

# 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** – *PrincipalComponentAnalysis, dimensionalityreduction, 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

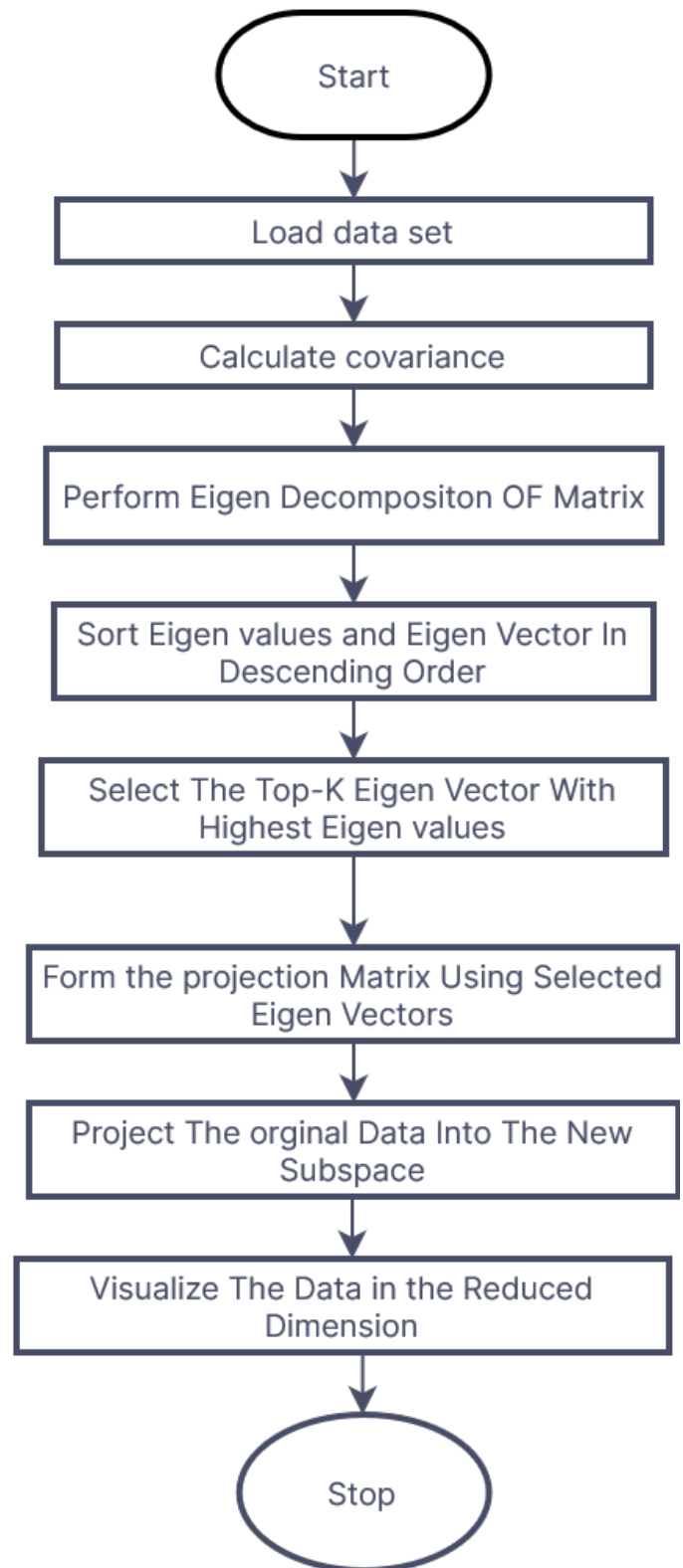


Fig. 1. System Block Diagram

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

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.

$$z = \frac{x - \mu}{\sigma} \quad (1)$$

Where,  $\mu$  is the mean of independent features,  $\sigma$  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:

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

where  $X$  is the centered data matrix and  $\mu$  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.

$$|A - \lambda V| \quad (2)$$

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

$$PX = Y \quad (3)$$

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

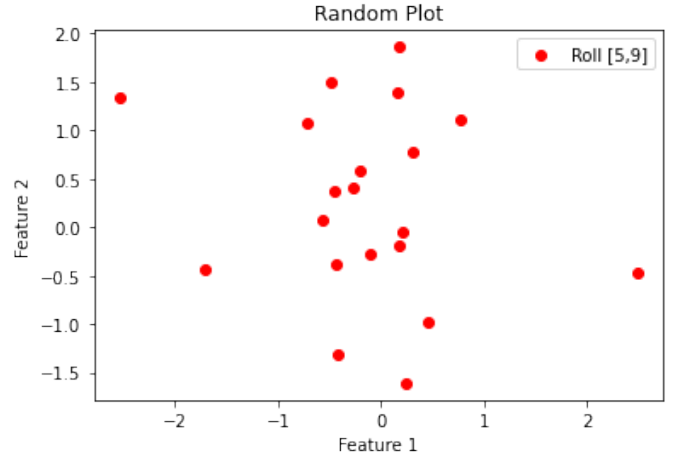


Fig. 2. 20 Random generated numbers

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.

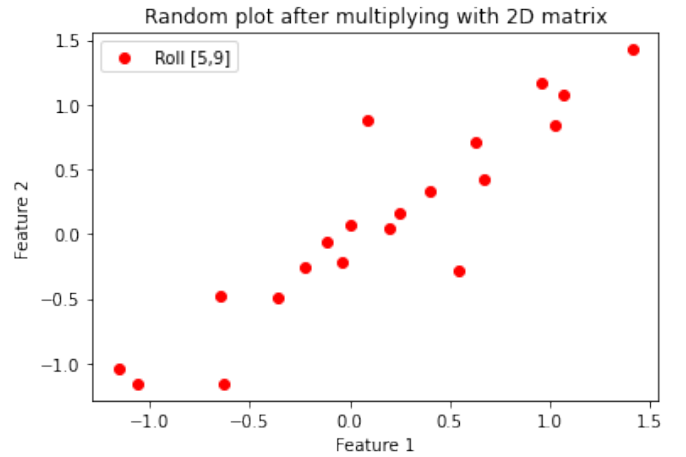


Fig. 3. 20 Random generated numbers after 2D transformation

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.

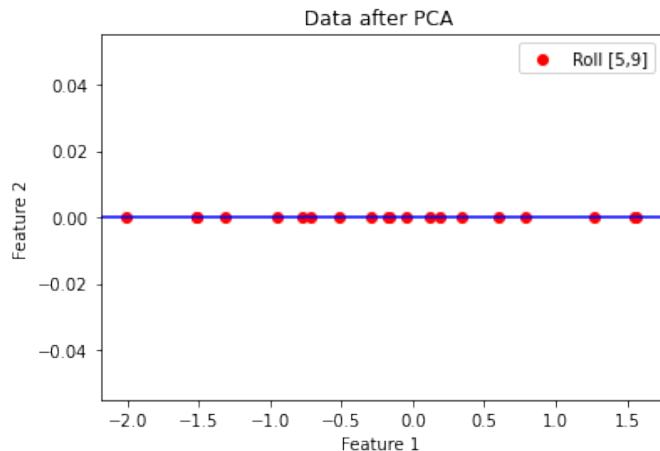


Fig. 4. 20 Random generated numbers after PCA

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.

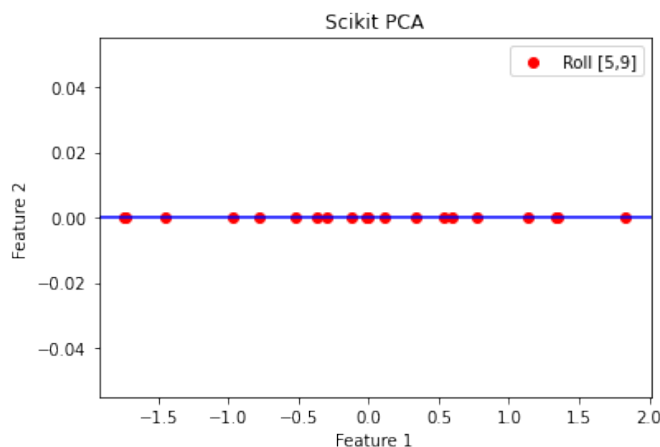


Fig. 5. 20 Random generated numbers after PCA using scikit

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

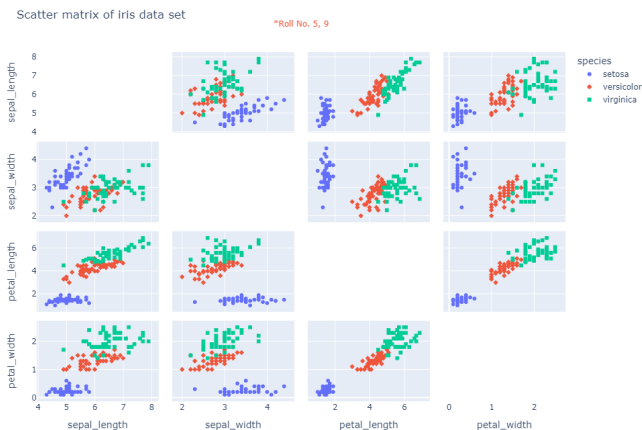


Fig. 6. Data visualization of Irish Dataset

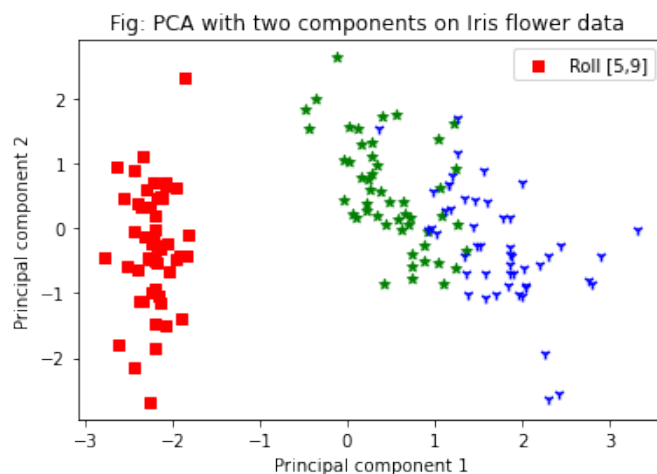


Fig. 7. PCA on iris dataset

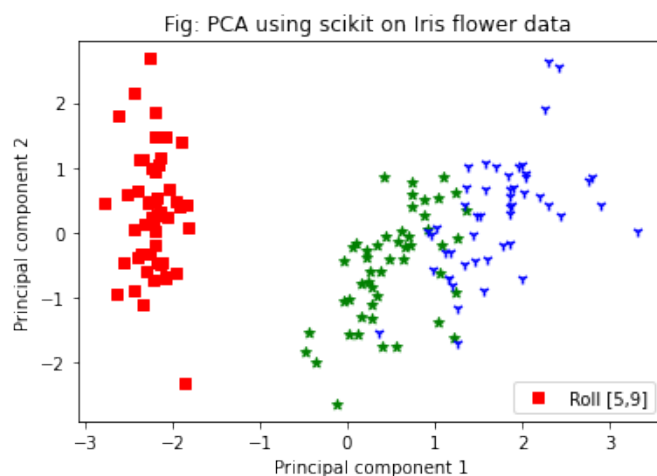


Fig. 8. PCA on iris dataset using scikit

discern meaningful clusters within the Iris flower data.

Fig: PCA with two components on Iris flower data using 2nd and 3rd PC

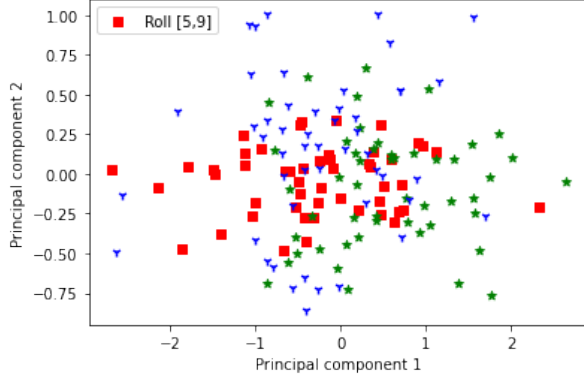


Fig. 9. PCA on Iris flower data using 2nd and 3rd PC

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: 3 components PCA on Irish Dataset

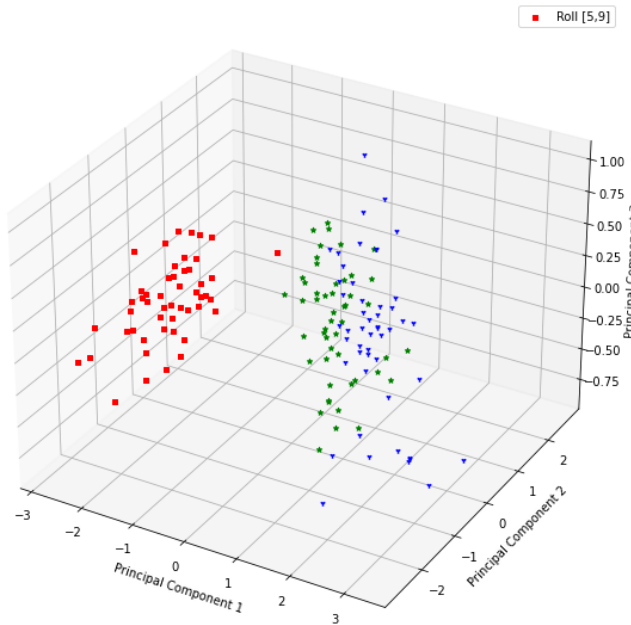


Fig. 10. PCA on Iris flower data using 3 components

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

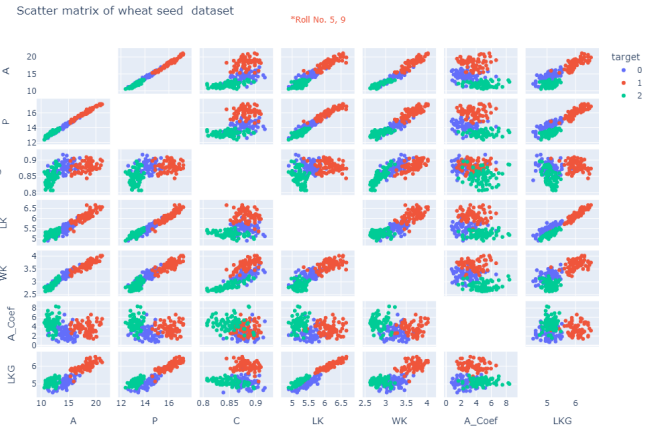


Fig. 11. Dataset visualization of seed Dataset

Fig: 2 components PCA on Seed Dataset

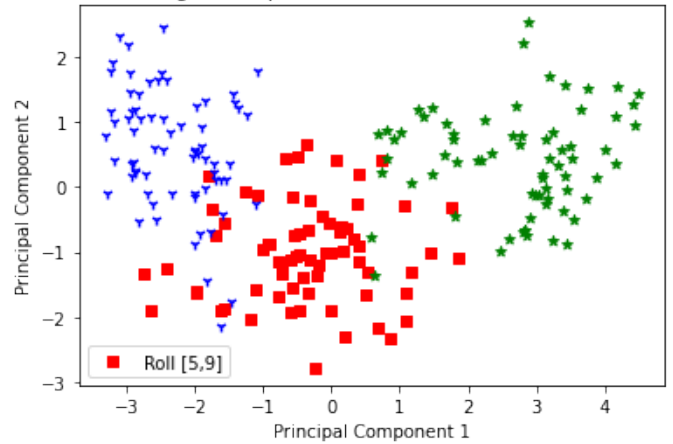


Fig. 12. 2D PCA on seed dataset

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: 3 components PCA on Seed Dataset

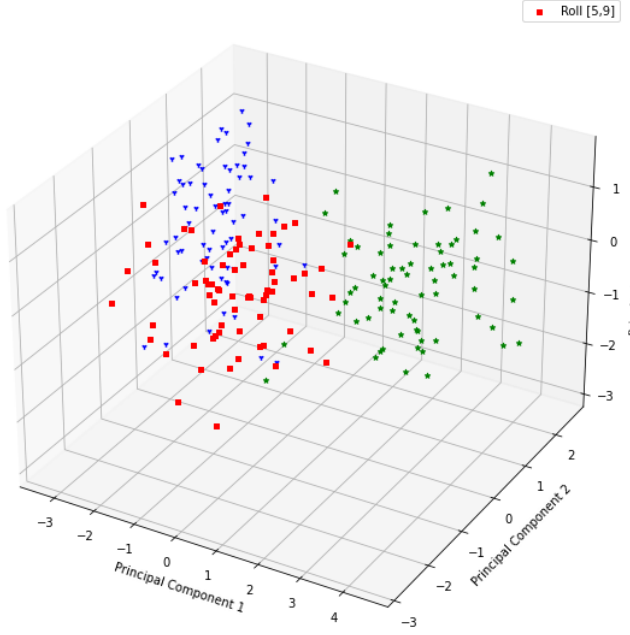


Fig. 13. 3D PCA on seed dataset

Fig: 3 components PCA on Seed Dataset using scikit

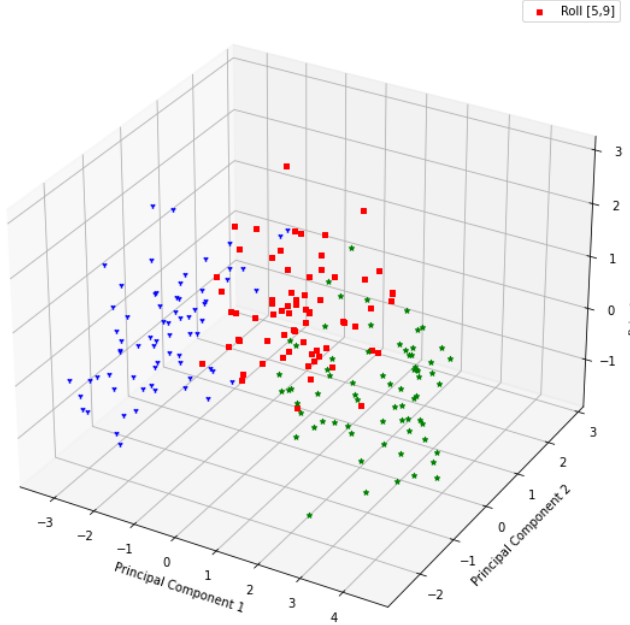


Fig. 14. 3D PCA using the sci-kit learn on seed dataset

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: 2 components PCA on Seed Dataset using scikit

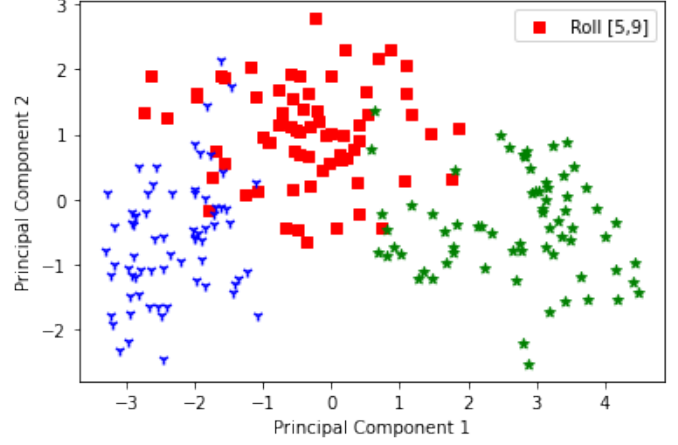


Fig. 15. 2 component PCA on seed dataset using scikit

Fig: 2 components PCA on Seed Dataset using 3rd and 4th PC

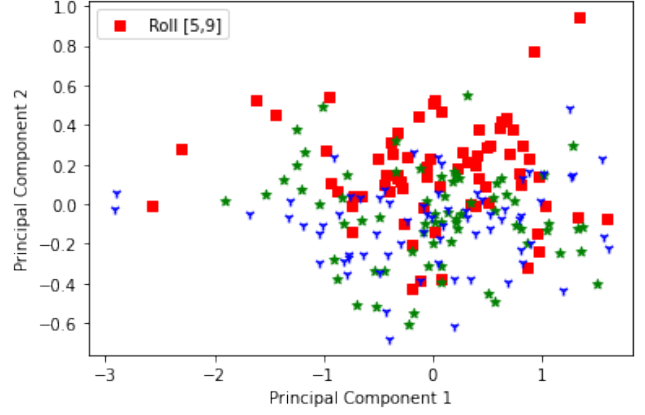


Fig. 16. PCA using 3rd and 4th component on seed dataset

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

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import random
4
5 # Function to plot the data
6 def visualize_data(x, y, title='Random Plot',
7 target=[0], label=['Feature 1', 'Feature 2
8 ']):
9     colormap = np.array(['r', 'g', 'b'])
10     plt.scatter(x, y, c=colormap[target])
11     plt.title(title)
12     plt.xlabel(label[0])
13     plt.ylabel(label[1])
14     plt.axhline(0, color='b')
15     # Add a legend with roll numbers
16     plt.legend([f'Roll [5,9]'])
17     plt.show()
18
19 # Returns the covariance and eigen values and
20 vectors.
21 def calculate_cov_eig(new_x):
22     cov_mat = np.cov(new_x.T)
23     eigenvalues, eigenvectors = np.linalg.eig(
24         cov_mat)
25     print(f'Covariance Matrix: {cov_mat}\
26 nEigen Values: {eigenvalues}\nEigen
27 Vectors: {eigenvectors}')
28     return cov_mat, eigenvalues, eigenvectors
29
30 # Implement PCA
31 def apply_pca(eigen_row, x):
32     pca_data = np.matmul(eigen_row, x.T)
33     pca_data = pca_data.T
34     return pca_data
35
36 x1 = np.random.randn(20)
37 x2 = np.random.randn(20)
38
39 visualize_data(x1, x2)
40
41 d = np.random.rand(2, 2)
42 x = [x1, x2]
43 x = np.array(x).T
44 print(x)
45
46 new_x = np.matmul(x, d)
47 visualize_data(new_x[:, 0], new_x[:, 1], '
48 Random plot after multiplying with 2D
49 matrix')
50
51 cov, eigenvalues, eigenvectors =
52 calculate_cov_eig(new_x)
53
54 data = apply_pca(eigenvectors[:, 1], new_x)
55
56 visualize_data(data, np.zeros_like(data), '
57 Data after PCA')
58
59 # use of sklearn
60 from sklearn.decomposition import PCA
```

```
51
52 # Assume you have a dataset called 'data' with
53 dimensions (m, n)
54
55 # Create an instance of PCA with the desired
56 number of components
57 n_components = 1 # Specify the number of
58 principal components you want to retain
59 pca = PCA(n_components=n_components)
60
61 # Fit the PCA model to the data
62 pca.fit(new_x)
63
64 # Transform the data to the lower-dimensional
65 space
66 transformed_data = pca.transform(new_x)
67
68 # Access the explained variance ratio
69 visualize_data(transformed_data, np.zeros_like
70 (transformed_data), 'Scikit PCA')
```

### B. Irish Flower Dataset

```
1 from sklearn import datasets
2 from sklearn.preprocessing import
3 StandardScaler
4 import numpy as np
5 import pandas as pd
6 import matplotlib.pyplot as plt
7
8 # Visualize the data
9 def visualize_data(x, y, title='Random Plot',
10 target=[0], label=['Principal component 1'
11 , 'Principal component 2']):
12     colormap = np.array(['r', 'g', 'b'])
13     plt.scatter(x, y, c=colormap[target])
14     plt.title(title)
15     plt.xlabel(label[0])
16     plt.ylabel(label[1])
17     plt.legend([f'Roll [5,9]'])
18     plt.show()
19
20 # Calculate covariance matrix and eigen values
21 and eigen vectors
22 def cal_cov_eig(new_x):
23     cov_mat = np.cov(new_x.T)
24     eigenvalues, eigenvectors = np.linalg.eig(
25         cov_mat)
26     print(f'Covariance Matrix:\n{cov_mat}\
27 nEigen Values:\n{eigenvalues}\nEigen
28 Vectors:\n{eigenvectors}')
29     return cov_mat, eigenvalues, eigenvectors
30
31 # Implement PCA
32 def apply_pca(eigen_row, x):
33     pca_data = np.matmul(eigen_row, x.T)
34     pca_data = pca_data.T
35     return pca_data
36
37 # Calculate the proportion of each eigen value
38 def cal_prop(e):
39     prop = []
40     sum = np.sum(e)
41     for i in range(len(e)):
42         prop.append((e[i] / sum) * 100)
43     return prop
```



```

37
38 # Return row-wise eigen vector
39 def sort_evect(eval, evect):
40     pair = dict()
41     for i in range(len(eval)):
42         pair[np.abs(eval[i])] = evect[:, i]
43     sorted_pair = dict(sorted(pair.items(),
44                             reverse=True))
45     evect_sorted = np.array(list(sorted_pair.
46                             values()))
47     return evect_sorted
48
49 # Load the dataset
50 from sklearn.datasets import load_iris
51 ds = load_iris()
52 df = pd.DataFrame(data=ds.data, columns=ds.
53                   feature_names)
54 df['target'] = pd.Series(ds.target)
55
56 x = df.iloc[:, 0:4].values
57 x_std = StandardScaler().fit_transform(x)
58
59 cov_mat, eigenvalues, eigenvectors =
60     cal_cov_eig(x_std)
61 p = cal_prop(eigenvalues)
62
63 # Number of principal components
64 k = 2
65 evect_sorted = sort_evect(eigenvalues,
66                           eigenvectors)
67 output = apply_pca(evect_sorted[:, k, :], x_std)
68
69 visualize_data(output[:, 0], output[:, 1], '
70               Fig: PCA with two components on Iris
71               flower data', df.iloc[:, 4])
72
73 # Using scikit-learn's PCA
74 from sklearn.decomposition import PCA
75
76 n_components = 2 # Specify the number of
77                 principal components you want to retain
78 pca = PCA(n_components=n_components)
79
80 pca.fit(x_std)
81 transformed_data = pca.transform(x_std)
82 explained_variance_ratio = pca.
83     explained_variance_ratio_
84 principal_components = pca.components_
85
86 visualize_data(transformed_data[:, 0],
87                 transformed_data[:, 1], 'Fig: PCA using
88                 scikit on Iris flower data', df.iloc[:,
89                 4])
90
91 # PCA using the 2nd and 3rd largest
92 eigenvalues
93 evect_sorted = sort_evect(eigenvalues,
94                           eigenvectors)
95 output = apply_pca(evect_sorted[1:3, :], x_std
96 )
97 visualize_data(output[:, 0], output[:, 1], '
98               Fig: PCA with two components on Iris
99               flower data using 2nd and 3rd PC', df.iloc
100              [:, 4])

```

### C. Wheat Seed Dataset

```

1 import numpy as np
2 from sklearn.preprocessing import
3     StandardScaler
4 import matplotlib.pyplot as plt
5 import pandas as pd
6
7 # Visualizing 3D PCA on Seed Dataset
8 def plot_3d(output, title='Fig: 3 components
9 PCA on Seed Dataset'):
10     fig = plt.figure(figsize=(12, 10))
11     ax = fig.add_subplot(111, projection='3d')
12     colormap = np.array(['r', 'g', 'b'])
13     ax.scatter(
14         output[:, 0], # X coordinates of
15         output[:, 1], # Y coordinates of
16         output[:, 2], # Z coordinates of
17         c=colormap[df.iloc[:, 7]]
18     )
19     plt.title(title)
20     plt.legend([f'Roll [5,9]'])
21     ax.set_xlabel('Principal Component 1')
22     ax.set_ylabel('Principal Component 2')
23     ax.set_zlabel('Principal Component 3')
24     plt.show()
25
26 # Visualizing 2D PCA on Seed Dataset
27 def plot_2d(output, title='Fig: 2 components
28 PCA on Seed Dataset'):
29     colormap = np.array(['r', 'g', 'b'])
30     plt.scatter(
31         output[:, 0], # X coordinates of
32         output[:, 1], # Y coordinates of
33         c=colormap[df.iloc[:, 7]]
34     )
35     plt.title(title)
36     plt.legend([f'Roll [5,9]'])
37     plt.xlabel('Principal Component 1')
38     plt.ylabel('Principal Component 2')
39     plt.show()
40
41 # Sorting eigen vector based on eigen value
42 def sort_evect(eval, evect):
43     pair = dict()
44     for i in range(len(eval)):
45         pair[np.abs(eval[i])] = evect[:, i]
46     sorted_pair = dict(sorted(pair.items(),
47                             reverse=True))
48     evect_sorted = np.array(list(sorted_pair.
49                             values()))
50     return evect_sorted
51
52 # Calculating proportion of eigen value
53 def cal_prop(e):
54     prop = []
55     sum = np.sum(e)
56     for i in range(len(e)):
57         prop.append((e[i] / sum) * 100)
58     return prop
59
60 df = pd.read_csv('Seed_Data.csv')

```

```

57 data = df.iloc[:, :7].to_numpy()
58 data_std = StandardScaler().fit_transform(data
59 )
60 covariance = np.cov(data_std.T)
61 eigen_value, eigen_vect = np.linalg.eig(
62     covariance)
63 prop = cal_prop(eigen_value)
64 eigen_vect_sorted = sort_evect(eigen_value,
65     eigen_vect)
66 k = 3 # Number of principal components
67
68 output_data = np.matmul(eigen_vect_sorted[:k -
69     1, :], data_std.T)
70 output = output_data.T
71 output_3d = np.matmul(eigen_vect_sorted[:k,
72     :], data_std.T)
73 output_3d = output_3d.T
74
75 plot_2d(output, title='Fig: 2 components PCA
76     on Seed Dataset')
77 plot_3d(output_3d, title='Fig: 3 components
78     PCA on Seed Dataset')
79
80 from sklearn.decomposition import PCA
81
82 pca2d = PCA(n_components=2)
83 pca3d = PCA(n_components=3)
84
85 pca2d.fit(data_std)
86 pca3d.fit(data_std)
87
88 data2d = pca2d.transform(data_std)
89 data3d = pca3d.transform(data_std)
90
91 plot_2d(data2d, title='Fig: 2 components PCA
92     on Seed Dataset using scikit')
93 plot_3d(data3d, title='Fig: 3 components PCA
94     on Seed Dataset using scikit')
95
96 #PCA using 3 and 4 th larger eigen value
97 output_data = np.matmul(eigen_vect_sorted
98     [2:4, :], data_std.T)
99 output_new= output_data.T
100 # output_3d = np.matmul(eigen_vect_sorted[:k
101     ,:], data_std.T)
102 # output_3d = output_3d.T
103 plot_2d(output_new, title='Fig: 2 components
104     PCA on Seed Dataset using 3rd and 4th PC')

```