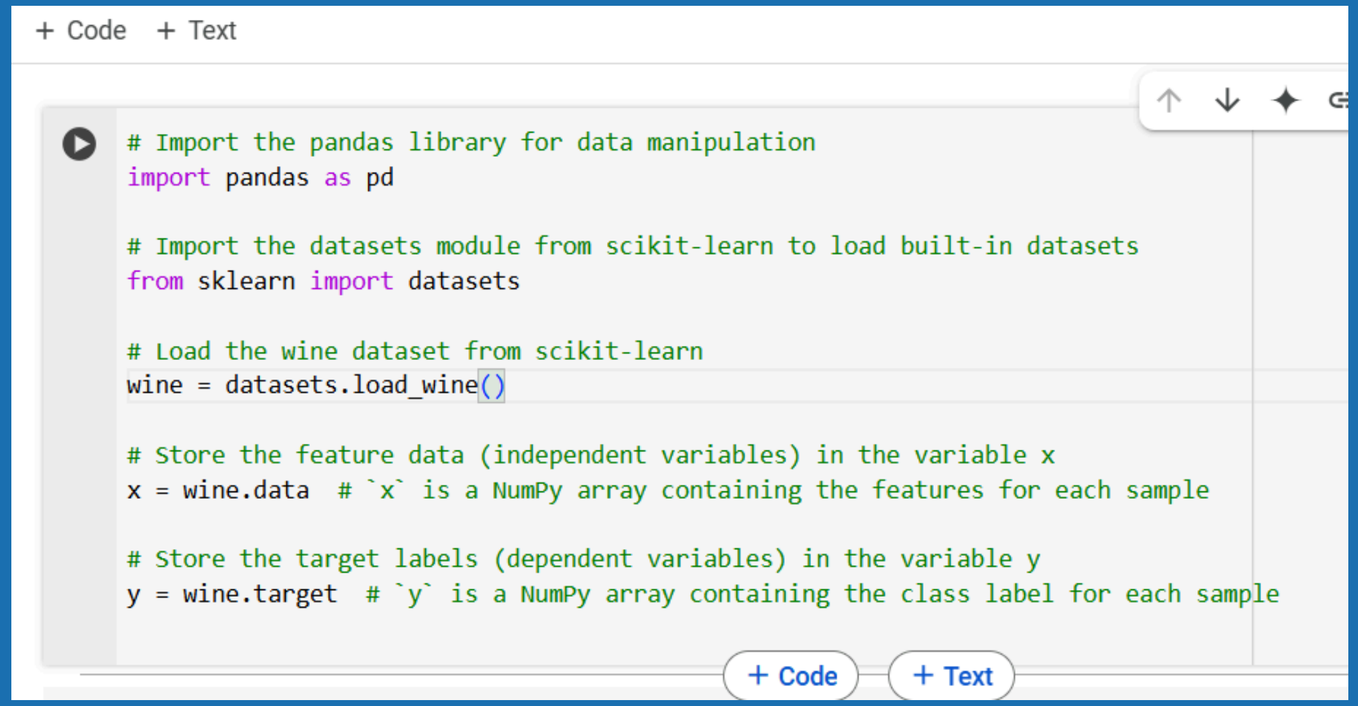
MACHINE LEARNING CLASIFICATION

Tool Used Google Colabs,pyhton & ETC

CODE EXPLANATION



1.IMPORTING LIBRARIES :

- PANDAS for data manipulation.
- DATASETS from scikit-learn to load built-in datasets.

2.LOADING THE WINE DATASETS

- wine = datasets.load_wine() : Loads the wine dataset, which contains 13 chemical features used to classify wines into 3 categories.

3.STORING DATA

- x = wine.data: Contains feature data (independent variables).
- y = wine.target: Contains class labels (dependent variable).

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.CREATE A DATAFRAME FOR FEATURES:

- df_x = pd.DataFrame(x, columns=wine.feature_names):Converts the feature data (x) into a pandas DataFrame with column names taken from wine.feature_names.

2.CREATE A SERIES FOR THE TARGET:

- df_y = pd.Series(y, name='target'): Converts the target data (y) into a pandas Series and names it target.

3.COMBINE FEATURES AND TARGET:

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

4.DISPLAY THE FIRST 10 ROWS:

- print(df.head(10)): Displays the first 10 rows of the combined DataFrame to inspect the data.

PRINT RESULTS :

	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
9	3.55	1045.0	0

CODE EXPLANATION

```
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])
```

1.DF.INFO() :

- The DataFrame has 178 rows and 14 columns (13 numerical features as float64 and 1 target column as int64).
- No missing values (all columns have 178 non-null entries).
- Memory usage: 19.6 KB.

2.DF['TARGET'].UNIQUE()

- isplays the unique values in the target column:
- [0, 1, 2]: Indicates there are three wine classes in the dataset.

CODE EXPLANATION

```
df.describe()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	17
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	

```
[9] # 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
)
```

1.DATA SUMMARY (DF.DESCRIBE()):

- Provides key statistics for each feature (count, mean, std, min, max, percentiles).
- Helps understand the data distribution and variability.

2.DATASET SPLITTING (TRAIN_TEST_SPLIT)

- Splits data into:Training set (80%): For model training.Testing set (20%): For evaluating model performance.
- Parameters:test_size=0.2: 20% for testing. random_state=42: Ensures reproducibility.

CODE EXPLANATION

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

```python
[11] # 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%
```

1.MODEL CREATION AND TRAINING:

- Imported DecisionTreeClassifier from scikit-learn.
- Created a model with random_state=42 for reproducibility.
- Trained the model using the training data (x_train, y_train).

2.MODEL EVALUATION:

- Used the trained model to predict labels for the test set (y_pred = model.predict(x_test)).
- Calculated accuracy with accuracy_score, comparing predictions (y_pred) to actual labels (y_test).
- Achieved an accuracy of 94.44%.

CODE EXPLANATION

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

```python
[11] # 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%
```

1.MODEL CREATION AND TRAINING:

- Imported DecisionTreeClassifier from scikit-learn.
- Created a model with random_state=42 for reproducibility.
- Trained the model using the training data (x_train, y_train).

2.MODEL EVALUATION:

- Used the trained model to predict labels for the test set (y_pred = model.predict(x_test)).
- Calculated accuracy with accuracy_score, comparing predictions (y_pred) to actual labels (y_test).
- Achieved an accuracy of 94.44%.

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 LIBRARIES:

- matplotlib.pyplot for plotting.
- tree module from sklearn to handle decision trees.

2.FIGURE SETUP:

- plt.figure(figsize=(15, 10)): Configures the plot dimensions (Width: 15, Height: 10).

3.PLOT THE DECISION TREE:

- tree.plot_tree: Visualizes the trained decision tree.
- Parameters:
- model: The trained decision tree model.
- feature_names: Names of the input features.
- class_names: Labels for target classes.
- filled=True: Colors the nodes to represent different classes.

4.ADD TITLE & DISPLAY:

- plt.title: Adds a title to the plot.
- plt.show: Renders the plot.


THANKS FOR WATCHING