Assignment 1: Dimensionality Reduction of the Iris Dataset using Principal Component Analysis

Sanand Tripathi

Mtech AI, IIT Patna

Roll Number- 2511AI03

Date: September 19, 2025

Abstract

This report details the application of Principal Component Analysis (PCA), a dimensionality reduction technique, to the well-known Iris flower dataset. The dataset's original four features were reduced to two principal components to assess the separability of the three distinct Iris species. The data was first standardized to ensure feature equity. The results demonstrate that the first two principal components successfully capture **95.81%** of the original data's variance. The resulting 2D visualization clearly shows that the three species form distinct clusters, validating the effectiveness of PCA for pattern recognition and data visualization in this context.

1. Introduction

In the fields of machine learning and data science, datasets often contain a large number of features or dimensions. While rich in information, high-dimensional data can be difficult to process and visualize, a problem often referred to as the "curse of dimensionality." Dimensionality reduction techniques are employed to transform data from a high-dimensional space into a lower-dimensional space while retaining as much meaningful information as possible.

Principal Component Analysis (PCA) is one of the most widely used unsupervised techniques for this purpose. It identifies the directions, or principal components, along which the variation in the data is maximal.

This project aims to apply PCA to the Iris flower dataset, a classic benchmark dataset in pattern recognition. The objective is to reduce its four dimensions (sepal length, sepal width, petal length, petal width) to two dimensions and to visually determine if the three flower species (Setosa, Versicolor, Virginica) can be distinguished in this reduced feature space.

2. Methodology

The analysis was conducted using the Python programming language with the scikit-learn, pandas, and matplotlib libraries. The process involved the following steps:

2.1. Data Loading and Preparation

The Iris dataset was loaded directly from the scikit-learn library. It consists of 150 samples, with 50 samples for each of the three species. The four numerical features were separated from the categorical target variable (species).

2.2. Feature Scaling

Before applying PCA, feature scaling is a critical step. PCA is sensitive to the variance of the features, and if they are on different scales, features with larger variances can unduly influence the outcome. To prevent this, the **StandardScaler** from scikit-learn was used to standardize the features. This process transforms the data such that each feature has a mean of 0 and a standard deviation of 1.

2.3. Principal Component Analysis

The PCA model from scikit-learn was configured to reduce the data to **two principal components** (n_components=2). The standardized four-dimensional data was then fit to the model, which computed the principal components. These components are new, orthogonal axes that are linear combinations of the original features, ordered by the amount of variance they explain.

3. Results

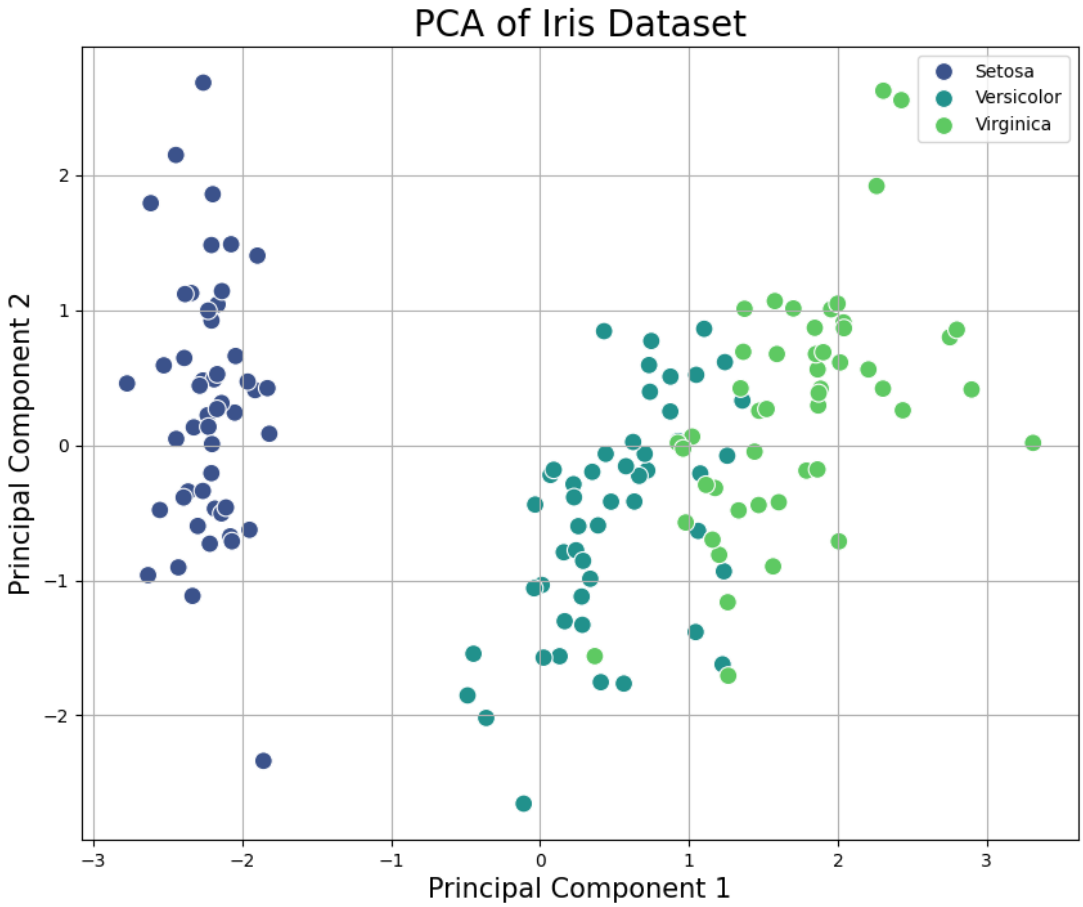
The application of PCA yielded two new dimensions (Principal Component 1 and Principal Component 2) that summarize the original four features.

The explanatory power of these components was quantified by their explained variance ratio:

- **Principal Component 1** accounts for **72.96%** of the total variance.
- **Principal Component 2** accounts for **22.85%** of the total variance.

Together, the two components capture a cumulative total of **95.81%** of the variance present in the original dataset, indicating a highly effective and informative reduction in dimensionality.

The 150 samples were plotted on a 2D scatter plot using their new principal component



values, with each point colored according to its species.

4. Analysis and Discussion

The results strongly support the effectiveness of PCA for this dataset. The high cumulative explained variance of 95.81% confirms that the two principal components are excellent

summaries of the original data, and very little information was lost in the dimensionality reduction process.

The visualization in Figure 1 provides a clear graphical confirmation of this. The three species of Iris form distinct clusters in the 2D space:

- The **Setosa** species (typically shown in one color) forms a tight, well-separated cluster, indicating it is very different from the other two.
- The **Versicolor** and **Virginica** species are also largely separated from each other, though they exhibit some minor overlap.

This visual separation implies that even after reducing the dataset's complexity by 50% (from four dimensions to two), the fundamental structure that distinguishes the species is not only preserved but enhanced.

5. Conclusion

This project successfully demonstrated the application of Principal Component Analysis for dimensionality reduction and visualization. By transforming the four-dimensional Iris dataset into a two-dimensional space, we were able to create a meaningful visualization that clearly distinguishes the three flower species. The analysis confirmed that PCA is a powerful tool for exploratory data analysis, capable of simplifying complex datasets while retaining their most critical structural information.

Appendix: Python Code

```
#Sanand Tripathi
#Mtech AI
#2511AI03
#Assignment 1

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris

def run_iris_pca():
    """
```

```
    This function performs PCA on the Iris dataset and saves a
visualization.
    """
    #load the dataset
    iris = load_iris()
    # The data is in iris.data, and the target (species) is in
iris.target.
    # The feature names are in iris.feature_names.
    df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
    df['target'] = iris.target

    # Map target numbers to actual species names for better
plotting
    target_names = {0: 'Setosa', 1: 'Versicolor', 2: 'Virginica'}
    df['species'] = df['target'].map(target_names)

    print("Dataset loaded successfully. First 5 rows:")
    print(df.head())
    print("\n")

    # Prepare the Data
    # Separate features (X) from the target (y)
    features = ['sepal length (cm)', 'sepal width (cm)', 'petal
length (cm)', 'petal width (cm)']
    X = df[features].values
    y = df['species'].values

    # Scale the features. This is CRITICAL for PCA.
    # It standardizes the data so that features with larger values
don't dominate the algorithm.
    X_scaled = StandardScaler().fit_transform(X)

    # Perform PCA
    # We are reducing the 4 features down to 2 principal
components.
    pca = PCA(n_components=2)
    principal_components = pca.fit_transform(X_scaled)

    # Create a new DataFrame with the principal components
    pca_df = pd.DataFrame(data=principal_components, columns=['PC
1', 'PC 2'])

    # Add the species column back for visualization
    pca_df['species'] = y

    print("PCA completed. First 5 rows of the new PCA DataFrame:")
    print(pca_df.head())
    print("\n")

    # Analyze the Results
    # 'explained_variance_ratio_' shows how much information
(variance)
```

```
    # is captured by each principal component.
    explained_variance = pca.explained_variance_ratio_
    print(f"Explained variance by PC1:
{explained_variance[0]:.2%}")
    print(f"Explained variance by PC2:
{explained_variance[1]:.2%}")
    print(f"Total variance captured by both components:
{explained_variance.sum():.2%}")
    print("\n")


    # Visualize the PCA results
    # This plot is the key result .
    plt.figure(figsize=(10, 8))
    sns.scatterplot(x='PC 1', y='PC 2', hue='species',
data=pca_df, palette='viridis', s=100)

    plt.title('PCA of Iris Dataset', fontsize=20)
    plt.xlabel('Principal Component 1', fontsize=15)
    plt.ylabel('Principal Component 2', fontsize=15)
    plt.legend()
    plt.grid()

    # Save the plot to a file
    plt.savefig('iris_pca_plot.png')
    print("Plot saved as 'iris_pca_plot.png'. You can now see the
file.")

    # Show the plot
    plt.show()


# main function
if __name__ == '__main__':
    run_iris_pca()
```