Learning Scree Plots: A Step-by-Step Guide to PCA Visualization in Python

Authored by

Mohammed looti

November 2, 2025

Principal Component Analysis (PCA) is a fundamental technique in statistical analysis and dimensionality reduction. Its primary goal is to transform a large set of variables into a smaller set of variables, called principal components, while retaining the vast majority of information present in the original dataset. These principal components are carefully constructed linear combinations of the predictor variables, chosen specifically to explain the maximum amount of variance possible. By effectively reducing the complexity of high-dimensional data, PCA makes subsequent visualization, modeling, and analysis significantly more manageable and efficient.

When applying PCA, a crucial task is determining the optimal number of components required to adequately represent the underlying data structure. We are intensely interested in quantifying exactly what percentage of the total variation within the dataset can be attributed to and explained by each newly generated principal component. If the initial few components successfully capture a substantial majority of the variance, we can safely discard the remaining components without incurring a major loss of analytical fidelity or predictive power.

The most straightforward and universally accepted method for visualizing this explained variance ratio is through the construction of a **scree plot**. This essential graphical tool plots the eigenvalues--or, more commonly, the percentage of variance explained--against the corresponding index of the principal component. The resulting graph offers a crystal-clear visual guide, enabling data scientists to determine the optimal number of components to retain by identifying the point of inflection, often referred to as the "elbow." This comprehensive tutorial provides a detailed, step-by-step guide on how to successfully generate and interpret a scree plot using the powerful Python programming environment.

Step 1: Data Acquisition and Preprocessing

For this practical example, we will utilize the well-known USArrests dataset. This dataset contains crucial information regarding the number of arrests per 100,000 residents across each U.S. state in 1973, categorized by various crime types. Before performing PCA, it is fundamentally important to standardize the data. Standardization ensures that all features contribute equally to the distance calculations; without it, variables with larger numerical scales would disproportionately influence the principal components.

The following code block demonstrates the necessary steps to import this dataset using the Pandas library, select the relevant features for analysis, and apply the StandardScaler from the Scikit-learn preprocessing module. This preparation step is vital for ensuring the validity and reliability of the subsequent PCA results.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
```

```
#define URL where dataset is located
url = "https://raw.githubusercontent.com/JWarmenhoven/ISLR-python/master/Notebooks/Data/USArrests.csv"

#read in data
data = pd.read_csv(url)

#define columns to use for PCA
df = data.iloc

#define scaler
scaler = StandardScaler()

#create copy of DataFrame
scaled_df=df.copy()

#created scaled version of DataFrame
scaled_df=pd.DataFrame(scaler.fit_transform(scaled_df), columns=scaled_df.columns)
```

Step 2: Implementing Principal Components Analysis

Once the data has been properly standardized, the next logical step is to execute the Principal Components Analysis itself. We leverage the highly optimized `PCA()` function, which is readily available within the decomposition module of the powerful Scikit-learn (sklearn) library. Since our initial dataset contains four distinct features (variables), we specify that we want to extract four principal components by setting the `n_components=4` parameter. Although we aim for dimensionality reduction later, we initially calculate all possible components to understand the full variance structure.

The code below defines the PCA model and then fits it to our previously scaled DataFrame (`scaled_df`). The fitting process calculates the eigenvectors and eigenvalues necessary to transform the original data coordinates into the new coordinate system defined by the principal components. These eigenvalues are fundamental, as they directly correspond to the amount of variance explained by each component.

```
from sklearn.decomposition import PCA

#define PCA model to use
pca = PCA(n_components=4)

#fit PCA model to data
```

```
pca_fit = pca.fit(scaled_df)
```

Step 3: Generating the Scree Plot Visualization

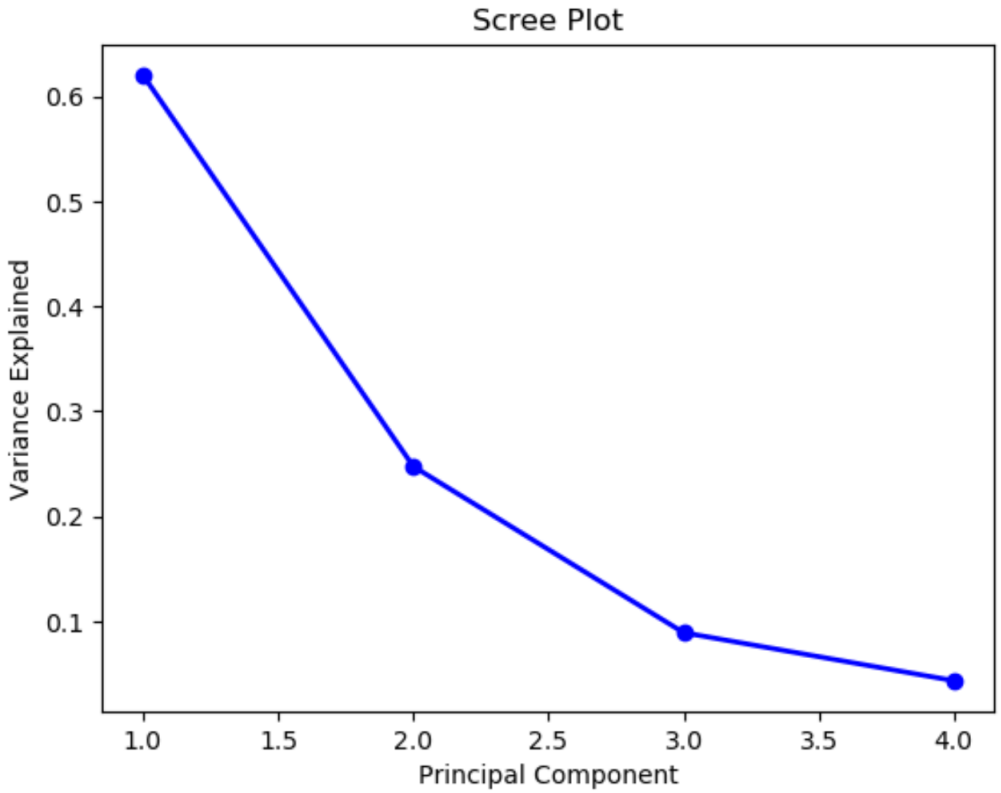
With the PCA model trained, we can now access the critical attribute `explained_variance_ratio_` from the fitted model. This array contains the percentage of total variance explained by each principal component, ordered from highest to lowest. To visually inspect this relationship and determine the optimal number of components, we must construct the scree plot using the renowned data visualization library, Matplotlib.

The plotting code below first generates an array representing the component numbers (1 through 4) for the x-axis. It then plots these component indices against the explained variance ratios (the y-axis values). We apply standard plotting functions to label the axes appropriately and assign a clear title, making the resulting visualization immediately interpretable for any analyst.

```
import matplotlib.pyplot as plt
import numpy as np

PC_values = np.arange(pca.n_components_) + 1
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```

Executing the above code generates the following graphical representation:



Step 4: Interpreting Explained Variance

The visualization provided by the scree plot is highly informative. The x-axis clearly delineates the principal component number (PC1, PC2, etc.), while the y-axis quantifies the percentage of the total variance captured by that specific individual component. A sharp drop-off in the curve indicates that subsequent components contribute significantly less to explaining the dataset's overall variation. This phenomenon is critical for determining the optimal subset of components to retain.

While the plot offers an intuitive understanding, it is often beneficial to examine the precise numerical contribution of each component. We can easily access these exact ratios by printing the `explained_variance_ratio_` attribute of the fitted PCA object. This output confirms the magnitude of the explanatory power held by the first few principal components compared to the latter ones.

print(pca.explained_variance_ratio_)

By translating these ratios into percentages, we gain a quantitative perspective on the information contained within the principal components:

The first principal component (PC1) explains a substantial **62.01%** of the total variation in the dataset. This indicates that PC1 is the most critical axis of variance.
The second principal component (PC2) explains an additional **24.74%** of the total variation.
The third principal component (PC3) explains **8.91%** of the total variation, showing a significant drop in explanatory power.
The fourth principal component (PC4) explains the smallest portion, accounting for only **4.34%** of the total variation.

It is important to note that, by design, the sum of these percentages equals 100% of the total variance originally present in the four features of the scaled dataset.

Step 5: Applying the Elbow Rule for Component Selection

The primary utility of the scree plot lies in its ability to facilitate component selection using the "elbow rule." The elbow rule suggests that the optimal number of components to retain is located immediately before the point where the curve begins to flatten out into a straight line. This flattening indicates that subsequent components are contributing only marginally to the explained variance, essentially representing noise rather than significant structural information.

In the generated plot above, we observe a dramatic decline in explained variance from the first principal component to the second, followed by a less steep but still notable drop to the third. The curve appears to significantly flatten out after the second component (PC2). Therefore, based on the elbow rule, a data scientist would typically select **two** principal components, as PC1 and PC2 together account for approximately 86.75% (62.01% + 24.74%) of the total variation, achieving significant PCA dimensionality reduction with minimal information loss.

Mastering the creation and interpretation of the scree plot is an essential skill for anyone working with Principal Component Analysis, providing a robust and justifiable method for selecting the correct number of components for subsequent modeling tasks.

You can find many more statistical and machine learning tutorials on this site.