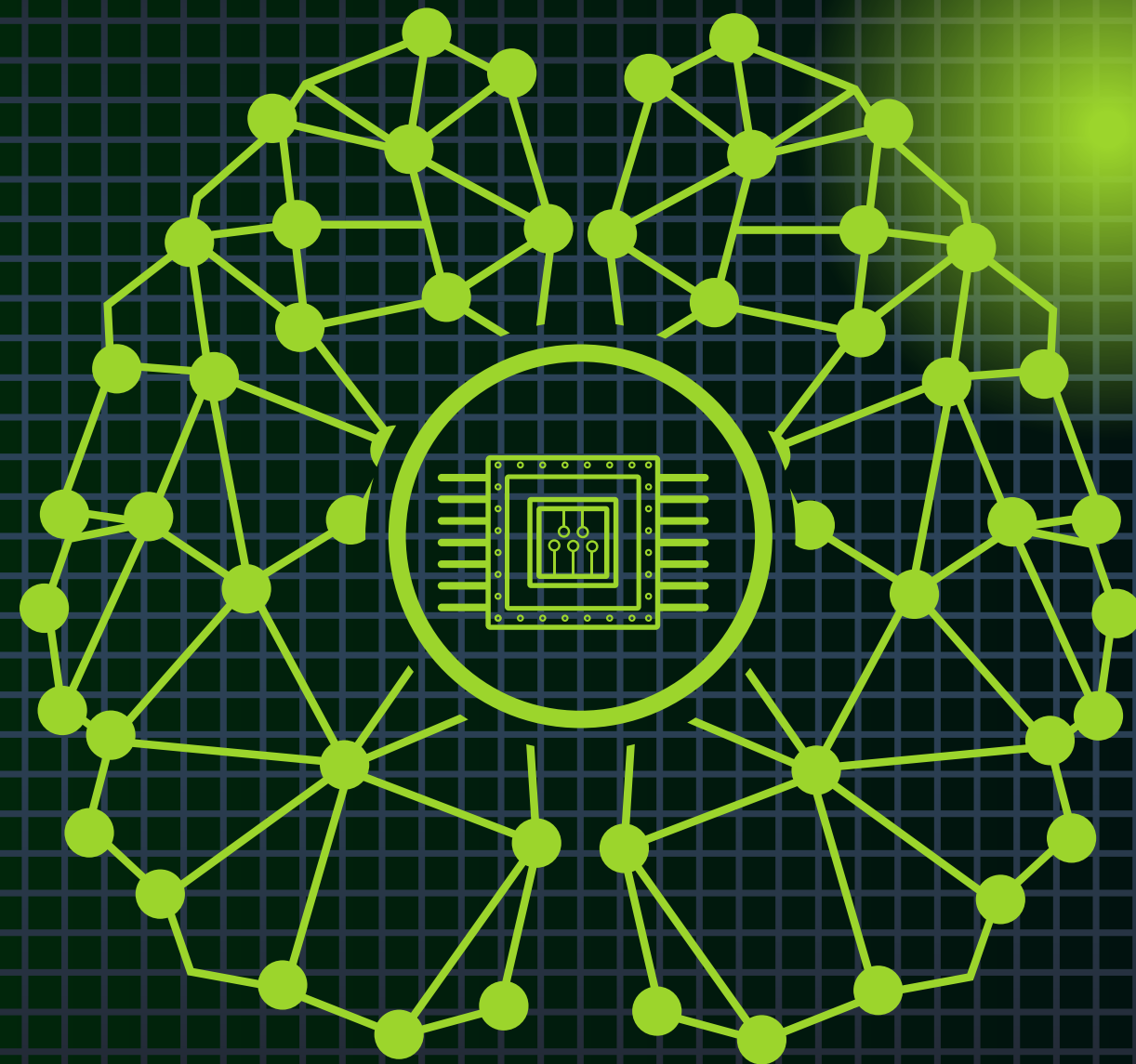
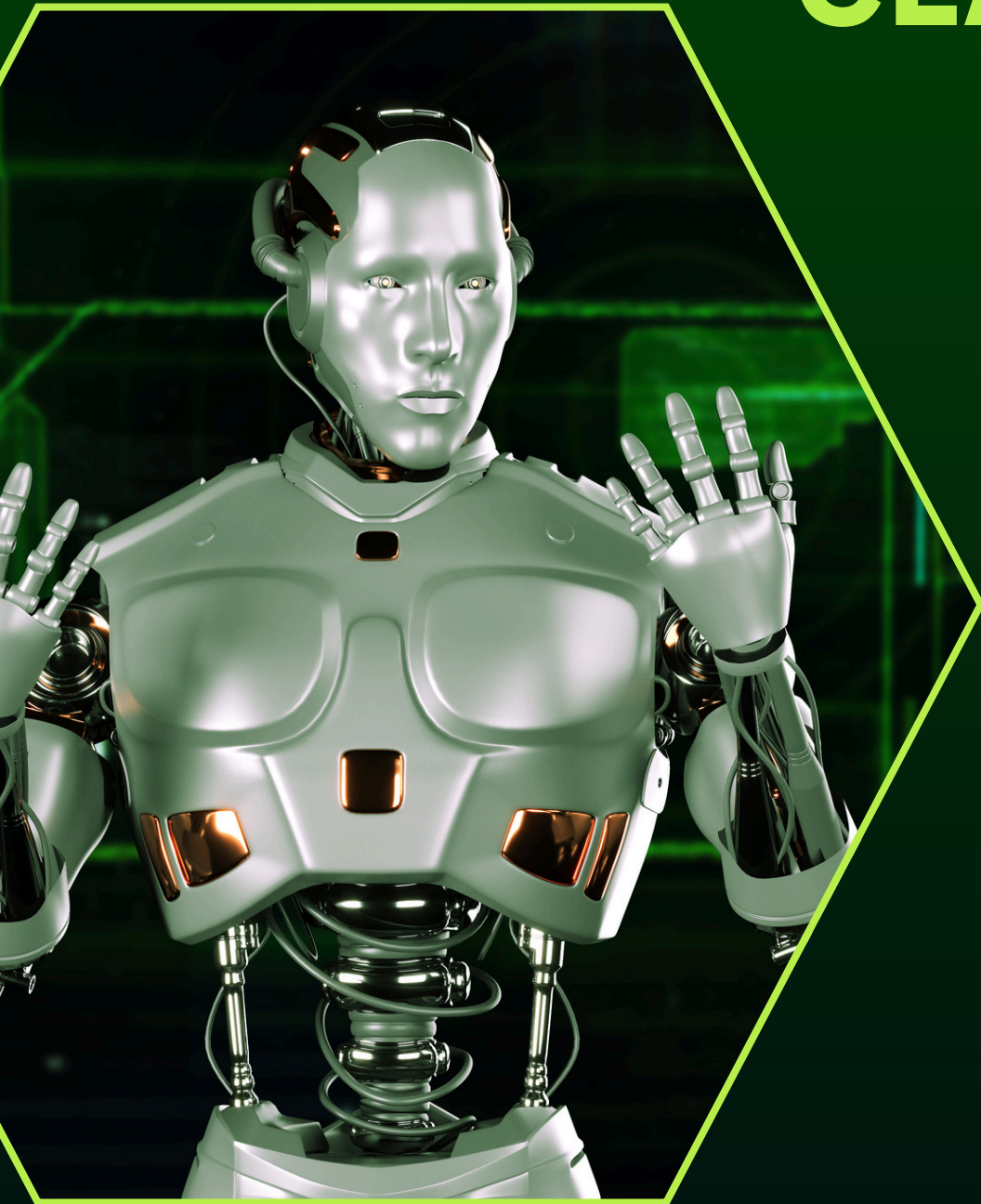


MACHINE LEARNING CLASSIFICATION

BY MITA MIRANTI



WHAT IS MACHINE LEARNING CLASSIFICATION?



Machine Learning Classification is a method in machine learning used to categorize data into predefined classes or categories. This process works by training a model using labeled data, where the model learns patterns from the data and then applies them to predict the class of new, unseen data.

There are three main types of classification:

1. Binary Classification – Data is categorized into two classes, such as spam vs. not spam or pass vs. fail.
2. Multiclass Classification – Data is classified into more than two classes, such as types of diseases or types of flowers.
3. Multilabel Classification – Each data sample can belong to more than one label, such as a movie having multiple genres.

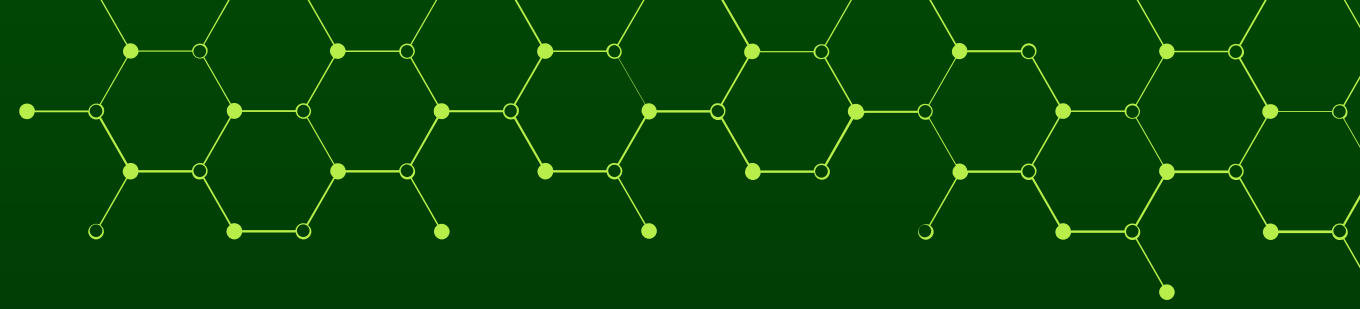
Common algorithms used in classification include Logistic Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), Naïve Bayes, K-Nearest Neighbors (KNN), and Neural Networks. Classification is widely applied in various fields, such as spam detection, facial recognition, sentiment analysis, and disease diagnosis.

TOOLS USED



Matplotlib

READ DATASET



```
import pandas as pd
from sklearn import datasets

# Load wine dataset from scikit-learn
wine = datasets.load_wine()
x = wine.data #Data features
y = wine.target #data target (label)

# Convert data features and targets into a DataFrame
df_x = pd.DataFrame(x, columns=wine.feature_names)
df_y = pd.Series(y, name='wine-class')

#Combine features and targets into one DataFrame
df = pd.concat([df_x, df_y], axis=1)

#display DataFrame
df
```

1. Importing Libraries

- Using pandas to manage data in a tabular format (DataFrame).
- Using datasets from sklearn to load the wine dataset.

2. Loading the Wine Dataset

- Retrieving the wine dataset from Scikit-learn.
- Splitting the data into two parts:
 - Features (x): Contains the chemical characteristics of the wine.
 - Target (y): Contains class labels indicating the type of wine.

3. Converting Data into a DataFrame

- Transforming the feature data into a table with column names corresponding to feature names.
- Converting the target data into a separate column named *wine-class*.

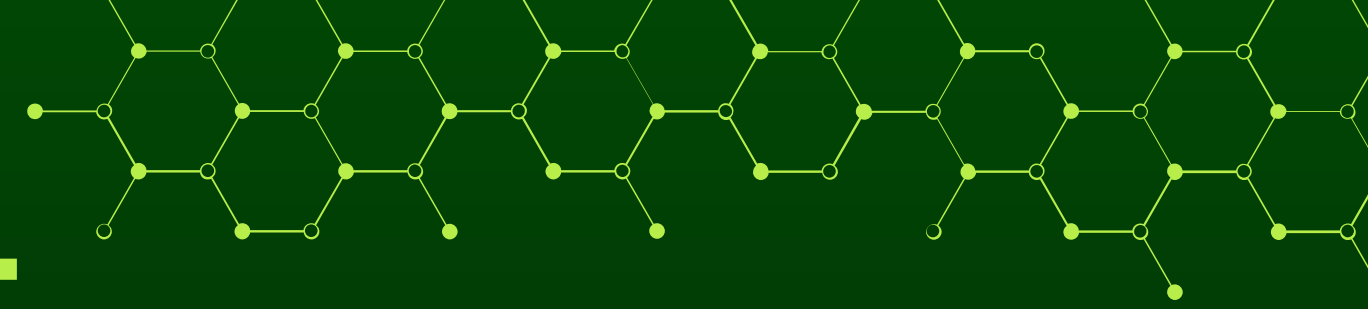
4. Merging Features and Target Data

- Combining the feature and target data into a single table for easier analysis.

5. Displaying the Data

- Presenting the final dataset in a structured table containing all features and wine class labels.

WINE DATASET



	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines	proline	wine-class
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	3.92	1065.0	0
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	3.40	1050.0	0
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	3.17	1185.0	0
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	3.45	1480.0	0
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	2.93	735.0	0
...
173	13.71	5.65	2.45	20.5	95.0	1.68	0.61	0.52	1.06	7.70	0.64	1.74	740.0	2
174	13.40	3.91	2.48	23.0	102.0	1.80	0.75	0.43	1.41	7.30	0.70	1.56	750.0	2
175	13.27	4.28	2.26	20.0	120.0	1.59	0.69	0.43	1.35	10.20	0.59	1.56	835.0	2
176	13.17	2.59	2.37	20.0	120.0	1.65	0.68	0.53	1.46	9.30	0.60	1.62	840.0	2
177	14.13	4.10	2.74	24.5	96.0	2.05	0.76	0.56	1.35	9.20	0.61	1.60	560.0	2

DATA PROCESSING

1. Number of Rows and Columns

- The dataset has 178 entries (rows), with indexes ranging from 0 to 177.
- There are 14 columns in the dataset.

2. Column Names and Data Types

- Each column has a name representing the chemical features of the wine, such as alcohol, malic_acid, ash, magnesium, and so on.
- All columns, except for wine-class, have the float64 type (decimal numbers).
- The wine-class column has the int64 type (integer), used to store the wine class labels.

3. Non-null Data Count

- All columns have *178 non-null values*, meaning there are no missing values in the dataset.

4. Memory Usage

- The dataset uses approximately 19.6 KB of computer memory.

In conclusion, this dataset is clean (no missing values) and contains 13 numerical features and 1 class label for wine classification analysis..

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   alcohol                              178 non-null    float64
1   malic_acid                           178 non-null    float64
2   ash                                  178 non-null    float64
3   alcalinity_of_ash                    178 non-null    float64
4   magnesium                            178 non-null    float64
5   total_phenols                        178 non-null    float64
6   flavanoids                           178 non-null    float64
7   nonflavanoid_phenols                 178 non-null    float64
8   proanthocyanins                      178 non-null    float64
9   color_intensity                      178 non-null    float64
10  hue                                  178 non-null    float64
11  od280/od315_of_diluted_wines         178 non-null    float64
12  proline                              178 non-null    float64
13  wine-class                            178 non-null    int64
dtypes: float64(13), int64(1)
memory usage: 19.6 KB
```

DATA PROCESSING

1. Checking for Missing Values

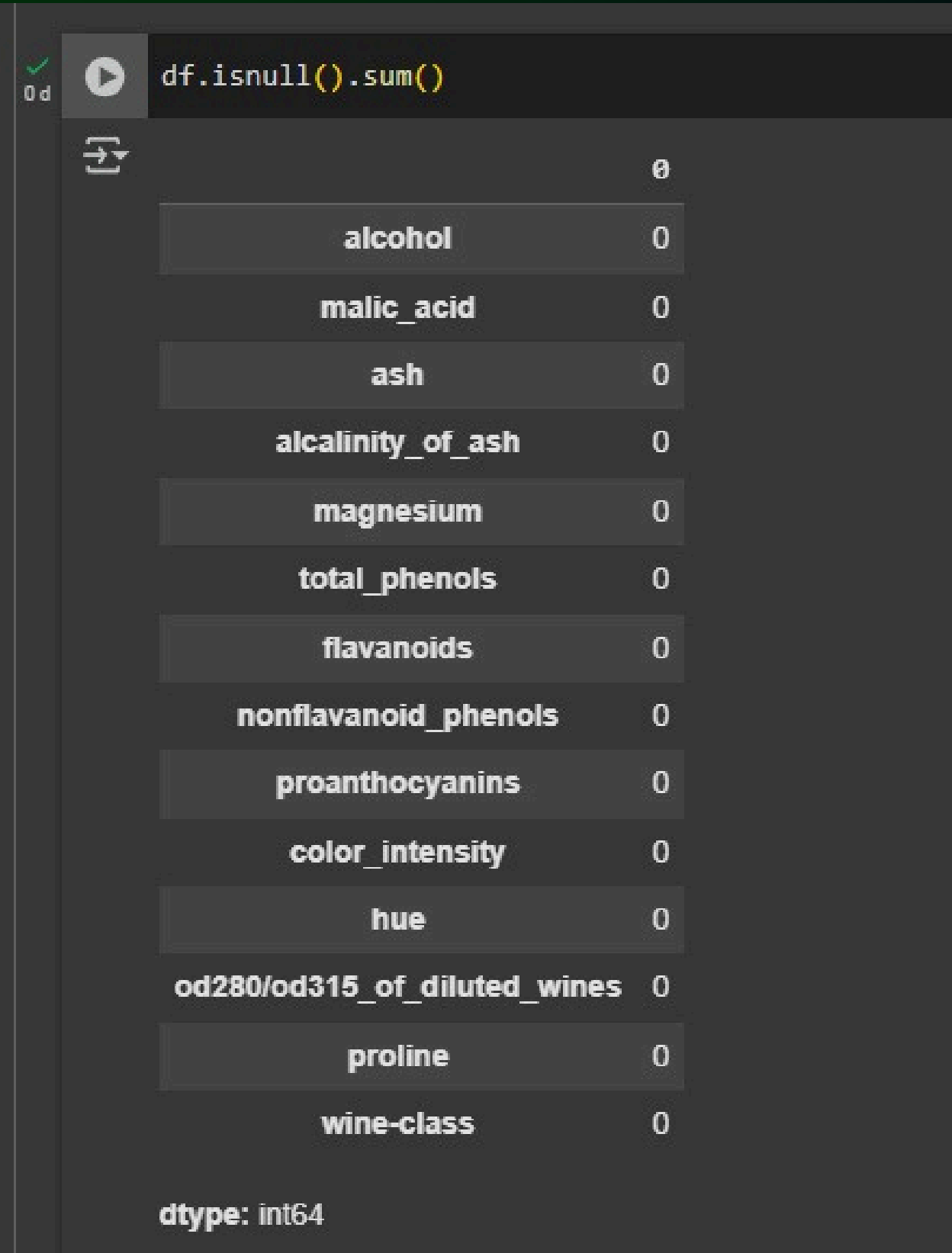
- The table displays various columns such as "alcohol," "malic_acid," "ash," etc.
- Each column has a corresponding number indicating the count of missing values.


2. All Columns Have Zero Missing Values

- Every column has a value of 0, meaning there are no missing data points.
- The dataset is complete, with no NaN (Not a Number) or empty cells.

3. Final Conclusion

- Since there are no missing values, no additional steps like data imputation or cleaning are required.
- The dataset is ready for further analysis or processing.



0 ✓  df.isnull().sum()

	0
alcohol	0
malic_acid	0
ash	0
alcalinity_of_ash	0
magnesium	0
total_phenols	0
flavanoids	0
nonflavanoid_phenols	0
proanthocyanins	0
color_intensity	0
hue	0
od280/od315_of_diluted_wines	0
proline	0
wine-class	0

dtype: int64

DATA PROCESSING

```
✓ 0 d # Mengakses kolom 'wine_class' yang unique  
df['wine-class'].unique()  
→ array([0, 1, 2])
```

- The dataset contains a column called 'wine-class'.
- By checking the unique values in this column, we find that there are three distinct values: 0, 1, and 2.
- These values represent different *wine classes* in the dataset, with each class corresponding to a specific type of wine.

DATA PROCESSING

1. Counting Wine Classes

- The image shows the result of counting the occurrences of each unique value in the "wine-class" column.
- This helps in understanding how many samples belong to each class.

2. Results


- Class 1: 71 samples
- Class 0: 59 samples
- Class 2: 48 samples

3. Meaning

- The dataset consists of three wine classes: 0, 1, and 2.
- Class 1 has the most samples (71), while Class 2 has the least (48).
- This distribution provides insights into the balance of the dataset.

4. Conclusion

- The dataset is slightly imbalanced, with Class 1 having more instances than the other two.
- This information is useful for further analysis, such as classification modeling or statistical evaluations.

```
0 d  # Calculate each wine class  
df['wine-class'].value_counts()
```

	count
wine-class	
1	71
0	59
2	48

dtype: int64

DATA PROCESSING

1. Checking for Outliers Using Boxplot

- The code generates a boxplot to visualize the distribution of four selected columns:

- alcohol
- malic_acid
- ash
- alcalinity_of_ash

- The boxplot helps identify outliers, median values, and data spread.

2. Boxplot Interpretation

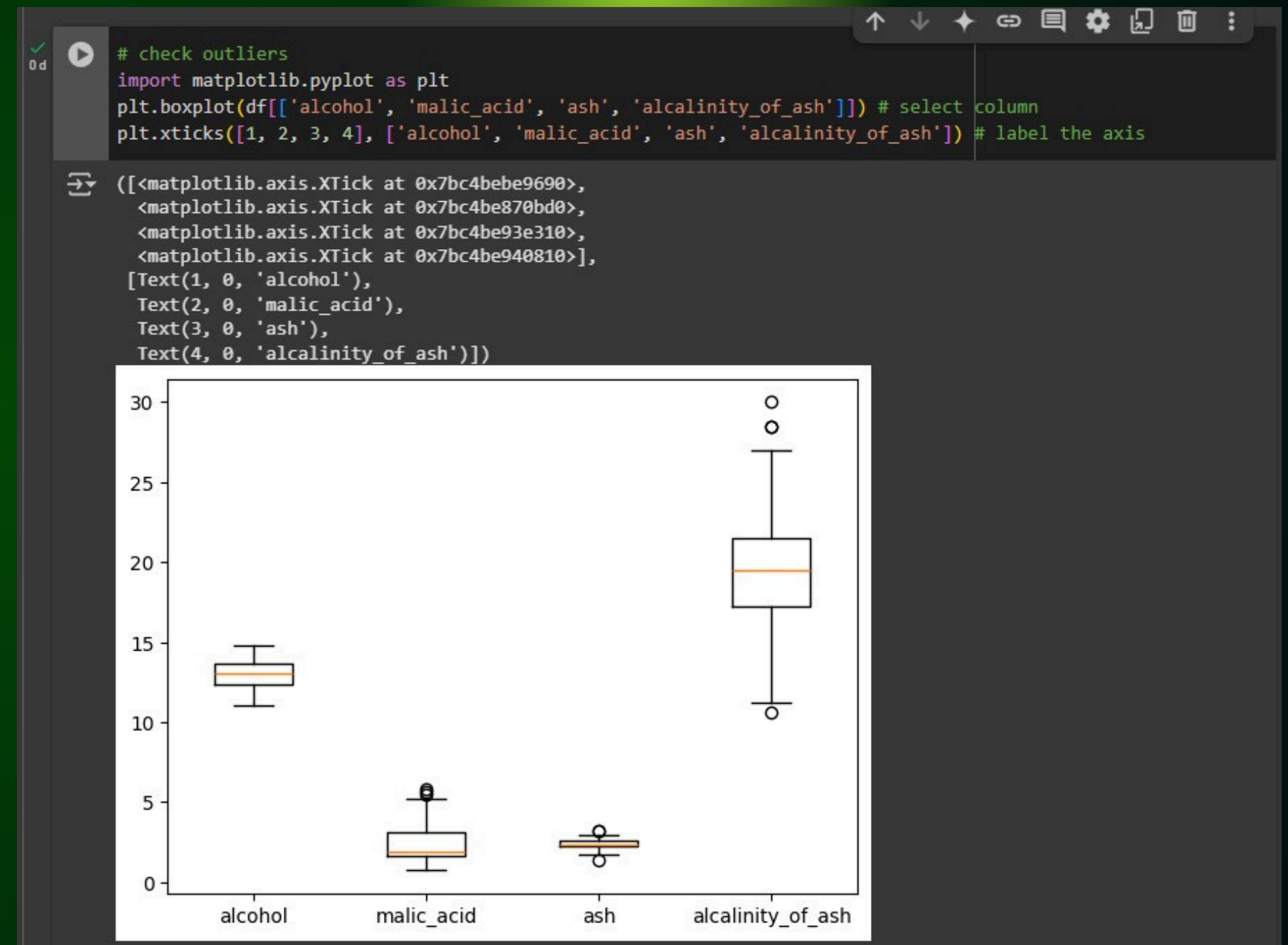
- Box (Interquartile Range - IQR): Represents the middle 50% of the data.
- Line inside the box: Represents the median value.
- Whiskers: Represent the range of data within 1.5 times the IQR.
- Dots (Outliers): Data points beyond the whiskers, which are considered potential outliers.

3. Observations from the Boxplot

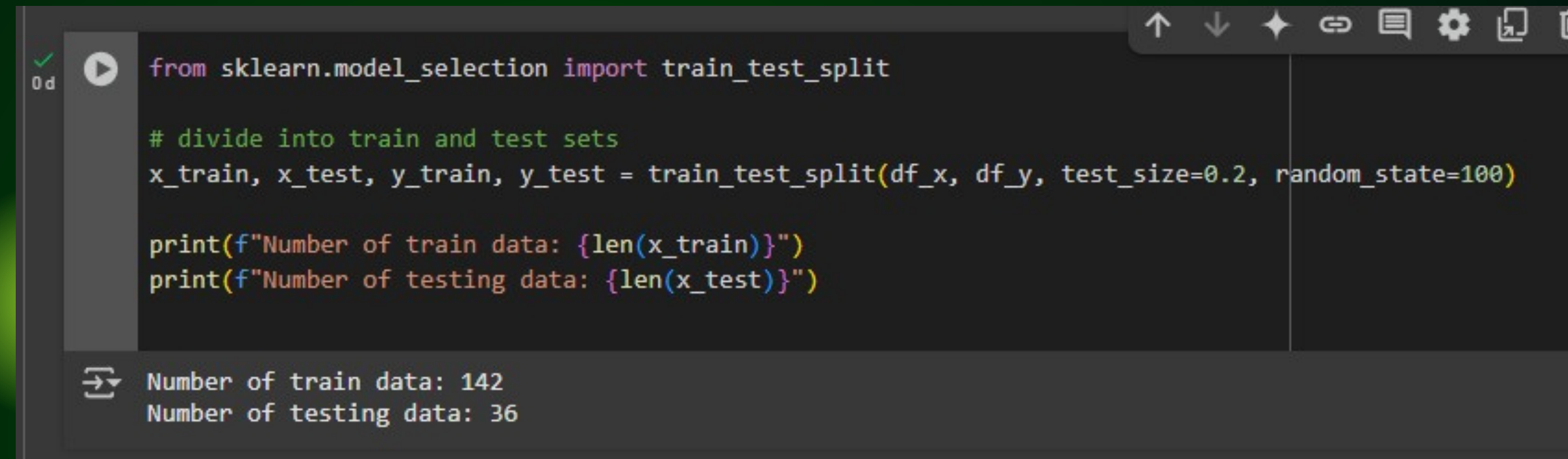
- Malic Acid and Alcalinity of Ash show multiple outliers (dots above whiskers).
- Alcohol and Ash have a more compact distribution with fewer outliers.
- Alcalinity of Ash has a wider spread, indicating a higher variance in data.

4. Conclusion

- Outliers may indicate anomalies or natural variations in data.
- Further analysis is needed to determine whether these outliers should be removed, transformed, or kept.



DATA PROCESSING



```
from sklearn.model_selection import train_test_split

# divide into train and test sets
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.2, random_state=100)

print(f"Number of train data: {len(x_train)}")
print(f"Number of testing data: {len(x_test)}")
```

Number of train data: 142
Number of testing data: 36

1. Purpose: The data is split to train and test a model in machine learning.
2. Split Ratio:
 - 80% of the data is used for training.
 - 20% of the data is used for testing.
3. Data Count:
 - Training data: 142 samples
 - Testing data: 36 samples
4. Randomness: The split is done using a fixed random state (random state = 100), ensuring the same result every time the code runs.
5. Output: The program displays the number of training and testing data on the screen.

This explanation provides a general overview without directly using coding terms.

SPLIT DATA



The future of AI and ML holds immense promise, with continued advancements expected in areas like natural language processing, computer vision, and robotics. As these technologies evolve, they will further transform our lives, leading to new innovations, increased efficiency, and a more interconnected world.

```
from sklearn.tree import DecisionTreeClassifier

# Creating a Decision Tree model
model = DecisionTreeClassifier(random_state = 100)

# Training the model with train
model.fit(x_train, y_train)
```



```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=100)
```


TRAIN THE MODEL

1. Library Used: The DecisionTreeClassifier is imported from scikit-learn.
2. Model Creation: A Decision Tree model is initialized with a fixed random state (random_state = 100) to ensure consistent results.
3. Training the Model: The model is trained using training data (x_train and y_train).
4. Confirmation: The model setup is displayed, confirming the DecisionTreeClassifier with the specified random state.

This explanation provides an overview without directly using coding terms.

```
from sklearn.tree import DecisionTreeClassifier

# Creating a Decision Tree model
model = DecisionTreeClassifier(random_state = 100)

# Training the model with train
model.fit(x_train, y_train)
```

DecisionTreeClassifier ⓘ ?

DecisionTreeClassifier(random_state=100)



PREDICT & EVALUATE

✓
0 d



```
from sklearn.metrics import accuracy_score

# Make predictions on test data
y_pred = model.predict(x_test)

# Calculating model accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy*100:.2f}%")
```



Accuracy: 77.78%

1. Library Used: The `accuracy_score` function is imported from `scikit-learn` to measure model performance.

2. Making Predictions: The trained model is used to predict outcomes for the test data.

3. Calculating Accuracy: The predicted results are compared with the actual test data labels to compute the accuracy.

4. Output: The accuracy of the model is displayed as 77.78%, indicating how well the model performs on the test data.

This explanation provides a general understanding without directly using coding terms.

VISUALIZATION

6. Visualization

```
import matplotlib.pyplot as plt
from sklearn import tree

# decision tree model visualization
plt.figure(figsize = (20,10))
tree.plot_tree(model,
               feature_names = wine.feature_names,
               class_names = wine.target_names,
               filled = True)

plt.show()
```

1. Libraries Used:

- matplotlib.pyplot is used for plotting.
- tree from scikit-learn is used to visualize the Decision Tree.

2. *Figure Size*: The visualization is set to a large size (20x10) for better readability.

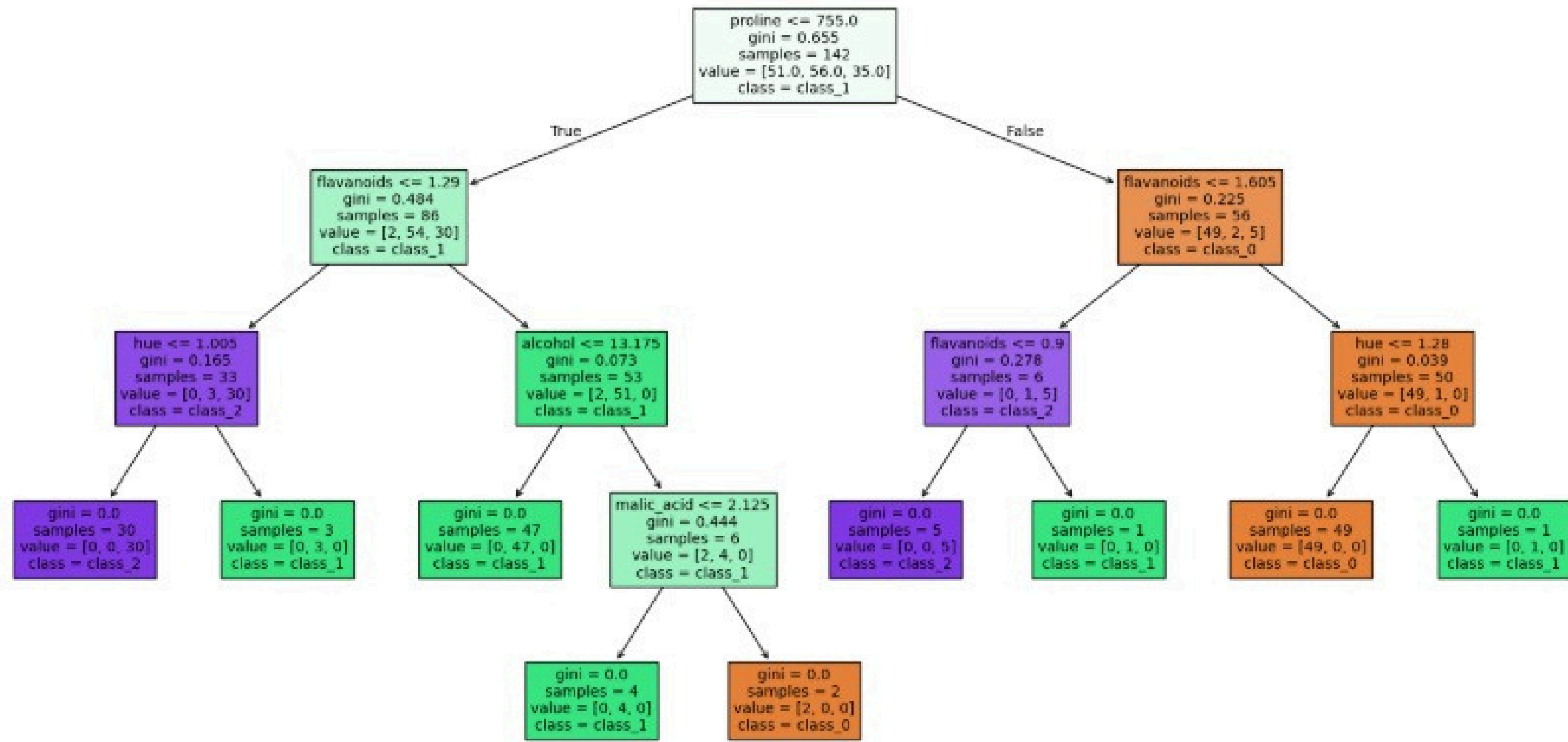
3. Tree Plot:

- The Decision Tree model is displayed using plot_tree().
- Feature names and class names from the dataset are included for clarity.
- The tree nodes are color-filled to enhance visualization.

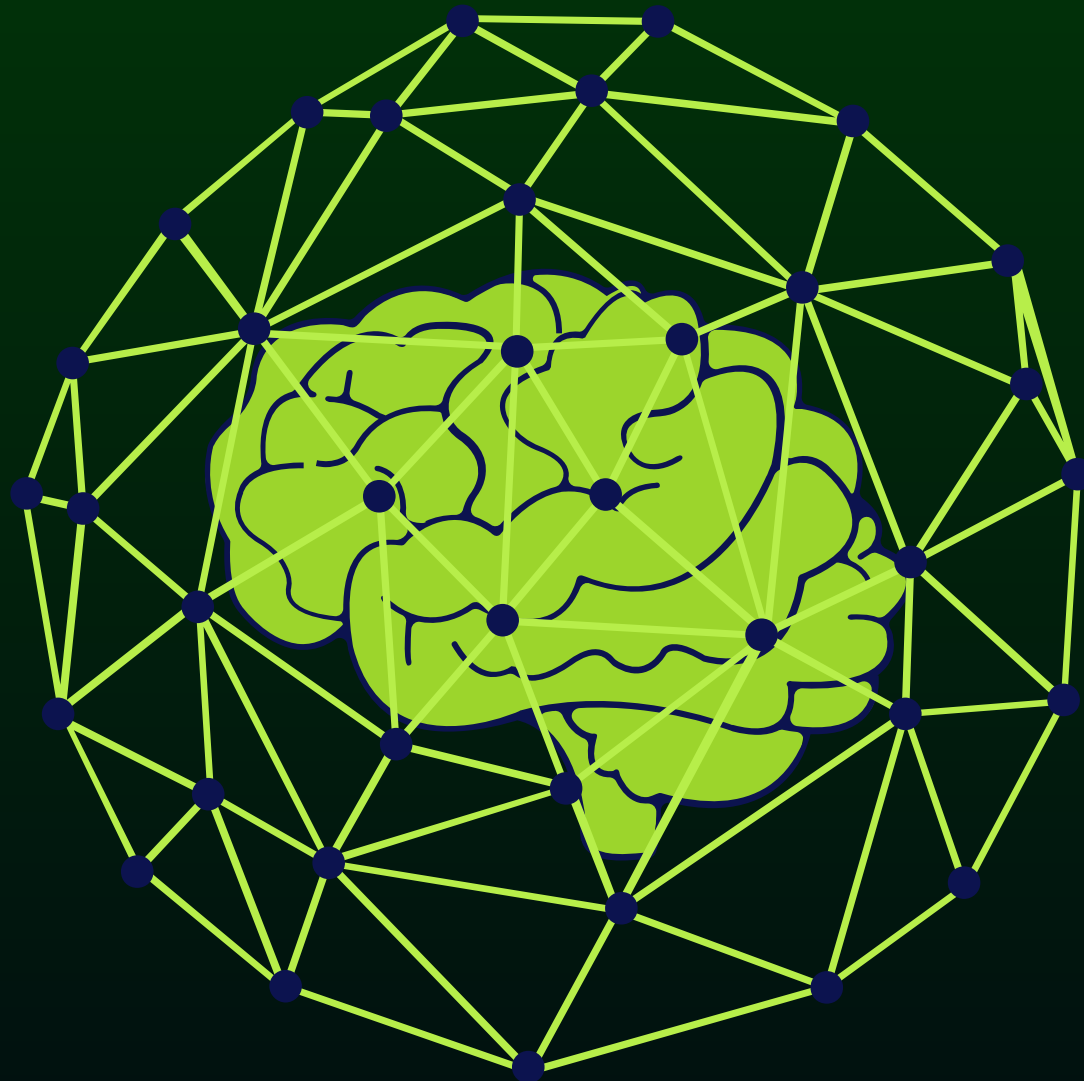
4. Displaying the Tree: The visualization is shown using plt.show().

This explanation provides an overview without directly using coding terms.

VISUALIZATION



CONCLUSION



- The Wine Dataset (178 samples, 13 features, 3 classes) was used for classification.
- Data was clean, with some outliers in Malic Acid and Alcalinity of Ash.
- A Decision Tree Classifier was trained (80% train, 20% test) and achieved 77.78% accuracy.
- The decision tree was visualized, showing key decision points.
- Improvements: Tuning hyperparameters, trying other models (Random Forest, SVM), and feature selection.

THANK YOU!

