KECE407 HW1

2019150445 신백록

---

0. Data Preprocessing

In [1]:
```python
import scipy.io
import pandas as pd
```

In [2]:
```python
class1=scipy.io.loadmat('class1.mat')
```

In [3]:
```python
print(type(class1))
```

```
<class 'dict'>
```

In [4]:
```python
class1=class1['R']
```

In [5]:
```python
print("size :",len(class1), "X", len(class1[0]))
```

```
size : 100 X 2
```

In [6]:
```python
class1=pd.DataFrame(class1,columns=('length','weight'))
```

In [7]:
```python
class2=scipy.io.loadmat('class2.mat')
```

In [8]:
```python
class2=class2['R']
```

In [9]:
```python
print("size :",len(class2), "X", len(class2[0]))
```

```
size : 100 X 2
```

In [10]:
```python
class2=pd.DataFrame(class2,columns=('length','weight'))
```

In [11]:
```python
class2
```

Out[11]:

	length	weight
0	83.258704	26.550746
1	81.653279	34.364317
2	74.045521	30.962814
3	78.973286	36.676235
4	79.076269	34.651967
...	...	...
95	80.505545	32.957943
96	74.001280	33.723384
97	82.602495	20.903814
98	84.281584	24.366071
99	85.424777	28.537182

100 rows × 2 columns

In [12]:
```python
obs=scipy.io.loadmat('observation.mat')
```

In [13]:
```python
obs=obs['obs']
```

In [14]:
```python
print("size:",len(obs),"X",len(obs[0]))
obs=pd.DataFrame(obs,columns=('length','weight'))
```

```
size: 6 X 2
```

In [15]:
```python
obs
```

Out[15]:

	length	weight
0	65	22
1	75	25
2	80	31
3	90	36
4	60	20
5	70	30

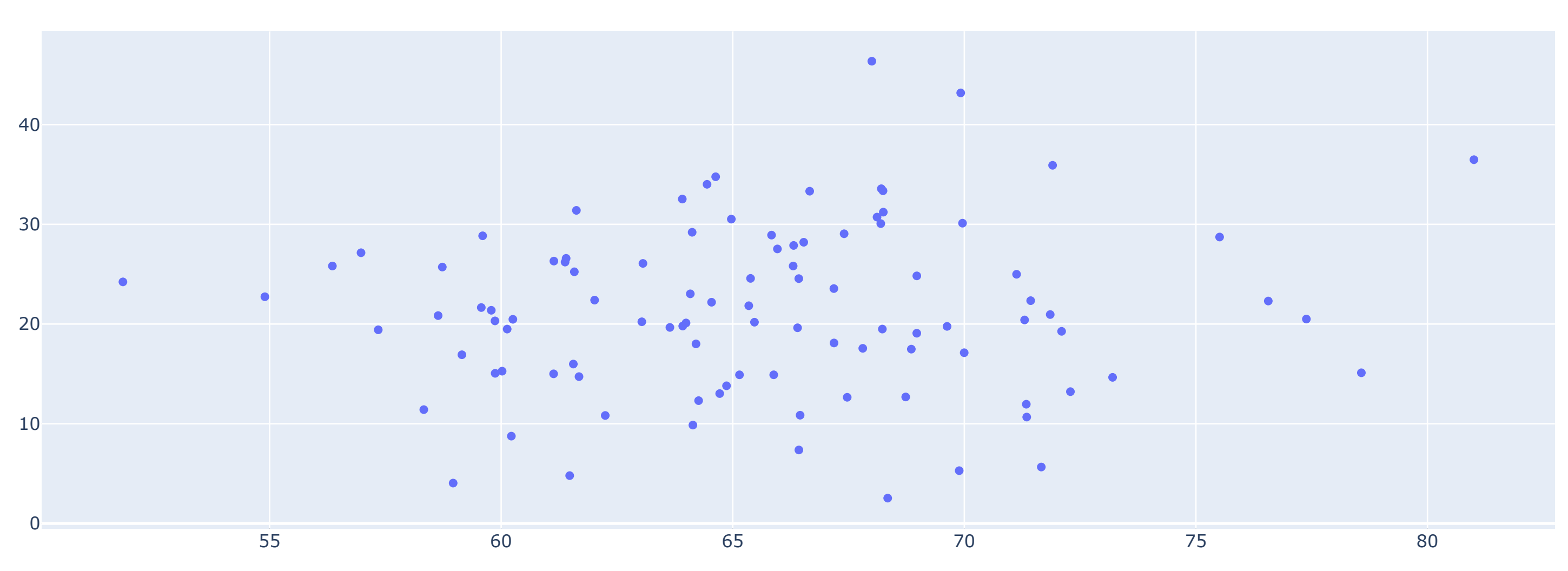
---

1. Data Visualization

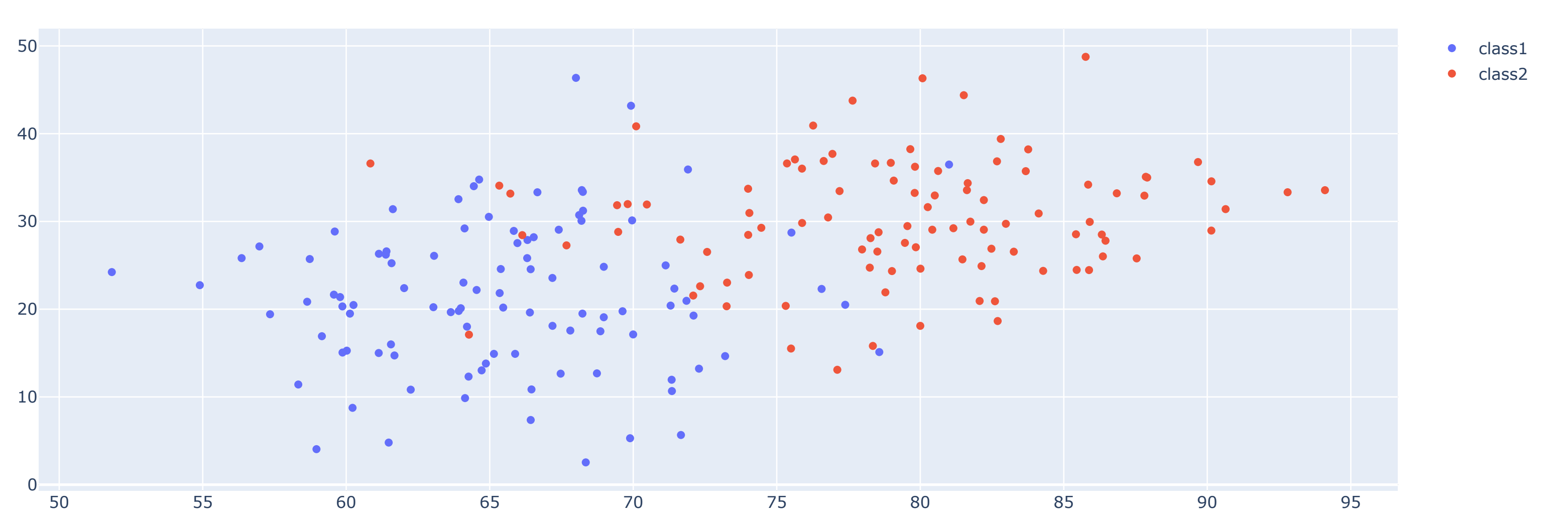
In [78]:
```python
import plotly.graph_objects as go
from plotly.offline import init_notebook_mode, plot_mpl
pyo.init_notebook_mode()

fig=go.Figure()
```

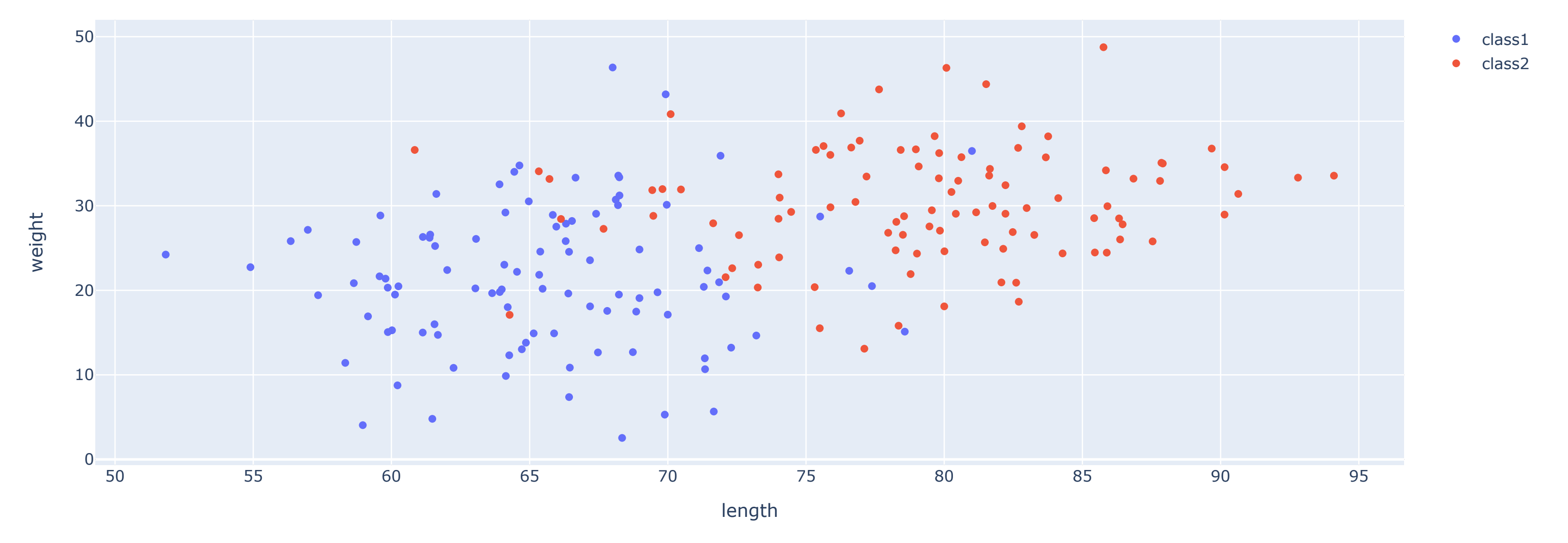
In [79]:
```python
fig.add_trace(go.Scatter(x=class1['length'],y=class1['weight'],mode='markers',name='class1'))
```



In [80]:
```python
fig.add_trace(go.Scatter(x=class2['length'],y=class2['weight'],mode='markers',name='class2'))
```



In [81]:
```python
fig.update_xaxes(title_text='length')
fig.update_yaxes(title_text='weight')
iplot(fig)
```



---

2. Data statistic

In [20]:
```python
import numpy as np
class1.aggregate([np.mean])
```

Out[20]:

	length	weight
mean	65.550454	21.397417

Mean of the class1's length is 65.550454 & mean of the class1's weight is 21.397417

In [21]:
```python
class1.cov()
```

Out[21]:

	length	weight
length	27.023425	3.567138
weight	3.567138	70.775602

Variance of the class1's length is 27.023425

Variance of the class1's weight is 70.775602

Covariance of the class1's length & weight is 3.578138.

In [22]:
```python
class2.aggregate([np.mean])
```

Out[22]:

	length	weight
mean	79.332975	30.232327

Mean of the class2's length is 79.332975 & mean of the class1's weight is 30.232327

In [23]:
```python
class2.cov()
```

Out[23]:

	length	weight
length	41.427751	5.016526
weight	5.016526	44.896465

Variance of the class2's length is 41.427751

Variance of the class2's weight is 44.896465

Covariance of the class2's length & weight is 5.016526

---

3. Make predictions

3.a)

In [24]:
```python
obs.aggregate([np.mean])
```

Out[24]:

	length	weight
mean	73.333333	27.333333

Mean of the test set's length is 73.333333 & mean of the test set's weight is 27.33333

In [25]:
```python
obs.cov()
```

Out[25]:

	length	weight
length	116.666667	59.666667
weight	59.666667	36.666667

Variance of the test set's length is 116.666667

Variance of the test set's weight is 36.666667

Covariance of the test set's length & weight is 59.666667

In [26]:
```python
from scipy.stats import multivariate_normal
class1_obs=multivariate_normal.pdf(obs,mean=np.mean(class1),cov=class1.cov())
class2_obs=multivariate_normal.pdf(obs,mean=np.mean(class2),cov=class2.cov())
```

In [27]:
```python
print(class1_obs) #Likelihood of class1
print(class2_obs) #Likelihood of class2
```

```
[3.61921681e-03 6.72669372e-04 5.03300753e-05 2.28731204e-08
 2.05844000e-03 1.60302040e-03]
[1.93322815e-04 2.30652645e-03 3.67604637e-03 7.50553466e-04
 2.02111574e-05 1.28679336e-03]
```

3.b)

In [28]:
```python
class1_obs>class2_obs #1,2,2,2,1,1
```

Out[28]:
```
array([ True, False, False, False,  True,  True])
```

For the first sample of the test set, likelihood of class1 is greater than likelihood of class2. So it is reasonable to give class1 to the first sample.

In this way, we can classify 1,5,6 sample to class1 & 2,3,4 sample to class2.

This is only taken into account when it is assumed that the attributes of class 1&2 follow a multivariate normal distribution. The probability of samples of the test set coming from the distribution of class 1&2 was calculated. After that, it was decided which distribution the sample came from by the probability. </br>

3.c)

In [29]:
```python
from scipy.stats import norm
```

In [30]:
```python
class1_obs_len=scipy.stats.norm(np.mean(class1['length']),np.std(class1['length'])).pdf(obs['length'])
class2_obs_len=scipy.stats.norm(np.mean(class2['length']),np.std(class2['length'])).pdf(obs['length'])
```

In [31]:
```python
print(class1_obs_len)
print(class2_obs_len)
```

```
[7.66942696e-02 1.45362254e-02 1.55791799e-03 1.08449517e-06
 4.33677733e-02 5.32751185e-02]
[0.0050906  0.04955    0.06195711 0.01555987 0.00065394 0.02154118]
```

In [32]:
```python
class1_obs_len>class2_obs_len
```

Out[32]:
```
array([ True, False, False, False,  True,  True])
```

In this way, we can classify 1,5,6 sample to class1 & 2,3,4 sample to class2. It is same as above.

3.d)

In [33]:
```python
class1_obs_wei=scipy.stats.norm(np.mean(class1['weight']),np.std(class1['weight'])).pdf(obs['weight'])
class2_obs_wei=scipy.stats.norm(np.mean(class2['weight']),np.std(class2['weight'])).pdf(obs['weight'])
```

In [34]:
```python
print(class1_obs_wei)
print(class2_obs_wei)
```

```
[0.0475363  0.04344389 0.0246822  0.01040669 0.04700011 0.02810629]
[0.02791848 0.043978   0.0594439  0.0411591  0.01842784 0.05980296]
```

In [35]:
```python
class1_obs_wei>class2_obs_wei
```

Out[35]:
```
array([ True, False, False, False,  True, False])
```

In this way, we can classify 1,5 sample to class1 & 2,3,4,6 sample to class2. It is different from above.

6th sample may have simmilar length with class1's train set but may have simmilar weight with class2's train set.

And it is true because length of 6th sample is 70. mean length of class1's train set is 65.55, mean length of class2's train set is 79.33. So in 3.c), we can classify 6th sample as class1 since it is much more closer to class1's train set.

But, when we consider weight, mean weight of class1's train set is 21.4 & mean weight of class2's train set is 30.24. So when we consider only weight in classification, because weight of 6th sample is 30 so it is assigned to class2.

3.e

Length is the better feature for classification

By the plot, when we project points to length-axis or weight-axis, we obviously can see that they are well divided when we project to length-axis because the points are not overlapped which is different from the weight-axis.

It means same that when we concatenate the class1&class2 data and calculate the variance, variance of length is 81.3 > weight of the variance is 76.7.

Mean difference of length between class1 & class2 are bigger than difference of weight. So, class1&class2 are distributed apart from each class when we consider length as the only feature.

So we have to choose length as the only feature.

In [36]:
```python
np.var(pd.concat([class1,class2]))
```

Out[36]:
```
length    81.372799
weight    76.771585
dtype: float64
```

In [ ]: