Name: Ujwal Sahu

Branch: BVoc (AI&DS)

Division: B

Roll no: 34

EXPERIMENT NO 9

Title: To implement K-Means algorithm and Dimensionality Reduction using PCA for a given dataset.

Tools: Anaconda (Jupyter Notebook).

Theory:

K-Means Clustering Algorithm: K-Means is an unsupervised learning algorithm used to group data into K distinct clusters based on similarity.

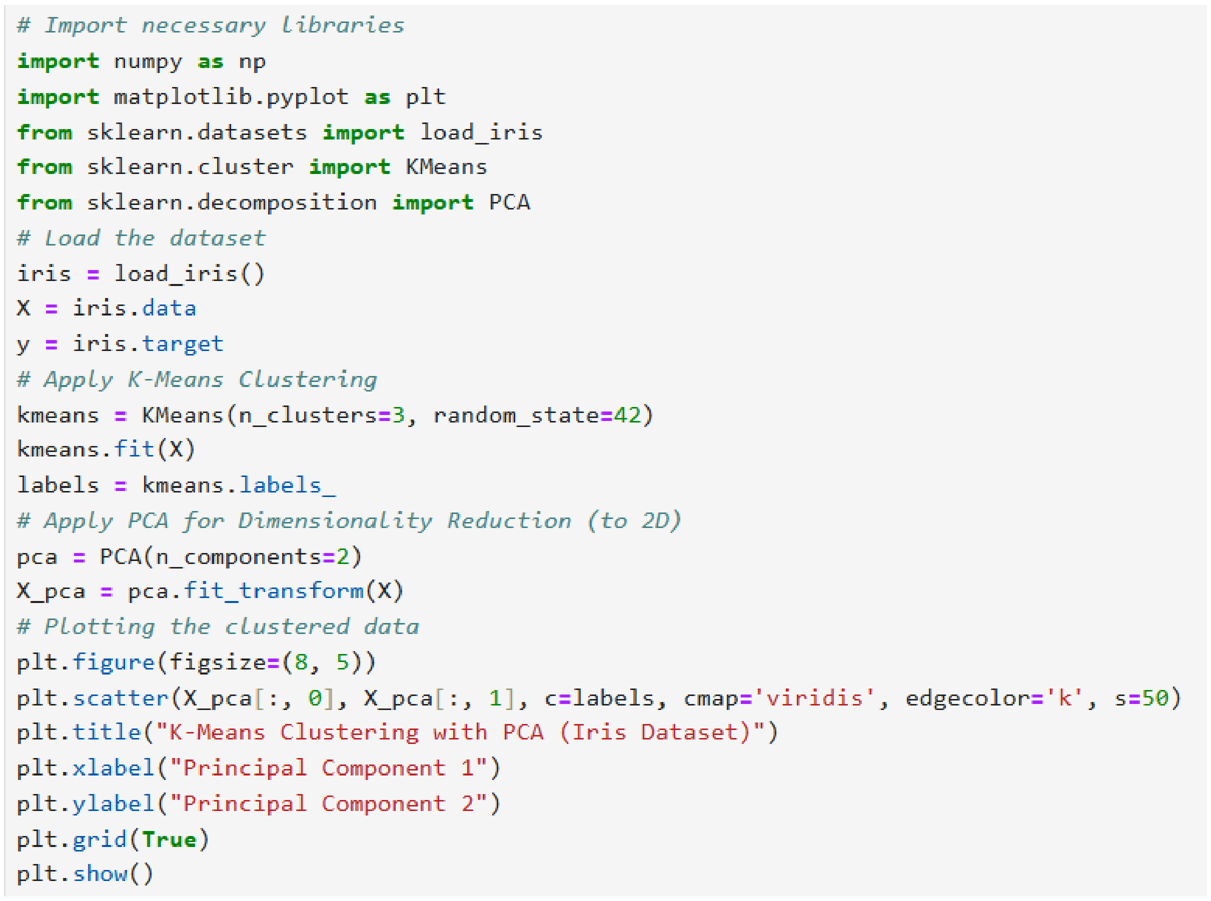
Principal Component Analysis (PCA): PCA is a dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional form while preserving as much variability as possible.

Code:

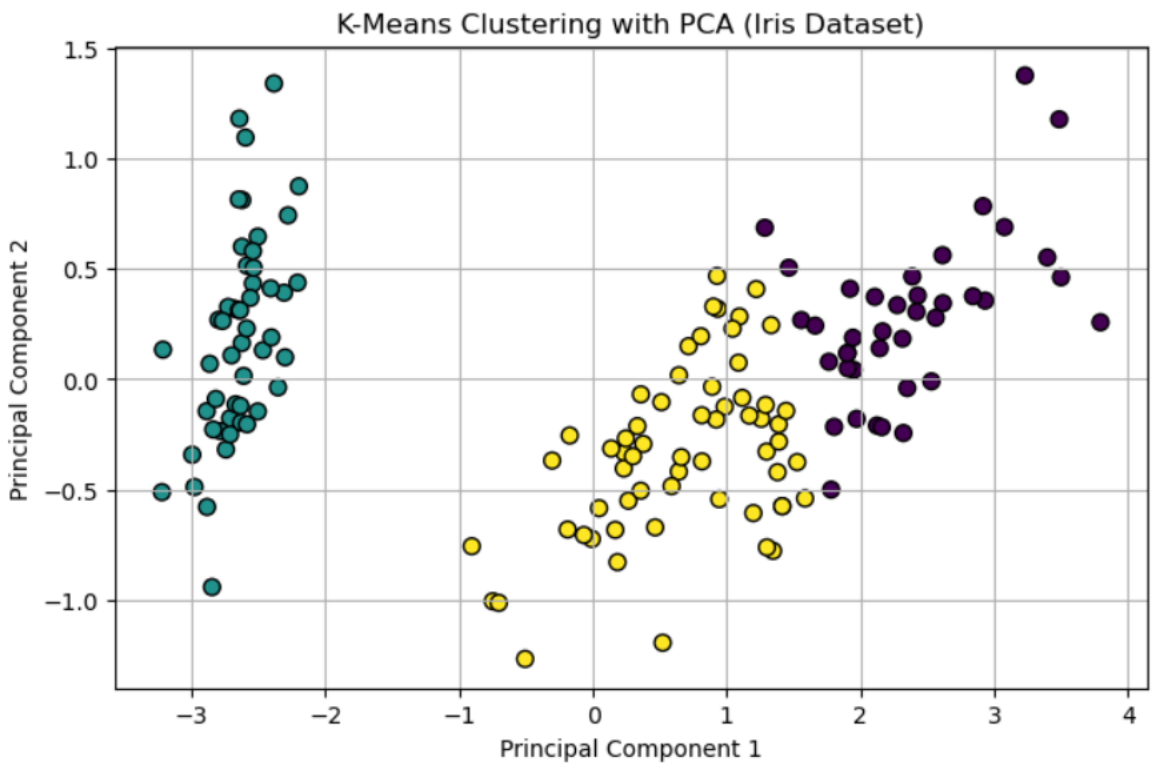
```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
# Load the dataset
iris = load_iris()
X = iris.data
y = iris.target
# Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_
# Apply PCA for Dimensionality Reduction (to 2D)
pca = PCA(n_components=2)
```

```
X_pca = pca.fit_transform(X)

# Plotting the clustered data

plt.figure(figsize=(8, 5))

plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis', edgecolor='k', s=50)

plt.title("K-Means Clustering with PCA (Iris Dataset)")

plt.xlabel("Principal Component 1")

plt.ylabel("Principal Component 2")

plt.grid(True)

plt.show()
```

Output:



```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
# Load the dataset
iris = load_iris()
X = iris.data
y = iris.target
# Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
labels = kmeans.labels_
# Apply PCA for Dimensionality Reduction (to 2D)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
# Plotting the clustered data
plt.figure(figsize=(8, 5))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=labels, cmap='viridis', edgecolor='k', s=50)
plt.title("K-Means Clustering with PCA (Iris Dataset)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.grid(True)
plt.show()
```



Conclusion: In this experiment, we successfully understood and implemented the Gradient Descent algorithm for optimizing model parameters, the Grid Search method for hyperparameter tuning, and various model evaluation techniques to assess model performance. Gradient Descent helped in minimizing the loss function, while Grid Search provided the best combination of hyperparameters. Evaluation metrics such as accuracy, precision, and mean squared error enabled us to measure how well the model performs on unseen data. These techniques together are essential for building accurate and efficient machine learning models.

For Faculty Use

Correction Parameters	Formative Assessment [40%]	Timely completion of Practical [40%]	Attendance / Learning Attitude [20%]	
Marks Obtained				