COS40007 Artificial Intelligence for Engineering

Week 9 Studio Activities

ILO	Understand Clustering Algorithms.
Aim	• Learn what *k*-Means and DBScan algorithms are. • Learn how to evaluate the quality of clustering models. • Learn the different distance measurement methods for clustering.
Resources	Books: 1. Prosise, Jeff. Applied machine learning and AI for engineers. " O'Reilly Media, Inc.", 2022. 2. Raschka, Sebastian, Yuxi Hayden Liu, and Vahid Mirjalili. Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python. Packt Publishing Ltd, 2022. Web Resources: 1. https://www.geeksforgeeks.org/clustering-in-machine-learning/ 2. https://scikit-learn.org/stable/modules/clustering.html 3. https://developers.google.com/machine-learning/clustering/overview
Requirements for to be marked as complete	Demonstrate the ability to derive accurate classes from unlabelled datasets using clustering algorithms.

Studio Activity 1: Preprocessing and Feature Selection

In this studio activity, we will learn how to use clustering algorithms to derive classes from unlabeled data sets. We will use the One Year Industrial Component Degradation dataset for this studio. This studio task aims to detect sensor anomalies by clustering the normally operating sensors.

Similar to classification models, unlabelled datasets must be cleaned and preprocessed before training the clustering model. So,

1) Identify the datatypes in the dataset. We must eliminate non-numerical features, categorical features, and labels in the dataset. The *k*-means algorithm is accurate with numerical features.
2) Drop duplicates and outliers from the dataset.

Features that are most suitable for accurately detecting sensor anomalies need to be derived. So, perform an exploratory data analysis (EDA) on the dataset, which includes,

1) Probability density distribution of features (identify the features having multiple distinctive peaks) and
2) Correlation of features (identify feature pairs which may have strong correlations)

Based on the analysis above, select the best *k*-features before moving on to the next step.

Studio Activity 2: Training a *K*-Means model and Deciding the Number of Clusters

Follow this documentation to develop a simple *K*-means model. Follow this video or discuss with your tutor how the *K*-Means algorithm executes.

You might identify that the number of clusters is one of the parameters to train the model. Iteratively, train multiple *K*-Means models using the dataset for between 2 to 10 clusters. Visualise each *K*-Means model's Within Cluster Sum of Squares (or inertia) for the given *k* value.

Using the graph, identify the accurate number of clusters using the elbow method under the guidance of your tutor. You may even use Yellowbricks KelbowVisualizer to help you decide the number of clusters.

Visualise the clustered data in a scatter plot, using colours corresponding to the cluster label.

Studio Activity 3: Assessing the quality of clustering

A clustering model could be identified to be optimally performing when:

1) the distance between the clusters is maximised, and/or

2) the average distance between the data points with a cluster is minimised.

We used the inertial above to measure how close the data points were in each cluster. Similarly, refactor the code to measure the Dunn Index and DB Index for different *k* values. Follow this tutorial to implement these indexes.

Discuss the index values in the plot with your tutor.

Silhouette Score is a clustering quality measure that underpins inter and intra-cluster distances. Follow this tutorial to implement a code to retrieve the silhouette score for different *k* values.

Studio Activity 4: Understanding different distance measurements

Many distance measures could be used to cluster data. For instance, cosine similarity is more suitable for clustering when data points are represented using binary vectors (e.g., word embedding). So,

1) Identify the different distance measurements in this tutorial,
2) Follow this tutorial, which uses matrix embedding to transform images into features and use cosine distance measurement to identify the images that are similar to each other and
3) In the above implementation, change the embedding parameter to the other distance measurement methods identified in Step 1 and compare their clustering accuracy.

Studio Activity 5: Density-based clustering algorithms

Follow this tutorial as a guide, and refactor the *K*-Means implementation in Activity 2 to use DBScan – a density-based clustering algorithm- instead of distance-based algorithms such as *K*-Means.

Compare the clustering performance of the two algorithms.