INTELIGENTNI SISTEMI

as. ms Vladimir Jocović
as. ms Adrian Milaković

LINEARNA REGRESIJA

Python

10

sklearn BIBLIOTEKA

Šta je sklearn biblioteka?

- Scikit-learn (sklearn) je besplatna softverska biblioteka napisana za programski jezik Pajton namenjena mašinskom učenju.
- Podržava različite algoritme mašinskog učenja:
  - Klasifikacija: stabla odlučivanja, logistička regresija, *SVM* (*support vector machines*), najbliži susedi (*k-nearest neighbours*) itd.
  - Regresija: regresiona stabla, linearna regresija, Bajesova regresija itd.
  - Klasterizacija: *k-means*, propagacija afiniteta itd.
- Predstavlja nadgradnju na popularnu Python biblioteku, koja pruža podršku za višedimenzione nizove i matematičke funkcije – NumPy.

Brzi tutorijali:

- sklearn: https://scikit-learn.org/stable/tutorial/index.html

matplotlib i pyplot BIBLIOTEKA

Šta je matplotlib biblioteka?

- **Matplotlib** je besplatna softverska biblioteka napisana za programski jezik Pajton namenjena iscrtavanju i vizuelizaciji statičkih, dinamičkih i interaktivnih objekata.

Šta je pyplot biblioteka?

- **Pyplot** je biblioteka napisana za programski jezik Pajton, koja se nalazi kao modul u okviru biblioteke **matplotlib**. Predstavlja kolekciju funkcija koje omogućavaju da biblioteka matplotlib radi kao MATLAB.

Brzi tutorijali:

- Matplotlib: https://matplotlib.org/3.3.3/tutorials/index.html
- pyplot: https://matplotlib.org/tutorials/introductory/pyplot.html

sklearn, matplotlib, pyplot

Kako preuzeti navedene biblioteke?

- Sklearn i matplotlib nisu standardne Python biblioteke. Moguće je preuzeti ih na više različitih načina:
  - Iz PyCharm Python okruženja: File -> Settings -> Project: *Project Name* -> Python Interpreter, a zatim izabrati dugme **+** za preuzimanje novog paketa. Zatim je potrebno pretražiti **scikit-learn**, odnosno **matplotlib** i instalirati paket na dugme **Install Package**. Na dugme **-** moguće je ukloniti instalirani paket.
  - Iz PyCharm Python Terminala komandom **pip install scikit-learn**, odnosno **pip install matplotlib**.
  - Pisanjem naredbe **import pandas**, odnosno **import matplotlib (import matplotlib.pyplot)** u fajlu sa Pajton izvornim kodom, a zatim prelaskom mišem preko naredbe (*hover*) iz tooltip prozora izabrati opciju **Install package pandas (matplotlib)**.
  - Korišćenjem popularnog package manager-a **Anaconda**.

Zadatak - CENA NEKRETNINA

Dat je skup podataka koji predstavlja nekretnine, za čiju datu površinu je neophodno napraviti predikciju cene. Površina nekretnina zadata je u kvadratnim metrima, a cena u američkim dolarima.

Potrebno je realizovati model linearne regresije koristeći:

- Algoritam gradijentnog spusta uz minimizaciju funkcije greške.
- Model linearne regresije iz Python biblioteke sklearn.

Zadatak - Rešenje

```
# Uvoz pandas modula za manipulaciju nad podacima.
# Alias pd za pandas se koristi po konvenciji.
import pandas as pd

# Uvoz pyplot modula za vizuelizaciju podataka.
# Alias plt za pyplot se koristi po konvenciji.
import matplotlib.pyplot as plt

# Uvoz numpy modula za rad sa visedimenzionim nizovima.
# Alias np za numpy se koristi po konvenciji.
import numpy as np

# Mapa boja (colormap) za bojenje funkcije greske
from matplotlib import cm

# Obican model Linearne regresije
from sklearn.linear_model import LinearRegression
```

Zadatak - Rešenje

```
# Citanje .csv fajla i kreiranje DataFrame-a od njega.
# Zaglavlje .csv fajla predstavlja imena kolona DataFrame-a.
data = pd.read_csv('datasets/house_prices.csv')

# Ispis prvih 5 redova DataFrame-a
print(data.head())

# Izgled prvih 5 redova DataFrame-a
#     area   price
# 0  122.07  207500
# 1   91.70   94500
# 2  157.01  181000
# 3   83.80  125500
# 4  126.35  187500
```

Zadatak - Rešenje

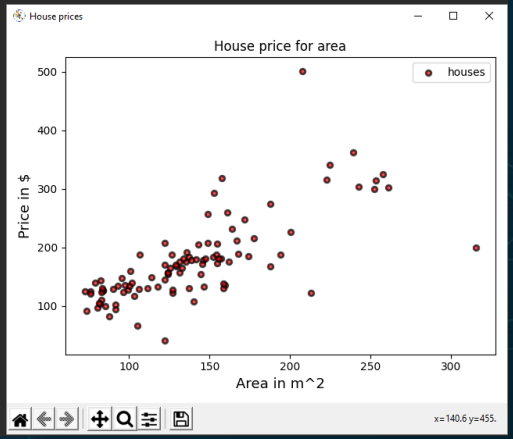
```
# Uzimamo sve redove (:) i kolonu area.
# Predstavlja skup atributa neophodan za predikciju izlaza.
# ['area'], a ne 'area', jer hocemo DataFrame, a ne Series
X = data.loc[:, ['area']]

# Uzimamo kolonu price.
# Predstavlja skup labela za date atribute (area).
y = data['price']

# Skaliramo vrednosti labela na skalu povrsina nekretnina.
# Bitno je da su vrednosti na priblizno slicnoj skali.
# U suprotnom, korak ucenja bi bio prilagodjen samo
# koeficijentima atributa sa vrednostima na istoj skali.
# Time ubrzavamo algoritam gradijentnog spusta.
y = y / 1000
```

Zadatak - Rešenje

```
# Kreiranje (aktiviranje) nove figure.
# To ce biti i naslov novog prozora.
plt.figure('House prices')

# Rasejavanje tacaka u Dekartovom koordinatnom sistemu.
# X - podaci za X osu, y - podaci za y osu
# s - velicina markera, c - boja markera
# marker - izgled markera, alpha - transparentnost markera
# edgecolors - boja ivice markera
# linewidth - velicina ivice markera
# label - labela za legendu za tacke
plt.scatter(X, y, s=23, c='red', marker='o', alpha=0.7,
            edgecolors='black', linewidths=2, label='houses')
```

Zadatak - Rešenje

```
# Labele za ose i naslov.
plt.xlabel('Area in m^2', fontsize=13)
plt.ylabel('Price in $', fontsize=13)
plt.title('House price for area')

# Prikazivanje legende na grafiku.
# Postavljena je labela samo za prvu tacku,
# ali sve tacke predstavljaju kuce.
plt.legend()

# Grafik zauzima celu povrsinu prozora.
plt.tight_layout()
# plt.show()
```



Zadatak - Rešenje

```
# Potrebno je napraviti predikciju cene nekretnine
# na osnovu njene povrsine.
# Na grafiku je prikazano da je cena
# linearna funkcija povrsine (koeficijenti c1, c0):
# y_target = X_features * c1 + c0
class LinearRegressionGradientDescent:
    def __init__(self):
        self.coeff = None
        self.features = None
        self.target = None
        self.mse_history = None

    def set_coefficients(self, *args):
        # Mapiramo koeficijente u niz oblika (n + 1) x 1
        self.coeff = np.array(args).reshape(-1, 1)
```

Zadatak - Rešenje

```
# Racuna se koristeci Mean-square error.
# m - broj uzoraka (redova u DataFrame-u)
# y_predicted - m x 1 niz predvidjenih cena uzoraka
# y_target - m x 1 niz stvarnih cena uzoraka
# MS_error = (1 / 2 * m) * sum ((y_predicted - y_target) ^ 2)
def cost(self):
    predicted = self.features.dot(self.coeff)
    s = pow(predicted - self.target, 2).sum()
    return (0.5 / len(self.features)) * s

# Argument mora biti DataFrame
def predict(self, features):
    features = features.copy(deep=True)
    features.insert(0, 'c0', np.ones((len(features), 1)))
    features = features.to_numpy()
    return features.dot(self.coeff).reshape(-1, 1).flatten()
```

Zadatak - Rešenje

```
# Jedan korak u algoritmu gradijentnog spusta.
def gradient_descent_step(self, learning_rate):
    # learning_rate - korak ucenja; dimenzije ((n + 1) x 1);
    # korak ucenja je razlicit za razlicite koeficijente
    # m - broj uzoraka
    # n - broj razlicitih atributa (osobina)
    # features - dimenzije (m x (n + 1));
    # n + 1 je zbog koeficijenta c0
    # self.coeff - dimenzije ((n + 1) x 1)
    # predicted - dimenzije (m x (n + 1)) x ((n + 1) x 1) = (m x 1)
    predicted = self.features.dot(self.coeff)

    # koeficijeni se azuriraju po formuli:
    # coeff(i) = coeff(i) - learning_rate * gradient(i)
    # za i-ti koeficijent koji mnozi i-ti atribut
    # gledaju se samo vrednosti i-tog atributa za sve uzorke
    # gradient(i) = (1 / m) * sum(y_predicted - y_target) * features(i)
```

Zadatak - Rešenje

```
# (predicted - self.target) - dimenzije (m x 1)
# features - dimenzije (m x (n + 1));
# transponovana matrica ima dimenzije ((n + 1) x m)
# gradient - dimenzije ((n + 1) x m) x (m x 1) = (n + 1) x 1
s = self.features.T.dot(predicted - self.target)
gradient = (1. / len(self.features)) * s
self.coeff = self.coeff - learning_rate * gradient
return self.coeff, self.cost()

def perform_gradient_descent(self, learning_rate, num_iterations=100):
    # Istorija Mean-square error-a kroz iteracije gradijentnog spusta.
    self.mse_history = []
    for i in range(num_iterations):
        _, curr_cost = self.gradient_descent_step(learning_rate)
        self.mse_history.append(curr_cost)
    return self.coeff, self.mse_history
```

Zadatak - Rešenje

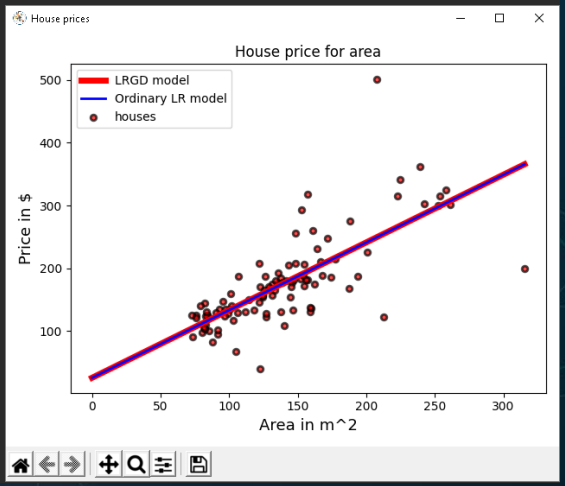
```
# features mora biti DataFrame
def fit(self, features, target):
    self.features = features.copy(deep=True)

    # Pocetna vrednost za koeficijente je 0.
    # self.coeff - dimenzije ((n + 1) x 1)
    coeff_shape = len(features.columns) + 1
    self.coeff = np.zeros(shape=coeff_shape).reshape(-1, 1)

    # Unosi se kolona jedinica za koeficijent c0,
    # kao da je vrednost atributa uz c0 jednaka 1.
    self.features.insert(0, 'c0', np.ones((len(features), 1)))

    # self.features - dimenzije (m x (n + 1))
    self.features = self.features.to_numpy()
    # self.target - dimenzije (m x 1)
    self.target = target.to_numpy().reshape(-1, 1)
```

Zadatak - Rešenje

```
# Generisanje 200 vrednosti za x osu podjednako udaljenih
# u opsegu [0, maksimalna_povrsina_nekretnine]
# koje ce predstavljati povrsine nekretnina
# za koje se radi predikcija.
spots = 200
estates = pd.DataFrame(data=np.linspace(0, max(X['area']), num=spots))

# Kreiranje i obucavanje modela
lrgd = LinearRegressionGradientDescent()
lrgd.fit(X, y)
learning_rates = np.array([[0.17], [0.0000475]])
res_coeff, mse_history = lrgd.perform_gradient_descent(learning_rates, 20)

# Vizuelizacija modela
plt.figure(1)
line, = plt.plot(estates[0], lrgd.predict(estates), lw=5, c='red')
line.set_label('LRGD model')
```

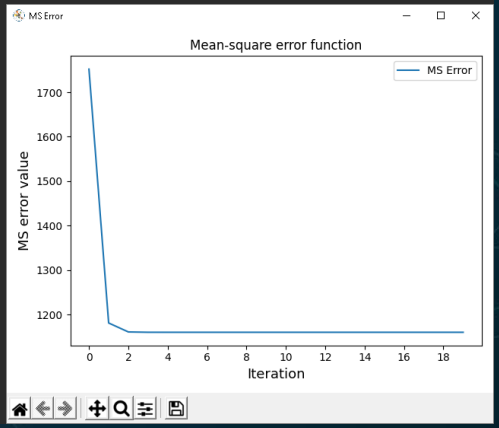
Zadatak - Rešenje

```
# Kreiranje i obucavanje sklearn.LinearRegression modela
lr_model = LinearRegression()
lr_model.fit(X, y)

# Vizuelizacija modela
line, = plt.plot(estates[0], lr_model.predict(estates), lw=2, c='blue')
line.set_label('Ordinary LR model')

# Lokacija legende (gore levo)
plt.legend(loc='upper left')
plt.show()
```



Zadatak – Rešenje

```
# Vizuelizacija MS_error funkcije kroz iteracije
# za model koji koristi gradijentni spust.
plt.figure('MS Error')
plt.plot(np.arange(0, len(mse_history), 1), mse_history)
plt.xlabel('Iteration', fontsize=13)
plt.ylabel('MS error value', fontsize=13)
plt.xticks(np.arange(0, len(mse_history), 2))
plt.title('Mean-square error function')
plt.tight_layout()
plt.legend(['MS Error'])
plt.show()
```

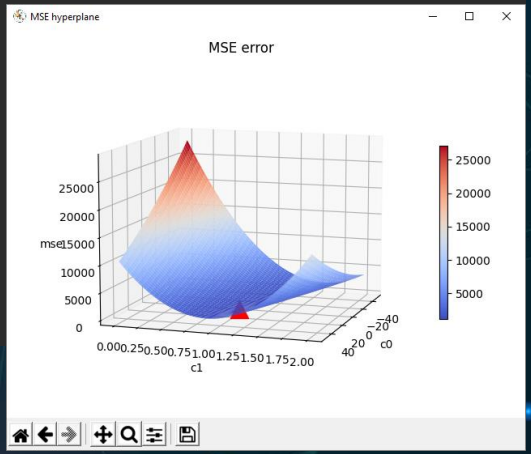


Zadatak - Rešenje

```
# Vizuelizacija Mean-square error funkcije
# u zavisnosti od vrednosti koeficijenata c0 i c1.
# Pravimo meshgrid, odnosno kombinacije svake dve vrednosti:
# c0 = -50, c1 = 0; c0 = -50, c1 = 0.02 ... ;
# c0 = -49, c1 = 0; c0 = -49, c1 = 0.02 ...;
# ...
# c0 = 50, c1 = 2; ...
spots = 100
c0, c1 = np.meshgrid(np.linspace(-50, 50, spots), np.linspace(0, 2, spots))

# Od visedimenzionih nizova pravimo jednodimenzione nizove
c0 = c0.flatten()
c1 = c1.flatten()
mse_values = []
for i in range(len(c0)):
    lrgd.set_coefficients(c0[i], c1[i])
    mse_values.append(lrgd.cost())
```

Zadatak – Rešenje

```
fig = plt.figure('MSE hyperplane')
# U okviru figure-a dodaju se axes-i (subplot-ovi u okviru plot-a).
# Figure objekat je 1x1, a trenutni axes je 1 (numeracija krece od 1).
ax = plt.subplot(1, 1, 1, projection='3d')
surf = ax.plot_surface(c0.reshape(spots, spots), c1.reshape(spots, spots),
                       np.array(mse_values).reshape(spots, spots),
                       cmap=cm.coolwarm)

min_mse_ind = mse_values.index(min(mse_values))
ax.scatter(c0[min_mse_ind], c1[min_mse_ind], mse_values[min_mse_ind],
           c='r', s=250, marker='^')
fig.colorbar(surf, shrink=0.5)
ax.set_xlabel('c0')
ax.set_ylabel('c1')
ax.set_zlabel('mse')
plt.title('MSE error')
plt.tight_layout()
plt.show()
```



Zadatak - Rešenje

```
# Testiranje predikcije oba modela nad jednim uzorkom
# price = c1 * area + c0
example_estate_sqm = 122
example_estate = pd.DataFrame(data=[example_estate_sqm])
lrgd.set_coefficients(res_coeff)
print(f'LRGD price for {example_estate_sqm}sqm house is '
      f'{lrgd.predict(example_estate)[0]:.2f} thousand $')
print(f'LRGD c0: {lrgd.coeff.flatten()[0]:.2f}, '
      f'c1: {lrgd.coeff.flatten()[1]:.2f}')
print(f'LR price for {example_estate_sqm}sqm house is '
      f'{lr_model.predict(example_estate)[0]:.2f} thousand $')
print(f'LR c0: {lr_model.intercept_:.2f}, '
      f'c1: {lr_model.coef_[0]:.2f}')

# Ispis:
# LRGD price for 122sqm house is 156.88 thousand $
# LRGD c0: 25.11, c1: 1.08
# LR price for 122sqm house is 156.88 thousand $
# LR c0: 25.10, c1: 1.08
```

Zadatak - Rešenje

```
# Stampanje mse za oba modela
lrgd.set_coefficients(res_coeff)
print(f'LRGD MSE: {lrgd.cost():.2f}')

c = np.concatenate((np.array([lr_model.intercept_]), lr_model.coef_))
lrgd.set_coefficients(c)
print(f'LR MSE: {lrgd.cost():.2f}')

# Restauracija koeficijenata
lrgd.set_coefficients(res_coeff)

# Ispis:
# LRGD MSE: 1160.05
# LR MSE: 1160.05
```

Zadatak - Rešenje

```
# Racunanje score-a za oba modela
data_test = pd.read_csv('datasets/house_prices_test.csv')
X = data_test[['area']]
y = data_test['price'] / 1000

# Zapamte se koeficijenti LR modela,
# da bi se postavili LRGD koeficijenti i izracunao LR score.
lr_coef_ = lr_model.coef_
lr_int_ = lr_model.intercept_
lr_model.coef_ = lrgd.coeff.flatten()[1:]
lr_model.intercept_ = lrgd.coeff.flatten()[0]
print(f'LRGD score: {lr_model.score(X, y):.2f}')

# Restauriraju se koeficijenti LR modela
lr_model.coef_ = lr_coef_
lr_model.intercept_ = lr_int_
print(f'LR score: {lr_model.score(X, y):.2f}')

# Ispis:
# LRGD score: 0.49
# LR score: 0.49
```

PITANJA?

http://ri4es.etf.rs/