Pattern Recognition and Machine Learning

(Winter 2022)

Assignment 5

Report

Question - 1:

Task - 1: **Pre-process the data and split in training and test set in 70:30 ratio and make sure all classes are present in test data in approximately the same number. Visualize the training data with scatterplot, using all possible combinations of two attributes.**

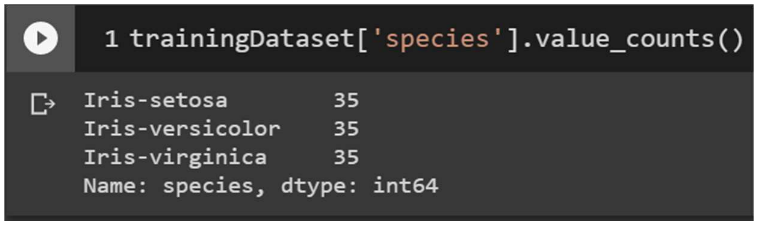
I first took the input of the given iris dataset. Then I performed the data pre-processing steps on the given dataset.

I checked if there are any null values present in the dataset. It appears that there are no null values present in the dataset.

I also checked the type of each column of the dataset to check that the datatype of each column is correct and there is no error in the given dataset.

Finally, I performed the splitting of the dataset into training datasets and testing datasets. Since, we have to make sure that all classes are present in test data in approximately in same number. So, for this task I just made the subset of given dataset based on species. So, I have now three data frames categorized based on species. Now, for splitting I split all three data frames into training and test dataset ratio of 70:30. Finally, I concat the three training data frames into one complete training dataset and the three testing data frames into one complete testing dataset.

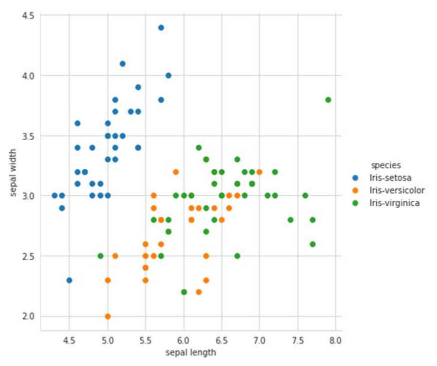
The image confirms that all the classes that are present in the training and testing data are in the same number.

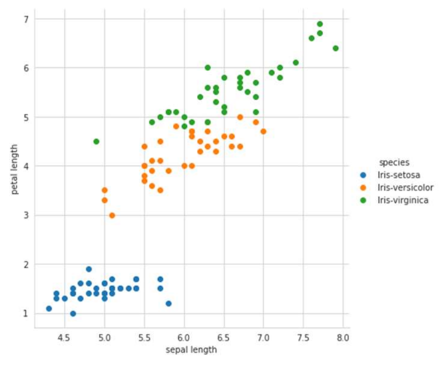


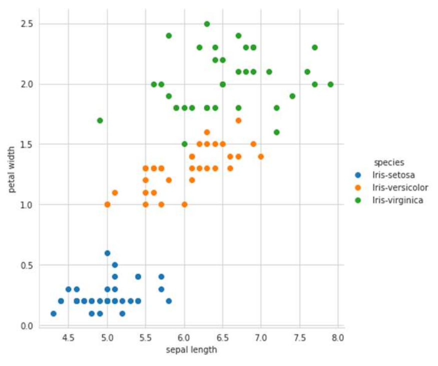
```
trainingDataset['species'].value_counts()
```

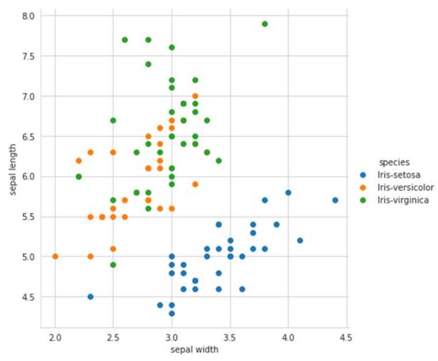
```
Iris-setosa        35
Iris-versicolor    35
Iris-virginica     35
Name: species, dtype: int64
```

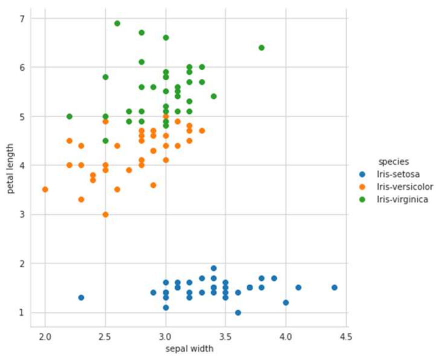
Then, I visualized the training data with scatterplot using all possible combinations of two attributes. The plot that I got for all possible combinations of two attributes are attached in the next page.

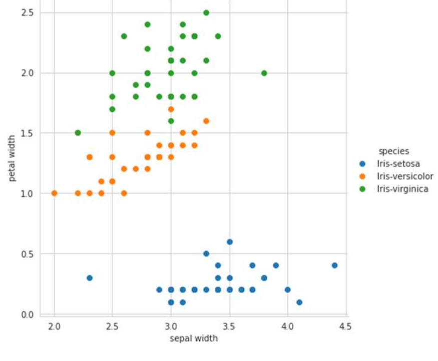


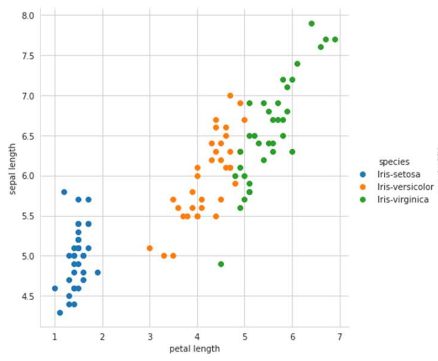


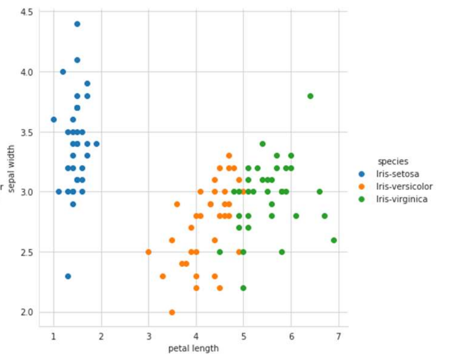


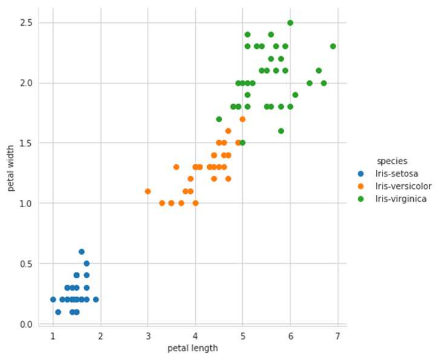


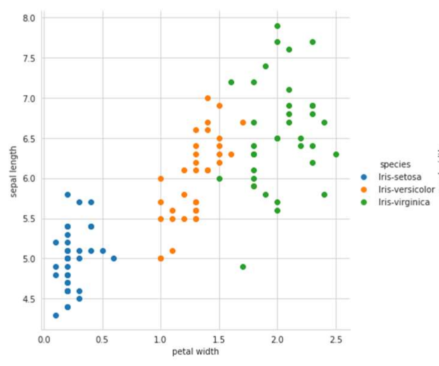


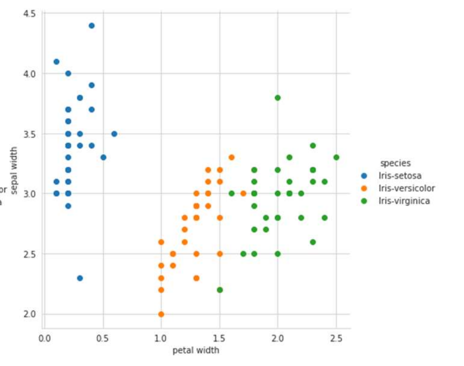


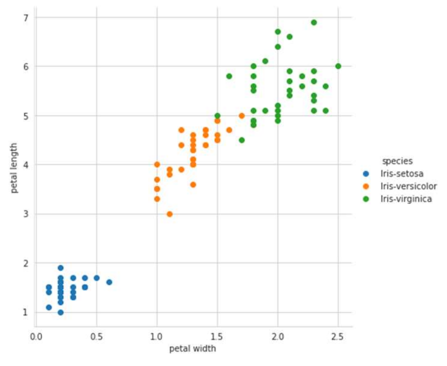




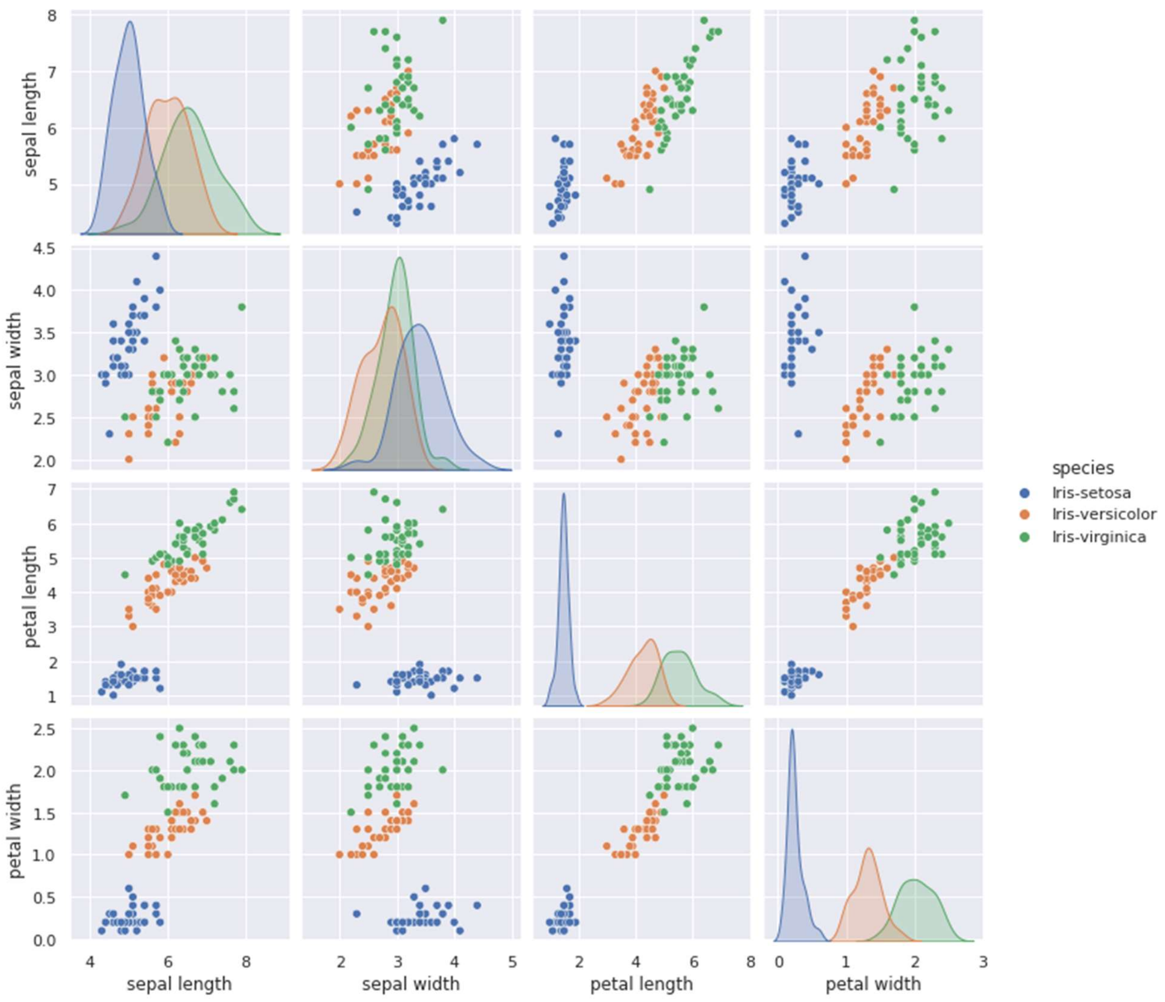








Then I also plotted the pair plot that also helps to visualize the scatter plot of all the possible combinations of two attributes in a single plot. The plot that I got is attached in next page.



<u>Task - 2:</u> **Choose any three pairs of features (which you think will give good results), Train QDA models for each pair, for the classification task. You can use the QuadraticDiscriminantAnalysis function from sklearn.**

For this task, I chose the three pair of features that I thought will give the good results. So, for choosing the three pairs of features, I observed the distribution of the pair of features on the scatterplot that I plotted in the previous task. The three features that I chose are sepal length v/s petal width sepal, width v/s petal width and petal length v/s petal width.

Then, I imported the QuadraticDiscriminantAnalysis function from sklearn library as QDA and first instantiated the QDA(store_covariance=True). Then, I took the training sets and trained the QDA model using fit() method for each pair of features that I have chosen for the classification task. I trained three models as

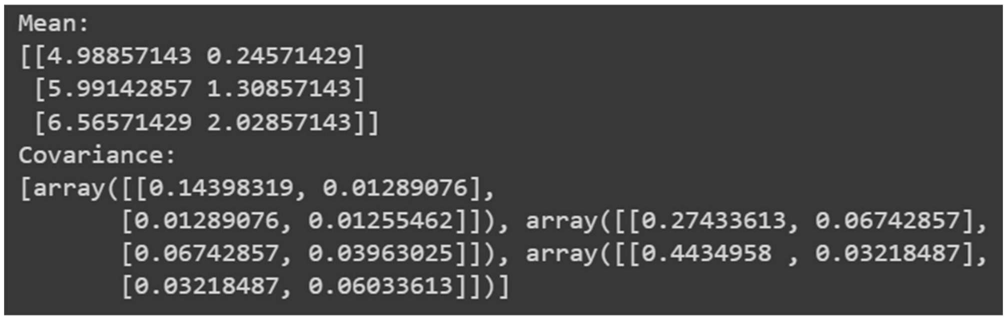
- model_1 for sepal length v/s petal width,
- model_2 for sepal width v/s petal width,
- model_3 for petal length v/s petal width.

<u>Task - 3:</u> Report the mean and covariance of the distributions found from each QDA model.

For this task, I reported the mean and covariance of the distributions found from each QDA model using the methods of QDA i.e., .means_ and .covariance_ respectively.

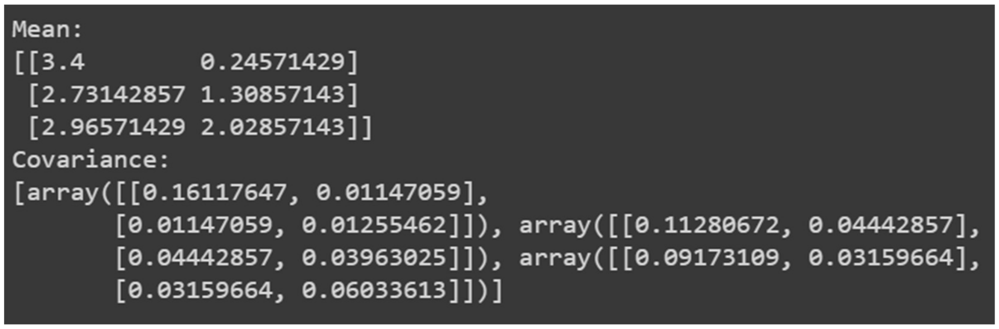
The mean and covariance that I got for the three models are:

For model_1:



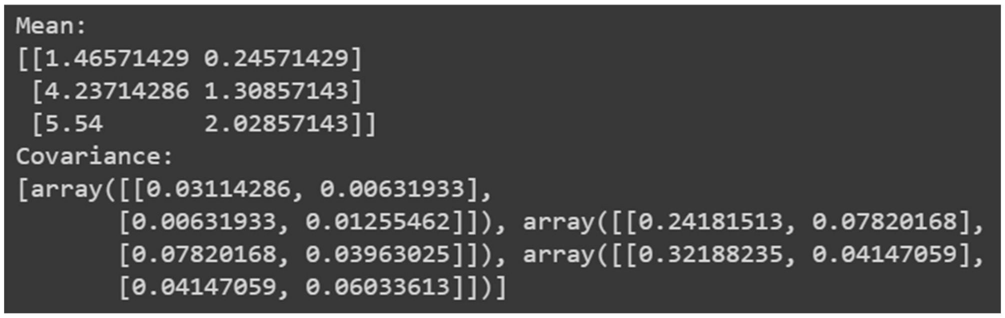
```
Mean:
[[4.98857143 0.24571429]
 [5.99142857 1.30857143]
 [6.56571429 2.02857143]]
Covariance:
[array([[0.14398319, 0.01289076],
       [0.01289076, 0.01255462]]), array([[0.27433613, 0.06742857],
       [0.06742857, 0.03963025]]), array([[0.4434958 , 0.03218487],
       [0.03218487, 0.06033613]])]
```

For model_2:



```
Mean:
[[3.4        0.24571429]
 [2.73142857 1.30857143]
 [2.96571429 2.02857143]]
Covariance:
[array([[0.16117647, 0.01147059],
       [0.01147059, 0.01255462]]), array([[0.11280672, 0.04442857],
       [0.04442857, 0.03963025]]), array([[0.09173109, 0.03159664],
       [0.03159664, 0.06033613]])]
```

For model_3:



```
Mean:
[[1.46571429 0.24571429]
 [4.23714286 1.30857143]
 [5.54       2.02857143]]
Covariance:
[array([[0.03114286, 0.00631933],
       [0.00631933, 0.01255462]]), array([[0.24181513, 0.07820168],
       [0.07820168, 0.03963025]]), array([[0.32188235, 0.04147059],
       [0.04147059, 0.06033613]])]
```

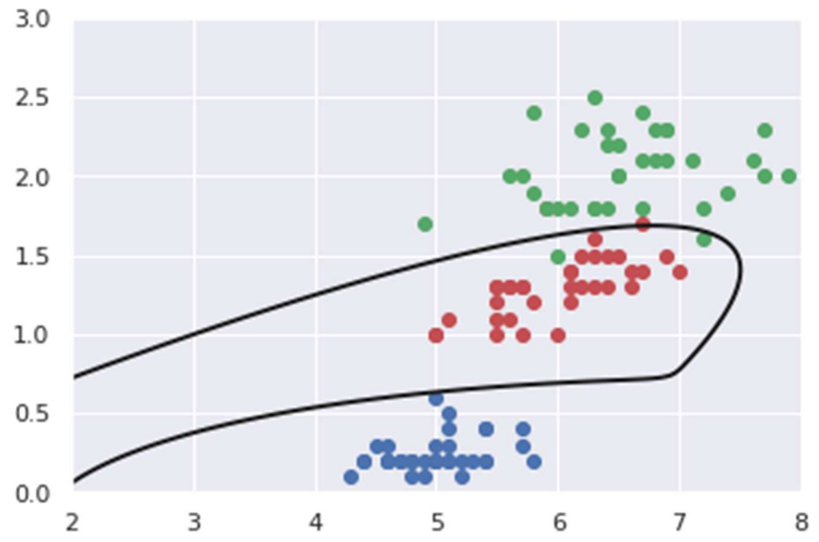
__Task - 4:__ **Plot the decision boundary given by the QDA models on top of the corresponding scatterplot visualization of the data.**

For this task, I plotted the decision boundary given by the QDA model on the top of the corresponding scatterplot visualization of the data.

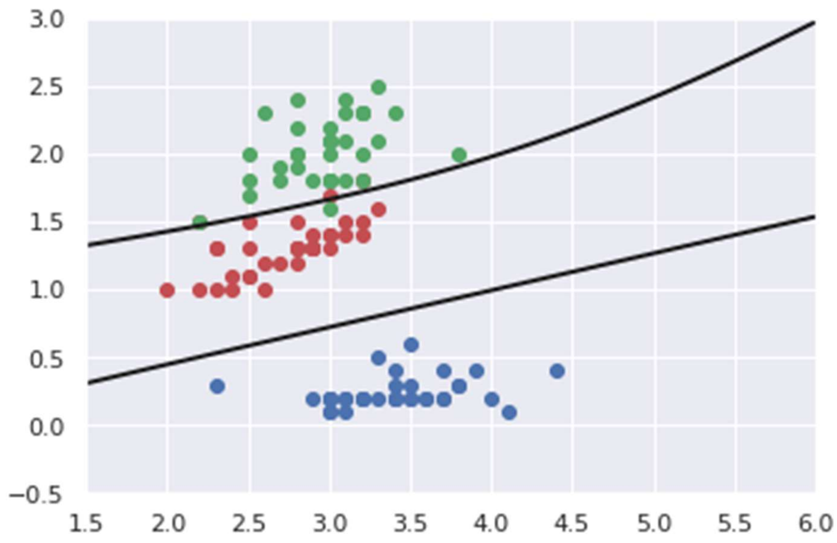
For plotting, I created a mesh grid for x axis and y axis, then I formed a grid named X_grid. Then, I calculated the probabilities using the predict_proba() method and stored these probabilities in a variable named probailitites_qda.

Finally, I first plotted the scatter plot of the feature and then on the same graph using contour curve function/method, I plotted the decision boundary on the same graph.

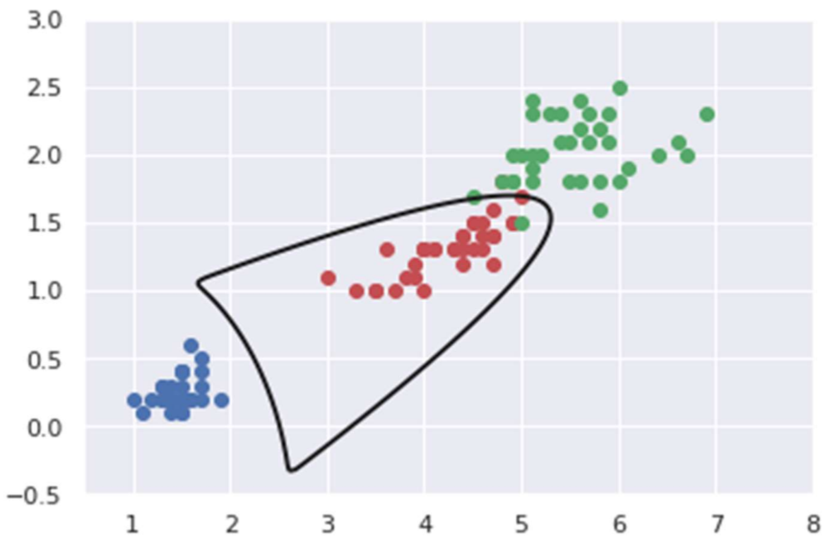
For model_1, I got the plot as:



For model_2, I got the plot as:

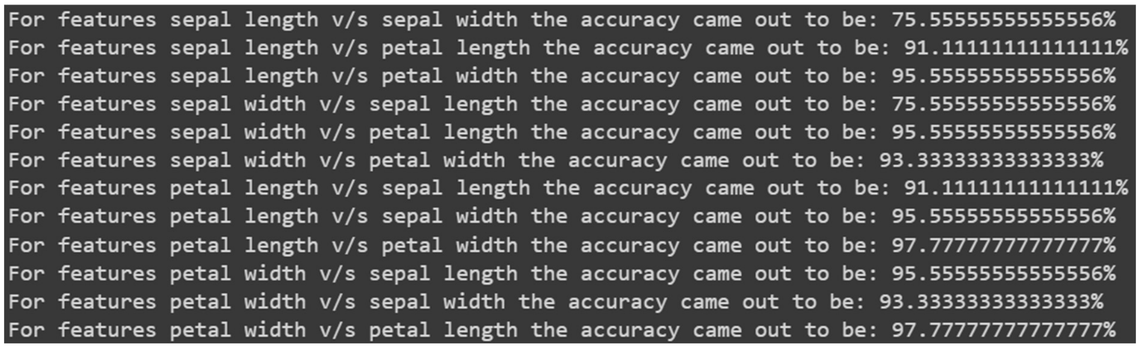


For model_3, I got the plot as:



<u>Task - 5:</u> Predict the test data and report error rate for each case. Which pair of features do you think gives the best result and why?

For this task, I took all possible combination of two features and train the model with those features from training dataset and then predicted the test data and calculated the accuracy for each pair of features. I got the results as:



```
For features sepal length v/s sepal width the accuracy came out to be: 75.55555555555556%
For features sepal length v/s petal length the accuracy came out to be: 91.11111111111111%
For features sepal length v/s petal width the accuracy came out to be: 95.55555555555556%
For features sepal width v/s sepal length the accuracy came out to be: 75.55555555555556%
For features sepal width v/s petal length the accuracy came out to be: 95.55555555555556%
For features sepal width v/s petal width the accuracy came out to be: 93.33333333333333%
For features petal length v/s sepal length the accuracy came out to be: 91.11111111111111%
For features petal length v/s sepal width the accuracy came out to be: 95.55555555555556%
For features petal length v/s petal width the accuracy came out to be: 97.77777777777777%
For features petal width v/s sepal length the accuracy came out to be: 95.55555555555556%
For features petal width v/s sepal width the accuracy came out to be: 93.33333333333333%
For features petal width v/s petal length the accuracy came out to be: 97.77777777777777%
```

From the accuracies that I got the pair of features that gave the best accuracy is petal length v/s petal width. The pair of features petal length v/s petal width gave the best accuracy can be explained form the scatter plot as well because the data was much better distributed in this pair feature on the scatter plot. Hence, this pair of features gave the best accuracy.

Task - 6: Take the pair of features that has given the best result and train LDA model with same training data. You can use the LinearDiscriminantAnalysis function from sklearn.

For this task, I first imported the LinearDiscriminantAnalysis function from sklearn library as LDA and first instantiated the LDA(store_covariance=True) model. Then, I took the data of pair of best features (i.e., petal length v/s petal width) from training datasets and trained the LDA model using fit() method.

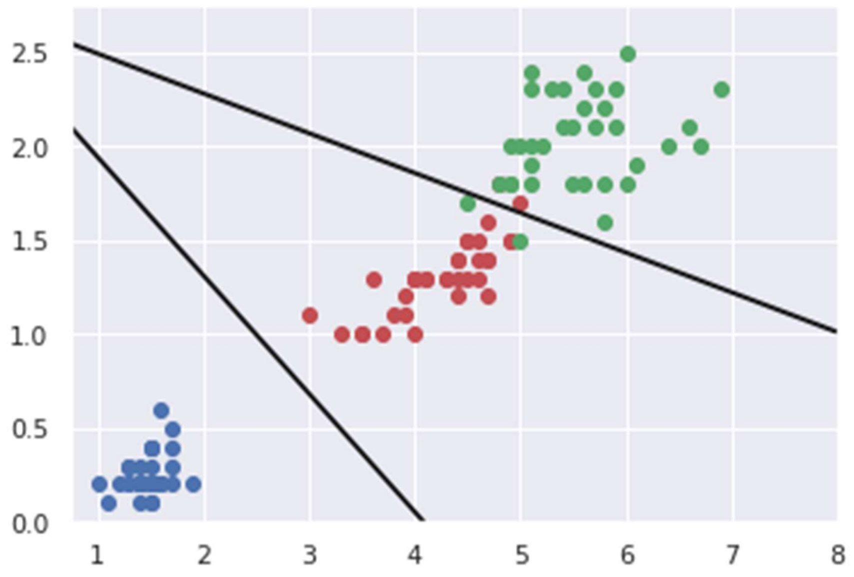
Task - 7: Plot the decision boundary given by the LDA model on top of the scatterplot visualization of the data.

For this task, I plotted the decision boundary given by the LDA model on the top of the corresponding scatterplot visualization of the data.

For plotting, I created a mesh grid for x axis and y axis, then I formed a grid named X_grid. Then, I calculated the probabilities using the predict_proba() method and stored these probabilities in a variable named probailitites_qda.

Finally, I first plotted the scatter plot of the best pair of features and then on the same graph using contour curve function/method, I plotted the decision boundary on the same graph.

For LDA model, I got the graph as:



Task - 8: Report the error rate on the test data for LDA model. Which one between LDA and QDA has performed better do you think? Justify your answer.

For this task, I first predicted the test data on the LDA model trained on the pair of best features. Then, I calculated the accuracy, It came out as 95.55555555555556%.

Hence, Among LDA and QDA, the LDA model (LinearDiscriminantAnalysis) performed better than QDA model(QuadraticDiscriminantAnalysis). As, I got the accuracies for the pair of best features as:

For QDA Model:

Accuracy = 97.77777777777777%

For LDA Model:

Accuracy = 95.55555555555556%

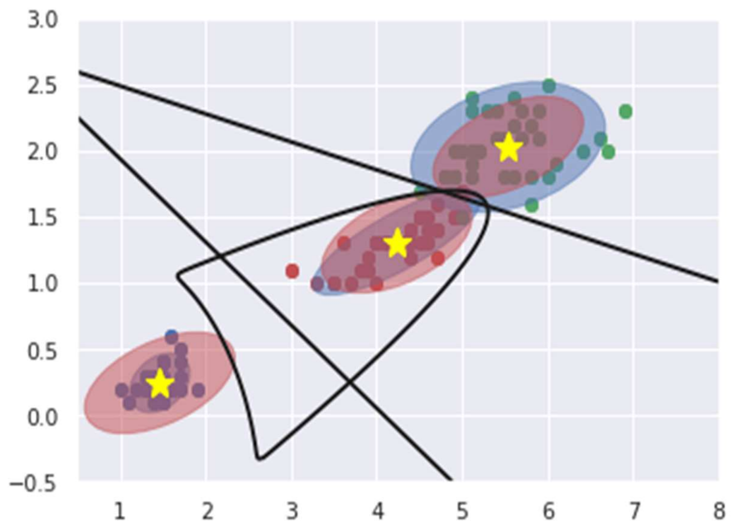
The QDA Model performed better than LDA Model because it allows for more flexibility for the covariance matrix and tends to fit the data better than LDA. Hence, helps to give better classification of the data as compared to LDA model.

Task - 9: Visualize the gaussian distributions obtained from QDA and LDA. (you can draw ellipse as shown in demo or you can use any other function to visualize.).

To visualize the gaussian distributions obtained form QDA and LDA, I implement a function that helps to plot ellipse. Then, I just plotted the scatter plot of the best pair of features and even plotted the decision boundaries and ellipse to shown the perfect comparison between the QDA and LDA model. The plot I got is:

Here, the ellipse in blue colour shows the gaussian distribution of QDA model.

And, the ellipse in red colour shows the gaussian distribution of LDA model.

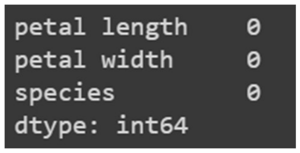


Question - 2:

- **Data Pre-Processing.**

I first took the input of the given iris dataset. Then I performed the data pre-processing on the given dataset.

I checked if there are any null values present in the dataset. It appears that there are no null values present in the dataset.



```
petal length    0
petal width     0
species         0
dtype: int64
```

Since no categorical data was present in the dataset, hence, no categorical encoding was required.

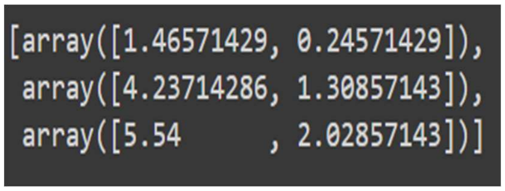
Then, I just split the data into training and testing datasets such that all classes are present in test data in approximately the same number.

Task - 1: From the data, find out the sample mean and sample covariance matrix of each class and visualize the gaussian distribution using the obtained sample mean and sample covariance matrix.

For this task, I calculated the sample mean and sample covariance using the function .mean() and .cov() respectively.

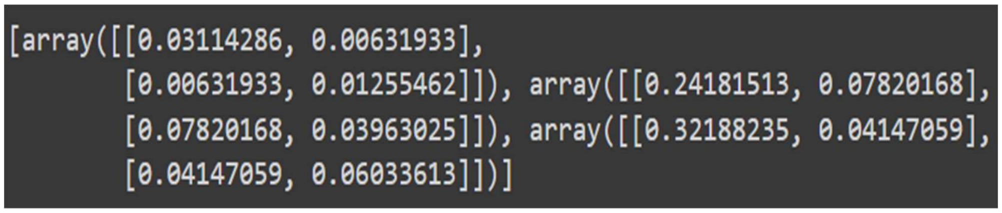
The sample mean and sample covariance that I got for the dataset is:

Sample Mean:



```
[array([1.46571429, 0.24571429]),
 array([4.23714286, 1.30857143]),
 array([5.54      , 2.02857143])]
```

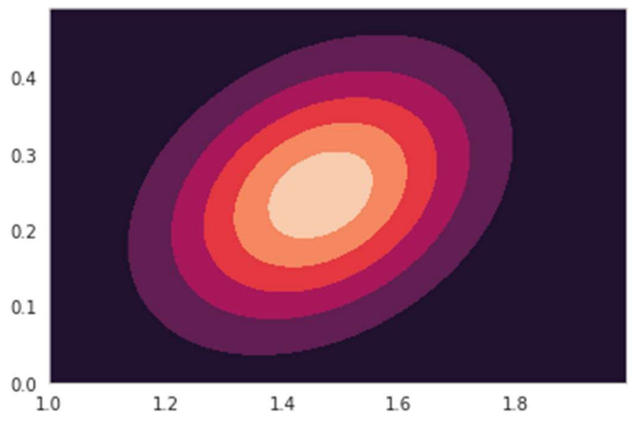
Sample Cov:



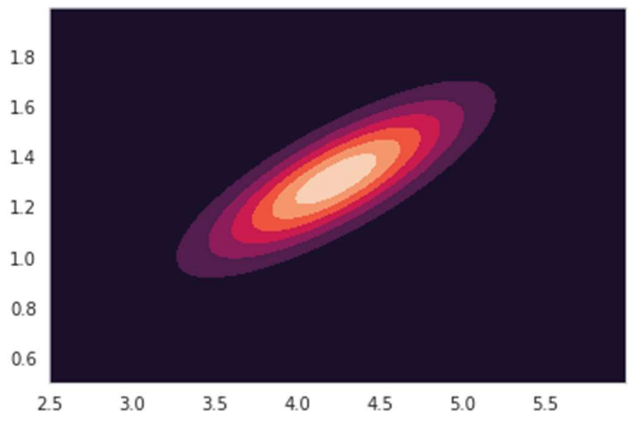
```
[array([[0.03114286, 0.00631933],
        [0.00631933, 0.01255462]]), array([[0.24181513, 0.07820168],
        [0.07820168, 0.03963025]]), array([[0.32188235, 0.04147059],
        [0.04147059, 0.06033613]])]
```

For the visualization using the obtained sample mean and sample covariance matrix. I plotted 2d contours for the visualization:

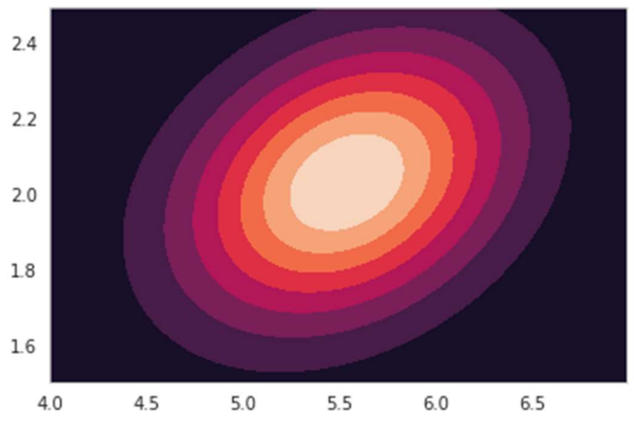
So, Multivariate Gaussian Distribuition for class1 obtained using sampleMean[0] and sampleCov[0] is:



Multivariate Gaussian Distribuition for class2 obtained using sampleMean[1] and sampleCov[1] is:



Multivariate Gaussian Distribuition for class3 obtained using sampleMean[2] and sampleCov[2] is:



<u>Task - 2:</u> **Write a function compute_likelihood to compute the likelihood of data given the parameters mean and covariance matrix assuming gaussian distribution.**

For this part, I wrote a function to compute the likelihood of the data. Since we know that the joint probability density function of Multivariate normal distribution is given by:

$$f_X(x_j) = (2\pi)^{-K/2}|\det(V_0)|^{-1/2}\exp\left(-\frac{1}{2}(x_j-\mu_0)^{\top}V_0^{-1}(x_j-\mu_0)\right)$$

And, the likelihood function is given by:

$$L(\mu, V; x_1, \ldots, x_n) = (2\pi)^{-nK/2}|\det(V)|^{-n/2}\exp\left(-\frac{1}{2}\sum_{j=1}^{n}(x_j-\mu)^{\top}V^{-1}(x_j-\mu)\right)$$

where:

K is dimension (for our case K = 2)

μ is K*1 mean vector.

V is K*K covariance matrix.

Hence, I implemented the formula of likelihood shown above and hence calculated the likelihood of the data given the parameters mean and covariance matrix assuming gaussian distribution.

<u>Task - 3:</u> **Write a function to perform maximum likelihood estimation over the training dataset to determine mean and covariance and classify it using Bayes classifier. Report the parameters obtained for each class.**

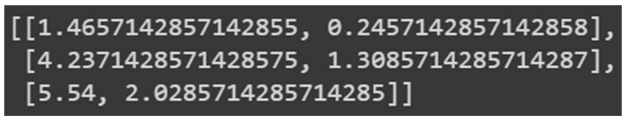
For this task, I wrote a function to perform maximum likelihood estimator over the training dataset to determine the mean and covariance.

Since, we know that given data in form of a matrix X of dimensions m×p, if we assume that the data follows a p-variate Gaussian distribution with parameters mean μ (p×1) and covariance matrix Σ (p×p) the Maximum Likelihood Estimators are given by

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{x^{(i)}} = \bar{\mathbf{x}}$$
$$\hat{\Sigma} = \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x^{(i)}} - \hat{\mu})(\mathbf{x^{(i)}} - \hat{\mu})^T$$

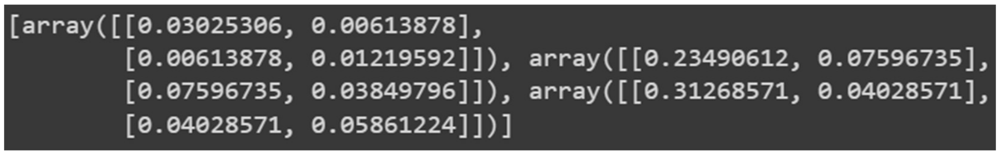
Hence, the parameters that I obtained for each class after performing MLE are:

Mean:



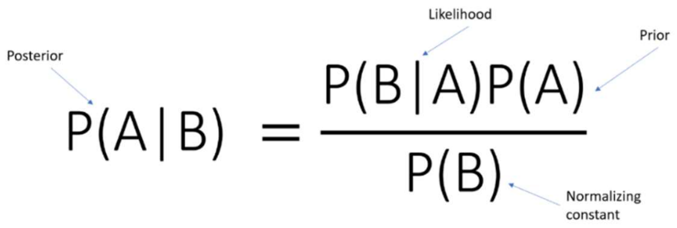
```
[[1.4657142857142855, 0.2457142857142858],
 [4.2371428571428575, 1.3085714285714287],
 [5.54, 2.0285714285714285]]
```

Covariance:



```
[array([[0.03025306, 0.00613878],
       [0.00613878, 0.01219592]]), array([[0.23490612, 0.07596735],
       [0.07596735, 0.03849796]]), array([[0.31268571, 0.04028571],
       [0.04028571, 0.05861224]])]
```

For Classification I implemented a function bayes_classify() to perform Bayes Classification.



$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

So, for this I firstly calculated the prior of each class. Then using the likelihood function created in the previous task, I calculated the likelihood of the data for each class. Hence, I substituted this value in Bayes Classifier formula and also divided the values by evidence/normalizing constant.

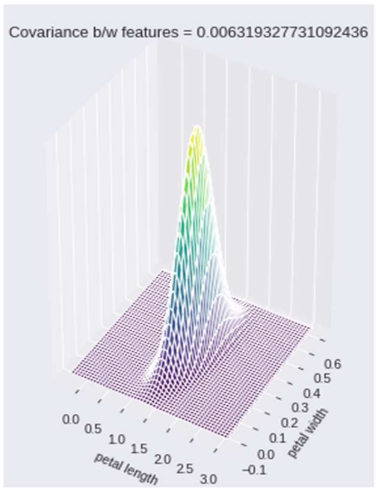
Then, I got the list of posterior probabilities for the entire dataset for each class. Hence, I classified each sample of the data to that class for which the posterior probability was highest.

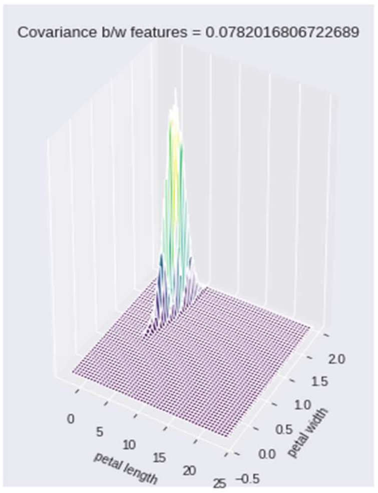
<u>Task - 4:</u> **Visualize the gaussian distribution from mean and covariance of both the above parts (a and c) in a single plot.**

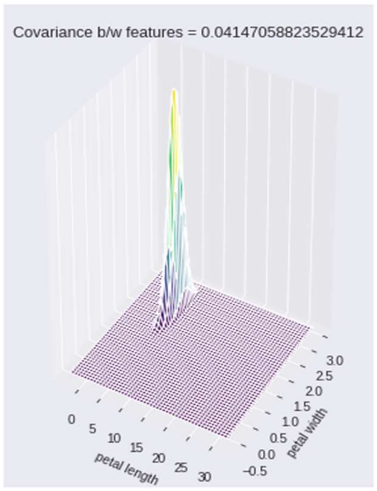
For the visualization of the gaussian distribution from mean and covariance of both the parts, I plotted the graph as:

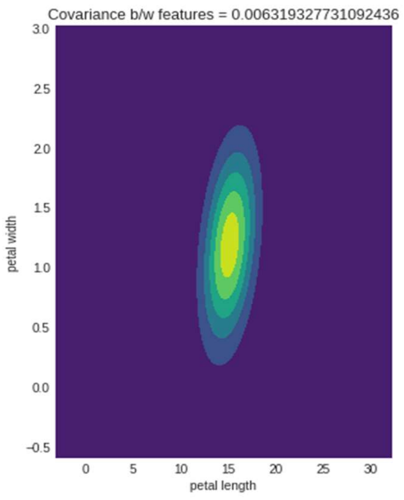
I have plotted the 3-d density function as well as 2-d contours for the distribution using mean and covariance from both the parts.

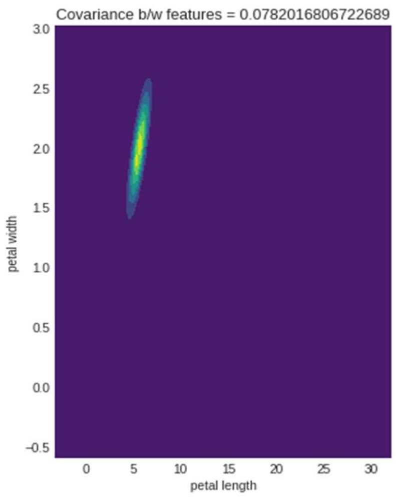
For gaussian distribution from mean and covariance of part a:

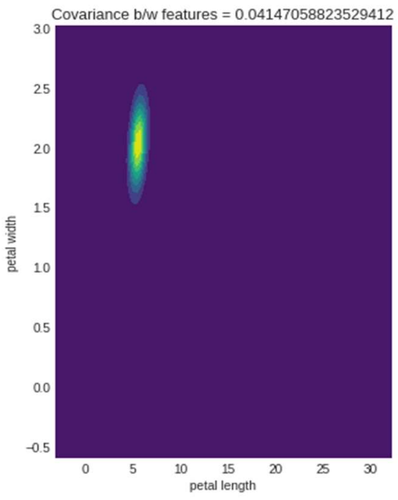




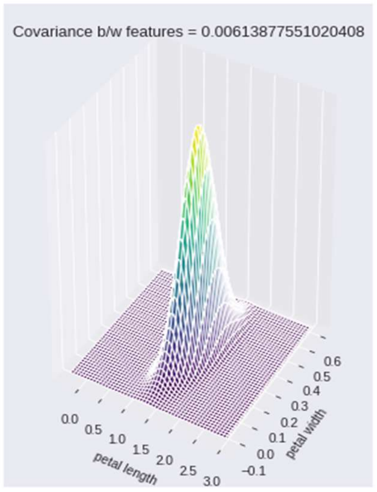


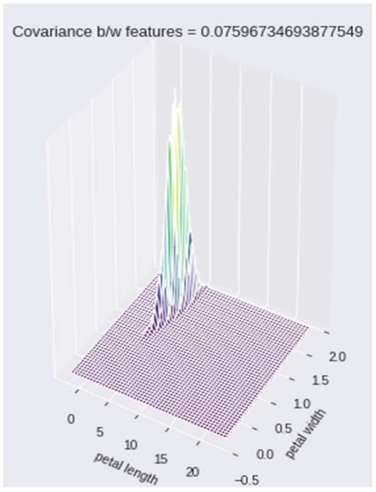


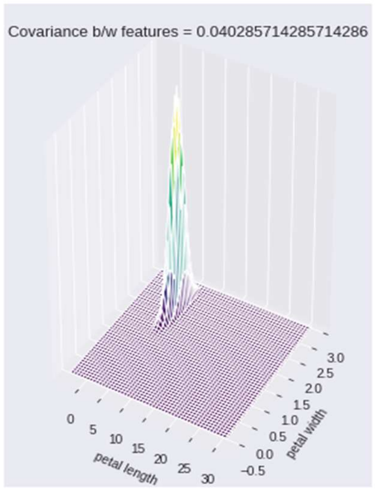


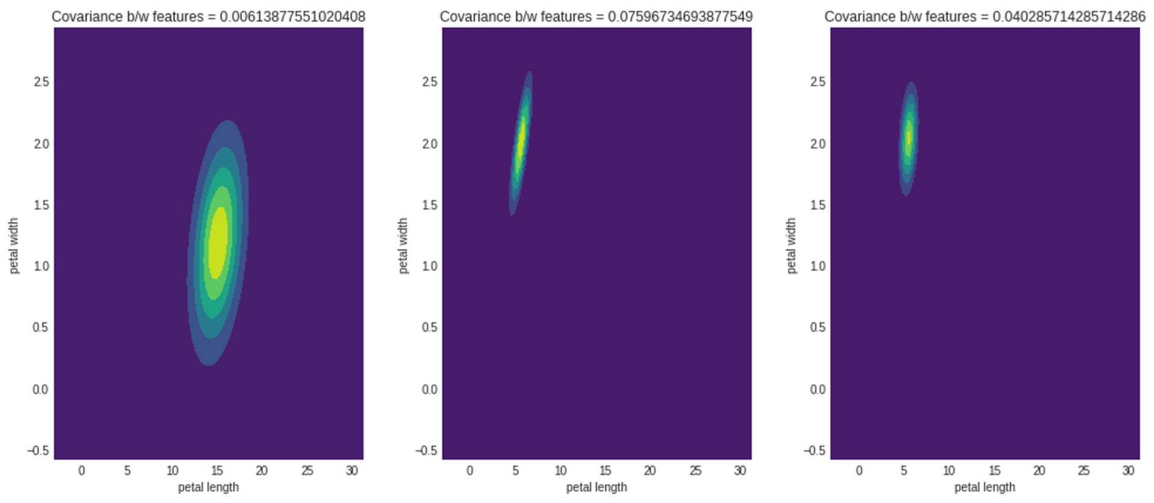


For gaussian distribution from mean and covariance of part b:









<u>Task - 5:</u> Predict the test data and compare the performance obtained in question1(with QDA).

For this task, I used the previously implemented bayes_classify() function to predict the test data. So, I passed the test data to the function and stored the predictions in a list. The accuracy score for the prediction came out be as:

Accuracy = 97.77777777777777%.

So, The Accuracy from bayes classification came out as: 97.77777777777777%.

And, The Accuracy from QDA (question1) came out as: 97.77777777777777%.

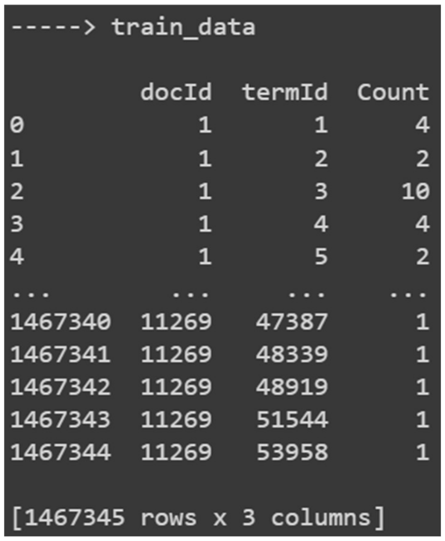
Hence, we can see that from both the models the accuracy came out to be approximately same. So, we can say that both are good approaches to classify the iris dataset.

Question - 3:

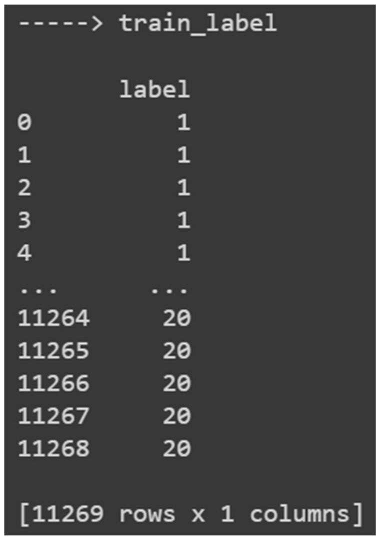
- **Data Pre-Processing.**

I first took the input of the given dataset.

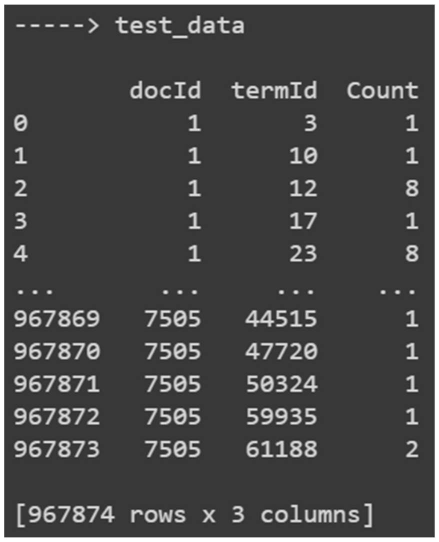
I also labelled the columns of the dataset. The dataset looks like:



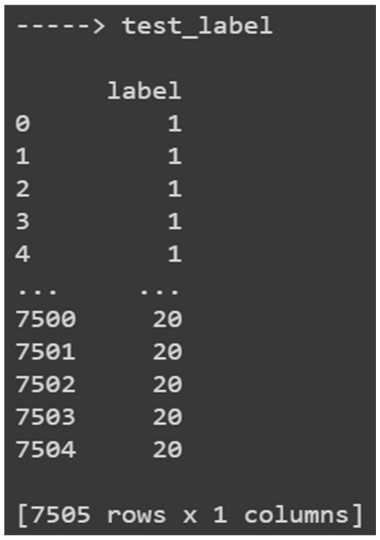
```
-----> train_data

         docId  termId  Count
0            1       1      4
1            1       2      2
2            1       3     10
3            1       4      4
4            1       5      2
...        ...     ...    ...
1467340  11269   47387      1
1467341  11269   48339      1
1467342  11269   48919      1
1467343  11269   51544      1
1467344  11269   53958      1

[1467345 rows x 3 columns]
```



```
-----> train_label

       label
0          1
1          1
2          1
3          1
4          1
...      ...
11264     20
11265     20
11266     20
11267     20
11268     20

[11269 rows x 1 columns]
```

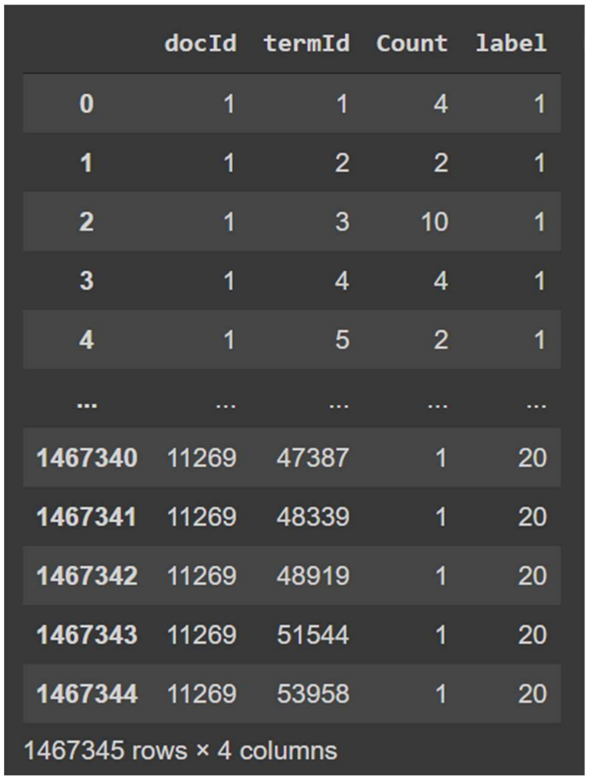


```
-----> test_data

        docId  termId  Count
0           1       3      1
1           1      10      1
2           1      12      8
3           1      17      1
4           1      23      8
...       ...     ...    ...
967869   7505   44515      1
967870   7505   47720      1
967871   7505   50324      1
967872   7505   59935      1
967873   7505   61188      2

[967874 rows x 3 columns]
```



```
-----> test_label

      label
0         1
1         1
2         1
3         1
4         1
...     ...
7500     20
7501     20
7502     20
7503     20
7504     20

[7505 rows x 1 columns]
```

I also created a function that maps the label corresponding to the document id and returns the final dataset. So, I passed the train data and train labels and got final dataset as:



	docId	termId	Count	label
0	1	1	4	1
1	1	2	2	1
2	1	3	10	1
3	1	4	4	1
4	1	5	2	1
...	...	...	...	...
1467340	11269	47387	1	20
1467341	11269	48339	1	20
1467342	11269	48919	1	20
1467343	11269	51544	1	20
1467344	11269	53958	1	20

1467345 rows × 4 columns

I even plotted the data to visualize that how the data is distributed across the features of the dataset.

<u>Task - 1:</u> Compute the likelihood of the training data-set(without using any standard library).

For computing the likelihood of the training dataset. Since we know that Likelihood is the conditional probability of a word occurring in a document given that the document belongs to a particular category. So, I have already separated the data frames based on the label. Hence, I calculated the likelihood of each termId present in each class.

P(termId/label) = (Number of occurrence of the termId in all the documents from a particular label) / (Total count of all termId in every document from a label)

<u>Task - 2:</u> Smoothing is a technique that helps tackle the problem of zero probability in the Naïve Bayes algorithm. Apply Laplace smoothing to solve the zero observations problem on a training data-set from scratch (without using any standard library).

For this task, I implemented a function smoothing() that helps tackle the problem of zero probability in the Naïve Bayes algorithm.

So, in this function “smoothing()”, I implemented the Add-1 (Laplace) smoothing principle i.e., “Assume every seen or unseen event occurred once more than it did in the training data”. Hence, I updated each of the termId count by 1.

Also, I updated the likelihood values by following the principle.

$$\text{Add One} \quad P(w_i) = \frac{C(w_i)+1}{\sum_j (C(w_j)+1)} = \frac{C(w_i)+1}{N+V}$$

Hence, after applying smoothing function to my dataset, I was able to tackle the zero observations problem on the training dataset. Also, now each of the likelihood values is greater than 1. Hence, the problem of zero probability in the Naïve Bayes algorithm has been resolved by the help of implemented smoothing function.

<u>Task - 3:</u> naiveBayesClassify(trainData, trainLabels, testData) - Classifies the data using the Naive Bayes algorithm.

For this task, it is asked to implement a function naiveBayesClassify(trainData, trainLabels, testData) that classifies the data using the Naïve Bayes Algorithm.

So, I made a function “naiveBayesClassify(trainData, trainLabels, testData)” In this, I first divided the testing dataset on the basis of docID since we have to classify the documents.

Then, I calculated the posterior probabilities of testing dataset with the help of likelihood and prior probabilities that are calculated in the previous tasks.

I used the naïve Bayes Classifier approach to calculate the posterior probabilities.

$$\overset{\text{Posterior}}{P(A|B)} = \frac{\overset{\text{Likelihood}}{P(B|A)}\,\overset{\text{Prior}}{P(A)}}{\underset{\text{Normalizing constant}}{P(B)}}$$

Hence, I substituted the values of likelihood and prior and calculated the posterior probabilities and applied the principle of naïve bayes, that classifies the document as label for which the posterior probability was highest.

Hence, the testing dataset was then classified into the 20 different categories.