Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Лабораторная работа №5

по дисциплине «Машинное обучение»

Выполнил
студент гр. 3530904/00103 Плетнева А.Д.

Руководитель Селин И.А.

Санкт-Петербург
2023

Оглавление

Задание

1. Загрузите данные из файла reglab1.txt. Постройте по набору данных регрессии, используя модели с различными зависимыми переменными. Выберите наиболее подходящую модель.
2. Реализуйте следующий алгоритм для уменьшения количества признаков, используемых для построения регрессии: для каждого $k \in \{0,1,...,d\}$ выбрать подмножество признаков мощности k^1, минимизирующее остаточную сумму квадратов *RSS*. Используя полученный алгоритм, выберите оптимальное подмножество признаков для данных из файла reglab.txt. Объясните свой выбор.
3. Загрузите данные из файла cygage.txt. Постройте регрессию, выражающую зависимость возраста исследуемых отложений от глубины залегания, используя веса наблюдений. Оцените качество построенной модели.
4. Загрузите данные из файла longley.csv. Данные состоят из 7 экономических переменных, наблюдаемых с 1947 по 1962 годы (*n=16*). Исключите переменную Population. Разделите данные на тестовую и обучающую выборки равных размеров случайным образом. Постройте линейную регрессию по признаку Employed. Постройте гребневую регрессию для значений $\lambda = 10^{-3+0.2 \cdot i}$, $i = 0,...,25$. Подсчитайте ошибку на тестовой и обучающей выборке для линейной регрессии и гребневой регрессии на данных значениях λ, постройте графики. Объясните полученные результаты.
5. Загрузите данные из файла eustock.csv. Данные содержат ежедневные котировки на момент закрытия фондовых бирж: Germany DAX (Ibis), Switzerland SMI, France CAC, и UK FTSE. Постройте на одном графике все кривые изменения котировок во времени. Постройте линейную регрессию для каждой модели в отдельности и для всех моделей вместе. Оцените, какая из бирж имеет наибольшую динамику.
6. Загрузите данные из файла JohnsonJohnson.csv. Данные содержат поквартальную прибыль компании Johnson & Johnson с 1960 по 1980 гг. Постройте на одном графике все кривые изменения прибыли во времени. Постройте линейную регрессию для каждого квартала в отдельности и для всех кварталов вместе. Оцените, в каком квартале компания имеет наибольшую и наименьшую динамику доходности. Сделайте прогноз по прибыли в 2016 году во всех кварталах и в среднем по году.
7. Загрузите данные из файла cars.csv. Данные содержат зависимости тормозного пути автомобиля (футы) от его скорости (мили в час). Данные получены в 1920 г. Постройте регрессионную модель и оцените длину тормозного пути при скорости 40 миль в час.
8. Загрузите данные из файла svmdata6.txt. Постройте регрессионный алгоритм метода опорных векторов (sklearn.svm.SVR) с параметром C = 1, используя ядро "rbf". Отобразите на графике зависимость среднеквадратичной ошибки на обучающей выборке от значения параметра ε. Прокомментируйте полученный результат
9. Загрузите набор данных из файла nsw74psid1.csv. Постройте регрессионное дерево (sklearn.tree.DecisionTreeRegressor) для признака re78. Постройте линейную регрессионную модель и SVM-регрессию для этого набора данных. Сравните качество построенных моделей, выберите оптимальную модель и объясните свой выбор.

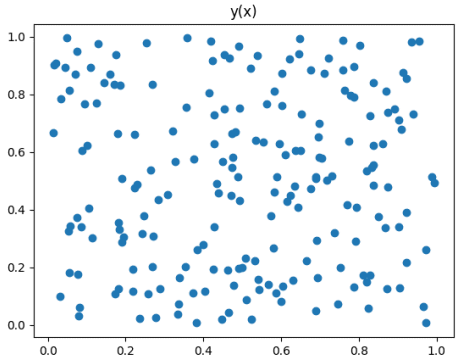
Ход работы

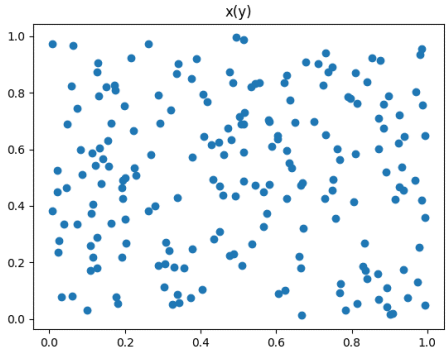
Задание 1

Построили по набору данных регрессии, используя модели с различными зависимыми переменными. Сравним точность полученных моделей:

```
Coef R^2 x(z): 0.2252452590211148
Coef R^2 y(z): 0.5157048933756456
Coef R^2 z(x): 0.17401001314671116
Coef R^2 y(x): -0.07041686820906734
Coef R^2 z(y): 0.5090946149908573
Coef R^2 x(y): -0.053729739027498
Coef R^2 x(z,y): 0.8976651955330801
Coef R^2 y(z,x): 0.9356838747028364
Coef R^2 z(x,y): 0.9516456233150588
```

Значение R^2 отрицательное, так как модель предсказывает результаты хуже случайного угадывания. Графики y(x) и x(y) выглядят следующим образом:





Для данных зависимостей нельзя использовать линейную регрессию
Лучший результат получается для модели z(x,y)

Задание 2

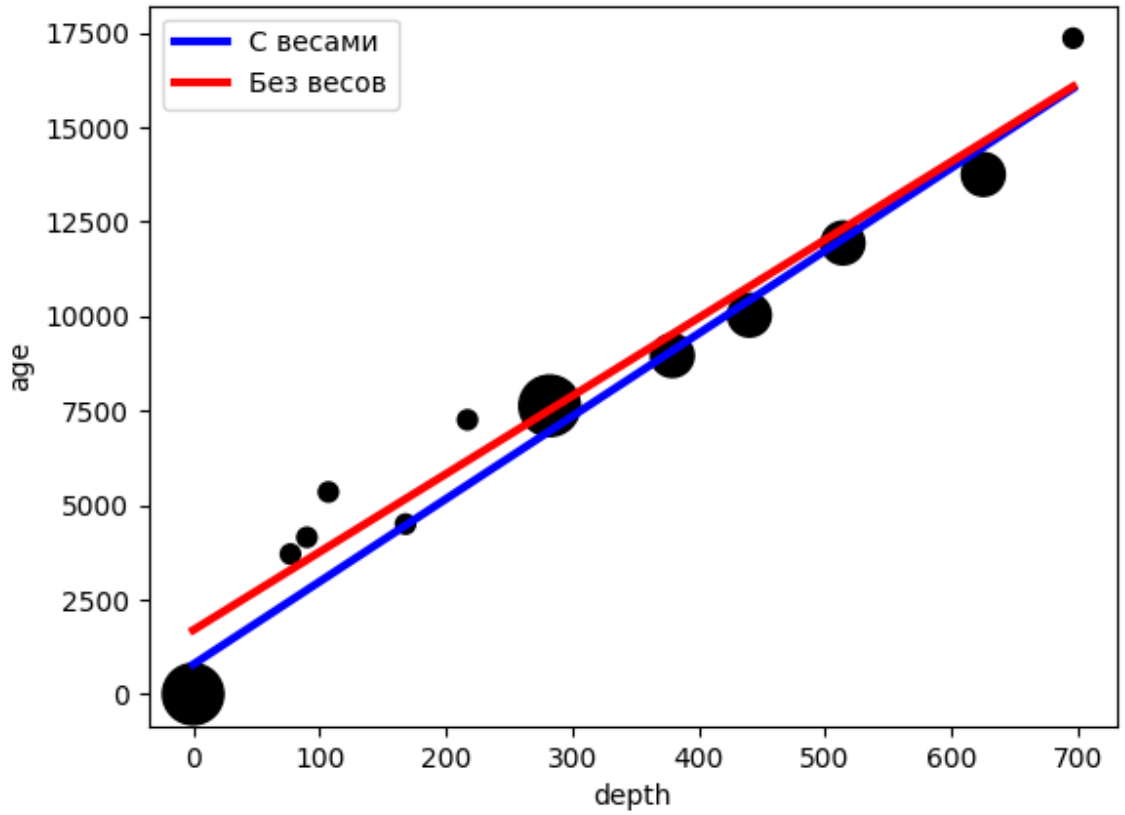
RSS (Residual Sum of Squares) - это сумма квадратов ошибок (остатков) регрессионной модели, которая говорит, насколько хорошо модель подходит к наблюдаемым данным. Найдем RSS для каждого подмножества признаков согласно алгоритму, описанному в задании:

```
k=1
['x1']: RSS=157.2197758284528
['x2']: RSS=268.2457708399214
['x3']: RSS=393.49046860549686
['x4']: RSS=394.5904974603926
k=2
['x1', 'x2']: RSS=0.5379617105782545
['x1', 'x3']: RSS=156.35406574563234
['x1', 'x4']: RSS=157.2192683152796
['x2', 'x3']: RSS=267.7954542430099
['x2', 'x4']: RSS=267.8061361302551
['x3', 'x4']: RSS=393.458728053751
k=3
['x1', 'x2', 'x3']: RSS=0.3322662149737569
['x1', 'x2', 'x4']: RSS=0.36196824827729474
['x1', 'x3', 'x4']: RSS=156.3483397038426
['x2', 'x3', 'x4']: RSS=267.4415471943948
k=4
['x1', 'x2', 'x3', 'x4']: RSS=0.192863541483843
```

Для k=1 лучшее подмножество: ['x1']: RSS=157.2197758284528
Для k=2 лучшее подмножество: ['x1', 'x2']: RSS=0.5379617105782545
Для k=3 лучшее подмножество: ['x1', 'x2', 'x3']: RSS=0.3322662149737569
Для k=4 лучшее подмножество: ['x1', 'x2', 'x3', 'x4']: RSS=0.192863541483843
Заметим, что уже для k=2 регрессор дает хороший результат, а если мы убираем один из признаков x1 или x2 результат заметно ухудшается, поэтому оптимальное подмножество признаков ['x1', 'x2'].

Задание 3

Построим две регрессионные модели: с весами и без весов, видим, что модель с весами дает лучший результат




```
С весами: 0.9736839487233144
Без весов: 0.9592555361125507
```

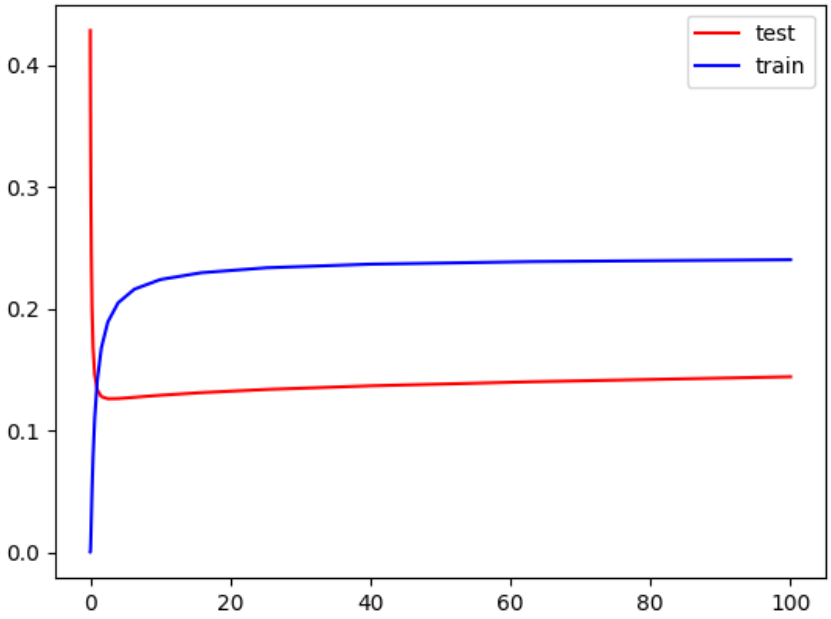
Задание 4

Для линейной регрессии получаем следующие результаты:

```
MSE на тестовой выборке для линейной регрессии: 0.43063263455488443
MSE на обучающей выборке для линейной регрессии: 0.000326876613255013
```

На обучающей выборке получаем очень маленькую ошибку, а на тестовой напротив очень большую, то есть модель переобучена

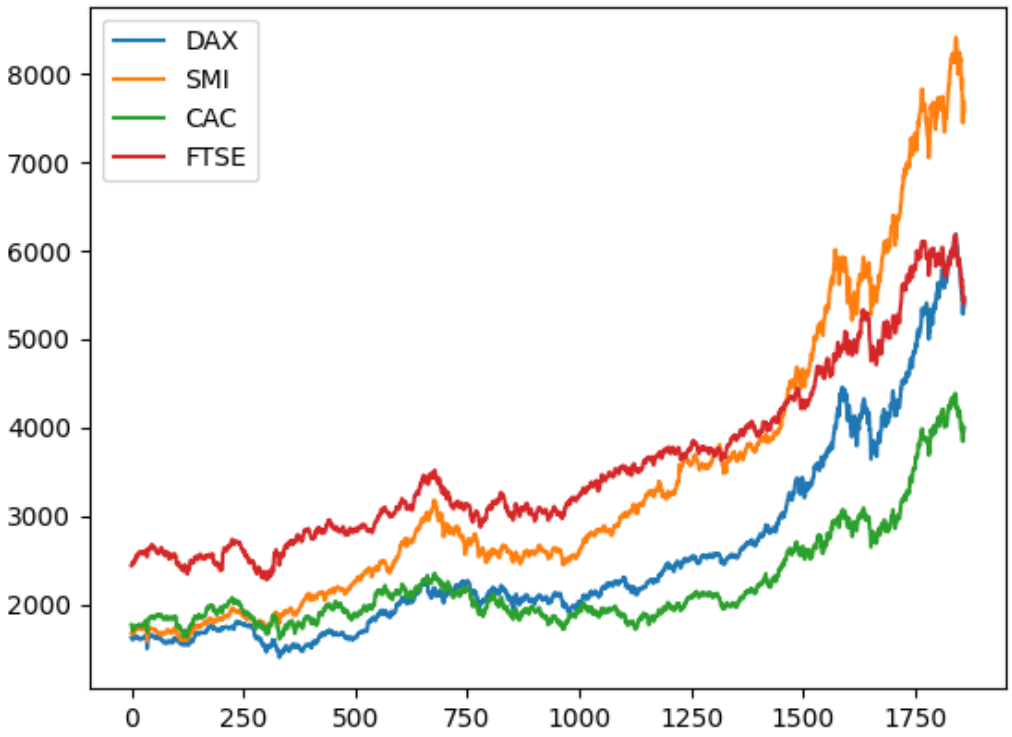
Используем гребневую регрессию для регуляризации:



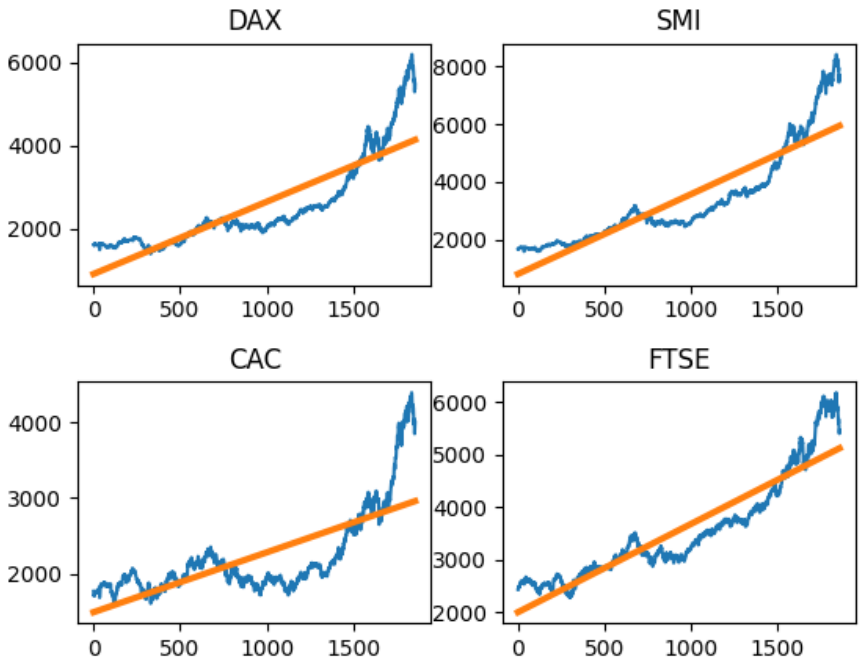
Видим, что при добавлении регуляризационного слагаемого, ошибка на тестовой выборке заметно уменьшается.

Задание 5

Построили на одном графике все кривые изменения котировок во времени



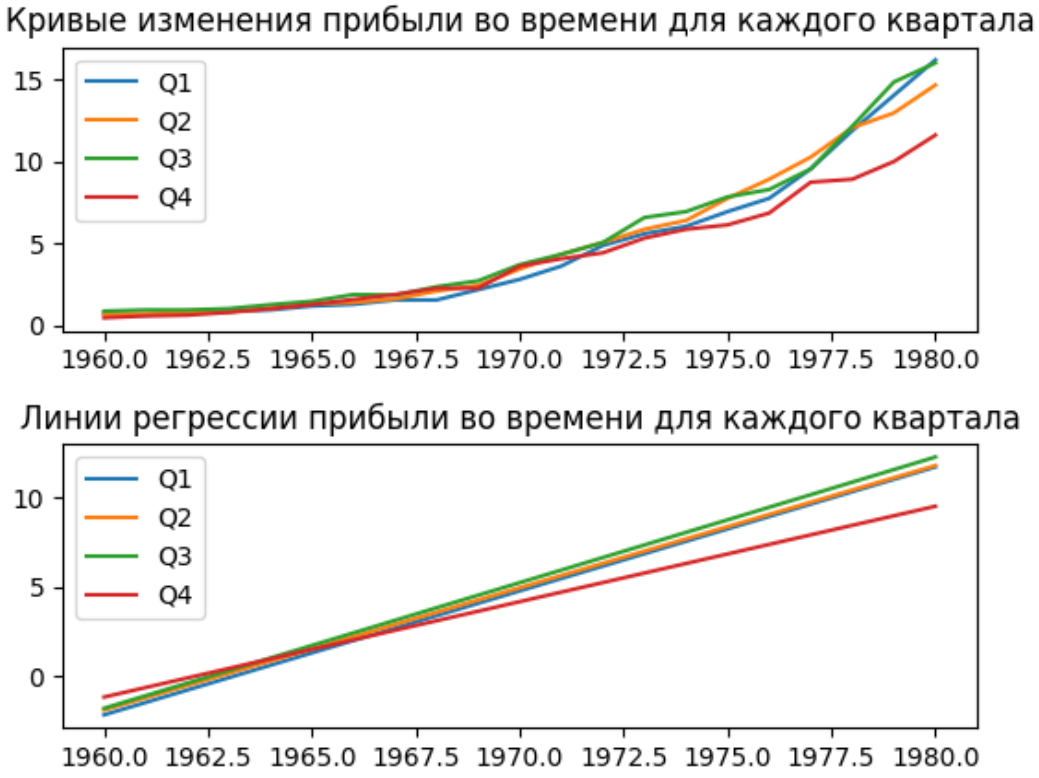
Линейная регрессия для каждой акции:



Линия регрессии SMI имеет наибольший наклон, то есть SMI имеет наибольшую динамику.

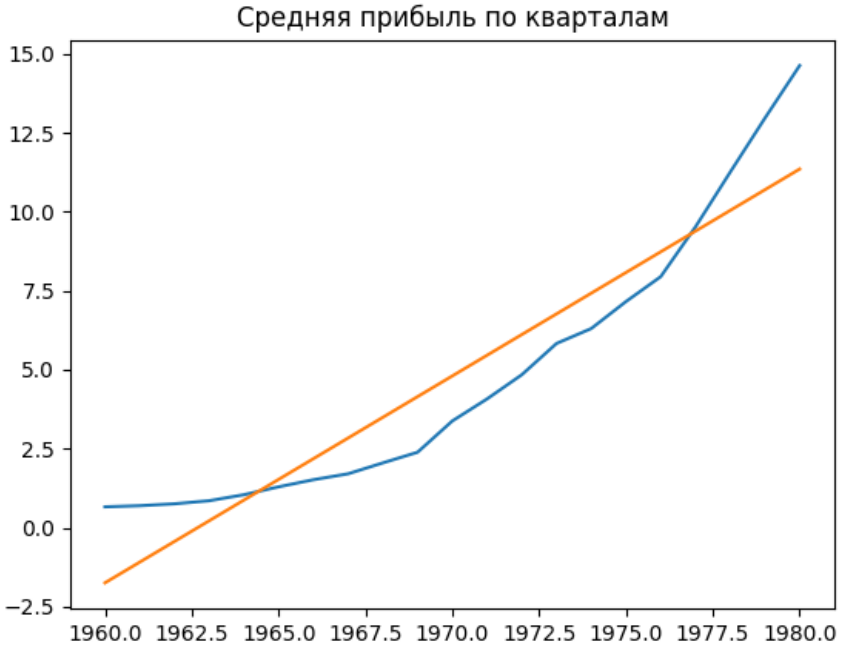
Задание 6

Построили на одном графике все кривые изменения прибыли во времени, а также лини регрессии для каждого квартала:



Наименьшая динамика наблюдается в 4 квартале, а наибольшая – в 3.

Также построили линию регрессии для средней прибыли по кварталам

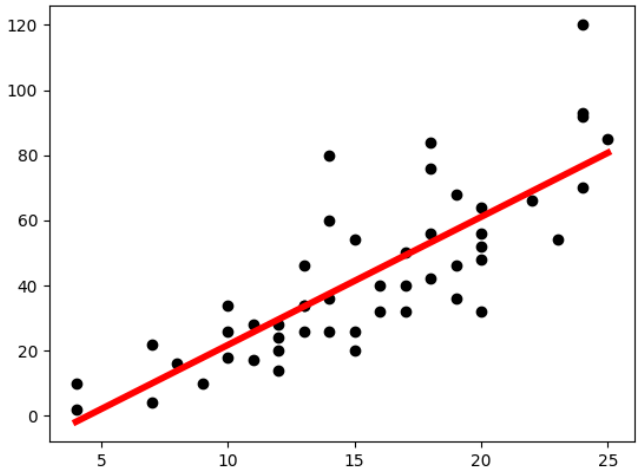


Сделали прогноз по прибыли в 2016 году во всех кварталах и в среднем по году, наибольшая прибыль ожидается в 3 квартале, наименьшая – в 4. Предсказание прибыли в среднем по году соответствует среднему значению предсказаний для каждого квартала.

```
Предсказание прибыли для квартала Q1 в 2016 году: [36.75963636]
Предсказание прибыли для квартала Q2 в 2016 году: [36.48945455]
Предсказание прибыли для квартала Q3 в 2016 году: [37.65393939]
Предсказание прибыли для квартала Q4 в 2016 году: [28.79391342]
Предсказание прибыли в среднем по году в 2016 году: [34.92423593]
```

Задание 7

Построили регрессионную модель:
Зависимость тормозного пути автомобиля (футы) от его скорости (мили в час)

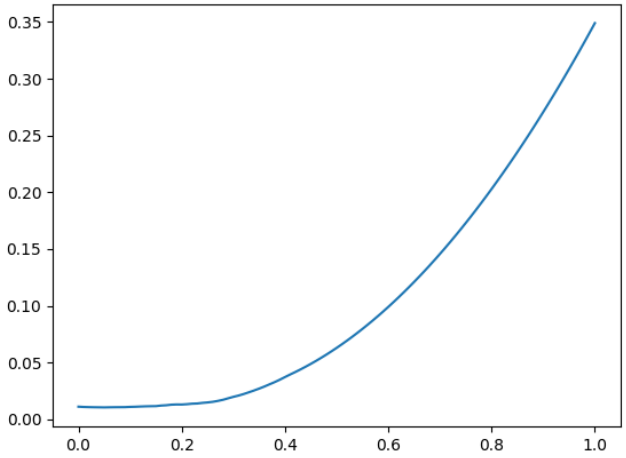


Оценка длины тормозного пути при скорости 40 миль в час:

```
Длина тормозного пути при скорости 40 миль в час: [139.71725547]
```

Задание 8

Построили регрессионный алгоритм метода опорных векторов.
На графике отображена зависимость среднеквадратичной ошибки на обучающей выборке от значения параметра ε:



Параметр ε определяет минимальное расстояние между предсказанными значениями и границами «окна» или зазора (margin), в котором находятся объекты обучающей выборки, чем больше этот зазор, тем меньше точность.

Задание 9

Построили регрессионное дерево, линейную регрессионную модель и SVM-регрессию для этого набора данных для признака re78, посчитали R^2:

```
R^2 SVR: 0.018116038588998462
R^2 LinearRegression: 0.6897459694931868
R^2 DecisionTreeRegressor: 0.1919311621692874
```

Линейная регрессионная модель дает наилучший результат, поэтому является оптимальной моделью.

Текст программы

Ex1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.model_selection import train_test_split

data = pd.read_csv('reglab1.txt', sep='\t')

Z = data.iloc[:, 0]
X = data.iloc[:, 1]
Y = data.iloc[:, 2]
ZX = data.iloc[:, :-1]
ZY = data.iloc[:, [0, 2]]
XY = data.iloc[:, 1:]
array_xyz = [Z, X, Y]

for i in range(len(array_xyz)):
    for j in range(len(array_xyz)):
        if i != j:
            X_train, X_test, y_train, y_test = train_test_split(array_xyz[i],
array_xyz[j], test_size=0.2,

random_state=1)
            reg = linear_model.LinearRegression()

            reg.fit(np.array(X_train).reshape(-1, 1), np.array(y_train))
            print(
                f'Coef R^2 {y_train._name}({X_train._name}):
{reg.score(np.array(X_test).reshape(-1, 1), np.array(y_test))}')


X_train, X_test, y_train, y_test = train_test_split(ZY, X, test_size=0.2,
                                                    random_state=1)
reg = linear_model.LinearRegression()
reg.fit(np.array(X_train), np.array(y_train))
print(
    f'Coef R^2 {y_train._name}({X_train.columns[0]},{X_train.columns[1]}):
{reg.score(np.array(X_test), np.array(y_test))}')

X_train, X_test, y_train, y_test = train_test_split(ZX, Y, test_size=0.2,
                                                    random_state=1)
reg.fit(np.array(X_train), np.array(y_train))
print(
    f'Coef R^2 {y_train._name}({X_train.columns[0]},{X_train.columns[1]}):
{reg.score(np.array(X_test), np.array(y_test))}')

X_train, X_test, y_train, y_test = train_test_split(XY, Z, test_size=0.2,
                                                    random_state=1)
reg.fit(np.array(X_train), np.array(y_train))
print(
    f'Coef R^2 {y_train._name}({X_train.columns[0]},{X_train.columns[1]}):
{reg.score(np.array(X_test), np.array(y_test))}')

plt.scatter(X, Y)
plt.title("y(x)")
plt.show()
plt.scatter(Y, X)
plt.title("x(y)")
plt.show()
```

ex2

```
from itertools import combinations

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

data = pd.read_csv('reglab.txt', sep='\t')
Y = data.iloc[:, 0]
all_x = [data.iloc[:, i] for i in range(1, 5)]

for i in range(4):
    print(f'k={i+1}')
    for comb in combinations(all_x, i+1):
        features = []
        for feature in comb:
            features.append(feature)
        df_x = pd.concat(features, axis=1)
        reg = linear_model.LinearRegression().fit(df_x, Y)
        RSS = mean_squared_error(Y, reg.predict(df_x)) * len(Y)
        print(f'{[df_x.columns[j] for j in range(i+1)]}: RSS={RSS}')
```

ex3

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import linear_model

data = pd.read_csv('cygage.txt', sep='\t')

age = data.iloc[:, 0]
depth = np.array(data.iloc[:, 1]).reshape(-1, 1)
weights = data.iloc[:, 2]

reg = linear_model.LinearRegression().fit(depth, age, weights)
print(f'С весами: {reg.score(depth, age, weights)}')
# Plot outputs
sizes = [500 * w for w in weights]
plt.plot(depth.reshape(-1, 1), reg.predict(depth), color="blue", linewidth=3)

# plt.xticks(())
# plt.yticks(())

reg = linear_model.LinearRegression().fit(depth, age)
print(f'Без весов: {reg.score(depth, age)}')
# Plot outputs
plt.plot(depth.reshape(-1, 1), reg.predict(depth), color="red", linewidth=3)

# plt.xticks(())
# plt.yticks(())
plt.legend(['С весами', 'Без весов'])
plt.scatter(depth.reshape(-1, 1), age, color="black", s=sizes)
plt.xlabel('depth')
plt.ylabel('age')
plt.show()
```

ex4

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import linear_model
```

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

data = pd.read_csv('longley.csv', sep=',')
data = data.drop(columns=['Population'])
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5,
random_state=10)
reg = linear_model.LinearRegression().fit(X_train, y_train)
print(f'MSE на тестовой выборке для линейной регрессии:
{mean_squared_error(y_test, reg.predict(X_test))}')
print(f'MSE на обучающей выборке для линейной регрессии:
{mean_squared_error(y_train, reg.predict(X_train))}')

r2_test = []
r2_train = []
lambdas = [10 ** (-3 + 0.2 * i) for i in range(26)]
for lambda_ in lambdas:
    ridge_reg = Ridge(alpha=lambda_).fit(X_train, y_train)
    r2_test.append(mean_squared_error(y_test, ridge_reg.predict(X_test)))
    r2_train.append(mean_squared_error(y_train, ridge_reg.predict(X_train)))

plt.plot(lambdas, r2_test, color='red')
plt.plot(lambdas, r2_train, color='blue')
plt.legend(['test', 'train'])
plt.show()
```

ex5

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import linear_model
from sklearn.linear_model import Ridge, LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

data = pd.read_csv('eustock.csv', sep=',')
DAX = data['DAX']
SMI = data['SMI']
CAC = data['CAC']
FTSE = data['FTSE']

x = [i for i in range(len(data))]
plt.plot(x, DAX)
plt.plot(x, SMI)
plt.plot(x, CAC)
plt.plot(x, FTSE)

plt.legend(['DAX', 'SMI', 'CAC', 'FTSE'])

plt.show()
x = np.array(x).reshape(-1, 1)
reg_DAX = LinearRegression().fit(x, DAX)
reg_SMI = LinearRegression().fit(x, SMI)
reg_CAC = LinearRegression().fit(x, CAC)
reg_FTSE = LinearRegression().fit(x, FTSE)
reg_all = LinearRegression().fit(x, data)

fig, ax = plt.subplots(2, 2)
fig.subplots_adjust(hspace=0.4)

ax[0][0].plot(x, DAX)
```

```
ax[0][0].plot(x, reg_DAX.predict(x), linewidth=3)
ax[0][0].set_title('DAX')

ax[0][1].plot(x, SMI)
ax[0][1].plot(x, reg_SMI.predict(x), linewidth=3)
ax[0][1].set_title('SMI')

ax[1][0].plot(x, CAC)
ax[1][0].plot(x, reg_CAC.predict(x), linewidth=3)
ax[1][0].set_title('CAC')

ax[1][1].plot(x, FTSE)
ax[1][1].plot(x, reg_FTSE.predict(x), linewidth=3)
ax[1][1].set_title('FTSE')

plt.show()
```

ex6

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas import DataFrame
from sklearn.linear_model import Ridge, LinearRegression

data = pd.read_csv('JohnsonJohnson.csv', sep=',')
Q1 = DataFrame()
Q2 = DataFrame()
Q3 = DataFrame()
Q4 = DataFrame()
for index, row in data.iterrows():
    if row['index'].endswith('Q1'):
        Q1 = pd.concat([Q1, row], axis=1)
    if row['index'].endswith('Q2'):
        Q2 = pd.concat([Q2, row], axis=1)
    if row['index'].endswith('Q3'):
        Q3 = pd.concat([Q3, row], axis=1)
    if row['index'].endswith('Q4'):
        Q4 = pd.concat([Q4, row], axis=1)

all_Q = [Q1.transpose(), Q2.transpose(), Q3.transpose(), Q4.transpose()]
fig, ax = plt.subplots(2, 1)
fig.subplots_adjust(hspace=0.4)
for q in all_Q:
    x = np.array(q['index'].str[:4].astype(int)).reshape(-1, 1)
    y = np.array(q['value'])
    reg = LinearRegression().fit(x, y)
    print(
        f'Предсказание прибыли для квартала {list(q["index"])[0][5:]} в 2016
году: {reg.predict(np.array([2016]).reshape(-1, 1))}')
    ax[0].plot(x, y)
    ax[1].plot(x, reg.predict(x))

ax[0].set_title('Кривые изменения прибыли во времени для каждого квартала')
ax[1].set_title('Линии регрессии прибыли во времени для каждого квартала')
ax[0].legend(['Q1', 'Q2', 'Q3', 'Q4'])
ax[1].legend(['Q1', 'Q2', 'Q3', 'Q4'])

plt.show()
avg_value = np.mean([q['value'] for q in all_Q], axis=0)
date = [i for i in range(1960, 1981)]
date = np.array(date).reshape(-1, 1)
reg = LinearRegression().fit(date, avg_value)
print(f'Предсказание прибыли в среднем по году в 2016 году:
{reg.predict(np.array([2016]).reshape(-1, 1))}')
```

```
plt.plot(date, avg_value)
plt.plot(date, reg.predict(date))
plt.title('Средняя прибыль по кварталам')
plt.show()
```

ex7

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas import DataFrame
from sklearn.linear_model import Ridge, LinearRegression

data = pd.read_csv('cars.csv', sep=',')
speed = np.array(data['speed']).reshape(-1, 1)
dist = data['dist']
reg = LinearRegression().fit(speed, dist)
print(f'Длина тормозного пути при скорости 40 миль в час:
{reg.predict(np.array([40]).reshape(-1, 1))}')
plt.scatter(speed, dist, color='black')
plt.plot(speed, reg.predict(speed), color='red', linewidth=4)
plt.show()
```

ex8

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error
from sklearn.svm import SVR
from pandas import DataFrame
from sklearn.linear_model import Ridge, LinearRegression

data = pd.read_csv('svmdata6.txt', sep='\t')
X = np.array(data['X']).reshape(-1, 1)
y = data['Y']
eps = [i * 0.01 for i in range(0, 101)]
mse = []
for epsilon in eps:
    reg = SVR(C=1, epsilon=epsilon).fit(X, y)
    mse.append(mean_squared_error(y, reg.predict(X)))

plt.plot(eps, mse)
plt.show()
```

ex9

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from pandas import DataFrame
from sklearn.linear_model import Ridge, LinearRegression

data = pd.read_csv('nsw74psid1.csv')
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)

svr = SVR().fit(X_train, y_train)
print(f'R^2 SVR: {svr.score(X_test, y_test)}')
```

```
lr = LinearRegression().fit(X_train, y_train)
print(f'R^2 LinearRegression: {lr.score(X_test, y_test)}')

dtr = DecisionTreeRegressor().fit(X_train, y_train)
print(f'R^2 DecisionTreeRegressor: {dtr.score(X_test, y_test)}')
```