# Project Based Learning Report

On

# K-means clustering for students grades in FLNNGA Unit Test-1 marks of 2023-24 and compare it with C-means clustering

Submitted in the partial fulfilment of the requirement

For the Project Based Learning in **Fuzzy Logic, Neural Networks and Genetic Algorithm**

In

### Electronics & Communication Engineering

By

### 2214110458 ADITYA SINGH

**2214110420 M. HAMZA MENON**

### 2214110432 SAMEER ABDUL SHARIF

Under the guidance of **V.P. Kaduskar**



Department of Electronics & Communication Engineering Bharati Vidyapeeth

(Deemed to be University) College of Engineering Pune – 410043

Academic Year: 2024-25

**Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune – 411043**

**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**

## CERTIFICATE

# Certified that the Project Based Learning report entitled, “K-means clustering for students grades in FLNNGA Unit Test-1 marks of 2023-24 and compare it with C-means clustering”

is work done by

### 2214110458 Aditya Singh

**2214110420 M. Hamza Menon**

### 2214110432 Sameer Abdul Sharif

in partial fulfilment of the requirements for the award of credits for Project Based Learning (PBL) in of **Fuzzy Logic, Neural Networks and Genetic Algorithm** Bachelor of Technology Semester V, in Electronics & Communication Engineering.

### Date:

**Prof. V.P. Kaduskar Dr. Arundhati A. Shinde Course In-charge Professor & Head**

# Index

|  |  |  |
| --- | --- | --- |
| S.no | Topic | Page No |
| 1 | Introduction | 1 |
| 2 | Problem Statement | 2 |
| 3 | K means clustering and Fuzzy C means clustering | 3-4 |
| 4 | Python language | 5 |
| 5 | Algorithm used and steps | 6-11 |
| 6 | Code output | 11-12 |
| 7 | Result and discussion | 13-14 |
| 8 | Conclusion | 15 |

**Introduction**

K-means clustering and fuzzy C-means clustering are both techniques for grouping data, but they differ significantly in their approach to cluster assignment. K-means is a hard clustering algorithm that partitions data into KKK distinct, non-overlapping clusters. It works by randomly selecting initial centroids, assigning each data point to the nearest centroid, and iteratively recalculating centroids until assignments stabilize. In contrast, fuzzy C-means (FCM) is a soft clustering algorithm that allows each data point to belong to multiple clusters with varying degrees of membership. It assigns membership values based on the distance to cluster centers and updates those centers using weighted contributions from all points. As a result, FCM is more effective in scenarios where clusters overlap, whereas K-means tends to struggle with such datasets. The key distinction lies in how each algorithm handles cluster membership, with K-means offering a definitive assignment and FCM providing a more nuanced approach that can better accommodate complex data distributions.

1

## Problem Statement

K-means clustering for students grades in FLNNGA Unit Test-1 marks of 2023-24 and compare it with C-means clustering

## Objective:

Utilize K-means clustering in jupyter notebook to effectively segment marks of students based on relevant marks they obtained and comparing it with fizzy C-means clustering.

It assigns membership values based on the distance to cluster centers and updates those centers using weighted contributions from all points.

2

**K Means Clustering and Fuzzy C means Clustering**

K-means clustering and fuzzy C-means clustering can be effectively applied to analyze student marks (e.g., UT marks) to group students based on their performance.

**K-means Clustering**

In the context of student UT marks, K-means clustering would involve the following steps:

1. Data Preparation: Gather the UT marks for all students and organize them in a numerical format.
2. Choosing KKK: Decide on the number of clusters (e.g., high performers, average performers, low performers).
3. Initialization: Randomly select KKK initial centroids from the marks data.
4. Assignment: Assign each student to the nearest centroid based on their marks.
5. Centroid Update: Calculate the new centroids by taking the mean of the marks in each cluster.
6. Iteration: Repeat the assignment and update steps until the cluster assignments do not change significantly.

This approach would yield distinct groups of students, making it easy to identify performance categories.

**Fuzzy C-means Clustering**

Fuzzy C-means clustering, on the other hand, would provide a more nuanced view of student performance:

1. Data Preparation: Similar to K-means, start with the UT marks dataset.
2. Choosing CCC: Decide on the number of clusters, but recognize that students may belong to more than one cluster.
3. Initialization: Select CCC initial cluster centers.
4. Membership Assignment: Calculate the degree of membership for each student to all clusters based on their marks.
5. Centroid Update: Update the cluster centers based on the weighted average of the students' marks, considering their membership degrees.
6. Iteration: Continue updating memberships and centroids until they stabilize.

Using FCM allows students to be classified into multiple categories (e.g., a student may be a strong performer in math but average in science), providing a richer understanding of their strengths and weaknesses.

3

**Comparison**

* K-means will create clear divisions, categorizing students strictly into groups, which is useful for straightforward performance assessments.
* Fuzzy C-means offers flexibility, recognizing that many students may not fit neatly into one category, which can help in personalized education strategies.

Both methods can provide valuable insights for educators, enabling targeted interventions based on student performance profiles.

4

**Python language**

Python is a high-level, interpreted programming language known for its simplicity and readability. It is widely used for various applications, including web development, data analysis, artificial intelligence, machine learning, scientific computing, and more.

**Jupyter Notebook**

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It is particularly popular in data science and academic settings.

**Key Features**:

* **Interactive Computing**: Users can write and execute code in chunks (cells) and see results immediately, facilitating experimentation and iteration.
* **Rich Media Support**: It supports Markdown for text formatting, enabling users to combine code with rich documentation, images, and interactive visualizations.
* **Integration**: Easily integrates with libraries like Matplotlib and seaborn for data visualization, as well as other tools for machine learning and analysis.

5

**Algorithm used and steps**

**K-Means Clustering**

**Algorithm Used:** K-Means Clustering

**Steps:**

1. **Data Preparation:**
   * Load the student data into a DataFrame.
   * Replace non-numeric values ('ABSENT') with -1 to handle missing data.
   * Convert grades to numeric format.
2. **Clustering:**
   * Choose the number of clusters (e.g., 3).
   * Initialize cluster centroids randomly.
   * Assign each data point to the nearest centroid.
   * Recalculate centroids based on the assigned points.
   * Repeat the assignment and centroid update steps until convergence (i.e., centroids do not change significantly).
3. **Visualization:**
   * Use a scatter plot to visualize the students' grades colored by their cluster assignments.

**Fuzzy C-Means Clustering**

**Algorithm Used:** Fuzzy C-Means Clustering

**Steps:**

1. **Data Preparation:**
   * Load the student data into a DataFrame.
   * Replace non-numeric values ('ABSENT') with -1 to handle missing data.
   * Convert grades to numeric format.
2. **Clustering:**
   * Choose the number of clusters (e.g., 3) and the fuzziness parameter (e.g., 2).
   * Initialize cluster centers randomly or use a specified method.
   * Calculate the degree of membership of each data point to each cluster based on distances.
   * Update cluster centers based on the membership values.
   * Repeat the membership assignment and center updating steps until convergence (i.e., the changes in membership values are below a set threshold).
3. **Visualization:**
   * Use a scatter plot to visualize the students' grades colored by their fuzzy cluster memberships.

**Key Differences**

* **K-Means:** Each data point belongs to one cluster, creating hard boundaries.
* **Fuzzy C-Means:** Each data point can belong to multiple clusters with varying degrees of membership, leading to softer boundaries

6

# Code 1:

# pip install pandas numpy plotly scikit-learn

# import pandas as pd

# import numpy as np

# from sklearn.cluster import KMeans

# import matplotlib.pyplot as plt

# # Provided student data

# data = {

# 'Roll No': list(range(1, 59)),

# 'Unit\_Test\_I\_Grades': [12, 6, 9, 2, 14, 'ABSENT', 10, 'ABSENT', 12, 'ABSENT', 'ABSENT', 14, 8, 'ABSENT', 14, 12, 'ABSENT', 14, 'ABSENT',

# 6, 15, 8, 14, 'ABSENT', 13, 8, 14, 15, 5, 'ABSENT', 11, 'ABSENT', 'ABSENT', 6, 9, 13, 'ABSENT', 7, 'ABSENT',

# 12, 13, 9, 10, 2, 9, 7, 8, 0, 10, 'ABSENT', 8, 8, 12, 13, 16, 11, 'ABSENT', 'ABSENT']

# }

# # Create a DataFrame from the provided data

# df = pd.DataFrame(data)

# # Handle missing or non-numeric values (in this case, replacing 'ABSENT' with -1)

# df['Unit\_Test\_I\_Grades'] = pd.to\_numeric(df['Unit\_Test\_I\_Grades'], errors='coerce').fillna(-1)

# # Perform K-Means clustering

# kmeans = KMeans(n\_clusters=3, random\_state=42)

# df['Cluster'] = kmeans.fit\_predict(df['Unit\_Test\_I\_Grades'].values.reshape(-1, 1))

# # Visualize the clusters

# plt.scatter(df['Roll No'], df['Unit\_Test\_I\_Grades'], c=df['Cluster'], cmap='viridis', s=50, alpha=0.7)

# plt.xlabel('Roll No')

# plt.ylabel('Unit Test I Grades')

# plt.title('K-Means Clustering of Students')

# plt.show()

# import pandas as pd

# import numpy as np

# import matplotlib.pyplot as plt

# import skfuzzy as fuzz

# 7

# # Provided student data

# data = {

# 'Roll No': list(range(1, 59)),

# 'Unit\_Test\_I\_Grades': [12, 6, 9, 2, 14, 'ABSENT', 10, 'ABSENT', 12, 'ABSENT', 'ABSENT', 14, 8, 'ABSENT', 14, 12, 'ABSENT', 14, 'ABSENT',6, 15, 8, 14, 'ABSENT', 13, 8, 14, 15, 5, 'ABSENT', 11, 'ABSENT', 'ABSENT', 6, 9, 13, 'ABSENT', 7, 'ABSENT',12, 13, 9, 10, 2, 9, 7, 8, 0, 10, 'ABSENT', 8, 8, 12, 13, 16, 11, 'ABSENT', 'ABSENT']

# }

# # Create a DataFrame from the provided data

# df = pd.DataFrame(data)

# # Handle missing or non-numeric values (replacing 'ABSENT' with -1)

# df['Unit\_Test\_I\_Grades'] = pd.to\_numeric(df['Unit\_Test\_I\_Grades'], errors='coerce').fillna(-1)

# # Prepare the data for clustering

# grades = df['Unit\_Test\_I\_Grades'].values.reshape(-1, 1)

# # Perform Fuzzy C-Means clustering

# cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(

# grades.T, 3, 2, error=0.005, maxiter=1000, init=None, seed=42)

# # Get the cluster membership for each student

# df['Cluster'] = np.argmax(u, axis=0)

# # Visualize the clusters

# plt.scatter(df['Roll No'], df['Unit\_Test\_I\_Grades'], c=df['Cluster'], cmap='viridis', s=50, alpha=0.7)

# plt.xlabel('Roll No')

# plt.ylabel('Unit Test I Grades')

# plt.title('Fuzzy C-Means Clustering of Students')

# plt.show()

# 8

# K-Means Clustering

# Algorithm Used: K-Means Clustering

# Steps:

# Load Data:

# Read student grades from a CSV file into a Pandas DataFrame.

# Data Preparation:

# Convert the Unit\_Test\_I\_Grades column to numeric, replacing non-numeric values ('ABSENT') with -1.

# Reshape the data for clustering.

# Clustering:

# Set the number of clusters (e.g., k = 3).

# Initialize the K-Means algorithm with the specified number of clusters and a random state for reproducibility.

# Fit the K-Means model to the grades data and predict the cluster assignments.

# Visualization:

# Create a scatter plot using Plotly to visualize the grades, coloring the points based on their assigned clusters.

# Optionally, add a trend line to show the overall trend of grades.

# Fuzzy C-Means Clustering

# Algorithm Used: Fuzzy C-Means Clustering

# Steps:

# Load Data:

# Read student grades from a CSV file into a Pandas DataFrame (same as in K-Means).

# Data Preparation:

# Convert the Unit\_Test\_I\_Grades column to numeric, replacing non-numeric values ('ABSENT') with -1.

# Reshape the data for clustering.

# Clustering:

# Set the number of clusters (e.g., n\_clusters = 3).

# Use the Fuzzy C-Means algorithm to calculate cluster centers and membership values for each student based on their grades.

# Assign each student to the cluster for which they have the highest membership value.

# Visualization:

# Create a scatter plot using Plotly to visualize the grades, coloring the points based on their assigned fuzzy clusters.

# Optionally, add a trend line to show the overall trend of grades.

# 9

# Key Differences Between K-Means and Fuzzy C-Means

# K-Means: Each data point is assigned to exactly one cluster, creating hard boundaries.

# Fuzzy C-Means: Each data point can belong to multiple clusters with varying degrees of membership, allowing for softer boundaries.

# Code 2:

# pip install scikit-fuzzy

# import pandas as pd

# import numpy as np

# from sklearn.cluster import KMeans

# import plotly.express as px

# # Load data from CSV

# data = pd.read\_csv(r"C:\Users\Aditya\Downloads\student\_grades.csv") # Using raw string literal for the path

# # Handle missing or non-numeric values (in this case, replacing 'ABSENT' with -1)

# data['Unit\_Test\_I\_Grades'] = pd.to\_numeric(data['Unit\_Test\_I\_Grades'], errors='coerce').fillna(-1)

# # Perform K-Means clustering

# k = 3 # Number of clusters

# X = data['Unit\_Test\_I\_Grades'].values.reshape(-1, 1)

# kmeans = KMeans(n\_clusters=k, random\_state=42)

# data['Cluster'] = kmeans.fit\_predict(X)

# # Convert range object to a list

# x\_values = list(range(1, len(data) + 1))

# # Interactive visualization using Plotly

# fig = px.scatter(data, x=x\_values, y='Unit\_Test\_I\_Grades', color='Cluster',

# sequence for clusters

# labels={'x': 'Student ID', 'Unit\_Test\_I\_Grades': 'Grades'},

# title='K-Means Clustering of Students Based on Unit Test I Grades')

# # Optional: Add a trend line (if needed)

# fig.add\_scatter(x=x\_values, y=data['Unit\_Test\_I\_Grades'],

# mode='lines', line=dict(color='red', width=2),

# name='Grades Trend')

# 10

# fig.update\_layout(

# showlegend=True,

# xaxis=dict(title='Student ID'),

# yaxis=dict(title='Grades'),

# coloraxis\_colorbar=dict(title='Cluster') # This line is optional and only applicable for continuous color scales

# )

# Code Output

# 

# Fig 1 a): Visualisation of K means clustering:

# Cluster assigned to one data

# 

# Fig 1 b): Visualisation of fuzzy C means clustering:

# Cluster assigned to more groups/data

# 11

# 

# Fig 2 a): Thread line analysis of fuzzy K means clustering:

# K-Means forces students into one category, which simplifies interpretation but may miss nuances

# 

# Fig 2 b): Thread line Analysis of fuzzy C means clustering:

# overlapping cluster memberships and a soft assignment of students to clusters

12

**Result and discussion**

K-Means Clustering: Final Output

1. K-Means Clustering with Matplotlib Visualization

In this case, the final output will be a static scatter plot generated using matplotlib, where:

X-axis: Student ID (or Roll No.).

Y-axis: Unit Test I Grades.

Points: Each student’s grades, color-coded by the cluster they belong to.

Clusters: K-Means assigns each student to one of the 3 clusters based on their grades. The clusters are distinct, with each point belonging to exactly one cluster (hard clustering).

The clusters represent groups of students based on their grades, where K-Means aims to minimize the intra-cluster variance. However, the grades of students absent for the test (replaced by -1) will likely form a separate cluster since they will differ significantly from non-absent grades.

Example Visualization:

Cluster 0: Students with low grades.

Cluster 1: Students with average grades.

Cluster 2: Students with high grades.

2. K-Means Clustering with Plotly Interactive Visualization

Here, the final output is an interactive scatter plot where:

The same X and Y axes represent Student ID and Grades.

Clusters are represented by different colors, and you can interact with the plot, zoom in/out, and hover over points for more details.

An optional trend line can show the general trend in student grades.

Again, each student is hard-assigned to a single cluster.

Fuzzy C-Means Clustering: Final Output

1. Fuzzy C-Means Clustering with Matplotlib Visualization

In Fuzzy C-Means (FCM) clustering, the final output will also be a scatter plot, but it differs from K-Means because FCM allows soft clustering, where a student can belong to multiple clusters with a membership value between 0 and 1.

For visualization purposes, each student will be assigned to the cluster where they have the highest membership, but the underlying clustering model does not force hard assignments. Therefore:

X-axis: Student ID.

Y-axis: Unit Test I Grades.

Clusters: Based on the fuzzy membership values, students are assigned to clusters. However, students near the boundary of two clusters could belong partly to both, though only the dominant cluster will be visualized.

Difference: Students with borderline grades (e.g., those around the middle grade range between clusters) may have partial membership in multiple clusters, unlike K-Means, which forces students into one cluster.

13

2. Fuzzy C-Means Clustering with Plotly Interactive Visualization

The final output will be an interactive scatter plot, similar to the K-Means version but reflecting FCM's soft clustering:

X-axis: Student ID.

Y-axis: Unit Test I Grades.

Clusters: Represented by different colors. The visualization may show a dominant cluster for each student, but behind the scenes, FCM allows partial membership in clusters.

Comparison of Outputs: K-Means vs Fuzzy C-Means

Clustering Approach:

K-Means: Hard clustering. Each student is assigned to one cluster only.

Fuzzy C-Means: Soft clustering. Each student has a membership value for each cluster (e.g., 60% in cluster 1, 40% in cluster 2). In practice, we visualize the cluster with the highest membership, but soft membership information is available.

Visualization:

Both methods provide scatter plots with the same axes (Student ID and Grades).

The K-Means clusters are distinct, while the Fuzzy C-Means clusters may have overlap, especially for students with borderline grades.

Handling of Borderline Students:

K-Means will assign a student to the nearest cluster, which may lead to arbitrary assignments for students with borderline grades.

Fuzzy C-Means accounts for partial membership, meaning borderline students can be associated with multiple clusters to a certain degree.

Effect of ABSENT Values:

In both methods, the ABSENT values replaced with -1 will likely form a separate cluster since they significantly differ from the other grades.

Interpretation:

K-Means forces students into one category, which simplifies interpretation but may miss nuances (e.g., a student whose grades are borderline between two clusters).

Fuzzy C-Means offers a more nuanced approach, acknowledging that students can belong to multiple categories based on their grades.

Which to Use?

K-Means is ideal when clear, distinct groups are needed, and when you want a straightforward clustering where each student belongs to one cluster.

Fuzzy C-Means is better when you expect overlap between clusters and want to capture the uncertainty or partial memberships in the clustering process.

In summary, the final output for K-Means is more rigid and distinct, while for Fuzzy C-Means, it's flexible, with overlapping cluster memberships and a soft assignment of students to clusters.

14

**Conclusion**

Both clustering algorithms identify three distinct groups of students based on their Unit Test I grades.

K-means provides a more rigid classification, assigning each student to a single cluster. Fuzzy C-means offers a more flexible approach, allowing for overlapping memberships and a more nuanced understanding of student performance. The choice between K-means and fuzzy C-means depends on the specific research question and the desired level of granularity in the analysis. The thread line analysis provides a visual representation of the cluster structure and the degree of overlap between clusters. K-means clustering produces distinct clusters with minimal overlap, while fuzzy C-means clustering allows for overlapping memberships and a more flexible assignment of students to clusters.The choice between K-means and fuzzy C-means depends on the specific research question and the desired level of granularity in the analysis.

15