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

Лабораторная работа №7

по дисциплине «Машинное обучение»

Выплолнил студент
гр. 33534/5 Стойкоски Н.С.

Руководитель И.А. Селин

Санкт-Петербург
2019 г.

Оглавление

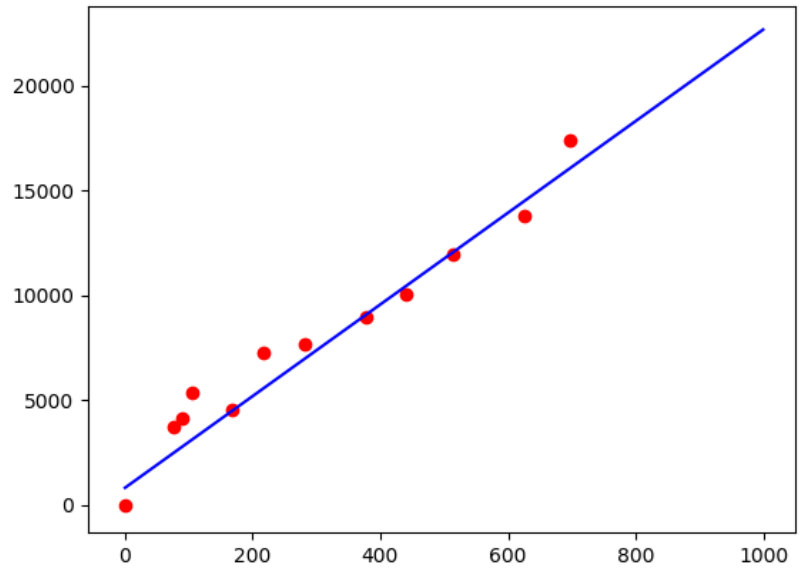
Постановка задачи

1. Загрузите данные из файла cygage.txt. Постройте регрессию, выражающую зависимость возраста исследуемых отложений от глубины залегания, используя веса наблюдений. Оцените качество построенной модели.
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 longley.csv. Данные состоят из 7 экономических переменных, наблюдаемых с 1947 по 1962 годы (*n=16*):
GNP.deflator - дефлятор цен,
GNP - валовой национальный продукт,
Unemployed – число безработных
Armed.Forces – число людей в армии
Population – население, возраст которого старше 14 лет
Year - год
Employed – количество занятых
Построить регрессию по признаку Employed .
Исключите из набора данных longley переменную "Population". Разделите данные на тестовую и обучающую выборки равных размеров случайным образом. Постройте гребневую регрессию для значений $\lambda = 10^{-3+0.2 \cdot i}$, $i = 0,...,25$, подсчитайте ошибку на тестовой и обучающей выборке для данных значений λ, постройте графики. Объясните полученные результаты.
4. 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.
5. 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.
6. Загрузите данные sunspot.year.csv. Данные содержат количество солнечных пятен с 1700 по 1988 гг. Постройте на графике кривую изменения числа солнечных пятен во времени. Постройте линейную регрессию для данных.
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

1. Была построена регрессия выражающую зависимость возраста исследуемых отложений от глубины залегания, используя веса наблюдений

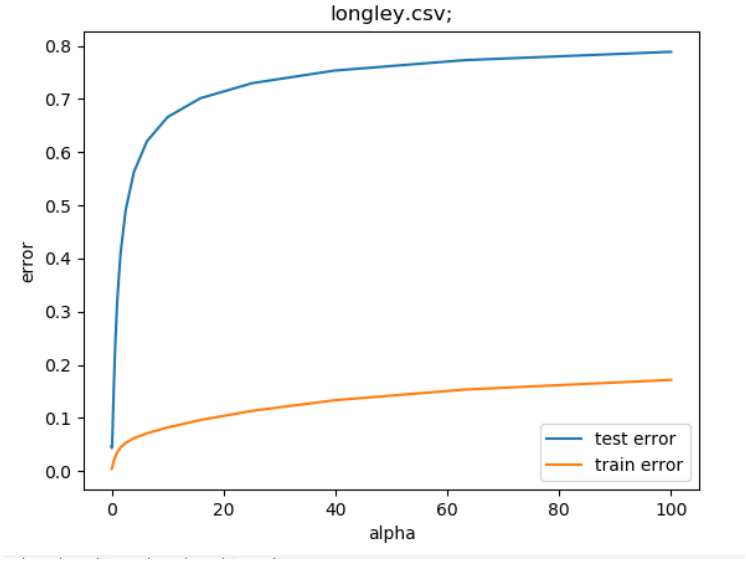



```
LineaerRegression:
 score = 0.973
 coef_ = [21.9091861]
 intercept = 784.569
```

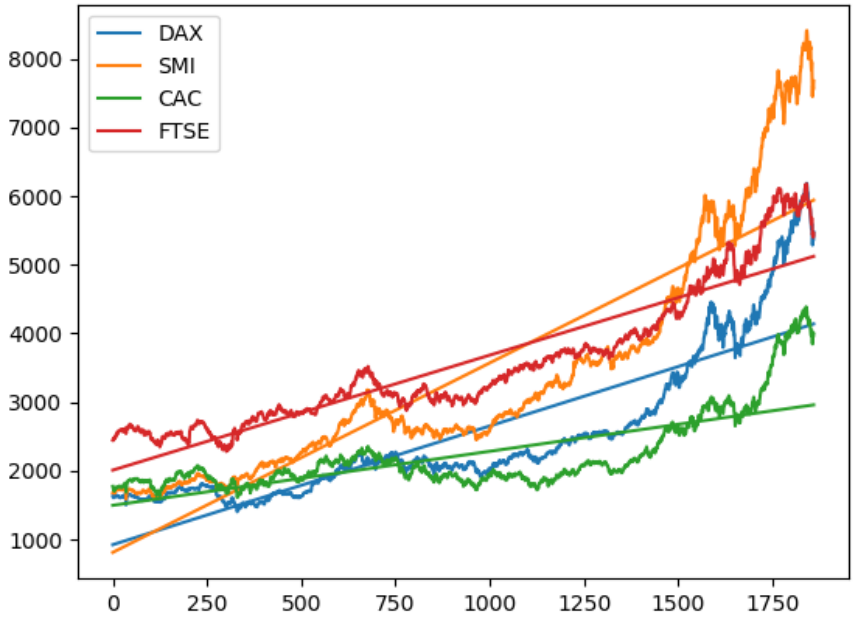
2. Был реализован алгоритм для уменьшения количества признаков, используемых для построения регресии. Было найдено, что оптимальное подмножество содержит всех признаков потому что так минимизиеруется остаточная сумма квадратов.

```
idx_comb = (0,), RSS = 157.22
idx_comb = (1,), RSS = 268.25
idx_comb = (2,), RSS = 393.49
idx_comb = (3,), RSS = 394.59
idx_comb = (0, 1), RSS = 0.54
idx_comb = (0, 2), RSS = 156.35
idx_comb = (0, 3), RSS = 157.22
idx_comb = (1, 2), RSS = 267.80
idx_comb = (1, 3), RSS = 267.81
idx_comb = (2, 3), RSS = 393.46
idx_comb = (0, 1, 2), RSS = 0.33
idx_comb = (0, 1, 3), RSS = 0.36
idx_comb = (0, 2, 3), RSS = 156.35
idx_comb = (1, 2, 3), RSS = 267.44
idx_comb = (0, 1, 2, 3), RSS = 0.19
best features: [0, 1, 2, 3], score = 0.9995
```

3. Данные из набора Longley после исключения переменной “Population” были разделены на тестовую и обучающую выборки равных размеров и была построена гребневая регресия для значений $\lambda = 10^{-3+0.2 \cdot i}$, $i = 0,...,25$. Был построен соответствующий график отображающий ошибку на тестовой и обучающей выборке для данных значений λ. Значение λ используется для уменьшение эффекта переобучения т.е. с увеличением значения λ сложность модели уменьшается. Однако данный набор содержит довольно малое количество экземпляров (n=16), и увеличение этого параметра только приводит к недообучение.

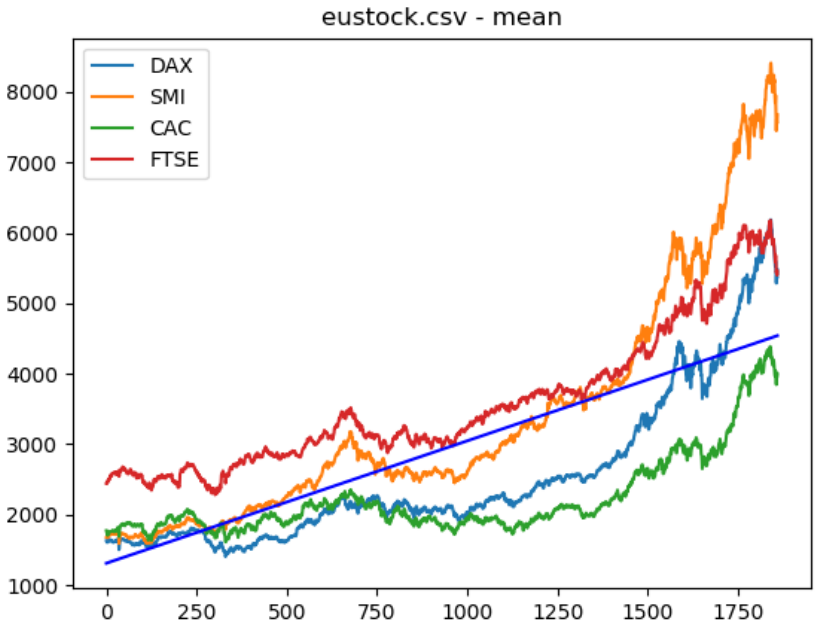


4. Для набора eustock.csv была построена линейная регрессия для каждой модели в отдельности и для всех моделей вместе.

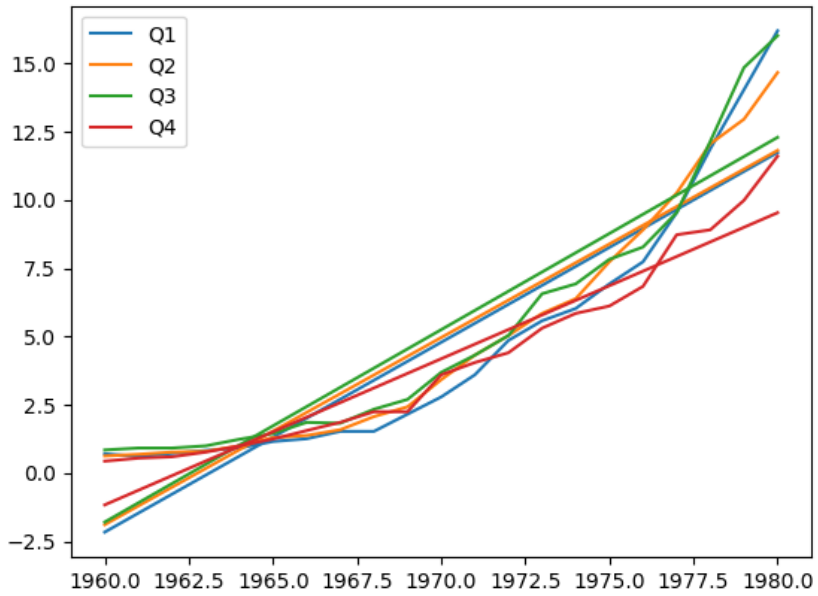


По графику можно сделать вывод что биржа SMI имеет наибольшую динамику. График биржы SMI так же имеет найболее высокое отклонение от нашей линейной модели:

```
data = DAX rss = 5.838e+08
data = SMI rss = 1.057e+09
data = CAC rss = 2.941e+08
data = FTSE rss = 2.693e+08
```

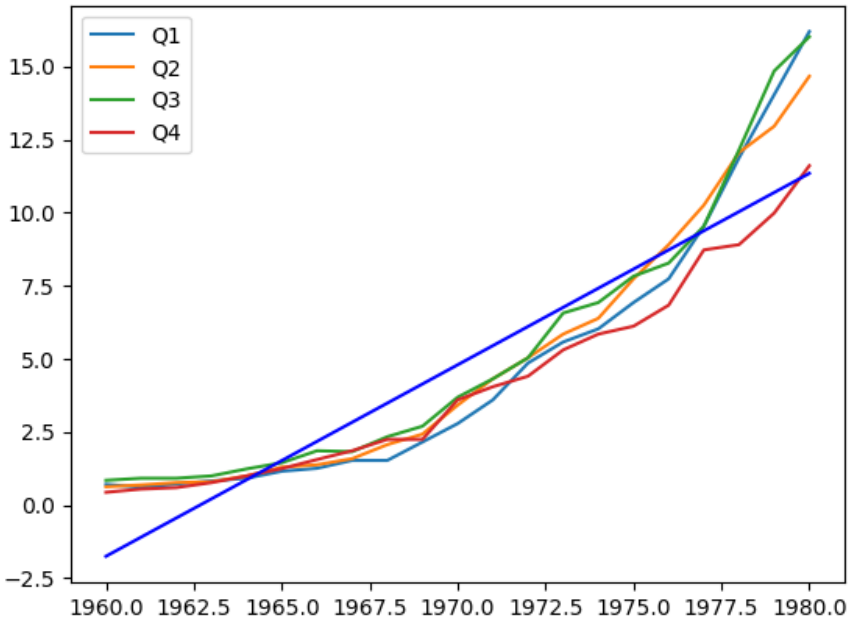


5. Были построени графики изменения прибыли во времени для каждого кваратала, и была построена линейная регрессия для каждой из них в отдельности и для всех кварталов вместе.



```
Q1 rss = 7.271e+01
Q2 rss = 4.397e+01
Q3 rss = 5.951e+01
Q4 rss = 1.703e+01
```

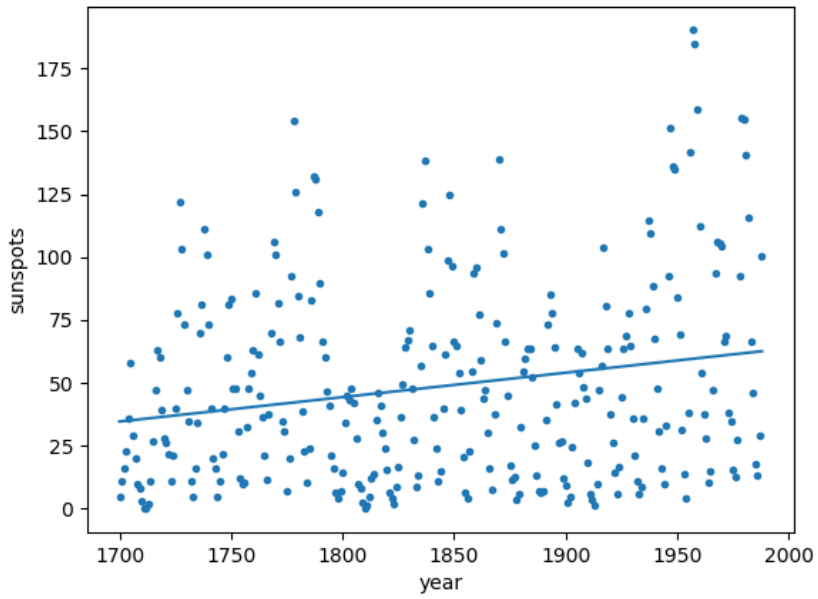
Наибольшую динамику имеет 3 квартал (Q3).



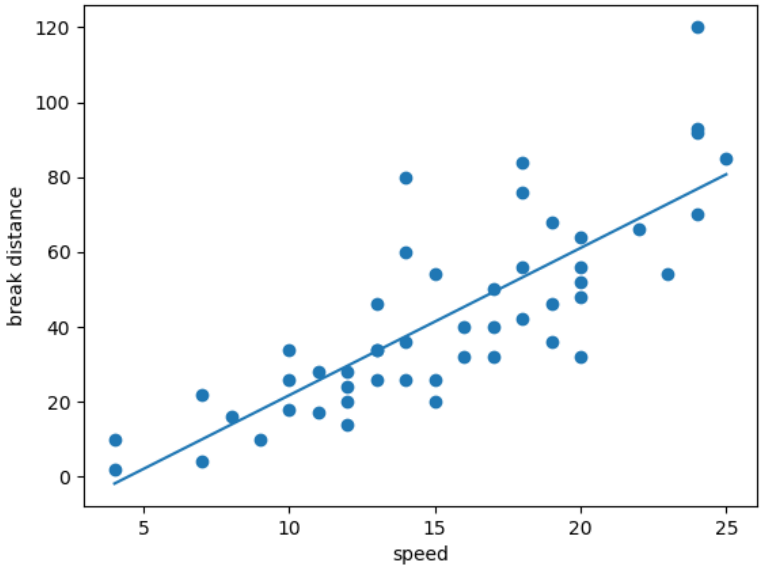
Прогноз по прыбыли в 2016 году:

```
Q1, prediction for 2016: [36.75963636]
Q2, prediction for 2016: [36.48945455]
Q3, prediction for 2016: [37.65393939]
Q4, prediction for 2016: [28.79391342]
Mean, prediction for 2016: [34.92423593]
```

6. Был построен график изменения числа солнечных пятен во времени. Для данных была построена линейная регрессия.



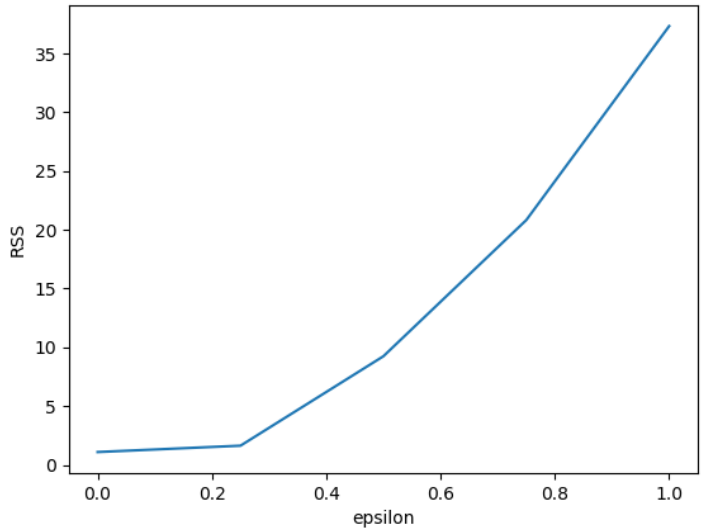
7. Была построена регресионная модель для набора cars.csv



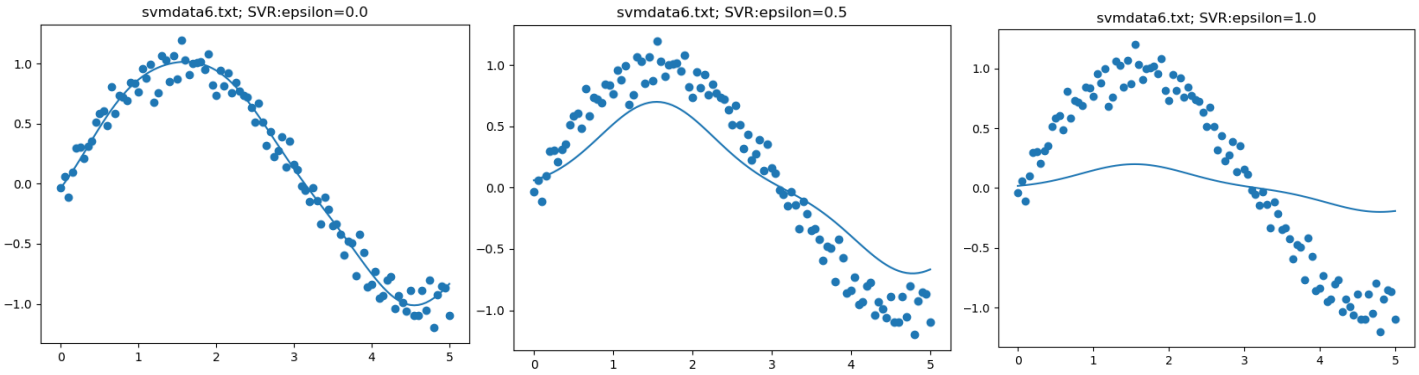
Оценка длины тормозного пути при скорости 40 миль в час:

speed = 40, break distance = [139.71]

8. Был построен регресионный алгоритм метода опорных векторов (sklearn.svm.SVR) с параметром C=1 и ядром “rbf”. Был построен график зависимостри среднеквадратичной ошибки на выборке от значения параметра epsilon.



Параметр epsilon допускает предел в которой не учитывается штрафная функция



9. На наборе nsw74psid1.csv было построено регресионное дерево (sklearn.tree.DecisionTreeRegressor) для признака re78. Так же построена линейная модель и SVM регрессия.

```
DecisionTree score = 0.997, RSS = 1.83e+09
SVR score = -0.004, RSS = 6.56e+11
LinearRegression score = 0.586, RSS = 2.70e+11
```

По результатам видно, что регресионное дерево на данном наборе работает найболее эффективно.

Вывод

В ходе данной лабораторной работы был получен опыт работы с алгоритмов регресии LinearRegression, Ridge, SVR, DecisionTreeRegressor. Было исследованое поведение и качество этих алгоритмов на разных наборах данных.

Текст программы

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

def task1():
    df = pd.read_csv('cygage.txt', delim_whitespace=True)
    X = np.array(df['Depth'].values).reshape(-1, 1)
    Y = df['calAge'].values
    w = df['Weight']

    reg = LinearRegression().fit(X, Y, w)
    print('LineaerRegression:')
    print(f' score = {reg.score(X, Y, w)}')
    print(f' coef_ = {reg.coef_}')
    print(f' intercept = {reg.intercept_}')

    plt.figure()
    plt.plot(range(1, 1000), reg.predict(np.array(range(1, 1000)).reshape(-1, 1)),
c='blue')
    plt.scatter(X, Y, c='red')


def findBestFeatures(X, Y, print_all=False):

    bestFeatures, bestRSS = None, np.inf

    for k in range(1, X.shape[1] + 1):

        idx_combinations = list(itertools.combinations(range(0, X.shape[1]), k))
        for comb in idx_combinations:

            reg = LinearRegression().fit(X[:, list(comb)], Y)
            yy = reg.predict(X[:, list(comb)])
            RSS = np.sum([(Y[i] - yy[i])**2 for i in range(len(Y))])

            if RSS < bestRSS:
                bestRSS, bestFeatures = RSS, list(comb)
            if print_all:
                print('idx_comb = {}, RSS = {:.2f}'.format(comb, RSS))

    return bestFeatures
```

```
def task2():
    df = pd.read_csv('reglab.txt', delim_whitespace=True)
    X = np.array(df.iloc[:, 1:].values)
    Y = df['y'].values

    best_features = findBestFeatures(X, Y, print_all=True)
    print(f'best features: {best_features}')
    X = X[:, best_features]

    reg = LinearRegression().fit(X, Y)
    print(f'score = {reg.score(X, Y)}')

def task3():
    df = pd.read_csv('longley.csv')
    df = df.drop('Population', 1)

    Y = df['Unemployed'].values
    X = df.drop('Unemployed', 1).values

    x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.5,
random_state=1)

    test_errors, train_errors = [], []
    for i in range(26):
        alpha = 10**(-3+0.2*i)
        reg = Ridge(alpha=alpha).fit(x_train, y_train)
        test_errors.append(1 - reg.score(x_test, y_test))
        train_errors.append(1 - reg.score(x_train, y_train))

    print([10**(-3+0.2*i) for i in range(26)])
    plt.figure()
    plt.plot([10**(-3+0.2*i) for i in range(26)], test_errors, label='test error')
    plt.plot([10**(-3+0.2*i) for i in range(26)], train_errors, label='train error')
    plt.xlabel('alpha')
    plt.ylabel('error')
    plt.legend()
    plt.title('longley.csv;')

def task4():
    df = pd.read_csv('eustock.csv')
    x = np.array(range(df.shape[0])).reshape(-1, 1)

    plt.figure()
    for i in range(4):
        y = df.iloc[:, i].values
        reg = LinearRegression().fit(x, y)
        y_pred = reg.predict(x)
        rss = np.sum([(y[i] - y_pred[i])**2 for i in range(len(y))])
        print(f'data = {df.columns[i]}', 'rss = {:.3e}'.format(rss))
        plt.plot(x, reg.predict(x), color=f'C{i}')
        plt.plot(x, df.iloc[:, i], color=f'C{i}', label=df.columns[i])
    plt.legend()

    y = [np.mean(df.iloc[i, :]) for i in range(df.shape[0])]
    reg = LinearRegression().fit(x, y)

    plt.figure()
    for i in range(4):
        plt.plot(x, df.iloc[:, i], label=f'{df.columns[i]}')
    plt.plot(x, reg.predict(x), color='b')
    plt.legend()
    plt.title(f'eustock.csv - mean')
```

```
def task5():
    df = pd.read_csv('JohnsonJohnson.csv')
    x = np.array(range(1960, 1981)).reshape(-1, 1)
    q = [[df.iloc[i, 1] for i in range(j, df.shape[0], 4)] for j in range(4)]

    plt.figure()
    for i in range(4):
        reg = LinearRegression().fit(x, q[i])

        y_pred = reg.predict(x)
        rss = np.sum([(q[i][j] - y_pred[j])**2 for j in range(len(q[i]))])
        print(f'Q{i+1}', 'rss = {:.3e}'.format(rss))
        print(f'Q{i+1}, prediction for 2016: {reg.predict([[2016]])}')
        plt.plot(x, q[i], color=f'C{i}')
        plt.plot(x, reg.predict(x), color=f'C{i}', label=f'Q{i+1}')
    plt.legend()

    qmean = [np.mean([q[j][i] for j in range(4)]) for i in range(x.shape[0])]
    plt.figure()
    for i in range(4):
        plt.plot(x, q[i], label=f'Q{i+1}')
    reg = LinearRegression().fit(x, qmean)
    plt.plot(x, reg.predict(x), color='blue')
    plt.legend()
    print(f'Mean, prediction for 2016: {reg.predict([[2016]])}')


def task6():
    df = pd.read_csv('sunspot.year.csv')
    x, y = np.array(df['index'].values).reshape(-1, 1), df['value'].values
    reg = LinearRegression().fit(x, y)
    plt.figure()
    plt.scatter(x, y, marker='.')
    plt.plot(x, reg.predict(x))
    plt.xlabel('year')
    plt.ylabel('sunspots')

def task7():
    df = pd.read_csv('cars.csv')
    x, y = np.array(df['speed'].values).reshape(-1, 1), df['dist'].values
    reg = LinearRegression().fit(x, y)
    plt.figure()
    plt.scatter(x, y)
    plt.plot(x, reg.predict(x))
    plt.xlabel('speed')
    plt.ylabel('break distance')
    print(f'speed = {40}, break distance = {reg.predict([[40]])}')

def task8():
    df = pd.read_csv('svmdata6.txt', delim_whitespace=True)
    x = np.array(df['X'].values).reshape(-1, 1)
    y = df['Y'].values

    eps_vals, rss_vals = np.linspace(0, 1, 5), []
    for eps in eps_vals:
        reg = SVR(C=1, kernel='rbf', epsilon=eps, gamma='auto').fit(x, y)
        y_pred = reg.predict(x)
        rss_vals.append(np.sum([(y[i] - y_pred[i])**2 for i in range(len(y))]))

        plt.figure()
        plt.scatter(x, y)
        plt.plot(x, reg.predict(x))
```

```
        plt.title(f'svmdata6.txt; SVR:epsilon={eps}')

    plt.figure()
    plt.plot(eps_vals, rss_vals)
    plt.xlabel('epsilon')
    plt.ylabel('RSS')

def task9():
    df = pd.read_csv('nsw74psid1.csv')
    x = df.iloc[:, 0:-1].values
    y = df['re78'].values

    regs = {'DecisionTree': DecisionTreeRegressor(),
            'SVR': SVR(gamma='auto'),
            'LinearRegression': LinearRegression()}
    for (key, reg) in regs.items():
        reg.fit(x, y)
        y_pred = reg.predict(x)
        rss = np.sum([(y[i] - y_pred[i]) ** 2 for i in range(len(y))])
        print(key, 'score = {:.3f}, RSS ={:.2e}'.format(reg.score(x, y), rss))

#task1()
#task2()
#task3()
#task4()
#task5()
#task6()
#task7()
#task8()
task9()

plt.show()
```