

Name: Umesh Dhakal

Course: MSCS634

Professor: Dr. Satish Penmatsa

February 13, 2026

## Lab 3- Clustering Analysis Using K-Means and K-Medoids Algorithms

### Load and Prepare the Dataset

The screenshot displays a Jupyter Notebook environment within the Anaconda Toolbox. The left sidebar contains navigation options: Anaconda Cloud (Create a New Project, Create a New Notebook, My Projects), Code Snippets (Manage Code Snippets), and Anaconda AI Assistant (Chat-based Python Help). The main area shows a notebook titled 'MSCS634L3' with the following code:

```
## MSCS634L3
## Lab 3: Clustering Analysis Using K-Means and K-Medoids Algorithms

[11]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, adjusted_rand_score
from sklearn.metrics import pairwise_distances

[12]: # Loading wine dataset
wine_data = load_wine()

x_data = wine_data.data
y_label = wine_data.target
col_name = wine_data.feature_names

wine_df = pd.DataFrame(x_data, columns=col_name)
wine_df["real_class"] = y_label

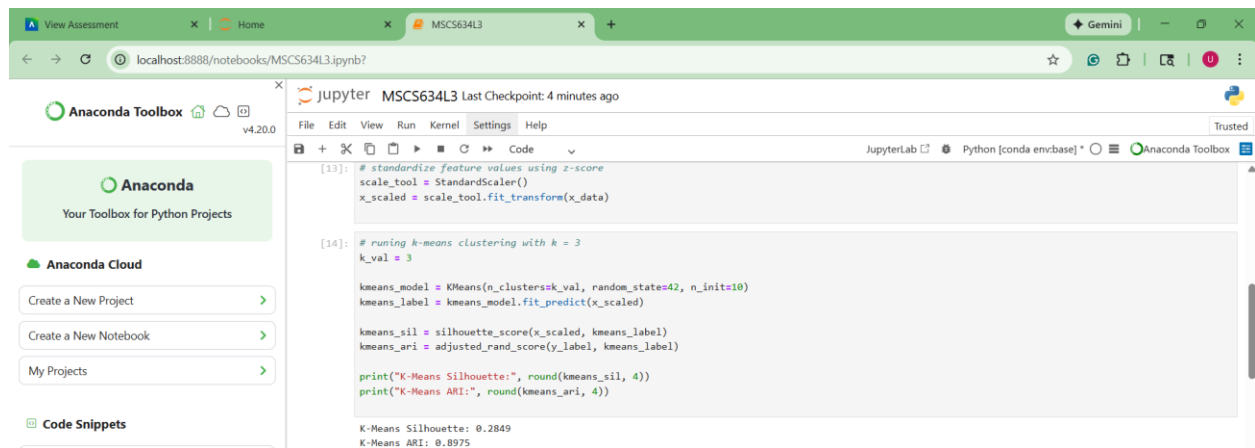
print("Data shape:", x_data.shape)
print(wine_df["real_class"].value_counts().sort_index())

Data shape: (178, 13)
real_class
0    59
1    71
2    48
Name: count, dtype: int64

[13]: # standardize feature values using z-score
scale_tool = StandardScaler()
x_scaled = scale_tool.fit_transform(x_data)
```

The output of the code shows the data shape and the distribution of the 'real\_class' variable.

### Implement K-Means Clustering



The screenshot shows a JupyterLab interface with a sidebar on the left containing 'Anaconda Toolbox', 'Anaconda Cloud', and 'Code Snippets'. The main area displays a notebook titled 'MSCS634L3' with two code cells. The first cell standardizes feature values using z-score. The second cell runs K-Means clustering with k=3, calculates the silhouette score and adjusted Rand score, and prints the results.

```
[13]: # standardize feature values using z-score
scale_tool = StandardScaler()
x_scaled = scale_tool.fit_transform(x_data)

[14]: # running k-means clustering with k = 3
k_val = 3

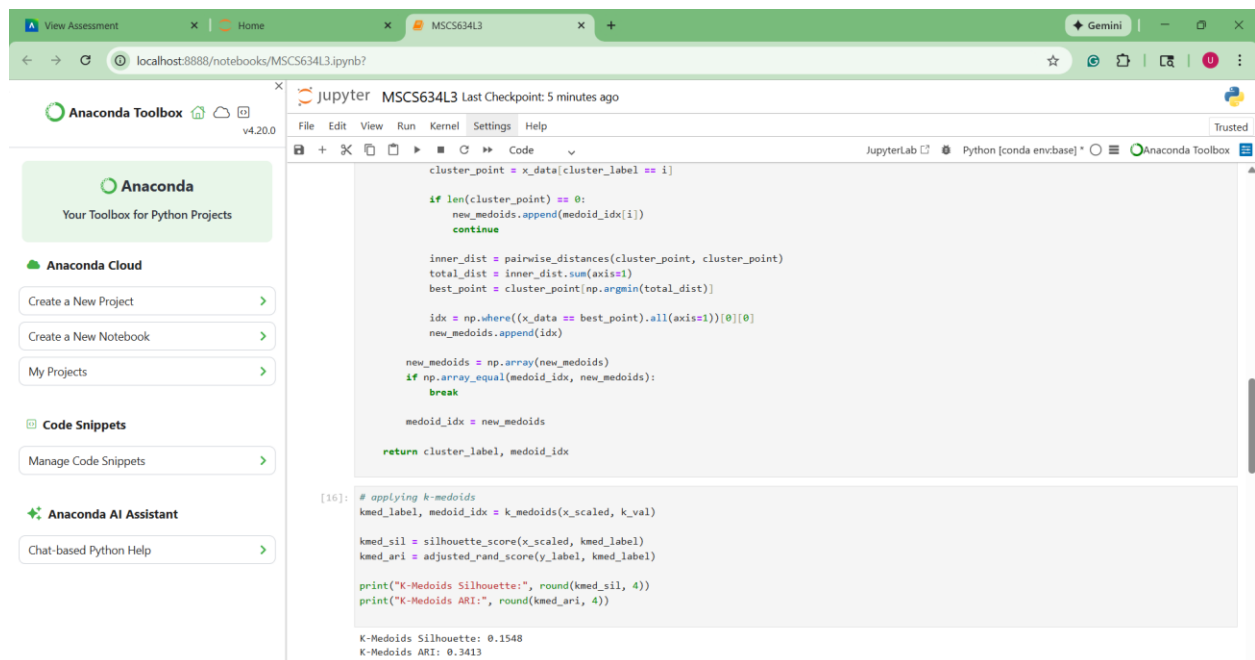
kmeans_model = KMeans(n_clusters=k_val, random_state=42, n_init=10)
kmeans_label = kmeans_model.fit_predict(x_scaled)

kmeans_sil = silhouette_score(x_scaled, kmeans_label)
kmeans_ari = adjusted_rand_score(y_label, kmeans_label)

print("K-Means Silhouette:", round(kmeans_sil, 4))
print("K-Means ARI:", round(kmeans_ari, 4))

K-Means Silhouette: 0.2849
K-Means ARI: 0.8975
```

## Implement K-Medoids Clustering



The screenshot shows a JupyterLab interface with a sidebar on the left containing 'Anaconda Toolbox', 'Anaconda Cloud', 'Code Snippets', and 'Anaconda AI Assistant'. The main area displays a notebook titled 'MSCS634L3' with two code cells. The first cell implements a function to find the best medoid for a cluster. The second cell applies K-Medoids clustering with k=3, calculates the silhouette score and adjusted Rand score, and prints the results.

```
cluster_point = x_data[cluster_label == i]

if len(cluster_point) == 0:
    new_medoids.append(medoid_idx[i])
    continue

inner_dist = pairwise_distances(cluster_point, cluster_point)
total_dist = inner_dist.sum(axis=1)
best_point = cluster_point[np.argmin(total_dist)]

idx = np.where((x_data == best_point).all(axis=1))[0][0]
new_medoids.append(idx)

new_medoids = np.array(new_medoids)
if np.array_equal(medoid_idx, new_medoids):
    break

medoid_idx = new_medoids

return cluster_label, medoid_idx

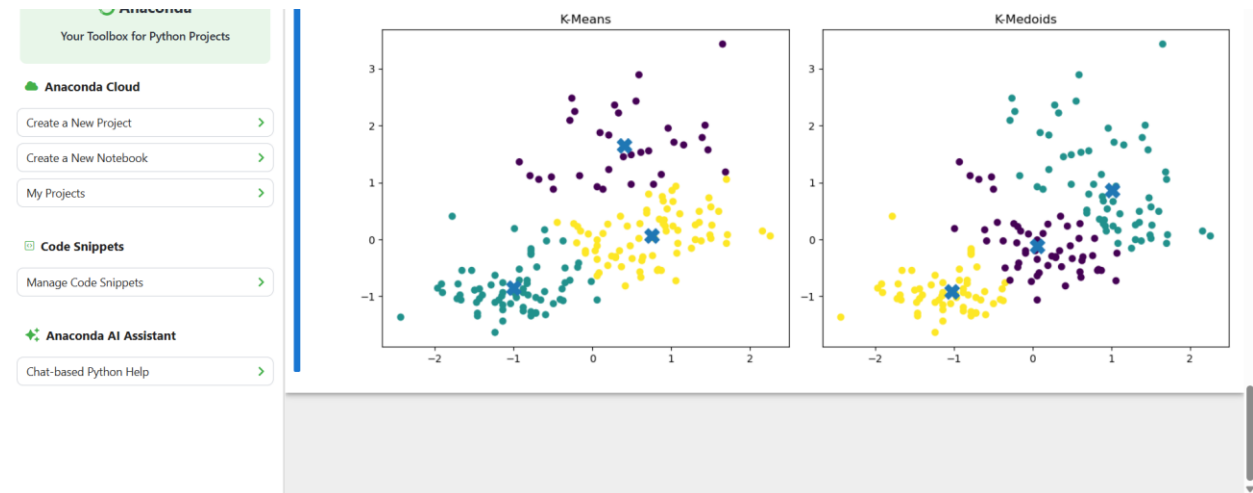
[16]: # applying k-medoids
kmed_label, medoid_idx = k_medoids(x_scaled, k_val)

kmed_sil = silhouette_score(x_scaled, kmed_label)
kmed_ari = adjusted_rand_score(y_label, kmed_label)

print("K-Medoids Silhouette:", round(kmed_sil, 4))
print("K-Medoids ARI:", round(kmed_ari, 4))

K-Medoids Silhouette: 0.1548
K-Medoids ARI: 0.3413
```

## Visualization



## Insight and analysis

Based on the clustering results, both K-Means and K-Medoids were able to group the Wine dataset into three clusters, which matches the actual number of wine classes in the dataset. Comparing the the silhouette scores, K-Means produced slightly tighter clusters overall. This shows that the data points inside each cluster were more closely grouped together, which means the cluster boundaries were more clearly define. The Adjusted Rand Index also showed that K-Means had a slightly better match with the real class labels compared to K-Medoids, even though both methods performed reasonably well.

One of the difference I notice between the two algorithms was how the cluster centers were handled. K-Means uses the mean of the data points as the cluster center which makes it efficient and effective when the data is evenly distributed. However, this also means it can be sensitive to extreme values. On the other hand, K-Medoids selects actual data points as the cluster centers which makes it more stable and less affected by potential outliers. Because of this, the K-Medoids clusters appeared slightly more spread out, but also more realistic in terms of representing actual data samples. K-Means performed slightly better for this dataset because the

Wine data is well-structured and does not contain extreme outliers. K-Medoids still provided meaningful clusters and demonstrated its advantage in robustness.