Assignment Title: Unsupervised Learning with Dimensionality Reduction and Clustering

Authors: Koustab Ghosh[1] & Sujoy Kumar Biswas[2]

Affiliation:

1. Researcher, IDEAS-TIH, Indian Statistical Institute, Kolkata
2. Head of Research & Innovation, IDEAS-TIH, Indian Statistical Institute, Kolkata

Dated: Sep 07th, 2025

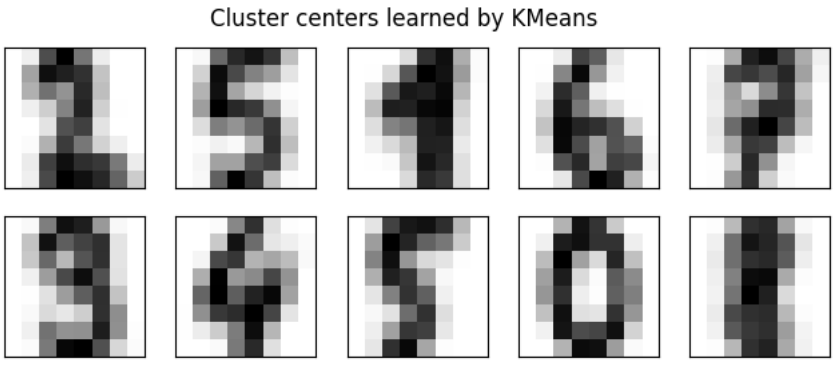
We shall work with the MNIST handwritten digits' image dataset. The details about the dataset is available here.

We need the scikit-learn library to import the various machine leanring models for our study.

Question 1. Complete the following lines of code for K-Means clustering

```
# Import libraries
from sklearn.datasets import load_digits
from sklearn.cluster import KMeans
from matplotlib import pyplot as plt

# Load the MNIST digits dataset (8x8 images → 64 features)
digits = load_digits()
print("Digits data shape:", digits.data.shape)   # (1797, 64)

# KMeans clustering
kmeans = KMeans(n_clusters=10, random_state=0)
clusters = kmeans.fit_predict(digits.data)
print("Cluster centers shape:", kmeans.cluster_centers_.shape)  # (10, 64)

# Visualize the cluster centers as images
fig, ax = plt.subplots(2, 5, figsize=(8, 3))
centers = kmeans.cluster_centers_.reshape(10, 8, 8)

for axi, center in zip(ax.flat, centers):
    axi.set(xticks=[], yticks=[])
    axi.imshow(center, interpolation="nearest", cmap=plt.cm.binary)

plt.suptitle("Cluster centers learned by KMeans")
plt.show()
```

```
Digits data shape: (1797, 64)
Cluster centers shape: (10, 64)
```



We see that even without the labels, KMeans is able to find clusters whose centers are recognizable digits.

Next, we shall apply dimensioanlity reduction of MNIST handwritten datasets with PCA

```
data = digits.data
```

```
data
```

```
array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ..., 10.,  0.,  0.],
       [ 0.,  0.,  0., ..., 16.,  9.,  0.],
       ...,
```

```
[ 0.,  0.,  1., ...,  6.,  0.,  0.],
[ 0.,  0.,  2., ..., 12.,  0.,  0.],
[ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

Question 2. Complete the following lines of code for

Step 1. Dimensionality reduction with PCA. The 8x8=64 dimensional data need to be reduced to 2-dimensional.

Step 2. K-means clustering should be done on the reduced diemnsional data. Initial K value should be set to 10 like before.

Step 3. Visualization code is supplied below.

Step 1: Dimensionality reduction with PCA

```
from sklearn.decomposition import PCA
data = digits.data
model = PCA(n_components=2)
reduced_data = model.fit_transform(data)
```

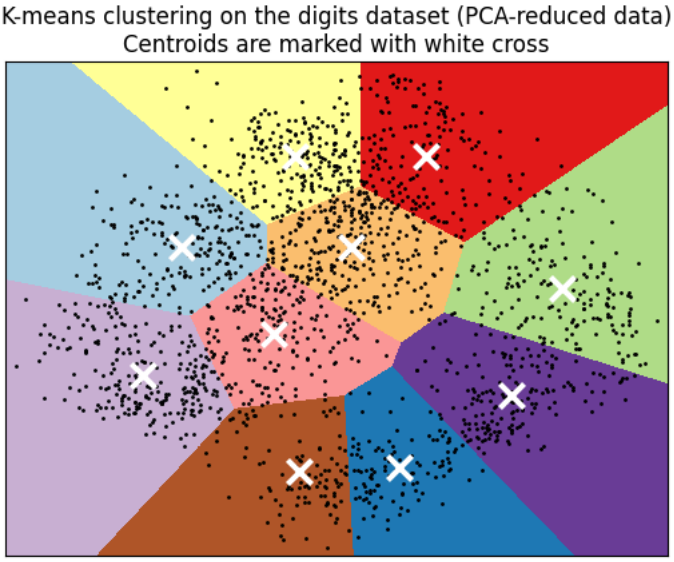
Step 2: K-means clustering on reduced data

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=10, random_state=0)
clusters = kmeans.fit_predict(reduced_data)
```

Data Visualization

We shall visualize the reduced diemnsion and cluster data (overlapped) with the help of the following code snippet.

```
import numpy as np
import matplotlib.pyplot as plt

h = 0.02  # step size of the mesh
x_min, x_max = reduced_data[:, 0].min() - 1, reduced_data[:, 0].max() + 1
y_min, y_max = reduced_data[:, 1].min() - 1, reduced_data[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))

clusters = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])
clusters = clusters.reshape(xx.shape)

plt.figure(1)
plt.clf()
plt.imshow(clusters, interpolation="nearest",
           extent=(xx.min(), xx.max(), yy.min(), yy.max()),
           cmap=plt.cm.Paired, aspect="auto", origin="lower")

plt.plot(reduced_data[:, 0], reduced_data[:, 1], "k.", markersize=2)

# Plot centroids
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker="x", s=169, linewidths=3,
            color="w", zorder=10)

plt.title("K-means clustering on the digits dataset (PCA-reduced data)\n"
          "Centroids are marked with white cross")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```



Question 3.

Find a high dimensional dataset of you choice. Show how you load the dataset. Do the basic exploratory data analysis to become familiar with the dataset.

```
# Step 1: Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Step 2: Load the dataset
file_path = "/content/student-mat.csv"   # Update the path if needed
data = pd.read_csv(file_path, sep=";")
data
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	5	4	4	5	4
391	MS	M	17	U	LE3	T	3	1	services	services	...	2	4	5	3	4	2
392	MS	M	21	R	GT3	T	1	1	other	other	...	5	5	3	3	3	3
393	MS	M	18	R	LE3	T	3	2	services	other	...	4	4	1	3	4	5
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	3	2	3	3	3	5

395 rows × 33 columns

```
# Step 3: Basic information about dataset
print("Shape of dataset:", data.shape)     # rows and columns
print("\nColumns:\n", data.columns.tolist())
print("\nInfo:\n")
print(data.info())
print("\nMissing values:\n", data.isnull().sum())
```

```
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
None

Missing values:
 school        0
sex            0
age            0
address        0
famsize        0
Pstatus        0
Medu           0
Fedu           0
Mjob           0
Fjob           0
reason         0
guardian       0
traveltime     0
studytime      0
failures       0
schoolsup      0
famsup         0
paid           0
activities     0
nursery        0
higher         0
internet       0
romantic       0
famrel         0
freetime       0
goout          0
Dalc           0
Walc           0
health         0
absences       0
G1             0
G2             0
G3             0
dtype: int64
```

```
# Step 4: Preview the data
print("\nFirst 5 rows:\n")
print(data.head())
```

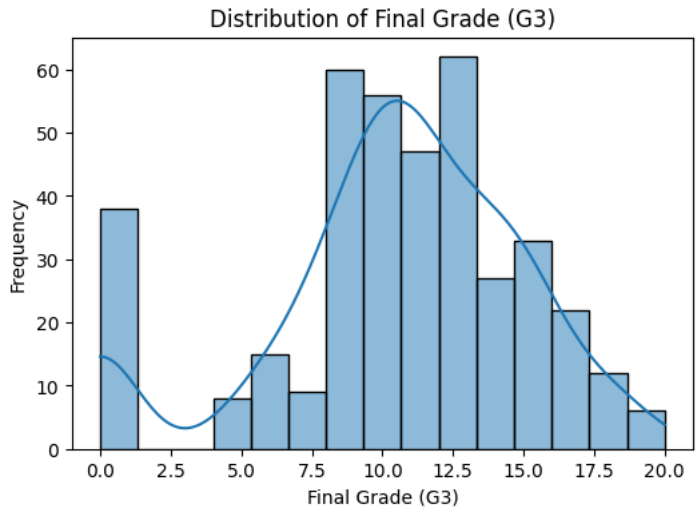
```
First 5 rows:

  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  ...  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher  ...
1     GP   F   17       U     GT3       T     1     1  at_home     other  ...
2     GP   F   15       U     LE3       T     1     1  at_home     other  ...
3     GP   F   15       U     GT3       T     4     2   health  services  ...
4     GP   F   16       U     GT3       T     3     3    other     other  ...

  famrel freetime  goout  Dalc  Walc health absences  G1  G2  G3
0      4        3      4     1     1      3        6   5   6   6
1      5        3      3     1     1      3        4   5   5   6
2      4        3      2     2     3      3       10   7   8  10
3      3        2      2     1     1      5        2  15  14  15
4      4        3      2     1     2      5        4   6  10  10

[5 rows x 33 columns]
```

```
print("\nStatistical summary:\n")
print(data.describe())
```

```
Statistical summary:

              age        Medu        Fedu  traveltime   studytime    failures  \
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000
mean    16.696203    2.749367    2.521519    1.448101    2.035443    0.334177
std      1.276043    1.094735    1.088201    0.697505    0.839240    0.743651
min     15.000000    0.000000    0.000000    1.000000    1.000000    0.000000
25%     16.000000    2.000000    2.000000    1.000000    1.000000    0.000000
50%     17.000000    3.000000    2.000000    1.000000    2.000000    0.000000
75%     18.000000    4.000000    3.000000    2.000000    2.000000    0.000000
```

```
max     22.000000    4.000000    4.000000    4.000000    4.000000    3.000000

           famrel    freetime       goout        Dalc        Walc      health  \
count  395.000000  395.000000  395.000000  395.000000  395.000000  395.000000
mean     3.944304    3.235443    3.108861    1.481013    2.291139    3.554430
std      0.896659    0.998862    1.113278    0.890741    1.287897    1.390303
min      1.000000    1.000000    1.000000    1.000000    1.000000    1.000000
25%      4.000000    3.000000    2.000000    1.000000    1.000000    3.000000
50%      4.000000    3.000000    3.000000    1.000000    2.000000    4.000000
75%      5.000000    4.000000    4.000000    2.000000    3.000000    5.000000
max      5.000000    5.000000    5.000000    5.000000    5.000000    5.000000

          absences          G1          G2          G3
count  395.000000  395.000000  395.000000  395.000000
mean     5.708861   10.908861   10.713924   10.415190
std      8.003096    3.319195    3.761505    4.581443
min      0.000000    3.000000    0.000000    0.000000
25%      0.000000    8.000000    9.000000    8.000000
50%      4.000000   11.000000   11.000000   11.000000
75%      8.000000   13.000000   13.000000   14.000000
max     75.000000   19.000000   19.000000   20.000000
```

```
plt.figure(figsize=(6,4))
sns.histplot(data['G3'], kde=True, bins=15)
plt.title("Distribution of Final Grade (G3)")
plt.xlabel("Final Grade (G3)")
plt.ylabel("Frequency")
plt.show()
```




```
data
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
390	MS	M	20	U	LE3	A	2	2	services	services	...	5	5	4	4	5	4
391	MS	M	17	U	LE3	T	3	1	services	services	...	2	4	5	3	4	2
392	MS	M	21	R	GT3	T	1	1	other	other	...	5	5	3	3	3	3
393	MS	M	18	R	LE3	T	3	2	services	other	...	4	4	1	3	4	5
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	3	2	3	3	3	5

395 rows × 33 columns

```
# List of categorical columns
categorical_cols = data.select_dtypes(include=['object']).columns.tolist()

# Remove 'internet' from the list
categorical_to_remove = [col for col in categorical_cols if col != 'internet']
```

```
# Drop the unneeded categorical features
data_cleaned = data.drop(columns=categorical_to_remove)

# Check remaining columns
print("Remaining columns:\n", data_cleaned.columns.tolist())
print("Shape after removing categorical features:", data_cleaned.shape)
```

```
Remaining columns:
 ['age', 'Medu', 'Fedu', 'traveltime', 'studytime', 'failures', 'internet', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'hea
Shape after removing categorical features: (395, 17)
```

```
data= data_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         395 non-null    int64
 1   Medu        395 non-null    int64
 2   Fedu        395 non-null    int64
 3   traveltime  395 non-null    int64
 4   studytime   395 non-null    int64
 5   failures    395 non-null    int64
 6   internet    395 non-null    object
 7   famrel      395 non-null    int64
 8   freetime    395 non-null    int64
 9   goout       395 non-null    int64
 10  Dalc        395 non-null    int64
 11  Walc        395 non-null    int64
 12  health      395 non-null    int64
 13  absences    395 non-null    int64
 14  G1          395 non-null    int64
 15  G2          395 non-null    int64
 16  G3          395 non-null    int64
dtypes: int64(16), object(1)
memory usage: 52.6+ KB
```

```
print(data_cleaned['internet'].value_counts())
```

```
internet
yes    329
no      66
Name: count, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder

# Copy dataset
df = data_cleaned.copy()

# Initialize LabelEncoder
le = LabelEncoder()

# Fit and transform 'internet'
data_cleaned['internet'] = le.fit_transform(data_cleaned['internet'])

# Check result
print(data_cleaned[['internet']].head())
```

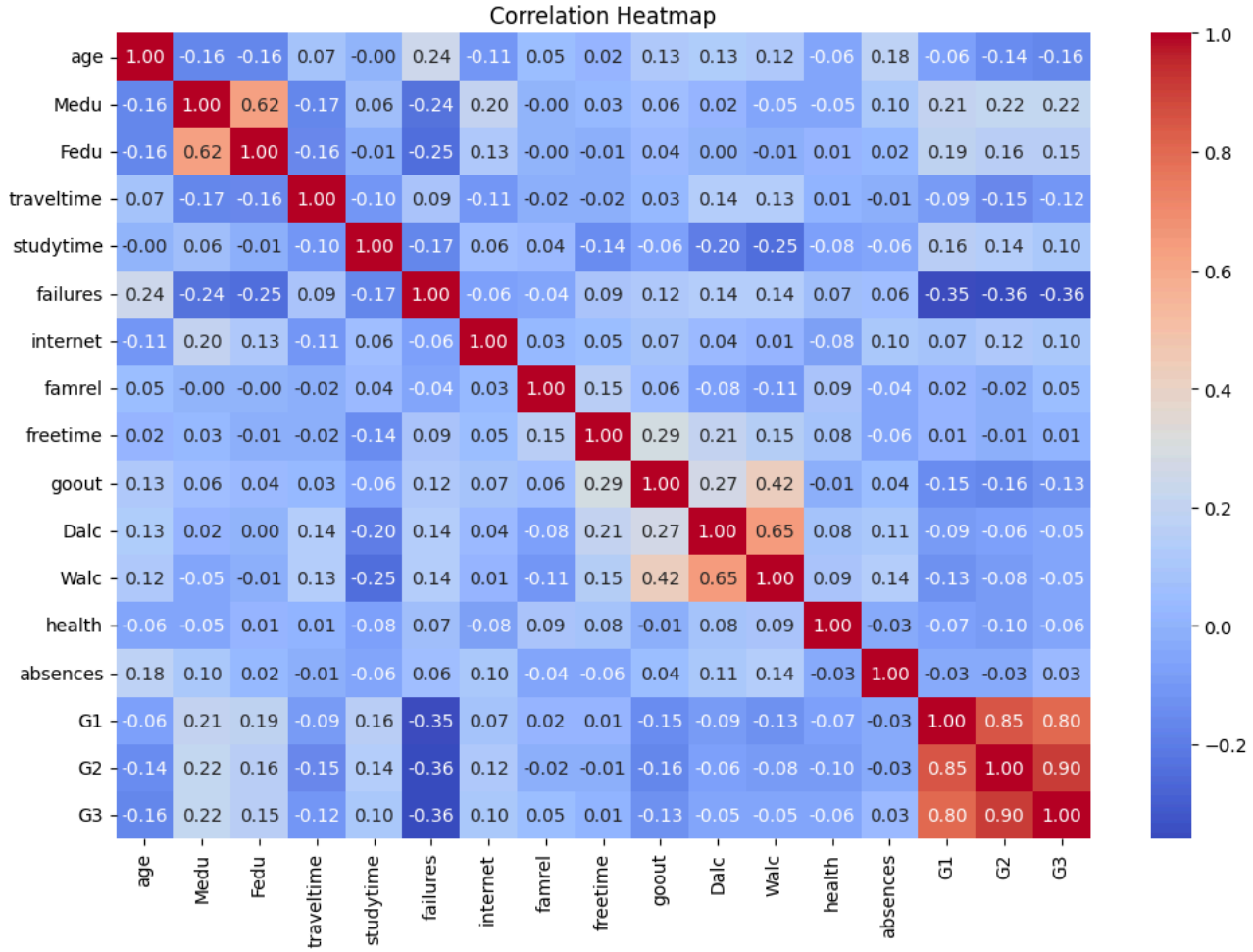
```
   internet
0         0
1         1
2         1
3         1
4         0
```

```
data= data_cleaned.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 17 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   age         395 non-null    int64
 1   Medu        395 non-null    int64
 2   Fedu        395 non-null    int64
 3   traveltime  395 non-null    int64
 4   studytime   395 non-null    int64
 5   failures    395 non-null    int64
 6   internet    395 non-null    int64
 7   famrel      395 non-null    int64
 8   freetime    395 non-null    int64
 9   goout       395 non-null    int64
```

```
 10  Dalc        395 non-null    int64
 11  Walc        395 non-null    int64
 12  health      395 non-null    int64
 13  absences    395 non-null    int64
 14  G1          395 non-null    int64
 15  G2          395 non-null    int64
 16  G3          395 non-null    int64
dtypes: int64(17)
memory usage: 52.6 KB
```

```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



Question 4.

Next, the objective would be to reduce the dimension of your dataset and do the clustering on it. Complete the following code for clustering in an object-oriented manner. Do the exact process as above: PCA dimension reduction followed by clustering.

A template code is provided below for your guidance.

```
class YourDataClustering:
    def __init__(self, n_clusters=3):
        self.n_clusters = n_clusters
        self.data = ______
        self.labels = ______
        self.kmeans = ______
        self.scaled_data = ______

    def load_data(self):
        """Load the Iris dataset"""
        iris = ________.________()
        self.data = iris.________
        return ________
```

```
    def preprocess_data(self):
        """Standardize the dataset"""
        scaler = _________.__________()
        self.scaled_data = scaler._________(_________)
        return _________

    def apply_kmeans(self):
        """Apply KMeans clustering"""
        self.kmeans = _________._________(n_clusters=self.n_clusters, random_state=42)
        self.labels = self.kmeans._________(_________)
        return _________

    def evaluate_clusters(self):
        """Compute silhouette score"""
        score = _________._________(_________, _________)
        print(f"Silhouette Score: {score:.3f}")
        return _________

    def visualize_clusters_matplotlib(self):
        """Visualize clustering result using Matplotlib"""
        plt.scatter(_________[:, 0], _________[:, 1], c=_________, cmap='viridis')
        plt.title("KMeans Clustering on Iris Dataset (Matplotlib)")
        plt.xlabel("_________")
        plt.ylabel("_________")
        plt.show()

    def visualize_clusters_opencv(self):
        """Visualize clustering result using OpenCV"""
        canvas = np.ones((______, ______, 3), dtype=np.uint8) * 255
        colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255)]
        scaled = (_________[:, :2] * 100 + 250).astype(int)

        for i, point in enumerate(_________):
            cv2.circle(canvas, tuple(point), 5, colors[_________ % 3], -1)

        cv2.imshow("KMeans Clustering (OpenCV)", _________)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

The following code executes all teh parts of teh complete system.

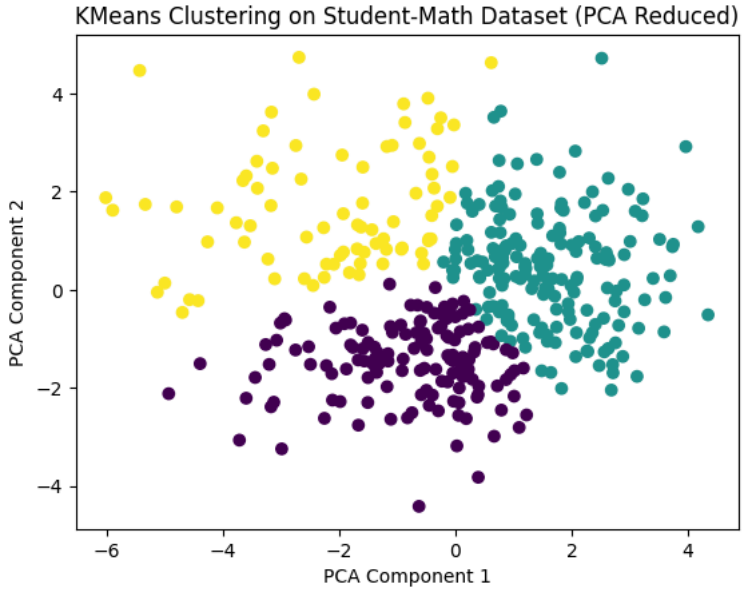
```
# Step 1: Create clustering object
  clustering = _________(n_clusters=3)

  # Step 2: Load dataset
  data = clustering._________()

  # Step 3: Preprocess dataset
  scaled_data = clustering._________()

  # Step 4: Apply KMeans clustering
  labels = clustering._________()

  # Step 5: Evaluate clusters
  score = clustering._________()

  # Step 6: Visualize with Matplotlib
  clustering._________()

  # Step 7: Visualize with OpenCV
  clustering._________()
```

```
  import numpy as np
  import pandas as pd
  import matplotlib.pyplot as plt
  import cv2

  from sklearn.preprocessing import StandardScaler, LabelEncoder
```

```
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score


class YourDataClustering:
    def __init__(self, n_clusters=3):
        self.n_clusters = n_clusters
        self.data = None
        self.labels = None
        self.kmeans = None
        self.scaled_data = None
        self.reduced_data = None
        self._label_encoders = {}

    def load_data(self):
        """Load the Student Performance (Math) dataset from CSV"""
        df = pd.read_csv("student-mat.csv", sep=';')
        self.data = df
        return df

    def preprocess_data(self):
        """Encode categorical columns, standardize the dataset and apply PCA to reduce to 2D"""
        df = self.data.copy()

        # Label encode string (object) columns
        for col in df.select_dtypes(include=['object']).columns:
            le = LabelEncoder()
            df[col] = le.fit_transform(df[col].astype(str))
            self._label_encoders[col] = le

        # Convert to float and scale
        numeric = df.astype(float).values
        scaler = StandardScaler()
        self.scaled_data = scaler.fit_transform(numeric)

        # PCA to 2 dimensions
        pca = PCA(n_components=2)
        self.reduced_data = pca.fit_transform(self.scaled_data)

        return self.reduced_data

    def apply_kmeans(self):
        """Apply KMeans clustering"""
        self.kmeans = KMeans(n_clusters=self.n_clusters, random_state=42)
        self.labels = self.kmeans.fit_predict(self.reduced_data)
        return self.labels

    def evaluate_clusters(self):
        """Compute silhouette score"""
        score = silhouette_score(self.reduced_data, self.labels)
        print(f"Silhouette Score: {score:.3f}")
        return score

    def visualize_clusters_matplotlib(self):
        """Visualize clustering result using Matplotlib"""
        plt.scatter(self.reduced_data[:, 0], self.reduced_data[:, 1], c=self.labels, cmap='viridis')
        plt.title("KMeans Clustering on Student-Math Dataset (PCA Reduced)")
        plt.xlabel("PCA Component 1")
        plt.ylabel("PCA Component 2")
        plt.show()

    def visualize_clusters_opencv(self):
        """Visualize clustering result using OpenCV"""
        canvas = np.ones((500, 500, 3), dtype=np.uint8) * 255
        colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255)]
        scaled = (self.reduced_data[:, :2] * 100 + 250).astype(int)

        for i, point in enumerate(scaled):
            cv2.circle(canvas, tuple(point), 5, colors[self.labels[i] % 3], -1)

        cv2.imshow("KMeans Clustering (OpenCV)", canvas)
        cv2.waitKey(0)
        cv2.destroyAllWindows()


# -------------------------
# Running the Full System
# -------------------------

# Step 1: Create clustering object
clustering = YourDataClustering(n_clusters=3)
```

```
# Step 2: Load dataset
data = clustering.load_data()

# Step 3: Preprocess dataset (encode, scale, PCA)
scaled_data = clustering.preprocess_data()

# Step 4: Apply KMeans clustering
labels = clustering.apply_kmeans()

# Step 5: Evaluate clusters
score = clustering.evaluate_clusters()

# Step 6: Visualize with Matplotlib
clustering.visualize_clusters_matplotlib()

# Step 7: Visualize with OpenCV (uncomment to show OpenCV window)
# clustering.visualize_clusters_opencv()
```

```
Silhouette Score: 0.384
```



Start coding or generate with AI.