ELG 5255[EG] Applied Machine Learning Summer

Assignment -Two (Parametric methods)

(Group 34)

Ahmed Hallaba   Omar Sorour

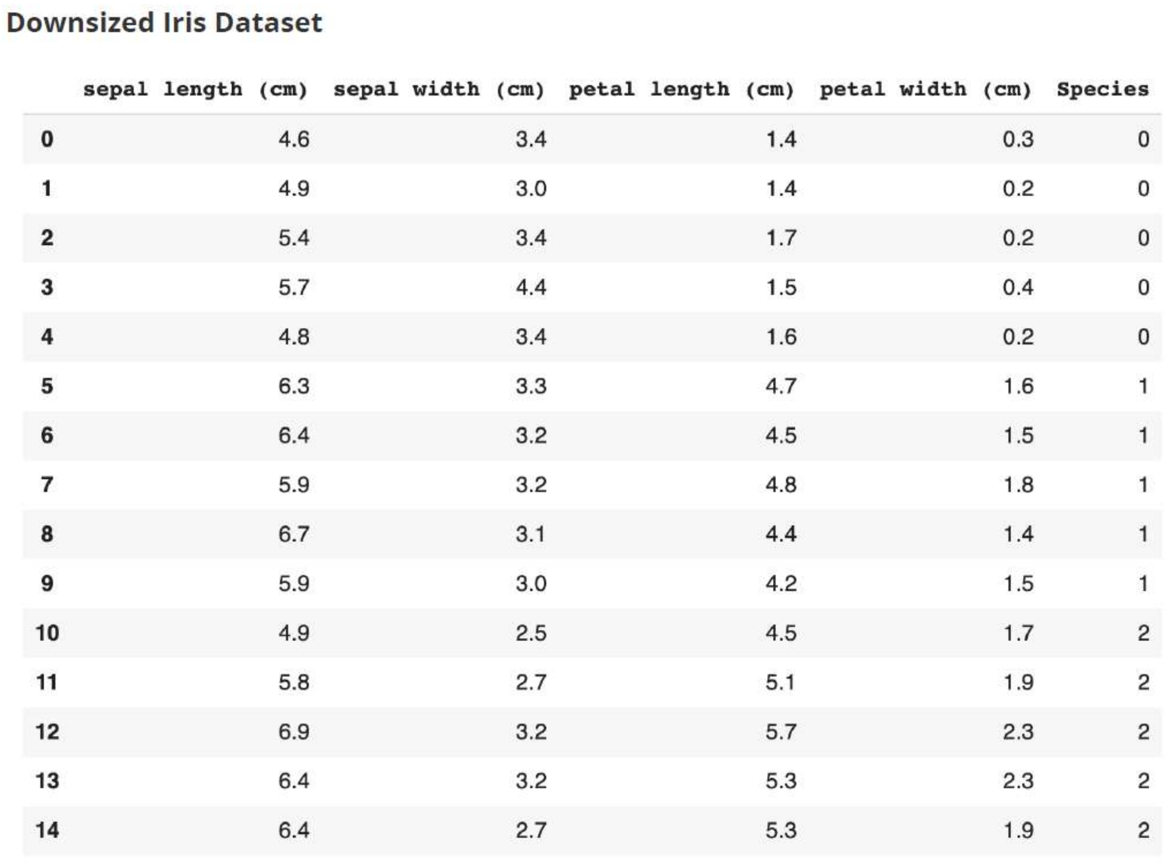
Kamel El-Sehly

Part1

Given the training data in the table (Downsized Iris Dataset), predict the class of following example using Naïve Bayes Classification.

(Species: 'Setosa' : 0, 'Versicolor' : 1, 'Virginica' : 2)

Sepal length=6.9, Sepal width=3.1, Petal length=5.4, Petal width=2.1



Downsized Iris Dataset

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	4.6	3.4	1.4	0.3	0
1	4.9	3.0	1.4	0.2	0
2	5.4	3.4	1.7	0.2	0
3	5.7	4.4	1.5	0.4	0
4	4.8	3.4	1.6	0.2	0
5	6.3	3.3	4.7	1.6	1
6	6.4	3.2	4.5	1.5	1
7	5.9	3.2	4.8	1.8	1
8	6.7	3.1	4.4	1.4	1
9	5.9	3.0	4.2	1.5	1
10	4.9	2.5	4.5	1.7	2
11	5.8	2.7	5.1	1.9	2
12	6.9	3.2	5.7	2.3	2
13	6.4	3.2	5.3	2.3	2
14	6.4	2.7	5.3	1.9	2

Solution:

Step 1 calculation the mean and the standard deviation for each class

Mean μ for each feature per class (cm)

Class / Feature	**Sepal length**	**Sepal width**	**Petal length**	**Petal width**
Setosa	5.08	3.25	1.52	0.26
Versicolor	6.24	3.16	4.52	1.56
Virginica	6.08	2.86	5.18	2.02

Standard deviation σ for each feature per class (cm)

Class / Feature	**Sepal length**	**Sepal width**	**Petal length**	**Petal width**
Setosa	0.4	0.466	0.116	0.08
Versicolor	0.3	0.1	0.213	0.135
Virginica	0.685	0.287	0.391	0.24

Standard deviation equation (1.1)

$$\sigma = \sqrt{\frac{1}{N}\sum_{\mathrm{i}=1}^{N}(\mathcal{X}_{\mathrm{i}} - \mu)^2} \qquad (1.1)$$

Step2: Calculating the likelihood of each feature given the class using the probability distribution function PDF (1.2)

$$\rho(\mathcal{X}|\theta) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(\chi-\mu)^2}{2\sigma^2}} \qquad (1.2)$$

$\rho(\text{Sepal length} = 6.9|0) = 0.0004$

$\rho(\text{Sepal length} = 6.9|1) = 0.129$

$\rho(\text{Sepal length} = 6.9|2) = 0.284$

$\rho(\text{Sepal width} = 3.1|0) = 0.570$

$\rho(\text{Sepal width} = 3.1|1) = 3.290$

$\rho(\text{Sepal width} = 3.1|2) = 0.979$

$\rho(\text{Petal length} = 3.1|0) = 0$

$\rho(\text{Petal length} = 3.1|1) = 0.0003$

$\rho(\text{Petal length} = 3.1|2) = 0.869$

$\rho(\text{Petal width} = 2.1|0)$ $=0.1$

$\rho(\text{Petal width} = 2.1|1) = 0.001$

$\rho(\text{Petal width} = 2.1|2) = 1.57$

Posterior probability=$\frac{Likelihood*prior\ probability}{evidence}$

P(Class=0)=P(Class=1)=P(Class=2)= $\frac{1}{3}$

Since the prior probability for each class are equal and equal to 1/3, and the evidence is the same, Therefore the maximum likelihood corresponds to the maximum posterior. So, we will only compute the likelihood P(features|class) for each class

$\rho(Features|class\ 0)$ =0.0004*0.57*0*0.1=0

$\rho(Features|class\ 1)$ =.129*3.29*.0003*.001

= 1.7e-7

$\rho(Features|class\ 2) =$

=0.284*0.979*0.869*1.57

= 0.38

If we were to calculate the evidence:

$evidence =$

$$[P(class0) * P(\text{Sepal length} = 6.9|0) * P(\text{Sepal width} = 3.1|0) * P(\text{Petal length} = 3.1|0) * P(\text{Petal width} = 2.1|0)] +$$

$$[P(class1) * P(\text{Sepal length} = 6.9|1) * P(\text{Sepal width} = 3.1|1) * P(\text{Petal length} = 3.1|1) * (\text{Petal width} = 2.1|0)] +$$

$$[P(class2) * P(\text{Sepal length} = 6.9|2) * P(\text{Sepal width} = 3.1|2) * P(\text{Petal length} = 3.1|2) * P(\text{Petal width} = 2.1|2)] = 0 + 4.24e - 8 + 0.126 = \mathbf{0.127}$$

Hence

P$((Class\ 0\ |feature\ set) = \frac{0*\frac{1}{3}}{0.127} = 0$

P$((Class\ 1\ |feature\ set) = \frac{1.7\text{e}{-}7*\frac{1}{3}}{0.127} = 4.48*10^{-7}$

P$((Class\ 2\ |feature\ set) = \frac{0.38*\frac{1}{3}}{0.127} = 0.997$

Hence the classifier will predict the class number 2 (Virginica)

Part2

2.1- Load the Iris dataset

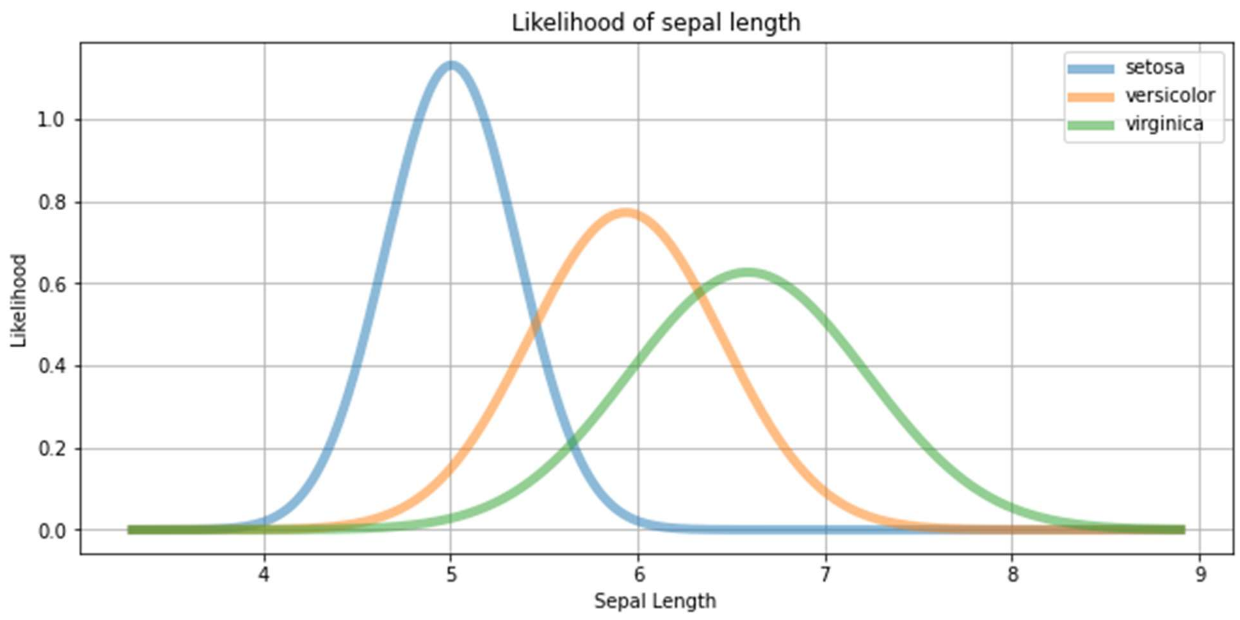
```
iris = load_iris()


df = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
                     columns= iris['feature_names'] + ['target'])
labels=iris.target_names
```

2.2- Drop the petal length and petal width features to form a 2D Iris dataset.

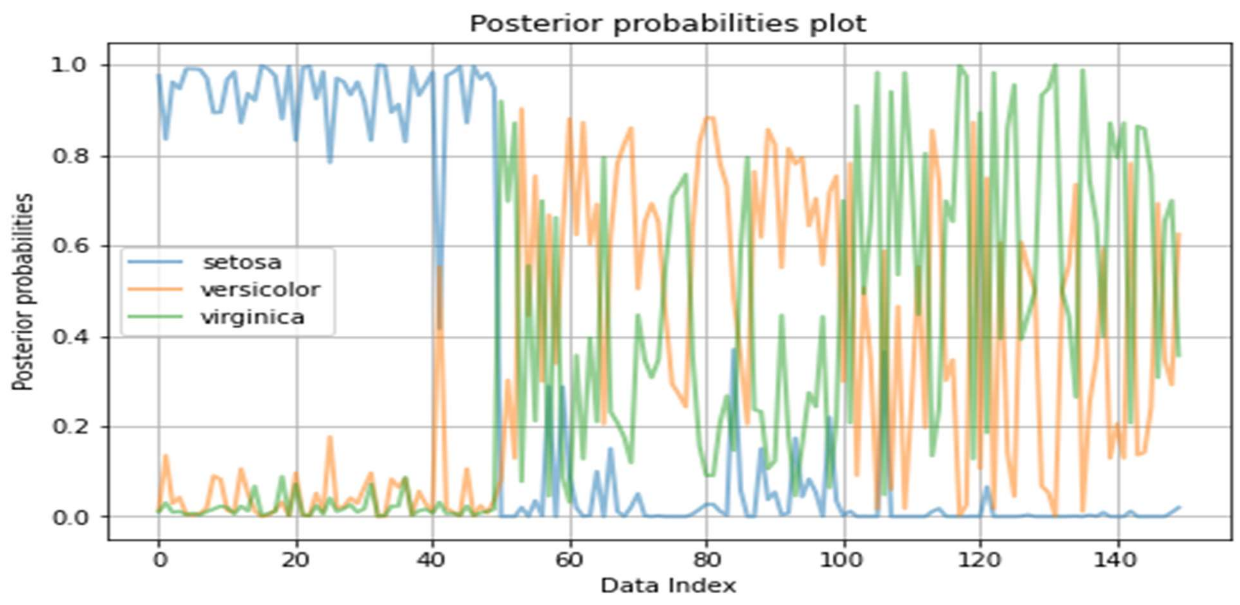
```
Iris2d=df.drop(columns=["petal length (cm)","petal width (cm)"]) #dropping the last two features
```

2.3- Plot the likelihoods of first feature (Sepal length) for each class as given below and apply Naïve Bayes Classifier to 2D Iris dataset to predict classes. Plot posterior probabilities and calculate the accuracy.

Using the plot_likelihood function defined in the appendix B, we plotted the likelihood of the first feature Sepal length for each class.



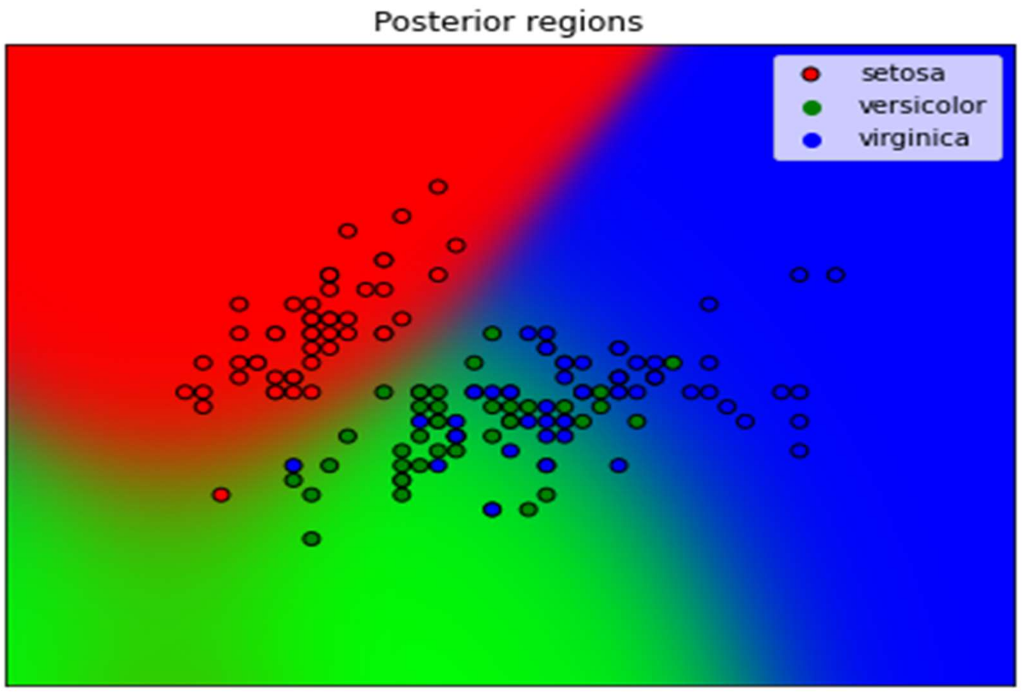
Using the plot_posterior function defined in the appendix C, we plotted the posterior probability for each class V.S the index of the data



The accuracy of this model was **0.78** given the following mean and standard deviation for each class per Sepal length feature.

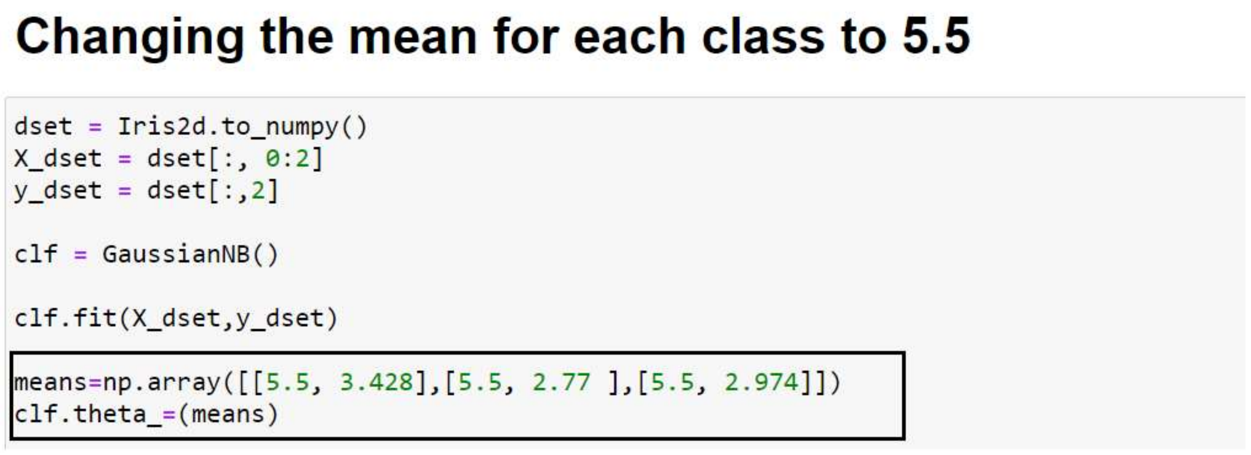
Class / Feature	Sepal length μ	Sepal length σ
Setosa	5.005	0.3524
Versicolor	5.936	0.5161
Virginica	6.587	0.6358

The following figure shows the boundary decision for each class.



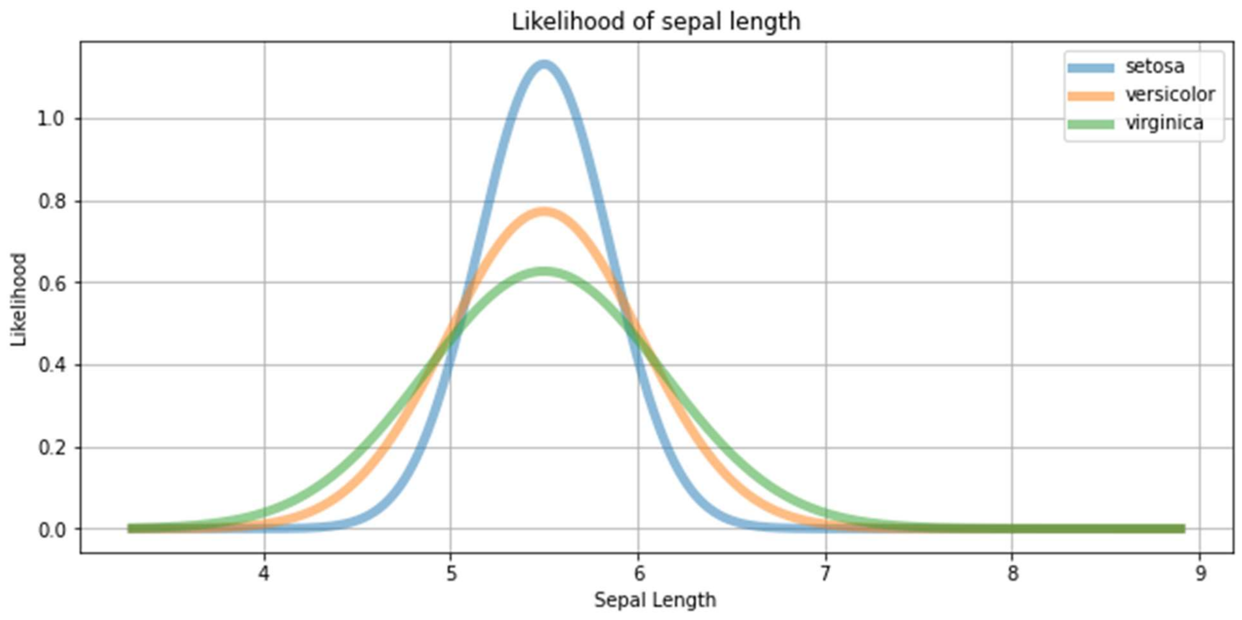
2.4.1 - $\mu_1 = \mu_2 = \mu_3 = 5.5$ and keep the actual values of σ_1, σ_2, and $\sigma_3$for the first feature. Plot the likelihoods and apply Naïve Bayes Classifier. Plot posterior probabilities and calculate the accuracy.

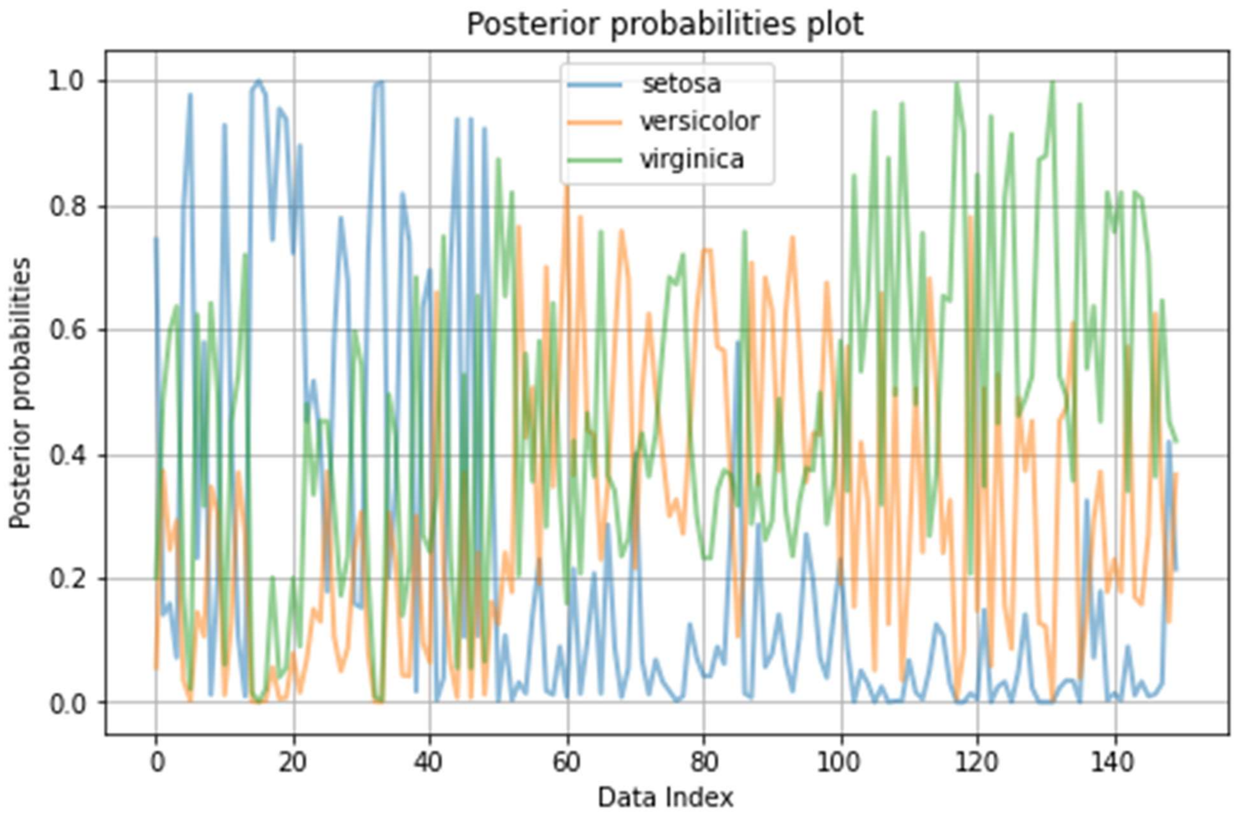
This has been implemented by changing the theta attribute of the scikit learn's GaussianNB() object as in the following code:



Changing the mean for each class to 5.5

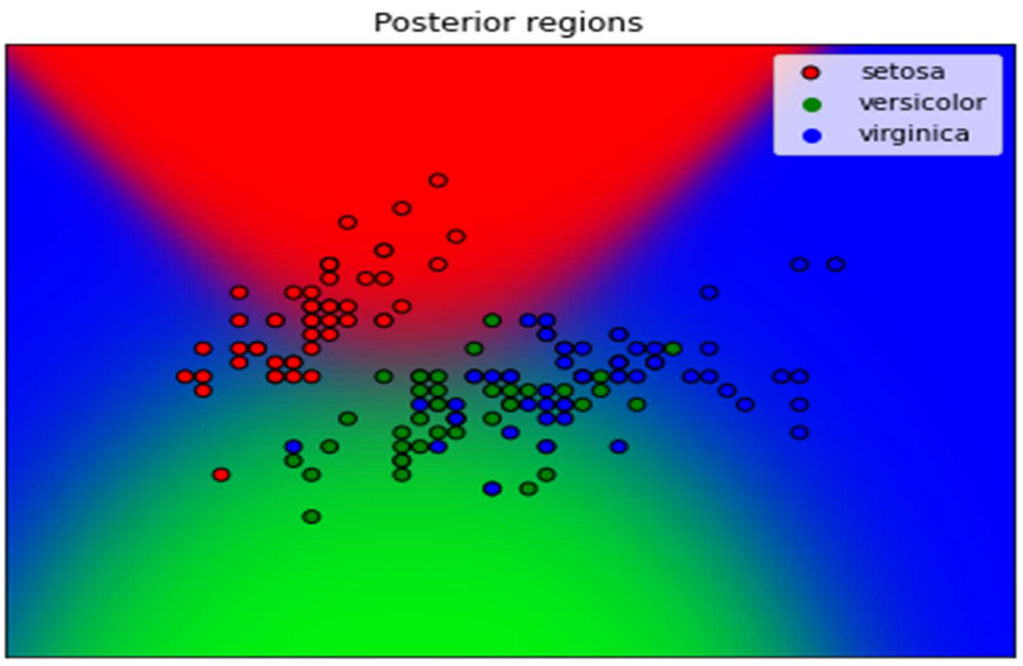
```
dset = Iris2d.to_numpy()
X_dset = dset[:, 0:2]
y_dset = dset[:,2]

clf = GaussianNB()

clf.fit(X_dset,y_dset)

means=np.array([[5.5, 3.428],[5.5, 2.77 ],[5.5, 2.974]])
clf.theta_=(means)
```


Likelihood of sepal length
Likelihood
Sepal Length
setosa
versicolor
virginica
0.0
0.2
0.4
0.6
0.8
1.0
4
5
6
7
8
9


Posterior probabilities plot
Posterior probabilities
Data Index
setosa
versicolor
virginica
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
100
120
140

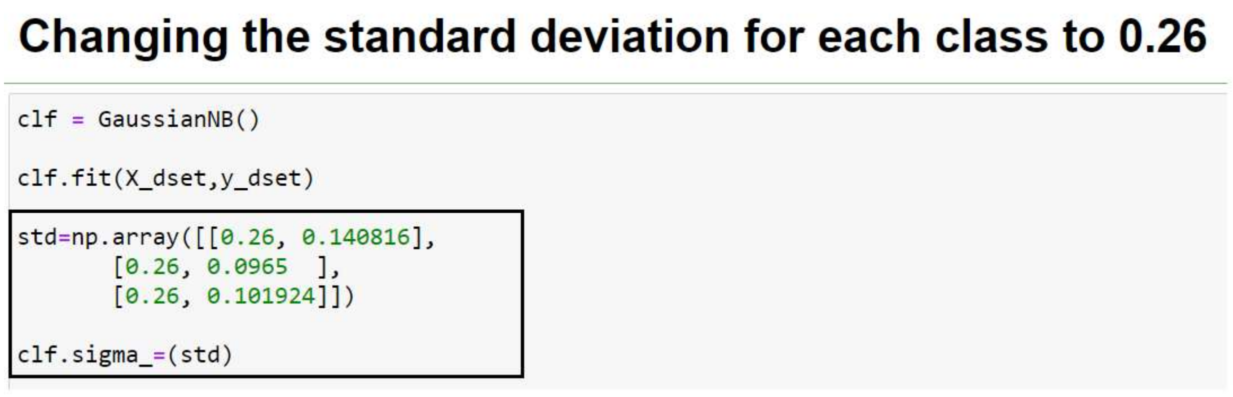
The accuracy of this model was **0.626** given the following mean and standard deviation for each class per Sepal length feature.

Class / Feature	Sepal length μ	Sepal length σ
Setosa	5.5	0.3524
Versicolor	5.5	0.5161
Virginica	5.5	0.6358



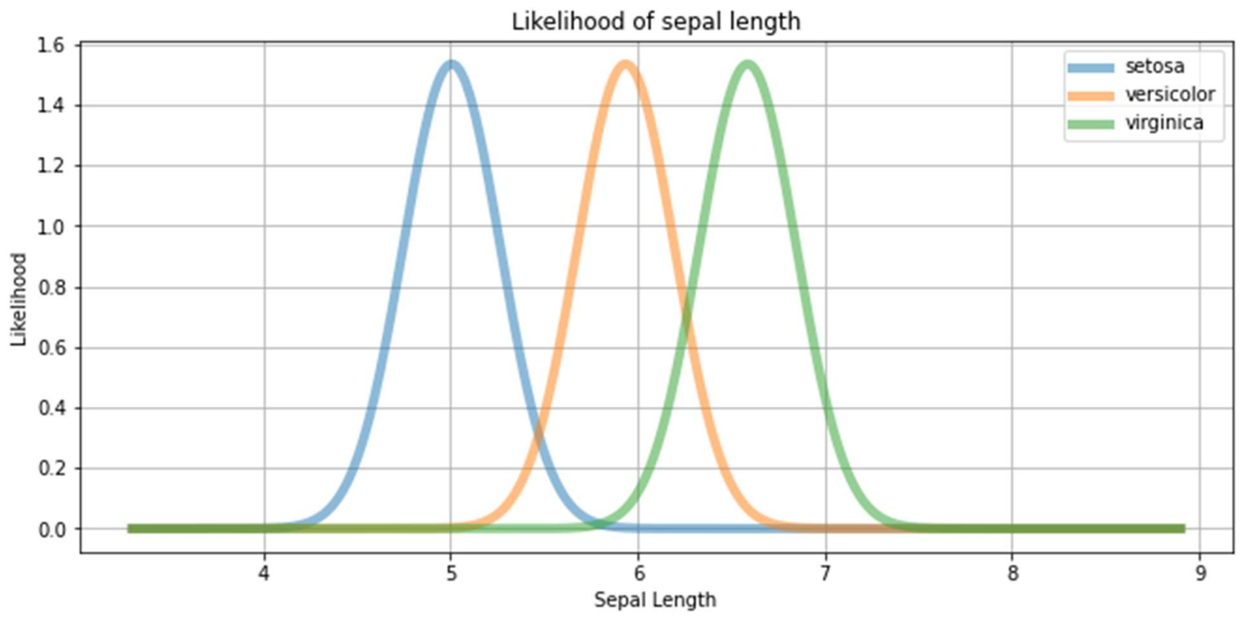
2.4.1 - $\sigma_1 = \sigma_2 = \sigma_2 = 0.26$ and keep the actual values of μ_1, μ_2, and $\mu_3$for the first feature. Plot the likelihoods and apply Naïve Bayes Classifier. Plot posterior probabilities and calculate the accuracy.

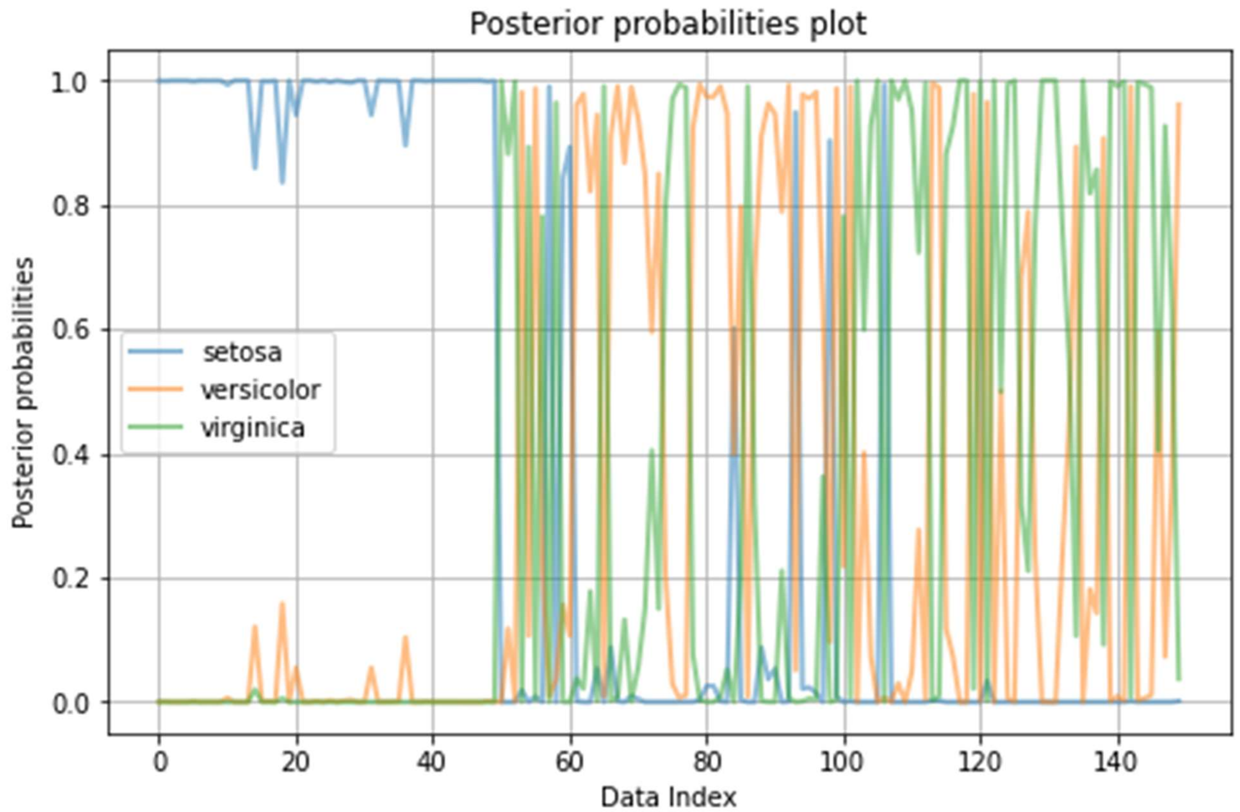
This has been implemented by changing the sigma attribute of the scikit learn's GaussianNB() object as in the following code:



Changing the standard deviation for each class to 0.26

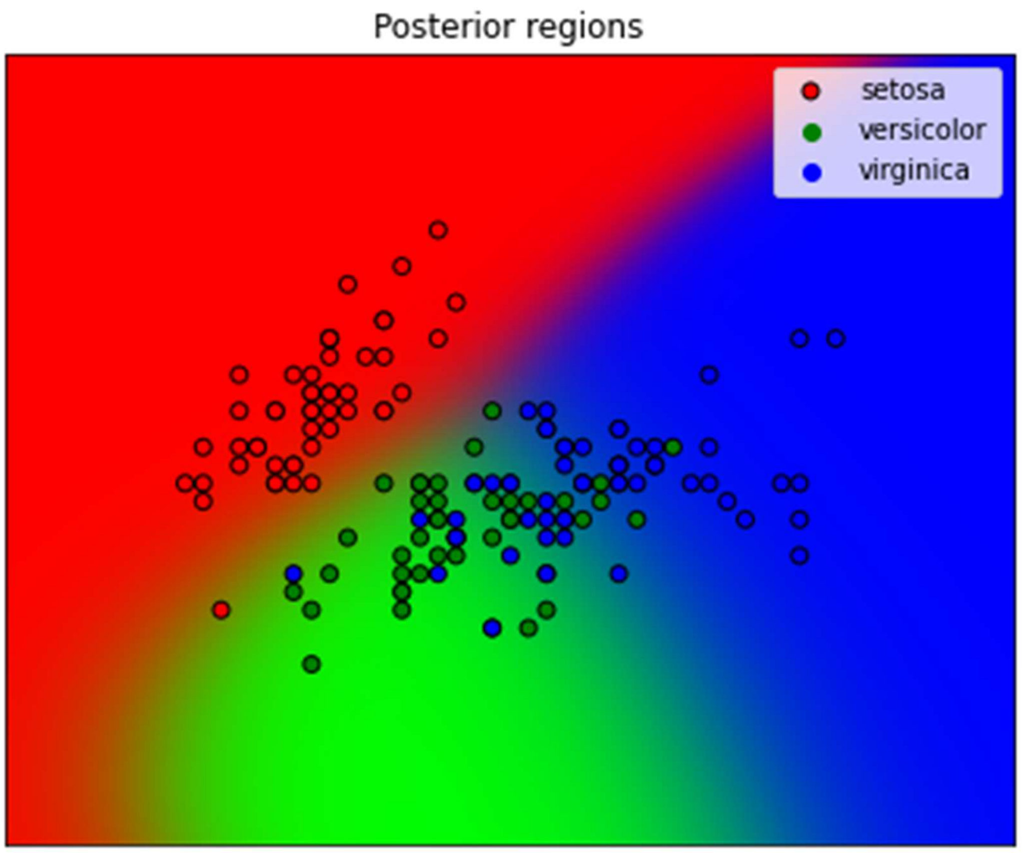
```
clf = GaussianNB()

clf.fit(X_dset,y_dset)

std=np.array([[0.26, 0.140816],
       [0.26, 0.0965  ],
       [0.26, 0.101924]])

clf.sigma_=(std)
```


Likelihood of sepal length
Likelihood
Sepal Length
setosa
versicolor
virginica


Posterior probabilities plot
Posterior probabilities
Data Index
setosa
versicolor
virginica

The accuracy of this model was **0.8** given the following mean and standard deviation for each class per Sepal length feature.

Class / Feature	Sepal length μ	Sepal length σ
Setosa	5.005	0.26
Versicolor	5.936	0.26
Virginica	6.587	0.26



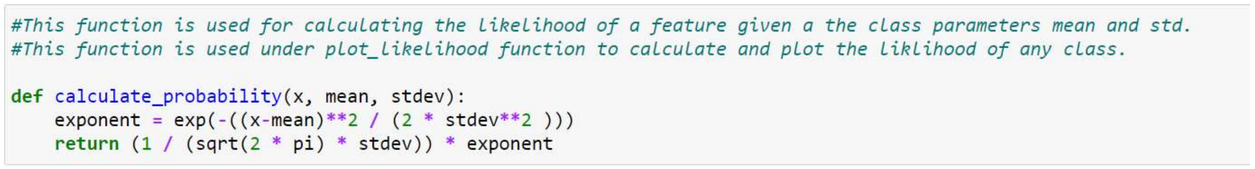
2.4.3 Compare the accuracy values and make a comment based on it.

The best accuracy obtained when the variance was set to 0.26 which was 80%. And the lowest accuracy was 62.7% obtained when all classes had the same mean 5.5 with different variance. Therefore, we can conclude that the accuracy of the naive bayes model is affected by difference of the features' means between each class, and the variance of the points of each class (how condense they are centered around the mean).

The higher the differences between the classes means along with the lower the variance of each class, the higher the accuracy of the model.

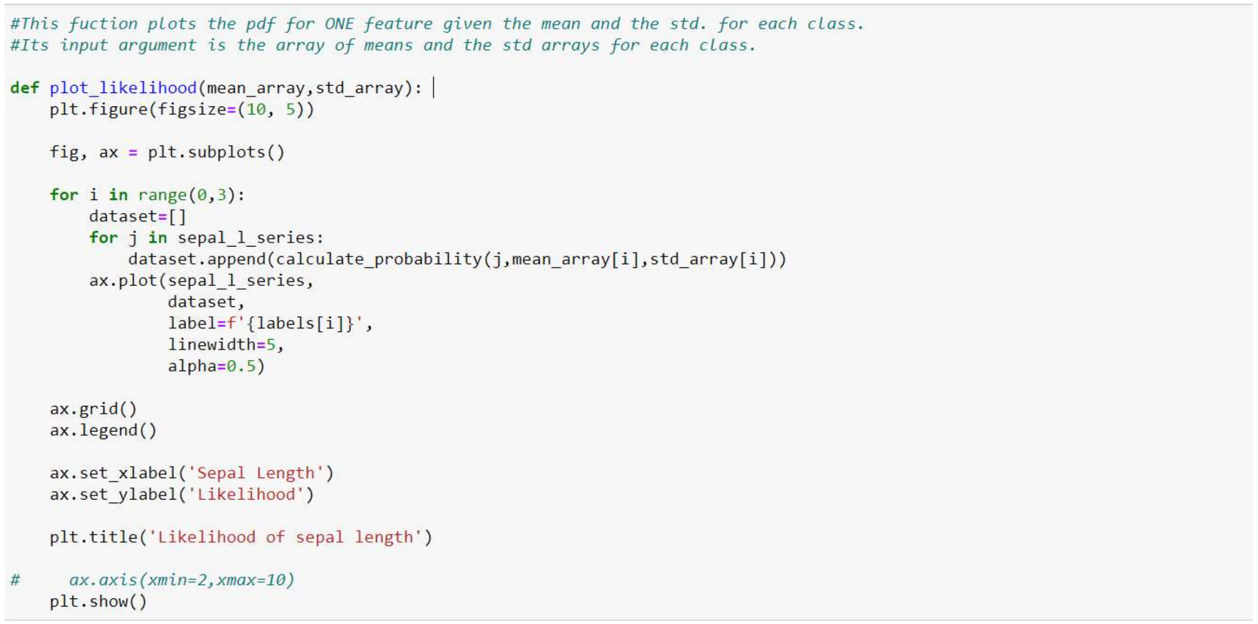
Appendix

A) Calculating the likelihood function.



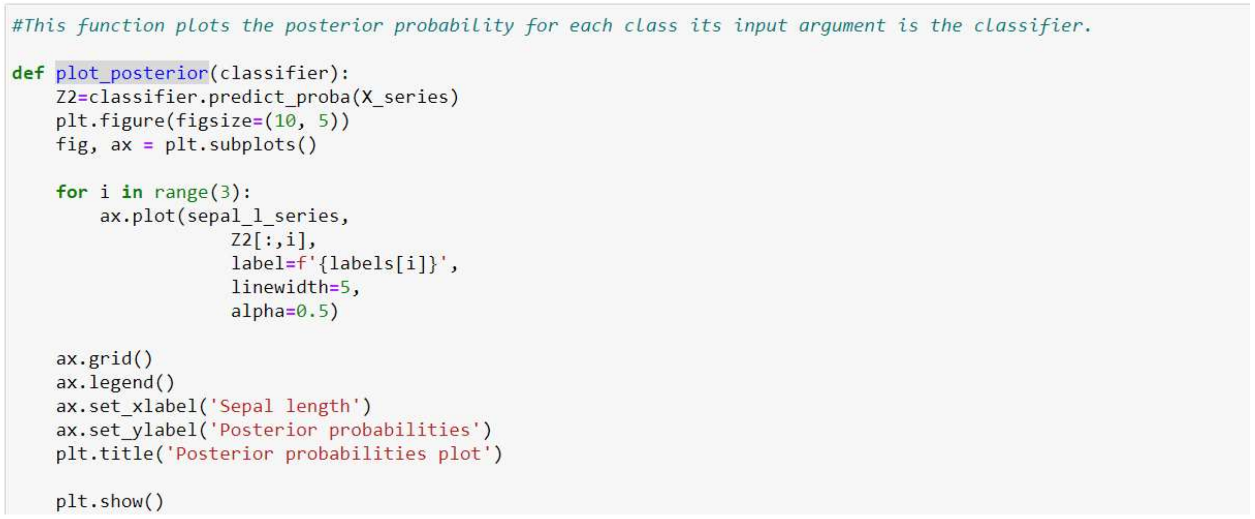
```
#This function is used for calculating the likelihood of a feature given a the class parameters mean and std.
#This function is used under plot_likelihood function to calculate and plot the liklihood of any class.

def calculate_probability(x, mean, stdev):
    exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
    return (1 / (sqrt(2 * pi) * stdev)) * exponent
```

B) Plotting the likelihood function.



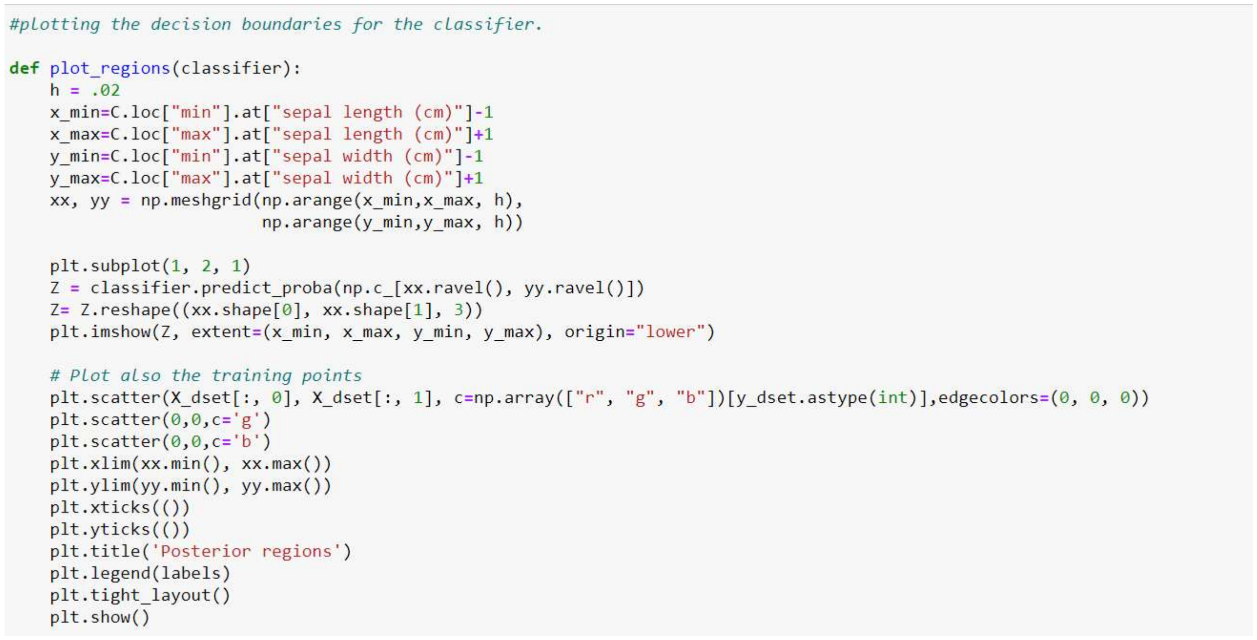
```
#This fuction plots the pdf for ONE feature given the mean and the std. for each class.
#Its input argument is the array of means and the std arrays for each class.

def plot_likelihood(mean_array,std_array):
    plt.figure(figsize=(10, 5))

    fig, ax = plt.subplots()

    for i in range(0,3):
        dataset=[]
        for j in sepal_l_series:
            dataset.append(calculate_probability(j,mean_array[i],std_array[i]))
        ax.plot(sepal_l_series,
                dataset,
                label=f'{labels[i]}',
                linewidth=5,
                alpha=0.5)

    ax.grid()
    ax.legend()

    ax.set_xlabel('Sepal Length')
    ax.set_ylabel('Likelihood')

    plt.title('Likelihood of sepal length')

#     ax.axis(xmin=2,xmax=10)
    plt.show()
```

C) Plotting the posterior probability function.



```
#This function plots the posterior probability for each class its input argument is the classifier.

def plot_posterior(classifier):
    Z2=classifier.predict_proba(X_series)
    plt.figure(figsize=(10, 5))
    fig, ax = plt.subplots()

    for i in range(3):
        ax.plot(sepal_l_series,
                    Z2[:,i],
                    label=f'{labels[i]}',
                    linewidth=5,
                    alpha=0.5)

    ax.grid()
    ax.legend()
    ax.set_xlabel('Sepal length')
    ax.set_ylabel('Posterior probabilities')
    plt.title('Posterior probabilities plot')

    plt.show()
```

D)Plotting the decision boundary function.



```
#plotting the decision boundaries for the classifier.

def plot_regions(classifier):
    h = .02
    x_min=C.loc["min"].at["sepal length (cm)"]-1
    x_max=C.loc["max"].at["sepal length (cm)"]+1
    y_min=C.loc["min"].at["sepal width (cm)"]-1
    y_max=C.loc["max"].at["sepal width (cm)"]+1
    xx, yy = np.meshgrid(np.arange(x_min,x_max, h),
                         np.arange(y_min,y_max, h))

    plt.subplot(1, 2, 1)
    Z = classifier.predict_proba(np.c_[xx.ravel(), yy.ravel()])
    Z= Z.reshape((xx.shape[0], xx.shape[1], 3))
    plt.imshow(Z, extent=(x_min, x_max, y_min, y_max), origin="lower")

    # Plot also the training points
    plt.scatter(X_dset[:, 0], X_dset[:, 1], c=np.array(["r", "g", "b"])[y_dset.astype(int)],edgecolors=(0, 0, 0))
    plt.scatter(0,0,c='g')
    plt.scatter(0,0,c='b')
    plt.xlim(xx.min(), xx.max())
    plt.ylim(yy.min(), yy.max())
    plt.xticks(())
    plt.yticks(())
    plt.title('Posterior regions')
    plt.legend(labels)
    plt.tight_layout()
    plt.show()
```