1. Introduction:

Fruit is a vital component of the human diet, offering a plethora of nutritional benefits. They are the mature ovaries of flowering plants, usually containing seeds and are enveloped in edible flesh. The diversity in fruit varieties is vast, encompassing a range of shapes, sizes, colors, and flavors. Common types of fruits include berries, drupes, pomes, and citrus fruits. Their nutritional profile is rich in vitamins, minerals, fiber, and antioxidants, contributing significantly to overall health. Regular consumption of fruits has been linked to numerous health benefits, including enhanced digestion, reduced risk of chronic diseases, and strengthened immune function.

In contemporary agriculture, the use of pesticides, insecticides, and weedicides is prevalent to boost fruit production. While these chemicals aid in increasing yield, their overuse can compromise the quality of the fruits. Ensuring high product quality is essential for the reliable marketing of fruits and crops. Quality is determined by comparing the standard of a product against others of a similar kind, reflecting its degree of excellence.

Fruit Quality Detection (FQD) leverages technologies such as image processing, machine learning, and deep learning algorithms to assess and classify the quality of fruits. FQD systems distinguish between high and low-quality fruits by extracting key features from images, including color, texture, edges, shapes, objects, and patterns. These features enable a non-destructive method of fruit classification and gradation. Techniques such as Histogram of Oriented Gradients (HOG), color analysis, and mean value calculations are employed to enhance the accuracy of FQD systems.

The integration of FQD into automated systems offers significant improvements in the speed and accuracy of fruit quality assessment and sorting. This automation can save time and labor compared to traditional manual inspections, which are often slow and prone to errors. Implementing FQD technologies can streamline packaging, transportation, and marketing operations, thereby enhancing the efficiency of the entire supply chain from farm to table.

This study utilizes a dataset consisting of 12 classes covering six fruits: apple, banana, guava, orange, lime, and pomegranate. Each fruit is categorized into bad and good classes. The project involves extracting features from the images and employing classification models to categorize the images. The primary application of FQD lies in the food industry, where it is used to grade fruits based on a quality index.

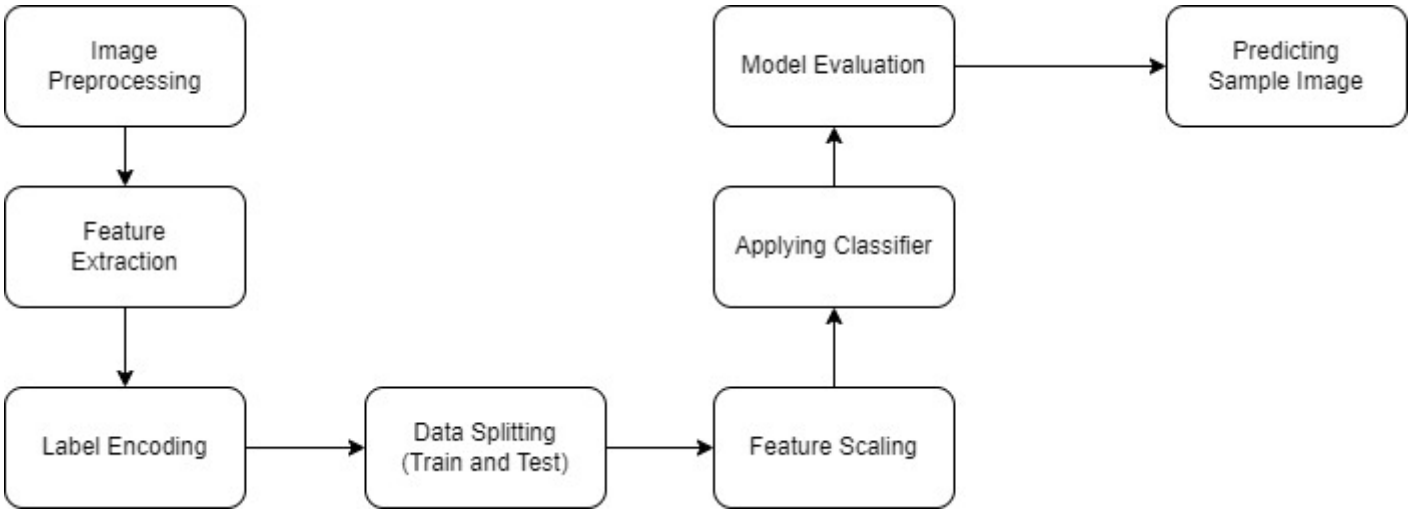
The research compares the performance of various machine learning models, including K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Random Forest, Logistic Regression, Naive Bayes, Artificial Neural Networks (ANN), Convolutional Neural Networks (CNN), Graph Neural Network (GNN), Linear Regression, and YOLOv8. These models are evaluated based on their accuracy and precision in detecting fruit quality. By comparing these models, the study aims to identify the most effective approach for FQD.

Additionally, the study explores the potential for creating a grading system that segregates fruits based on quality ranges using YOLOv8. This grading system would classify fruits into categories such as 80-95% as best, 65-75% as good, 40-60% as okay, and 10-35% as bad. Such a system would further refine the process of fruit quality assessment, providing a more detailed and nuanced understanding of fruit quality and enabling better decision-making in the food industry.

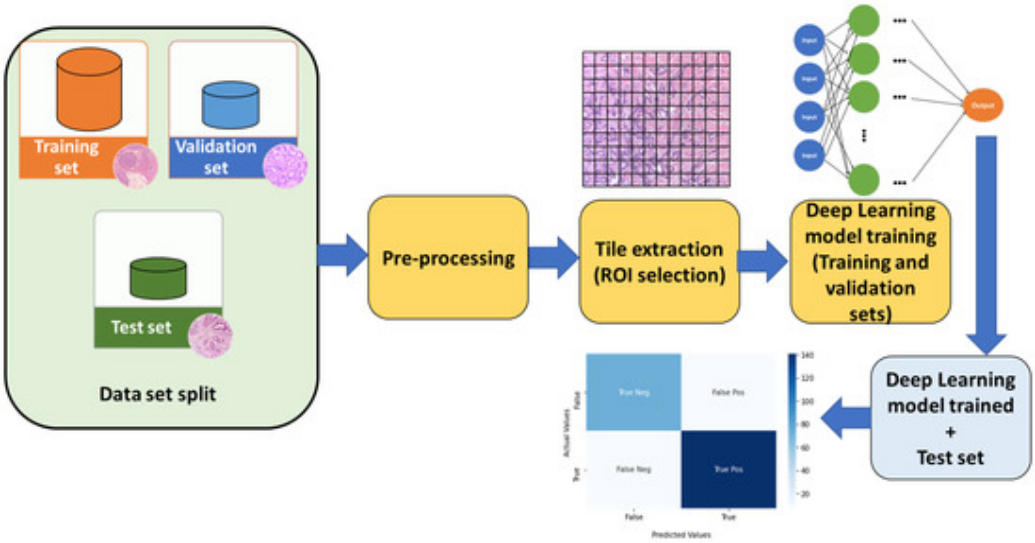
2. Research Methodology:

The Fruit Quality Detection (FQD) system is designed to identify and classify the quality of fruits based on input images provided by the user.his process encompasses several key stages, each crucial for ensuring the accuracy and efficiency of the system which includes Image Preprocessing, Feature Extraction, Label Encoding, Data Splitting, Feature Scaling, Applying Classifier, Model Evaluation, and Predicting Sample Images.

KNN, SVM, Naive Bayes, Logistic Regression, and Linear Regression: These models follow the standard workflow involving image preprocessing, feature extraction, label encoding, data splitting, feature scaling, and classifier application. Each model has specific parameters and hyperparameters that are tuned to optimize performance.



CNN and ANN: These deep learning models also use the standard workflow but benefit from advanced techniques that automatically extract and learn features from raw images during the training process. CNNs, in particular, are adept at recognizing patterns in images, making them suitable for complex image classification tasks.



YOLOv8: For YOLOv8, the process begins with annotating images by bounding-boxing objects and assigning them respective classes, facilitated by platforms like Roboflow which streamline the labeling process. Roboflow provides comprehensive tools for data labeling, model training, deployment, and evaluation, enhancing the efficiency and accuracy of YOLOv8 training. Annotated images then undergo preprocessing and augmentation, such as rotation, scaling, and flipping, to diversify the training data and

improve model robustness. Finally, the YOLOv8 model is trained on these images to accurately detect and classify fruits, with the training process optimizing model parameters to minimize loss and ensure precise object detection and classification.

2.1 Pre-processing Steps:

Preprocessing is a crucial step in image processing and computer vision tasks, as it enhances the quality of the input image to facilitate better analysis and feature extraction.

Image Resizing:

Image resizing involves changing the dimensions of an image to a standard size. This is crucial for ensuring consistency in the dataset and reducing computational load. In machine learning, standardizing the input image size helps maintain uniformity, which is necessary for the efficient training of models and for reducing variability that could affect the performance of the algorithm.

Convert Image to Grayscale:

Converting an image to grayscale simplifies the image data by reducing it to one color channel. This is particularly useful in applications where color information is not critical. Grayscale conversion reduces the complexity of the data and helps focus on the structural and textural features of the image, which are essential for tasks like edge detection and text recognition.

Image Loading and Validation:

Loading an image correctly ensures that it is read into the system in the proper format. Validation checks if the image meets the necessary criteria for further processing. This step is crucial for preventing errors that can arise from corrupted or incompatible image files. Proper loading and validation ensure that only valid images are processed, thereby maintaining the integrity of the dataset.

Label Encoding:

Label encoding converts categorical labels into numerical values, which are required for training machine learning models. This step transforms qualitative data into a quantitative format that algorithms can process. It is a fundamental step in preparing the dataset for classification tasks, ensuring that the labels are in a format suitable for model training.

Train-Test Split:

The train-test split is a technique to divide the dataset into two parts: one for training the model and the other for testing its performance. This split is essential for evaluating how

well the model generalizes to unseen data. By testing on a separate set, we can get an unbiased estimate of the model's accuracy and other performance metrics.

Feature Scaling:
Feature scaling standardizes the range of independent variables or features. In machine learning, this is crucial because many algorithms perform better when numerical attributes are on a similar scale. Scaling helps accelerate the convergence of gradient descent algorithms and improves the model's performance by ensuring that features contribute equally to the result.

2.2 Feature Extraction:

Feature extraction is a fundamental concept in image processing, used to obtain details and patterns of objects within images. It plays a crucial role in pattern recognition and serves as a method of dimensionality reduction. The main objective of feature extraction is to capture the most relevant information from the original data and represent it in a lower-dimensional space.

This process is typically performed after preprocessing, as it enhances the accuracy of identifying important features. In pattern recognition, the primary task is to analyze an input pattern and correctly classify it into one of the predefined categories.

Importance of Feature Extraction:
Feature extraction is an important step in the construction of any pattern classification and aims at the extraction of the relevant information that characterizes each class.

1. Improves efficiency in Classification by reducing data dimensionality, facilitating easier differentiation between classes.
2. It focuses on extracting the most relevant information from raw data, minimizing within-class variability and maximizing between-class variability, thus ensuring robust and accurate classification outcomes.
3. Reduces the Redundancy present in the data by transforming input data into a concise set of features with the help of Feature Selection which eliminates redundancy and retains only essential information.
4. Improvement of Recognition Rates by choosing the right feature extraction technique is crucial for high recognition performance. Effective methods ensure the extracted features are both discriminative and representative, significantly boosting recognition rates.

Feature Selection:

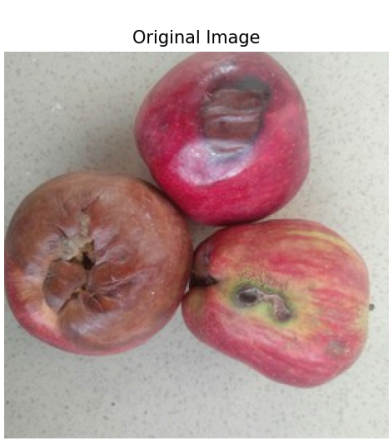
Feature selection is a crucial process in handling high-dimensional data, which often contains irrelevant, misleading, or redundant features. These extraneous features increase the search space size, complicating data processing and hindering the learning process. Feature Selection Algorithms (FSAs) are employed to address this issue by identifying and retaining only the most significant features.

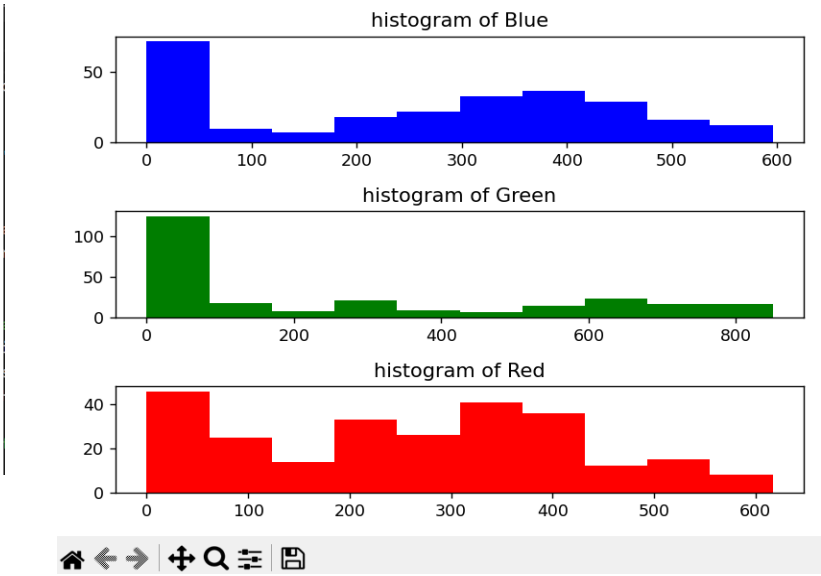
The importance of feature selection includes:

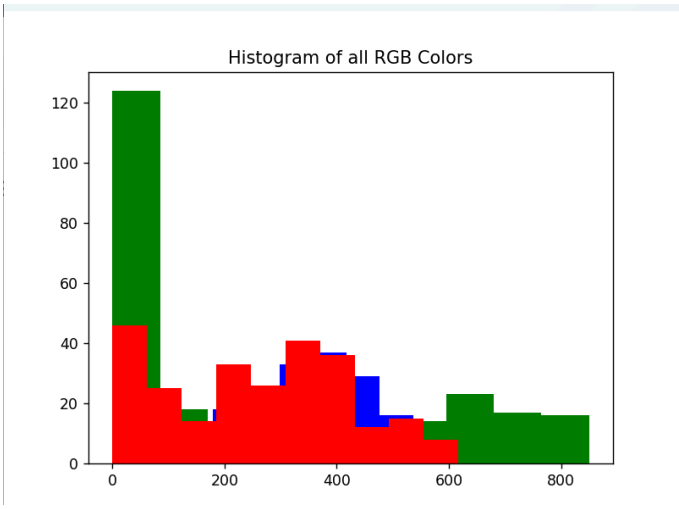
1. It performs dimensional reduction by limiting the storage space needed for data, it simplifies data management and analysis.
2. It enhances model accuracy by focusing on the most relevant features, allowing the model to learn more effectively.
3. Enhanced Data Quality by eliminating noise and redundant information, feature selection improves the overall quality of the dataset.
4. It reduces ambiguity and simplifies the learning process, making the data easier to work with and accelerating computational processes.

Color Histogram

A color histogram represents the distribution of colors within an image. It counts the number of pixels for each color value and is often used to describe the image's color composition. This technique is especially useful in computer vision tasks for object recognition and image retrieval. Histogram equalization can be applied to grayscale images to improve contrast, especially when the pixel distribution is uneven. For color images, the histogram is normalized and flattened into a feature vector, which helps in comparing and classifying images based on their color content.

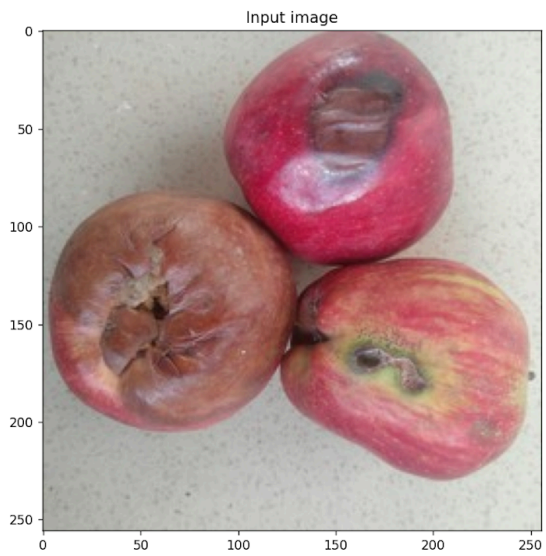


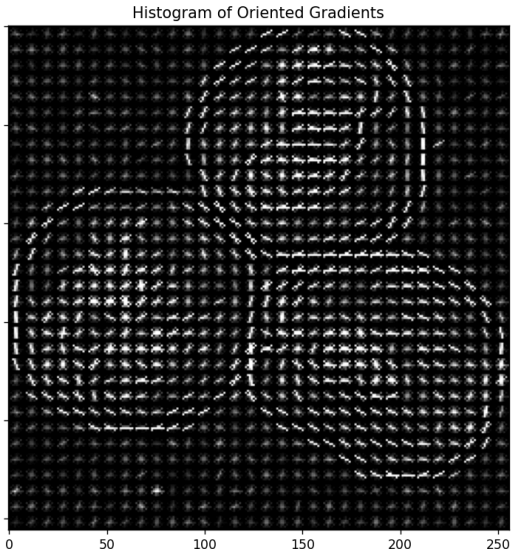




Histogram of Oriented Gradients (HOG)

HOG is a feature descriptor used for object detection. It works by dividing an image into small regions called cells and computing a histogram of gradient directions or edge orientations within each cell. The combined histograms represent the image's overall structure and are used to identify and localize objects. HOG is particularly effective for detecting objects with well-defined edges and has been widely applied in pedestrian detection and other computer vision applications.





Mean Color

Mean color is a straightforward yet powerful feature that calculates the average color values of an image. It involves computing the mean of the RGB (Red, Green, Blue) channels, providing a simple representation of the image's color. This feature is beneficial for distinguishing images based on their dominant color characteristics and is often used in image retrieval systems.

```
Shape of Image Features

(34596,)

Mean Color of the Image

[0.69869271 0.5632867  0.55680291]
```

Combining Features

Combining multiple features can significantly enhance the performance of image classification systems. By integrating color histograms, HOG, mean color, and other features, a more comprehensive representation of the image is achieved. This multi-faceted approach captures various aspects of the image, improving the accuracy and robustness of classification algorithms. Feature combination is particularly useful in complex tasks such as object recognition, where different features contribute to a more holistic understanding of the image.

```
Shape of Combined Features

(35111,)
```

Models:

K-Nearest Neighbors (KNN):

K-Nearest Neighbors (KNN) is a fundamental classification and prediction model used in machine learning. It works by identifying the K nearest neighbors of a data point based on the Euclidean distance and classifying the data point based on the majority class among its neighbors. The model's simplicity and effectiveness make it a popular choice for various applications.

One of the critical challenges in using the KNN algorithm is selecting the optimal value of K. The value of K determines the number of neighbors to consider for classifying a new data point. Choosing an incorrect value of K can lead to poor classification performance. For instance, a small value of K (e.g., K=1) can lead to high variance and sensitivity to noise, while a large value of K can result in bias, making the model overly smooth and less sensitive to the underlying data structure.

Small K Value: A small K value, such as K=1, means that the model relies on the closest neighbor for classification. This can cause high variance and sensitivity to noisy data, as the classification heavily depends on individual data points.

Large K Value: A large K value averages the classifications over a broader range of neighbors, which can introduce bias by smoothing the predictions. This can result in the model overlooking local patterns in the data.

We employed K-Nearest Neighbors (KNN) to classify and identify fruit quality, distinguishing between good and bad based on metrics such as color, texture, and size. KNN determines the category of an unknown data point by analyzing the values of its nearest neighbors. Effective feature extraction and selection are crucial, as they significantly enhance recognition and classification rates, leading to more accurate and reliable fruit quality assessments.

Classification Report:

K = 5

```
Train accuracy: 93.19%
Test accuracy: 88.17%
Precision: 87.86%
Recall: 88.17%
F1 Score: 87.48%
```

K = 10

```
Train accuracy: 89.03%
Test accuracy: 85.38%
Precision: 85.27%
Recall: 85.38%
F1 Score: 84.67%
```

Confusion matrix for KNN:

K = 5

```
array([[192,   0,   0,   0,   0,   0,   1,   0,   0,   2,   0,   6],
       [ 11,  83,   2,   2,  12,  17,  19,   3,  12,  12,  14,  14],
       [  0,   1, 198,   0,   2,   1,   4,   0,   4,   0,   0,   0],
       [  0,   1,   2, 201,   1,   0,   1,   0,   8,   1,   1,   0],
       [  3,   0,   2,   0, 185,   3,   2,   0,   0,   0,   2,   2],
       [  1,  16,   0,   0,   2, 165,   8,   4,   0,   6,   0,   7],
       [  0,   3,   4,   0,   6,   5, 165,   0,   2,   0,   1,   0],
       [  0,   0,   0,   0,   0,   3,   0, 192,   0,   0,   0,   0],
       [  0,   0,   1,   0,   3,   0,   5,   0, 173,   4,   0,   1],
       [  1,   1,   0,   1,   1,   2,   0,   0,   3, 171,   0,   7],
       [  2,   0,   1,   0,   1,   0,   0,   0,   0,   0, 220,   1],
       [  2,   4,   0,   0,   0,   0,   0,   0,   1,   4,   2, 171]],
      dtype=int64)
```

K = 10

```
array([[185,   0,   0,   0,   1,   0,   1,   0,   1,   2,   3,   8],
       [ 12,  78,   3,   2,  10,  14,  22,   4,  11,  14,  16,  15],
       [  0,   2, 186,   0,   8,   0,   8,   0,   5,   0,   0,   1],
       [  0,   0,   2, 196,   2,   0,   1,   0,  13,   1,   1,   0],
       [  2,   0,   4,   0, 184,   3,   2,   0,   0,   0,   2,   2],
       [  1,  18,   0,   0,   3, 160,   6,   6,   0,   8,   0,   7],
       [  0,   2,   4,   1,   8,   6, 159,   0,   3,   0,   1,   2],
       [  0,   0,   0,   0,   3,   4,   0, 188,   0,   0,   0,   0],
       [  0,   0,   3,   0,   2,   0,   4,   0, 173,   4,   0,   1],
       [  0,   2,   0,   4,   2,   2,   1,   1,   5, 163,   0,   7],
       [  9,   0,   1,   0,   4,   0,   1,   0,   0,   0, 206,   4],
       [  2,   1,   1,   0,   0,   0,   0,   0,   1,   4,   4, 171]],
      dtype=int64)
```

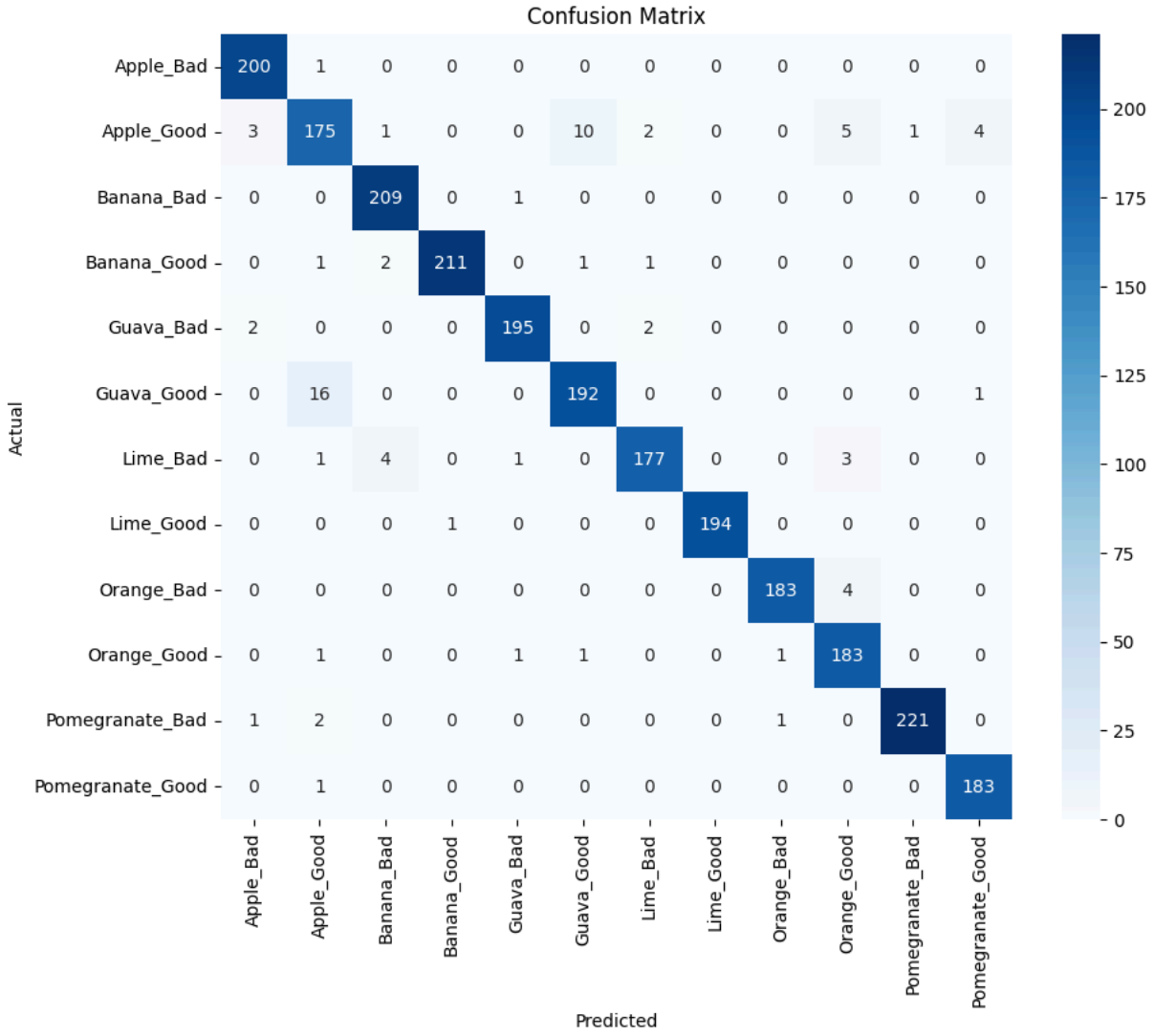
SVM:

Support Vector Machine (SVM) is a supervised machine learning algorithm commonly used for classification tasks. SVM works by plotting each data item as a point in n-dimensional space, where n is the number of features. The objective of SVM is to find a hyperplane that best separates the data points of different classes. In a two-dimensional space, this hyperplane is a line, but in higher dimensions, it can be a plane or a hyperplane.

The Hyperplane is the decision boundary that separates different classes. SVM aims to maximize the margin between the data points of different classes. Support Vectors are data points that are closest to the hyperplane and influence its position and orientation. These points are critical in defining the hyperplane. SVM uses kernel functions to transform the input data into a higher-dimensional space to make it easier to classify non-linear data. Linear, Polynomial, and Radial Basis Function (RBF) Kernel are commonly used kernel functions.

An SVM with a linear kernel is trained on the scaled features to identify the optimal hyper-plane that separates different classes of fruit quality. GridSearchCV is used to find the best hyperparameters for the SVM model. The SVM algorithm is beneficial in this application due to its effectiveness in high-dimensional spaces and its ability to model non-linear relationships using kernel functions. This makes it particularly suitable for image-based classification tasks where the data may be complex and multidimensional. The use of SVM ensures a robust and accurate classification of fruit quality, aiding in automated and consistent quality control processes.

Confusion matrix for SVM:



<u>Random Forest:</u>

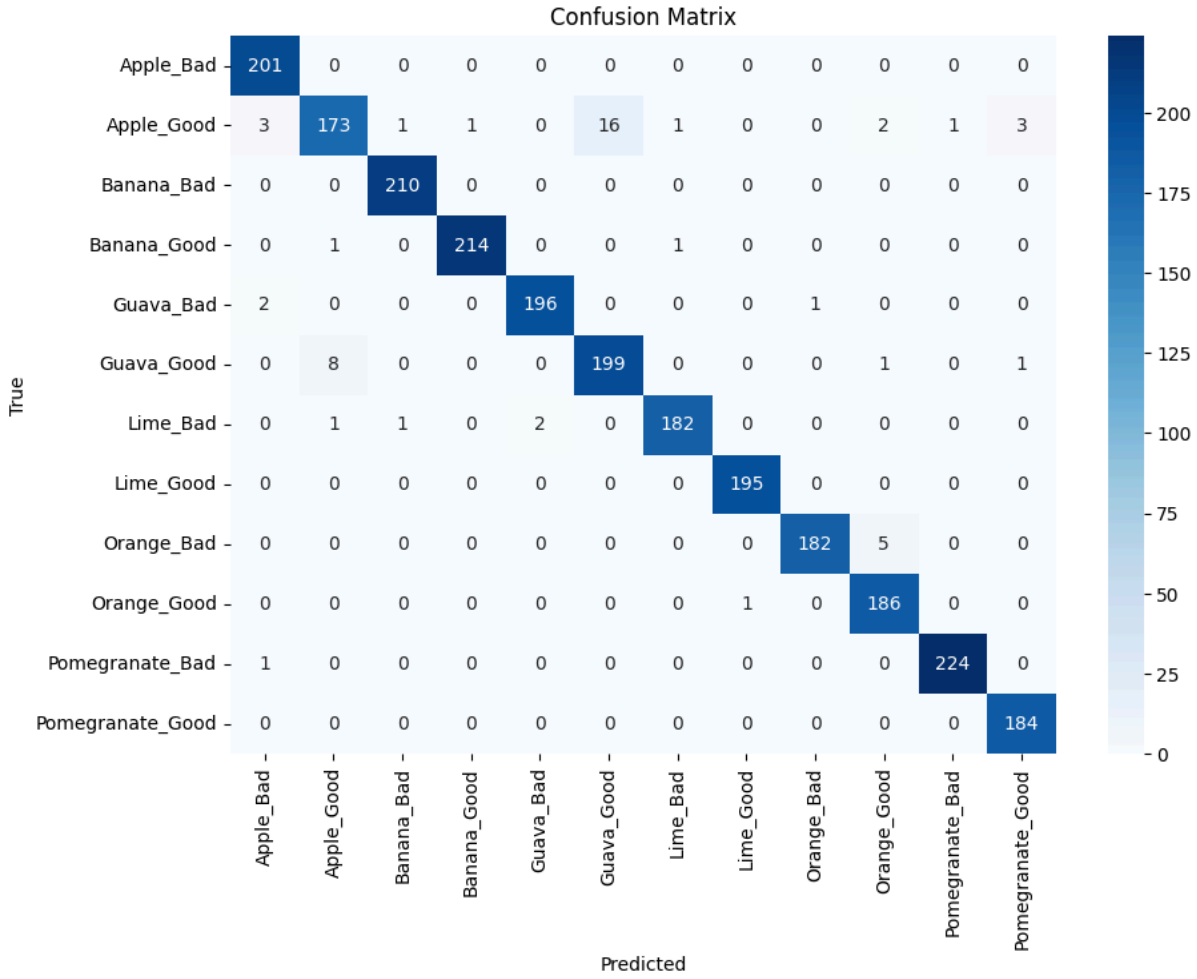
Random Forest is a machine learning algorithm that leverages multiple decision trees during its training phase. Each tree is constructed using a random subset of the dataset and measures a random subset of features at each partition. This randomness introduces variability among individual trees, which reduces the risk of overfitting and enhances the overall prediction performance. The algorithm aggregates the results of all trees by voting in classification tasks and averaging in regression tasks. Because multiple decision trees are involved, the results obtained from Random Forest are both stable and precise. Random Forest is particularly effective in handling complex data, reducing overfitting, and providing reliable insights for decision-making.

Before applying Random Forest to image classification, relevant features like color histograms, texture descriptors, and shape descriptors need to be extracted and transformed into numerical representations. During training, the Random Forest algorithm constructs multiple decision trees using these features from training images, each tree trained on different bootstrap samples and feature subsets to ensure variability.

For classifying new images, the extracted features are passed through each tree in the forest, with each tree providing a classification vote. The final class label is determined by majority voting. This ensemble approach enhances accuracy and robustness.

Random Forest is beneficial for image classification due to its ability to handle high-dimensional data and its robustness against overfitting, capturing complex patterns and interactions for reliable and precise classification outcomes.

Confusion matrix for Random Forest :



Artificial Neural Networks (ANN)

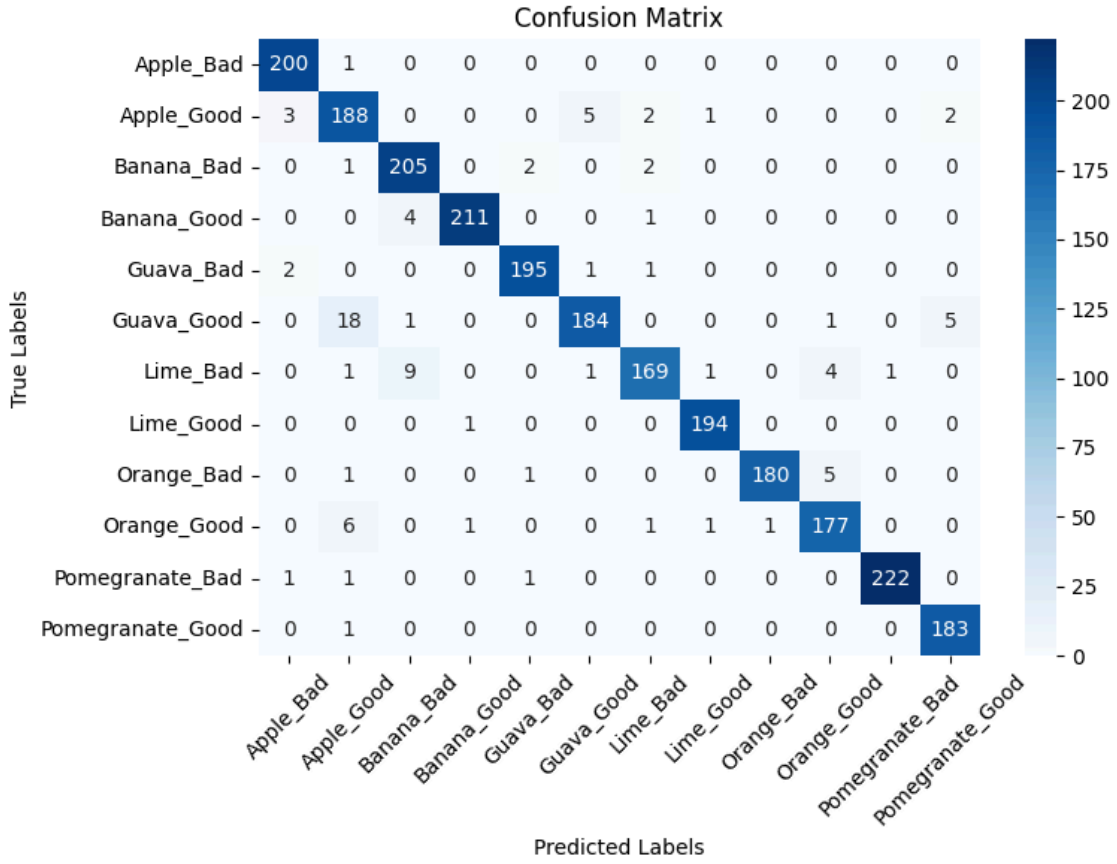
Artificial Neural Networks (ANN) are computational models inspired by the human brain's neural structure. They consist of interconnected nodes (artificial neurons) organized into layers, where data or information moves through these nodes. During training, the network adjusts the connection weights to learn from the data, enabling it to recognize patterns, make predictions, and solve various tasks. ANNs can range from single-layer to multi-layer architectures, typically comprising an input layer, one or more hidden layers, and an output layer. The hidden layers extract relevant patterns from the inputs and pass them on for further analysis, improving network efficiency by focusing on essential information.
A sequential neural network model was constructed using Keras for model building and training. The model architecture included:

- An input layer matches the dimension of the feature vector.
- Two dense hidden layers with ReLU activation functions and dropout layers for regularization.
- An output layer with a softmax activation function for multi-class classification.

The model was compiled using the Adam optimizer and sparse categorical cross-entropy loss. The training was conducted over 20 epochs.

Artificial Neural Networks play a crucial role in fruit quality detection due to their ability to model complex patterns and handle high-dimensional data. The combination of traditional feature extraction techniques and ANNs allows for efficient and accurate classification of fruit quality. The use of color histograms, HOG features, and mean color ensures that both color and texture information are captured, which are critical factors in determining fruit quality.

Confusion matrix for ANN:



<u>CNN:</u>

Convolutional Neural Networks (CNNs) are a type of deep neural network that excels in image classification, object detection, and image recognition tasks. Inspired by the human visual system, CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images. This makes them particularly effective for processing structured grid data such as images.

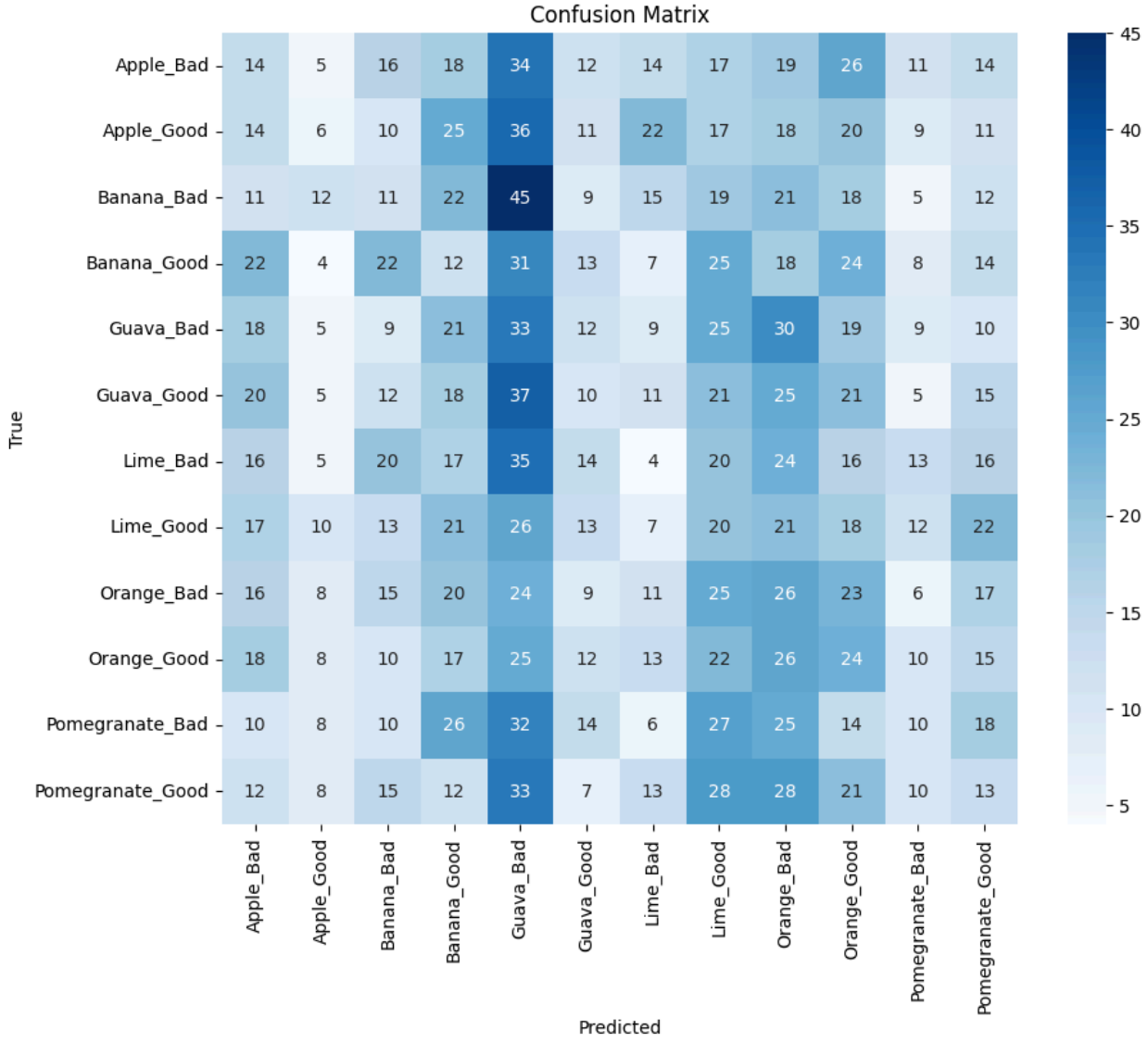
Some basic Terminologies and Concepts used in CNN:

- **Convolutional Layer:** Applies filters (kernels) to the input image to produce feature maps, detecting features like edges and textures.
- **Filter (Kernel):** Small weight matrices that act as feature detectors, identifying characteristics such as vertical or horizontal edges.
- **Activation Function:** Non-linear function, like ReLU, applied to convolution outputs to introduce non-linearity and enable learning of complex patterns.
- **Pooling Layer:** Reduces spatial dimensions of feature maps, decreasing computational complexity and reducing overfitting. Commonly uses Max Pooling.

- **Fully Connected Layer:** Neurons are connected to every neuron in the previous layer, combining features for final classification.
- **Dropout:** Regularization technique that randomly sets a fraction of input units to zero during training, preventing overfitting and improving generalization.
- **Batch Normalization:** Normalizes inputs of each layer to have zero mean and unit variance, accelerating training and stabilizing the learning process.

We used `ImageDataGenerator` from TensorFlow Keras for data augmentation, including rescaling, rotation, shifts, shear, zoom, and horizontal flip, to enhance model robustness and generalization.
The model architecture employs transfer learning with the pre-trained MobileNetV2, retaining its weights and adding custom layers: global average pooling, a dense layer with 512 units and ReLU activation, dropout for regularization, and a final softmax layer for classification.
The model was compiled with the Adam optimizer (learning rate 1e-4) and categorical cross-entropy loss, tracking accuracy as the evaluation metric.
Class weights were computed using `compute_class_weight` from sci-kit-learn to address class imbalance.
The training was optimized using `ReduceLROnPlateau` to adjust the learning rate dynamically and `EarlyStopping` to prevent overfitting by halting training when validation loss did not improve.

Confusion matrix for CNN:



Logistic Regression:

Logistic Regression is a statistical method used for binary and multiclass classification problems. It models the probability of a categorical outcome based on one or more predictor variables. Unlike linear regression, which predicts a continuous output, logistic regression predicts a probability that can be mapped to a categorical outcome.

Logistic regression uses the sigmoid function to map the predicted values to probabilities:

$$\sigma(z) = \frac{1}{1+e^{z}}$$

where $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots\ldots + \beta_n x_n$

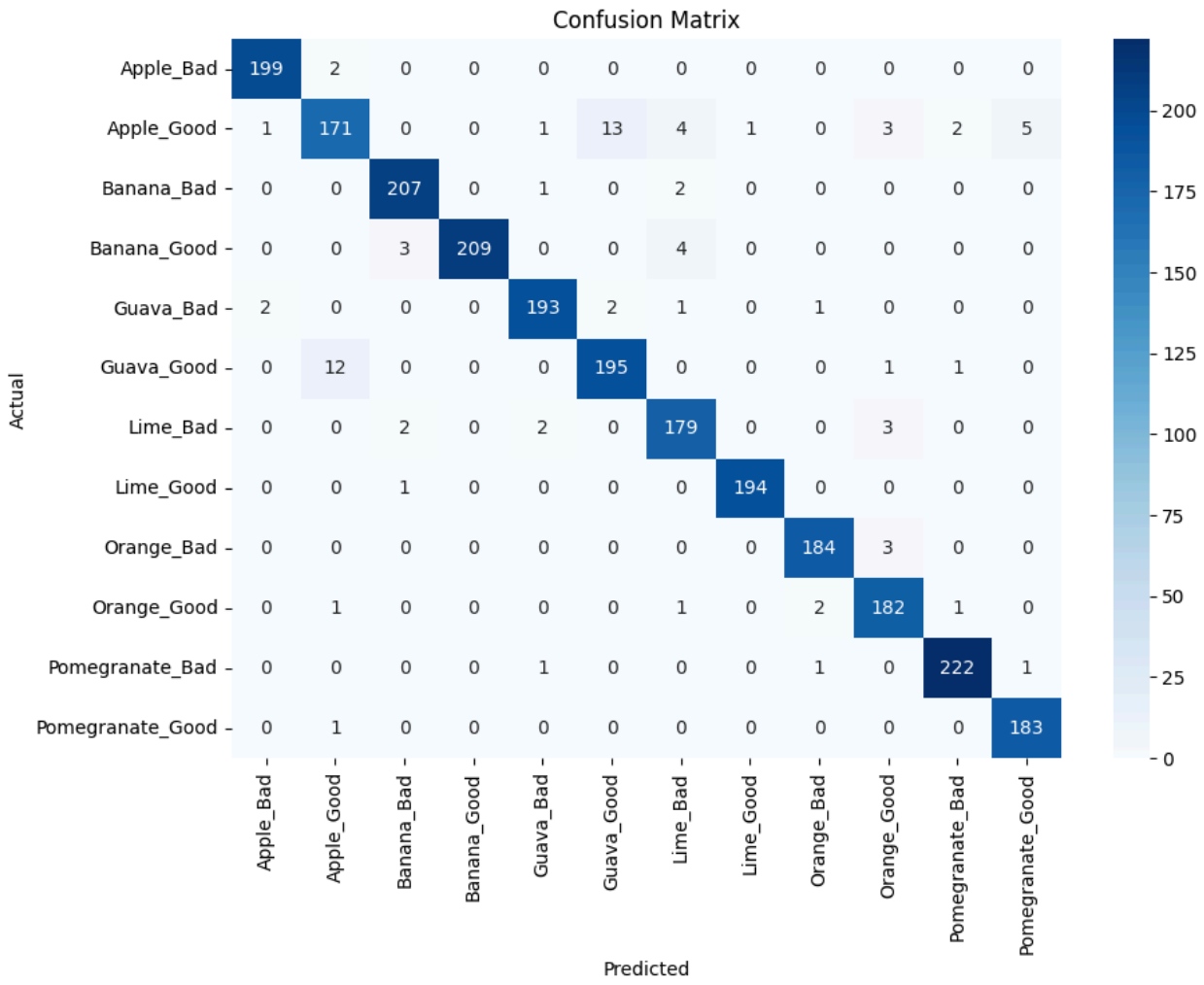
The decision boundary is determined by a threshold, usually 0.5. If the predicted probability is greater than or equal to the threshold, the output is classified as one class; otherwise, it is classified as the other class.

The cost function used in logistic regression is the log loss, also known as cross-entropy loss:

$$J(\theta) = -\frac{1}{m} \sum_{i=-1}^{m} [y_i \, log(\, h_\theta (x_i)) + (1 - y_i) log(1 - h_\theta (x_i)) \,]$$

where m is the number of training examples, y_i is the actual label, and $h_\theta (x_i)$ is the predicted probability.

The logistic regression model is initialized. In sci-kit-learn, this is done using the LogisticRegression class. The Model Parameter max_iter is set to "1000". This parameter specifies the maximum number of iterations the algorithm will run to find the optimal parameters. The use of max_iter=1000 ensures sufficient iterations for convergence, resulting in a reliable model for fruit quality detection.


Confusion Matrix
Actual
Predicted
Apple_Bad
Apple_Good
Banana_Bad
Banana_Good
Guava_Bad
Guava_Good
Lime_Bad
Lime_Good
Orange_Bad
Orange_Good
Pomegranate_Bad
Pomegranate_Good
199 2 0 0 0 0 0 0 0 0 0 0
1 171 0 0 1 13 4 1 0 3 2 5
0 0 207 0 1 0 2 0 0 0 0 0
0 0 3 209 0 0 4 0 0 0 0 0
2 0 0 0 193 2 1 0 1 0 0 0
0 12 0 0 0 195 0 0 0 1 1 0
0 0 2 0 2 0 179 0 0 3 0 0
0 0 1 0 0 0 0 194 0 0 0 0
0 0 0 0 0 0 0 0 184 3 0 0
0 1 0 0 0 0 1 0 2 182 1 0
0 0 0 0 1 0 0 0 1 0 222 1
0 1 0 0 0 0 0 0 0 0 0 183
200
175
150
125
100
75
50
25
0

Naive Bayers:

Naive Bayes classifiers are a type of supervised machine learning algorithm based on Bayes' theorem of probability. They are designed to predict the class of unknown data sets by modeling the distribution of inputs within specific classes. Naive Bayes classifiers, known for their simplicity and effectiveness, offer a probabilistic approach to classification based on Bayes' theorem.
Key characteristics of Naive Bayes classifiers include:

1. Bayes' Theorem: This theorem calculates the probability of a hypothesis given the observed data and prior knowledge. It is expressed as:

$$P(H \mid D) = \frac{P(D \mid H).\, P(H)}{P(D)}$$

   where P(H | D) is the probability of hypothesis H given data D, P(D | H) is the probability of data D given hypothesis H, P(H) is the prior probability of hypothesis H, and P(D) is the prior probability of data D.

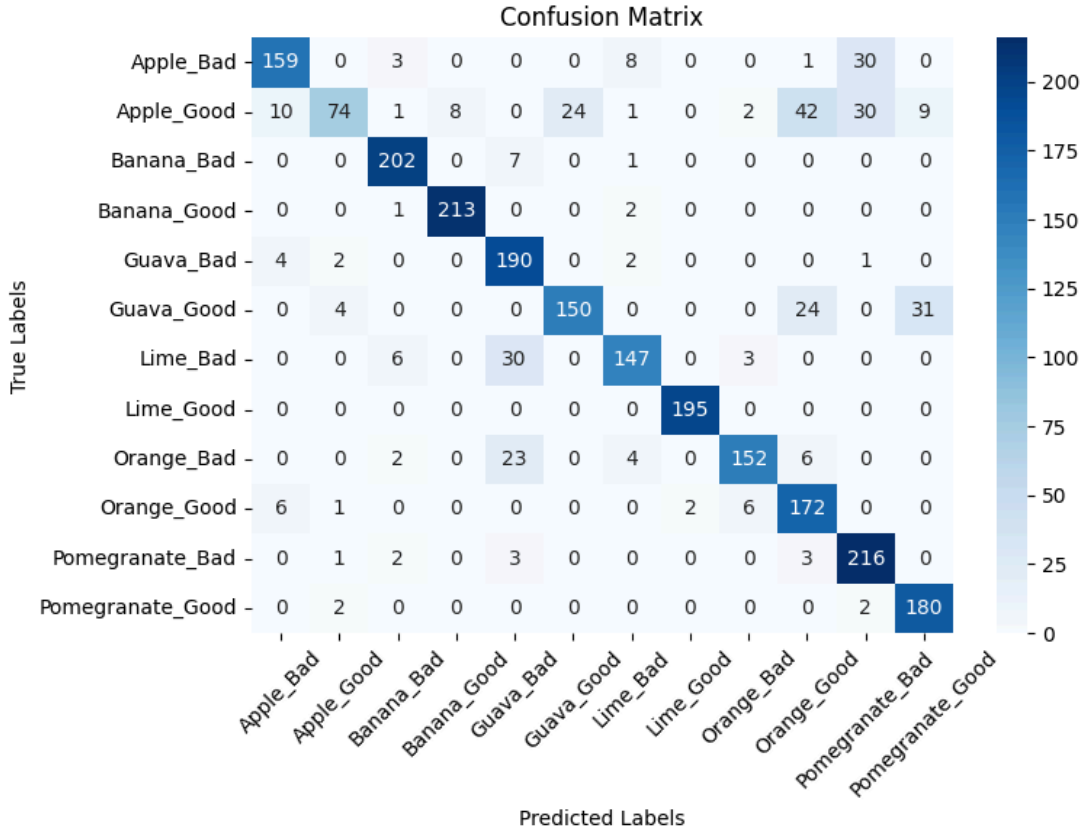
2. Feature Independence: Naive Bayes assumes that all features in the input data are independent of each other, which simplifies the computation and allows for efficient training and prediction.

3. Kernel Density Estimation: This technique can be used to estimate the probability density function of input data, enhancing the classifier's performance in scenarios where data distribution is complex and not well-defined.

The Naive Bayes classifier is trained on the extracted features. The dataset is split into training (80%) and testing (20%) sets. The features are scaled using StandardScaler to normalize them before training.

Naive Bayes is Suitable for Fruit Quality Detection because of its computational efficiency making it suitable for real-time applications. Its training and prediction times are fast, which is essential when processing a large number of images.
Naive Bayes performs well with high-dimensional data. The combination of color histograms, HOG features, and mean color creates a rich feature set, and Naive Bayes can effectively handle this multi-dimensional input. Naive Bayes provides probabilistic outputs, which can be useful for decision-making processes. Naive Bayes can utilize kernel density estimation to handle complex data distributions. This is beneficial when dealing with natural variations in fruit appearance due to different lighting conditions, angles, or inherent quality differences.

Confusion matrix for Naive Bayes :

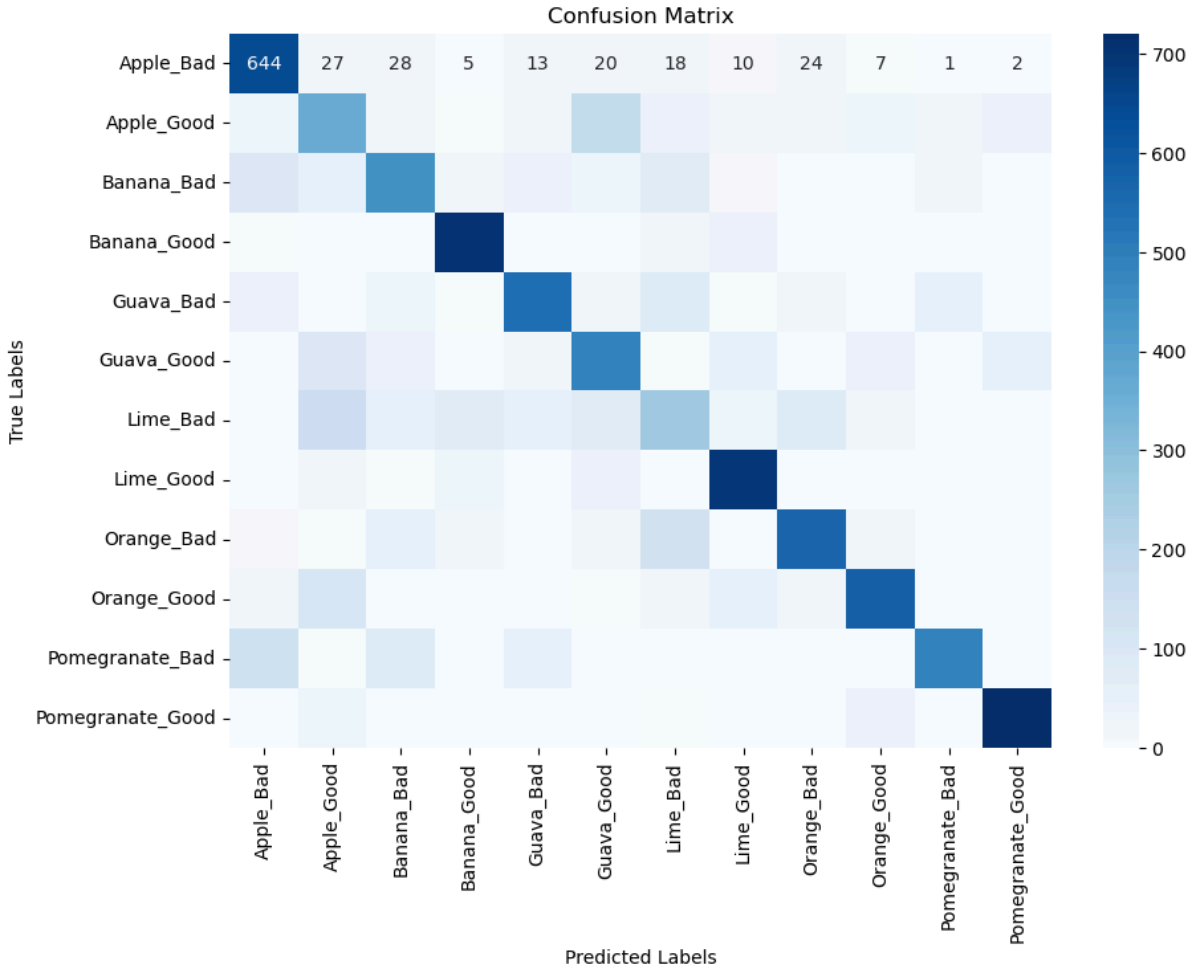


GNN:

The concept of the earliest graph neural network (GNN) was introduced in several foundational papers. Micheli proposed an early version of a spatial-based graph convolutional network (GCN) using non-recursive layers. Over recent years, numerous variants of spatial-based GCNs have emerged. Spectral-based GCNs were first developed by Bruna et al., incorporating graph convolution based on spectral graph theory. Since then, many efforts have been made to enhance spectral-based GCNs. GCNs are typically utilized in graph data contexts such as social networks, citation networks, and biochemical graphs. In computer vision, GCN applications include point cloud classification, scene graph generation, and action recognition. Point clouds, which are 3D point sets collected by LiDAR, have been classified and segmented using GCNs. Scene graph generation, which parses images into graphs of objects and their relationships, often combines object detectors with GCNs. For human action recognition, GCNs process graphs of connected human joints. While GCNs are effective for visual tasks with naturally constructed graphs, a GCN-based backbone network is needed to handle general computer vision applications that directly process image data.

.

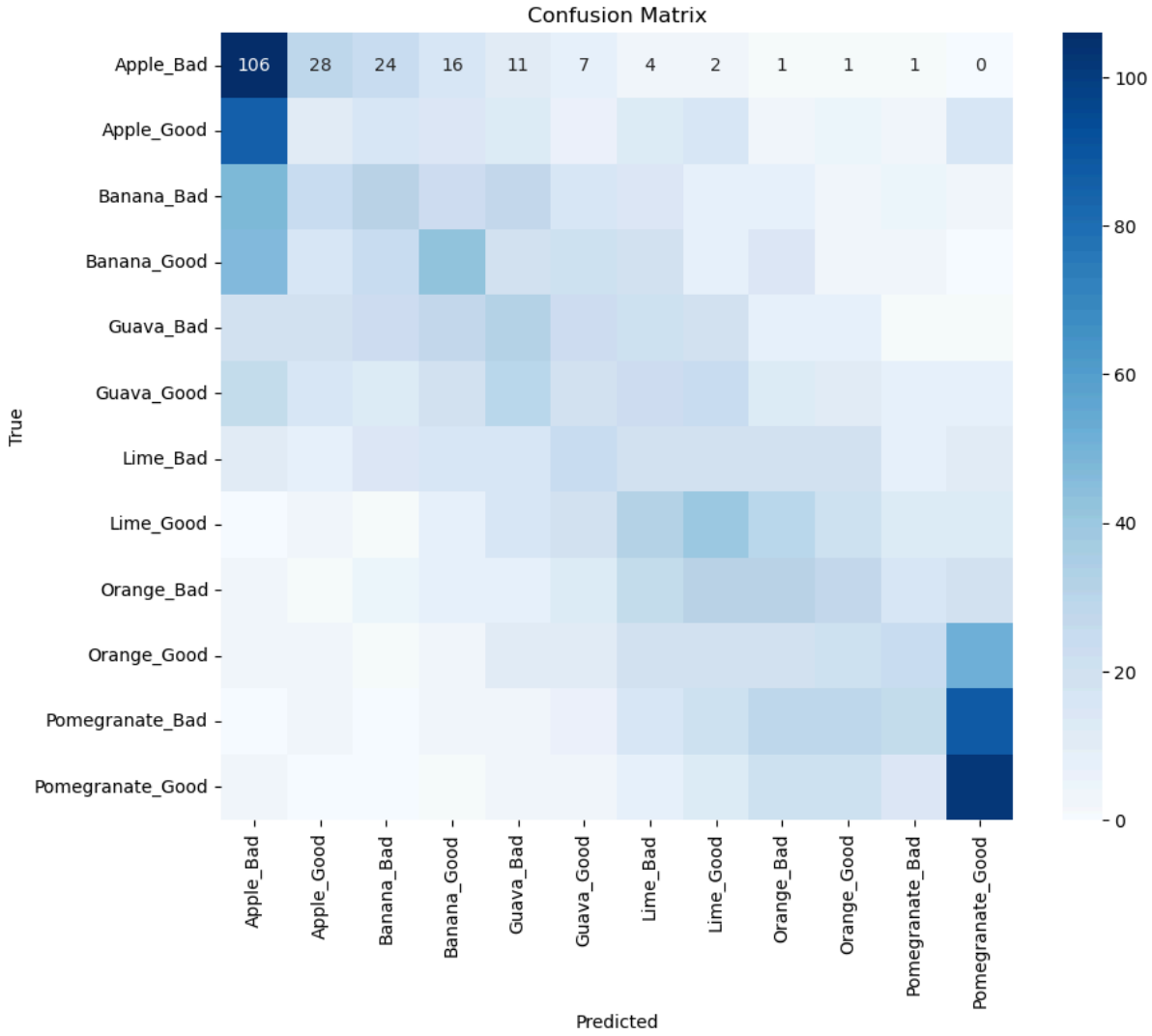
Confusion matrix for GNN:



Linear Regression:

Linear regression analysis is employed to forecast the value of one variable based on the value of another. The variable being predicted is known as the dependent variable, while the variable used for prediction is called the independent variable. This analytical method calculates the coefficients of the linear equation, involving one or more independent variables, to best predict the dependent variable's value. Linear regression aims to fit a line or surface that reduces the differences between predicted and actual values. Simple linear regression calculators often use the "least squares" method to find the optimal line for paired data, allowing the estimation of the dependent variable (X) from the independent variable (Y).

Confusion matrix for Linear regression:



Quality Evaluation Techniques:

(Model Evaluation Metrics)

Evaluating a machine learning model is crucial for verifying its accuracy and effectiveness. Ensuring that models deployed in production are both optimal and reliable is a key step in the development process.

1. **Accuracy**: This metric evaluates the overall correctness of the model by calculating the ratio of correct predictions to total predictions. While it is straightforward, accuracy can be misleading for imbalanced datasets where one class is more prevalent than others.
2. **Precision**: Precision measures the proportion of true positive predictions among all positive predictions. It is particularly important in scenarios where the cost of false positives is high, ensuring that the model's positive predictions are reliable.
3. **Recall**: Also known as sensitivity, recall calculates the ratio of true positives to the sum of true positives and false negatives. It is crucial in situations where missing a positive instance can have significant consequences, ensuring that most positive cases are identified.
4. **F1 score**: The F1 score is the harmonic mean of precision and recall, providing a balance between the two. It is especially useful when both false positives and false negatives need to be considered, offering a comprehensive measure of the model's performance.

Model	Train Accuracy	Test Accuracy	Precision	F1 Score	Recall
ANN	99.94%	96.42%	96%	96%	96%
Naive Bayes	87.63%	85.42%	86.52%	84.57%	87.63%
KNN	93.19%	88.17%	87.86%	87.48%	88.17%
Logistic Regression	100%	96.58%	96.57%	96.56%	96.58%
Random Forest	100%	97.75%	98%	98%	98%
SVM	100%	96.79%	96.67%	96.79%	95.83%
CNN	90.97%	94.60%	91.26%	91.80%	92.35%
GNN	61.12%	67.65%	64.09%	61.36%	61.12%
Linear Regression	20.12%	87.67%	20.47%	21.10%	21.78%

5. **Execution Time and Time Complexity:**

In our study on Fruit Quality Detection (FQD), the execution time and time complexity of each model are critical factors, particularly given our dataset of 12,000 images. Time complexity provides a theoretical estimate of the computational effort required, while execution time gives practical insights based on actual implementation.

K-Nearest Neighbors (KNN) has a time complexity of O(n * d), where n is the number of images and d is the number of dimensions, resulting in relatively slow execution with large datasets. Support Vector Machines (SVM) also exhibit considerable training time with a complexity of O(n^2 * d) but are faster in prediction. Random Forest models, with a time complexity of O(nt * n log n) where nt is the number of trees, offer a balanced trade-off between training time and accuracy. Logistic Regression and Naive Bayes, with complexities of O(n * d^2) and O(n * d) respectively, are efficient and fast, making them suitable for real-time applications. Advanced models like Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN), despite their high computational requirements during training, provide faster predictions once trained, with complexities of O(n * d * h) and O(n * d * f). YOLOv8, with its end-to-end training complexity of O(n * d * k), stands out for its real-time prediction capabilities.

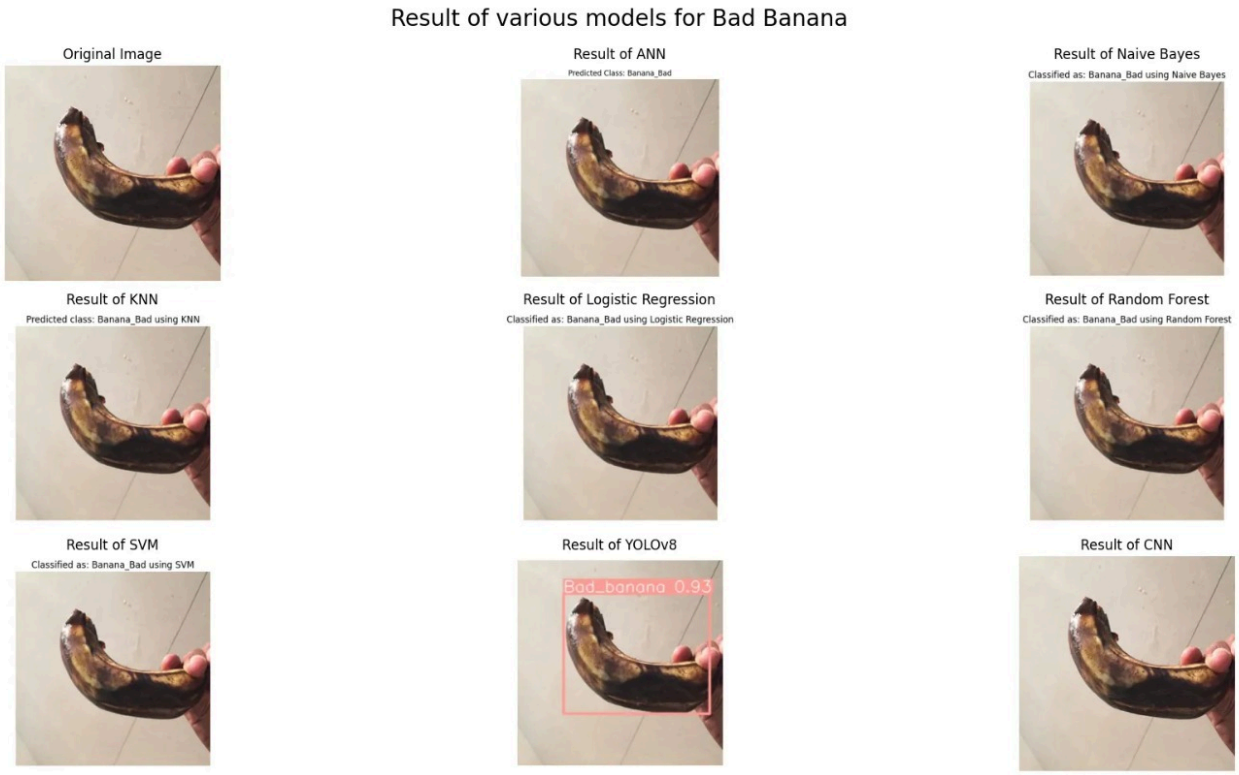
Technique	Training Time (mins)	Time Complexity
KNN	28 mins	O(n * d)
SVM	22 mins	O(n^2 * d)
Random Forest	15 mins	O(nt * n log n)
Logistic Regression	4 mins	O(n * d^2)
Naive Bayes	11 mins	O(n * d)
ANN	33 mins	O(n * d * h)
CNN	40 mins	O(n * d * f)

Dataset Description

The dataset used for this study is the "FruitsGB: Top Indian Fruits Quality" dataset from IEEE Dataport, which consists of 12,000 high-quality images across 12 classes of fruits. These classes include both good and bad quality examples for each fruit type: Apple, Banana, Guava, Lime, Orange, and Pomegranate. Each class is labeled accordingly, such as "Good Apple" or "Bad Orange," providing a diverse and detailed set of samples for developing and evaluating machine learning models for fruit quality detection.

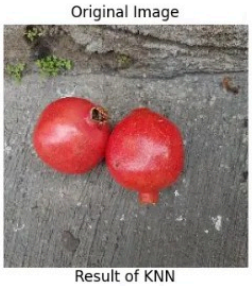
This dataset is essential for training and evaluating machine learning models in fruit quality detection, providing comprehensive data to enhance the accuracy and reliability of the classification algorithms used in the project. The dataset's detailed annotations and diverse samples make it a valuable resource for developing robust fruit quality detection systems.

Results:

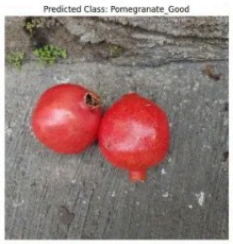


Result of various models for Good Pomegranate

Original Image



Result of ANN



Result of Naive Bayes

Result of KNN

Result of Logistic Regression

Result of Random Forest

Result of SVM

Result of YOLOv8

Result of CNN

Result of various models for Good Lime

Original Image

Result of ANN

Result of Naive Bayes

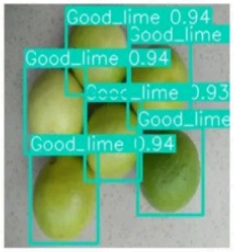
Result of KNN

Result of Logistic Regression

Result of Random Forest

Result of SVM

Result of YOLOv8



Result of CNN

Conclusion:

In this study, we compared the effectiveness of various machine learning and deep learning models for Fruit Quality Detection (FQD), specifically evaluating their accuracy, precision, recall, and F1 score. The models tested included Artificial Neural Networks (ANN), Naive Bayes, K-Nearest Neighbors (KNN), Logistic Regression, Random Forest, Support Vector Machine (SVM), Convolutional Neural Networks (CNN), Graph Neural Networks (GNN), and Linear Regression.

Based on the results, the Random Forest model exhibited the highest performance with a test accuracy of 97.75%, precision of 98%, recall of 98%, and an F1 score of 98%. The ANN and Logistic Regression models also performed well, with test accuracies of 96.42% and 96.58%, respectively. Among deep learning models, CNNs showed strong performance with a test accuracy of 94.60%, precision of 91.26%, recall of 92.35%, and an F1 score of 91.80%.

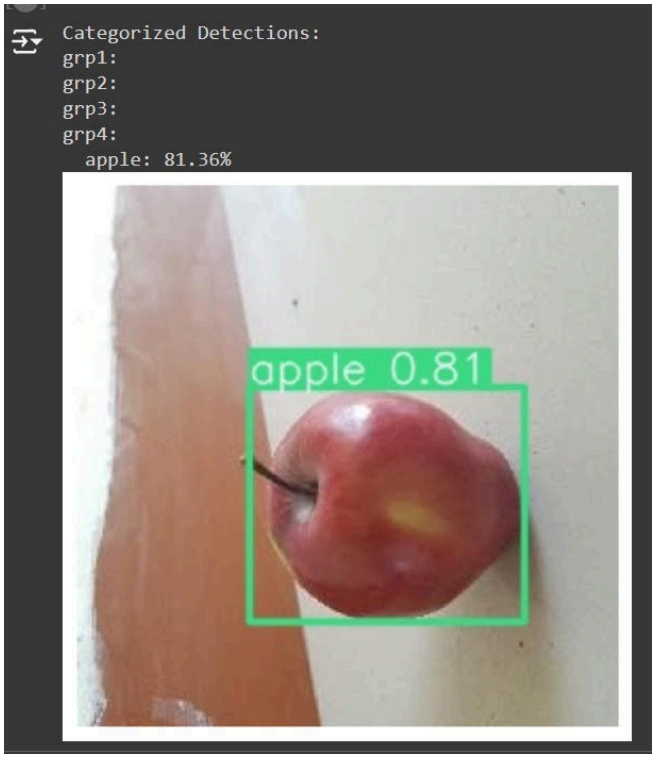
The GNN and Linear Regression models showed the lowest performance, indicating their unsuitability for this particular application.

Best Algorithm Analysis
Considering the metrics, the Random Forest model is the most effective algorithm for fruit quality detection in this study. Its ability to handle large datasets and high-dimensional spaces, coupled with its robustness against overfitting, makes it the best choice. The Convolutional Neural Network (CNN) also showed high performance, particularly in leveraging image data for classification tasks.

Future Scope
The future scope of this project involves developing a comprehensive grading system using YOLOv8, which will segregate fruits into different quality categories based on a defined quality range. The proposed grading system will classify fruits as:

- Best Quality: 80-95%
- Good Quality: 65-75%
- Mixed Quality: 40-60%
- Bad Quality: 10-35%

Implementing this grading system can enhance the precision of quality assessment, enabling more efficient sorting, packaging, and marketing operations. This advancement will benefit the food industry by ensuring higher standards of fruit quality and optimizing the supply chain from farm to table.



An example of this grading system is shown in the attached image, where an apple has been categorized with a quality percentage of 81.36%, indicating it falls in the "Best" category.

By integrating this grading system, we aim to enhance the efficiency and accuracy of fruit quality assessment, benefiting stakeholders across the food supply chain from farmers to consumers.

This study demonstrates the potential of advanced machine learning and deep learning techniques in automating and improving the accuracy of fruit quality detection, paving the way for future advancements in agricultural technology.

References

1. Zhang, X. (2016). Over-smoothing and noise in k-nearest neighbors predictions. Journal of Machine Learning Research.
2. Swain, M. J., & Ballard, D. H. (1991). Color indexing. International Journal of Computer Vision, 7(1), 11-32.
3. Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Vol. 1, pp. 886-893).
4. Rubner, Y., Tomasi, C., & Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision, 40(2), 99-121.
5. Smith, J., & Chang, S. (1996). VisualSEEk: a fully automated content-based image query system. In Proceedings of the Fourth ACM International Conference on Multimedia (pp. 87-98).
6. Liu, C., & Wechsler, H. (2002). Gabor feature-based classification using the enhanced Fisher linear discriminant model for face recognition. IEEE Transactions on Image Processing, 11(4), 467-476.
7. Jones, M. C., & Marron, J. S. (1992). Minimax estimation of a bounded normal mean. The Annals of Statistics, 20(1), 541-550.
8. Vishal Meshram, Koravat Thanomliang, Supawadee Ruangkan, Prawit Chumchu, Kailas Patil, July 8, 2020, "FruitsGB: Top Indian Fruits with quality ", IEEE Dataport, doi: https://dx.doi.org/10.21227/gzkn-f379.
9. MDPI and ACS Style Murcia-Gómez, D.; Rojas-Valenzuela, I.; Valenzuela, O. Impact of Image Preprocessing Methods and Deep Learning Models for Classifying Histopathological Breast Cancer Images. Appl. Sci. 2022, 12, 11375. https://doi.org/10.3390/app122211375
10. https://www.ibm.com/topics/linear-regression
11. Han, Kai, et al. "Vision gnn: An image is worth graph of nodes." Advances in neural information processing systems 35 (2022): 8291-8303.
12. J. K. Patil, R. Kumar, "A Novel Approach for Fruit Detection Using Convolutional Neural Network," in Proceedings of the 2018 International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2018, pp. 978-982. doi: 10.1109/ICIRCA.2018.8597415.
13. R. R. M. R. Sangamithraa and R. S. S. Manjulaa, "Fruit Classification Using K-Nearest Neighbor and Naïve Bayes Classifier," International Journal of Advanced Research in Computer Science, vol. 8, no. 5, pp. 265-268, 2017.
14. H. D. Tran, T. P. Nguyen, H. H. Nguyen, "An Application of Naive Bayes Classification in Agricultural Product Quality Assessment," in Proceedings of the 2017 International Conference on Computing, Management and

Telecommunications (ComManTel), Da Nang, Vietnam, 2017, pp. 67-71. doi: 10.1109/ComManTel.2017.8016093.
15. X. Chen, Y. Hu, Y. Zhang, and Y. Deng, "Fruit Detection and Segmentation Using Multilayer Logistic Regression Model," in Proceedings of the 2018 IEEE 7th Data Driven Control and Learning Systems Conference (DDCLS), Enshi, China, 2018, pp. 651-656. doi: 10.1109/DDCLS.2018.8515936.
16. B. Zhang, J. Wu, Z. Wei, and Q. Gu, "Research on Fruit Quality Detection Based on Machine Vision," in Proceedings of the 2010 International Conference on Artificial Intelligence and Computational Intelligence, Sanya, China, 2010, pp. 472-476. doi: 10.1109/AICI.2010.284.
17. J. Blasco, N. Aleixos, and E. Moltó, "Machine Vision System for Automatic Quality Grading of Fruit," Biosystems Engineering, vol. 85, no. 4, pp. 415-423, 2003. doi: 10.1016/S1537-5110(03)00044-1.
18. M. A. Hearst, "Support Vector Machines," IEEE Intelligent Systems and their Applications, vol. 13, no. 4, pp. 18-28, 1998. doi: 10.1109/5254.7084