Amirkabir University of Technology
(Tehran Polytechnic)

Department of
Computer Engineering

Course : Machine Learning

Homework 1

Najmeh Mohammadbagheri

99131009

بخش تشریحی

سوال اول

یادگیری نظارتی: در این روش از یادگیری، داده‌های آموزشی دارای برچسب هستند و در واقع آموزش پارامترها توسط این برچسب‌ها انجام می‌شود. در این آموزش برچسب‌ها نقش معلم یا ناظر را دارند و به همین دلیل معروف به یادگیری بانظارت است.

یادگیری نیمه نظارتی: در این روش تعداد کمی از داده‌ها دارای برچسب‌اند و تعداد زیادی از آنها برچسب ندارند و الگوریتم‌های یادگیری نیمه نظارتی به گونه‌ای هستند که یادگیریشان را با هردو دسته از داده‌های برچسب‌دار و بدون برچسب انجام می‌دهند.

یادگیری بدون نظارت: در این روش هیچ برچسبی نداریم و مدل باید خودش داده‌ها را براساس ویژگی‌هایشان جدا کند. این روش بیشتر در مسائل خوشه‌بندی مشاهده می‌شود و چون برچسبی برای نظارت درست بودن عملکرد مدل وجود ندارد به یادگیری بدون نظارت معروف است.

یادگیری تقویتی: در این یادگیری فقط یک سیگنال به عنوان معلم داریم که تنها بیان می‌کند خروجی تابع مطلوب هست یا خیر، جواب اصلی را نمی‌گوییم. این روش در عامل‌ها و رباتیک خیلی موضوعیت دارد. درواقع این روش از یادگیری نظارتی ضعیف‌تر است و زمانی سراغ این روش می‌آییم که نتوانیم از نظارتی استفاده کنیم. یعنی داده‌ها برچسب نداشته باشند.

یادگیری عمیق: یادگیری عمیق یک عمل هوش مصنوعی است که نحوه‌ی کارکردن مغز انسان را تقلید می‌کند تا داده‌ها را پردازش کند، الگوها را شناسایی کند و تصمیم‌گیری انجام دهد. یادگیری عمیق زیرمجموعه‌ای از یادگیری ماشین است که یادگیری را روی داده‌های بدون ساختار و برچسب انجام می‌دهد. این روش به عنوان یادگیری عصبی عمیق و شبکه عصبی عمیق نیز شناخته شده‌است.

رگرسیون: در رگرسیون سعی براین است که یک عدد را به عنوان خروجی تخمین بزنیم. در رگرسیون، داده ها دارای یک برچسب عددی هستند و مدل باید به گونه‌ای آموزش ببیند که در نهایت با دریافت یک داده‌ی جدید که قبلا آن را ندیده است یک مقدار مناسب تخمین بزند. رگرسیون بیشتر شبیه یک تابع است. ($y = f(x)$)

یادگیری برخط: در این روش داده‌ها به صورت متوالی و پشت سرهم در دسترس قرار می‌گیرند. این داده‌ها برای به روزرسانی بهترین مدل برای داده‌های آینده، در هر مرحله از یادگیری استفاده می‌شود. یادگیری برخط

یک روش رایج است و در مواردی از یادگیری ماشین مورد استفاده قرار می گیرد که از نظر محاسباتی آموزش کل مجموعه داده غیرممکن است و نیاز به الگوریتم های خارج از هسته است. همچنین در شرایطی که لازم است الگوریتم به صورت پویا با الگوهای جدید داده‌ها سازگار شود، یا وقتی داده‌ها خود به عنوان تابعی از زمان تولید می شوند، مورد استفاده قرار می‌گیرد.

یادگیری فعال: در مواردی که داده‌های بسیار زیادی داریم که تعداد زیادی از آنها برچسب ندارند از این یادگیری استفاده می‌کنیم. در واقع در این روش، از داده‌های برچسب‌خورده استفاده می‌شود تا مدل یادگیری را انجام دهد سپس از همین مدل استفاده می‌شود تا تعدادی از داده‌های بدون برچسب، برچسب گذاری شوند. بعد از آن دوباره یادگیری روی داده‌های جدید برچسب خورده و داده‌های قبلی انجام می‌شود. این فرایند انقدر تکرار می‌شود تا تمام داده‌ها برچسب بخورند.

دسته‌بندی: در مسایل دسته‌بندی داده‌های ما داری برچسب هستند که مشخص می‌کند هر داده مربوط به کدام دسته است و تعداد کل دسته‌ها چندتا است. در این مسائل مدل به گونه‌ای آموزش می‌بینند که با دریافت داده‌ی جدید بتواند حدس بزند آن داده به کدام دسته تعلق دارد؛ یعنی به داده‌های کدام دسته بیشترین شباهت را دارد. دسته‌بندی‌ها از الگوریتم‌های بانظارت هستند.

خوشه‌بندی: برخلاف دسته‌بندی، در مسائل خوشه‌بندی داده‌ها دارای برچسب نیستند و یادگیری به صورت بدون نظارت انجام می‌شود. در حل این مسائل الگوریتم‌ها به گونه‌ای هستند که براساس فاصله یا دیگر پارامترها داده‌های نزدیک بهم را در یک خوشه قرار می‌دهند. تعداد این خوشه‌ها از قبل مشخص نیست و این پارامتر را نیز باید خود مدل یاد بگیرد. پس از اینکه خوشه‌ها مشخص شدند داده‌ی جدید که وارد شد معمولا با یک معیاری با نماینده‌ی هر خوشه مقایسه می‌شود و در نهایت به شبیه‌ترین خوشه وارد می‌شود.

بیش برازش و کم برازش: در مرحله‌ی یادگیری مدل باید پارامترهای مدل را با استفاده از داده‌های آموزشی محاسبه کرد؛ بنابراین لازم است که مدل رفتار داده‌ها را یاد بگیرد. اگر زیاد از حد مدل یادگیری انجام دهد و سعی کند که داده‌ها را حفظ کند و مدل پیچیده‌ای در نهایت شود، به این حالت می‌گوییم مدل بیش برازش شده است. از طرف دیگر اگر مدل کمتر از میزان لازم رفتار داده‌ها را یاد بگیرد، یک مدل ساده باقی می‌ماند و خطای زیادی دارد. در این حالت می‌گوییم کم برازش رخ داده است. در حالت کم برازش بایاس مدل زیاد است و در حالت بیش برازش واریانس بالاست که هردو خوب نیستند و باید تعادل بین این دو معیار برقرار کرد.

سوال دوم

همبستگی به این معناست که با تغییر میزان شدت یک ویژگی در داده‌ها، ویژگی دیگر هم تغییرات معنادار دارد. به طور مثال با افزایش ویژگی یک، میزان ویژگی دو نیز افزایش (یا کاهش) منظمی دارد.

همبستگی بین دو ویژگی را میتوان از ماتریس کواریانس تشخیص داد. اگر منفی باشد این مقدار، یعنی با افزایش یک ویژگی، ویژگی دیگر کاهش میابد یا به عبارتی تغییرات این دو ویژگی خلاف جهت هم است و اگر مثبت باشد یعنی تغییرات این دو ویژگی هم جهت و اگر صفر باشد یعنی هیچ همبستگی ای بین این دو ویژگی وجود ندارد.

سوال سوم

هر سه معیار میانگین تفاوت مقدار پیش‌بینی شده با مقدار واقعی را محاسبه می‌کنند و علامت هر سه مثبت است. در هر سه معیار هرچه حاصل جمع ها بیشتر باشد خطای مدل بیشتر است.

در یک مقایسه‌ی یکسان مقدارها بصورت مقابل است: $MSE > MAE > RMSE$

باید دقت شود که در یک یادگیری تنها از یک معیار استفاده می‌شود.

حساسیت MSE به داده‌های نویزی بیشتر است. چون در محاسبه‌ی خطا، فاصله‌ی بین داده‌ی نویزی و واقعیت را بیشتر از دو روش دیگر بولد می‌کند(چون دارد به توان دو می‌رساند). بعد از MSE معیار MAE به داده‌های پرت و نویزی حساس است.

سوال چهارم

در روش گرادیان نزولی باید نرخ یادگیری مناسب را پیدا کنیم همچنین باید تعداد تکرار مناسب را بیابیم اما در روش معادله نرمال این مشکلات را نداریم.

در روش معادله نرمال بحث معکوس پذیر نبودن ماتریس داده‌ها وجود دارد ولی در گرادیان این محدودیت نیست.

گرادیان نزولی با داده‌هایی که تعداد ویژگی‌های زیادی دارند خوب کار می‌کند ولی معادله نرمال با تعداد کم ویژگی خوب کار میکند.

پیچیدگی زمانی گرادیان نزولی $kn2$ است که k تعداد تکرار و n تعداد ویژگی است اما پیچیدگی زمانی معادله نرمال n^3 است.

در معادله نرمال نیازی به نرمال سازی داده‌ها نیست درحالی که در گرادیان نزولی میتواند استفاده شود.

سوال پنجم

The “LASSO” stands for **L**east **A**bsolute **S**hrinkage and **S**election **O**perator.

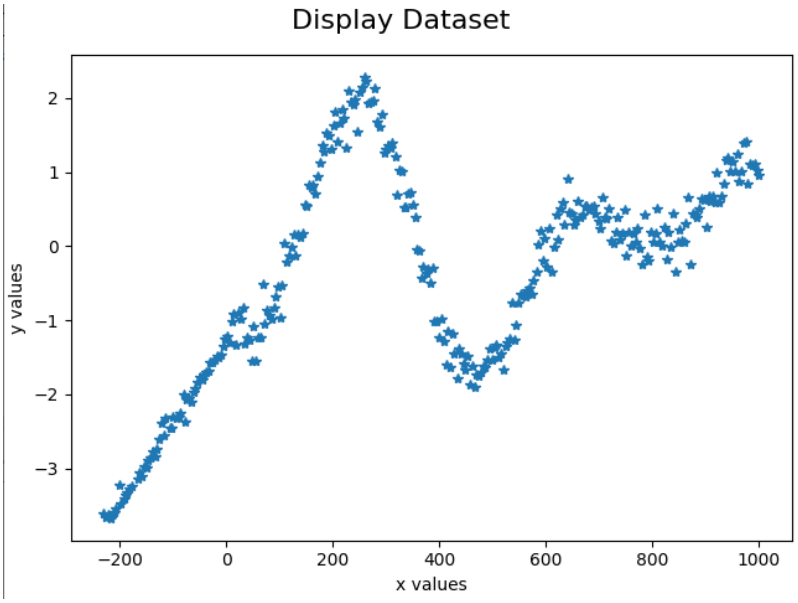
این رگرسیون یک روش رگولاریزیشن کردن است و برای دقت بیشتر پیشبینی از این رگرسیون استفاده می‌شود. این روش زمانی کاربرد دارد که داده‌ها به سمت یک نقطه‌ی مرکزی به عنوان میانگین متمایل می‌شوند. رگرسیون lasso بیشتر مدل‌ها را تشویق می‌کند که ساده باشند و پارامترهای کمتری داشته باشند. تفاوت این روش با رگرسیون خطی در این است که در lasso به ازای پیچیده شدن مدل یک جریمه داریم و باید در طول یادگیری حواسمان به بیش برازش باشد اما در رگرسیون خطی مدل میتواند خیلی پیچیده شود و در واقع بیش برازش رخ دهد. چون در ازای پیچیده شدن و پارامترهای بالا گرفتن جریمه نمی‌شود. فرمول تابع هزینه در رگرسیون lasso در زیر آمده است. بتا همان پاراتر مساله است.

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

بخش پیاده‌سازی

بخش اول

یک.

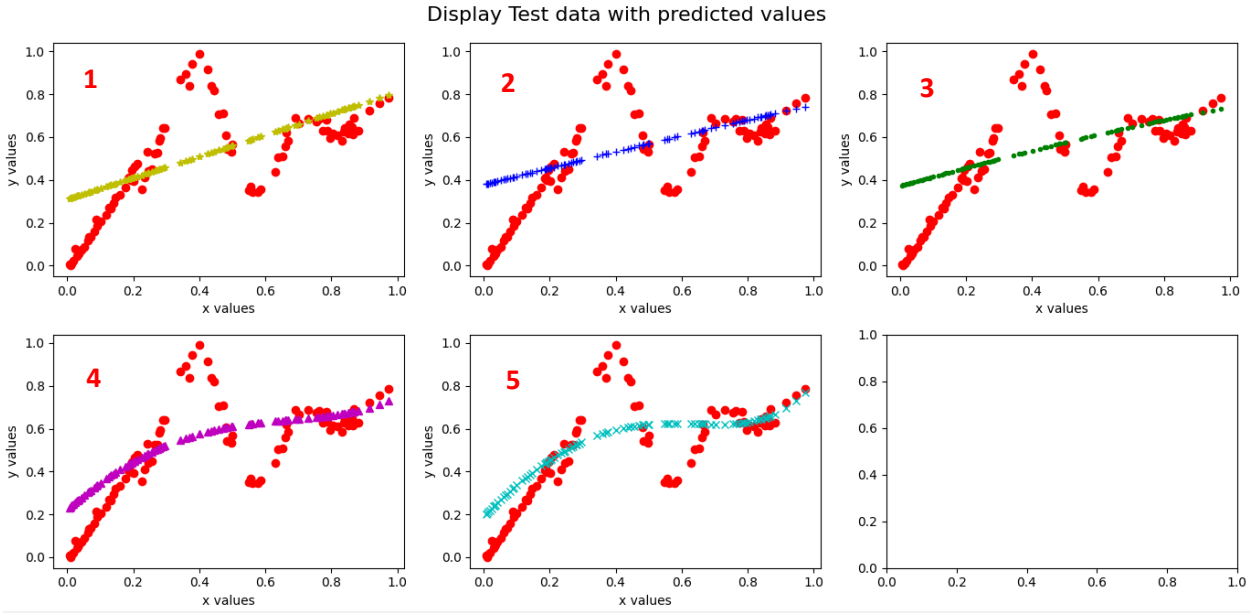


دو. شافل کردن برای داده‌هایی که ترتیب در آنها وجود دارد انجام می‌شود. همانطور که می‌دانیم برای یادگیری پارامترهای مساله داده‌ها را به دو قسمت آموزش و تست تقسیم می‌کنیم و از داده-های تست برای یادگیری استفاده نمی‌کنیم. به همین دلیل اگر در داده‌های مساله ترتیب وجود داشته باشد و ما قسمتی از آن را برای یادگیری جدا نکنیم، مدل یادگیری خوبی ندارد. مثلا فرض کنیم یکی از ویژگی داده‌ها قد باشد و ترتیب داده‌های مساله بصورتی باشد که قد نمونه‌ها بصورت افزایشی باشد، اگر ما ۲۰٪ این داده‌ها را جدا کنیم مثلا از قد ۱۷۰ به بعد در داده‌های تست قرار می‌گیرد و در داده‌های آموزشی وجود ندارد و ممکن است از همین قد ۱۷۰ به بعد نحوه‌ی تاثیر قد در مدل متفاوت از قدهای کمتر از ۱۷۰ باشد. پس در فرایند یادگیری مدل ما نمی‌تواند پیش-بینی بخشی از داده‌ها را که ندیده است درست انجام دهد. برای جلوگیری از این خطا، قبل از جدا کردن داده‌های آموزش و تست بهتر است عمل شافل کردن را انجام دهیم. یعنی ترتیب داده‌ها را بهم بزنیم و به صورت تصادفی داده‌ها را مرتب کنیم. مجموعه داده‌ی اول نیاز به شافل کردن دارد، چون در مقادیر x ترتیب وجود دارد و داده‌ها بصورت نزولی از ۱۰۰۰ به ۲۳۰- رفته‌اند.

نرمال سازی داده‌ها به این منظور انجام می‌شود که داده‌ها به دامنه‌ای برده شوند که کار با آنها راحت باشد. بطور مثال برای محاسبه‌ی خطا اگر مقادیر در مقیاس ۱۰۰۰۰ باشند و هر داده به توان دو برسد، اعداد خیلی بزرگ می‌شوند و کار محاسبات سنگین می‌شود. در این موارد، داده‌ها را به یک بازه‌ی کوچکتر میبریم. مجموعه داده‌ی اول نیاز به نرمال سازی نیز دارد. چون مقادیر بزرگ هستند و اگر به بازه‌ی ۱- تا ۱ انتقال یابند راحت‌تر میتوانیم عملیات را روی آن انجام دهیم.

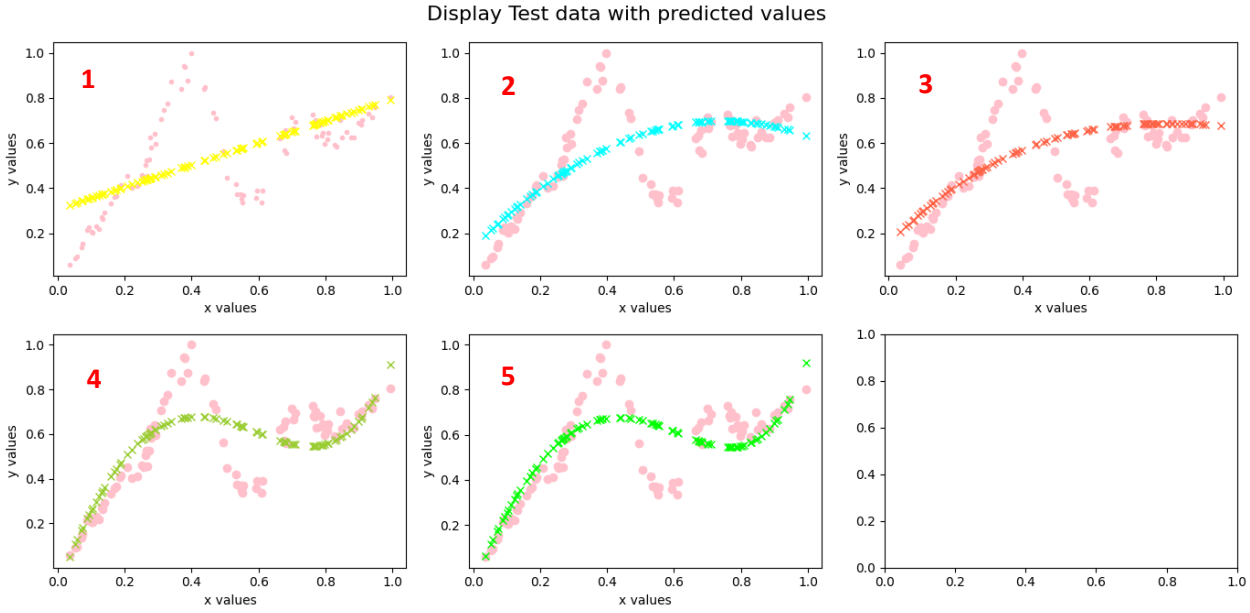
سه. از این قسمت گزارش تا انتها در نمودارهای به منظور مشاهده‌ی دقیق‌تر عملکرد مدل، تنها داده‌ی تست و میزان پیش‌بینی شده کشیده شده‌است.

plot	degree	Repeat	Learning rate	MSE
1	1	1000	0.01	۰.۰۳۳۳۱
2	2	1000	0.1	۰.۰۳۷۶۸
3	3	1000	0.1	۰.۰۳۷۰۲
4	4	1000	1	۰.۰۲۳۸۱
5	5	1000	1	۰.۰۲۰۷۶

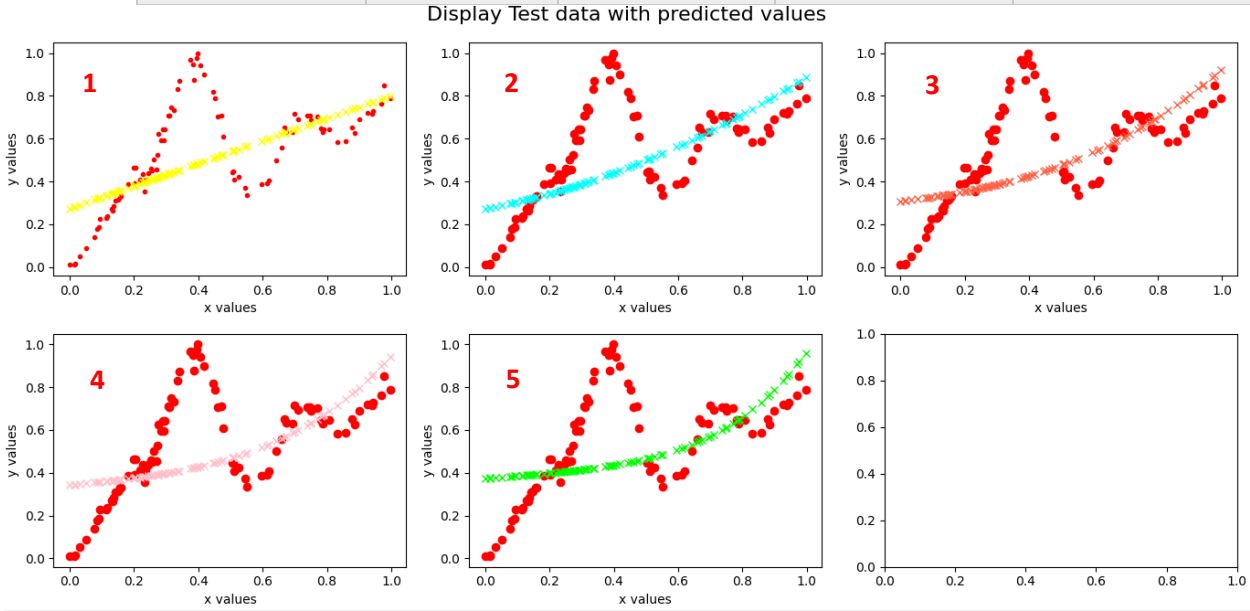


---

plot	degree	Repeat	Learning rate	MSE
1	1	15000	0.01	۰.۰۳۰۸۰
2	2	15000	0.1	۰.۰۲۵۷۸
3	3	15000	0.1	۰.۰۲۴۷۶
4	4	15000	1	۰.۰۱۷۷۴
5	5	15000	1	۰.۰۱۸۱۹



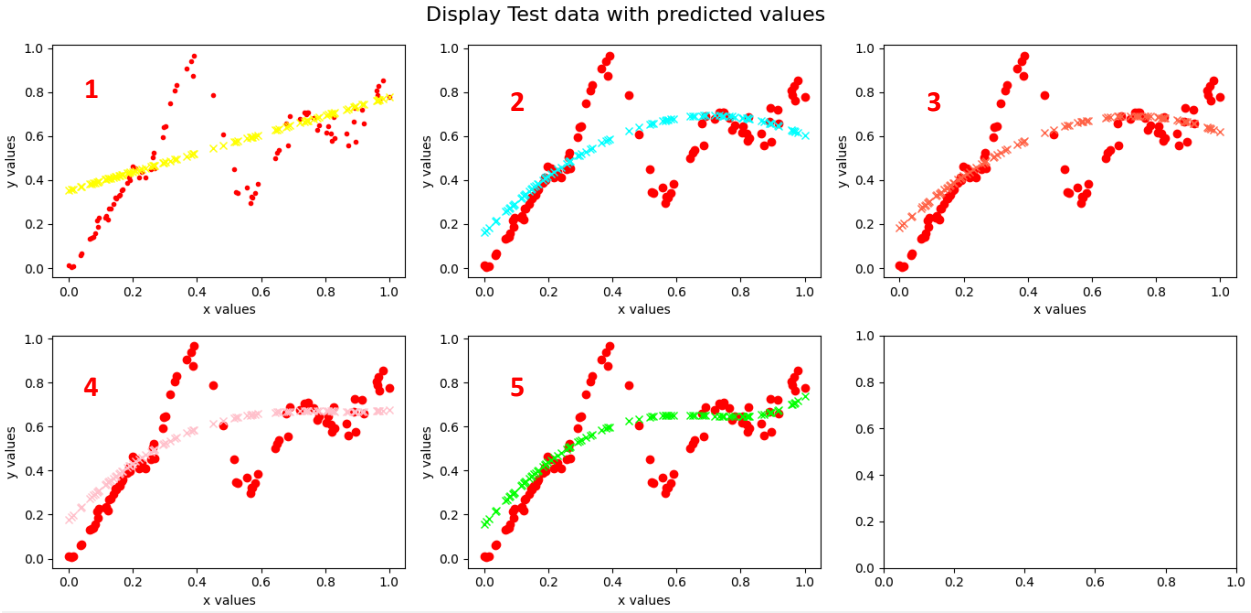
plot	degree	Repeat	Learning rate	MSE
1	1	100	0.1	۰.۰۳۸۲۴
2	2	100	0.1	۰.۰۴۶۰۲
3	3	100	0.1	۰.۰۴۸۹۵
4	4	100	0.1	۰.۰۴۹۵۵
5	5	100	0.1	۰.۰۴۹۴۹



همانطور که در سه آزمایش بالا مشاهده کردیم هرچه تعداد تکرار کمتر باشد مدل ساده‌تر باقی می‌ماند و خطا زیاد. از خطاهای بدست آمده متوجه می‌شویم که تعداد تکرار ۱۵۰۰۰ برای این مجموعه داده خوب است. همچنین در تعداد زیاد تکرار دستمان برای انتخاب نرخ یادگیری باز است اما در تعداد تکرار کم باید در انتخاب نرخ یادگیری دقت کنیم؛ زیرا مدل زیاد فرصت ندارد تا به حالت پایدار برسد. کمترین خطا نیز برای درجه‌ی ۴ است.

چهار.

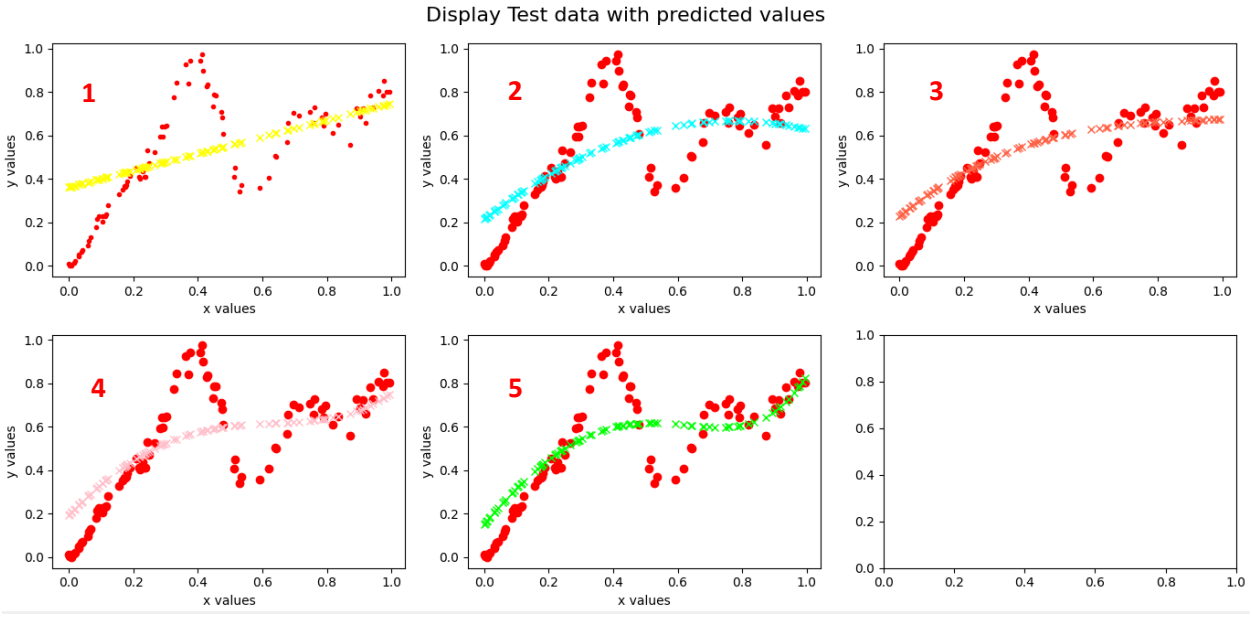
در آزمایش زیر تعداد تکرار ۱۵۰۰۰و نرخ یادگیری ۰.۱ و ضریب رگولاریزیشن ۰.۱ است. درجه‌ی هر نمودار نیز با رنگ قرمز بر روی آن مشخص شده است.



در تصویر زیر نیز خطای تست و آموزش و مقادیر تتا برای هر نمودار چاپ شده است.

```
MSE1 = 0.03196065327671836
MSE2 = 0.02552254932372446
MSE3 = 0.025842357459229682
MSE4 = 0.023701791934598038
MSE5 = 0.021617995202826937
train error
MSE1 = 0.040621702019379015
MSE2 = 0.040621702019379015
MSE3 = 0.040621702019379015
MSE4 = 0.040621702019379015
MSE5 = 0.040621702019379015
theta vector
[0.35325428 0.42569499]
[ 0.16104221  1.50187022 -1.05913126]
[ 0.18466739  1.35159752 -0.85503511 -0.06306747]
[ 0.17825547  1.49699629 -1.04996426 -0.53158862  0.58162678]
[ 0.15718525  1.65488045 -1.07850426 -0.82086938  0.06888874  0.75615551]
```

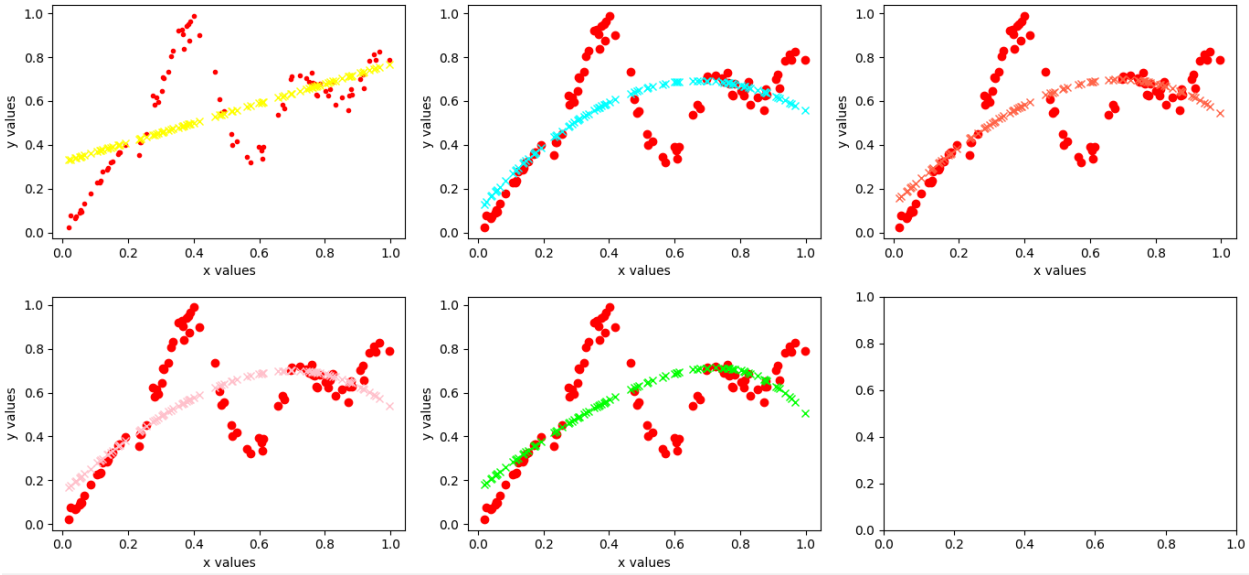
در آزمایش زیر تعداد تکرار ۱۵۰۰۰و نرخ یادگیری ۰.۱ و ضریب رگولاریزیشن ۰.۰۱ است. درجه‌ی هر نمودار نیز با رنگ قرمز بر روی آن مشخص شده است.



```
test error
MSE1 = 0.04144662585043384
MSE2 = 0.02840927277352614
MSE3 = 0.028823882119420066
MSE4 = 0.024594311738203013
MSE5 = 0.02079258661346093
train error
MSE1 = 0.03681220238688658
MSE2 = 0.03681220238688658
MSE3 = 0.03681220238688658
MSE4 = 0.03681220238688658
MSE5 = 0.03681220238688658
theta vector
[0.36245458 0.38610818]
[ 0.21542697  1.17244863 -0.75887219]
[ 0.23042317  1.21584828 -1.14062544  0.36856991]
[ 0.1933586   1.56397814 -1.44963653 -0.62286688  1.07138604]
[ 0.1514705   1.80964964 -1.41161148 -1.06737878  0.15272521  1.20493636]
```

در آزمایش زیر تعداد تکرار ۱۵۰۰۰و نرخ یادگیری ۰.۱ و ضریب رگولاریزیشن ۰.۵ است. درجه‌ی هر نمودار نیز با رنگ قرمز بر روی آن مشخص شده است.

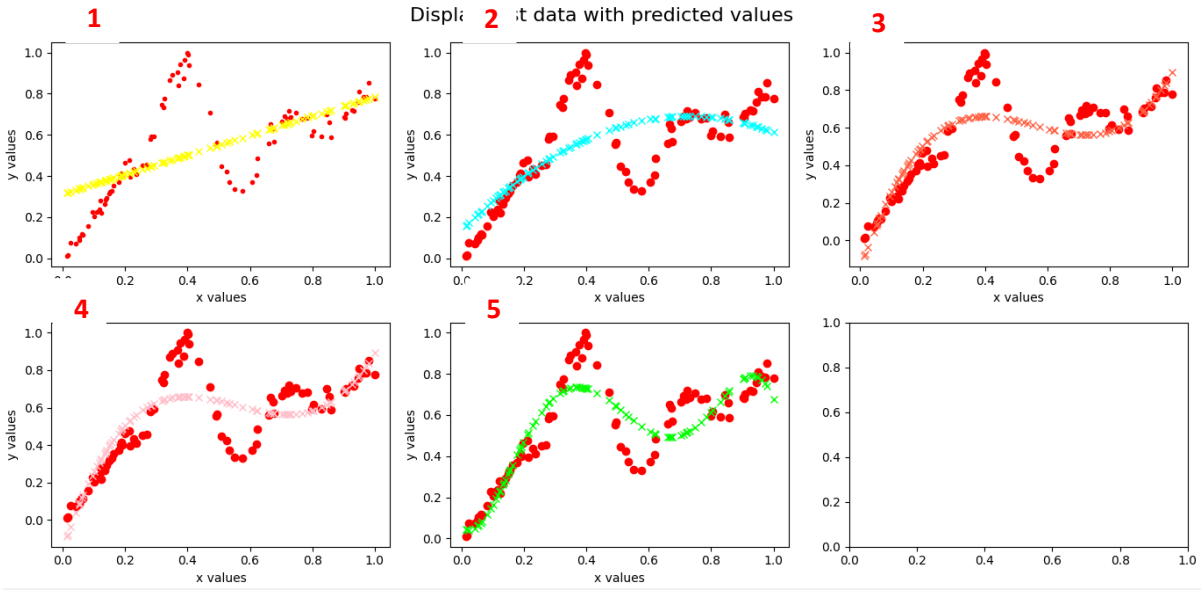
Display Test data with predicted values



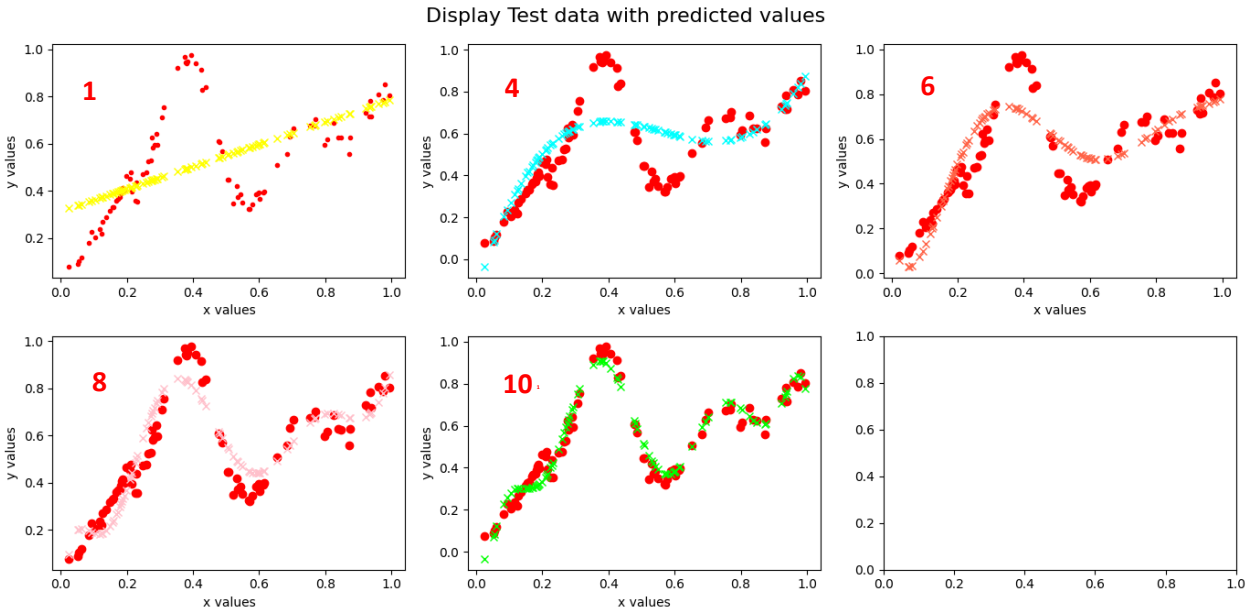

```
test error
MSE1 = 0.04055207940173032
MSE2 = 0.030852493048623506
MSE3 = 0.03270485074326227
MSE4 = 0.03355893506681515
MSE5 = 0.03580121072261523
train error
MSE1 = 0.0363632333726146
MSE2 = 0.0363632333726146
MSE3 = 0.0363632333726146
MSE4 = 0.0363632333726146
MSE5 = 0.0363632333726146
theta vector
[0.32237657 0.44641749]
[ 0.0969455   1.76772117 -1.3108206 ]
[ 0.12977863  1.4434895  -0.60232645 -0.42719279]
[ 0.14160555  1.36496797 -0.58428879 -0.1629841  -0.22249964]
[ 0.15727477  1.27170445 -0.64563511  0.04997919  0.28486224 -0.61756582]
```

همانطور که از خطاها مشخص است کمترین خطا مربوط به ضریب رگولاریزیشن ۰.۱ است و در این حالت خط پیشبینی بر روی داده‌ها مطابقت کامل ندارد. بنابراین کار این ضریب این است که مقادیر پارامتر تتا را کاهش دهد تا بیش از اندازه مدل بر داده‌ها منطبق نشود.

پنج.




```
MSE1 = 0.038074174849562575
MSE2 = 0.028734533540733468
MSE3 = 0.017627546877935246
MSE4 = 0.017666318552994074
MSE5 = 0.013070912714719076
```

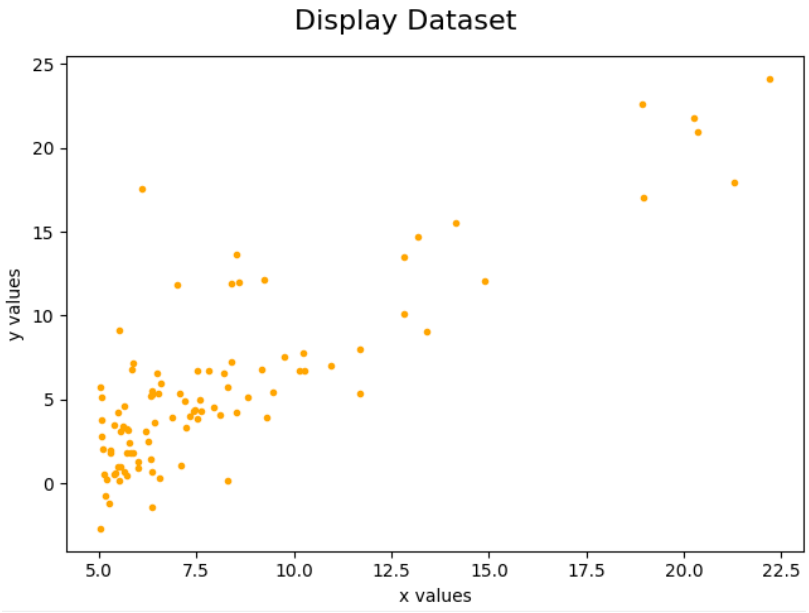



```
MSE1 = 0.03650415464351504
MSE2 = 0.021893131637559574
MSE3 = 0.014894963665638222
MSE4 = 0.008048858857868008
MSE5 = 0.002659126317930498
```

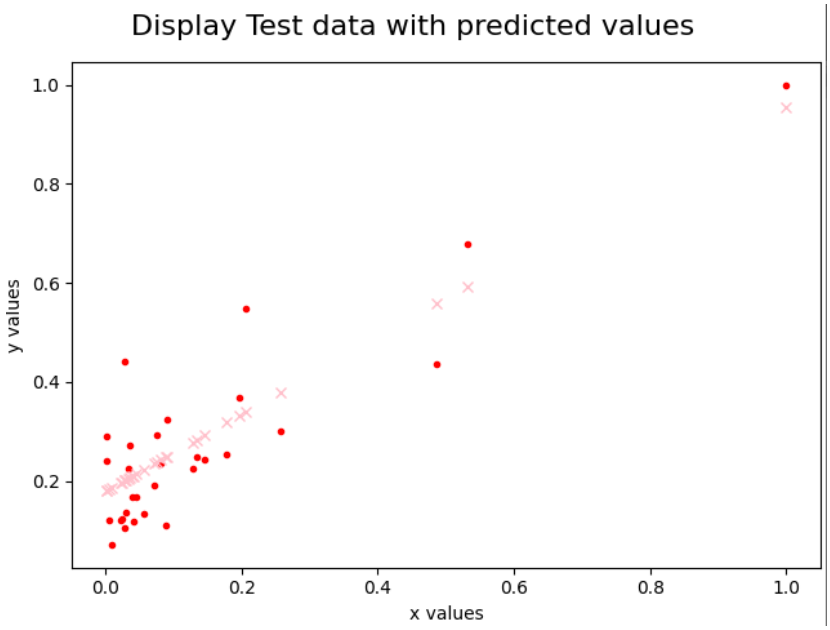
در ۸ آزمایش مجزایی که در بالا مشاهده کردیم کمترین خطا مربوط به درجه‌ی ۱۰ بود. بنابراین هرچه درجه بالاتر برود دقت مدل بیشتر می‌شود.

بخش دوم

یک.



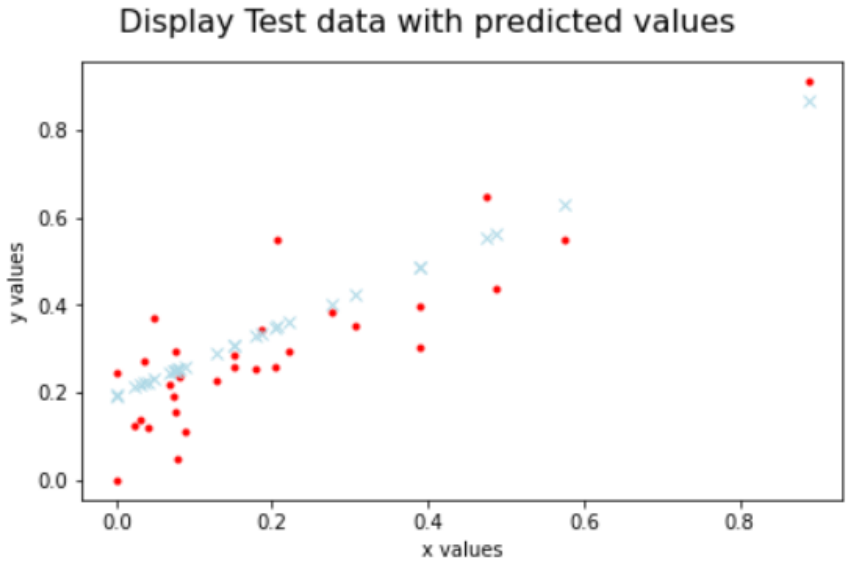
دو.




```
MSE = 0.008437519933923195
```

سه.

```
MSE_sklearn = [0.01041571]
```



این قسمت در کولب انجام شد. به همین دلیل کد این سوال را در ادامه قرار داده‌ام.

```
from google.colab import drive
drive.mount('/content/drive/')
import pandas as pd
import random import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model

def shuffeling(dataFrame):
    for i in range(0,int(dataFrame.size/2)):
        r1 = random.randint(0,dataFrame.size/2 - 1)
        r2 = random.randint(0,dataFrame.size/2 - 1)
        x = dataFrame.at[r1,'x']
        y = dataFrame.at[r1,'y']
        dataFrame.at[r1,'x'] = dataFrame.at[r2,'x']
        dataFrame.at[r1,'y'] = dataFrame.at[r2,'y']
        dataFrame.at[r2,'x'] = x
        dataFrame.at[r2,'y'] = y
    return dataFrame

def normalizer(dataFrame):
    maxX = dataFrame['x'].max()
    minX = dataFrame['x'].min()
    maxY = dataFrame['y'].max()
    minY = dataFrame['y'].min()
    dataFrame['x'] = pd.to_numeric(dataFrame['x'], downcast="float")
    dataFrame['y'] = pd.to_numeric(dataFrame['y'], downcast="float")
```

```
    for i in range(0,int(dataFrame.size / 2)):
        dataFrame.at[i,'x'] = (dataFrame.at[i,'x'] - minX )/(maxX - minX)
        dataFrame.at[i,'y'] = (dataFrame.at[i,'y'] - minY )/(maxY - minY)
    return dataFrame

def calculate_error(predicted,real , index):
     error = 0
     for i in range(0,predicted.size):
         error = pow((predicted.__getitem__(i) - real.at[index + i , 'y'])
 , 2) + error
     error = error / predicted.size
     return error



df = pd.read_csv('/content/drive/My Drive/Dataset2.csv', delimiter=',', en
coding="utf-8-sig")
# shuffling dataset
df = shuffeling(df)
# normalizing dataset
df = normalizer(df)
m = int(df.size / 2)
s = int(0.7 * m)
train = df.iloc[:s, :]
test = df.iloc[s:, :]
reg = linear_model.LinearRegression().fit(train['x'].values.reshape(-
1, 1),train['y'].values.reshape(-1, 1))
predicted_values_sklearn = reg.predict(test['x'].values.reshape(-1, 1))
# print(predicted_values_sklearn)
# print(predicted_values_sklearn.__class__)
MSE_prim = calculate_error(predicted_values_sklearn, test,s)
print("MSE_sklearn = " + str(MSE_prim))
# display outputs
fig, axs = plt.subplots(1,1, constrained_layout=True)
axs.plot(test['x'], test['y'], "." , color = "red")
axs.plot(test['x'], predicted_values_sklearn,"x",color="lightblue")
axs.set_xlabel('x values')
axs.set_ylabel('y values')
fig.suptitle('Display Test data with predicted values', fontsize=16)
plt.show()
```