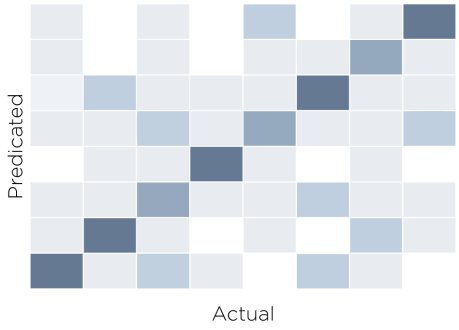
PRO-C118

WhiteHat Jr
Live Online Coding for Kids

Clustering



What is our GOAL for this MODULE?

The goal of this module is to use the k means algorithm for cluster analysis or clustering.

What did we ACHIEVE in the class TODAY?

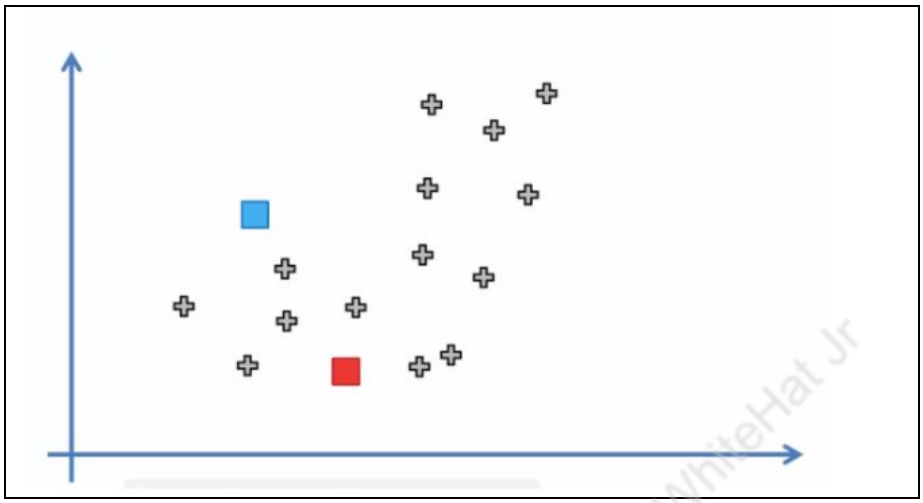
We used the elbow method to find the best value for K and using K means algorithm found the proper cluster points.

Which CONCEPTS/CODING BLOCKS did we cover today?

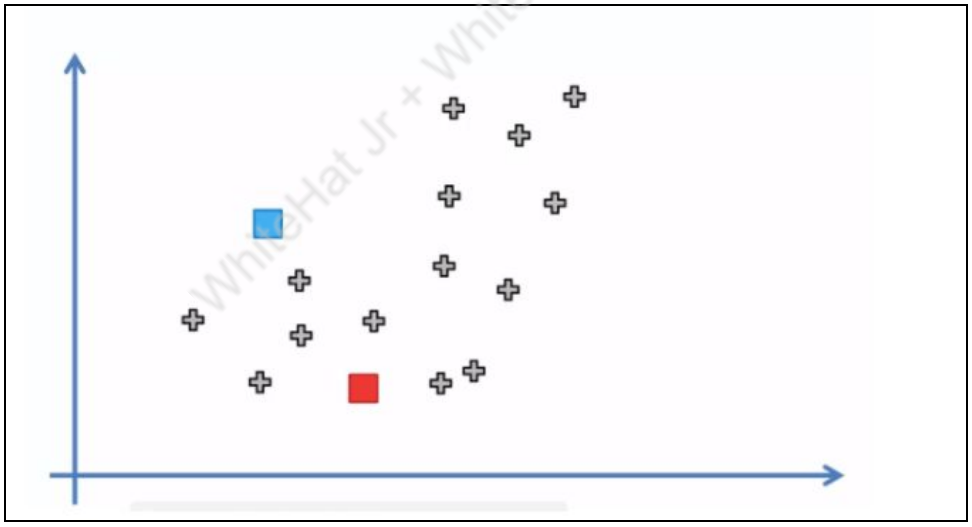
- Elbow method
- K-means algorithm

How did we DO the activities?

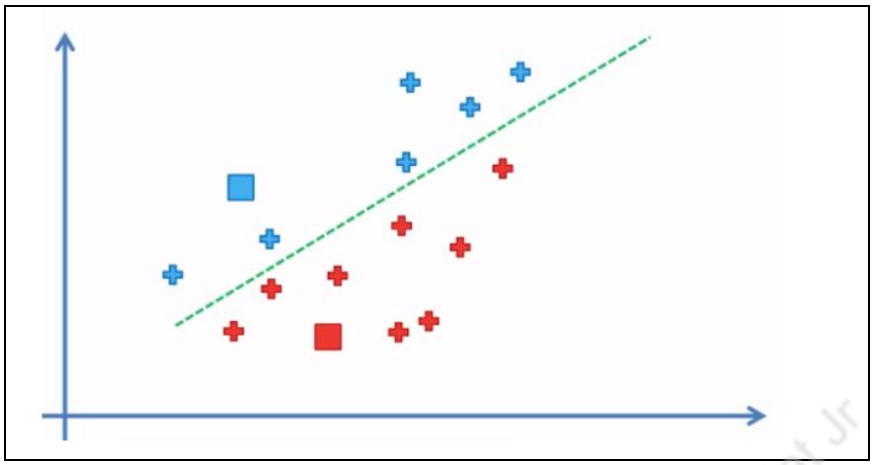
1. We explored the steps to find the K-means algorithm.
    - **Choose the number K of clusters**



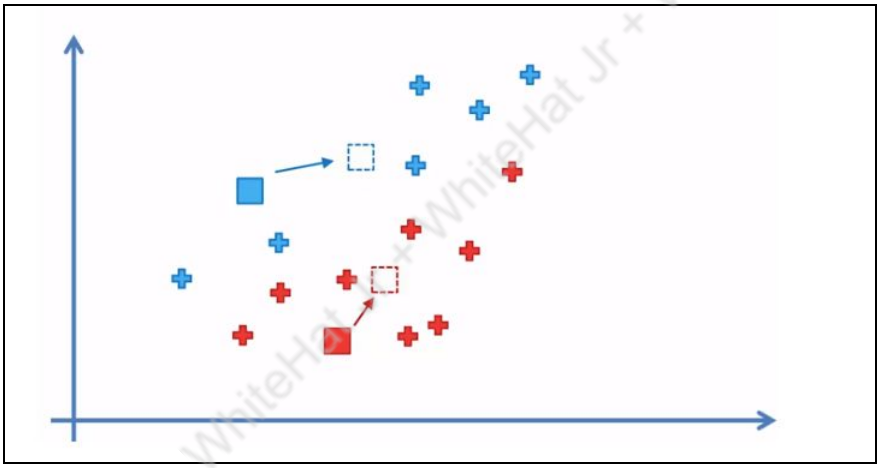
- **Select randomly the center points (centroids) for the K clusters (2 in this case)**



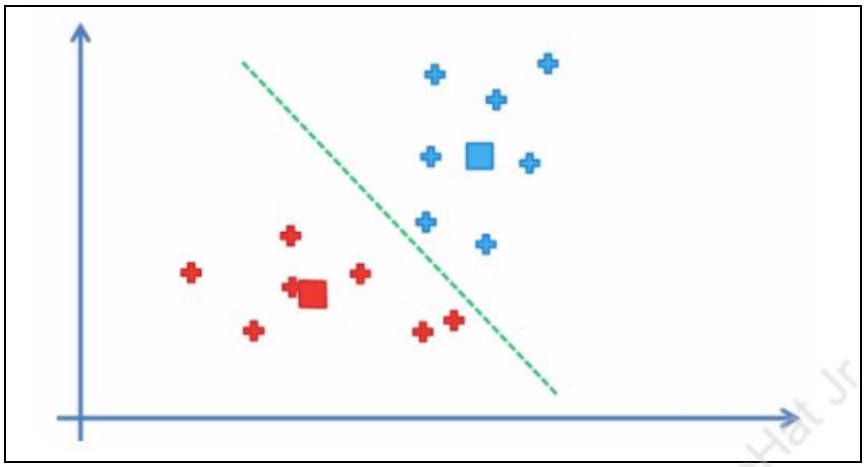
- **Assign each data point to the closest centroid.**



- **Shift the centroids a little for all the clusters.**



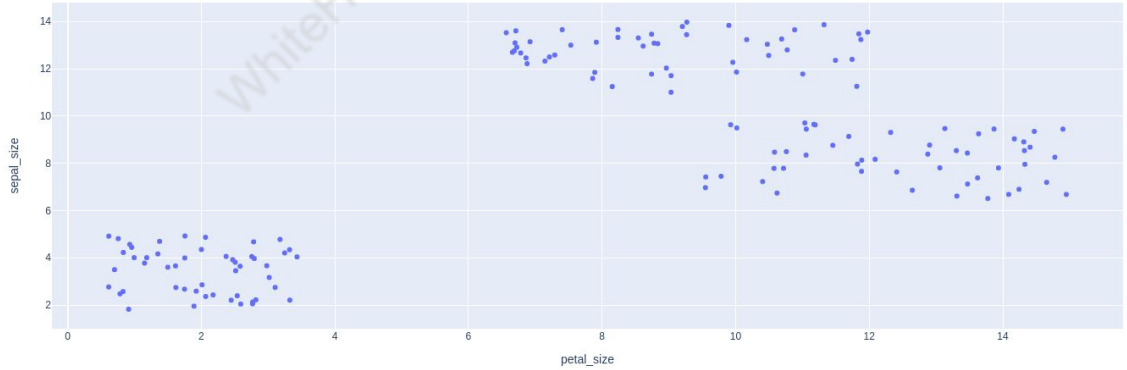
- **Re-assign each data point to the new closest centroid. If any points got reassigned, repeat `Step 4` again otherwise the model is ready.**



2. We took the data for petals and sepals to find the clustering points.

```
[ ] import pandas as pd
    import plotly.express as px

    df = pd.read_csv("petals_sepals.csv")

    print(df.head())

    fig = px.scatter(df, x="petal_size", y="sepal_size")
    fig.show()
```

```
   petal_size  sepal_size
0   11.323484   13.866161
1    9.265842   13.443414
2   14.329944    7.956200
3   11.883902    7.658534
4    9.957722   12.273535
```

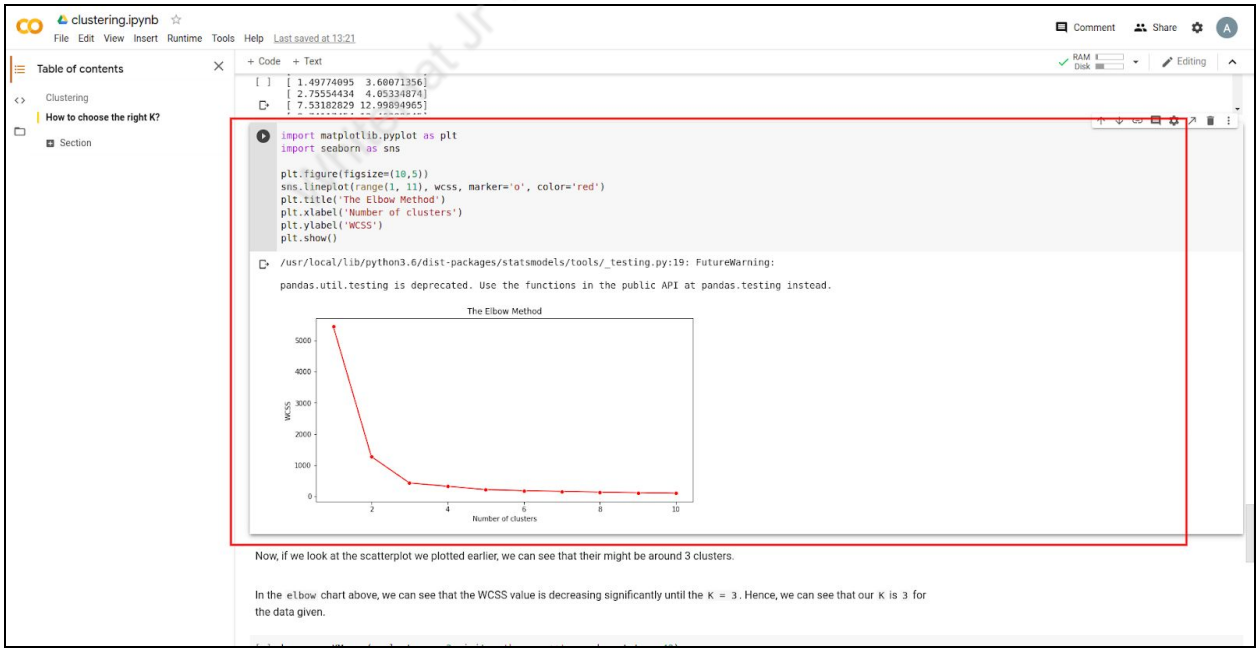


3. We choose the center points from a cluster using the WCSS.

```
[ ] from sklearn.cluster import KMeans

    X = df.iloc[:, [0, 1]].values

    print(X)

    wcss = []
    for i in range(1, 11):
        kmeans = KMeans(n_clusters=i, init='k-means++', random_state = 42)
        kmeans.fit(X)
        # inertia method returns wcss for that model
        wcss.append(kmeans.inertia_)
```

```
[[11.32348369 13.86616131]
 [ 9.26584161 13.4434136 ]
 [14.32994392  7.95619956]
 [11.88390198  7.65853411]
 [ 9.95772216 12.27353488]
 [11.87446585 13.23783855]
 [11.05434664  8.34645832]
 [ 9.92501036  9.63140484]
 [ 6.72330556 12.91052608]
 [ 1.7547028   4.92229755]
 [ 2.53760792  2.39274409]
 [ 0.82826409  2.57057886]
 [14.17308088  9.03309242]
 [ 2.8166071   2.21911623]
 [ 8.6152154  12.96116714]
 [12.87654335  8.38760135]
 [14.08781072  6.68177744]
 [ 2.59059319  2.04203334]
 [ 3.32057276  4.34097779]
 [ 3.32553533  2.20737103]
 [10.01773429  9.49527624]
 [ 9.20235232 13.7895536 ]
 [10.47443458 13.03790983]
 [11.45457896  8.76001507]
 [11.03565171  9.70704578]
 [13.46897961  8.43272357]
 [14.40798387  8.68145304]
 [11.49414942 12.35569869]
 [11.88685783  8.13176978]
 [ 8.54247125 13.30436616]
 [13.86822339  9.45088543]
 [10.49468563 12.56398709]
```

4. We choose the best value for K using the elbow method.

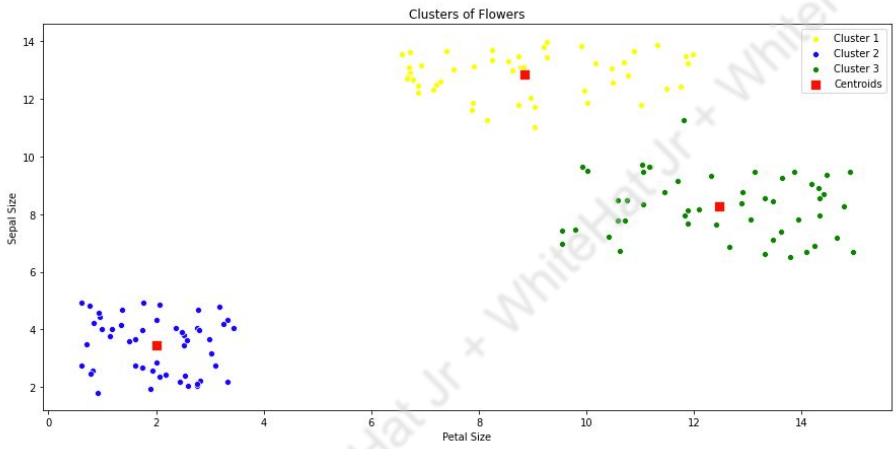


5. Then using the k means algorithm we found the proper cluster points.

```
[ ] kmeans = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
    y_kmeans = kmeans.fit_predict(X)
```

6. Plotted the cluster points on the plots.

```
] plt.figure(figsize=(15,7))
  sns.scatterplot(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], color = 'yellow', label = 'Cluster 1')
  sns.scatterplot(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], color = 'blue', label = 'Cluster 2')
  sns.scatterplot(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], color = 'green', label = 'Cluster 3')
  sns.scatterplot(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], color = 'red', label = 'Centroids',s=100,marker=',')
  plt.grid(False)
  plt.title('Clusters of Flowers')
  plt.xlabel('Petal Size')
  plt.ylabel('Sepal Size')
  plt.legend()
  plt.show()
```



We concluded that there are 3 cluster points in the data.

What's NEXT?

In the next class, we will explore more algorithms for clustering..

EXTEND YOUR KNOWLEDGE:

Learn more about the clustering using k means algorithm:
https://realpython.com/k-means-clustering-python/