A Comparative Analysis of Machine Learning Techniques with Principal Component Analysis for Diabetes Diagnosis

Siva Anand
Concordia ID: 40279665
Github link : https://github.com/Siva-Anand006/INSE-6220-Final_Project/blob/main/6220_final_project.ipynb

***Abstract*—Diabetes mellitus remains a prevalent and challenging health problem worldwide, underscoring the significance of accurate and efficient diagnostic approaches. In this study, we investigate how the merging of Principal Component Analysis (PCA) with machine learning techniques can enhance the detection of diabetes. We utilize PCA on a dataset containing diverse physiological attributes like glucose levels, blood pressure, and body mass index to reduce dimensionality and pinpoint crucial features essential for diagnosis.By comparing different machine learning algorithms like Naive Bayes, Logistic Regression, and Linear Discriminant Analysis, we assess how PCA affects classification accuracy and performance. Findings suggest that using PCA for dimensionality reduction can greatly enhance the accuracy and efficiency of diabetes diagnosis, with some algorithms like Naive Bayes showing better performance after the transformation. In addition, we evaluate how effective evaluation metrics like precision, recall, and F1-score are in measuring classification performance, offering understanding about the advantages and disadvantages of each method.**

Keywords—Principal Component Analysis, PCA, dimensionality reduction, medical diagnostics, diabetic classification, Linear Discriminant Analysis,hyperparameter tuning, performance metrics, F1 score, confusion matrix, decision boundaries, Naive Bayes, Logistic Regression, machine learning

I. INTRODUCTION

In the medical diagnostics field, the incorporation of machine learning methods has brought about a significant transformation, providing new possibilities for early disease detection and precise disease classification. In the wide range of machine learning techniques available, Principal Component Analysis (PCA) is highlighted as a strong method for reducing dimensionality, especially useful for managing the extensive and intricate datasets often seen in medical research.

The main goal of this report is to examine how effective PCA is in diabetic classification. Diabetes mellitus, a long-term metabolic condition marked by high levels of blood sugar, presents major public health issues worldwide. Detecting diabetes early and accurately categorizing cases is essential for successful disease control and treatment.

In order to reach this goal, we utilize PCA on a dataset containing numerous related variables linked to diabetic and non-diabetic people. PCA allows the conversion of this dataset with many dimensions into a space with fewer dimensions, all while retaining important information, making data analysis and interpretation more effective.

Next, we utilize two different classification methods, Linear Discriminant Analysis and Decision Tree Analysis, on both the initial and PCA-transformed data sets. Our goal is to evaluate how reducing dimensionality through PCA affects the accuracy and robustness of diabetic classification by comparing algorithm performance pre- and post-transformation.

We also fine-tune model hyperparameters to enhance the performance of metrics like F1 score and confusion matrix. Seeing the decision boundaries visually helps in comprehending how well the model fits the data, giving insights into its ability to discriminate.

A notable result of the study is the enhanced performance of the Naive Bayes classifier after undergoing PCA transformation, underscoring the effectiveness of PCA in improving classification accuracy.

II. PRINCIPAL COMPONENT ANALYSIS

PCA is a commonly employed method for reducing dimensions in the fields of machine learning and statistics. Its goal is to convert a dataset with possibly related variables into a fresh group of variables that are not correlated, which are referred to as principal components. The main components are created by combining the original variables in a linear way and are ranked based on how much variance they account for in the data.

The basic concept of PCA is to reduce the dimensions of a dataset to retain crucial information, simplifying analysis while preserving variability. PCA makes it easier to visualize, interpret, and analyze data by decreasing its dimensionality.

A. PCA Algorithm

The PCA method can be utilized on a data set X with a size of n × p by following these steps:

1) **Standardization:** The first stage includes normalizing the variables to guarantee they have the same impact on the analysis. The average of all values in each column of the dataset is calculated to find the mean vector $\bar{x}$ for that column.

   In mathematical terms, the mean vector $\bar{x}$ is formally described as

   $$\bar{x} = \frac{1}{n} \times \sum_{i=1}^{n} x_i \quad (1)$$

   The next step involves standardizing the data by subtracting the average of each column from each element in the data matrix. The centered data matrix Y can be written as:

   $$Y = H \times X \quad (2)$$

where H is the matrix used for centering.

2) **Covariance Matrix Computation :** Computing the covariance matrix allows us to understand the underlying relationships between variables and provides essential information for deriving the principal components through Eigen decomposition, enabling dimensionality reduction and data transformation in PCA.

The p × p covariance matrix is computed as follows:

$$S = \frac{1}{n-1} Y^T \times Y \quad (3)$$

3) Eigen Decomposition : It is possible to calculate the eigenvalues and eigenvectors of S. Eigenvectors represent the direction of each principal component (PC) whereas eigenvalues represent the variance captured by each PC.

$$S = A \wedge A^T \quad (4)$$

where A means the p × p orthogonal matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues.

4) **Principal components:** It computes the transformed matrix Z that is the size of n×p. The rows of Z represent the observations and columns of Z represent the PCs. The number of PCs is equal to the dimension of the original data matrix. The equation of Z can be given by:

$$Z = Y A \quad (5)$$

B. Logistic Regression (LR)

Logistic Regression (LR) is a statistical technique employed for assessing the connection between a binary result variable and multiple predictor variables.LR aims to create a model that effectively links input features with class labels in classification tasks.

LR places samples into one of two categories, typically labeled as either 1 or 0. The logistic function transforms input data into a probability value between 0 and 1 for classification purposes. If the probability score exceeds a certain threshold (usually 0.5), the sample is classified as belonging to the positive category (1); if not, it is classified as part of the negative category (0).

C. Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a method of supervised classification that is utilized to discover a linear blend of characteristics that effectively distinguishes between two or more classes of entities or occurrences. In contrast to Logistic Regression, which predicts the probability of each class, LDA aims to enhance class separation by identifying the linear discriminants that effectively distinguish between classes.

The objective of LDA is to reduce the dimensionality of input features by optimizing between-class variance and minimizing within-class variance. This is accomplished by calculating the eigenvectors and eigenvalues of the covariance matrix of the input characteristics. The discriminant directions, also called eigenvectors, show the best axes to separate the data, while the eigenvalues measure the variance explained by each discriminant.

The threshold value defines the linear decision boundary in LDA. Examples are grouped into various categories depending on the location on either side of the decision boundary following projection onto the discriminatory axes.

D. Naive Bayes

Naive Bayes is a straightforward but effective classification algorithm that is rooted in Bayes' theorem. Despite being simple, Naive Bayes is effective in practice and commonly applied in text classification, spam filtering, and medical diagnosis.

Naive Bayes determines the likelihood of each class based on input features, choosing the class with the highest probability as the predicted outcome. This computation is carried out with the help of Bayes' theorem, which states that the probability of a class after observing the input features is proportional to the product of the class' prior probability and the probability of the features given the class, divided by the overall probability of the features.

III. DATA SET DESCRIPTION

The dataset used in this project for analysis is the diabetes diagnosis dataset. This data set is organized to establish if someone has diabetes or not, offering important information for diagnostic reasons. Consisting of 768 entries, the dataset includes a wide range of features aimed at aiding in precise diagnosis.

In particular, the dataset displays 8 unique characteristics:

1. Pregnancies: Represents the total number of pregnancies the person has experienced.
2. Glucose: Indicates the person's level of glucose concentration.
3. BloodPressure: Represents the person's blood pressure reading.
4. SkinThickness: Indicates the thickness of the person's skin.
5. Insulin: Shows the amount of insulin in the person's system.
6. BMI (Body Mass Index): Is a measure of a person's body weight relative to their height, calculated using weight and height.
7. DiabetesPedigreeFunction: Indicates the level of genetic predisposition to diabetes within a family.
8. Age: Indicates the person's age.

figure.1 : Pair Plot

These characteristics together give a complete perspective on different physiological aspects that are important for diagnosing diabetes. Additionally, the dataset includes an essential column named "Outcome," which acts as the class label to determine if a person has been diagnosed with diabetes. Through the utilization of this dataset, our analysis seeks to utilize machine learning algorithms in order to effectively categorize diabetic cases and aid in the progression of medical diagnostic processes.

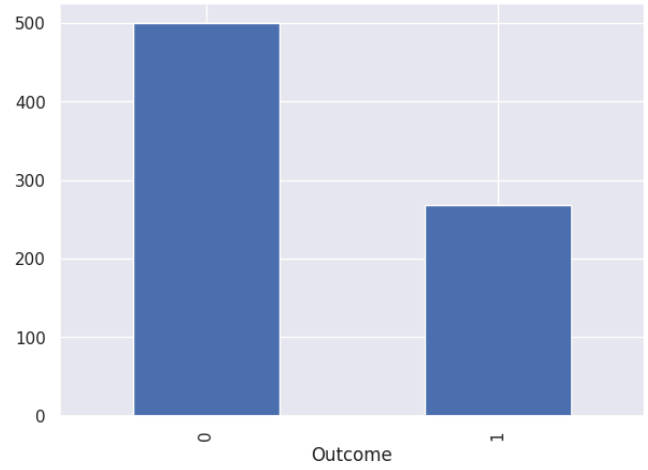

figure.1 : Outcome Plot

Utilizing the box and whisker plots and their five number summaries on dataset, the distributions, central values and variability of the features were measured. Fig. 1 illustrates the box plot of the features of diabetes dataset. It can be observed from Fig. 1 that most of the features have outliers.

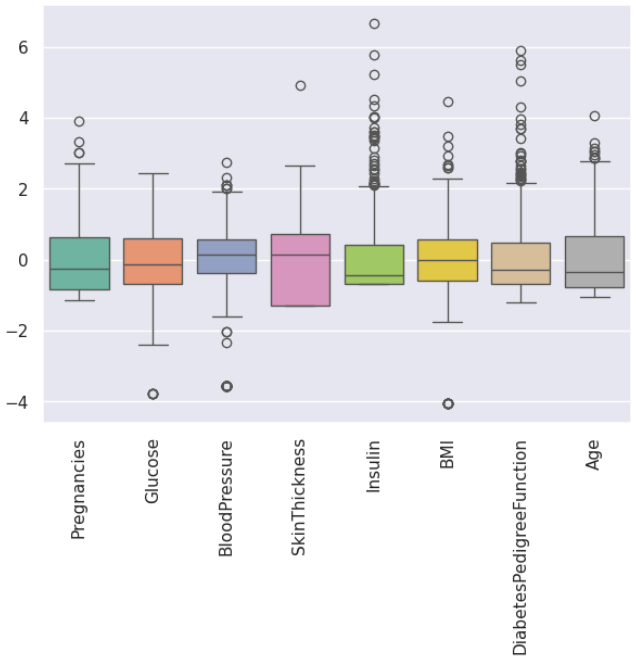

figure.2 : Box Plot

The relationships between different characteristics in a diabetes dataset are displayed in the correlation matrix. Every number in the matrix indicates the correlation coefficient of a pair of features.

The correlation coefficient varies between -1 and 1, with: A score of 1 denotes a complete positive correlation. As one feature's value rises, the other feature's value also rises.
A score of -1 represents a complete negative relationship. As one feature's value goes up, the value of the other feature goes down. A value of 0 indicates there is no relationship between the variables. There is no correlation between the alterations in the two characteristics.

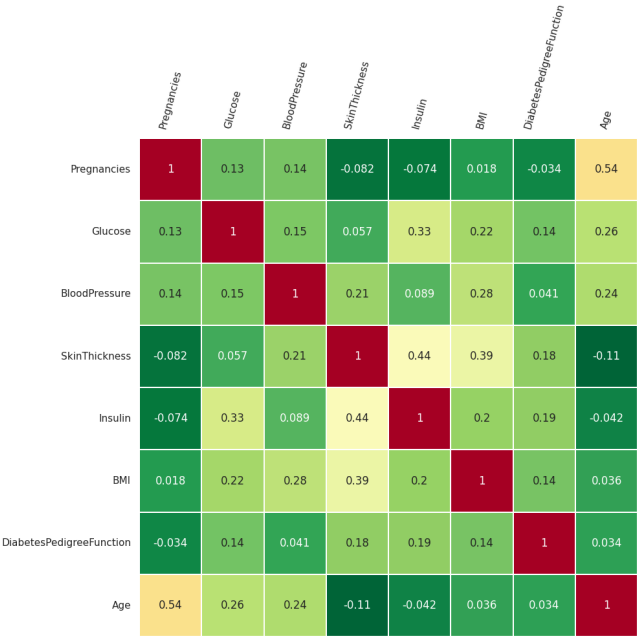

figure.3 :Correlation Matrix

From the correlation matrix, one can analyze the following observations. Age is moderately correlated with number of pregnancies (0.54) and glucose levels (0.26). Body mass index (BMI) is slightly positively correlated with glucose (0.22) and insulin (0.2).
There is a moderate positive correlation between insulin and skin thickness, with a correlation coefficient of 0.44. Blood pressure is slightly positively correlated with BMI (0.28) and age (0.24).

It is crucial to understand that correlation does not indicate causation. The correlation between two features does not imply causation.

IV. PCA RESULTS

Principal Component Analysis (PCA) is employed on the diabetes dataset for dimensionality reduction and feature extraction. There are two primary approaches to implement PCA: (1) developing PCA from scratch using standard Python libraries like NumPy, and (2) utilizing established PCA libraries that offer pre-implemented functions and streamlined workflows. In the Google Colab notebook accompanying this report, both implementation methods are showcased.

Although the results obtained from both approaches are comparable in terms of accuracy and effectiveness, employing a dedicated PCA library offers distinct advantages. These libraries provide users with enhanced flexibility and convenience, often requiring only a single line of code to execute complex PCA operations. Moreover, they come with extensive documentation and community support, facilitating easier implementation and troubleshooting.

In this report, the figures and plots presented are generated from the implementation utilizing a PCA library.
Figures 4 and 5 depict the scree plot and pareto plot, respectively, illustrating the variance explained by each principal component (PC). These plots offer valuable insights into the contribution of individual PCs to the overall variability within the dataset.

The scree plot and pareto plot serve as graphical representations of the eigenvalues associated with each PC. They allow us to visualize the proportion of variance explained by each PC, aiding in the determination of the optimal number of principal components to retain for dimensionality reduction.

In particular, Percentage of Variance Explained by j-th PC

$$Percentage\ of\ Explained\ variance = \frac{\lambda j}{\sum_{i=1}^{p} \lambda_i} \qquad (6)$$

where λ_j represents the eigenvalue corresponding to the j-th PC, and p denotes the total number of principal components.

By analyzing these plots, we can identify the principal components that capture the most significant amount of variance in the dataset, thereby guiding the selection of the optimal number of PCs for subsequent analysis and interpretation.

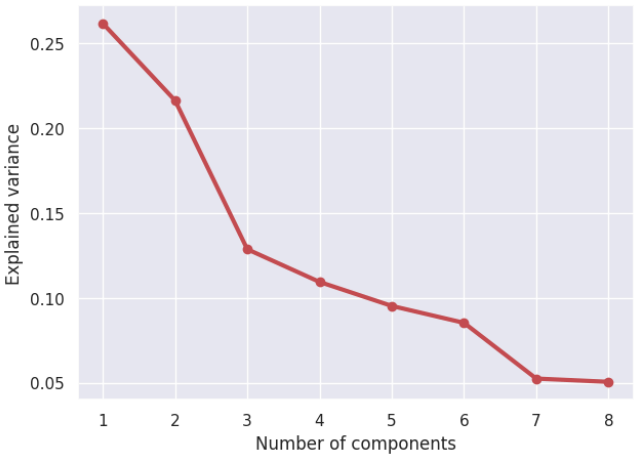

Fig. 4: Scree Plot

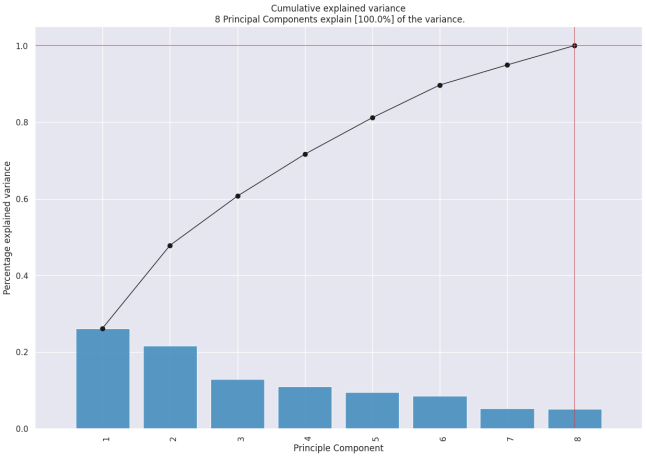

Fig. 5: Pareto Plot

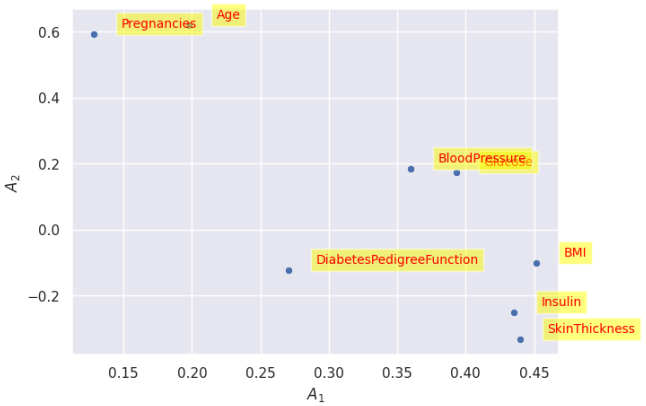

Fig. 6: PC coefficient plot

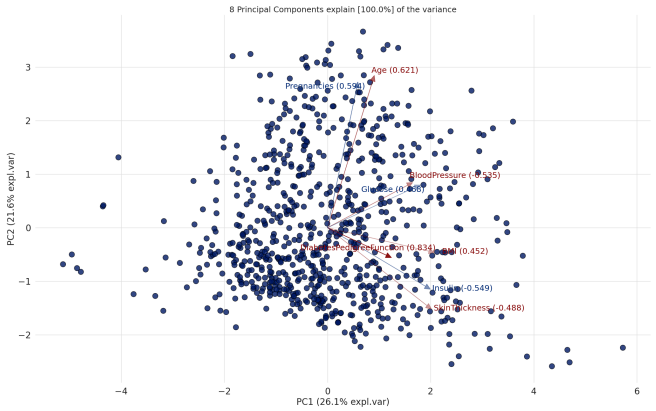

Fig. 7: Biplot

Figure 6 illustrates the PC coefficient plot, providing a visual depiction of the contribution of each feature to the first two principal components (PCs). This plot enables us to discern the relative importance of each feature in defining the principal components and, consequently, in capturing the underlying structure of the data.

On the other hand, Figure 7 presents a biplot, offering an alternative visual representation of the first two PCs. In the biplot, the axes correspond to the first two principal components, allowing for a comprehensive visualization of the relationships between both observations and features in the reduced-dimensional space. Additionally, the biplot displays the direction and magnitude of each feature's contribution to the principal components, aiding in the interpretation of the underlying structure of the data.

V. CLASSIFICATION RESULTS

In this section, we delve into the performance evaluation of three widely-used classification algorithms on the Diabetes dataset. To assess the impact of Principal Component Analysis (PCA) on the dataset, these classification algorithms are applied to both the original dataset and the dataset after PCA transformation, with three principal components extracted. The classification tasks are executed utilizing the PyCaret library in Python.

To ensure robustness and reproducibility, the original dataset is split into training and testing sets in a 70:30 ratio. The session id is set to 123 for consistency across iterations.

Through PyCaret, a comprehensive performance comparison table is generated, showcasing the accuracy metrics of all available classification algorithms on the target dataset. This enables the identification of the optimal model with the highest accuracy.

Analysis of the performance comparison table (Fig. 8) reveals that, prior to applying PCA, the top-performing classification models are Linear Discriminant Analysis and Logistic Regression, boasting the highest accuracies.

Subsequently, after PCA application, the comparison among classification models is presented in Fig. 9. Notably, the analysis highlights Naive Bayes as the standout model, exhibiting the highest accuracy among all models considered.

These findings underscore the efficacy of PCA in enhancing classification accuracy, with Naive Bayes emerging as the optimal choice for diabetes classification post-PCA transformation.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
lda	Linear Discriminant Analysis	0.7807	0.8406	0.5286	0.6899	0.5792	0.4114	0.4288
lr	Logistic Regression	0.7483	0.8100	0.5286	0.6869	0.5769	0.4073	0.4256
ridge	Ridge Classifier	0.7483	0.8106	0.5219	0.6882	0.5747	0.4056	0.4234
nb	Naive Bayes	0.7461	0.7890	0.5638	0.6521	0.6000	0.4213	0.4296
rf	Random Forest Classifier	0.7390	0.7808	0.5290	0.6415	0.5730	0.3901	0.3974
et	Extra Trees Classifier	0.7344	0.7685	0.4757	0.6776	0.5446	0.3686	0.3875
ada	Ada Boost Classifier	0.7227	0.7594	0.5219	0.6224	0.5555	0.3599	0.3704
gbc	Gradient Boosting Classifier	0.7182	0.7866	0.5090	0.6067	0.5408	0.3445	0.3542
lightgbm	Light Gradient Boosting Machine	0.7157	0.7847	0.5024	0.6057	0.5389	0.3395	0.3478
xgboost	Extreme Gradient Boosting	0.7134	0.7629	0.4952	0.6041	0.5364	0.3348	0.3418
knn	K Neighbors Classifier	0.7133	0.7034	0.4619	0.6088	0.5086	0.3184	0.3308
qda	Quadratic Discriminant Analysis	0.7087	0.7705	0.5157	0.5943	0.5389	0.3316	0.3406
dummy	Dummy Classifier	0.6574	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
dt	Decision Tree Classifier	0.6292	0.5791	0.4210	0.4540	0.4314	0.1601	0.1612
svm	SVM - Linear Kernel	0.5616	0.5552	0.3171	0.2037	0.2195	-0.0052	-0.0224

Fig. 8: Comparison among classification models before applying PCA

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
nb	Naive Bayes	0.7319	0.7574	0.4338	0.6863	0.5226	0.3496	0.3724
lda	Linear Discriminant Analysis	0.7296	0.7564	0.4481	0.6729	0.5300	0.3511	0.3701
lr	Logistic Regression	0.7273	0.7552	0.4414	0.6707	0.5248	0.3449	0.3643
ridge	Ridge Classifier	0.7226	0.7564	0.4276	0.6646	0.5114	0.3307	0.3514
qda	Quadratic Discriminant Analysis	0.7203	0.7543	0.4343	0.6527	0.5148	0.3290	0.3467
gbc	Gradient Boosting Classifier	0.6877	0.7101	0.4357	0.5560	0.4836	0.2677	0.2731
knn	K Neighbors Classifier	0.6853	0.6996	0.4557	0.5540	0.4932	0.2702	0.2763
rf	Random Forest Classifier	0.6784	0.6997	0.4081	0.5503	0.4593	0.2410	0.2503
et	Extra Trees Classifier	0.6783	0.6956	0.3929	0.5458	0.4474	0.2333	0.2425
ada	Ada Boost Classifier	0.6761	0.6946	0.4024	0.5310	0.4527	0.2333	0.2382
svm	SVM - Linear Kernel	0.6691	0.7179	0.6293	0.4620	0.4479	0.2377	0.2424
dummy	Dummy Classifier	0.6574	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
xgboost	Extreme Gradient Boosting	0.6528	0.6553	0.4300	0.4901	0.4545	0.2041	0.2059
lightgbm	Light Gradient Boosting Machine	0.6481	0.6606	0.3890	0.4834	0.4225	0.1788	0.1834
dt	Decision Tree Classifier	0.5921	0.5476	0.4071	0.3999	0.3997	0.0933	0.0936

Fig. 9: Comparison among classification models after applying PCA

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.7907	0.7738	0.4667	0.8750	0.6087	0.4833	0.5278
1	0.7442	0.7119	0.3333	0.8333	0.4762	0.3458	0.4094
2	0.6512	0.7762	0.4667	0.5000	0.4828	0.2201	0.2204
3	0.7442	0.7071	0.4667	0.7000	0.5600	0.3897	0.4056
4	0.7674	0.7857	0.5333	0.7273	0.6154	0.4543	0.4655
5	0.8605	0.9190	0.6667	0.9091	0.7692	0.6726	0.6892
6	0.6279	0.6548	0.3333	0.4545	0.3846	0.1269	0.1300
7	0.7209	0.6601	0.4286	0.6000	0.5000	0.3138	0.3224
8	0.7442	0.7512	0.2857	0.8000	0.4211	0.3013	0.3672
9	0.7143	0.8342	0.3571	0.6250	0.4545	0.2800	0.3001
Mean	0.7365	0.7574	0.4333	0.7024	0.5272	0.3508	0.3838
Std	0.0626	0.0762	0.1074	0.1479	0.1078	0.1445	0.1497

Fig. 10: NB metrics score after hyperparameter tuning

Both the original and transformed datasets undergo training, tuning, and evaluation using the three classification algorithms. While both experiments are detailed in the Google Colab notebook, this report concentrates solely on the results obtained after applying PCA to the transformed dataset.

To optimize model performance, hyperparameter tuning plays a crucial role. In PyCaret, hyperparameter tuning involves three key steps: creating the model, tuning it, and evaluating its performance. Initially, a classification model is generated for each algorithm. Subsequently, the tune_model() function is employed to fine-tune the model with optimal hyperparameters. This function automatically tunes the model using a predefined search space and assesses its performance via stratified K-fold cross-validation. By default, PyCaret employs 10-fold stratified K-fold validation for the three algorithms, ensuring robust evaluation of model performance.

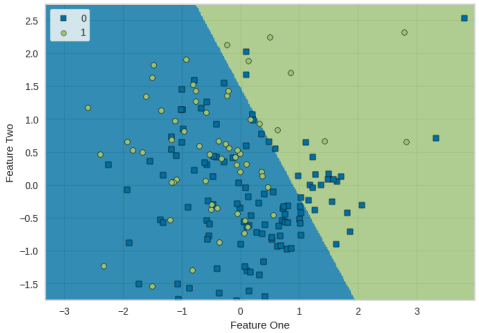

Fig. 11(a) : Decision Boundary LDA

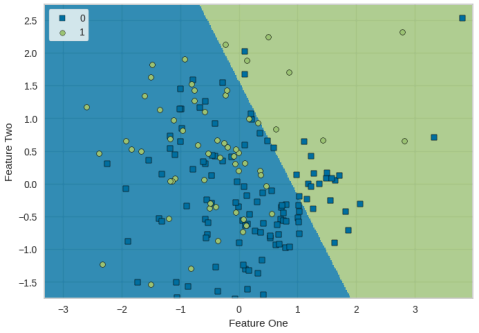

Fig. 11(b) : Decision Boundary LR

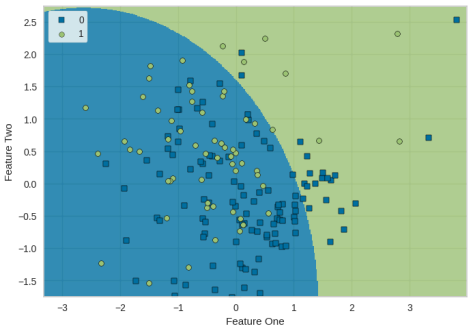

Fig. 11(c) : Decision Boundary NB

Figures 11(a)(b)(c) visually represents the decision boundaries generated by the model on the transformed dataset. A decision boundary can be conceptualized as a hyperplane that delineates different classes within the dataset, effectively partitioning data points into distinct categories. As data points approach the decision boundary, the algorithm discerns which class they belong to, enabling accurate classification.

The first principal component (PC) corresponds to the x-axis in Figure 11, and the second principal component corresponds to the y-axis. This visualization provides a useful understanding of how the model organizes and classifies data points using their principal components. Analyzing the decision boundaries helps us comprehend the model's discriminative abilities and its success in classifying instances in the modified dataset.

Since the diabetes dataset is a binary classification problem, evaluating precision and recall for each class individually provides valuable insights into the performance of the classification models. Precision and recall are two essential metrics used to assess the effectiveness of classification algorithms.

Precision is defined as the fraction of relevant instances among all retrieved instances, whereas recall represents the fraction of retrieved instances among all relevant instances.

Figure 12 shows precision and recall metrics results using confusion matrices. A confusion matrix is a table that displays the comparison between predicted and actual class instances. It outlines accurate and inaccurate forecasts, separated for each category. Figure 12 displays confusion

matrix tables for the three algorithms used on the modified dataset. These tables provide a detailed analysis of the algorithms' classification performance by showing the breakdown of true positives, false positives, true negatives, and false negatives for each class.

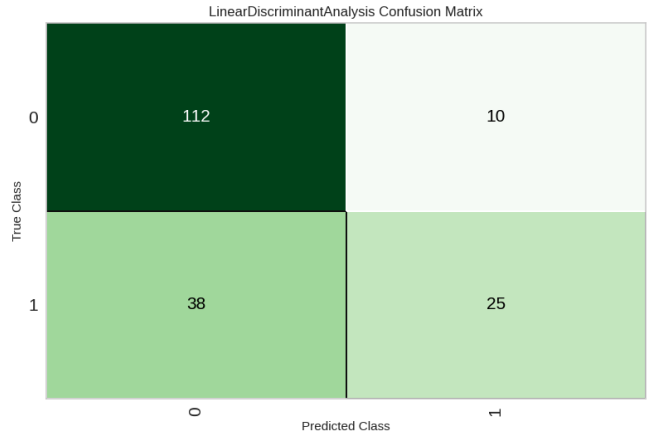

Fig. 12(a) : Confusion matrix LDA

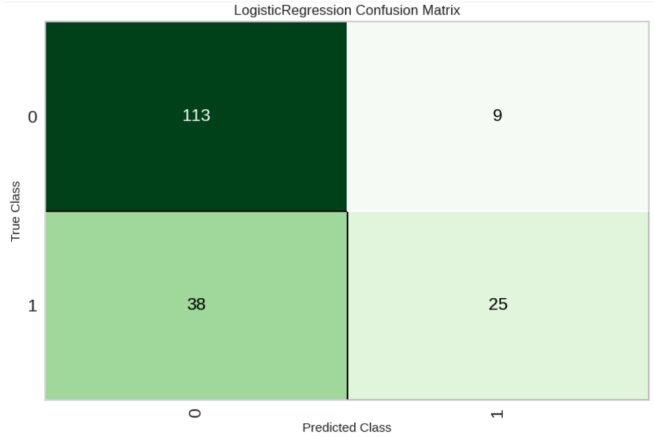

Fig. 12(b) : Confusion matrix LR

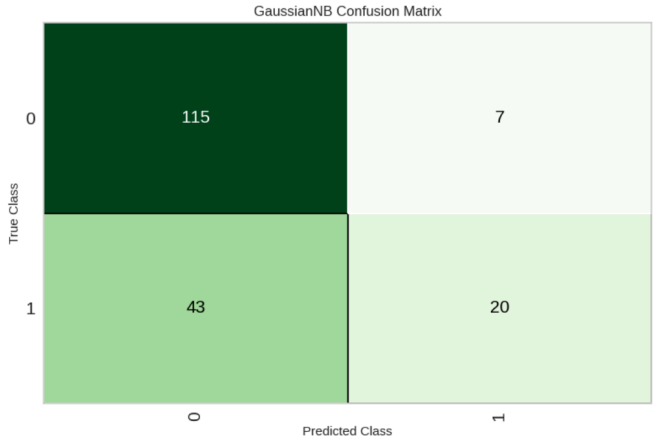

Fig. 12(c) : Confusion matrix NB

The F1-score is another significant metric for evaluating performance, as it combines precision and recall into one balanced assessment. The F1-score is especially handy for assessing and contrasting the effectiveness of various classifiers.

From a mathematical standpoint, the F1-score is determined as the harmonic average of precision and recall. It is determined by utilizing the subsequent equation:

$$F1 - Score = 2 \times \frac{precision \times recall}{precision + recall} \quad (7)$$

The F1-score can vary from 0 to 1, with superior performance reflected by higher values. It serves as a robust indicator for assessing classifiers, especially when one classifier excels in precision while another excels in recall. The F1-score evaluates how well a classifier performs by considering both precision and recall to accurately detect important cases and minimize incorrect identifications.

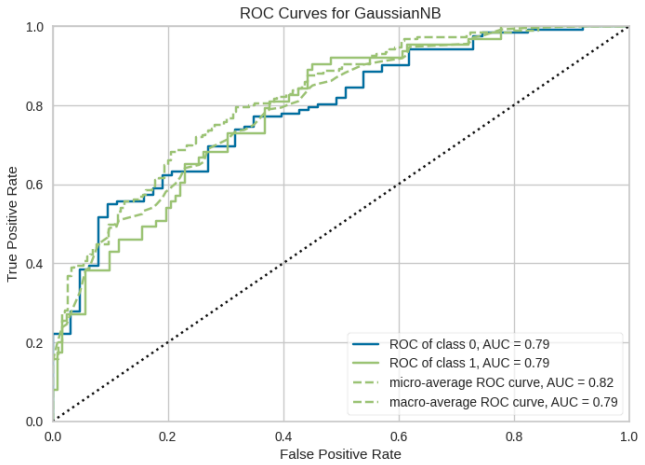

Fig. 13) : ROC curve NB

The ROC curve for the Naive Bayes (NB) algorithm is shown in Figure 13 as the last part of the analysis. The ROC curve visually shows how well a classification model performs at different thresholds for classifying data.

The ROC curve illustrates the True Positive Rate (TPR) and False Positive Rate (FPR), which are both important parameters. TPR, also called sensitivity, is the ratio of true positive instances correctly identified by the model, while FPR is the ratio of false positive instances wrongly classified as positive by the model.

These aspects are essential elements in constructing a confusion matrix, which offers a comprehensive analysis of predicted and actual class instances. Hence, the ROC curve and the confusion matrix are intricately connected and serve as various visualizations of the identical underlying evaluation.

VI. CONCLUSION

Our study highlights the effectiveness of Principal Component Analysis (PCA) in improving classification accuracy for the diabetes dataset. Through various classification algorithms and performance metrics, we found that PCA significantly enhances model performance, with

Naive Bayes emerging as the top-performing model post-transformation. Analysis also demonstrates the utility of precision, recall, F1-score, and ROC curves in evaluating classification performance. Overall, PCA proves to be a valuable tool for dimensionality reduction and feature extraction, offering insights that can aid in the development of accurate and reliable classification models, particularly in medical diagnostics.

VII. REFERENCES

[1] Maulud, Dastan & Mohsin Abdulazeez, Adnan. (2020). A Review on Linear Regression Comprehensive in Machine Learning. Journal of Applied Science and Technology Trends. 1. 140-147. 10.38094/jastt1457.

[2] Desai, Chitra. (2022). Understanding Linear Discriminant Analysis for Classification and Dimensionality Reduction.

[3] Cichosz, Pawel. (2015). Naïve Bayes classifier. 10.1002/9781118950951.ch4.

[4] Greenacre, Michael & Groenen, Patrick & Hastie, Trevor & Iodice D'Enza, Alfonso & Markos, Angelos & Tuzhilina, Elena. (2022). Principal component analysis. Nature Reviews Methods Primers. 2. 100. 10.1038/s43586-022-00184-w.

[5] A. B. Hamza, *Advanced Statistical Approaches to Quality*. Unpublished.