|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| k-Means Clustering - Modifications | | | | | | | |
|  | **Section** | **Change** | **Previous code** | **New code** | **Description** | **Level of difficulty (1-10)** | **Time taken** |
| 1 |  | Change the style | plt.style.use('seaborn-whitegrid') | plt.style.use('seaborn-v0\_8-whitegrid') | Changed 'seaborn-whitegrid’ to 'seaborn-v0\_8-whitegrid' | 2 | 5 minutes |
| 2 | Introducing k-Means | Improved visual | plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')  centers = kmeans.cluster\_centers\_  plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200); | plt.figure(figsize=(8, 6))  plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')  centers = kmeans.cluster\_centers\_  plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X', label='Centroids')  plt.title('K-Means Clustering Results with Centroids')  plt.xlabel('Feature 1')  plt.ylabel('Feature 2')  plt.legend()  plt.show() | This code provides improved visual than the previous visual. Main changes are associate with the Centroid. | 8 | 15 minutes |
| 3 | Introducing k-Means | Model evaluation |  | inertia = kmeans.inertia\_  silhouette\_avg = silhouette\_score(X, y\_kmeans)  print(f'Inertia: {inertia}')  print(f'Silhouette Score: {silhouette\_avg:.3f}') | Inertia is Sum of squared distances of samples to their closest cluster center. Lower values indicate tighter clusters.  Silhouette Score measures how similar an object is to its own cluster compared to other clusters. Ranges from -1 (incorrect clustering) to +1 (highly dense clustering). Here, Inertia: 212.00  Silhouette Score: 0.682 These figure suggests the model performance in average level. | 8 | 20 minutes |
| 4 | Introducing k-Means | Elbow method to identify k value |  | inertias = []  k\_range = range(1, 10)  for k in k\_range:  kmeans\_model = KMeans(n\_clusters=k, random\_state=0)  kmeans\_model.fit(X)  inertias.append(kmeans\_model.inertia\_) | In this notebook 4 has chosen as the k value. Here I plotted an elbow curve to check the optimal k value. The chart also suggested that 4 is the optimal value in 1-10 range | 6 | 20 minutes |

**My Example**

**Application of K-means - Iris dataset**

The Iris dataset has details of three types of Iris flowers: Setosa, Versicolor, and Virginica. Also, it has four features per flower such as Sepal length, Sepal width, Petal length, Petal width. We can use k-Means clustering to group the flowers into clusters without using the actual species labels.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Purpose | Code | Description | Level of difficulty (1-10) | Time spent |
| 1 | Importing Libraries | from sklearn.datasets import load\_wine | For this ‘load\_iris’ was imported from ‘sklearn.datasets’. All libraries have included at the beginning of the notebook to avoid redundancy. | 3 | 5 minutes |
| 2 | Loading dataset | iris = load\_iris()  data\_iris = pd.DataFrame(iris.data, columns=iris.feature\_names) | Loaded the iris dataset and made it into a dataframe. | 3 | 5 minutes |
| 3 | Pairplots | sns.pairplot(data\_iris.iloc[:, :5])  plt.show() | Plot the features in the dataset in pairwise to study the correlation between features. | 6 | 10 minutes |
| 4 | Data Preprocessing | scaler = StandardScaler()  scaled\_iris = scaler.fit\_transform(data\_iris) | Normalized data using standard scaler. In k-means it is essential as it ensures that all features contribute equally to the distance calculations that determine the clusters. For this ‘StandardScaler’ was imported from ‘sklearn.preprocessing’ | 5 | 5 minutes |
| 5 | Finding Optimal Number of Clusters (Elbow Method) | inertia\_iris = []  for k in range(1, 11):  kmeans = KMeans(n\_clusters=k, init='k-means++', random\_state=42)  kmeans.fit(scaled\_iris)  inertia\_iris.append(kmeans.inertia\_)  plt.figure(figsize=(8, 5))  plt.plot(range(1, 11), inertia\_iris, 'bx-')  plt.xlabel('Number of Clusters (k)')  plt.ylabel('Inertia')  plt.title('Elbow Method for Iris Dataset')  plt.show() | Elbow method can be used to identify the optimal number of clusters. Here range of 1 – 11 was selected as the number of clusters. Among them 3 can be selected as the optimum number of clusters according to the elbow chart. | 7 | 10 minutes |
| 6 | Building the Model (k=3 based on elbow curve) | kmeans\_iris = KMeans(n\_clusters=3, init='k-means++', random\_state=42)  kmeans\_iris.fit(scaled\_iris)  labels\_iris = kmeans\_iris.labels\_  data\_iris['Cluster'] = labels\_iris | In this part initialise the k-mean by defining number of clusters = 3 (data should be grouped into three clusters, which aligns with the three species in the Iris dataset). The init='k-means++' parameter ensures that the initial cluster centroids are selected using the k-means++ algorithm, which improves convergence speed. random\_state=42 parameter ensures reproducibility (output will be same for every time that the code runs) | 7 | 10 minutes |
| 7 | Visualizing the Clusters | plt.figure(figsize=(8, 5))  colors = ['red', 'green', 'blue']  for i in range(3):  plt.scatter(  scaled\_iris[labels\_iris == i, 0],  scaled\_iris[labels\_iris == i, 1],  s=100, c=colors[i], label=f'Cluster {i+1}'  )  centers\_iris = kmeans\_iris.cluster\_centers\_  plt.scatter(centers\_iris[:, 0], centers\_iris[:, 1], c='yellow', s=300, label='Centroids', marker='X')  plt.xlabel('Feature 1 (scaled)')  plt.ylabel('Feature 2 (scaled)')  plt.title('Iris Dataset Clustering Results')  plt.legend()  plt.show() | The code in this section is used to visualise the clusters after fitting the k-means model. | 7 | 10 minutes |
| 8 | Evaluation | sil\_score\_iris = silhouette\_score(scaled\_iris, labels\_iris)  print(f'Iris Dataset Silhouette Score: {sil\_score\_iris:.3f}') | ‘Silhouette Score’ has used as the evaluation parameter in this scenario. Silhouette Score measures how similar an object is to its own cluster compared to other clusters. Ranges from -1 (incorrect clustering) to +1 (highly dense clustering). In this case Silhouette Score: 0.480. This value indicates that model is not that much accurate. | 8 | 15 minutes |
| 9 | Improving the model using feature selection and result evaluation | plt.figure(figsize=(8, 6))  sns.heatmap(data\_iris.corr(), annot=True, cmap='coolwarm')  plt.title('Correlation Matrix of Iris Dataset Features')  plt.show()  selected\_features = ['petal length (cm)', 'petal width (cm)']  feature\_data = data\_iris[selected\_features]  print(f"Selected Features for Clustering: {selected\_features}")  scaled\_selected = scaler.fit\_transform(feature\_data)  kmeans\_selected = KMeans(n\_clusters=3, random\_state=42)  kmeans\_selected.fit(scaled\_selected)  labels\_selected = kmeans\_selected.labels\_  data\_iris['Cluster\_Feature\_Selected'] = labels\_selected | Feature selection can be performed to increase the model performance. To select the best features correlation matrix was analysed. 'petal length (cm)' and 'petal width (cm)' was selected as they were strongly correlated with the target variable. Then scaling was applied again to ensure that both features contribute equally to the clustering process. Finally, a K-Means model is built using these scaled features, grouping the data into three clusters. Then checked Silhouette Score for this improved model and it was increased to 0.674. That is: this improved model is performing better than previous k-means model. | 8 | 20 minutes |
| 10 | Model comparison | models = ['Original', 'Feature Selected']  scores = [sil\_score\_iris, sil\_score\_selected]  plt.figure(figsize=(8, 5))  sns.barplot(x=models, y=scores, palette='viridis')  plt.ylabel('Silhouette Score')  plt.title('Comparison of Silhouette Scores')  plt.ylim(0, 1)  plt.show()  for model, score in zip(models, scores):  print(f'{model}: Silhouette Score = {score:.3f}') | Then I plotted Silhouette Scores for both models in a bar chart for easy understanding. | 8 | 15 minutes |
| 11 | Final Cluster Interpretation | final\_clusters = data\_iris.groupby('Cluster\_Feature\_Selected').mean()  print("\nMean values of selected features for each cluster:")  print(final\_clusters[selected\_features])  # Mapping cluster labels to actual species for interpretation  species\_mapping = {0: 'Cluster 0', 1: 'Cluster 1', 2: 'Cluster 2'}  data\_iris['Predicted\_Species'] = data\_iris['Cluster\_Feature\_Selected'].map(species\_mapping) | Finally interpreted the characteristics of each cluster by calculating mean values of petal length (cm) and petal width (cm). | 9 | 15 minutes |