23CSE301 Machine Learning

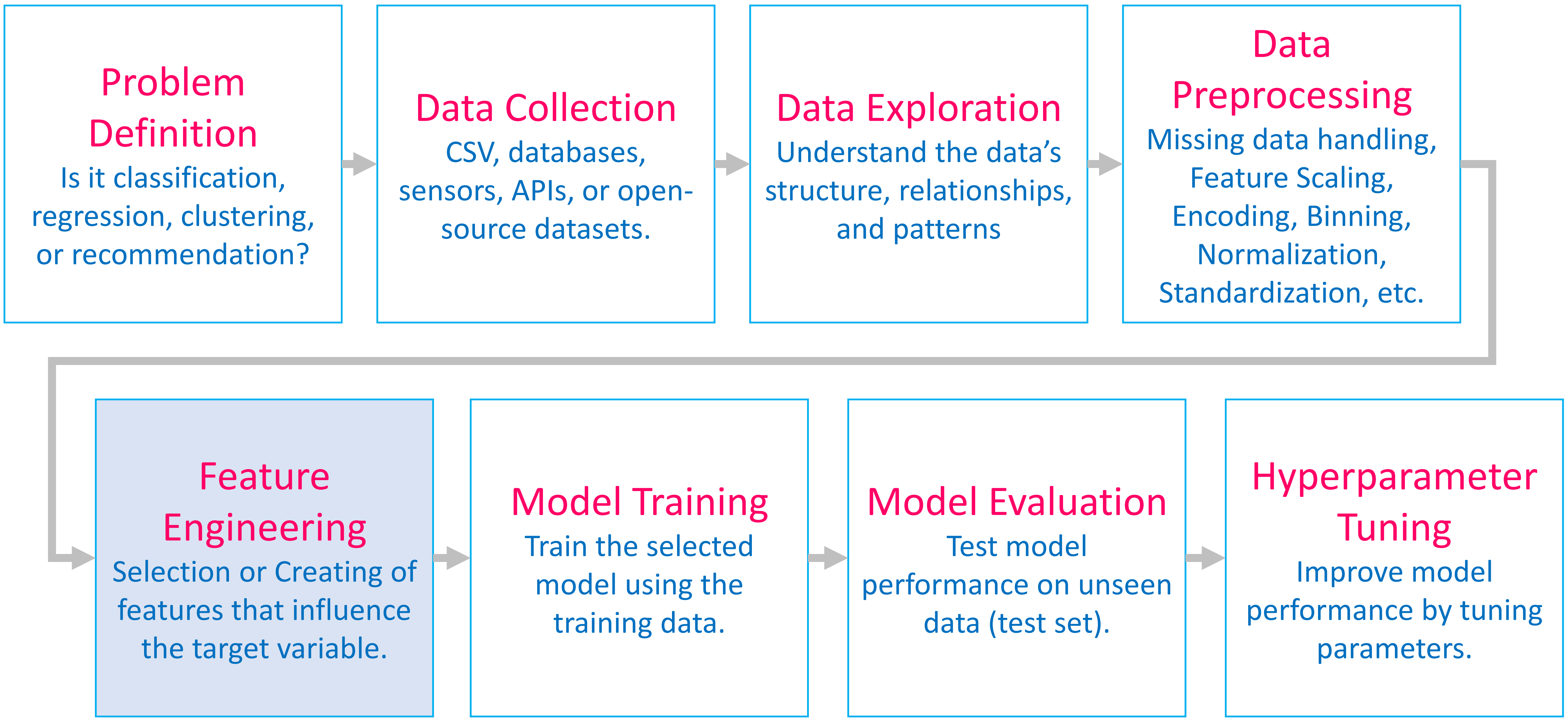
V Sem. CSE B

Practical – Week 5

Course Instructor: Dr. M. Anbazhagan

Pract. #	Experiment Title
P1-P3	• Introduction: Python, Pandas (scikit learn and other libraries) • Pre-processing: Dataset selection, Exploratory Data analysis and Feature engineering; Introduction to Colab/Jupyter Notebook, Pandas(Data Frames); Data Selection (iloc, loc); Sorting, Grouping merge, join, concat; Crosstab; Missing data treatment(fillna, dropna), Converting categorical values, Visualization(Line chart, Bar Chart, Pie chart, Scatter plot, Box plot); Distributions; Summary statistics.
Lab 1 Evaluation (P1 to P3)	
P4	Dimensionality Reduction Technique: PCA
P5	Feature Selection
P6	Regression Algorithms: Linear Regression
P7	Regression Algorithms: Logistic Regression
P8	Classification Algorithms: Decision Tree Classifier Classification Algorithms: K-Nearest Neighbor Classifier
Lab 2 Mid-Term exam (P1 to P8)	
P9	Classification Algorithms: Random Forest Classifier, ensemble learning.
P10	Classification Algorithms: Support Vector Machines
P11	Classification Algorithms: Perceptron
P12	Clustering: 1. K-Means Clustering 2. Agglomerative Clustering
Lab 3 Evaluation (P1 to P12)	

End-to-End Machine Learning Pipeline

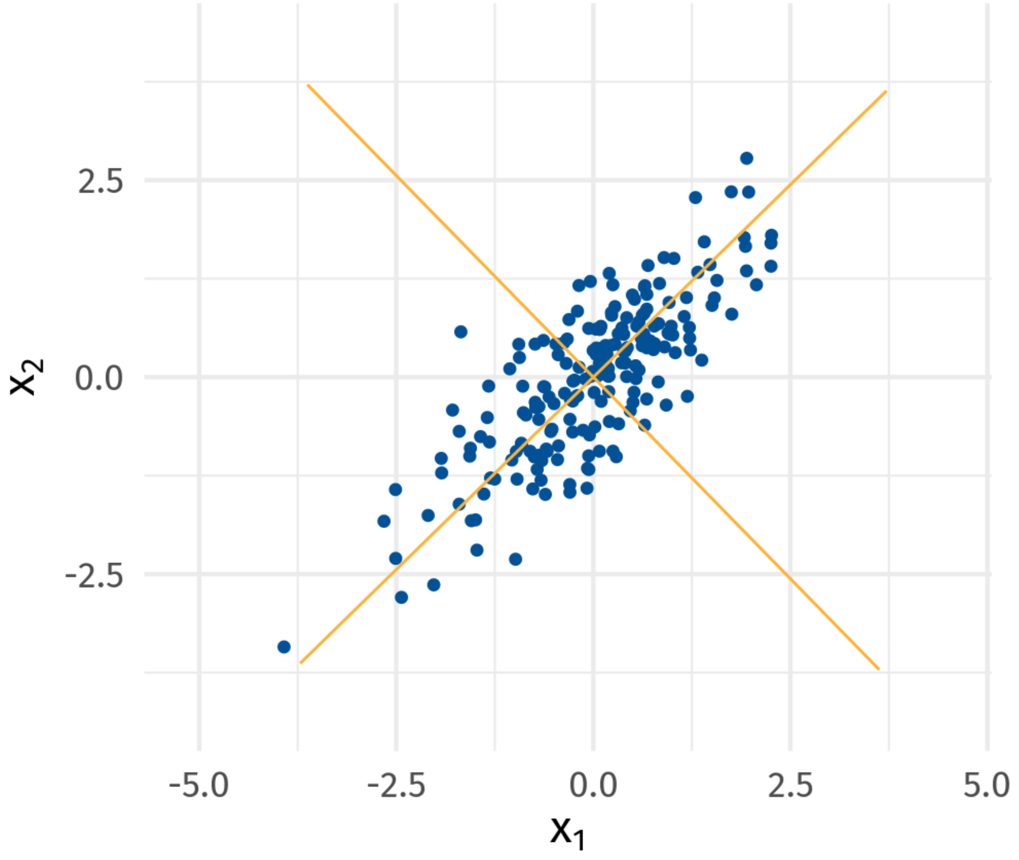


Principal Component Analysis

PCA transforms the data into new variables, called principal components, which capture the maximum variance

- PCA rotates the coordinate system to align with the directions where the data is most spread out

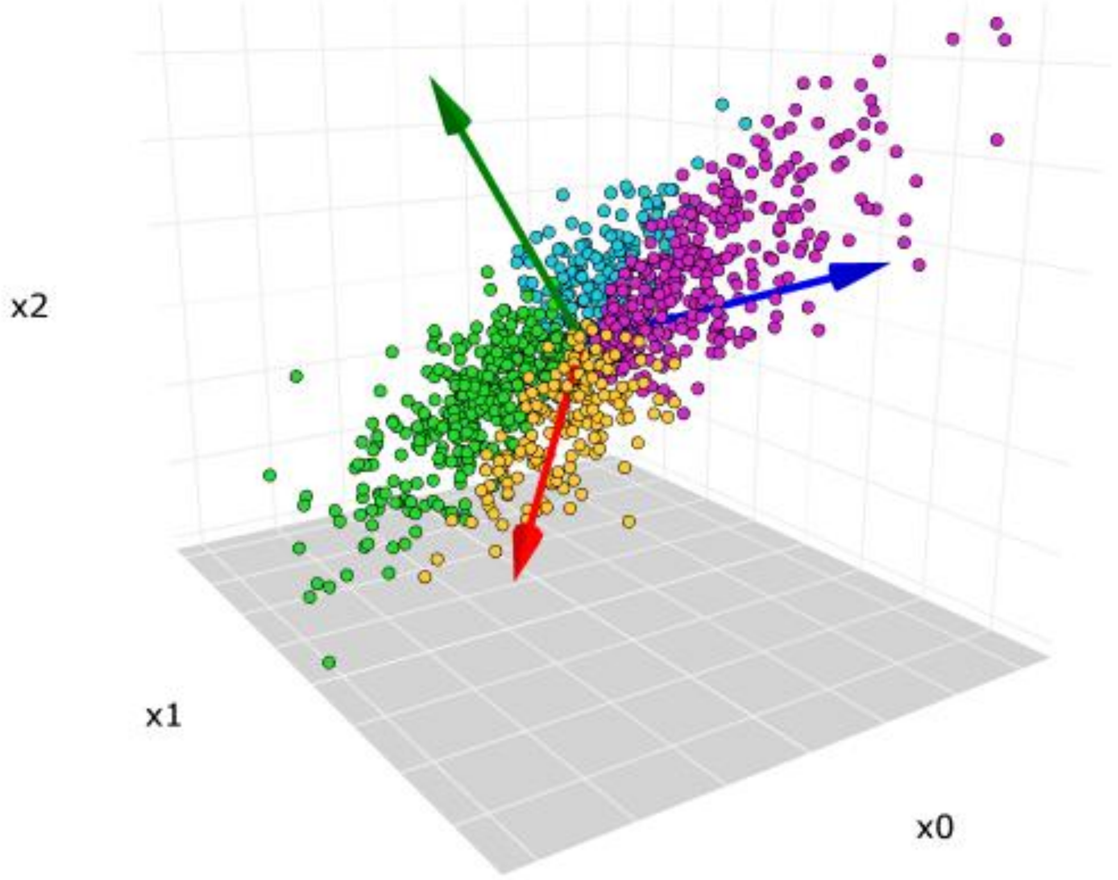
- The first principal component points along the longest stretch of the cloud, the second is perpendicular to it, and so on.



Principal Component Analysis

When there are more than two principal components, PCA still follows the same core logic. But instead of just rotating the coordinate system in 2D, it does so in higher-dimensional space.

Each subsequent component is orthogonal to all previous ones and captures the next most variance



PCA in Scikit-learn

PCA is implemented via the sklearn.decomposition.PCA class

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

Parameter	Description
n_components	Number of principal components to keep
whiten	If True, scales components to unit variance. Useful for downstream models
svd_solver	Algorithm used for decomposition: 'auto', 'full', 'arpack', ‘randomized’, etc.

Attribute	Description
components_	Eigen vectors
explained_variance_	Variance captured by each component
explained_variance_ratio	Proportion of total variance explained
singular_values	Singular values from SVD

Applying PCA to a Synthetic Data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns
```

```
# 1. Generate Synthetic Data
np.random.seed(42)
n_samples = 150
# Create three correlated features
x1 = np.random.normal(5, 2, n_samples)
x2 = 0.5 * x1 + np.random.normal(0, 1, n_samples)
x3 = -0.2 * x1 + 0.8 * x2 + np.random.normal(0, 1, n_samples)
```

Generates n_samples values from a normal distribution

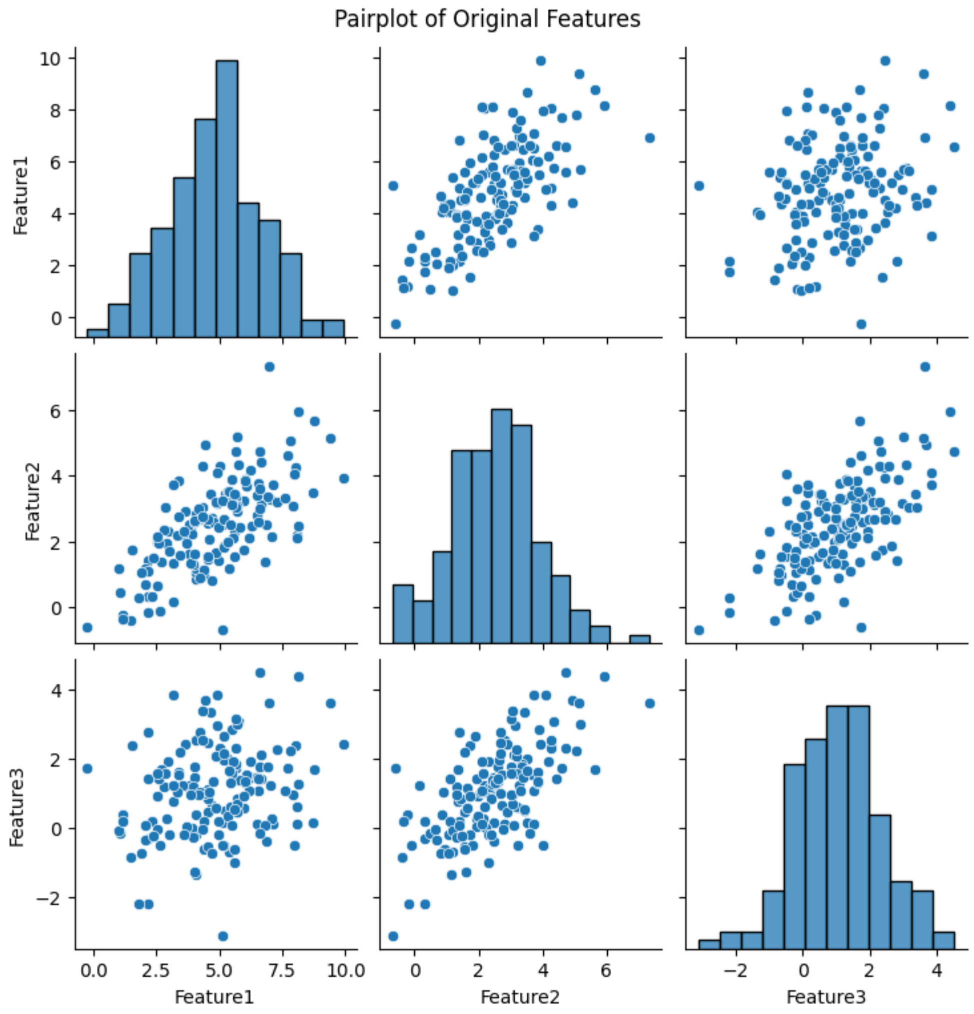
Half of x2 comes from x1, the rest is randomness

Mostly influenced by x2, slightly negatively influenced by x1

Applying PCA to a Synthetic Data

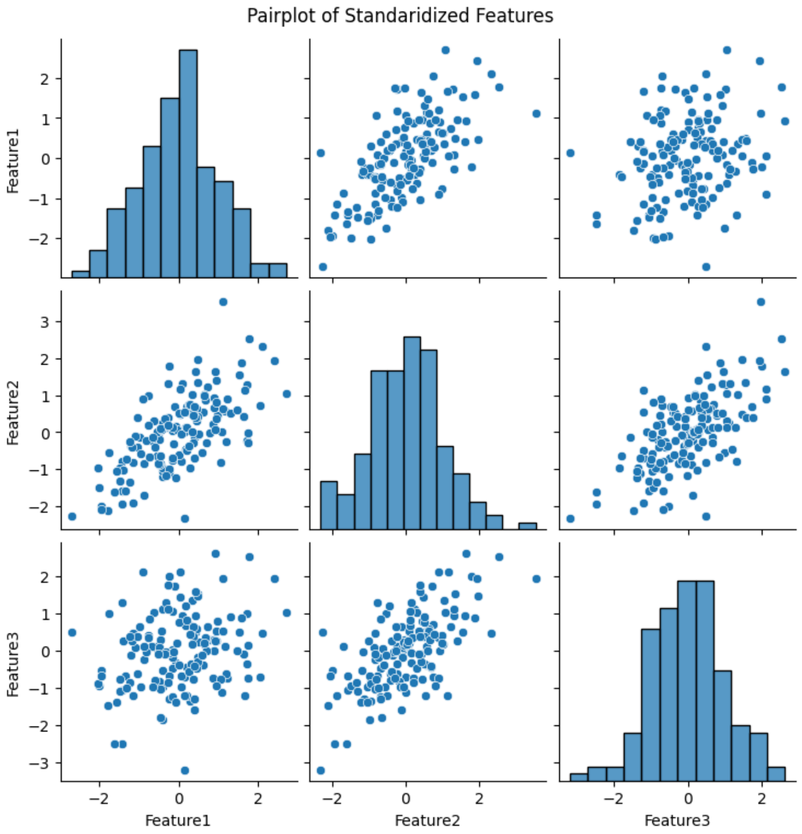
```
# Combine into a DataFrame
df = pd.DataFrame({'Feature1': x1, 'Feature2': x2, 'Feature3': x3})
print("Original Data Head:\n", df.head())
# 2. Visualize Pairplot
sns.pairplot(df)
plt.suptitle("Pairplot of Original Features", y=1.02)
plt.show()
```

	Feature1	Feature2	Feature3
0	5.993428	3.247207	0.570085
1	4.723471	2.708184	0.661672
2	6.295377	2.467664	1.462349
3	8.046060	4.255284	2.405385
4	4.531693	2.558919	1.119895



Applying PCA to a Synthetic Data

```
# 3. Standardize the Data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df)
scaled_df = pd.DataFrame(scaled_data, columns=['Feature1', 'Feature2', 'Feature3'])
```



Applying PCA to a Synthetic Data

```
# 4. PCA Transformation
pca = PCA()
pca_data = pca.fit_transform(scaled_data)
# Create PCA DataFrame
pca_df = pd.DataFrame(pca_data, columns=['PC1', 'PC2', 'PC3'])
print(pca.explained_variance_ratio_)


# Key Parameters
print("\nExplained Variance (Eigen values):")
print(pca.explained_variance_)
print("\nPrincipal Axes (components/Eigen Vectors):")
print(pca.components_)
print("\nMean of each feature before transformation:")
print(pca.mean_)
```

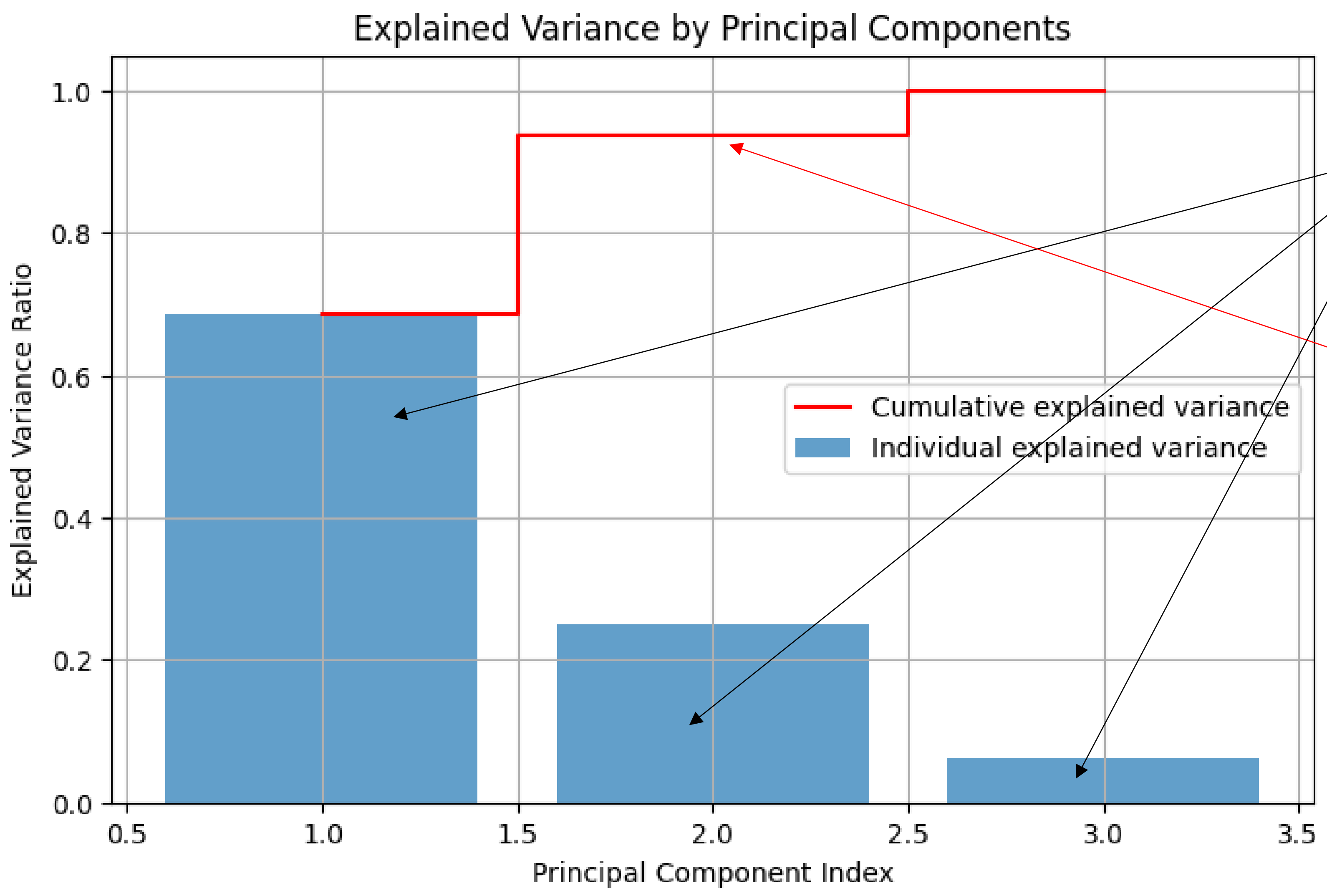
	PC1	PC2	PC3
0	0.492119	-0.712767	0.295728
1	-0.093916	-0.189688	0.292759
2	0.561230	-0.328085	-0.516940
3	2.301203	-0.469391	-0.307035
4	-0.036742	0.134006	0.102043

```
[0.68709989 0.2512522  0.06164792]

Explained Variance (Eigen values):
[2.07513388 0.75881536 0.18618498]

Principal Axes (components/Eigen Vectors):
[[ 0.53728421  0.65905299  0.52628399]
 [-0.69329002 -0.01022554  0.72058614]
 [-0.48028599  0.75202699 -0.45142084]]

Mean of each feature before transformation:
[-2.96059473e-16  3.43428989e-16  5.92118946e-17]
```

Applying PCA to a Synthetic Data

```python
# 5. Explained Variance Plot
explained_var = pca.explained_variance_ratio_


plt.figure(figsize=(8,5))
plt.bar(range(1, 4), explained_var, alpha=0.7, align='center', label='Individual explained variance')
plt.step(range(1, 4), np.cumsum(explained_var), where='mid', label='Cumulative explained variance',
color='red')
plt.xlabel('Principal Component Index')
plt.ylabel('Explained Variance Ratio')
plt.title('Explained Variance by Principal Components')
plt.legend()
plt.grid(True)
plt.show()
```

Applying PCA to a Synthetic Data



Explained Variance Plot

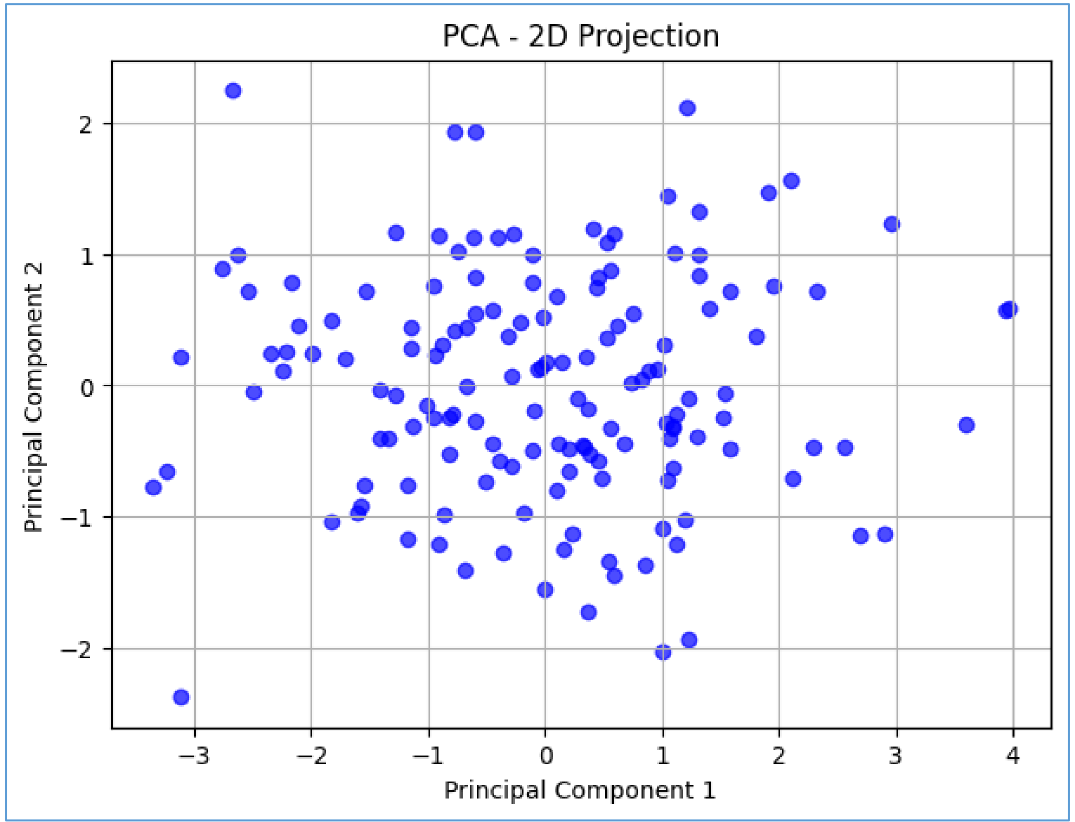
Variance explained by each individual principal component

Cumulative explained variance across components

You can reduce from 3 IVs → 2 using only PC1 and PC2, and still retain 95% of the data's information

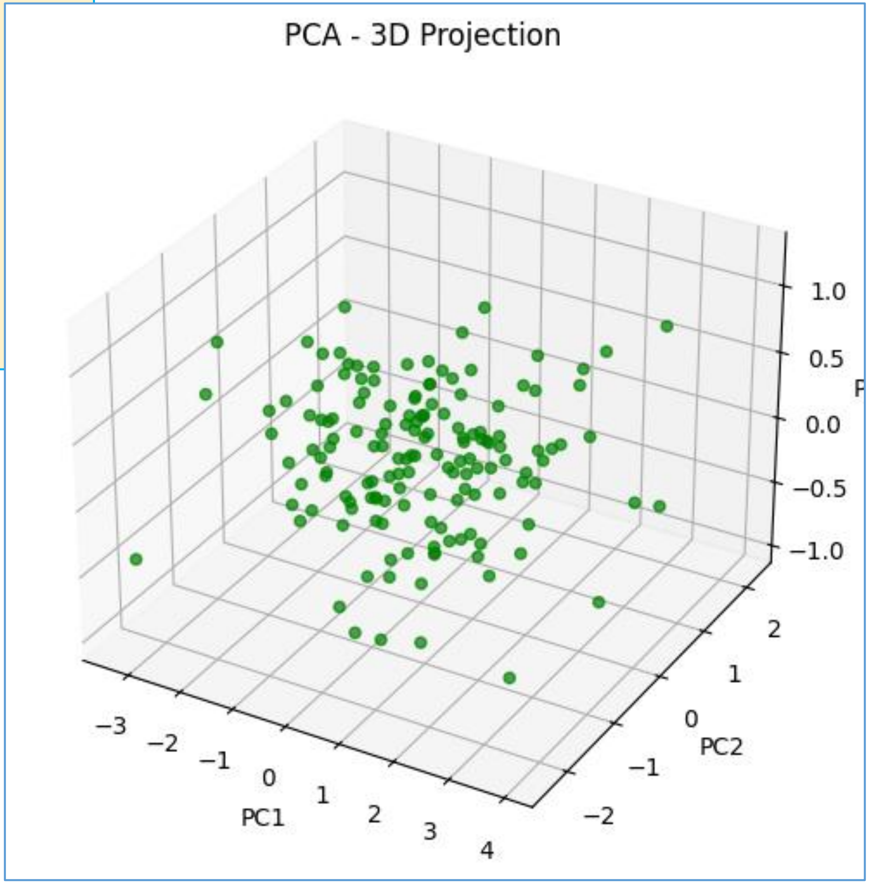
Applying PCA to a Synthetic Data

```
# 6. 2D PCA Scatter Plot (PC1 vs PC2)
plt.figure(figsize=(7, 5))
plt.scatter(pca_df['PC1'], pca_df['PC2'], c='blue', alpha=0.7)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('PCA - 2D Projection')
plt.grid(True)
plt.show()
```



Applying PCA to a Synthetic Data

```
# 7. 3D Scatter Plot (All 3 PCs)
fig = plt.figure(figsize=(8, 6))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(pca_df['PC1'], pca_df['PC2'], pca_df['PC3'], c='green', alpha=0.7)
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_zlabel('PC3')
ax.set_title('PCA - 3D Projection')
plt.show()
```



Exercise 1 - Week 5

- Dataset: Wine Recognition Dataset
  - You are tasked with applying PCA to the Wine Recognition dataset. Begin by loading the dataset and exploring its structure, print the shape, feature names, and class distribution. Standardize the feature matrix using StandardScaler to ensure each feature contributes equally. Then, perform PCA to reduce the dataset from 13 dimensions to 2 principal components. Display the principal components, their corresponding eigenvalues, and the explained variance ratios. Visualize the PCA-transformed data using a 2D scatter plot, with points color-coded by wine class. Additionally, plot the explained variance ratio for each component and include a cumulative variance line to determine how much variance is retained as components are added. Finally, interpret your results: how many components are required to retain at least 95% of the total variance, and how well are the wine classes separated in the 2D space? As a bonus, you may extend the PCA to 3 components and visualize the result in 3D to observe class separability more clearly.
  - Note: Inference can be written into a text/comment cell at the very last.

Exercise 2 - Week 5

- Dataset: UCI Human Activity Recognition (HAR) dataset
  - You are tasked with performing PCA on the UCI Human Activity Recognition dataset, which contains 561 features derived from smartphone accelerometer and gyroscope signals. Begin by loading the dataset and inspecting its shape, feature distribution, and class labels representing six human activities (e.g., walking, sitting, standing). Standardize the features using StandardScaler to normalize the input space. Apply PCA to reduce the dimensionality of the dataset while preserving most of the variance. Identify how many principal components are needed to retain at least 90% and 95% of the total variance. Plot the explained variance ratio and the cumulative variance to support your analysis. Visualize the data projected onto the first two and first three principal components using 2D and 3D scatter plots, color-coded by activity class. Discuss whether PCA helps in visualizing class separability in lower dimensions.
  - Note: Inference can be written into a text/comment cell at the very last.