دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

تکلیف شماره ۱

درس یادگیری ماشین

استاد
دکتر ناظرفرد

امیرحسین کاشانی

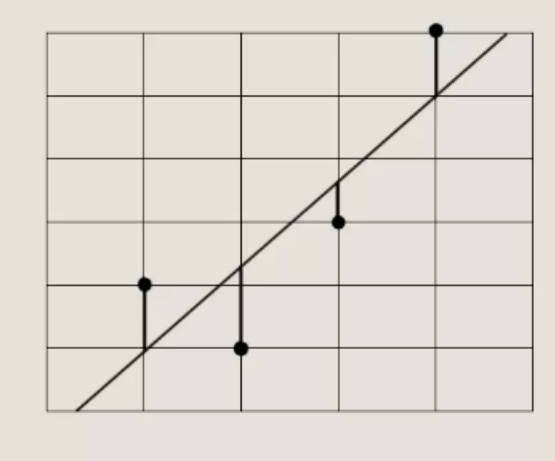
فهرست مطالب

Q1)

a)

Least square error:

در این حالت به دنبال یافتن یک خط با بیشترین تناسب نسبت به مجموعه داده ورودی هستیم به بیان دیگر به دنبال خطی می‌گردیم که مجموع توان دو فاصله نقاط از نقطه متناظرآن ها در آن خط کمترین مقدار ممکن باشد. این تسک دقیقا مشابه عملی است که در رگرسیون صورت می گرد و w های بدست آمده همان معادله خطی هستند که least square error به آن نیاز دارد.



برنامه نویسی خطی

یک مساله برنامه نویس خطی به شکل ماکسیمم کردن یا مینیمم کردن یک معادله خطی با درنظر گرفتن یک قیود خطی بر روی متغیر های به کار رفته در معادله ی اولیه مطرح می شود.

$$z = 0.6x_1 + 0.35x_2$$

$$\begin{aligned} 5x_1 + 7x_2 &\geq 8 \\ 4x_1 + 2x_2 &\geq 15 \\ 2x_1 + x_2 &\geq 3 \\ x_1 \geq 0, x_2 &\geq 0. \end{aligned}$$

مثال فوق یک نمونه از مقدار دهی به معادله ax1 + bx2 می باشد که شرط های زیر را برقرار کرده است.

بهینه سازی محدب

مساله ای است که همه قیود آن حالت محدب دارند و تابع هدفمان نیز بهینه نمودن یک مساله محدب است. از آنجا که ترکیب خطی یک مساله محدب است در نتیجه برنامه نویسی خطی نیز یک حالت خاص از بهینه سازی محدب محسوب می گردد.

b)

رگرسیون ساده از آنجایی که به شکل

$$y = \alpha + \beta x$$

تعریف می شود پارامتر complexity مدل برای آن مطرح نمی گردد اما برای تعداد iteration می توان از دوسیاست حداکثر iteration یا مینیمم خطای قابل قبول استفاده کرد و در صورت رسیدن به هر یک از این شروط اجرای الگوریتم را متوقف نمود.

** بصورت کلی برای انتخاب میزان پیچیدگی میتوان از یک سیاست جست و جوی باینری بهره برد برای مثال در ابتدا یک مدل با درجه پیچیدگی یک و یک مدل با ماکسیمم درجه پیچیدگی مورد نظر ایجاد کرد (برای مثال M) سپس یک مدل با درجه M/2 و بعد از آن در بخشی که نتیجه بهتری داشته را انتخاب کنیم (فرض کنیم بخش راست با پیچیدگی درجه M بهتر بوده) در گام بعدی درجه پیچیدگی ما روند شده 0.75 M خواهد بود و این جستو جوی باینری را ادامه می دهیم تا به یک حالت بهینه نسبت به فضای موجود دست پیدا کنیم.

البته لازم به ذکر است که درحالت هایی که بازه گزینه ها برای انتخاب پیچیدگی کوچک است می توانیم ۳ تا، ۳تا (برای مثال ۳ فرض شده) پیچیدگی را زیاد کنیم و در نقطه ای که بهترین جواب وجود داشت تمام پیچیدگی های درجه ای نزدیک را بررسی کنیم.

c)

به این پارامتر، پارامتر بایاس گفته می شود. در صورت عدم وجود این پارامتر الگوریتم رگرسیون فقط توانایی محاسبه خطوطی که از مبدا می گذرند را خواهد داشت به بیان دیگر نشانگر عرض از مبدا خط پیشنهادی این روش می باشد و درصورت نبودن این پارامتر حاصل اجرای الگوریتم معادل شکل زیر می باشد.

$$\hat{y} = \mathbf{w}^T \mathbf{x}$$

اما وجود پارامتر θ_0 به ما این امکان را می دهد تا همه توابع خطی موجود را بتوانیم ایجاد کنیم.

$$\hat{y} = \mathbf{w}^T \mathbf{x} + b$$

d)

روش k-fold داده آماری را به k دسته تقسیم می کند و هر بار یکی را به عنوان تست بر می گزیند و بقیه را به عنوان داده آموزش مصرف می کند.(این کار را k بارتکرار می کند)

روش leave-one-out یک داده را به عنوان تست بر می دارد و کل مابقی دیتا ست را آموزش می بیند و این کار را k بار تکرار می‌کند. این کار باعث می شود که هر با آموزش دادن این مدل مدت زمان قابل توجهی به طول انجامد اما در زمانی که حجم داده کم است بهتر است از این روش استفاده شود.

I. در مقیاس کوچک leave one out و در مقیاس بزرگ k-fold به دلیل اینکه بار محاسباتی کمتری در هر یادگیری خواهد داشت بهتر است.
II. باتوجه به اینکه روش leave-one-out فقط یک داده را در هر دور ملاک قضاوت قرار می دهد نسبت به داده پرت مقاومت خوبی ندارد در نتیجه **k-fold در مقابل نویز مقاوم تر است.**
III. K-fold، در هر مرحله بخش کوچکتری از داده ها را به عنوان داده آموزش بر می گزیند
IV. به طور کلی نمی توان روش مشخصی برای تعیین k بیان نمود به عنوان یک روش دمه دستی k بین ۵ تا۱۰ توصیه می‌شود که بیانگر این است که در هر تکرار بین ۱۰ تا ۲۰ درصد داده ها را به عنوان تست برداریم. لازم به ذکر است که درصورتی که حجم داده های ما کم است باید از k های بزرگتر استفاده کنیم تا میزان داده مناسب برای اجرای فرآیند آموزش داشته باشیم ولی در زمانی که حجم داده معقول است k با مقدار کمتر زمان کمتر در اجرای آموزش را برای ما به ارمغان می آورد و همین طور به فرآیند آموزش لطمه نمی زند.

e)

regularization یکی از راه های پر طرف دار برای جلوگیری از overfit در اجرای الگوریتم می باشد و به این صورت عمل می‌کند که در تابع هزینه وزن های انتخاب شده را جمع می کند. مانند شکل زیر

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\min_\theta J(\theta)$$

این کار باعث این می شود که از وزن های به شدت زیاد که نتیجه را به سمت خود نا متوازن می کنند جلوگیری شود. حال در صورتی که از توان دو برای θ استفاده کنیم به آن L2 یا *Ridge Regression* گفته می شود و درصورتی که از قدر مطلق استفاده گردد به آن L1 یا *Lasso Regression* گفته می شود. به طور کلی با توجه به اینکه L2 توان ۲ ضرایب را استفاده می کند نتیجه وزن های نهایی کوچک خواهد بود و برای زمانی مناسب است که تعداد ویژگی ها زیاد هستند و feature engineering به شکل

مناسب انجام نشده است اما Lasso Regression سخت گیری کمتری در وزن ها استفاده می کند و وزن های کمتری را صفر می‌کند(این کار باعث حذف فیچر می شود) در نتیجه برای زمان هایی مناسب است که فیچرها رو با دقت انتخاب کرده ایم.

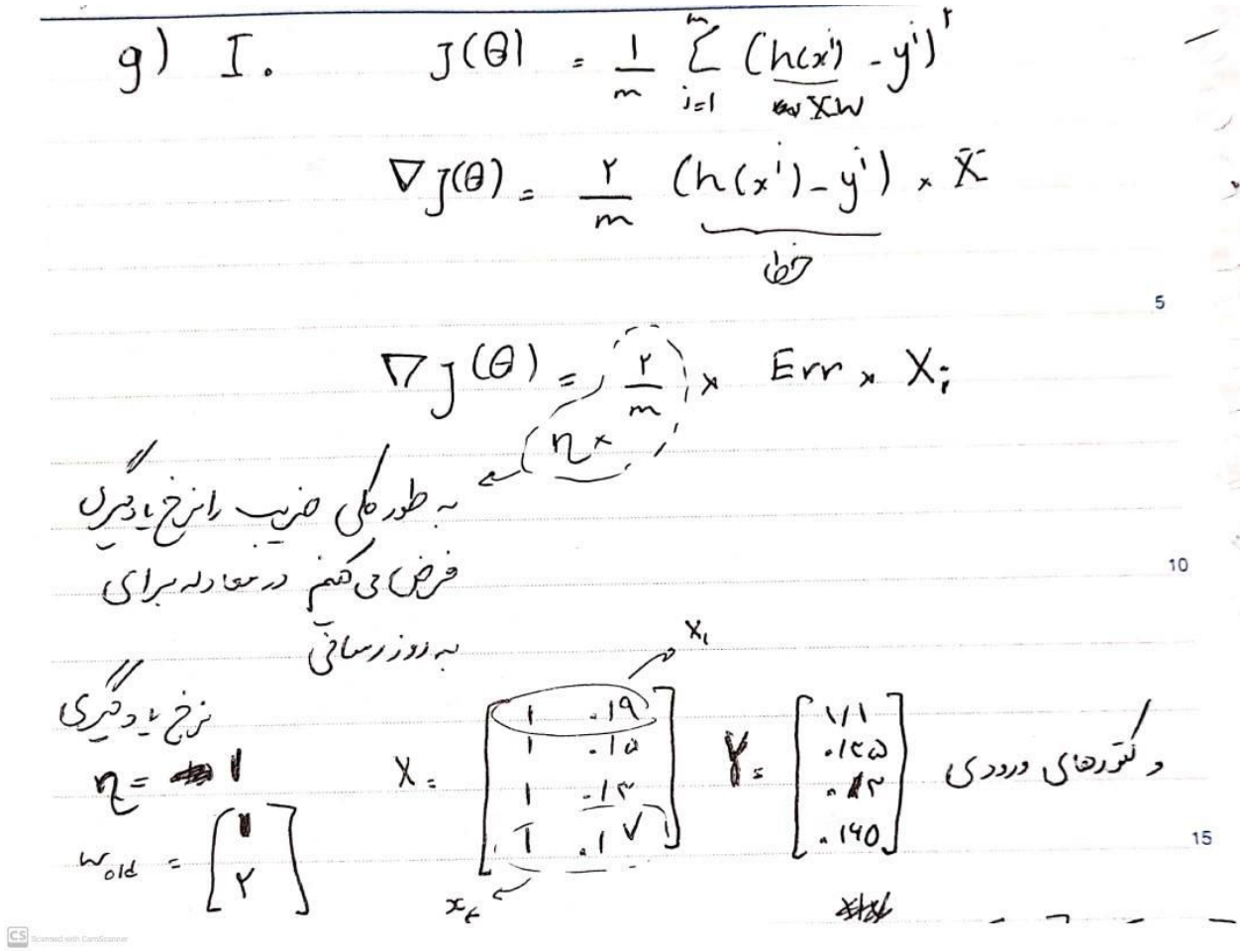
علت استفاده از این نرمال سازی کمک به تشخیص فیچرهای مناسب برای بررسی مدل های مورد نظر و جلوگیری از بیش برازش می باشد.

f)

از آن جهت که تابع در انتخاب نمونه های آزمایش و تست بصورت رندوم انتخاب می کنیم وجود نویز در دوسته تقسیم خواهد شده ونیازی تغییر در این نسبت نخواهیم داشت.

افزایش درجه برای یادگیری نویز کار چندان پسندیده ای نیست و به طور کلی اشتباه است(در صورتی که اصل موضوع آموزش یادگیری نویز نباشد) و با افزودن درجه، مدل را به سمتی سوق می دهیم که داده های آموزشی را با نویز هایشان حفظ کند ولی این کار باعث می شود که در بخش تست عملکرد ضعیف تر بدلیل overfitting از خود نمایش دهد. درصورتی که مدل قبل از افزودن نویز امکان یادگیری کامل داشته باید درجه قبلی را حفظ کند یا اینکه مقدار کمی به درجه آن اضافه گردد.

g)


g) I. $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h(x^i) - y^i)^2$
$\nabla J(\theta) = \frac{2}{m} (h(x^i) - y^i) \times X$
خطا
$\nabla J(\theta) = \frac{2}{m} \times Err \times X_j$
η
به طور کلی ضریب را نرخ یادگیری
فرض می کنیم در معادله برای
به روز رسانی
نرخ یادگیری
$\eta =$
$w_{old} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$
$X = \begin{bmatrix} 1 & .19 \\ 1 & .15 \\ 1 & .13 \\ 1 & .17 \end{bmatrix}$
$Y = \begin{bmatrix} 1.1 \\ .125 \\ .12 \\ .140 \end{bmatrix}$
و بردارهای ورودی
x_1
x_2

Subject

Year Month Date

$$w_{new} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + 1 \times \frac{(111 - 218)}{-117} \times \begin{bmatrix} 1 \\ 0.19 \end{bmatrix}$$

$$= \begin{bmatrix} -0.17 \\ 0.247 \end{bmatrix}$$

$$w_{new} = \begin{bmatrix} -0.17 \\ 0.247 \end{bmatrix} + 1 \times \frac{(+0.17 - 0.125 + 0.125)}{0.153} \times \begin{bmatrix} 1 \\ 0.13 \end{bmatrix}$$

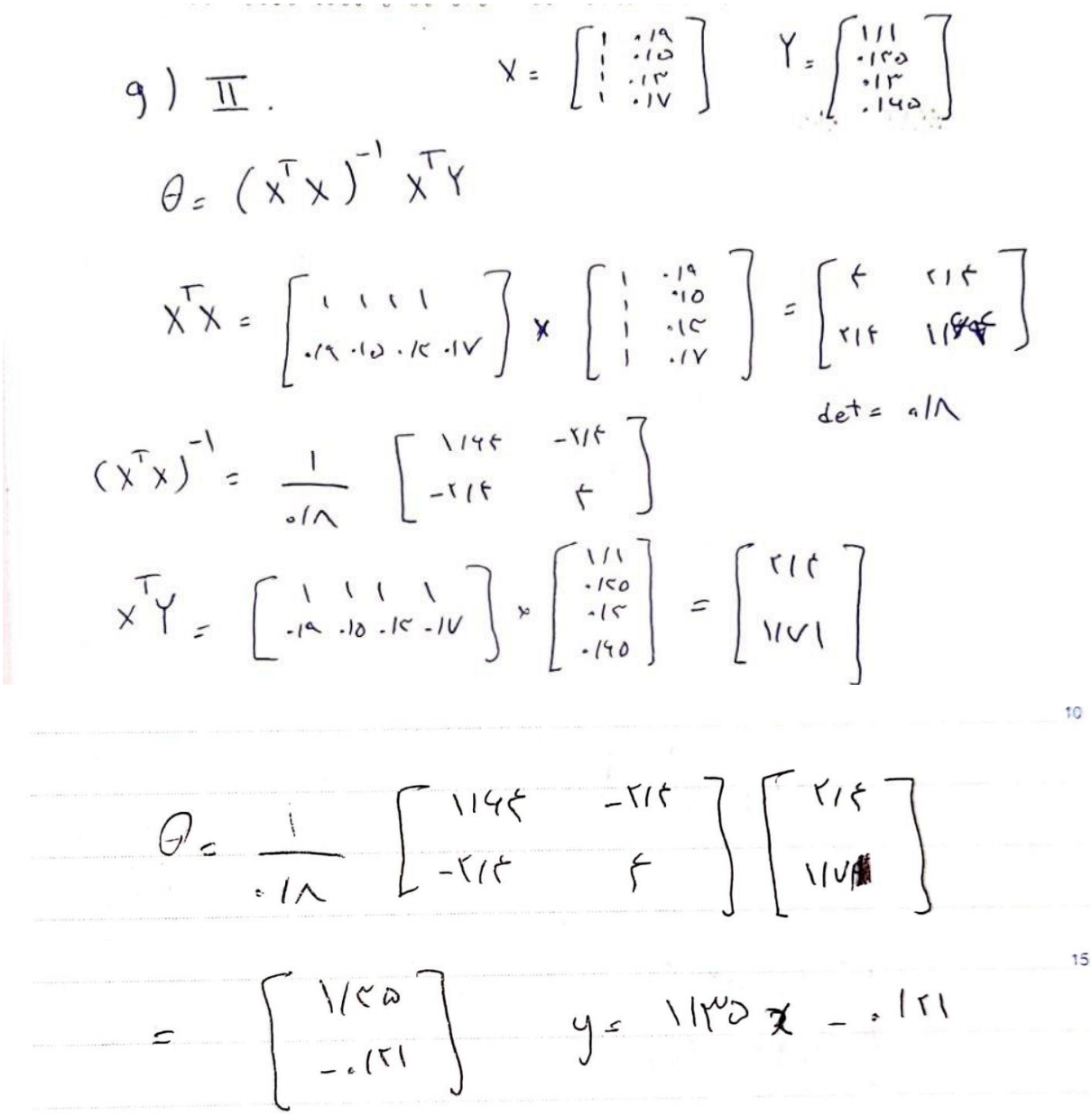
$$= \begin{bmatrix} -0.115 \\ 0.2765 \end{bmatrix}$$

این کار را ۱۰ مرتبه‌ی دیگر انجام می‌دهیم

$$w = \begin{bmatrix} -1.9052 \\ 0.3242 \end{bmatrix}$$

نتیجه نهایی

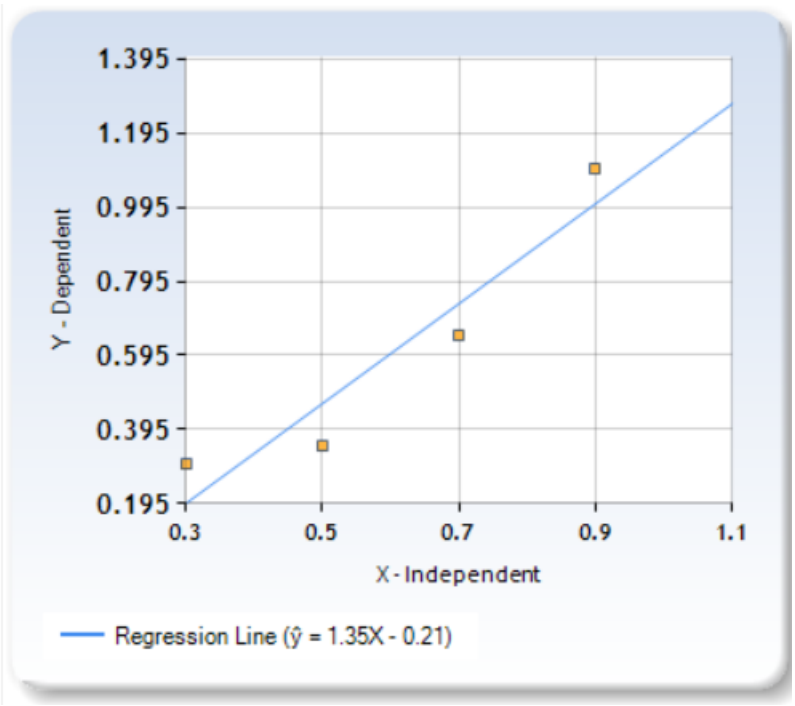
$$w = \begin{bmatrix} 0.009 \\ 0.19687 \end{bmatrix}$$



III)

برای حالت normal question

Sum of squares (SS_X) = 0.2
Sum of products (SP) = 0.27



h)

به طور کلی به روز رسانی وزن ها به ازای هر داده موجب ناپایداری در روند همگرایی می شود. احتمال اینکه این ناپایدار باعث عدم همگرایی در انتها شود کم است، اما روند همگرایی را کند تر می کند و نسبت به داده های پرت واکنش بدی نشان می‌دهد حال چه در اجرای gradient descent عادی به ازای هر داده این کار تکرار شود یا در stochastic gradient descent بر روی یک نمونه محدود از کل داده ها اجرا شود.

برای بهبود این مشکل می توان از Batch gradient descent استفاده نمود که به ازای هر داده به روز رسانی را انجام نمی دهد. بلکه به ازای هر دسته میانگین خطا هر دسته از ورودی ها اینکار را انجام می دهد. از مفروضات این الگوریتم این است که همه دیتاست حتما مورد بررسی قرار گرفت و این فرض دلیل اصلی کند بودن این روش در اجرا می باشد اما روند اجرا پایدار تر بوده و نوسانات شدید ندارد.

در آخر با ترکیب دو روش stochastic , Batch به روش mini Batch می رسیم که فرض استفاده کامل از دیتاست را حذف می کند ولی بصورت Batch وزن ها را به روز رسانی می کند.

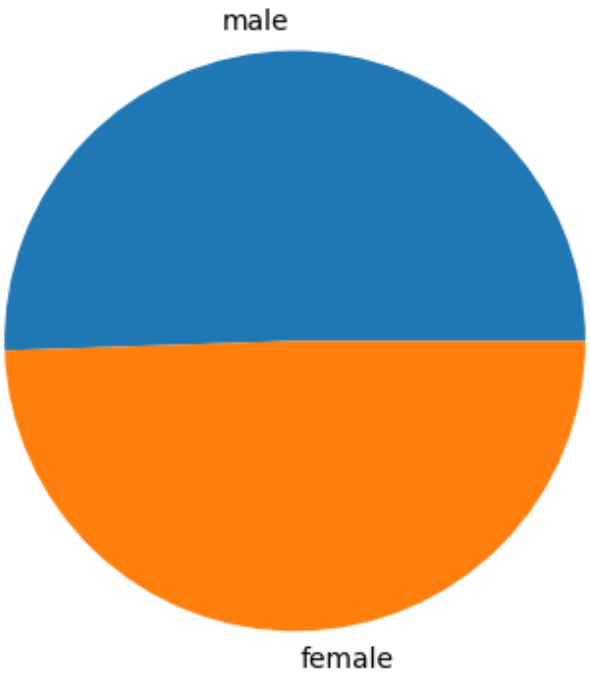
i)

در رابطه با شباهت این دو روش می توان گفت که هر دو آن ها در تلاش هستند تا یک مدل با کمترین خطا مجموع مربعات را برای داده های ورودی خود پیدا کنند تا بتوانند برخی ویژگی دیگر را استخراج نمایند اما تفاوت بنیادی در رفتار این دو این است که در function estimation می توان همه متغیرها را متغیرهای آزاد فرض کرد به بیان دیگر هدف یافتن وزن ها است اما در regression هدف پیش بینی مقدار متغیروابسته (هدف) بر اساس سایر متغیرهای آزاد است .

Q2)

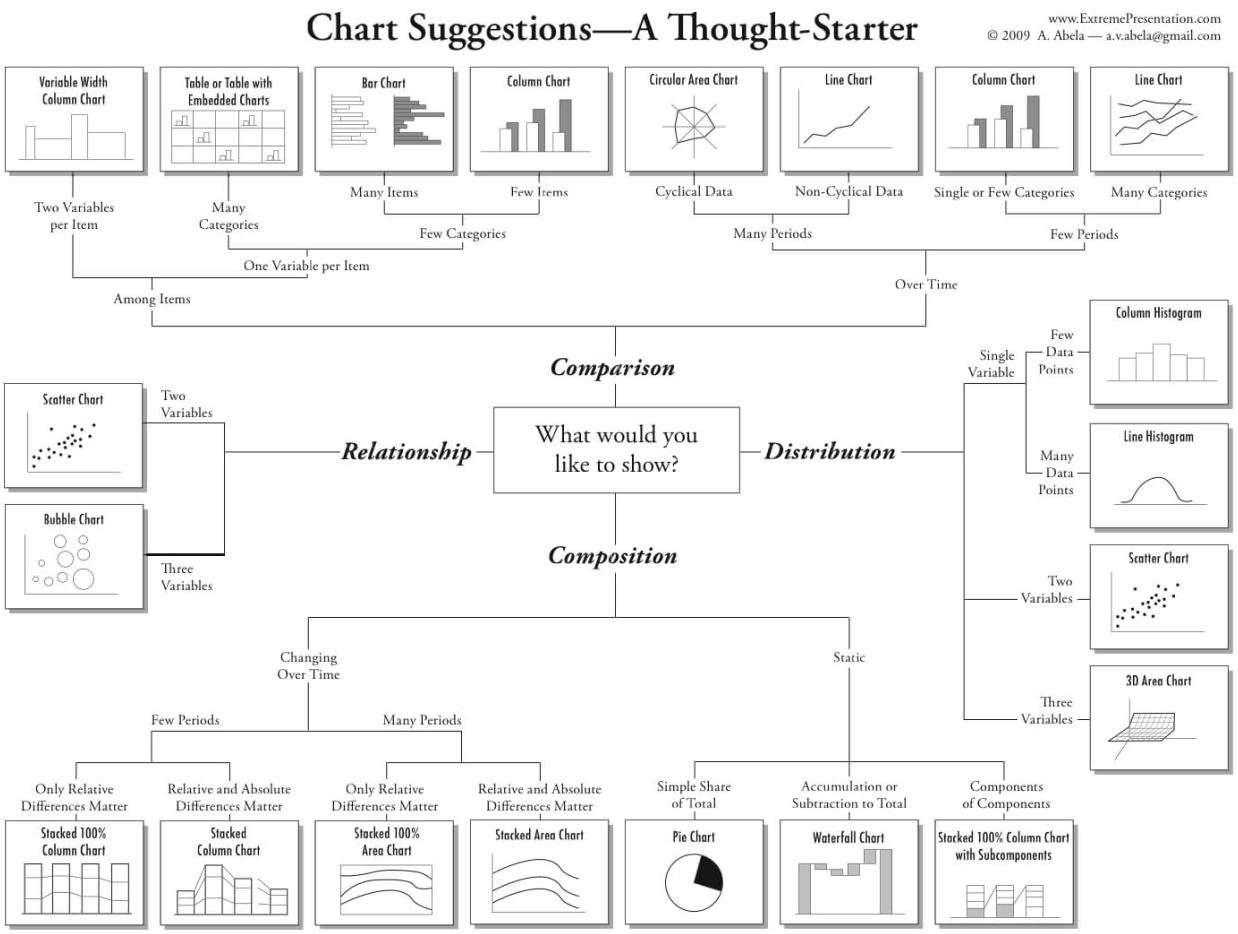
a)

```
C:\Anaconda\python.exe G:/university/master/term3/machineLearning/codes/Q2.py
male      676
female    662
Name: sex, dtype: int64
```

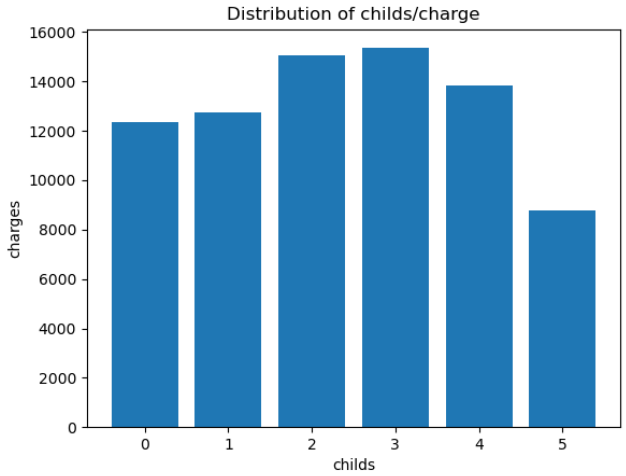


نتایج حاکی از این بود که مقدار اندکی تعداد آقایان بیشتر از خانم ها می باشد.

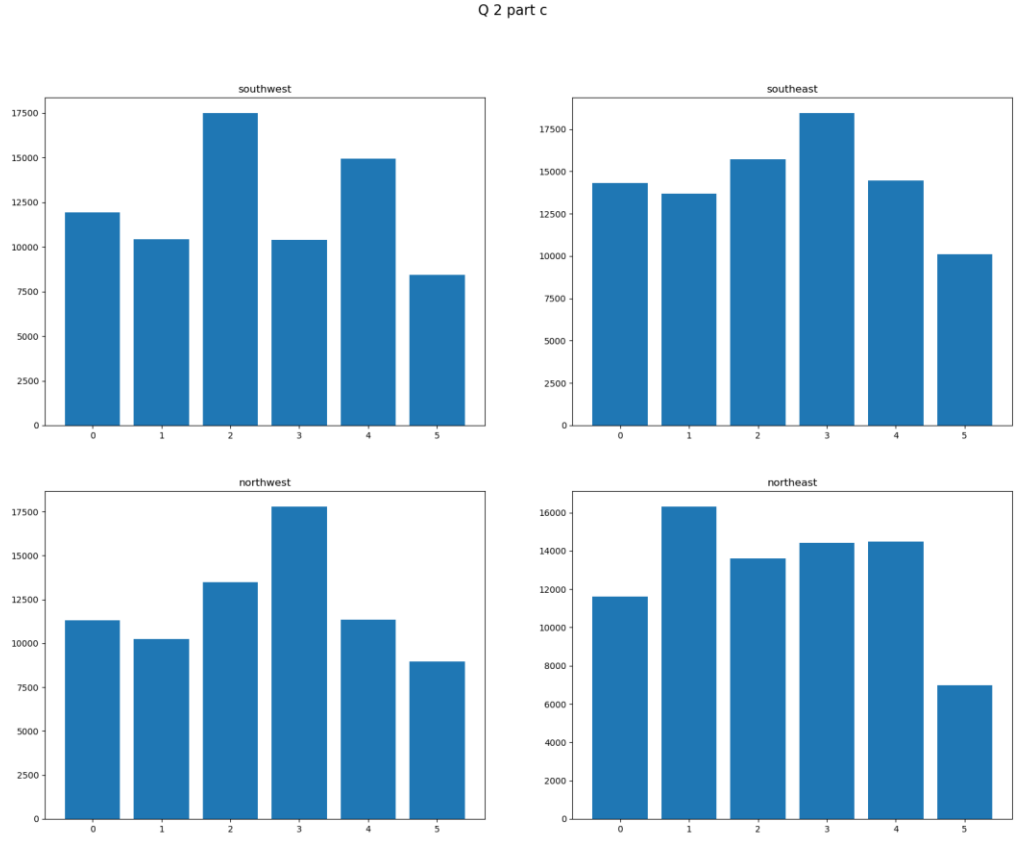
b)



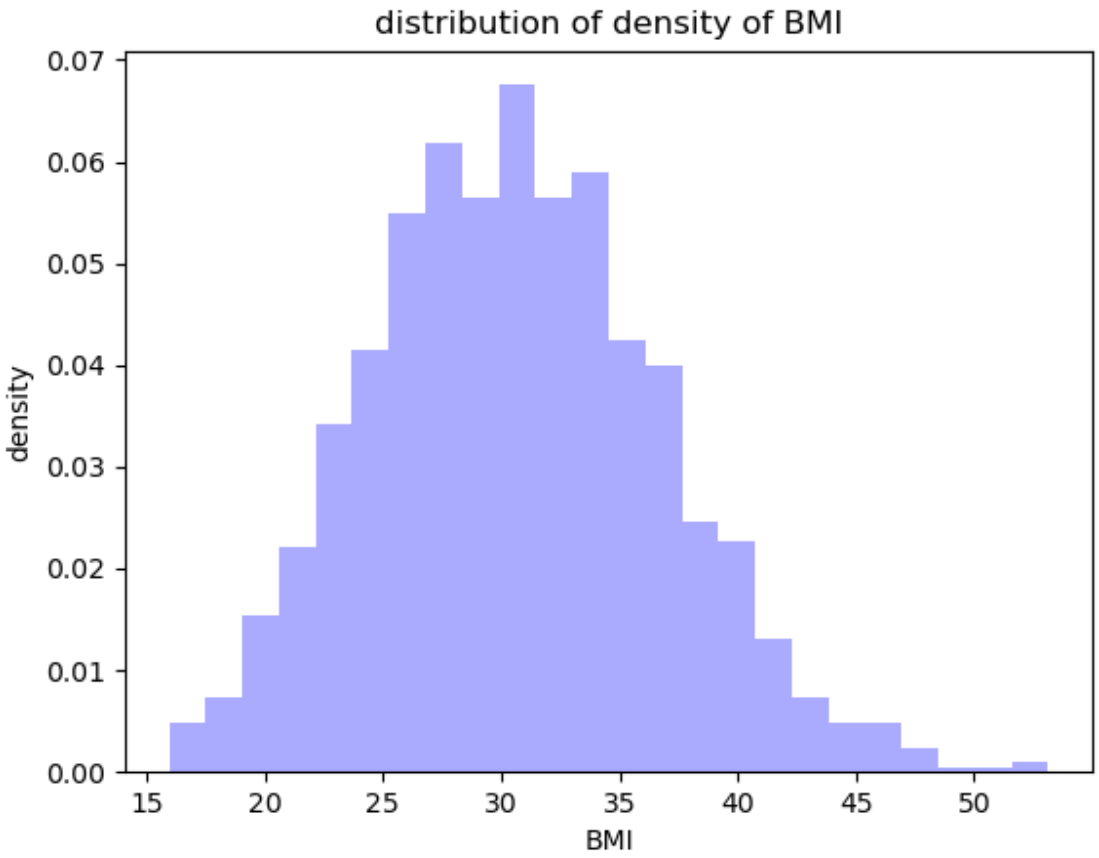
از بین نمودار های ممکن scatter plot نموداری است که برای بررسی ارتباط دو نمودار انتخاب گردیده است



c)

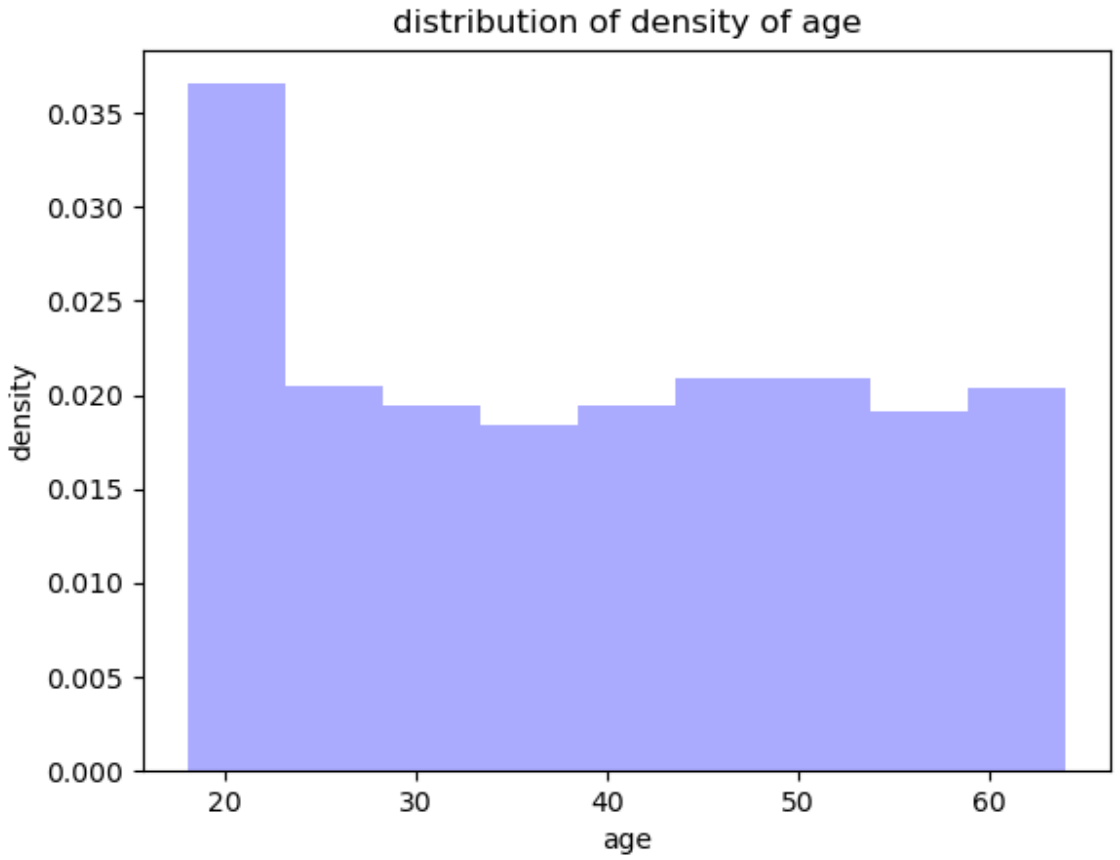


D)


distribution of density of BMI
density
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
15
20
25
30
35
40
45
50
BMI

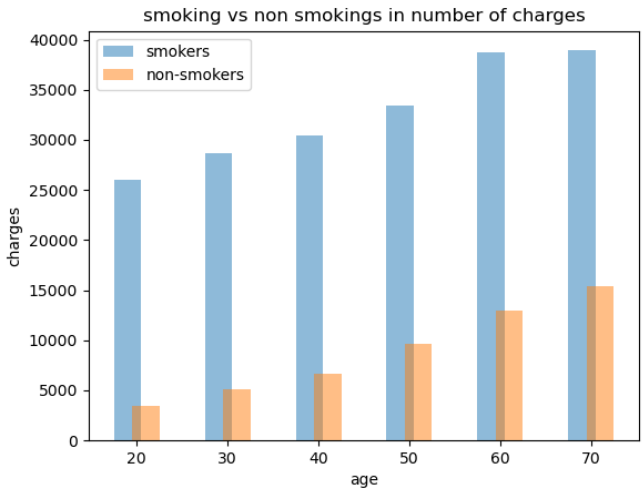
e)

توزیع داده های سن به گونه ای است که به جز بازه ی سنی حدود ۲۰ سال که ۳۵ درصد جامعه را به خود اختصاص داده در بقیه سنین توزیع افراد برابر می باشد.

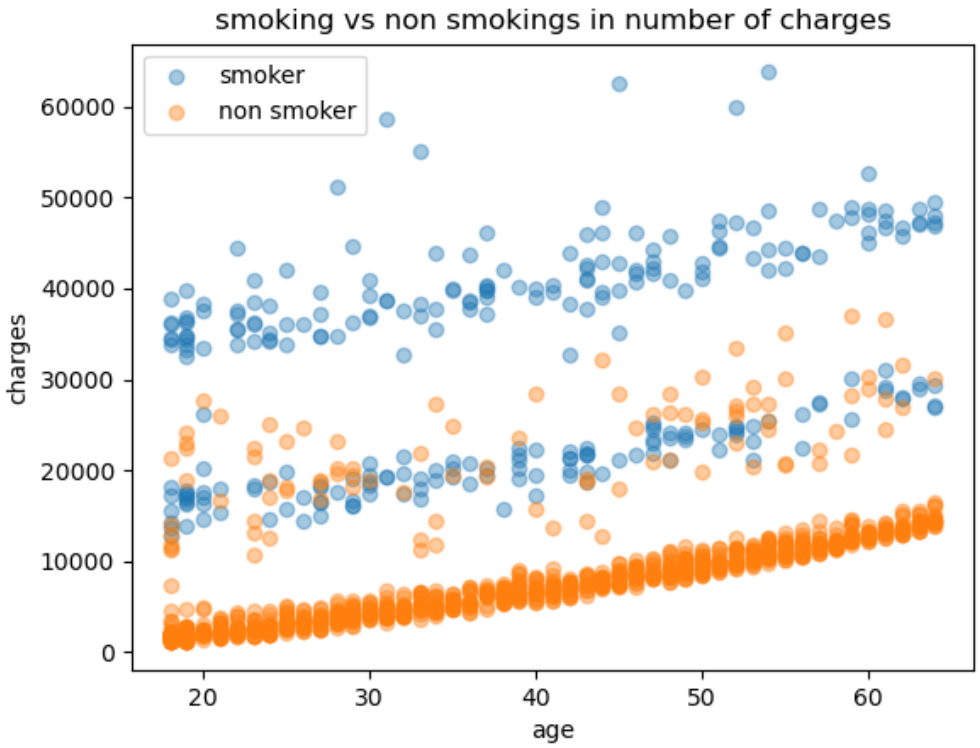


f)

نمودار اولیه بیانگر این است که هیچ وقت smokers از non-smokers میزان دریافتی (charges) کمتری نداشته اند اما برای بررسی دقیق تر می توانیم از نمودار دوم نیز استفاده کنیم.

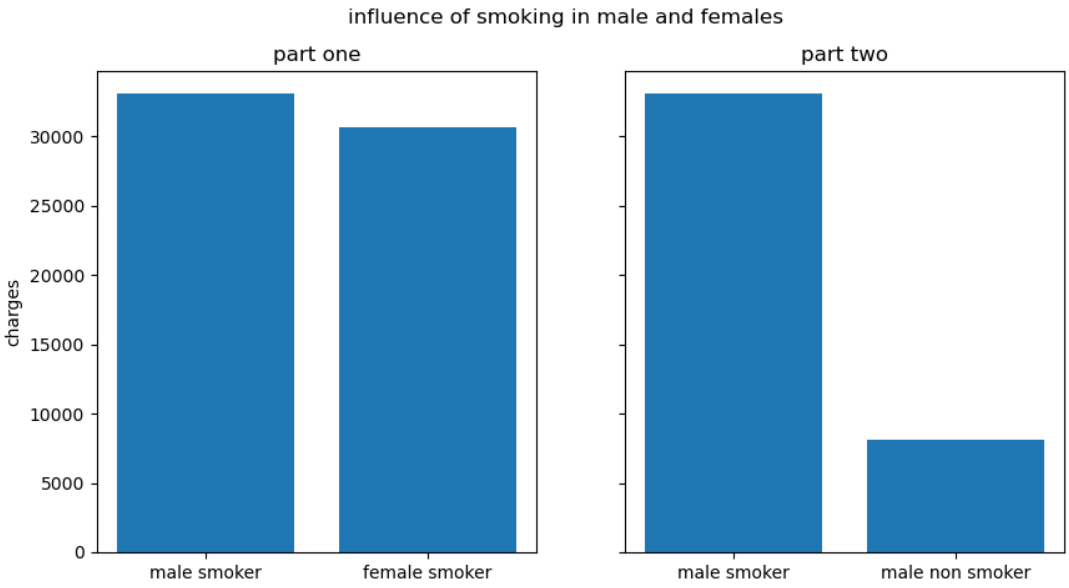


با توجه به توزیع داده ها می توان گفت در همه بازه های سنی کسانی که سیگار مصرف نمی کنند احتمال کمتری دارند اما در بازه ۲۰ تا ۳۰ تعداد افرادی که سیگار مصرف نمی کنند و charge بیشتری داشته اند قابل ملاحظه است همچنین در بازه ۵۰ تا ۶۰ سال نیز این پترن تکرار شده است اما شدت و تراکم آن کمتر است.



g)

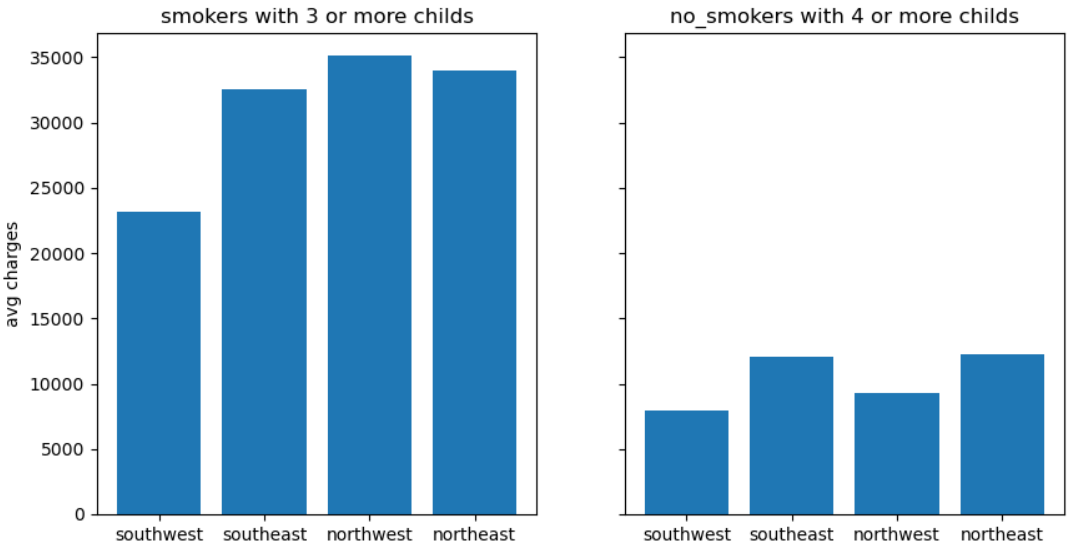
همان طور که در شکل ها مشاهده می کنید میانگین مردان و زنان سیگاری بسیار به یک دیگر نزدیک است (آقایان بیشتر هست البته) اما مقدار میانگین در بین آقاین دسته ای که دخانیات مصرف نمی کنند آمار charges کمتری دارند در میانگین.



h)

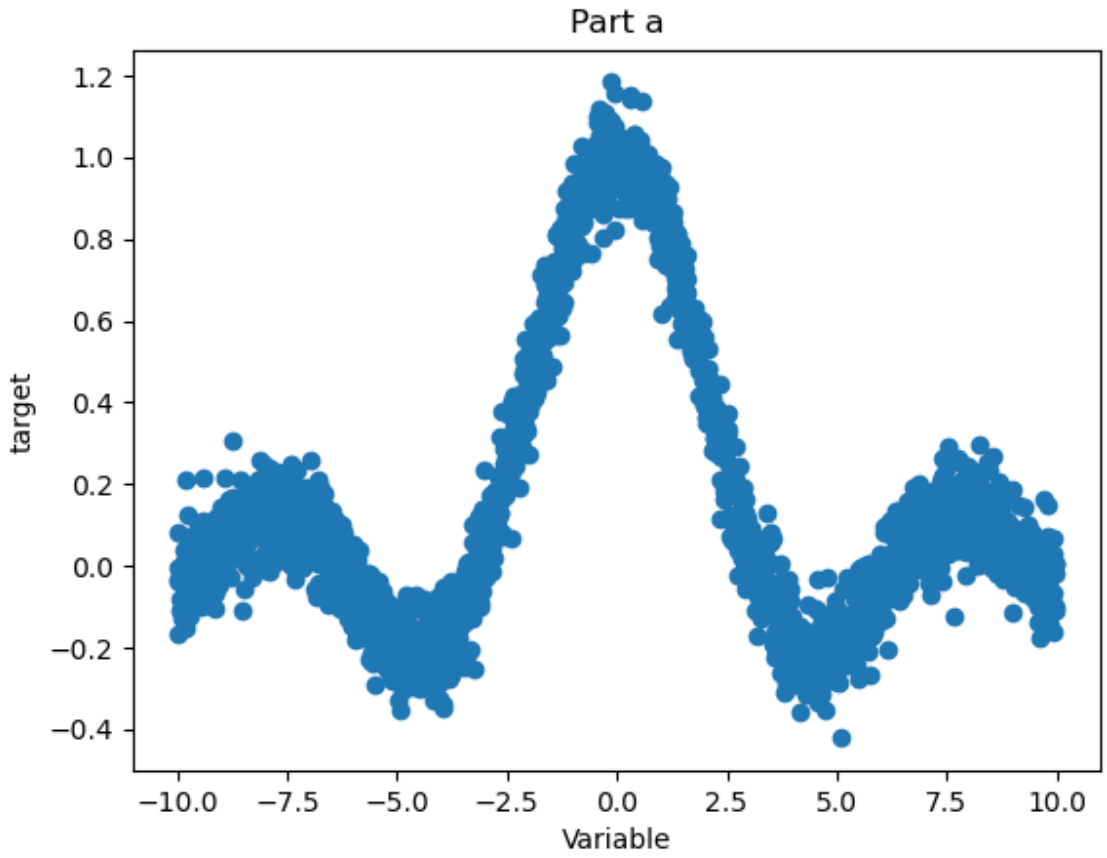
با توجه به شکل های بدست آمده

در همه بخش ها باتوجه به نتایج بدست آمده کسانی که دخانیات مصرف می کنند و بیشتر از ۳ فرزند دارند charges بیشتر خواهند داشت.



Q3)

a)


Part a
target
Variable
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

b)

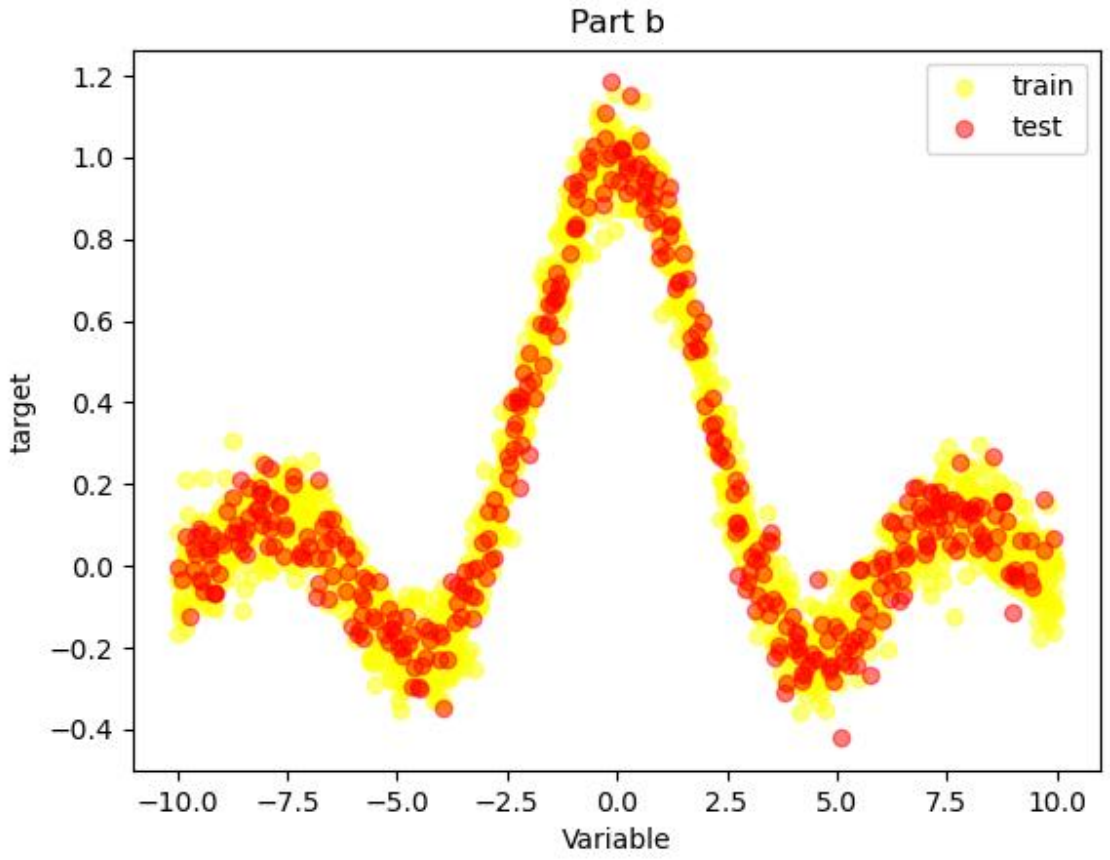
برای اینکه بتوانیم چک کنیم که آیا داده مورد نظر sort شده اند ابتدا دو المان اول را چک کرده که آیا صعودی هستند یا نزولی سپس بقیه المان های داخل سطر نیز باید از این قاعده تبعیت کنند، یعنی اگر صعودی هستند ، صعودی چیده شوند و اگر نزولی هستند نزولی چیده شوند . این فرآیند با یک بار چرخش بر روی داده ها در O(n) بدست می آید.

نتیجه کد

```
C:\Anaconda\python.exe G:/university/master/term3/machineLearning/HW1/codes/Q3.py
its not sorted
```

c)

با توجه به شکل به دست آمده توزیع داده های موجود در آموزش و تست یکسان می باشد. لازم به ذکر است که برای اینکار ما از تابع sample استفاده کرده ایم که عملیات shuffling را به طور ضمنی با انتخاب رندوم از بین داده هال انجام می دهد.



d)

repeat until convergence: {

$$w_j = w_j - \alpha \frac{\partial J(\mathbf{w}, b)}{\partial w_j} \qquad \text{for j = 0..n-1}$$

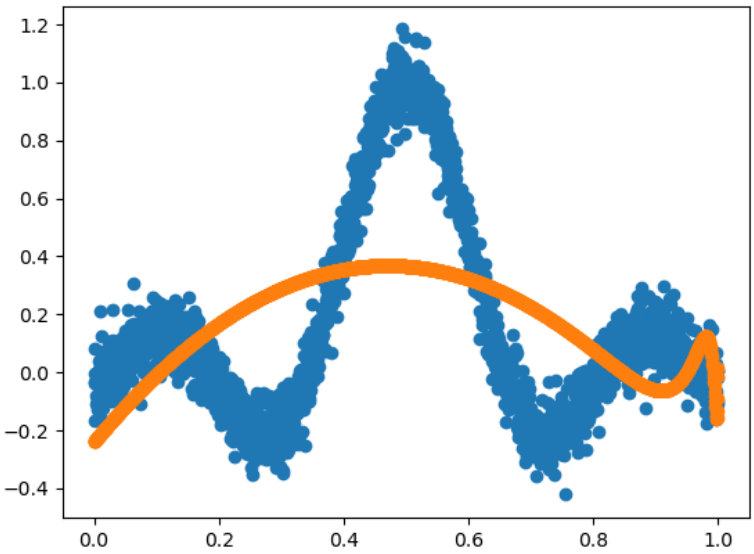
$$b = b - \alpha \frac{\partial J(\mathbf{w}, b)}{\partial b}$$

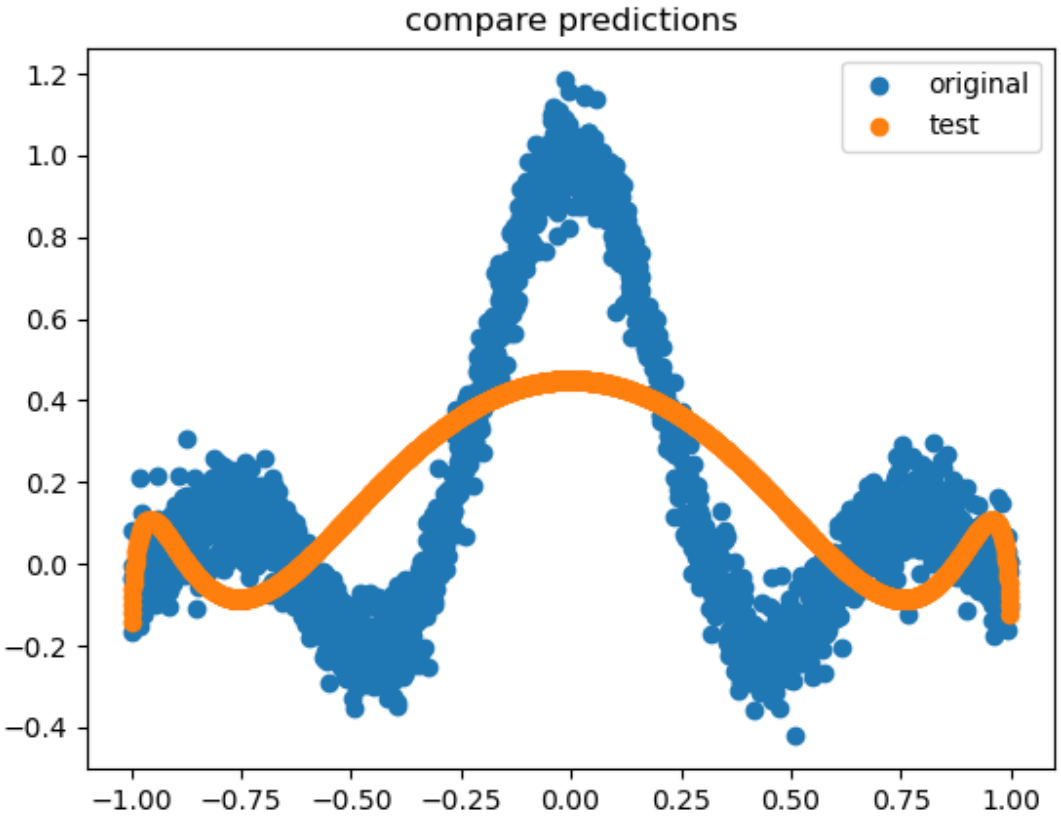
}

$$\frac{\partial J(\mathbf{w}, b)}{\partial w_j} = \frac{1}{m} \sum_{i=0}^{m-1} (f_{\mathbf{w},b}(\mathbf{x}^{(i)}) - y^{(i)}) x_j^{(i)}$$

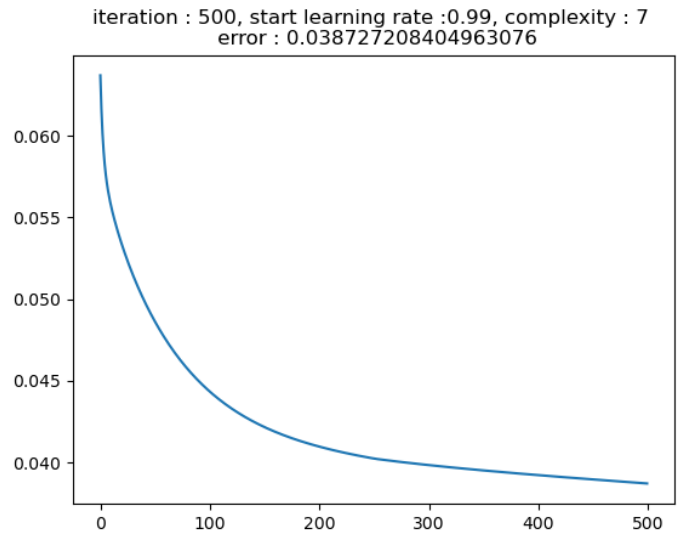
$$\frac{\partial J(\mathbf{w}, b)}{\partial b} = \frac{1}{m} \sum_{i=0}^{m-1} (f_{\mathbf{w},b}(\mathbf{x}^{(i)}) - y^{(i)})$$

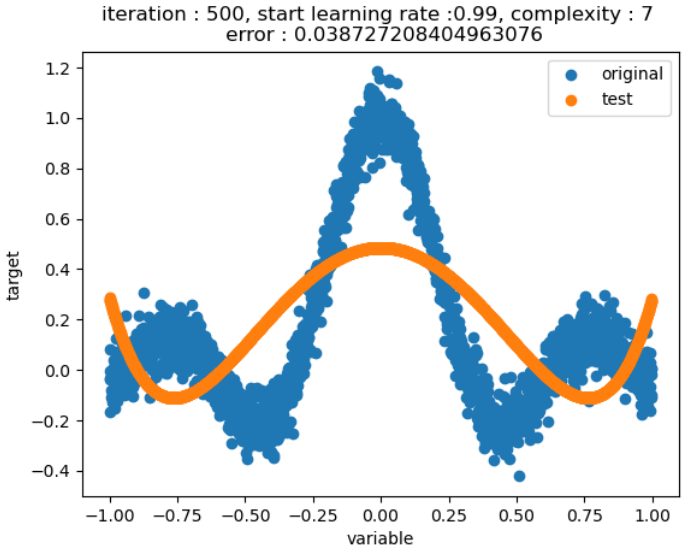
برای پیاده سازی این سوال ابتدا تابع cost function را پیاده سازی کردیم و سپس از روی آن compute_gradient را اجرا می‌کنیم و شیب تغییرات را به دست می آوریم. با استفاده از اعداد بدست آمده در تابع gradient_descent مساله را به شکل مورد نظر حل می کند. لازم به ذکر است تابع نهایی ما نیاز دارد تا ستون های مورد نظر قبل از ورود به تابع ایجاد شده باشند به این معنی که اگر تابع از درجه ۵ می باشد باید ستون های بعدی به آن اضافه شده باشند.



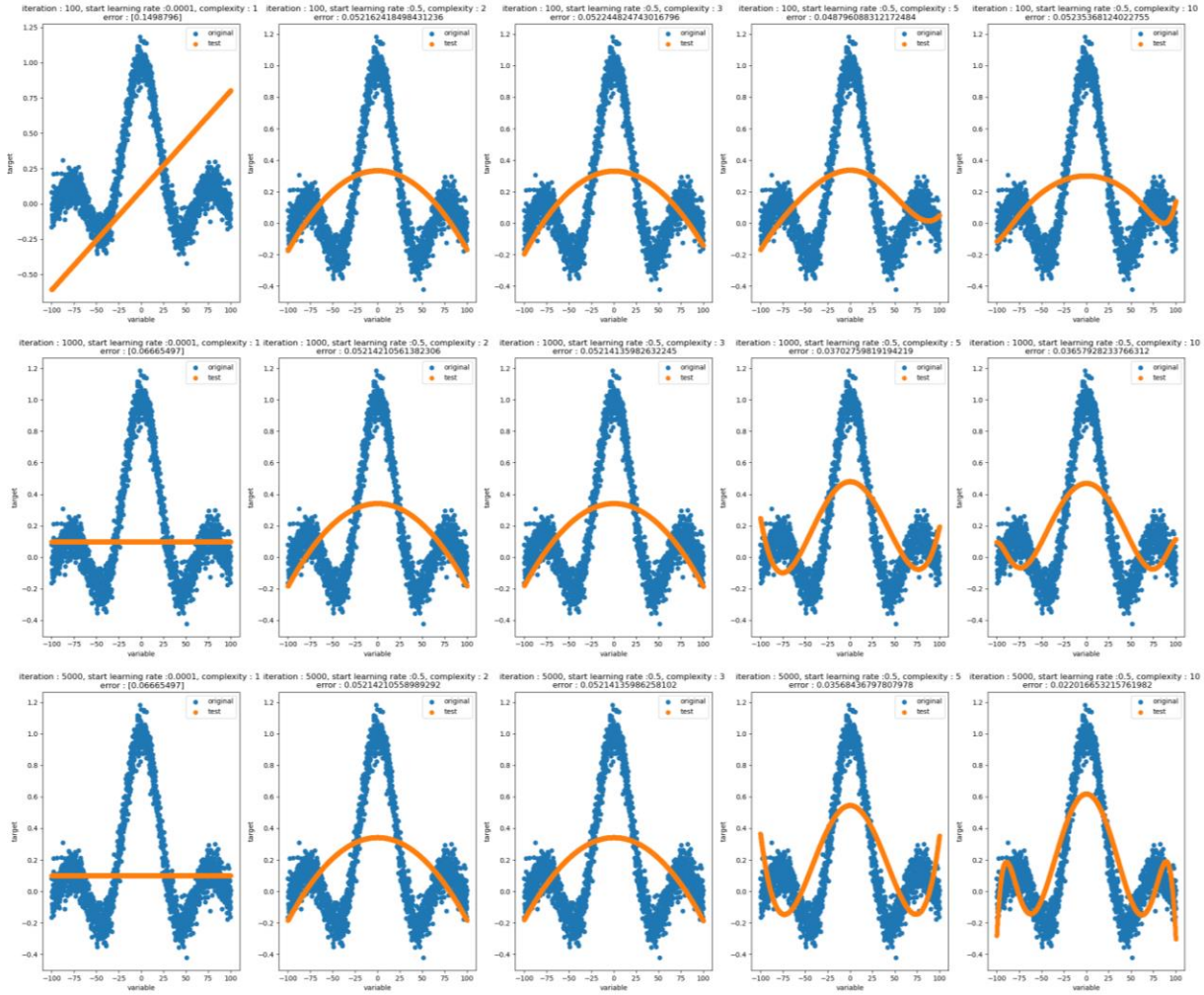


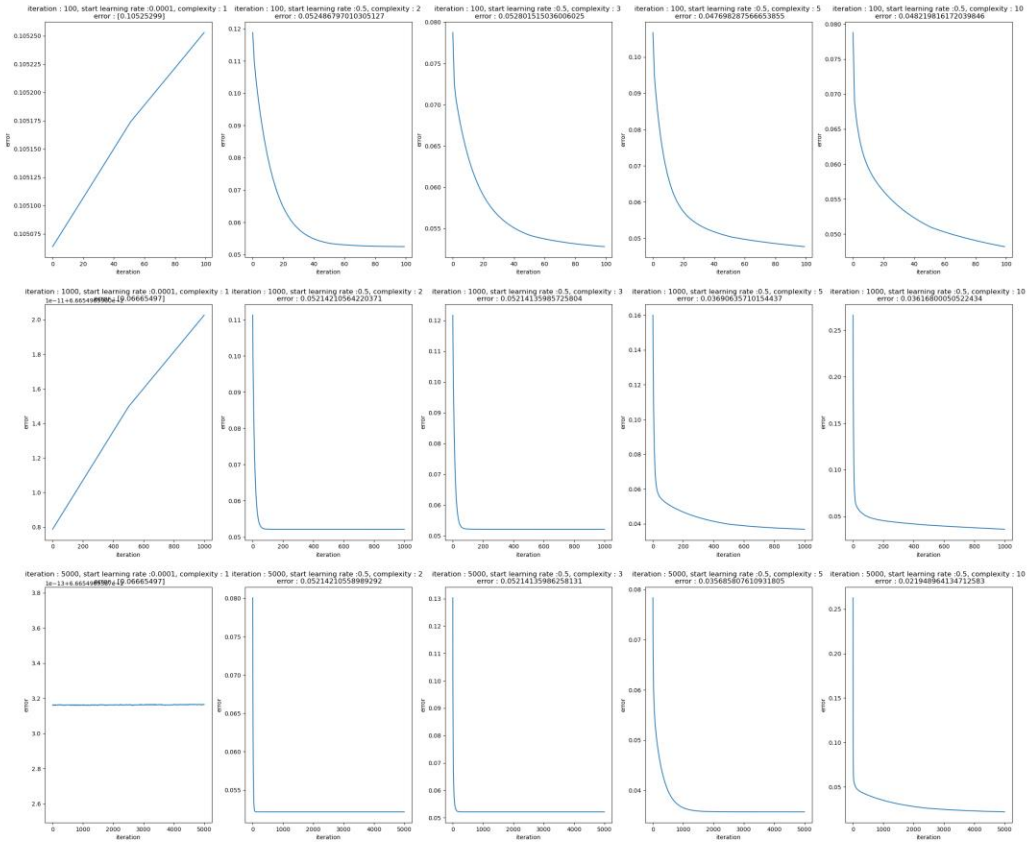
شکل های فوق چند نمونه از اجرای الگوریتم می باشد که شکل آخر حاصل اجرای الگوریتم با ۵ بعد(به علاوه مقدار ثابت که با فرض آن باید گفت شش بعد) در ۵۰۰ گام اجرا شده است. و همچنین به مرور زمان پارامتر آموزش تغییر پیدا می کند.زمانی که ۵۰ درصد iteration ها را انجام داده در 0.75 ضرب می شود و بعد از اینکه ۷۵ درصد iteration ها رو انجام داد به 0.4 مقدار اولیه تبدیل می شود و در ۱۰ درصد انتهایی مقدار 0.1 مقدار اولیه را دارد.(لازم به ذکر است باتوجه به اینکه توان های داده های از ماکسیمم مقدار float قابل محاسبه بیشتر می شد مقادیر ورودی در ابتدای کار بر ۱۰ تقسیم می شوند، حالت های مختلف نورمال سازی بررسی شد اما در نهایت همین scale ساده بهترین نتیجه را داد، اولین نمودار این بخش مربوط به زمانی است که هر کامپلکسیتی را مستقل از بقیه نورمال کنیم این کار بعضا باعث از بین رفتن تقارن در شکل نهایی بدست آمده می شد)


iteration : 500, start learning rate :0.99, complexity : 7
error : 0.038727208404963076
0.060
0.055
0.050
0.045
0.040
0
100
200
300
400
500


iteration : 500, start learning rate :0.99, complexity : 7
error : 0.038727208404963076
original
test
target
variable
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

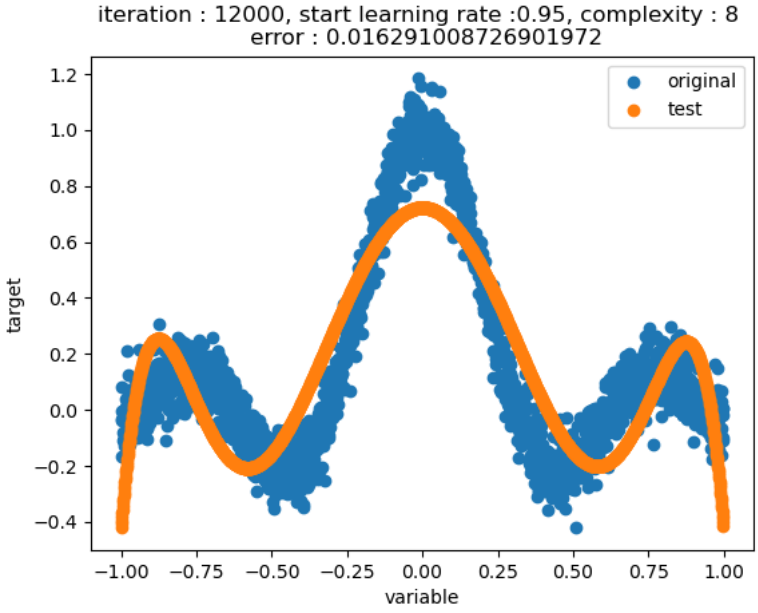
e)

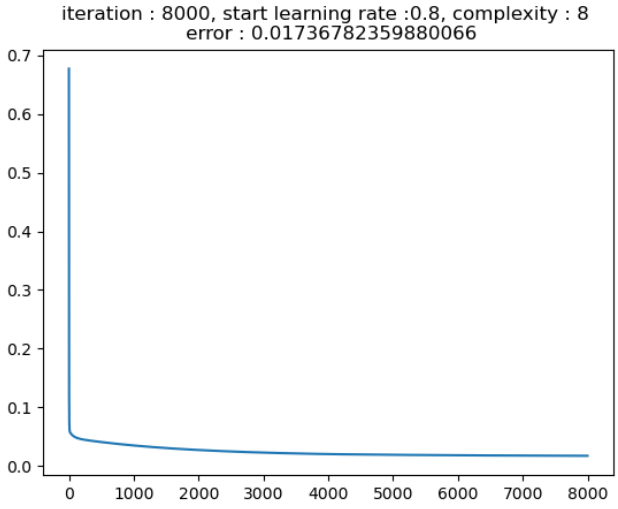

iteration : 100, start learning rate :0.0001, complexity : 1
error : [0.1498796]
iteration : 100, start learning rate :0.5, complexity : 2
error : 0.052162418498431236
iteration : 100, start learning rate :0.5, complexity : 3
error : 0.052244824743016796
iteration : 100, start learning rate :0.5, complexity : 5
error : 0.04879608831212484
iteration : 100, start learning rate :0.5, complexity : 10
error : 0.05235368124022755
iteration : 1000, start learning rate :0.0001, complexity : 1
error : [0.06665497]
iteration : 1000, start learning rate :0.5, complexity : 2
error : 0.05214210561382306
iteration : 1000, start learning rate :0.5, complexity : 3
error : 0.05214135982632245
iteration : 1000, start learning rate :0.5, complexity : 5
error : 0.03702759819194219
iteration : 1000, start learning rate :0.5, complexity : 10
error : 0.03657928233766312
iteration : 5000, start learning rate :0.0001, complexity : 1
error : [0.06665497]
iteration : 5000, start learning rate :0.5, complexity : 2
error : 0.05214210558989292
iteration : 5000, start learning rate :0.5, complexity : 3
error : 0.05214135986258102
iteration : 5000, start learning rate :0.5, complexity : 5
error : 0.035684367978079 78
iteration : 5000, start learning rate :0.5, complexity : 10
error : 0.022016653215761982
original
test
target
variable


iteration : 100, start learning rate :0.0001, complexity : 1
error : [0.10525299]
iteration : 100, start learning rate :0.5, complexity : 2
error : 0.05248679701030512 7
iteration : 100, start learning rate :0.5, complexity : 3
error : 0.052801515036006025
iteration : 100, start learning rate :0.5, complexity : 5
error : 0.047698287566653855
iteration : 100, start learning rate :0.5, complexity : 10
error : 0.048219816172039846
iteration : 1000, start learning rate :0.0001, complexity : 1
iteration : 1000, start learning rate :0.5, complexity : 2
error : 0.05214210564220371
iteration : 1000, start learning rate :0.5, complexity : 3
error : 0.052141359857258 04
iteration : 1000, start learning rate :0.5, complexity : 5
error : 0.03690635710154437
iteration : 1000, start learning rate :0.5, complexity : 10
error : 0.03616800050522434
iteration : 5000, start learning rate :0.0001, complexity : 1
iteration : 5000, start learning rate :0.5, complexity : 2
error : 0.05214210558989292
iteration : 5000, start learning rate :0.5, complexity : 3
error : 0.052141359862581 31
iteration : 5000, start learning rate :0.5, complexity : 5
error : 0.035685807612931805
iteration : 5000, start learning rate :0.5, complexity : 10
error : 0.021948896413471258 3
error
iteration

F)

با توجه به مشاهدات و متقارن بودن تابع نسبت به محور y قطعا توابع با درجه توانی زوج کاندید های مناسب تری هستند نظر شخصی بنده استفاده از درجه ۸ بود با iteration بیشتر که نتیجه نهایی را می توانید در زیر مشاهده کنید



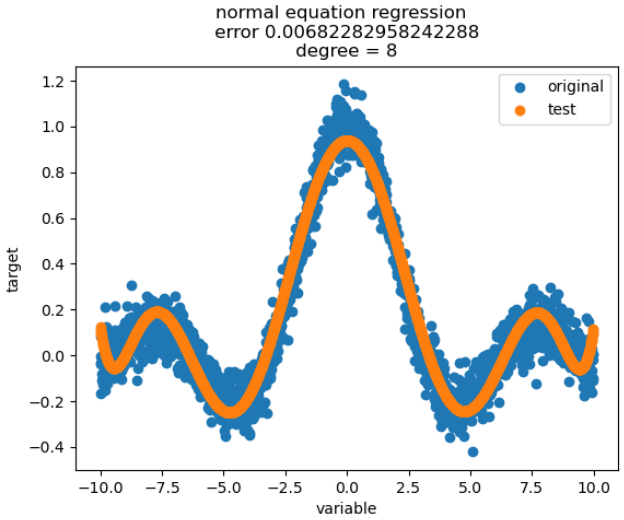


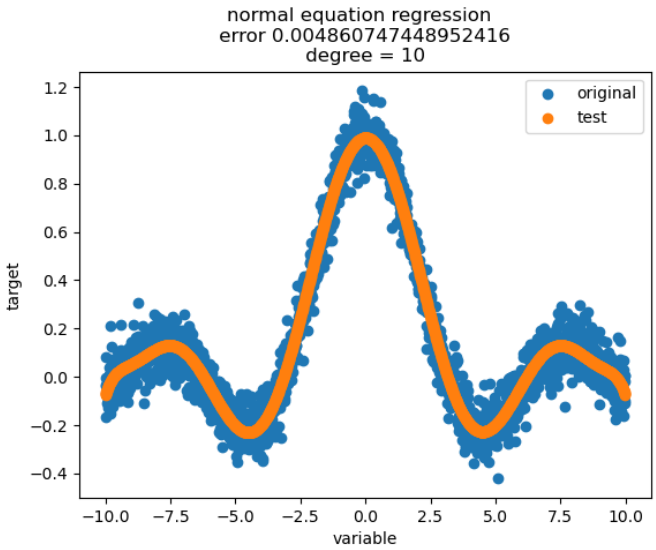
g)

خطای ذکر شده MSE می باشد

همانطور که در اسلاید های درس نیز گفته شد برای استفاده از normal equation به شکل زیر عمل شده است.

$$\boldsymbol{\theta = (X^T X)^{-1} X^T y}$$



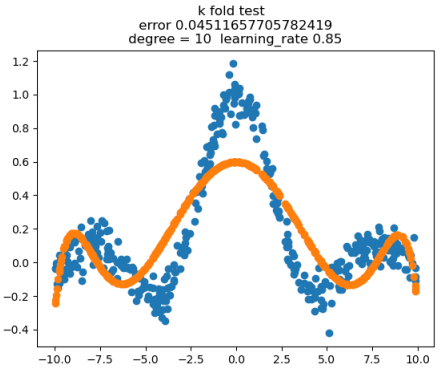


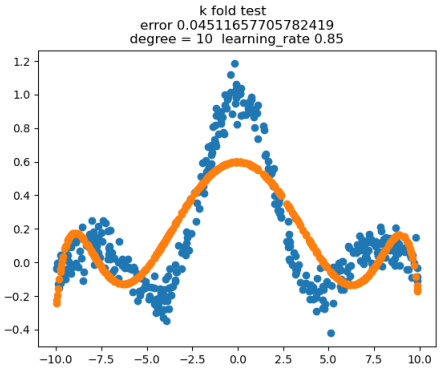
H)

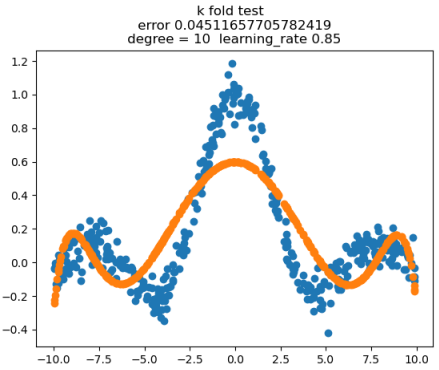
برای حالات مختلف میانگین خطا را در جدول زیر مشاهده می کنید در همه حالات ۲۵۰۰ iteration مورد نظر قرار گرفته همچنین درجه مستقل از بایاس بیان شده است. دلیل استفاده نکردن از ۰.۹۹ در همه حالات این است که این کار در برخی مدل ها باعث ایجاد ناپایداری می شود.

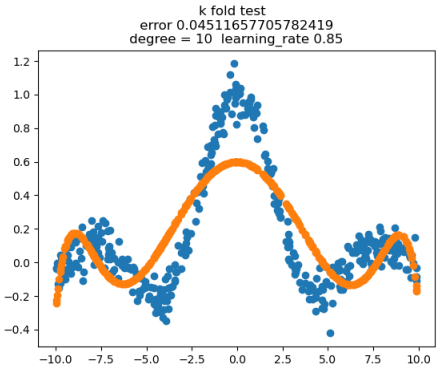
Learning rate (initial)	complexity	err
0.99	5	۰/۰۷۱۷۷۵۱۶۴۹۸۳۴۲۹۸۳
0.99	6	۰/۰۵۹۹۰۱۵۳۳۰۳۴۳۳۴۹۴۴
0.99	8	۰/۰۴۶۳۰۲۳۶۰۴۵۷۶۵۸۱۴
0.75	10	۰/۰۴۷۱۹۰۸۸۷۱۹۴۰۸۲۱۵۴
0.85	10	۰/۰۴۵۶۸۵۸۱۰۱۸۳۲۵۶۱۴
0.75	12	۰/۰۴۸۰۲۶۷۴۷۵۱۴۶۵۲۱۹

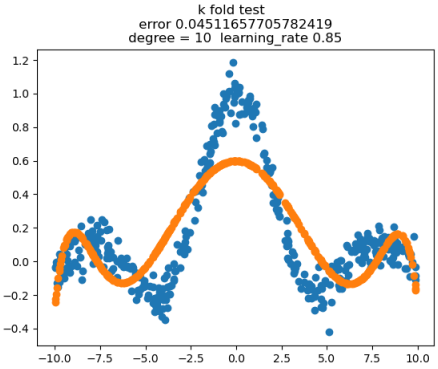
۵ تصویر مرتبط با بهترین نتیجه می باشد که در آن ۰.۸۵ نرخ یادگیری و پیچیدگی ۱۰ می باشد.










k fold test
error 0.04511657705782419
degree = 10 learning_rate 0.85
1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

Q4)

a)

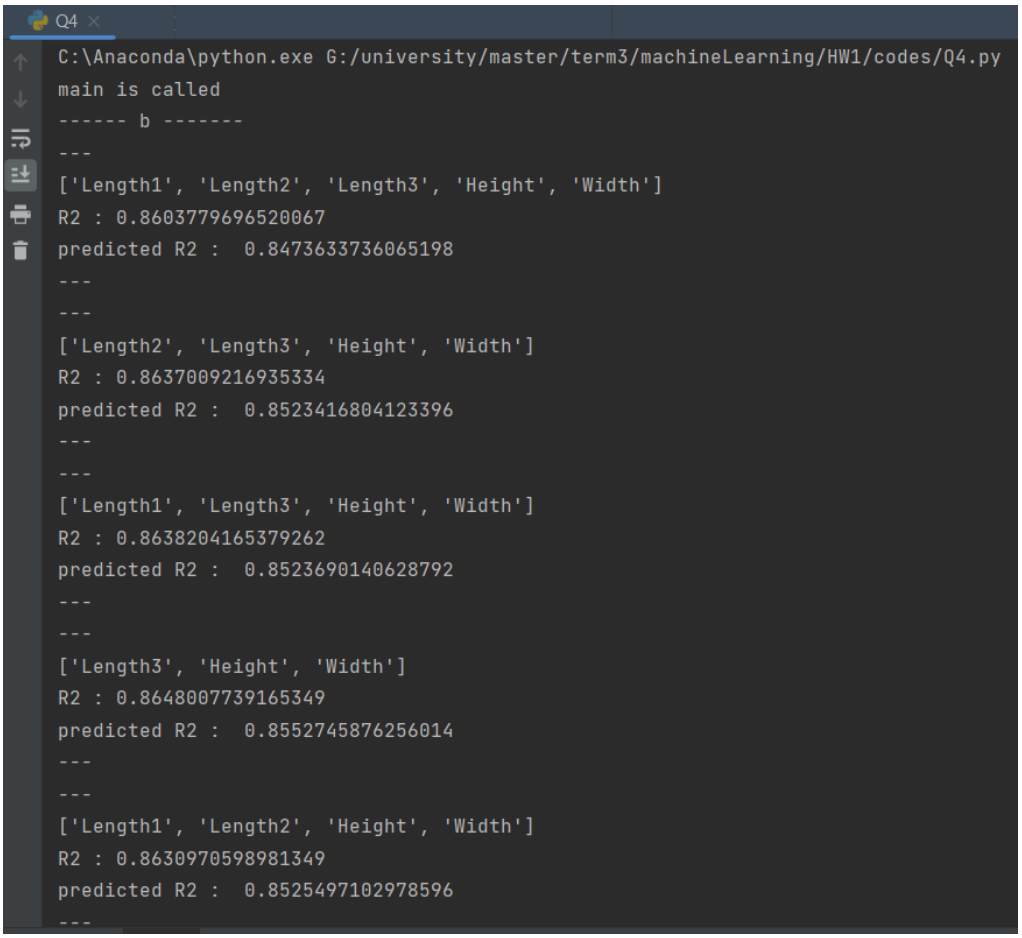
نتیجه اجرای الگوریتم با نرخ یادگیری ۰/۰۱ به شکل زیر بوده است

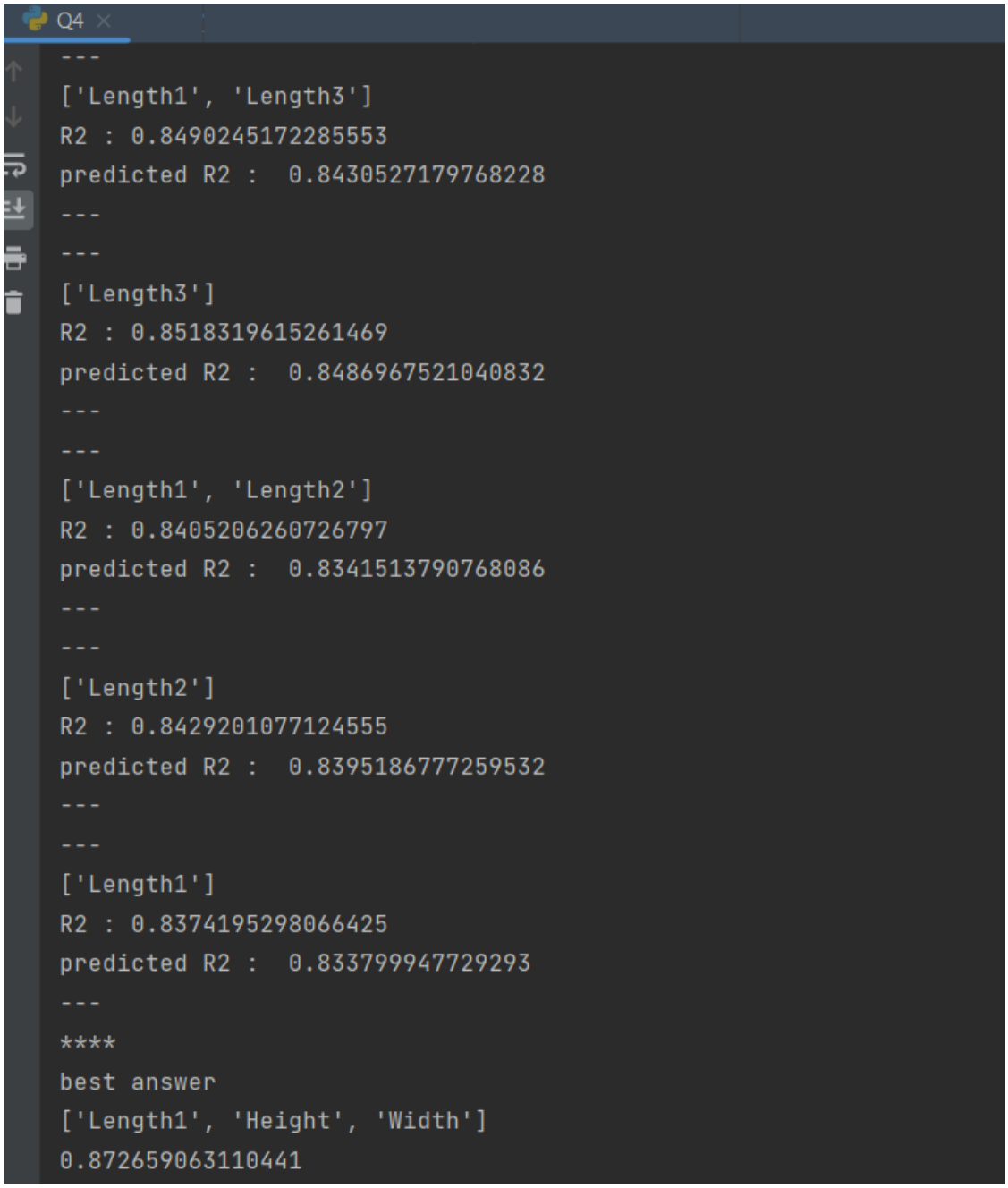
```
C:\Anaconda\python.exe G:/university/master/term3/machineLearning/HW1/codes/Q4.py
main is called
------ a -------
R2 erro :0.8288824795479934
R2 prediction 0.8110443153477049
***

Process finished with exit code 0
```

b)

در این جا یک مجموعه توانی (همه زیرمجموعه های ممکن از مجموعه ویژگی ها) را ایجاد می کنیم و مقدار خطاها را برای هر یک محاسبه می کنیم و درنهایت بهترین را نیز اعلام می کنیم. بدلیل کمبود جا بخشی از خروجی ها را در پایین آورده ایم





در نهایت بهترین جواب lenght1 , Height , Width

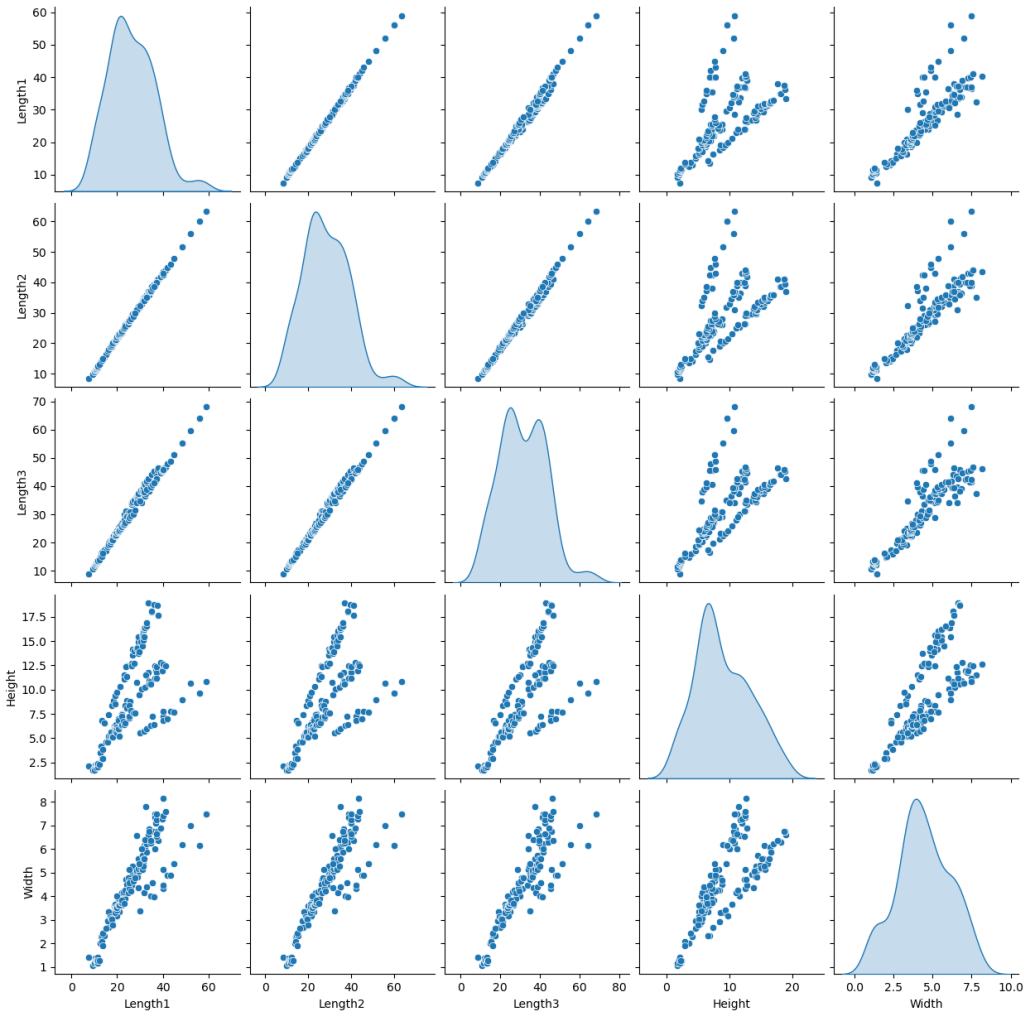
c)

```
Q4
C:\Anaconda\python.exe G:/university/master/term3/machineLearning/HW1/codes/Q4.py
main is called
------ c -------
Weight ~ Length3 + Width + 1
0.8757636700691216
***

Process finished with exit code 0
```

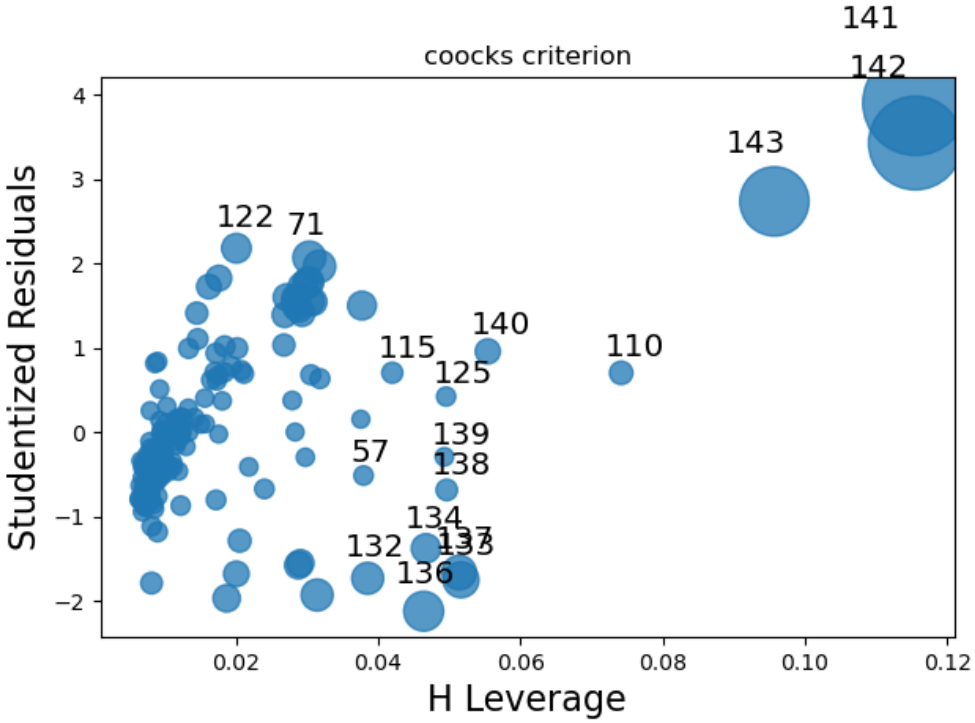
نتیجه این روش تقریبا با روش قبلی از نظر خطا برابر است اما با یک عنصر کمتر که این کار باعث بهبود الگوریتم می شود.

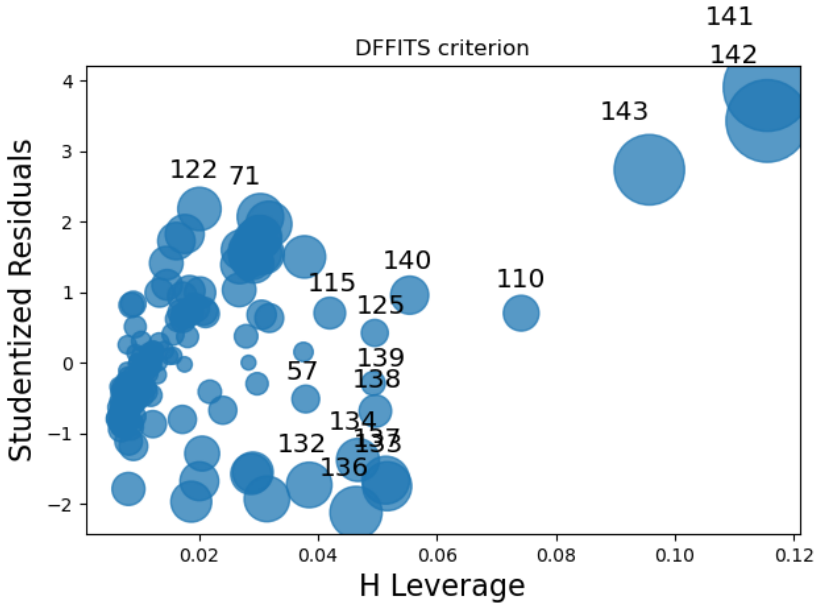
d)



نمودار فوق نسبت دو به دوی پارامتر های ورودی ما را نمایش می دهد از آن جا که تعدادی از خانه های جدول حالت خطی به گرفته اند در نتیجه از روی یکی دیگری را می توان محاسبه نمود و درنتیجه multicollinearity در داده های ما وجود دارد.

e)


coocks criterion
Studentized Residuals
H Leverage
141
142
143
122
71
140
115
110
125
139
57
138
134
132
136
0.02
0.04
0.06
0.08
0.10
0.12


DFFITS criterion
Studentized Residuals
H Leverage
141
142
143
122
71
140
115
110
125
139
57
138
134
132
136
0.02
0.04
0.06
0.08
0.10
0.12

f)

هنوز ایده ای ندارم

g)

از دو فیچری که در بخش c بدست آمد استفاده کردیم و نورمال سازی نیز انجام دادیم و مساله را با درجه ۲ تحلیل نمودیم.

حاصل مدل نهایی به شکل زیر بدست آمد.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                 Weight   R-squared:                       0.966
Model:                            OLS   Adj. R-squared:                  0.965
Method:                 Least Squares   F-statistic:                     862.1
Date:                Sat, 29 Oct 2022   Prob (F-statistic):          1.37e-109
Time:                        13:44:54   Log-Likelihood:                -885.69
No. Observations:                 158   AIC:                             1783.
Df Residuals:                     152   BIC:                             1802.
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
```

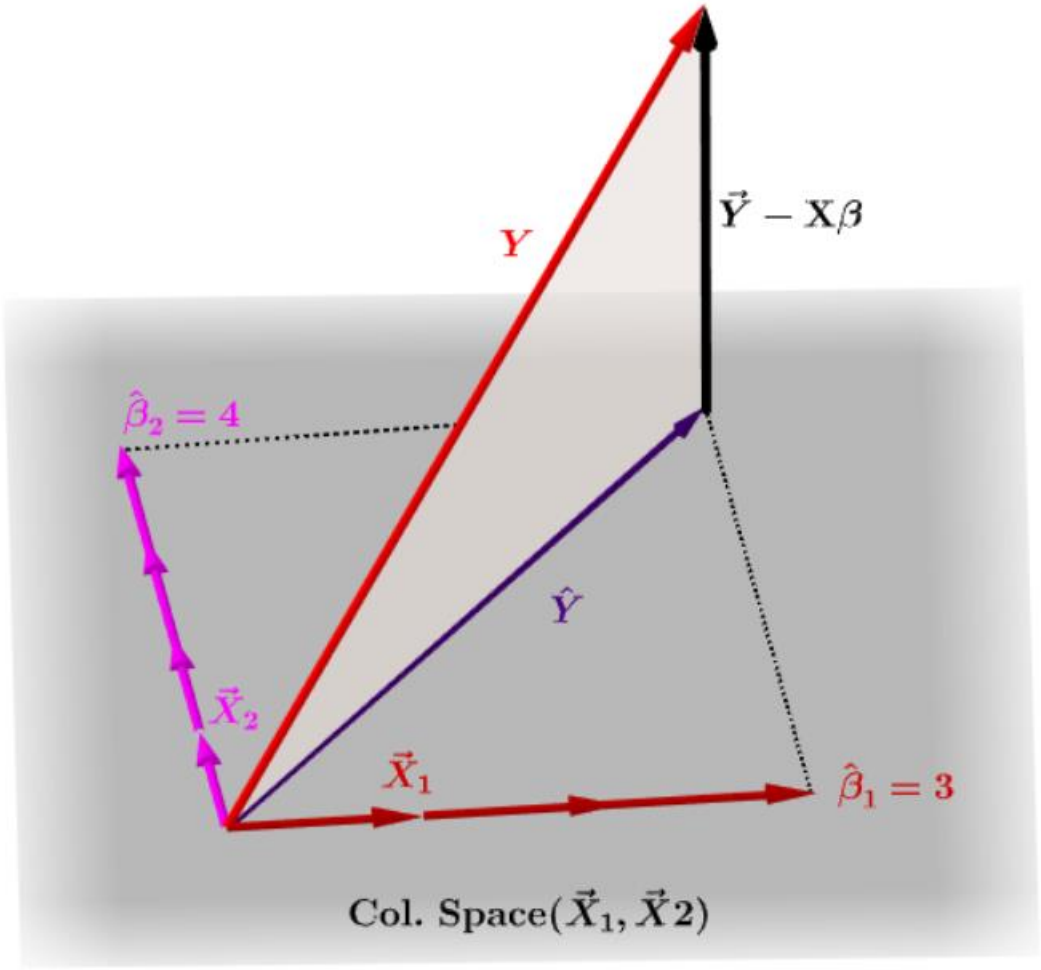
Q5)

a)

ماتریس Hat ماتریسی است که با اعمال شدن به مقدار واقعی داده ورودی مقدار تخمین زده شده توسط رگرسیون را برای آن نشان می دهد. به بیان دیگر مقدار واقعی داده ورودی را بر روی شکل نهایی رگرسیون تصویر می کند.(orthogonal projection)

$$\hat{y} = Hy$$

به کمک این ماتریس می توانیم میزان تاثیر گذاری هر المان در مدل نهایی را با استفاده از مولفه عمود بر مدل نهایی بدست آوریم.



$$\mathbf{H} = \mathbf{X}(\mathbf{X^T X})^{-1}\mathbf{X^T}.$$

همچنین برای بدست آوردن هر یک از β_i ها ازحاصل ضرب زیر کمک می گیریم که همان رابطه ای است که در بالا آورده شده است.

$$(\mathbf{X^T X})^{-1}\mathbf{X^T Y}.$$

b)

step wise regression یک مدل iterative در ساخت رگرسیون می‌باشد که در آن در هر مرحله به انتخاب متغیرهای مستقل (وزن فیچرها) یا حذف آن ها می پردازد. در این روش در هر مرحله سعی در انتخاب بهترین متغیر و یا حذف بدترین متغیر در جهت ساخت مدل رگرسیون ما انجام می شود.

به طور کلی ۳ روش برای استفاده از step wise بیان می شود:(منظور از میزان تاثیر گذاری در حذف یا افزودن متغیرها در اینجا F-test و T-test می باشد.)

۱. Forward selection : در این حالت مدل از حالتی که هیچ متغیری ندارد آغاز می کند و در هر مرحله بهترین وزن یا متغیری که بیشتری ن تاثیر آماری در مدل را دارد به مساله اضافه می کند و این کار را تا زمانی انجام می دهد که به جواب optimal دست یابد.(به بیان دیگر تا زمانی ادامه می دهد که وزن های بعدی تاثیر چندانی در نتیجه نهایی نداشته باشند)
۲. Deleting one at time: بر خلاف روش قبلی فرض می کند همه فیچر ها در مدل رگرسیون حضور دارند سپس هر کدام را به تنهایی تست می کنیم. المانی که کمترین تاثیر را بر روی مدل داشت حذف می کنیم. (لازم به ذکر است که در این حالت یک threshold مد نظر می گیریم که حتما یال حذف شده باید از آن threshold تاثیر کمتری داشته باشد. همچنین مساله زمانی به پایان می رسد که هیچ متغیری کمتر از threshold تاثیرگذاری نداشته باشد.)
۳. Bidirectional elimination: این حالت حالت ترکیبی از یک و دو می باشد که در برخی از گام ها ممکن است که یک متغیر انتخاب شده را حذف کند و متغیری جدید را به مجموعه در همان گام اضافه کند.

c)

Multicollinearity

این مشکل زمانی رخ می دهد که دو یا چند تا از متغیرهای مستقل (ویژگی ها) همبستگی زیادی نسبته به هم داشته باشند و به بیان دیگر به توان از روی یکی دیگری را محاسبه نمود. این مشکل باعث ایجاد خطا چندانی در پیش بینی ما نخواهد شد و همچنین به قابل اطمینان بودن مدل ما نیز خدشه ای وارد نمی کند. اما ممکن است وزن هایی که به متغیرهایی که همبستگی زیاد دارند به شکل غیر قابل تفسیر اختصاص داده شود اما با وجود این شرایط نتیجه مجموع وزن ها به درستی کار می کند.(برای مثال فرض کنید ما یک مفهوم را دو بار به عنوان ویژگی به مدل می دهیم ممکن است که به یکی از آن ها وزن ۰.۳ و به دیگری ۰.۷ داده شود در صورتی که منطقا با توجه به اینکه یک مفهوم بوده اند باید وزنی معادل به آن ها داده می شد)

راه شناسایی

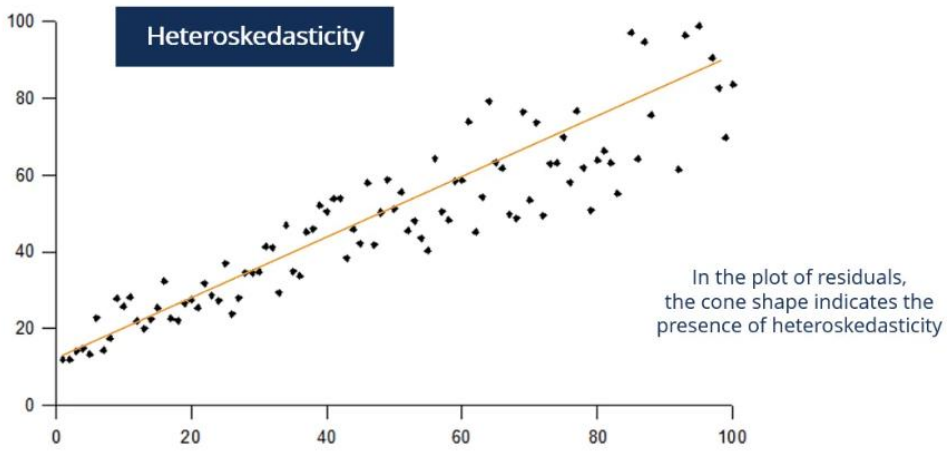
۱. تغییرات زیاد در وزن های رگرسیون با حذف یا افزودن یک متغیر **VIF (Variable Inflation Factors)**
۲. وجود ضرایب نا متناسب نسبت به بقیه ضرایب استفاده شده در مدل رگرسیون
۳. بعضا علامت ضرایب با آن تئوری منطبق نیست و پارامتر های تاثیر گذاردر جهت مثبت، تاثیر منفی به خود بگیرند.

راه حل

همان طور که واضح هست باید ستون هایی که شناسایی شده اند که با بقیه همبستگی دارند را حذف کنیم، در مواردی که همبستگی نه به قدری نیست که بتوانیم حذف و نه به قدری است بتوانیم نگه داریم می توانیم از تعریف متغیر جدید که ترکیبی از این دو می باشد استفاده کنیم و برای آن یک ضریب در مدل رگرسیونمان اختصاص دهیم.

Heteroscedasticity

این پدیده زمانی اتفاق می افتد که خطا (residual) متغیر پیش بینی شـده در بازه های متقاوت یکسـان نباشـد برای مثال داده هایی که از مقدار آلفا بزرگ تر باشند احتمال خطای بیشتری داشته باشند نسبت به آنهایی که کوچکت از آلفا پیش بینی شده اند. یکی از شایع ترین حالات این مشکل این است که نمودار خطا نسبت به مقدار پیش بینی شده به شکل مخروطی در می آید.



این پدیده برای حالتی که از روش ordinary list square (ols) regression برای بدست آوردن ضرایب استفاده می کنیم ایجاد خطا می کند زیرا این روش برای محاسبه وزن ها فرض بر ثابت بودن واریانس متغیرهای آزاد جدید در همه بازه‌های زمانی می‌کند و در صورتی که پدیده Heteroscedasticity اتفاق بیفتد پاسخ این روش دیگر مناسب نخواهد بود.

شناسایی

راه اول برای شناسایی آن نمایش آن و یافتن رابطه مخروطی شکل در داده های موجود است.

راه دوم : heteroscedasticity به دو دسته pure , impure تقسیم می شود در حالت خالص روش ترسیم توصیه می شود اما در حالت impure علت بوجود آمدن مشـکل وجود تعداد زیاد فیچرها یا تعداد خیلی کم فیچ ها اسـت در این حالت می توان با اصـلاح ویژگی ها این مشکل را برطرف نمود

راه سوم : استفاده از p_value که در صورتی که برای ویژگی های انتخاب شده این مقدار پایین باشد می تواند از علائم این مشکل باشد.

راه حل های حل مساله

- استفاده از مقادیر لگاریتمی به عنوان ویژگی که جلوی ایجاد واریانس های زیاد را در حالت نهایی می گیرد. (این کار برای داده هایی که بازه مقادیری گسترده دارند بیشتر توصیه می شود)
- استفاده از تبدیل های غیر خطی برای ستون هایی که این مشکل را دارند (این حالت، در اصل حالت کلی تر پیشنهاد اول می باشد)
- استفاده از روش هایی به جز ols مانند , Heteroscedasticity-consistent standard errors , MINQUE weighted least squares

d)

DFFITS

معیار DFFITS اندازه گیری تغییر در مقدار پیش بینی شده(میزان تاثیرگذاری) برای مشاهده iام است و با حذف مشاهده i ام محاسبه می شود. یک مقدار بزرگ نشان می دهد که داده‌ی ورودی جدید در این فضا برای مدل رگرسیون تاثیر بسزایی داشته است.

DFFIT حاصل مقداری پیشنهادی با حضور داده منهای مقدار پیشنهادی بدون داده مورد نظر است.

S خطای استاندراد در حالت بدون در نظر گرفتن داده

h برابر leverage می باشد که همان ماتریس hat می باشد که از قطر آن در فرمول DFFITS استفاده شده است.

$$\mathrm{DFFIT} = \widehat{y_i} - \widehat{y_{i(i)}}$$

$$\mathrm{DFFITS} = \frac{\mathrm{DFFIT}}{s_{(i)}\sqrt{h_{ii}}}$$

$$h_{ii} = \frac{\partial \widehat{y}_i}{\partial y_i}$$

Cooks distance

یک معیار آماری که با آن در رگرسیون خطی به تشخیص outlier ها می پردازند. در این معیار بر اساس دوعامل اندازه گیری می‌شود. میزان فاصله مقدار تخمین زده شده با مقدار اصلی (residual) و میزان نفوذ leverage که در بالا تعریف شده است.

$$D_i = \frac{\sum_{j=1}^{n}(\widehat{Y}_j - \widehat{Y}_{j(i)})^2}{(p+1)\,\widehat{\sigma}^2}$$

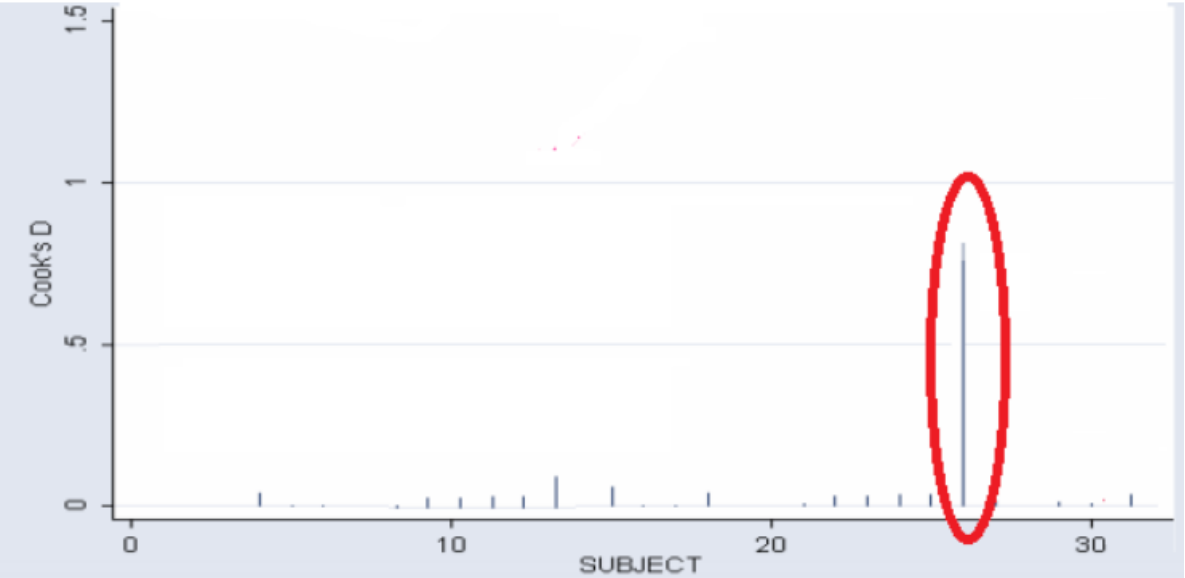
در صورت فرمول مربع خطای بین مقدار تخمین زده شده و مقدار حقیقی با حذف داده مورد نظر از دیتاست جمع می شود.

P تعداد ضرایب موجود در مدل می باشد.

σ نیز انحراف معیار خطا به توان ۲ می باشد.

$$\widehat{\sigma}^2 = \frac{1}{n-m}\sum_{j=1}^{n}\widehat{\varepsilon}_j^{\,2}.$$

البته لازم به ذکر است که حالات مختلفی از این فرمول وجود دارد که فرق هایی جزئی با یکدیگر دارند.



نمونه کشف یک outlier

e)

بخاطر اینکه در این روش ازنمایش یک خط در فضای حالات برای متغیر وابسته استفاده می کنیم، زمانی که واریانس خطا از تابع نرمال پیروی نمی کند به این معنا است که استفاده از رگرسیون خطی اشتباه بوده و جواب مساله در بازه های مختلف ضرایب خطاهای متفاوتی خواهد داشت (میزان خطای پیش بینی شده به شدت متفاوت می باشد) از این رو می بایست از روش های غیر خطی یا روش هایی مانند weighted least squares , Heteroscedasticity-consistent standard errors , MINQUE که این مشکل را ندارند استفاده نمود.