گزارش کد

سوال چهارم

(a

P4a.py

فراخوانی کتابخانه‌های لازم

```
import pandas as pd
import matplotlib.pyplot as plt
```

خواندن دیتاست. هدر نداریم و جداکننده داده‌ها کاما است

```
Data_Raw = pd.read_csv('iris.data',sep=',',header=-1)
```

ابتدا یک شکل باز می‌کنیم که تنها یک سطر و ستون دارد و نیاز به زیر پلات نیست تا سطر و ستون‌های بیشتری را داشته باشیم

```
plt.subplots(nrows=1, ncols=1)
```

با دستور hist2d اقدام به نمایش هیستوگرام دوبعدی داده‌ها می‌کنیم . ویژگی‌های انتخابی ویژگی اول و دوم است که با اندیس صفر و یک نمایش دادیم.

```
plt.hist2d(list(Data_Raw.values[:,0]),list(Data_Raw.values[:,1]), bins=100)
```

برچسب محور افقی را متغیر ۱ و برچسب محور عمودی را متغیر ۲ گذاشته‌ایم

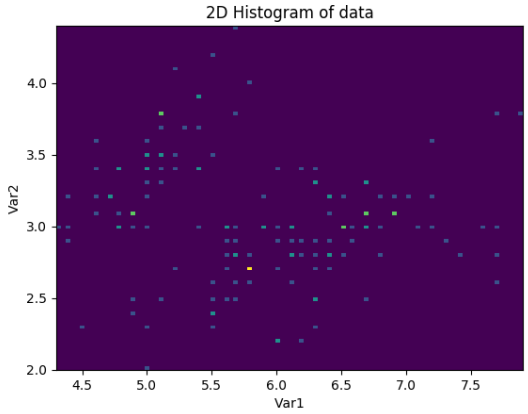
```
plt.xlabel('Var1')
plt.ylabel('Var2')
```

عنوان نمایش

```
plt.title('2D Histogram of data')
```

نمایش پلات

```
plt.show(block=True)
```



(b

P4b.py

فراخوانی کتابخانه‌های لازم

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
```

خواندن دیتاست. هدر نداریم و جداکننده داده‌ها کاما است

```
Data_Raw = pd.read_csv('iris.data',sep=',',header=-1)
```

ساخت یک فیگر برای نمایش

```
fig = plt.figure()
```

خصیصه شکل را به صورت سه بعدی تغییر می‌دهیم تا برای ترسیم سه بعدی آماده باشد

```
ax = fig.add_subplot(111, projection='3d')
```

جدا کردن دو ویژگی اول برای نمایش

```
data 2d = Data Raw.values[:,0:1]
```

تبدیل کردن ویژگی‌ها به ماتریس numpy

```
data_array = np.array(data_2d)
```

ساخت مش برای ترسیم دو بعد از سه بعد

```
x_data, y_data = np.meshgrid( np.arange(data_array.shape[1]), np.arange(data_array.shape[0]) )
```

تبدیل کرد داده‌های دو ویژگی به حالت flatt

```
x_data = x_data.flatten()
y_data = y_data.flatten()
```

تبدیل کردن مقادیر دو ویژگی به یک سطح برای ترسیم بعد سوم

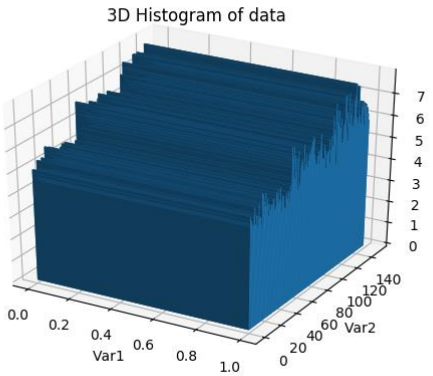
```
z_data = data_array.flatten()
```

ترسیم سه بعدی داده‌های تولید شده

```
ax.bar3d( x_data,y_data, np.zeros(len(z_data)),1, 1, z_data )
```

برچسب محورها و عنوان نمودار

```
plt.xlabel('Var1')
plt.ylabel('Var2')
plt.title('3D Histogram of data')
plt.show()
```



(c

P4c.py

فراخوانی کتابخانه‌های لازم

```
import pandas as pd
import matplotlib.pyplot as plt
```

خواندن دیتاست. هدر نداریم و جداکننده داده‌ها کاما است

```
Data_Raw = pd.read_csv('iris.data',sep=',',header=-1)
```

ابتدا برچسب‌های یکتا را استخراج می‌کنیم که مشخص کنیم چه تعداد کلاس داریم

. برچسب‌ها در ستون پنجم قرار دارد

```
Unique_Label = pd.unique(Data_Raw.values[:,4])
```

سپس برچسب‌های رشته‌ای را به برچسب‌های عددی متناظر می‌کنیم

```
NUmeric_Label = Data_Raw[4].apply(list(Unique_Label).index)
```

به ازای هر کلاس، داده‌های آن کلاس را با scatter در دو بعد ترسیم می‌کنیم و در هر ترسیم یکی از رنگ‌ها را انتخاب کرده و داده‌ها را با آن عکس نایش می‌دهیم

```
count = 0
colors = ['red','green','blue','purple']
for i in Unique_Label:
```

انتخاب ویژگی اول و دومی که در کلاس i ام از برچسب‌های یکتا قرار دارند

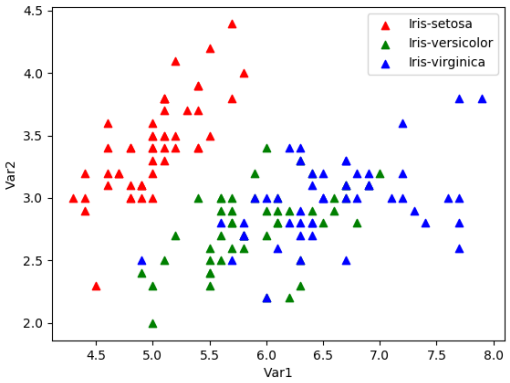
```
    Temp = Data Raw[(Data Raw[4]==i)][[0,1]]
```

ترسیم نمونه‌های هر کلاس در دو بعد

```
    plt.scatter(Temp[:][0],Temp[:][1], marker='^', c=colors[count], label = i)
    count = count +1
```

اطلاعات نمایش

```
plt.legend()
plt.xlabel('Var1')
plt.ylabel('Var2')
plt.show()
```



(d

P4d.py

فراخوانی کتابخانه‌های لازم

```
import pandas as pd
import matplotlib.pyplot as plt
```

خواندن دیتاست. هدر نداریم و جداکننده داده‌ها کاما است

```
Data_Raw = pd.read_csv('iris.data',sep=',',header=-1)
```

پیدا کردن میانگین و واریانس ویژگی‌ها

```
Mean Arr = Data Raw.mean().values
Var Arr = Data Raw.var().values
```

اسم هر ویژگی را به طور دلخواه مشخص می‌کنیم

```
index = ['Var1', 'Var2', 'Var3' , 'Var4']
```

با استفاده از دیتا فریم ترسیم نمودار میله‌ای را انجام می‌دهیم

```
Data_Frame_Obj =  pd.DataFrame({'Mean': Mean_Arr,'Variance': Var_Arr}, index=index)
```

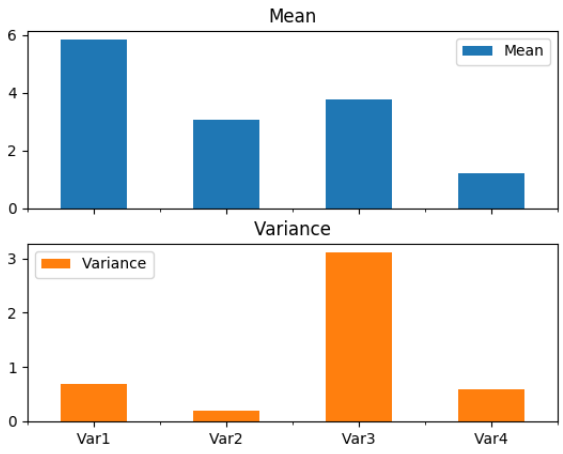
ترسیم میانگین و واریانس

```
Plot Obj = Data Frame Obj.plot.bar(rot=0)
```

ابتدا نمودارها میانگین و واریانس در کنار یکدیگر قرار می‌گیرند. در این کد قصد داریم تا این دو را جدا کنیم و در دو زیر پلات ترسیم کنیم.

```
axes = Data_Frame_Obj.plot.bar(rot=0, subplots=True)
axes[1].legend(loc=2)

plt.show()
```



(e

P4e.py

فراخوانی کتابخانه‌های لازم

```
import pandas as pd
import numpy as np
```

خواندن دیتاست. هدر نداریم و جداکننده داده‌ها کاما است

```
Data_Raw = pd.read_csv('iris.data',sep=',',header=-1)

# >>>>>>>>>>>>> Cov Matrix
```

پیدا کردن برچسب‌های یکتا

```
Unique Label = pd.unique(Data Raw.values[:,4])
```

برچسب اول و دوم را انتخاب کرده و داده‌های این دو کلاس را جدا می‌کنیم

```
Class1 = Data Raw[(Data Raw[4]==Unique Label[0])].values[:,0:4]
Class2 = Data_Raw[(Data_Raw[4]==Unique_Label[1])].values[:,0:4]
```

ابتدا میانگین هر کلاس را از داده‌های آن کلاس کم می‌کنیم

```
Class1 Normalized = Class1 - np.mean(Class1)
Class2 Normalized = Class2 - np.mean(Class2)
```

ساخت یک ماتریس ۲در ۲

```
Cov Mat = [[0,0],[0,0]]
```

محاسبه هر المان از ماتریس کوواریانس مطابق با فرمول پایین.

دستور size تعداد نمونه‌ها را می‌دهد و دستور multiply ضرب المان در المان داده‌های دو کلاس را انجام می‌دهد

```
Cov Mat[0][0] = np.sum((np.multiply(Class1 Normalized,Class1 Normalized))/(np.size(Class1 Normalized)-1))
Cov_Mat[0][1] = np.sum((np.multiply(Class1_Normalized,Class2_Normalized))/(np.size(Class1_Normalized)-1))
Cov_Mat[1][0] = np.sum((np.multiply(Class2_Normalized,Class1_Normalized))/(np.size(Class1_Normalized)-1))
Cov_Mat[1][1] = np.sum((np.multiply(Class2_Normalized,Class2_Normalized))/(np.size(Class1_Normalized)-1))
```

پرینت ماتریس کوواریانس

```
print(Cov_Mat)
```

[[3.4135788944723644, 2.3860713567839187]

, [2.3860713567839187, 3.106001005025126]]

برای محاسبه ماتریس کوواریانس از رابطه زیر استفاده کرده‌ایم. مسلم است که کوواریانس $\text{cov}(A,B)$ و $\text{cov}(B,A)$ باید یکی شود.

$$\text{cov}(A,B) = \frac{1}{N-1}\sum_{i=1}^{N}(A_i-\mu_A)^*(B_i-\mu_B)$$

$$C = \begin{pmatrix} \text{cov}(A,A) & \text{cov}(A,B) \\ \text{cov}(B,A) & \text{cov}(B,B) \end{pmatrix}.$$

(f

P4f.py

فراخوانی کتابخانه‌های لازم

```
import pandas as pd
import numpy as np
```

خواندن دیتاست. هدر نداریم و جداکننده داده‌ها کاما است

```
Data_Raw = pd.read_csv('iris.data',sep=',',header=-1)

# >>>>>>>>>>>>> Cov Matrix
```

توضیحات مشابه بخش قبل

```
Unique_Label = pd.unique(Data_Raw.values[:,4])
Class1 = Data_Raw[(Data_Raw[4]==Unique_Label[0])].values[:,0:4]
Class2 = Data_Raw[(Data_Raw[4]==Unique_Label[1])].values[:,0:4]

Class1_Normalized = Class1 - np.mean(Class1)
Class2_Normalized = Class2 - np.mean(Class2)
Cov_Mat = [[0,0],[0,0]]
Cov_Mat[0][0] = np.sum((np.multiply(Class1_Normalized,Class1_Normalized))/(np.size(Class1_Normalized)-1))
Cov_Mat[0][1] = np.sum((np.multiply(Class1_Normalized,Class2_Normalized))/(np.size(Class1_Normalized)-1))
Cov_Mat[1][0] = np.sum((np.multiply(Class2_Normalized,Class1_Normalized))/(np.size(Class1_Normalized)-1))
Cov_Mat[1][1] = np.sum((np.multiply(Class2_Normalized,Class2_Normalized))/(np.size(Class1_Normalized)-1))
```

در این قسمت ماتریس همبستگی را طبق رابطه با تقسیم بر انحرافات معیار به دست می‌آوریم.

```
Corr_Mat = Cov_Mat / (np.std(Class1)*np.std(Class2))
[Corr_Mat[0][0],Corr_Mat[1][1]] = [round(Corr_Mat[0][0]),round(Corr_Mat[1][1])]

print(Corr_Mat)
```

برای محاسبه مقادیر همبستگی کافی است تا ماتریس کوواریانس را تقسیم بر انحرافات معیار دو کلاس مورد نظر کنیم

$$\rho(A,B) = \frac{\operatorname{cov}(A,B)}{\sigma_A \sigma_B}.$$

خروجی به دست آمده به صورت زیر خواهد بود.

[[1. 0.73646918]

[0.73646918 1.]]

(g

P4g.py

فراخوانی کتابخانه‌های لازم

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

خواندن دیتاست. هدر نداریم و جداکننده داده‌ها کاما است

```
Data_Raw = pd.read_csv('iris.data',sep=',',header=-1)
```

محاسبه میانگین و انحراف معیار داده‌های کلاس ویرجینیکا

```
Means = Data_Raw[Data_Raw[4]=='Iris-virginica'].mean()
Std  = Data_Raw[Data_Raw[4]=='Iris-virginica'].std()


import matplotlib.mlab as mlab
```

به ازای هر ویژگی که میانگین و انحراف معیار را پیدا کردیم، نمودار توزیع گاوسی آن را رسم می‌کنیم

```
for i in range(0,4,1):
    x = np.linspace(Means[i] - 3*Std[i], Means[i] + 3*Std[i], 100)
    plt.plot(x,mlab.normpdf(x, Means[i], Std[i]))
```

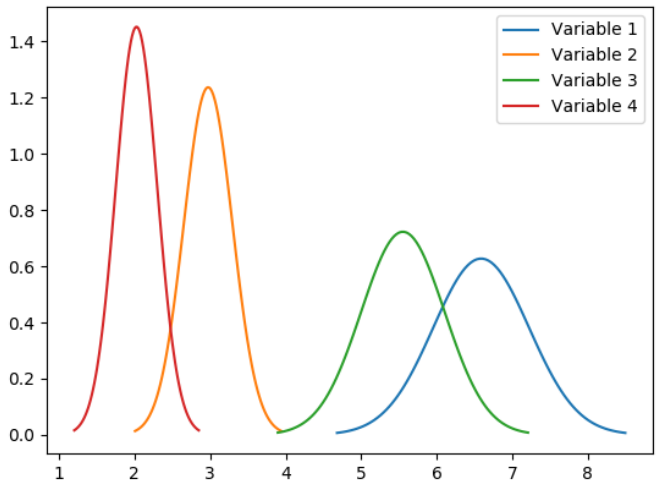
عنوان ترسیم‌ها را مشخص می‌کنیم

```
plt.legend(['Variable 1','Variable 2','Variable 3','Variable 4'])
```

پیدا کردن ضرایب همبستگی ویژگی‌ها

```
Corr_Of_Features = np.corrcoef((Data_Raw[[0,1,2,3]][Data_Raw[4]=='Iris-virginica']),rowvar=False)
print(Corr Of Features)

plt.show()
```



[[1. 0.45722782 0.86422473 0.28110771]

[0.45722782 1. 0.40104458 0.53772803]

[0.86422473 0.40104458 1. 0.32210822]

[0.28110771 0.53772803 0.32210822 1.]]

(h

P4h.py

فراخوانی کتابخانه‌های لازم

```
import pandas as pd
from sklearn.feature_selection import mutual_info_classif
```

خواندن دیتاست. هدر نداریم و جداکننده داده‌ها کاما است

```
Data_Raw = pd.read_csv('iris.data',sep=',',header=-1)
```

دادن دیتاست به ابزار MI برای پیدا کردن بهترین ویژگی و مشخص کردن ارتباط بین ویژگی‌ها و برچسب کلاس‌ها

```
MI_Predicted = mutual_info_classif(Data_Raw[[0,1,2,3]].values,Data_Raw[4].values)

print(MI_Predicted)
```

خروجی این تابع به صورت زیر است

[0.48467537 0.19982519 0.99096098 0.99312168]

ابتدا ماتریس ویژگی‌ها را ترانهاده می‌کنیم برای استفاده آینده

```
    Input_Transpose = Input.transpose()
```

آرایه‌ای برای ذخیره خطای حین آموزش

```
    SSE_Array = np.zeros(Iter)
```

تا تعداد تکرار مشخص آموزش را انجام می‌دهیم

```
    for in_Iter in range(0, Iter):
```

ابتدا با توجه به وزن‌های فعلی مقدار خروجی را محاسبه می‌کنیم

```
        Predicted_Target = np.dot(Input, Weights)
```

خطای پیش‌بینی را محاسبه می‌کنیم

```
        Error = Predicted_Target - Target
```

مقدار خطای خواسته شده در سوال محاسبه می‌شود

```
        SSE = np.sum(Error ** 2)
```

گرادیان را به دست می‌آوریم با ضرب خطا در ورودی

```
        gradient = np.dot(Input_Transpose, Error) / Dimen
```

سپس وزن‌ها را به‌روزرسانی می‌کنیم

```
        Weights = Weights - Learning_Rate * gradient
```

مقدار خطا را ذخیره می‌کنیم

```
        SSE_Array[in_Iter] = SSE
```

نمایش اطلاعات آموزشی

```
        print("Iter %d with SSE: %f" % (in_Iter, SSE))
```

وزن‌ها و آرایه خطا را برمی‌گردانیم

```
    return Weights, SSE_Array
```

خواندن داده‌ها

```
# Main Code
Data_Raw = np.load('data.npz')
x1 = Data_Raw.f.x1
x1 test = Data Raw.f.x1 test
x2 = Data Raw.f.x2
x2_test = Data_Raw.f.x2_test
y = Data_Raw.f.y
y_test = Data_Raw.f.y_test
```

آماده کردن مقادیر داده‌ها و اضافه کردن ترم بایاس. داده‌های آموزش و تست جداگانه آماده می‌شوند.

این داده‌ها با توجه به رابطه صورت مسئله آماده می‌شوند.

```
Bias Train = np.ones([np.shape(x1)[0],Gradien Order])
Bias Test = np.ones([np.shape(x1 test)[0],Gradien Order])
x_train = np.column_stack((np.multiply(x1,x2 **2),x2 **2,x1,Bias_Train))
x_test = np.column_stack((np.multiply(x1_test,x2_test **2),x2_test **2,x1_test,Bias_Test))
```

به آخر ویژگی‌ها مقدار بایاس ۱ را اضافه می‌کنیم

```
x = np.column stack((x1,x2,Bias))
```

سایز نمونه‌ها

```
Sample Size, Dimen = np.shape(x train)
```

تعداد تکرار الگوریتم

```
Iter= 1000
```

نرخ یادگیری

```
Learning Rate = 0.0000001
```

مقداردهی اولیه ماتریس وزن‌ها

```
Weights = np.ones(Dimen)
```

آموزش گرادیان

```
Weights, SSE Array = Gradient Descent(x train, y, Weights, Learning Rate, Sample Size, Iter)
```

محاسبه خطای پیش‌بینی داده‌های آموزش و آزمایش با وزن‌های به دست آمده

```
y_p_test = Weights[0]*x_test[:,0] + Weights[1]*x_test[:,1] + Weights[2]*x_test[:,2]+
Weights[3]*x_test[:,3]
y_p_train = Weights[0]*x train[:,0] + Weights[1]*x train[:,1] + Weights[2]*x train[:,2]+
Weights[3]*x_train[:,3]


Error = y_p_test - y_test
SSE Test = np.sum(Error ** 2)
Error = y_p_train - y
SSE_Train = np.sum(Error ** 2)

print("SSE > Test = %f   &  Train =  %f " % (SSE_Test,SSE_Train))
```

ترسیم داده‌های تست و داده‌های پیش‌بینی شده

```
# Plot Target and Perdicted Target
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.scatter(x1_test, x2_test, y_test , c='red')
ax.scatter(x1_test, x2_test, y_p_test , c='blue')
plt.xlabel('x1')
plt.ylabel('x2')
plt.ylabel('y')
ax.legend(['Target','Prediction'])
```

ترسیم نمودار خطای حین آموزش

```
# Plot Train SSE line
fig = plt.figure()
plt.plot(range(0,Iter),SSE_Array,c='red')
plt.xlabel('Iteration')
plt.ylabel('SSE')
plt.legend(['Train SSE Line'])
plt.show()
```