Insights In Wheat Seed And Irish Flower Data Through PCA

Amit Raj Pant* and Arahanta Pokhrel*
*Insitute of Engineering, Thapathali Campus

ABSTRACT This study focuses on the application of Principal Component Analysis (PCA) to the Wheat Seed Dataset and Irish flower data, aiming to unravel valuable insights hidden within the dataset's features. It suggests that the study explores the transformation of raw data obtained from wheat fields into meaningful features through the utilization of PCA. By employing PCA, the study aims to analyze and extract valuable insights from the dataset, enabling a deeper understanding of the characteristics and patterns present in wheat seeds. Plots and graphs were generated to visualize the findings. The report discusses the theoretical background, mathematical formulas, and instrumentation, and presents the insights obtained from the data analysis.
INDEX TERMS $--Principal Component Analysis, dimensionality reduction, visualize$

I. Introduction

In the real world, data often consists of a vast number of attributes or features, which may include both relevant and irrelevant information. Sometimes, these attributes are unnecessary and fail to provide significant insights or valuable information. In such cases, it becomes crucial to extract meaningful data from the raw dataset through a process known as data mining.One common approach to extracting relevant information from high-dimensional datasets is dimensionality reduction. The curse of dimensionality refers to the challenges and limitations that arise when working with high-dimensional data. As the number of attributes increases, the data becomes sparser, and it becomes more difficult to analyze and interpret effectively.

Principal Component Analysis (PCA) is an effective technique for overcoming the curse of dimensionality.The curse of dimensionality is a collection of phenomena that state as dimensionality increases, the manageability and effectiveness of data tend to decrease. On a high level, the curse of dimensionality is related to the fact that as dimensions (variables/features) are added to a data set, the average and minimum distance between points (records/observations) increase. PCA enables analysts to reduce the number of dimensions in a dataset while preserving the essential characteristics and variability of the data. By transforming the original high-dimensional space into a lower-dimensional space, PCA aims to capture the maximum amount of information with the minimum number of dimensions.It involves identifying the principal components, which are the directions in the feature space along which the data exhibits the highest variance. These principal components are orthogonal to each other, meaning they are uncorrelated. The first principal component captures the most significant amount of variance, followed by the second, third, and so on.

II. Methodology

A. Breif Theory and Working Pipeline

The primary objective of this lab experiment was to develop a custom implementation of Principal Component Analysis (PCA) and compare it with the PCA implementation provided by the scikit-learn library.PCA is a technique that transforms data into a lower-dimensional space while maximizing the preservation of important information. By leveraging the covariance matrix, PCA identifies new dimensions that are orthogonal and ordered according to their variance, with the first component capturing the most significant variance.The analysis was conducted on three different data-sets that are a randomly generated dataset, the Iris dataset, and the Wheat Seed Prediction dataset. By performing PCA on these datasets, we aimed to identify the principal components and assess the performance of the custom implementation in comparison to scikit-learn's implementation.

The methodology performed on the random dataset involved several steps. Initially, a pseudo-random number generator was initialized with a random seed for reproducibility. Sampling was done from a normal distribution, resulting in an array of 20 values, which were then plotted to visualize the distribution. A 2x2 matrix was created and filled with random values between 0 and 1 .The use of the 2x2 matrix as a transformation matrix provides a powerful tool for modifying and manipulating data.We use it to manipulate and analyze data by applying linear transformations.Applying linear transformations through matrix multiplication, it allows for a range of operations such as stretching, rotating, and scaling.Then,the sampled values were multiplied by the random matrix, and the resulting

values were plotted. Variance was computed along the data axes, and the covariance matrix was calculated to describe relationships between variables. Eigen decomposition of the covariance matrix yielded eigenvalues and eigenvectors, enabling a change of basis. The proportion of variance explained by both eigenvectors was determined, and the data values were plotted in the new basis.

The Iris dataset [2] is a widely used benchmark dataset in the field of machine learning and data analysis. It consists of measurements of iris flowers belonging to three different species: setosa, versicolor, and virginica. The dataset contains a total of 150 samples, with 50 samples for each iris species. For each sample, four features are recorded which are sepal length, sepal width, petal length, and petal width. The methodology for performing Principal Component Analysis (PCA) on the Iris dataset involves several steps. First, the dataset is loaded and prepared by extracting the feature values and adding the target variable. The data is then standardized to ensure uniform scaling. The covariance matrix and the corresponding eigenvalues and eigenvectors are computed to understand the relationships and directions of the principal components. Proportions of explained variance are calculated to assess the information captured by each component. The desired number of principal components is selected, and PCA is applied to transform the data into a lower-dimensional space. The results are visualized using scatter plots. Additionally, scikit-learn's PCA functionality is utilized, providing convenient methods for fitting the model, transforming the data, and accessing the explained variance ratio and principal components.

The Wheat Seed Prediction dataset [1] is a well-known dataset used for classification tasks in machine learning. It contains measurements of various attributes of wheat seeds, along with their corresponding wheat seed types. The dataset consists of 210 samples, with each sample representing a wheat seed. For each seed, seven numerical attributes are recorded, including area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient, and length of kernel groove. These attributes provide information about the physical characteristics and shape of the seeds. The wheat seed types are classified into three categories: Kama, Rosa, and Canadian.

The methodology for performing Principal Component Analysis (PCA) on the Seed dataset involves importing necessary libraries and defining visualization functions. The dataset is read into a pandas DataFrame and prepared by standardizing the feature values. The covariance matrix is computed, and eigenvalues with corresponding eigenvectors are obtained. The eigenvectors are sorted, and a chosen number of principal components is used to transform the data to a lower-dimensional space. The transformed data is visualized in 2D and 3D using custom-defined functions. Furthermore, scikit-learn's PCA functionality is employed, fitting PCA models to the standardized data and transforming it. The explained variance ratio and principal components can be accessed from the models. The transformed data is again visualized in 2D and 3D using the scikit-learn PCA results. By following this methodology, PCA is applied to the Seed dataset, allowing for understanding patterns, reducing dimensionality, and visualizing the data in a lower-dimensional space using both custom

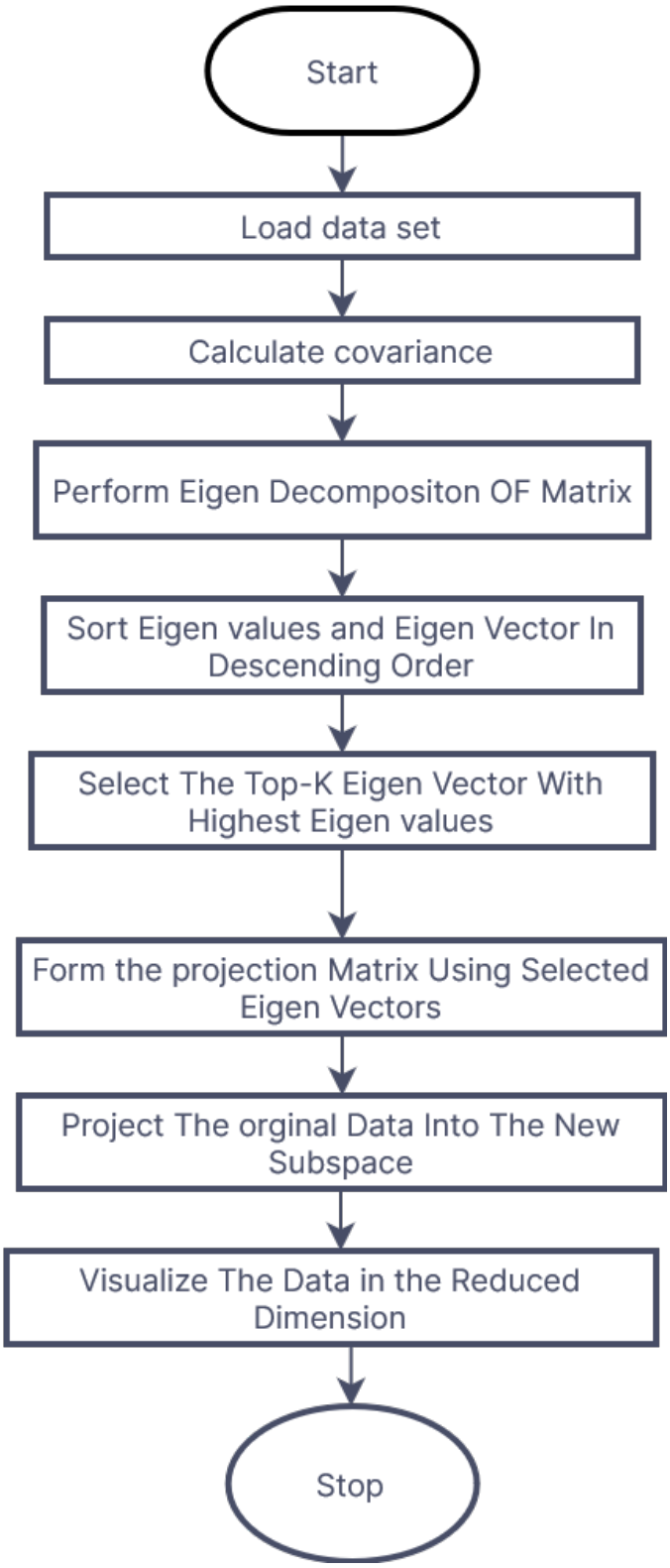

Fig. 1. System Block Diagram

implementation and scikit-learn's PCA capabilities.

B. Major mathematical Formula

Principal Component Analysis (PCA) involves several important mathematical formulas. Here are the key formulas used in PCA analysis:

1) Normalization of the Data: In order to prevent any individual variable from overpowering the analysis due to its larger scale, it is necessary to standardize the data before applying PCA. This step ensures that the variables which may be measured on different scales, are brought to a common scale.

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

Where, μ is the mean of independent features, σ is the standard deviation.

2) Covariance Matrix: The covariance matrix is calculated to measure the relationships between variables in the dataset. Given a dataset with n variables, the covariance matrix is represented as CV and is computed as:

$$CV = \frac{(X - \mu)(X - \mu)^T}{(n - 1)} \tag{1}$$

where X is the centered data matrix and μ is the mean vector.

3) Eigenvalue and Eigenvector: PCA aims to find the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors represent the principal components, while the corresponding eigenvalues indicate the amount of variance explained by each principal component.finding the roots of below equation find eigen values.And for each of the eigen values there will be eigen vector.

$$|A - \lambda V| \tag{2}$$

4) Change Of Biasis: PCA involves projecting the original data onto the new coordinate system defined by the principal components

$$PX = Y \tag{3}$$

where X is original record, Y is rerepresentation of dataset,P is a matrix that transform X into Y

C. Instrumentation

The experiment was conducted using the Python programming language, which provided a versatile and efficient platform for data analysis. The NumPy library, known for its powerful numerical computation capabilities, was employed for matrix operations and mathematical calculations. The scikit-learn library, renowned for its comprehensive machine-learning functionalities, was utilized for the pre-existing PCA implementation and importing datasets.Matplotlib is used for visually representing the data,It provided a range of tools and functions to create informative and visually appealing plots, charts, and graphs, aiding in the effective representation of the findings. pandas for data manipulation and analysis. It enabled us to efficiently handle the datasets and extract meaningful insights. The combination of these instrumental components, along with the use of the random function to create sample datasets, ensured a robust and reliable analysis. The Python programming language, along with NumPy, matplotlib, pandas, and scikit-learn, provided a powerful and well-defined framework for the experiment, enabling efficient data handling, visualization, and exploration, while ensuring the accuracy and reliability of the results.

III. Results

A. Random Dataset

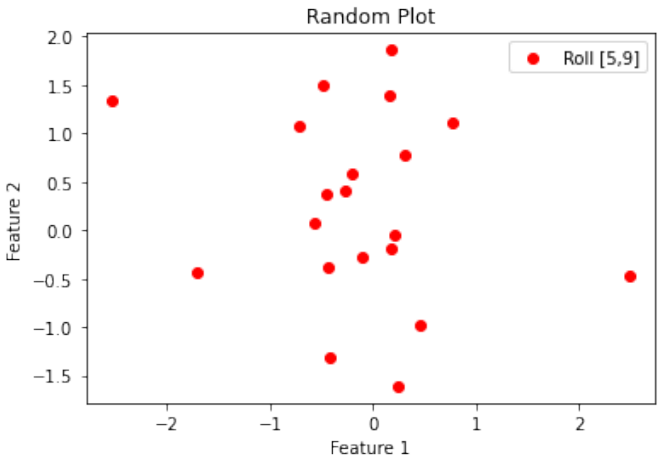

Fig. 2. 20 Random generated numbers

The plot depicted in Figure 2 displays a collection of 20 sample points that were randomly selected from a standard distribution. They are spread randomly across all the directions so PCA cannot be applied to it.

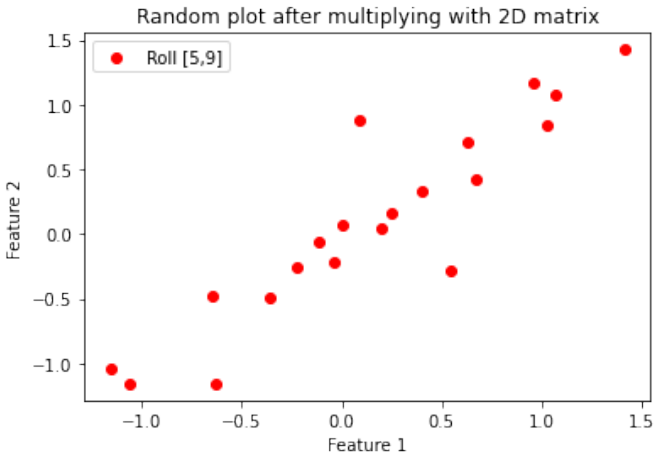

Fig. 3. 20 Random generated numbers after 2D transformation

Figure 3 visualizes the alignment of the original data points when they undergo a transformation using a randomly selected

2D matrix from a uniform distribution. The transformation resulted in the data points being aligned in a specific direction.

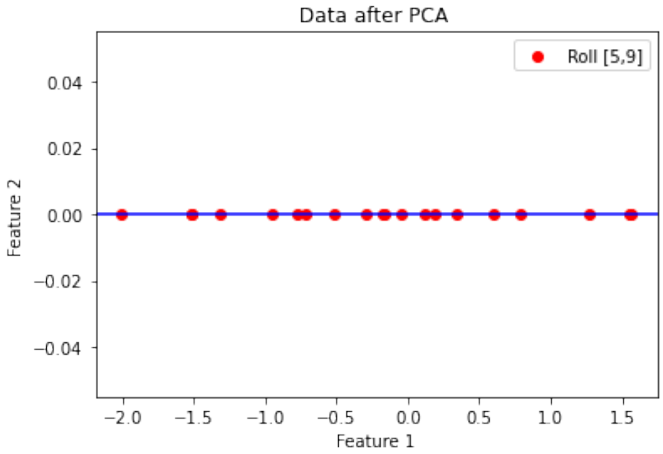

Fig. 4. 20 Random generated numbers after PCA

Figure 4 is the plot after applying principal component analysis. All the data points are now aligned in one horizontal line as only one principal component was selected to transform the data.

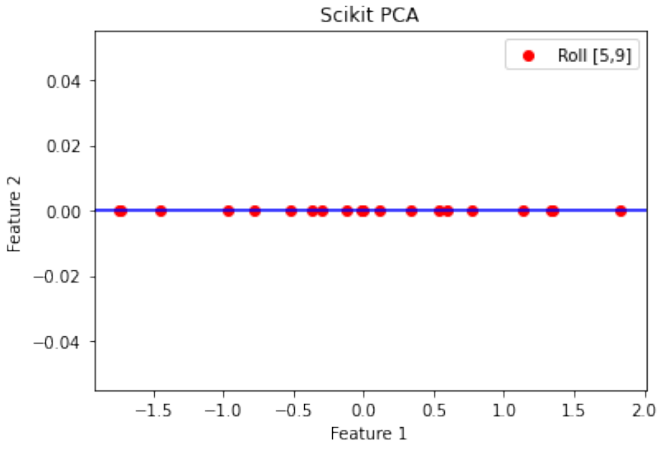

Fig. 5. 20 Random generated numbers after PCA using scikit

Figure 5 is a visual representation of the outcomes obtained through the application of Principal Component Analysis (PCA) using sci-kit-learn library.

B. Iris flower data set

Similar to the random dataset, PCA was applied to Iris flower dataset by taking two principal components, the results obtained can be observed in the 7. It reveals a clear separation of the data into three clusters. This implies that the first two principal components effectively captured the underlying patterns and variances present in the dataset, enabling us to discern meaningful clusters within the Iris flower data.

Fig. 6. Data visualization of Irish Dataset

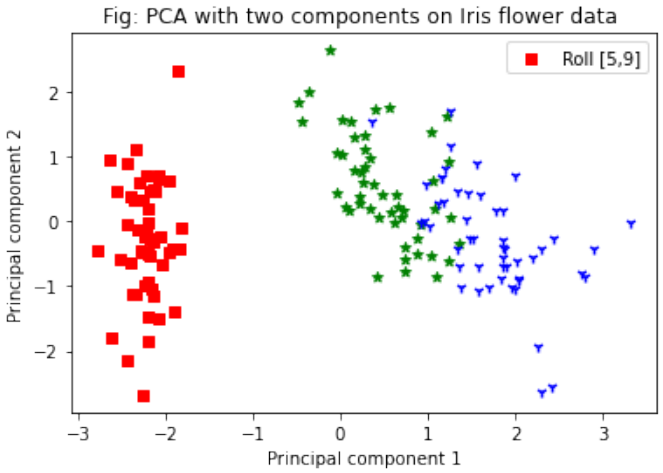

Fig. 7. PCA on iris dataset

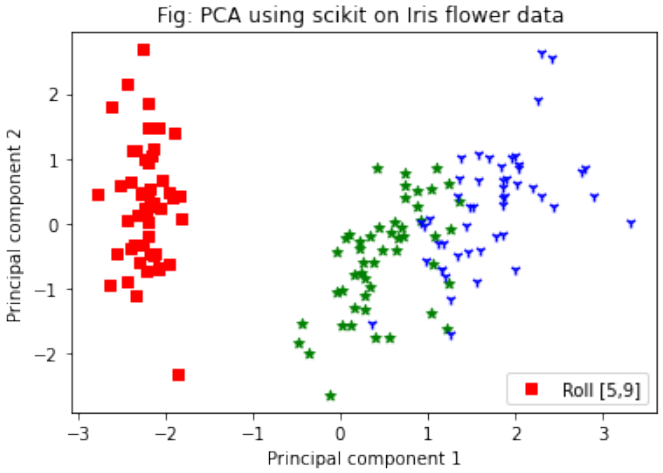

Fig. 8. PCA on iris dataset using scikit

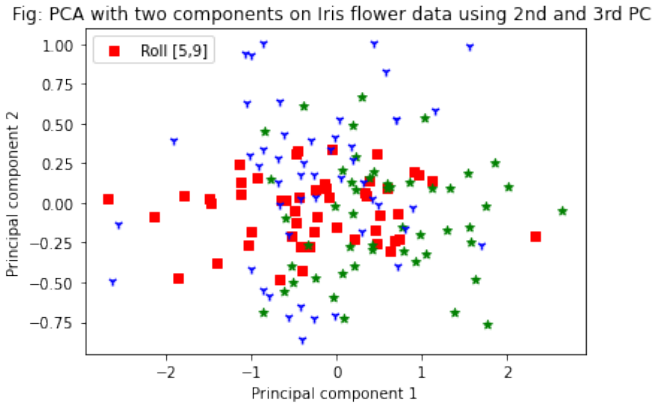

Fig. 9. PCA on Iris flower data using 2nd and 3rd PC

Figure 8 is the plot of results obtained by using the sci-kit learn library on the iris dataset. The plot slightly varies from the results we obtained through our own implementation of PCA.

Figure 9 is the plot PCA with two components on Iris flower data using 2nd and 3rd highest eigen values.

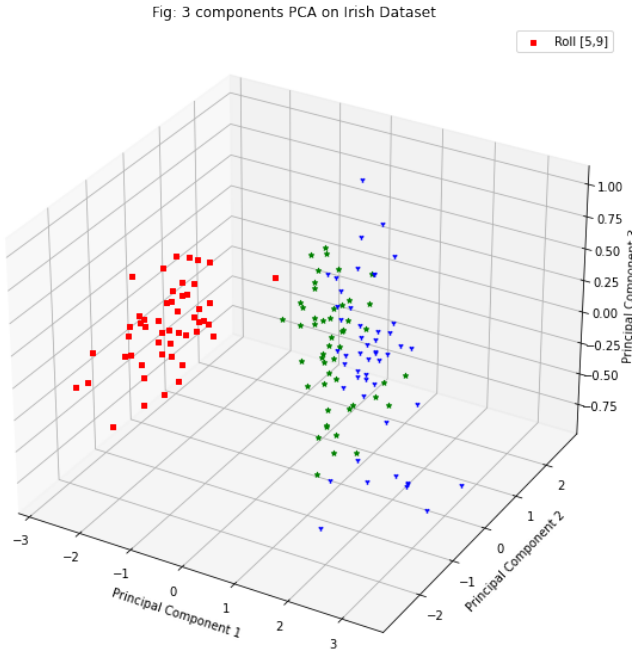

Fig. 10. PCA on Iris flower data using 3 components

Figure 10 is the plot PCA with three components on Iris flower data using 1st,2nd and 3rd highest eigen values.

C. Seed data set

When 2 principal components with proportions 71% and 17% were used to change the basis, the plot in 12 was obtained. These two components captured the majority of the

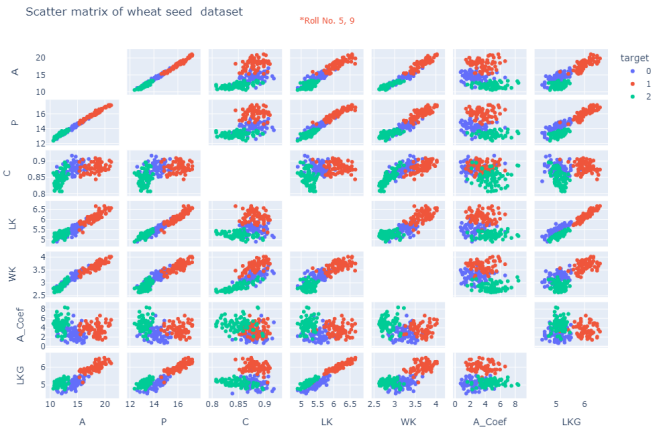

Fig. 11. Dataset visualization of seed Dataset

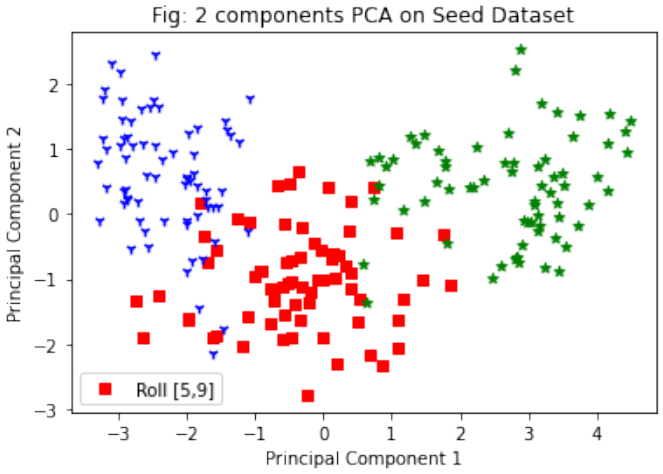

Fig. 12. 2D PCA on seed dataset

data so a clustered pattern in the results can be observed.

In figure 13, we can observe the outcomes of performing Principal Component Analysis (PCA) by considering the first three components. Interestingly, both the 3D and 2D plots demonstrate similar information gain, indicating that the third component may not contribute significantly to capturing the essence of the data. Consequently, we can draw the conclusion that the first two components alone encompass a vast majority of the information contained within the dataset.

In figure 14, we can observe the outcomes of performing Principal Component Analysis (PCA) using the sci-kit learn library by considering the first three components.

Figure 15 is the plot of results obtained by using the sci-kit learn library on the wheat seed dataset. The plot slightly varies from the results we obtained through our own implementation of PCA.

Figure 16 is the two-component PCA on Seed data using the 3rd and 4th highest eigenvalues. The data points aren't distinct

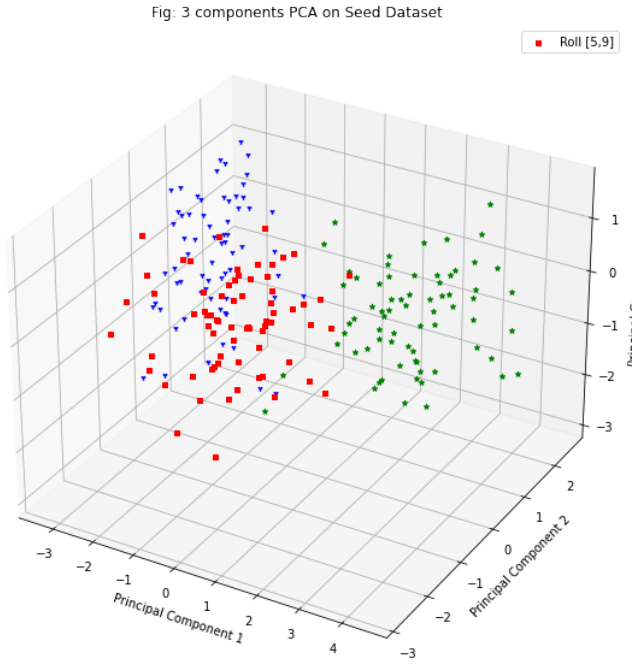

Fig. 13. 3D PCA on seed dataset

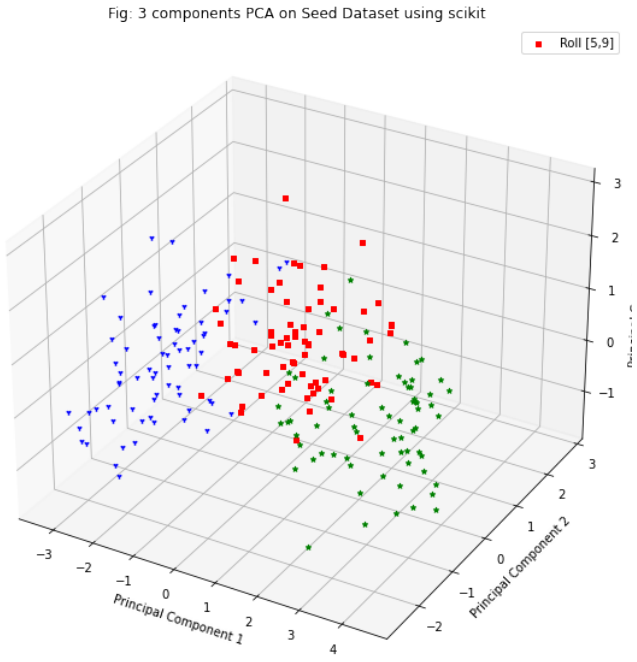

Fig. 14. 3D PCA using the sci-kit learn on seed dataset

and are randomly distributed with no insights into data even after applying PCA.

IV. Discussion & Analysis

In our analysis of the randomly selected dataset consisting of 20 numbers, when the data points were transformed by a

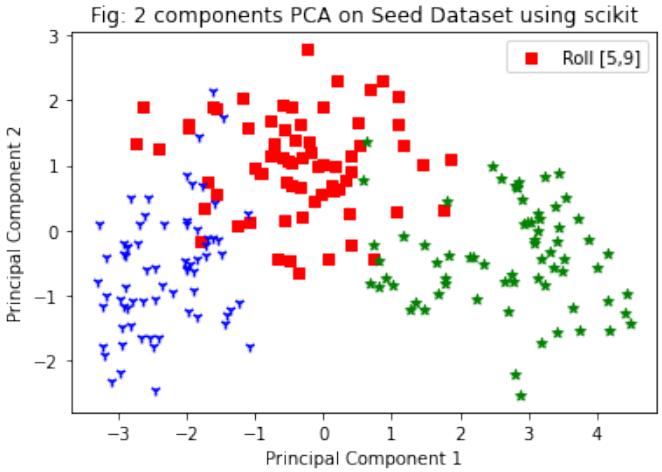

Fig. 15. 2 component PCA on seed dataset using scikit

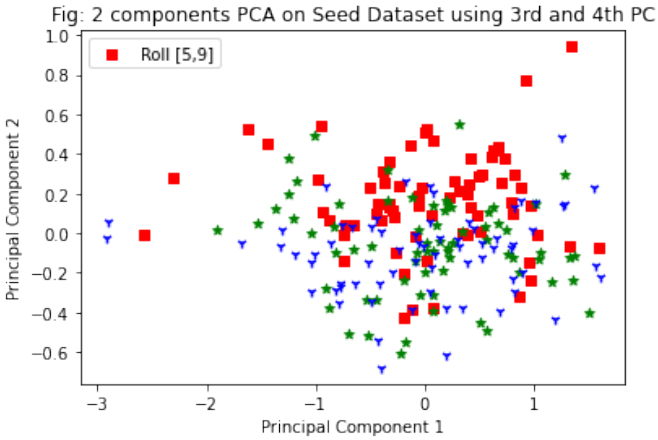

Fig. 16. PCA using 3rd and 4th component on seed dataset

2D matrix drawn from uniform distribution they got aligned in a particular direction. This alignment can be attributed to the initial data points, which were generated based on a standard deviation centered around the mean (i.e., 0), forming a circular arrangement. A 2D matrix that has all the elements positive can be attributed to having shearing effects and when this transformation is applied the circular space, caused the data space to undergo a shearing effect, ultimately aligning the points in a specific direction, in this case, from left to right.

In Irish dataset, after analyzing the different component utilization, we can conclude even with the utilization of only two principal components, the entire dataset can be accurately represented. The two principal components are able to accurately represent the entire dataset, which is also described by the resulting plots. The results on using three components are almost similar to the results obtained when we used 2 components. This is also reflected by the percentage of variance for every component.

Furthermore, when we extended our analysis to the seed

dataset, we made an interesting observation regarding the utilization of three principal components instead of the traditional two. Surprisingly, the outcomes did not exhibit significant disparities. This finding aligns with the proportional distribution of eigenvalues, which indicates that the additional principal component does not contribute substantially to the separation of the data.

When employing two principal components, we can observe that the data points exhibit equal separation, forming distinct clusters. However, when we expanded the analysis to include three principal components, the clustering pattern remained relatively unchanged. The additional principal component did not introduce a substantial shift in the separation of the data points or result in a noticeably different clustering pattern.

This observation suggests that the inherent structure and variations present in the seed dataset can be sufficiently captured and represented by the first two principal components. The third principal component may not carry significant additional information that contributes to further separation or clustering of the data.

Also, one thing to note is there is a slight variation in the results obtained by our implementation of PCA and sci-kit learn implementation. These differences can occur due to factors such as numerical precision, differences in data preprocessing methods, and parameter settings.

Both in flower and seed dataset, when we explore the application of Principal Component Analysis (PCA) using principal components other than the first and second, the resulting data exhibits a mixed-up or differently clustered pattern, deviating from the anticipated arrangement. The primary reason behind this can be attributed to the limitations of the third and fourth principal components in effectively capturing the significant variations present within the data. These particular components lack the capacity required to accurately represent the underlying structure of the dataset.

As a result, when these less informative components are considered in the analysis, they can introduce noise or incorporate unrelated information. This, in turn, leads to the mixing of data points and an alteration of the clustering pattern that would have been observed if we had solely focused on the first and second principal components. So, it can be concluded that in our seed dataset, the first and second principal components are enough to capture and represent the most substantial variations in the data, thus providing a more reliable representation of the dataset's structure.

V. Conclusion

In nutshell, We successfully explored and applied PCA on different datasets, including random data, the Iris dataset, and the wheat seed prediction dataset. By implementing PCA from scratch and comparing it with the scikit-learn PCA implementation, we gained valuable insights into dimensionality reduction and feature selection.The eigenvalues derived from the covariance matrices provided insight into the importance of each principal component. The comparison between the custom PCA implementation and scikit-learn PCA showed consistent results, validating their accuracy. Visualizations of the data in the new basis highlighted clear distinctions and patterns. Hence, PCA proved to be a robust technique for dimensionality reduction and data exploration, offering valuable insights for analysis and decision-making.

The lab objectives were met as we successfully performed PCA, implemented the necessary mathematical formulas using inbuilt function, and analyzed the results. The comparison between our custom PCA implementation and the scikit-learn PCA showcased the consistency and reliability of both approaches. Overall, this lab enhanced our understanding of PCA as a powerful tool for dimensionality reduction, data visualization, and feature extraction, enabling us to extract meaningful insights from complex datasets and make informed decisions based on the transformed data.

References

[1] Charytanowicz, M., Niewczas, J., Kulczycki, P., Kowalski, P., & Lukasik, S. (2012). *seeds*. UCI Machine Learning Repository. Retrieved from *DOI: https://doi.org/10.24432/C5H30K10.24432/C5H30K*

[2] Fisher, R. A. (1988). *Iris*. UCI Machine Learning Repository. Retrieved from *DOI: https://doi.org/10.24432/C56C7610.24432/C56C76*

Amit Raj Pant is a dedicated student pursuing his studies in the Department of Electronics and Computers at Thapathali Engineering Campus, Tribhuvan University, located in Kathmandu, Nepal. With a strong passion for technology. His interest includes computer vision, and machine learning on resource-constrained edge devices, which involves performing computational tasks on local devices rather than relying solely on remote servers.

Arahanta Pokhrel born in 1999 in Biratnagar, Nepal, is a dedicated individual with a strong passion for learning and research. Currently pursuing a Bachelor's degree in Computer Technology at the Institute of Engineering, Thapathali Campus, He is in the final year of his studies.Throughout His academic journey, He has developed a keen interest in machine learning and data science.Additionally, He has a curious mind and actively engages in quizzes and current affairs to stay updated with the latest information and committed to acquiring new skills.

VI. Appendix

The code was written using Jupyter Notebook to implement PCA. The snippets are attached below.

A. Random Dataset

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

# Function to plot the data
def visualize_data(x, y, title='Random Plot', target=[0], label=['Feature 1', 'Feature 2']):
    colormap = np.array(['r', 'g', 'b'])
    plt.scatter(x, y, c=colormap[target])
    plt.title(title)
    plt.xlabel(label[0])
    plt.ylabel(label[1])
    plt.axhline(0, color='b')
    # Add a legend with roll numbers
    plt.legend([f'Roll [5,9]'])
    plt.show()

# Returns the covariance and eigen values and vectors.
def calculate_cov_eig(new_x):
    cov_mat = np.cov(new_x.T)
    eigenvalues, eigenvectors = np.linalg.eig(cov_mat)
    print(f'Covariance Matrix: {cov_mat}\nEigen Values: {eigenvalues}\nEigen Vectors: {eigenvectors}')
    return cov_mat, eigenvalues, eigenvectors

# Implement PCA
def apply_pca(eigen_row, x):
    pca_data = np.matmul(eigen_row, x.T)
    pca_data = pca_data.T
    return pca_data

x1 = np.random.randn(20)
x2 = np.random.randn(20)

visualize_data(x1, x2)

d = np.random.rand(2, 2)
x = [x1, x2]
x = np.array(x).T
print(x)

new_x = np.matmul(x, d)
visualize_data(new_x[:, 0], new_x[:, 1], 'Random plot after multiplying with 2D matrix')

cov, eigenvalues, eigenvectors = calculate_cov_eig(new_x)

data = apply_pca(eigenvectors[:, 1], new_x)

visualize_data(data, np.zeros_like(data), 'Data after PCA')

# use of skleannr
from sklearn.decomposition import PCA

# Assume you have a dataset called `data` with dimensions (m, n)

# Create an instance of PCA with the desired number of components
n_components = 1  # Specify the number of principal components you want to retain
pca = PCA(n_components=n_components)

# Fit the PCA model to the data
pca.fit(new_x)

# Transform the data to the lower-dimensional space
transformed_data = pca.transform(new_x)

# Access the explained variance ratio
visualize_data(transformed_data, np.zeros_like(transformed_data), 'Scikit PCA')
```

B. Irish Flower Dataset

```
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Visualize the data
def visualize_data(x, y, title='Random Plot', target=[0], label=['Principal component 1', 'Principal component 2']):
    colormap = np.array(['r', 'g', 'b'])
    plt.scatter(x, y, c=colormap[target])
    plt.title(title)
    plt.xlabel(label[0])
    plt.ylabel(label[1])
    plt.legend([f'Roll [5,9]'])
    plt.show()

# Calculate covariance matrix and eigen values and eigen vectors
def cal_cov_eig(new_x):
    cov_mat = np.cov(new_x.T)
    eigenvalues, eigenvectors = np.linalg.eig(cov_mat)
    print(f'Covariance Matrix:\n{cov_mat}\nEigen Values:\n{eigenvalues}\nEigen Vectors:\n{eigenvectors}')
    return cov_mat, eigenvalues, eigenvectors

# Implement PCA
def apply_pca(eigen_row, x):
    pca_data = np.matmul(eigen_row, x.T)
    pca_data = pca_data.T
    return pca_data

# Calculate the proportion of each eigen value
def cal_prop(e):
    prop = []
    sum = np.sum(e)
    for i in range(len(e)):
        prop.append((e[i] / sum) * 100)
    return prop
```

```

# Return row-wise eigen vector
def sort_evect(eval, evect):
    pair = dict()
    for i in range(len(eval)):
        pair[np.abs(eval[i])] = evect[:, i]
    sorted_pair = dict(sorted(pair.items(),
        reverse=True))
    evect_sorted = np.array(list(sorted_pair.
        values()))
    return evect_sorted

# Load the dataset
from sklearn.datasets import load_iris
ds = load_iris()
df = pd.DataFrame(data=ds.data, columns=ds.
    feature_names)
df['target'] = pd.Series(ds.target)

x = df.iloc[:, 0:4].values
x_std = StandardScaler().fit_transform(x)

cov_mat, eigenvalues, eigenvectors =
    cal_cov_eig(x_std)
p = cal_prop(eigenvalues)

# Number of principal components
k = 2
evect_sorted = sort_evect(eigenvalues,
    eigenvectors)
output = apply_pca(evect_sorted[:k, :], x_std)

visualize_data(output[:, 0], output[:, 1], '
    Fig: PCA with two components on Iris
    flower data', df.iloc[:, 4])

# Using scikit-learn's PCA
from sklearn.decomposition import PCA

n_components = 2  # Specify the number of
    principal components you want to retain
pca = PCA(n_components=n_components)

pca.fit(x_std)
transformed_data = pca.transform(x_std)
explained_variance_ratio = pca.
    explained_variance_ratio_
principal_components = pca.components_

visualize_data(transformed_data[:, 0],
    transformed_data[:, 1], 'Fig: PCA using
    scikit on Iris flower data', df.iloc[:,
    4])

# PCA using the 2nd and 3rd largest
    eigenvalues
evect_sorted = sort_evect(eigenvalues,
    eigenvectors)
output = apply_pca(evect_sorted[1:3, :], x_std
    )
visualize_data(output[:, 0], output[:, 1], '
    Fig: PCA with two components on Iris
    flower data using 2nd and 3rd PC', df.iloc
    [:, 4])
```

C. Wheat Seed Dataset

```
import numpy as np
from sklearn.preprocessing import
    StandardScaler
import matplotlib.pyplot as plt
import pandas as pd

# Visualizing 3D PCA on Seed Dataset
def plot_3d(output, title='Fig: 3 components
    PCA on Seed Dataset'):
    fig = plt.figure(figsize=(12, 10))
    ax = fig.add_subplot(111, projection='3d')
    colormap = np.array(['r', 'g', 'b'])
    ax.scatter(
        output[:, 0],  # X coordinates of
            principal components
        output[:, 1],  # Y coordinates of
            principal components
        output[:, 2],  # Z coordinates of
            principal components
        c=colormap[df.iloc[:, 7]]
    )
    plt.title(title)
    plt.legend([f'Roll [5,9]'])
    ax.set_xlabel('Principal Component 1')
    ax.set_ylabel('Principal Component 2')
    ax.set_zlabel('Principal Component 3')
    plt.show()

# Visualizing 2D PCA on Seed Dataset
def plot_2d(output, title='Fig: 2 components
    PCA on Seed Dataset'):
    colormap = np.array(['r', 'g', 'b'])
    plt.scatter(
        output[:, 0],  # X coordinates of
            principal components
        output[:, 1],  # Y coordinates of
            principal components
        c=colormap[df.iloc[:, 7]]
    )
    plt.title(title)
    plt.legend([f'Roll [5,9]'])
    plt.xlabel('Principal Component 1')
    plt.ylabel('Principal Component 2')
    plt.show()

# Sorting eigen vector based on eigen value
def sort_evect(eval, evect):
    pair = dict()
    for i in range(len(eval)):
        pair[np.abs(eval[i])] = evect[:, i]
    sorted_pair = dict(sorted(pair.items(),
        reverse=True))
    evect_sorted = np.array(list(sorted_pair.
        values()))
    return evect_sorted

# Calculating proportion of eigen value
def cal_prop(e):
    prop = []
    sum = np.sum(e)
    for i in range(len(e)):
        prop.append((e[i] / sum) * 100)
    return prop


df = pd.read_csv('Seed_Data.csv')
```

```
data = df.iloc[:, :7].to_numpy()
data_std = StandardScaler().fit_transform(data
    )
covariance = np.cov(data_std.T)
eigen_value, eigen_vect = np.linalg.eig(
    covariance)
prop = cal_prop(eigen_value)
eigen_vect_sorted = sort_evect(eigen_value,
    eigen_vect)

k = 3  # Number of principal components

output_data = np.matmul(eigen_vect_sorted[:k -
     1, :], data_std.T)
output = output_data.T
output_3d = np.matmul(eigen_vect_sorted[:k,
    :], data_std.T)
output_3d = output_3d.T

plot_2d(output, title='Fig: 2 components PCA
    on Seed Dataset')
plot_3d(output_3d, title='Fig: 3 components
    PCA on Seed Dataset')

from sklearn.decomposition import PCA

pca2d = PCA(n_components=2)
pca3d = PCA(n_components=3)

pca2d.fit(data_std)
pca3d.fit(data_std)

data2d = pca2d.transform(data_std)
data3d = pca3d.transform(data_std)

plot_2d(data2d, title='Fig: 2 components PCA
    on Seed Dataset using scikit')
plot_3d(data3d, title ='Fig: 3 components PCA
    on Seed Dataset using scikit')

#PCA using 3 and 4 th larger eigen value
output_data = np.matmul(eigen_vect_sorted
    [2:4,:], data_std.T)
output_new= output_data.T
# output_3d = np.matmul(eigen_vect_sorted[:k
    ,:], data_std.T)
# output_3d = output_3d.T
plot_2d(output_new, title ='Fig: 2 components
    PCA on Seed Dataset using 3rd and 4th PC')
```