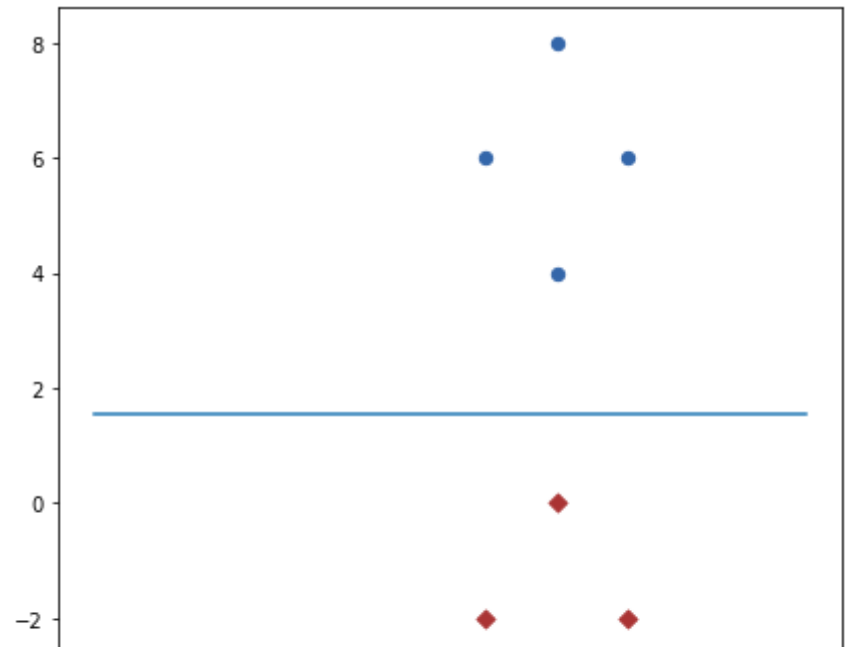
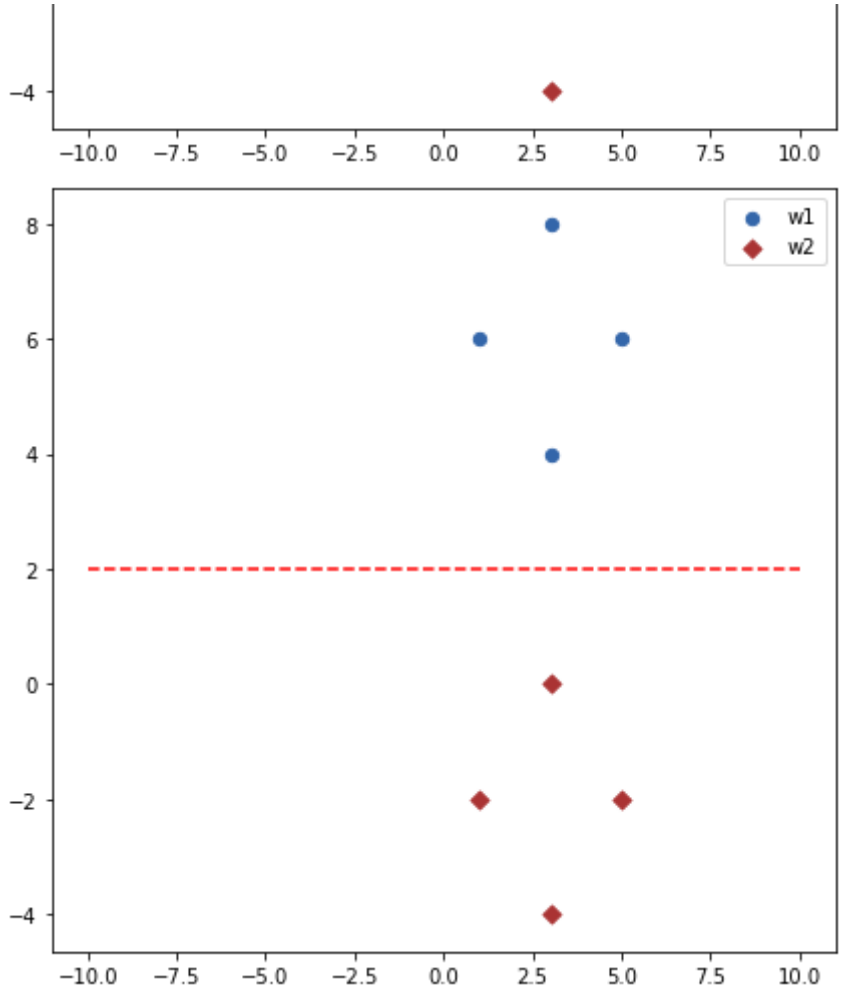
```
#%%
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt


def decision_boundary(x, w, x0):
    return(w.dot((x - x0).transpose()))


def main():
    w1 = np.array([[1, 6], [3, 4], [3, 8], [5, 6]])
    w2 = np.array([[3, 0], [1, -2], [3, -4], [5, -2]])

    print("w1 = \n", w1, "\nw2 = \n", w2)

    w1_mean = np.mean(w1, axis = 0)
    w2_mean = np.mean(w2, axis = 0)

    print("w1_mean = \n", w1, "\nw2_mean = \n", w2)

    z1 = w1 - w1_mean
    z2 = w2 - w2_mean
    w1_covariance = (1 / 3) * z1.transpose().dot(z1)
    w2_covariance = (1 / 3) * z2.transpose().dot(z2)

    print("w1_covariance = \n", w1_covariance, "\nw2_covariance = \n", w2_covarianc

    print("Since covariance(w1) = covariance(w2) = k.I, the decision boundary falls

    sigma_sq = 8/3

    print("When p(w1) = 0.8 and p(w2) = 0.2 :")
    print(np.array([w1_mean]).transpose())
    x0 = 0.5 * (w1_mean + w2_mean) - sigma_sq * np.log(0.8 / 0.2) * (w1_mean - w2_m
    w = (w1_mean - w2_mean)
    print("x0 ", x0)
    print("w ", w)
    print([decision_boundary(ele, w, x0) for ele in w1])
    print([decision_boundary(ele, w, x0) for ele in w2])

    f, ax = plt.subplots(figsize=(7, 7))
    c1, c2 = "#3366AA", "#AA3333"
    ax.scatter(*w1.T, c=c1, s=40)
    ax.scatter(*w2.T, c=c2, marker="D", s=40)
    x_vec = np.linspace(-10, 10, 5)
    y_vec = ((w1_mean[0] - w2_mean[0])*x_vec - (w1_mean[0] - w2_mean[0])* x0[0] +
    print(y_vec)
    plt.plot(x_vec, y_vec)

    print("When p(w1) = p(w2) :")
    print(np.array([w1_mean]).transpose())
    x0 = 0.5 * (w1_mean + w2_mean) - sigma_sq * np.log(1) * (w1_mean - w2_mean) /
```

```
    x0 = 0.5 * (w1_mean + w2_mean) - sigma_sq * np.log(1) / (w1_mean - w2_mean)
    w = (w1_mean - w2_mean)
    print("x0", x0)
    print([decision_boundary(ele, w, x0) for ele in w1])
    print([decision_boundary(ele, w, x0) for ele in w2])

    f, ax = plt.subplots(figsize=(7, 7))
    c1, c2 = "#3366AA", "#AA3333"
    ax.scatter(*w1.T, c=c1, s=40, label = "w1")
    ax.legend()
    ax.scatter(*w2.T, c=c2, marker="D", s=40, label = "w2")
    ax.legend()
    x_vec = np.linspace(-10, 10, 5)
    y_vec = (w[1]*x0[1] - w[0]*(x_vec - x0[0]))/w[1]
    print(y_vec)
    plt.plot(x_vec, y_vec, 'r--')
    plt.show()



if __name__ == "__main__":
    main()

# %%
```

```
w1 =
 [[1 6]
 [3 4]
 [3 8]
 [5 6]]
w2 =
 [[ 3  0]
 [ 1 -2]
 [ 3 -4]
 [ 5 -2]]
w1_mean =
 [[1 6]
 [3 4]
 [3 8]
 [5 6]]
w2_mean =
 [[ 3  0]
 [ 1 -2]
 [ 3 -4]
 [ 5 -2]]
w1_covariance =
 [[2.66666667 0.        ]
 [0.         2.66666667]]
w2_covariance =
 [[2.66666667 0.        ]
 [0.         2.66666667]]
Since covariance(w1) = covariance(w2) = k.I, the decision boundary falls in th
When p(w1) = 0.8 and p(w2) = 0.2 :
[[3.]
 [6.]]
x0  [3.         1.53790188]
w  [0. 8.]
[35.69678496298637, 19.696784962986374, 51.69678496298637, 35.69678496298637]
[-12.303215037013626, -28.303215037013626, -44.30321503701363, -28.30321503701
[1.53790188 1.53790188 1.53790188 1.53790188 1.53790188]
When p(w1) = p(w2) :
[[3.]
 [6.]]
x0 [3. 2.]
[32.0, 16.0, 48.0, 32.0]
[-16.0, -32.0, -48.0, -32.0]
[2. 2. 2. 2. 2.]
```

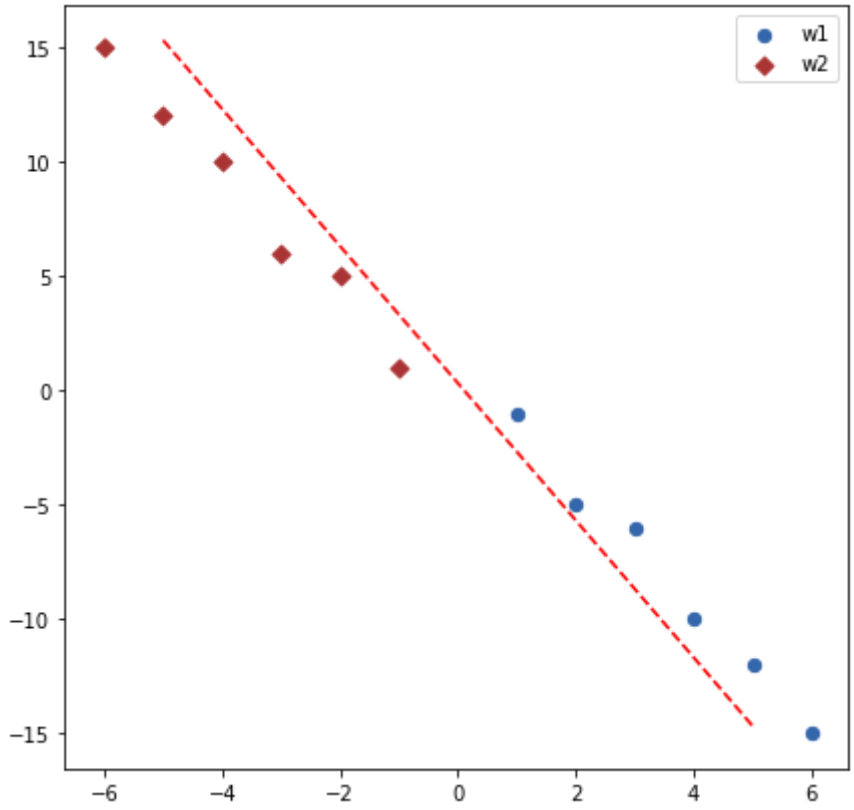





```
#%%
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

def decision_boundary(x, w, x0):
    return(w.dot((x - x0).transpose()))


def main():
    w1 = np.array([[1, -1], [2, -5], [3, -6], [4, -10], [5, -12], [6, -15]])
    w2 = (-1)*w1
    w1_mean = np.mean(w1, axis = 0)
    w2_mean = np.mean(w2, axis = 0)

    z1 = np.array([list(map((lambda num : num - w1_mean[0]), w1[:, 0].tolist())),
    z2 = np.array([list(map((lambda num : num - w2_mean[0]), w2[:, 0].tolist())),

    w1_covariance = (1 / 5) * z1.dot(z1.transpose())
    w2_covariance = (1 / 5) * z2.dot(z2.transpose())

    print(w1_covariance, "\n", w2_covariance)

    print("Since covariance(w1) = covariance(w2) , the decision boundary falls und

    sigma_sq = 8/3
```

```
        sigma_sq = 0.5

        print("When p(w1) = 0.8 and p(w2) = 0.2 :")
        x0 = 0.5 * (w1_mean + w2_mean) - sigma_sq * np.log(0.3 / 0.7) * (w1_mean - w2_m
        print("x0 : ", x0)
        w = np.linalg.inv(w1_covariance).dot(w1_mean - w2_mean)
        print("w : ", w)

        print([decision_boundary(ele, w, x0) for ele in w1])
        print([decision_boundary(ele, w, x0) for ele in w2])


        x_vec = np.linspace(-5, 5, 10)
        y_vec = (w[1]*x0[1] - w[0]*(x_vec - x0[0]))/w[1]
        print(x_vec)
        print(y_vec)
        f, ax = plt.subplots(figsize=(7, 7))
        c1, c2 = "#3366AA", "#AA3333"
        ax.scatter(*w1.T, c=c1, s=40, label  = "w1")
        ax.scatter(*w2.T, c=c2, marker="D", s=40, label = "w2")
        ax.legend()
        plt.plot(x_vec, y_vec, 'r--')

    if __name__ == "__main__":
        main()

    # %%
```

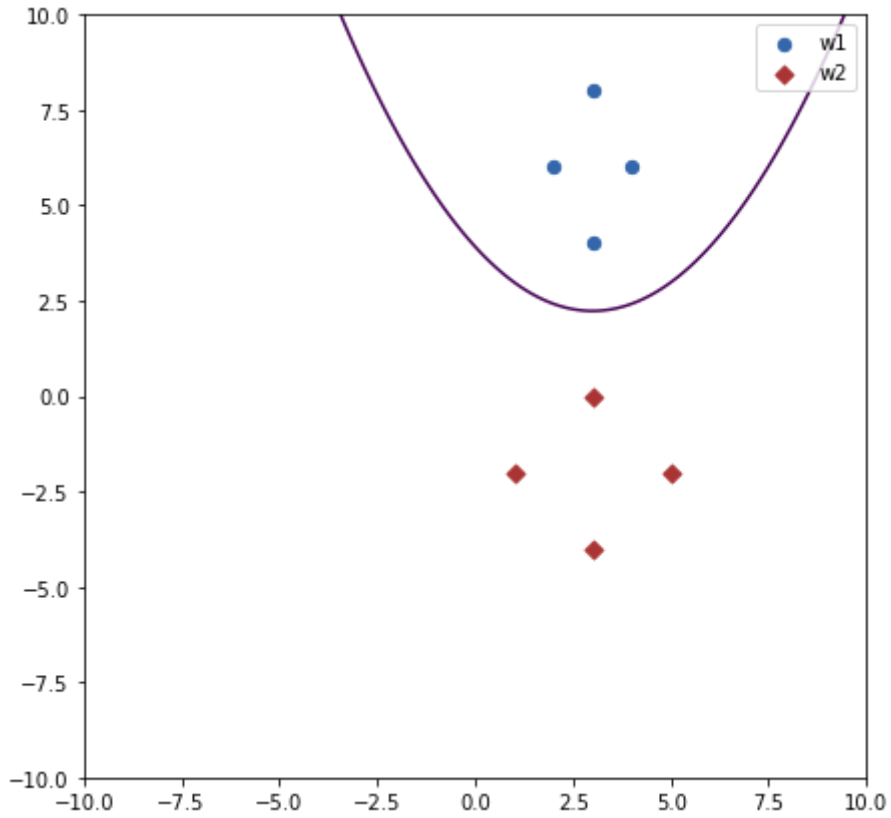
```
[[ 3.5        -9.5       ]
 [-9.5        26.16666667]]
 [[ 3.5        -9.5       ]
 [-9.5        26.16666667]]
Since covariance(w1) = covariance(w2) , the decision boundary falls under the
When p(w1) = 0.8 and p(w2) = 0.2 :
x0 :  [ 0.48417021 -1.12973048]
w :  [21.  7.]
[11.740539038967617, 4.740539038967558, 18.740539038967725, 11.740539038967668
[-16.259460961032698, -9.25946096103263, -23.259460961032786, -16.259460961032
[-5.         -3.88888889 -2.77777778 -1.66666667 -0.55555556  0.55555556
  1.66666667  2.77777778  3.88888889  5.        ]
[ 15.32278014  11.9894468    8.65611347   5.32278014   1.9894468
  -1.34388653  -4.67721986  -8.0105532  -11.34388653 -14.67721986]
```




```
#%%
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

def decision_boundary(x, W, w, x0):
    return(x.dot(W.dot(x.transpose())) + w.dot(x.transpose()) + x0)

def main():
    w1 = np.array([[2, 6], [3, 4], [3, 8], [4, 6]])
    w2 = np.array([[3, 0], [1, -2], [3, -4], [5, -2]])
    w1_mean = np.mean(w1, axis = 0)
    w2_mean = np.mean(w2, axis = 0)

    z1 = np.array([list(map((lambda num : num - w1_mean[0]), w1[:, 0].tolist())),
    z2 = np.array([list(map((lambda num : num - w2_mean[0]), w2[:, 0].tolist())),
```

```
    w1_covariance = (1 / 3) * z1.dot(z1.transpose())
    w2_covariance = (1 / 3) * z2.dot(z2.transpose())

    print(w1_covariance, "\n", w2_covariance)

    print("Since covariance(w1) = covariance(w2) = k.I, the decision boundary falls

    sigma_sq = 8/3

    print("When p(w1) = p(w2) :")
    w10 = -(0.5) * ((w1_mean.transpose().dot(np.linalg.inv(w1_covariance).dot(w1_me
    W1 = -(0.5) * np.linalg.inv(w1_covariance)
    w20 = -(0.5) * ((w2_mean.transpose().dot(np.linalg.inv(w2_covariance).dot(w2_me
    W2 = -(0.5) * np.linalg.inv(w2_covariance)
    w11 = (w1_mean).dot(np.linalg.inv(w1_covariance))
    w22 = (w2_mean).dot(np.linalg.inv(w2_covariance))
    W = W1 - W2
    w = w11 - w22
    x0 = w10 - w20

    print("When p(w1) = p(w2) :")
    print(np.array([w1_mean]).transpose())
    print("x0", x0)
    print([decision_boundary(ele, W, w, x0) for ele in w1])
    print([decision_boundary(ele, W, w, x0) for ele in w2])

    f, ax = plt.subplots(figsize=(7, 7))
    c1, c2 = "#3366AA", "#AA3333"
    ax.scatter(*w1.T, c=c1, s=40, label = "w1")
    ax.scatter(*w2.T, c=c2, marker="D", s=40, label = "w2")
    ax.legend()
    y, x = np.ogrid[-10 : 10 : 1000j, -10 : 10 : 1000j]
    plt.contour(x.ravel(), y.ravel(), W[0][0] * x ** 2 + x*y*(W[0][1] + W[1][0]) +

    # print(y_vec)
    # plt.plot(x_vec, y_vec, 'r--')
    plt.show()

    print("w22", w22)



if __name__ == "__main__":
    main()

# %%
```

```
[[0.66666667 0.        ]
 [0.         2.66666667]]
 [[2.66666667 0.        ]
 [0.         2.66666667]]
Since covariance(w1) = covariance(w2) = k.I, the decision boundary falls in th
When p(w1) = p(w2) :
When p(w1) = p(w2) :
[[3.]
 [6.]]
x0 -11.755647180559944
[10.744352819440056, 5.306852819440056, 17.306852819440056, 10.744352819440056
[-6.693147180559944, -14.943147180559944, -18.693147180559944, -14.943147180559
```




```
w22 [ 1.125 -0.75 ]
```

```
#%%
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from google.colab import files
uploaded = files.upload()
import io


def discriminant_function(x, covariance, mean):
    g = (0.5) * (-(x - mean).dot(np.linalg.inv(covariance).dot((x - mean).T)) - np
    return(g)

def main():
    data = pd.read_csv(io.BytesIO(uploaded['iris.csv']))
    variety = data['variety'].unique()
    column = data.columns
```

```
    setosa_train, t1 = train_test_split(data[0 : 50], test_size=0.2)
    versicolor_train, t2 = train_test_split(data[51 : 100], test_size=0.2)
    virginica_train, t3 = train_test_split(data[101 : 150], test_size=0.2)
    frames = [t1, t2, t3]
    test = pd.concat(frames)
    print("setosa train : \n{}\nversicolor : \n{}\nvirginica : \n{}\n, test : \n{}"
    print(data.describe())
    setosa_mean = np.mean(setosa_train, axis = 0)
    versicolor_mean = np.mean(versicolor_train, axis = 0)
    virginica_mean = np.mean(virginica_train, axis = 0)
    print(column)
    print("setosa_mean : \n{}\nversicolor_mean : \n{}\nvirginica_mean : \n{}\n".fo
    setosa_z = np.array([list(map((lambda num : num -  setosa_mean[0]), setosa_tra
    virginica_z = np.array([list(map((lambda num : num -  virginica_mean[0]), virg
    versicolor_z = np.array([list(map((lambda num : num -  versicolor_mean[0]), ve
    print("setosa_z : \n{}\nversicolor_z : \n{}\nvirginia_z : \n{}\n".format(setosa

    setosa_covariance = (1/39) * (setosa_z.dot(setosa_z.T))
    versicolor_covariance = (1/39) * (versicolor_z.dot(versicolor_z.T))
    virginica_covariance = (1/39) * (virginica_z.dot(virginica_z.T))
    print("setosa_covariance : \n{}\nversicolor_covariance : \n{}\nvirginia_covaria

    count = 0

    for index, row in test.iterrows():
        test_vector = np.array([row[0], row[1], row[2], row[3]])

        g1 = discriminant_function(test_vector, setosa_covariance, setosa_mean)
        g2 = discriminant_function(test_vector, versicolor_covariance, versicolor_m
        g3 = discriminant_function(test_vector, virginica_covariance, virginica_mea



        print(g1, g2, g3)
        if(g1 > g2 and g1 > g3):
            if(row[4] == 'Setosa'):
                count = count + 1
            print("setosa")
        elif(g2 > g3):
            if(row[4] == 'Versicolor'):
                count = count + 1

            print("versicolor")
        else:
            if(row[4] == 'Virginica'):
                count = count + 1
            print("virginica")

    print("Accuracy is : ", count / 30)


if(__name__ == "__main__"):
    main()

# %%
```

Choose Files No file chosen Upload widget is only available when the cell has been executed

```
Saving iris.csv to iris.csv
setosa train :
    sepal.length  sepal.width  petal.length  petal.width variety
34           4.9          3.1           1.5          0.2  Setosa
32           5.2          4.1           1.5          0.1  Setosa
30           4.8          3.1           1.6          0.2  Setosa
5            5.4          3.9           1.7          0.4  Setosa
48           5.3          3.7           1.5          0.2  Setosa
1            4.9          3.0           1.4          0.2  Setosa
24           4.8          3.4           1.9          0.2  Setosa
2            4.7          3.2           1.3          0.2  Setosa
25           5.0          3.0           1.6          0.2  Setosa
44           5.1          3.8           1.9          0.4  Setosa
29           4.7          3.2           1.6          0.2  Setosa
8            4.4          2.9           1.4          0.2  Setosa
39           5.1          3.4           1.5          0.2  Setosa
0            5.1          3.5           1.4          0.2  Setosa
33           5.5          4.2           1.4          0.2  Setosa
46           5.1          3.8           1.6          0.2  Setosa
6            4.6          3.4           1.4          0.3  Setosa
22           4.6          3.6           1.0          0.2  Setosa
19           5.1          3.8           1.5          0.3  Setosa
40           5.0          3.5           1.3          0.3  Setosa
7            5.0          3.4           1.5          0.2  Setosa
26           5.0          3.4           1.6          0.4  Setosa
20           5.4          3.4           1.7          0.2  Setosa
9            4.9          3.1           1.5          0.1  Setosa
11           4.8          3.4           1.6          0.2  Setosa
17           5.1          3.5           1.4          0.3  Setosa
41           4.5          2.3           1.3          0.3  Setosa
15           5.7          4.4           1.5          0.4  Setosa
21           5.1          3.7           1.5          0.4  Setosa
36           5.5          3.5           1.3          0.2  Setosa
18           5.7          3.8           1.7          0.3  Setosa
31           5.4          3.4           1.5          0.4  Setosa
43           5.0          3.5           1.6          0.6  Setosa
13           4.3          3.0           1.1          0.1  Setosa
16           5.4          3.9           1.3          0.4  Setosa
49           5.0          3.3           1.4          0.2  Setosa
14           5.8          4.0           1.2          0.2  Setosa
47           4.6          3.2           1.4          0.2  Setosa
27           5.2          3.5           1.5          0.2  Setosa
4            5.0          3.6           1.4          0.2  Setosa
versicolor :
    sepal.length  sepal.width  petal.length  petal.width     variety
56           6.3          3.3           4.7          1.6  Versicolor
97           6.2          2.9           4.3          1.3  Versicolor
62           6.0          2.2           4.0          1.0  Versicolor
66           5.6          3.0           4.5          1.5  Versicolor
88           5.6          3.0           4.1          1.3  Versicolor
99           5.7          2.8           4.1          1.3  Versicolor
83           6.0          2.7           5.1          1.6  Versicolor
67           5.8          2.7           4.1          1.0  Versicolor
59           5.2          2.7           3.9          1.4  Versicolor
75           6.6          3.0           4.4          1.4  Versicolor
82           5.8          2.7           3.9          1.2  Versicolor
52           6.9          3.1           4.9          1.5  Versicolor
65           6.7          3.1           4.4          1.4  Versicolor
61           5.9          3.0           4.2          1.5  Versicolor
53           5.5          2.3           4.0          1.3  Versicolor
```

76	6.8	2.8	4.8	1.4	Versicolor
70	5.9	3.2	4.8	1.8	Versicolor
78	6.0	2.9	4.5	1.5	Versicolor
57	4.9	2.4	3.3	1.0	Versicolor
89	5.5	2.5	4.0	1.3	Versicolor
55	5.7	2.8	4.5	1.3	Versicolor
96	5.7	2.9	4.2	1.3	Versicolor
79	5.7	2.6	3.5	1.0	Versicolor
64	5.6	2.9	3.6	1.3	Versicolor
86	6.7	3.1	4.7	1.5	Versicolor
69	5.6	2.5	3.9	1.1	Versicolor
51	6.4	3.2	4.5	1.5	Versicolor
71	6.1	2.8	4.0	1.3	Versicolor
93	5.0	2.3	3.3	1.0	Versicolor
90	5.5	2.6	4.4	1.2	Versicolor
80	5.5	2.4	3.8	1.1	Versicolor
60	5.0	2.0	3.5	1.0	Versicolor
84	5.4	3.0	4.5	1.5	Versicolor
98	5.1	2.5	3.0	1.1	Versicolor
73	6.1	2.8	4.7	1.2	Versicolor
87	6.3	2.3	4.4	1.3	Versicolor
81	5.5	2.4	3.7	1.0	Versicolor
92	5.8	2.6	4.0	1.2	Versicolor
68	6.2	2.2	4.5	1.5	Versicolor

virginica :

	sepal.length	sepal.width	petal.length	petal.width	variety
139	6.9	3.1	5.4	2.1	Virginica
126	6.2	2.8	4.8	1.8	Virginica
149	5.9	3.0	5.1	1.8	Virginica
112	6.8	3.0	5.5	2.1	Virginica
117	7.7	3.8	6.7	2.2	Virginica
101	5.8	2.7	5.1	1.9	Virginica
123	6.3	2.7	4.9	1.8	Virginica
132	6.4	2.8	5.6	2.2	Virginica
143	6.8	3.2	5.9	2.3	Virginica
129	7.2	3.0	5.8	1.6	Virginica
138	6.0	3.0	4.8	1.8	Virginica
109	7.2	3.6	6.1	2.5	Virginica
102	7.1	3.0	5.9	2.1	Virginica
133	6.3	2.8	5.1	1.5	Virginica
140	6.7	3.1	5.6	2.4	Virginica
110	6.5	3.2	5.1	2.0	Virginica
103	6.3	2.9	5.6	1.8	Virginica
146	6.3	2.5	5.0	1.9	Virginica
106	4.9	2.5	4.5	1.7	Virginica
121	5.6	2.8	4.9	2.0	Virginica
124	6.7	3.3	5.7	2.1	Virginica
122	7.7	2.8	6.7	2.0	Virginica
104	6.5	3.0	5.8	2.2	Virginica
144	6.7	3.3	5.7	2.5	Virginica
119	6.0	2.2	5.0	1.5	Virginica
128	6.4	2.8	5.6	2.1	Virginica
120	6.9	3.2	5.7	2.3	Virginica
116	6.5	3.0	5.5	1.8	Virginica
107	7.3	2.9	6.3	1.8	Virginica
136	6.3	3.4	5.6	2.4	Virginica
127	6.1	3.0	4.9	1.8	Virginica
142	5.8	2.7	5.1	1.9	Virginica
115	6.4	3.2	5.3	2.3	Virginica
108	6.7	2.5	5.8	1.8	Virginica
137	6.4	3.1	5.5	1.8	Virginica

```
111           6.4          2.7           5.3          1.9  Virginica
134           6.1          2.6           5.6          1.4  Virginica
148           6.2          3.4           5.4          2.3  Virginica
147           6.5          3.0           5.2          2.0  Virginica
, test :
     sepal.length  sepal.width  petal.length  petal.width     variety
37            4.9          3.6           1.4          0.1      Setosa
10            5.4          3.7           1.5          0.2      Setosa
42            4.4          3.2           1.3          0.2      Setosa
45            4.8          3.0           1.4          0.3      Setosa
3             4.6          3.1           1.5          0.2      Setosa
28            5.2          3.4           1.4          0.2      Setosa
12            4.8          3.0           1.4          0.1      Setosa
38            4.4          3.0           1.3          0.2      Setosa
23            5.1          3.3           1.7          0.5      Setosa
35            5.0          3.2           1.2          0.2      Setosa
77            6.7          3.0           5.0          1.7  Versicolor
58            6.6          2.9           4.6          1.3  Versicolor
54            6.5          2.8           4.6          1.5  Versicolor
94            5.6          2.7           4.2          1.3  Versicolor
72            6.3          2.5           4.9          1.5  Versicolor
74            6.4          2.9           4.3          1.3  Versicolor
91            6.1          3.0           4.6          1.4  Versicolor
85            6.0          3.4           4.5          1.6  Versicolor
95            5.7          3.0           4.2          1.2  Versicolor
63            6.1          2.9           4.7          1.4  Versicolor
145           6.7          3.0           5.2          2.3   Virginica
130           7.4          2.8           6.1          1.9   Virginica
113           5.7          2.5           5.0          2.0   Virginica
118           7.7          2.6           6.9          2.3   Virginica
125           7.2          3.2           6.0          1.8   Virginica
141           6.9          3.1           5.1          2.3   Virginica
105           7.6          3.0           6.6          2.1   Virginica
135           7.7          3.0           6.1          2.3   Virginica
131           7.9          3.8           6.4          2.0   Virginica
114           5.8          2.8           5.1          2.4   Virginica

       sepal.length  sepal.width  petal.length  petal.width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.057333      3.758000     1.199333
std        0.828066     0.435866      1.765298     0.762238
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
Index(['sepal.length', 'sepal.width', 'petal.length', 'petal.width',
       'variety'],
      dtype='object')
setosa_mean :
sepal.length    5.0425
sepal.width     3.4725
petal.length    1.4750
petal.width     0.2525
dtype: float64
versicolor_mean :
sepal.length    6.474359
sepal.width     2.964103
petal.length    5.464103
petal.width     1.984615
dtype: float64
virginica mean :
```

```
sepal.length    5.841026
sepal.width     2.723077
petal.length    4.171795
petal.width     1.300000
dtype: float64

setosa_z :
[[-0.1425  0.1575 -0.2425  0.3575  0.2575 -0.1425 -0.2425 -0.3425 -0.0425
   0.0575 -0.3425 -0.6425  0.0575  0.0575  0.4575  0.0575 -0.4425 -0.4425
   0.0575 -0.0425 -0.0425 -0.0425  0.3575 -0.1425 -0.2425  0.0575 -0.5425
   0.6575  0.0575  0.4575  0.6575  0.3575 -0.0425 -0.7425  0.3575 -0.0425
   0.7575 -0.4425  0.1575 -0.0425]
 [-0.3725  0.6275 -0.3725  0.4275  0.2275 -0.4725 -0.0725 -0.2725 -0.4725
   0.3275 -0.2725 -0.5725 -0.0725  0.0275  0.7275  0.3275 -0.0725  0.1275
   0.3275  0.0275 -0.0725 -0.0725 -0.0725 -0.3725 -0.0725  0.0275 -1.1725
   0.9275  0.2275  0.0275  0.3275 -0.0725  0.0275 -0.4725  0.4275 -0.1725
   0.5275 -0.2725  0.0275  0.1275]
 [ 0.025   0.025   0.125   0.225   0.025  -0.075   0.425  -0.175   0.125
   0.425   0.125  -0.075   0.025  -0.075  -0.075   0.125  -0.075  -0.475
   0.025  -0.175   0.025   0.125   0.225   0.025   0.125  -0.075  -0.175
   0.025   0.025  -0.175   0.225   0.025   0.125  -0.375  -0.175  -0.075
  -0.275  -0.075   0.025  -0.075 ]
 [-0.0525 -0.1525 -0.0525  0.1475 -0.0525 -0.0525 -0.0525 -0.0525 -0.0525
   0.1475 -0.0525 -0.0525 -0.0525 -0.0525 -0.0525 -0.0525  0.0475 -0.0525
   0.0475  0.0475 -0.0525  0.1475 -0.0525 -0.1525 -0.0525  0.0475  0.0475
   0.1475  0.1475 -0.0525  0.0475  0.1475  0.3475 -0.1525  0.1475 -0.0525
  -0.0525 -0.0525 -0.0525 -0.0525]]
versicolor_z :
[[ 4.58974359e-01  3.58974359e-01  1.58974359e-01 -2.41025641e-01
  -2.41025641e-01 -1.41025641e-01  1.58974359e-01 -4.10256410e-02
  -6.41025641e-01  7.58974359e-01 -4.10256410e-02  1.05897436e+00
   8.58974359e-01  5.89743590e-02 -3.41025641e-01  9.58974359e-01
   5.89743590e-02  1.58974359e-01 -9.41025641e-01 -3.41025641e-01
  -1.41025641e-01 -1.41025641e-01 -1.41025641e-01 -2.41025641e-01
   8.58974359e-01 -2.41025641e-01  5.58974359e-01  2.58974359e-01
  -8.41025641e-01 -3.41025641e-01 -3.41025641e-01 -8.41025641e-01
  -4.41025641e-01 -7.41025641e-01  2.58974359e-01  4.58974359e-01
  -3.41025641e-01 -4.10256410e-02  3.58974359e-01]
 [ 5.76923077e-01  1.76923077e-01 -5.23076923e-01  2.76923077e-01
   2.76923077e-01  7.69230769e-02 -2.30769231e-02 -2.30769231e-02
  -2.30769231e-02  2.76923077e-01 -2.30769231e-02  3.76923077e-01
   3.76923077e-01  2.76923077e-01 -4.23076923e-01  7.69230769e-02
   4.76923077e-01  1.76923077e-01 -3.23076923e-01 -2.23076923e-01
   7.69230769e-02  1.76923077e-01 -1.23076923e-01  1.76923077e-01
   3.76923077e-01 -2.23076923e-01  4.76923077e-01  7.69230769e-02
  -4.23076923e-01 -1.23076923e-01 -3.23076923e-01 -7.23076923e-01
   2.76923077e-01 -2.23076923e-01  7.69230769e-02 -4.23076923e-01
  -3.23076923e-01 -1.23076923e-01 -5.23076923e-01]
 [ 5.28205128e-01  1.28205128e-01 -1.71794872e-01  3.28205128e-01
  -7.17948718e-02 -7.17948718e-02  9.28205128e-01 -7.17948718e-02
  -2.71794872e-01  2.28205128e-01 -2.71794872e-01  7.28205128e-01
   2.28205128e-01  2.82051282e-02 -1.71794872e-01  6.28205128e-01
   6.28205128e-01  3.28205128e-01 -8.71794872e-01 -1.71794872e-01
   3.28205128e-01  2.82051282e-02 -6.71794872e-01 -5.71794872e-01
   5.28205128e-01 -2.71794872e-01  3.28205128e-01 -1.71794872e-01
  -8.71794872e-01  2.28205128e-01 -3.71794872e-01 -6.71794872e-01
   3.28205128e-01 -1.17179487e+00  5.28205128e-01  2.28205128e-01
  -4.71794872e-01 -1.71794872e-01  3.28205128e-01]
 [ 3.00000000e-01 -2.22044605e-16 -3.00000000e-01  2.00000000e-01
  -2.22044605e-16 -2.22044605e-16  3.00000000e-01 -3.00000000e-01
   1.00000000e-01  1.00000000e-01 -1.00000000e-01  2.00000000e-01
```

```
   1.00000000e-01  2.00000000e-01 -2.22044605e-16  1.00000000e-01
   5.00000000e-01  2.00000000e-01 -3.00000000e-01 -2.22044605e-16
  -2.22044605e-16 -2.22044605e-16 -3.00000000e-01 -2.22044605e-16
   2.00000000e-01 -2.00000000e-01  2.00000000e-01 -2.22044605e-16
  -3.00000000e-01 -1.00000000e-01 -2.00000000e-01 -3.00000000e-01
   2.00000000e-01 -2.00000000e-01 -1.00000000e-01 -2.22044605e-16
  -3.00000000e-01 -1.00000000e-01  2.00000000e-01]]
virginia_z :
[[ 0.42564103 -0.27435897 -0.57435897  0.32564103  1.22564103 -0.67435897
  -0.17435897 -0.07435897  0.32564103  0.72564103 -0.47435897  0.72564103
   0.62564103 -0.17435897  0.22564103  0.02564103 -0.17435897 -0.17435897
  -1.57435897 -0.87435897  0.22564103  1.22564103  0.02564103  0.22564103
  -0.47435897 -0.07435897  0.42564103  0.02564103  0.82564103 -0.17435897
  -0.37435897 -0.67435897 -0.07435897  0.22564103 -0.07435897 -0.07435897
  -0.37435897 -0.27435897  0.02564103]
 [ 0.13589744 -0.16410256  0.03589744  0.03589744  0.83589744 -0.26410256
  -0.26410256 -0.16410256  0.23589744  0.03589744  0.03589744  0.63589744
   0.03589744 -0.16410256  0.13589744  0.23589744 -0.06410256 -0.46410256
  -0.46410256 -0.16410256  0.33589744 -0.16410256  0.03589744  0.33589744
  -0.76410256 -0.16410256  0.23589744  0.03589744 -0.06410256  0.43589744
   0.03589744 -0.26410256  0.23589744 -0.46410256  0.13589744 -0.26410256
  -0.36410256  0.43589744  0.03589744]
 [-0.06410256 -0.66410256 -0.36410256  0.03589744  1.23589744 -0.36410256
  -0.56410256  0.13589744  0.43589744  0.33589744 -0.66410256  0.63589744
   0.43589744 -0.36410256  0.13589744 -0.36410256  0.13589744 -0.46410256
  -0.96410256 -0.56410256  0.23589744  1.23589744  0.33589744  0.23589744
  -0.46410256  0.13589744  0.23589744  0.03589744  0.83589744  0.13589744
  -0.56410256 -0.36410256 -0.16410256  0.33589744  0.03589744 -0.16410256
   0.13589744 -0.06410256 -0.26410256]
 [ 0.11538462 -0.18461538 -0.18461538  0.11538462  0.21538462 -0.08461538
  -0.18461538  0.21538462  0.31538462 -0.38461538 -0.18461538  0.51538462
   0.11538462 -0.48461538  0.41538462  0.01538462 -0.18461538 -0.08461538
  -0.28461538  0.01538462  0.11538462  0.01538462  0.21538462  0.51538462
  -0.48461538  0.11538462  0.31538462 -0.18461538 -0.18461538  0.41538462
  -0.18461538 -0.08461538  0.31538462 -0.18461538 -0.18461538 -0.08461538
  -0.58461538  0.31538462  0.01538462]]

setosa_covariance :
[[0.12404487 0.10299359 0.01391026 0.01001923]
 [0.10299359 0.15589103 0.00955128 0.01071154]
 [0.01391026 0.00955128 0.03269231 0.0049359 ]
 [0.01001923 0.01071154 0.0049359  0.01076282]]
versicolor_covariance :
[[0.24754767 0.08289941 0.17320842 0.05282051]
 [0.08289941 0.09921105 0.08629191 0.04435897]
 [0.17320842 0.08629191 0.22458909 0.07410256]
 [0.05282051 0.04435897 0.07410256 0.04051282]]
virginia_covariance :
[[0.29370151 0.0852334  0.22907955 0.0539645 ]
 [0.0852334  0.09768573 0.0717883  0.05944773]
 [0.22907955 0.0717883  0.23460881 0.05149901]
 [0.0539645  0.05944773 0.05149901 0.07412229]]

4.798773404340389 -63.547301066194414 -133.66743553787666
setosa
5.571483582011266 -62.90066338172752 -143.12100974100215
setosa
4.145379589938168 -43.41382142210721 -103.31234183831633
setosa
5.262364971309232 -33.24925020718591 -97.46826026801394
setosa
5.525974710351561 -38.03669860628337 -95.53213842668622
```

```
setosa
5.778106708495176 -51.627280093452335 -128.39002522062535
setosa
4.90522496045759 -41.00659241486755 -105.40305446959692
setosa
4.579873049588019 -36.98417790379129 -95.66148393235342
setosa
2.688735228000658 -32.30646829415598 -94.80511762554676
setosa
4.796815122487667 -47.02676044614264 -124.32944019277802
setosa
-252.42633546136673 2.6350375114407227 0.5026643641659394
versicolor
-183.83182695180417 3.6609077264745227 -7.211937640707451
versicolor
-200.12578158959408 3.9274600152248795 -3.1488771703809393
versicolor
-144.0621289785059 5.114336837328381 -2.0840071325112204
versicolor
-231.42969323850994 2.0484584953041636 2.019557744433179
versicolor
-156.09929756867453 4.370040947015563 -9.619086590726543
versicolor
-181.7275422368388 4.641368392694816 -2.302924250866911
versicolor
-184.1671057059061 2.583281323159863 -5.636308470010349
versicolor
-132.5906051344239 3.1001693268843526 -7.6401526191883
versicolor
-192.46754372633757 4.452515172772268 -0.489743723479136
versicolor
-351.009404542904 -17.92509046357925 1.6337398533537186
virginica
-415.135082051485 -7.289112491868434 2.2416369110087366
virginica
-291.4395553855189 -11.32536468494602 2.29470644512392
virginica
-589.1743393623356 -27.90251530885761 -6.520896064689233
virginica
-374.40368450512403 -2.4908666403594095 2.6434855140119033
virginica
-342.17624889193104 -18.955498784091425 -0.9427961960366682
virginica
-501.57038976548154 -12.063423162601758 1.538469643557681
virginica
-467.13518164112617 -16.824277237256105 -0.37488222376626457
virginica
-444.0236419227657 -6.549275096058609 -3.1076484251460927
virginica
-358.04101400783384 -27.40720299716361 0.6112543361545297
virginica
Accuracy is :  1.0
```

```
#%%
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import datasets
```

```
import numpy as np
import matplotlib.pyplot as plt
from mlxtend.plotting import plot_decision_regions
import matplotlib.gridspec as gridspec
import itertools



# Initializing Classifiers
clf1 = GaussianNB()

# Loading some example data
iris = datasets.load_iris()
X = iris.data[:, [2,3]]
y = iris.target
gs = gridspec.GridSpec(2, 2)

fig = plt.figure(figsize=(10,8))

labels = ['Naive Bayes']
for clf, lab, grd in zip([clf1], labels, itertools.product([0, 1], repeat=2)):
    clf.fit(X, y)
    ax = plt.subplot(gs[grd[0], grd[1]])
    fig = plot_decision_regions(X=X, y=y, clf=clf, legend=2)
    plt.title(lab)

plt.show()

# %%
```