CSBB 311 : MACHINE LEARNING

LAB ASSIGNMENT 9 : Feature Extraction Using PCA

Submitted By:

Name: KARTIK MITTAL

Roll No: 221210056

Branch: CSE

Semester: 5th Sem

Group: 2

Submitted To: Dr. Preeti Verma

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

2024

Code -

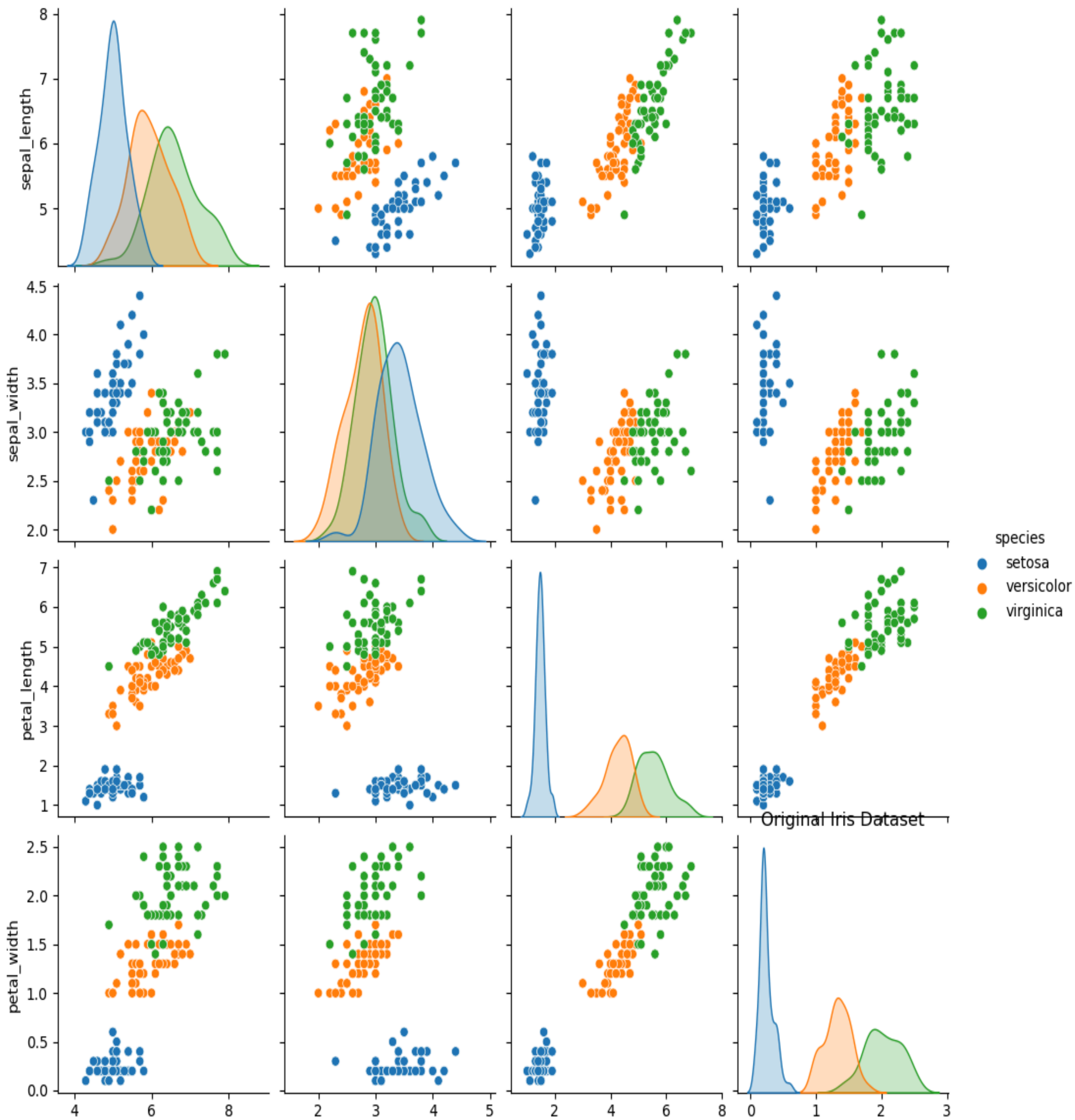
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D

# Load the Iris dataset from a CSV file
df = pd.read_csv('iris.csv')

# Check the first few rows to understand the dataset
print(df.head())

# Separate features and target labels
X = df.drop('species', axis=1)  # Features
y = df['species']  # Target labels (species)

# Visualize original data using pairplot
sns.pairplot(df, hue='species')
plt.title("Original Iris Dataset")
plt.show()

# Apply PCA and reduce to 1 component
pca = PCA(n_components=1)
X_pca_1 = pca.fit_transform(X)
plt.figure(figsize=(8, 6))
plt.scatter(X_pca_1, np.zeros_like(X_pca_1), c=y.map({'setosa': 0, 'versicolor': 1,
                                                       'virginica': 2}), cmap='viridis')
plt.title("PCA with 1 Component")
plt.xlabel("Principal Component 1")
plt.show()
```

```python
# Apply PCA and reduce to 2 components
pca = PCA(n_components=2)
X_pca_2 = pca.fit_transform(X)
plt.figure(figsize=(8, 6))
plt.scatter(X_pca_2[:, 0], X_pca_2[:, 1], c=y.map({'setosa': 0, 'versicolor': 1,
                                                    'virginica': 2}), cmap='viridis')
plt.title("PCA with 2 Components")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()

# Apply PCA and reduce to 3 components
pca = PCA(n_components=3)
X_pca_3 = pca.fit_transform(X)
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X_pca_3[:, 0], X_pca_3[:, 1], X_pca_3[:, 2], c=y.map({'setosa': 0, 'versicolor': 1,
                                                                  'virginica': 2}), cmap='viridis')
ax.set_title("PCA with 3 Components")
ax.set_xlabel("Principal Component 1")
ax.set_ylabel("Principal Component 2")
ax.set_zlabel("Principal Component 3")
plt.show()

# Apply PCA and reduce to all components (4 components)
pca = PCA(n_components=4)
X_pca_4 = pca.fit_transform(X)

# Plot the explained variance ratio for each component (how much each component contributes)
plt.figure(figsize=(8, 6))
plt.bar(range(1, 5), pca.explained_variance_ratio_, alpha=0.6, color='g', label='Explained variance ratio')
plt.xlabel('Principal Components')
plt.ylabel('Variance Ratio')
plt.title('Variance Explained by Each Principal Component')
plt.show()
```

```python
# **Find the eigenvalues** (explained variance) for each principal component
eigenvalues = pca.explained_variance_
print("Eigenvalues (Explained Variance) for each component:")
print(eigenvalues)

# Plot the eigenvalues (Explained Variance)
plt.figure(figsize=(8, 6))
plt.bar(range(1, 5), eigenvalues, alpha=0.6, color='b', label='Eigenvalues')
plt.xlabel('Principal Components')
plt.ylabel('Eigenvalue (Explained Variance)')
plt.title('Eigenvalues (Explained Variance) for Each Principal Component')
plt.show()

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# List to store accuracy results
accuracies = []

# Loop over 1 to 4 principal components
for n in range(1, 5):
    # Apply PCA
    pca = PCA(n_components=n)
    X_train_pca = pca.fit_transform(X_train)
    X_test_pca = pca.transform(X_test)

    # KNN classifier
    knn = KNeighborsClassifier(n_neighbors=3)
    knn.fit(X_train_pca, y_train)

    # Predict and evaluate accuracy
    y_pred = knn.predict(X_test_pca)
    accuracy = accuracy_score(y_test, y_pred)
    accuracies.append(accuracy)
```

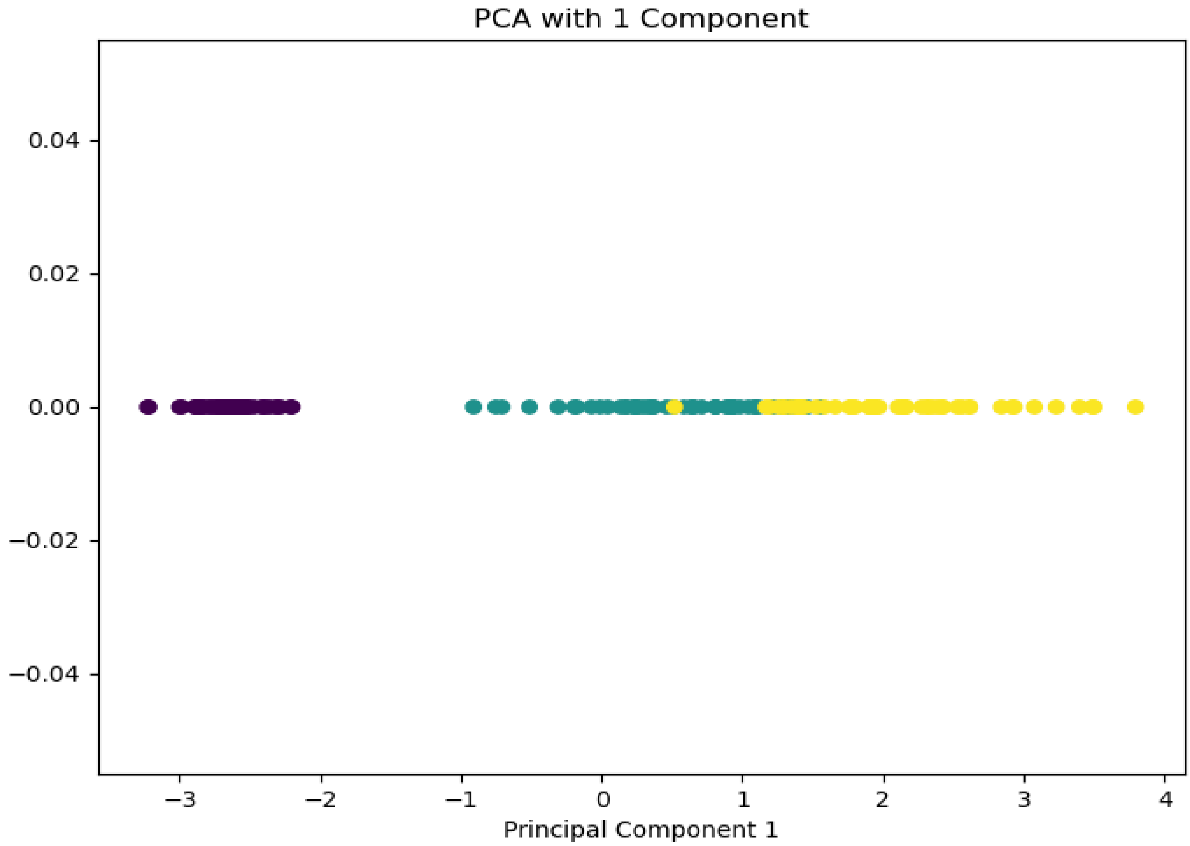
```python
# Plot accuracy vs number of PCA components
plt.figure(figsize=(8, 6))
plt.plot(range(1, 5), accuracies, marker='o', color='b')
plt.title("KNN Accuracy vs Number of PCA Components")
plt.xlabel("Number of PCA Components")
plt.ylabel("Accuracy")
plt.xticks(range(1, 5))
plt.grid(True)
plt.show()
```

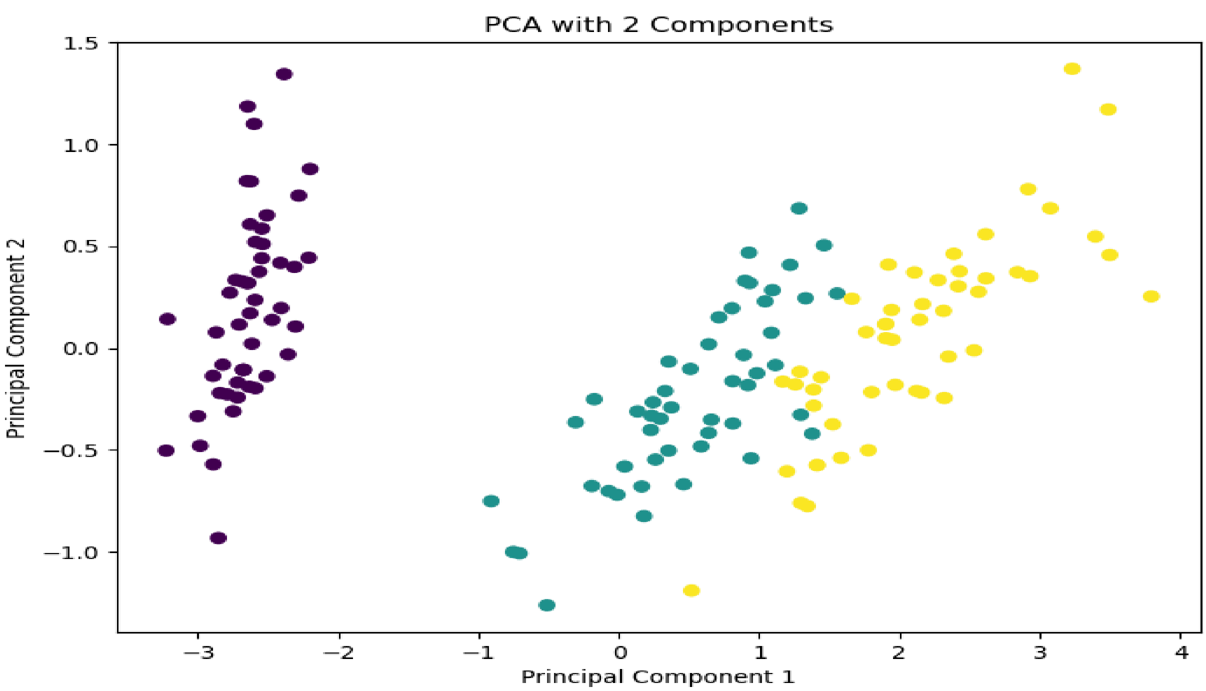
Output -

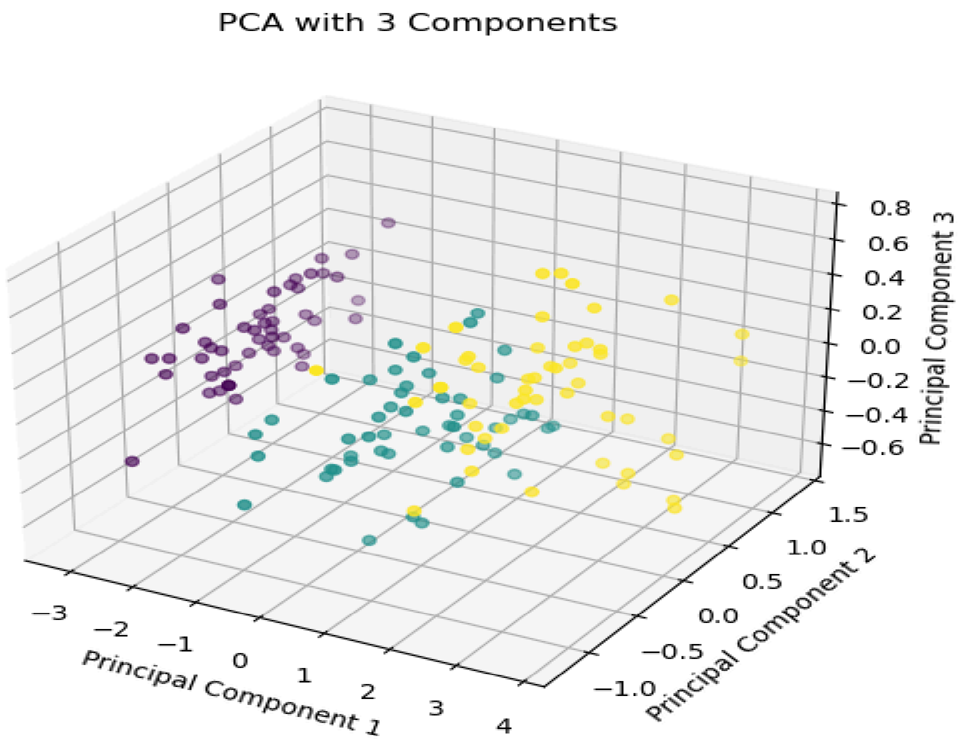


PairPlot

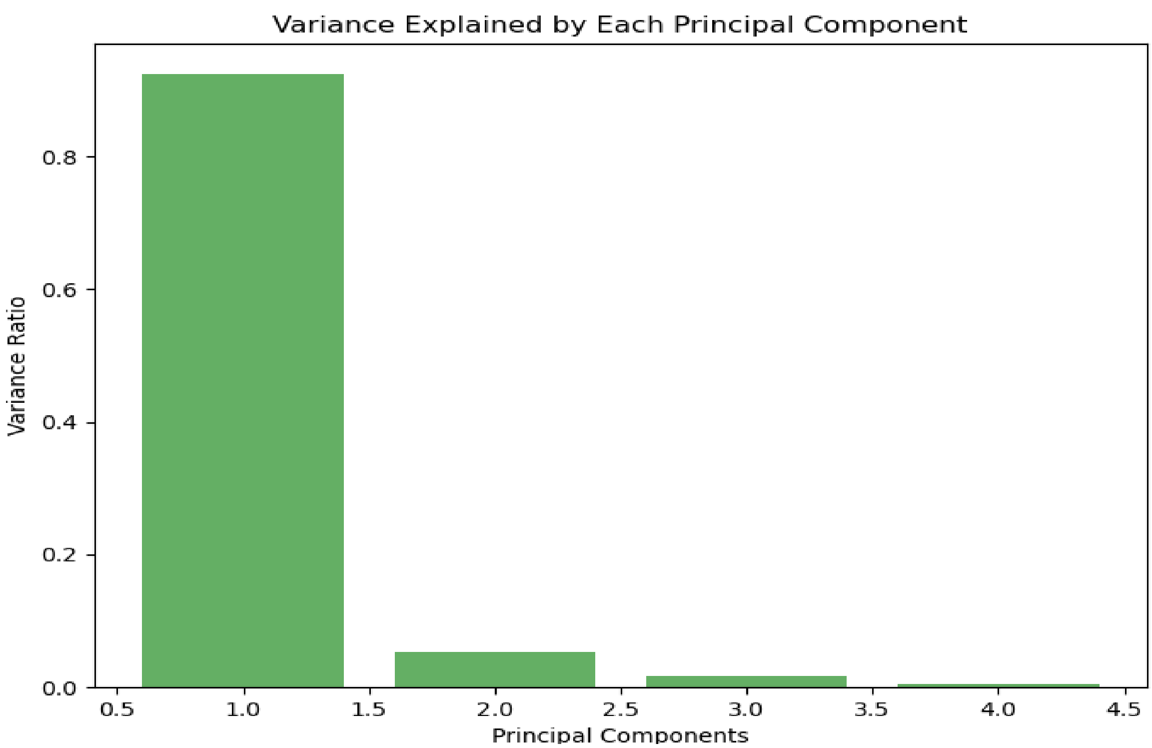
Scatter Plots with Various Components :-





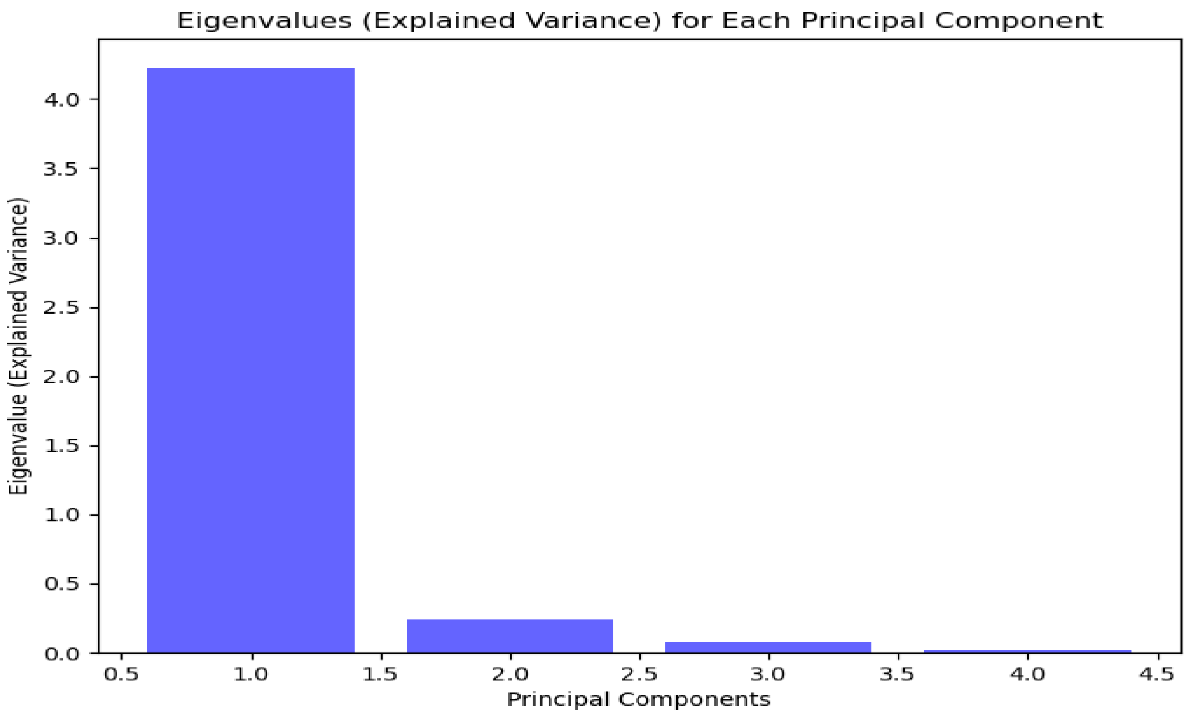


Variance Plot :-



EigenValues For Each Feature :-

```
Eigenvalues (Explained Variance) for each component:
[4.22484077 0.24224357 0.07852391 0.02368303]
```



Accuracy Graph :-