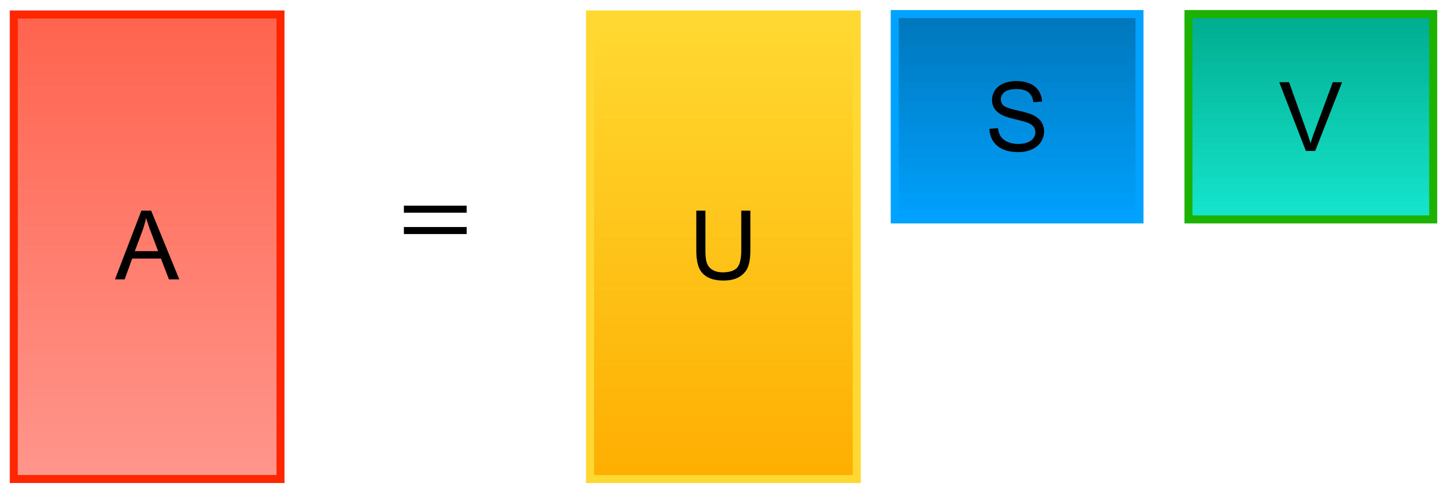
Quick Review of SVD

$$A = USV^T$$



$U : n \times d$ orthogonal matrix (left singular vectors)
$S : d \times d$ diagonal matrix (singular values)
$V : d \times d$ orthogonal matrix (right singular vectors)

Use Python build-in function to compute SVD.

Application: Orthogonal Procrustes Analysis

Problem:

Find the rotation $R*$ that minimizes distance between two $d \times k$ matrices A, B:

$$R* = \arg \min \|\mathrm{RA} - \mathrm{B}\|^2, \mathrm{s.t.} \mathrm{R^T R} = \mathrm{I}$$

Solution:

Let $U \Sigma V^T$ be the SVD of BA^T, then

$$R* = UV^T$$

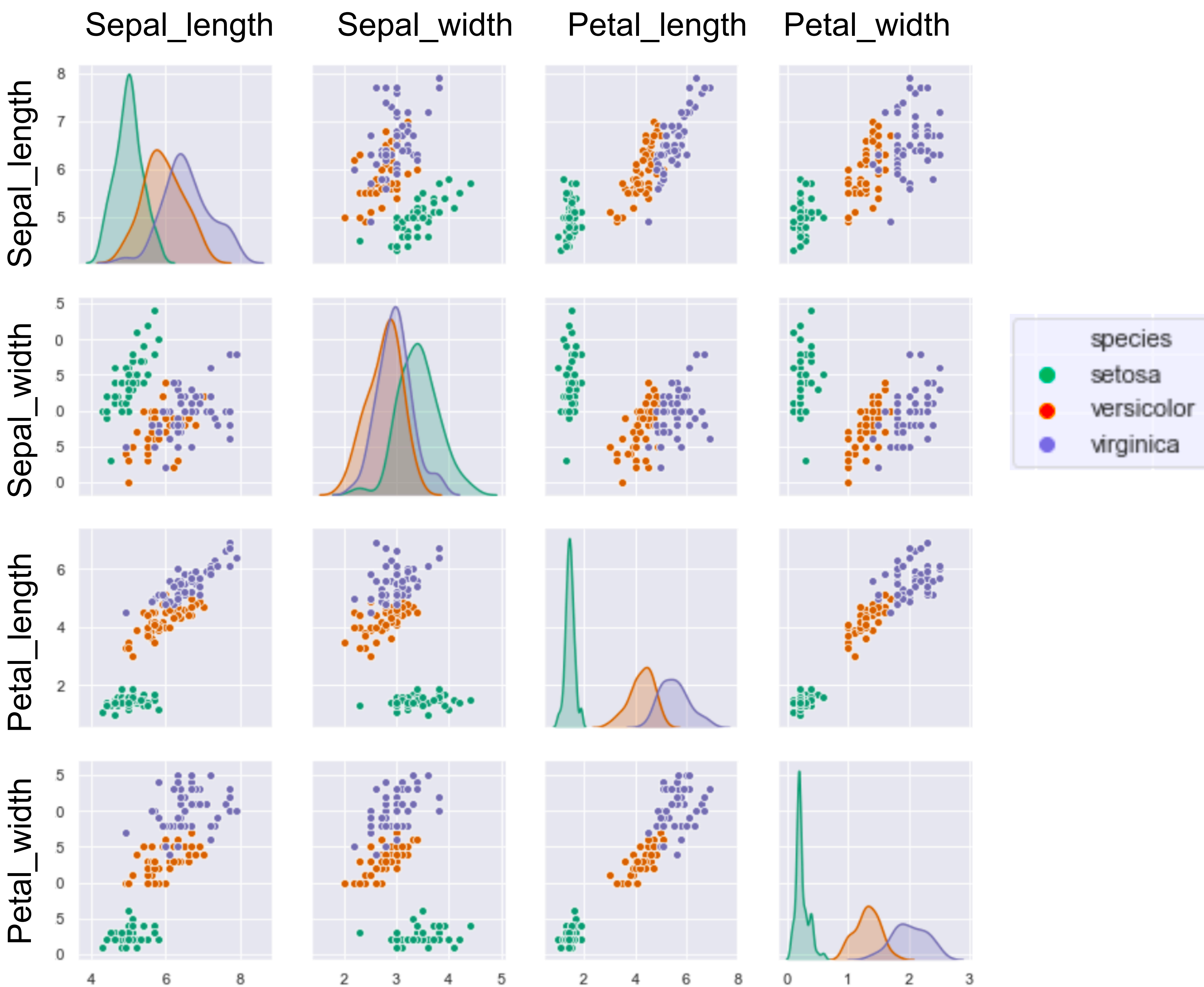
Proof is available at: https://en.wikipedia.org/wiki/Orthogonal_Procrustes_problem

Principal Component Analysis (PCA)

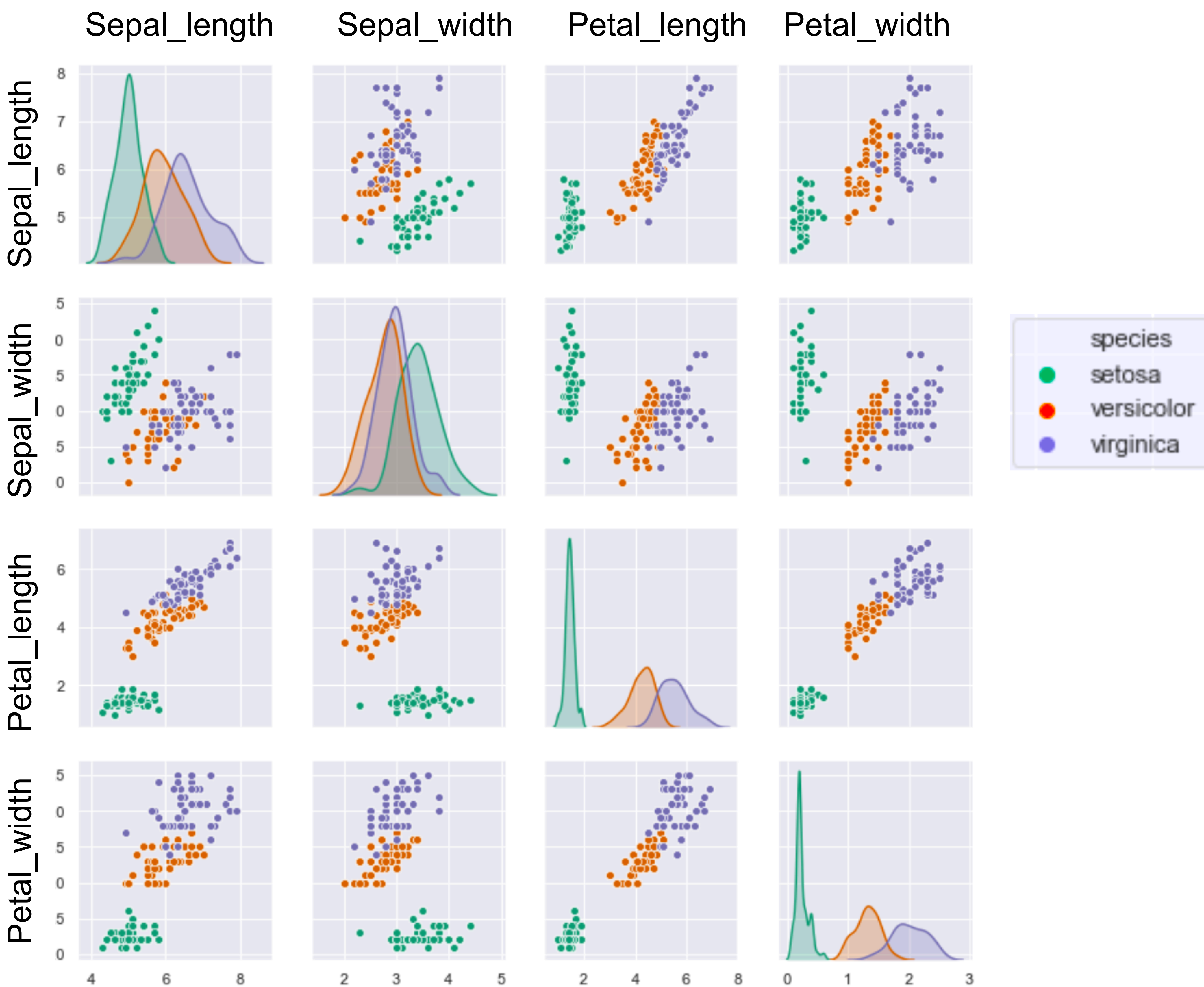
Foundations of Data Analysis

March 22, 2023

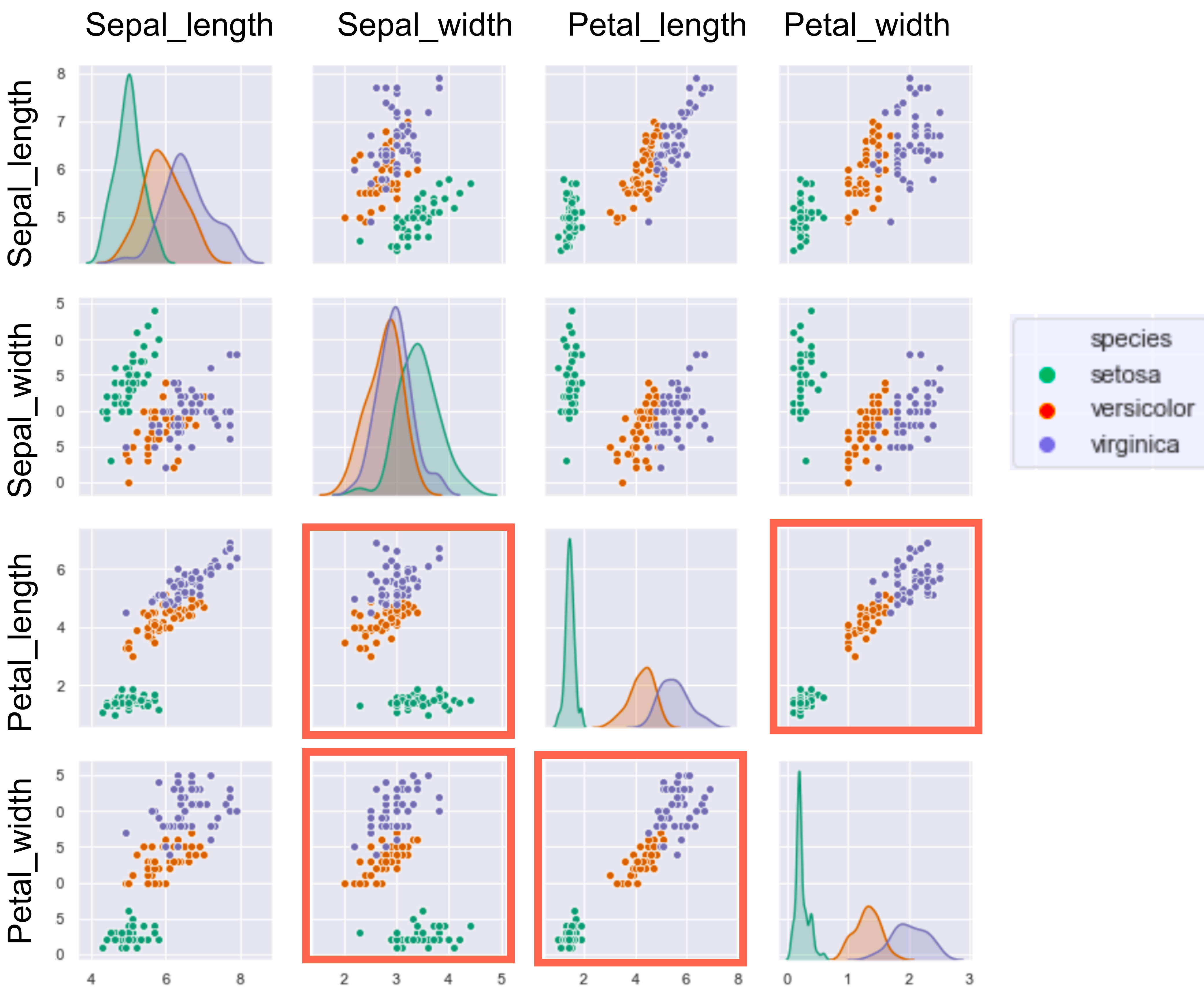
Example: Iris Data



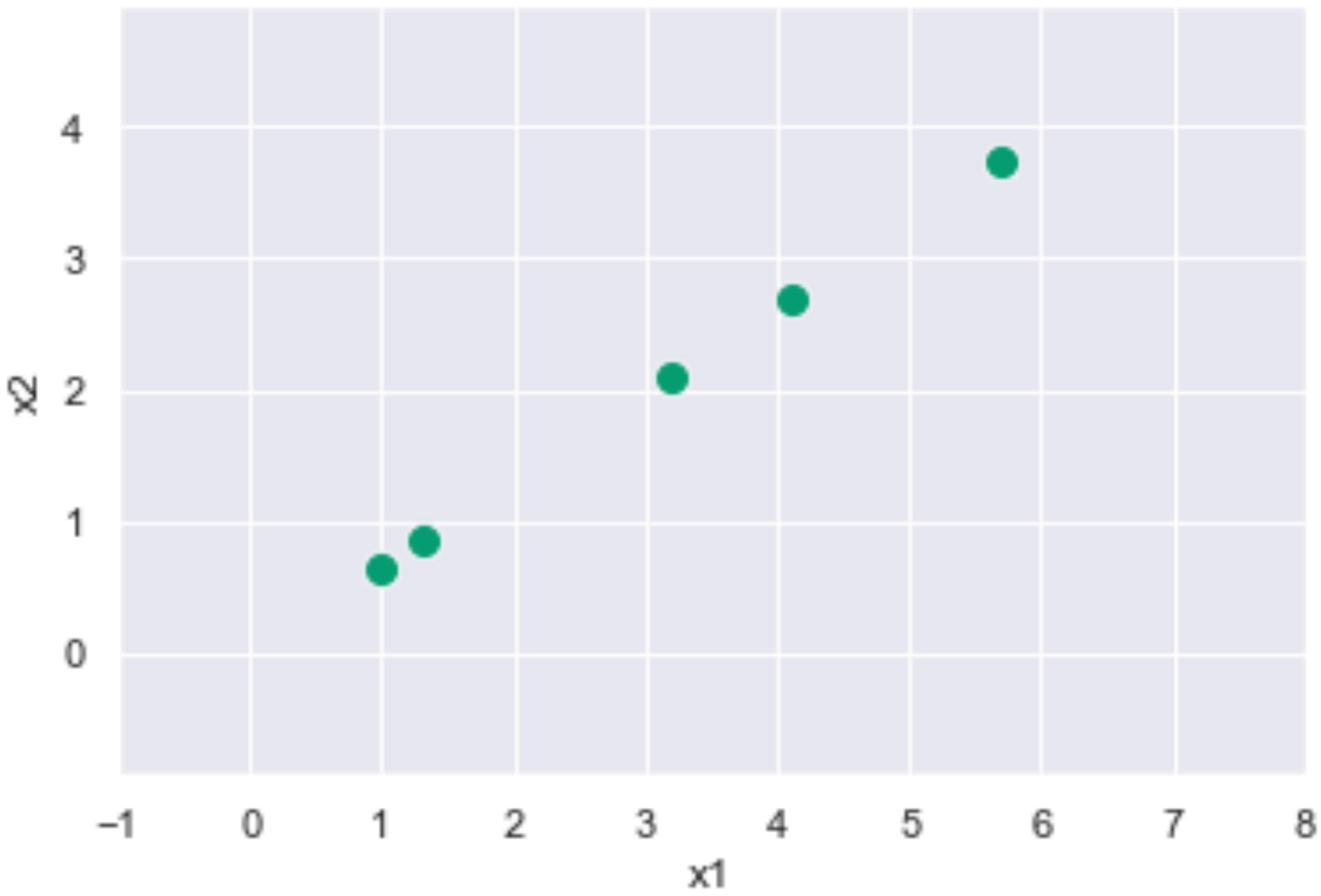
Do We Need All Dimensions of Features?



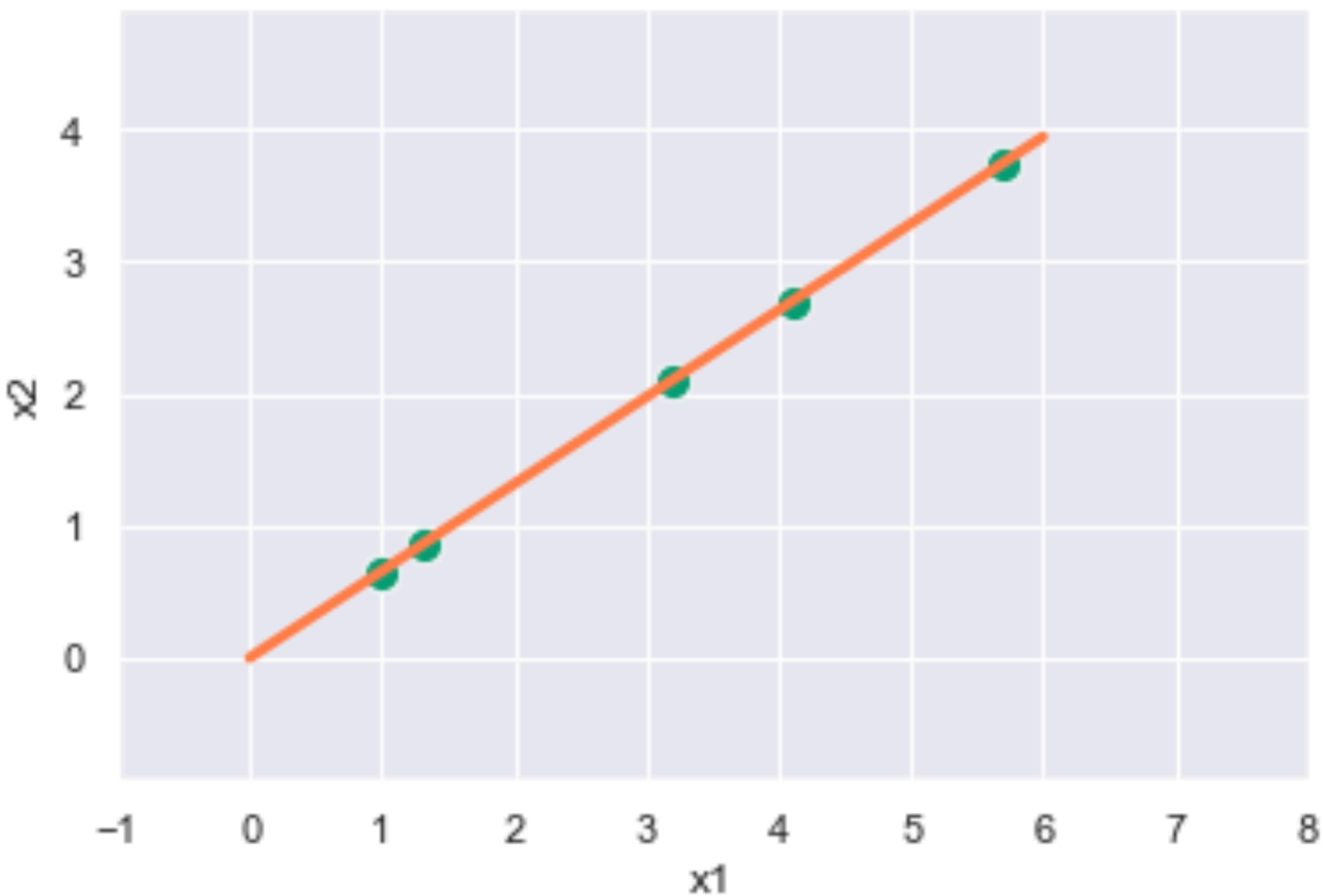
Do We Need All Dimensions of Features?



How Many Dimensions Are In Your Data?



How Many Dimensions Are In Your Data?



This data can be represented by 1-D if using the orange line (basis).

Covariance

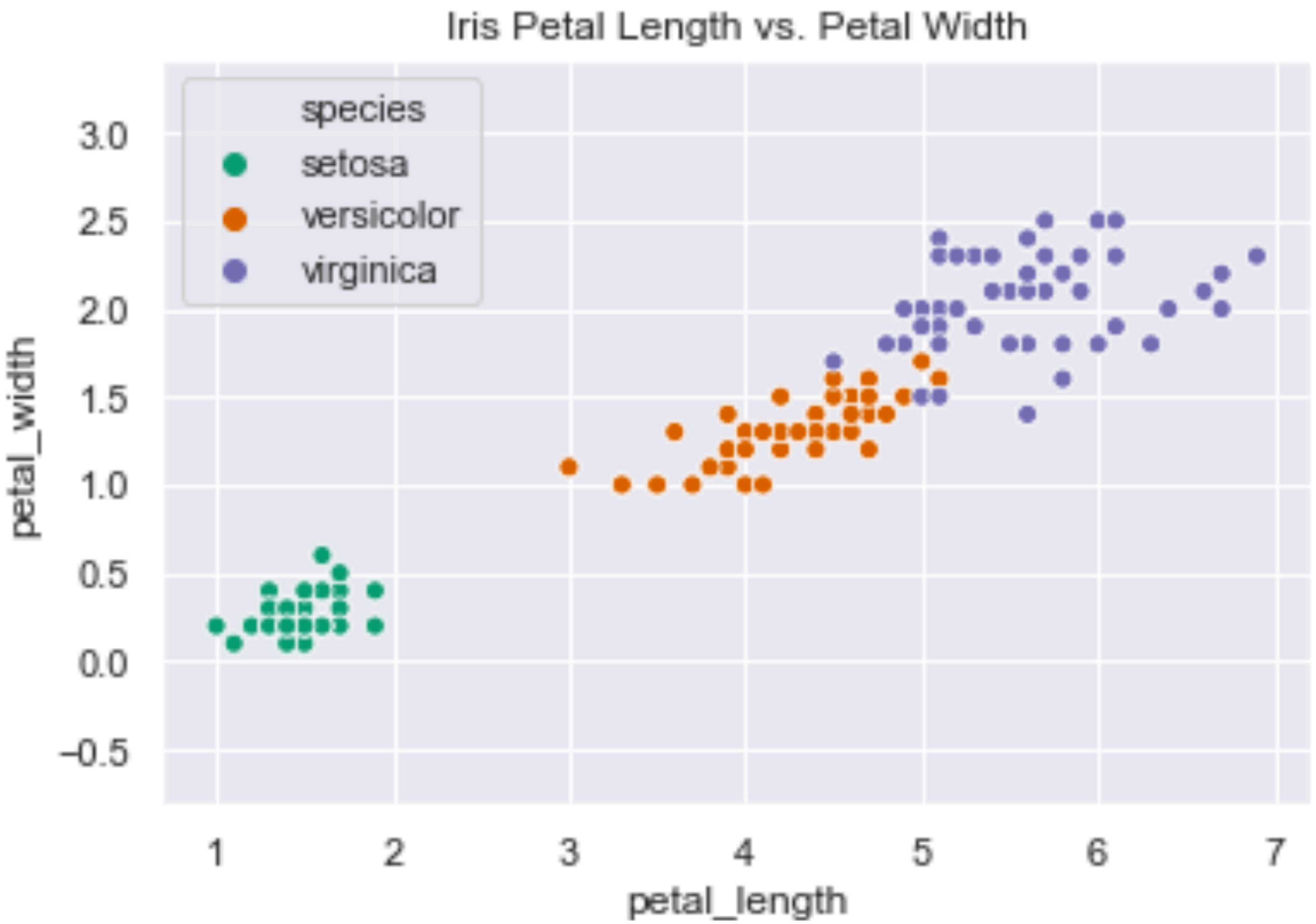
Covariance between two random samples: $x_i, y_i \in \mathbb{R}$

$$\text{cov}(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

Measures how x "covaries" with y

Symmetric: $\text{cov}(x, y) = \text{cov}(y, x)$

Example: Iris Data



$$Covariance = 1.2869720000000002$$

Centering a Data Matrix

Data matrix $X : n \times d$

n rows (data points)

d columns (dimensions, or features)

Mean of data (rows):

$$\mu = \frac{1}{n} \sum_{i=1}^{n} X_{i\bullet}$$

Centered data (subtract mean from each row):

$$\tilde{X}_{i\bullet} = X_{i\bullet} - \mu$$

Covariance Matrix

Sample covariance matrix:

$$\Sigma = \frac{1}{n-1}\tilde{X}^T\tilde{X}$$

$\Sigma_{i,j}$ is the covariance between the ith and jth dimension (feature)

$$\Sigma_{i,j} = \frac{1}{n-1}\sum_{k=1}^{n}(X_{ki}-\mu_i)(X_{kj}-\mu_j) = \mathrm{cov}(X_{\bullet i}, X_{\bullet j})$$

Properties

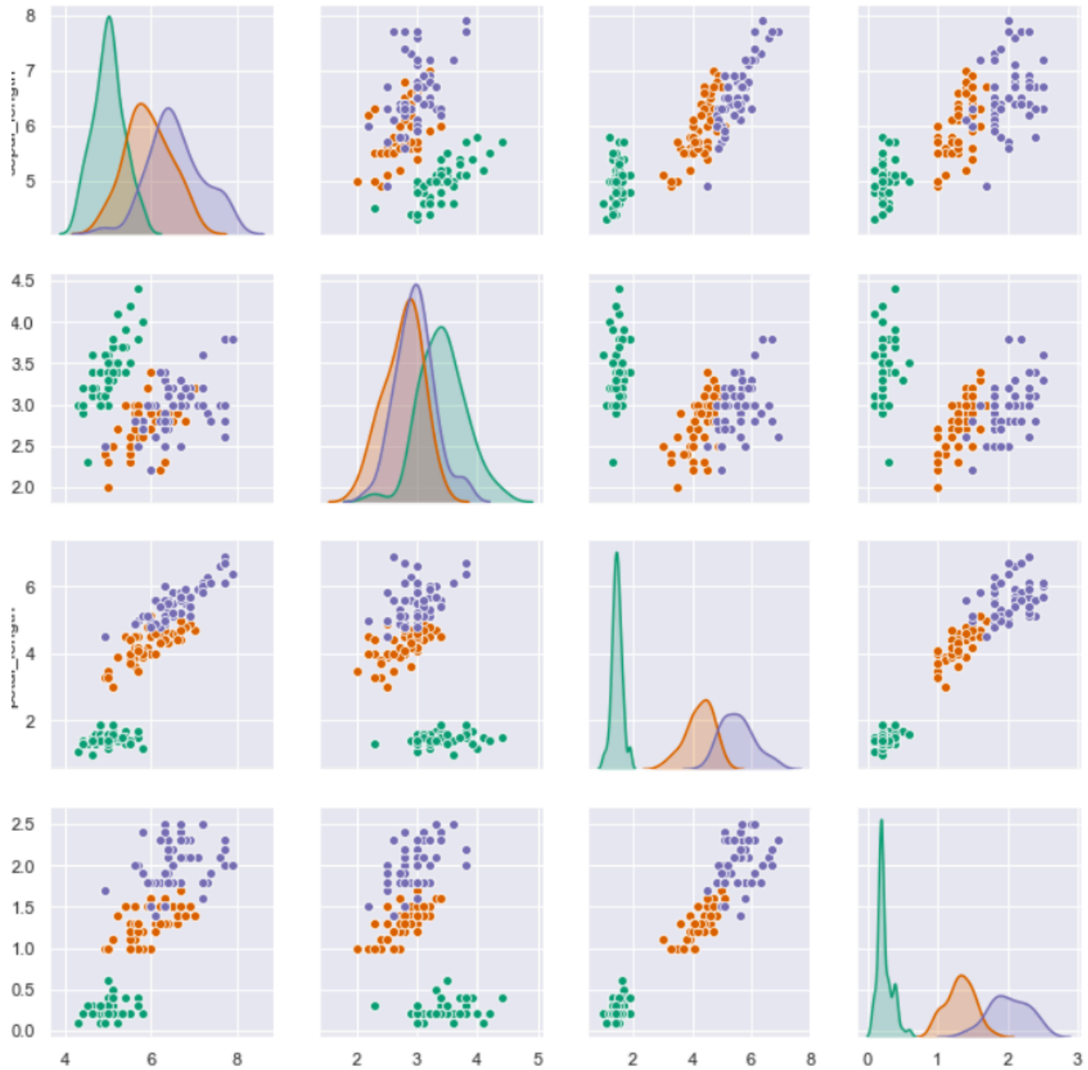
Covariance is **symmetric**: $\Sigma = \Sigma^T$

$$\Sigma_{i,j} = \text{cov}(X_{\bullet i}, X_{\bullet j}) = \text{cov}(X_{\bullet j}, X_{\bullet i}) = \Sigma_{j,i}$$

Covariance is **positive-semidefinite**:

$$v^T \Sigma v \geq 0$$

Example: Iris Data



Covariance matrix:

$$\Sigma = \begin{pmatrix} 0.685, & -0.04243, & 1.274, & 0.5163 \\ -0.04243, & 0.1900, & -0.3297, & -0.1216 \\ 1.274, & -0.3297, & 3.116, & 1.296 \\ 0.5163, & -0.1216, & 1.296, & 0.5810 \end{pmatrix}$$

Eigenvectors, Eigenvalues

Square matrix $A : d \times d$

Eigenvector $v \in \mathbb{R}^d$ and eigenvalue $\lambda \in \mathbb{R}$:

$$Av = \lambda v$$

Meaning: The transformation A is a scaling when applied to v.

Eigen Analysis of a Symmetric Matrix

Fact: If A is a $d \times d$ symmetric matrix, it has exactly d real eigenvalues $\lambda_k \in \mathbb{R}$ (possibly with repeats).

Each eigenvalue λ_k has a corresponding eigenvector $v_k \in \mathbb{R}^d$.

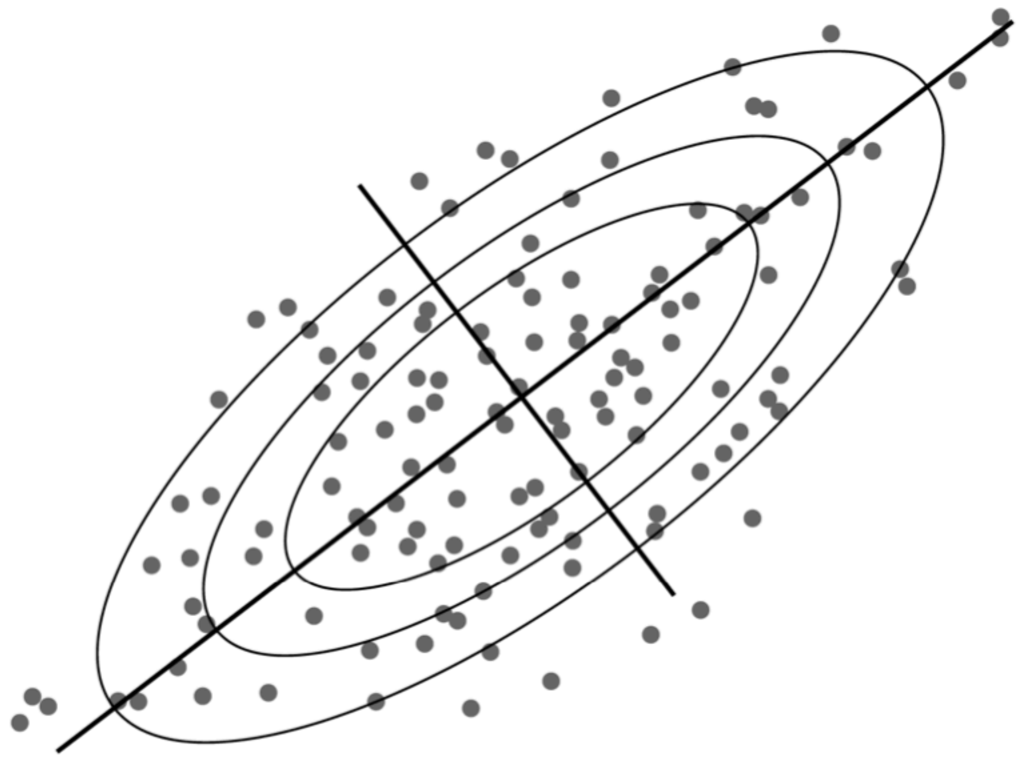
Eigen Analysis of a Symmetric Matrix

The SVD of a symmetric, positive-semidefinite matrix looks like this:

$$A = VSV^T$$

- The singular values are the eigenvalues: $S_k = \lambda_k$.
- The left and right singular vectors are the *same* and are the eigenvectors, v_k.

Principal Component Analysis



PCA is an eigen analysis of the covariance matrix:

$$\Sigma = V \Lambda V^T$$

- Eigenvectors: $v_k = V_{\bullet k}$ are **principal components**
- Eigenvalues: λ_k are the **variance** of the data in the v_k direction

PCA Algorithm Summary

Input: Data matrix $X : n \times d$

1. Compute centered data $\tilde{X}$
2. Compute covariance matrix:

$$\text{cov}(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

3. Eigenanalysis of covariance:

$$\Sigma = V \Lambda V^T$$

Hint: *numpy.linalg.eigh* computes an eigen analysis of a symmetric matrix!

Dimensionality Reduction

Goal: Find a k-dimensional subspace, v_k, that best fits our data

Least-squares fit:

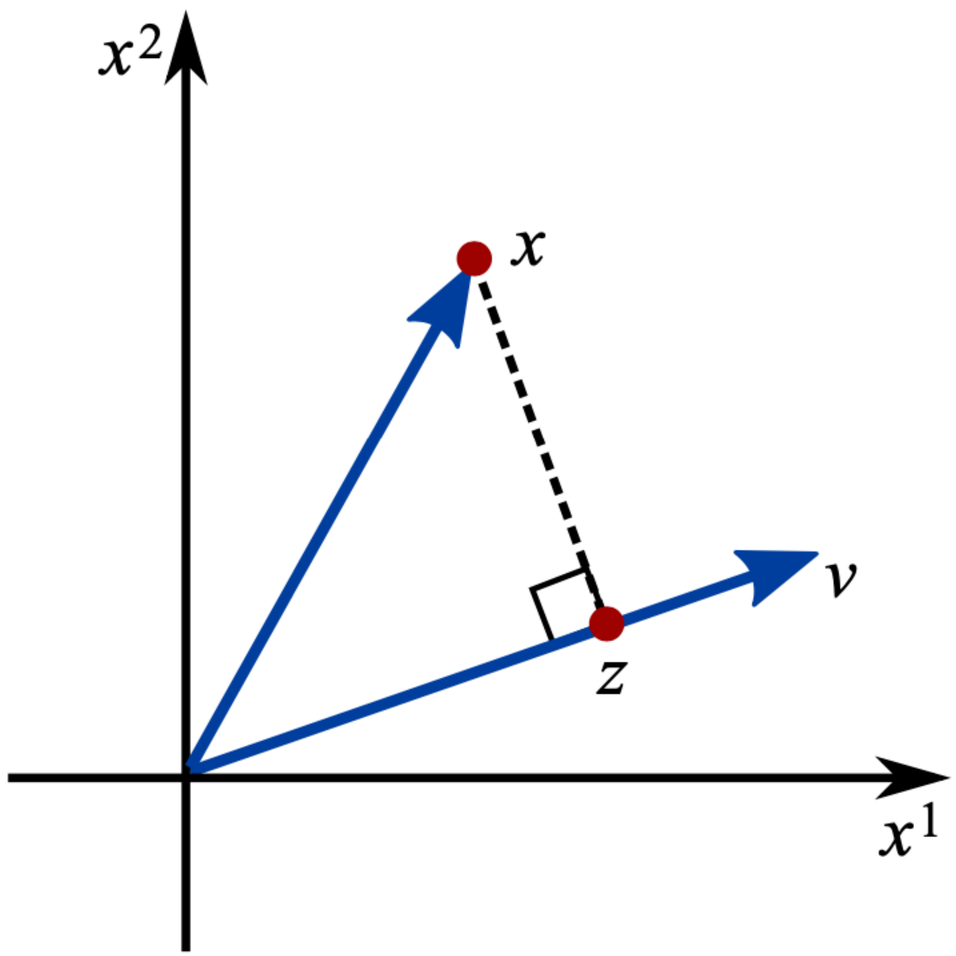
$$\arg\min_{V_k} \sum_{i=1}^{n} \text{distance}(V_k, x_i)^2$$

Solution: Use first k principal components:

$$V_k = \text{span}(v_1, v_2, \cdots, v_k)$$

Maximizing Variance of Projected Data

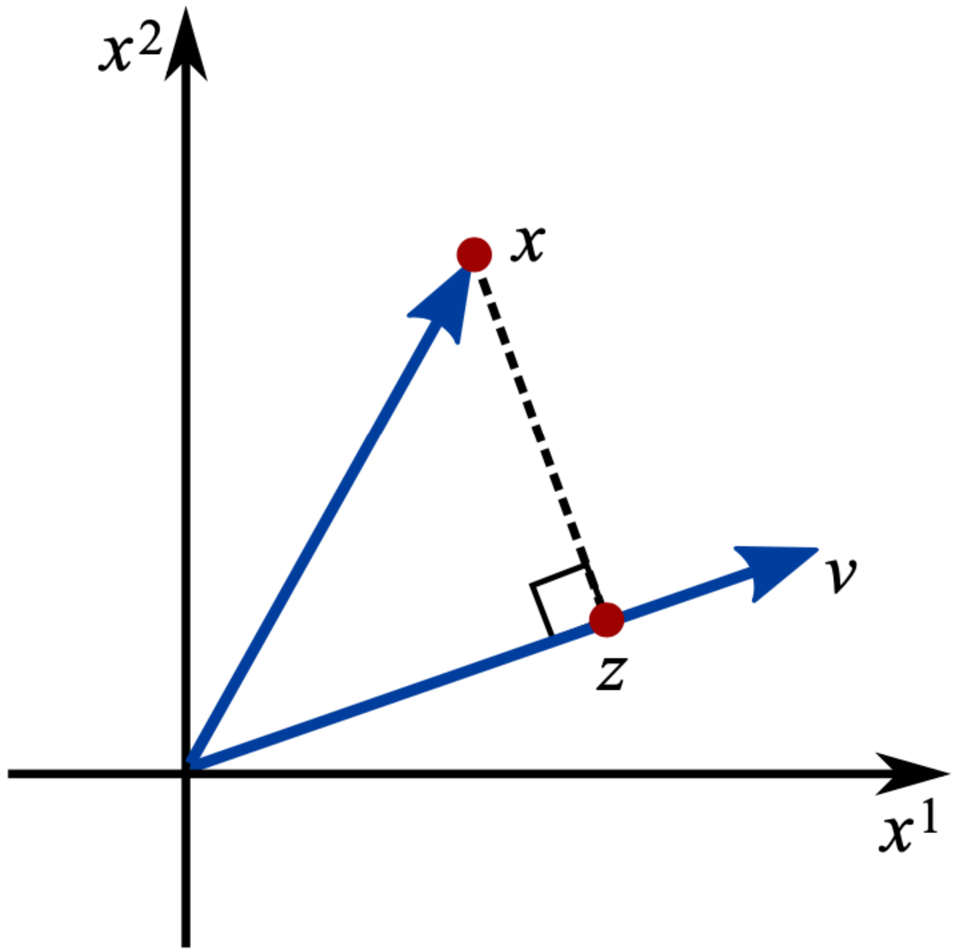
Fact: PCA finds dimensions that maximize variance



Given direction $v \in \mathbb{R}^d$, with $\|v\| = 1$,
project data point $x \in \mathbb{R}^d$ onto v:

$$z = < v, x >$$

Maximizing Variance of Projected Data

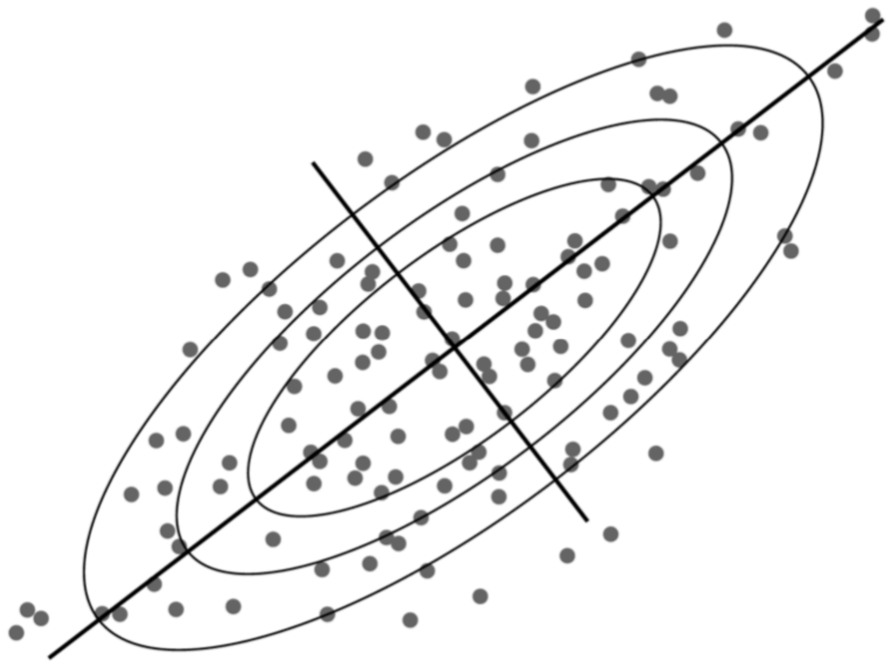


Given mean-centered data, x_i,

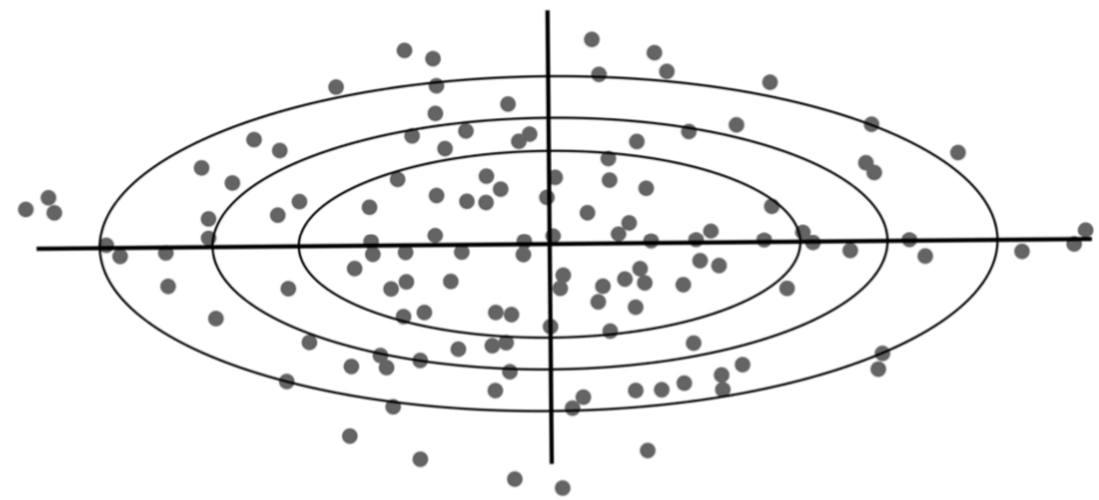
first principal component, v_1 maximizes variance:

$$v_1 = \arg\max_{\|v\|=1} \sum_{i=1}^{n} < v, x_i >^2$$

PC's as Rotation



X



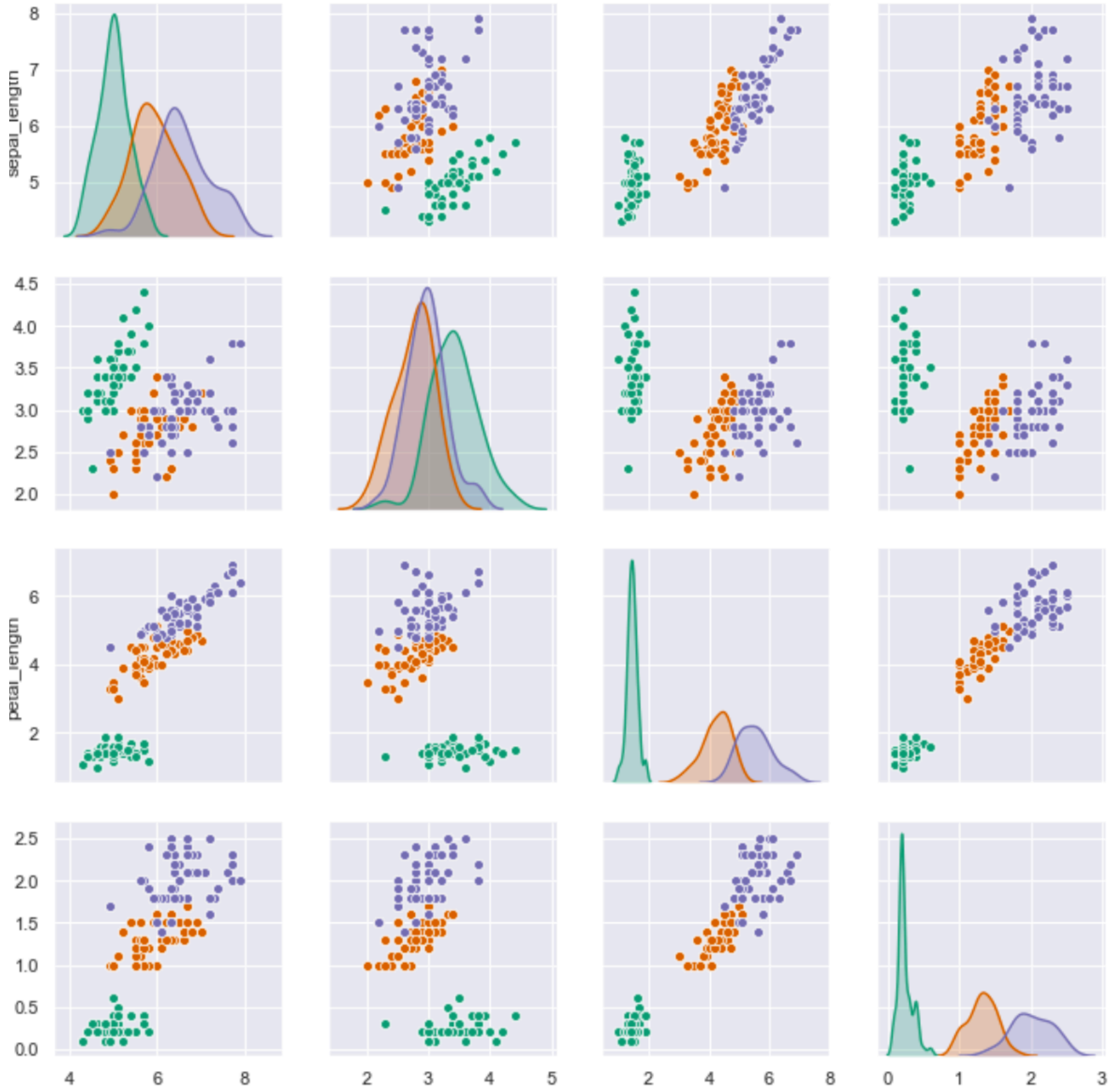
Z

The principal components matrix, V, acts as a rotation:

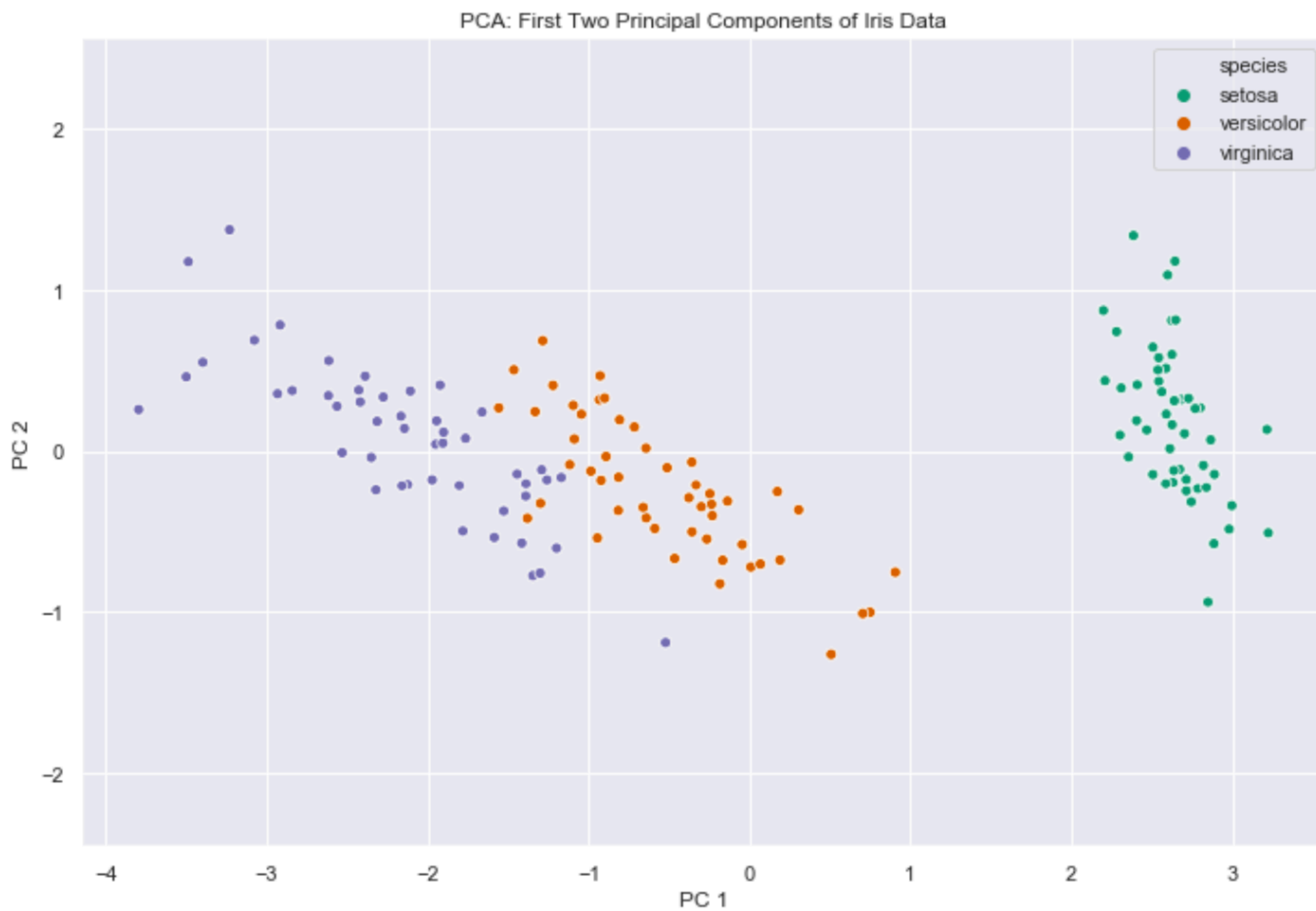
$$Z = XV$$

Columns of Z are new coordinates, called **loadings**.

Example: Iris Data



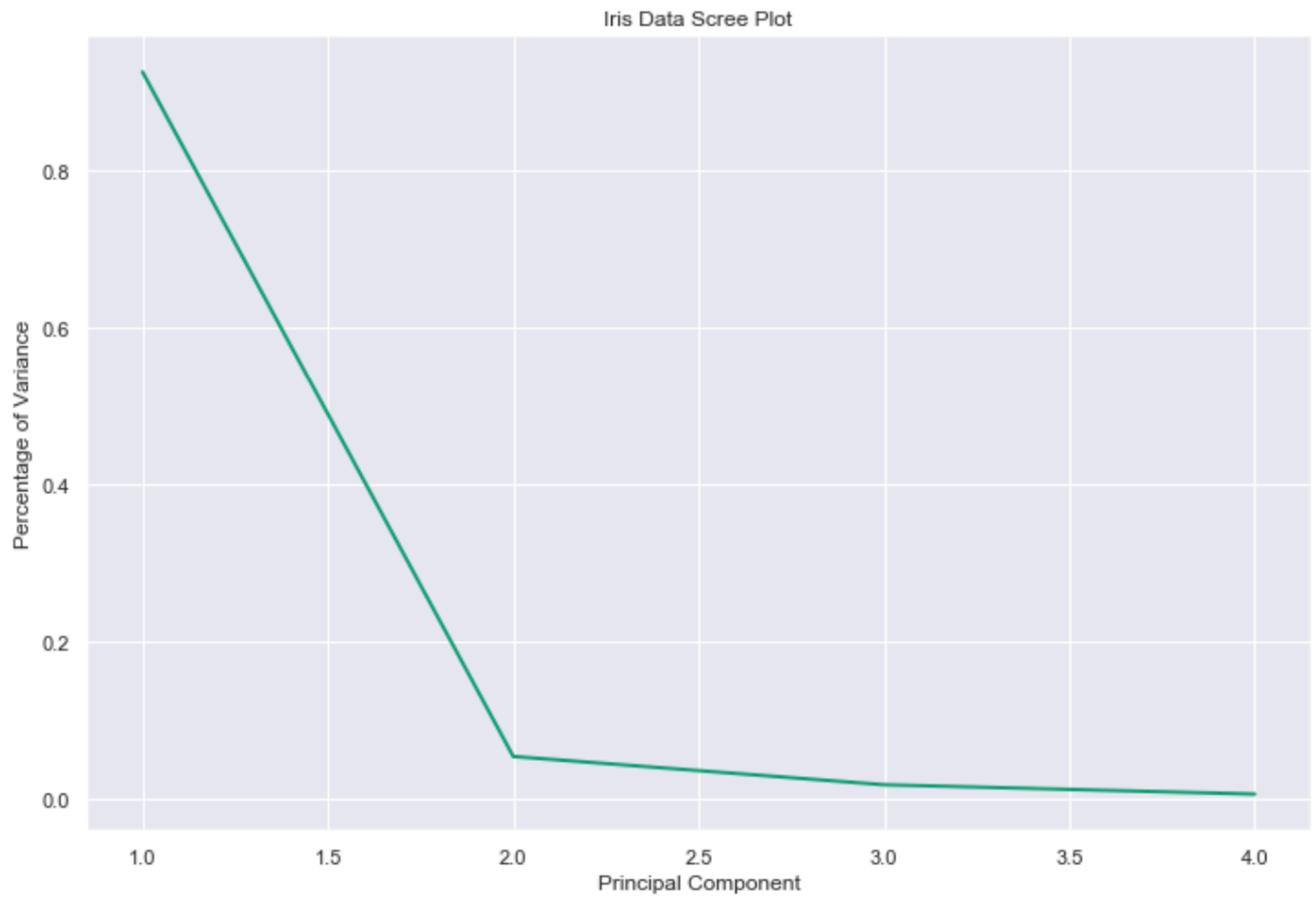
Example: Iris Data PCA



Eigenvectors: $V = \begin{pmatrix} -0.361387, & 0.656589, & 0.582030, & 0.315487 \\ 0.084523, & 0.730161, & -0.597911, & -0.319723 \\ -0.856671, & -0.173373, & -0.076236, & -0.479839 \\ -0.358289, & -0.075481, & -0.545831, & 0.753657 \end{pmatrix}$

Eigenvalues: $\lambda = (\boxed{4.22824171, 0.24267075,} 0.0782095, 0.02383509)$

Scree Plot: Eigenvalues (Variance)



Horizontal axis: which principal component (index k)

Vertical axis: proportion of variance: $\dfrac{\lambda_k}{\sum_{j=1}^{d} \lambda_j}$