University of Science and Technology of Hanoi

ICT3.011 Machine Learning and Data Mining II

PROJECT REPORT: Fruit Quality Classification

Class: Data Science - B2
[Group 2]

Name	Student ID
Hoang Kim Huong	22BA13158
Nguyen Thi Ngoc Lan	22BA13186
Doan Ngoc Linh	22BA13190
Le Sy Han	23BI14150
Le Hoang Dat	23BI14087

Academic year: 2024 - 2025

Contents

Abstract

This report examines the application of convolutional neural networks (CNNs) in the classification of fruit images to distinguish between three types of fruits (apples, bananas, and oranges) and their freshness status (fresh/rotten) samples. Ensuring fruit quality is crucial in the food industry to prevent health risks and reduce waste. A dataset of 1800 images was collected, consisting of equal numbers of 6 classes: 'Fresh_Apple', 'Fresh_Banana', 'Fresh_Orange', 'Rotten_Apple', 'Rotten_Banana', 'Rotten_Orange' and preprocessed using resizing, normalization techniques. A CNN model was developed using TensorFlow and trained with a categorical cross-entropy loss function and the Adam optimizer. The model achieved an accuracy of 95.37% on the validation set and 91% on the test set, indicating a strong ability to generalize. Performance was further analyzed using a confusion matrix and precision-recall metrics. These results suggest that deep learning, specifically CNNs, can be effectively utilized for automated food quality control. Future work may involve expanding the dataset, testing on real-time systems, and integrating the model into supply chain monitoring solutions.

1 Introduction

Background and context

Fruit freshness classification is crucial in the agricultural and food industries for consumer health, reducing food waste, and improving supply chain efficiency. Post-harvest spoilage contributes to global food losses and compromises other produce quality. Rapid detection of rotten fruits is essential for product quality and minimizing losses. Currently, fruit quality testing relies on manual methods, which are time-consuming and prone to errors. The increasing scale of agricultural production and demand for standardized, high-quality fruits necessitate more efficient, reliable, and scalable methods for freshness classification. Addressing these challenges is essential for developing accurate, efficient, and scalable fruit freshness classification systems.

Project objectives

The project aims to create an automated system that accurately classifies the freshness of fruits using image-based analysis. This will improve the efficiency of quality control processes in agricultural supply chains, reduce post-harvest losses, support food safety initiatives, and contribute to smart agriculture and modern supply chain management practices. The project aims to address the growing need for scalable, accurate, and efficient technologies in food production and distribution.

Scope

The project is developed by applying Convolutional Neural Network (CNN) model for multi-class fruit classification, focusing on three types of fruit: apples, bananas, and oranges. The model will be preprocessed for training and evaluation. The project is limited to these three fruits and does not classify fruits into different levels of rottenness (only rotten and fresh). Limiting the model to simple tasks will make the model highly accurate, effective, and lay the foundation for expanding to more fruits and rottenness levels in the future.

2 Related Work

Initially, the project was developed by applying the K-Means Clustering Algorithm to cluster and detect the rotten region. This model is good at highlighting the regions by the difference between colors and textures, therefore, it yields good results for fruits with high contrast and a noticeable

difference between colors. However, in some cases, if the fruit has a harmonious color, the rotten region is not clearly separated, or the background region has a similarity to the color of the fruit, the method will make mistakes when clustering.

K-Means Clustering Algorithm will use the Elbow method to determine the optimal number of clusters (k) and cluster the image based on its pixels, then use a threshold to convert the original RGB format into the binary format. The masks will be created to browse through each pixel, then decide if it is the rotten region, where its intensity is less than the threshold. After that, the images are converted into feature vectors, which are the input for the Random Forest model. This model will utilize the feature vectors and their label to learn and train, then it will be used to predict labels (rotten or fresh fruit).

However, the threshold plays an important role in identifying the rotten region and extracting features, resulting in influences the performance of the classification. Some fruits do not have a clear difference between the rotten and normal regions, making it challenging to determine a threshold. Additionally, in some cases, there are many rotten colors on the same fruit (for example, when oranges are moldy, they have white mold and green mold), it might be necessary to choose multiple thresholds, leading to a more complex model. Therefore, if applying this model for various kinds of fruits, there should be a corresponding coding part for each threshold choice.

3 Data Description

The dataset, which is images of a fruit in RGB color format used in this project, was obtained from Kaggle. There are three kinds of fruits: Apple, Banana, and Orange; each kind has two statuses: rotten and fresh. Specifically, the model classifies each input image into one of the following six categories: Fresh_Apple, Rotten_Apple, Fresh_Banana, Rotten_Banana, Fresh_Orange, Rotten_Orange.

To prepare the dataset for model training, a series of preprocessing steps was applied to the raw images. The following outlines the preprocessing pipeline:

- **Image Collection:** Images were organized into six folders, each corresponding to one class label. For each folder, there are 300 images, ensuring a balanced dataset across all classes, resulting in a total of 1,800 images.
- **Image Loading**: Each image was read from disk using OpenCV (cv2.imread). Images that failed to load were skipped, and a warning was logged.
- **Resizing:** All images were resized to a uniform dimension of 256×256 pixels to standardize input size across the dataset. This ensures compatibility with the input requirements of Convolutional Neural Network (CNN) architectures.
- **Normalization:** Pixel values, initially in the range [0, 255], were scaled to the [0.0, 1.0] range. This was achieved by converting pixel data to 32-bit floating-point format (float32) and dividing by 255. Normalization accelerates the convergence of deep learning models and improves numerical stability during training.
- **Label Extraction:** Labels were assigned based on the folder names where the images were stored. Each label represents a combination of fruit type and freshness condition.
- **Data Storage:** All preprocessed images were stored in a NumPy array with the shape (1800, 256, 256, 3), representing:

  - 1800 images
  - Each image of size 256×256 pixels
  - 3 color channels (RGB)

- **Encode categorical labels into numeric form**
  - Since the machine learning model cannot understand categorical labels like "fresh banana", "fresh orange" ..., the labels must be converted into numeric form so that the machine learning model can understand and perform learning.
  - The labels are encoded in two ways:
    * Label encoding: converts each unique category into an integer, using LabelEncoder from the sklearn package:

Label	Label Encoding
freshapples	0
freshbanana	1
freshoranges	2
rottenapples	3
rottenbanana	4
rottenoranges	5

    * One hot encoding: converts each category into a binary vector, using OneHotEncoder from the sklearn package:

Label	One Hot Encoding
freshapples	[1; 0; 0; 0; 0; 0]
freshbanana	[0; 1; 0; 0; 0; 0]
freshoranges	[0; 0; 1; 0; 0; 0]
rottenapples	[0; 0; 0; 1; 0; 0]
rottenbanana	[0; 0; 0; 0; 1; 0]
rottenoranges	[0; 0; 0; 0; 0; 1]

- **Shuffle & split the dataset:** Shuffle the ordered dataset ('freshapples', 'freshbanana', 'freshoranges', 'rottenapples', 'rottenbanana', 'rottenoranges') into the unordered dataset to the patterns will randomly distribute and reduce the bias. Split the dataset into three subsets, the splitting process was conducted in two steps:
  - Initial Split: The dataset was split into a training + validation set (90%) and a testing set (10%).
  - Secondary Split: The training + validation set was further divided into the training set (80% of the 90% subset) and the validation set (20% of the 90% subset).

4 Model

4.1 Algorithm: Convolutional Neural Network (CNN)

4.1.1 Introduction to CNN

A Convolutional Neural Network (CNN), also known as a ConvNet, is a specialized deep learning model primarily designed for object recognition tasks, including image classification, detection, and segmentation. CNNs are employed in various practical scenarios, including autonomous vehicles, security camera systems, and others. [1]

CNN is chosen for this problem because of these advantages [1]:

- CNNs are distinguished from classic machine learning algorithms such as SVMs and decision trees by their ability to autonomously extract features at a large scale, bypassing the need for manual feature engineering and thereby enhancing efficiency.
- The convolutional layers grant CNNs their translation-invariant characteristics, empowering them to identify and extract patterns and features from data irrespective of variations in position, orientation, scale, or translation.

4.1.2 Components

The convolutional neural network consists of four main parts [1]:

1. **Convolutional layers:** The first and the most important building block of a CNN. Convolution involves moving a filter or kernel, across an image (a grid of pixel values). At each step, the filter looks at a small part of the image and performs a simple calculation. There are several filters of equal size applied, and each filter is used to recognize a specific pattern from the image, such as the curving of the digits, the edges, the whole shape of the digits, and more.
2. **Activation Function:** Helps the network learn non-linear relationships between the features in the image, hence making the network more robust for identifying different patterns.
3. **Pooling layers:** Used to pull out the most significant features from the convoluted matrix. This is done by applying some aggregation operations, which reduce the dimension of the feature map (convoluted matrix), hence reducing the memory used while training the network.
4. **Fully connected layers:** The last layer of the CNN, and its inputs correspond to the flattened one-dimensional matrix generated by the last pooling layer. The final label of the image is predicted here.

In this model, 2 activation functions have been used:

- **ReLU:** Short for "Rectified Linear Unit", it is very important and popular for deep learning models, as it is the default choice for many architectures due to its simplicity and efficiency. [2]

$$f(x) = \max(0, x)$$

where x is the input to the neuron.

- **Softmax:** A mathematical function that transforms a vector of raw model outputs, known as logits, into a probability distribution. [3]

$$\sigma(z_i) = \frac{\exp(z_i)}{\sum_{j=1}^{K} \exp(z_j)}$$

where:

  - z_i: the input value (logit) for class i.
  - $\exp(z_i)$: the exponential function applied to z_i.
  - K: the number of classes.

4.2 Architecture

The CNN model is implemented by the Keras Sequential API in the 'Tensorflow' library. It was designed for multi-class image classification with 6 output categories. The model takes RGB images of size 256x256 as input and applies a series of convolutional, pooling, and dropout layers to extract features and prevent overfitting.

The model contains 3 convolutional blocks and a fully-connected layer:

1. **First Convolutional Block**

Layer Type	Hyperparameter	Output Shape
Conv2D	32 filters of size 3x3, ReLU	(254, 254, 3)
MaxPooling2D	3x3 pool, stride (2,2)	(126, 126, 32)
Dropout	Rate = 0.3	(126, 126, 32)

2. **Second Convolutional Block**

Layer Type	Hyperparameter	Output Shape
Conv2D	64 filters of size 3x3, ReLU	(124, 124, 64)
MaxPooling2D	3x3 pool, stride (2,2)	(61, 61, 64)
Dropout	Rate = 0.3	(61, 61, 64)

3. **Third Convolutional Block**

Layer Type	Hyperparameter	Output Shape
Conv2D	128 filters of size 2x2, ReLU	(60, 60, 128)
MaxPooling2D	2x2 pool, stride (2,2)	(30, 30, 128)
Dropout	Rate = 0.3	(30, 30, 128)

4. **Fully-connected layer**

Layer Type	Hyperparameter	Output Shape
Flatten		115200
Dense	256 units, ReLU	256
Dropout	Rate = 0.3	256
Output Dense	6 units, softmax activation	6

Note:

- Output shape (x, y, z) means the features map has the size $x \times y$ and has z chanels.
- **Conv2D:** 2D Convolutional Layer.
- **MaxPooling2D:** Max pooling operation for 2D spatial data.
- **Dropout:** Randomly turns off some neurons in the feature map (For regularization).
- **Dense**: Flattens the 2D feature maps into a 1D vector and applies a fully connected (dense) layer.
- **Output Dense**: the final fully connected (dense) layer in a neural network that produces the final prediction. The number of units is the number of classes.

In conclusion, this CNN model progressively extracts hierarchical features from input images using three convolutional layers with increasing filter sizes ($32 \rightarrow 64 \rightarrow 128$), interleaved with max pooling for downsampling and dropout for regularization. The final fully connected layers learn to map the extracted features to one of six classes. The softmax activation in the output layer provides class probabilities. The model is trained with the cross-entropy loss function and the Adam optimizer.

5 Result and Evaluation

After applying the model on the dataset, evaluations will be considered, including the comparison between training and validation performance, the confusion matrix and classification metrics.

5.1 Training vs Validation Performance

After applying the model to the training subset and the validation subset, the accuracy is shown in Figure 1. The model achieved a maximum training accuracy of 98. 98% and a maximum validation accuracy of 95.37%. These results indicate that the model has learned to generalize very well.

Along with the accuracy, the loss value is also calculated by using the categorical cross-entropy (CCE) loss function, which is a loss function used in multi-class classification problems. Given a true class label (as a one-hot encoded vector) and a predicted probability distribution (typically output by a softmax activation), it measures the difference between the predicted probability distribution and the true distribution of classes. The smaller the loss function value, the better the model performs. [4]

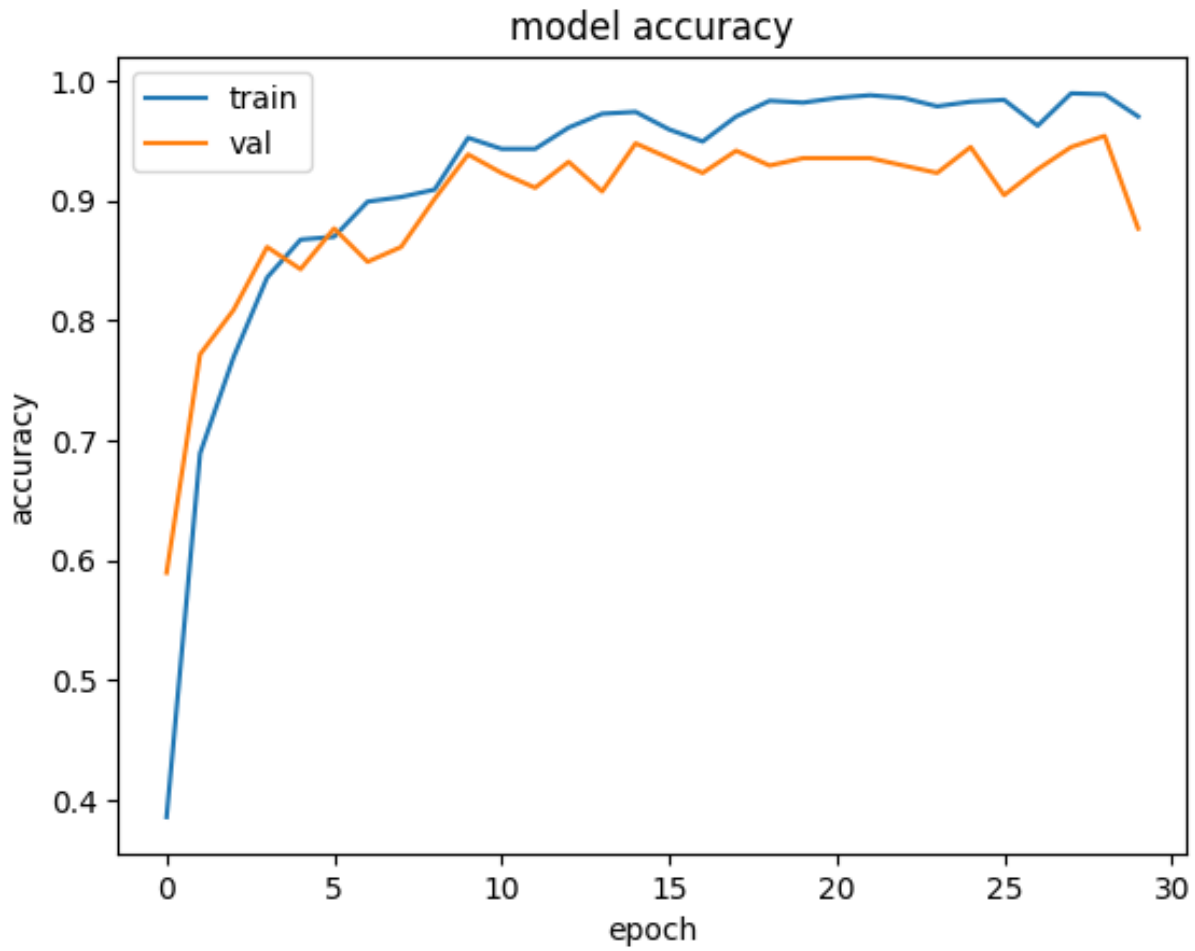

Figure 1: Training and Validation Performance of the Fruit Quality Classification Model

The formula of the CCE loss function is:

$$L(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

Where:

- $L(y, \hat{y})$ is the categorical cross-entropy loss.
- y_i is the true label (0 or 1 for each class) from the one-hot encoded target vector.
- $\hat{y}_i$ is the predicted probability for class i.
- N is the number of classes.

The loss curves after applying the model to the training subset and the validation subset are plotted in Figure 2. It can be seen that both curves reach very low values. The training loss curve fell from the loss value of 7.0639 to the lowest value of 0.0242, while the validating loss curve improved the loss value from 1.2338 to the lowest point of 0.2385, showing that the model performs fairly well after training and validating.

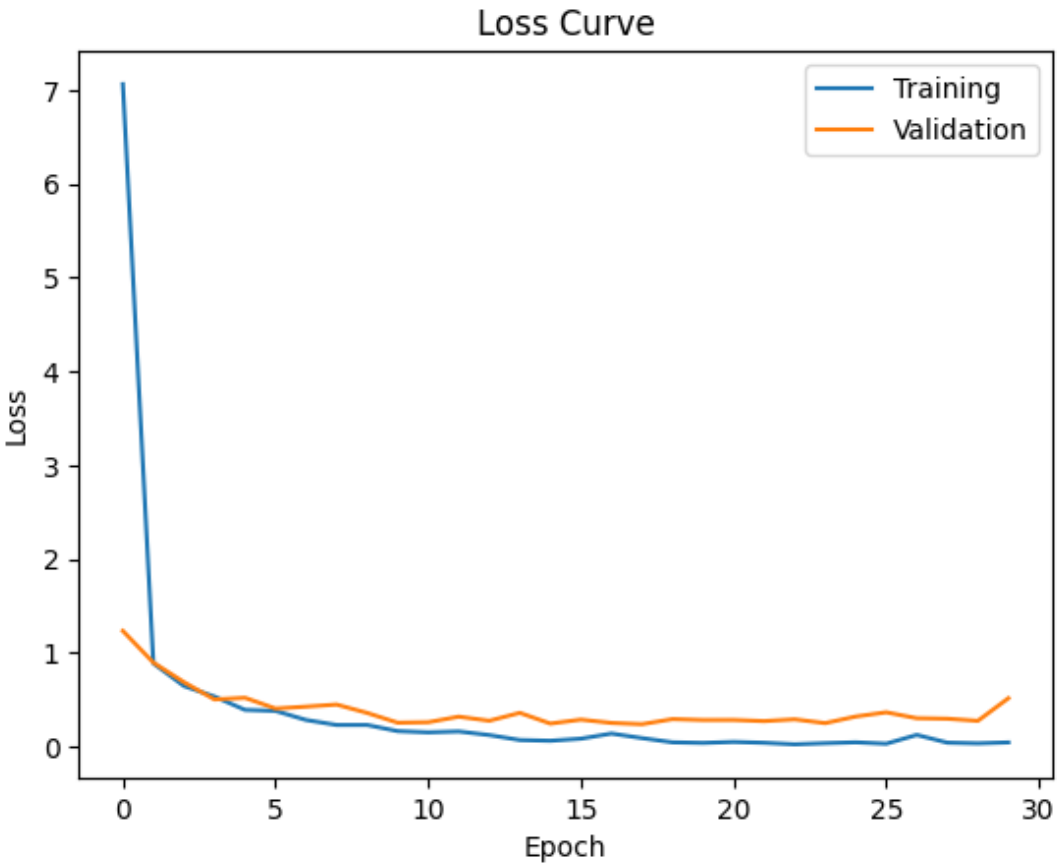

Figure 2: Training and Validation Loss Curve of the Fruit Quality Classification Model

5.2 Confusion Matrix

The confusion matrix for classification on the testing data is obtained in Table 1

		Actual					
		Class 0	**Class 1**	**Class 2**	**Class 3**	**Class 4**	**Class 5**
Predicted	**Class 0**	30	0	0	2	0	0
	Class 1	0	32	0	0	1	0
	Class 2	0	1	36	0	0	2
	Class 3	3	0	0	22	1	0
	Class 4	0	1	0	0	18	0
	Class 5	1	1	3	1	0	25

Table 1: Confusion Matrix of the Fruit Quality Classification

It can be seen that the model performs very well with most of the images labeled correctly. Most of the False Positive (FP) values lie in class 3(rottenapples) and class 5(rottenoranges), showing that the model confused objects with these two classes the most. On the other hand, class 0(freshapples) has the most False Positive (FP) values, indicating that this is the class with the most incorrectly labeled objects.

5.3 Classification Metrics

Obtained by using the "classification_report" function in the module "metrics" in the "sklearn" library, Table 2 illustrates some classification metrics of the model across 6 classes, like precision, recall, F1-Score, along with the number of data points (support) in the test dataset. There are also "Macro average", which calculates the mean of each metric for all classes, and "Weighted average", which calculates the weighted average of each metric by support. [5]

Class	**Precision**	**Recall**	**F1-score**	**Support**
0	0.88	0.94	0.91	32
1	0.91	0.97	0.94	33
2	0.92	0.92	0.92	39
3	0.88	0.85	0.86	26
4	0.90	0.95	0.92	19
5	0.93	0.81	0.86	31
Accuracy			0.91	180
Macro average	0.90	0.91	0.90	180
Weighted average	0.91	0.91	0.90	180

Table 2: Classification Report of the Fruit Quality Classification model

The formula for these metrics are:

$$\text{precision} = \frac{\text{TP}}{\text{TP+FP}}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP+FN}}$$

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

In general, all the metric results are larger than or equal to 0.80. Class 1(freshbanana) performs the best metrics with a precision of 0.91, a near-perfect recall of 0.97, and the F1 score of 0.94. The remaining classes' metrics range from moderate to good. The overall accuracy of the classification is 0.91. In conclusion, the model is very good and efficient for the dataset.

6 Discussion

The results of this study demonstrate that CNNs are highly effective in classifying fruit images by both type and freshness. The model achieved a validation accuracy of 95.37% and a test accuracy of 91%, demonstrating good generalization to unseen data. These results confirm the potential of CNNs in automating fruit quality classification tasks, which are traditionally performed manually and subjectively.

The high accuracy suggests that the chosen preprocessing methods (resizing and normalization) and the CNN architecture were appropriate for this classification task. However, some misclassifications were observed, particularly between fresh and rotten samples of the same fruit, which may be attributed to subtle visual differences or inconsistent lighting conditions in the dataset.

Despite the promising results, the study is limited by the relatively small dataset size (1,800 images), which may restrict the model's robustness across broader conditions, such as different lighting, angles, or fruit varieties. Moreover, real-time deployment was not tested in this study.

Future work should focus on increasing dataset diversity, incorporating data augmentation, testing the model in real-world environments (e.g., conveyor belt systems), and optimizing for speed and performance on embedded devices for potential use in smart agriculture or retail settings.

7 Conclusion

Using the K-Means Clustering Algorithm to detect and cluster rotten regions, but it is not effective, so CNN is used as an alternative and a more effective model. By eliminating the requirement for manual feature engineering, CNN can extract features on a wide scale autonomously, increasing efficiency. Convolutional layers provide CNNs their translation-invariant properties, which allow them to identify and extract features and patterns from input despite changes in scale, location, orientation, or translation.

In the near future, the dataset needs to be expanded (diversify the types of fruits). Enhancing the model's ability to evaluate and classify fruits into different levels of rottenness, or evaluate multiple types of fruits and their freshness status at the same time. In addition, it is possible to develop not only to detect damage on the outside of fruits, but also to detect it on the inside, and deploy the model into real life.

References

[1] Zoumana Keita, "What is a Convolutional Neural Network (CNN)?". Available: `https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns`. [Accessed: 07-Jun-2025]

[2] GeeksforGeeks, "ReLU Activation Function in Deep Learning". Available: `https://www.geeksforgeeks.org/relu-activation-function-in-deep-learning/`. [Accessed: 08-Jun-2025]

[3] Rajesh Kumar, "Softmax Activation Function in Python: A Complete Guide". Available: `https://www.datacamp.com/tutorial/softmax-activation-function-in-python`. [Accessed: 08-Jun-2025]

[4] GeeksforGeeks, "Categorical Cross-Entropy in Multi-Class Classification". Available: `https://www.geeksforgeeks.org/categorical-cross-entropy-in-multi-class-classification/`. [Accessed: 08-Jun-2025]

[5] scikit-learn, "precision_recall_fscore_support - scikit-learn 1.6.1 documentation". Available: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html#sklearn.metrics.precision_recall_fscore_support`. [Accessed: 07-Jun-2025]

Appendix: Source Code Repository

The complete source code used in this project is available at the following GitHub repository:
https://github.com/watermelon-3012/ML2-Group2-Fruit_Quality_Classification

The repository includes the following files:

- `Fruit.ipynb` – Contains everything from data preprocessing, model and performance evaluation, and metrics.
- `README.md` – Introduction of the project
- `Group 2 - ML2 - Fruit Quality Classification.pdf` – The report for the project.
- `Clustering.ipynb` – Model for Related Works