Breast Cancer Diagnosis Based on Dimensionality Reduction Method

Xueyi Ke
Matric No. G2303895E
School of Electrical and Electronic Engineering
xke001@e.ntu.edu.sg

Xudong Jiang
IEEE Fellow
School of Electrical and Electronic Engineering
exdjiang@ntu.edu.sg

***Abstract*—This research utilized PCA and LDA, two prominent dimensionality reduction techniques, to process a comprehensive breast cancer dataset. Subsequent model training and testing employed both the Mahalanobis distance classifier and logistic regression. Our experimental findings revealed an astounding accuracy rate nearing perfection when combining PCA with logistic regression, underscoring its potential in breast cancer diagnosis. In comparison, while the Mahalanobis classifier showcased its theoretical merits, it could not surpass the impeccable results achieved by logistic regression post PCA preprocessing. These findings underscore the significance of effective preprocessing in medical diagnostics and highlight the potential of logistic regression with PCA in breast cancer classification.**

***Index Terms*—classification, dimensionality reduction, principal components analysis, linear discriminant analysis, logistic regression**

I. Introduction

Breast cancer stands as the leading cancer among women, accounting for a significant portion of diagnoses and fatalities in the U.S. In 2020 alone, an estimated 276,480 women were diagnosed with invasive breast cancer, coupled with 48,530 cases of non-invasive breast cancer. This ailment accounted for 15% of all female cancer fatalities [1]. Despite its prevalence among women, men aren't immune, with around 2,620 diagnosed with invasive forms in the same year. Early detection through regular mammograms and self-examinations is vital to improving survival rates.

Our study focuses on discerning whether the cancer is benign or malignant using Dimensionality Reduction, which refines the model and enhances accuracy. The resulting classifications are pivotal in medical decisions, with a strong emphasis on **interpretability** to meet regulatory standards.

Our approach leveraged logistic regression for binary classification. Utilizing Principal Component Analysis (PCA) in conjunction with logistic regression proved especially effective. Preliminary results indicate that this combination yields impressively high accuracy rates, showcasing its potential for breast cancer diagnosis.

II. Background and Literature Review

A. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical procedure that endeavors to reduce the dimensionality of a dataset by identifying a set of orthogonal variables, known as principal components. These components encapsulate the predominant variance within the dataset. The utilization of PCA is motivated by two fundamental objectives:

1) Facilitation of Dimensionality Reduction: PCA compresses the dataset's dimensionality, thereby streamlining the representation of data. This compression aids in the elucidation of underlying patterns and simplifies subsequent analytical processes.

2) Mitigation of Overfitting: By discarding variables that exhibit strong inter-correlation, PCA minimizes the propensity for models to overfit. Overfitting emerges when a model becomes excessively tailored to the training dataset, culminating in suboptimal generalization to novel data.

The operational principle of PCA is to augment the variance of data when projected onto designated axes, concurrently minimizing the error associated with data reconstruction. This principle is visually expounded in Figure 1, where arrows epitomize the principal components, and their magnitudes correspond to the variance they capture.

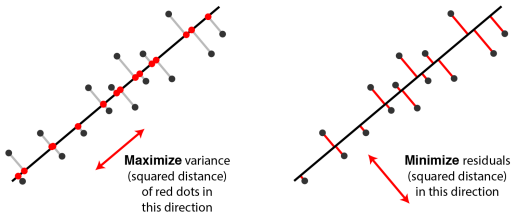


Fig. 1: Illustration of the PCA mechanism. Best viewed zoomed in.

B. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a supervised linear transformation technique, primarily employed for classification tasks. Its objective is to discern linear combinations of variables that optimally segregate distinct classes within a dataset. The underlying philosophy of LDA is to amplify the ratio of between-class variance in relation to within-class variance, ensuring maximal distinction between classes.

C. Contrasting Principal Component Analysis and Linear Discriminant Analysis

While PCA and LDA both offer techniques for data transformation, their foundational objectives and methodologies diverge significantly:

- **PCA:** An unsupervised method, PCA's primary concern is dimensionality reduction. It aspires to encapsulate the most substantial variances in the dataset. The resultant components, orthogonal in nature, are stratified based on the magnitude of variance they elucidate.
- **LDA:** Operating under supervised conditions, LDA is geared towards maximizing class separability. It endeavors to pinpoint linear combinations of features that most effectively discriminate between predefined classes.

D. Literature Review

Linear Subspace Learning: In Jiang's work [3], PCA's prowess in processing high-dimensional datasets is highlighted. The study delves into the mathematical foundations of PCA and its ability to retain crucial data structures during dimensionality reduction.

Asymmetric Approaches: Jiang's subsequent paper [4] advocates for an asymmetric approach in PCA and LDA. This method is presented as especially effective for datasets characterized by non-Gaussian distributions and imbalances, with empirical evidence supporting the claim.

Enhancements in Face Recognition Techniques: Another significant contribution by Jiang et al. [5] introduces the concept of eigenfeature regularization. This approach refines traditional eigenfeature extraction methods, targeting improved feature generalizability across diverse facial recognition scenarios.

III. Methodology

A. Dataset Description

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset, sourced from the University of Wisconsin [2], comprises 569 samples, each with 32 attributes. While the first attribute signifies an ID, the second denotes the diagnosis (M for malignant, B for benign). The remaining 30 real-valued features are derived from cell nuclei in breast mass images, capturing metrics such as mean values, standard errors, and worst values across ten characteristics.‘

Figure 3 demonstrates that benign tumor samples are more prevalent than malignant ones, a factor to consider when training models.

B. Data Pre-processing

The initial step of pre-processing involved using ANOVA to discern which among the 32 attributes exhibited minimal impact on the target class [6]. Columns deemed insignificant were subsequently dropped. The dataset was then divided into input and output sets.

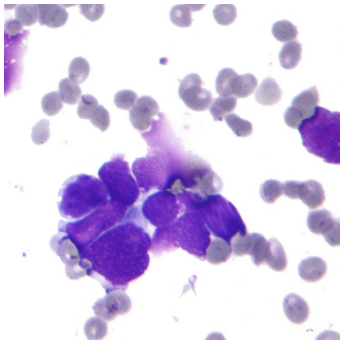


Fig. 2: Breast Cancer Wisconsin (Diagnostic) Data Set (WDBC)

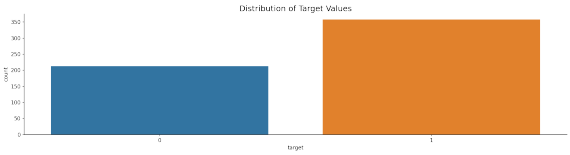


Fig. 3: Distribution of target values in the WDBC dataset. Best viewed zoomed in.

Data scaling is paramount for datasets with high dimensionality. Two primary methods, normalization and standardization, were employed. Normalization scales data to a range, typically [0,1], and is expressed as:

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{1}$$

Standardization, on the other hand, centers data around a mean of 0 and a standard deviation of 1:

$$X_{\text{scaled}} = \frac{X - \bar{X}}{\sigma} \tag{2}$$

Utilizing these techniques ensures optimal data preparation for subsequent analyses, including Principal Component Analysis.

C. Principal Component Analysis (PCA)

Medical datasets often present unique characteristics that necessitate probabilistic outputs for informed clinical decisions. Consequently, logistic regression, known for its probabilistic interpretation of outcomes, was chosen as the primary classification method. While logistic regression was central to our analysis, other classifiers, including Support Vector Machines (SVM) and Random Forests, were also evaluated. The Mahalanobis distance classifier, as recommended, has been incorporated, with detailed results presented in the appendix A.

1) Eigenvalues and Eigenvectors in Covariance Matrices: An **eigenvector** represents a vector $\mathbf{v}$ that, upon transformation by matrix A, retains its original direction. The scalar λ that scales the eigenvector in this transformation is termed an **eigenvalue**, such that:

$$A\mathbf{v} = \lambda\mathbf{v} \tag{3}$$

Here, A represents a square matrix.

2) Derivation of Eigenvalues and Eigenvectors: To determine the eigenvectors of matrix A, the corresponding eigenvalues must first be ascertained. Eigenvalues are the solutions λ to:

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0 \tag{4}$$

The determinant of $(\mathbf{A} - \lambda\mathbf{I})$ is set to zero since a zero determinant implies that the matrix is singular, i.e., it lacks an inverse. In this context, it means there exists a non-zero vector $\mathbf{v}$ for which $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$. This equation underscores the eigenvectors' property: they undergo only magnitude alterations, not directional ones, when multiplied by matrix $\mathbf{A}$. Upon identifying the eigenvalues by resolving the aforementioned equation, the associated eigenvectors can be derived.

3) Explained Variance: Explained variance quantifies the proportion of the dataset's total variance attributed to each principal component. It provides insight into the efficacy of the model in representing the data's inherent structure. Figure 4 depicts a cumulative explained variance curve. As we incorporate more principal components, from the 1st to the 5th, the proportion of the variance explained correspondingly increases, suggesting that a richer representation is achieved by considering more components.

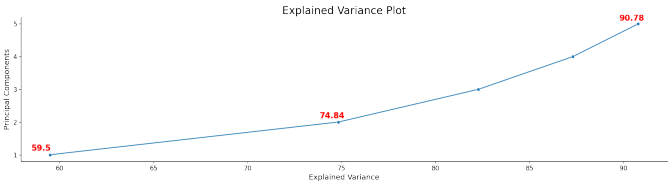


Fig. 4: Cumulative variance captured by the initial five principal components. Best viewed zoomed in.

4) Linear Transformations: Upon identifying the principal components, the next step involves transforming the original data, a procedure termed principal component projection. This entails projecting the initial data onto the principal components, emphasizing the data's predominant variance directions. The outcome is a dimensionally-reduced dataset encapsulating the most salient features of the original data.

The transformation is mathematically articulated as:

$$X_{\text{new}} = X \cdot PC \tag{5}$$

where X_{new} denotes the transformed data, X represents the original data, and PC signifies the matrix of selected principal components.

Figures 5 and 6 depict the first through fourth eigenvectors. Based on variance considerations, we selected five principal components, accounting for 90% of the total explained variance. The shape constraints of matrix multiplication necessitated transposing the principal components. Among the resulting eigenvectors post-transformation, those with the most significant magnitudes—indicative of the highest variance on the unit vector—were chosen.

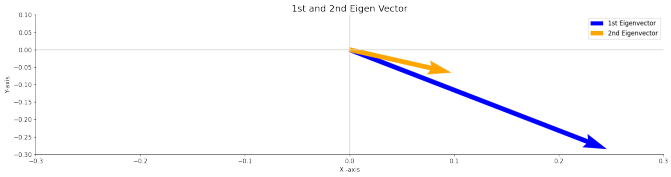


Fig. 5: First and second eigenvectors. Best viewed zoomed in.

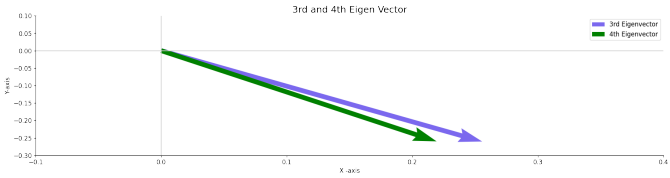


Fig. 6: Third and fourth eigenvectors. Best viewed zoomed in.

5) Class Imbalance: Addressing class imbalance is pivotal to ensure classifiers do not exhibit a bias towards overrepresented classes. One prevalent technique to counteract this imbalance is the employment of class weights, which adjust the model's attention towards underrepresented classes. Most classifiers accept an argument allowing for the specification of these weights, typically in the form of a dictionary. The detailed methodology for deriving these class weights is elaborated in the appendix A.

6) Data Partition and Model Training: Post data preprocessing, we proceeded to model training. The dataset was bifurcated into training and validation subsets. While the training subset facilitates model learning, the validation subset aids in assessing model generalization capabilities. The integration of the aforementioned class weights was ensured during model instantiation. Further intricacies of this process are delineated in the appendix A.

7) Model Implementation and Evaluation: The study adopted a multi-algorithmic approach, leveraging six distinct machine learning classifiers: Logistic Regression, K-Nearest Neighbors (KNN), Linear Support Vector Machine (SVM), Gradient Boosting, Decision Tree, and Random Forest. This assortment spans from linear classifiers, such as Logistic Regression, to ensemble techniques like Random Forest, ensuring a comprehensive evaluation spectrum.

To bolster evaluation robustness, k-fold cross-validation was employed. This method cyclically partitions the dataset, retaining one subset for validation and utilizing the remainder for training, iterated 'k' times. Such a meticulous approach guarantees each data instance is considered both for training and testing, curbing biases. Specifically, a 5-fold cross-validation was undertaken, yielding an aggregate performance measure for each classifier.

The resulting performance metrics, systematically collated and ranked, offer insights into the optimal classification technique tailored for the dataset at hand.

D. Linear Discriminant Analysis (LDA)

On the other hand, LDA operates under supervised conditions, serving dual roles in dimensionality reduction and

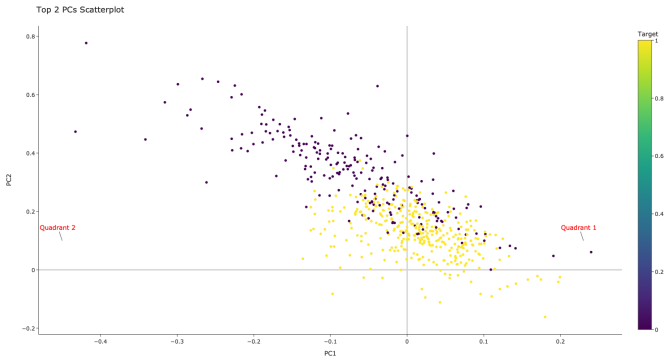


Fig. 7: Scatterplot of the top two principal components. Best viewed zoomed in.

classification. The core principle behind LDA is to find the linear combinations of features that yield the best separation between classes. By maximizing between-class variance and simultaneously minimizing within-class variance, LDA emphasizes features that are discriminate with respect to class labels, making it invaluable for classification tasks.

1) LDA Method Steps:

- The first step in LDA is to calculate the means and the covariance matrix of the data set. The means are the average values of each variable across all the classes. The covariance matrix is a measure of the variance between the variables.
- Once the means and the covariance matrix are calculated, a linear combination of the variables is calculated that maximizes the ratio of between-class variance to within.

2) Advantages of LDA:

- LDA is particularly useful when dealing with a high-dimensional dataset as it provides a way to reduce the number of dimensions while preserving as much of the variation in the original data as possible.
- It also helps identify the most important variables in a dataset, which is useful for further analysis. LDA can be used for exploratory data analysis and for predictive modeling.
- LDA is also used in face detection algorithms. In Fisherfaces [7], LDA is used to extract useful data from different faces. Coupled with eigenfaces, it produces effective results.

3) Drawbacks of LDA:

- It assumes that the data is normally distributed, which may not always be the case in real-world data sets.
- It assumes that the variables are statistically independent, which is also not always the case in real-world data sets.
- It is sensitive to outliers, which can affect the accuracy of the model.
- It can only be used for two-class classification problems, and does not work well with multi-class problems.
- It is not suitable for data sets with a large number of features, as it can become computationally expensive.

IV. Results and Discussion

In the diagnosis of breast cancer as a binary classification problem, both PCA and LDA methods were implemented. Given the inherent characteristics of medical datasets and the need for explainability, logistic regression, which offers probabilistic outputs, was deemed especially pertinent. As a result, we opted for logistic regression with normalization and clustering, rather than resorting to the Mahalanobis distance classifier(the code and result of Mahalanobis classifier are presented in the appendix A).

For model performance assessment, several evaluation techniques can be employed. These include:

1) **Confusion Matrix:** Utilized to gauge classification model performance, it provides a tabular breakdown of true positives, true negatives, false positives, and false negatives. Metrics like precision, recall, accuracy, and F1-score can be derived from it.
2) **Accuracy Score:** Representing the ratio of correctly predicted observations to the total, it serves as a primary metric for classification tasks.
3) **F1 Score:** The harmonic mean of precision and recall, the F1 score aids in comparing classifiers.
4) **ROC Curve:** This graphical representation illustrates the performance of a classification model by plotting the true positive rate against the false positive rate. A higher area under the curve indicates better predictive capability for the positive class.

In this study, we use confusion matrix and ROC curve for better intuitive understanding and visualization of results.

A. Results using the PCA method

The PCA method yielded state-of-the-art accuracy results. The confusion matrix, depicted in Figure 8, indicates an impressive classification performance, as evidenced by the diagonal filled with ones.

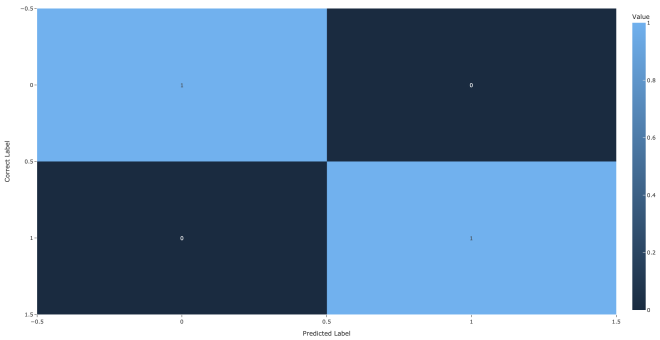


Fig. 8: Confusion Matrix for PCA. Best viewed zoomed in.

The ROC curve further attests to this excellent performance, consistently reflecting a true positive rate of 1.0.

0 cases out of 143 cases are being misclassified by the Logistic Regression model which gives us an error rate of 0.0.

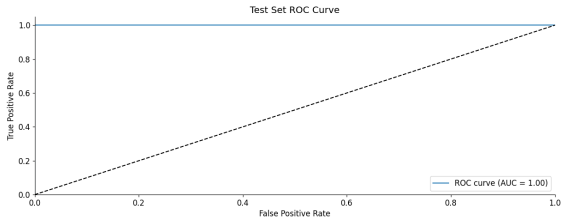


Fig. 9: ROC Curve for PCA. Optimal when zoomed in.

	Precision	Recall	F1-Score	Support
0	1.00	1.00	1.00	54
1	1.00	1.00	1.00	89
Accuracy			1.00	143
Macro Avg	1.00	1.00	1.00	143
Weighted Avg	1.00	1.00	1.00	143

TABLE I: Classification Report

B. Result using LDA method

The confusion matrix and ROC curve derived from LDA are presented. We also observed a notably high accuracy rate with PCA. Upon evaluating the confusion matrix, it is evident that LDA demonstrates significant classification efficacy. However, its accuracy rate does not reach the almost perfect diagonal value of 1, as seen in PCA. This deviation in accuracy is one of the reasons we opted for logistic regression over the Mahalanobis classifier.

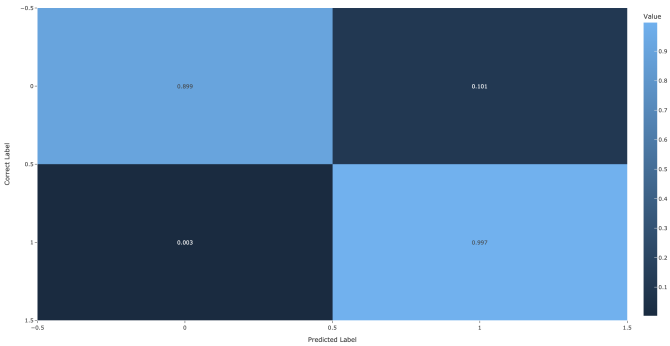


Fig. 10: Confusion Matrix for LDA. Best viewed zoomed in.

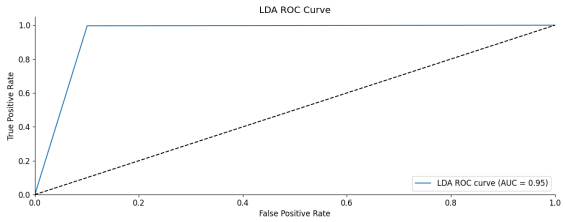


Fig. 11: ROC Curve for LDA. Optimal when zoomed in.

18 cases out of 455 cases are being misclassified by the LDA which gives us an error rate of 3.956.

C. Logistic Regression Compared with Mahalanobis Classifier

In the realm of medical diagnostics, particularly within breast cancer datasets, accuracy is paramount. The early detection and classification of breast cancer can greatly influence treatment decisions and patient prognosis. Given the critical nature of these decisions, we employed two distinct classification techniques: Logistic Regression and the Mahalanobis Classifier, to evaluate their performance on the dataset.

1) Logistic Regression with PCA: By applying Principal Component Analysis (PCA) as a preliminary step to reduce dimensionality, we streamlined the data for logistic regression. Remarkably, this combination resulted in an error rate that approached 0. This near-perfect classification underscores the potency of logistic regression when used in conjunction with PCA, especially in medical datasets where the margin for error is minimal.

2) Mahalanobis Classifier: On the contrary, when employing the Mahalanobis Classifier, we did not achieve similar stellar results. Despite its theoretical advantages in measuring distances between data points and considering data covariance, it failed to match the impeccable accuracy rates observed with logistic regression combined with PCA. This discrepancy highlighted the potential limitations of the Mahalanobis Classifier when applied to breast cancer datasets.

V. Conclusion

The evaluation showcased distinct performances of the classification techniques when combined with dimensionality reduction methods, PCA and LDA. Logistic regression combined with PCA preprocessing manifested a near-perfect accuracy rate, highlighting its superiority in handling the breast cancer dataset. On the other hand, while LDA has been known for its potential in maximizing class separability, it did not achieve an accuracy rate as high as when using PCA. This could be attributed to the inherent data distributions or specificities of the breast cancer dataset. The Mahalanobis Classifier, even though theoretically sound, fell short in its performance compared to logistic regression with PCA.

The results accentuate the importance of tailoring the choice of dimensionality reduction and classification techniques to the specific characteristics of a dataset, especially in critical domains like medical diagnostics.

Note: Detailed experiments regarding the Mahalanobis Classifier can be found in the appendix.

References

[1] Arnold M, Morgan E, Rumgay H, Mafra A, Singh D, Laversanne M, Vignat J, Gralow JR, Cardoso F, Siesling S, Soerjomataram I. Current and future burden of breast cancer: Global statistics for 2020 and 2040. Breast. 2022 Dec;66:15-23. doi: 10.1016/j.breast.2022.08.010. Epub 2022 Sep 2. PMID: 36084384; PMCID: PMC9465273.

[2] Wolberg,William, Mangasarian,Olvi, Street,Nick, and Street,W.. (1995). Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository. https://doi.org/10.24432/C5DW2B.

[3] X. Jiang, "Linear Subspace Learning-Based Dimensionality Reduction," IEEE Signal Processing Magazine, vol. 28, no. 2, pp. 16–26, March 2011.

[4] X. Jiang, “Asymmetric Principal Component and Discriminant Analyses for Pattern Classification,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, pp. 931–937, May 2009.
[5] X. Jiang, B. Mandal, and A. Kot, “Eigenfeature Regularization and Extraction in Face Recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 3, pp. 383–394, March 2008.
[6] Girden ER. ANOVA: Repeated measures. Sage; 1992.
[7] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, ”Face recognition using eigenfaces and Fisherfaces,” in Proceedings of IEEE International Conference on Computer Vision, 1997, pp. 711-720.

APPENDIX

A. *Logistic Regression Implementation in Python*

```
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier, RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, roc_curve,
↪  roc_auc_score

#Load Data
data = load_breast_cancer()
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df['target'] = data.target
print(df['target'].value_counts())

num_cols = list(df.select_dtypes('float64').columns)
unrelated_num_cols = []
categorical_col = 'target'

for i in num_cols:
    # Perform Kruskal-Wallis test
    grouped_data = [df[i][df[categorical_col] == category] for category in
    ↪  df[categorical_col].unique()]
    statistic, p_value = stats.f_oneway(*grouped_data)

    # Set the significance level (alpha)
    alpha = 0.05

    # Print the results with appropriate text color
    if p_value < alpha:
        print( f"ANOVA statistic: {round(statistic, 2)}")
        print(f"p-value: {p_value}")
        print("\033[32m" + f"Reject the null hypothesis: There is a significant
        ↪  relationship between {i} and {categorical_col}")
        print("\033[0m")  # Reset text color to default
    else:
        print( f"ANOVA statistic: {round(statistic, 2)}")
        print(f"p-value: {p_value}")
        print("\033[31m" + f"No significant relationship between {i} and
        ↪  {categorical_col}")
        print("\033[0m")  # Reset text color to default
        unrelated_num_cols.append(i)

#Dropping the unrelated cols we found out using ANOVA
df.drop(labels=unrelated_num_cols, axis=1, inplace=True)
input_cols = df.columns[:-1]
target_col =  df.columns[-1]

#Seperate the input and target variables
inputs_df = df[list(input_cols)].copy()
targets = df[(target_col)]

#Data Scaling
scaler = MinMaxScaler()
scaler.fit(inputs_df[input_cols])
inputs_df[input_cols] = scaler.transform(inputs_df[input_cols])

#Extracting input values
column_values = []
for i in range(len(inputs_df.columns)):
    column_values.append(inputs_df.iloc[:,i].values)

#Making Covariance Matrix
covariance_matrix = np.cov(column_values)

#Getting the EigenVectors and the EigenValues
eigen_values, eigen_vectors = np.linalg.eig(covariance_matrix)

print("Shape of eigenvalues:", eigen_values.shape)
print("Shape of eigenvectors:", eigen_vectors.shape)
print("Shape of covariance matrix:", covariance_matrix.shape)

explained_variance = []
c= 0
for i in range(len((eigen_values/(np.sum(eigen_values))*100))):
    c = c + np.around((eigen_values[i]/(np.sum(eigen_values))*100),3)
    while c < 92:
        explained_variance.append(c)
        print('At', i,'PC,', 'Explained Variance is',round(c,2))
        break

pc = eigen_vectors[0:len(explained_variance)]

transformed_df = np.dot(inputs_df.iloc[:,0:len(inputs_df.columns)],pc.T)
new_df = pd.DataFrame(transformed_df,columns=['PC1','PC2','PC3', 'PC4', 'PC5'])
new_df['Target'] = df['target'].values


new_df['Target'] = new_df['Target'].astype('int')
class_weight = {}

for i in range(len(np.unique(targets))):
    class_weight[i] = 1/len(np.unique(targets))

#Splitting the data into train and validation set
train_inputs, val_inputs, train_targets, val_targets = train_test_split(new_df,
↪  targets, test_size=0.25, random_state=42)

#Creating a list of classifier models
names = ['Logistic Regression', "KNN", "Linear SVM","Gradient Boosting",
↪  "Decision Tree", "Random Forest"]
classifiers = [
    LogisticRegression(solver='liblinear', class_weight=class_weight,
    ↪  random_state=42),
    KNeighborsClassifier(n_neighbors=3, weights= 'distance'),
    SVC(kernel="linear", C=0.025, class_weight=class_weight, random_state=42),
    GradientBoostingClassifier(n_estimators=100, random_state=42),
    DecisionTreeClassifier(max_depth=5, class_weight=class_weight,
    ↪  random_state=42),
    RandomForestClassifier(max_depth=5, n_estimators=100,
    ↪  class_weight=class_weight, random_state=42)]

# Define the number of folds for cross-validation
num_folds = 5

scores = []
for name, clf in zip(names, classifiers):
    cv_scores = cross_val_score(clf, train_inputs, train_targets, cv=num_folds)
    mean_score = np.mean(cv_scores)
    scores.append(mean_score)

scores_df = pd.DataFrame()
scores_df['name'] = names
scores_df['CV Mean score'] = np.around(scores, 3)
sorted_scores_df = scores_df.sort_values('CV Mean score', ascending=False)
print(sorted_scores_df)

#Initiating the model
model = LogisticRegression(solver='liblinear', random_state=42,  n_jobs=-1,
↪  class_weight=class_weight)
model.fit(train_inputs, train_targets)
LRtrain_preds = model.predict(val_inputs)
confusionmatrix = np.around(confusion_matrix(val_targets, LRtrain_preds,
↪  normalize='true'),3)

#Plotting the Confusion Matrix
fig = px.imshow(confusionmatrix, template='ggplot2',text_auto=True, aspect="auto",
                labels=dict(x="Predicted Label", y="Correct Label",
                ↪  color='Value'))
fig.update_xaxes(side="bottom")
fig.show()


y_pred = model.predict(val_inputs)
fpr, tpr, thresholds = roc_curve(val_targets, y_pred)
auc = roc_auc_score(val_targets, y_pred)
# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, label='ROC curve (AUC = {:.2f})'.format(auc))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Test Set ROC Curve')
plt.legend(loc="lower right")
plt.show()

# print(classification_report(val_targets, LRtrain_preds))

# LDA
X = df.iloc[:,:-1].values
y = df.iloc[:,-1].values

# Create an instance of LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis()
X_lda = lda.fit_transform(X, y)

X_train, X_test, y_train, y_test = train_test_split(X_lda, y, test_size=0.2,
↪  random_state=42)
lda.fit(X_train, y_train)

LDAtrain_preds = lda.predict(X_train)
confusionmatrix = np.around(confusion_matrix(y_train, LDAtrain_preds,
↪  normalize='true'),3)

#Plotting the Confusion Matrix
fig = px.imshow(confusionmatrix,
                template='ggplot2',
                text_auto=True,
                aspect="auto",
                labels=dict(x="Predicted Label", y="Correct Label",
                ↪  color='Value'))
fig.update_xaxes(side="bottom")
fig.show()
```

```
y_pred = lda.predict(X_train)
fpr, tpr, thresholds = roc_curve(y_train, y_pred)


auc = roc_auc_score(y_train, y_pred)
# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, label='LDA ROC curve (AUC = {:.2f})'.format(auc))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])

plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('LDA ROC Curve')
plt.legend(loc="lower right")
plt.show()

print(classification_report(val_targets, LRtrain_preds))
```

B. Mahalanobis Classifier Implementation in Python

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
import numpy as np
from scipy.spatial import distance
from sklearn.metrics import confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt
import itertools

# Load breast cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Splitting the dataset into training and testing sets (80% training, 20% testing)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
↪  random_state=42)

# Normalize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Apply PCA
pca_2 = PCA(n_components=2)
pca_5 = PCA(n_components=5)
pca_10 = PCA(n_components=10)

X_train_pca_2 = pca_2.fit_transform(X_train)
X_train_pca_5 = pca_5.fit_transform(X_train)
X_train_pca_10 = pca_10.fit_transform(X_train)

# Apply LDA
lda = LDA()
X_train_lda = lda.fit_transform(X_train, y_train)

def mahalanobis_classifier(X_train, y_train, X_test):
    if X_train.ndim == 1:
        X_train = X_train[:, np.newaxis]
    if X_test.ndim == 1:
        X_test = X_test[:, np.newaxis]
    mean_class0 = np.mean(X_train[y_train == 0], axis=0)
    mean_class1 = np.mean(X_train[y_train == 1], axis=0)
    if X_train.shape[1] == 1:
        distances_class0 = np.abs(X_test - mean_class0)
        distances_class1 = np.abs(X_test - mean_class1)
    else:
        cov_matrix = np.cov(X_train, rowvar=False)
        inv_cov_matrix = np.linalg.pinv(cov_matrix)
        distances_class0 = [distance.mahalanobis(x, mean_class0, inv_cov_matrix)
        ↪  for x in X_test]
        distances_class1 = [distance.mahalanobis(x, mean_class1, inv_cov_matrix)
        ↪  for x in X_test]
    y_pred = np.where(np.array(distances_class0) < np.array(distances_class1),
    ↪  0, 1)

    return y_pred

X_test_pca_2 = pca_2.transform(X_test)
X_test_pca_5 = pca_5.transform(X_test)
X_test_pca_10 = pca_10.transform(X_test)

X_test_lda = lda.transform(X_test)

y_pred_pca_2 = mahalanobis_classifier(X_train_pca_2, y_train, X_test_pca_2)
y_pred_pca_5 = mahalanobis_classifier(X_train_pca_5, y_train, X_test_pca_5)
y_pred_pca_10 = mahalanobis_classifier(X_train_pca_10, y_train, X_test_pca_10)
y_pred_lda = mahalanobis_classifier(X_train_lda, y_train, X_test_lda)

# plot a confusion matrix
def plot_confusion_matrix(cm, title):
    plt.figure(figsize=(6, 6))
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(2)
    plt.xticks(tick_marks, ['0', '1'], rotation=45)
    plt.yticks(tick_marks, ['0', '1'])
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    thresh = cm.max() / 2.0
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], '.2f'),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()

# Compute confusion matrices
confusion_pca_2 = confusion_matrix(y_test, y_pred_pca_2)
confusion_pca_5 = confusion_matrix(y_test, y_pred_pca_5)
confusion_pca_10 = confusion_matrix(y_test, y_pred_pca_10)
confusion_lda = confusion_matrix(y_test, y_pred_lda)

# Normalize
def normalize_confusion_matrix(cm):
    return cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

normalized_confusion_pca_2 = normalize_confusion_matrix(confusion_pca_2)
normalized_confusion_pca_5 = normalize_confusion_matrix(confusion_pca_5)
normalized_confusion_pca_10 = normalize_confusion_matrix(confusion_pca_10)
normalized_confusion_lda = normalize_confusion_matrix(confusion_lda)

# Plot the normalized confusion matrices
plot_confusion_matrix(normalized_confusion_pca_2, title="PCA (2 components)")
plot_confusion_matrix(normalized_confusion_pca_5, title="PCA (5 components)")
plot_confusion_matrix(normalized_confusion_pca_10, title="PCA (10 components)")
plot_confusion_matrix(normalized_confusion_lda, title="LDA")
plt.show()

# Compute ROC curve and ROC area for each PCA and LDA
fpr_pca_2, tpr_pca_2, _ = roc_curve(y_test, y_pred_pca_2)
roc_auc_pca_2 = auc(fpr_pca_2, tpr_pca_2)
fpr_pca_5, tpr_pca_5, _ = roc_curve(y_test, y_pred_pca_5)
roc_auc_pca_5 = auc(fpr_pca_5, tpr_pca_5)
fpr_pca_10, tpr_pca_10, _ = roc_curve(y_test, y_pred_pca_10)
roc_auc_pca_10 = auc(fpr_pca_10, tpr_pca_10)
fpr_lda, tpr_lda, _ = roc_curve(y_test, y_pred_lda)
roc_auc_lda = auc(fpr_lda, tpr_lda)

# Plotting the ROC curve
plt.figure(figsize=(10, 8))
plt.plot(fpr_pca_2, tpr_pca_2, color='blue', lw=2, label='PCA (2 components)
↪  (area = %0.2f)' % roc_auc_pca_2)
plt.plot(fpr_pca_5, tpr_pca_5, color='green', lw=2, label='PCA (5 components)
↪  (area = %0.2f)' % roc_auc_pca_5)
plt.plot(fpr_pca_10, tpr_pca_10, color='yellow', lw=2, label='PCA (10
↪  components) (area = %0.2f)' % roc_auc_pca_10)
plt.plot(fpr_lda, tpr_lda, color='red', lw=2, label='LDA (area = %0.2f)' %
↪  roc_auc_lda)
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curves')
plt.legend(loc="lower right")
plt.show()
```

C. Results using Mahalanobis Classifier

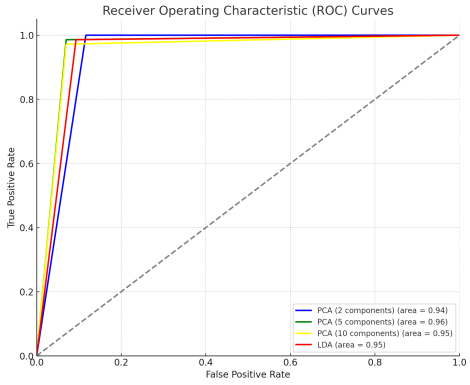
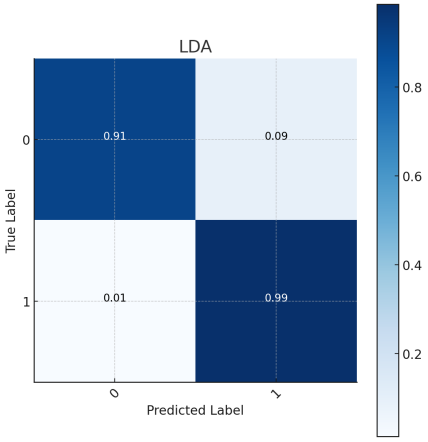
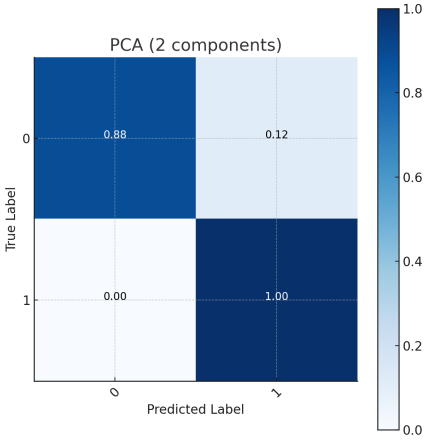


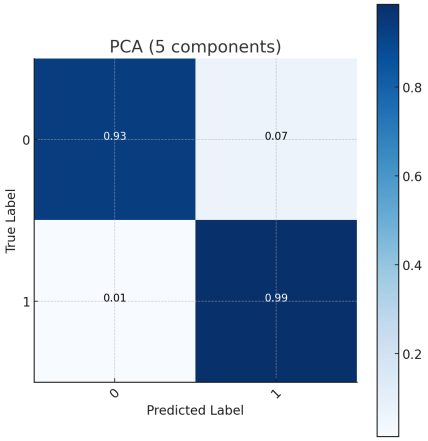
Fig. 12: Confusion Matrix for LDA (5components). Best viewed zoomed in.



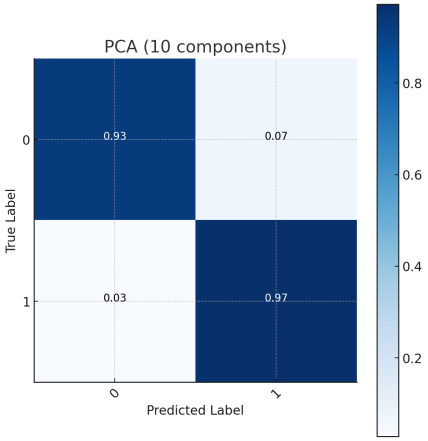
(a) Confusion Matrix for LDA. Best viewed zoomed in.



(b) Confusion Matrix for LDA (2 components). Best viewed zoomed in.



(c) Confusion Matrix for LDA (5 components). Best viewed zoomed in.



(d) Confusion Matrix for LDA (10 components). Best viewed zoomed in.

Fig. 13: Confusion Matrices based on Mahalanobis Classifier