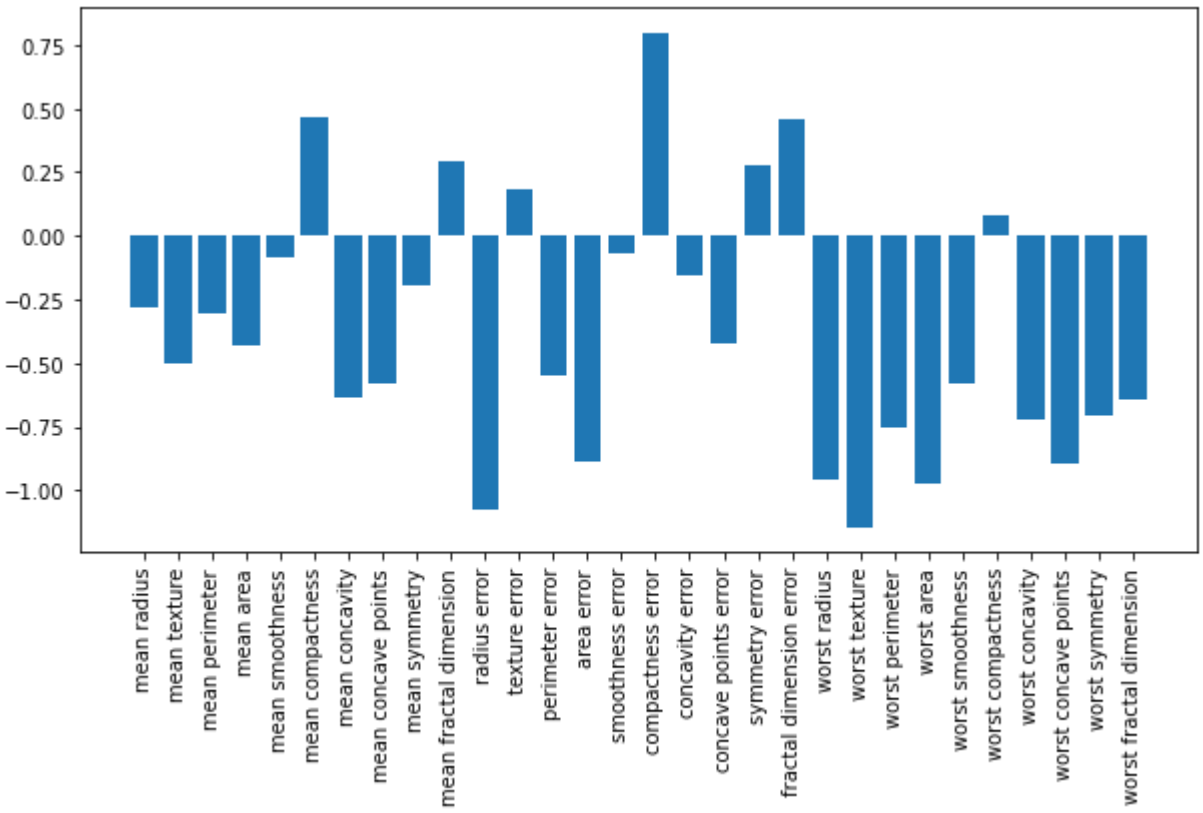
In [25]:
```
model.coef_
```

Out[25]:
```
array([[-0.28488259, -0.5036704 , -0.30845374, -0.43124456, -0.0885936 ,
         0.46530026, -0.63431362, -0.5799997 , -0.19622748,  0.29418451,
        -1.0734335 ,  0.17985058, -0.551678  , -0.88684956, -0.06873348,
         0.79862352, -0.1564331 , -0.41965439,  0.27938466,  0.45749103,
        -0.95434897, -1.14556574, -0.75325776, -0.975012  , -0.58396992,
         0.08148169, -0.72493983, -0.89869581, -0.70564949, -0.64264675]])
```

Interpretacija vrednosti koeficijenata je važna zbog razumevanja samog modela i uticaja atributa.

In [26]:
```
N = len(data.feature_names)
values = model.coef_[0]
plt.figure(figsize = (10, 5))
plt.bar(np.arange(0, N),values)
plt.xticks(np.arange(0, N), data.feature_names, rotation='vertical')

plt.show()
```



Na primer, na ovaj način se može uvideti da atributi kao što su radius error, worst texture

ili worst area važni u predviđanju negativne klase tj. maligni tumor, doku je atribut compactness error važan za predikciju pozitivne klase tj. benigni tumor.

Evaluacija

In [27]:
```
y_test_predicted = model.predict(X_test)
```

Sve mere od interesa se nalaze u metrics biblioteci.

In [28]:
```
metrics.accuracy_score(y_test, y_test_predicted)
```

Out[28]:
```
0.9787234042553191
```

In [29]:
```
metrics.precision_score(y_test, y_test_predicted)
```

Out[29]:
```
0.9913793103448276
```

In [30]:
```
metrics.recall_score(y_test, y_test_predicted)
```

Out[30]:
```
0.9745762711864406
```

In [31]:
```
metrics.f1_score(y_test, y_test_predicted)
```

Out[31]:
```
0.982905982905983
```