Parametric Methods

Assignment 2_G4

Ahmed Yousry **Bassel Hamshary**
Mohamed El-Namoury

uOttawa
Faculté de génie
Faculty of Engineering

JUNE 10, 2021
UNIVERSITY OF OTTAWA
Ottawa, Canada

Table of Contents

Table of Figures

1. Objectives

This assignment aims to understand Naïve Bayes Classification and how mean and variance values affect posterior probabilities. the well-known Iris flower dataset is used.
Iris dataset contains 150 samples, 4 features (i.e., sepal length, sepal width, petal length, petal width), and 3 classes (i.e., Iris-Setosa, Iris-Versicolor, Iris-Virginica).

2. Implementation

2.1. Part One

Given the training data in the table (Downsized Iris Dataset), predict the class of the 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

Table 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

2.1.1. Calculating the Mean & Standard Deviation

$$\mu = \frac{\sum_{i=1}^{n} Xi}{n} \qquad \sigma = \sqrt{\frac{\sum_{i=1}^{n}(Xi - \mu)^2}{n}}$$

- Apply the mean and the standard deviation for each column in the table above.

Sepal Length	Sepal Width	Petal Length	Petal Width
Mean (SL_0) = **5.08** Mean (SL_1) = **6.24** Mean (SL_2) = **6.08**	Mean (SW_0) = **3.52** Mean (SW_1) = **3.16** Mean (SW_2) = **2.86**	Mean (PL_0) = **1.52** Mean (PL_1) = **4.52** Mean (PL_2) = **5.18**	Mean (PW_0) = **0.26** Mean (PW_1) = **1.56** Mean (PW_2) = **2.02**
St. Dev (SL_0) = **0.406939799** St. Dev (SL_1) = **0.30724583** St. Dev (SL_2) = **0.685273668**	St. Dev (SW_0) = **0.466476152** St. Dev (SW_1) = **0.10198039** St. Dev (SW_2) = **0.287054002**	St. Dev (PL_0) = **0.116619038** St. Dev (PL_1) = **0.213541565** St. Dev (PL_2) = **0.391918359**	St. Dev (PW_0) = **0.08** St. Dev (PW_1) = **0.1356466** St. Dev (PW_2) = **0.24**

- Using the probability density function for the normal distribution to calculate the following:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

f(SL = 6.9 | class 0)	$4.4454 * 10^{-5}$	f(PL = 5.4 | class 0)	$1.4619 * 10^{-240} \approx 0$
f(SL = 6.9 | class 1)	0.129246374	f(PL = 5.4 | class 1)	0.000383472
f(SL = 6.9 | class 2)	0.284526189	f(PL = 5.4 | class 2)	0.869541787
f(SW = 3.1 | class 0)	0.570226839	f(PW = 2.1 | class 0)	$6.7132 * 10^{-115} \approx 0$
f(SW = 3.1 | class 1)	3.290235952	f(PW = 2.1 | class 1)	0.001064607
f(SW = 3.1 | class 2)	0.979837855	f(PW = 2.1 | class 2)	1.572430115

- To which class does the example belong?

$$P(\ Class \mid Evidence)$$

$$= \frac{\text{f(SL } = \text{ 6.9 | class)} * \text{ f(SW } = \text{ 3.1 | class)} * \text{ f(PL } = \text{ 5.4 | class)} * \text{ f(PW } = \text{ 2.1 | class)} * P(Class)}{P(Evidence)}$$

P(Class 0 | Evidence)	**0**
P(Class 1 | Evidence)	**5.78692 * 10^{-8}**
P(Class 2 | Evidence)	**0.127062389**

Since the Posterior Probability of class 2 is higher, therefore, the Naïve Bayes classifier predicts Class = 'Virginica' (2) for the example.

Mean				
class_0	5.08	3.52	1.52	0.26
class_1	6.24	3.16	4.52	1.56
class_2	6.08	2.86	5.18	2.02

ST_Dev				
class_0	0.406939799	0.46647615	0.116619038	0.08
class_1	0.30724583	0.10198039	0.213541565	0.1356466
class_2	0.685273668	0.287054	0.391918359	0.24

PDF	
f(SL = 6.9 | class 0)	4.4454E-05
f(SL = 6.9 | class 1)	0.129246374
f(SL = 6.9 | class 2)	0.284526189
f(SW = 3.1 | class 0)	0.570226839
f(SW = 3.1 | class 1)	3.290235952
f(SW = 3.1 | class 2)	0.979837855
f(PL = 5.4 | class 0)	1.4619E-240
f(PL = 5.4 | class 1)	0.000383472
f(PL = 5.4 | class 2)	0.869541787
f(PW = 2.1 | class 0)	6.7132E-115
f(PW = 2.1 | class 1)	0.001064607
f(PW = 2.1 | class 2)	1.572430115

Posterior Probability	
P(Class 0 | Evidence)	0
P(Class 1 | Evidence)	5.7869E-08
P(Class 2 | Evidence)	0.12706239

Figure 1: Part one Calculations on Excel

2.2. Part Two

2.2.1. Loading the iris dataset

```
iris = load_iris()
```

```
df_data = pd.DataFrame(iris.data)
```

```
df_data
```

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

2.2.2. Drop the Petal length and Petal width Features

```
df_data.drop(columns=[2,3], inplace= True)
df_data.head()
```

	0	1
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6

```
df_target = pd.DataFrame(iris.target)
```

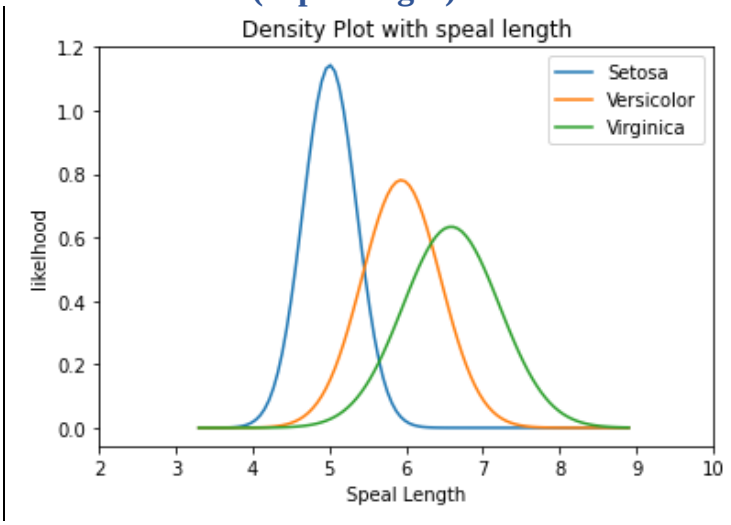
```
df_data.rename(columns= {0:"sepal_length", 1:"sepal_width"}, inplace = True)
df_data.head()
```

	sepal_length	sepal_width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6

2.2.3. Training and Plotting of the Data

2.2.3.1. Plotting the likelihood for the first feature (Sepal length)

```
classes = ['Setosa' , 'Versicolor' , 'Virginica']

plt.plot(x_values,class_0_likelhood2)
plt.plot(x_values,class_1_likelhood2)
plt.plot(x_values,class_2_likelhood2)

# Plot formatting
plt.legend(['Setosa' , 'Versicolor' , 'Virginica'])
plt.title('Density Plot with speal length')
plt.xlabel('Speal Length')
plt.xlim(2, 10)
plt.ylabel("likelhood")
```



2.2.3.2. Applying the Naive Bayes Classifier to the iris dataset.

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X_train = df_data
y_train = df_iris["y"]
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_train)
predictions  = gnb.predict_proba(X_train)
predictions_0 = predictions[:,0]
predictions_1 = predictions[:,1]
predictions_2 = predictions[:,2]
```

```
df_iris["posterior_0"] = predictions_0
df_iris["posterior_1"] = predictions_1
df_iris["posterior_2"] = predictions_2
```

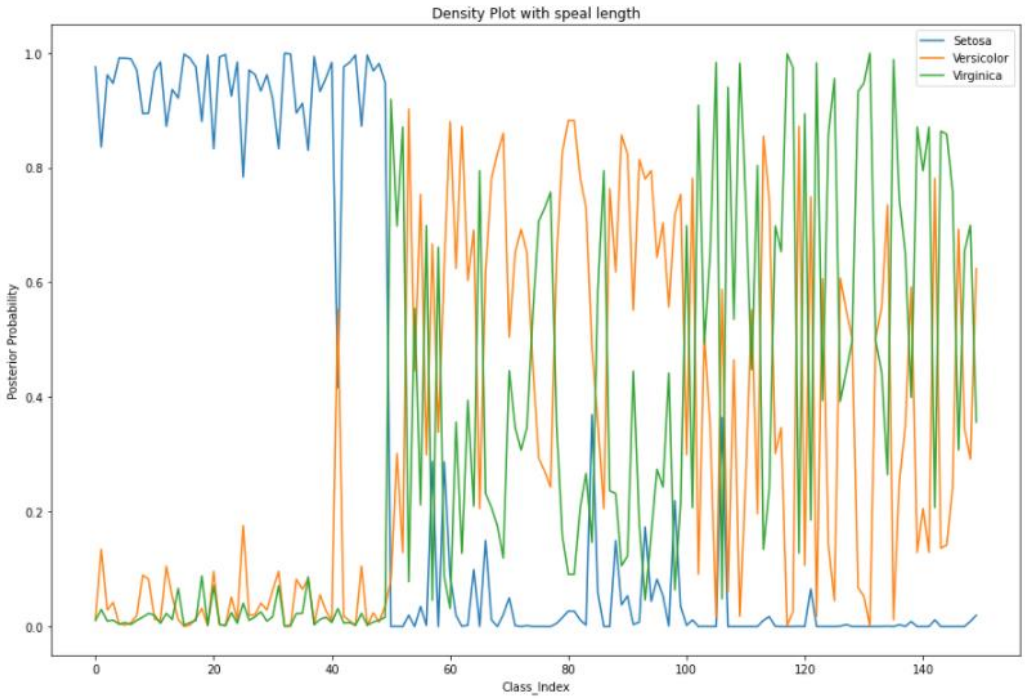
```
predictions
```

```
       [1.98063263e-02, 9.02292246e-01, 7.79014281e-02],
       [2.57135077e-05, 4.45309879e-01, 5.54664407e-01],
       [3.48160348e-02, 7.53248039e-01, 2.11935926e-01],
       [1.95340303e-03, 2.99425864e-01, 6.98620733e-01],
       [2.87504771e-01, 6.66902603e-01, 4.55926262e-02],
       [1.14040968e-05, 3.38739601e-01, 6.61248995e-01],
       [2.86700742e-01, 6.26185632e-01, 8.71136267e-02],
       [8.83434558e-02, 8.79909495e-01, 3.17470491e-02],
       [1.95472527e-02, 6.23994551e-01, 3.56458196e-01],
       [4.68859336e-04, 8.71701649e-01, 1.27829492e-01],
       [2.29138881e-03, 6.03102638e-01, 3.94605973e-01],
       [9.90380254e-02, 6.90922232e-01, 2.10039742e-01],
       [6.97245509e-06, 2.05481396e-01, 7.94511631e-01],
       [1.49943267e-01, 6.18193707e-01, 2.31863026e-01],
       [1.15137845e-02, 7.81229918e-01, 2.07256297e-01],
       [8.33769189e-05, 8.24530487e-01, 1.75386136e-01],
       [2.08333262e-02, 8.59945232e-01, 1.19221442e-01],
       [4.98812330e-02, 5.04293156e-01, 4.45825611e-01],
```

2.2.3.3. Plotting the Posterior Probabilities

```
In [29]: plt.figure(figsize=(15,10))

         plt.plot(predictions_0)
         plt.plot(predictions_1)
         plt.plot(predictions_2)
         # Plot formatting
         plt.legend(['Setosa' , 'Versicolor' , 'Virginica'])
         plt.title('Density Plot with speal length')
         plt.xlabel('Class_Index')
         plt.ylabel("Posterior Probability")
```

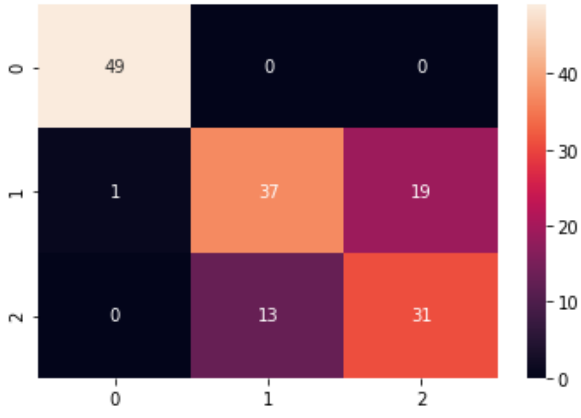
```
Out[29]: Text(0, 0.5, 'Posterior Probability')
```



2.2.3.4. Calculating the Accuracy

```
from sklearn.metrics import classification_report , confusion_matrix

print(classification_report(y_pred , df_iris["y"]))
print(confusion_matrix(y_pred , df_iris["y"]))

sns.heatmap(confusion_matrix(y_pred , df_target), annot=True)
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        49
           1       0.74      0.65      0.69        57
           2       0.62      0.70      0.66        44

    accuracy                           0.78       150
   macro avg       0.78      0.78      0.78       150
weighted avg       0.78      0.78      0.78       150

[[49  0  0]
 [ 1 37 19]
 [ 0 13 31]]
```

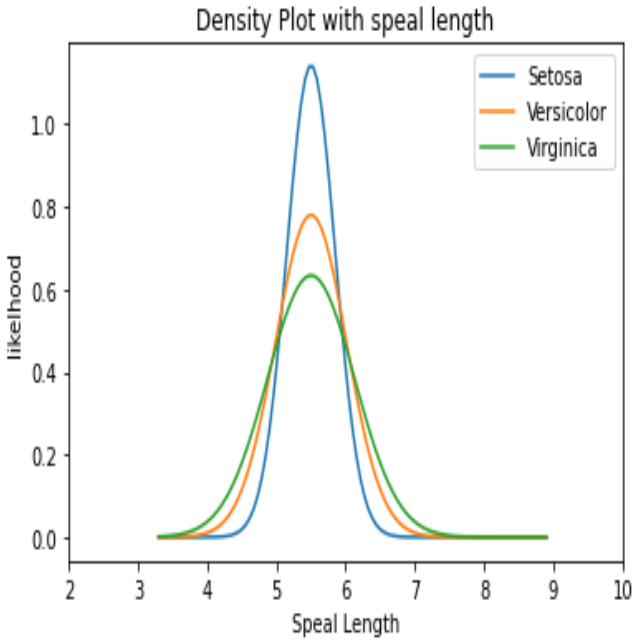


2.2.4. Training and Plotting of the Data after Changing Mean St.Dev

2.2.4.1. Plotting the likelihoods and changing the mean and applying the naive Bayes classifier to the new mean. Then calculating the accuracy

```
class_0_likelihood2_4 = likelyhood2(x_values,5.5, var_0)
class_1_likelihood2_4 = likelyhood2(x_values,5.5, var_1)
class_2_likelihood2_4 = likelyhood2(x_values,5.5, var_2)
```

```
classes = ['Setosa' , 'Versicolor' , 'Virginica']

plt.plot(x_values,class_0_likelihood2_4)
plt.plot(x_values,class_1_likelihood2_4)
plt.plot(x_values,class_2_likelihood2_4)

# Plot formatting
plt.legend(['Setosa' , 'Versicolor' , 'Virginica'])
plt.title('Density Plot with speal length')
plt.xlabel('Speal Length')
plt.xlim(2, 10)
plt.ylabel("likelihood")
```



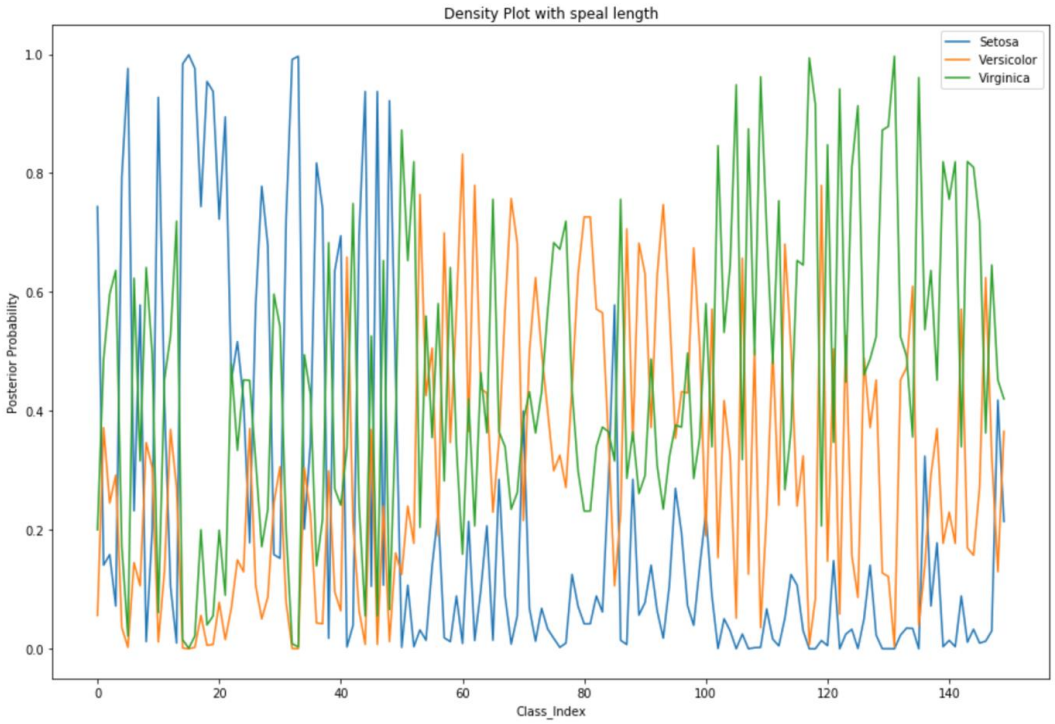


Figure 2: The posterior probabilities for new Mean

```
gnb2 = GaussianNB()
gnb2.fit(X_train, y_train)
y_pred = gnb2.predict(X_train)
gnb2.theta_[:,0] = [5.5 ,5.5 ,5.5]
y_pred2 = gnb2.predict(X_train)
predictions3  = gnb2.predict_proba(X_train)
#gnb.fit(X_train, y_train)
print(classification_report(y_pred2 , df_iris["y"]))
```

```
              precision    recall  f1-score   support

           0       0.58      0.94      0.72        31
           1       0.56      0.67      0.61        42
           2       0.74      0.48      0.58        77

    accuracy                           0.63       150
   macro avg       0.63      0.69      0.64       150
weighted avg       0.66      0.63      0.62       150
```

```
sns.heatmap(confusion_matrix(y_pred2 , df_target), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb16b425190>
```

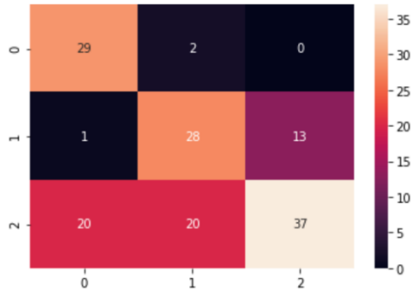
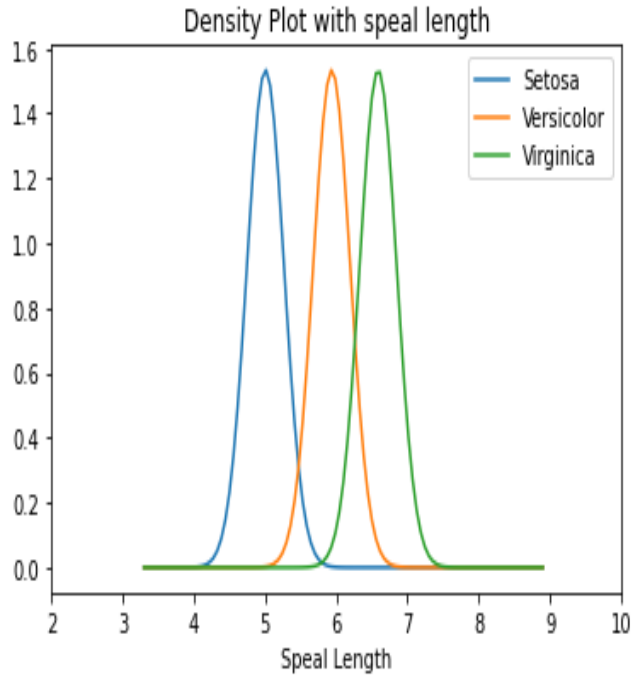


Figure 3: Naive Bayes Classifier on the new Mean

2.2.4.2. Plotting the likelihoods and changing the variance and applying the naive Bayes classifier to the new variance. Then calculating the accuracy

```
class_0_likelhood2_5 = likelyhood2(x_values,mean_0, 0.26)
class_1_likelhood2_5 = likelyhood2(x_values,mean_1, 0.26)
class_2_likelhood2_5 = likelyhood2(x_values,mean_2, 0.26)
```

```
classes = ['Setosa' , 'Versicolor' , 'Virginica']

plt.plot(x_values,class_0_likelhood2_5)
plt.plot(x_values,class_1_likelhood2_5)
plt.plot(x_values,class_2_likelhood2_5)

# Plot formatting

plt.legend(['Setosa' , 'Versicolor' , 'Virginica'])
plt.title('Density Plot with speal length')
plt.xlabel('Speal Length')
plt.xlim(2, 10)
```



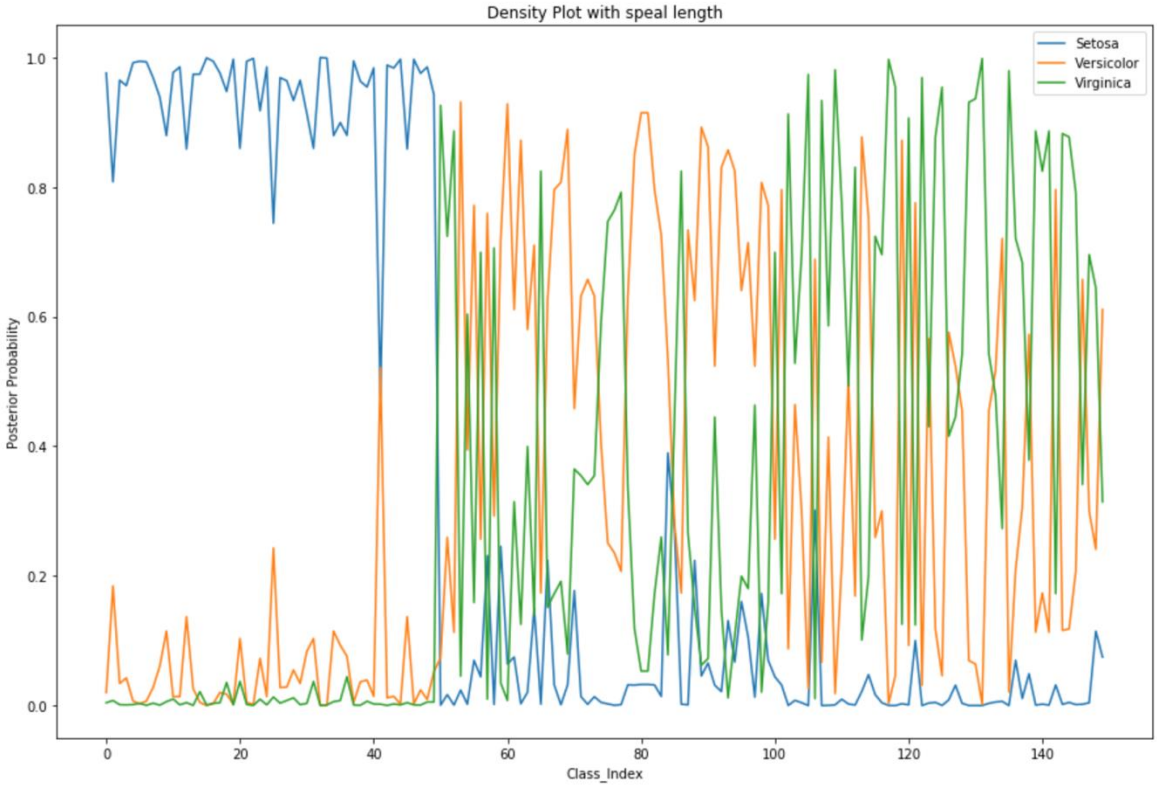


Figure 4: The posterior probabilities for new variance

```
gnb1 = GaussianNB()
gnb1.fit(X_train, y_train)
y_pred = gnb1.predict(X_train)
gnb1.sigma_[:,0] = [0.26,0.26,0.26]
y_pred1 = gnb1.predict(X_train)
print(classification_report(y_pred1 , df_iris["y"]))
predictions2  = gnb1.predict_proba(X_train)
predictions_01 = predictions2[:,0]
predictions_11 = predictions2[:,1]
predictions_21 = predictions2[:,2]
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99        49
           1       0.74      0.69      0.71        54
           2       0.68      0.72      0.70        47

    accuracy                           0.80       150
   macro avg       0.80      0.80      0.80       150
weighted avg       0.80      0.80      0.80       150
```

```
sns.heatmap(confusion_matrix(y_pred1, df_target), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8ea3a2dc90>
```

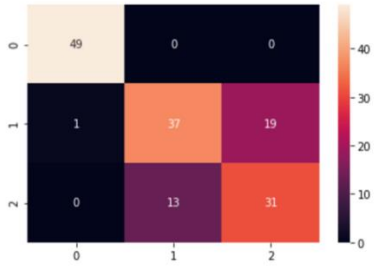


Figure 5: Naive Bayes Classifier on the new Variance

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

1- For the normal classifier with calculated mean and variance for every class the accuracy was 0.78
2- For constant variance 0.26 over all classes, the accuracy was 0.8.
3- For constant mean 5.5 over all classes, the accuracy was 0.63.

$$\mu = \frac{\sum_{i=1}^{n} Xi}{n} \quad \left| \quad \sigma = \sqrt{\frac{\sum_{i=1}^{n}(Xi - \mu)^2}{n}} \right. \quad \left| \quad f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}} \right.$$

Hence from these equations, we can conclude that in the first step when mean and variance changed from one class to another the accuracy was 0.78, and when the variance is constant and how the data distributed for each class it caused the model to get a higher result as it shown in the graph that it made it easier for the model to predict correctly, that lead the accuracy to increase from 0.78 to 0.8, but when the mean was constant and the data was distributed around the same axes with different variance from one class to another the intersection caused the model to be confused when predicting the class that lead the accuracy to fall from 0.78 to 0.63.