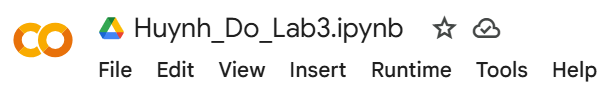
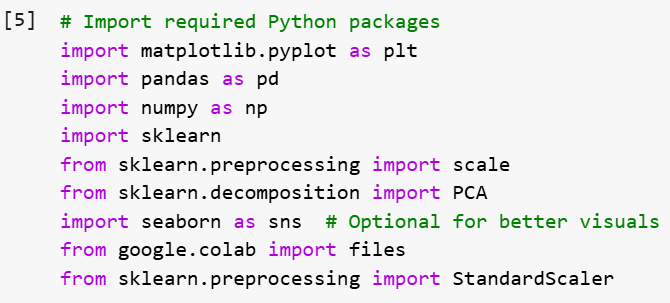
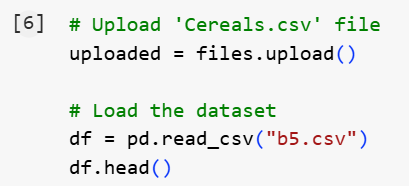
**Huynh Do Lab#3:  
Objective**:   
This lab assignment works with Principal Component Analysis (PCA) to expose how they can determine the best explained variance ratio. This is a statistical procedure that is used to reduce 

1. **Import libraries**

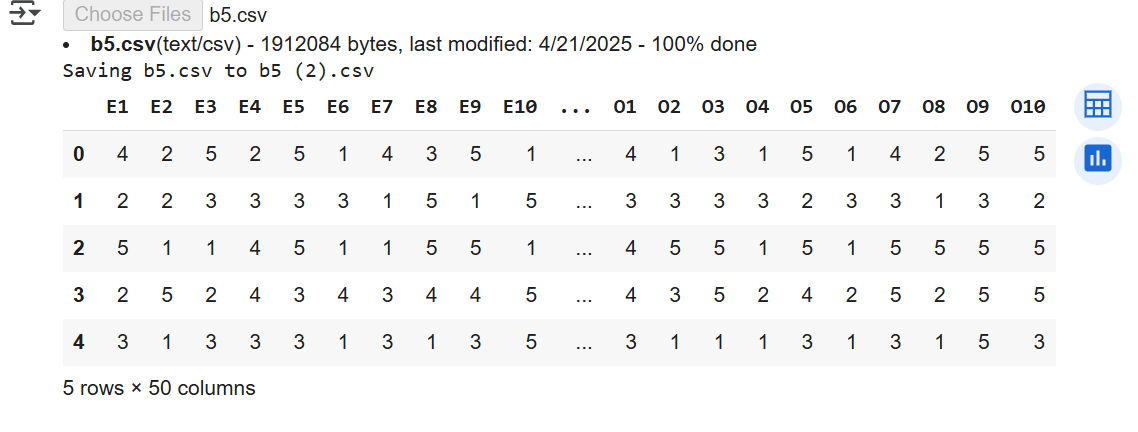
  
The code above imports several Python libraries commonly used in data analysis.

* 1. **matplotlib.pyplot** – Used to create basic graphs like scatter plots, line charts, and bar graphs.
  2. **pandas** – Loads and manages tabular data (like CSVs) into an easy-to-use DataFrame.
  3. **numpy** – Powers fast mathematical operations, especially with large arrays of numbers.
  4. **sklearn** – Provides machine learning algorithms and data processing tools.
  5. **scale** – Instantly standardizes your data to have a mean of 0 and variance of 1.
  6. **PCA** – Reduces the number of variables while keeping the important patterns in your data.
  7. **seaborn** – Makes fancier, cleaner-looking graphs with less code than plain matplotlib.
  8. **google.colab.files** – Lets you upload and download files when working inside Google Colab.
  9. **StandardScaler** – Another way to standardize data, but more control across train/test splits.

1. **Upload file b5.csv**

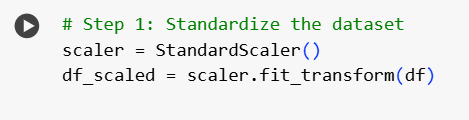


The above screen shot is used to upload and load a CSV file named **b5.csv** into a Pandas Data Frame in a Google Colab environment.

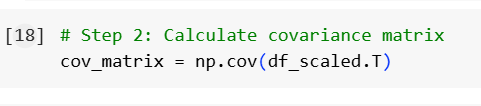
When upload is done:  


1. **Process PCA**

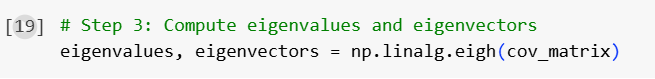
* **Step 1**



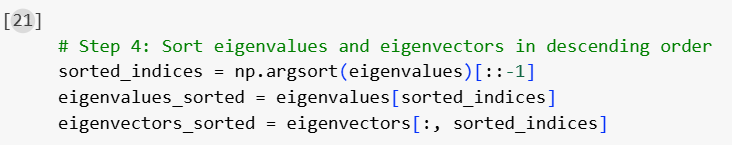
* **scaler = StandardScaler()** Creates a scaler object to standardize features (zero mean, unit variance).
* **df\_scaled = scaler.fit\_transform(df)** Fits the scaler to the dataset and transforms the data into a standardized form.
* **Step 2**



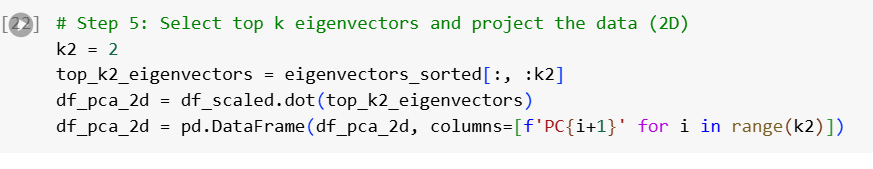
* **cov\_matrix = np.cov(df\_scaled.T)** Calculates the covariance matrix by transposing the standardized data so features are treated as variables.
* **Step 3**



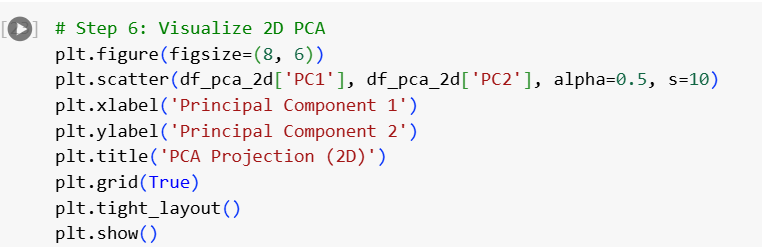
* **eigenvalues, eigenvectors = np.linalg.eigh(cov\_matrix)** Computes the eigenvalues and eigenvectors of the covariance matrix to find directions and magnitudes of variance.
* **Step 4**



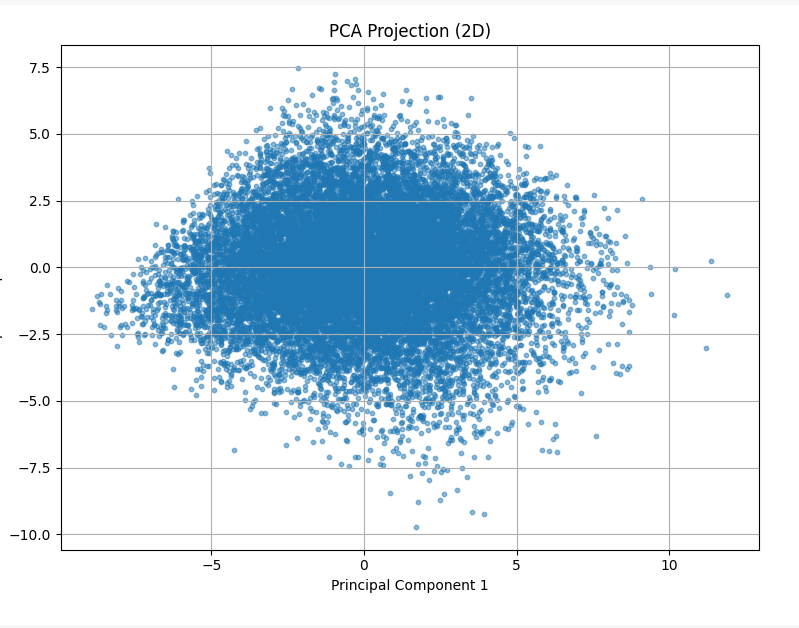
* **sorted\_indices = np.argsort(eigenvalues)[::-1]** Finds indices that sort eigenvalues from largest to smallest.
* **eigenvalues\_sorted = eigenvalues[sorted\_indices]** Reorders eigenvalues in descending order.
* **Step** 5



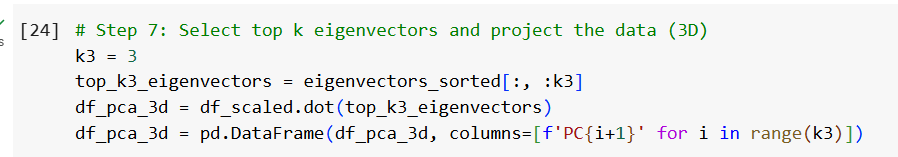
* **k2 = 2** Sets the number of principal components to keep at 2 for 2D projection.
* **top\_k2\_eigenvectors = eigenvectors\_sorted[:, :k2]** Selects the first two sorted eigenvectors.
* **df\_pca\_2d = df\_scaled.dot(top\_k2\_eigenvectors)** Projects the standardized data onto the 2D principal component space.
* **df\_pca\_2d = pd.DataFrame(df\_pca\_2d, columns=[f'PC{i+1}' for i in range(k2)])** Converts the projected data into a labeled DataFrame with PC1 and PC2.
* **Step 6**



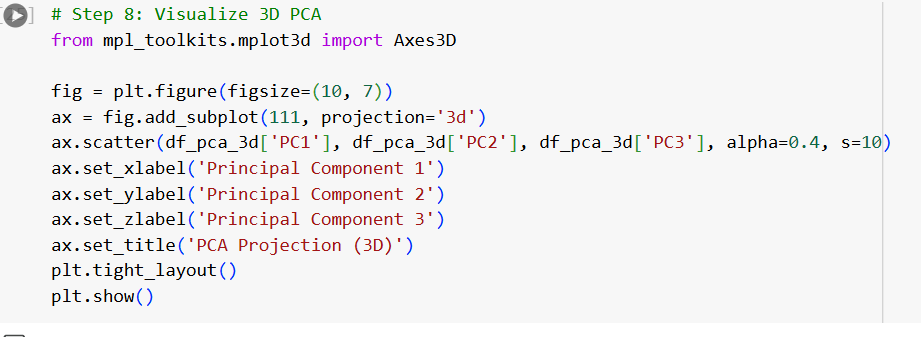
* **plt.figure(figsize=(8, 6))** Creates a new figure with specified size.
* **plt.scatter(df\_pca\_2d['PC1'], df\_pca\_2d['PC2'], alpha=0.5, s=10)** Plots the 2D PCA result as a scatter plot.
* **plt.xlabel('Principal Component 1')** Labels the x-axis.
* **plt.ylabel('Principal Component 2')** Labels the y-axis.
* **plt.title('PCA Projection (2D)')** Adds a title to the plot.
* **plt.grid(True)** Enables grid lines for better readability.



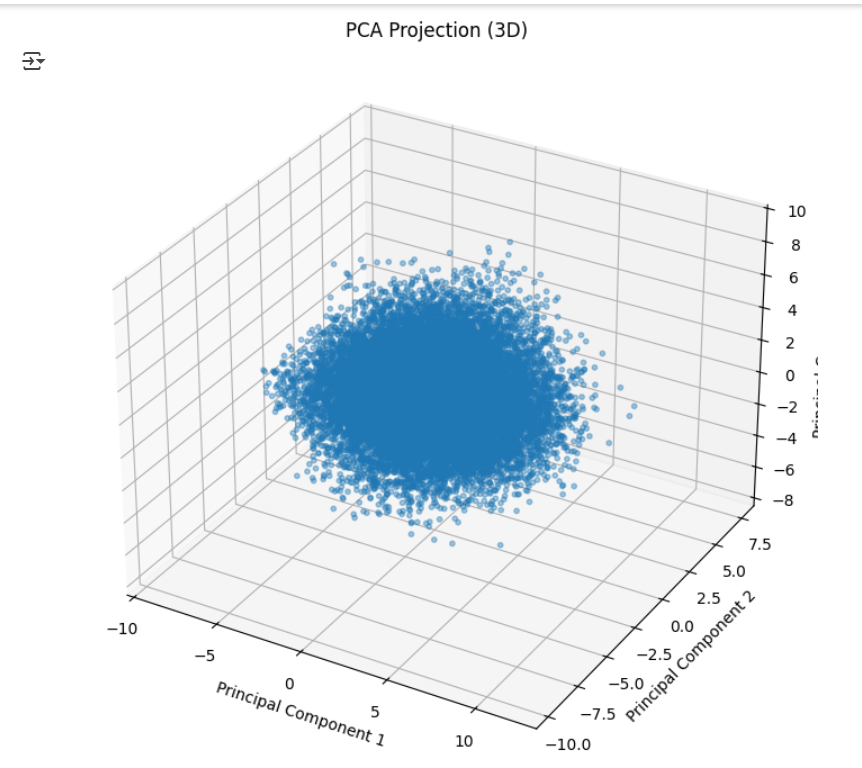
* **Step 7**



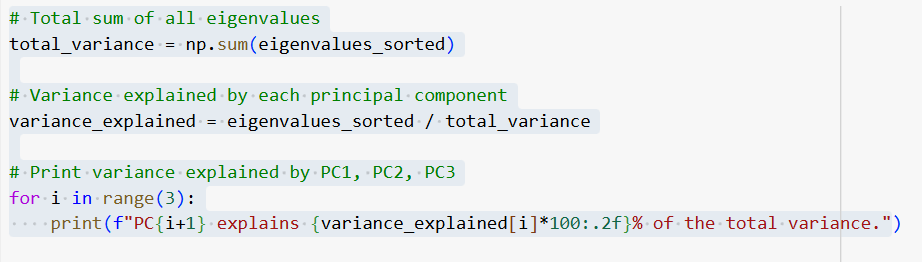
* **k3 = 3** Sets the number of principal components to 3 for 3D projection.
* **top\_k3\_eigenvectors = eigenvectors\_sorted[:, :k3]** Selects the first three sorted eigenvectors.
* **df\_pca\_3d = df\_scaled.dot(top\_k3\_eigenvectors)** Projects the standardized data onto the 3D principal component space.
* **df\_pca\_3d = pd.DataFrame(df\_pca\_3d, columns=[f'PC{i+1}' for i in range(k3)])** Converts the projected data into a labeled DataFrame with PC1, PC2, and PC3.
* **Step 8**



* **from mpl\_toolkits.mplot3d import Axes3D** Imports 3D plotting tools for matplotlib.
* **fig = plt.figure(figsize=(10, 7))** Creates a new figure with specified size.
* **ax = fig.add\_subplot(111, projection='3d')** Adds a 3D subplot to the figure.
* **ax.scatter(df\_pca\_3d['PC1'], df\_pca\_3d['PC2'], df\_pca\_3d['PC3'], alpha=0.4, s=10)** Plots the 3D PCA result as a scatter plot.
* **ax.set\_xlabel('Principal Component 1')** Labels the x-axis.
* **ax.set\_ylabel('Principal Component 2')** Labels the y-axis.
* **ax.set\_zlabel('Principal Component 3')** Labels the z-axis.
* **ax.set\_title('PCA Projection (3D)')** Adds a title to the 3D plot.



* Total summary



PC1 explains 16.10% of the total variance.

PC2 explains 9.25% of the total variance.

PC3 explains 7.53% of the total variance.