Principal Component Analysis (PCA) on Iris Dataset

Introduction

This notebook demonstrates how to perform Principal Component Analysis (PCA) on the Iris dataset.

This example shows a well known decomposition technique known as Principal Component Analysis (PCA) on the Iris dataset.

This dataset is made of 4 features: sepal length, sepal width, petal length, petal width. We use PCA to project this 4 feature space into a 3-dimensional space.

Loading the Iris dataset

The Iris dataset is directly available as part of scikit-learn. It can be loaded using the :func:`~sklearn.datasets.load_iris` function. With the default parameters, a :class:`~sklearn.utils.Bunch` object is returned, containing the data, the target values, the feature names, and the target names.

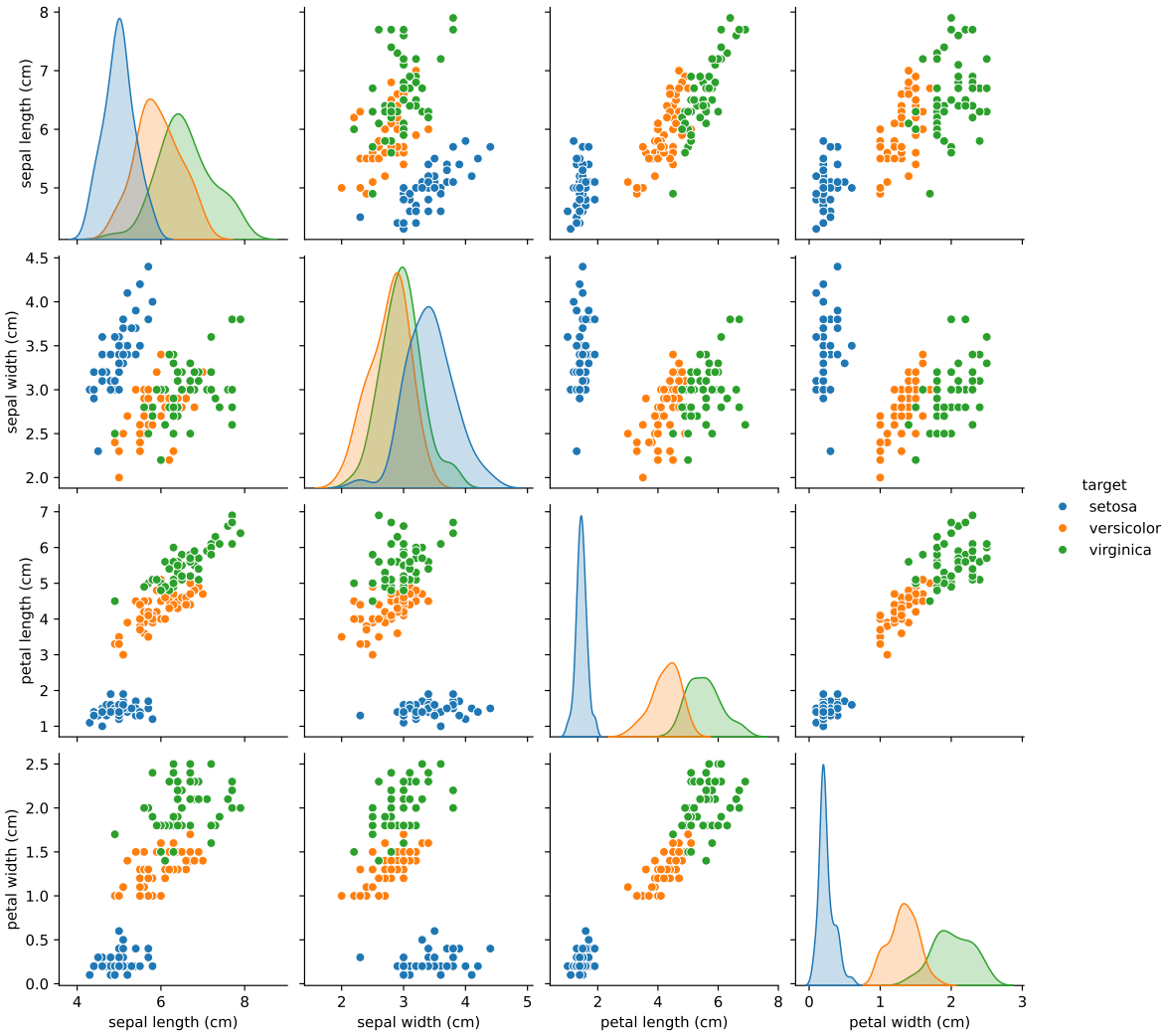
```
from sklearn.datasets import load_iris

iris = load_iris(as_frame=True)
print(iris.keys())
```

```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename',
```

Plot of pairs of features of the Iris dataset

Let's first plot the pairs of features of the Iris dataset.

```
import seaborn as sns

# Rename classes using the iris target names
iris.frame["target"] = iris.target_names[iris.target]
_ = sns.pairplot(iris.frame, hue="target")
```



Each data point on each scatter plot refers to one of the 150 iris flowers in the dataset, with the color indicating their respective type (Setosa, Versicolor, and Virginica).

You can already see a pattern regarding the Setosa type, which is easily identifiable based on its short and wide sepal. Only considering these two dimensions, sepal width and length, there's still overlap between the Versicolor and Virginica types.

The diagonal of the plot shows the distribution of each feature. We observe that the petal width and the petal length are the most discriminant features for the three types.

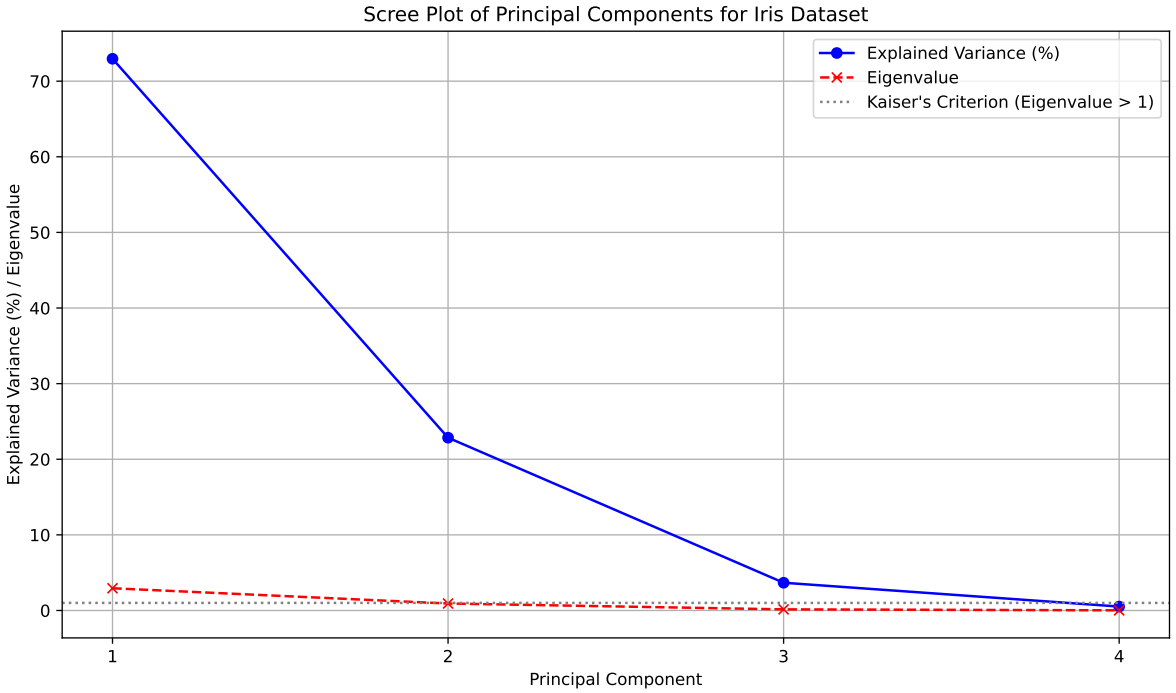
Explained Variance and Scree Plot

In PCA, it is important to understand how much variance each principal component captures. This is often visualized using a Scree Plot, which shows the explained variance for each principal component. We will also compute the eigenvalues of the covariance matrix to understand the significance of each principal component.

```
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np # For cumulative sum

# Load the Iris dataset
iris = load_iris(as_frame=True)
X = iris.data # Features DataFrame
feature_names = iris.feature_names # Original feature names for labeling

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
# Convert back to DataFrame for easier handling and to retain feature names
X_scaled_df = pd.DataFrame(X_scaled, columns=feature_names)
```

```
# Perform PCA on the standardized data, keeping all components to analyze variance
pca_full = PCA()
pca_full.fit(X_scaled)

# Get explained variance ratio and eigenvalues
explained_variance_ratio = pca_full.explained_variance_ratio_
eigenvalues = pca_full.explained_variance_
```

```
# Create a DataFrame for explained variance for a table
pc_labels = [f'PC{i+1}' for i in range(len(explained_variance_ratio))]
explained_variance_df = pd.DataFrame({
    'Principal Component': pc_labels,
    'Explained Variance Ratio (%)': explained_variance_ratio * 100,
    'Cumulative Explained Variance (%)': np.cumsum(explained_variance_ratio) * 100,
    'Eigenvalue': eigenvalues
})

# Plotting the Scree Plot
plt.figure(figsize=(10, 6))
plt.plot(range(1, len(explained_variance_ratio) + 1), explained_variance_ratio * 100,
         marker='o', linestyle='-', color='b', label='Explained Variance (%)')
plt.plot(range(1, len(eigenvalues) + 1), eigenvalues,
         marker='x', linestyle='--', color='r', label='Eigenvalue') # Added eigenvalue plot
plt.axhline(y=1, color='gray', linestyle=':', label="Kaiser's Criterion (Eigenvalue > 1)") #
plt.title('Scree Plot of Principal Components for Iris Dataset')
plt.xlabel('Principal Component')
plt.ylabel('Explained Variance (%) / Eigenvalue')
plt.xticks(range(1, len(explained_variance_ratio) + 1))
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

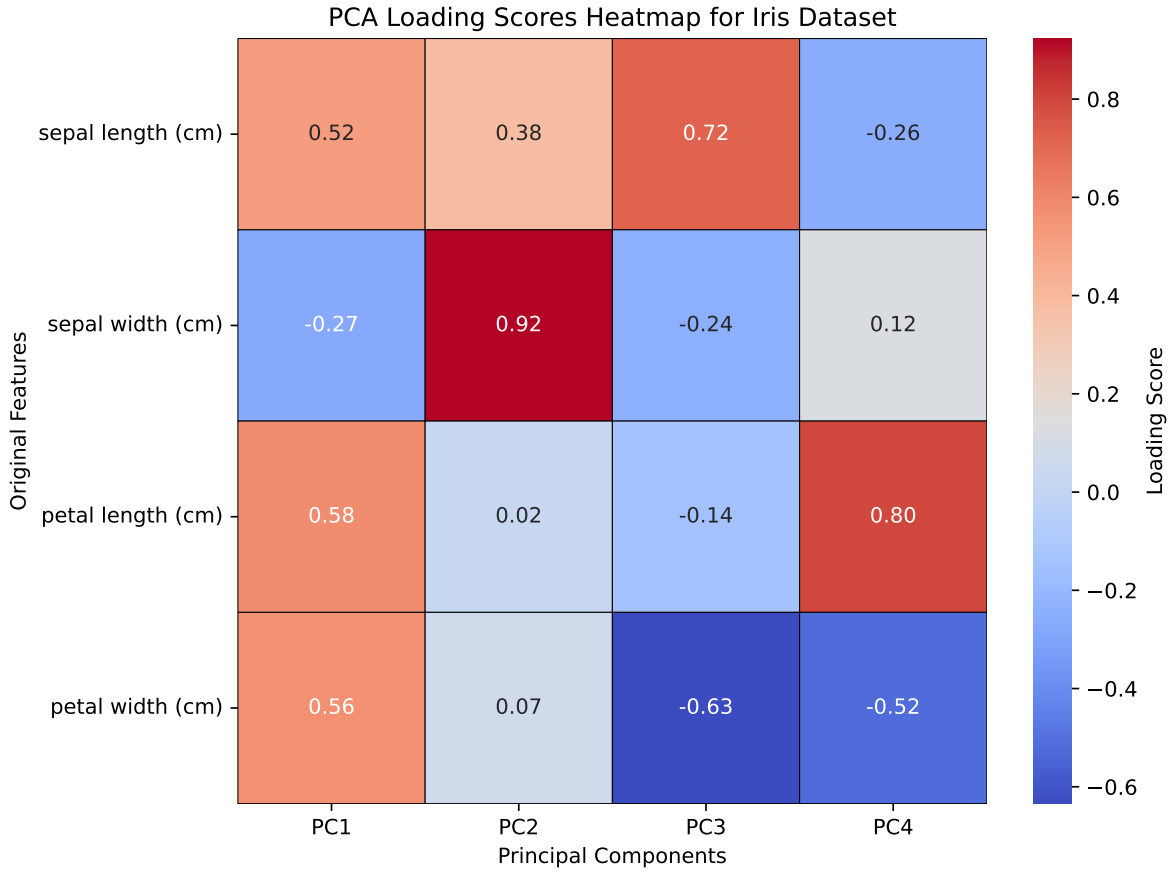


Loading Scores Heatmap

The loading scores indicate how much each original feature contributes to each principal component. We will visualize these loading scores using a heatmap.

```
# Use the pca_full object from the scree plot section, which was fitted on X_scaled
# pca_full.components_ has shape (n_components, n_features)
# Transpose it to have (n_features, n_components) for the heatmap
loadings = pca_full.components_.T

# Create a DataFrame for better visualization with feature names and PC labels
pc_cols = [f'PC{i+1}' for i in range(loadings.shape[0])] # e.g., ['PC1', 'PC2', 'PC3', 'PC4']
loadings_df = pd.DataFrame(loadings, index=feature_names, columns=pc_cols)

# Plotting the Loading Scores Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(loadings_df, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5, linecolor='bl
            cbar_kws={'label': 'Loading Score'})
plt.title('PCA Loading Scores Heatmap for Iris Dataset')
plt.xlabel('Principal Components')
plt.ylabel('Original Features')
```

```
plt.tight_layout()
plt.show()
```

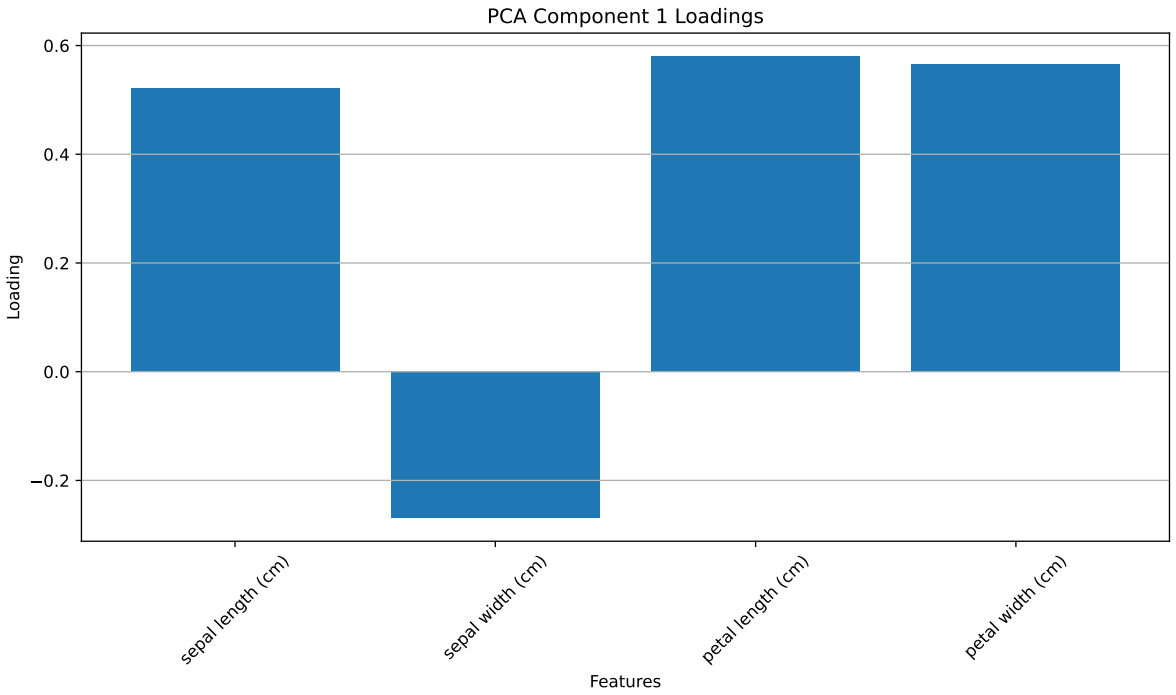


Visualizing the First Principal Component Loadings

The first principal component often captures the most variance in the data. We can visualize the loadings of the first principal component to see how each feature contributes to it. This can help us understand which features are most influential in the first principal component. These values are identical to the first row of the `loadings_df` DataFrame created above.

```
plt.figure(figsize=(10, 6))
plt.bar(range(1, len(pca_full.components_[0]) + 1), pca_full.components_[0], tick_label=featu
plt.title('PCA Component 1 Loadings')
plt.xlabel('Features')
plt.ylabel('Loading')
```

```
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

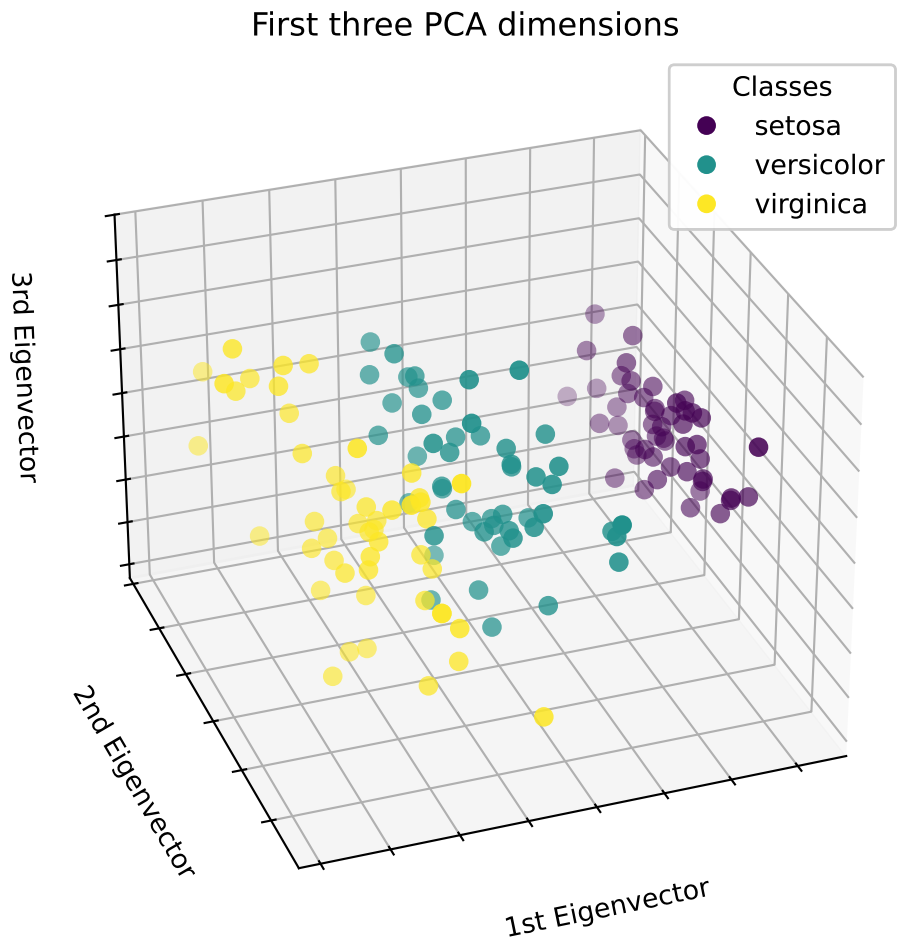


Plot a PCA representation

Let's apply a Principal Component Analysis (PCA) to the iris dataset and then plot the irises across the first three PCA dimensions. This will allow us to better differentiate among the three types!

```
import matplotlib.pyplot as plt

# unused but required import for doing 3d projections with matplotlib < 3.2
import mpl_toolkits.mplot3d  # noqa: F401

from sklearn.decomposition import PCA

fig = plt.figure(1, figsize=(8, 6))
ax = fig.add_subplot(111, projection="3d", elev=-150, azim=110)
```

```
X_reduced = PCA(n_components=3).fit_transform(iris.data)
scatter = ax.scatter(
    X_reduced[:, 0],
    X_reduced[:, 1],
    X_reduced[:, 2],
    c=iris.target,
    s=40,
)

ax.set(
    title="First three PCA dimensions",
    xlabel="1st Eigenvector",
    ylabel="2nd Eigenvector",
    zlabel="3rd Eigenvector",
)
ax.xaxis.set_ticklabels([])
ax.yaxis.set_ticklabels([])
ax.zaxis.set_ticklabels([])

# Add a legend
legend1 = ax.legend(
    scatter.legend_elements()[0],
    iris.target_names.tolist(),
    loc="upper right",
    title="Classes",
)
ax.add_artist(legend1)

plt.show()
```



PCA will create 3 new features that are a linear combination of the 4 original features. In addition, this transformation maximizes the variance. With this transformation, we see that we can identify each species using only the first feature (i.e., first eigenvector).