# **K-Means Clustering for Iris dataset**

**Created By:**

Arijit Ganguly

1001871460

# Problem Statement

K-Means Clustering is an unsupervised learning algorithm. It is very useful in finding natural groups in the dataset. For our problem, we are using K-Means on the Iris data set for clustering.

Main challenges faced on the K-Means dataset is that the clusters can be formed anywhere depending where the centroids are picked. Hence, picking the correct centroids is a crucial part of the algorithm. The second challenge that arrives on K-Means is the metric used for calculating the distances between the sample vectors. Different vectors can lead to different end results.

# Data

The Iris Data set contains 5 columns – Sepal Length, Sepal Width, Petal Length, Petal Width and Species.

We use the first 4 columns as our features – Sepal length, Sepal width, Petal length and Petal width. All the columns for features are float values. The values in these columns make up the A matrix.

Last column is going to be used as the labels. It states the species of the plant which have the corresponding features. The data set contains 3 species – Iris setosa, Iris virginica and Iris versicolor.

For purposes of

**Data Distribution based on species:**

|  |  |
| --- | --- |
| **Species** | **Number of Records** |
| Iris Setosa | 50 |
| Iris Virginica | 50 |
| Iris Versicolor | 50 |

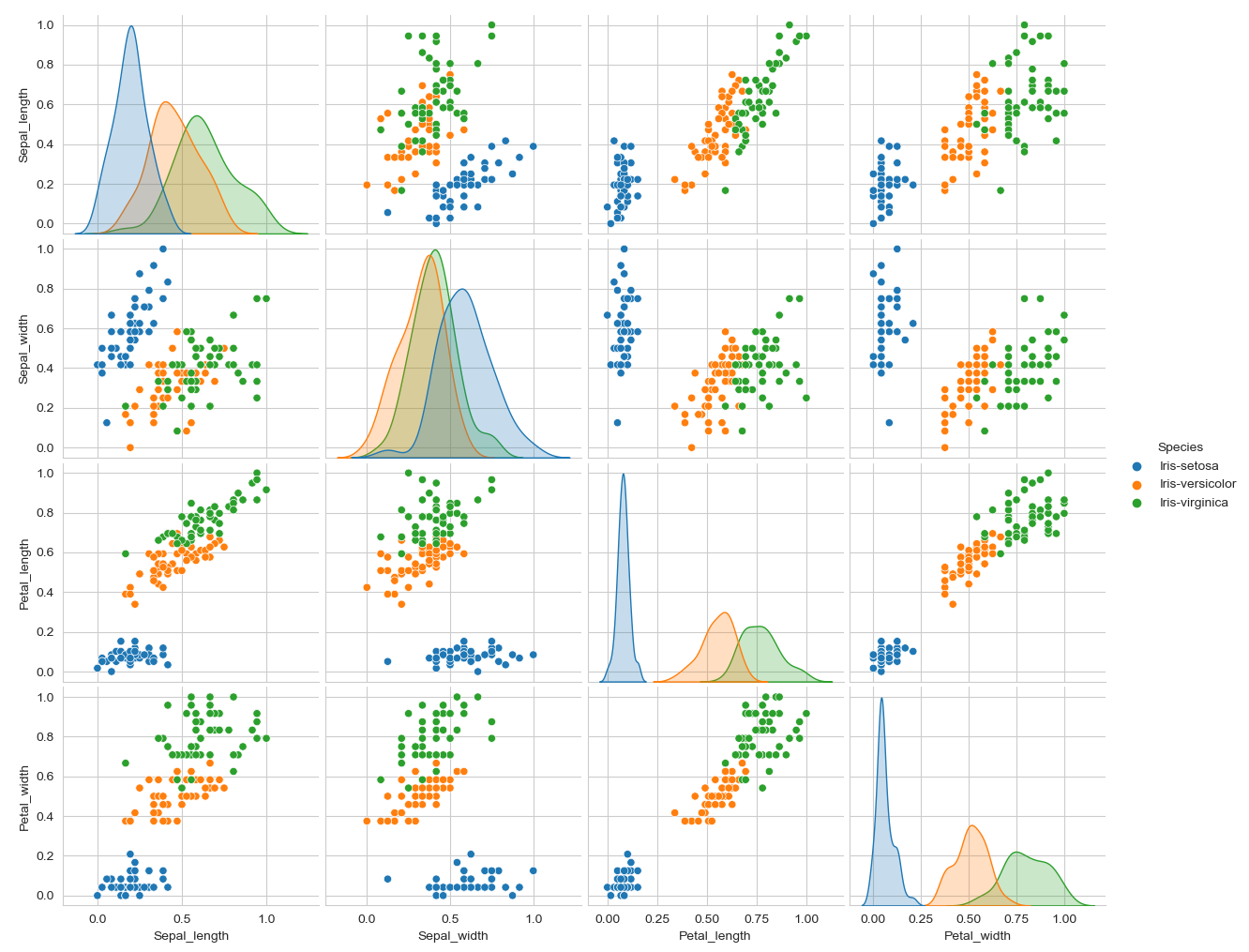
**Information on the data received from method pandas.info()**

|  |  |  |  |
| --- | --- | --- | --- |
| **Index number** | **Column** | **Non-Null Count** | **Data Type** |
| 0 | Sepal Length | 150 | Float64 |
| 1 | Sepal Width | 150 | Float64 |
| 2 | Petal Length | 150 | Float64 |
| 3 | Petal Width | 150 | Float64 |
| 4 | Species | 150 | category |
| 5 | Species\_cat | 150 | Int8 |

**Data Description (Received from method pandas.describe())**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Sepal Length | Sepal Width | Petal Length | Petal Width |
| count | 150.000000 | 150.000000 | 150.000000 | 150.000000 |
| mean | 0.428704 | 0.439167 | 0.467571 | 0.457778 |
| std | 0.230018 | 0.180664 | 0.299054 | 0.317984 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.222222 | 0.333333 | 0.101695 | 0.083333 |
| 50% | 0.416667 | 0.416667 | 0.567797 | 0.500000 |
| 75% | 0.583333 | 0.541667 | 0.694915 | 0.708333 |
| max | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

**Data visualization:**



# Strategy

The strategy used for K-Means is to initialize centroids using the first 3 records of the X values. Then we calculate the distance from each point to the centroid and mark that record with the closest centroid value. Then re-create the centroid by finding the mean of each cluster. Then re calculate the distances and re- create the clusters based on the closest distances. The distances changes will provide the error value, if the error value goes to zero then the clusters have not changed. This becomes the final the clusters.

We repeat this process with different metrics to calculate the distances from the centroids to sample points. For this project, I am using:

1. Euclidean Distance
2. Manhattan Distance

## Method Used:

### Fit Method

K-Means fit method is only storing the data in the array. For my code, I am executing this inside the \_\_init\_\_ function of the K\_Means\_Cluster class.

### Predict Method

This method initializes the centroids and uses the strategy to find the clusters to be used for clustering the data.

### Predict\_all Method

This method executes the predict method for the 2 different metrics and compares the accuracy of the clusters formed.

### Data Visualization Method

This method builds the box plots for the each of the feature to compare the actual data and the predicted data. This method also builds the cross-tab data to compare how many records are correctly clustered together.

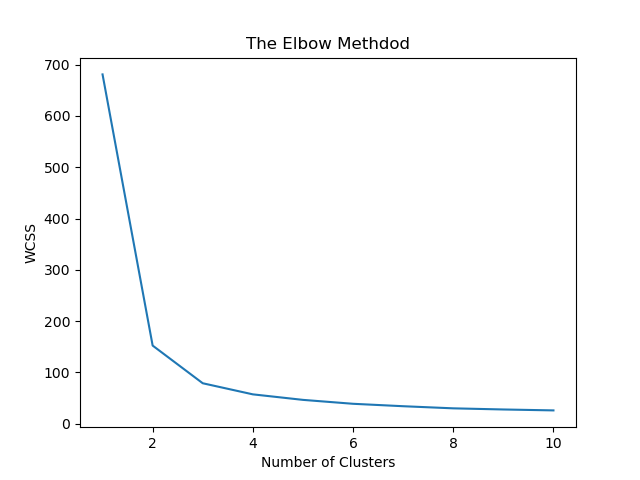
### Check Accuracy Method

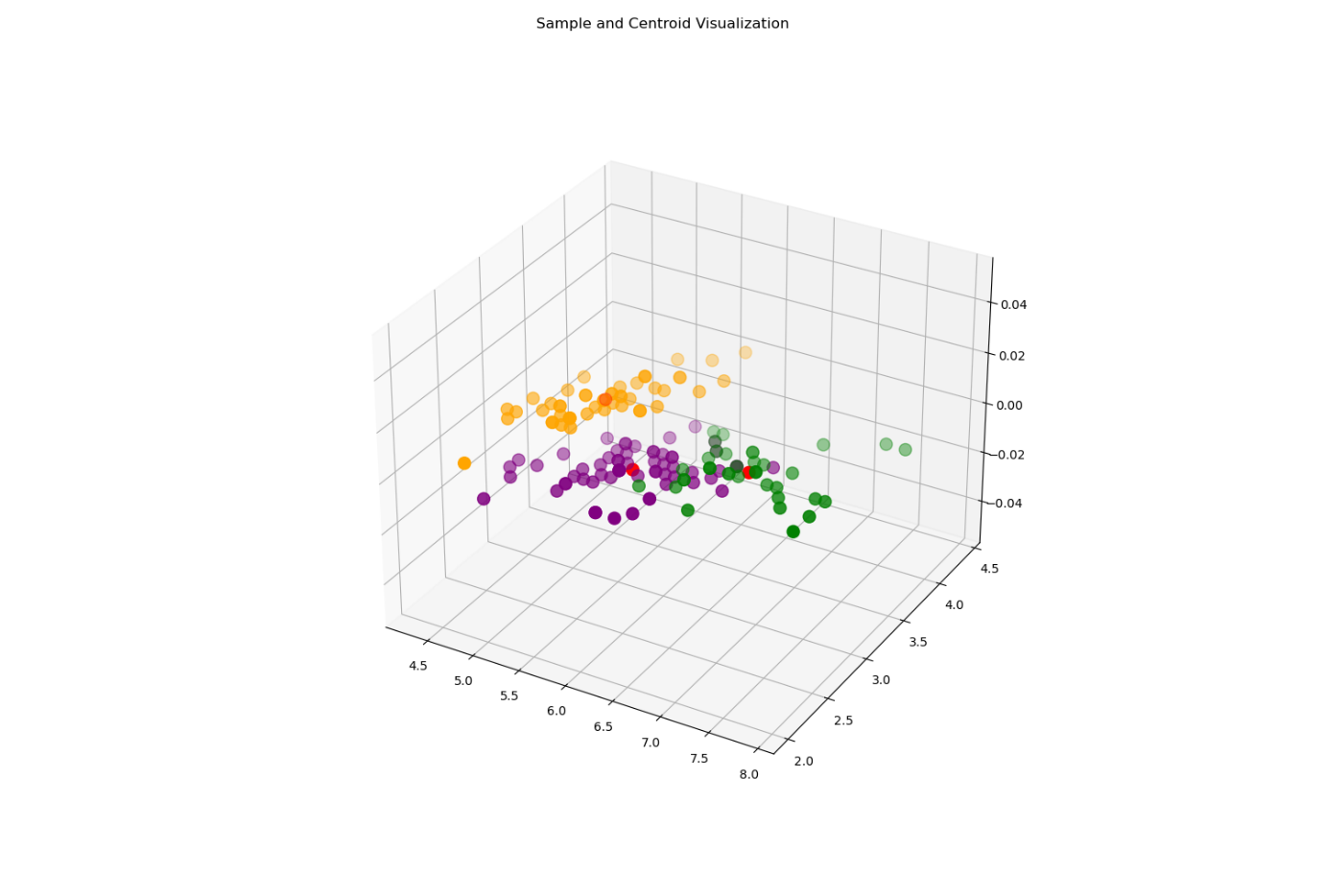
Based on the metric we used, the clusters change the relation. Hence calculate the count of records that have been correctly labelled

# Results

We use the Elbow method to find the correct number of clusters to be used. Based on the graph, we can use the 3 or 4 clusters can be used to correctly build the clusters of the data.

Additionally, visualize the scatter plot and the centroids.



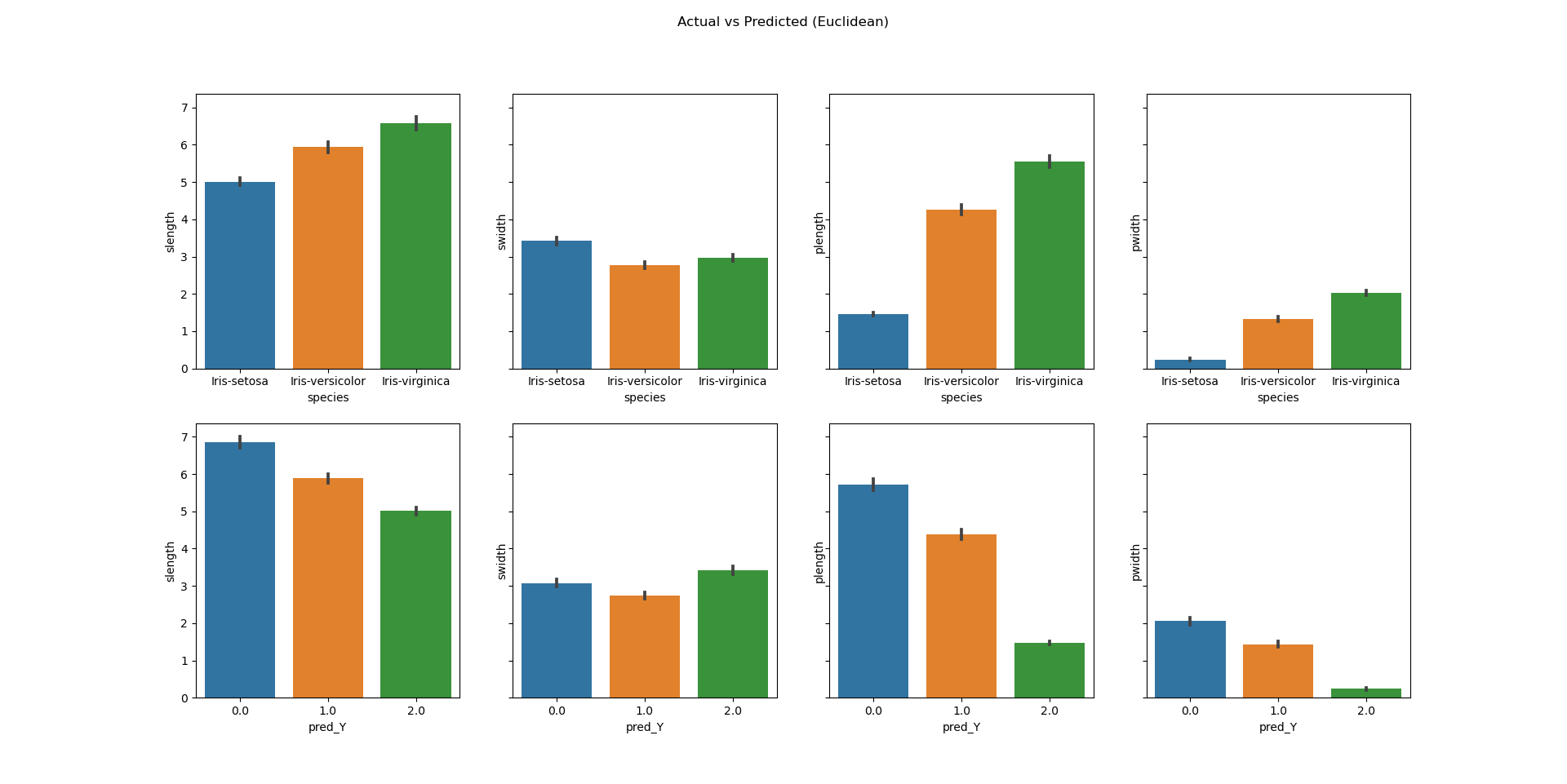


Using the Euclidean distance, we get the results mentioned below. We can see the relation of the clusters and the actual labels.

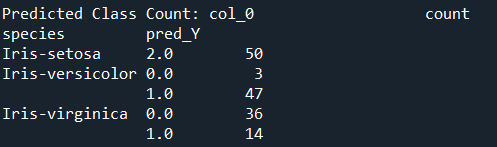
Iris-setosa -> Cluster 2

Iris-virginica-> Cluster 0

Iris-versicolor-> Cluster 1



Building the crosstab gives us information on the correctly predicted data.



This can be used to get the accuracy of the algorithm using the check-accuracy function to get 88. 667 % accuracy.

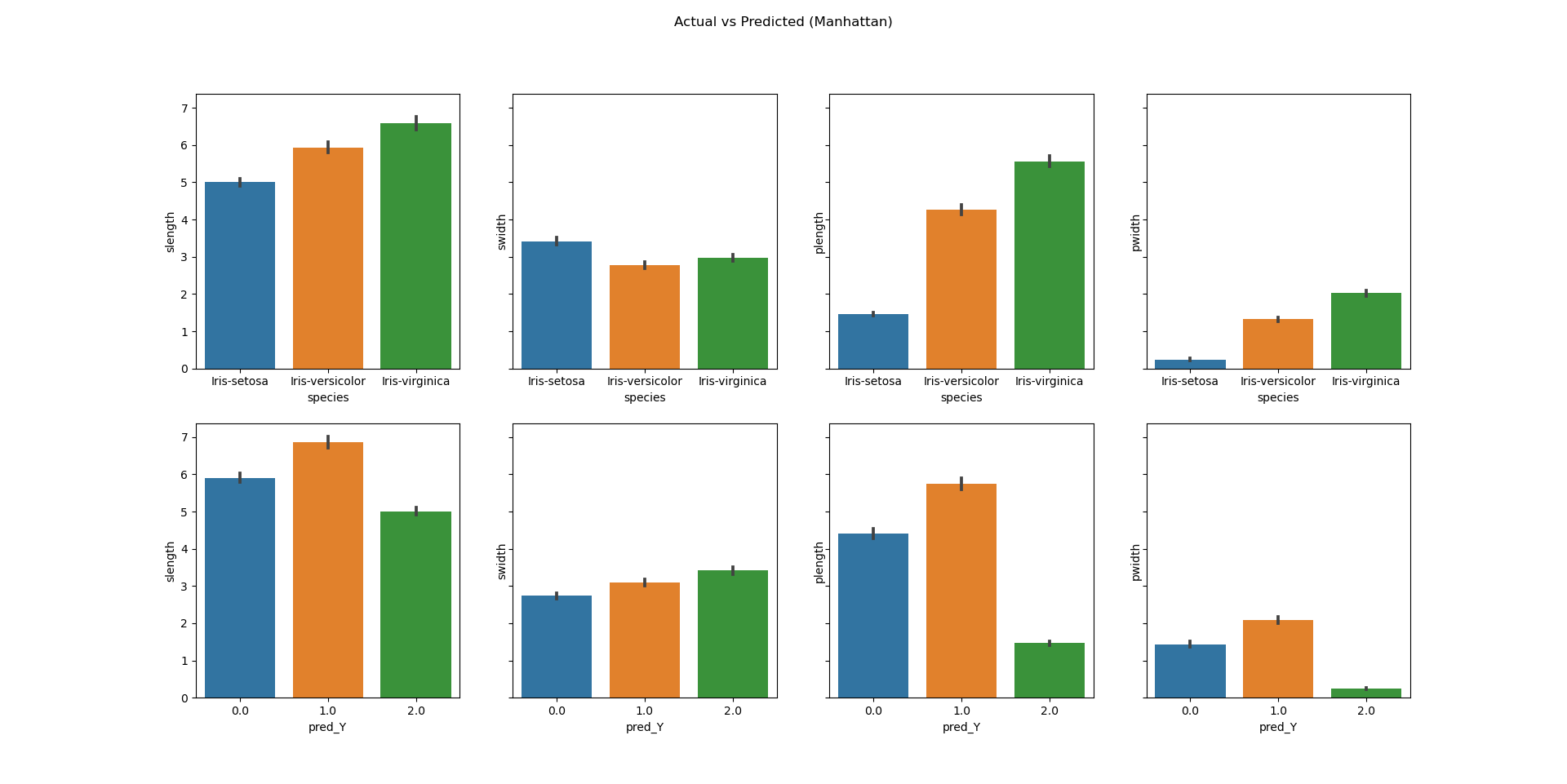


Then we run the same for the Manhattan distance to get the relation of clusters and actual labels.

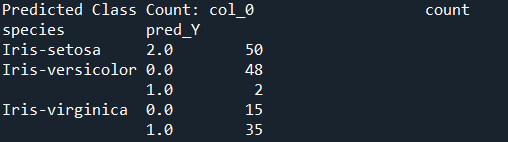
Iris-setosa -> Cluster 2

Iris-virginica-> Cluster 1

Iris-versicolor-> Cluster 0



Building the crosstab for Manhattan distance as well for getting the correctly predicted data



Overall Accuracy using the Manhattan distance to get accuracy of 88.667%



# References

## My github repository for the code and project report

[Github](https://github.com/agx01/iris_kmeans)

## For Iris dataset visualization and K-means reference materials

<https://www.kaggle.com/khotijahs1/k-means-clustering-of-iris-dataset/notebook>

## Data Visualization for subplots

<https://stackoverflow.com/questions/27426668/row-titles-for-matplotlib-subplot>