PORTOFOLIO

RIAN ALFA ZALIKA

JANUARI/2025

DATA SCIENCE

Classification in machine learning is a predictive modeling process by which machine learning models use classification algorithms to predict the correct label for input data.

As AI models learn to analyze and classify data in their training datasets, they become more proficient at identifying various data types, discovering trends and making more accurate predictions.

CODE EXPLANATION

```
[ ]  # Import the pandas library for data manipulation
     import pandas as pd

     # Import the datasets module from scikit-learn to load built-in datasets
     from sklearn import datasets

     # Load the wine dataset from scikit-learn
     wine = datasets.load_wine()

     # Store the feature data (independent variables) in the variable x
     x = wine.data  # `x` is a NumPy array containing the features for each sample

     # Store the target labels (dependent variables) in the variable y
     y = wine.target  # `y` is a NumPy array containing the class label for each sample
```

1. pandas is imported as pd for data manipulation.
2. The datasets module from scikit-learn is imported to access built-in datasets.
3. The load_wine() function loads the Wine dataset, which contains chemical analysis data of wines grown in the same region but derived from different cultivars (classes).
4. x stores the feature data (independent variables), which is a NumPy array containing various attributes for each wine sample.
5. y stores the target labels, which represent the class labels of the wines.

Code Explanation

```
# Create a DataFrame from the feature data (x) with column names from wine.feature_names
df_x = pd.DataFrame(x, columns=wine.feature_names)

# Create a Series from the target data (y) and name it 'target'
df_y = pd.Series(y, name='target')

# Combine the feature DataFrame (df_x) and the target Series (df_y) into a single DataFrame
df = pd.concat([df_x, df_y], axis=1)

# Display the first 5 rows of the DataFrame to inspect the data
print(df.head(10))
```

1. Converts the feature data (x) into a pandas DataFrame. The column names are set using wine.feature_names, which contains the names of the features.
2. Converts the target labels (y) into a pandas Series.The Series is named 'target' to indicate that it represents the class labels.
4. Merges the feature DataFrame (df_x) and the target Series (df_y) into a single DataFrame. axis=1 ensures that they are combined as columns.
5. Prints the first 10 rows of the DataFrame to inspect the data.

Print Result :

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	\
0	14.23	1.71	2.43	15.6	127.0	2.80	
1	13.20	1.78	2.14	11.2	100.0	2.65	
2	13.16	2.36	2.67	18.6	101.0	2.80	
3	14.37	1.95	2.50	16.8	113.0	3.85	
4	13.24	2.59	2.87	21.0	118.0	2.80	
5	14.20	1.76	2.45	15.2	112.0	3.27	
6	14.39	1.87	2.45	14.6	96.0	2.50	
7	14.06	2.15	2.61	17.6	121.0	2.60	
8	14.83	1.64	2.17	14.0	97.0	2.80	
9	13.86	1.35	2.27	16.0	98.0	2.98	

	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	\
0	3.06	0.28	2.29	5.64	1.04	
1	2.76	0.26	1.28	4.38	1.05	
2	3.24	0.30	2.81	5.68	1.03	
3	3.49	0.24	2.18	7.80	0.86	
4	2.69	0.39	1.82	4.32	1.04	
5	3.39	0.34	1.97	6.75	1.05	
6	2.52	0.30	1.98	5.25	1.02	
7	2.51	0.31	1.25	5.05	1.06	
8	2.98	0.29	1.98	5.20	1.08	
9	3.15	0.22	1.85	7.22	1.01	

	od280/od315_of_diluted_wines	proline	target
0	3.92	1065.0	0
1	3.40	1050.0	0
2	3.17	1185.0	0
3	3.45	1480.0	0
4	2.93	735.0	0
5	2.85	1450.0	0
6	3.58	1290.0	0
7	3.58	1295.0	0
8	2.85	1045.0	0

Code Explanation

df.info()

- Shows that there are 178 entries (rows) and 14 columns.
- Lists column names, non-null counts, and data types.
- Most columns are of type float64, except for 'target', which is int64.

df['target'].unique()

- Retrieves the unique class labels in the 'target' column.
- The output array([0, 1, 2]) indicates that the dataset contains three distinct wine classes (0, 1, and 2).

```
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  target                        178 non-null    int64  
dtypes: float64(13), int64(1)
memory usage: 19.6 KB
```

```
[ ] df['target'].unique()
```

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

The Code

```
[ ] df.describe()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315_of_di
count	178.000000	178.000000	178.000000	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	1.590899	5.058090	0.957449	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	0.572359	2.318286	0.228572	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	0.410000	1.280000	0.480000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	1.250000	3.220000	0.782500	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	1.555000	4.690000	0.965000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	1.950000	6.200000	1.120000	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	3.580000	13.000000	1.710000	

```python
# Import the train_test_split function from scikit-learn for splitting the dataset
from sklearn.model_selection import train_test_split

# Split the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(
    df_x,       # Feature data (independent variables)
    df_y,       # Target data (dependent variable)
    test_size=0.2,  # Proportion of the dataset to include in the test set (20%)
    random_state=42  # Ensures reproducibility by setting a random seed
)
```

Code Explanation

1. Dataset Summary (df.describe()):

- The first part of the image shows the summary statistics of a dataset using the df.describe() function.
- The dataset appears to contain numeric features related to wine composition (e.g., alcohol content, malic acid, magnesium, flavonoids, etc.).

2. Train-Test Split using train_test_split:

- The second part of the image shows a Python script that splits the dataset into training and testing sets using train_test_split from sklearn.model_selection.
- Imports the train_test_split function from scikit-learn.
- Splits the dataset into training (x_train, y_train) and testing (x_test, y_test) subsets.
- df_x contains the independent variables (features).
- df_y contains the dependent variable (target).
- test_size=0.2 specifies that 20% of the dataset is reserved for testing.
- random_state=42 ensures that the split is reproducible.

The Code

```python
# Import the DecisionTreeClassifier from scikit-learn
from sklearn.tree import DecisionTreeClassifier

# Create an instance of the DecisionTreeClassifier
model = DecisionTreeClassifier(random_state=42)  # random_state ensures reproducibility

# Train the decision tree model using the training dataset
model.fit(x_train, y_train)  # x_train: features, y_t
```

```
DecisionTreeClassifier(random_state=42)
```

```python
# Import the accuracy_score function from scikit-learn for evaluating model accuracy
from sklearn.metrics import accuracy_score

# Use the trained model to make predictions on the test set
y_pred = model.predict(x_test)  # Predict the target labels for the test set features

# Calculate the accuracy of the model by comparing predicted and actual target labels
accuracy = accuracy_score(y_test, y_pred)  # Returns the proportion of correct predictions

# Print the classification report (accuracy of the model)
print("Classification Report")
print(f"Accuracy: {accuracy * 100:.2f}%")  # Display accuracy as a percentage with 2 decimal places
```

```
Classification Report
Accuracy: 94.44%
```

Code Explanation

1. Importing and Training a Decision Tree Classifier

- Imports the DecisionTreeClassifier from sklearn.tree.
- Initializes the classifier with a fixed random_state=42 to ensure reproducibility.
- Trains the decision tree model using x_train (features) and y_train (target labels).

2. Making Predictions and Evaluating Accuracy

- Imports accuracy_score from sklearn.metrics to evaluate the model's accuracy.
- Uses the trained model to predict target values for the test set (x_test).
- Computes accuracy by comparing predicted labels (y_pred) with actual labels (y_test).
- Prints the classification accuracy in percentage format.

3. Output

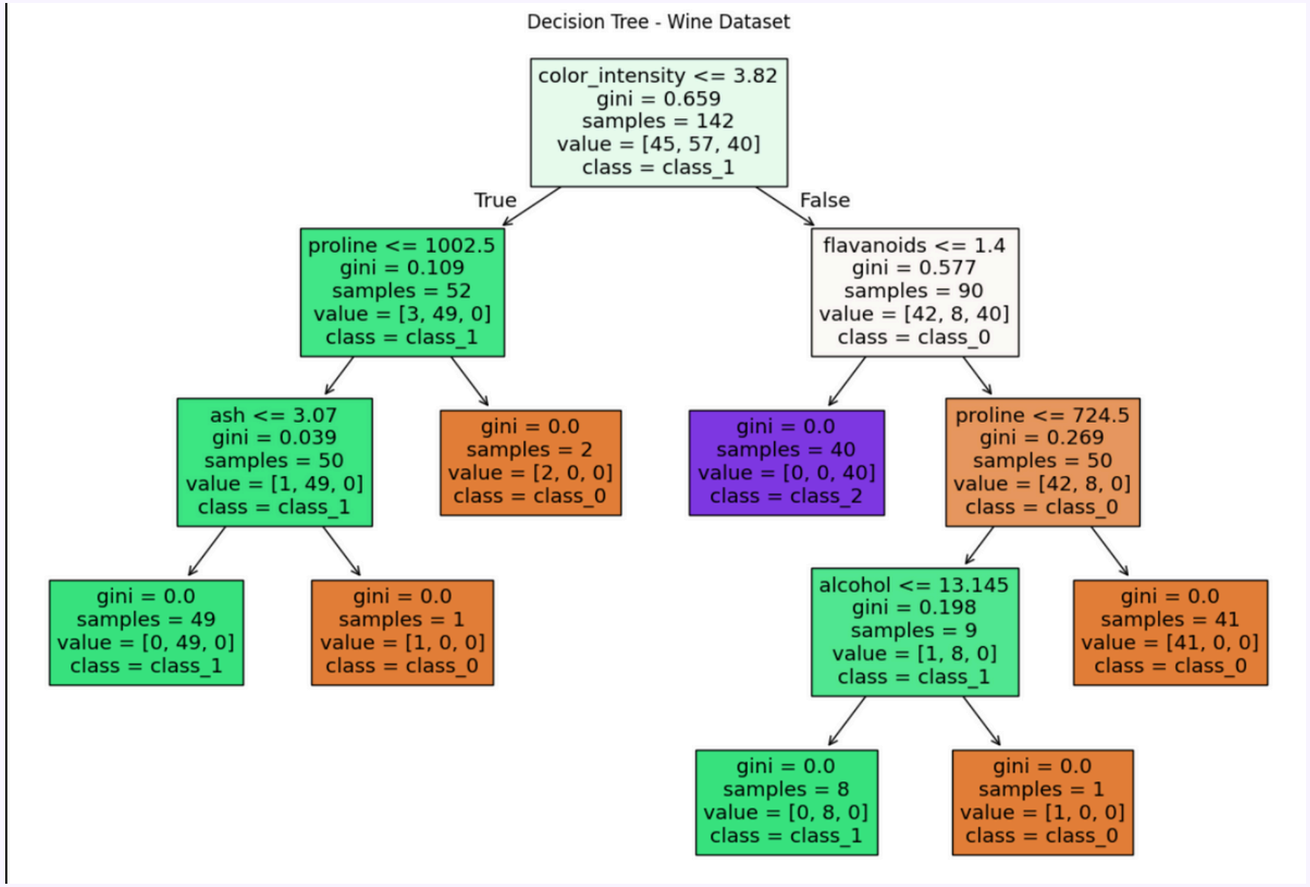
- A Decision Tree model is trained using DecisionTreeClassifier.
- The model is tested on unseen data (x_test), and predictions are made.
- The accuracy of the model is calculated and displayed.
- The model achieves an accuracy of 94.44%, indicating strong performance.

Code Explanation

```python
# Import matplotlib for plotting
import matplotlib.pyplot as plt

# Import the tree module from scikit-learn for visualizing decision trees
from sklearn import tree

# Create a figure for the plot with specified dimensions
plt.figure(figsize=(15, 10))  # Width: 15, Height: 10

# Plot the decision tree
tree.plot_tree(
    model,  # The trained decision tree model
    feature_names=wine.feature_names,  # Correct spelling: Feature names from the dataset
    class_names=wine.target_names,  # Class names corresponding to the target labels
    filled=True  # Fill nodes with colors representing different classes
)

# Add a title to the plot
plt.title("Decision Tree - Wine Dataset")

# Display the plot
plt.show()
```

1. Import Necessary Libraries
2. Create a Figure for the Plot
3. Plot the Decision Tree
4. Add a Title to the Plot
5. Display the Plot

Print Result :



Thank You