Principal Component Analysis Breakdown (July 2023)

Anuj Rayamajhi[1], UG, Aayush Regmi[1], UG. and Editor Arun Subedi[1], UG
[1]Institute of Engineering Thapathali Campus, Tribhuvan University, Kathmandu, Nepal

ABSTRACT Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction in machine learning. In this study, we investigate the impact of PCA on the performance of logistic regression for breast cancer classification. The original breast cancer dataset with 30 features is evaluated using logistic regression, achieving an accuracy of 96% and excellent precision and recall scores. By applying PCA and selecting six major principal components, we obtain comparable results, demonstrating the effectiveness of dimensionality reduction without significant loss in classification accuracy. Additionally, reduced datasets with two and three principal components maintain strong precision and recall values, albeit with slightly lower accuracy. These findings highlight the significance of PCA in simplifying the breast cancer dataset while preserving essential characteristics for accurate classification. This study provides insights into the benefits of PCA and its implications for logistic regression in breast cancer classification.

KEYWORDS Principal Component Analysis, Dimensionality reduction, Logistic regression, Eigen value decomposition

I. INTRODUCTION

In modern days, data used for analysis is often characterized by high-dimensional features due to advancements in data collection technologies and the availability of massive datasets. These high-dimensional datasets present challenges in terms of computational complexity, model interpretability, and overfitting. Therefore, effective dimensionality reduction techniques are crucial for extracting meaningful information and improving the performance of machine learning algorithms.

The curse of dimensionality is a well-known issue associated with high-dimensional data. As the number of features increases, the available data becomes sparse, resulting in increased computational complexity and decreased predictive accuracy. Moreover, high-dimensional data often contains irrelevant or redundant features, which can negatively impact model performance and generalizability. Addressing the curse of dimensionality is essential for enhancing the efficiency and effectiveness of machine learning algorithms.

Various dimensionality reduction methods have been developed to tackle the curse of dimensionality. Two commonly used techniques are Feature Selection and Feature Extraction. Feature Selection methods aim to identify and select a subset of relevant features from the original dataset. On the other hand, Feature Extraction methods transform the original features into a new set of lower-dimensional features, typically referred to as "latent variables" or "principal components."

In this study, our main objective is to employ Principal Component Analysis (PCA), a widely used feature extraction technique, to address the curse of dimensionality in the context of breast cancer classification. We aim to transform the original high-dimensional breast cancer dataset into a lower-dimensional representation, capturing the most informative patterns and preserving essential characteristics for accurate classification. Additionally, we extend our analysis to include other datasets, such as the Iris dataset, to gain a broader understanding of PCA's effectiveness. Moreover, we explore the application of PCA from scratch by utilizing a (20,2) matrix of random numbers drawn from a normal distribution. Through comprehensive evaluations, including logistic regression, we will assess the impact of PCA on classification performance and interpretability for each dataset. This study

aims to provide valuable insights into the practical applications of PCA as a dimensionality reduction technique in various classification tasks, including breast cancer classification.

II. METHODOLOGY

A. THEORY

Principal Component Analysis (PCA) is a widely used dimensionality reduction technique that aims to transform a high-dimensional dataset into a lower-dimensional representation while retaining the most important information. It accomplishes this by finding a set of orthogonal axes, called principal components, that capture the maximum variance in the data.

The steps involved in applying PCA can be summarized as follows:

1) STANDARDIZATION

The input dataset is first standardized by subtracting the mean and dividing by the standard deviation of each feature. This step ensures that all features have a similar scale, preventing dominance by features with larger magnitudes.

Mathematical Equation: For a dataset with *N* samples and *D* features, the standardized dataset is computed as follows:

$$X_{standardized} = \frac{X - mean(X)}{std(X)} \quad (1)$$

Here, *X* represents the original dataset, *mean(X)* represents the mean value of each feature, and *std(X)* represents the standard deviation of each feature.

2) COVARIANCE MATRIX COMPUTATION

The covariance matrix is computed from the standardized dataset. The covariance matrix provides information about the relationships and variances between different features.

Mathematical Equation: The covariance matrix *C* is computed as:

$$C = \frac{1}{N-1}\left(X_{std} - mean(X_{std})\right)^T (X_{std} - mean(X_{std})) \quad (2)$$

Here, *N* represents the number of samples, X_{std} represents the standardized dataset, and $mean(X_{std})$ represents the mean value of the standardized dataset.

3) EIGEN DECOMPOSITION

The covariance matrix is then subjected to Eigen decomposition, yielding a set of eigenvectors and eigenvalues. The eigenvectors represent the principal components, while the eigenvalues indicate the amount of variance explained by each principal component.

Mathematical Equation: The Eigen decomposition of the covariance matrix *C* is computed as:

$$C = C \cdot V \cdot V^T \quad (3)$$

Here, *V* represents the matrix of eigenvectors, and *D* is a diagonal matrix containing the eigenvalues.

4) SELECTION OF PRINCIPAL COMPONENTS

The eigenvectors corresponding to the largest eigenvalues are selected as the principal components. These components capture the most significant information in the dataset.

The principal components are selected based on the eigenvalues. The eigenvectors corresponding to the largest eigenvalues capture the most significant information. Eigen vectors are selected based on eigen values.

5) PROJECTION

The original dataset is projected onto the new lower-dimensional space spanned by the selected principal components. This projection allows for dimensionality reduction while preserving the most important information for subsequent analysis or classification tasks.

The projection of the dataset "*X*" onto the lower-dimensional space spanned by the principal components is computed as:

$$X_{projected} = X_{standardized} \cdot V_{selected} \quad (4)$$

Here, $V_{selected}$ represents the matrix of selected eigenvectors corresponding to the principal components

By applying PCA, we can reduce the dimensionality of the dataset while maintaining a substantial portion of its variance, allowing for a more efficient representation and potentially improving the performance of subsequent analysis or classification tasks.

B. SYSTEM BLOCK DIAGRAM

1) INPUT

The initial step of the system is to provide the input dataset, which contains the samples and their associated features.

2) FEATURE EXTRACTION

The dataset is separated into the feature matrix and the target vector. The feature matrix contains the input features, while the target vector represents the corresponding target values for classification or analysis.

3) STANDARDIZATION

The feature matrix undergoes a standardization process, where each feature is centered around zero and scaled by its standard deviation. This step ensures that all features have a similar scale, preventing any one feature from dominating the PCA process.

4) COVARIENCE MATRIX

The standardized feature matrix is used to calculate the covariance matrix. The covariance matrix provides information about the relationships and variances between different features.

5) EIGENVALUE DECOMPOSITION

The covariance matrix is subjected to eigenvalue decomposition, resulting in a set of eigenvalues and eigenvectors. The eigenvalues represent the amount of variance explained by each principal component, while the eigenvectors represent the directions of the principal components.

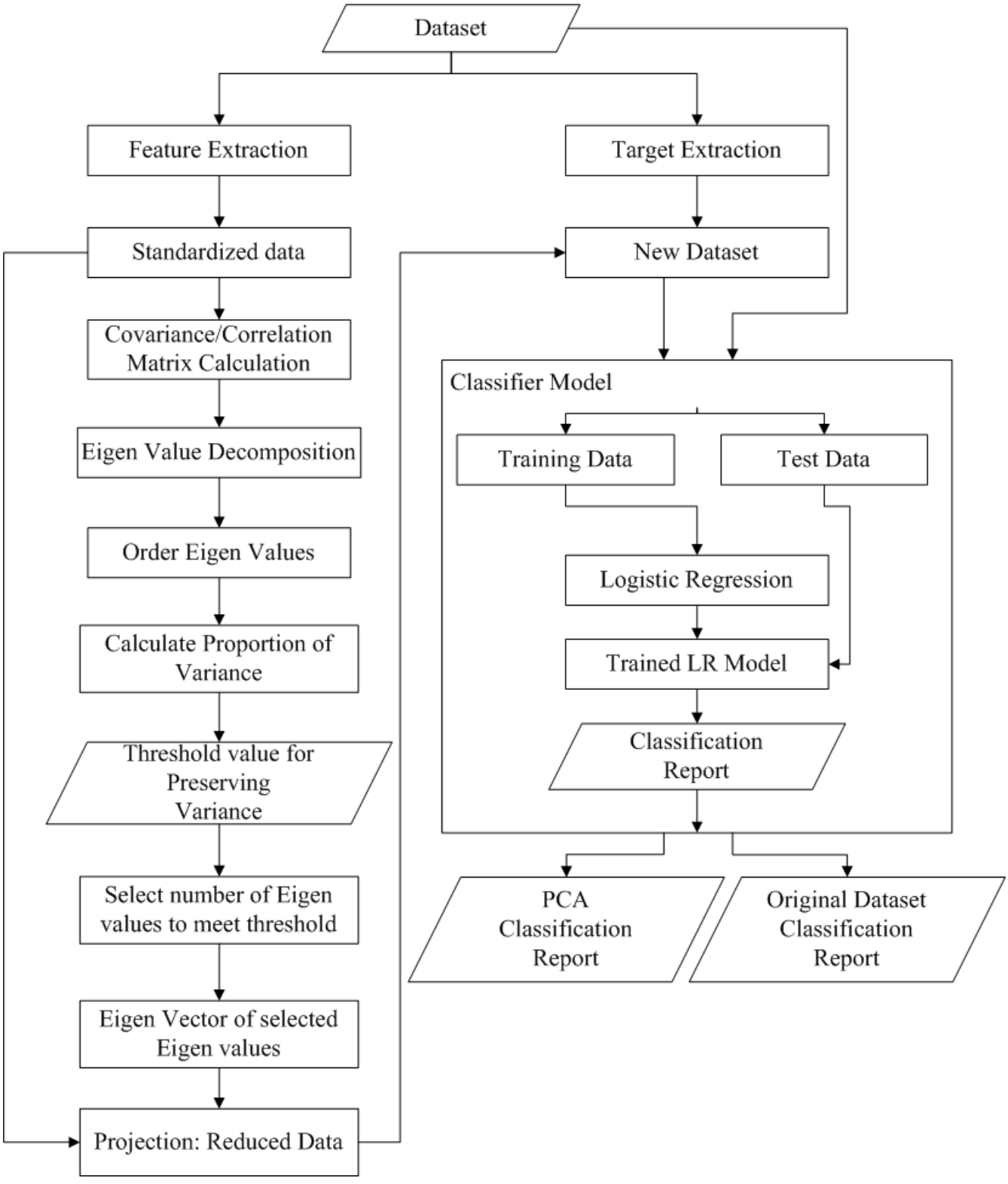

Figure 1 : System Block Diagram

6) ORDER EIGEN VALUES AND REEORGANIZED EIGEN VECTORS

The eigenvalues are ordered in descending order, and the corresponding eigenvectors are reorganized accordingly. This step ensures that the principal components are aligned with the most significant variance.

7) PROPORTION OF VARIENCE CALCULATION

The proportion of variance explained by each principal component is calculated by dividing the eigenvalues by the sum of all eigenvalues. This information provides insights into the relative importance of each principal component.

8) THRESHOLD VALUE FOR PRESERVING VARIENCE

A threshold value is set to determine the desired proportion of variance to be preserved. This threshold value acts as a criterion for selecting the number of principal components to retain.

9) SELECT EIGENVALUES TO MEET THRESHOLD

The eigenvalues are examined in descending order, and the corresponding eigenvectors are selected until the cumulative proportion of variance meets or exceeds the threshold value.

10) EIGEN VECTORS OF SELECTED EIGENVALUES

The selected eigenvectors corresponding to the chosen eigenvalues are retained. These eigenvectors represent the directions in the feature space that capture the most important information.

11) PROJECTION

The original standardized feature matrix is projected onto the lower-dimensional space spanned by the selected eigenvectors. This projection reduces the dimensionality of the dataset while preserving the most significant information for subsequent analysis or classification tasks.

12) MERGE PRJECTION AND TARGET EXTRACTION

The projected feature matrix is combined with the target vector to form a new reduced dataset. This dataset contains the transformed features in the lower-dimensional space, ready for further analysis or classification.

The system block diagram illustrates the sequential steps involved in PCA, starting from the input dataset and culminating in a new reduced dataset.

III. RESULTS

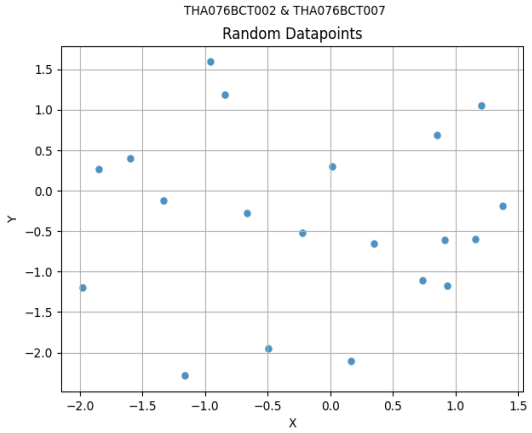

Figure 2 : Random Datapoints

This plot is the representation of 20 sample datapoints which are randomly generated using Numpy library from the normal distribution. These values are taken from the normal distribution to generate a standard data, which means the mean is centered near 0 and standard deviation is near 1.

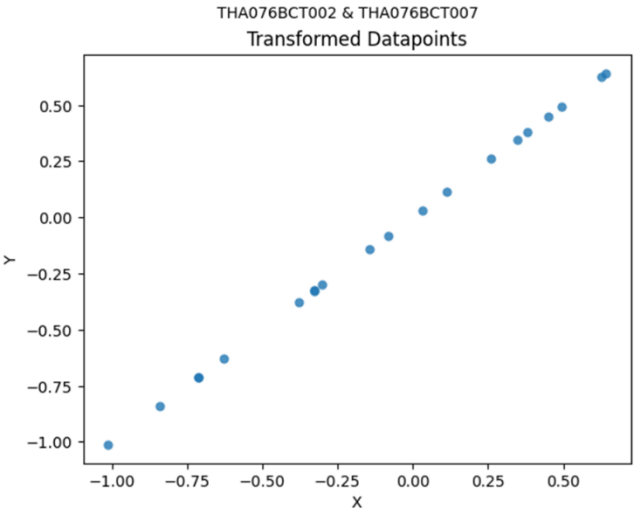

Figure 3 : Transformed Datapoints

The sample data is then transformed using a matrix to simulate a linear transformation or rotation of the data points. In PCA, the transformation matrix is typically derived from the covariance matrix of the original data. However, in this code, a random matrix is generated and used for the transformation. This is done to demonstrate the effect of a transformation on the data and how PCA can still capture the underlying structure even after the transformation.

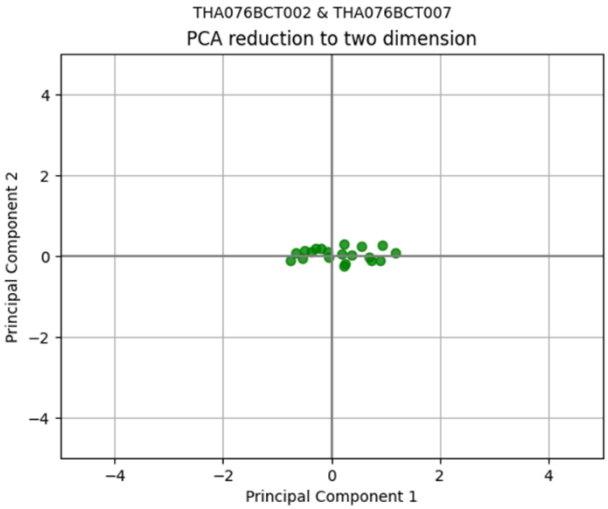

Figure 4 : PCA reduction to two dimensions

This scatter plot visually represents how the original data points have been transformed and mapped onto the new coordinate system defined by the principal components, principal component 1 at x-axis and principal component 2 at y-axis. The plot allows us to observe the distribution, clustering, and any patterns or trends in the reduced data.

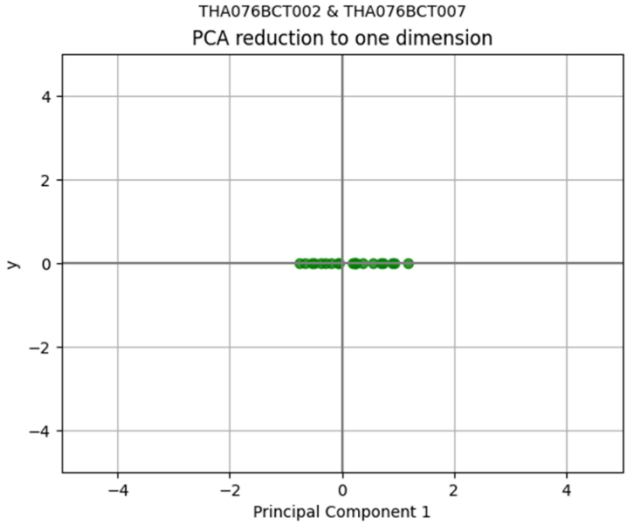

Figure 5 : PCA reduction to one dimension

The plot represents the result of PCA reduction to one dimension. In this plot, the x-axis represents the principal component (PC) 1, which is the single dimension obtained after reducing the data using PCA. The y-axis represents the constant value of 0. Since we have reduced the data to one dimension, each data point is now represented by a single value along the x-axis. The scatter plot shows the distribution of the data points in this reduced space.

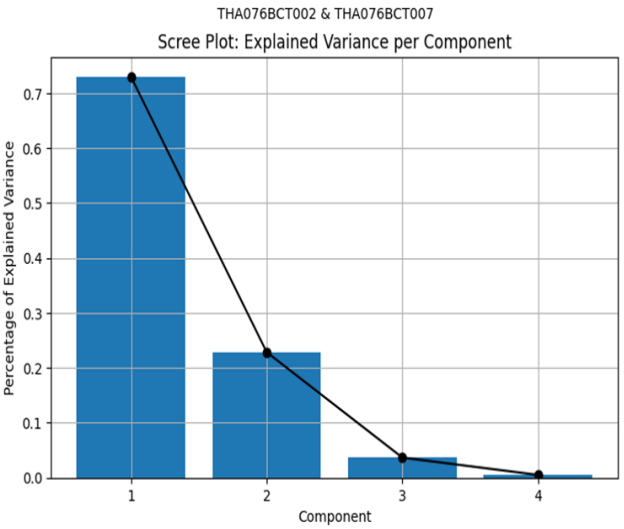

Figure 6 : Scree Plot : Explained Variance per Component

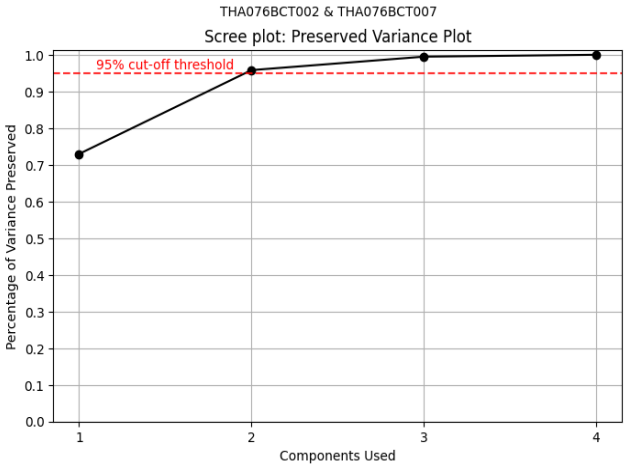

Figure 7 : Scree plot: Preserved Variance Plot

The plot generated at step 8 is known as a scree plot, which displays the proportion of variance (POV) explained by each component and the cumulative sum of the explained variance along each component. The x-axis represents the components, and the y-axis represents the percentage of explained variance.

In the scree plot, each component is represented by a point or a bar. The height of each bar or the position of each point on the y-axis represents the percentage of variance explained by that component.

Looking at the specific values of POV and explained variance per component provided:

- For the first component (PC1), the POV is 0.7296, which means it explains approximately 72.96% of the variance in the data.
- For the second component (PC2), the POV is 0.2285, explaining around 22.85% of the variance.
- For the third component (PC3), the POV is 0.0367, explaining about 3.67% of the variance.
- For the fourth component (PC4), the POV is 0.0052, accounting for approximately 0.52% of the variance.

The cumulative sum of explained variance at each component is also calculated. It shows the accumulated percentage of variance explained up to each component. In this case, the cumulative sum is as follows:

- After considering PC1, the cumulative sum is 0.7296, indicating that PC1 alone explains 72.96% of the variance.
- After including PC2, the cumulative sum becomes 0.9581, meaning that PC1 and PC2 together explain 95.81% of the variance.
- When PC3 is added, the cumulative sum reaches 0.9948, indicating that PC1, PC2, and PC3 combined explain 99.48% of the variance.
- Finally, with the inclusion of PC4, the cumulative sum becomes 1.0, indicating that all four components collectively explain 100% of the variance.

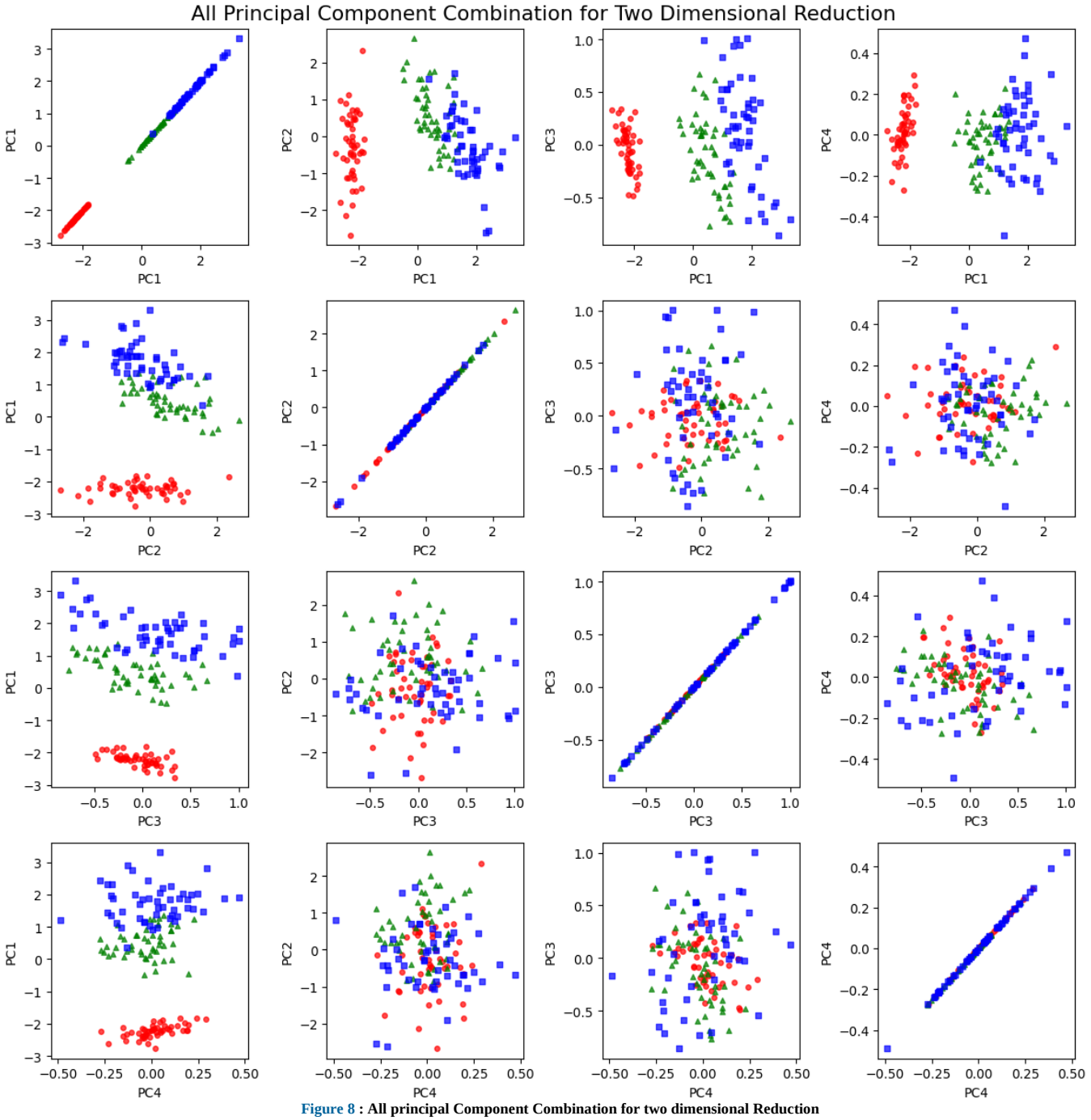

Figure 8 : All principal Component Combination for two dimensional Reduction

The scree plot helps in determining the number of components to retain in the dimensionality reduction. In this case, it suggests that retaining the first two components (PC1 and PC2) would preserve a significant amount of variance (95.81%), making them the most informative components for the Iris dataset.

In the scatter plot of figure (9), the x-axis represents the first principal component (PC1), and the y-axis represents the second principal component (PC2). Each data point from the Iris dataset is projected onto these two principal components. The plot includes three different colors and markers to represent the three target classes of the Iris dataset: Setosa, Versicolor, and Virginica. Each class is visualized separately, with data points belonging to the same class having the same color and marker. By reducing the data to two dimensions using PCA, the plot allows to visualize the separation and distribution of the data points in a lower-dimensional space.

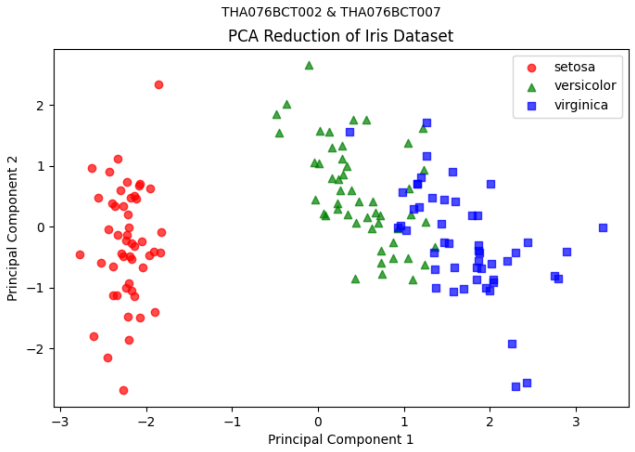

Figure 9 : PCA Reduction of Iris Dataset

The goal of PCA is to maximize the variance along the principal components, so the plot shows how the data is spread out along the PC1 and PC2 axes. The plot provides insights into the relationships and patterns within the Iris dataset, allowing us to observe any clustering or grouping of the data points. It helps us understand the separability of the different classes and can be useful for further analysis or classification tasks.

The plot in figure (8) displays all possible combinations of two principal components (PCs) in a grid of scatter plots. Each scatter plot represents a different pair of PCs, with the x-axis representing one PC and the y-axis representing another PC.

The purpose of this plot is to visualize the different views of the data in reduced dimensions. By examining each scatter plot, we can gain insights into how the data is distributed and how well it separates the different classes.

In this specific case, the Iris dataset has four original features: sepal length, sepal width, petal length, and petal width. After performing PCA, the data is reduced to a lower-dimensional space represented by the principal components. The scatter plots in this plot show the distribution of the data points in the reduced space, capturing the relationships between different pairs of PCs.

Analyzing the plot, we can observe the following:

- Each scatter plot corresponds to a specific pair of PCs. For example, the scatter plot in the top-left corner represents the relationship between PC1 and PC1, which is simply a plot of PC1 against itself.
- The scatter plots show the distribution of the data points for each pair of PCs. The colors and markers used in the scatter plots represent the different classes in the Iris dataset (setosa, versicolor, and virginica).
- By examining the scatter plots, we can observe how well the different classes are separated in the reduced space. Ideally, we would like to see distinct clusters or patterns for each class, indicating good separability.
- The position and dispersion of the data points in each scatter plot provide information about the relationships between the PCs. For example, if the data points form a tight cluster or exhibit a clear linear trend, it suggests a strong correlation between the corresponding PCs.

This plot allows us to visually explore the relationships and separability of the data in the reduced space. It can help us understand the effectiveness of PCA in capturing the most significant patterns and variations in the Iris dataset and provide insights into the data's intrinsic structure.

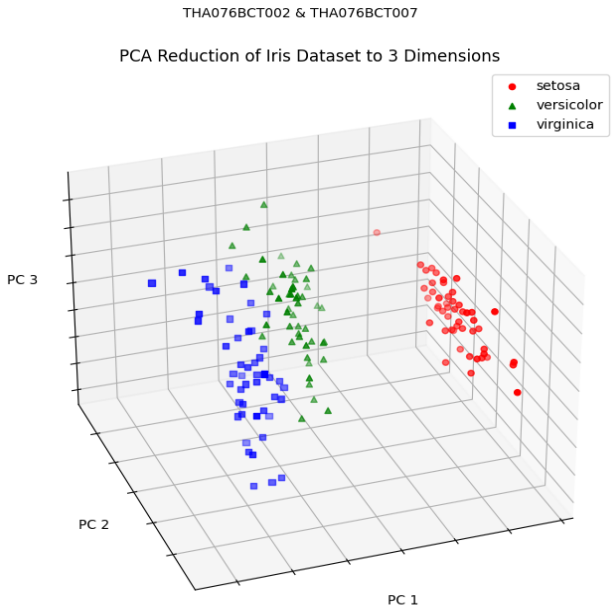

Figure 10 : PCA Reduction of Iris Dataset to 3 Dimension

The plot represents the reduction of the Iris dataset to three dimensions using PCA. It is a 3D scatter plot where each data point is represented by its values along the three major principal components (PCs). The x-axis represents PC 1, the y-axis represents PC 2, and the z-axis represents PC 3. In the plot, each data point is visualized as a marker with a specific color and shape corresponding to its target class (setosa, versicolor, or virginica). This allows to observe the distribution of the data points in the reduced three-dimensional space. By reducing the dimensionality of the dataset to three components, significant amount of the original variance in the data is captured. This allows to visualize the data in a lower-dimensional space while still preserving important information.

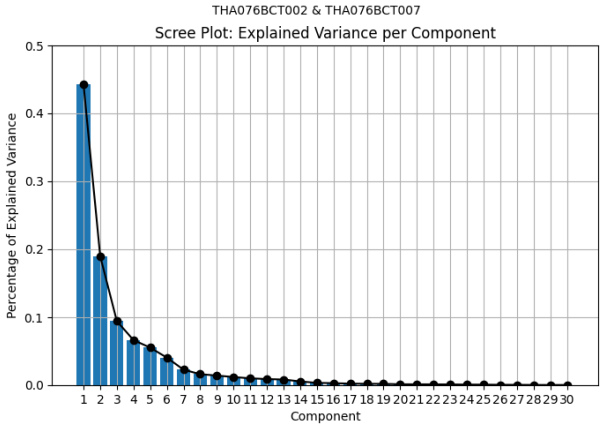

Figure 11 : Scree Plot: Explained Variance per Component

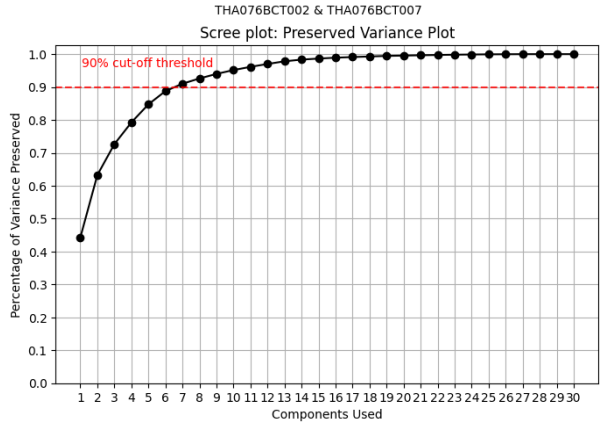

Figure 12 : Scree Plot : Preserved Variance plot

The plot generated at step 9 is a scree plot, which shows the proportion of variance explained by each principal component in descending order. It helps in determining the number of principal components to retain for dimensionality reduction.

The scree plot displays the percentage of the explained variance and preserved variance. The x-axis represents the number of components, ranging from 1 to 30. The y-axis represents the percentage of explained variance and preserved variance along the components. The scree plot shows a steep drop in the percentage of explained variance for the first few components, followed by a more gradual decline. The plot helps identify an "elbow" point, where the explained variance significantly decreases. This point indicates the number of components that retain most of the variance in the data.

Looking at the calculated information, cumulative sum starts at 44.27% for the first component. As we add more components, the cumulative sum increases. By the 2nd component, the cumulative sum reaches 63.24%, indicating that the first two components explain a significant portion of the variance. By the 3rd component, the cumulative sum reaches 72.64%. The scree plot visually represents this information. It shows a steep drop from the first component to the second component, indicating that the first component explains a large portion of the variance. As we add more components, the decrease becomes more gradual, suggesting diminishing returns in terms of explained variance.

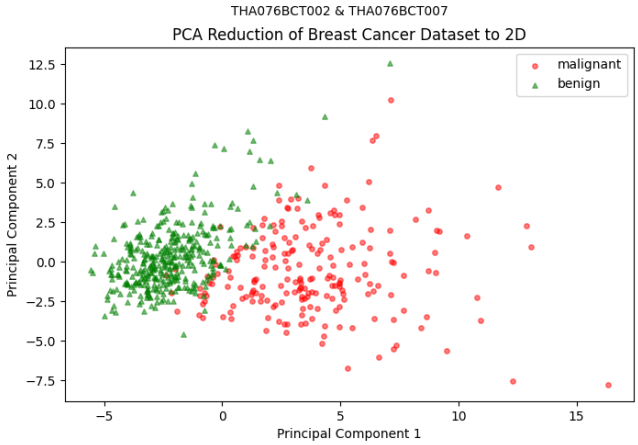

Figure 13 : PCA Reduction of Breast Cancer Dataset to 2D

The 2D plot represents the Breast Cancer dataset after reducing it to two dimensions using PCA. The x-axis represents the first principal component (PC1), and the y-axis represents the second principal component (PC2). Each point on the plot represents a data sample from the dataset. In the scatter plot, the points are color-coded and shape-coded based on the target classes, Malignant and Benign. This 2D plot allows us to visualize the distribution and separation of the data samples in a reduced feature space. We can observe how the different classes are distributed and if there is any clustering or overlap between them. The goal is to achieve a clear separation between the classes, indicating that PCA has successfully captured the relevant information for classification.

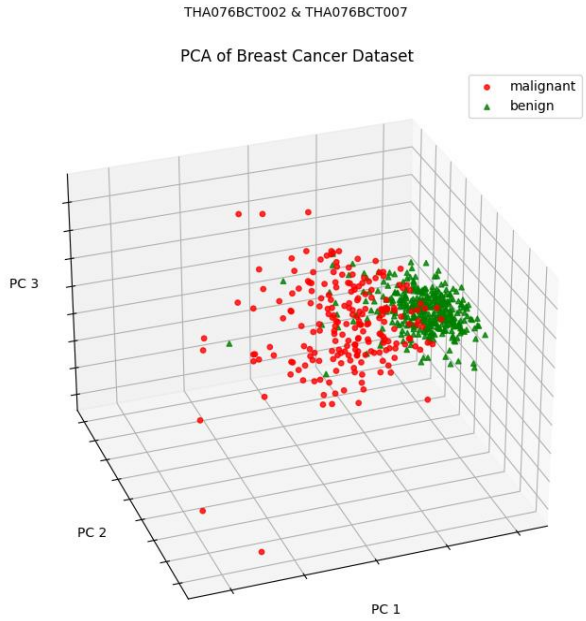

Figure 14 : PCA of Breast Cancer Dataset

The 3D plot represents the Breast Cancer dataset after reducing it to three dimensions using PCA. Similar to the 2D plot, the axes represent the first three principal components (PC1, PC2, and PC3). Each point in the plot corresponds to a data sample, and the color and shape of the markers indicate the target classes (Benign and Malignant) as in the 2D plot.

The 3D plot provides additional information compared to the 2D plot by visualizing the dataset in three dimensions. It allows us to observe the distribution of the data samples from different perspectives and potentially discover more complex patterns or relationships between the classes.

By visualizing the data after dimensionality reduction, these plots help us assess the effectiveness of PCA in capturing the essential information and separating the classes. If the classes are well-separated and show distinct clusters in the reduced feature space, it indicates that PCA has successfully captured the underlying structure of the data, making it easier to classify the samples.

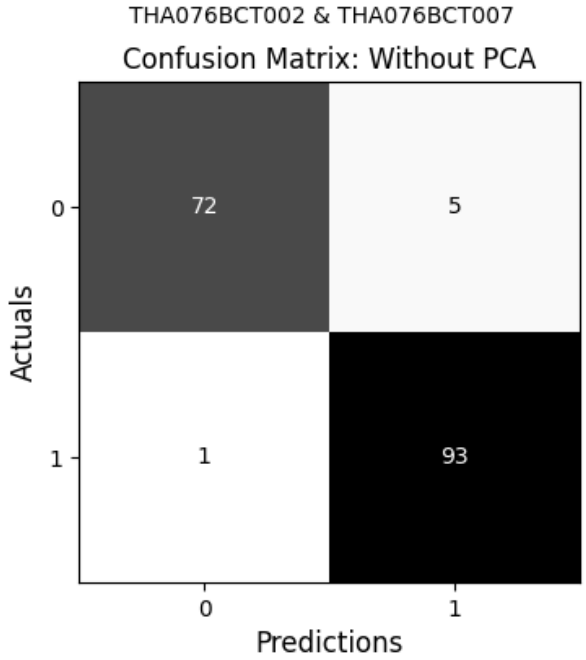

Figure 15 : Confusion Matrix: Without PCA

The results obtained from the logistic regression performed on the original dataset are:

- Accuracy: 0.96
- Precision: 0.99 (Malignant, 0), 0.95 (Benign, 1)
- Recall: 0.94 (Malignant, 0), 0.99 (Benign, 1)
- F1-Score: 0.96 (Malignant, 0), 0.97 (Benign, 1)

The logistic regression model achieved high accuracy (96%) on the original data. It shows excellent precision and recall values for both classes, indicating a good balance between correctly identifying positive and negative samples. The F1-scores are also high, indicating a reliable overall performance of the model. The low number of false positives and false negatives, as shown in the confusion matrix, further supports the effectiveness of the model on the original dataset.

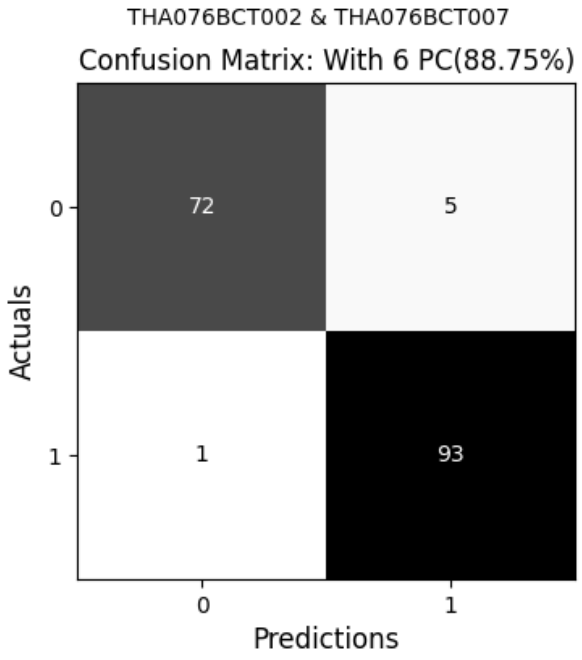

Figure 16 : Confusion Matrix: With 6 PC (88.75%)

- Accuracy: 0.96
- Precision: 0.99 (Malignant, 0), 0.95 (Benign, 1)
- Recall: 0.94 (Malignant, 0), 0.99 (Benign, 1)
- F1-Score: 0.96 (Malignant, 0), 0.97 (Benign, 1)

The regression model achieved the same high performance on the dataset reduced to six principal components as on the original data. This indicates that the reduced dataset retains the essential information required for accurate classification. The similar precision, recall, F1-scores, and confusion matrix values demonstrate that the reduction to six principal components effectively captures the discriminative features for classification.

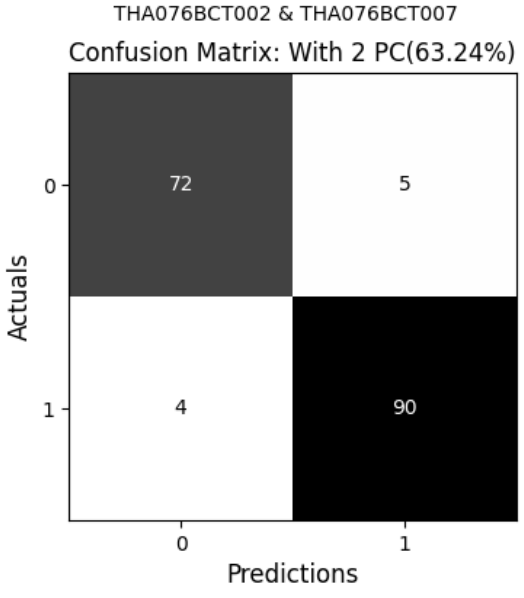

Figure 17 : Confusion Matrix: With 2 PC (63.24%)

- Accuracy: 0.95
- Precision: 0.95 (Malignant, 0), 0.95 (Benign, 1)
- Recall: 0.94 (Malignant, 0), 0.96 (Benign, 1)
- F1-Score: 0.94 (Malignant, 0), 0.95 (Benign, 1)

The regression model on the dataset reduced to two principal components achieved slightly lower accuracy compared to the original and six-component reduced datasets. However, the precision, recall, and F1-scores for both classes remain high. The confusion matrix shows a slightly higher number of false negatives (4) compared to the other results, indicating that the reduced dataset may have lost some discriminative information. Despite this, the model still performs well in accurately classifying most of the samples.

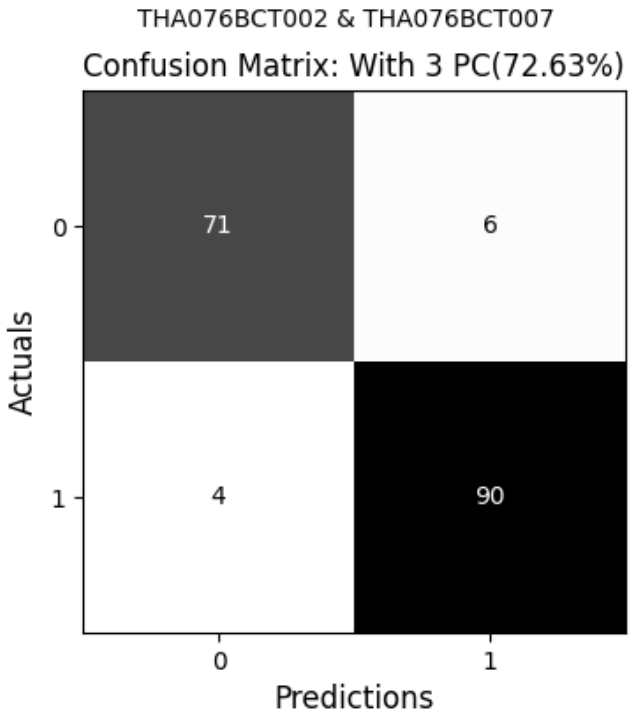

Figure 18 : Confusion Matrix : With 3 PC (72.63%)

- Accuracy: 0.94
- Precision: 0.95 (class 0), 0.94 (class 1)
- Recall: 0.92 (class 0), 0.96 (class 1)
- F1-Score: 0.93 (class 0), 0.95 (class 1)

The regression model on the dataset reduced to three principal components shows a slightly lower performance compared to the previous results. The accuracy, precision, recall, and F1-scores have decreased slightly, and the confusion matrix indicates a higher number of false positives and false negatives compared to the other results. This suggests that reducing the dataset to only three principal components might have led to a loss of some relevant information, impacting the model's performance.

The classifier model performs consistently well on both the original and reduced datasets, demonstrating the effectiveness of PCA in preserving the essential information required for accurate classification. Reducing the dataset to six principal components retains the key discriminative features, resulting in similar performance to the original data. Using two or three principal components still yields reasonably good results, although there is a slight degradation in performance compared to the original and six-component reduced datasets. This suggests that a small number of principal components might not capture all the crucial details of the data, leading to a slight decrease in accuracy and higher misclassification rates.

These insights highlight the trade-off between dimensionality reduction and classification performance. While reducing the dimensionality of the dataset can improve efficiency and computational cost, it's crucial to strike a balance between dimensionality reduction and retaining sufficient information for accurate classification.

IV. DISCUSSION AND ANALYSIS

A. DISCUSSION

This study investigated the impact of PCA on the performance of logistic regression for breast cancer classification. The findings of this study have important implications for data analysis and visualization. Dimensioality reduction technique such as PCA can be used to simplify complex datasets and improve the performance of classification methods in some cases. The reduction in the number of features also simplifies the classification model. It is important to note that use of PCA may not always reults in improved performance. The effectiveness of PCA depends on the dataset at hand. In some cases, PCA may result in loss of important information and lead to reducted classification accuracy.

B. ANALYSIS

In this study, PCA was implemented using covariance matrix decomposition. Whereas, the PCA class available in scikit-learn library utilizes singular value decomposition (SVD) for PCA. Both the techniques result in different time and space complexities.

For PCA using covariance matrix decomposition, the algorithm can be summarized by the steps given below:

- Calculate the mean centered data.
- Compute the covariance matrix.
- Compute the eigenvalue decomposition of covariance matrix.

Here, the computationally intensive steps are step 2 and step 3.

The time and space complexity for computing the covariance matrix are $O(N \cdot n^2)$ and $O(n^2)$ respectively where, N is the number of rows/observations and n is the number of features.

Similarly, the time and space complexity for eigenvalue value decomposition are $O(n^3)$ and $O(n^2)$.

Overall, the time and space complexity for PCA using covariance matrix are $O(N \cdot n^2 + n^3)$ and $O(n^2)$.

Using SVD in PCA, the only computationally intensive step is the calculation of SVD. The time and space complexity of SVD is $O(\min(N^2 \cdot n, N * n^2))$ and $O(N^n)$ respectively, where N is the number of rows and n is the number of features. The time complexity depends on the dimensions of the input matrix (either n > m or m > n).

Overall, the time and space complexity for PCA using SVD is $O(\min(N^2 \cdot n, N * n^2))$ and $O(mn)$.

As this study implements PCA using covariance matrix as compared to the SVD technique implemented by the scikit-learn library. The PCA class from scikit-learn tends to be more efficient, when the number of features is much smaller than the number of observations. When the number of features is comparable the number of observations PCA implemented in this paper seems to be more efficient.

V. CONSLUSION

The results of this study demonstrate the effectiveness of Principal Components Analysis in reducing the dimensionality of the dataset without significant loss of variance. The use of PCA for iris and breast cancer dataset resulted in new datapoints with different smaller dimensions reducing the number of features and hence reducing the effect of curse of dimensionality. The use of PCA resulted in reduction of the number of features without much reduction in the classification accuracy of the model. The plot bar plot generated in this study shows that the majority of the information in the dataset can be captured by a small number of features. Overall, the findings of this study provide valuable insights into the benefits of using Principal Component Analysis in dimensionality reduction.

VI. REFERENCES

[1] Shlens, Jonathon. *A Tutorial on Principal Component Analysis*. arXiv, 3 Apr. 2014. arXiv.org

[2] Jolliffe, Ian T., and Jorge Cadima. "*Principal Component Analysis: A Review and Recent Developments.*" Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 374, no. 2065, Apr. 2016, p. 20150202. DOI.org

ANUJ RAYAMAJHI is an ambitious individual currently in the final year of his Bachelor's degree program in Computer Engineering at the esteemed Institute of Engineering Thapathali Campus. With a keen interest in various fields such as Artificial Intelligence, Machine Learning, Data Science, Software Development, and Engineering, Anuj possesses a diverse range of skills and a thirst for knowledge. He constantly seeks out opportunities to enhance his understanding of these subjects through research, online courses, and practical experience.

Apart from his academic pursuits, Anuj has a passion for music, sports, and computer games. In his leisure time, he enjoys exploring different genres of music, engaging in sports activities to stay active, and immersing himself in the virtual worlds of computer games.

With a strong foundation in computer engineering and a multifaceted interest in emerging technologies, Anuj is driven to make significant contributions in the fields of AI, machine learning, and data science. He strives to stay up-to-date with the latest advancements in these domains, continuously expanding his knowledge and honing his skills. Anuj's dedication, enthusiasm, and well-rounded interests make him a promising individual poised to excel in the ever-evolving field of technology

AAYUSH REGMI is a dynamic individual currently pursuing his Bachelor's degree in Computer Engineering at the renowned Institute of Engineering Thapathali Campus. With a strong inclination towards technology, Aayush has developed a keen interest in a wide range of fields, including Artificial Intelligence, Machine Learning, Data Science, Web Development, Quantum Computing, and more. His diverse expertise reflects his commitment to exploring various facets of computer science and pushing the boundaries of innovation.

In addition to his academic pursuits, Aayush finds solace and joy in his hobbies. As an avid sports enthusiast, he actively engages in cricket and football, relishing the thrill of competition and teamwork. Aayush also has a creative side and enjoys playing musical instruments, finding harmony in the melodies he creates.

With an ever-curious mind and a passion for learning, Aayush strives to stay ahead in the rapidly evolving world of technology. He is particularly intrigued by the emerging field of Quantum Computing and its potential to revolutionize the computing landscape. Aayush's dedication, coupled with his diverse skill set, positions him to make significant

contributions in web development, quantum computing, and other innovative domains.

Driven by a relentless pursuit of knowledge and a desire to create a positive impact, Aayush is determined to shape the future through his expertise in computer engineering. His enthusiasm, adaptability, and holistic interests make him a promising individual poised to excel in the dynamic and ever-expanding field of technology.

APPENDIX A : CODE

```
# # Principal Component Analysis

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

from sklearn.decomposition import PCA
from sklearn import datasets
from sklearn.preprocessing import StandardScaler


# ### PCA on randomly generated numbers from Normal
Distribution

np.random.seed(69)

sampleData = np.random.standard_normal((20,2))
sampleData.mean(), sampleData.std()


plt.figure(figsize=(7,5))
plt.scatter(sampleData[:,0], sampleData[:,1],
linewidth = 0.05, alpha = 0.8)
plt.title("Random Datapoints")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid()
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


mat = np.random.rand(2,2)


print(mat)
prod = sampleData.dot(mat)
plt.scatter(prod[:,1], prod[:,1], linewidth=0.05,
alpha=0.8)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Transformed Datapoints')
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


covMat = np.cov(prod.T)
covMat


eigVal, eigVec = np.linalg.eig(covMat)
eigVal, eigVec


#sorting the eigenvalues and rearranging the eigen
vectors to match the sorted eigen values
eigValVec = np.vstack((eigVal, eigVec))
temp = list(zip(*sorted(zip(*eigValVec),reverse =
True)))
eigValVec = np.array(temp)
eigVal = eigValVec[0,:]
eigVec = eigValVec[1:,:]
eigVal, eigVec


#calculating the proportion of variance each eigen
value is able to retain
pov = np.ndarray((2,))
sum_eigVal = np.sum(eigVal)
csum = np.ndarray((2,))
temp = 0
print("POV\tVarianceExplained\tCummulativeSum")
for i in range(0, len(eigVal)):
    pov[i] = eigVal[i]/sum_eigVal
    temp += pov[i]
    csum[i] = temp
    print(f'pov{i+1}\t{pov[i]}\t{csum[i]}')


#perfroming the reduction Y =PX
y = prod.dot(eigVec)
y


plt.scatter(y[:,0],y[:,1],alpha=0.8, c='green')
plt.title('PCA reduction to two dimension')
plt.xlabel('Principal Component 1')
plt.xlim(-5,5)
plt.ylim(-5,5)
plt.ylabel('Principal Component 2')
plt.axhline(y=0, color='grey')
plt.axvline(x=0, color="grey")
plt.grid()
```

```
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


#Performing the reduction in form of Y=PX, reducing to
one dimension, retaining 95.57% variance
Y_pca = prod.dot(eigVec[:,0])
Y_pca
plt.scatter(Y_pca, np.zeros((20,)),alpha=0.8,
c='green')
plt.title('PCA reduction to one dimension')
plt.xlabel('Principal Component 1')
plt.xlim(-5,5)
plt.ylim(-5,5)
plt.ylabel('y')
plt.axhline(y=0, color='grey')
plt.axvline(x=0, color="grey")
plt.grid()
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


# ### PCA for Iris Dataset

#loading the Iris dataset into a dataframe
iris = datasets.load_iris()
irisDf = pd.DataFrame(iris.data,
columns=iris.feature_names)
irisDf['target'] = pd.Series(iris.target)
irisDf.head(5)


#separating the feature values in numpy array X
X_iris = irisDf.iloc[:,0:4].values
#we must check whether the data is standardized or not
by seeing mean and standard deviation
print(X_iris.mean(), X_iris.std())


#since the data is not standard, we use StandardScaler
to make it standard
X_std = StandardScaler().fit_transform(X_iris)
X_std.mean(), X_std.std()


#Now we calculate the covariance matrix for the
standard data. Correlation matrix may also be used
varMatIris = np.cov(X_std.T)
varMatIris


# Performing Eigen Value Decomposition using the
obtained Covariance Matrix
eigValIris, eigVecIris = np.linalg.eig(varMatIris)
eigValIris, eigVecIris


# Sorting and rearranging the eigen values and eigen
Matrix, useful when the eigvalues are not ordered
eigValVec = np.vstack((eigValIris, eigVecIris))
temp = list(zip(*sorted(zip(*eigValVec),reverse =
True)))
eigValVec = np.array(temp)
eigValIris = eigValVec[0,:]
eigVecIris = eigValVec[1:,:]
eigValIris, eigVecIris


# Calculating the proportion of variance each eigen
value is able to retain
povIris = np.ndarray((4,))
sum_eigValIris = np.sum(eigValIris)
csumIris = np.ndarray((4,))
temp = 0
print("POV\tVarianceExplained\tCummulativeSum")
for i in range(0, len(eigValIris)):
    povIris[i] = eigValIris[i]/sum_eigValIris
    temp += povIris[i]
    csumIris[i] = temp
    print(f'pov{i+1}\t{povIris[i]}\t{csumIris[i]}')


#plotting scree plot for explained variance by each
component and
# cummulative preserved variance along each component
plt.figure(figsize=(8,5))
plt.plot(np.linspace(1,4,4), povIris, 'o-',c='black')
plt.bar(np.linspace(1,4,4),povIris)
plt.title('Scree Plot: Explained Variance per
Component')
plt.grid()
plt.xticks(np.arange(1,5,1))
plt.xlabel("Component")
plt.ylabel("Percentage of Explained Variance")
```

```
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


plt.figure(figsize=(8,5))
plt.plot(np.linspace(1,4,4), csumIris, 'o-',c='black')
plt.title('Scree plot: Preserved Variance Plot')
plt.xlabel('Components Used')
plt.ylabel('Percentage of Variance Preserved')
plt.xticks(np.arange(1,5,1))
plt.yticks(np.arange(0,1.1,0.1))

plt.axhline(y=0.95, color='red', linestyle='--
',alpha=0.8)
plt.text(1.1,0.96, '95% cut-off threshold',
color='red', fontsize=10)
plt.grid()
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


# Reducing to two dimensions for preserving 95.81%
variance
yIris = X_std.dot(eigVecIris[:,0:2])
yIris.shape



# Plotting the Information along the two major
Principle Components
plt.figure(figsize=(8,5))
targetValIris = irisDf.iloc[:,-1].values
for c, i, targetName, mark in zip('rgb', [0,1,2],
iris.target_names, "o^s"):
    plt.scatter(yIris[targetValIris==i, 0],
yIris[targetValIris==i, 1],
                c=c, label=targetName, marker=mark,
alpha = 0.7)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend()
plt.title('PCA Reduction of Iris Dataset')
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)
```

```
# calculating the values of all priciple components
yIrisAll = X_std.dot(eigVecIris)
yIrisAll.shape


# Plotting all combination two principle components
rows = 4
cols = 4

fig, axes = plt.subplots(rows,cols,figsize=(12,12))
fig.suptitle('All Principal Component Combination for
Two Dimensional Reduction',fontsize=16)

for row in range(0,rows):
    for col in range(0,cols):
        for c, i, targetName, mark in zip('rgb',
[0,1,2], iris.target_names, "o^s"):
            axes[row][col].scatter(yIrisAll[targetValI
ris==i, row], yIrisAll[targetValIris==i, col],
                                   c=c, label=targetName,
marker=mark, alpha = 0.7, s=15)
#         ax = sns.scatterplot(x=yIrisAll[:,row],
#                              y=yIrisAll[:,col],
#                              ax=axes[row,col])
        axes[row][col].set_xlabel(f'PC{row+1}')
        axes[row][col].set_ylabel(f'PC{col+1}')
plt.tight_layout()
plt.figtext(0.5,1.00,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)



fig = plt.figure(1, figsize=(8,8))
ax = fig.add_subplot(111, projection="3d", elev = -
150, azim = 110)
for c, i, targetName, mark in zip("rgb", [0,1,2],
iris.target_names, "o^s"):
    ax.scatter(yIrisAll[targetValIris==i, 0],
yIrisAll[targetValIris==i, 1],
yIrisAll[targetValIris==i, 2],
               marker=mark, c=c, label=targetName)
ax.set_xlabel("PC 1")
ax.xaxis.set_ticklabels([])
ax.set_ylabel("PC 2")
ax.yaxis.set_ticklabels([])
ax.set_zlabel("PC 3")
ax.zaxis.set_ticklabels([])
```

```
ax.legend()
ax.set_title('PCA Reduction of Iris Dataset to 3
Dimensions')
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


# ### PCA for Breast Cancer Dataset

breastCancer = datasets.load_breast_cancer()
cancerDf = pd.DataFrame(breastCancer.data,
columns=breastCancer.feature_names)
cancerDf['target'] = pd.Series(breastCancer.target)
cancerDf.head(5)


cancerDf['target'] = cancerDf['target'].apply(lambda
x: "Benign" if x==1 else "Malignant")


X_cancer = cancerDf.iloc[:,0:30].values
X_cancer.mean(), X_cancer.std()


Xbc_std = StandardScaler().fit_transform(X_cancer)
Xbc_std.mean(), Xbc_std.std()


varMatCancer = np.cov(Xbc_std.T)
varMatCancer.shape
S

eigValCancer, eigVecCancer =
np.linalg.eig(varMatCancer)
# eigValCancer, eigVecCancer


# Since the recieved Eigen Values are not in ordered
form, reorganizing them
eigValVec = np.vstack((eigValCancer, eigVecCancer))
temp = list(zip(*sorted(zip(*eigValVec),reverse =
True)))
eigValVec = np.array(temp)
eigValCancer = eigValVec[0,:]
eigVecCancer = eigValVec[1:,:]
# eigValCancer, eigVecCancer


# Calculating the proportion of variance each eigen
value is able to retain
povCancer = np.ndarray((30,))
sum_eigValCancer = np.sum(eigValCancer)
csumCancer = np.ndarray((30,))
temp = 0
print("POV\tVarianceExplained\tCummulativeSum")
for i in range(0, len(eigValCancer)):
    povCancer[i] = eigValCancer[i]/sum_eigValCancer
    temp += povCancer[i]
    csumCancer[i] = temp
    print(f'pov{i+1}\t{povCancer[i]}\t{csumCancer[i]}'
)


plt.figure(figsize=(8,5))
plt.plot(np.linspace(1,30,30), povCancer, 'o-
',c='black')
plt.bar(np.linspace(1,30,30),povCancer)
plt.title('Scree Plot: Explained Variance per
Component')
plt.grid()
plt.xticks(np.arange(1,31,1))
plt.xlabel("Component")
plt.ylabel("Percentage of Explained Variance")
plt.ylim(0,0.5)
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


plt.figure(figsize=(8,5))
plt.plot(np.linspace(1,30,30), csumCancer, 'o-
',c='black')
plt.title('Scree plot: Preserved Variance Plot')
plt.xlabel('Components Used')
plt.ylabel('Percentage of Variance Preserved')

plt.xticks(np.arange(1,31,1))
plt.yticks(np.arange(0,1.1,0.1))

plt.axhline(y=0.90, color='red', linestyle='--
',alpha=0.8)
plt.text(1.1,0.96, '90% cut-off threshold',
color='red', fontsize=10)
plt.grid()
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)
```

```
# perform 2 dim reduction and 3 dim reduction for
visualization purpose
Ybc_2 = Xbc_std.dot(eigVecCancer[:,0:2])

plt.figure(figsize=(8,5))
ybc = breastCancer.target
for c, i, targetName, mark in zip("rgb", [0,1],
breastCancer.target_names, 'o^'):
    plt.scatter(Ybc_2[ybc==i, 0], Ybc_2[ybc==i, 1],
c=c, label=targetName, alpha=0.5, marker=mark, s=15)
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend()
plt.title('PCA Reduction of Breast Cancer Dataset to
2D')
plt.figtext(0.5,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


Ybc_3 = Xbc_std.dot(eigVecCancer[:,0:3])

fig = plt.figure(1, figsize=(8,8))
ax = fig.add_subplot(111, projection="3d", elev = -
150, azim = 110)
for c, i, targetName, mark in zip("rgb", [0,1,2],
breastCancer.target_names, 'o^'):
    ax.scatter(Ybc_3[ybc==i, 0], Ybc_3[ybc==i, 1],
Ybc_3[ybc==i, 2], c=c, label=targetName, marker=mark,
s=15, alpha=0.8)
ax.set_xlabel("PC 1")
ax.xaxis.set_ticklabels([])
ax.set_ylabel("PC 2")
ax.yaxis.set_ticklabels([])
ax.set_zlabel("PC 3")
ax.zaxis.set_ticklabels([])
ax.legend()
ax.set_title('PCA of Breast Cancer Dataset')
plt.figtext(0.51,0.95,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


Xbc_std.nbytes


Ybc_6 = Xbc_std.dot(eigVecCancer[:,0:6])

Ybc_6.shape, Ybc_6.nbytes


# #### Using Logistic Regression to analyse the
effectiveness of PCA

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report,
confusion_matrix


from mlxtend.plotting import plot_confusion_matrix


#separating features and target values
features_nopca = Xbc_std
features_pca = Ybc_6
# features_nopca.shape, features_pca.shape
target = breastCancer.target


# ##### Evaluation on Original Data


#spliting the dataset into training and testing set
X_train, X_test, y_train, y_test =
train_test_split(features_nopca, target,
                                                 te
st_size=0.3, random_state=69)
#Creating a Logistic Regression Model
NoPcaModel = LogisticRegression()

#Training this model on original data
NoPcaModel.fit(X_train, y_train)

#Making prediction using this model on test set
y_pred = NoPcaModel.predict(X_test)

#Generating a Classification Report
report = classification_report(y_test, y_pred)
conf = confusion_matrix(y_test,y_pred)
print('Classification Report:\n', report)
print('Confusion Matrix:\n', conf)

fig, ax = plot_confusion_matrix(conf, figsize=(4,4),
cmap=plt.cm.Greys)
```

```
plt.xlabel('Predictions', fontsize = 12)
plt.ylabel('Actuals', fontsize = 12)
plt.title('Confusion Matrix: Without PCA')
plt.figtext(0.5,0.97,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)


# ##### Evalution for PCA data

#spliting the dataset into training and testing set
X_ptrain, X_ptest, y_ptrain, y_ptest =
train_test_split(features_pca, target,
                                                    te
st_size=0.3, random_state=69)
#Creating a Logistic Regression Model
PcaModel = LogisticRegression()

#Training this model on original data
PcaModel.fit(X_ptrain, y_ptrain)

#Making prediction using this model on test set
y_ppred = PcaModel.predict(X_ptest)

#Generating a Classification Report
reportPca = classification_report(y_ptest, y_ppred)
confpca = confusion_matrix(y_ptest,y_ppred)
print('Classification Report:\n', reportPca)
print('Confusion Matix:\n', confpca)

fig, ax = plot_confusion_matrix(confpca,
figsize=(4,4), cmap=plt.cm.Greys)
plt.xlabel('Predictions', fontsize = 12)
plt.ylabel('Actuals', fontsize = 12)
plt.title('Confusion Matrix: With 6 PC(88.75%)')
plt.figtext(0.5,0.97,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)
plt.show()


# ##### Evaluation for 2 and 3 dim reduction

features_2 = Ybc_2

#spliting the dataset into training and testing set
X_ptrain, X_ptest, y_ptrain, y_ptest =
train_test_split(features_2, target,
                                                    te
st_size=0.3, random_state=69)
#Creating a Logistic Regression Model
PcaModel = LogisticRegression()

#Training this model on original data
PcaModel.fit(X_ptrain, y_ptrain)

#Making prediction using this model on test set
y_ppred = PcaModel.predict(X_ptest)

#Generating a Classification Report
reportPca = classification_report(y_ptest, y_ppred)
confpca = confusion_matrix(y_ptest,y_ppred)
print('Classification Report:\n', reportPca)
print('Confusion Matix:\n', confpca)

fig, ax = plot_confusion_matrix(confpca,
figsize=(4,4), cmap=plt.cm.Greys)
plt.xlabel('Predictions', fontsize = 12)
plt.ylabel('Actuals', fontsize = 12)
plt.title('Confusion Matrix: With 2 PC(63.24%)')
plt.figtext(0.5,0.97,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)
plt.show()


features_3 = Ybc_3
#spliting the dataset into training and testing set
X_ptrain, X_ptest, y_ptrain, y_ptest =
train_test_split(features_3, target,
                                                    te
st_size=0.3, random_state=69)
#Creating a Logistic Regression Model
PcaModel = LogisticRegression()

#Training this model on original data
PcaModel.fit(X_ptrain, y_ptrain)

#Making prediction using this model on test set
y_ppred = PcaModel.predict(X_ptest)

#Generating a Classification Report
reportPca = classification_report(y_ptest, y_ppred)
confpca = confusion_matrix(y_ptest,y_ppred)
print('Classification Report:\n', reportPca)
```

```
print('Confusion Matix:\n', confpca)

fig, ax = plot_confusion_matrix(confpca,
figsize=(4,4), cmap=plt.cm.Greys)
plt.xlabel('Predictions', fontsize = 12)
plt.ylabel('Actuals', fontsize = 12)
plt.title('Confusion Matrix: With 3 PC(72.63%)')
plt.figtext(0.5,0.97,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)
plt.show()


from sklearn.decomposition import PCA

pca = PCA(n_components=6)


pca.fit(features_nopca)


x_pcaSk = pca.transform(features_nopca)


x_pcaSk.shape


#spliting the dataset into training and testing set
X_train, X_test, y_train, y_test =
train_test_split(x_pcaSk, target,
                                                te
st_size=0.3, random_state=69)
#Creating a Logistic Regression Model
NoPcaModel = LogisticRegression()

#Training this model on original data
NoPcaModel.fit(X_train, y_train)

#Making prediction using this model on test set
y_pred = NoPcaModel.predict(X_test)

#Generating a Classification Report
report = classification_report(y_test, y_pred)
conf = confusion_matrix(y_test,y_pred)
print('Classification Report:\n', report)
print('Confusion Matrix:\n', conf)
```

```
fig, ax = plot_confusion_matrix(conf, figsize=(4,4),
cmap=plt.cm.Greys)
plt.xlabel('Predictions', fontsize = 12)
plt.ylabel('Actuals', fontsize = 12)
plt.title('Confusion Matrix: Sklearn PCA')
plt.figtext(0.5,0.97,'THA076BCT002 & THA076BCT007',
ha='center',fontsize=10)
```