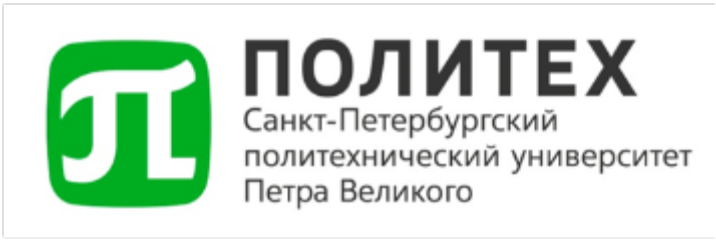
ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и технологий

Высшая школа программной инженерии



ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине «Машинное обучение»

Студент А. М. Потапова
гр. 3530202/90202

Руководитель И. А. Селин

Санкт-Петербург
2022 г

Содержание

Задание 1

Загрузите данные из файла reglab1.txt. Постройте по набору данных регрессии, используя модели с различными зависимыми переменными. Выберите наиболее подходящую модель.

Исходные данные:

	z	x	y
0	2.836772	0.271010	0.308331
1	4.987167	0.589598	0.514913
2	6.412325	0.651744	0.730453
3	4.641998	0.581983	0.461400
4	2.793941	0.463688	0.191102
...	...	...	...
195	2.917468	0.745790	0.073874
196	4.813905	0.474947	0.582424
197	6.419296	0.773870	0.636785
198	4.231917	0.049090	0.994537
199	3.928354	0.180229	0.664546

Наборы переменных - x(y, z), y(x, z), z(x, y), x(y), z(x), y(z)

Для каждого набора зависимых переменных строим регрессионную модель. Воспользуемся функцией score модуля LinearRegression, которая возвращает коэффициент детерминации (чем коэффициент ближе к 1, тем более подходящим будет модуль):

```
linear = LinearRegression()
linear.fit(YZ, X)
print('Determination of x(y, z):', round(linear.score(YZ, X), 10))

linear = LinearRegression()
linear.fit(XZ, Y)
print('Determination of y(x, z):', round(linear.score(XZ, Y), 10))

linear = LinearRegression()
linear.fit(XY, Z)
print('Determination of z(x, y):', round(linear.score(XY, Z), 10))

linear = LinearRegression()
linear.fit(Y.values.reshape(-1, 1), X)
print('Determination of x(y):', round(linear.score(Y.values.reshape(-1, 1), X), 10))

linear = LinearRegression()
linear.fit(Z.values.reshape(-1, 1), Y)
print('Determination of y(z):', round(linear.score(Z.values.reshape(-1, 1), Y), 10))

linear = LinearRegression()
linear.fit(X.values.reshape(-1, 1), Z)
print('Determination of z(x):', round(linear.score(X.values.reshape(-1, 1), Z), 10))
```

Результат:

```
Determination of x(y, z): 0.9186677284
Determination of y(x, z): 0.9505275437
Determination of z(x, y): 0.9686286787
Determination of x(y): 0.0001560505
Determination of y(z): 0.6143421894
Determination of z(x): 0.3659820403
```

Вывод

Исходя из полученных результатов можно заметить, что наиболее подходящей моделью стала модель с зависимостью z(x, y), поскольку значение коэффициента детерминации оказалось очень близко к 1 (0,9686).

Задание 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.

Исходные данные:

	y	x1	x2	x3	x4
0	3.469720	0.233628	0.835549	0.102965	0.457428
1	0.768448	0.117920	0.090544	0.258778	0.283951
2	2.880374	0.091520	0.797592	0.198528	0.699287
3	3.745485	0.876722	0.062935	0.615415	0.176058
4	1.853966	0.207406	0.303490	0.775967	0.667351
...	...	...	...	...	...
195	0.820625	0.163545	0.033599	0.383312	0.645846
196	4.676726	0.983839	0.225527	0.127676	0.757216
197	3.471228	0.661587	0.231570	0.800238	0.609558
198	3.633997	0.828217	0.076275	0.129719	0.965084
199	5.117337	0.830951	0.546032	0.348426	0.629570

Переберем все варианты зависимости переменной y от 1, 2 или 3. Выведем значения коэффициента детерминации для различных вариантов зависимости переменных:

```
linear = LinearRegression()
linear.fit(x1, y)
print('Determination of y(x1):', round(linear.score(x1, y), 10))

linear = LinearRegression()
linear.fit(x2, y)
print('Determination of y(x2):', round(linear.score(x2, y), 10))

linear = LinearRegression()
linear.fit(x3, y)
print('Determination of y(x3):', round(linear.score(x3, y), 10))

linear = LinearRegression()
linear.fit(x4, y)
print('Determination of y(x4):', round(linear.score(x4, y), 10))

linear = LinearRegression()
linear.fit(x1x2, y)
print('Determination of y(x1, x2):', round(linear.score(x1x2, y), 10))

linear = LinearRegression()
linear.fit(x1x3, y)
print('Determination of y(x1, x3):', round(linear.score(x1x3, y), 10))

linear = LinearRegression()
linear.fit(x1x4, y)
print('Determination of y(x1, x4):', round(linear.score(x1x4, y), 10))

linear = LinearRegression()
linear.fit(x2x3, y)
print('Determination of y(x2, x3):', round(linear.score(x2x3, y), 10))

linear = LinearRegression()
linear.fit(x2x4, y)
print('Determination of y(x2, x4):', round(linear.score(x2x4, y), 10))

linear = LinearRegression()
linear.fit(x3x4, y)
print('Determination of y(x3, x4):', round(linear.score(x3x4, y), 10))

linear = LinearRegression()
linear.fit(x1x2x3, y)
print('Determination of y(x1, x2, x3):', round(linear.score(x1x2x3, y), 10))

linear = LinearRegression()
linear.fit(x1x2x4, y)
print('Determination of y(x1, x2, x4):', round(linear.score(x1x2x4, y), 10))

linear = LinearRegression()
linear.fit(x1x3x4, y)
print('Determination of y(x1, x3, x4):', round(linear.score(x1x3x4, y), 10))

linear = LinearRegression()
linear.fit(x2x3x4, y)
print('Determination of y(x2, x3, x4):', round(linear.score(x2x3x4, y), 10))
```

Результаты:

```
Determination of y(x1): 0.6016481121
Determination of y(x2): 0.3203386236
Determination of y(x3): 0.0030028334
Determination of y(x4): 0.0002156613
Determination of y(x1, x2): 0.9986369522
Determination of y(x1, x3): 0.6038415846
Determination of y(x1, x4): 0.601649398
Determination of y(x2, x3): 0.3214796027
Determination of y(x2, x4): 0.3214525377
Determination of y(x3, x4): 0.0030832553
Determination of y(x1, x2, x3): 0.9991581283
Determination of y(x1, x2, x4): 0.9990828715
Determination of y(x1, x3, x4): 0.6038560928
Determination of y(x2, x3, x4): 0.3223763063
```

Вывод

Исходя из полученных результатов можно заметить, что наиболее подходящими зависимостями переменных стали y(x1, x2), y(x1, x2, x3) и y(x1, x2, x4). Оптимальное подмножество признаков - y(x1, x2).

Задание 3

Загрузите данные из файла cygage.txt. Постройте регрессию, выражающую зависимость возраста исследуемых отложений от глубины залегания, используя веса наблюдений. Оцените качество построенной модели.

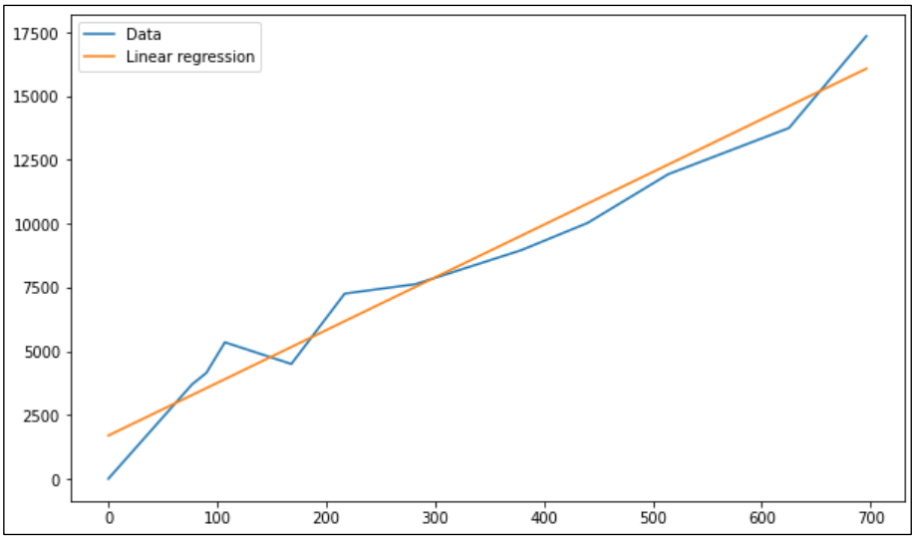
Исходные данные:

	calAge	Depth	Weight
0	0	0	1.0
1	3707	77	0.1
2	4150	90	0.1
3	5350	107	0.1
4	4500	168	0.1
5	7260	217	0.1
6	7630	282	1.0
7	8960	379	0.5
8	10030	440	0.5
9	11940	514	0.5
10	13755	625	0.5
11	17360	696	0.1

Для построения регрессии используем LinearRegression и веса наблюдений:

```
X = data.Depth.values.reshape(-1, 1)
y = data.calAge.values.reshape(-1, 1)
weights = data.Weight.values.reshape(-1, 1)

linear = LinearRegression()
linear.fit(X, y)
```

Результат:



Оценим точность модели:

```
linear.score(X, y)
```

```
0.9592555361125507
```

Вывод

Полученные результаты говорят о достаточно хорошем качестве построенной модели, поскольку коэффициент детерминации оказался равен 0,9592.

Задание 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.

Исключим переменную Population:

```
data = pd.read_csv('/content/drive/MyDrive/longley.csv').drop('Population', axis=1)
```

Исходные данные:

	GNP.deflator	GNP	Unemployed	Armed.Forces	Year	Employed
0	83.0	234.289	235.6	159.0	1947	60.323
1	88.5	259.426	232.5	145.6	1948	61.122
2	88.2	258.054	368.2	161.6	1949	60.171
3	89.5	284.599	335.1	165.0	1950	61.187
4	96.2	328.975	209.9	309.9	1951	63.221
5	98.1	346.999	193.2	359.4	1952	63.639
6	99.0	365.385	187.0	354.7	1953	64.989
7	100.0	363.112	357.8	335.0	1954	63.761
8	101.2	397.469	290.4	304.8	1955	66.019
9	104.6	419.180	282.2	285.7	1956	67.857
10	108.4	442.769	293.6	279.8	1957	68.169
11	110.8	444.546	468.1	263.7	1958	66.513
12	112.6	482.704	381.3	255.2	1959	68.655
13	114.2	502.601	393.1	251.4	1960	69.564
14	115.7	518.173	480.6	257.2	1961	69.331
15	116.9	554.894	400.7	282.7	1962	70.551

Разделим данные на тестовую и обучающую выборки равных размеров случайным образом:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=1479)
```

Построим линейную регрессию по признаку Employed:

```
linear = LinearRegression()
linear.fit(X_train, y_train)
```

```
LinearRegression()
```

Посчитаем для нее среднеквадратичную ошибку на обучающей и тестовой выборке:

```
print('Mean squad error on train data, linear regression:', mean_squared_error(y_train, linear.predict(X_train)))
print('Mean squad error on test data, linear regression:', mean_squared_error(y_test, linear.predict(X_test)))
```

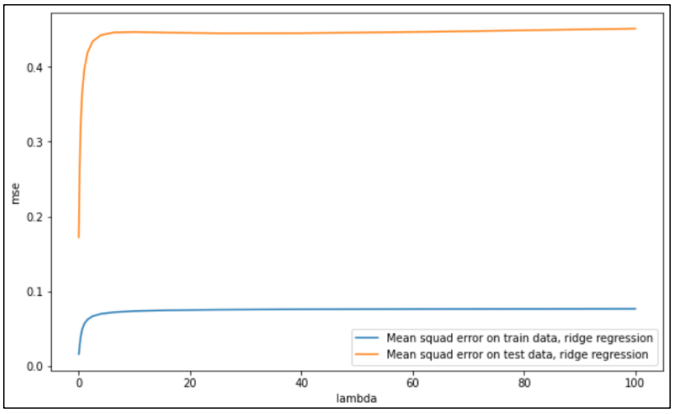
```
Mean squad error on train data, linear regression: 0.016062203299175206
Mean squad error on test data, linear regression: 0.17146444645933206
```

Построим гребневую регрессию для значений $\lambda = 10^{-3+0.2 \cdot i}$, $i = 0,...,25$

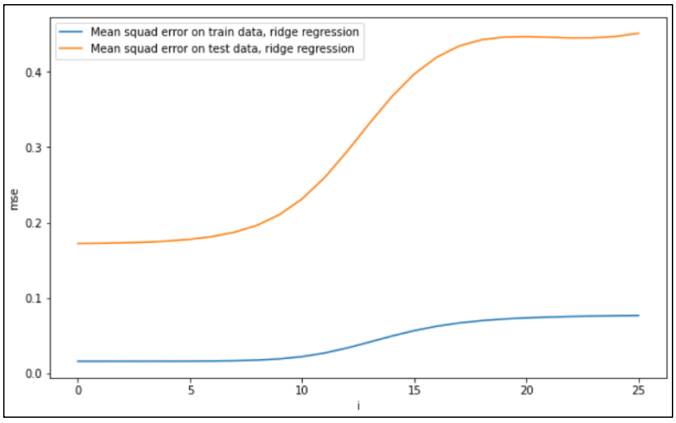
```
i_arr = range(0, 26)
lamda_arr = [10**(-3 + 0.2*i) for i in i_arr]

test_mse = []
train_mse = []
for lamda in lamda_arr:
    ridge = Ridge(alpha=lamda, random_state=0)
    ridge.fit(X_train, y_train)
    test_mse.append(mean_squared_error(y_test, ridge.predict(X_test)))
    train_mse.append(mean_squared_error(y_train, ridge.predict(X_train)))
```

Посчитаем для них среднеквадратичную ошибку на обучающей и тестовой выборке:

Для λ:



Для i:



```
print('Min MSE, train:', min(train_mse))
print('Min MSE, test',  min(test_mse))
```

```
Min MSE, train: 0.016063512046953478
Min MSE, test 0.17207525840134216
```

Вывод

Исходя из полученных результатов замечаем, что при увеличении параметра λ ошибка как на обучающей, так и на тестовой выборках для гребневой регрессии увеличивается. Линейная регрессия показала себя лучше как на обучающей, так и на тестовой выборках, т. к. ее среднеквадратичная ошибка была меньше.

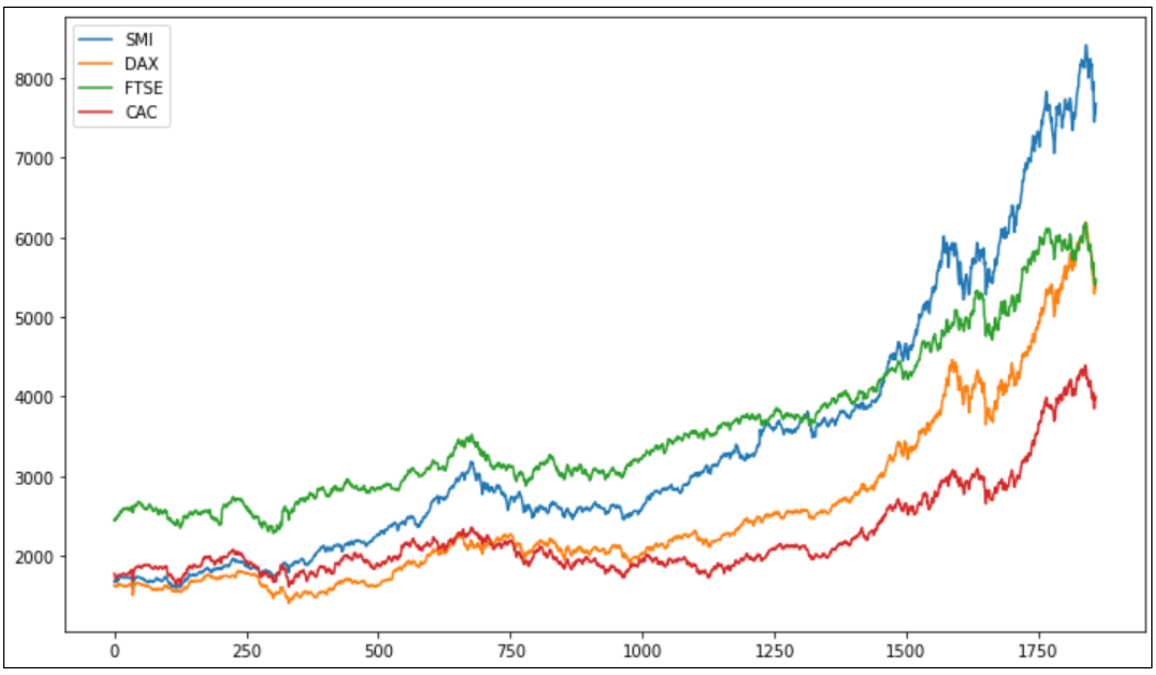
Задание 5

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.

Исходные данные:

	DAX	SMI	CAC	FTSE
0	1628.75	1678.1	1772.8	2443.6
1	1613.63	1688.5	1750.5	2460.2
2	1606.51	1678.6	1718.0	2448.2
3	1621.04	1684.1	1708.1	2470.4
4	1618.16	1686.6	1723.1	2484.7
...	...	...	...	...
1855	5460.43	7721.3	3939.5	5587.6
1856	5285.78	7447.9	3846.0	5432.8
1857	5386.94	7607.5	3945.7	5462.2
1858	5355.03	7552.6	3951.7	5399.5
1859	5473.72	7676.3	3995.0	5455.0

Построим на одном графике все кривые изменения котировок во времени:

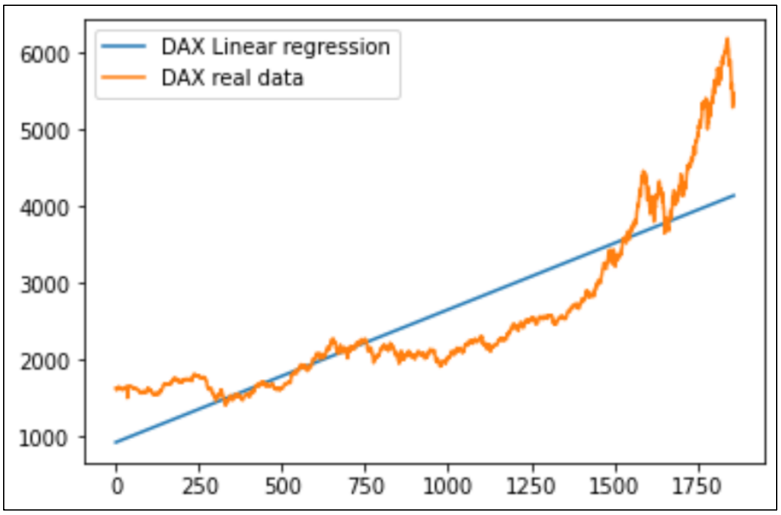


Построим линейную регрессию для каждой модели в отдельности и для всех моделей вместе:

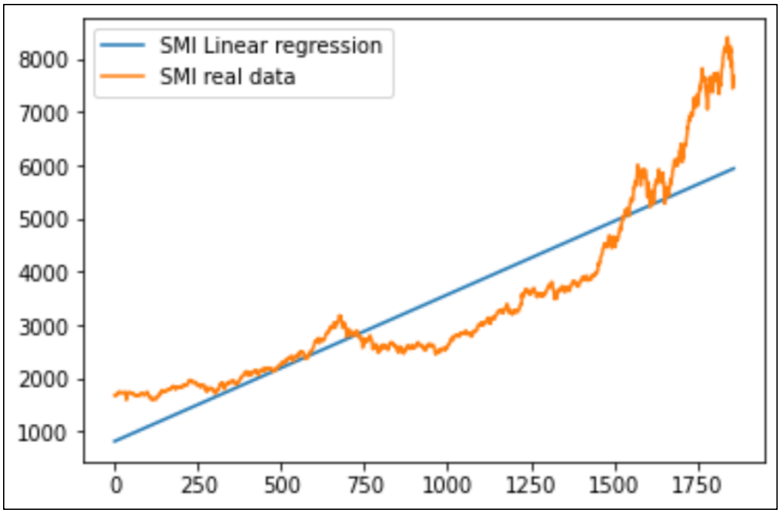
```
linear_1 = LinearRegression()
linear_1.fit(data.index.values.reshape(-1, 1), DAX.values.reshape(-1, 1))
linear_2 = LinearRegression()
linear_2.fit(data.index.values.reshape(-1, 1), SMI.values.reshape(-1, 1))
linear_3 = LinearRegression()
linear_3.fit(data.index.values.reshape(-1, 1), CAC.values.reshape(-1, 1))
linear_4 = LinearRegression()
linear_4.fit(data.index.values.reshape(-1, 1), FTSE.values.reshape(-1, 1))
linear_avg = LinearRegression()
linear_avg.fit(data.index.values.reshape(-1, 1), avg_data.reshape(-1, 1))
linear_sum = LinearRegression()
linear_sum.fit(data.index.values.reshape(-1, 1), sum_data.values.reshape(-1, 1))
```

```
LinearRegression()
```

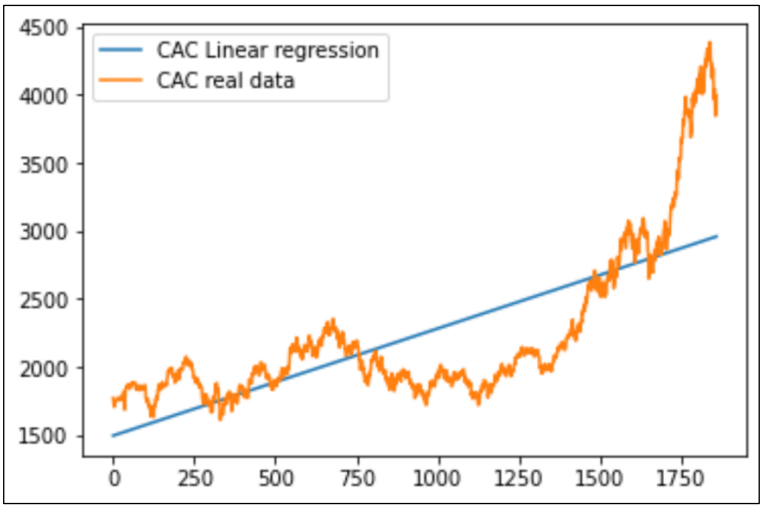
- Линейная регрессия для DAX



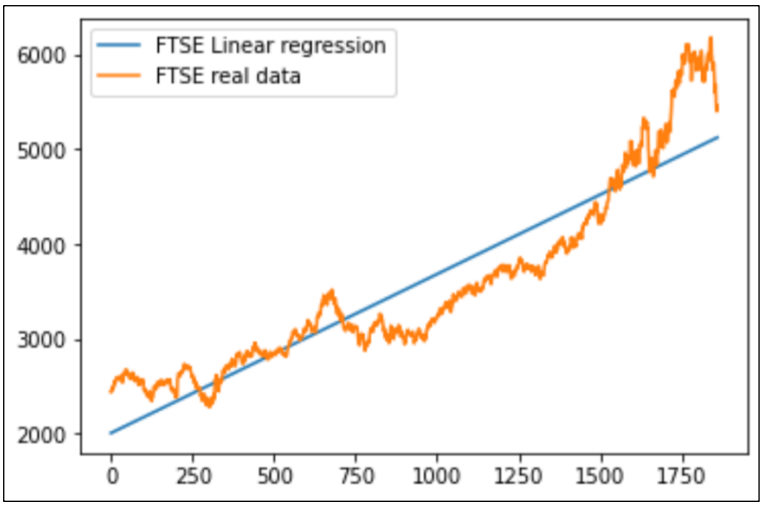
- Линейная регрессия для SMI



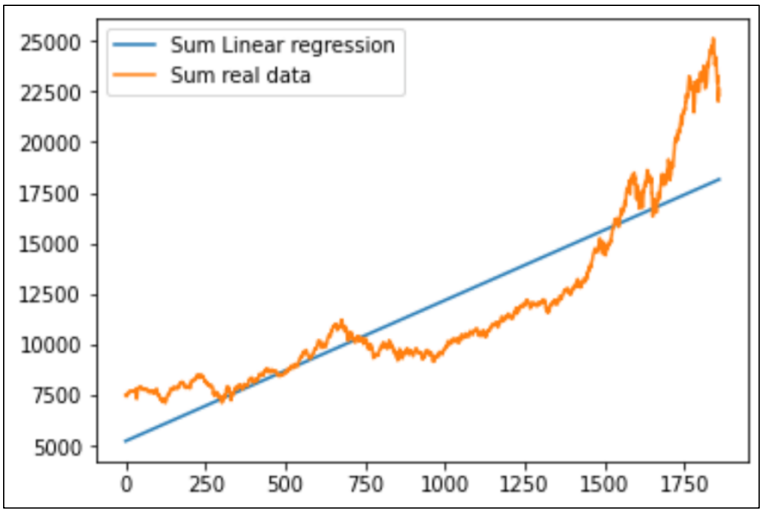
- Линейная регрессия для CAC



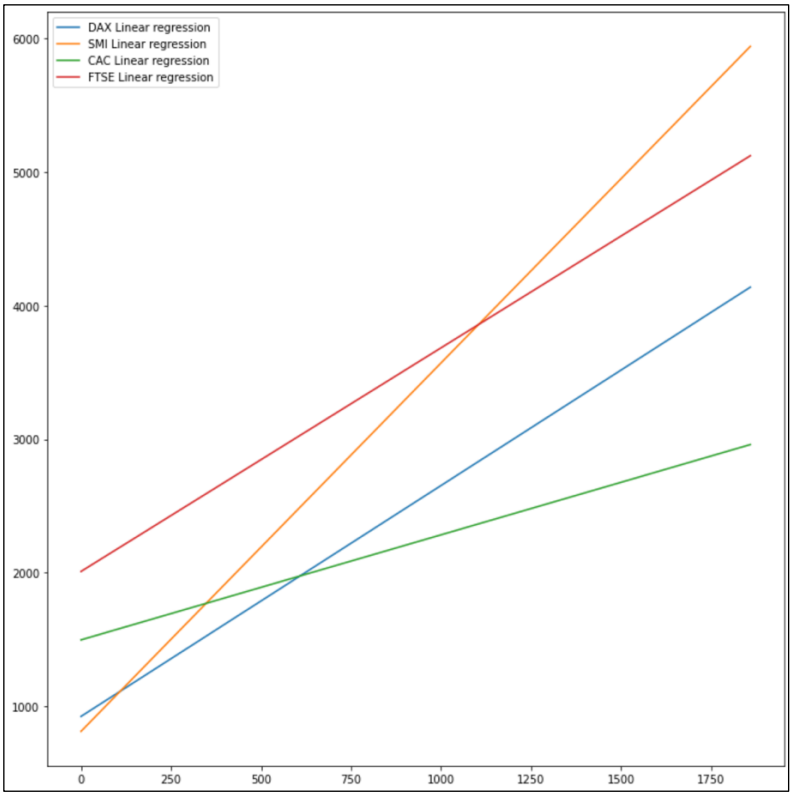
- Линейная регрессия для FNSE



- Линейная регрессия для всех моделей:



Отобразим линейные регрессии всех моделей по-отдельности:



Вывод

Исходя из полученных результатов замечаем, что биржа SMI имеет наибольшую динамику.

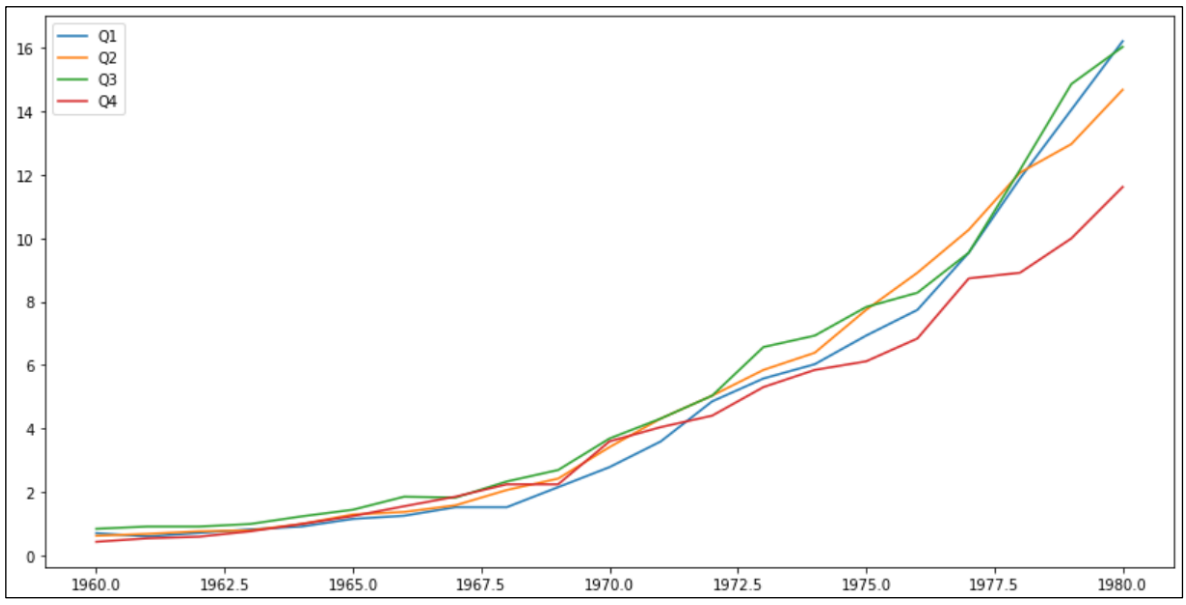
Задание 6

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.

Исходные данные:

	index	value
0	1960 Q1	0.71
1	1960 Q2	0.63
2	1960 Q3	0.85
3	1960 Q4	0.44
4	1961 Q1	0.61
...	...	...
79	1979 Q4	9.99
80	1980 Q1	16.20
81	1980 Q2	14.67
82	1980 Q3	16.02
83	1980 Q4	11.61

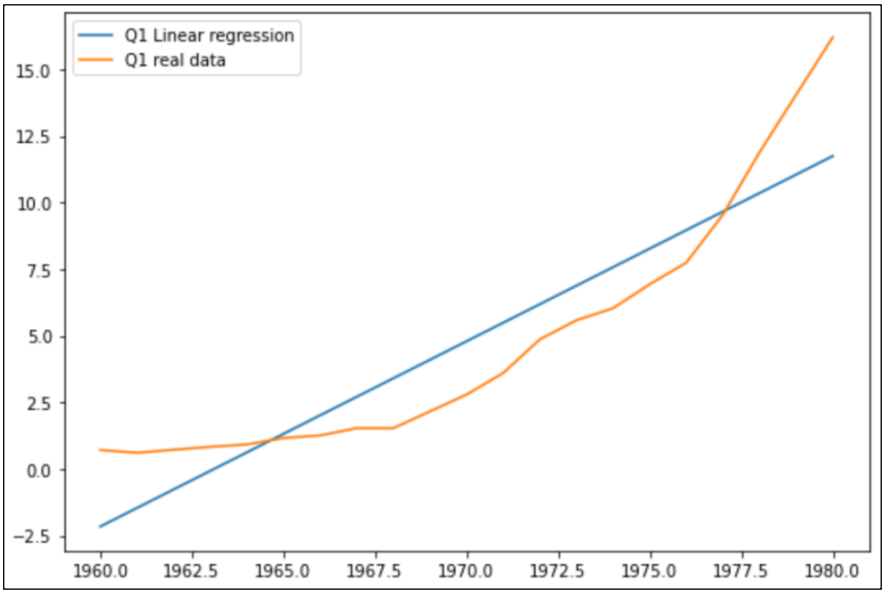
Построим на одном графике все кривые изменения прибыли во времени:



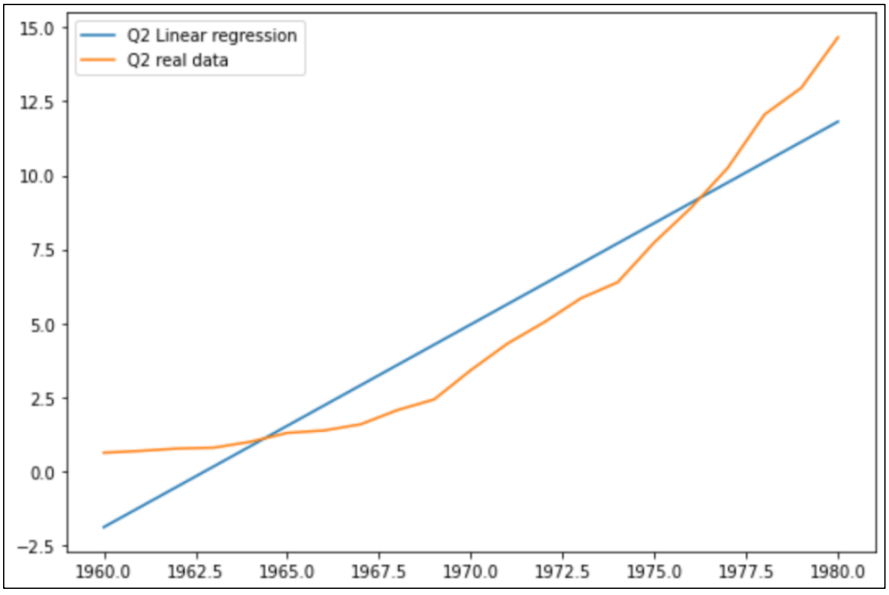
Построим линейную регрессию для каждого квартала в отдельности и для всех кварталов вместе:

```
linear_1 = LinearRegression()
linear_1.fit(Q1.years.values.reshape(-1, 1), Q1.value)
linear_2 = LinearRegression()
linear_2.fit(Q2.years.values.reshape(-1, 1), Q2.value)
linear_3 = LinearRegression()
linear_3.fit(Q3.years.values.reshape(-1, 1), Q3.value)
linear_4 = LinearRegression()
linear_4.fit(Q4.years.values.reshape(-1, 1), Q4.value)
linear_avg = LinearRegression()
linear_avg.fit(avg_data.years.values.reshape(-1, 1), avg_data.value)
```

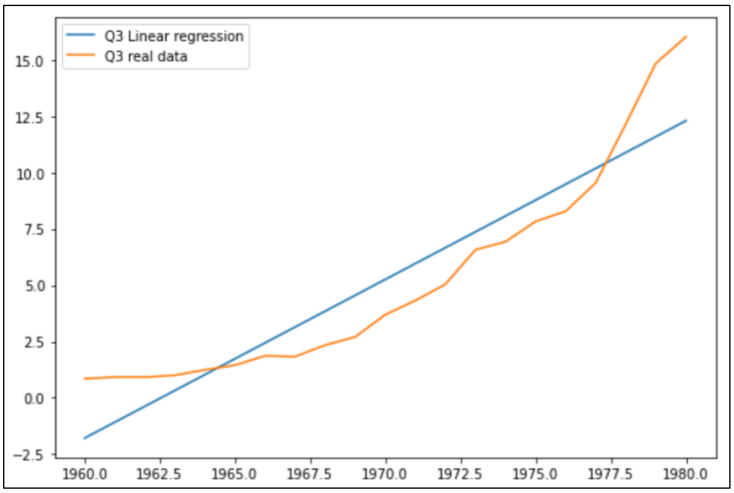
- Линейная регрессия для Q1:



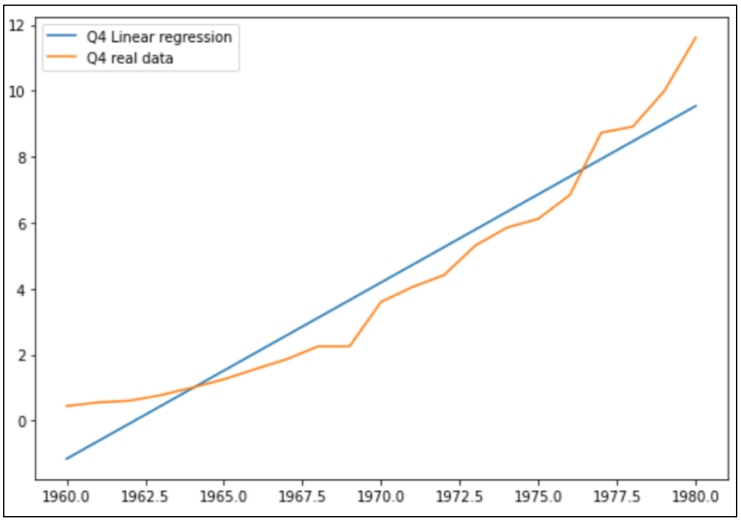
- Линейная регрессия для Q2:



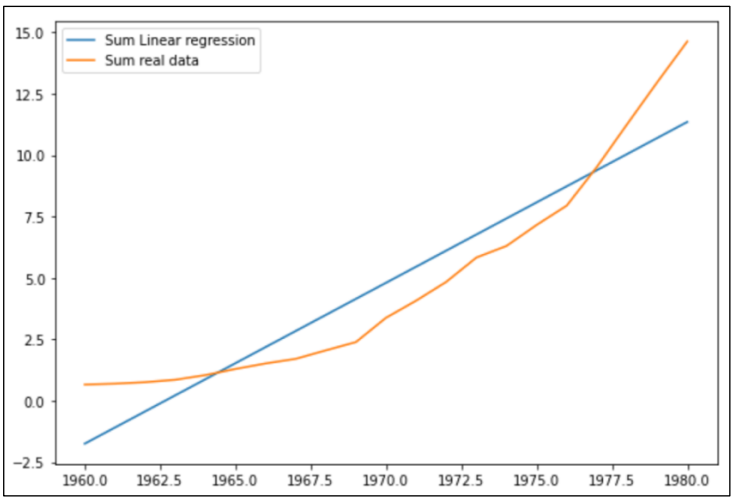
- Линейная регрессия для Q3:



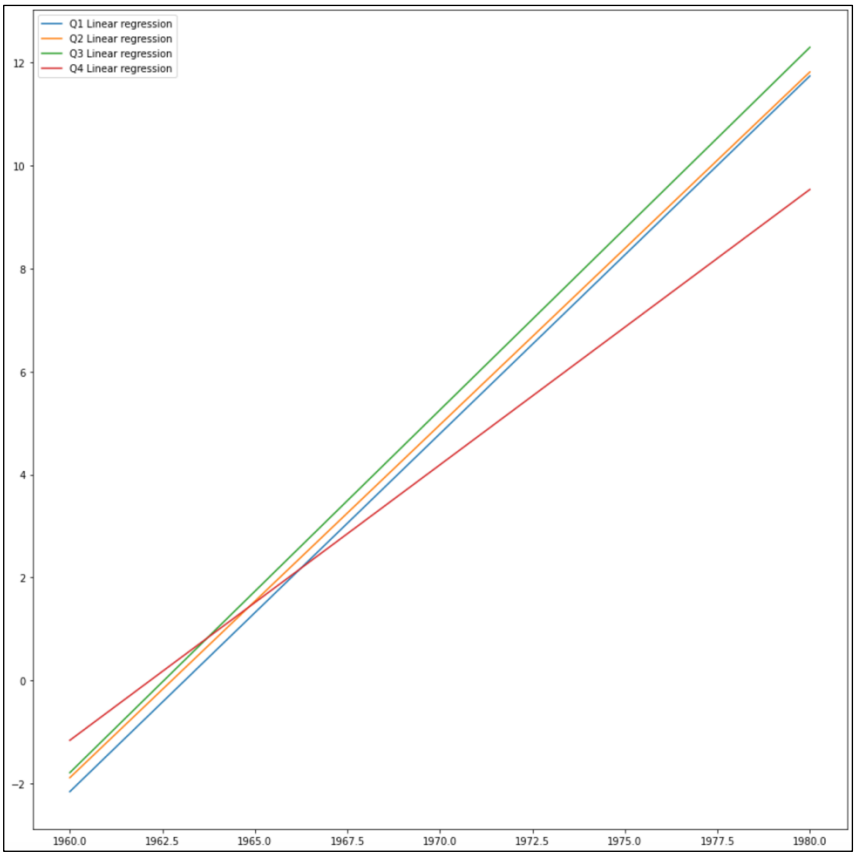
- Линейная регрессия для Q4:



- Линейная регрессия для всех кварталов вместе:



- Линейная регрессия для всех кварталов по-отдельности:



Сделаем прогноз по прибыли в 2016 году во всех кварталах и в среднем по году:

```
Profit forecast in 2016, Q1: 36.75963636363622
Profit forecast in 2016, Q2: 36.48945454545469
Profit forecast in 2016, Q3: 37.653939393939254
Profit forecast in 2016, Q4: 28.79391341991345
Profit forecast in 2016, avg: 34.924235930735904
```

Вывод

Исходя из полученных результатов замечаем, что в квартале Q3 компания имеет наибольшую динамику доходности, а в Q4 – наименьшую. Прогноз соответствует увиденной динамике.

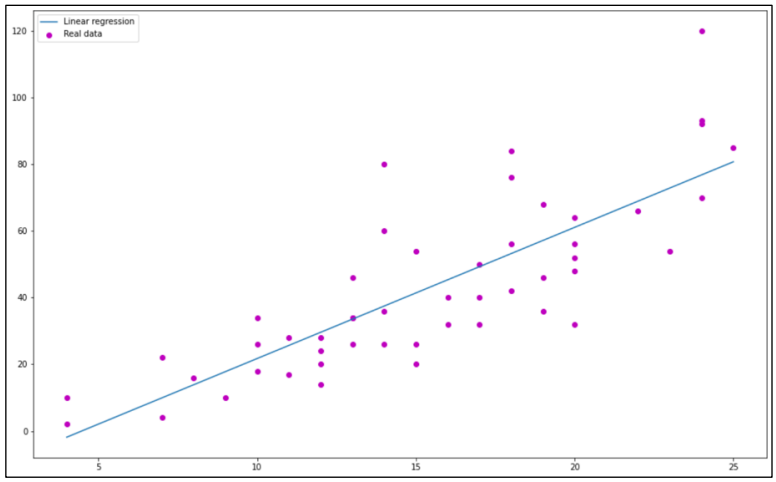
Задание 7

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.

Исходные данные:

	speed	dist
0	4	2
1	4	10
2	7	4
3	7	22
4	8	16
5	9	10
6	10	18
7	10	26
8	10	34
9	11	17
10	11	28

Линейная регрессия:



Оценка длины тормозного пути при скорости 40 миль в час:

```
X_predict = np.array(40).reshape(1, -1)
print('Prediction of distance with speed=40:',
      linear.predict(X_predict)[0][0])
```

```
Prediction of distance with speed=40: 139.7172554744526
```

Вывод

При помощи регрессионной модели мы оценили длину тормозного пути при скорости 40 миль в час. Она составила 139,7172 фута.

Задание 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.

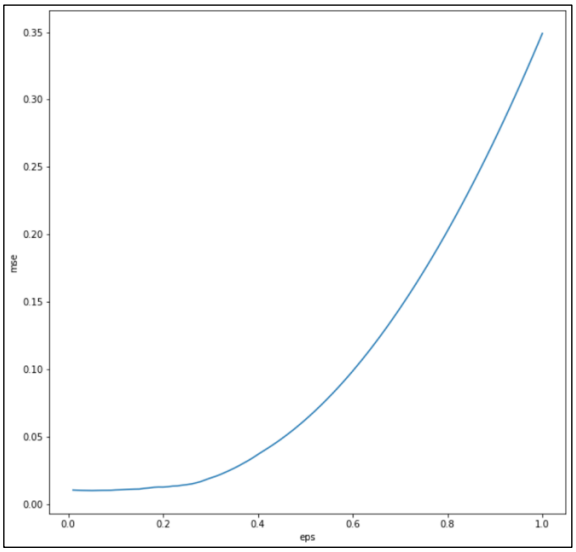
Исходные данные:

	X	Y
1	0.00	-0.035662
2	0.05	0.059784
3	0.10	-0.111344
4	0.15	0.099728
5	0.20	0.297808
...	...	...
97	4.80	-1.199663
98	4.85	-0.926104
99	4.90	-0.855847
100	4.95	-0.867529
101	5.00	-1.098171

Построим регрессионный алгоритм метода опорных векторов (sklearn.svm.SVR) с параметром C = 1, используя ядро "rbf".

```
mse = []
eps_range = np.arange(0.01, 1.01, 0.01)
for eps in eps_range:
    svr = SVR(epsilon=eps)
    svr.fit(X, y)
    mse.append(mean_squared_error(y, svr.predict(X)))
```

Графике зависимости среднеквадратичной ошибки на обучающей выборке от значения параметра ε:



Вывод

Исходя из полученного результата мы видим, что чем меньше значения параметра ε, тем меньше среднеквадратичная ошибка. Это происходит из-за того, что данный параметр отвечает за допустимое отклонение. Следовательно, чем больше отклонение, тем хуже регрессия.

Задание 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.

Исходные данные:

	trt	age	educ	black	hisp	marr	nodeg	re74	re75	re78
0	0	47	12	0	0	0	0	0.00	0.00	0.000
1	0	50	12	1	0	1	0	0.00	0.00	0.000
2	0	44	12	0	0	0	0	0.00	0.00	0.000
3	0	28	12	1	0	1	0	0.00	0.00	0.000
4	0	54	12	0	0	1	0	0.00	0.00	0.000
...	...	...	...	...	...	...	...	...	...	...
2670	1	33	12	1	0	1	0	20279.95	10941.35	15952.600
2671	1	25	14	1	0	1	0	35040.07	11536.57	36646.950
2672	1	35	9	1	0	1	1	13602.43	13830.64	12803.970
2673	1	35	8	1	0	1	1	13732.07	17976.15	3786.628
2674	1	33	11	1	0	1	1	14660.71	25142.24	4181.942

Построим регрессионное дерево (sklearn.tree.DecisionTreeRegressor) для признака re78:

```
parameters = {'criterion': ['mse', 'friedman_mse', 'mae'],
              'max_depth': range(1, 20),
              'min_samples_leaf': range(1, 20),
              'min_samples_split': range(1, 20)}
tree = DecisionTreeRegressor()
rand_search_cv_tree = RandomizedSearchCV(tree, parameters, cv=5, n_jobs=-1, n_iter=500)
rand_search_cv_tree.fit(X_train, y_train)
best_tree = rand_search_cv_tree.best_estimator_
print('Score of DecisionTreeRegressor:', best_tree.score(X_test, y_test))
```

Получили следующее качество:

```
Score of DecisionTreeRegressor: 0.5597624440847127
```

Построим линейную регрессионную модель и SVM-регрессию для этого набора данных:

```
linear = LinearRegression()
linear.fit(X_train, y_train)
print('Score of Linear regression model:', linear.score(X_test, y_test))
```

```
svr = SVR()
rand_search_cv_svc = RandomizedSearchCV(svr, {'C': range(1, 500), 'epsilon' : np.arange(0.01, 1, 0.01)},
                                        cv=5, n_jobs=-1, n_iter=500)
rand_search_cv_svc.fit(X_train, y_train)
best_svr = rand_search_cv_svc.best_estimator_
print('Score of SVR:', best_svr.score(X_test, y_test))
```

Получили следующее качество:

```
Score of Linear regression model: 0.5967892230342389
```

```
Score of SVR: 0.4771195991021452
```

Вывод

Исходя из полученных результатов мы видим, что оптимальной моделью можно считать регрессионную модель, т. к. ее значение коэффициента детерминации ближе к 1.