ELG5255[EG]: Applied Machine Learning

Assignment 2

Parametric Methods

Group: 25

The report will go as follows: #) for steps to perform the intended tasks including screenshots of the code we used to tackle this question along with any description needed or relevant figures.

Part #1

For part one we used python for some calculations as the mean and variance for each feature per class, the likelihood –after we've hard coded the equation, given the feature value (a.k.a: x), mean and variance we can get the likelihood.
The steps for the solution is provided in the following screenshots along with a glimpse of the basic codes used for assistance in calculation.

Manual steps

class	mean(SL)	mean(SW)	mean(PL)	mean(PW)
0	5.08	3.52	1.52	0.26
1	6.24	3.16	4.52	1.56
2	6.08	2.86	5.18	2.02

class	Var(SL)	Var(SW)	Var(PL)	Var(PW)
0	0.17	0.218	0.0136	0.0064
1	0.09	0.045	0.010	0.00184
2	0.47	0.082	0.154	0.0576

$$p(C=0 \mid SL, SW, PL, PW)$$

$$= \frac{p(0)\,p(SL|0)\,p(SW|0)\,p(PL|0)\,p(PW|0)}{\text{evidance}}$$

$$\text{evidance} = p(0)\,p(SL|0)\,p(SW|0)\,p(PL|0)\,p(PW|0) + p(1)\,p(SL|1)\,p(SW|1)\,p(PL|1)\,p(PW|1) + p(2)\,p(SL|2)\,p(SW|2)\,p(PL|2)\,p(PW|2)$$

$$p(SL|0) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-(x-\mu)^2/2\sigma^2} \simeq 4.4465 \times 10^{-5}$$

$$p(SW|0) \simeq 0.5704 \qquad p(PL|0) \simeq 1.4623 \times 10^{-240}$$

$$p(PW|0) \simeq 6.7149 \times 10^{-115}$$

$p(sL|1) \simeq 0.1293$ $\quad p(sw|1) \simeq 3.2911$

$p(pL|1) \simeq 0.00038$ $\quad p(pw|1) \simeq 0.00106$

$p(sL|2) \simeq 0.2846$ $\quad p(sw|2) \simeq 0.98009$

$p(pL|2) \simeq 0.8697$ $\quad p(pw|2) \simeq 1.5728$

$p(0) = p(1) = p(2) = \frac{5}{15} = \frac{1}{3}$

$p(0|6.9, 3.1, 5.4, 2.1)$

$$= \frac{\frac{1}{3} * 4.4465 \times 10^{-5} * 0.5704 * 1.4623 \times 10^{-240} * 6.7149 \times 10^{-155}}{\frac{1}{3} * 4.4465 \times 10^{-5} * 0.5704 * 1.4623 \times 10^{-240} * 6.7149 \times 10^{-155} + \frac{1}{3} * 0.1293 * 3.2911 * 0.00038 * 0.00106 + \frac{1}{3} * 0.2846 * 0.98009 * 0.8697 * 1.5728}$$

$$= \boxed{0.0}$$

$p(1|6.9, 3.1, 5.4, 2.1)$

$$= \frac{\frac{1}{3} * 0.1293 * 3.2911 * 0.00038 * 0.00106}{0.12719}$$

$$\simeq \boxed{4.55439 \times 10^{-7}}$$

$$P(2 \mid 6.9, 3.1, 5.4, 2.1)$$

$$= \frac{\frac{1}{3} * 0.2846 * 0.98009 * 0.8697 * 1.5728}{0.12719}$$

$$\approx \boxed{0.9999}$$

$\therefore P(C=2 \mid sL, sw, pL, pw) > P(C=0 \mid sL, sw, pL, pw)$

and

$P(C=2 \mid sL, sw, pL, pw) > P(C=1 \mid sL, sw, pL, pw)$

$\therefore$ The example features belong to class $\underline{2}$

Example for Some Calculations by python

Variance and mean

of each feature for each class using: np.var and mean

```
iris2 = pd.read_csv('iris.csv')
print(iris2.head())
print('##########################')

## Sepal Length with 3 classes: Class_0 --> [:5], Class_1 --> [5:10], Class_2 --
print(np.var(iris2['sepalLength'][:5]))
print(np.var(iris2['sepalLength'][5:10]))
print(np.var(iris2['sepalLength'][10:15]))
print('##########################')
```

μ Class_0 = 5.08, 3.52, 1.52, 0.26 ----> σ Class_0 = 0.17, 0.218, 0.0136, 0.0064
μ Class_1 = 6.24, 3.16, 4.52, 1.56 ----> σ Class_1 = 0.09, 0.045, 0.010, 0.00184
μ Class_2 = 6.08, 2.86, 5.18, 2.02 ----> σ Class_2 = 0.47, 0.082, 0.154, 0.0576

Likelihood Calculation

```
# var --> the variance you want
# mean --> mean you want to insert
# x --> the feature value

var = 0.0576
mean = 2.02
x = 2.1

num = (1/(mt.sqrt(2*3.14*var)))
num2 = -pow((x-mean),2)
den2 = 2*var
likelihood = num* mt.exp(num2/den2)
print(likelihood)
```

Prediction

Since the posterior probability of class 2 was the highest then, performing argmax (just comparing manually) it's predicted that these features account for a **Virginica (class 2)** flower.

Part #2

1) Importing needed libraries

```
from sklearn import datasets
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
import pandas as pd
import numpy as np
import math as mt
import matplotlib.pyplot as plt
```

2) Loading the Iris dataset and converting it to 2D Iris.

Loading IRIS Dataset

```
iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target, columns=["Species"])
# print(X.head(10))
# print(y.head(10))
```

Dropping Petal features

```
X = X.drop(["petal length (cm)","petal width (cm)" ],axis=1)
```

3) Steps to plot the likelihood –hard coded function- (for this since the same goes for each class; hence we only provide a sample of the screenshots for one class to ease the scanning, whilst you can find the whole code in the .ipynb file)

Steps to calculate likelihood ¶

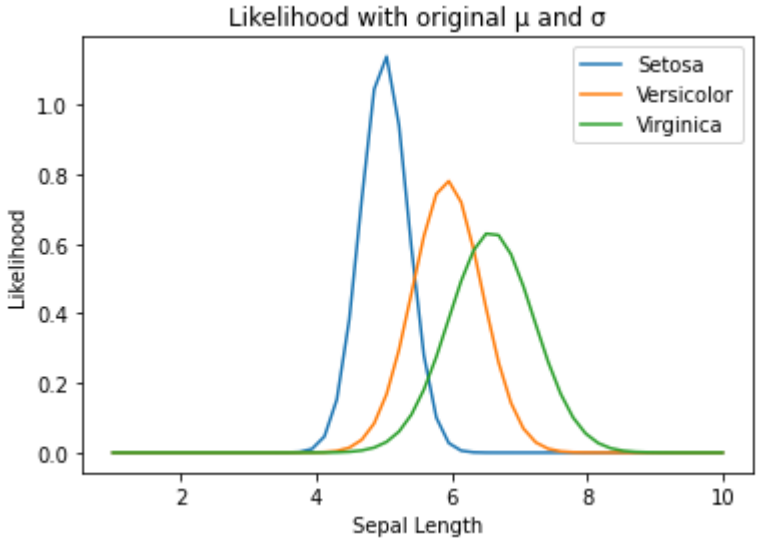
```
sepal_length_setosa = X['sepal length (cm)'][:50]
sepal_length_Versicolor = X['sepal length (cm)'][50:100]
sepal_length_Virginica = X['sepal length (cm)'][100:]
```

```
# Compute sample mean and stdev, for use as model parameter value guesses
mu_setosa= np.mean(sepal_length_setosa)
sigma_setosa = np.std(sepal_length_setosa)
# Sorting to make the plot appear fancier
sepal_length_setosa = sepal_length_setosa.sort_values(ascending=True)
```

```
def like_setosa(xaxis, mu_setosa, sigma_setosa):
    li_setosa  = []
    for x in xaxis :
            num = (1/(mt.sqrt(2*3.14*pow(sigma_setosa,2))))
            num2 = -pow((x-mu_setosa ),2)
            den2 = 2*pow(sigma_setosa ,2)
            point = num* mt.exp(num2/den2)
            li_setosa.append(point)
    return li_setosa
```

```
## Original Features
xaxis = np.linspace(1, 10, num=50)
plt.plot(xaxis, like_setosa(xaxis, mu_setosa, sigma_setosa))
plt.plot(xaxis, like_Versi(xaxis, mu_Versicolor, sigma_Versicolor))
plt.plot(xaxis, like_Virgi(xaxis, mu_Virginica, sigma_Virginica))
plt.xlabel('Sepal Length')
plt.ylabel('Likelihood')
plt.title('Likelihood with original μ and σ')
plt.legend( ['Setosa', 'Versicolor', 'Virginica'])
plt.show()
```

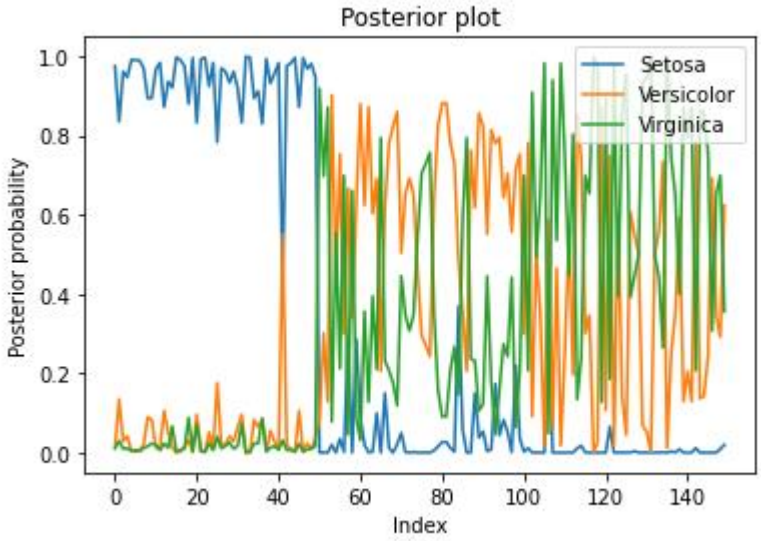
Result/Plot:



4) Applying Naïve bayes

```
model = GaussianNB()
model.fit(X, y)
y_pred = model.predict(X)
# print(y_pred)
```

5) Plot posterior probability
(where the posterior probability encompasses the same meaning as predict_proba that calculates the probability belonging to each class)



6) Accuracy

```
print("Gaussian Naive Bayes model accuracy: {} %".format(metrics.accuracy_score(y, y_pred)*100))
```

```
Gaussian Naive Bayes model accuracy: 78.0 %
```

7) Change parameters: **μ = 5.5**

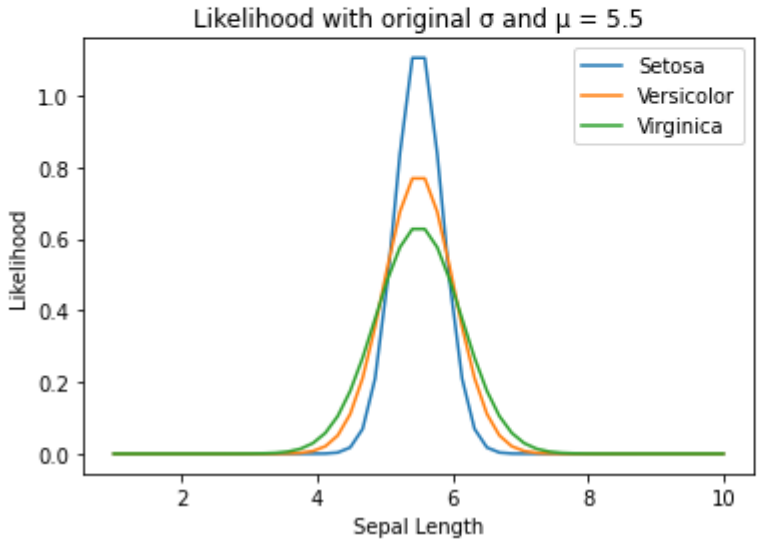
```
model2 = GaussianNB()
model2.fit(X, y)

model2.theta_[:,0] = [5.5, 5.5, 5.5]
print(model2.theta_)
```

```
[[5.5   3.428]
 [5.5   2.77 ]
 [5.5   2.974]]
```

8) Repeating the previous steps

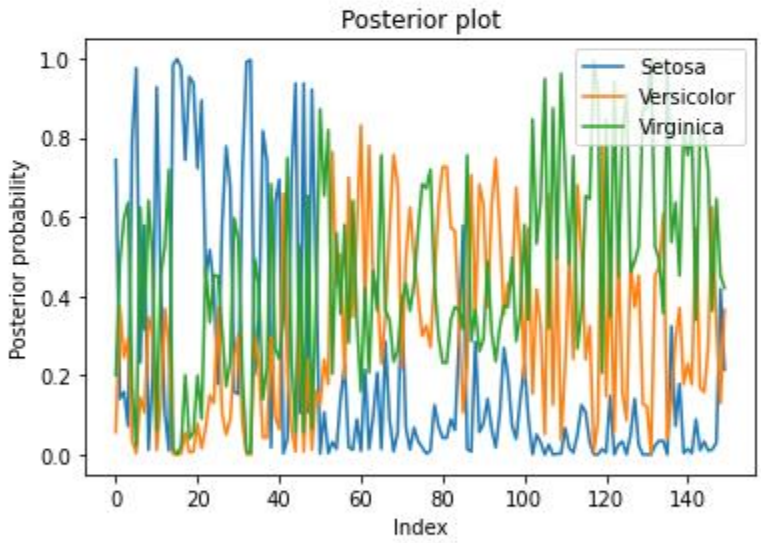
```
## μ = 5.5
plt.plot(xaxis,like_setosa(xaxis,5.5, sigma_setosa))
plt.plot(xaxis,like_Versi(xaxis,5.5, sigma_Versicolor))
plt.plot(xaxis,like_Virgi(xaxis,5.5, sigma_Virginica))
plt.xlabel('Sepal Length')
plt.ylabel('Likelihood')
plt.title('Likelihood with original σ and μ = 5.5')
plt.legend( ['Setosa', 'Versicolor', 'Virginica'])
plt.show()
```



Apparently, after unifying the mean the 3 graphs (likelihoods of 3 classes) too much overlapped.

```
y_pred2 = model2.predict(X)
```

```
y_pred2_proba = model2.predict_proba(X)
plt.plot(y_pred2_proba)
plt.xlabel('Index')
plt.ylabel('Posterior probability')
plt.title('Posterior plot')
plt.legend(['Setosa', 'Versicolor', 'Virginica'], loc=1)
plt.show()
```



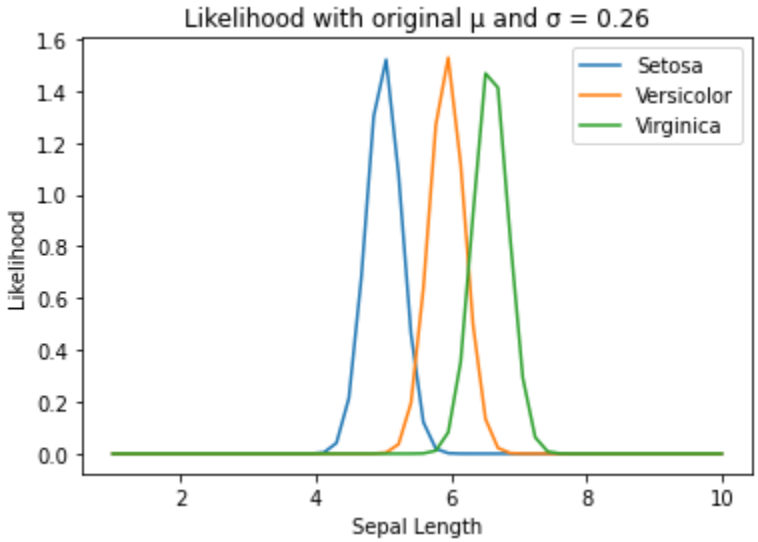

```
print("Gaussian Naive Bayes model accuracy: {} %".format(metrics.accuracy_sc
```

```
Gaussian Naive Bayes model accuracy: 62.66666666666667 %
```

9) Change parameters: **σ = 0.26**

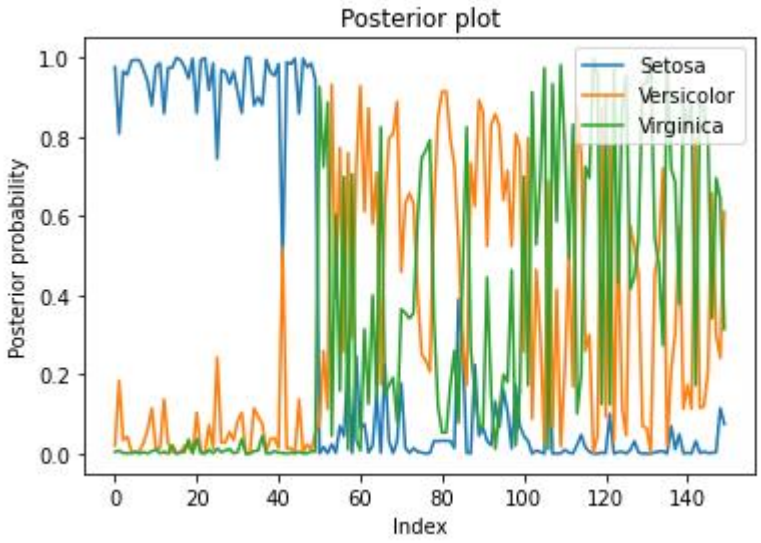
```
model3.sigma_[:,0]= [0.26, 0.26, 0.26]
```

Likelihood:



Here the likelihood graphs seem to be more further from each other.

Posterior:



Accuracy:

```
Gaussian Naive Bayes model accuracy: 80.0 %
```

Features	Accuracy
original	78.0 %
μ = 5.5	62.67 %
σ = 0.26	80.0 %

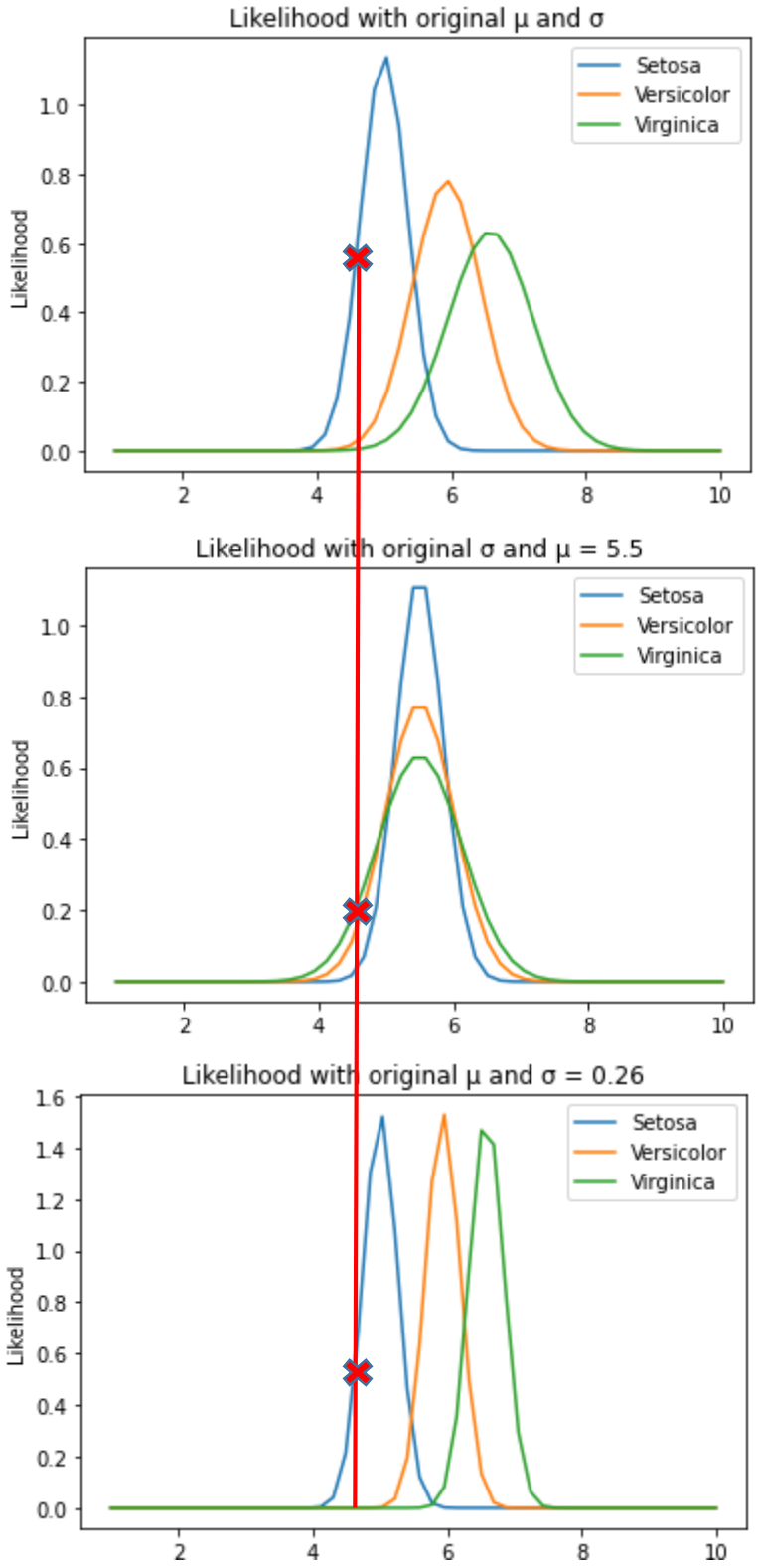
10) **Conclusion**

From the graphs of the likelihood, the original features maintained quite good discrimination between the three classes, however, on changing the mean value and unifying it to 5.5, the three graphs were superimposed over each other. This could tell that the model would be confused to decide which class should it predict for a given point. Nevertheless, with the unification of the variance, the 3 graphs were far away from each other with very little overlapping which might imply that the model can discriminate quite well.

An example for this, is the figures below: for a given point (x = 4.5) what is the likelihood of the three cases?

With original features the model can discriminate well at this point and predict its class with very little confusion, while when we changed the mean it seems that the model got confused more by these overlaps, however, on changing the sigma the model confusion seems to be zero and it could discriminate much better.

This is an evidence that naïve bayes gets affected easily with slight changes in mean and variance, and this may imply that it can be affected by noisy data and outliers.


Likelihood with original μ and σ
Setosa
Versicolor
Virginica
Likelihood
Likelihood with original σ and μ = 5.5
Setosa
Versicolor
Virginica
Likelihood
Likelihood with original μ and σ = 0.26
Setosa
Versicolor
Virginica
Likelihood