K-NN vs Decision tree: classification of apples based on the quality

Vasyl Yarmolenko

Alanya University, Faculty of Engineering and Natural Sciences
Computer Engineering Department

TE1414 Introduction to Data Science, Spring 2024–2025

May 15, 2025

1. 1. Introduction

1.1 Background

This study utilizes a dataset composed of various physical and chemical properties of apples, with the objective of classifying fruit quality. The dataset encompasses multiple numerical attributes such as size, weight, sweetness, crunchiness, juiciness, ripeness, and acidity, along with a qualitative target label representing fruit quality. These attributes provide a comprehensive basis for building predictive models capable of distinguishing high-quality apples from lower-quality ones based on their measurable features.

1.2 Problem Statement

The central research question addressed in this project is the identification of the most suitable classification algorithm for analyzing structured tabular data with both continuous features and a categorical target. Specifically, the aim is to evaluate and compare the predictive performance of two supervised learning algorithms—K-Nearest Neighbors (K-NN) and Decision Tree Classifier—in the context of fruit quality classification. This comparative analysis provides insights into their effectiveness when applied to real-world datasets with multivariate characteristics.

1.3 Objectives

The main objectives of this project are as follows:

- **Exploratory Data Analysis and Cleaning**: To assess the structure and integrity of the dataset by identifying missing values, outliers, and data types, and applying necessary preprocessing steps.
- **Feature Scaling and Normalization**: To apply appropriate scaling and transformation techniques (e.g., Min-Max normalization, Z-score standardization) for ensuring that numerical features contribute proportionately to model training.
- **Model Development**: To implement and train K-NN and Decision Tree classifiers, optimizing hyperparameters to improve predictive accuracy and reduce overfitting.

- **Model Evaluation and Comparison**: To assess model performance using standard classification metrics such as precision, recall, F1-score, and error-based metrics (MAE, MSE, RMSE), and determine which model generalizes better to unseen data.

1.4 Structure of the Report

The report is organized into several key sections to systematically present the methodology and findings of this project:

- **Introduction** – Provides the background context, outlines the problem, defines the objectives, and describes the report's organization.
- **Methodology** – Details the analytical approach, tools, and techniques applied in data preprocessing and model development.
- **Implementation**
  - **3.1 Dataset Description** – Explains the data source, attributes, and potential use cases.
  - **3.2 Data Preprocessing** – Describes the procedures used for cleaning, transforming, and preparing the data.
  - **3.3 Model Training and Evaluation** – Discusses the training process, evaluation metrics, and experimental design.
  - **3.4 Model Iterations and Results** – Presents the performance results from various model configurations.
- **Conclusion** – Summarizes the study's findings and provides insights for future work and potential improvements.
- **References and Appendices** – Lists the data sources, libraries, and tools used, along with supplementary materials such as code and visualizations.

2. Methodology

2.1 Approach Overview

The methodological framework adopted for this project follows a standard supervised machine learning pipeline. It begins with data exploration and preprocessing, continues through feature engineering and normalization, and concludes with the implementation and evaluation of classification models. The primary aim is to assess and compare the predictive capabilities of K-Nearest Neighbors (K-NN) and Decision Tree classifiers for binary classification of apple quality. The study emphasizes rigorous data preparation, model training, and iterative performance evaluation to ensure reliability and generalizability.

2.2 Tools and Technologies

The following tools and libraries were employed throughout the project:

- **Python** – Core programming language used for implementation.
- **Pandas** – For data manipulation, cleaning, and analysis.
- **NumPy** – For numerical operations and array handling.
- **Matplotlib & Seaborn** – For data visualization and plotting.
- **Scikit-learn (sklearn)** – For implementing machine learning models, data splitting, preprocessing, and evaluation metrics.

All experiments were conducted in a cloud-based environment using **Google Colab**, which facilitated seamless code execution and visualization.

2.3 Data Preprocessing Strategy

The data preprocessing stage included multiple steps to ensure the quality and readiness of the dataset for modeling.

Initial Inspection

- Used df.head(), df.shape, df.describe(), and df.dtypes to inspect the structure, size, and statistical properties of the dataset.
- Verified that the dataset contained 4,000 instances and 9 columns, including 7 input features and 1 target variable (Quality).

Missing Values

- Performed null value analysis using df.isnull().sum() and confirmed the absence of any missing values.

Duplicate Records

- Checked for duplicate entries with df.duplicated() and removed any if present to avoid redundancy and bias in model training.

Outlier Detection

- Employed the Interquartile Range (IQR) method to detect outliers in numerical features.
- Visualized feature distributions using box plots to identify extreme values that could influence model performance.

Feature Type Classification

- Categorized features as continuous numerical variables.
- Target variable (Quality) was identified as binary categorical, later encoded as 0 and 1.

Feature Elimination

- Dropped the A_id column as it served merely as an identifier and carried no predictive information.

Normalization and Scaling

Three standard normalization strategies were evaluated:

1. **Z-Score Standardization** – Applied to center data around zero with unit variance.
2. **Log-Normalization** – Applied to features with skewed distributions based on Shapiro-Wilk test results.
3. **Min-Max Scaling** – Scaled values to a fixed range [0, 1] to preserve data structure and minimize distortion.

After comparative analysis, **Min-Max Scaling** was selected due to its superior performance in compressing feature ranges and improving model convergence.

2.4 Modeling Method

Two supervised learning algorithms were implemented:

- **K-Nearest Neighbors (K-NN)**: A non-parametric instance-based learner that classifies samples based on majority voting among their k-nearest neighbors in feature space. Various values of k were tested (1, 3, 7) to evaluate the impact of neighbor count on overfitting and generalization.
- **Decision Tree Classifier**: A hierarchical tree-based algorithm that recursively partitions the feature space. Key hyperparameters such as max_depth and min_samples_split were tuned in different configurations to reduce overfitting and enhance test accuracy.

2.5 Rationale for Methodological Choices

The choice of K-NN and Decision Tree classifiers was motivated by their interpretability, low computational cost, and suitability for small-to-medium-sized tabular datasets. K-NN is effective in cases with well-separated classes and is sensitive to the scale of features, necessitating normalization. Decision Trees offer transparent decision-making logic and can handle both categorical and numerical variables. By comparing these two fundamentally different approaches—distance-based and tree-based—this study aims to understand the trade-offs in classification accuracy, generalization, and robustness.

3. Implementation

3.1 Dataset Description

The dataset employed in this study comprises detailed measurements of fruit attributes, particularly apples. Each instance in the dataset represents an individual fruit, described by a range of physicochemical characteristics. The available features include a unique identifier (A_id), physical dimensions (Size, Weight), organoleptic qualities (Sweetness, Crunchiness, Juiciness, Ripeness), and chemical composition (Acidity). The target variable, Quality, indicates the categorical classification of the fruit as either "good" or "bad."

Key Features:

- **A_id**: Unique numerical identifier for each fruit (excluded from modeling)
- **Size**: Dimensional size of the fruit
- **Weight**: Mass of the fruit in grams
- **Sweetness**: Subjective measure of sugar concentration
- **Crunchiness**: Sensory texture rating
- **Juiciness**: Fluid content level
- **Ripeness**: Maturity stage of the fruit
- **Acidity**: pH-related sourness indicator
- **Quality**: Binary classification label indicating overall fruit quality

Potential Applications:

- **Fruit Classification**: Implement supervised classification algorithms to label fruit based on input features.
- **Quality Prediction**: Construct predictive models to infer fruit quality from measurable characteristics.

3.2 Data Preprocessing

The preprocessing phase commenced with a structural inspection of the dataset to ensure data integrity. An extraneous note present in the final row was identified and removed, as it did not constitute a valid data entry. All feature columns were verified to be continuous quantitative variables, while the target variable (Quality) was categorical and binary in nature. The identifier variable A_id was later excluded due to its lack of predictive utility.

No missing values or duplicate records were detected during preliminary analysis. The dataset consisted of 4,000 samples across 9 columns, comprising one target, seven input features, and one identifier. Summary statistics generated using .describe() and structural insights from .shape confirmed the dataset's readiness for further transformation (Fig. 1).

Subsequently, the target variable was label-encoded, mapping "good" and "bad" to 1 and 0, respectively. Statistical measures such as mean, median, and interquartile ranges (IQR) were evaluated to assess distribution symmetry. Among all features, Juiciness exhibited the most balanced distribution, as indicated by equidistant quartiles.

Outlier detection was performed using the IQR method for each feature. Although the target variable exhibited no outliers, all input features contained varying degrees of extreme values. These findings were further supported through histogram and box plot visualizations (Fig. 2, Fig. 3). While most features exhibited near-normal distributions, the Quality variable deviated from normality, with an imbalanced class distribution. Also I handled outliers in a dataset, but these actions does not affect significantly on the result, so I did left them in the dataset.

To assess correlation between features and the target, scatter plots were analyzed (Fig. 4). No linear relationships were observed, suggesting the need for non-linear classifiers.

Three normalization techniques were evaluated to ensure numerical stability in downstream model training:

- **Z-Score Standardization**: Subtracted the mean and scaled by standard deviation
- **Log-Normalization**: Applied to features with skewed distributions
- **Min-Max Scaling**: Scaled features to the [0, 1] interval

Upon empirical comparison, Min-Max normalization yielded the most effective standardization, maintaining range consistency across all features while minimizing distortion, and was thus selected for final model training.

3.3 Model Training and Evaluation

Following data preprocessing and normalization, the dataset was divided into training and testing subsets using an **80:20** stratified split to preserve class distribution across both sets. This division facilitated unbiased performance evaluation on unseen data.

Two supervised classification algorithms were employed in this study:

- **K-Nearest Neighbors (K-NN)**: A non-parametric, instance-based learning algorithm that assigns a class label to a data point based on the majority label of its k nearest neighbors in the feature space. The choice of k significantly influences model performance, with smaller values potentially leading to overfitting and larger values favoring smoother decision boundaries. Several values of k were evaluated to determine the optimal configuration.

- **Decision Tree Classifier**: A tree-structured model that uses recursive partitioning to divide the feature space into homogenous regions based on information gain or impurity measures such as Gini index or entropy. Internal nodes represent feature-based decision rules, while terminal nodes indicate class predictions. Decision Trees are interpretable and capable of capturing non-linear relationships but are prone to overfitting if not properly regularized through parameters such as tree depth and minimum split size.

Model performance was assessed using both classification and error-based evaluation metrics, including:

- **Confusion Matrix**
- **Mean Absolute Error (MAE)**
- **Mean Squared Error (MSE)**
- **Root Mean Squared Error (RMSE)**
- **Precision, Recall, and F1-score**

These metrics provided comprehensive insights into the accuracy, bias, and variance of each model on both training and test data.

3.4 Model Iterations and Results

3.4.1 Model 1 – Baseline K-NN Classifier (k=1)

The initial model used k=1, which yielded perfect accuracy on the training data. However, generalization to the test set was suboptimal due to evident overfitting. This limitation necessitated increasing the number of neighbors to improve model robustness.

3.4.2 Model 2 – K-NN Classifier (k=3)

Upon adjusting k to 3, the classifier achieved more balanced results. While training accuracy slightly decreased, test performance improved in terms of generalization. Evaluation metrics for training and test sets are presented below:

Mean Absolute Error (MAE)	0.0556
Mean Squared Error (MSE)	0.0556
Root Mean Squared Error (RMSE)	0.2358

Table 1. Training Set Evaluation

Classes	0	1
Precision	**0.95**	**0.94**
Recall	0.94	0.95
F1-score	0.94	0.94

Table 2. Classification Report of Training Set

Mean Absolute Error (MAE)	0.0925
Mean Squared Error (MSE)	0.0925
Root Mean Squared Error (RMSE)	0.3041

Table 3. Test Set Evaluation

Classes	0	1
Precision	**0.92**	**0.90**
Recall	0.89	0.92
F1-score	0.91	0.91

Table 4. Classification Report of Test Set

3.4.3 Model 3 – K-NN Classifier (k=7)

Increasing k to 7 further mitigated overfitting. The performance metrics of the training and test sets became more aligned, indicating improved generalization:

Mean Absolute Error (MAE)	0.0766
Mean Squared Error (MSE)	0.0766
Root Mean Squared Error (RMSE)	0.2767

Table 5. Training Set Evaluation

Classes	0	1
Precision	0.93	0.92
Recall	0.92	0.93
F1-score	0.92	0.92

Table 6. Classification Report of Training Set

Mean Absolute Error (MAE)	0.0950
Mean Squared Error (MSE)	0.0950
Root Mean Squared Error (RMSE)	0.3082

Table 7. Test Set Evaluation

Classes	0	1

Precision	**0.92**	**0.89**
Recall	0.88	0.93
F1-score	0.90	0.91

Table 8. Classification Report of Test Set

3.4.4 Model 1 – Decision Tree Classifier (Depth=9, Min Samples Split=2)

The initial Decision Tree classifier achieved high accuracy on the training data. However, test performance revealed moderate overfitting. The following results were recorded:

Mean Absolute Error (MAE)	0.0966
Mean Squared Error (MSE)	0.0966
Root Mean Squared Error (RMSE)	0.3107

Table 9. Training Set Evaluation

Classes	0	1
Precision	0.87	0.94
Recall	0.95	0.86
F1-score	0.91	0.90

Table 10. Classification Report of Training Set

Mean Absolute Error (MAE)	0.1975
Mean Squared Error (MSE)	0.1975
Root Mean Squared Error (RMSE)	0.4444

Table 11. Test Set Evaluation

Classes	0	1
Precision	0.77	0.85
Recall	0.86	0.74
F1-score	0.81	0.79

Table 12. Classification Report of Test Set

3.4.5 Model 2 – Decision Tree Classifier (Depth=10, Min Samples Split=5)

To further refine the Decision Tree model, the maximum depth was extended to 10, and the minimum samples split was increased to 5. This adjustment improved training performance, though test performance remained less consistent:

Mean Absolute Error (MAE)	0.0794
Mean Squared Error (MSE)	0.0794
Root Mean Squared Error (RMSE)	0.2817

Table 13. Training Set Evaluation

Classes	0	1
Precision	**0.90**	**0.94**
Recall	0.94	0.90
F1-score	0.92	0.92

Table 14. Classification Report of Training Set

Mean Absolute Error (MAE)	0.1850
Mean Squared Error (MSE)	0.1850
Root Mean Squared Error (RMSE)	0.4301

Table 15. Test Set Evaluation

Classes	0	1
Precision	0.79	0.84
Recall	0.85	0.78
F1-score	0.82	0.81

Table 16. Classification Report of Test Set

While additional tuning and entropy-based splitting criteria were explored, no further improvement in classification performance was achieved.

4. Conclusion

This project investigated the application of supervised machine learning techniques to classify apple quality using a structured dataset of physicochemical features. Through comprehensive preprocessing, normalization, and model evaluation, two classification algorithms—K-Nearest Neighbors (K-NN) and Decision Tree—were trained, optimized, and compared.

The K-NN classifier demonstrated strong performance, particularly with k=7, achieving a balance between training accuracy and generalization to unseen data. Lower error rates and closely aligned evaluation metrics between training and test sets indicated minimal overfitting. Conversely, the Decision Tree model, while initially overfitting the training data, improved after hyperparameter tuning (max depth = 10, min samples split = 5), yet still exhibited greater variance between training and test performance compared to K-NN.

Overall, the K-NN algorithm outperformed the Decision Tree classifier in terms of consistent accuracy and lower error rates across multiple configurations. This suggests that instance-based learning may be better suited to this specific classification problem, especially when combined with effective feature scaling.

Future enhancements may include:

- Implementing automated hyperparameter optimization (e.g., Grid Search or Randomized Search)
- Evaluating ensemble methods such as Random Forest or Gradient Boosted Trees
- Applying additional feature engineering or dimensionality reduction techniques (e.g., PCA)
- Expanding the dataset to improve model generalization and reduce variance

This project demonstrates the effectiveness of systematic data preparation and iterative evaluation in building reliable machine learning models for binary classification tasks in the domain of agricultural product quality assessment.

5. References

1. Pandas Documentation - https://pandas.pydata.org/docs/reference/frame.html
2. NumPy Documentation - https://numpy.org/devdocs/reference/routines.html
3. Kaggle Dataset Source, 2024, Nidula Elgiriyewithana - https://www.kaggle.com/datasets/nelgiriyewithana/apple-quality
4. Google Colab Notebook Project of Codes, 2025, Vasyl Yarmolenko- https://colab.research.google.com/drive/18LpjabUszoemOSzJ4eUyiwUB6nWgI9TA

6. Appendices

Appendix 1. Python Code:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.metrics import mean_squared_error, mean_absolute_error, classification_report, confusion_matrix
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier, plot_tree
df = pd.read_csv('/content/sample_data/apple_quality.csv')
df .head()
# Convert 'Quality' to numerical
df['Quality'] = df['Quality'].map({'good': 1, 'bad': 0})
print("\nOriginal data dimension:\n")
print(df.shape)
print("\nSummary:\n")
print((df).describe())
missing_values = df.isnull().sum()
print("\nMissing Values (Original Data):\n")
print(missing_values)
# Identify duplicates in the DataFrame
duplicates = df[df.duplicated()]
# Print the duplicate rows
print("\nDuplicates: \n")
print(duplicates)
# Identify data types and inconsistencies
print("\nData Types:")
print(df.dtypes)
df.drop(columns=['A_id'], inplace=True)
numerical_cols = df.columns
#Check for outliers based on IQR
for col in numerical_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]
    print(f"\nOutliers for {col}:\n", outliers)
# Histograms for all numerical features
plt.figure(figsize=(15, 12))
```

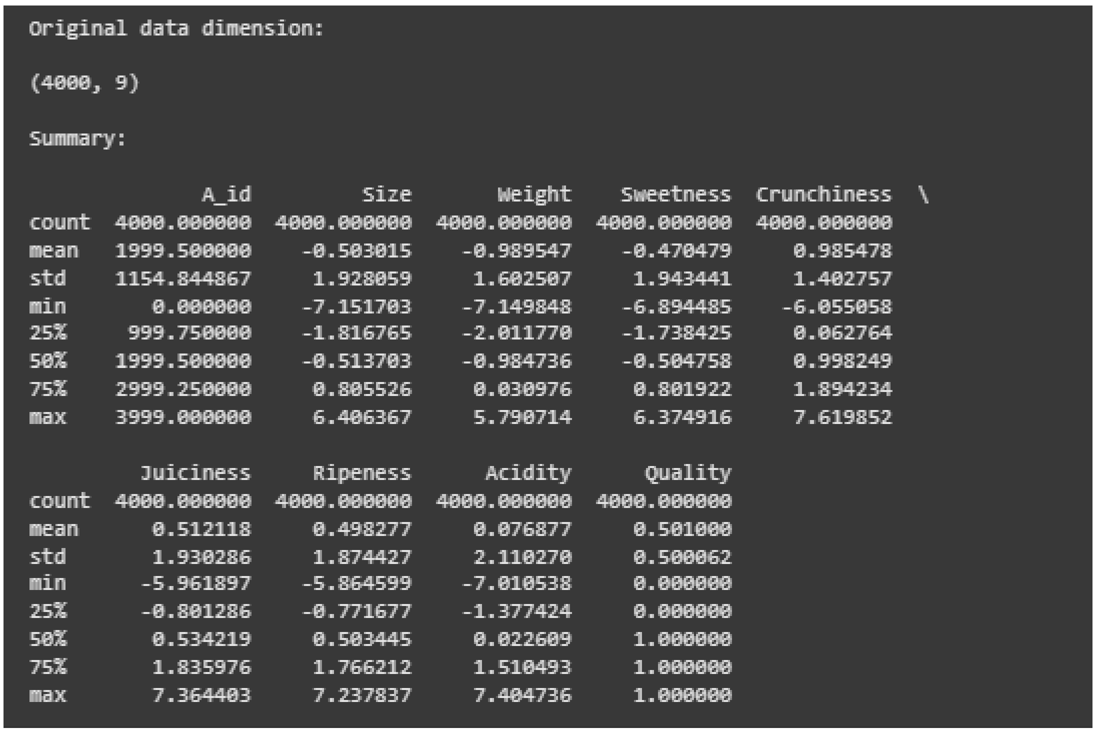
```
for i, col in enumerate(numerical_cols):
    plt.subplot(4, 4, i + 1)
    sns.histplot(df[col], kde=True)
    plt.title(f'Distribution of {col}')
plt.tight_layout()
plt.show()
def boxplot_outliers(df, numerical_cols):
  # Box plots for all numerical features
  plt.figure(figsize=(15, 12))
  for i, col in enumerate(numerical_cols):
      plt.subplot(4, 4, i + 1)
      sns.boxplot(y=df[col])
      plt.title(f'Box Plot of {col}')
  plt.tight_layout()
  plt.show()
boxplot_outliers(df, numerical_cols)
target = 'Quality'
# Scatter plots of 'Quantity' vs. other numerical features
plt.figure(figsize=(15, 12))
for i, col in enumerate(numerical_cols):
    if col != target:
        plt.subplot(4, 4, i + 1)
        sns.scatterplot(x=df[col], y=df[target], hue=df[target], palette='coolwarm')
        plt.title(f"{col} vs {target}")
        plt.xlabel(col)
        plt.ylabel(target)
plt.tight_layout()
plt.show()
# Scale numerical features using StandardScaler
scaler = StandardScaler()
df_scaled = pd.DataFrame(scaler.fit_transform(df[numerical_cols]), columns=numerical_cols)
boxplot_outliers(df_scaled, numerical_cols)
for i, col in enumerate(numerical_cols):
  df_scaled[col] = np.log1p(df[col])
boxplot_outliers(df_scaled, numerical_cols)
min_max_scaler = MinMaxScaler()
df_scaled[numerical_cols] = min_max_scaler.fit_transform(
    df[numerical_cols]
)
boxplot_outliers(df_scaled, numerical_cols)
df_scaled = df.copy()
min_max_scaler = MinMaxScaler()
df_scaled[numerical_cols] = min_max_scaler.fit_transform(
    df[numerical_cols]
)
boxplot_outliers(df_scaled, numerical_cols)
# Separate features and target
X = df.drop(columns=target)
y = df[target]
```

```python
# Split data: 80% training, 20% testing
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=210029807, stratify=y
)
# Check shapes of training and set sets
print("Training set shape:", X_train.shape)
print("Test set shape:", X_test.shape)
# Create and train KNN model (you can tune n_neighbors)
knn_model = KNeighborsClassifier(n_neighbors=7)
knn_model.fit(X_train, y_train)

# Predict on training set
y_train_pred_knn = knn_model.predict(X_train)

# Evaluation metrics
mse_knn = mean_squared_error(y_train, y_train_pred_knn)
mae_knn = mean_absolute_error(y_train, y_train_pred_knn)
rmse_knn = np.sqrt(mse_knn)

print(" KNN Classifier - Training Set Evaluation:")
print(f" MSE: {mse_knn:.4f}")
print(f" MAE: {mae_knn:.4f}")
print(f" RMSE: {rmse_knn:.4f}\n")

# Classification report
print(" Classification Report:")
print(classification_report(y_train, y_train_pred_knn))

# Confusion matrix
conf_matrix_knn = confusion_matrix(y_train, y_train_pred_knn)
sns.heatmap(conf_matrix_knn, annot=True, fmt='d', cmap='Purples')
plt.title("Confusion Matrix - KNN (Train Set)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
# Predict on test set
y_test_pred_knn = knn_model.predict(X_test)

# Evaluation
mse_knn_test = mean_squared_error(y_test, y_test_pred_knn)
mae_knn_test = mean_absolute_error(y_test, y_test_pred_knn)
rmse_knn_test = np.sqrt(mse_knn_test)

print("KNN - Test Set:")
print(f"MSE: {mse_knn_test:.4f}")
print(f"MAE: {mae_knn_test:.4f}")
print(f"RMSE: {rmse_knn_test:.4f}\n")
print("Classification Report:")
print(classification_report(y_test, y_test_pred_knn))
```

```python
# Confusion Matrix
sns.heatmap(confusion_matrix(y_test, y_test_pred_knn), annot=True, cmap='Purples')
plt.title("Confusion Matrix - KNN (Test Set)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
# Create and train the Decision Tree model
tree_model = DecisionTreeClassifier(max_depth=9, min_samples_split=2, random_state=210029807)

tree_model.fit(X_train, y_train)

# Predict on training set
y_train_pred_tree = tree_model.predict(X_train)

# Evaluation metrics
mse_tree = mean_squared_error(y_train, y_train_pred_tree)
mae_tree = mean_absolute_error(y_train, y_train_pred_tree)
rmse_tree = np.sqrt(mse_tree)

print("Decision Tree - Training Set Evaluation:")
print(f"MSE: {mse_tree:.4f}")
print(f"MAE: {mae_tree:.4f}")
print(f"RMSE: {rmse_tree:.4f}\n")

# Classification report
print("Classification Report:")
print(classification_report(y_train, y_train_pred_tree))

# Confusion matrix
conf_matrix_tree = confusion_matrix(y_train, y_train_pred_tree)
sns.heatmap(conf_matrix_tree, annot=True, fmt='d', cmap='Greens')
plt.title("Confusion Matrix - Decision Tree (Train Set)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Decision Tree
plt.figure(figsize=(15, 10))
plot_tree(tree_model, filled=True, feature_names=X_train.columns, class_names=['Good', 'Bad'], rounded=True)
plt.title("Decision Tree Visualization")
plt.show()


# Predict on test set
y_test_pred_tree = tree_model.predict(X_test)

# Evaluation metrics for test set
mse_tree_test = mean_squared_error(y_test, y_test_pred_tree)
```

```
mae_tree_test = mean_absolute_error(y_test, y_test_pred_tree)
rmse_tree_test = np.sqrt(mse_tree_test)

print("Decision Tree - Test Set Evaluation:")
print(f"MSE: {mse_tree_test:.4f}")
print(f"MAE: {mae_tree_test:.4f}")
print(f"RMSE: {rmse_tree_test:.4f}\n")

# Classification report
print("Classification Report:")
print(classification_report(y_test, y_test_pred_tree))

# Confusion Matrix for test set
sns.heatmap(confusion_matrix(y_test, y_test_pred_tree), annot=True, cmap='Greens')
plt.title("Confusion Matrix - Decision Tree (Test Set)")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Plot the Decision Tree (same tree used for training)
plt.figure(figsize=(15, 10))
plot_tree(tree_model, filled=True, feature_names=X_train.columns, class_names=['Good', 'Bad'], rounded=True)
plt.title("Decision Tree Visualization (Test Set)")
plt.show()
```

Appendix. Figures



```
Original data dimension:

(4000, 9)

Summary:

              A_id         Size       Weight    Sweetness  Crunchiness  \
count  4000.000000  4000.000000  4000.000000  4000.000000  4000.000000
mean   1999.500000    -0.503015    -0.989547    -0.470479     0.985478
std    1154.844867     1.928059     1.602507     1.943441     1.402757
min       0.000000    -7.151703    -7.149848    -6.894485    -6.055058
25%     999.750000    -1.816765    -2.011770    -1.738425     0.062764
50%    1999.500000    -0.513703    -0.984736    -0.504758     0.998249
75%    2999.250000     0.805526     0.030976     0.801922     1.894234
max    3999.000000     6.406367     5.790714     6.374916     7.619852

         Juiciness     Ripeness      Acidity      Quality
count  4000.000000  4000.000000  4000.000000  4000.000000
mean      0.512118     0.498277     0.076877     0.501000
std       1.930286     1.874427     2.110270     0.500062
min      -5.961897    -5.864599    -7.010538     0.000000
25%      -0.801286    -0.771677    -1.377424     0.000000
50%       0.534219     0.503445     0.022609     1.000000
75%       1.835976     1.766212     1.510493     1.000000
max       7.364403     7.237837     7.404736     1.000000
```

Fig. 1. Original data dimension and summary of descriptive statistics

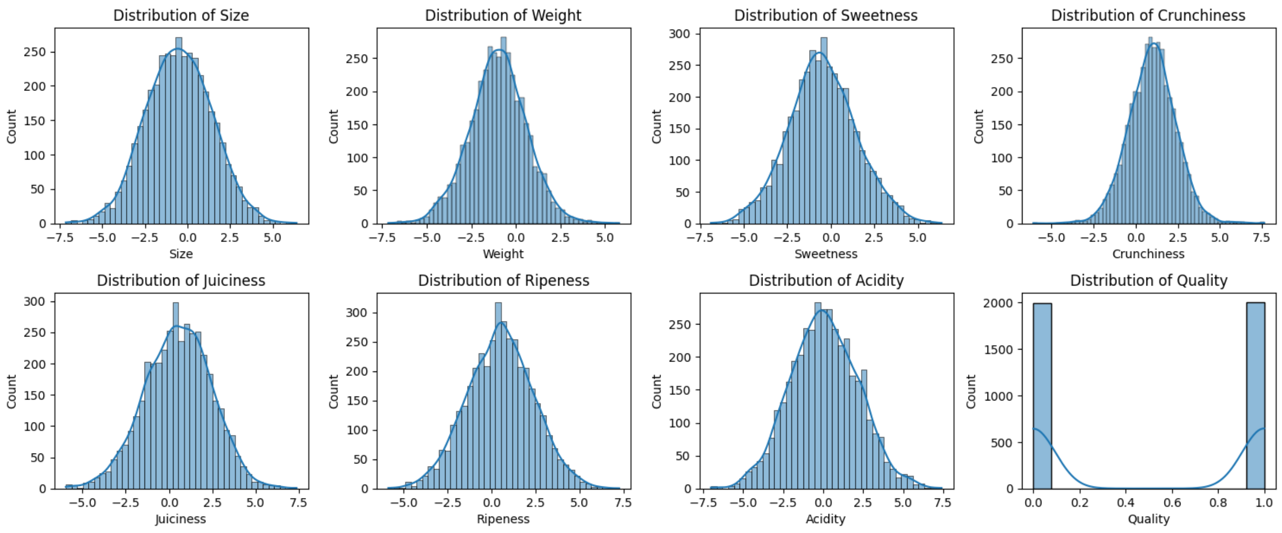


Fig. 2. Histograms for all numerical variables

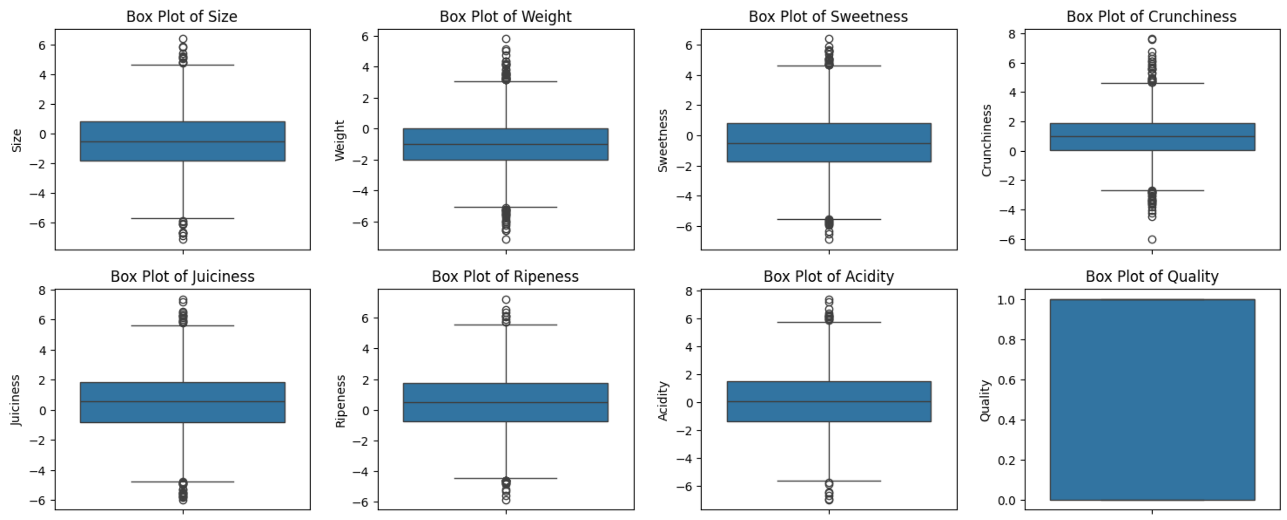


Fig. 3 Box plots for all numerical variables

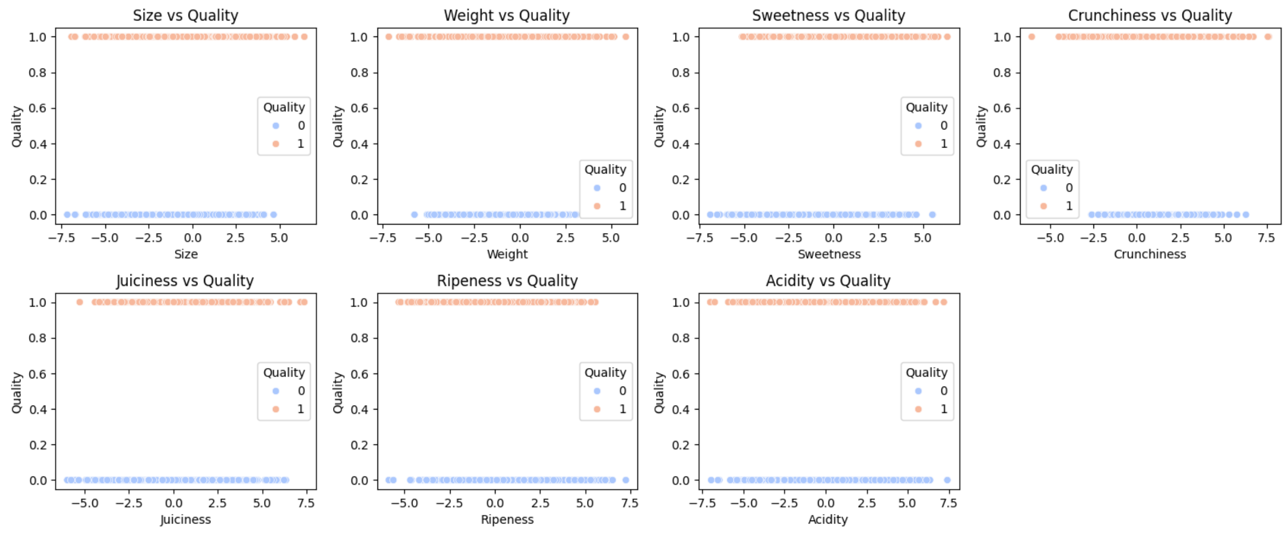


Fig. 4. Scatter plots of 'Quality' vs. numerical features

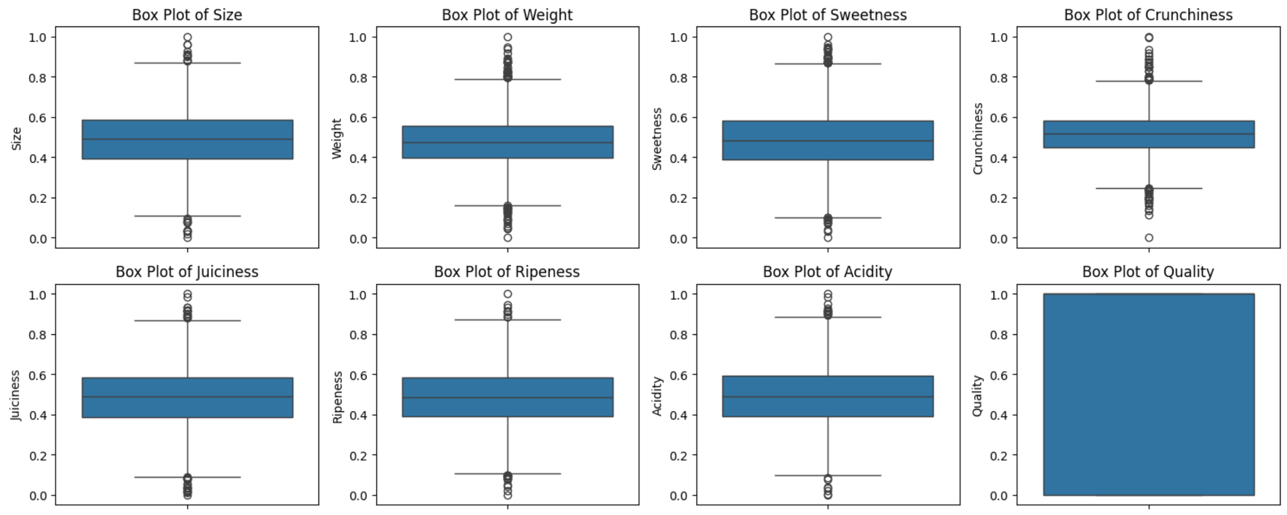


Fig. 4. Box plots for all scaled numerical features