Pattern Recognition Assignment

Siddhartha Singh Bhadauriya(2k18/se/123)

Problem Statement

In this assignment, the problem is basically to understand the concept of K-means algorithm and as we know that finding the value of K in K-means algorithm i snot an easy task, so we will have to use elbow method to find optimal number of clusters, we can divide our dataset into.

Objectives:

(1) Load the dataset and perform splitting into training and validation sets with a 70:30 ratio.

(2) Implement the K Means algorithm using sklearn. You need to find the optimal number of clusters using the elbow method. Plot the error vs number of clusters graph while using the elbow method. Report the optimal number of clusters found.

(3) Use Scatter plot to visualize the dataset to depict the clusters formed (optimal).

(4) Report the training and the validation accuracy. Comment on the accuracy obtained for both the sets.

Solution:

Dataset used: We will use Iris dataset to perform this k-means algorithm.

The number of rows in the data set is 150, and the number of columns are 5. First 4 columns contain features and the last column contain the label associated with them.

data : *Bunch*

Dictionary-like object, with the following attributes.

data : ***{ndarray, dataframe} of shape (150, 4)***

The data matrix. If `as_frame=True`, `data` will be a pandas DataFrame.

target: {ndarray, Series} of shape (150,)

The classification target. If `as_frame=True`, `target` will be a pandas Series.

feature_names: list

The names of the dataset columns.

target_names: list

The names of target classes.

frame: DataFrame of shape (150, 5)

Only present when `as_frame=True`. DataFrame with `data` and `target`.

New in version 0.23.

DESCR: str

The full description of the dataset.

```
In [19]: print (iris.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
```

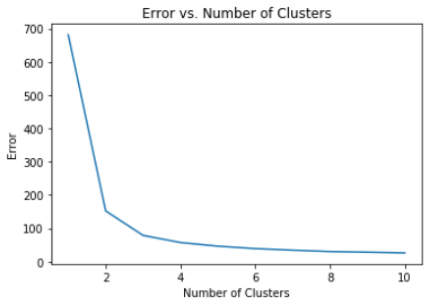
Elbow Method for determining number of clusters:

Here, currently we don't know the value of what K is to be taken, so we started trying with K=5, which for sure will divide the dataset into clusters, but what we want is to use the Elbow method, to get the optimal value of K.

Optimal number of clusters can be get as follows:

1. Compute clustering algorithm (e.g., k-means clustering) for different values of k. For instance, by varying k from 1 to 10 clusters.
2. For each k, calculate the total within-cluster sum of square (wss).
3. Plot the curve of wss according to the number of clusters k.
4. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.

```
In [6]: Error= []
        for i in range(1,11):
            kmeans = KMeans(n_clusters = i).fit(iris.data)
            kmeans = kmeans.fit(iris.data)
            Error.append(kmeans.inertia_)
        plt.plot(range(1,11), Error)
        plt.title("Error vs. Number of Clusters")
        plt.xlabel("Number of Clusters")
        plt.ylabel("Error")
        plt.show()
```



As we can clearly see from this graph, that between 2 and 4, we started noticing a linear change in error with respect to the increase in number of clusters so we will basically chose number of clusters to be 3.

Splitting dataset into training set and validation set as well as training:

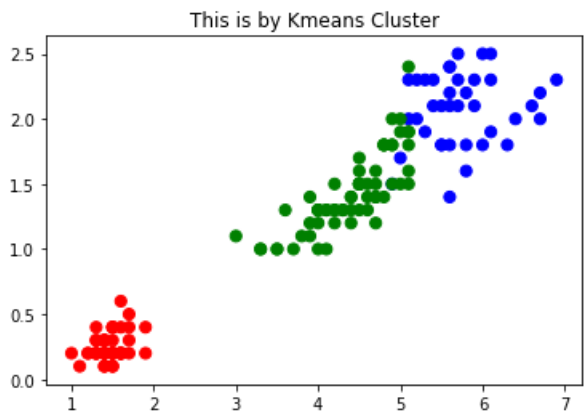
Finding the number of clusters as 3, we need to split our dataset into a 70:30 ratio for training and validation purposes.

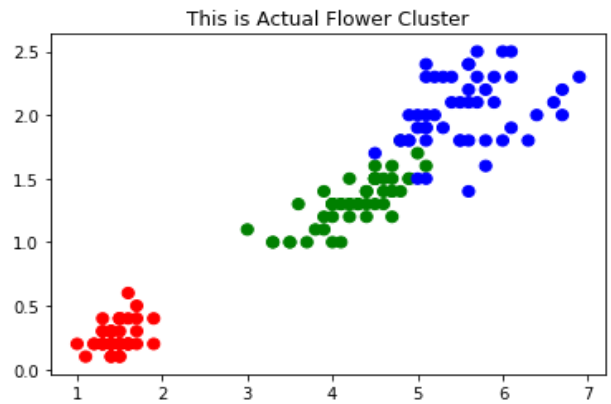
After that, we will simply use scikit learn to perform K-means algorithm in training dataset as well as validation dataset.

```
In [14]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target, test_size=0.30)

In [15]: # Doing training for traing set first
         kmeans = KMeans(n_clusters = 3)
         y_trainpredicted = kmeans.fit_predict(X_train)
         print(y_trainpredicted)

         [2 2 2 1 0 0 2 1 1 1 1 1 1 0 1 2 2 1 2 1 0 0 1 1 2 1 1 1 0 1 1 0 1 2 0 2 1
          1 1 1 0 0 2 2 1 1 1 0 0 1 0 1 2 1 0 1 0 2 0 0 1 1 1 0 0 1 0 1 0 0 1 2 0 2
          2 2 0 1 1 1 2 1 0 0 0 2 0 1 1 2 0 1 1 2 0 0 1 1 0 0 2 1 2 2 2]

In [16]: # Now doing for Validation set of 30%
         kmeans = KMeans(n_clusters = 3)
         y_testpredicted = kmeans.fit_predict(X_test)
         print(y_testpredicted)

         [0 0 0 1 1 1 0 1 0 2 2 2 1 2 0 0 2 0 2 1 2 2 0 2 0 1 1 2 1 1 0 2 0 0 2 1 0
          2 2 2 1 2 2 0 0]
```

Scatter plot for actual dataset vs. dataset after clustering:

```
In [19]: iris_df = pd.DataFrame(iris.data)
         iris_df.columns = ['sepal_length','sepal_width','petal_length','petal_width']
         y1 = pd.DataFrame(iris.target)
         y1.columns =['Targets']
         color_theme = np.array(['red','green','blue'])

         plt.scatter(x=iris_df.petal_length, y=iris_df.petal_width, c= color_theme[iris.target],s=50)
         plt.title ("This is Actual Flower Cluster")
```





Accuracy:

Accuracies for both Training dataset and Validation Dataset with confusion matrix:

Printing accuracy for training set

print(accuracy_score(y_train,y_trainpredicted)) confusion_matrix(y_train,y_trainpredicted)

In [17]:
```
print(accuracy_score(y_train,y_trainpredicted))
confusion_matrix(y_train,y_trainpredicted)
```

```
0.8857142857142857
```

Out[17]:
```
array([[33,  0,  0],
       [ 0, 35,  1],
       [ 0, 11, 25]], dtype=int64)
```

Printing accuracy for validation now ¶

In [18]:
```

print(accuracy_score(y_test,y_testpredicted))
confusion_matrix(y_test,y_testpredicted)
```

```
0.022222222222222223
```

Out[18]:
```
array([[ 0,  0, 17],
       [13,  1,  0],
       [ 3, 11,  0]], dtype=int64)
```