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

# Отчет по лабораторной работе №5

# «Регрессия»

# по дисциплине «Машинное обучение»

Выполнил
студент гр. 3530904/90102 Афанасьев Е.Д.

Руководитель
старший преподаватель ВШПИ Селин И. А.

Санкт-Петербург 2022

# Оглавление

## Задачи

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.

## reglab1.txt различные зависимые переменные

Построим регрессионные модели с зависимыми переменными x, y и z.

Были получены следующие результаты

```
('x', 0.922682223722205),
('y', 0.9393710177397852),
('z', 0.9727096270687096)
```

Все модели продемонстрировали хороший результат, но лучшей из них является модель с зависимой переменной z.

# Уменьшение количества признаков

Построим модели для всех вариантов зависимых переменных

Получился следующий результат

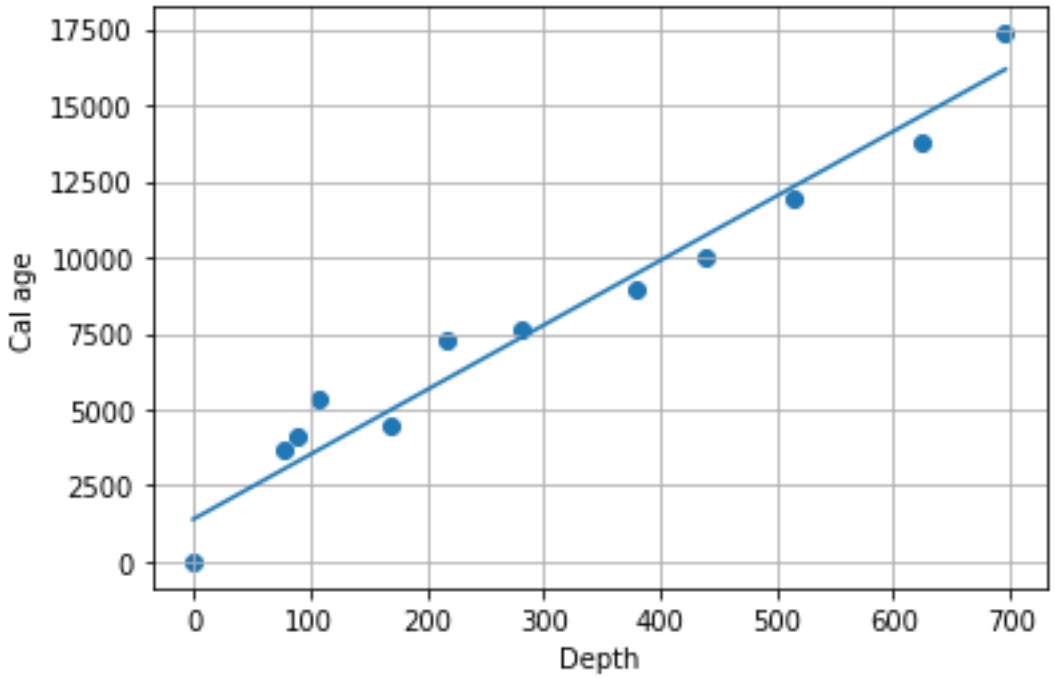
```
x1 x2 x3 x4
Residiual sum of squares 0.0008812820680217011
Score 0.999454961080623
x1 x2 x3
Residiual sum of squares 0.0016283387958568138
Score 0.9992723101823325
x1 x2 x4
Residiual sum of squares 0.0022451592661334426
Score 0.998984864778062
x1 x3 x4
Residiual sum of squares 0.9656426277351251
Score 0.47989993713027534
x2 x3 x4
Residiual sum of squares 1.4633376298318945
Score 0.11699746708126313
x1 x2
Residiual sum of squares 0.002894054418116864
Score 0.9986470333654877
x1 x3
Residiual sum of squares 0.9589019263112102
Score 0.4124852785620212
x2 x3
Residiual sum of squares 1.1709269593641218
Score 0.33311840693884875
x1 x4
Residiual sum of squares 0.8515689551674057
Score 0.5883313677907658
x2 x4
Residiual sum of squares 1.3088274508279567
Score 0.3156657750052335
x3 x4
Residiual sum of squares 1.8699663349636755
Score -0.1000629426553481
x1
Residiual sum of squares 0.7215074134563162
Score 0.5933892921022614
x2
Residiual sum of squares 1.1691542242897437
Score 0.37436670823994855
x3
Residiual sum of squares 1.782791913904695
Score -0.09734815188037005
x4
Residiual sum of squares 2.527428449793898
Score -0.1129239833185649
```

При наличии всех параметров получается лучшая точность и меньшие rss. Но если необходимо уменьшить количество параметров для построения регрессии, то следуют исключать в следующем порядке x4, x3, x2, x1. Что видно, в последних случаях с одним параметром.

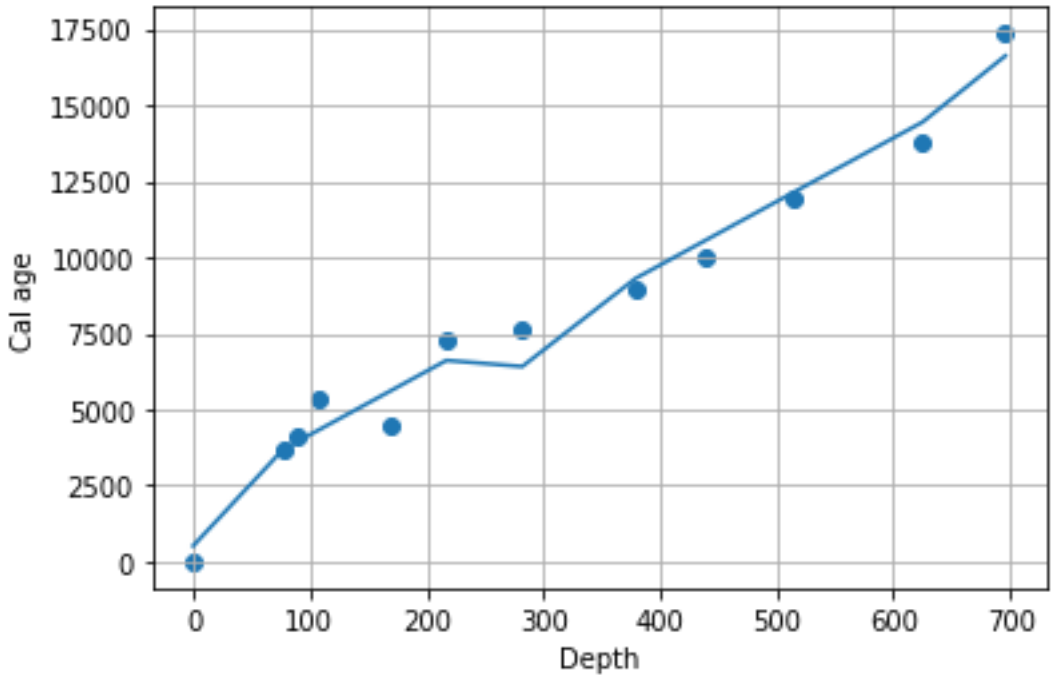
## cygage.txt построение регрессии с использованием весов

Построим модель, не использующие веса, а затем с весами.

```
Score train:  0.9609963361830721
Score test:  0.9358199759951694
```




```
Score train:  0.9754870385659205
Score test:  0.9866613864944062
```




Получаем лучшую точность с использованием весов, что видно не только из результатов тестирования, но и визуально точки из датасета ближе расположены ко второй модели.

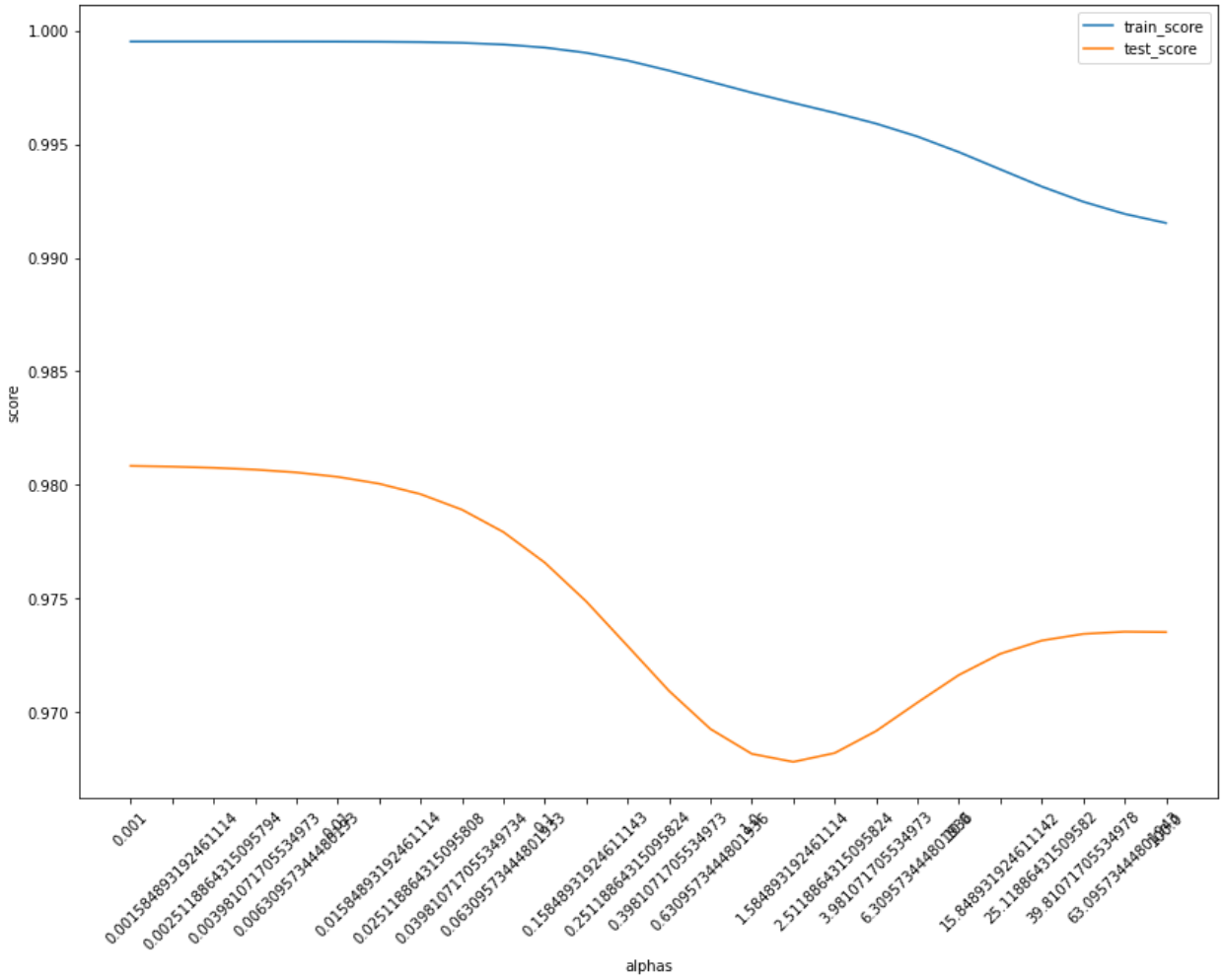
# longley.csv линейная и гребневая регрессия с различными параметрамия

Для линейной регрессии были получены хорошие результаты

```
Score train:  0.9995200489579817
Score test:  0.9808946284511154
```

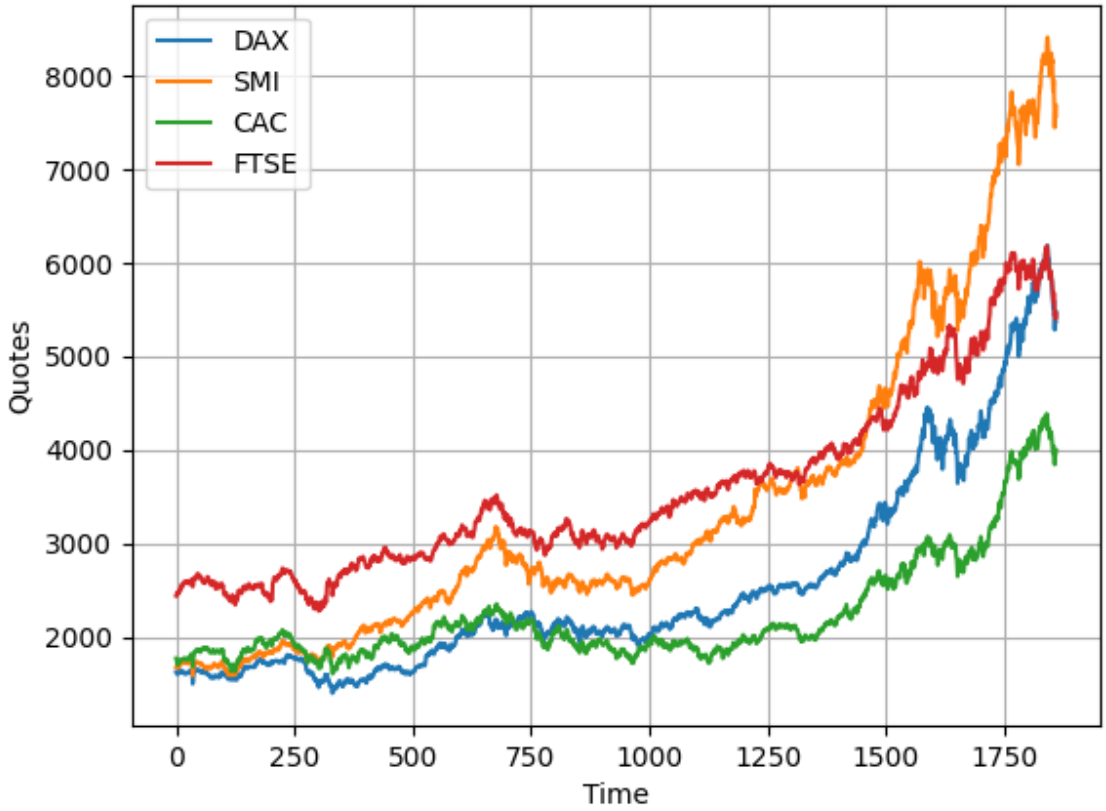
Для гребневой регрессии лучшие результат был получен при $\lambda = 0.001$,

С увеличением $\lambda$ точность падала, параметр $\lambda$ управляет степенью разреженности расчетных коэффициентов, что в данном случае положительно сказывается на точности модели.



# Биржи, оценка динамики

Визуализация данных:



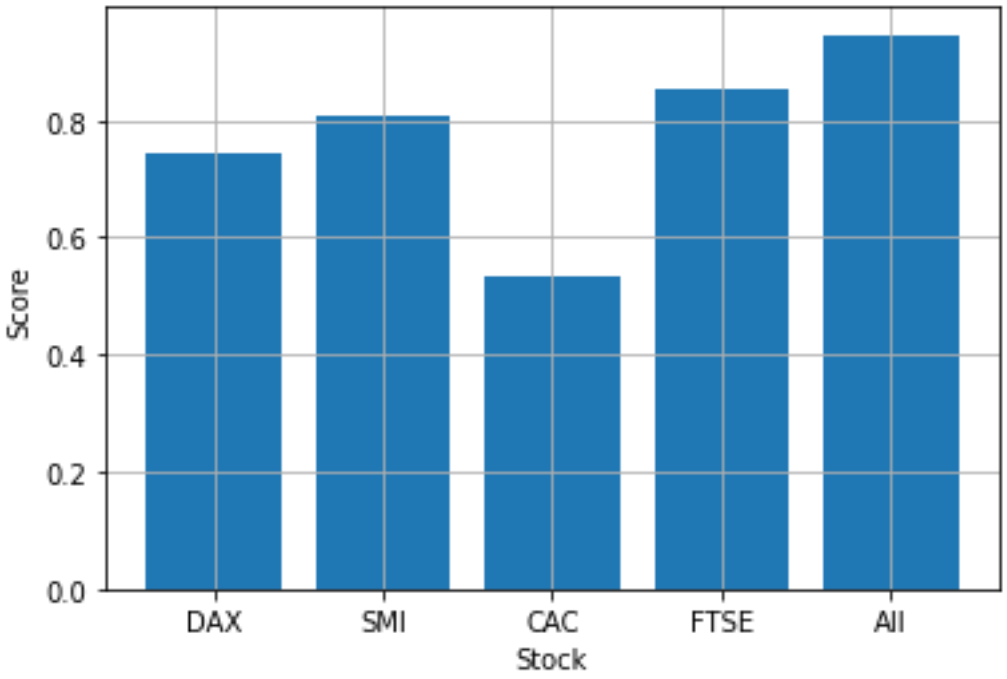

Результаты построения моделей:

```
DAX SMI CAC FTSE
Score train:  0.7196562327811973
Score test:  0.7420498574298932
Coefficient:  [[1.69755735]
 [2.72945159]
 [0.76824345]
 [1.6589993 ]]
DAX
Score train:  0.7352550637340787
Score test:  0.7262356688902087
Coefficient:  [1.73922482]
SMI
Score train:  0.7973422943443509
Score test:  0.7839467362764962
Coefficient:  [2.78436565]
CAC
Score train:  0.5289919587273235
Score test:  0.5337604070229254
Coefficient:  [0.780623]
```

```
FTSE
Score train:  0.8447897453568441
Score test:  0.8573937756008436
Coefficient:  [1.66978873]
```

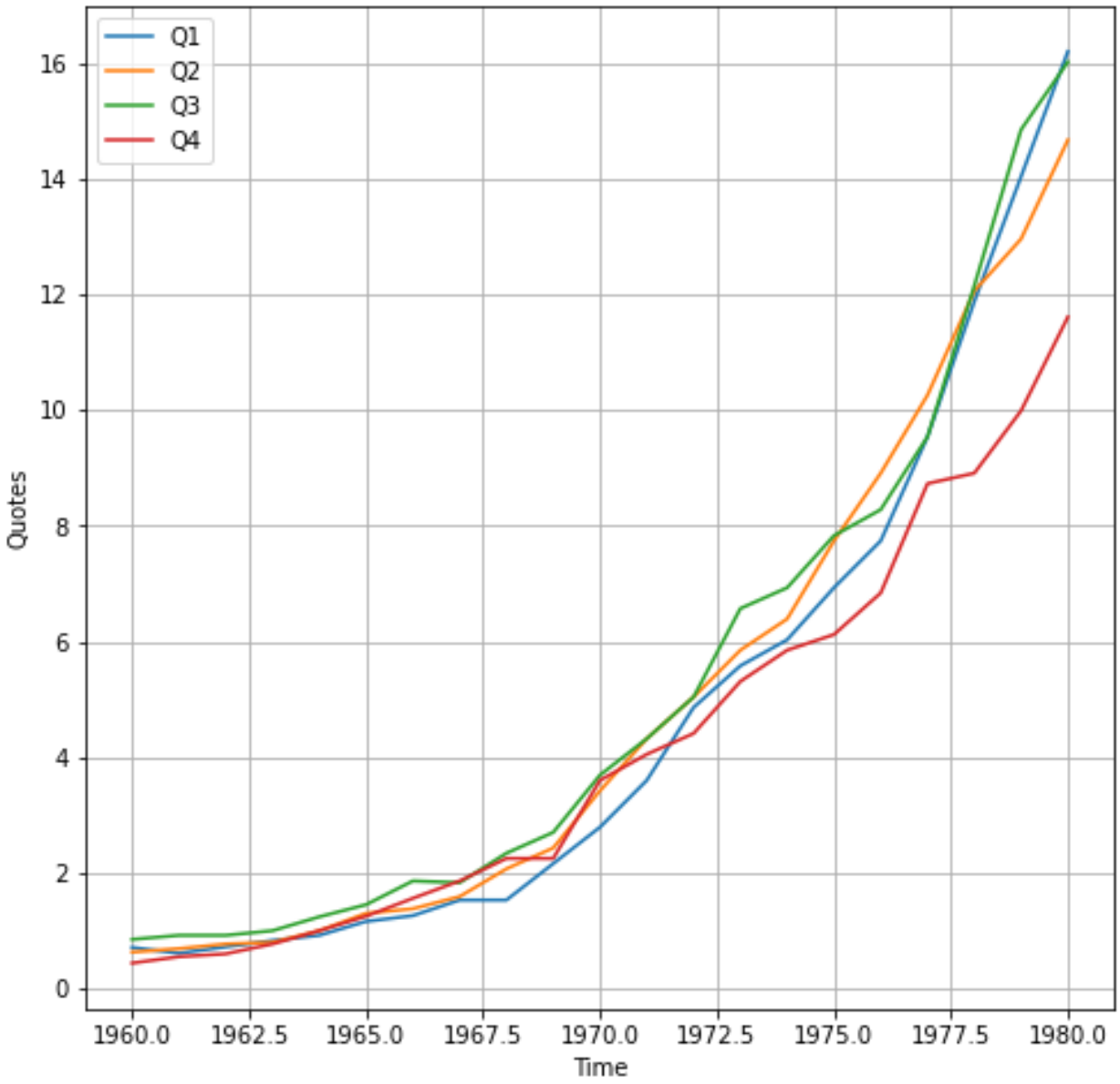
Наибольшую динамику имеет биржа SMI.

Также получил значение score для всех вариантов бирж.



# Johnson & Johnson

Визуализация данных:




Результаты построения моделей:

```
Q1
Score:  0.5266411475370645
Coefficient:  0.7977442865966647
index 2016:  41.32875602223589
Q2
Score:  0.714329293173791
Coefficient:  0.7652810376775787
index 2016:  39.95308214947477
Q3
Score:  0.5758223251205876
Coefficient:  0.8062297714638668
index 2016:  42.186190240889346
Q4
Score:  0.8234939032114907
Coefficient:  0.5825299567634343
index 2016:  30.834954910438455
```

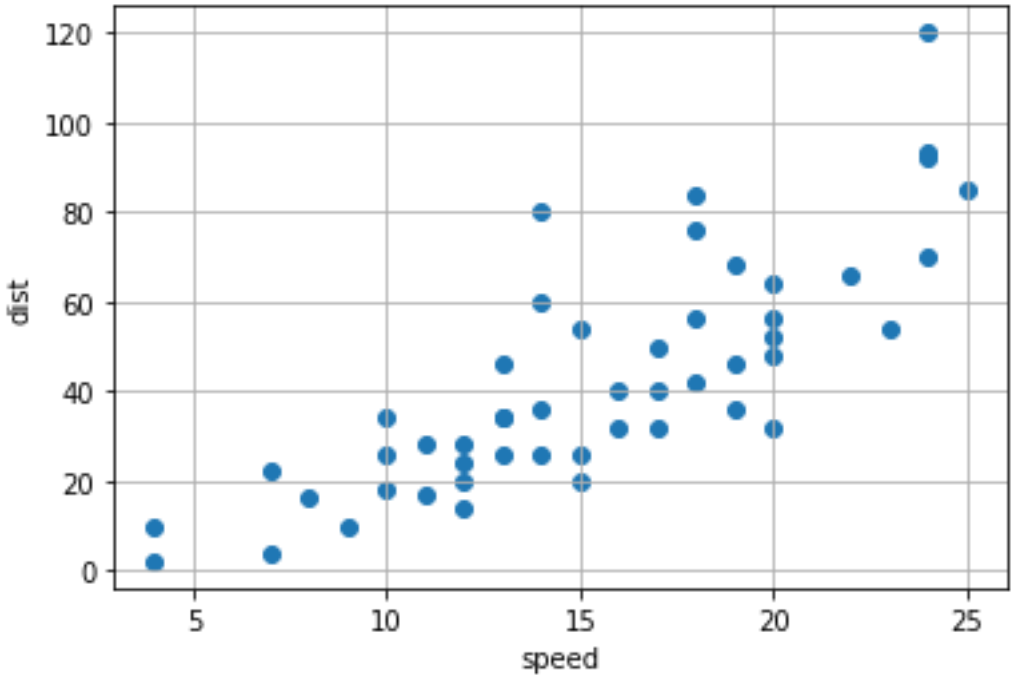
```
All
Score:  0.674666917666243
Coefficient:  2.9517850525015445
index 2016:  154.30298332303846
```

На 2016 год модель предсказывает индекс 154.3.

Наибольшая динамика наблюдается в третьем квартале, наименьшая в четвертом.

## Прогнозирование тормозного пути

Визуализация данных:




При построении модели линейной регрессии были получены следующие результаты:

```
Score:  0.6157734182280231
40 m/h dist:  133.29360796370747
```

Стоит отметить, что данные для модели имеют нелинейных характер и поэтому получаем не самую большую точность. Стоит использовать другие регрессионные модели.

## Регрессия на опорных векторах

Визуализация данных:

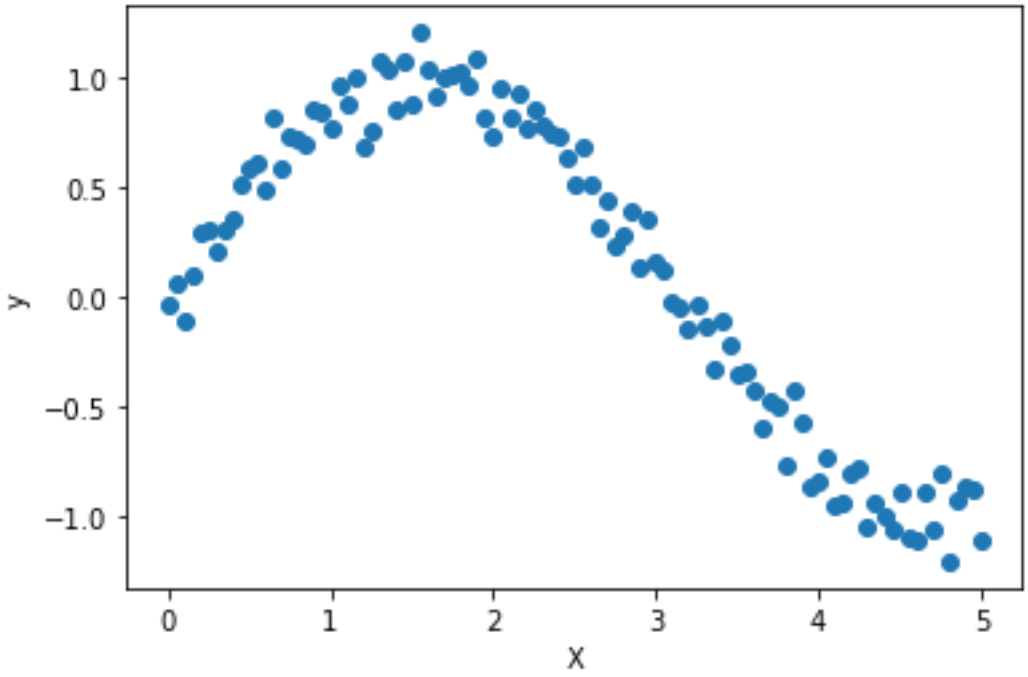
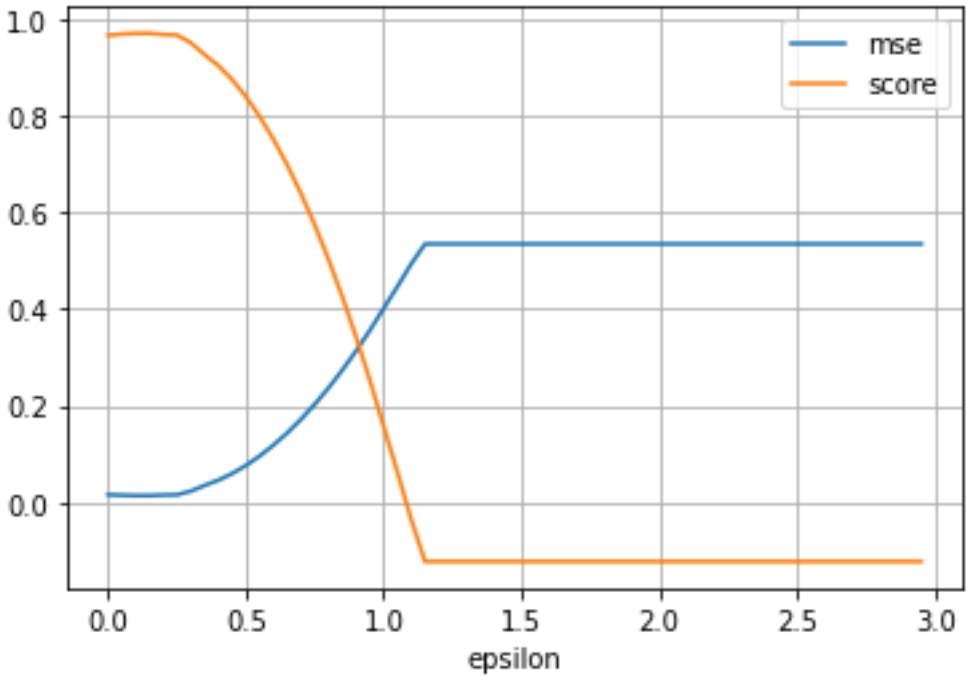



График зависимость среднеквадратичной ошибки от значения параметра ε




По результатам мы видим, что точность наблюдается наилучшей точность достигается при ε= 0, до ε = 1.2 точность падает, а затем перестает изменяться. Также и с квадратом ошибок, они тесно связаны.

## Различные модели

При запуске программы со значениями по умолчанию для каждой модели были получены следующие результаты

```
Linear regression
Score: 0.6065627352621804
Decision tree
Score: 0.19633749301789982
SVR
Score: 0.012802347710630158
```

Лучше всех справилась модель с линейной регрессией, регрессионное дерево решений дает плохой результат, SVR вовсе не обучается на приведенных данных.

Для данного датасета будет выбрана модель с LinearRegression().

# Приложения

## 1_reglab.py

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:



import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')



# In[2]:



data = pd.read_table('reglab1.txt')
variants = [
    (['x'], ['y', 'z']),
    (['y'], ['x', 'z']),
    (['z'], ['y', 'x'])
]



# In[3]:
```

```
score = []

for pred, rel in variants:
    train_X, test_X, train_y, test_y = train_test_split(data[rel], data[pred],
train_size=0.8)

    reg = LinearRegression().fit(train_X, train_y)
    score.append((pred[0], reg.score(test_X, test_y)))



# In[6]:



score
```

## 2_features.py

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:



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

```
get_ipython().run_line_magic('matplotlib', 'inline')


# In[4]:


data = pd.read_table('reglab.txt')
variants = [
    ['x1', 'x2', 'x3', 'x4'],
    ['x1', 'x2', 'x3'],
    ['x1', 'x2', 'x4'],
    ['x1', 'x3', 'x4'],
    ['x2', 'x3', 'x4'],
    ['x1', 'x2'],
    ['x1', 'x3'],
    ['x2', 'x3'],
    ['x1', 'x4'],
    ['x2', 'x4'],
    ['x3', 'x4'],
    ['x1'],
    ['x2'],
    ['x3'],
    ['x4'],
]


# In[12]:
```

```
rss_score = []


for var in variants:
    train_X, test_X, train_y, test_y = train_test_split(data[var], data['y'],
train_size=0.8)


    reg = LinearRegression().fit(train_X, train_y)
    pred_y = reg.predict(test_X)
    rss_score.append((var, mean_squared_error(test_y, pred_y), reg.score(test_X,
test_y)))



# In[17]:



for var, sq, sc in rss_score:
    print(*var)
    print('Residiual sum of squares', sq)
    print('Score', sc)
```

## 3_cygage.py

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:



import numpy as np
```

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')


# In[2]:


data = pd.read_table('cygage.txt')
n = data.to_numpy()


# In[3]:


x_without_w = [[n[i, 1], 1] for i in range(len(n))]
x_with_w = [[n[i, 1], n[i, 2]] for i in range(len(n))]
y = n[:, 0]


# In[4]:


for x in [x_without_w, x_with_w]:
    train_X, test_X, train_y, test_y = train_test_split(x, y, train_size=0.75,
random_state=211)
```

```
    regression = LinearRegression().fit(train_X, train_y)
    print('Score train: ', regression.score(train_X, train_y))
    print('Score test: ', regression.score(test_X, test_y))
    prediction = regression.predict(x)
    plt.plot(n[:,1], prediction)
    plt.scatter(n[:, 1], n[:, 0])
    plt.ylabel("Cal age")
    plt.xlabel("Depth")
    plt.grid(True)
    plt.show()


# In[5]:



plt.scatter(n[:, 1], n[:, 0])
```

## 4_longley.py

```
#!/usr/bin/env python
# coding: utf-8

# In[15]:



import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.model_selection import train_test_split
```

```
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')


# In[13]:


data = pd.read_csv('longley.csv')
data = data.drop('Population', axis=1)

X, y = data.drop('Employed', axis=1), data.Employed
train_X, test_X, train_y, test_y = train_test_split(X, y, train_size=0.5,
random_state=42)


# In[14]:


lin_reg = LinearRegression().fit(train_X, train_y)

print('Score train: ', lin_reg.score(train_X, train_y))
print('Score test: ', lin_reg.score(test_X, test_y))


# In[16]:


alphas = [10**(-3 + 0.2*i) for i in range(26)]
train_score = []
```

```
test_score = []

for alpha in alphas:
    ridge = Ridge(alpha=alpha).fit(train_X, train_y)

    train_score.append(ridge.score(train_X, train_y))
    test_score.append(ridge.score(test_X, test_y))


# In[29]:


plt.figure(figsize=(14, 10))
plt.plot(train_score, label='train_score')
plt.plot(test_score, label='test_score')
plt.xticks(range(len(alphas)), alphas, rotation=45)
plt.xlabel('alphas')
plt.ylabel('score')
plt.legend(loc='best')
plt.show()
```

## 5_eustock.py

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:
```

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')


# In[2]:


data = pd.read_csv('eustock.csv')
y = np.arange(len(data['DAX']))


# In[3]:


variants = [
    'DAX',
    'SMI',
    'CAC',
    'FTSE',
]
scores = []

for var in variants:
    train_X, test_X, train_y, test_y = train_test_split(data[var].to_numpy().reshape(-
1, 1), y, train_size=0.8, random_state=42)
```

```
    reg = LinearRegression().fit(train_X, train_y)
    scores.append(reg.score(test_X, test_y))
    plt.plot(y, data[var], label=var)
    print(var, reg.coef_)

train_X, test_X, train_y, test_y = train_test_split(data, y, train_size=0.8,
random_state=42)
reg = LinearRegression().fit(train_X, train_y)
scores.append(reg.score(test_X, test_y))
print('All', reg.coef_)

plt.legend(loc='best')
plt.show()
variants.append('All')



# In[8]:



plt.bar(variants, scores)
plt.xlabel('Stock')
plt.ylabel('Score')
plt.grid()
plt.show()
```

## 6_JJ.py

```
#!/usr/bin/env python
# coding: utf-8
```

# In[1]:

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
```

# In[2]:

```
data = pd.read_csv('JohnsonJohnson.csv').to_numpy()
Q1 = np.array([[int(data[i,0].split(' ')[0]), data[i, 1]] for i in range(0, len(data), 4)])
Q2 = np.array([[int(data[i,0].split(' ')[0]), data[i, 1]] for i in range(1, len(data), 4)])
Q3 = np.array([[int(data[i,0].split(' ')[0]), data[i, 1]] for i in range(2, len(data), 4)])
Q4 = np.array([[int(data[i,0].split(' ')[0]), data[i, 1]] for i in range(3, len(data), 4)])
All = np.array([[int(data[i,0].split(' ')[0]), data[i:i+4, 1].sum()] for i in range(0,
len(data), 4)])
```

# In[3]:

```
plt.figure(figsize=(8,8))
plt.plot(Q1[:, 0], Q1[:, 1], label="Q1")
plt.plot(Q2[:, 0], Q2[:, 1], label="Q2")
```

```
plt.plot(Q3[:, 0], Q3[:, 1], label="Q3")
plt.plot(Q4[:, 0], Q4[:, 1], label="Q4")
plt.xlabel('Time')
plt.ylabel('Quotes')
plt.grid(True)
plt.legend(loc='best')
plt.show()



# In[4]:



def do(X, y, label):
    train_X, test_X, train_y, test_y = train_test_split(X, y, train_size=0.8,
random_state=42)
    regression = LinearRegression().fit(train_X, train_y)


    print(label)
    print('Score: ', regression.score(test_X, test_y))
    print('Coefficient: ', *regression.coef_)
    print('index 2016: ', *regression.predict(np.array([2016]).reshape(-1, 1)))



# In[5]:



do(Q1[:, 0].reshape(-1, 1), Q1[:, 1], 'Q1')
do(Q2[:, 0].reshape(-1, 1), Q2[:, 1], 'Q2')
do(Q3[:, 0].reshape(-1, 1), Q3[:, 1], 'Q3')
```

```
do(Q4[:, 0].reshape(-1, 1), Q4[:, 1], 'Q4')
do(All[:, 0].reshape(-1, 1), All[:, 1], 'All')
```

## 7_cars.py

```
#!/usr/bin/env python
# coding: utf-8

# In[17]:



import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')



# In[18]:



data = pd.read_csv('cars.csv').to_numpy()
X, y = data[:, 0], data[:, 1]



# In[19]:
```

```
plt.scatter(X, y)
plt.grid(True)
plt.xlabel('speed')
plt.ylabel('dist')
plt.show()


# In[20]:


train_X, test_X, train_y, test_y = train_test_split(X.reshape(-1, 1), y,
train_size=0.8, random_state=42)

reg = LinearRegression().fit(train_X, train_y)

print('Score: ', reg.score(test_X, test_y))
print('40 m/h dist: ', *reg.predict(np.array([40]).reshape(-1, 1)))
```

## 8_SVR.py

```
#!/usr/bin/env python
# coding: utf-8

# In[10]:


import numpy as np
import pandas as pd
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')


# In[5]:


data = pd.read_table('svmdata6.txt').to_numpy()

X = data[:, 0]
y = data[:, 1]


# In[8]:


plt.scatter(X, y)
plt.xlabel('X')
plt.ylabel('y')
plt.show()


# In[9]:


train_X, test_X, train_y, test_y = train_test_split(X, y, train_size=0.8,
random_state=42)
```

```
# In[25]:


mse = []
scores = []

epsilons = np.arange(0, 3, 0.05)

for eps in epsilons:
    reg = SVR(C=1, kernel='rbf', epsilon=eps).fit(train_X.reshape(-1, 1), train_y)

    pred = reg.predict(test_X.reshape(-1, 1))
    mse.append(mean_squared_error(test_y, pred))
    scores.append(reg.score(test_X.reshape(-1, 1), test_y))


# In[26]:


plt.plot(epsilons, mse, label='mse')
plt.plot(epsilons, scores, label='score')
plt.xlabel('epsilon')
plt.legend()
plt.grid(True)
```

## 9_nsv.py

```
#!/usr/bin/env python
```

```
# coding: utf-8

# In[1]:



import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')



# In[2]:



data = pd.read_csv('nsw74psid1.csv')
X, y = data.drop('re78', axis=1), data.re78

train_X, test_X, train_y, test_y = train_test_split(X, y, train_size=0.8,
random_state=42)



# In[3]:
```

```
regressions = [
    ('Linear regression', LinearRegression()),
    ('Decision tree', DecisionTreeRegressor()),
    ('SVR', SVR())
]


# In[4]:


for title, reg in regressions:
    print(title)
    reg.fit(train_X, train_y)
    print('Score:', reg.score(test_X, test_y))
```