DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Homework Assignment No. 1:

HW No. 1: Gaussian Distributions

submitted to:

Professor Joseph Picone
ECE 8527: Introduction to Pattern Recognition and Machine Learning
Temple University
College of Engineering
1947 North 12th Street
Philadelphia, Pennsylvania 19122

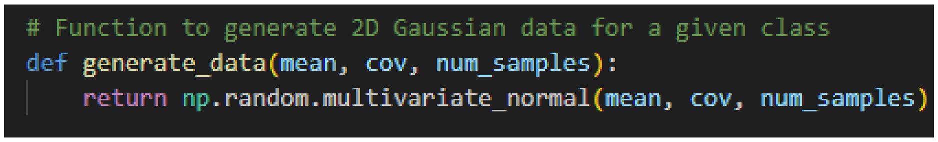
January 22, 2024

prepared by:

Abraham Paroya
Email: paroyaabraham@temple.edu

A. DESCRIPTION OF THE TASK 1

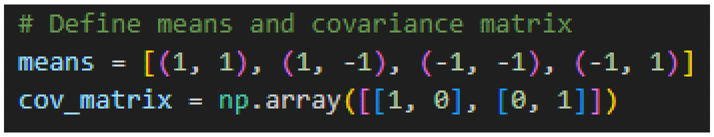
For the first task, code was developed to generate data for distinct classes, with each class being centered around a unique mean. Four classes were generated centered around (1,1), (1,-1), (-1,1) and (-1,-1) respectively. Using python, a data set of 400 different vectors was generated so that each class had 100 different entries for their respective means. To generate this data, two python libraries NumPy and pandas. NumPy was used for its support of large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. Pandas was used because of its handling of data structures for efficiently storing and manipulating large datasets, as well as tools for working with structured data. The main generate data works as follows:



```python
# Function to generate 2D Gaussian data for a given class
def generate_data(mean, cov, num_samples):
    return np.random.multivariate_normal(mean, cov, num_samples)
```

Figure 1. Generate Data Code

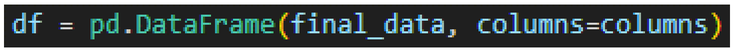
The data generation function takes in the mean, covariance matrix and number of samples and then generates the data accordingly. For task 1, the covariance matrix is $\begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$. The code utilizes a function within the NumPy library called multivariate_normal which generates any set of correlated real-valued random variables each of which clusters around a mean value. To automate the process of generating this data, each mean for the 2-D space was defined and put into an array called mean. The covariance matrix was also defined as cov_matrix.



```python
# Define means and covariance matrix
means = [(1, 1), (1, -1), (-1, -1), (-1, 1)]
cov_matrix = np.array([[1, 0], [0, 1]])
```

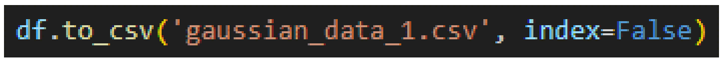
Figure 2. Mean and covariance definitions

The primary data generation occurs within a loop that iterates over the specified means for each class. For each mean, the data is processed through the `generate_data` function. Simultaneously, class labels are assigned to the data within the loop, reflecting the mean of the respective class. Then data frames are created using pandas to reflect the format of the data asked for in the homework description: Class, feature vector 1 and feature vector 2. Finally the data is saved into a csv file.



```python
df = pd.DataFrame(final_data, columns=columns)
```

Figure 3. Pandas Data Frame



```
df.to_csv('gaussian_data_1.csv', index=False)
```

Figure 3. Data Saved to CSV

Once the data is saved into a CSV file it can then be imported into the JMP software. Once imported into the software, the data was graphed onto a scatterplot with feature vector 1 on the x-axis and feature vector-2 on the y-axis and the class put into the color section to distinguish the data of each class.

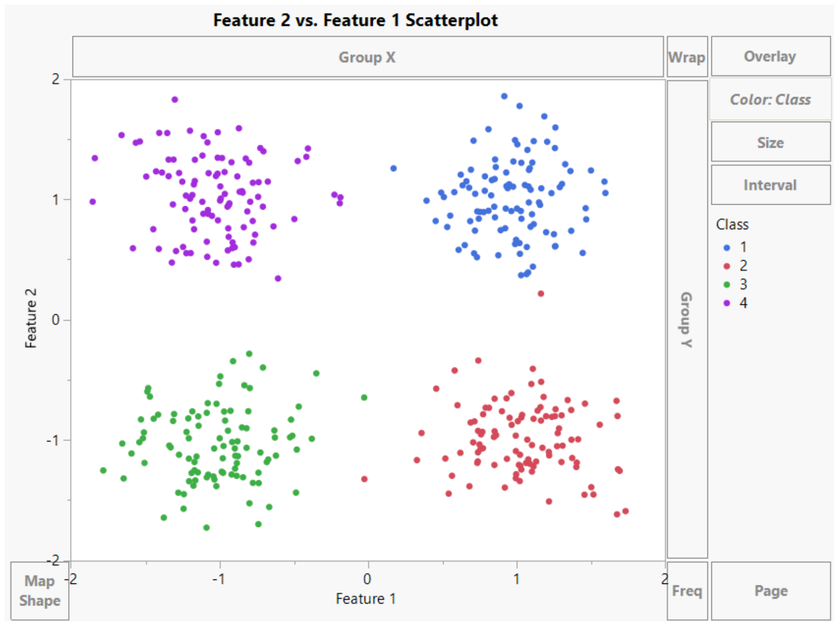
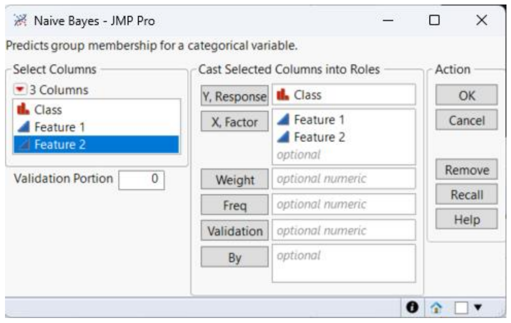


Figure 4. Data Scatterplot

As shown in the graph, there are four distinguishable classes each centered around the points mentioned above. Then the naïve bayes predictive model is run on the data with the following settings:



Predicts group membership for a categorical variable.

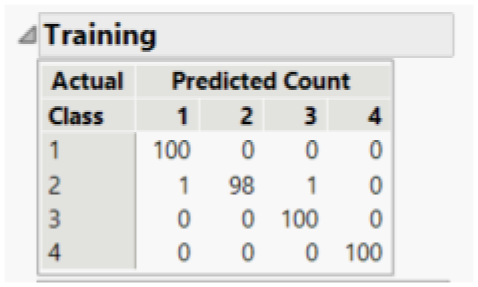
3 Columns
Class
Feature 1
Feature 2

Validation Portion 0

Y, Response	Class
X, Factor	Feature 1 Feature 2 optional
Weight	optional numeric
Freq	optional numeric
Validation	optional numeric
By	optional

Figure 5. Naïve Bayes Settings

The Y response is the class value 1 through 4 and X factors are the feature vectors for each class. Through training, the following confusion matrix is produced:



Training

Actual	Predicted Count			
Class	1	2	3	4
1	100	0	0	0
2	1	98	1	0
3	0	0	100	0
4	0	0	0	100

Figure 6. Confusion Matrix

In total, the naïve bayes predicted 101 data points for class 1, 98 for class 2, 101 for class 3 and 100 for class 4. This makes sense looking at the scatterplot graphically because for class two there are two data points that are out of the predicted bounds.

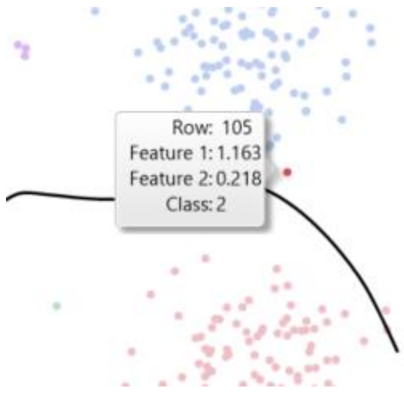


Figure 7. Incorrect Prediction

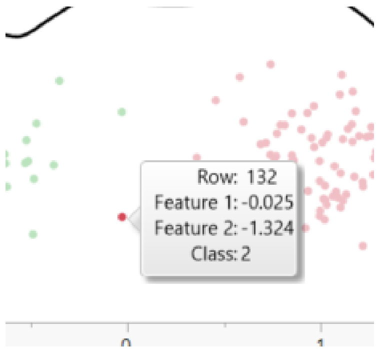
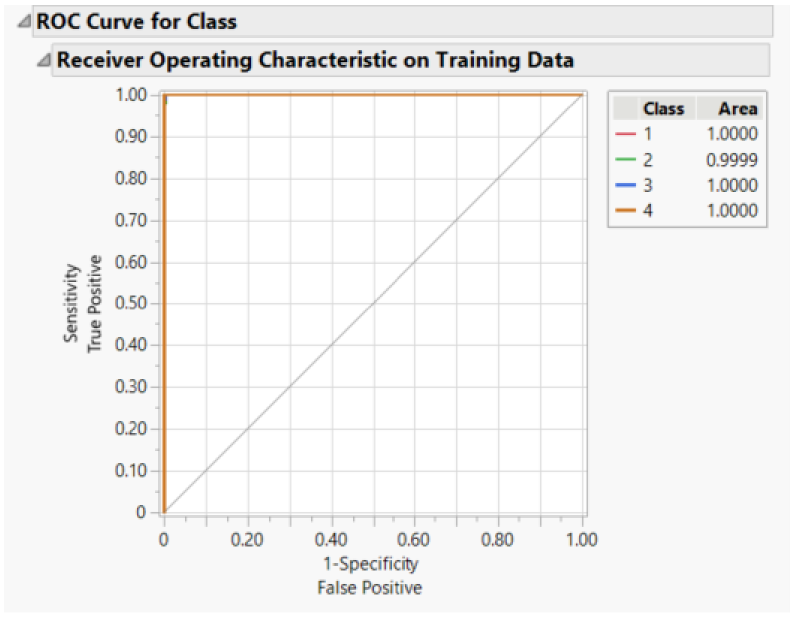


Figure 8. Incorrect Prediction

The first value falls above the x-axis at the because of the y value 0.218. The second value falls to the left of the y-axis because of its x value -0.025. Overall the predictive model has a misclassification rate of 0.00500 with a total of 2 misclassifications. The ROC curve for each class was also given as follows:



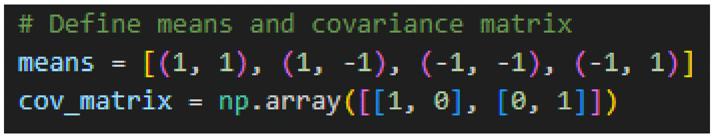
Class	Area
1	1.0000
2	0.9999
3	1.0000
4	1.0000

Figure 9. ROC Curve

Classes 1, 3 and 4 the true-positive rate was 1 and the false positive rate was 0 giving the optimized curve a boxed look. The area under the ROC curve was 1 meaning that it had a perfect overall performance as a binary classifier. Class 2 had an area of 0.9999 meaning that the true positive and false positive rates were not perfectly 1 and 0 respectively. This means the performance was not perfect overall as a binary classifier.

B. DESCRIPTION OF THE TASK 2

For task two, the same process as task 1 was followed but when generating the data for the four classes, the convolution matrix was now:



```
# Define means and covariance matrix
means = [(1, 1), (1, -1), (-1, -1), (-1, 1)]
cov_matrix = np.array([[1, 0], [0, 1]])
```

Figure 10. Mean and covariance definitions.

This means that distribution for the data should now be equal. The expected output of using an identity covariance matrix is that in each quadrant of the scatter plot there should be circular data distributions. The following output was received:

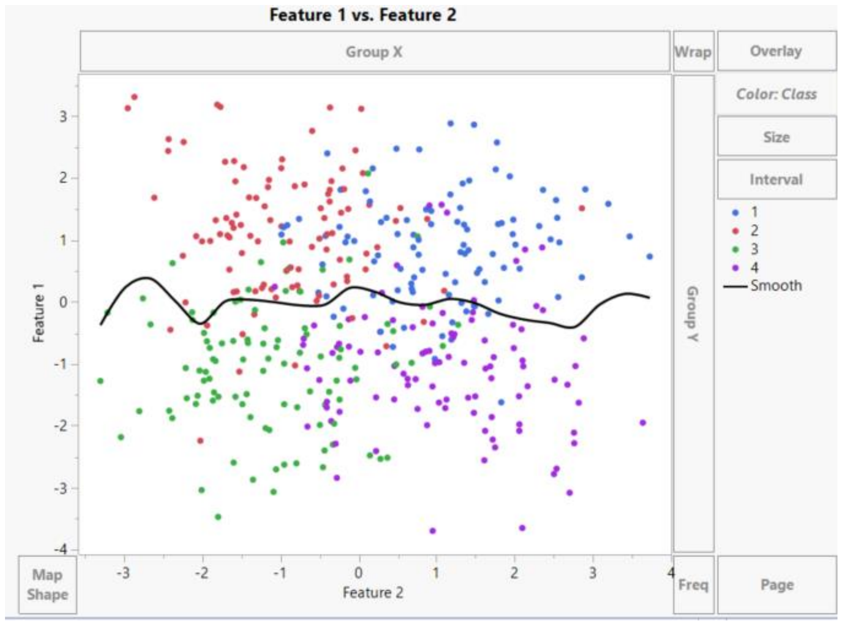
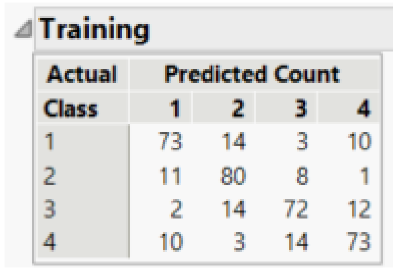


Figure 11. Data Scatterplot

Here the data tends to still generally fall into the areas of their respective central means but is more scattered than the data previously shown in task 1. This goes contrary to the expected output of perfect circles and currently I am not aware of the reason as to why this is. Going from this observed scatter plot it is safe to assume that using the naïve bayes algorithm the misclassification rate will be much higher for this set of data as there is mixing in the center. The following confusion matrix was produced from the data:



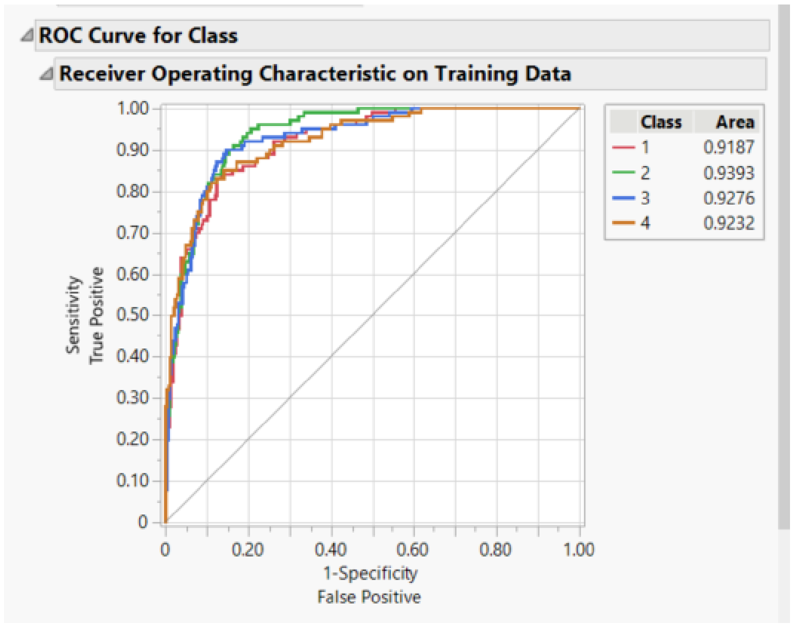
Training

Actual	Predicted Count			
Class	1	2	3	4
1	73	14	3	10
2	11	80	8	1
3	2	14	72	12
4	10	3	14	73

Figure 12. Confusion Matrix

For each class the binary classification predicted an inaccurate amount of the class to be present. Overall, the system has a misclassification rate of 0.25500 and a total of 102 misclassifications.

The ROC curve for the training was:



Class	Area
1	0.9187
2	0.9393
3	0.9276
4	0.9232

Figure 13. ROC Curve

The ROC curve is more of an actual curve as opposed to the curve in task 1. For each class the area was above 0.9 meaning that generally each class has a high ability to distinguish between the positive and negative classes.

C. SUMMARY

This assignment has provided me with a foundational understanding of generating Gaussian distribution data using Python. However, I am currently grappling with the challenge of comprehending why the data in task 2 became intertwined when the covariance identity matrix suggests it should be evenly spread and distributed in spheres. It is likely that there is a bug in the Python code responsible for generating these numbers, leading to this unexpected outcome.

Furthermore, this assignment has deepened my insight into how the Naïve Bayes algorithm functions and its application in classifying Gaussian data. I've learned that the Naïve Bayes algorithm endeavors to establish an optimized boundary within the data to differentiate between various classes. It appears that Naïve Bayes excels in scenarios where data is well-distributed with minimal overlap. However, in situations with significant overlap, as observed in task 2, the Naïve Bayes algorithm encounters challenges in optimizing a boundary to distinguish different portions of the data.

Additionally, this homework assignment has enhanced my proficiency in utilizing JMP as a statistical analysis tool, particularly in navigating the graph builder and predictive modeling sections of the tool.