7:08 pm

~ ~Naveen Kumar Red... +91 63020 62458

Forwarded

2

```
!touch example.db
import sqlite3
# Connect to the database
conn = sqlite3.connect('example.db')

cursor = conn.cursor()
cursor.execute('''CREATE TABLE
example_table (
            id INTEGER PRIMARY KEY,
            name TEXT NOT NULL,
            age INTEGER
        )''')
cursor.execute('''INSERT INTO
example_table (name, age) VALUES
            ('John', 30),
            ('Mary', 25),
            ('Bob', 40)''')
# Execute a query to retrieve data
cursor.execute('SELECT * FROM
example_table')
rows = cursor.fetchall()
conn.commit()
cursor.execute('SELECT * FROM
example_table')
rows = cursor.fetchall()
for row in rows:
    print(row)
```

7:37 pm

~ ~Naveen Kumar Red... +91 63020 62458

Forwarded

And follow this for the 2nd experiment.

7:37 pm

OUTPUT:

Training Data :

['Sunny','warm','Normal','strong','warm','Same','Yes']
['Sunny','warm','High','strong','warm','Same','Yes']
['Rainy','cold','High','strong','warm','change','NO']
['Sunny','warm','High','strong','cool','change','Yes']

Most specific Hypothesis

['Sunny','warm','?','strong','?','?']

3

Result: Hence, the implementation and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given training data samples is verified Succesfully.

Output:

Predicted classification for VAR1 = 0.906 and VAR2 = 0.606 : class 1

5

Result: Hence, The classifications for nine combinations of VAR1 and VAR2 is predicted Succesfully.

Output

Em Algorithm clustering labels :
[2 2 2 .... 1 1 1]
k-means algorithm clustering labels :
[2 2 2 ... 1 1 1]
Silhoutte score Em algorithm :
0.5992103291678713
Silhoutee score k-means algorithm :
0.5985361996316 17

6

Result:

Hence, applying Em algorithm to cluster a set of data stored in a .csv file

Source Code:

7

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Load the data from a .CSV file
data = pd.read_csv('D:\ML\credit_data.csv')

# Extract features from the data
X = data.iloc[:, :-1].values

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Perform PCA for dimensionality reduction
pca = PCA(n_components=2)  # specify the number of components (dimensions) to reduce to
X_pca = pca.fit_transform(X_scaled)

# Print the reduced dimensionality data
print("Reduced Dimensionality Data (First 5 rows):")
print(X_pca[:5, :])
```

Output

```
Reduced Dimensionality Data (first 5 rows):
[[ -2.91302459   0.09562058]
 [  0.42991133  -0.58815567]
 [ -0.28522508  -0.45517441]
 [ -2.93242265   1.69555507]
 [  1.03357587   0.13665871]]
```

Result: Hence, the implementation prin'of principle Component analysis for Dimensionality Reduction.

```
        if y_test[i] == y_pred[i]:
            print("Correct prediction: Expected:", y_test[i], " Predicted:", y_pred[i])
            correct_predictions += 1
        else:
            print("Wrong prediction: Expected:", y_test[i], " Predicted:", y_pred[i])
            wrong_predictions += 1

    print("Total Correct Predictions: ", correct_predictions)
    print("Total Wrong Predictions: ", wrong_predictions)

    # Print confusion matrix
    confusion = confusion_matrix(y_test, y_pred)
    print("Confusion Matrix:\n", confusion)
```

8

OUTPUT

Accuracy : 1.0
Correct prediction : Expected: Iris-versicolor predicted : Iris-Versicolor
Correct prediction : Expected: Iris-Setosa predicted : Iris-Setosa
Correct prediction : Expected: Iris-versicolor predicted : Iris-Versicolor
Correct prediction : Expected: Iris-Verginica predicted : Iris-Verginica
Total Correct predictions : 30
Total Wrong predictions : 0
Confusion matrix :
[[10 0 0]
[0 9 0]
[0 0 11]]

Result: Hence, the implementation of k-Nearest Neighbour algorithm to classify the iris data set is executed & verified Succesfully.

```
from sklearn.datasets import
load_iris
from sklearn.tree import
DecisionTreeClassifier
from sklearn.model_selection import
train_test_split
from sklearn.metrics import
confusion_matrix, accuracy_score
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import tree

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and
testing datasets
X_train, X_test, y_train, y_test =
train_test_split(X, y, random_state=0)

# Train the Decision Tree Classifier
with Gini Index
clf_gini = DecisionTreeClassifier(crit
erion="gini")
clf_gini.fit(X_train, y_train)

# Predict on the test data
y_pred = clf_gini.predict(X_test)

# Calculate confusion matrix
cm = confusion_matrix(y_test,
y_pred)
print("Confusion Matrix:")
print(cm)
```

10

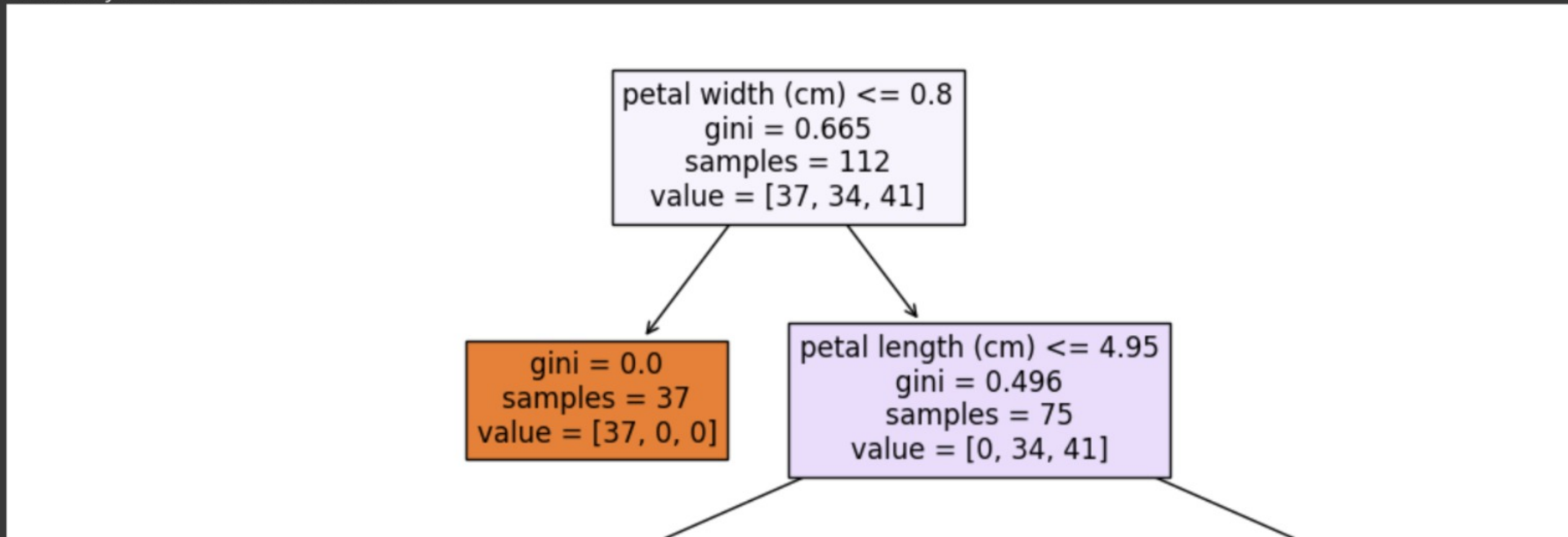
10

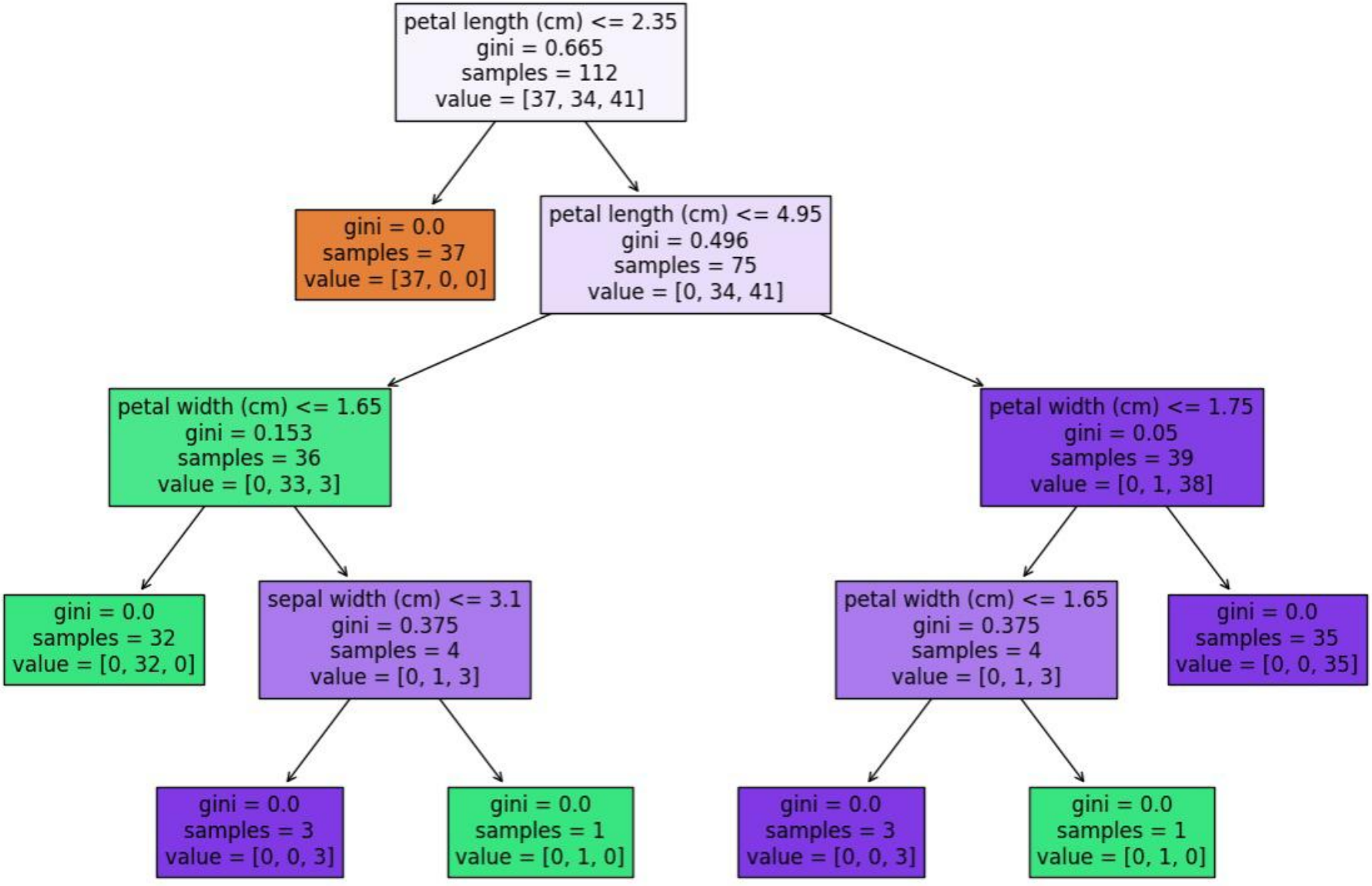
```
# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the data into training and
testing datasets
X_train, X_test, y_train, y_test =
train_test_split(X, y, random_state=0)

# Train the Decision Tree Classifier
with Gini Index
clf_gini = DecisionTreeClassifier(crit
erion="gini")
clf_gini.fit(X_train, y_train)

# Predict on the test data
y_pred = clf_gini.predict(X_test)

# Calculate confusion matrix
cm = confusion_matrix(y_test,
y_pred)
print("Confusion Matrix:")
print(cm)

# Calculate accuracy
accuracy = accuracy_score(y_test,
y_pred)
print("\nAccuracy:", accuracy)
# Visualize the Decision Tree with
Gini Index
plt.figure(figsize=(15,10))
tree.plot_tree(clf_gini, filled=True,
feature_names=iris.feature_names)
plt.show()
```

4:27 pm

```
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy:", accuracy)
# Visualize the Decision Tree with Gini Index
plt.figure(figsize=(15,10))
tree.plot_tree(clf_gini, filled=True, feature_names=iris.feature_names)
plt.show()
```

```
Confusion Matrix:
[[13  0  0]
 [ 0 15  1]
 [ 0  0  9]]

Accuracy: 0.9736842105263158
```




petal length (cm) <= 2.35
gini = 0.665
samples = 112
value = [37, 34, 41]
gini = 0.0
samples = 37
value = [37, 0, 0]
petal length (cm) <= 4.95
gini = 0.496
samples = 75
value = [0, 34, 41]
petal width (cm) <= 1.65
gini = 0.153
samples = 36
value = [0, 33, 3]
petal width (cm) <= 1.75
gini = 0.05
samples = 39
value = [0, 1, 38]
gini = 0.0
samples = 32
value = [0, 32, 0]
sepal width (cm) <= 3.1
gini = 0.375
samples = 4
value = [0, 1, 3]
petal width (cm) <= 1.65
gini = 0.375
samples = 4
value = [0, 1, 3]
gini = 0.0
samples = 35
value = [0, 0, 35]
gini = 0.0
samples = 3
value = [0, 0, 3]
gini = 0.0
samples = 1
value = [0, 1, 0]
gini = 0.0
samples = 3
value = [0, 0, 3]
gini = 0.0
samples = 1
value = [0, 1, 0]