Install and Import Libraries

```
!pip install ucimlrepo --quiet
```

Start coding or generate with AI.

```
import pandas as pd
import numpy as np

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

from ucimlrepo import fetch_ucirepo
```

Set Random Seed

Make sure that you use this on every single call to the `train_test_split` function

```
RANDOM_SEED = 123456789
```

Fetch and One-Hot Encode Datasets

```
# fetch dataset
bank_marketing = fetch_ucirepo(id=222)

# data (as pandas dataframes)
bank_marketing_features = bank_marketing.data.features
bank_marketing_features_onehot = pd.get_dummies(bank_marketing_features)
bank_marketing_labels = bank_marketing.data.targets

# metadata
print(bank_marketing.metadata)

# variable information
print(bank_marketing.variables)
```

```
{'uci_id': 222, 'name': 'Bank Marketing', 'repository_url': 'https://archive.ics.uci.edu/dataset/222/bank+marketing', 'd
           name     role         type      demographic  \
0           age  Feature      Integer              Age
1           job  Feature  Categorical       Occupation
2       marital  Feature  Categorical   Marital Status
3     education  Feature  Categorical  Education Level
4       default  Feature       Binary             None
5       balance  Feature      Integer             None
6       housing  Feature       Binary             None
7          loan  Feature       Binary             None
8       contact  Feature  Categorical             None
9   day_of_week  Feature         Date             None
10        month  Feature         Date             None
11     duration  Feature      Integer             None
12     campaign  Feature      Integer             None
13        pdays  Feature      Integer             None
14     previous  Feature      Integer             None
15     poutcome  Feature  Categorical             None
16            y   Target       Binary             None

                                          description  units missing_values
0                                                None   None             no
1   type of job (categorical: 'admin.','blue-colla...   None             no
2   marital status (categorical: 'divorced','marri...   None             no
3   (categorical: 'basic.4y','basic.6y','basic.9y'...   None             no
4                              has credit in default?   None             no
5                                average yearly balance  euros            no
6                                   has housing loan?   None             no
7                                  has personal loan?   None             no
8   contact communication type (categorical: 'cell...   None            yes
9                           last contact day of the week   None          no
10  last contact month of year (categorical: 'jan'...   None             no
11   last contact duration, in seconds (numeric). ...   None             no
12  number of contacts performed during this campa...   None             no
13  number of days that passed by after the client...   None            yes
14  number of contacts performed before this campa...   None             no
15  outcome of the previous marketing campaign (ca...   None            yes
```

```
16         has the client subscribed a term deposit?   None             no
```

```
cc_default = fetch_ucirepo(id=350)

# data (as pandas dataframes)
cc_default_features = cc_default.data.features
cc_default_features_onehot = pd.get_dummies(cc_default_features, columns=['X2', 'X3', 'X4', 'X6', 'X7', 'X8', 'X9', 'X10', '
cc_default_labels = cc_default.data.targets

# metadata
print(cc_default.metadata)

# variable information
print(cc_default.variables)
```

```
{'uci_id': 350, 'name': 'Default of Credit Card Clients', 'repository_url': 'https://archive.ics.uci.edu/dataset/350/def
   name     role     type      demographic                 description units  \
0    ID       ID  Integer             None                        None  None
1    X1  Feature  Integer             None                   LIMIT_BAL  None
2    X2  Feature  Integer              Sex                         SEX  None
3    X3  Feature  Integer  Education Level                   EDUCATION  None
4    X4  Feature  Integer   Marital Status                    MARRIAGE  None
5    X5  Feature  Integer              Age                         AGE  None
6    X6  Feature  Integer             None                       PAY_0  None
7    X7  Feature  Integer             None                       PAY_2  None
8    X8  Feature  Integer             None                       PAY_3  None
9    X9  Feature  Integer             None                       PAY_4  None
10  X10  Feature  Integer             None                       PAY_5  None
11  X11  Feature  Integer             None                       PAY_6  None
12  X12  Feature  Integer             None                   BILL_AMT1  None
13  X13  Feature  Integer             None                   BILL_AMT2  None
14  X14  Feature  Integer             None                   BILL_AMT3  None
15  X15  Feature  Integer             None                   BILL_AMT4  None
16  X16  Feature  Integer             None                   BILL_AMT5  None
17  X17  Feature  Integer             None                   BILL_AMT6  None
18  X18  Feature  Integer             None                    PAY_AMT1  None
19  X19  Feature  Integer             None                    PAY_AMT2  None
20  X20  Feature  Integer             None                    PAY_AMT3  None
21  X21  Feature  Integer             None                    PAY_AMT4  None
22  X22  Feature  Integer             None                    PAY_AMT5  None
23  X23  Feature  Integer             None                    PAY_AMT6  None
24    Y   Target   Binary             None  default payment next month  None

   missing_values
0              no
1              no
2              no
3              no
4              no
5              no
6              no
7              no
8              no
9              no
10             no
11             no
12             no
13             no
14             no
15             no
16             no
17             no
18             no
19             no
20             no
21             no
22             no
23             no
24             no
```

Q1

Construct Default Decision Tree for Bank Marketing

```
# Change the code below to construct, train, and evaluate the default decision tree on the bank marketing dataset
# Make sure you report accuracy on the training set as well!
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
# Split the data into training and testing sets (80-20 split)
bank_marketing_features_train, bank_marketing_features_test, bank_marketing_labels_train, bank_marketing_labels_test = trair
    bank_marketing_features_onehot,  # Preprocessed features
    bank_marketing_labels,          # Target labels
    test_size=0.2,                  # 20% for testing
    random_state=42                 # Reproducibility
)

# Initialize the Decision Tree Classifier with default parameters
bank_marketing_dt_default = DecisionTreeClassifier(random_state=42)

# Train the decision tree on the training data
bank_marketing_dt_default.fit(bank_marketing_features_train, bank_marketing_labels_train)

# Make predictions on both the training and testing sets
bank_marketing_train_predictions = bank_marketing_dt_default.predict(bank_marketing_features_train)
bank_marketing_test_predictions = bank_marketing_dt_default.predict(bank_marketing_features_test)

# Calculate accuracy for both training and testing sets
train_accuracy = accuracy_score(bank_marketing_labels_train, bank_marketing_train_predictions)
test_accuracy = accuracy_score(bank_marketing_labels_test, bank_marketing_test_predictions)

# Output the training and testing accuracy
print(f"Training Accuracy: {train_accuracy*100:.2f}%")
print(f"Testing Accuracy: {test_accuracy*100:.2f}%")
```

```
Training Accuracy: 100.00%
Testing Accuracy: 87.67%
```

Q2

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets (80-20 split)
bank_marketing_features_train, bank_marketing_features_test, bank_marketing_labels_train, bank_marketing_labels_test = trair
    bank_marketing_features_onehot,  # Preprocessed features
    bank_marketing_labels,          # Target labels
    test_size=0.2,                  # 20% for testing
    random_state=42                 # Reproducibility
)

# Maximum depths to limit to
max_depths = [1, 2, 3, 5, 7, 10, 15, 20]

# Initialize a list to store results
results = []

# Loop through different maximum depths
for depth in max_depths:
    # Initialize a decision tree classifier with the specified max depth
    dt = DecisionTreeClassifier(max_depth=depth, random_state=42)

    # Train the model
    dt.fit(bank_marketing_features_train, bank_marketing_labels_train)

    # Make predictions
    train_predictions = dt.predict(bank_marketing_features_train)
    test_predictions = dt.predict(bank_marketing_features_test)

    # Calculate accuracies
    train_accuracy = accuracy_score(bank_marketing_labels_train, train_predictions)
    test_accuracy = accuracy_score(bank_marketing_labels_test, test_predictions)

    # Append results
    results.append({"Depth": depth, "Training Accuracy": train_accuracy, "Testing Accuracy": test_accuracy})

# Convert results to a DataFrame for better visualization
results_df = pd.DataFrame(results)

# Print the table
print(results_df)

# Optional: Save the table to a CSV
results_df.to_csv("decision_tree_depth_analysis.csv", index=False)
```

```
   Depth  Training Accuracy  Testing Accuracy
0      1           0.883931          0.879354
1      2           0.896925          0.893177
2      3           0.902372          0.896495
3      5           0.906741          0.897158
4      7           0.912077          0.897822
5     10           0.925265          0.897269
6     15           0.958665          0.894062
7     20           0.985291          0.884552
```

Finish Q2 like Q1, but limiting maximum depth

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Split the dataset (reuse for consistency)
bank_marketing_features_train, bank_marketing_features_test, bank_marketing_labels_train, bank_marketing_labels_test = trair
    bank_marketing_features_onehot,  # Features with one-hot encoding
    bank_marketing_labels,          # Target labels
    test_size=0.2,                  # 80% train, 20% test split
    random_state=42                 # Ensure reproducibility
)

# Maximum depths to evaluate
max_depths = [1, 2, 3, 5, 7, 10, 15, 20]

# Results storage
results = []

# Loop through each depth
for depth in max_depths:
    # Create and train a decision tree with the specified max depth
    dt = DecisionTreeClassifier(max_depth=depth, random_state=42)
    dt.fit(bank_marketing_features_train, bank_marketing_labels_train)

    # Evaluate performance
    train_accuracy = accuracy_score(bank_marketing_labels_train, dt.predict(bank_marketing_features_train))
    test_accuracy = accuracy_score(bank_marketing_labels_test, dt.predict(bank_marketing_features_test))

    # Store the depth and accuracies
    results.append({"Depth": depth, "Training Accuracy": train_accuracy, "Testing Accuracy": test_accuracy})

# Convert results into a DataFrame
results_df = pd.DataFrame(results)

# Print results table
print(results_df)

# Save results as CSV for further use (optional)
results_df.to_csv("decision_tree_max_depth_analysis.csv", index=False)
```


```
   Depth  Training Accuracy  Testing Accuracy
0      1           0.883931          0.879354
1      2           0.896925          0.893177
2      3           0.902372          0.896495
3      5           0.906741          0.897158
4      7           0.912077          0.897822
5     10           0.925265          0.897269
6     15           0.958665          0.894062
7     20           0.985291          0.884552
```


Q3

Construct Default Decision Tree for Credit Card Default Dataset

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets (80-20 split)
cc_default_features_train, cc_default_features_test, cc_default_labels_train, cc_default_labels_test = train_test_split(
    cc_default_features_onehot,  # Preprocessed features with one-hot encoding
```

```
    cc_default_labels,          # Target labels
    test_size=0.2,              # 20% for testing
    random_state=42             # Reproducibility
)

# Create a Decision Tree Classifier with default parameters
dt_default = DecisionTreeClassifier(random_state=42)

# Train the model on the training set
dt_default.fit(cc_default_features_train, cc_default_labels_train)

# Make predictions on both the training and test sets
train_predictions = dt_default.predict(cc_default_features_train)
test_predictions = dt_default.predict(cc_default_features_test)

# Calculate accuracy for training and test sets
train_accuracy = accuracy_score(cc_default_labels_train, train_predictions)
test_accuracy = accuracy_score(cc_default_labels_test, test_predictions)

# Report accuracies
print(f"Training Accuracy: {train_accuracy * 100:.2f}%")
print(f"Testing Accuracy: {test_accuracy * 100:.2f}%")
```

```
Training Accuracy: 99.95%
Testing Accuracy: 72.57%
```

Q4

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets (80-20 split)
cc_default_features_train, cc_default_features_test, cc_default_labels_train, cc_default_labels_test = train_test_split(
    cc_default_features_onehot,  # Preprocessed features with one-hot encoding
    cc_default_labels,          # Target labels
    test_size=0.2,              # 20% for testing
    random_state=42             # Reproducibility
)

# List of maximum depths to evaluate
max_depths = [1, 2, 3, 5, 7, 10, 15, 20]

# Initialize a list to store results
results = []

# Loop through each depth
for depth in max_depths:
    # Create a Decision Tree Classifier with the current max depth
    dt = DecisionTreeClassifier(max_depth=depth, random_state=42)

    # Train the model on the training set
    dt.fit(cc_default_features_train, cc_default_labels_train)

    # Make predictions on the training and test sets
    train_predictions = dt.predict(cc_default_features_train)
    test_predictions = dt.predict(cc_default_features_test)

    # Calculate accuracy for training and test sets
    train_accuracy = accuracy_score(cc_default_labels_train, train_predictions)
    test_accuracy = accuracy_score(cc_default_labels_test, test_predictions)

    # Append the results to the list
    results.append({
        "Depth": depth,
        "Training Accuracy": train_accuracy,
        "Testing Accuracy": test_accuracy
    })

# Convert results into a DataFrame for better readability
results_df = pd.DataFrame(results)

# Print results table
print(results_df)

# Save results as CSV for further use (optional)
```

```
results_df.to_csv("decision_tree_depth_analysis.csv", index=False)
```

```
   Depth  Training Accuracy  Testing Accuracy
0      1           0.812125          0.815667
1      2           0.812958          0.816333
2      3           0.818750          0.818833
3      5           0.823625          0.820500
4      7           0.829375          0.820000
5     10           0.848042          0.809500
6     15           0.892667          0.795667
7     20           0.937417          0.768667
```

Finish Q4 like Q3, but limiting maximum depth

Question 5

Similarities Between Logistic Regression and Classification Tree

Both are supervised learning MachineLearning algorithms

Differences Between Logistic Regression and Classification Tree

1. **Model Type:**
   - **Logistic Regression**: Assume a linear model
   - **Classification Tree**: Non-linear model that splits data based on feature values.
2. **Interpretability:**
   - **Logistic Regression**: Coefficients represent feature influence, but harder to visualize.
   - **Classification Tree**: Easy to interpret with a tree structure, showing decision-making steps.
3. **Assumptions:**
   - **Logistic Regression**: Assumes linearity and independent errors.
   - **Classification Tree**: Makes no assumptions about data distribution.
4. **Feature Handling:**
   - **Logistic Regression**: Works best with numerical features (needs encoding for categorical).
   - **Classification Tree**: Handles both numerical and categorical features directly.
5. **Overfitting:**
   - **Logistic Regression**: Less prone to overfitting if regularized.
   - **Classification Tree**: More prone to overfitting, especially without pruning.

When Would They Perform Similarly?

- **Scenario: Linearly Separable Data** (e.g., a clear boundary separating classes in feature space).
  - Both algorithms would perform similarly as logistic regression would fit a linear decision boundary and classification trees would make clear splits.

When Would They Perform Differently?

- **Scenario: Non-Linearly Separable Data** (e.g., classes are intertwined or have complex patterns).
  - **Logistic Regression**: Struggles with non-linear separability and performs poorly.
  - **Classification Tree**: Performs well by making complex splits to capture non-linear relationships.

Summary

- **Similar Performance:** When the data is linearly separable.
- **Different Performance:** In complex, non-linear datasets, classification trees tend to outperform logistic regression due to their flexibility.

Question 6

I noticed that my coworker's decision tree is overfitting because the depth is set too high (10). With a small dataset (~~500 data points), a deep tree becomes too complex and memorizes specific details and noise from the training data. This results in near-perfect training accuracy (~~100%) but poor testing accuracy (~60%) because the model struggles to generalize. In small datasets, the model becomes overly specific to the training set, capturing noise instead of general patterns. To fix this, I would recommend reducing the tree's depth or using pruning techniques to prevent overfitting and improve generalization.

Question 7

No, the logistic regression and decision tree do not match. Logistic regression uses a continuous, linear relationship between the features (weight, height, and age) and retention, as shown by the equation. In contrast, the decision tree creates discrete splits based on thresholds (e.g., weight > 200 or age > 40). This results in a non-linear, piecewise decision-making process. While both predict retention, the logistic regression model gives a smooth, continuous prediction, whereas the decision tree makes predictions based on specific feature thresholds.

```
Start coding or generate with AI.
```