# Dimentionality reduction using principal component analysis and Linear discriminant analysis with machine learning

Arman Hosseinmardi - 40271290

https://github.com/armanhm/INSE6220Project

***Abstract*—**Principal Component Analysis (PCA) serves as a technique for reducing the dimensionality of extensive datasets by transforming a collection of correlated variables into a smaller, uncorrelated set, capturing the bulk of the information. This study applies PCA to a fast-food industrial dataset. Employing logistic regression (LR), k-nearest neighbor (KNN), and RFC (Random Forest classifier) as classification algorithms, both the original and transformed datasets post PCA are utilized for the type identification. Subsequently, fine-tuning each model with optimal hyperparameters enhances performance metrics, evaluated via F1 score, confusion matrices, and receiver operating characteristic (ROC) curves. Visualization of decision boundaries elucidates the model fitting process on the dataset. LR emerges as the top-performing model among various machine learning models in the PyCaret library. Moreover, an interpretive experiment employing Explainable AI through Shapley values utilizes an Extra Trees (ET) classifier model.

**Keywords:** ***Principal component analysis, binary classification, logistic regression, k-nearest neighbor, quadratic discriminant analysis, Random Forest***

## I. Introduction

The fast-food industry stands as a pivotal element in modern consumer culture, reflecting changing dietary habits and lifestyle choices. Within this landscape, a dataset containing various intrinsic features holds the key to understanding consumer preferences and the nutritional landscape of fast-food items.

Through comprehensive analysis utilizing machine learning and data mining techniques, this dataset aims to unravel patterns, correlations, and significant insights related to consumer behaviors, nutritional compositions, and trends within the fast-food industry. The subsequent sections will delve into methodologies, dataset descriptions, analyses of specific features, and discussions on the implications derived from this multifaceted dataset.

1. **Insights from Nutritional Composition:** Analyzing the nutritional composition within this fast food dataset presents a unique opportunity to comprehend the dietary landscape. The presence of varied macronutrients—such as proteins, fats, and carbohydrates—offers a comprehensive understanding of the balance between health considerations and consumer preferences. Moreover, the evaluation of ash content provides valuable insights into the mineral makeup of these food items, which can influence consumer choices based on perceived health benefits. The detailed breakdown of sodium levels, a crucial but often contentious element in fast food, aids in comprehending its prevalence in different food categories and its potential impact on public health concerns.
2. **Utilizing Machine Learning for Industry Insights:** Leveraging advanced machine learning methodologies, this dataset aims to uncover hidden patterns and correlations among the features encapsulated within. Through classification algorithms, predictive models, and dimensionality reduction techniques, the analysis seeks to discern trends in consumer behavior and preferences across various fast food categories. Exploring the interplay between moisture content, caloric value, and nutritional components like proteins and fats could provide invaluable insights into consumer perceptions and choices. Ultimately, the application of these data-driven techniques aims to assist stakeholders in the fast food industry in making informed decisions, optimizing product offerings, and potentially steering towards healthier alternatives aligned with consumer demands.

## II. Principal Component Analysis in Data Dimensionality Reduction

The majority of real-world datasets exhibit a characteristic of high dimensionality, posing challenges in processing, storage, and visualization. This issue becomes not only costly but also at times infeasible.

Techniques like Principal Component Analysis (PCA) offer a remedy by condensing a vast array of variables into a more manageable subset that retains the essential information from the original dataset.

PCA, as a method for simplifying high-dimensional data, retains inherent trends and patterns [4]. It achieves this objective by compressing the data into fewer dimensions that act as comprehensive feature summaries.

A. The PCA Algorithm: Steps for Dimension Reduction

PCA can be employed on a data matrix X with dimensions n × p through a sequence of defined steps [5]:

1. **Standardization**: This initial step aims to standardize the original variables to ensure uniform contribution to the analysis. It begins by computing the mean vector $\bar{x}$ for each column within the dataset. The mean vector, a p-dimensional representation, is expressed as:

Standardization ensues by subtracting the mean of each column from the respective items in the data matrix. The resultant centered data matrix (Y) is formulated as:

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

The data is standardized by subtracting the mean of each column from each item in the data matrix. The final centered data matrix (Y) can be expressed as follows:

$$Y = HX$$

where H denotes the centering matrix.

2. **Computation of Covariance Matrix**: This step endeavors to unveil relationships among variables, identifying potential redundancies owing to closely related variables. The p × p covariance matrix is calculated as:

$$S = \frac{1}{n-1} Y^T Y$$

3. **Eigen Decomposition**: Utilizing eigen decomposition, the eigenvalues and eigenvectors of S can be determined. Eigenvectors portray the direction of each principal component (PC), while eigenvalues represent the variance captured by each PC. Eigen decomposition is computed using the equation:

$$S = A\Lambda A^T$$

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

4. **Principal Components**: This step calculates the transformed matrix Z of size n × p. Z's rows correspond to observations, while its columns represent the PCs. The number of PCs aligns with the original data matrix's dimensions. The equation for Z is given by:

$$Z = YA$$

## III. Linear Discriminant Analysis (LDA) for Dimension Reduction

Similar to PCA, real-world datasets often grapple with high dimensionality, necessitating efficient methods for data simplification and pattern recognition. Linear Discriminant Analysis (LDA) emerges as a technique aiming to reduce data dimensionality while preserving crucial information regarding class discrimination.

LDA operates by projecting data into a lower-dimensional space, seeking optimal separation between classes. Unlike PCA, which emphasizes overall variance, LDA specifically emphasizes class separability. By maximizing between-class variance and minimizing within-class variance, LDA identifies the most discriminative features, aiding in classification tasks and enhancing interpretability.

Steps Involved in LDA:

1. Computing Class Means: LDA commences by calculating the means of each class within the dataset. This involves determining the mean vectors for different classes to capture class-specific information.
2. Scatter Matrices: Two types of scatter matrices are computed in LDA: within-class scatter matrix and between-class scatter matrix. These matrices encapsulate information about data distribution within classes and the separation between classes.
3. Eigen Decomposition: Through eigen decomposition of the scatter matrices, LDA extracts eigenvectors and eigenvalues. These eigenvalues aid in selecting the most informative discriminant axes, while the corresponding eigenvectors depict the directions of optimal class separation.
4. Projection onto Discriminant Axes: The final step involves projecting the data onto the selected discriminant axes. This transformation facilitates the creation of a lower-dimensional

subspace where the data is effectively separated based on class labels, enabling better classification performance.

## IV. MACHINE LEARNING-BASED CLASSIFICATION ALGORITHMS

A. **Logistic Regression (LR):** Logistic Regression (LR) serves as a statistical technique primarily applied in binary classification tasks. Despite its name, LR is employed for classification rather than regression problems. It models the relationship between a binary dependent variable and one or more independent variables by estimating the likelihood of an event occurring. Using the logistic function, LR compresses the output within the range of 0 to 1, interpreting these outputs as probabilities. The logistic function, fundamental to LR, is defined as:

$$P(Y = 1|X) = \frac{1}{1 + e^{-Z}}$$

Where $P(Y = 1|X)$ represents the probability of the outcome being 1 given the input features $X$, and $z$ is the linear combination of the input features and their respective coefficients.

B. **K-nearest neighbor (K-NN):** K-Nearest Neighbor is a non-parametric algorithm used for both classification and regression tasks. It operates based on the assumption that similar data points exist in close proximity to each other in the feature space. In classification, when presented with a new data point, K-NN identifies its k nearest neighbors based on distance metrics (such as Euclidean distance) and assigns the majority class among those neighbors to the new data point. The choice of the parameter 'k' impacts model performance, balancing between overfitting (with lower k) and underfitting (with higher k).

C. **Quadratic Discriminant Analysis (QDA):** Quadratic Discriminant Analysis is a classification algorithm that assumes different covariance matrices for each class. Unlike Linear Discriminant Analysis (LDA), CDA doesn't assume equal covariance matrices for the classes. QDA models the probability of belonging to each class using quadratic decision boundaries, allowing for more flexible decision boundaries compared to LDA. It estimates class conditional densities and applies Bayes' theorem to classify new data points into the class that yields the highest probability.

QDA involves estimating class conditional densities using Gaussian distributions and applying Bayes' theorem. For each class $c$, the probability density function $fc(x)$ for a data point $x$ is given by:

$$f_c(x) = \frac{1}{\sqrt{(2\pi)^n |\sum c|}} e^{-\frac{1}{2}(x-\mu_c)^T \sum_c^{-1}(x-\mu_c)}$$

where $\mu_c$ is the mean vector and $\Sigma c$ is the covariance matrix for class $c$, and $n$ represents the number of features.

D. **Decision Tree Classifier:** Decision Tree Classifier is a non-parametric supervised learning method used for classification tasks. It works by recursively partitioning the feature space into smaller regions based on the most informative features at each step. The tree structure is composed of nodes representing feature splits and leaf nodes representing the final predicted class. Decision trees are easy to interpret and visualize, making them advantageous for understanding feature importance, but they are susceptible to overfitting.

Decision trees create splits based on information gain or Gini impurity to minimize the uncertainty in classification. The information gain ($IG$) for a split in a decision tree is calculated using:

$$IG = Entropy(parent) - Weighted\ average\ of\ children$$

Where entropy is given by $H = -\sum p(x) log_2 p(x)$ and $p(x)$ is the probability of a certain class.

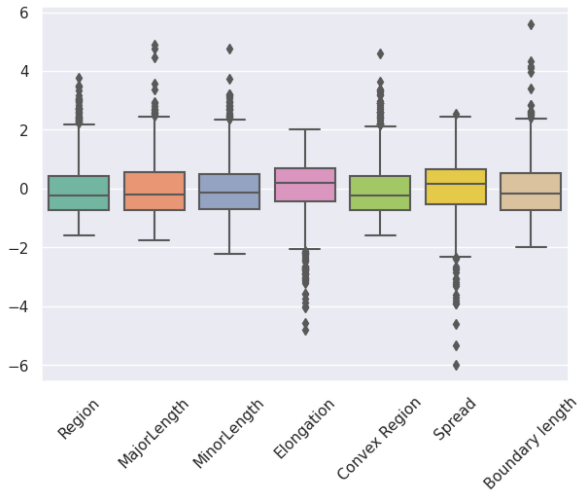
E. **Random Forest Model:** Random Forest is an ensemble learning technique that operates by constructing multiple decision trees during training and outputs the class that is the mode of the classes output by individual trees. Each tree in the forest is built on a bootstrapped sample from the dataset and utilizes a subset of features, introducing randomness and reducing overfitting. By aggregating predictions from multiple trees, Random Forest improves robustness and generalizability compared to individual decision trees.

## V. DATASET DESCRPITION

The food dataset employed for this analysis compiles a diverse array of culinary information sourced from distinct geographic regions. It encompasses a comprehensive range of data related to various food items, their characteristics, and classifications based on specific attributes. This dataset serves as a valuable resource for exploring regional culinary differences and food item categorizations.

The dataset comprises several essential columns, each offering unique insights into the characteristics of food items. It includes the following key features:

1. **Region:** This column delineates the geographic or cultural regions from which the food items originate. It potentially includes names, labels, or codes representing specific areas renowned for their culinary traditions.
2. **MajorLength:** This feature represents a numerical measurement associated with the major dimension or prominent characteristic of the food item. It could indicate the length or magnitude of a significant attribute of the cuisine.
3. **MinorLength:** This column likely contains numerical values representing the minor dimension or lesser characteristic of the food item. It might signify a secondary aspect or dimension of the cuisine, providing complementary information to the major dimension.
4. **Elongation:** This feature potentially contains numerical data indicating the elongation or stretching factor associated with the food item. It might describe how the food item deviates from a standard or typical shape in its category.
5. **Convex Region:** This column might contain categorical or numerical data describing whether the food item is considered convex or possesses convex characteristics based on its shape or structure.
6. **Spread:** This column likely contains numerical values indicating the spread or distribution of specific attributes within the food item. It might pertain to the variation or dispersion of certain characteristics across different regional cuisines.
7. **Boundary length:** This feature could encompass numerical values representing the length or extent of certain boundaries or distinct elements within the food items. It might relate to culinary specifics or unique attributes that set apart different regional dishes.
8. **Class:** This column potentially contains categorical labels or numerical values signifying the class or category to which each food item belongs. It could represent different groups, types, or classifications of regional cuisines based on specific culinary characteristics or traditions.




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 fast food dataset. It can be observed from Fig. 1 that most of the features follow approximately normal distribution. However, outliers exist in all features. All features except Spread and Elongation contain outliers on the left whereas Spread and Elongation contains outliers on both sides.


Fig. 2: Correlation matrix

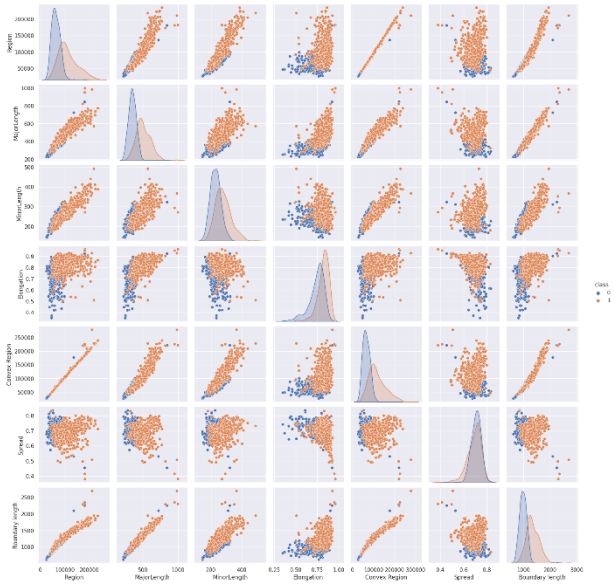


Fig. 3: Pair Plot

Fig. 2 shows the correlation matrix for the normalized features of the dataset. The features with large positive numbers are Region, MajorLength, Convex Region, and Boundary length. This evident implies that these three features are highly correlated. Other three features; i.e. MinorLength, Elongation, and Spread show less correlation with the other features in the dataset. The pair-plot in Fig. 3 supports this observation. The highly correlated features contain higher number of cells with regularly increasing line.

Utilizing statistical analyses, visualizations such as histograms, scatterplots, and geographical maps, an in-depth exploration of the dataset's culinary attributes and regional distinctions was undertaken. These visualizations offer insights into the distribution patterns, relationships among culinary features, and potential distinctions across various regional cuisines, enabling a nuanced understanding of culinary diversity and characteristics across different geographic areas.

## VI. PCA RESULTS

PCA is applied on fast-food dataset. Implementation of PCA can be done in two ways: (1) developing PCA from scratch using standard Python libraries such as numpy, (2) using popular and well documented PCA library. In the Kaggle notebook implementation of both methods is provided. Even though results obtained from both ways are similar, usage of PCA library brings more flexibility to the user and a lot can be done by writing only single line of code. In this report, the figures and plots are shown from the implementation using PCA library.

By applying the PCA steps, the feature set of 7 can be reduced to r numbers of features where r < 7. The original n × p dataset is reduced using eigenvector matrix A. Each column of the eigenvector matrix A is represented by a PC. Each PC captures an amount of data that determines the dimension (r). The obtained eigenvector matrix (A) for fast food data set is as follow:

$$\begin{bmatrix} 0.448 & -0.116 & 0.005 & -0.111 & -0.611 & -0.099 & -0.624 \\ 0.443 & 0.136 & -0.100 & 0.495 & 0.087 & -0.685 & 0.227 \\ 0.389 & -0.374 & 0.236 & -0.655 & 0.384 & -0.239 & 0.129 \\ 0.202 & 0.610 & -0.628 & -0.426 & 0.075 & 0.053 & 0.020 \\ 0.450 & -0.087 & 0.036 & 0.055 & -0.392 & 0.471 & 0.639 \\ -0.056 & -0.667 & -0.731 & 0.109 & 0.056 & 0.023 & -0.001 \\ 0.450 & 0.034 & 0.044 & 0.339 & 0.555 & 0.487 & -0.363 \end{bmatrix}$$

and the corresponding eigenvalues are:

$$\lambda = \begin{bmatrix} 4.837 \\ 1.454 \\ 0.629 \\ 0.056 \\ 0.021 \\ 0.006 \\ 0.001 \end{bmatrix}$$

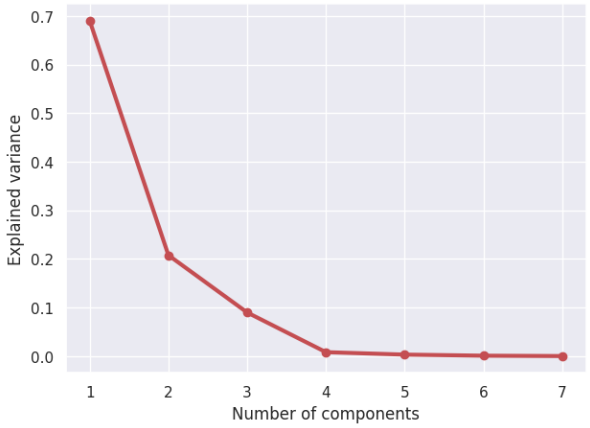



Fig. 4: Scree Plot

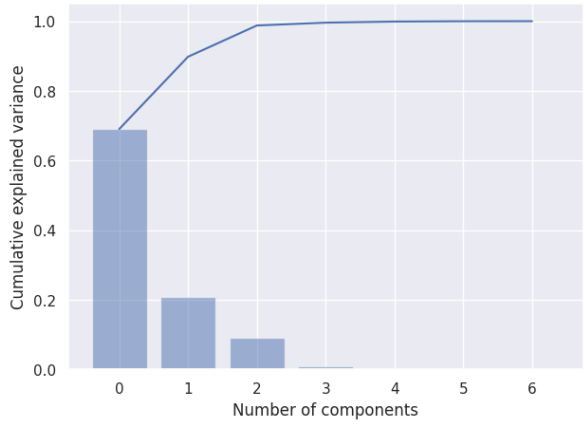



Fig. 5: Pareto Plot

Fig 4 and pareto plot in Fig. 5 demonstrate the scree plot and pareto plot of the PCs. The scree plot and pareto plot display the amount of variance explained by each principal component. The percentage of variance experienced by j-th PC can be evaluated using the following equation:

$$j = \frac{\lambda_j}{\sum_j^p \lambda_j} \times 100, j = 1,2, \ldots., p,$$

where $\lambda_j$ represents the eigenvalue and the amount of variance of the j-the PC. Fig 4 and 5 plot the number of PCs vs the explained variance. It can be observed from both figures that the variance of first two PC's contribute to 89.8% of the amount of variance of the original dataset; i.e. first PC holds 69% of variance ($l_1$ = 60%) and second PC holds 20.75% of variance ($l_2$ = 20.75%). The scree plot presents that the elbow is located on the second PC. These two observations imply that the dimension of the feature set can be reduce to two (r = 2). The first principal component Z1 is given by:

$$Z_1 = 0.448\,X_1 + 0.443\,X_2 + 0.389\,X_3 + 0.202\,X_4 + 0.450\,X_5 - 0.056\,X_6 + 0.450\,X_7$$

It can be observed from the first PC that X1 , X3 , X4 contributes to the most in first PC. However, none of the features have a negligible contribution to the first PC. The second principal component Z2 is given by:

$$Z_2 = -0.116\,X_1 + 0.136\,X_2 - 0.374\,X_3 + 0.610\,X_4 - 0.087\,X_5 - 0.667\,X_6 + 0.034\,X_7$$

From the second PC Z2 it can be seen that $X_4$ (Elongation) and $X_6$ (Spread) have the highest contribution in the second PC. The contributions for $X_5$ and $X_7$, $X_3$ are very small and hence negligible. Therefore, Z2 can be rewritten as follows:

$$Z_2 = -0.116\,X_1 + 0.136\,X_2 - 0.374\,X_3 + 0.610\,X_4 - 0.667\,X_6$$

Fig. 6 represents the PC coefficient plot. It visually represents the amount of contribution each feature has on the first two PCs. The figure supports the previous calculation of PCs and it can be clearly seen from the figure that Elongation, Minor and Major length has the highest contributions in the first PC.

The Biplot in Fig. 7 displays a different visual representation of the first two PCs. The axes of biplot represents the first two PCs. The rows of the eigenvector matrix is shown as a vector. Each of the observations in the dataset is drawn as a dot on the plot. The vectors for features namely, Elongation, and Spread show very small angle with the first PC and very large angle with the second PC. This evident supports that the analysis of the PC coefficient plot of Fig. 6. It implies that these two features have a large contribution to the first PC and very small contribution to the second PC.

The vectors representing (Major Length) and (Minor Length) display an inverse trend: they form a larger angle with the first PC and a smaller angle with the second PC, indicating a stronger association with the second PC. Moreover, vectors aligning in the same direction, like Boundary Length, Region, and Convex Region, signify a positive correlation among these features.

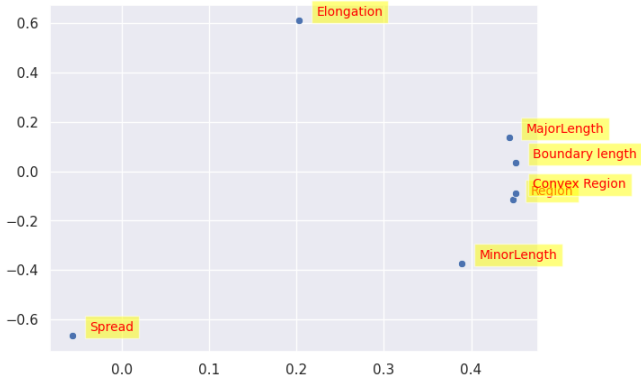



Fig. 6: PC coefficient plot

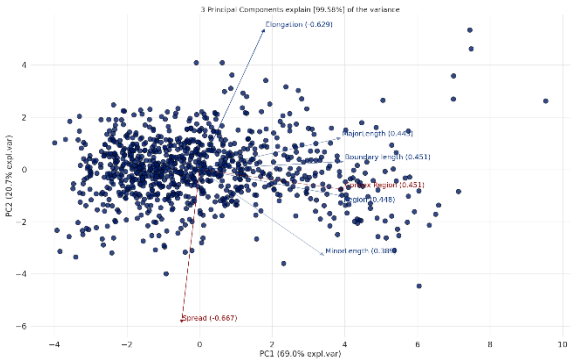



Fig. 7: Biplot

## VII. CLASSIFICATION RESULTS

In this segment, an analysis is conducted on the performance of three widely used classification algorithms using the fast-food dataset. The objective is to examine the impact of Principal Component Analysis (PCA) on this dataset. To assess this impact, the classification algorithms are applied to both the original dataset and the dataset transformed by PCA, specifically with three PCA components. Utilizing the PyCaret library in Python, the classification process entails the division of the original dataset into training and testing subsets, maintaining a split ratio of 75% for training and 25% for testing. Leveraging the functionalities offered by PyCaret, it becomes feasible to generate a comprehensive performance comparison table encompassing all available classification algorithms pertinent to the target dataset. This comparison table facilitates the identification of the most effective model by discerning the algorithm that exhibits the highest accuracy in classification.

The observations derived from Figure 8 showcase the top three classification models, with the highest accuracies, on the fast-food dataset before the application of Principal Component Analysis (PCA). These models are Linear Discriminant Analysis (LDA), Extra Trees Classifier (ET), and Gradient Boosting Classifier (GBC). In contrast, Figure 9 delineates the comparison among classification models subsequent to applying PCA. Notably, the three models exhibiting the highest accuracy on the transformed dataset are Logistic Regression (LR), K-Nearest Neighbors (K-NN), and Random Forest Classifier. Consequently, these specific algorithms are selected for further evaluation throughout the subsequent phases of the experiment. Although both sets of experiments – classification algorithms applied on the original dataset and those applied on the transformed dataset via PCA – are documented within the Kaggle notebook, this report primarily emphasizes the outcomes obtained subsequent to applying PCA. This focus enables a detailed exploration and analysis of the results stemming from the transformed dataset, specifically highlighting the performances of LR, K-NN, and Random Forest Classifier within this context.

| | Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| lr | Logistic Regression | 0.8852 | 0.9384 | 0.8634 | 0.9065 | 0.8829 | 0.7704 | 0.7736 | 0.351 |
| et | Extra Trees Classifier | 0.8763 | 0.9296 | 0.8387 | 0.9114 | 0.8720 | 0.7528 | 0.7572 | 0.181 |
| rf | Random Forest Classifier | 0.8693 | 0.9251 | 0.8246 | 0.9096 | 0.8635 | 0.7387 | 0.7439 | 0.228 |
| gbc | Gradient Boosting Classifier | 0.8641 | 0.9228 | 0.8248 | 0.9018 | 0.8586 | 0.7282 | 0.7346 | 0.126 |
| lda | Linear Discriminant Analysis | 0.8623 | 0.9286 | 0.8600 | 0.8695 | 0.8630 | 0.7245 | 0.7274 | 0.014 |
| ridge | Ridge Classifier | 0.8605 | 0.0000 | 0.8461 | 0.8775 | 0.8591 | 0.7210 | 0.7251 | 0.014 |
| qda | Quadratic Discriminant Analysis | 0.8603 | 0.9246 | 0.8036 | 0.9093 | 0.8519 | 0.7209 | 0.7273 | 0.014 |
| lightgbm | Light Gradient Boosting Machine | 0.8568 | 0.9227 | 0.8280 | 0.8819 | 0.8533 | 0.7136 | 0.7160 | 0.127 |
| ada | Ada Boost Classifier | 0.8534 | 0.9106 | 0.8283 | 0.8774 | 0.8509 | 0.7069 | 0.7097 | 0.103 |
| nb | Naive Bayes | 0.8377 | 0.9099 | 0.7618 | 0.9033 | 0.8254 | 0.6757 | 0.6854 | 0.013 |
| knn | K Neighbors Classifier | 0.8146 | 0.8654 | 0.7867 | 0.8400 | 0.8108 | 0.6294 | 0.6334 | 0.022 |
| dt | Decision Tree Classifier | 0.8075 | 0.8074 | 0.8212 | 0.8022 | 0.8107 | 0.6148 | 0.6166 | 0.015 |
| svm | SVM - Linear Kernel | 0.5105 | 0.0000 | 0.9000 | 0.4572 | 0.6063 | 0.0109 | 0.0240 | 0.015 |
| dummy | Dummy Classifier | 0.5035 | 0.5000 | 1.0000 | 0.5035 | 0.6698 | 0.0000 | 0.0000 | 0.011 |

Fig. 8: Comparison among classification models before applying PCA

| | Model | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC | TT (Sec) |
|---|---|---|---|---|---|---|---|---|---|
| knn | K Neighbors Classifier | 0.8604 | 0.9060 | 0.8108 | 0.9032 | 0.8541 | 0.7211 | 0.7255 | 0.022 |
| lr | Logistic Regression | 0.8570 | 0.9206 | 0.8424 | 0.8748 | 0.8561 | 0.7139 | 0.7179 | 0.018 |
| ridge | Ridge Classifier | 0.8569 | 0.0000 | 0.7969 | 0.9102 | 0.8486 | 0.7142 | 0.7212 | 0.013 |
| lda | Linear Discriminant Analysis | 0.8569 | 0.9213 | 0.7969 | 0.9102 | 0.8486 | 0.7142 | 0.7212 | 0.015 |
| lightgbm | Light Gradient Boosting Machine | 0.8553 | 0.9056 | 0.8283 | 0.8792 | 0.8517 | 0.7107 | 0.7137 | 0.046 |
| nb | Naive Bayes | 0.8552 | 0.9199 | 0.7933 | 0.9089 | 0.8467 | 0.7107 | 0.7171 | 0.014 |
| ada | Ada Boost Classifier | 0.8517 | 0.9077 | 0.8145 | 0.8839 | 0.8465 | 0.7036 | 0.7076 | 0.097 |
| qda | Quadratic Discriminant Analysis | 0.8463 | 0.9112 | 0.7865 | 0.8994 | 0.8370 | 0.6930 | 0.7011 | 0.013 |
| rf | Random Forest Classifier | 0.8429 | 0.9069 | 0.8004 | 0.8800 | 0.8366 | 0.6859 | 0.6909 | 0.226 |
| gbc | Gradient Boosting Classifier | 0.8412 | 0.9094 | 0.8144 | 0.8657 | 0.8376 | 0.6824 | 0.6858 | 0.100 |
| et | Extra Trees Classifier | 0.8357 | 0.9171 | 0.8001 | 0.8651 | 0.8290 | 0.6716 | 0.6763 | 0.191 |
| svm | SVM - Linear Kernel | 0.8184 | 0.0000 | 0.8387 | 0.8202 | 0.8257 | 0.6367 | 0.6430 | 0.015 |
| dt | Decision Tree Classifier | 0.8007 | 0.8004 | 0.8107 | 0.8023 | 0.8039 | 0.6010 | 0.6052 | 0.012 |
| dummy | Dummy Classifier | 0.5035 | 0.5000 | 1.0000 | 0.5035 | 0.6698 | 0.0000 | 0.0000 | 0.010 |

Fig. 9: Comparison among classification models after applying PCA

```
lr2pca=create_model('lr')
```

| | Accuracy | AUC | Recall | Prec. | F1 | Kappa | MCC |
|---|---|---|---|---|---|---|---|
| 0 | 0.7895 | 0.8608 | 0.7931 | 0.7931 | 0.7931 | 0.5788 | 0.5788 |
| 1 | 0.8772 | 0.9212 | 0.8621 | 0.8929 | 0.8772 | 0.7545 | 0.7549 |
| 2 | 0.8772 | 0.9495 | 0.9310 | 0.8438 | 0.8852 | 0.7539 | 0.7581 |
| 3 | 0.8246 | 0.8966 | 0.7586 | 0.8800 | 0.8148 | 0.6499 | 0.6563 |
| 4 | 0.8246 | 0.8966 | 0.8276 | 0.8276 | 0.8276 | 0.6490 | 0.6490 |
| 5 | 0.9474 | 0.9754 | 0.8929 | 1.0000 | 0.9434 | 0.8945 | 0.8995 |
| 6 | 0.8750 | 0.9286 | 0.8929 | 0.8621 | 0.8772 | 0.7500 | 0.7505 |
| 7 | 0.8571 | 0.9260 | 0.8214 | 0.8846 | 0.8519 | 0.7143 | 0.7161 |
| 8 | 0.9286 | 0.9643 | 0.8929 | 0.9615 | 0.9259 | 0.8571 | 0.8593 |
| 9 | 0.8571 | 0.8903 | 0.8929 | 0.8333 | 0.8621 | 0.7143 | 0.7161 |
| Mean | 0.8658 | 0.9209 | 0.8565 | 0.8779 | 0.8658 | 0.7316 | 0.7339 |
| SD | 0.0451 | 0.0339 | 0.0515 | 0.0594 | 0.0445 | 0.0901 | 0.0908 |

Fig. 10: LR metrics score after hyperparameter tuning

To enhance model performance, the refinement of hyperparameters stands as a pivotal undertaking. Within PyCaret, the process of hyperparameter tuning encompasses a structured sequence of three integral steps: model creation, parameter tuning, and subsequent performance evaluation. Initially, a classification model is instantiated for each algorithm. Subsequently, the 'tune_model()' function is employed to fine-tune the model, optimizing its hyperparameters for enhanced efficacy. This function adeptly explores a pre-defined search space to pinpoint optimal hyperparameters while assessing model performance through the prism of stratified K-fold cross-validation, typically set at a default of 10 folds in PyCaret. Moreover, the Logistic Regression (LR) model undergoes tuning with an L2 penalty, a regularization mechanism aimed at forestalling overfitting issues. For the K-Nearest Neighbors (K-NN) algorithm, the tuning process focuses on the adjustment of the number of nearest neighbors (K value), while the 'reg_param' parameter for regularization undergoes fine-tuning in the context of the Random Forest classifier. The discernible outcomes portrayed in Figure 10 illustrate the superiority of the tuned LR model metrics over the baseline metrics of the initial, non-tuned model, signifying the positive impact of hyperparameter optimization on model performance.

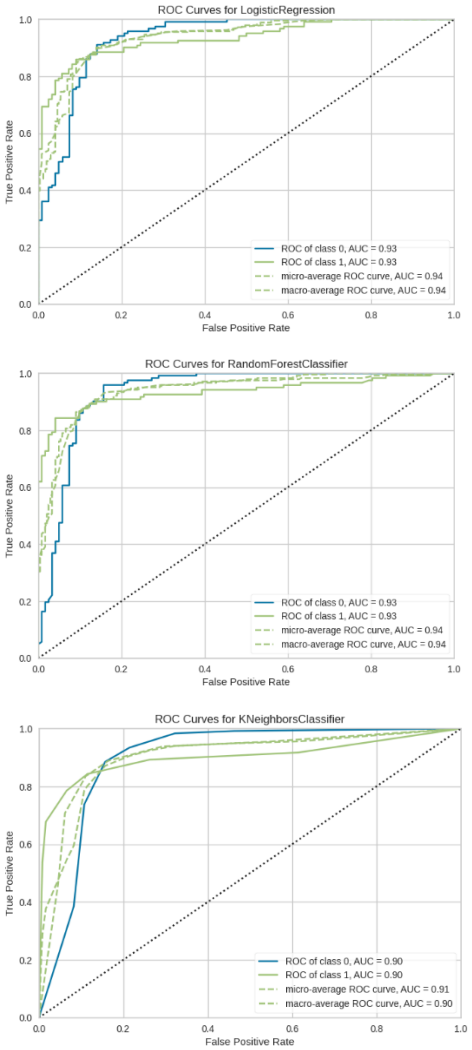



Fig. 11: Decision Boundaries of the three algorithms applied on transformed dataset

Figure 11 delineates the Receiver Operating Characteristic (ROC) curve generated by the model on the transformed dataset. This graphical representation offers insight into the performance of a classification model across various classification thresholds. It distinctly showcases the variances among the decision boundaries formed by the algorithms under consideration. Within the figure, the discernible differences in decision boundaries indicate that Logistic Regression boasts a more optimal decision boundary compared to K-Nearest Neighbors (K-NN) and Random Forest. The efficacy of Logistic Regression's decision boundary is apparent in its enhanced ability to accurately segregate data instances belonging to both classes. Moreover, precision and recall serve as pivotal metrics used to evaluate the classification performance. Precision denotes the ratio of relevant instances among all retrieved instances, emphasizing the accuracy of positive predictions. Conversely, recall represents the ratio of retrieved instances among all relevant instances, highlighting the model's ability to capture all positive instances effectively. These complementary measurements, when assessed together, offer a comprehensive evaluation of the classification model's performance. The obtained results from precision and recall are presented using the confusion matrices Fig. 12. The confusion matrix is defined as the matrix providing the mix of predicted vs. actual class instances. It illustrates correct and incorrect predictions with count values and breaks down for each class.

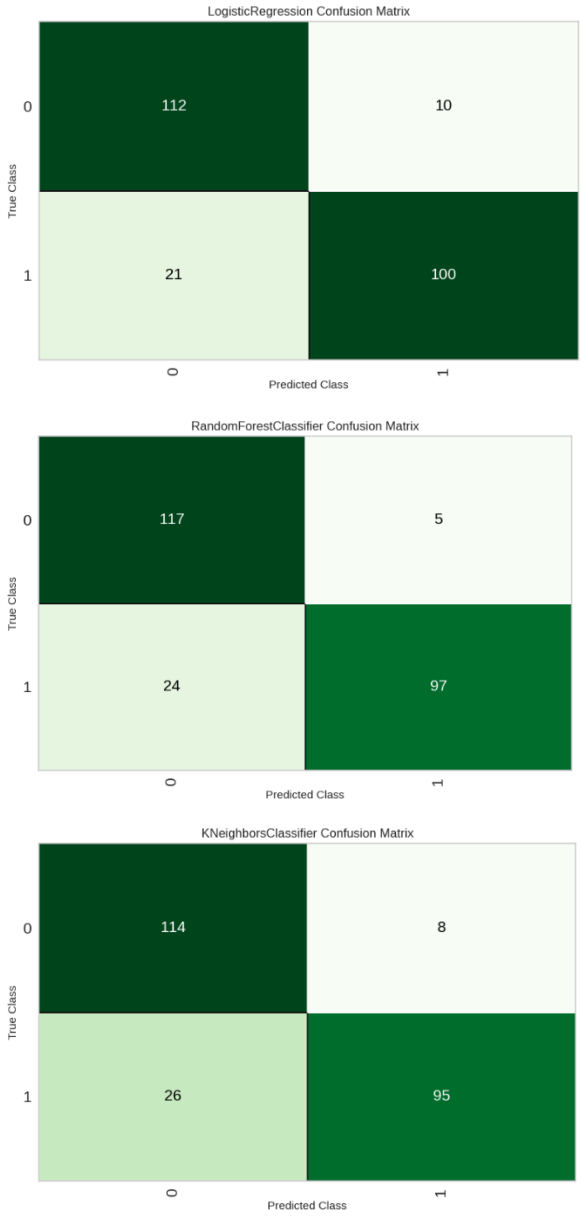



Fig. 12: Confusion matrices of the three classification algorithms applied on transformed dataset

The Fig. 12 shows the confusion matrix tables for the three algorithms which were applied on transformed dataset. The confusion matrices for the original dataset can be found in the Kaggle notebook. In the figure, the horizontal axis represents the class prediction and vertical axis represents the true label. First of all, the comparison model on Fig. 9 shows that LR outperforms all other compared models including K-NN and Random

Forest Classifier. This observation is also supported by the confusion matrices of Fig.12. LR misclassified the lowest numbers of instances. For example, LR misclassfied 4 instances from class 0 as class 1 and 5 instances of class 1 are misclassified as class 0. On the other hand, K-NN misclassified 6 instances of class 0 and 6 instances of class 1 whereas Random Forest Classifier misclassified 7 instances of class 0 and 4 instances of class 1. Another measurement of the performance evaluation is F1- score. The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is a great metric to compare the results among the classifiers. For example: classifier A has a higher recall, and classifier B has higher precision. In this situation, the F1-score helps to determine the better classifier. The function of F1- score can be defined as below:

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall}$$

From Figures 8 and 9, a notable enhancement in the F1-score of the Logistic Regression (LR), K-Nearest Neighbors (K-NN), and Random Forest Classifier is evident subsequent to the application of Principal Component Analysis (PCA). This improvement infers that dimension reduction through PCA contributes to a reduction in inter-feature dependencies. Moreover, after fine-tuning these models with their optimal hyperparameters, the F1-scores of LR and K-NN exhibit further augmentation. These trends collectively underline the advantages derived from employing PCA and optimizing hyperparameters in enhancing model performance. As a culminating analytical step, Figure 13 showcases the Receiver Operating Characteristic (ROC) curve for the LR algorithm. The ROC curve serves as a graphical representation illustrating the performance of a classification model across various classification thresholds. This curve delineates the True Positive Rate against the False Positive Rate, pivotal components used in constructing the confusion matrix. Hence, both the ROC curve and the confusion matrix offer distinct yet interconnected visual representations of the model's performance. While the ROC curves for K-NN and Random Forest Classifier are accessible in the Kaggle notebook, Figure 13 exclusively displays the ROC curve for LR. This curve not only exhibits the False Positive Rate (x-axis) versus the True Positive Rate (y-axis) across different threshold values but also includes the macro and micro average curves. The ROC curve's accompanying Area Under the Curve (AUC) values demonstrate LR's superior predictive capability for both classes, achieving a precision level of 98%.

## VIII. EXPLAINABLE AI WITH SHAPLEY

A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

Model interpretability is an important metric in the context of ML. There are several ways of enhancing the explain ability of a model and feature importance is one of them. Feature importance helps to estimate the contribution of each feature in the prediction process. Hence in order to get an overview of the most important features on the PCs, we use the SHAP values by importing the open source "Shap" library of Python. SHAP (Shapley Additive Explanations) is a method that was first introduced by Lundberg and Lee in 2016. SHAP explains individual predictions based on the optimal Shapley values using the concept of game theory. Specifically, SHAP can explain the prediction of an instance x by computing the contribution of each feature to the prediction. Borrowing the concept of game theory, each feature of a dataset act as players in a coalition. A player can be an individual feature value, e.g. for tabular data. or a group of feature values. Shapley values describes how to adequately distribute the prediction among the feature set. The Shap library of Python is still in its development stage and it only supports tree-based models (i.e; decision tree, random forest, extra trees classifier) for binary classification problem. Since, our fast food dataset is a binary classification problem, Shap analysis cannot be performed on LR, KNN, and Random Forest classifier. Therefore, for the Shap analysis, we choose the fourth best model of the transformed dataset that is "Extra trees classifier (ET)". Similar to other models, at first, an ET model is created and tuned with ideal hyperparameters. Then the tuned model is passed to the Shap library for producing the interpretation plots. For our case, each PC acts as a player in the coalition.

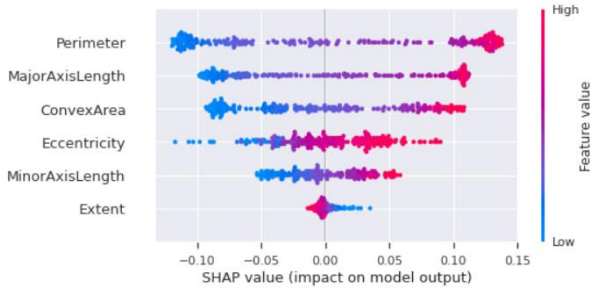



Fig. 14: Summary plot

Fig. 14 displays the summary plot of SHAP values. The summary plot combines feature importance with feature effects. Each point on the summary plot is a Shapley value for a PC and an instance. The y-axis represents the PCs and Shapley values are positioned on the x-axis. More specifically, component_1 represents the first PC, component_2 represents the second PC and component_3 represents the third PC. We can observe some jittered overlapping points in the direction of y-axis which indicates the distribution of the Shapley values per PC. All the PC's are ordered according to their importance. This observation supports the pareto plot and scree plot which indicates that the first PC holds the most feature variance. The red color indicates high PC value and blue color indicates low PC value. To interpret the summary plot, we can say that a low level of PC value has a high and positive impact on the classification. Similarly, a high level of PC value has a low and negative impact on the classification. More clearly, PCs are negatively correlated with the target variable.

Fig. 15: Force plot for a single observation

Fig 15 portrays a force plot dedicated to a singular observation. It's important to underscore that this plot is specifically tailored for one individual data point. Within this visualization, each feature contributes to either augmenting or reducing the model output from its base value. The term "base value" denotes the predicted value when no information about the features is available for the current output. In essence, it represents the mean prediction derived from the test set, which in this instance is established at 0.3. The prominently displayed value of 0.55 in bold font signifies the model's score for this particular observation. The interpretation of this plot involves understanding that higher scores incline the model towards predicting a class label of 1, while lower scores prompt a prediction of 0. In the visualization, the red color indicates that the second Principal Component (PC) contributes to a higher prediction, while the blue color associated with the first PC suggests a tendency to drive the prediction lower. Despite these contributions, this specific observation is classified as class 0 primarily because it is predominantly influenced towards the left direction. It is crucial to recognize that while this plot offers insights into the factors influencing a single prediction, it does not encapsulate the predictive output of the entire model.

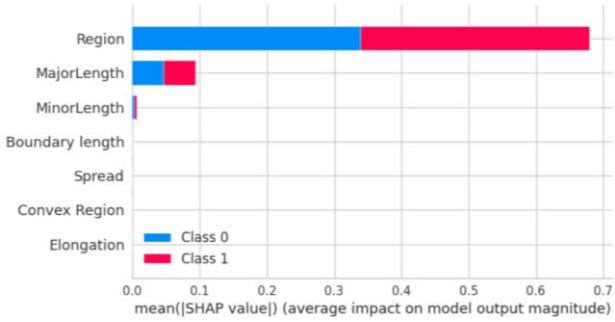



Fig. 16: Combined force plot

In Figure 16, the combined force plot of all Principal Components (PCs) is showcased. This composite plot integrates individual force plots, each rotated by 90 degrees, and aligns them horizontally. The y-axis within this plot corresponds to the x-axis of the individual force plots. With a total of 154 data points comprising the transformed test set, the x-axis delineates these 154 observations.

The primary objective of this combined force plot is to elucidate the impact of each Principal Component (PC) on the current prediction. Notably, values represented in blue signify a positive influence on the prediction, whereas those depicted in red indicate a negative influence on the prediction. This visualization aids in discerning the directional impact and relative strength of each PC in contributing to the predictive outcome.

## IX. Conclusion

In conclusion, PCA and three popular classification algorithms are applied on fast food dataset. At first, PCA is applied on the original dataset. The first two PC's apprehends 89.8% variance of the data. Hence, the feature-set is reduced to 2 from 7. Extensive experiments are conducted on the first two PC's and different plots are generated to validate the obtained results from different perspectives. To move forward, three classification algorithms, LR, K-NN and RFC, are applied on the original dataset as well as transformed dataset with first three components. Each algorithm is tuned with the ideal hyperparameter settings and performance evaluation is conducted by comparing confusion matrices, ROC curves and Fig. 16: Combined force plot F1-scores. It is observed that after hyperparameter tuning performance metrics score of each algorithm has improved significantly. The LDA, ET and GBC algorithms performed the best on the original dataset. Interestingly, after applying PCA, LR, K-NN and RFC performed the highest and showed the best performance metrics. Finally, in order to increase the interpretability of the model, several interpretation plots are produced using explainable AI values.

## References

[1] A. B. Hamza, "Advanced statistical approaches to quality," Dec 2022.

[2] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," Pattern recognition, vol. 40, no. 7, pp. 2038–2048, 2007.

[3] I. Jolliffe and J. Cadima, “Principal component analysis: A review and recent developments,” Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 374, p. 20150202, 04 2016.

[4] P. Karsmakers, K. Pelckmans, and J. A. Suykens, “Multiclass kernel logistic regression: a fixed-size implementation,” in 2007 International Joint Conference on Neural Networks, pp. 1756–1761, IEEE, 2007.

[5] H. Zhang, A. C. Berg, M. Maire, and J. Malik, “Svmknn: Discriminative nearest neighbor classification for visual category recognition,” in 2012.