

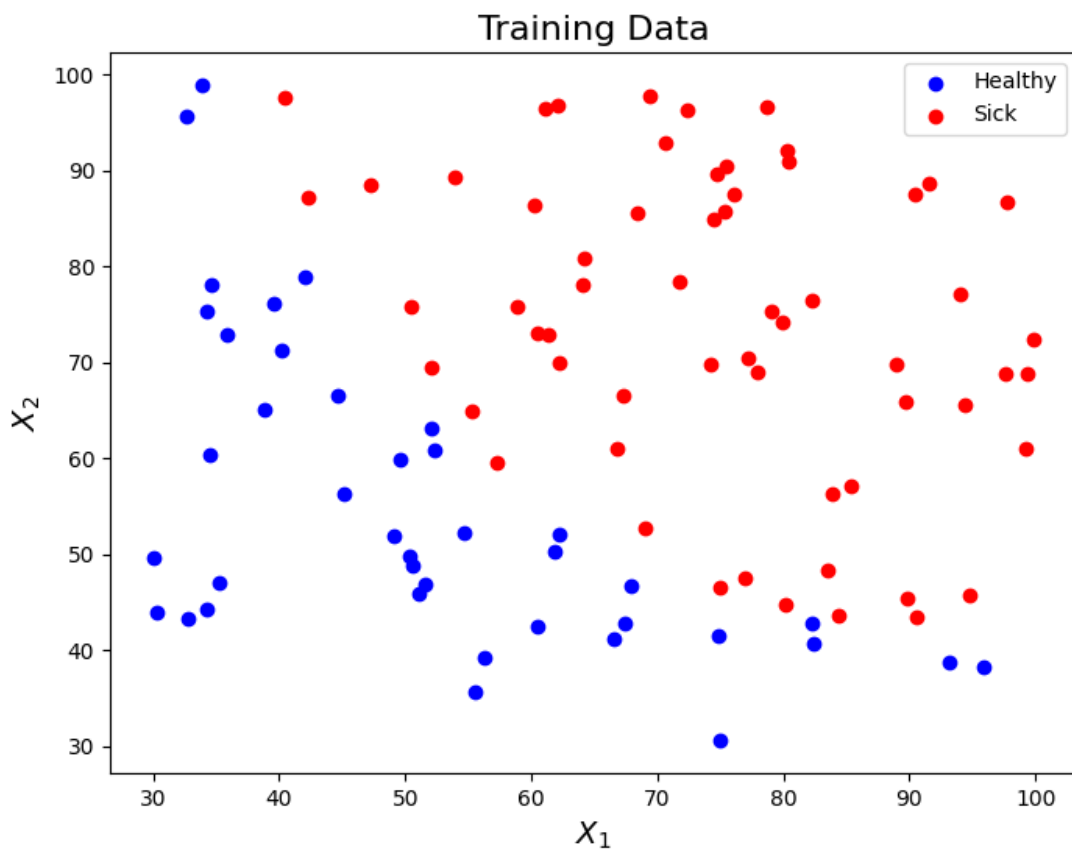
به نام خدا

گزارش کار تمرین 1 الگوریتم های یادگیری ماشین

امیرحسین مرتضی 810895050

سوال اول

1- نمودار داده های آموزشی براساس ۲ ویژگی و label



2- محاسبه بردار theta به طوری که تابع هزینه کمینه شود به روش GD

فرض ها:

m = features count

n = samples count

$$h\theta = \text{sigmond}(\theta^T x)$$

$$J(\theta) = -1/m * \sum(Y(i) * \log(h\theta(X(i))) - (1 - Y(i)) * \log(1 - h\theta(X(i))))$$

GD:

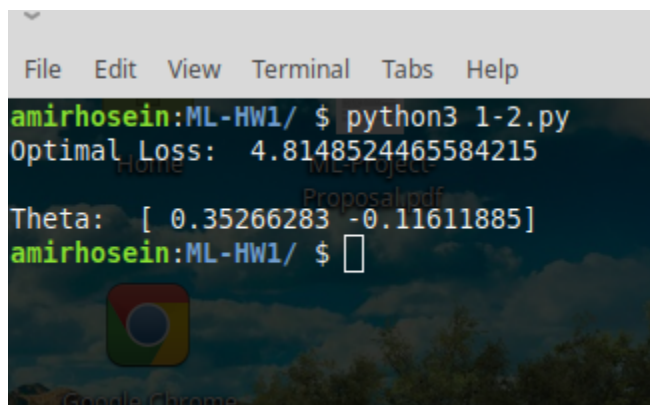
$$\alpha = 0.01$$

$$\text{Iteration} = 10000$$

$$\theta_j := \theta_j - \alpha * dJ/d\theta_j = \theta_j - \alpha/m * (h\theta(X(i)) - Y(i)) * X(i)_j$$

به صورت همزمان برای تمام داده ها اجرا شد و بردار theta را یافتیم

$$j = 0, 1, \dots, n$$



```

File Edit View Terminal Tabs Help
amirhosein:ML-HW1/ $ python3 1-2.py
Optimal Loss: 4.8148524465584215
Theta: [ 0.35266283 -0.11611885]
amirhosein:ML-HW1/ $ 

```

3- محاسبه theta با اضافه کردن ترم لاندا در زیگما مجذور theta به تابع هزینه

$$J(\theta) = -1/m * \sum(Y(i) * \log(h\theta(X(i))) - (1 - Y(i)) * \log(1 - h\theta(X(i)))) + L * \sum(\theta_j^2)$$

در حالت L2-Norm با Regularized GD داریم:

$$\theta_j := \theta_j * (1 - (\alpha * L) / m) - \alpha/m * (h\theta(X(i)) - Y(i)) * X(i)_j$$

با اجرای کد مشاهده می کنیم با افزایش L نرخ کاهش بیشتر میشود.

```
File Edit View Terminal Tabs Help
amirhosein:ML-HW1/ $ python3 1-3.py
L: 0
Optimal Loss: 4.8148524465584215
Theta: [ 0.35266283 -0.11611885]
amirhosein:ML-HW1/ $ python3 1-3.py
L: 1
Optimal Loss: 4.830877915498331
Theta: [ 0.35256905 -0.11590368]
amirhosein:ML-HW1/ $ python3 1-3.py
L: 2
Optimal Loss: 4.846797815437439
Theta: [ 0.35247458 -0.11568761]
amirhosein:ML-HW1/ $ python3 1-3.py
L: 10
Optimal Loss: 4.970819437544268
Theta: [ 0.35169223 -0.1139232 ]
amirhosein:ML-HW1/ $ python3 1-3.py
L: 100
Optimal Loss: 5.863672135165725
Theta: [ 0.33463941 -0.07901716]
amirhosein:ML-HW1/ $ python3 1-3.py
```

4- با استفاده از L قسمت قبل

L = 4800
Alpha = 0.01
GD iterates = 10000

```
0.6199369661032441
0.7024508742101149
0.6832645661651771
0.5234022370991166
0.7029902161755045
0.5877109392458475
0.6155150644162146
Accuracy: 0.62
amirhosein:ML-HW1/ $
```

سوال دوم

1- بهینه سازی ضرایب و محاسبه تابع خطا

چون فقط یک نوع ویژگی داریم تابع خط رگرسیون که با آن پیشبینی میکنیم به فرم زیر است

$$h(X_i) = t_0 + t_1 X_i$$

محاسبه ضرایب به روش least square

$$Y_i = t_0 + t_1 X_i + \text{Error}_i = h(X_i) + \text{Error}_i$$

$$J(t) = (1/2m) * \sum (\text{Error}_i^2) = 1/2n * \sum ((Y_i - h(X_i))^2)$$

روی بردار x عمل normalization انجام دادیم

$$X = (X - X_{\min}) / (X_{\max} - X_{\min})$$

چون تعداد ویژگی ها کم اند اینجا برای یافتن ضرایب t_0 , t_1 به طوری که J کمینه شود داریم

$$t_0 = \frac{\sum(Y_i X_i - mXY)}{\sum(x_i^2 - nX^2)}$$

$$t_1 = Y - t_0 X$$

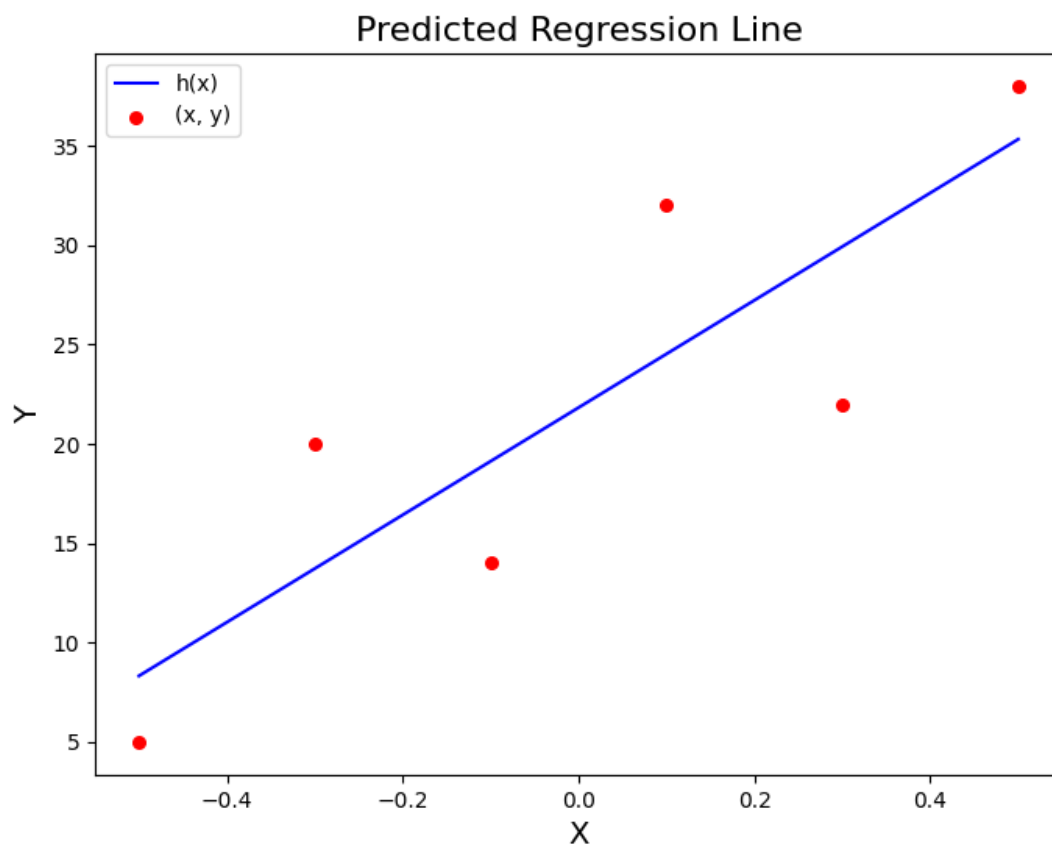
```
python3 Q2-(1-3\).py
amirhosein:ML-HW1/ $ python3 Q2-(1-3\).py
theta0: 8.333333333333332 theta1: 27.0
```

-2

$$Y_i = h(x_i) + \text{Error}_i \rightarrow \text{Error}_i = Y_i - (t_0 + t_1 X_i)$$

```
python3 Q2-(1-3\).py
amirhosein:ML-HW1/ $ python3 Q2-(1-3\).py
Error: [-3.33333333 6.26666667 -5.13333333 7.46666667 -7.93333333 2.66666667]
```

-3



-4

$$h(X(i)) = \theta_0 + \theta_1 * X1(i) + \theta_2 * X2(i)$$

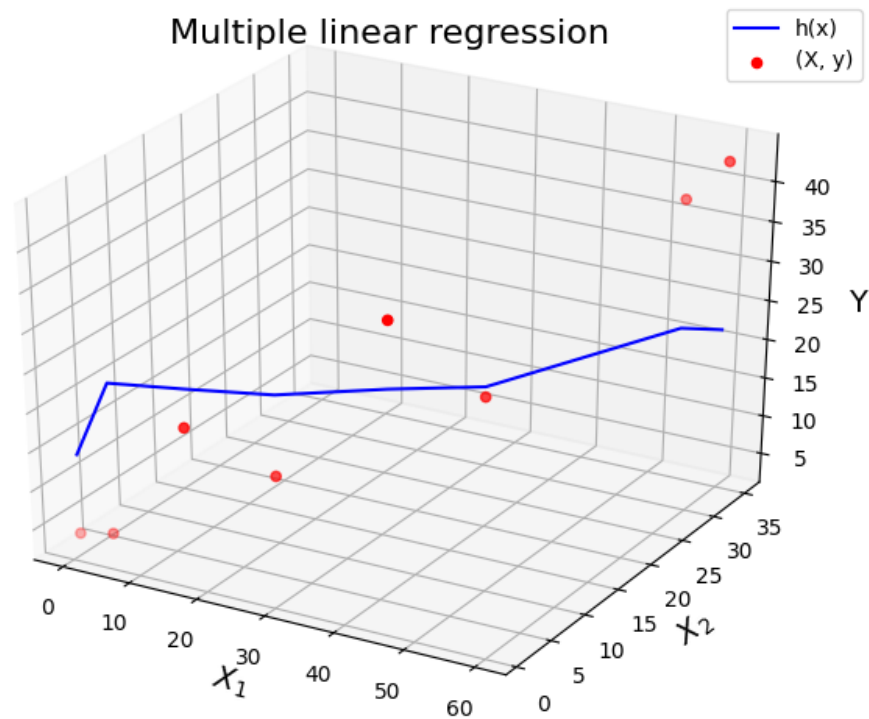
برای تخمین θ اینبار از Batch GD استفاده کردیم

```
amirhosein@amirhosein-Inspiron-N5110: ~/Downloads/ML/ML-HW1
amirhosein:ML-HW1/ $ python3 Q2-4.py
theta0: 13.515469913090769
theta1: 12.796306377157997
theta2: 0.7191635359327591
amirhosein:ML-HW1/ $
```

```

amirhosein:ML-HW1/ $ python3 Q2-4.py
theta0: 13.515468905639308
theta1: 12.796310070760704
theta2: 0.7191588348779945
Error: [-22.31177898 -21.91107801 -6.94633089 -12.91107801  5.14103495
        -4.8511481  11.24379702  16.23401544]
amirhosein:ML-HW1/ $

```



5- چون یک ویژگی داریم از normal equation استفاده میکنیم

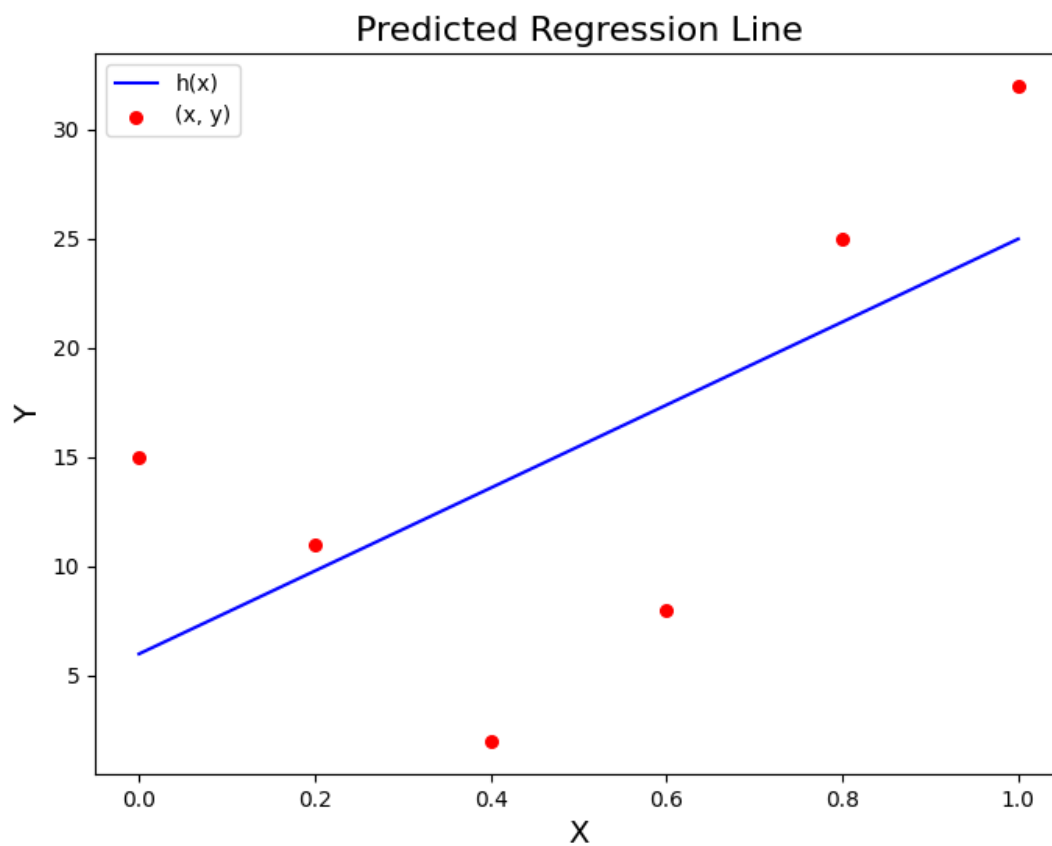
$$dJ(t)/dt = 0$$

After normalization, $\theta = (X.T X)^{-1}(X.T)(Y)$

```

File Edit view Terminal tabs Help
amirhosein:ML-HW1/ $ python3 Q5.py
theta_0 6.0000000000000036
theta_1 18.999999999999996
Error: [ 9.  1.2 -11.6 -9.4  3.8  7. ]

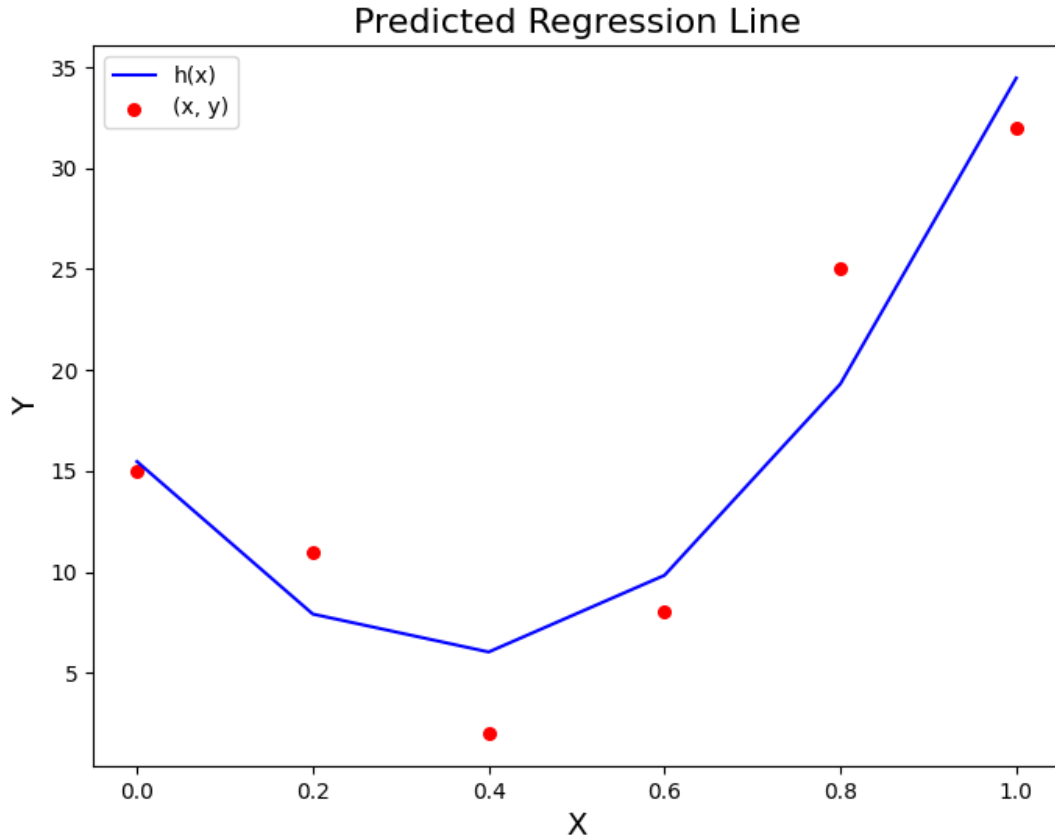
```



-6

با اضافه شدن ترم توان دوم x نمودار fit تر میشود

```
python3 Q2-6.py
amirhosein:ML-HW1/ $ python3 Q2-6.py
2
70.98 x2 - 51.98 x + 15.46
theta_0: 15.464285714285703
theta_1: -51.98214285714282
theta_2: 70.98214285714285
Error: [-0.46428571  3.09285714 -4.02857143 -1.82857143  5.69285714 -2.46428571]
```



سوال سوم

-1

```
xlsx_file = Path('ENB2012_data.xlsx')
wb_obj = openpyxl.load_workbook(xlsx_file)
sheet = wb_obj.active

training_data = []
for row in sheet.iter_rows(min_row=2, max_row=600, min_col=1, max_col=10):
    data = []
    for cell in row:
        data.append(cell.value)
    training_data.append(data)

testing_data = []
for row in sheet.iter_rows(min_row=600, max_row=769, min_col=1, max_col=10):
    data = []
    for cell in row:
        data.append(cell.value)
    testing_data.append(data)
```


2- تابع خطا و روش بهینه سازی

فرض کنیم مجموعه داده های آموزشی D زیر مجموعه کل دیتا N باشد به طوری که برای هر متغیر $X_1 \dots X_m$ و $Y_1 \dots Y_d$ داریم

$$D = \{(x(1), y(1)), \dots, (x(N), y(N))\}$$

که هر بخش x شامل مقدار تمام ویژگی ها باشد

$$x(l) = (x_1, \dots, x_j, \dots, x_m)$$

و خروجی y مجموع ای از target variable ها باشد

$$y(l) = (y_1, \dots, y_i, \dots, y_d)$$

$$i \in \{1, \dots, d\}, j \in \{1, \dots, m\}, \text{ and } l \in \{1, \dots, N\}.$$

باید با یادگیری یک مدل multi-target regression به یک تابع h برسیم

$$h: t \times 1 * \dots * t \times m \rightarrow t \times 1 * \dots * t \times d$$

$$x = (x_1, \dots, x_m) \rightarrow y = (y_1, \dots, y_d),$$

1- یک روش آن است که برای هر خروجی یک رگرسیون انجام دهیم و تمام پیشبینی ها را concat کنیم اما ارتباط بین target ها نادیده گرفته میشود (single-target method) برای یافتن ضرایب h از least square مثل قبل استفاده میشود

2- می توان در روش ها مرسوم طبقه بندی از رگرسیون به جای الگوریتم طبقه بندی استفاده کرد (multi-target regressor stacking, regressor chains, multi-output support vector regression)

Regressor chains (RC)

selecting a random chain of the set of target variables, then building a separate regression model for each target following the order of the selected chain.

The main problem is that it is sensitive to the selected chain ordering.

Multi-output support vector regression

It extends the original feature space and expresses the multi-output problem as an equivalent single-output problem so that it can then be solved using the single-output least squares.

$$(x(l), y(l)) \rightarrow [(l_1, x(l), y_1), \dots, (l_d, x, y_d)]$$

پس یک مجموعه داده آموزشی جدید D_i داریم و ضرایب را با روش مینیم سازی تابع هزینه J می یابیم

$$D_i = \{(l_i, x_i(l)), y_i(l)\}$$

$$J = \frac{1}{2} * ||w||^2 + \frac{1}{C} * \text{sum}(\text{sum}(e_i(l)^2))$$

$$y_i = w^T \phi(l_i, x(l)) + l_i b + e_i$$

$$w = (w_1, \dots, w_d) : \text{weights},$$

$\phi(\cdot)$: a nonlinear transformation to the feature space

$b = (b_1, \dots, b_d)^T$ is the bias vector.

e_i : fitting error

Algorithm adaption methods

simultaneously predicting all the targets using a single model that is able to capture all dependencies.

این روش بهتر است زیرا

It is easier to interpret a single multi-target model than many single-target models and it ensures better predictive performance especially when the targets are correlated.

3- داده ها را به ۳ دسته training , testing و validation تقسیم میکنند

بخش validation set برای model selection استفاده میشود درحالی که testing set برای مدل نهایی پیشبینی است

The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model.

برای تخمین اینکه مدل مناسبی train شده (بسته به سایز داده و ورودی) و ویژگی ها مدل مانند میزان خطا از validation set استفاده میشود که خود آن هم به دوبرخش تقسیم میشود.

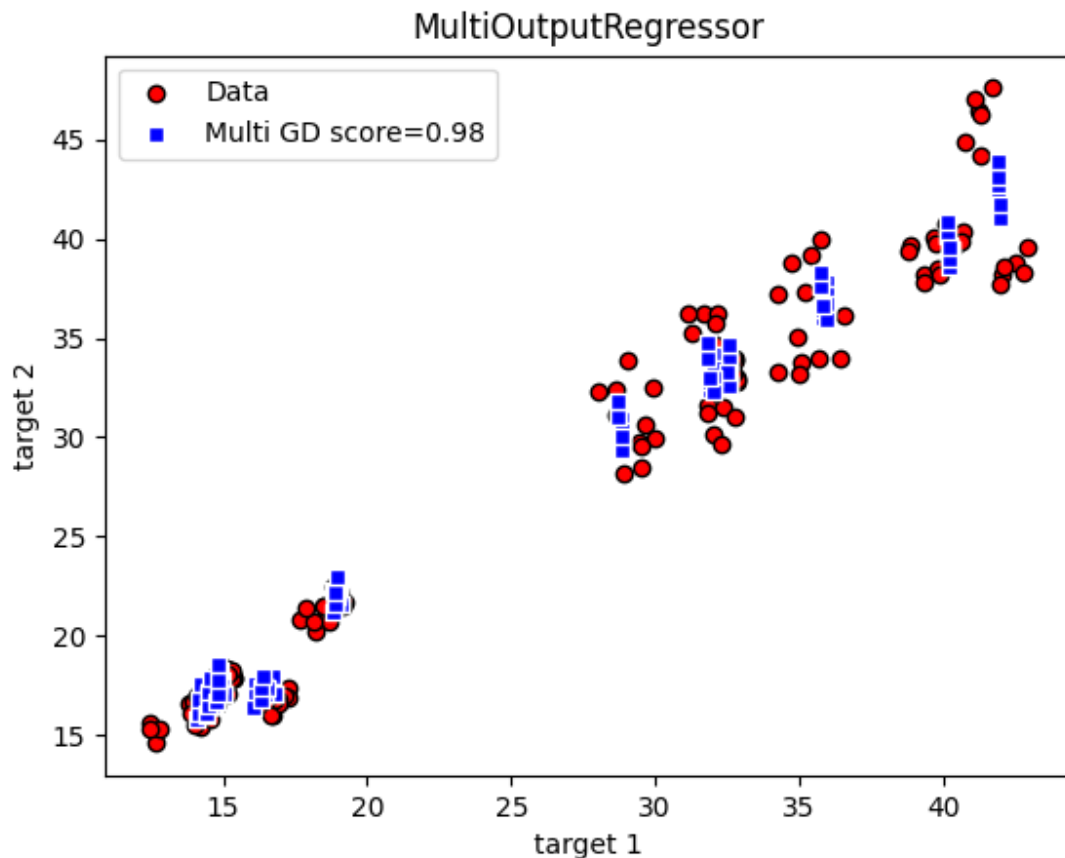
1. فقط مدل ها را بررسی و بهترین راه حل را انتخاب می کند (validation)

2. میزان دقت راه حل انتخابی را می سنجد (test)

-4

برای اعتبار سنجی از تابع cross_val_score استفاده میکنیم این تابع از استراتژی KFold بهره میبرد

```
amirhosein@amirhosein-Inspiron-N5110: ~/Downloads/ML/ML-HW1
amirhosein:ML-HW1/ $ python3 Q3-(4-5\).py
cross_val_score: [0.99868431 0.99771988 0.99589008 0.99693157 0.99821246 0.99601001
0.99866481 0.99895584 0.99760997 0.99870968]
```



-5 دقت score در قسمت قبل ذکر شدند
-6

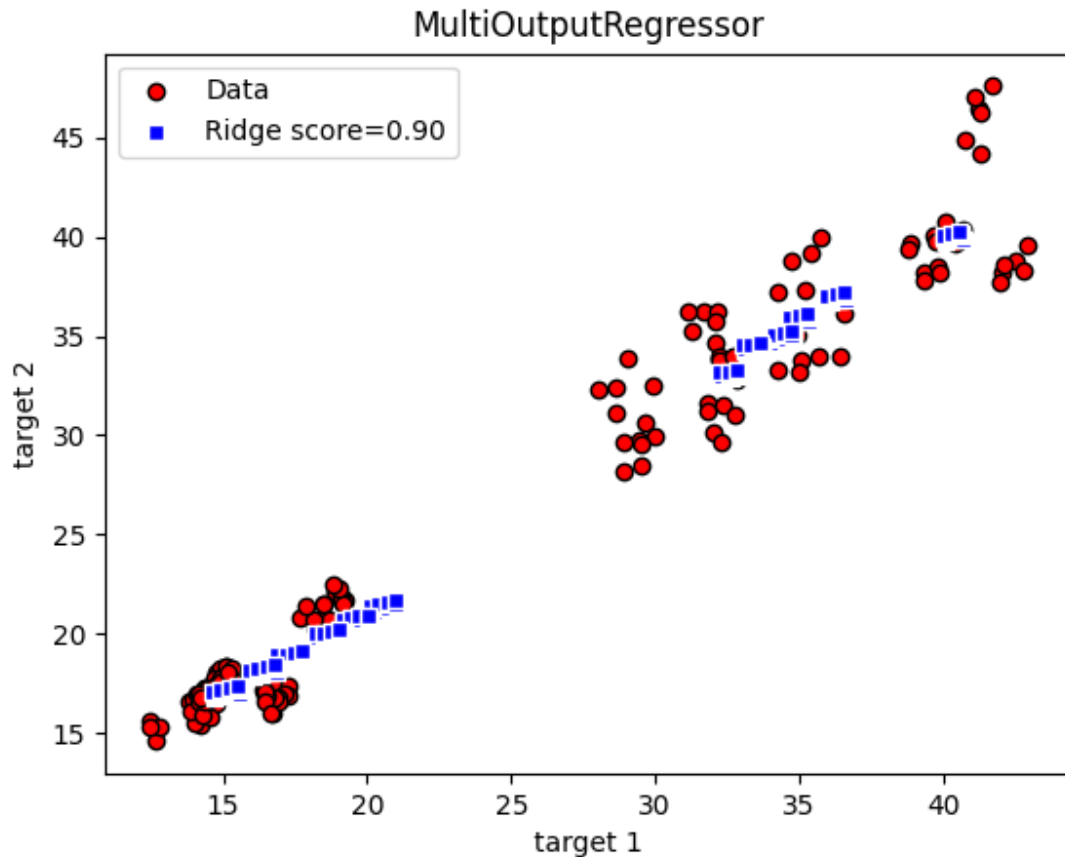
Regression feature selection

3 راه statistical و greedy search و Regularization وجود دارند
روش regularization شناسایی و اصلاح ویژگی ها مهم معمولا برای جلوگیری از overfitting بدون از بین بردن بخشی از داده original
برای به حداقل رساندن تأثیر ویژگیهای بی معنی و همبستگی با مدل ، نظم بخشی ضرایب این ویژگی ها را محدود می کند تا آنها در نتایج پیش بینی سهم نداشته باشند.
این هدف با اضافه کردن یک term penalty برای ضرایب به تابع هزینه محقق می شود که به ۲ راه اجرا میشود
1. اگر تغییر وزن روی خود مقدار ضرایب رخ دهد به آن L2-norm گوئیم و الگوریتم با Lasso regression شناخته میشود
2. اگر وزن دهی روی مجموع مجذور ضرایب رخ دهد (L2-norm) الگوریتم Ridge regression شناخته میشود.

$$J = 1/2m (y - Xw)^2 + \alpha * (w)^2$$

-7

از تابع RidgeCv از کتابخانه sklearn استفاده کردیم که علاوه بر L2-norm regularization عمل cross validation را نیز اجرا میکند. دقت کمتر اما خروجی ها بهم نزدیکتر شدند و مرتب تر شدند.



سوال چهارم

1- روش نیوتن

$$J = 2x^6 + 3x^4 - 10x^3 + 12x^2 - 60x$$

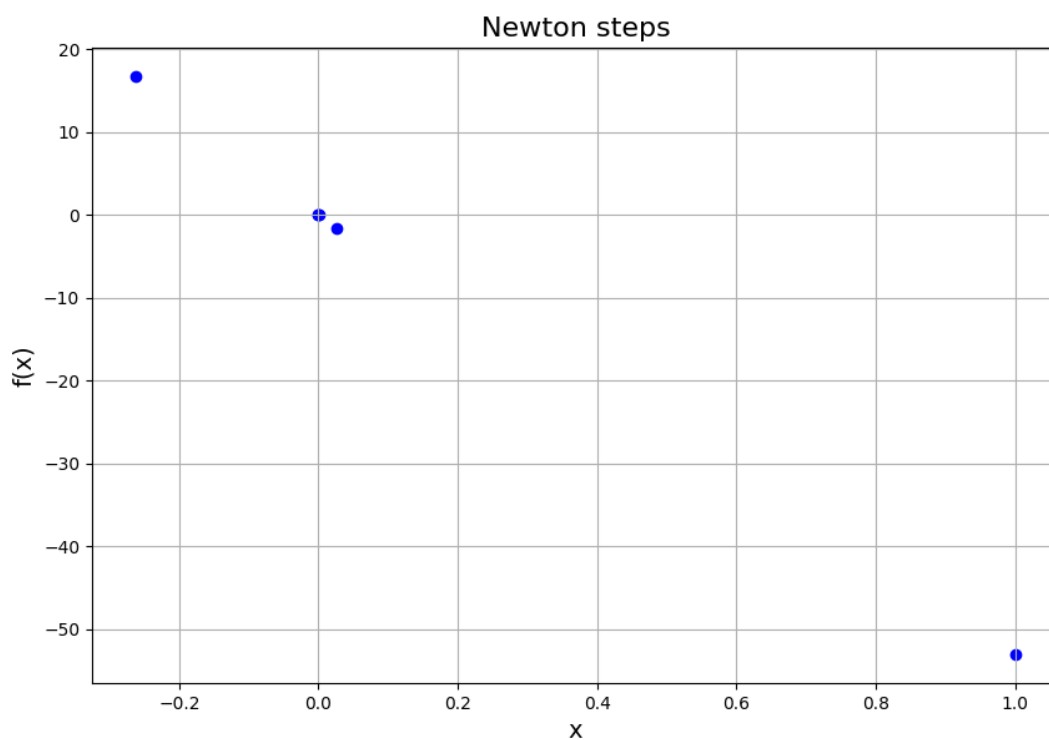
$$\theta := \theta - \frac{dJ}{d\theta}$$

$$\theta := \theta - \text{inverse}(H(n+1)(n+1)) * \text{grad}(J)$$

$$H_{ij} = \frac{dJ}{d\theta_i} * \frac{d\theta_j}{d\theta}$$

$$\text{grad}(J) = \text{transpose}[\frac{dJ}{d\theta_0}, \frac{dJ}{d\theta_1}, \dots, \frac{dJ}{d\theta_n}]$$

بسته به دقت و انتخاب اولیه زمان همگرایی متفاوت است اما در نهایت همگرا می شود تغییرات گام برابر نسبت مقدار تابع به مشتق آن در یک نقطه است باعث می شود اختلاف با نقطه ای که مقدار تابع اکسترمم است کم تر شود.



همچنین مقدار همگرایی پیاده سازی مان را با تابع newton کتابخانه scipy مقایسه کردیم

```
amirhosein@amirhosein-Inspiron-N5110: ~/Downloads/ML/ML-HW1  
amirhosein:ML-HW1/ $ python3 Q4.py  
Root: 2.1936800893584855e-18  
f(x): -1.3162080536150913e-16  
Root with scipy newton: 2.1936800893584855e-18  
f(x) with scipy newton: -1.3162080536150913e-16  
amirhosein:ML-HW1/ $
```