Dimensionality Reduction: Principal Component Analysis

Team Members:

Manthan Jethva

Pranav Kalariya

Instructor: Dr. Yasser Alginahi

Date: 10th November 2023

University of Windsor

Topics to be covered !!

1. Introduction to Dimensionality Reduction
2. What is PCA?
3. Mathematical Foundations
4. PCA Workflow
5. Variance and Information
6. Scree Plot and Eigenvalues
7. Implementing PCA
8. Advantages of PCA
9. Challenges and Limitations of PCA
10. Conclusion
11. References

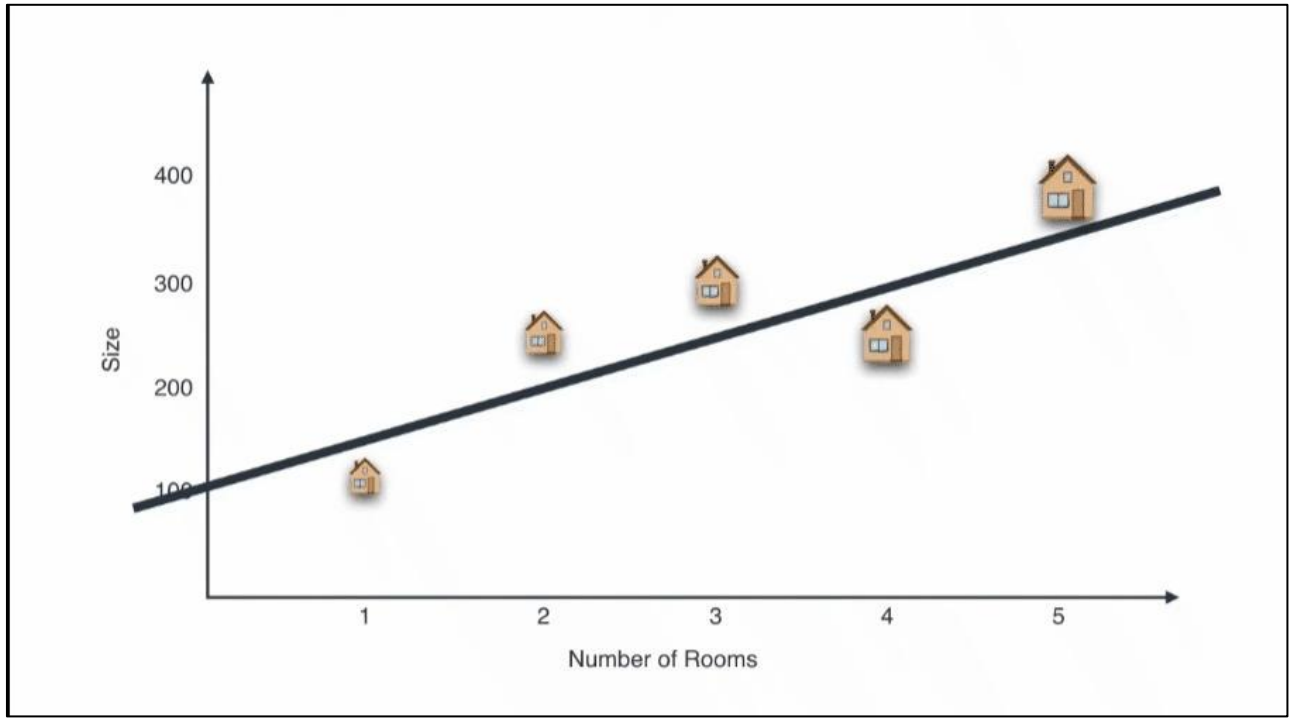
Introduction to Dimensionality Reduction

- Dimensionality reduction refers to the techniques that reduce the number of input variables in dataset.

- Why Dimensionality Reduction:
  - Less Computation and training time
  - Removing redundancy
  - Memory optimization
  - Makes data easy for plotting in graphs
  - Finding out most significant features and skip the rest

What is PCA?

PCA is a technique for reducing the dimensionality of dataset, increasing interpretability but at the same time minimizing information loss.

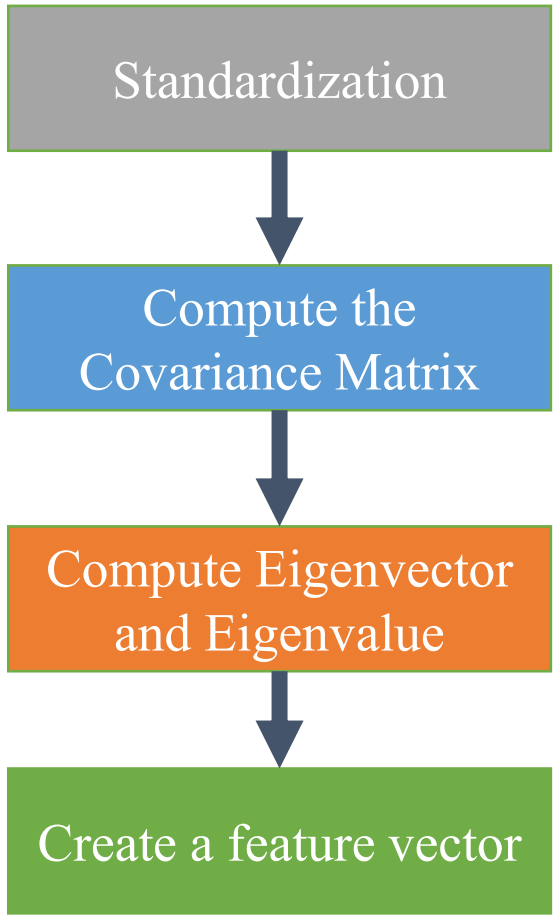


Source: https://youtu.be/g-Hb26agBFg?si=dx81XmSygRrFzMPf

Variance and Information

- PCA aims to find orthogonal axes (principal components) in the data such that the variance of the data along these axes is maximized.
- The first principal component accounts for the most significant variance, the second principal component accounts for the second most significant variance, and so on.
- This property helps in retaining as much information as possible while reducing the dimensionality of the data.
- One of the primary uses of PCA is to determine how much variance in the original data is retained by each principal component.
- We can calculate the proportion of variance explained by each principal component, which helps us understand which components contribute the most to the data's variability. (eigen values)

PCA Workflow



Standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.

Understand how the variables of the input data set are varying from the mean with respect to each other for checking relationship between them.

Eigenvectors and Eigenvalues are linear algebra concepts that we compute from the covariance matrix to determine **principal components**.

Check if need to discard components of lesser significance (of low eigenvalues) and form a matrix of vectors that we call Feature vector.

Mathematical Foundations

- **Co-variance Matrix**

  The covariance matrix is a $p \times p$ symmetric matrix (where p is the number of dimensions) that has entries of the covariances associated with all possible pairs of the initial variables. [2]

- **Eigenvectors**

  Eigenvectors are the linear combination of the original datapoints and in PCA they are the principal components.

- **Eigenvalue**

  Eigenvalues can be thought of as a measure of the amount of variation in the data that is explained by each principal component.

More Info on Eigenvalues and Eigenvectors:
https://lpsa.swarthmore.edu/MtrxVibe/EigMat/MatrixEigen.html

How to calculate Co-variance

In the context of Principal Component Analysis (PCA), you can calculate the variance of a set of data points in the original feature space or in the space of principal components.

Variance in the Original Feature Space: To calculate the variance of a set of data points in the original feature space, you can use the following formula.

$$\mathbf{cov(x, y)= [\ \sum_{i=0}^{n} (x_i - \bar{x})\ (y_i - \bar{y})\ ]\ /(n-1)}$$

Where:

- cov(x, y) is the variance.
- n is the number of data points in each column X and Y.
- Σ represents the summation.
- x_i and y_i is an individual data point in column x and y respectively.
- $\bar{x}$ is the mean (average) of the data points in column X.
- $\bar{y}$ is the mean (average) of the data points in column Y.

Variance in the Space of Principal Components

To calculate the variance of data along a particular principal component in the space of principal components, you can use the eigenvalues of the covariance matrix.

Each eigenvalue represents the variance of the data along the corresponding principal component.

Let $\lambda 1$, $\lambda 2$, $\lambda 3$, ... be the eigenvalues, and PC1, PC2, PC3, ... be the principal components in descending order of eigenvalues (variance). The variance explained by a specific principal component (PC_i) is given by:

- **Variance Explained by $PC_i = \lambda$**

Calculate Co-variance Matrix

For example, for a 3-dimensional data set with 3 variables *x*, *y*, and *z*, the covariance matrix is a 3×3 data matrix of this from:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}$$

Source: https://tinyurl.com/2ftxh3kx

Since the covariance of a variable with itself is its variance (Cov(a,a)=Var(a)), in the main diagonal, we actually have the variances of each initial variable.

Since the covariance is commutative (Cov(a,b)=Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal.[2]

Calculate Co-variance Matrix

x	y
3	7
2	4

x = (3+2)/2=5/2

$\bar{y}$= (7+4)/2=11/2

Cov(x,y)=(3-5/2)(7-11/2)+(7-5/2)(4-11/2)=-1

Var(x)= (3-5/2)^2 + (2-5/2)^2 = ½

Var(y)= (7-11/2)^2 + (4-11/2)^2 = 18/4

Covariance matrix = $\begin{bmatrix} cov(x,x) & cov(y,x) \\ cov(x,y) & cov(y,y) \end{bmatrix} = \begin{bmatrix} 1/2 & -1 \\ -1 & 9/2 \end{bmatrix}$

Example of Eigenvalue and Eigen vector

Equation for Eigen value problem:

$$\mathbf{A}\cdot\mathbf{v}=\lambda\cdot\mathbf{v}$$

- In this equation **A** is an n-by-n matrix, **v** is a non-zero n-by-1 vector and λ is a scalar (which may be either real or complex). Any value of λ for which this equation has a solution is known as an eigenvalue of the matrix **A**. It is sometimes also called the characteristic value. The vector, **v**, which corresponds to this value is called an eigenvector. The eigenvalue problem can be rewritten as

$$\mathbf{A}\cdot\mathbf{v}-\lambda\cdot\mathbf{v}=0$$
$$\mathbf{A}\cdot\mathbf{v}-\lambda\cdot\mathbf{I}\cdot\mathbf{v}=0$$
$$(\mathbf{A}-\lambda\cdot\mathbf{I})\cdot\mathbf{v}=0$$

- If **v** is non-zero, this equation will only have a solution if

$$|\mathbf{A}-\lambda\cdot\mathbf{I}|=0$$

Example of Eigenvalue

$A = \begin{matrix} 2 & 2 \\ 1 & 3 \end{matrix}$ => $\lambda = ?$

$|A-\lambda I| = 0$

$$\left\| \begin{matrix} 2-\lambda & 2 \\ 1 & 3-\lambda \end{matrix} \right\| = 0$$

$\lambda^2 - 5\lambda + 4 = 0$

$\lambda = 1$ or $\lambda = 4$

Example of Eigen vector

For $\lambda_1 = 1$,

$A \cdot v1 = \lambda_1 \cdot v1$

$(A - \lambda_1) \cdot v1 = 0$

$$\begin{Vmatrix} 2-\lambda_1 & 2 \\ 1 & 3-\lambda_1 \end{Vmatrix} \cdot v1 = 0$$

$$\begin{Vmatrix} 1 & 2 \\ 1 & 2 \end{Vmatrix} \cdot \begin{matrix} v_{1,1} \\ v_{1,2} \end{matrix} = 0 \quad (\therefore \lambda_1 = 1)$$

$v_{1,1} + 2\, v_{1,2} = 0$

$v_{1,1} = 2\, v_{1,2} \quad \Rightarrow \quad v_1 = k_1 \begin{Vmatrix} 1 \\ 2 \end{Vmatrix}$

For $\lambda_2 = 4$,

$A \cdot v_2 = \lambda_2 \cdot v_2$

$(A - \lambda_2) \cdot v_2 = 0$

$$\begin{Vmatrix} 2-\lambda_2 & 2 \\ 1 & 3-\lambda_2 \end{Vmatrix} \cdot v_2 = 0$$

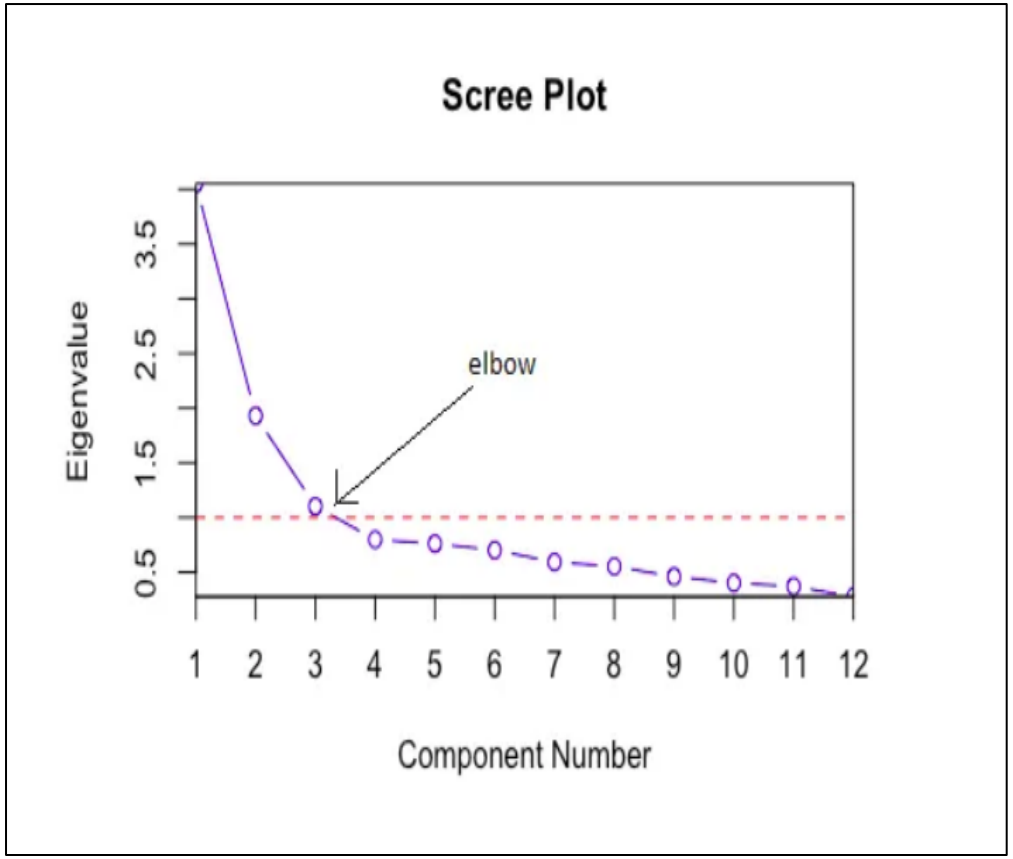
$$\begin{Vmatrix} -2 & 2 \\ 1 & -1 \end{Vmatrix} \cdot \begin{matrix} v_{2,1} \\ v_{2,2} \end{matrix} = 0 \quad (\therefore \lambda_2 = 4)$$

$v_{2,1} - v_{2,2} = 0$

$v_{2,1} = v_{2,2} \quad \Rightarrow \quad v_2 = k_2 \begin{Vmatrix} 1 \\ 1 \end{Vmatrix}$

Scree Plot

- A common method for determining the number of PCs to be retained is a graphical representation known as a scree plot.
- A Scree Plot is a simple line segment plot that shows the eigenvalues for each individual PC. It shows the eigenvalues on the y-axis and the number of factors on the x-axis.
- It always displays a downward curve.
- The scree plot criterion looks for the “elbow” in the curve and selects all components just before the line flattens out.



Source: https://tinyurl.com/47c8n7hs

Implementing PCA

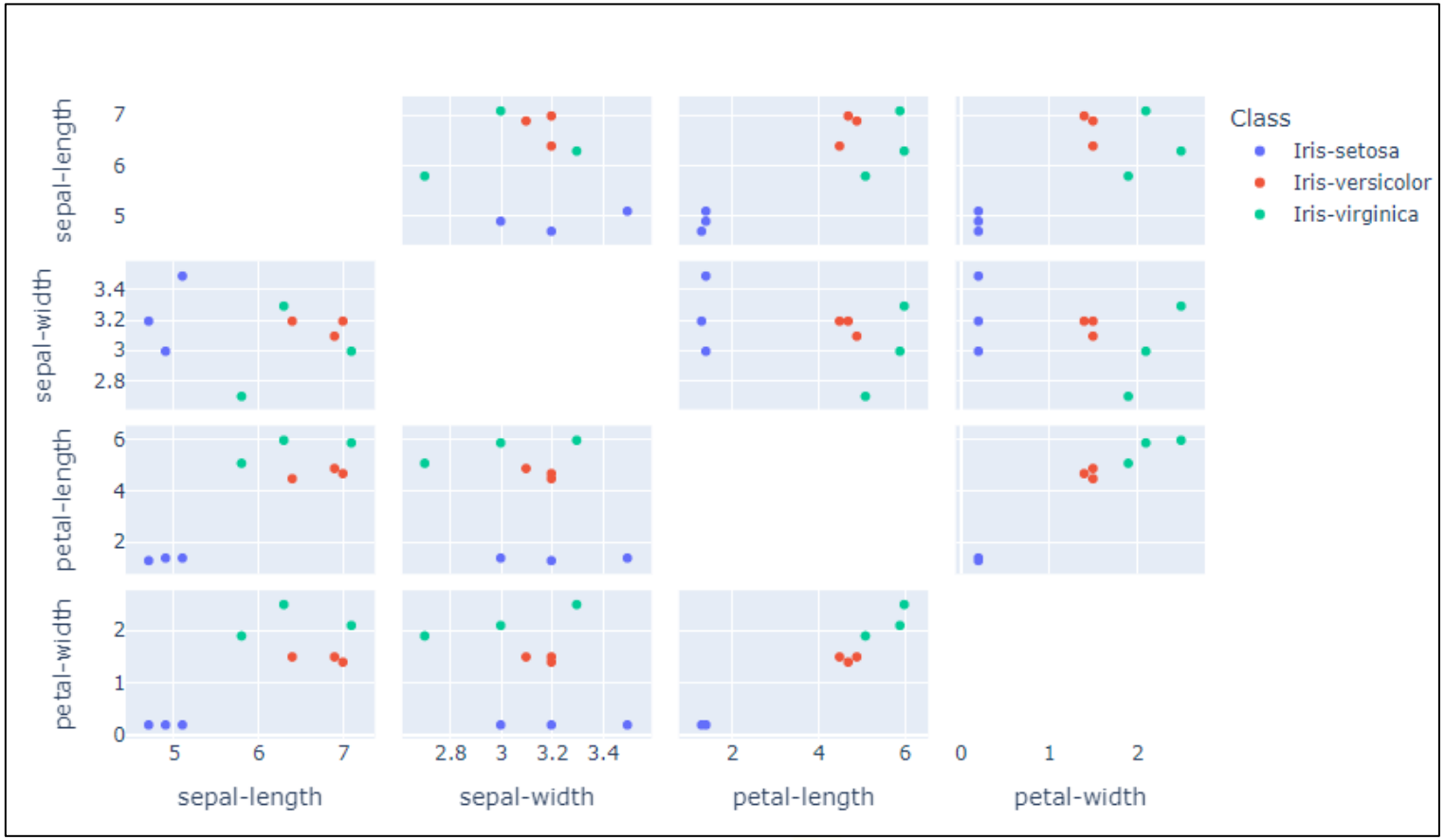
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import plotly.express as px
```

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class']
dataset = pd.read_csv(url, names=names)

dataset = dataset.groupby('Class').head(3)


dataset
```

	sepal-length	sepal-width	petal-length	petal-width	Class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
50	7.0	3.2	4.7	1.4	Iris-versicolor
51	6.4	3.2	4.5	1.5	Iris-versicolor
52	6.9	3.1	4.9	1.5	Iris-versicolor
100	6.3	3.3	6.0	2.5	Iris-virginica
101	5.8	2.7	5.1	1.9	Iris-virginica
102	7.1	3.0	5.9	2.1	Iris-virginica

Implementing PCA

```
X = dataset.drop('Class', 1)
y = dataset['Class']
fig = px.scatter_matrix(
    dataset,
    dimensions=X,
    color="Class"
)
fig.update_traces(diagonal_visible=False)
fig.show()
```



Implementing PCA

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=59)
classifier = RandomForestClassifier(max_depth=100, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print('Accuracy: ' + str(accuracy_score(y_test, y_pred)))
```

```
Accuracy: 1.0
```

Implementing PCA

```
sc = StandardScaler()
X = sc.fit_transform(X)
X
```

```
array([[-1.04454175,  1.73925271, -1.35065657, -1.30321735],
       [-1.27106888, -0.63245553, -1.35065657, -1.30321735],
       [-1.497596  ,  0.31622777, -1.40444378, -1.30321735],
       [ 1.10746595,  0.31622777,  0.42432131,  0.14778753],
       [ 0.42788457,  0.31622777,  0.31674689,  0.26870461],
       [ 0.99420239, -0.15811388,  0.53189573,  0.26870461],
       [ 0.31462101,  0.79056942,  1.12355502,  1.47787534],
       [-0.25169681, -2.05548048,  0.63947014,  0.7523729 ],
       [ 1.22072952, -0.63245553,  1.06976781,  0.99420705]])
```

Implementing PCA

```
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X)
X_pca
```

```
array([[ 2.48782748, -1.20033641,  0.09217077],
       [ 2.05813165,  1.12798853, -0.16836501],
       [ 2.42910565,  0.29093988,  0.16830729],
       [-0.84478218, -0.65506933, -0.62091116],
       [-0.49135873, -0.45992044, -0.05079488],
       [-1.02666335, -0.18373948, -0.52915376],
       [-1.4822095 , -0.99927635,  0.98731803],
       [-1.14595344,  1.95021883,  0.38174681],
       [-1.98409758,  0.12919477, -0.26031808]])
```

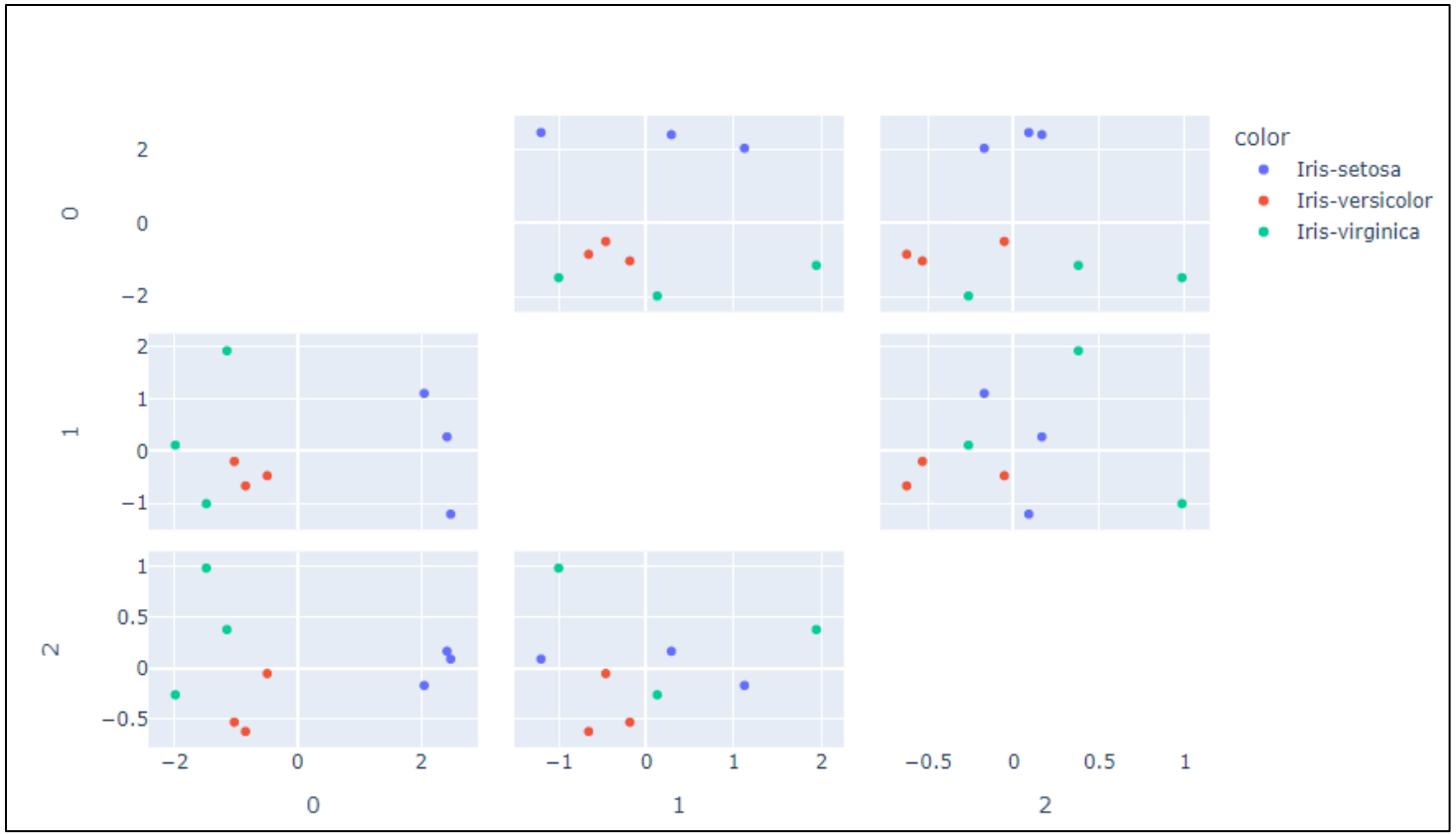
Implementing PCA

```
X_train, X_test, y_train, y_test = train_test_split(X_pca, y, test_size=0.2, random_state=59)
classifier = RandomForestClassifier(max_depth=100, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
print('Accuracy: ' + str(accuracy_score(y_test, y_pred)))
```

```
Accuracy: 0.5
```

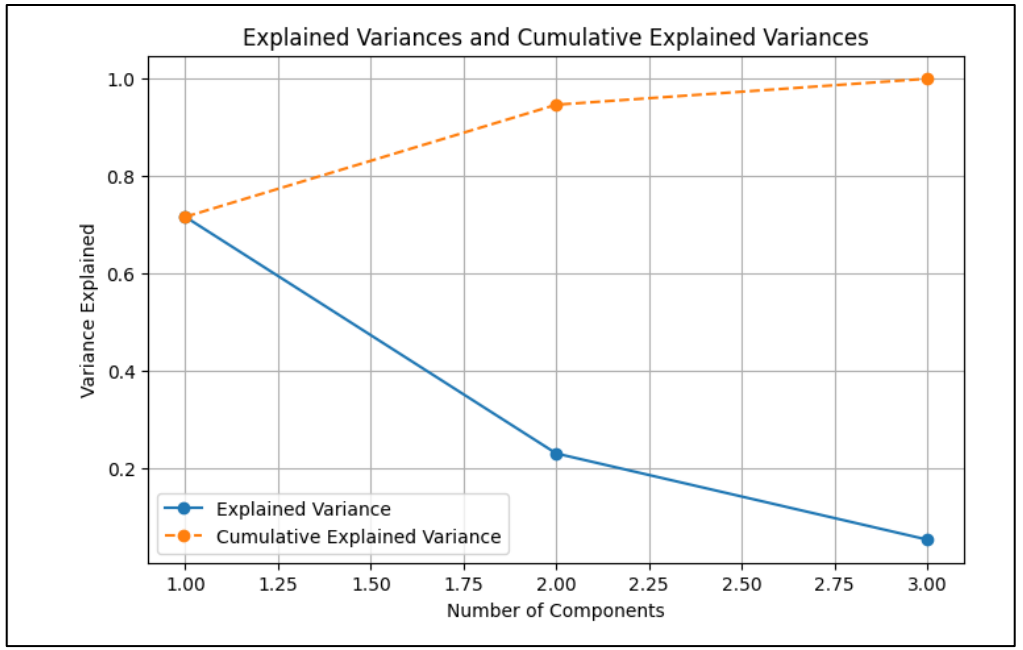
Implementing PCA

```
fig = px.scatter_matrix(
    X_pca,
    dimensions=range(3),
    color=dataset["Class"]
)
fig.update_traces(diagonal_visible=False)
fig.show()
```



Implementing PCA

```
explained_variances = pca.explained_variance_ratio_
cumulative_variances = np.cumsum(explained_variances)

# Plot explained variances and cumulative variances
plt.figure(figsize=(8, 5))
plt.plot(range(1, len(explained_variances) + 1),
         explained_variances, marker='o', linestyle='-', label='Explained Variance')
plt.plot(range(1, len(cumulative_variances) + 1),
         cumulative_variances, marker='o', linestyle='--',
         label='Cumulative Explained Variance')
plt.xlabel('Number of Components')
plt.ylabel('Variance Explained')
plt.title('Explained Variances and Cumulative Explained Variances')
plt.legend()
plt.grid(True)
plt.show()
```



Advantages of PCA

1. **Noise Reduction:** PCA can be effective in reducing noise and capturing the most important variance in the data, making it useful when dealing with noisy datasets.
2. **Linearity:** PCA is a linear technique, which can be an advantage when data relationships are predominantly linear.
3. **Computational Efficiency:** PCA is computationally efficient and can handle large datasets and a high number of dimensions, making it a practical choice for high-dimensional data.
4. **Variance Retention:** PCA allows you to retain a specified portion of the total variance in the data, giving you control over how much information is preserved in the reduced feature space.

Challenges and Limitations

1. **Linearity Assumption:** PCA assumes that the underlying data relationships are linear. If the data exhibits complex nonlinear structures the PCA is not a suitable technique.
2. **Information Loss:** PCA discards some information when reducing dimensionality. The trade-off between dimensionality reduction and information preservation should be carefully considered.
3. **Sensitive to Scaling:** PCA is sensitive to the scaling of features, so it's important to standardize or normalize data before applying PCA to avoid the impact of feature scale on the results.
4. **Assumption of Orthogonality:** PCA assumes that the principal components are orthogonal (uncorrelated). This assumption may not hold in some datasets.

Conclusion

- In conclusion, Principal Component Analysis, or PCA, is a powerful and widely-used technique for dimensionality reduction, data preprocessing, and feature extraction.
- In machine learning projects, PCA can be a valuable tool for enhancing data quality, reducing computation complexity, and improving the interpretability of your results.

References

[1] Wikipedia contributors. (2023, July 2). Scree plot. In *Wikipedia, The Free Encyclopedia*. Retrieved 19:32, November 9, 2023, from https://en.wikipedia.org/w/index.php?title=Scree_plot&oldid=1163045172

[2] Z. Jaadi, "A Step by Step Explanation of Principal Component Analysis," Built In, Sep. 04, 2019. https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[3] "Principal Component Analysis (PCA)," www.youtube.com. https://www.youtube.com/watch?v=FD4DeN81ODY&t=219s (accessed Nov. 09, 2023).

[4] "Principal Component Analysis (PCA)," www.youtube.com. https://www.youtube.com/watch?v=g-Hb26agBFg&t=272s (accessed Nov. 09, 2023).

[5] B. Hall, "An Intuitive Approach to PCA," The Startup, Jun. 23, 2020. https://medium.com/swlh/an-intuitive-approach-to-pca-fc4d05c14c19 (accessed Nov. 09, 2023).

[6] "| notebook.community," notebook.community. https://notebook.community/hershaw/data-science-101/course/class1/pca/iris/PCA%20-%20Iris%20dataset (accessed Nov. 09, 2023).

[7] "Eigenvalues and Eigenvectors," lpsa.swarthmore.edu. https://lpsa.swarthmore.edu/MtrxVibe/EigMat/MatrixEigen.html

Code Reference

PCA on Iris Dataset

Questions