Application of PCA and Machine Learning on Dry Bean Dataset

ISAAC

Github: Dry bean classifier

***Abstract*—This project investigates the application of Principal Component Analysis (PCA) and machine learning algorithms to classify Dry Bean types based on their morphological features. PCA effectively reduced the dataset from 16 features to four principal components, retaining 95% of the variance. Several machine learning models, including Gradient Boosting Classifier, Random Forest, and Linear Discriminant Analysis, were evaluated for their classification performance both with and without PCA. The Gradient Boosting Classifier achieved the best overall performance, balancing high accuracy and computational efficiency. The study highlights the advantages of combining dimensionality reduction with machine learning for efficient and accurate classification, offering insights for similar applications in agricultural and related fields.**

Keywords—principal component analysis(pca), machine learning, classiification

I. Introduction

Classification tasks involving high-dimensional datasets often face challenges related to computational efficiency and model interpretability. The Dry Bean dataset, containing morphological features of different bean types, provides an excellent testbed to evaluate dimensionality reduction and machine learning techniques. Principal Component Analysis (PCA) offers a solution to these challenges by transforming the dataset into a lower-dimensional space while preserving the variance. This study explores the application of PCA and compares the performance of several machine learning algorithms, including Gradient Boosting, Random Forest, and Quadratic Discriminant Analysis. By examining the impact of PCA on classification accuracy and computational time, this paper seeks to identify optimal methods for efficient and accurate classification of Dry Bean types.

II. Principal Component Analysis (PCA)

A. Principal Component Analysis (PCA) Algorithm

Principal Component Analysis (PCA) is a statistical method used for dimensionality reduction, transforming a dataset with p variables into a set of orthogonal principal components (PCs) ordered by their variance. Below is the step-by-step explanation of the PCA algorithm:

1. Standardization

The first step is to standardize the data so that all variables contribute equally to the analysis, especially when measured on different scales.

Compute the mean vector p-dimensional vector representing the mean of each column of the data matrix X :

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Center the data by subtracting the mean vector $\bar{x}$ from each observation, resulting in the adjusted data matrix X_{adj}

$$X_{\text{adj}} = X - \bar{x}$$

This ensures that the data has a mean of zero for each variable, facilitating unbiased covariance calculations.

2. Covariance Matrix Computation

The covariance matrix S captures the relationships between variables. It is computed from the adjusted data matrix X_{adj}

$$S = \frac{1}{n-1} X_{\text{adj}}^{T} X_{\text{adj}}$$

$where\ S\ is\ a\ p \times p\ symmetric\ matrix$

Diagonal elements of S represent variances of individual variables, while off-diagonal elements represent covariances between variables.

3. Eigen Decomposition

To identify the principal components, eigen decomposition is performed on the covariance matrix S:

- Solve for eigenvalues (λ) and eigenvectors (A) such that:

  $$S\,A = A\,\Lambda$$

  where Λ is a diagonal matrix containing eigenvalues, and A is a p×p matrix whose columns are the eigenvectors.
- Eigenvalues (λ) indicate the variance captured by each principal component.
- Eigenvectors (A) represent the directions of the principal components in the original variable space.

4. Principal Components

Project the standardized data X_{adj} onto the eigenvectors to compute the principal components. The transformation is given by:

$$Z = X_{\text{adj}} A$$

Z represent the observations in the new coordinate system, and the columns represent the principal components. Typically, the first k components (corresponding to the largest eigenvalues) are retained to reduce dimensionality while preserving most of the variance.

III. MACHINE LEARNING ALGORITHM

A. Principal Component Analysis (PCA) Algorithm

A. Light Gradient Boosting Machine (LightGBM)

LightGBM is a gradient boosting framework that uses decision trees for predictive tasks. It is designed to be highly efficient and scalable for large datasets and supports both classification and regression tasks. LightGBM improves over traditional Gradient Boosting by using a histogram-based learning algorithm, which reduces memory usage and speeds up computation.

It uses leaf-wise tree growth, meaning it splits the leaf with the largest loss reduction rather than level-wise growth. This can lead to deeper trees and better performance with fewer iterations.

Mathematically, the LightGBM algorithm optimizes the following objective function:

$$L\big(y, f(x)\big) + \Omega\big(f(x)\big),$$

where is the loss function, $f(x)$ represents the predicted values,

and is aregularizationinter

The advantages of LightGBM include its ability to handle large datasets efficiently, support for categorical features, and better performance with fewer tuning requirements. However, it may overfit small datasets or noise if not properly regularized.

B. Gradient Boosting Classifier (GBC)

Gradient Boosting Classifier is an ensemble method that builds models sequentially, with each new model attempting to correct the errors of the previous ones. It combines weak learners, typically decision trees, into a strong predictive model.

The model is trained by minimizing a differentiable loss function through gradient descent. For a classification task, the loss function might be the log loss:

$$L\big(y, f(x)\big) = -\sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)],$$

where $\hat{y}_i$ is the predicted probability for class y_i

GBC is powerful for datasets with complex structures but requires careful hyperparameter tuning, as it is prone to overfitting. It also has higher computational complexity compared to simpler models.

C. Random Forest Classifier (RF)

Random Forest is an ensemble learning algorithm that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees.

The trees are trained using different bootstrap samples, and at each split, only a random subset of features is considered, which introduces diversity among the trees.

The prediction is obtained by majority voting:

$$\hat{y} = \text{mode}\big(T_1(x), T_2(x), \dots, T_n(x)\big),$$

Random Forest is robust against overfitting and works well with high-dimensional data. However, it can be computationally expensive and less interpretable compared to single decision trees.

D. Extra Trees Classifier (ETC)

Extra Trees Classifier, or Extremely Randomized Trees, is an ensemble method similar to Random Forest but differs in how splits are made. Unlike RF, which selects the best split at each node, ETC chooses splits randomly within a range of values.

This increases randomness, making the model less prone to overfitting, especially in small datasets. Predictions are aggregated through majority voting, similar to RF.

The mathematical representation is analogous to RF but with added randomness in feature splits:

$$\hat{y} = \text{mode}\big(T_1(x), T_2(x), \dots, T_n(x)\big).$$

$T_i(x)$ is

ETC is computationally efficient and robust but may underperform if the dataset requires precise split decisions.

E. Quadratic Discriminant Analysis (QDA)

QDA is a probabilistic classification algorithm that models each class as having its own Gaussian distribution with distinct mean vectors and covariance matrices. The decision boundary between classes is quadratic, allowing QDA to separate data with non-linear boundaries.

The QDA decision rule is:

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k) + \log \pi_k,$$

where Σ is the shared covariance matrix,

$\mu_k$is the mean vector for class k, and

π_k is the prior probability for class

QDA works well when the covariance of the classes is distinct but can overfit when there are many features relative to the number of observations.

F. Linear Discriminant Analysis (LDA)

LDA is a classification algorithm similar to QDA but assumes that all classes share the same covariance matrix. This constraint simplifies the decision boundary to be linear.

The decision rule for LDA is:

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log\pi_k,$$

where Σ *is the shared covariance matrix,*

μ_k*is the mean vector for class* k*, and*

π_k *is the prior probability for class* k

LDA performs well with linearly separable data and is computationally efficient. However, its assumptions of Gaussian distribution and equal covariance across classes may not always hold, limiting its effectiveness in some scenarios.

IV. Dataset Description

A. General Description

The Dry Bean dataset gotten from UC Irivine Machine Learning repository consists of 16 numerical features representing morphological properties of beans, which include dimensions, shape factors, and ratios. These features contribute to the classification of seven distinct bean types. A brief description of the features is provided below:

1. Area (A): Total area of a bean seed in pixels.
2. Perimeter (P): The circumference of the bean.
3. Major axis length (L): The longest possible line drawn through the bean.
4. Minor axis length (l): The longest line perpendicular to the major axis.
5. Aspect ratio (K): The ratio of the major to minor axis lengths.
6. Eccentricity (Ec): The deviation of the bean's shape from a perfect circle.
7. Convex area (C): Pixels in the smallest convex polygon enclosing the bean.
8. Equivalent diameter (Ed): Diameter of a circle with the same area as the bean.
9. Extent (Ex): Ratio of bean area to the area of its bounding box.
10. Solidity (S): Ratio of convex hull pixels to bean pixels.
11. Roundness (R)
12. Compactness (CO)
13. - 16. ShapeFactor1-4 (SF1, SF2, SF3, SF4): Derived shape metrics capturing additional nuances of bean morphology.

	count	mean	std	min	25%	50%	75%	max
Area	13611.0	53048.284549	29324.095717	20420.000000	36328.000000	44652.000000	61332.000000	254616.000000
Perimeter	13611.0	855.283459	214.289696	524.736000	703.523500	794.941000	977.213000	1985.370000
MajorAxisLength	13611.0	320.141867	85.694186	183.601165	253.303633	296.883367	376.495012	738.860153
MinorAxisLength	13611.0	202.270714	44.970091	122.512653	175.848170	192.431733	217.031741	460.198497
AspectRation	13611.0	1.583242	0.246678	1.024868	1.432307	1.551124	1.707109	2.430306
Eccentricity	13611.0	0.750895	0.092002	0.218951	0.715928	0.764441	0.810466	0.911423
ConvexArea	13611.0	53768.200206	29774.915817	20684.000000	36714.500000	45178.000000	62294.000000	263261.000000
EquivDiameter	13611.0	253.064220	59.177120	161.243764	215.068003	238.438026	279.446467	569.374358
Extent	13611.0	0.749733	0.049086	0.555315	0.718634	0.759859	0.786851	0.866195
Solidity	13611.0	0.987143	0.004660	0.919246	0.985670	0.988283	0.990013	0.994677
roundness	13611.0	0.873282	0.059520	0.489618	0.832096	0.883157	0.916869	0.990685
Compactness	13611.0	0.799864	0.061713	0.640577	0.762469	0.801277	0.834270	0.987303
ShapeFactor1	13611.0	0.006564	0.001128	0.002778	0.005900	0.006645	0.007271	0.010451
ShapeFactor2	13611.0	0.001716	0.000596	0.000564	0.001154	0.001694	0.002170	0.003665
ShapeFactor3	13611.0	0.643590	0.098996	0.410339	0.581359	0.642044	0.696006	0.974767
ShapeFactor4	13611.0	0.995063	0.004366	0.947687	0.993703	0.996386	0.997883	0.999733

Fig. 1. Statistcical descrption of the dry bean data

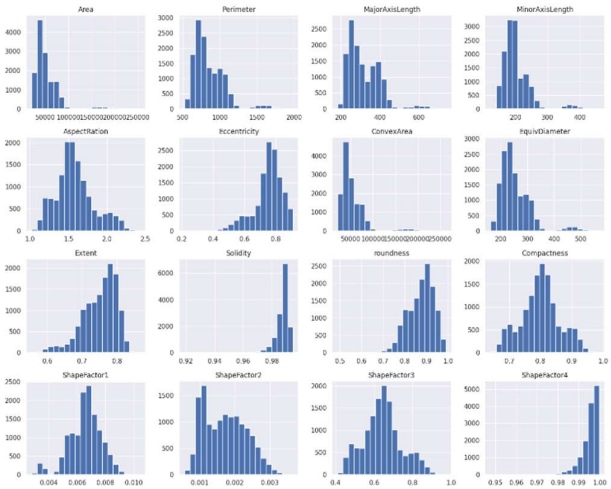

Fig. 2. Distribution of features

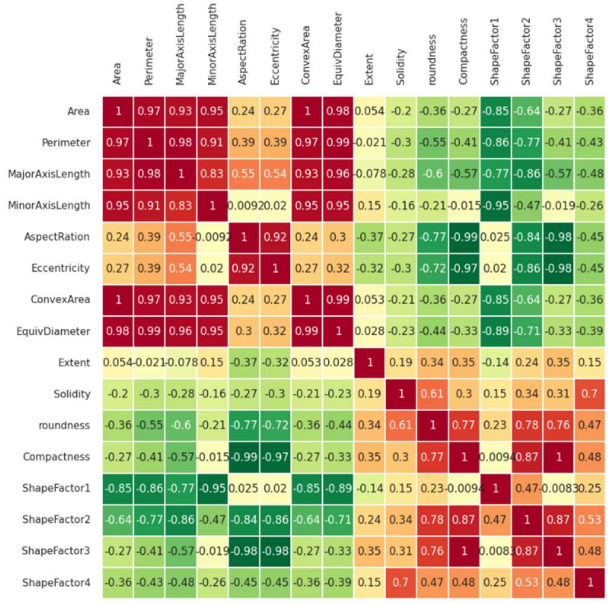

Fig. 3. Correlation matrix of features

V. PCA Results

A. Model Comparison

PCA was applied to the Dry Bean dataset to reduce its dimensionality and reveal key patterns in its 16 morphological features. PCA implementation can be approached in two ways:

(1) developing PCA from scratch using libraries like NumPy, or

(2) utilizing robust PCA libraries such as scikit-learn. In this analysis, the scikit-learn library was used, which simplifies the implementation with minimal code while providing flexibility for visualization and interpretation.

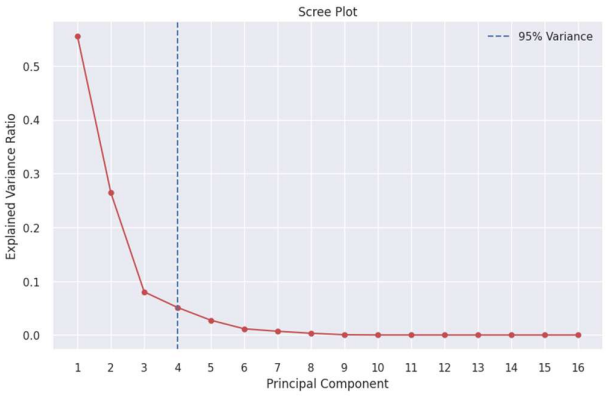

Fig. 4. Scree plot

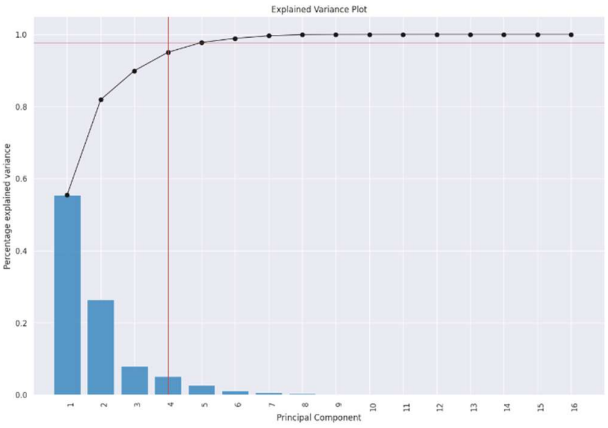

Fig. 5. Pareto plot

Figures 1 and 2 display the correlation matrix and the pair plot of the dataset, revealing significant correlations among features like Area, Perimeter, and MajorAxisLength. These strong correlations justify applying PCA to reduce redundancy and improve interpretability.

	Features	PC1	PC2	PC3	PC4
0	Area	0.282458	0.245882	-0.061447	-0.031546
1	Perimeter	0.310891	0.179303	-0.018853	-0.042468
2	MajorAxisLength	0.325824	0.100757	-0.084692	-0.006793
3	MinorAxisLength	0.236199	0.343461	0.007500	-0.061300
4	AspectRation	0.229298	-0.330844	-0.169058	0.053646
5	Eccentricity	0.231526	-0.319434	-0.163042	0.118389
6	ConvexArea	0.283200	0.244630	-0.053649	-0.030960
7	EquivDiameter	0.297484	0.222802	-0.049914	-0.032427
8	Extent	-0.059808	0.220619	-0.085258	0.948254
9	Solidity	-0.143016	0.103322	-0.738670	-0.049546
10	roundness	-0.248165	0.214805	-0.163325	0.067482
11	Compactness	-0.238378	0.328914	0.149701	-0.087156
12	ShapeFactor1	-0.221319	-0.332549	-0.032623	0.072330
13	ShapeFactor2	-0.314625	0.129419	0.120077	-0.046544
14	ShapeFactor3	-0.238983	0.327522	0.149570	-0.095679
15	ShapeFactor4	-0.198009	0.100061	-0.536903	-0.210120

Fig. 6. Principal component and features

By applying PCA, the 16-feature dataset was reduced to four principal components (PCs), which account for 95% of the total variance. The PCA transformation projects the original data using an eigenvector matrix A, with each column representing a principal component. The eigenvector matrix for the Dry Bean dataset is as follows:

$$A = \begin{pmatrix} 0.282 & 0.246 & -0.061 & -0.032 \\ 0.311 & 0.179 & -0.019 & -0.042 \\ 0.326 & 0.101 & -0.085 & -0.007 \\ 0.236 & 0.343 & 0.008 & -0.061 \\ 0.229 & -0.331 & -0.169 & 0.054 \\ 0.232 & -0.319 & -0.163 & 0.118 \\ \dots & \dots & \dots & \dots \\ -0.199 & 0.100 & -0.537 & -0.210 \end{pmatrix}$$

The corresponding eigenvalues:

$$\lambda = \begin{pmatrix} 5.54 \\ 2.64 \\ 0.80 \\ 0.51 \end{pmatrix}$$

From **the Figure 4 (Scree plot)**, it can be observed that the first four PCs collectively account for 95% of the variance. PC1 explains 55.5% of the variance, PC2 explains 26.4%, while PC3 and PC4 explain 8.0% and 5.1%, respectively. The elbow point in the scree plot occurs at PC4, confirming that the dataset can be effectively reduced to four dimensions

$$Z_1 = 0.282A + 0.311P + 0.326L + 0.236l + 0.229K + 0.232Ec + \cdots - 0.199SF4$$

$$Z_2 = 0.246A + 0.179P + 0.101L + 0.343l - 0.331K - 0.319Ec + \cdots + 0.100SF4$$

$$Z_3 = -0.061A - 0.019P - 0.085L + 0.008l - 0.169K - 0.163Ec + \cdots - 0.537SF4$$

$$Z_4 = -0.032A - 0.042P - 0.007L - 0.061l + 0.054K + 0.118Ec + \cdots - 0.210SF4$$

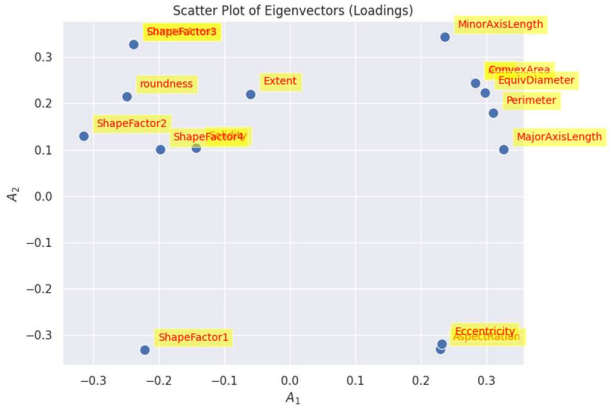

Fig. 7. Scatter plot of Eigenvectors

Fig 7 represents the PC coefficient plot, which visually demonstrates the contribution of each feature to the first two PCs. Features such as Area, Perimeter, and MajorAxisLength show significant contributions to PC1, while Extent and Solidity have dominant influences on PC2. Features facing similar directions (e.g., Area and Perimeter) are highly positively correlated, as confirmed by the angles between their vectors.

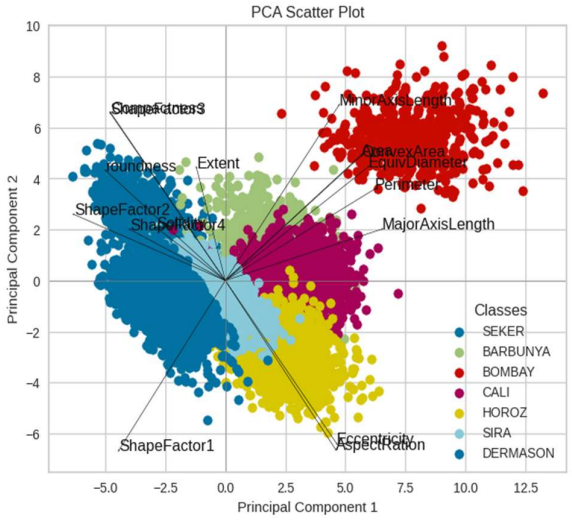

Fig. 8. biplot

The biplot (Fig 8) provides another visual representation of the first two PCs. In the biplot:

Features such as Area and Perimeter form small angles with PC1, confirming their strong contributions to it.

VI. Classification Results

A. Model Comparison (without PCA)

LightGBM (Light Gradient Boosting Machine) performs the best with an accuracy of 92.77% and the highest AUC of 0.9937. This suggests that LightGBM was well-tuned for the given features.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.9277	0.9937	0.9277	0.9282	0.9277	0.9125	0.9126	7.1660
gbc	Gradient Boosting Classifier	0.9260	0.0000	0.9260	0.9265	0.9259	0.9105	0.9106	22.7980
rf	Random Forest Classifier	0.9229	0.9923	0.9229	0.9232	0.9228	0.9067	0.9068	1.1420
et	Extra Trees Classifier	0.9193	0.9923	0.9193	0.9199	0.9193	0.9023	0.9024	0.6660
qda	Quadratic Discriminant Analysis	0.9151	0.0000	0.9151	0.9183	0.9154	0.8975	0.8981	0.0320
lda	Linear Discriminant Analysis	0.9071	0.0000	0.9071	0.9168	0.9086	0.8877	0.8894	0.1960
dt	Decision Tree Classifier	0.8941	0.9349	0.8941	0.8947	0.8941	0.8720	0.8721	0.1070
lr	Logistic Regression	0.8638	0.0000	0.8638	0.8649	0.8636	0.8350	0.8353	2.6870
ridge	Ridge Classifier	0.8601	0.0000	0.8601	0.8712	0.8537	0.8294	0.8360	0.0770
nb	Naive Bayes	0.7659	0.9642	0.7659	0.7664	0.7634	0.7170	0.7177	0.0380
knn	K Neighbors Classifier	0.7135	0.9262	0.7135	0.7134	0.7098	0.6518	0.6531	0.1700
ada	Ada Boost Classifier	0.6964	0.0000	0.6964	0.6672	0.6485	0.6284	0.6501	0.5260
svm	SVM - Linear Kernel	0.3040	0.0000	0.3040	0.2626	0.1910	0.1634	0.2170	0.1950
dummy	Dummy Classifier	0.2605	0.5000	0.2605	0.0679	0.1077	0.0000	0.0000	0.0830

Fig. 9. Comparison of Models trained without performance of pca

Gradient Boosting Classifier (GBC) also performs well, with an accuracy of 92.59%, but the AUC is 0, which is an anomaly. This could indicate issues with how the metrics were calculated or the inherent behavior of this model on the dataset.

Extreme Gradient Boosting (XGBoost) and Random Forest (RF) also show strong results with accuracies of 92.46% and 92.29%, respectively. These ensemble methods handle large datasets well and likely perform well without dimensionality reduction.

In terms of Computational Time, XGBoost has the fastest computation time of 1.629 seconds, which is significantly lower than GBC (52.948 seconds). This indicates that some algorithms, even when using all features, may require varying amounts of computation due to algorithmic complexity.

On the other hand, Naive Bayes (NB) and K Neighbors Classifier (KNN) take relatively short times (0.217s and 0.412s, respectively) but yield poor accuracy, which indicates that they are inefficient for this dataset, possibly due to the feature space's dimensionality.

In terms of Recall and Precision, Recall is a good measure for the ability to capture true positives, and it is high for LightGBM, GBC, and XGBoost, suggesting that these models are very good at identifying the positive class. However, SVM (with linear kernel) performs very poorly across all metrics, with a recall of just 0.304 and accuracy of 0.304%.

F1 scores are balanced between precision and recall, and they are strong across the board for ensemble methods like LightGBM and GBC. This indicates that these models perform well in capturing the underlying patterns of the data while maintaining a good balance between false positives and false negatives.

B. Model Comparison (with PCA)

GBC still leads with an accuracy of 88.91%, which is a decrease from the non-PCA scenario. However, its AUC score remains 0 (likely due to the way the data is transformed and evaluated).

Models like Logistic Regression (LR), Quadratic Discriminant Analysis (QDA), and Extra Trees (ET) show a noticeable drop in performance compared to when no PCA is applied.

SVM (with linear kernel) does much better with PCA, showing an accuracy of 77.18%, a substantial improvement from its previous 30.4%.

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Se
gbc	Gradient Boosting Classifier	0.8909	0.0000	0.8909	0.8907	0.8904	0.8680	0.8682	4.65
lr	Logistic Regression	0.8895	0.0000	0.8895	0.8898	0.8890	0.8663	0.8665	0.07
qda	Quadratic Discriminant Analysis	0.8885	0.0000	0.8885	0.8892	0.8880	0.8652	0.8655	0.05
et	Extra Trees Classifier	0.8873	0.9884	0.8873	0.8871	0.8866	0.8636	0.8638	0.37
rf	Random Forest Classifier	0.8871	0.9878	0.8871	0.8869	0.8864	0.8633	0.8635	0.71
lightgbm	Light Gradient Boosting Machine	0.8836	0.9885	0.8836	0.8836	0.8831	0.8591	0.8593	1.43
lda	Linear Discriminant Analysis	0.8822	0.0000	0.8822	0.8844	0.8814	0.8575	0.8581	0.04
knn	K Neighbors Classifier	0.8814	0.9735	0.8814	0.8815	0.8810	0.8565	0.8566	0.07
nb	Naive Bayes	0.8661	0.9822	0.8661	0.8661	0.8645	0.8379	0.8383	0.04
dt	Decision Tree Classifier	0.8463	0.9075	0.8463	0.8472	0.8465	0.8143	0.8144	0.05
svm	SVM - Linear Kernel	0.7718	0.0000	0.7718	0.7468	0.7501	0.7227	0.7282	0.06
ridge	Ridge Classifier	0.6536	0.0000	0.6536	0.5486	0.5673	0.5737	0.6041	0.03
ada	Ada Boost Classifier	0.4398	0.0000	0.4398	0.2950	0.3148	0.2874	0.3372	0.28
dummy	Dummy Classifier	0.2605	0.5000	0.2605	0.0679	0.1077	0.0000	0.0000	0.04

Fig. 10. Comparison of Models trained without performance of pca

In the evaluation of Computational Time, as expected, applying PCA has significantly reduced computation times for many models. For example, GBC runs faster (11.144 seconds vs. 52.948 seconds), and LightGBM runs faster (5.571 seconds vs. 13.529 seconds), thanks to the reduced feature space.

Models like Dummy Classifier show a minimal change in time but still produce low-quality results.

In terms of precision and recall, Models generally show a drop in recall when PCA is applied, especially in ensemble methods like LightGBM, GBC, and XGBoost. This suggests that while dimensionality reduction can make models run faster, it may also remove critical features that the models used to predict certain classes.

SVM's recall and precision show significant improvement, which supports the idea that PCA may help with certain algorithms by simplifying the decision boundaries.

F1 scores show a similar trend, with ensemble methods and SVM showing a decline after applying PCA. This suggests that while PCA can help with computation, it may not always be beneficial for every algorithm in terms of prediction performance, particularly with models that rely on rich feature sets.

C. Confusion Matrix

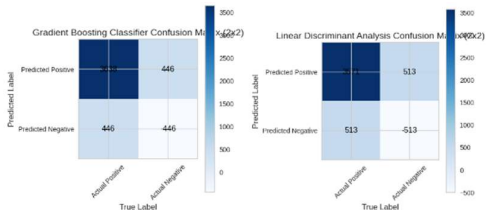

Fig. 11. Confusion matrix of gradiaent boosting classifier and linear discrimant analysis

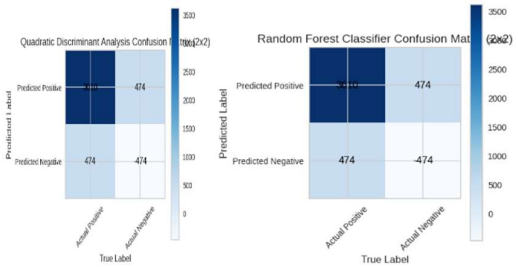

Fig. 12. Confusion matrix of random forest classifer and quadratic component anslysis

The confusion matrix (Figure 11 and 12) shows the following:

The Random Forest Classifier demonstrates balanced performance across positive and negative predictions. It correctly identifies 3510 true positives (TP) but misclassifies 474 false positives (FP) and 474 false negatives (FN). While the model captures most of the positive class effectively, the misclassification rates on both sides are symmetric, indicating a consistent level of error across the dataset. The model's balanced behavior suggests that it performs reliably but still leaves room for improvement in reducing false classifications.

The Quadratic Discriminant Analysis (QDA) confusion matrix is identical to that of the Random Forest Classifier. QDA also achieves 3510 true positives (TP) while misclassifying 474 false positives (FP) and 474 false negatives (FN). This similarity suggests that QDA aligns well with the dataset's structure, likely due to its assumption of Gaussian distributions in the data. QDA performs reliably but does not outperform Random Forest in any aspect, and both models share the same level of misclassification.

The Gradient Boosting Classifier outperforms the other algorithms by achieving the highest true positive count (3538) while reducing misclassification rates. It records 446 false positives (FP) and 446 false negatives (FN), showing improvement over Random Forest and QDA in both categories. This reduction in misclassification indicates that the model is more effective at identifying both classes correctly. The Gradient Boosting model's ability to minimize errors while maintaining strong performance in identifying positives makes it the best-performing algorithm for this dataset.

The Linear Discriminant Analysis (LDA) model performs better than Random Forest and QDA but does not achieve the same level of accuracy as Gradient Boosting. LDA identifies 3571 true positives (TP), which is higher than Random Forest and QDA, but it also records 513 false positives (FP) and 513 false negatives (FN). While the model improves its ability to capture positives, it does so at the cost of slightly increased misclassification compared to Gradient Boosting. Overall, LDA strikes a balance between the simpler models and Gradient Boosting but does not reach the latter's superior accuracy.

Among the four algorithms, the Gradient Boosting Classifier delivers the best performance by maximizing true positives and minimizing false positives and false negatives. Random Forest and QDA show identical and balanced performance but lag behind Gradient Boosting. Linear Discriminant Analysis improves on Random Forest and QDA but remains slightly less effective than Gradient Boosting. For

optimal results on this dataset, the Gradient Boosting Classifier is the most reliable choice, offering superior accuracy and reduced misclassification

VII. Conclusion

The study demonstrates the effective application of Principal Component Analysis (PCA) and various machine learning algorithms on the Dry Bean dataset. By reducing the dataset's dimensionality, PCA preserved 95% of the total variance within four principal components, providing a robust foundation for subsequent classification tasks. Among the machine learning models evaluated, the Gradient Boosting Classifier emerged as the most reliable choice, achieving the highest accuracy and minimized misclassifications. This analysis highlights the potential of combining dimensionality reduction techniques with advanced machine learning models to handle large datasets effectively while optimizing computational efficiency and predictive performance. These findings pave the way for applying such techniques to similar classification problems in agricultural and other domains.

References

[1] IBM, "What is Principal Component Analysis (PCA)?" [Online]. Available: https://www.ibm.com/think/topics/principal-component-analysis. [Accessed: Dec. 18, 2024].

[2] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016, pp. 785–794. [Online]. Available: https://arxiv.org/pdf/1603.02754. [Accessed: Dec. 18, 2024].

[3] S. Liu and W. Chen, "Performance analysis of boosting classifiers in recognizing morphological features," PMC PubMed Central, 2021. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7038216/. [Accessed: Dec. 18, 2024]

[4] J. Shlens, "A Tutorial on Principal Component Analysis," arXiv preprint arXiv:1404.1100, Apr. 2014.

[5] I. T. Jolliffe, Principal Component Analysis, 2nd ed. Springer, 2002.

[6] C. M. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.

[7] M. Koklu and I. A. Ozkan, "Dry Bean Dataset," UCI Machine Learning Repository, 2020. [Online]. Available: https://archive.ics.uci.edu/dataset/602/dry+bean+dataset. [Accessed: Dec. 19, 2024].

[8] M. Koklu and I. A. Ozkan, "Multiclass Classification of Dry Beans Using Computer Vision and Machine Learning Techniques," Computers and Electronics in Agriculture, vol. 174, p. 105507, 2020. DOI: 10.1016/j.compag.2020.105507