Project Title: Unsupervised Learning with Dimensionality Reduction and Clustering

Student's Name: Arghyadip Bain

Course: B.Sc (Hons) [Mathematics] / 2nd year

Institute: Bangabasi College (Day), Calcutta University

Period of Internship: 25th August 2025 - 19th September 2025

Report submitted to: IDEAS – Institute of Data Engineering, Analytics and Science Foundation , ISI Kolkata

Abstract :

This project demonstrates the application of unsupervised learning combined with dimensionality reduction on high-dimensional datasets.The MNIST handwritten digits dataset is used as the primary example to showcase clustering without labels.
K-Means clustering is first applied to the original 64-dimensional data to identify natural groupings of the digits.Subsequently, Principal Component Analysis (PCA) reduces the dimensionality to two components, enabling intuitive visualization of the data distribution. The reduced data is then reclustered using K-Means to assess the preservation of structure after dimensionality reduction.

Visualization of clusters in the reduced feature space highlights how similar samples are grouped together even without supervision.The methodology is also extended to another high-dimensional dataset to demonstrate the flexibility and scalability of the approach.Overall, the project provides a clear, practical pipeline for unsupervised learning, from data preparation and dimensionality reduction to clustering and visualization.

Introduction :

The internship introduced participants to core concepts of data preprocessing, exploratory data analysis, dimensionality reduction, and unsupervised clustering. In this project, I implemented a complete pipeline to handle high-dimensional data.
Technologies used: Python, scikit-learn, Matplotlib, OpenCV.

Topics Covered During First Two Weeks of Internship

1. Basics of Python programming (variables, loops, functions).

2. Introduction to data handling with Pandas and NumPy.

3. Data visualization using Matplotlib and Seaborn.

4. Fundamentals of Machine Learning – supervised vs. unsupervised learning.

5. Overview of classification algorithms (Logistic Regression, Decision Trees, Random Forest).

6. Concept of train/test split and cross-validation.

7. Evaluation metrics: accuracy, precision, recall, F1-score, confusion matrix.

8. Feature selection, correlation analysis, and handling redundant features.

9. Use of Jupyter Notebook/Colab for running ML projects.

10. Introduction to real-world datasets from UCI Machine Learning Repository.

Project Objectives

- Apply K-Means clustering to unlabelled datasets.
- Reduce dimensionality of high-dimensional data using PCA or Truncated SVD.
- Evaluate cluster quality using Silhouette Score.
- Visualize clustering results using Matplotlib and OpenCV.
- Demonstrate the methodology on both image and text datasets.

Methodology :

1. Data Collection: Image dataset (Digits dataset from scikit-learn) and Text dataset (20 Newsgroups from scikit-learn).
2. Preprocessing: Standardize numerical features (image data) and convert text to TF-IDF features(text data).
3. Dimensionality Reduction: PCA for image data, TruncatedSVD for text data.
4. Clustering: K-Means with appropriate number of clusters.
5. Evaluation: Silhouette Score computed to assess cluster separation.
6. Visualization: Matplotlib scatter plots and OpenCV drawing for cluster visualization.

Code for all steps was written in Python using scikit-learn.

Assignment Questions and Answers

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

```
from sklearn.cluster from import
KMeans matplotlib import pyplot #
kmeans = ... # clusters = ...
kmeans = KMeans(n_clusters= as plt
, random_state=0) clusters = 10
kmeans.fit_predict(digits.data)
print
(kmeans.cluster_centers_.shape)
```

```
(10, 64) fig, ax = plt.subplots(2 ,
5 , 3)) centers = , figsize=(8
kmeans.cluster_centers_.reshape(
for axi, center in zip (ax.flat,
centers): axi. set (xticks= [],
yticks= 10, 8 , 8) [])
axi.imshow(center, interpolation=
```

ANSWER :

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

digits = load_digits()
kmeans = KMeans(n_clusters=10, random_state=0)
clusters = kmeans.fit_predict(digits.data)
print(kmeans.cluster_centers_.shape)

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.show()
```

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.

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

```
# Step size of the mesh. Decrease to
increase the quality of the VQ. h = 0.02 #
point in the mesh [x_min, x_max]x[y_min,
y_max]. # Plot the decision boundary. For
that, we will assign a color to each 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)) # Obtain labels
for each point in mesh. Use last trained
model. clusters =
kmeans.predict(np.c_[xx.ravel(),
yy.ravel()]) # Put the result into a color
plot 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 the centroids as a white X 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()
```

ANSWER :

```
from sklearn.datasets import load_digits
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np

# Step 1: Load the digits data
digits = load_digits()
data = digits.data

# Step 2: Dimensionality reduction with PCA to 2 components
model = PCA(n_components=2)
reduced_data = model.fit_transform(data)

# Step 3: K-Means clustering with 10 clusters on reduced data
kmeans = KMeans(n_clusters=10, random_state=0)
clusters = kmeans.fit_predict(reduced_data)

# Visualization of decision boundaries and clusters
h = 0.02  # step size in 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_mesh = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])
clusters_mesh = clusters_mesh.reshape(xx.shape)

plt.figure(1)
plt.clf()
plt.imshow(
    clusters_mesh,
    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)

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)") # Fixed the unterminated string literal
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

After applying PCA, the original 64-dimensional digit images are reduced to just two principal components, making it possible to plot and visually inspect the data. K-Means with 10 clusters is then fitted on this reduced representation, and the decision boundaries along with the cluster centroids are shown in the visualization. This plot demonstrates that even after strong dimensionality reduction, the algorithm is still able to separate most of the digits into meaningful groups without using labels.

Question 3: Find a high dimensional dataset of your choice Show how you load the dataset. Do the basic exploratory data analysis to become familiar with the dataset

ANSWER :

For this task, I selected the **20 Newsgroups text dataset** from scikit-learn. It contains thousands of news articles spread across 20 different topics. Because each document is transformed into a large TF-IDF feature vector, it is a high-dimensional dataset suitable for unsupervised learning.

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

# 1. Load text data (downloads once, then caches locally)
newsgroups = fetch_20newsgroups(
    subset='all',
    remove=('headers', 'footers', 'quotes')
)
texts = newsgroups.data
print("Number of documents:", len(texts))

# 2. Convert text to high-dimensional TF-IDF features
vectorizer = TfidfVectorizer(
    max_features=2000,
    stop_words='english'
)
X = vectorizer.fit_transform(texts)
print("TF-IDF feature matrix shape:", X.shape)

# 3. Show some feature names
print("Sample feature names:", vectorizer.get_feature_names_out()[:10])

# 4. Basic exploratory data analysis
df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())
print(df.describe())  # summary of feature distributions
```

OUTPUT :

```
Number of documents: 18846
TF-IDF feature matrix shape: (18846, 2000)
Sample feature names: ['00' '000' '01' '02' '03' '04' '05' '06' '0d' '0t']
                 00            000             01             02             03  \
count  18846.000000   18846.000000   18846.000000   18846.000000   18846.000000
mean       0.004669       0.003484       0.000839       0.000965       0.000616
std        0.043082       0.031003       0.014151       0.015799       0.014567
min        0.000000       0.000000       0.000000       0.000000       0.000000
25%        0.000000       0.000000       0.000000       0.000000       0.000000
50%        0.000000       0.000000       0.000000       0.000000       0.000000
75%        0.000000       0.000000       0.000000       0.000000       0.000000
max        0.920820       0.879405       0.522280       0.615895       0.677259

                 04             05             06             0d             0t  \
count  18846.000000   18846.000000   18846.000000   18846.000000   18846.000000
mean       0.000781       0.000547       0.000386       0.000036       0.000029
std        0.013910       0.012155       0.008455       0.001752       0.001443
min        0.000000       0.000000       0.000000       0.000000       0.000000
25%        0.000000       0.000000       0.000000       0.000000       0.000000
50%        0.000000       0.000000       0.000000       0.000000       0.000000
75%        0.000000       0.000000       0.000000       0.000000       0.000000
max        0.561700       0.702706       0.480461       0.144160       0.138822

       ...          xview           yeah           year          years  \
count  ...   18846.000000   18846.000000   18846.000000   18846.000000
mean   ...       0.000649       0.002761       0.008630       0.007967
std    ...       0.015507       0.030017       0.041860       0.033958
min    ...       0.000000       0.000000       0.000000       0.000000
25%    ...       0.000000       0.000000       0.000000       0.000000
50%    ...       0.000000       0.000000       0.000000       0.000000
75%    ...       0.000000       0.000000       0.000000       0.000000
max    ...       0.742670       0.834497       0.678244       0.421725
```

```
                yes     yesterday          york         young           zip  \
count  18846.000000  18846.000000  18846.000000  18846.000000  18846.000000   
mean       0.006109      0.001377      0.001553      0.001753      0.001539   
std        0.034865      0.018469      0.017257      0.019135      0.023874   
min        0.000000      0.000000      0.000000      0.000000      0.000000   
25%        0.000000      0.000000      0.000000      0.000000      0.000000   
50%        0.000000      0.000000      0.000000      0.000000      0.000000   
75%        0.000000      0.000000      0.000000      0.000000      0.000000   
max        1.000000      0.820353      0.560961      0.685554      0.713797   

               zone  
count  18846.000000  
mean       0.001029  
std        0.018151  
min        0.000000  
25%        0.000000  
50%        0.000000  
75%        0.000000  
max        0.714897  

[8 rows x 2000 columns]
```

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.

ANSWER :

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import numpy as np
import cv2

class TextDataClustering:
    def __init__(self, n_clusters=20):
        self.n_clusters = n_clusters
        self.texts = None
        self.data = None
        self.reduced_data = None
        self.kmeans = None
        self.labels = None
        self.vectorizer = None

    def load_data(self):
        """Load the 20 Newsgroups dataset"""
        newsgroups = fetch_20newsgroups(subset='all', remove=('headers',
'footers', 'quotes'))
        self.texts = newsgroups.data
        return self.texts

    def vectorize_data(self):
        """Convert text into TF-IDF features"""
        self.vectorizer = TfidfVectorizer(max_features=2000,
stop_words='english')
        self.data = self.vectorizer.fit_transform(self.texts)
        return self.data

    def reduce_dimensionality(self):
        """Reduce dimensionality with Truncated SVD (2 components)"""
        svd = TruncatedSVD(n_components=2, random_state=42)
        self.reduced_data = svd.fit_transform(self.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.figure(figsize=(8, 6))
        plt.scatter(self.reduced_data[:, 0], self.reduced_data[:, 1],
c=self.labels, cmap='tab20', s=5)
        plt.title("KMeans Clustering on Text Dataset (2D SVD Reduced)")
        plt.xlabel("Component 1")
        plt.ylabel("Component 2")
        plt.show()

    def visualize_clusters_opencv(self):
        """Visualize clustering result using OpenCV"""
        canvas = np.ones((600, 600, 3), dtype=np.uint8) * 255
        colors = [tuple(np.random.randint(0, 255, 3).tolist()) for _ in
range(self.n_clusters)]
        scaled = (self.reduced_data * 20 + 300).astype(int)
        for i, point in enumerate(scaled):
            cv2.circle(canvas, tuple(point), 2, colors[self.labels[i] %
len(colors)], -1)

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

# Running the complete pipeline
clustering = TextDataClustering(n_clusters=20)
texts = clustering.load_data()
X = clustering.vectorize_data()
reduced_data = clustering.reduce_dimensionality()
labels = clustering.apply_kmeans()
score = clustering.evaluate_clusters()
clustering.visualize_clusters_matplotlib()
# clustering.visualize_clusters_opencv() # uncomment to view OpenCV window
```

Conclusion

This project successfully demonstrated how unsupervised learning techniques like PCA and K-Means can be combined to uncover structure in high-dimensional datasets. Both image and text data showed meaningful clustering after dimensionality reduction. Future work may involve comparing alternative clustering algorithms (DBSCAN, Agglomerative Clustering) or using more advanced embeddings for text data.

Appendices

References:

- AI software like DeepSeek , Chat gpt , Gemini
- scikit-learn official documentation
- MNIST / Digits dataset documentation
- 20 Newsgroups dataset documentation

Github Link :

https://github.com/Arghyadip881/internship-project-2025