Pattern Recognition and Machine Learning

Lab - 4 Assignment

Bayes Classification with Non-Linearity

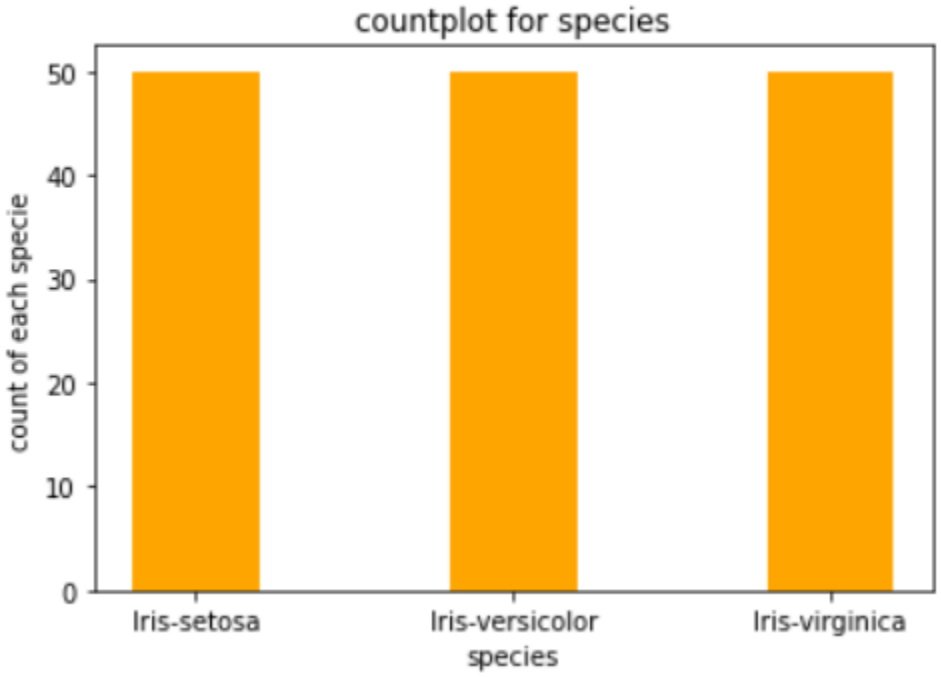
Akriti Gupta
B21AI005

Question 01: Gaussian Bayes Classifier

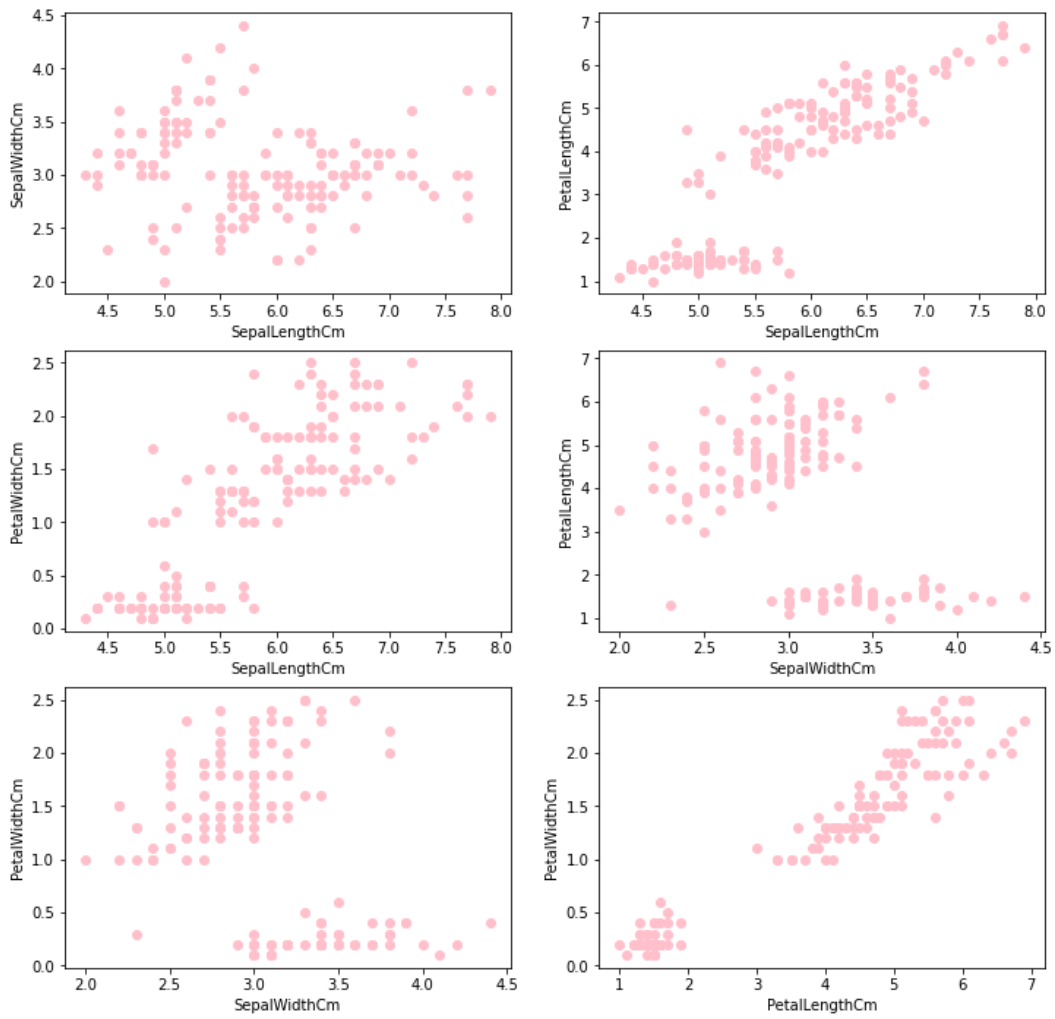
Preprocess and Exploratory analysis:

We first check if there are any nan/null values present in the dataframe. As no null values are there so we further check the data type fo each column. The ‘species’ column is of object datatype so we encode it by assigning integer values to each unique specie.

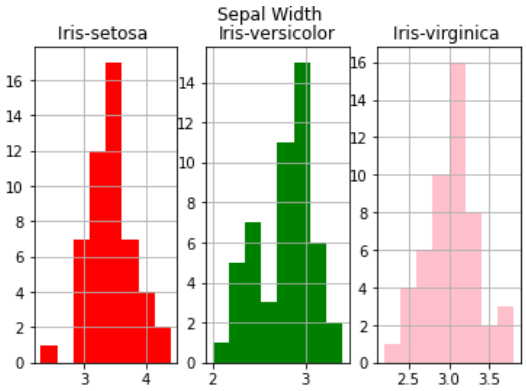
Exploratory analysis is done by visualizing the data:
We first plot a countplot for species by using barplot which is as follows:

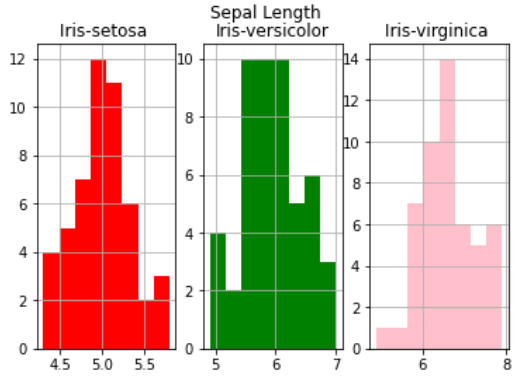


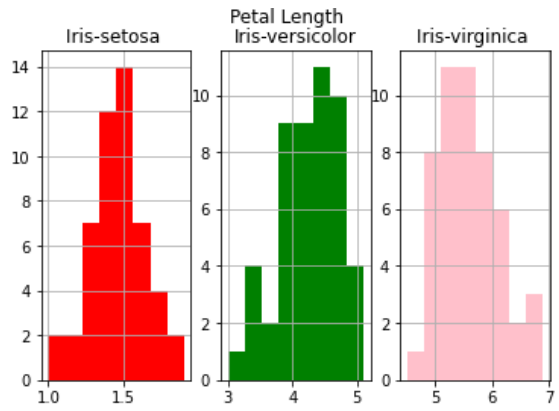
Next, we plot 6 scatterplots with each pair of features:

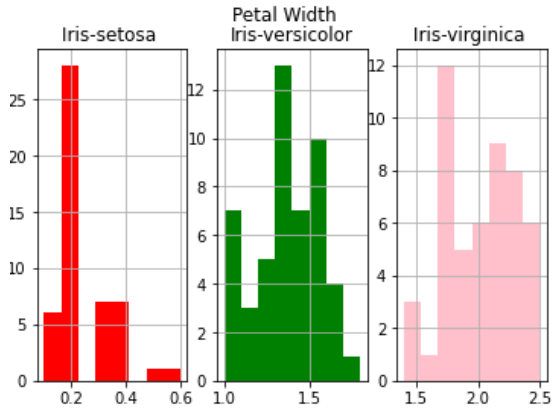


Lastly, we plot histograms for each feature for different species:









Splitting the data :

Now, we split the data into training and testing dataset in the ratio 70:30.

Subpart 1 and 2:

For these subparts we made class Gaussian with the following attributes:

- case(which case we are implementing)
- df(the dataframe we are working with, initialised to None)

The following are the methods we implement in our class:

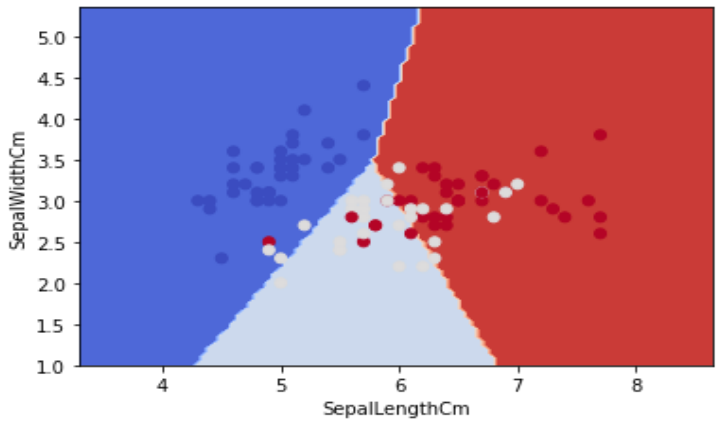
- train(takes x_train and y_train as input and sets the df of our model as the training data it obtains)
- g_i_x(takes class label and data point/feature vector of a single datapoint as input and depending on the case of our model computes the covariance matrix and also computes the mean vector and prior probability wrt the class label passed and returns discriminant function wrt class_label passed)
- predict(takes a single datapoint as input and in a list stores the discriminant function wrt all the class labels possible and then returns the class label wrt which discriminant function is maximum)
- test(takes x_test and y_test as input and wrt all the points in x_test predicts what its output will be and appends in y_pred array and also returns the accuracy)

- plot_decision_boundary(takes the 2 features wrt which decision boundary is to be plotted anf plots the decision boundary with points scattered on it)

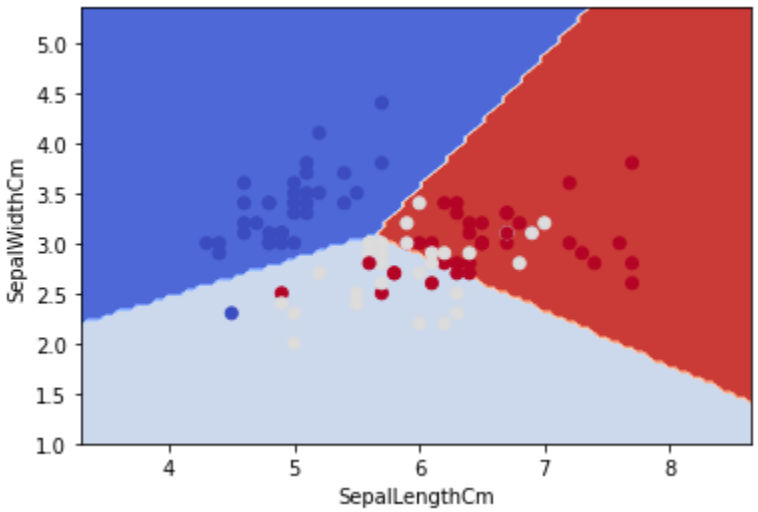
Subpart 3:

We train the model wrt the different cases we implemented in constructor and obtain the following accuracies and decision boundary:
For case 1:
Accuracy is 95.556 %

Decision Boundary is:

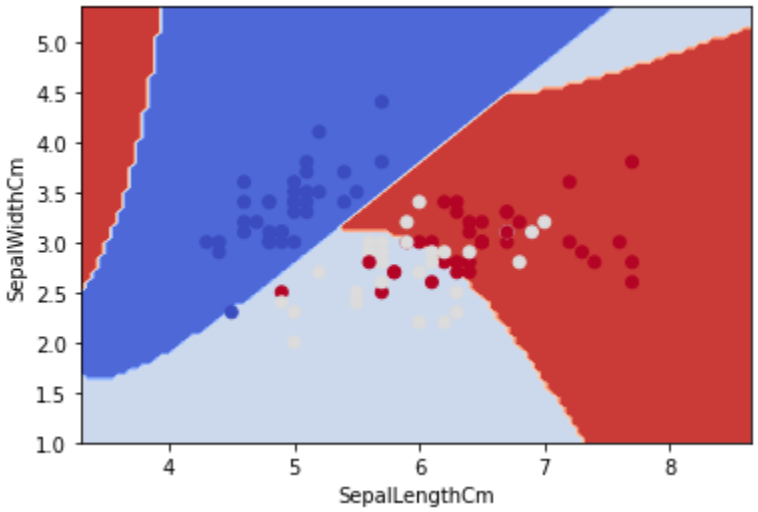


For case 2:
Accuracy is 80%
Decision Boundary is:



For case 3:
Accuracy is 100%

Decision Boundary is:



So, we see that w.r.t case 3 the model seems to fit the data best as the decision boundary is the best and the accuracy also comes out to be maximum thus we can infer that the model performs best w.r.t case 3.

Subpart 4:

We perform cross validation and obtain the following accuracies wrt all the training-testing pairs:

For case 1

```
[90.47619047619048, 85.71428571428571, 85.71428571428571,
95.23809523809523, 100.0]
```

For case 2

```
[66.66666666666666, 100.0, 90.47619047619048, 76.19047619047619,
85.71428571428571]
```

For case 3

```
[100.0, 95.23809523809523, 95.23809523809523, 100.0, 100.0]
```

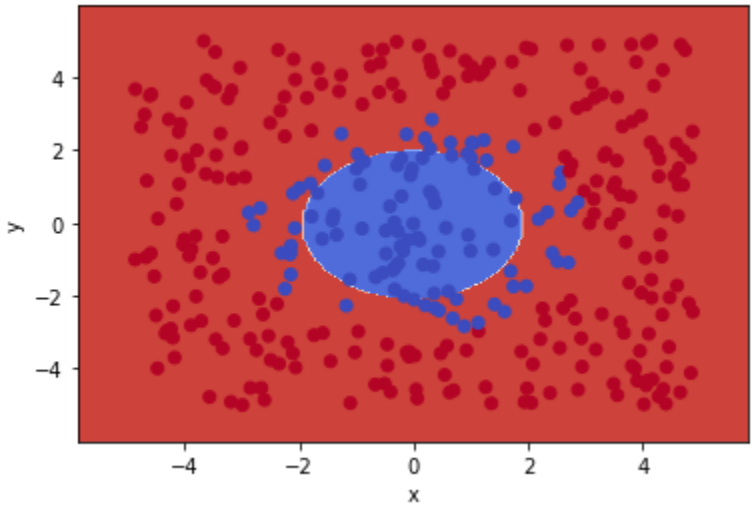
So, again we can say clearly the cross validation scores also concludes that case 3 performs the best as it has the maximum number of sets having 100% accuracy in cross validation sets.

Subpart 5:

We now generate 500 samples of points inside the circle x^2+y^2=25 and assign the outputs according to the conditions given in the question.

Now we train our model w.r.t case 3 and get accuracy as 83.33%

Now we plot decision boundary and obtain it as:

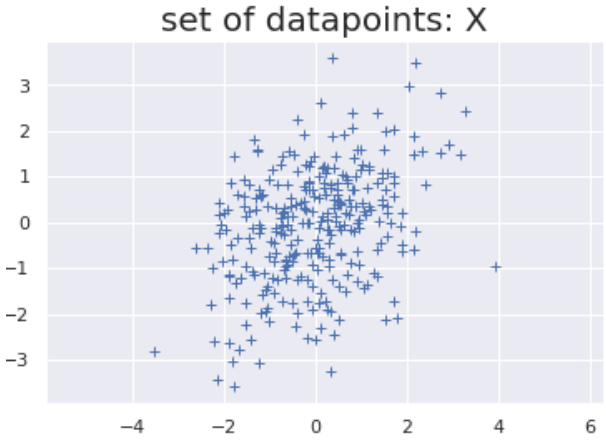


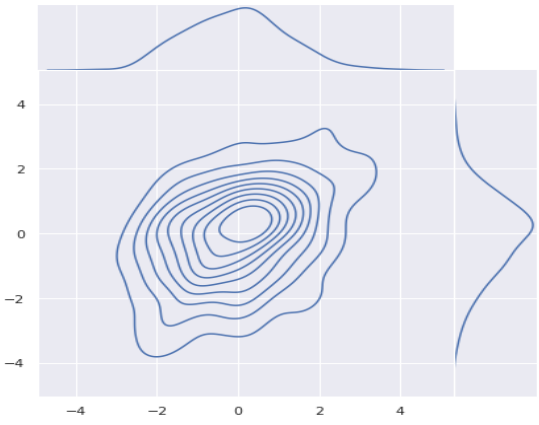
Thus, we can say even if the data is not linearly separable, the gaussian bayes fits the data very well and obtains a non linear decision boundary which as we can see fits the data well and predicts the output accurately.

Question 02: Mahalanobis Distance

Sampling random points:

We firstly write the covariance matrix and mean as given in the question and then sample 300 random points from the multivariate normal distribution.
Next we plot the scatterplot and probability distribution plot for the set of datapoints:





Subpart 1:

Now, we generate the covariance matrix of the above sampled datapoints by defining a cov(x,y) fuction and using it to calculate the covariance matrix that is:

```
[[1.44581137 0.54555525]
 [0.54555525 1.56025964]]
```

Next, we find the eigenvalues and eigenvectors of the above generated matrix with the help of numpy inbuilt method which are as follows:

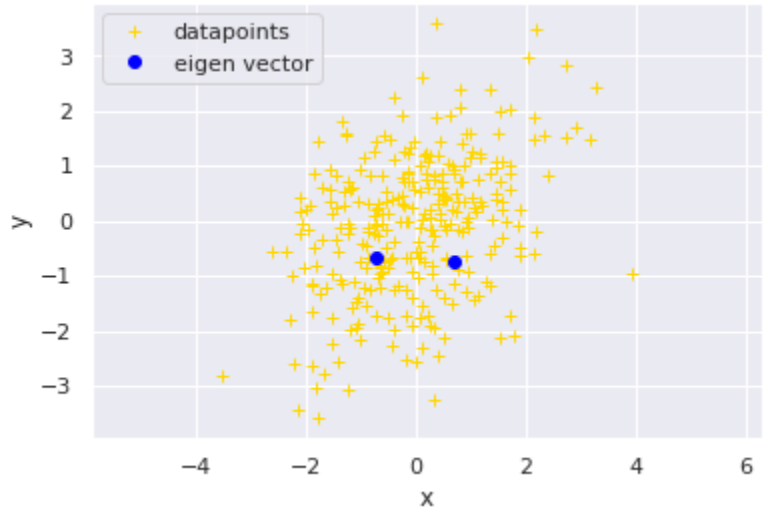
Eigen values:

```
[0.9544873  2.05158371]
```

Eigen vectors:

```
[[-0.74307444 -0.66920877]
 [ 0.66920877 -0.74307444]]
```

Lastly, the eigenvectors are plotted on the scatterplot by being superimposed on the datapoints X.



Subpart 2:

Now, we perform the transformation Y= (Σs^(−1/2))X by first calculating Σs^(−1/2) by using scipy.linalg.fractional_matrix_power and then transforming datapoints X to Y.

Next, we calculate the covariance matrix of Y datapoints by using the function cov(x,y) defined by us earlier.

Covariance matrix of Y datapoints:

```
[[ 1.00000000e+00 -4.75279422e-17]
 [-4.75279422e-17  1.00000000e+00]]
```

The obtained covariance matrix is similar to an identity matrix.

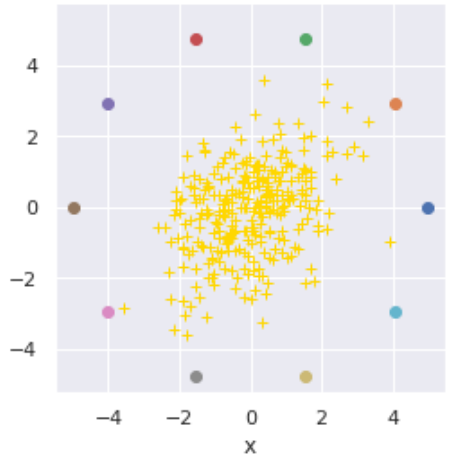
Also, the standard deviation is equal to unity and mean is given as [0,0].
Hence the transformation has resulted into the dataframe being standardized.

Subpart 3:

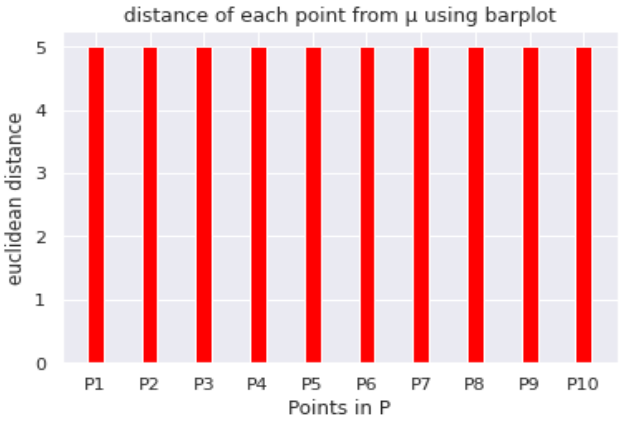
We uniformly sample 10 points on the curve x^2 + y^2 =25 by using a defining a function sampling(). These set of points are called P.

x_P: `[5.0, 4.045084971874737, 1.5450849718747373, -1.5450849718747373, -4.045084971874737, -5.0, -4.045084971874737, -1.5450849718747373, 1.5450849718747373, 4.045084971874737],`
y_P: `[0.0, 2.9389262614623655, 4.755282581475767, 4.755282581475767, 2.9389262614623655, 0.0, -2.9389262614623655, -4.755282581475767, -4.755282581475767, -2.9389262614623655])`

Now, we plot P along with dapoints X. The plot is as follows:



Each point is of different color.
The eucledian distance of each point from μ is 5 and the barplot is as follows:

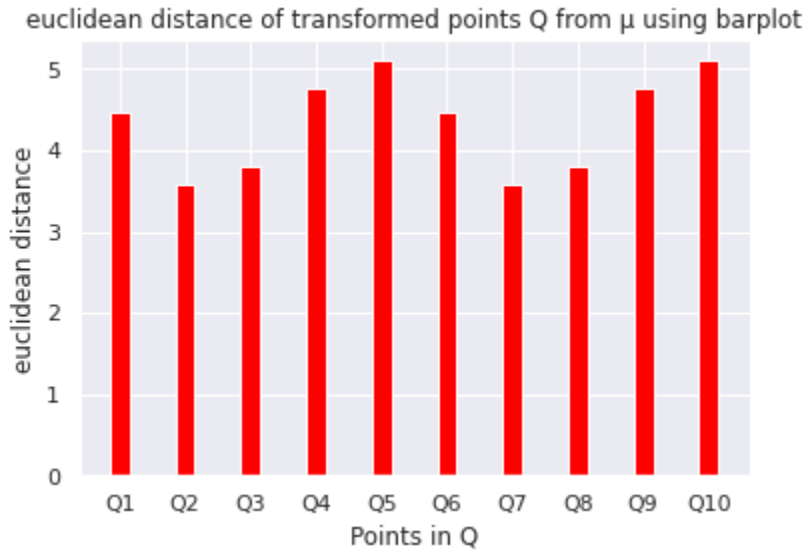


Subpart 4:

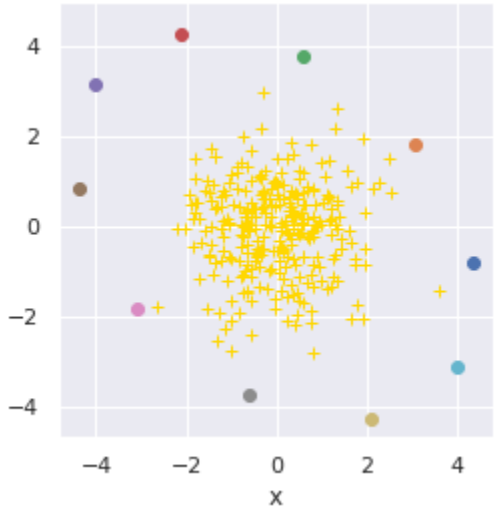
Now, we perform transformation on P datapoints to convert it to Q datapoints by Q= (Σs^(−1/2))P.
Next, we calculate the eucledian distance of transformed datapoints Q from μ which come out to be as:

```
[4.463121847775779, 3.576390053362857, 3.8084040667646972,
4.7635865460510916, 5.102815808489989, 4.463121847775779,
3.576390053362857, 3.8084040667646972, 4.7635865460510916,
5.102815808489989]
```

The barplot for the same is as follows:



We also plot a scatter plot by plotting datapoints Q along with datapoints Y.



The euclidean distance before the transformations of the point P from mean was equal for all i.e. 5 whereas after the transformation the value is different for different points. This is because after transformation, euclidean distance becomes mahalanobis distance which takes correlation between variables into consideration unlike euclidean distance which measures length of line segment between two points without taking the distribution under account. The points which are near to the distribution have less value of mahalanobis distance and the points which are far away have higher value. We get an elliptical shape of points after transformation.