VJEŽBA 4: LINEARNI MODELI ZA REGRESIJU.

I. Cilj vježbe:

Upoznati se s linearnim modelima za rješavanje regresijskih problema. Proširenje linearnih modela baznim funkcijama. Regularizacija.

II. Opis vježbe:

II.1. Nadgledano učenje. Regresija.

U ovoj vježbi razmatra se problem nadgledanog učenja gdje je cilj odrediti nepoznatu funkcionalnu ovisnost između *m* ulaznih veličina $X = [x_1, x_2, \dots, x_m]$ i izlazne veličine *y* na temelju podatkovnih primjera. Promatra se slučaj kada je izlazna veličina *y* kontinuirana veličina (regresija). Pri tome su podatkovni primjeri parovi koji se sastoje od vektora ulaznih veličina i vrijednosti izlazne veličine, stoga se *i*–ti podatkovni primjer ili uzorak može prikazati kao uređeni par $(\mathbf{x}^{(i)}, y^{(i)})$. Vektor ulaznih veličina zapisuje u obliku:

$$\mathbf{x}^{(i)} = \left[x_1^{(i)}, x_2^{(i)}, \dots, x_m^{(i)}\right]^T \tag{4-1}$$

Skup koji se sastoji od *n* raspoloživih mjernih podataka $\left(\mathbf{x}^{(1)}, y^{(1)}\right), \left(\mathbf{x}^{(2)}, y^{(2)}\right), \dots, (\mathbf{x}^{(n)}, y^{(n)})$ može se zapisati u matričnom obliku:

$$\mathbf{X} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_m^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_m^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \dots & x_m^{(n)} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}. \tag{4-2}$$

Prilikom nadgledanog učenja, često se pretpostavlja da su podaci generirani sljedećim mehanizmom:

$$y = f(\mathbf{x}) + \varepsilon\,, \tag{4-3}$$

pri čemu je $f(\mathbf{x})$ nepoznata funkcionalna ovisnost, a ε pogreška nezavisna od *X* i ima srednju vrijednost 0. Cilj nadgledanog učenja je upravo odrediti aproksimaciju $\hat{f}$ funkcionalne ovisnosti f koja se onda koristi za predikciju izlazne veličine za novi (nepoznati) uzorak ulaznih veličina:

$$\hat{y}(\mathbf{x}) = \hat{f}(\mathbf{x}) = h_\theta(\mathbf{x})\,. \tag{4-4}$$

Dobivena aproksimacija obično se naziva model koji se često označava i s h_θ. U parametarskom pristupu izgradnji modela algoritmima strojnog učenja, model se definira kao funkcija s konačnim brojem parametara θ. Pri tome je se od svih mogućih funkcija odabire samo mali podskup mogućih funkcija (prostor hipoteza), kao primjerice linearne funkcije, a postupkom učenja upravo se određuju ovi nepoznati parametri θ na temelju raspoloživih podataka.
Uz pretpostavku da su raspoloživi podaci (**X**, **y**) modela nezavisni i jednoliko distribuirani (i.i.d.), te uz izraz (4-3), metodom maksimalne vjerojatnosti moguće je doći do kriterijske funkcije, tj. kriterija prema kojem se mogu odrediti parametri θ modela na temelju raspoloživih podataka:

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\sum_{i=1}^{n}\left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2\,, \tag{4-5}$$

pa se optimalni parametri modela dobivaju minimizacijom kriterija (4-5):

$$\boldsymbol{\theta}_{\mathrm{ML}} = \operatorname{argmin}\left(J(\boldsymbol{\theta})\right). \tag{4-6}$$

II.2. Linearni modeli za regresiju

Kao aproksimaciju funkciju moguće je koristiti linearnu funkciju. Model je u tom slučaju oblika:

$$\hat{y}\left(\mathbf{x}^{(i)}\right) = h_\theta\left(\mathbf{x}^{(i)}\right) = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_m x_m^{(i)} = \theta_0 + \sum_{j=1}^{m} \theta_j x_j^{(i)} \,, \tag{4-7}$$

i naziva se višedimenzionalna linearna regresija (engl. multiple linear regression). Obično se vektor ulaznih veličina proširuje s dodatnom ulaznom veličinom x_0 koja je jednaka 1. U tom slučaju cijeli se model može jednostavnije zapisati:

$$\hat{y}\left(\mathbf{x}^{(i)}\right) = h_\theta\left(\mathbf{x}^{(i)}\right) = \sum_{j=0}^{m} \theta_j x_j^{(i)} = \boldsymbol{\theta}^T \mathbf{x}^{(i)} \,, \tag{4-8}$$

gdje je:

$$\begin{gathered} \mathbf{x}^{(i)} = \left[x_0^{(i)}, x_1^{(i)}, x_2^{(i)}, \ldots, x_m^{(i)}\right]^T \,, \\ \boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_m]^T \,. \end{gathered} \tag{4-9}$$

Vektor parametara $\boldsymbol{\theta}$ određuje se prema (4-6). Rješenje optimizacijskog problema u ovom slučaju postoji u zatvorenoj formi:

$$\boldsymbol{\theta}_{\mathrm{ML}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \,. \tag{4-10}$$

Drugi način rješavanja optimizacijskog problema (4-6) podrazumijeva korištenje iterativnog numeričkog postupka. Na primjer, metodom gradijentnog spusta moguće je pronaći optimalne vrijednosti parametara modela (4-8):

Metoda gradijentnog spusta za optimizaciju kriterija (4-5)

1. odaberi početnu vrijednost vektora parametara $\boldsymbol{\theta}$; definiraj duljinu koraka α
2. simultano osvježi svaki element vektora $\boldsymbol{\theta}$:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} \,, \qquad za\ j = 0, \ldots, m$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{n} \sum_{i=1}^{n} \left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right) x_j^{(i)}$$

3. ako je zadovoljen kriterij zaustavljanja (npr. dostignut je maksimalni broj iteracija) zaustavi optimiranje; u suprotnom idi na 2. korak

II.3. Proširenje linearnog modela. Regularizacija.

Proširenje linearnog modela (4-8) moguće je u smislu nelinearne transformacije ulaznih veličina. Na taj način model može opisati i nelinearne odnose između ulaznih i izlazne veličine, a problem optimizacije je i dalje linearne prirode. Često se koristi polinomsko proširenje, na način da se ulazna veličina transformira:

$$\phi(x) = x^k \quad za\ k = 0, \ldots, K \,, \tag{4-11}$$

pri čemu je K broj koji je potrebno unaprijed zadati. Optimalna vrijednost K ovisi o složenosti prisutne funkcionalne ovisnosti između ulaznih veličina i izlazne veličine. Ako se odabere preveliki K doći će pretjeranog usklađivanja na podatke (engl. overfitting) dok u slučaju premalog K model neće biti u mogućnosti dovoljno dobro opisati prisutnu funkcionalnu povezanost (engl. underfitting). Jedan od načina sprječavanja pretjeranog usklađivanja na podatke je odabir optimalnog K metodom unakrsnog vrjednovanja (engl. cross validation) ili regularizacijom kriterijske funkcije. Na primjer, često se koristi regularizacija oblika:

$$J(\boldsymbol{\theta}) = \frac{1}{2n}\left[\sum_{i=1}^{n}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{m} \theta_j^2\right] \,, \tag{4-12}$$

koja je poznata i pod nazivom ridge regresija. Parametar λ određuje koliki je kompromis između složenosti modela i usklađivanja na podatke. Rješenje optimizacijskog problema (4-12) i u ovom slučaju postoji u zatvorenoj formi:

$$\boldsymbol{\theta}_{\mathrm{RR}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \,, \tag{4-13}$$

gdje je $\mathbf{I}$ jedinična matrica koja na prvom mjestu ima 0 ($\mathbf{I} = diag\{0,1,1,\ldots,m+1\}$). Drugi način rješavanja optimizacijskog problema (4-13) podrazumijeva korištenje iterativnog numeričkog postupka. Na primjer, metodom gradijentnog spusta moguće je pronaći optimalne vrijednosti parametara modela (4-8):

Metoda gradijentnog spusta za optimizaciju kriterija (4-12)

1. odaberi početnu vrijednost vektora parametara $\boldsymbol{\theta}$; definiraj duljinu koraka α
2. simultano osvježi svaki element vektora $\boldsymbol{\theta}$:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j}, \qquad za\ j = 0, \dots, m$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_0} = \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)x_0^{(i)}$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{n}\sum_{i=1}^{n}\left(h_\theta\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{n}\theta_j, \quad za\ j = 1, \dots, m$$

3. ako je zadovoljen kriterij zaustavljanja (npr. dostignut je maksimalni broj iteracija) zaustavi optimiranje; u suprotnom idi na 2. korak

II.4. Vrjednovanje modela

Testiranje predikcijskih sposobnosti odnosno sposobnosti generalizacije potrebno je provesti na zasebnom skupu podataka koji se naziva skup za testiranje. Pri tome je moguće koristiti različite mjere od kojih je najpopularnija srednja kvadratna pogreška na podacima (engl. mean squared error):

$$MSE_{test} = \frac{1}{n_{test}}\sum_{i=1}^{n_{test}}\left(y^{(i)} - h_\theta\left(\mathbf{x}^{(i)}\right)\right)^2. \tag{4-14}$$

Naravno, modeli s dobrim generalizacijskim sposobnostima imaju manji iznos srednje kvadratne pogreške na testnim podacima.
Koeficijent determinacije R^2 pokazuje koliko je varijacija u podacima obuhvaćeno modelom:

$$R^2_{test} = 1 - \frac{\sum_{i=1}^{n_{test}}\left(y^{(i)} - h_\theta\left(\mathbf{x}^{(i)}\right)\right)^2}{\sum_{i=1}^{n_{test}}\left(y^{(i)} - \bar{y}\right)}, \tag{4-15}$$

gdje je $\bar{y} = \frac{1}{n_{test}}\sum_{i=1}^{n_{test}} y^{(i)}$. Najbolji model (savršeno procjenjuje vrijednosti izlazne veličine) ima vrijednost 1. Model oblika $h_\theta\left(\mathbf{x}^{(i)}\right) = \bar{y}$ ima vrijednost $R^2_{test} = 0$. Modeli koji lošije procjenjuju izlaznu veličinu od ovog konstantnog modela imaju negativan R^2_{test}.
Prilikom izgradnje linearnog regresijskog modela važno je provjeriti reziduale na skupu za učenje. Rezidual za neki i-ti podatak iz skupa za učenje definira se kao:

$$e^{(i)} = y^{(i)} - h_\theta\left(\mathbf{x}^{(i)}\right). \tag{4-16}$$

Prikaz reziduala podrazumijeva prikaz vrijednosti reziduala za svaki mjerni uzorak na osi ordinata, dok se na osi apscisa nanosi vrijednost ulazne veličine x_j. Ako linearni model zadovoljavajuće opisuje determinističku komponentu u izrazu (4-3), tada reziduali trebaju imati nasumičan karakter, tj. jednoliko su raspršeni oko horizontalne osi. U suprotnom, nelinearni model je prikladniji za dani problem.

II.5. Scikit-learn biblioteka

Scikit-learn je open source Python biblioteka za strojno učenje koja se temelji na NumPy, SciPy i matplotlib bibliotekama. Unutar biblioteke je implementiran velik broj metoda strojnog učenja za nadzirano (regresija, klasifikacija) i nenadzirano učenje (klasteriranje, smanjivanje dimenzionalnosti podataka, procjena gustoće vjerojatnosti). U ovoj biblioteci modeli su implementirani u obliku Python objekta koji implementiraju metode *fit* (procjena parametara modela) i *predict* (izračunavanje izlaza modela za uzorke ulaznih veličina). Sljedeći primjer demonstrira kako se inicijalizira linearni model te kako se procjenjuju parametri na temelju skupa za učenje koji su predstavljeni poljima *xtrain* i *ytrain*, te kako se računa izlaz modela za nove vrijednosti ulaznih veličina koje su pohranjene u numpy polju *xtest*.

```
import numpy as np
import sklearn.linear_model as lm

linearModel = lm.LinearRegression()
linearModel.fit(xtrain,ytrain)
```

```
ytest_pred = linearModel.predict(xtest)
```

III. Priprema za vježbu:

Nema posebne pripreme za vježbu.

IV. Rad na vježbi:

1. Klonirajte vaš repozitorij `PSU_LV` na računalo pomoću `git clone` naredbe. Kreirajte direktorij `LV4` unutar direktorija `PSU_LV`. U ovaj direktorij kopirajte sve datoteke vezane za ovu vježbu, a koje se nalaze na loomen stranici predmeta pod `LV4`.

2. Riješite dane zadatke, pri čemu Python skripte trebaju imati naziv `zad_x.py` (gdje je `x` broj zadatka) i trebaju biti pohranjene u direktorij `PSU_LV/LV4/`. Svaki zadatak rješavajte u zasebnoj *git* grani koju spojite s glavnom granom kada riješite pojedini zadatak. Pohranite skripte u lokalni *git* repozitorij kao i u `PSU_LV` repozitorij na vašem gitlab računu. Svaki puta kada načinite promjene koje se spremaju u *git* sustav napišite i odgovarajuću poruku prilikom izvršavanja `commit` naredbe.

3. Nadopunite postojeću tekstualnu datoteku `PSU_LV/LV4/Readme.md` s kratkim opisom vježbe i kratkim opisom rješenja vježbe te pohranite promjene u lokalnu bazu. Na kraju pohranite promjene u udaljeni repozitorij.

Zadatak 1

Pokrenite primjer 4.1. iz dodatka. U ovom primjeru generiraju se umjetni podaci te se izgrađuje linearni regresijski model. Razmislite koji dio programskog koda odgovara kojem dijelu teorije predstavljene u opisu ove vježbe.

Zadatak 2

Pokrenite primjer 4.2. iz dodatka. U ovom primjeru generiraju se umjetni podaci te se izgrađuje linearni regresijski model. Međutim, ovaj model se proširuje polinomskim članovima prema (4-11). Koja je razlika u odnosnu na rezultate dobivene u zadatku 1?

Zadatak 3

Prepravite skriptu tako da rezultat izvođenja skripte bude sljedeći:
- vektori *MSEtrain* i *MSEtest* koji sadrži srednju kvadratnu pogrešku na podacima za učenje i podacima za testiranje za tri različita stupnja dodatnih veličina u modelu (npr., *degree*= 2, 6 i 15)
- slika na kojoj je usporedba izlaza modela za tri različita stupnja dodatnih veličina u modelu (npr., *degree* 2, 6 i 15) s pozadinskom funkcijom koja je generirala podatke

Što se događa s ovim rezultatima ako je na raspolaganju veći odnosno manji broj uzoraka za učenje? Simulirajte.

Zadatak 4

Primjer 4.3. učitava skup podataka koji se nalazi u datoteci `car_processed.csv` u dataframe. Ova datoteka sadrži podatke koji su prikupljeni s online platforme cardekho.com koja se bavi prodajom vozila u Indiji. Za svaki automobil na raspolaganju je:

name:	ime automobila
year:	godina proizvodnje automobila
selling_price:	cijena po kojoj se automobil prodaje (logaritam cijene u indijskim rupijima)
km_driven:	prijeđeni broj kilometara
fuel:	tip motora (Diesel, Petrol, CNG, LPG)
seller_type:	prodavač (individual/dealer)

transmission:	tip mjenjača (automatic/manual)
owner:	broj prethodnih vlasnika
mileage:	potrošnja automobila
engine:	zapremnina motora u cm3
max_power:	maksimalna snaga motora (bhp)
seats	broj sjedala

Skripta crta određene grafove kako bi se dobio uvid u dataset. Proučite dobivene grafove te odgovorite na sljedeća pitanja

1. Koliko mjerenja (automobila) je dostupno u datasetu?
2. Kakav je tip pojedinog stupca u dataframeu?
3. Koji automobil ima najveću cijenu, a koji najmanju?
4. Koliko automobila je proizvedeno 2012. godine?
5. Koji automobil je prešao najviše kilometara, a koji najmanje?
6. Koliko najčešće automobili imaju sjedala?
7. Kolika je prosječna prijeđena kilometraža za automobile s dizel motorom, a koliko za automobile s benzinskim motorom?

Zadatak 5

Potrebno je izgraditi model koji procjenjuje cijenu automobila na temelju numeričkih ulaznih veličina.

1. Izbacite veličine koje nisu potrebne poput imena automobila. Koriste pandas funkciju `df.drop`.

2. Podijelite skup na train i test u omjeru 80% – 20%. Koristite ugrađenu sklearn funkciju:
   ```
   from sklearn.model_selection import train_test_split
   ```

3. Skalirajte ulazne podatke korištenjem sklearn funkcija:
   ```
   from sklearn.preprocessing import MinMaxScaler, StandardScaler
   ```

4. Odredite parametre linearnog regresijskog modela.

5. Evaluirajte izgrađeni model na trening i testnom skupu pomoću sklearn funkcija:

   ```
   from sklearn.metrics import mean_absolute_error, r2_score,
   mean_squared_error, max_error
   ```

6. Što se događa s pogreškom na testnom skupu kada mijenjate broj ulaznih veličina?

Zadatak 6

Dodajte u model iz prethodnog zadatka i kategoričke varijable. Pri tome koristite pandas funkciju `pd.get_dummies` za one hot kodiranje kategoričkih veličina. Jesu li se značajno poboljšali rezultati? Što ako dodate u model i polinomske članove?

Zadatak 7

Isprobajte i druge linearne modele na danom problemu procjene cijene automobila kao npr. Ridge regresija.

V. Izvještaj s vježbe

Kao izvještaj s vježbe prihvaća se web link na repozitorij pod nazivom `PSU_LV`.

VI. Dodatak

Primjer 4.1.

```
import numpy as np
import matplotlib.pyplot as plt
import sklearn.linear_model as lm
from sklearn.metrics import mean_squared_error

def non_func(x):
    y = 1.6345 - 0.6235*np.cos(0.6067*x) - 1.3501*np.sin(0.6067*x) - 1.1622 * np.cos(2*x*0.6067) - 0.9443*np.sin(2*x*0.6067)
    return y

def add_noise(y):
   np.random.seed(14)
   varNoise = np.max(y) - np.min(y)
   y_noisy = y + 0.1*varNoise*np.random.normal(0,1,len(y))
   return y_noisy

x = np.linspace(1,10,100)
y_true = non_func(x)
y_measured = add_noise(y_true)

plt.figure(1)
plt.plot(x,y_measured,'ok',label='mjereno')
plt.plot(x,y_true,label='stvarno')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc = 4)

np.random.seed(12)
indeksi = np.random.permutation(len(x))
indeksi_train = indeksi[0:int(np.floor(0.7*len(x)))]
indeksi_test = indeksi[int(np.floor(0.7*len(x)))+1:len(x)]

x = x[:, np.newaxis]
y_measured = y_measured[:, np.newaxis]

xtrain = x[indeksi_train]
ytrain = y_measured[indeksi_train]

xtest = x[indeksi_test]
ytest = y_measured[indeksi_test]

plt.figure(2)
plt.plot(xtrain,ytrain,'ob',label='train')
plt.plot(xtest,ytest,'or',label='test')
plt.xlabel('x')
plt.ylabel('y')
plt.legend(loc = 4)

linearModel = lm.LinearRegression()
linearModel.fit(xtrain,ytrain)

print('Model je oblika y_hat = Theta0 + Theta1 * x')
print('y_hat = ', linearModel.intercept_, '+', linearModel.coef_, '*x')

ytest_p = linearModel.predict(xtest)
MSE_test = mean_squared_error(ytest, ytest_p)

plt.figure(3)
plt.plot(xtest,ytest_p,'og',label='predicted')
plt.plot(xtest,ytest,'or',label='test')
plt.legend(loc = 4)

x_pravac = np.array([1,10])
x_pravac = x_pravac[:, np.newaxis]
y_pravac = linearModel.predict(x_pravac)
plt.plot(x_pravac, y_pravac)
```

Primjer 4.2.

```
import numpy as np
import matplotlib.pyplot as plt
import sklearn.linear_model as lm
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import PolynomialFeatures

def non_func(x):
    y = 1.6345 - 0.6235*np.cos(0.6067*x) - 1.3501*np.sin(0.6067*x) - 1.1622 * np.cos(2*x*0.6067) - 0.9443*np.sin(2*x*0.6067)
    return y

def add_noise(y):
    np.random.seed(14)
    varNoise = np.max(y) - np.min(y)
    y_noisy = y + 0.1*varNoise*np.random.normal(0,1,len(y))
    return y_noisy

x = np.linspace(1,10,50)
y_true = non_func(x)
y_measured = add_noise(y_true)

x = x[:, np.newaxis]
y_measured = y_measured[:, np.newaxis]

# make polynomial features
poly = PolynomialFeatures(degree=15)
xnew = poly.fit_transform(x)

np.random.seed(12)
indeksi = np.random.permutation(len(xnew))
indeksi_train = indeksi[0:int(np.floor(0.7*len(xnew)))]
indeksi_test = indeksi[int(np.floor(0.7*len(xnew)))+1:len(xnew)]

xtrain = xnew[indeksi_train,]
ytrain = y_measured[indeksi_train]

xtest = xnew[indeksi_test,]
ytest = y_measured[indeksi_test]

linearModel = lm.LinearRegression()
linearModel.fit(xtrain,ytrain)

ytest_p = linearModel.predict(xtest)
MSE_test = mean_squared_error(ytest, ytest_p)

plt.figure(1)
plt.plot(xtest[:,1],ytest_p,'og',label='predicted')
plt.plot(xtest[:,1],ytest,'or',label='test')
plt.legend(loc = 4)

#pozadinska funkcija vs model
plt.figure(2)
plt.plot(x,y_true,label='f')
plt.plot(x, linearModel.predict(xnew),'r-',label='model')
plt.xlabel('x')
plt.ylabel('y')
plt.plot(xtrain[:,1],ytrain,'ok',label='train')
plt.legend(loc = 4)
```

Primjer 4.3.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# ucitavanje ociscenih podataka
df = pd.read_csv('cars_processed.csv')
print(df.info())

# razliciti prikazi
sns.pairplot(df, hue='fuel')

sns.relplot(data=df, x='km_driven', y='selling_price', hue='fuel')
df = df.drop(['name','mileage'], axis=1)

obj_cols = df.select_dtypes(object).columns.values.tolist()
num_cols = df.select_dtypes(np.number).columns.values.tolist()

fig = plt.figure(figsize=[15,8])
for col in range(len(obj_cols)):
    plt.subplot(2,2,col+1)
    sns.countplot(x=obj_cols[col], data=df)

df.boxplot(by ='fuel', column =['selling_price'], grid = False)

df.hist(['selling_price'], grid = False)

tabcorr = df.corr()
sns.heatmap(df.corr(), annot=True, linewidths=2, cmap= 'coolwarm')

plt.show()
```