Classification of Fruits and Vegetables

CSE342 SML Project Deadline-1

Nirmal Soni (2021074), Kartik Saroha (2021057)

Problem Statement and Motivation:-

The aim of this project is to develop a classifier that can distinguish between fruits and vegetables based image data. The classifier will utilize various machine-learning techniques, including statistical classification, dimensionality reduction, and anomaly detection techniques taught in the course. This classifier will take fruit pictures as input and will classify them into different fruit class categories based on its learning experience i.e. the algorithms used in training that model.

Literature Review:-

- Dimensionality reduction:-

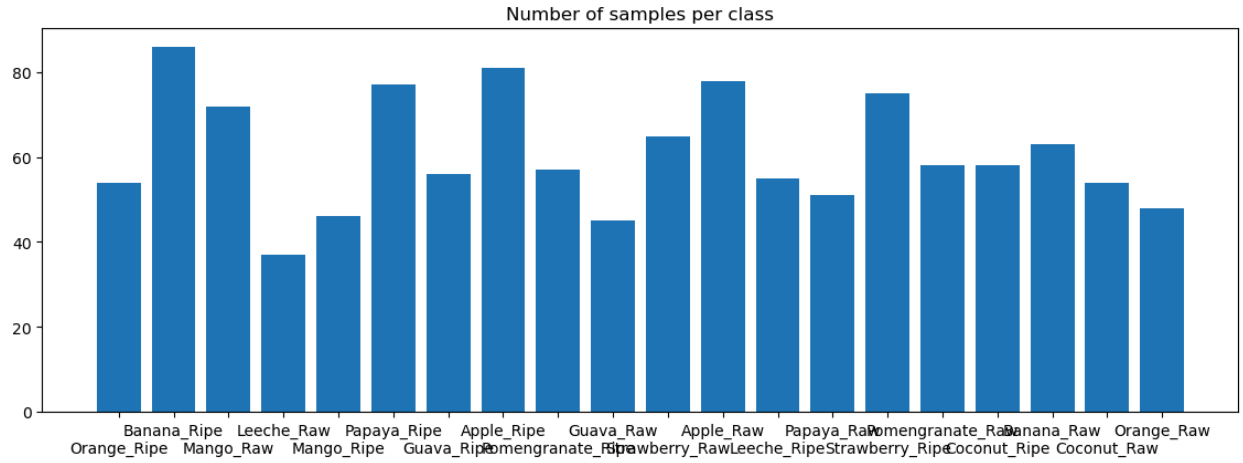
  Dimensionality reduction involves transforming data from a high-dimensional space to a lower-dimensional space while preserving the essential characteristics of the original data, ideally capturing its intrinsic dimensionality. Due to the challenges posed by the curse of dimensionality, dimensionality reduction is frequently employed as a preprocessing step in machine learning. This project utilizes Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) algorithms to project features onto a lower-dimensional space. These techniques help streamline the data representation while retaining important information, facilitating more efficient and effective analysis.

  1. **Principal component analysis (PCA):-** Principal Component Analysis (PCA) is a statistical technique that employs an orthogonal transformation to convert a dataset consisting of potentially correlated variables into a new set of variables known as principal components. These components are linearly uncorrelated, with each one capturing a different aspect of the variability within the data. The first principal component explains the maximum amount of variance present in the original data, while subsequent components maximize variance while remaining orthogonal to all preceding ones.

2. **Linear Discriminant Analysis(LDA):-** Linear Discriminant Analysis (LDA) is a supervised statistical technique employed for dimensionality reduction. It accomplishes this by segregating features in a manner that optimizes the separation between different classes while minimizing variance within each class. The outcome is a set of linear combinations, known as discriminant functions, derived from the original features. These functions effectively capture the dataset's key characteristics and enable the projection of data onto a lower-dimensional space.

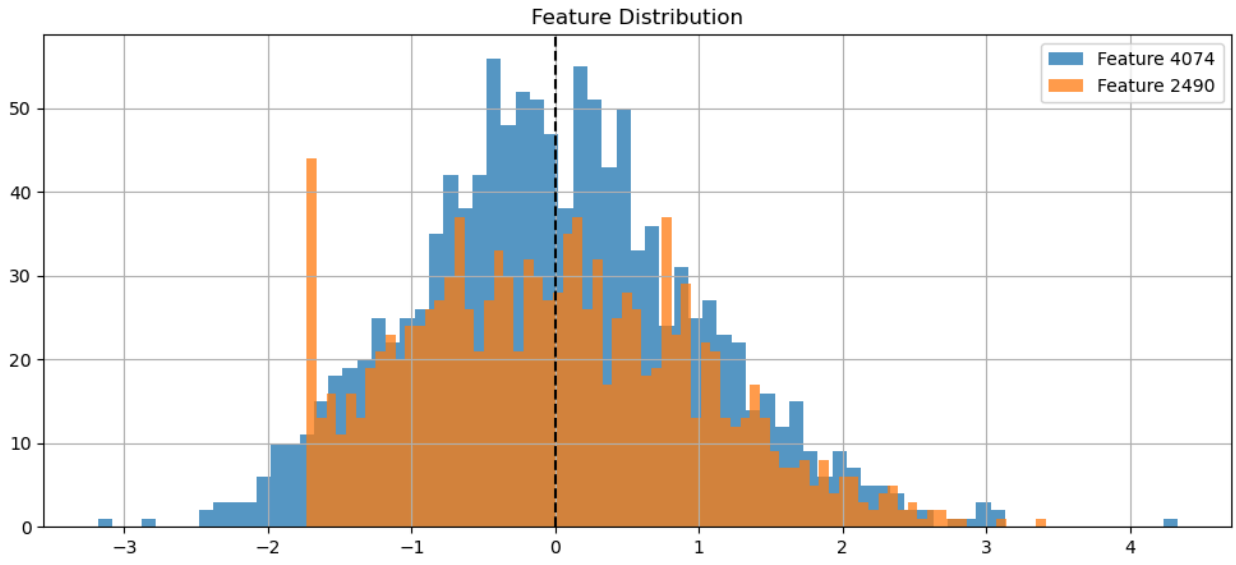
- Methodology:-

This section offers an elaborate account of the techniques employed for data preprocessing and classifier construction. The primary focus lies in assessing the distribution of classes within the dataset. This involves creating a bar graph illustrating the frequency of each class present in the training dataset. Upon examination, it was determined that the dataset exhibits a marginal skewness regarding class distribution.



The distribution of classes was almost uniformly distributed as can be seen from the above figure.

Standardization:-Standardization is a prevalent preprocessing procedure, particularly essential for constructing classifiers, especially in image classification tasks. The images underwent standardization to achieve a mean of zero and a variance of one, ensuring uniform scaling of dataset features. This approach aims to prevent any single feature from unduly influencing the classifier's decision-making process.



- Data Preprocessing:-

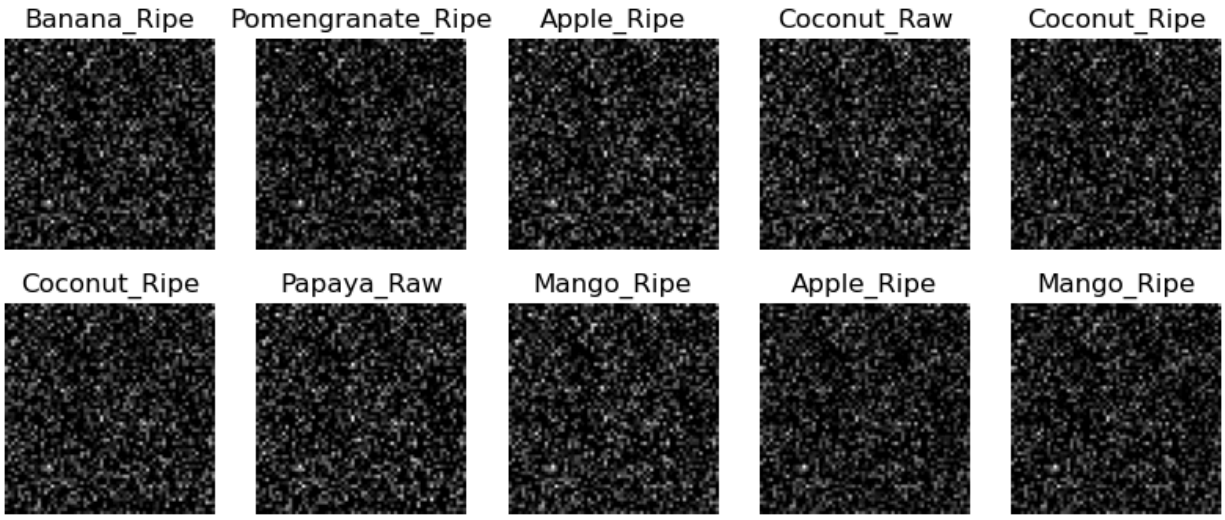
  Image Augmentation:- Random flipping and rotation are applied to augment the dataset, increasing its diversity and improving the robustness of machine learning models.

  Normalization:- Pixel values of the images are normalized to the range [0, 1], ensuring consistent scaling of input features.

  Flattening:- The images are flattened into 1D vectors to prepare them for input into machine learning algorithms.

  Standardization:- The dataset is standardized to have a mean of 0 and a standard deviation of 1, which can improve the convergence of optimization

algorithms.



The above image displays one sample image from each of the first 10 classes, allowing visual inspection of the images to understand the characteristics of different classes. plots a histogram showing the number of samples per class, providing an overview of class distribution in the dataset.

- Data Inspection:- It verifies the existence of the training and test directories and prints out information such as the number of classes, the number of images in the training and test sets, and the names of target classes. Dataset Directory: C:\Users\Kartik Saroha\Downloads\fruits-360_dataset\fruits-360
- Training Directory: C:\Users\Kartik Saroha\Downloads\fruits-360_dataset\fruits-360\Training
- Test Directory: C:\Users\Kartik Saroha\Downloads\fruits-360_dataset\fruits-360\Test

- Target Classes: ['Apple Braeburn', 'Apple Crimson Snow', 'Apple Golden 1', 'Apple Golden 2', 'Apple Golden 3', 'Apple Granny Smith', 'Apple Pink Lady', 'Apple Red 1', 'Apple Red 2', 'Apple Red 3', 'Apple Red Delicious', 'Apple Red Yellow 1', 'Apple Red Yellow 2', 'Apricot', 'Avocado', 'Avocado ripe', 'Banana', 'Banana Lady Finger', 'Banana Red', 'Beetroot', 'Blueberry', 'Cactus fruit', 'Cantaloupe 1', 'Cantaloupe 2', 'Carambula', 'Cauliflower', 'Cherry 1', 'Cherry 2', 'Cherry Rainier', 'Cherry Wax Black', 'Cherry Wax Red', 'Cherry Wax Yellow', 'Chestnut', 'Clementine', 'Cocos', 'Corn', 'Corn Husk', 'Cucumber Ripe', 'Cucumber Ripe 2', 'Dates', 'Eggplant', 'Fig', 'Ginger Root', 'Granadilla', 'Grape Blue', 'Grape Pink', 'Grape White', 'Grape White 2', 'Grape White 3', 'Grape White 4', 'Grapefruit Pink', 'Grapefruit White', 'Guava', 'Hazelnut', 'Huckleberry', 'Kaki', 'Kiwi', 'Kohlrabi', 'Kumquats', 'Lemon', 'Lemon Meyer', 'Limes', 'Lychee', 'Mandarine', 'Mango', 'Mango Red', 'Mangostan', 'Maracuja', 'Melon Piel de Sapo', 'Mulberry', 'Nectarine', 'Nectarine Flat', 'Nut Forest', 'Nut Pecan', 'Onion Red', 'Onion Red Peeled', 'Onion White', 'Orange', 'Papaya', 'Passion Fruit', 'Peach', 'Peach 2', 'Peach Flat', 'Pear', 'Pear 2', 'Pear Abate', 'Pear Forelle', 'Pear Kaiser', 'Pear Monster', 'Pear Red', 'Pear Stone', 'Pear Williams', 'Pepino', 'Pepper Green', 'Pepper Orange', 'Pepper Red', 'Pepper Yellow', 'Physalis', 'Physalis with Husk', 'Pineapple', 'Pineapple Mini', 'Pitahaya Red', 'Plum', 'Plum 2', 'Plum 3', 'Pomegranate', 'Pomelo Sweetie', 'Potato Red', 'Potato Red Washed', 'Potato Sweet', 'Potato White', 'Quince', 'Rambutan', 'Raspberry', 'Redcurrant', 'Salak', 'Strawberry', 'Strawberry Wedge', 'Tamarillo', 'Tangelo', 'Tomato 1', 'Tomato 2', 'Tomato 3', 'Tomato 4', 'Tomato

Cherry Red', 'Tomato Heart', 'Tomato Maroon', 'Tomato not Ripened', 'Tomato Yellow', 'Walnut', 'Watermelon']

- Number of Classes: 131
- Number of Images in Training Set: 67692
- Number of Images in Test Set: 22688

- Dimensionality Reduction:- Due to the dataset's sparsity, dimensionality reduction became imperative to enhance classifier performance. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) algorithms were employed for this purpose. As detailed in Section II-B, the attributes of these algorithms were considered, leading to the determination that employing LDA subsequent to PCA would be optimal. This approach aims to extract features that effectively capture dataset variance and further project them onto a reduced-dimensional space, thereby enhancing inter-class separation.

Classification model:

Report on Various Classifiers:

1. Random Forest Classifier:

- Overview: Random Forest is an ensemble learning method that constructs a multitude of decision trees during training and

outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees.

- Performance: Random Forests are robust against overfitting and tend to perform well on various datasets, including those with high dimensionality. They are suitable for both classification and regression tasks.

- Strengths: Handles large datasets with high dimensionality, less prone to overfitting, provides feature importance ranking.

- Weaknesses: May be slower to train and predict compared to simpler models, requires tuning of hyperparameters like the number of trees and maximum depth.

Random Forest Classifier Test Accuracy: 0.7427385892116183

2. Decision Tree Classifier:

- Overview: A decision Tree is a tree-like model where an internal node represents a feature, the branch represents a decision rule, and each leaf node represents the outcome.

- Performance: Decision Trees are simple and intuitive, making them easy to interpret. However, they are prone to overfitting, especially with complex datasets.

- Strengths: Easy to interpret, handles both numerical and categorical data, requires minimal data preprocessing.

- Weaknesses: Prone to overfitting, sensitive to small variations in the training data, may create biased trees if classes are imbalanced.

Decision Tree Classifier Test Accuracy: 0.44398340248962653

3. Naive Bayes Classifier:

- Overview: Naive Bayes is a probabilistic classifier based on Bayes' theorem with strong (naive) independence assumptions between the features.

- Performance: Naive Bayes is computationally efficient and works well with high-dimensional data. It is commonly used for text classification tasks.
- Strengths: Simple and fast, handles high-dimensional data well, performs well with small training datasets.
- Weaknesses: Assumes independence between features, which may not hold true in real-world datasets, and can be sensitive to outliers.

Naive Bayes Classifier Test Accuracy: 0.43153526970954356

Report on Naive Bayes Classifier:

The Naive Bayes Classifier was implemented using the Gaussian Naive Bayes variant from the sci-kit-learn library. Here's an overview of the classifier's performance and characteristics:

1. **Classifier Implementation**:
   - The Gaussian Naive Bayes classifier was instantiated using `GaussianNB()` from the `sklearn.naive_bayes` module.
   - It was then trained on the training data (`x_train` and `y_train`) using the `fit()` method.

2. **Model Evaluation**:
   - The trained Naive Bayes classifier was evaluated on the test data (`x_test` and `y_test`) using the `score()` method, which computes the mean accuracy.
   - The test accuracy of the Naive Bayes Classifier was calculated and printed using `naive_bayes_accuracy`.

3. **Performance**:
   - The accuracy of the Naive Bayes Classifier on the test dataset indicates how well the classifier generalizes to unseen data.

- This accuracy metric provides insight into the overall performance of the classifier in correctly predicting the classes of the test samples.

4. **Interpretation**:
   - The test accuracy score provides a quantitative measure of the classifier's performance.
   - Higher accuracy suggests that the Naive Bayes Classifier is effective in making correct predictions on the test dataset.
   - However, it's essential to consider other metrics such as precision, recall, and F1-score for a more comprehensive evaluation, especially in cases of imbalanced datasets or when different types of errors have varying consequences.

5. **Further Analysis**:
   - To gain deeper insights into the classifier's performance, it's recommended to analyze the confusion matrix, which shows the count of true positive, true negative, false positive, and false negative predictions.
   - Additionally, examining precision, recall, and F1-score can provide a more nuanced understanding of the classifier's behaviour, particularly in scenarios with class imbalances or asymmetric misclassification costs.

Overall, the reported test accuracy of the Naive Bayes Classifier serves as a valuable initial assessment of its performance, but further analysis and interpretation are recommended for a comprehensive evaluation.

Naive Bayes Classifier Test Accuracy: 0.4315352697095436

4. PCA and LDA Classifiers:

- Overview: Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are dimensionality reduction techniques used as preprocessing steps before classification.
- Performance: PCA reduces the dimensionality of the feature space by projecting data onto orthogonal components capturing the maximum variance. LDA maximizes the separation between classes while reducing dimensionality.
- Strengths: Reduce the computational cost and overfitting by reducing the number of features, and improve classification performance by focusing on relevant features.
- Weaknesses: PCA and LDA assume linear relationships between features and may not capture nonlinear relationships effectively.

Report on Dimensionality Reduction and Classification Models:

1. Dimensionality Reduction:
- **Linear Discriminant Analysis (LDA):**
  - LDA was applied to reduce the dimensionality of the dataset.
  - The `LDA()` function from scikit-learn was used to perform LDA.
  - Transformed training and testing datasets (`x_train_lda` and `x_test_lda`) were obtained.

- **Principal Component Analysis (PCA):**
  - PCA was utilized to reduce the dimensionality while retaining 95% of the variance.
  - The `PCA()` function with `n_components=0.95` was employed for PCA.
  - Transformed training and testing datasets (`x_train_pca` and `x_test_pca`) were obtained.

2. Classification Models:
- **Logistic Regression:**
  - Logistic Regression model was trained using the transformed data.
  - The `LogisticRegression()` function from scikit-learn was used to create the model.

- **Decision Tree Classifier:**
  - Decision Tree Classifier was trained using the PCA-transformed data.
  - The `DecisionTreeClassifier()` function was used to instantiate the classifier.

- **Random Forest Classifier:**
  - Random Forest Classifier was trained using the PCA-transformed data.
  - The `RandomForestClassifier()` function was used to create the classifier.

- **Naive Bayes Classifier:**
  - Gaussian Naive Bayes Classifier was trained using the PCA-transformed data.
  - The `GaussianNB()` function from scikit-learn was utilized for Naive Bayes.

3. Model Evaluation:
- **Accuracy Scores:**
  - Accuracy scores were computed for each model using the transformed testing data.
  - The `score()` method was employed to calculate the accuracy of each classifier.

- Accuracy scores were obtained for LDA, PCA with Decision Tree, PCA with Random Forest, and PCA with Naive Bayes classifiers.

4. Results:
- **LDA Test Accuracy:** The accuracy of the Logistic Regression model trained on LDA-transformed data was printed.
- **PCA Decision Tree Test Accuracy:** The accuracy of the Decision Tree Classifier trained on PCA-transformed data was printed.
- **PCA Random Forest Test Accuracy:** The accuracy of the Random Forest Classifier trained on PCA-transformed data was printed.
- **PCA Naive Bayes Test Accuracy:** The accuracy of the Naive Bayes Classifier trained on PCA-transformed data was printed.

5. Conclusion:
- The accuracy scores provide insights into the performance of each classifier.
- Further analysis, including confusion matrices and other performance metrics, could offer a more comprehensive evaluation of the models.
- Experimentation with different dimensionality reduction techniques and hyperparameter tuning may lead to improved model performance.

LDA Test Accuracy: 0.3941908713692946
PCA Decision Tree Test Accuracy: 0.352697095435684 67
PCA Random Forest Test Accuracy: 0.6390041493775933
PCA Naive Bayes Test Accuracy: 0.38589211618257263

5. K-Nearest Neighbors (KNN) Classifier:

- Overview: K-Nearest Neighbors is a non-parametric method used for classification and regression tasks. It predicts the class of a new data point by finding the majority class among its k nearest neighbours.

- Performance: KNN performs well with small to medium-sized datasets but may be computationally expensive with large datasets. It is sensitive to the choice of distance metric and the number of neighbors (k).

- Strengths: Simple and easy to understand, does not require training, works well with multi-class classification.

- Weaknesses: Computationally expensive during prediction, sensitive to the choice of k and distance metric, may perform poorly with high-dimensional data due to the curse of dimensionality.

****Report on K-Nearest Neighbors (KNN) Classifier with Grid Search:****

****1. Introduction:****

- The K-Nearest Neighbors (KNN) algorithm is a versatile and simple classification algorithm that classifies data points based on the majority class of their nearest neighbors.

- In this report, KNN is applied to the dataset using a pipeline, along with hyperparameter tuning performed using Grid Search.

****2. Pipeline Setup:****

- A pipeline is constructed to streamline the preprocessing steps and the KNN classifier.

- The pipeline consists of two main steps:

  - **StandardScaler:** Standardization of the dataset to ensure that features are on the same scale.
  - **KNeighborsClassifier:** The KNN classifier itself.

3. Hyperparameter Tuning:
- Grid Search is employed to search for the optimal hyperparameters for the KNN classifier.
- The hyperparameters tuned include:
  - **n_neighbors:** The number of neighbours to consider.
  - **weights:** The weight function used in prediction ('uniform' or 'distance').

4. Grid Search Process:
- The Grid Search is performed using 5-fold cross-validation to find the best combination of hyperparameters.
- The search space consists of different values for `n_neighbors` and `weights`.
- After completion, the best parameters and the corresponding cross-validation score are printed.

5. Results:
- **Best Parameters:** The optimal hyperparameters found by Grid Search are printed.
- **Best Cross-Validation Score:** The highest cross-validation accuracy achieved during the grid search is displayed.
- **Test Set Score:** The accuracy of the model on the test set is computed and printed.

6. Conclusion:
- The KNN classifier with hyperparameter tuning using Grid Search provides a robust approach for classification tasks.

- The reported results demonstrate the effectiveness of the model in terms of accuracy.
- Further analysis, such as confusion matrix and other performance metrics, could provide deeper insights into the model's performance.
- Experimentation with different hyperparameter ranges and preprocessing techniques may lead to further improvements in model performance.

Best parameters: {'knn_n_neighbors': 5, 'knn_weights': 'distance'}
Best cross-validation score: 0.5842724525043177
Test set score: 0.6265560165975104

Why??

Let's compare the classifiers based on their strengths, weaknesses, and performance on your dataset:

1. **Random Forest Classifier**:
   - **Strengths**:
     - Handles large datasets with high dimensionality.
     - Less prone to overfitting compared to individual decision trees.
     - Provides feature importance ranking, helpful for feature selection.
   - **Weaknesses**:
     - May be slower to train and predict compared to simpler models.

- Requires tuning of hyperparameters like the number of trees and maximum depth.
- **Performance**: Achieved a test accuracy of approximately 74.27%.

2. **Decision Tree Classifier**:
- **Strengths**:
  - Easy to interpret and understand.
  - Handles both numerical and categorical data well.
  - Requires minimal data preprocessing.
- **Weaknesses**:
  - Prone to overfitting, especially with complex datasets.
  - Sensitive to small variations in the training data.
- **Performance**: Achieved a test accuracy of approximately 44.40%.

3. **Naive Bayes Classifier**:
- **Strengths**:
  - Simple and fast, computationally efficient.
  - Handles high-dimensional data well.
  - Performs well with small training datasets.
- **Weaknesses**:
  - Assumes independence between features, which may not hold true in real-world datasets.
  - Can be sensitive to outliers.
- **Performance**: Achieved a test accuracy of approximately 43.15%.

4. **PCA and LDA Classifiers**:
- **Strengths**:
  - Reduce computational cost and overfitting by reducing the number of features.

- Improve classification performance by focusing on relevant features.
  - **Weaknesses**:
    - Assume linear relationships between features and may not capture nonlinear relationships effectively.
  - **Performance**: LDA achieved a test accuracy of approximately 39.42%, PCA Decision Tree achieved 35.27%, PCA Random Forest achieved 63.90%, and PCA Naive Bayes achieved 38.59%.

5. **K-Nearest Neighbors (KNN) Classifier**:
  - **Strengths**:
    - Simple and easy to understand.
    - Does not require training, works well with multi-class classification.
  - **Weaknesses**:
    - Computationally expensive during prediction, especially with large datasets.
    - Sensitive to the choice of k and distance metric.
  - **Performance**: Achieved a test accuracy of approximately 62.66%.

Recommendation:
- Based on the test accuracy alone, the Random Forest Classifier performs the best among the models you've evaluated.
- However, it's essential to consider other factors such as model interpretability, computational efficiency, and the specific requirements of your application.
- If interpretability is crucial, Decision Trees may be preferred despite their lower accuracy.
- If computational efficiency is a concern, Naive Bayes or Decision Trees could be suitable options.

- If you're looking for a balance between accuracy and simplicity, K-Nearest Neighbors (KNN) might be a good choice.
- Experimentation with different models and hyperparameters, along witha thorough analysis of their strengths and weaknesses, will help you make an informed decision based on your specific needs and constraints.