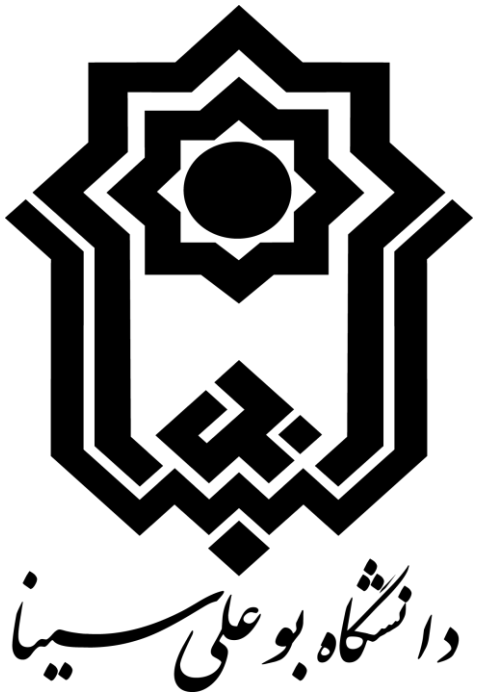
به نام خدا



نام و نام خانوادگی:سید فرهاد حسینی

شماره دانشجویی : ۹۶۱۲۳۵۸۰۱۶

نام درس : مبانی داده کاوی

استاد مربوطه : دکتر منصوری زاده

- صورت تمرین :

✓ پیاده سازی رگرسیون خطی و لاجستیک برروی دیتاست آیریس طبق دستور داده شده

- بررسی کد :

```
[1]  #------------------load dataset-------------
     from sklearn.datasets import load_iris
     import numpy as np
     import pandas as pd
     iris = load_iris()

[2]  #--------------onehot encoding---------
     target = pd.get_dummies(iris.target)

[3]  dataset = np.hstack((np.ones((150,1)),iris.data))
     dataset.shape  #(150, 5)

     (150, 5)

[4]  #-------------step 1 ----------------
     r=np.random.rand(3,5)*100 - 50
```

در سلول اول دیتاست آیریس را درون برنامه لود کردیم .

در سلول دوم عمل one hot encoding را برروی برچسب کلاس ها انجام دادیم

در سلول سوم یک ستون با مقادیر تماما ۱ به ویژگی ها متصل کردیم

در سلول بعدی گام ۱ دستورالعمل را انجام دادیم یعنی به تعداد ویژگی ها عدد رندوم تولید کردیم (سه سری)

```
[5]  #-------------step2 & 3 -----------------
     z= dataset[:,0:5] @ r.T
     z.shape  #(150, 3)

     (150, 3)

[6]  # ---------------hstack -------------
     dataset=np.hstack((dataset,z,target))
     dataset.shape

     (150, 11)

[7]  #----------shuffling-----------------
     from sklearn.utils import shuffle
     dataset=shuffle(dataset)
     dataset.shape  #(150, 8)

     (150, 11)

[8]  #-------------test & train -----------
     train=dataset[0:120,:]
     test=dataset[120:,:]
     test.shape                #(30, 11)
     train.shape              # (120, 11)

     (120, 11)
```

در سلول اول تصویر بالا ترکیب خطی ویژگی ها و اعداد رندوم مرحله قبل را برای سه سری از اعداد رندوم انجام دادیم

در سلول دوم ویژگی های قبلی را به ویژگی های جدید متصل کرده و نهایتا به برچسب کلاس ها که قبلا آنها را کد کردیم متصل کردیم . (اتصال افقی)

حال سطر های ماتریس را شافل کرده تا هیچ پیش زمینه و قصد خاصی در انتخاب مجوعه تست و ترین وجود نداشته باشد .

در سلول آخر نیز ۲۰ درصد سطر ها را به مجمعه تست اختصاص داده و با مابقی داده ها مدل را آموزش میدهیم .

```
[9] #------------linear regression-------------
    from scipy import linalg
    theta= linalg.pinv(train[:,0:8].T @ train[:,0:8]) @ train[:,0:8].T @ train[:,8:11]
    theta.shape   #(8, 3)
```

```
    (8, 3)
```

```
[10] #---------rate on train data-------------
     y_pred = train[:,0:8] @ theta
     y_pred = (y_pred > 0.5) * 1 #treshold

     r1=(120 - (np.sum((train[:,8] - y_pred[:,0] )**2))) / 120
     r2=(120 - (np.sum((train[:,9] - y_pred[:,1] )**2))) / 120
     r3=(120 - (np.sum((train[:,10] - y_pred[:,2] )**2))) / 120

     np.hstack(( np.reshape(train[:,9],(-1,1)) , np.reshape(y_pred[:,1],(-1,1))))

     r1 , r2 ,r3
     #compare
```

```
     (1.0, 0.725, 0.9333333333333333)
```

```
[11] #---------rate on test data-------------
     y_pred = test[:,0:8] @ theta
     y_pred = (y_pred > 0.5) * 1 #treshold

     r1=(30 - (np.sum((test[:,8] - y_pred[:,0] )**2))) / 30
     r2=(30 - (np.sum((test[:,9] - y_pred[:,1] )**2))) / 30
     r3=(30 - (np.sum((test[:,10] - y_pred[:,2] )**2))) / 30

     np.hstack(( np.reshape(test[:,9],(-1,1)) , np.reshape(y_pred[:,1],(-1,1))))

     r1 , r2 ,r3
```

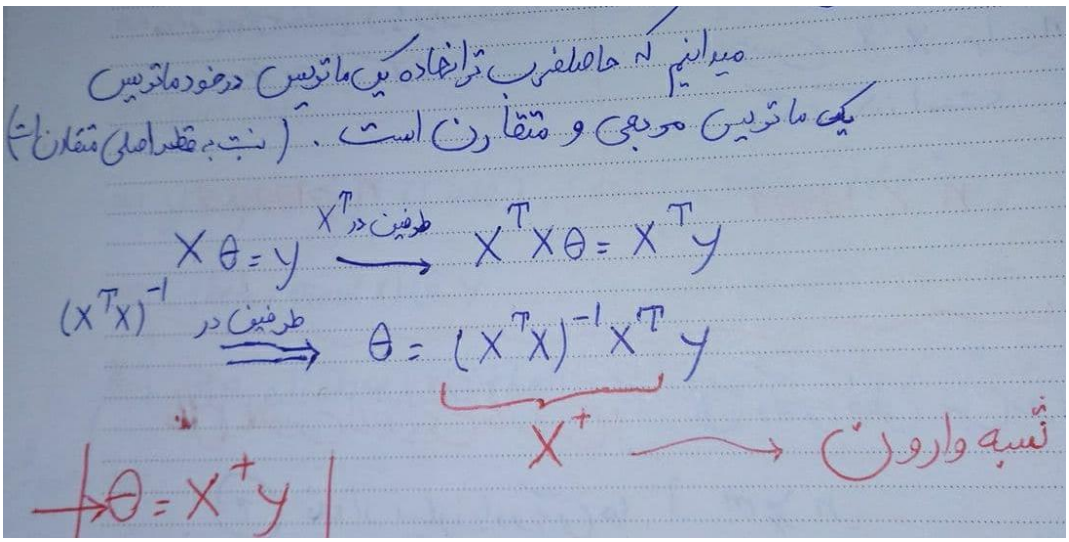
```
     (1.0, 0.7666666666666667, 0.9666666666666667)
```

تصویر بالا مربوط به پیاده سازی الگوریتم رگرسیون خطی و بررسی دقت آن برروی داده های تست و ترین میباشد .

توضیح الگوریتم رگرسیون خطی :

این الگوریتم بوسیله یک ترفند در جبر خطی پیاده سازی شده (در کلاس آنلاین اشاره ای به آن شد) .

اگر x ماتریس مقادیر ویژگی ها و y ماتریس برچسب کلاس ها و Θ ضرایب مجهول باشند داریم:



در سلول دوم دقت الگوریتم را برروی داده های ترین بررسی کردیم که دقت آنرا در تصویر برروی سه کلاس آیریس مشاهده میکنید .

در سلول سوم دقت الگوریتم را برروی داده های تست بررسی کردیم که دقت آنرا در تصویر برروی سه کلاس آیریس مشاهده میکنید . (عجیبه که بهتر جواب داده)

```
[12] #--------logestic regression ----------------
     theta1=np.random.rand(8,3)*100 - 50
     def logestic(u):
       return 1 /(1+(np.e ** (-1*u)))
```

```
[56] #----------gredient decsent----------------
     t=1 #tekrar
     alpha=0.0017
     while (t<400) :
       er =  train[:,8:11] - logestic(train[:,0:8] @ theta1)
       theta1 = theta1 + alpha * train[:,0:8].T @ er
       t+=1
```

حال به پیاده سازی الگوریتم رگرسیون لاجستیک به کمک گرادیان کاهشی میپردازیم.

در سلول اول ضرایب اولیه بصورت رندوم پر شده اند و یک تابع برای محاسبه تابع سیگموید نوشته ایم .

در سلول دوم الگوریتم گرادیان کاهشی که در کلاس آنلاین شبه کد آن نوشته شد را پیاده سازی کردیم .

(با چند بار تست کردن مقدار مناسب آلفا و تعداد تکرار های الگوریتم بدست آمد)

```
[57] #---------rate on train data-------------
     vv = logestic(train[:,0:8] @ theta1)


     y_pred1 = (vv > 0.5)  #treshold

     r1=(120 - (np.sum((train[:,8] - y_pred1[:,0] )**2))) / 120
     r2=(120 - (np.sum((train[:,9] - y_pred1[:,1] )**2))) / 120
     r3=(120 - (np.sum((train[:,10] - y_pred1[:,2] )**2))) / 120

     np.hstack(( np.reshape(train[:,9],(-1,1)) , np.reshape(y_pred1[:,1],(-1,1))))

     r1 , r2 ,r3
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: RuntimeWarning: overflow encountered in power
  after removing the cwd from sys.path.
(1.0, 0.775, 0.975)
```

```
#---------rate on test data-------------
vv = logestic(test[:,0:8] @ theta1)

y_pred1 = (vv > 0.5)  #treshold

r1=(30 - (np.sum((test[:,8] - y_pred1[:,0] )**2))) / 30
r2=(30 - (np.sum((test[:,9] - y_pred1[:,1] )**2))) / 30
r3=(30 - (np.sum((test[:,10] - y_pred1[:,2] )**2))) / 30

np.hstack(( np.reshape(test[:,10],(-1,1)) , np.reshape(y_pred1[:,2],(-1,1))))

r1 , r2 ,r3
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: RuntimeWarning: overflow encountered in power
  after removing the cwd from sys.path.
(1.0, 0.6666666666666666, 0.9666666666666667)
```

در سلول اول دقت الگوریتم را برروی داده های ترین بررسی کردیم که دقت آنرا در تصویر برروی سه کلاس آیریس مشاهده میکنید .

در سلول دوم دقت الگوریتم را برروی داده های تست بررسی کردیم که دقت آنرا در تصویر برروی سه کلاس آیریس مشاهده میکنید .

لینک notebook :

https://colab.research.google.com/drive/1ANXi4xrDTQ6BHRp4LLPhefvCdw5jLqoH?usp=sharing