**Project** 2 **Report** **Shivam Pandey** (160010003)
CS 386 September 7, 2018

## 1 Visualize the Iris dataset using scatterplots

Commencement of this project entails exploratory data analysis over the Iris Dataset by using python libraries such as sklearn, numpy, pandas, seaborn and matplotlib. Iris dataset provides information about three species of *Iris*, namely, Setosa, Versicolor and Virginica, in the form of four features, viz. sepal length, sepal width, petal length and petal width. It comprises of 150 data points, each species assigned to 50 datapoints. Firstly, we contemplate the matrix plot (pairplot) obtained by plotting each of the four features against themselves. We diagnose a strong correlation between petal length , petal width and sepal length from correlation matrix Also, we perceive that the setosa is clustered farther from the clusters of the versicolor and virginica in every scatter plot. In the scatter plot of sepal length against sepal width, the clusters of versicolor and virginica are highly intermixed with no distinguishable boundary. Summarizing over each scatter plot, versicolor and virginica are closer to each other compared to setosa, with not much distinguishable boundary.

## 2 Visualize the Iris dataset using boxplots and histograms

Upon scrutinizing the boxplots, we diagnose a big separation of petal length and petal width of setosa from those of other two species. Also, in the case of sepal length, petal length and petal width, as we go from setosa to versicolor to virginica, the features reveal an increment. In the case of sepal width, we discern significant amount of overlapping amongst the three species.Also, *histogram of sepal width* has the shape of a Gaussian distribution(or Normal distribution) function, i.e. it has a bell-shaped curve. Elseways, histograms of petal length and petal width don't have the characteristic of gaussian distribution.

## 3 Visualize the Iris dataset using 3D-plots

To visualize the 3D scatter plot, we chose the 3 features as *petal length, petal width* and *sepal width*, as this triplet gives the best distinguishable 3D clusters, putting together other possible triplets of features. Also, we detect that the setosa is clustered in a smaller volume of space(*smaller variance*) and farther away from other two clusters. Versicolor and virginica are also separately clustered in space, but with less distinguishable boundary.

## 4 Analyses of the $k$-means algorithm

By EDA, we decided the value of 'K' to be 3 in the run of K-Means algorithm. Upon execution of the k-means algorithm, we obtained a scatter plot with 3 clusters similar to that of the original iris dataset. Since, the centroids are randomly chosen, we end up in different number of iterations the each time we run k-means algorithm. Generally, we detect that the algorithm takes about 3-14 iterations to complete. We can also decide upon several stopping criteria, viz. when the value of J(Cost function) is not much changing , when the cluster centroids aren't updating or we can also put some maximum bound on the number of iterations.

## 5 Compare algorithms $k$-means and agglomerative clustering

I have come up with a self devised *Negative Performance Metric*. I define Negative Performance Metric as *"Percentage of data points wrongly marked by the run of respective algorithm"*. *Lesser* is the "Negative Performance Metric", *Better* is the run of the respective algorithm (k-Means/agglomerative). Variation in Negative Performance Metric (Error Percentage) :-

- K-Means Algorithm : 10.66 %
- Agglomerative Clustering Algorithm with linkage 'average' : 9.33 %
- Agglomerative Clustering Algorithm with linkage 'ward' : 10.66 %
- Agglomerative Clustering Algorithm with linkage 'complete' : 16.0 %

We discern that when we use the 'average' linkage strategy in Agglomerative clustering, i.e. combining two clusters with minimum centroid distance, we get the least error percentage,which is even better than K-means algorithm. Incase we use 'wards' or the 'complete' linkage strategy, then error percentage is more than k-means clustering. Difference in Execution Time :-

- K-means Clustering Algorithm : 0.032994 seconds
- Agglomerative Clustering Algorithm : 0.001739 seconds

We spot that Agglomerative Clustering is "about 19 times faster" than K-Means Clustering; although the speed of agglomerative approach over k-means approach might vary depending upon the initialization strategy or the linkage strategy.

## 6 Select $K$

1. **Elbow Method** : The Elbow method looks at the total within-cluster sum of square(WSS) as a function of the number of clusters. One should choose a number of clusters so that adding another cluster does not improve much better the total WSS. Compute clustering algorithm for different values of k. For instance, by varying k from 1 to 10 clusters. For each 'k', calculate the total WSS. Plot the curve of WSS according to the number of clusters k. The location of a bend (knee) in the plot is generally considered as an indicator of the appropriate number of clusters.
2. **Average Silhouette Method** : Briefly, it measures the quality of a clustering. That is, it determines how well each object lies within its cluster. A high average silhouette width hindicates a good clustering. Compute clustering algorithm for different values of k. For instance, by varying k from 1 to 10 clusters. For each k, calculate the average silhouette of observations. Plot the curve of avg.sil according to the number of clusters k. The location of the maximum is considered as the appropriate number of clusters.
3. **K = Squareroot(n/2)** : A quick (and rough) method is to take the square root of the number of data points(n) divided by two, and set that as the number of clusters , i.e. K = squareroot(n/2). But this is just a rough way of doing it.
4. **By EDA** : By performing Exploratory Data Analysis also, we can decide upon choosing the number of clusters.
5. **By Dendrogram**(*Specifically for Agglomerative Clustering*) : We can cut the dendrogram at a certain height (distance threshold) by diagnosing the density of clusters below the cut, and get the number of desired clusters.

## Attachments

agglomerativeClusteringScikit.py, boxplots.py, correlationMatrix.py, histograms.py, iris3d.py, kmeans.py, kmeansClusteringScikit.py, pairplot.py, petalLengthVsSepalWidth.py, sepalLength-VsSepalWidth.py