MACHINE LEARNING CLASSIFICATION

By Ayumi Syahraya

DSF 35.0 - Data Science

WHAT IS MACHINE LEARNING CLASSIFICATION?

Machine Learning Classification is a supervised learning technique used to categorize data into predefined classes based on its features. The model learns from labeled data and can then predict the class of new, unseen data.

One popular application of this technique is on the Wine Dataset, where the goal is to classify different types of wine based on their chemical properties. So, here I will provide an example of applying classification on the Wine Dataset, which is commonly used in machine learning to test various classification algorithms. The Wine Dataset from the UCI Machine Learning Repository contains 13 features that describe the chemical composition of each wine sample and classifies the data into three different types of wine.

TOOLS USED

pandas

matplotlib

scikit learn

READ DATASET

1. Read Dataset

[14]
```python
import pandas as pd
from sklearn import datasets

# Memuat dataset Wine dari scikit-learn
wine = datasets.load_wine()
x = wine.data  # Data fitur
y = wine.target  # Data target (label)

# Mengonversi data fitur dan target menjadi DataFrame
df_x = pd.DataFrame(x, columns=wine.feature_names)
df_y = pd.Series(y, name='wine_class')

# Menggabungkan fitur dan target dalam satu DataFrame
df = pd.concat([df_x, df_y], axis=1)

# Menampilkan DataFrame
df
```

- Imports necessary libraries: It imports pandas for data manipulation and datasets from scikit-learn to load the Wine dataset.
- Loads the Wine dataset: The datasets.load_wine() function loads the Wine dataset, which contains data on the chemical composition of wines.
- Extracts features and target: wine.data contains the features (chemical properties), and wine.target contains the target labels (wine classes).
- Converts data into a DataFrame: It converts the features into a DataFrame (df_x) and the target into a Series (df_y), assigning appropriate column names for the features.
- Combines the features and target: The features (df_x) and target (df_y) are combined into one DataFrame (df).
- Displays the DataFrame: Finally, it prints the combined DataFrame df, showing both features and their corresponding target (wine classes).

WINE DATASET

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_diluted_wines
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
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
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
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
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
...	...	...	...	...	...	...	...	...	...	...	...	...
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
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
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
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
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

178 rows × 14 columns

DATA PREPROCESSING

The df.info() function in pandas provides a concise summary of the DataFrame. It includes the following information:

- Number of entries: Displays the total number of rows in the DataFrame.
- Column names: Lists the names of all the columns.
- Non-null count: Shows how many non-null (non-missing) values there are in each column.
- Data types: Displays the data type of each column (e.g., float64, int64).
- Memory usage: Indicates the amount of memory the DataFrame occupies.

2. Data Pre-Processing

```
15] 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 PREPROCESSING

The df.isnull().sum() function in pandas is used to check for missing (null) values in the DataFrame. Here's what it does:

- df.isnull(): This checks each value in the DataFrame and returns True for any missing value (NaN) and False for non-missing values.
- .sum(): This sums up the True values (which represent missing values) for each column.

The result is the number of missing values in each column. If a column has no missing values, the result will be 0 for that column.

[16]
```python
# Mengecek missing value
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

DATA PREPROCESSING

The code df['wine_class'].unique() is used to access the unique values in the 'wine_class' column of the DataFrame df.

- df['wine_class']: This selects the 'wine_class' column from the DataFrame, which contains the target labels (i.e., the classes of wine).
- .unique(): This function returns the unique values in that column, which means it will list all the distinct wine classes present in the 'wine_class' column.

```
[19] # Mengakses kolom 'wine_class' yang unique
     df['wine_class'].unique()

array([0, 1, 2])
```

DATA PREPROCESSING

The code df['wine_class'].value_counts() is used to count the number of occurrences of each unique value in the 'wine_class' column.

- df['wine_class']: This selects the 'wine_class' column, which contains the target labels (wine types).
- .value_counts(): This function returns a count of how many times each unique value appears in the 'wine_class' column.

[20]
```
# Menghitung masing-masing wine_class

df['wine_class'].value_counts()
```

wine_class	count
1	71
0	59
2	48

dtype: int64

DATA PREPROCESSING

```
[21] # Statistik Deskriptif
     df.describe()
```

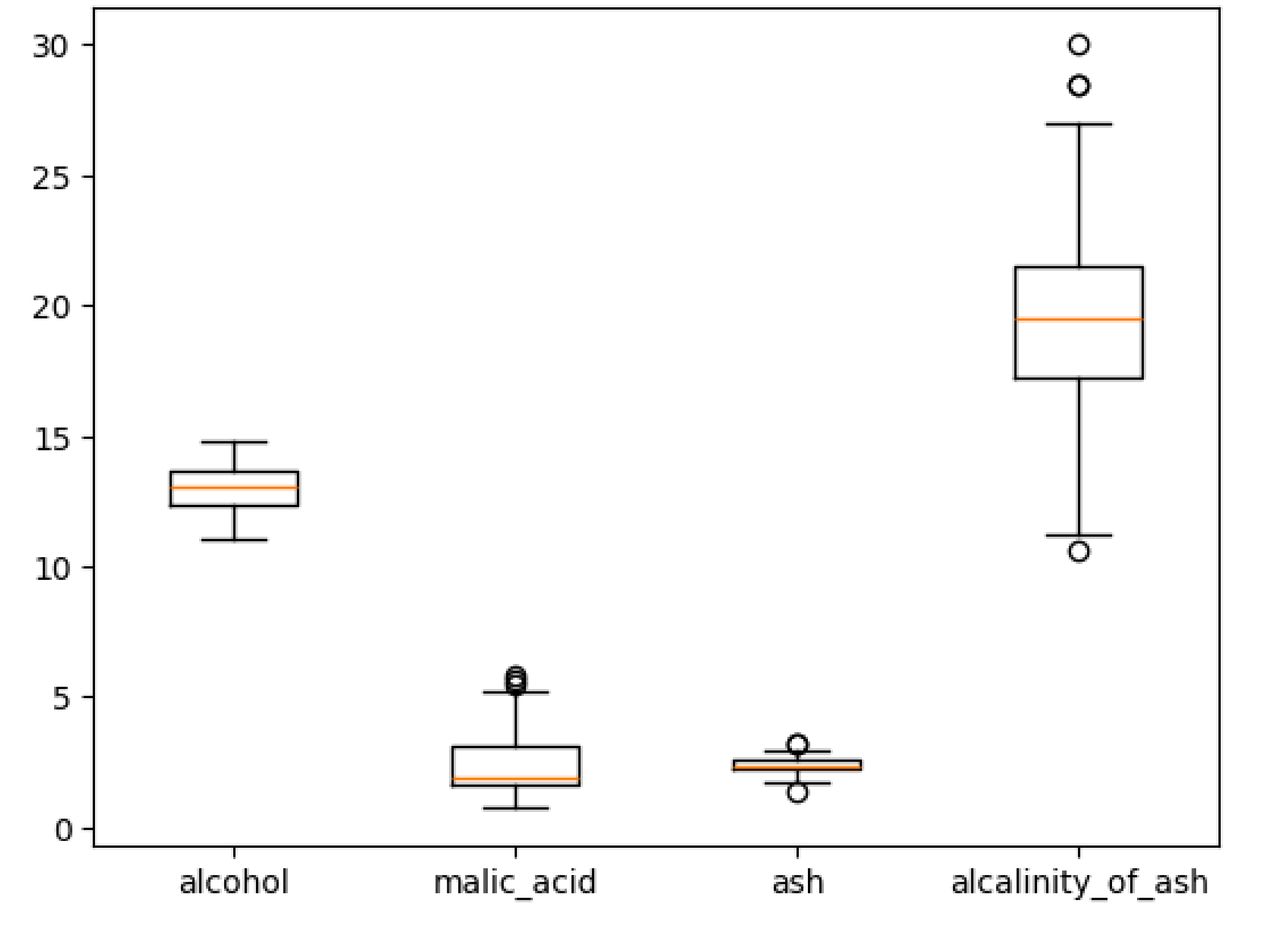
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	p
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	

The code df.describe() is used to generate descriptive statistics for the numerical columns in the DataFrame df.
Here's what it provides:

- Count: The number of non-null entries in each column.
- Mean: The average value of each numerical column.
- Standard Deviation (std): A measure of the spread or variability of the values.
- Min: The minimum value in each numerical column.
- 25th percentile (25%): The value below which 25% of the data falls.
- 50th percentile (50%): The median value (middle value) of each numerical column.
- 75th percentile (75%): The value below which 75% of the data falls.
- Max: The maximum value in each numerical column.

DATA PREPROCESSING

```
[23] # Mengecek outlier
     import matplotlib.pyplot as plt
     plt.boxplot(df[['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash']]) # Memilih kolom
     plt.xticks([1, 2, 3, 4], ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash']) # Memberi label pada axis
     plt.show()
```



This code creates a box plot to check for outliers in the columns 'alcohol', 'malic_acid', 'ash', and 'alcalinity_of_ash' in the df DataFrame:

- plt.boxplot(df[['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash']]): Generates the box plot for the selected columns to visualize the distribution and potential outliers.
- plt.xticks([1, 2, 3, 4], ['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash']): Labels the x-axis with the column names.
- plt.show(): Displays the plot.

The box plot helps identify outliers by showing data points outside the whiskers, which represent values significantly different from the rest of the data.

SPLIT DATA

[25]

```
#Memisahkan variabel prediktor dan variabel target


from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split

# Membagi data menjadi train dan test
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
```

This code visualizes the decision tree:

- plt.figure(figsize=(20, 10)): Sets the figure size for the plot.
- tree.plot_tree(...): Visualizes the decision tree:
- model: The trained decision tree.
- feature_names = wine.feature_names: Labels for the features.
- class_names = wine.target_names: Labels for the target classes.
- filled = True: Colors the nodes for better readability.
- plt.show(): Displays the tree plot.
- It shows how the model splits data and classifies wine types.

TRAIN THE MODEL

4. Train The Model

```
[26] from sklearn.tree import DecisionTreeClassifier

     # Membuat model Decision Tree
     model = DecisionTreeClassifier(random_state = 100)

     # Melatih model dengan data train
     model.fit(x_train, y_train)
```

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

The code provided creates and trains a Decision Tree model using the training data. Here's a breakdown of what each part does:

- from sklearn.tree import DecisionTreeClassifier: This imports the DecisionTreeClassifier class from scikit-learn, which is used to build a decision tree model for classification tasks.
- model = DecisionTreeClassifier(random_state=100): This initializes the decision tree classifier with a random_state of 100. Setting the random_state ensures that the results are reproducible by controlling the randomness during training (e.g., random splits of data).
- model.fit(x_train, y_train): This trains the decision tree model on the training data:
- x_train: The features from the training set.
- y_train: The target labels from the training set.

PREDICT & EVALUATE

5. Predict & Evaluate

```
[27] from sklearn.metrics import accuracy_score

     # Melakukan prediksi pada data test
     y_pred = model.predict(x_test)

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

```
Accuracy: 77.78%
```

The code provided is used to make predictions with the trained decision tree model and evaluate its accuracy on the test data. Here's a breakdown:

- from sklearn.metrics import accuracy_score: This imports the accuracy_score function from scikit-learn, which is used to calculate the accuracy of the model by comparing the predicted values with the actual values.
- y_pred = model.predict(x_test): This uses the trained decision tree model to make predictions (y_pred) on the test features (x_test).
- accuracy = accuracy_score(y_test, y_pred): This calculates the accuracy by comparing the predicted labels (y_pred) with the actual labels (y_test) from the test set. The accuracy score is the proportion of correctly predicted labels.
- print(f"Accuracy: {accuracy*100:.2f}%"): This prints the accuracy of the model as a percentage, formatted to two decimal places.

VISUALIZATION

This code visualizes the trained decision tree model using matplotlib and scikit-learn's plot_tree function. Here's a breakdown of what each part does:

1. import matplotlib.pyplot as plt: This imports matplotlib.pyplot, which is used for plotting graphs.
2. from sklearn import tree: This imports the tree module from scikit-learn, which contains functions for visualizing decision trees.
3. plt.figure(figsize = (20,10)): This creates a figure with a specified size (20 inches by 10 inches) for better visibility of the tree.
4. tree.plot_tree(...): This function visualizes the decision tree:
   - model: The trained decision tree model.
   - feature_names = wine.feature_names: Specifies the names of the features (chemical properties of wine) for labeling the tree nodes.
   - class_names = wine.target_names: Specifies the names of the wine classes (target labels) for labeling the leaf nodes.
   - filled = True: Fills the nodes with colors to represent the class labels, making it easier to interpret.
5. plt.show(): This displays the plot of the decision tree.

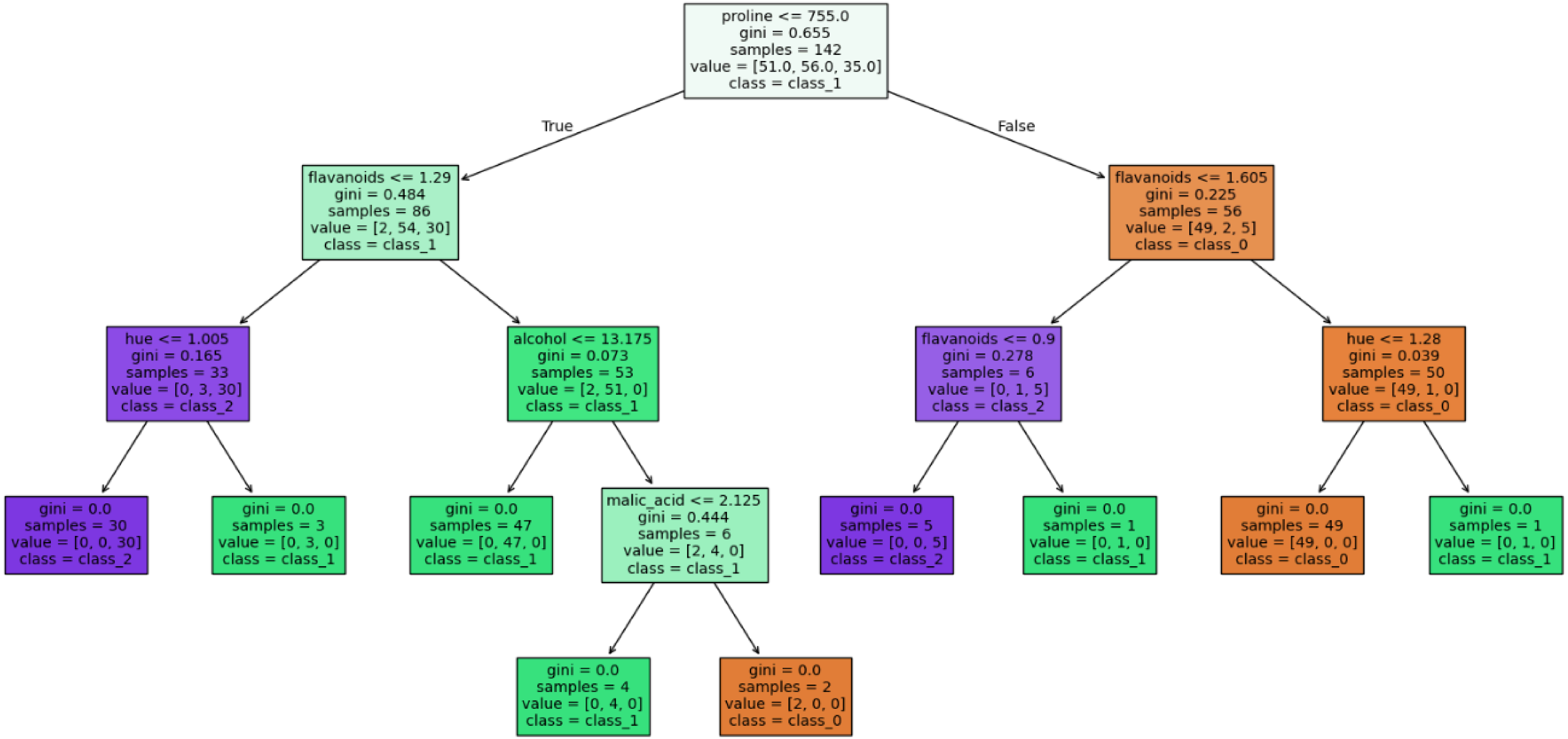
6. Visualization

2d [29]

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

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

VISUALIZATION



CONCLUSION

In applying Decision Tree classification to the Wine dataset, we successfully built and trained a model to classify wine types based on their chemical properties. The model's accuracy shows how well it predicts the wine type on test data. By using decision tree visualization, we can observe how the model makes classification decisions based on specific chemical features. This model also provides an easily interpretable explanation of how classification decisions are made, thanks to the clear and understandable tree structure.

THANK YOU