Machine Learning Algorithms

Exercise 5

Name: Chethan Kashyap Bangalore Muralidhara

Date: 20-04-2023

Completed Exercises: 1,2,3,4,5,6(All)

Solutions:

1. Code in python.

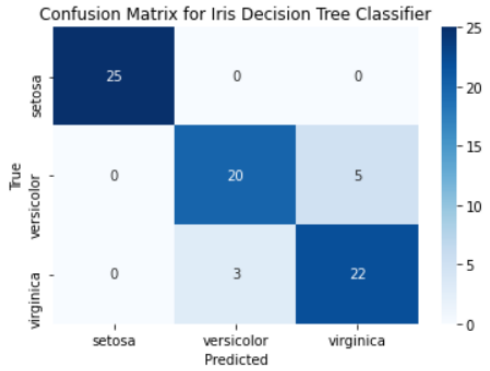
In [3]:
```
###Question 1

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

# Read the data into a Pandas dataframe
data = pd.DataFrame({
    'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny', 'Sunny', 'Rainy', 'Sunny', 'Overcas
    'Temperature': ['Hot', 'Hot', 'Hot', 'Mild', 'Cool', 'Cool', 'Cool', 'Mild', 'Cool', 'Mild', 'Mild', 'Mild', 'Hot', 'Mild'],
    'Humidity': ['High', 'High', 'High', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Normal', 'Normal', 'Normal', 'High', 'Nor
    'Wind': ['false', 'true', 'false', 'false', 'false', 'true', 'true', 'false', 'false', 'false', 'true', 'true', 'false', 'tru
    'PlayTennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
})

# Split the data into features (X) and target (y)
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

# Encode categorical features as integers
X = pd.get_dummies(X, columns=['Outlook', 'Temperature', 'Humidity', 'Wind'])

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Create a decision tree classifier using information gain as the splitting criterion
clf = DecisionTreeClassifier(criterion='entropy')

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Predict the target values for the testing data
y_pred = clf.predict(X_test)

# Print the accuracy score of the classifier
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.3333333333333333
```

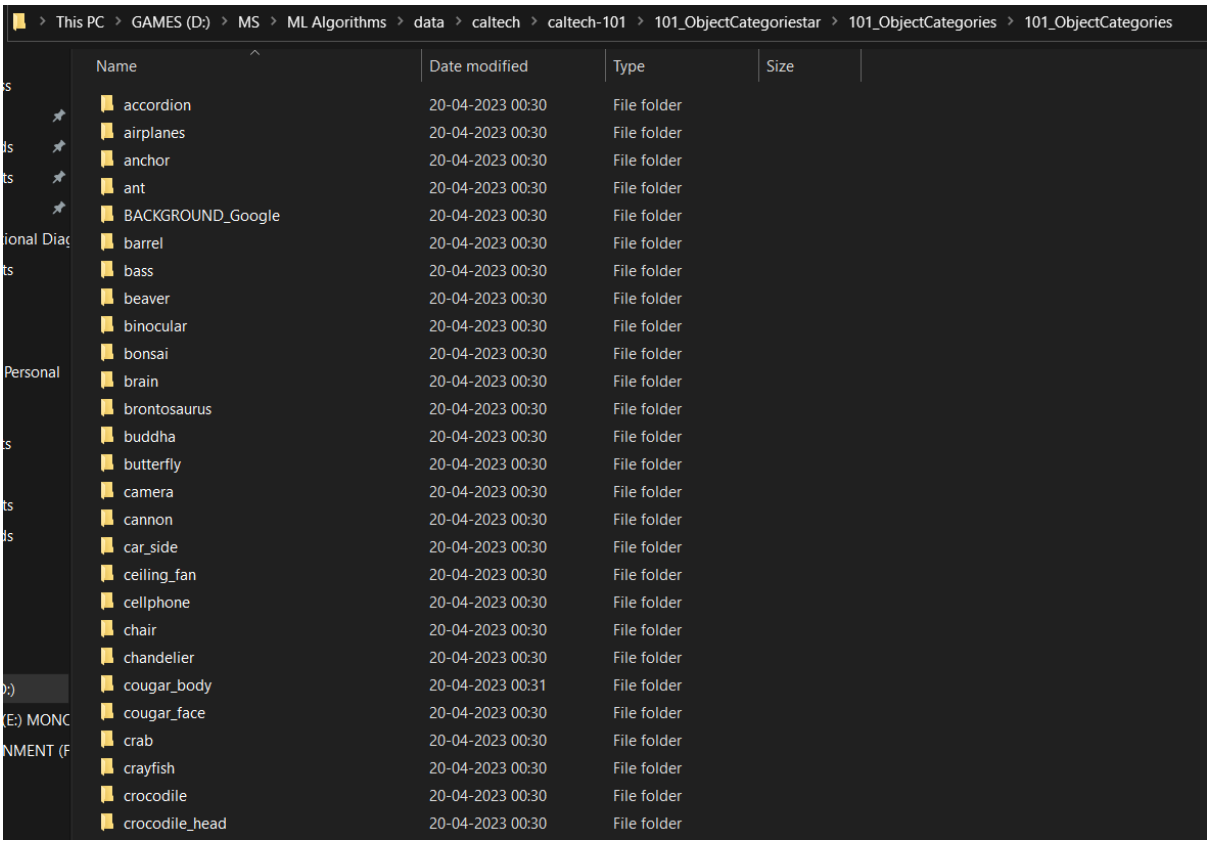
2.

In [5]:
```
###Question 2

import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

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

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, stratify=y, random_state=0)

# Filter the training set to contain only the first 25 cases of each class
class_indices = [0, 1, 2]
X_train_filtered = []
y_train_filtered = []
for class_idx in class_indices:
    class_X = X_train[y_train == class_idx]
    class_y = y_train[y_train == class_idx]
    X_train_filtered.append(class_X[:25])
    y_train_filtered.append(class_y[:25])
X_train_filtered = np.concatenate(X_train_filtered)
y_train_filtered = np.concatenate(y_train_filtered)

# Build a decision tree classifier
clf = DecisionTreeClassifier()
clf.fit(X_train_filtered, y_train_filtered)
```

```
# Predict the target values for the test set
y_pred = clf.predict(X_test)

# Create a confusion matrix
confusion_mat = confusion_matrix(y_test, y_pred)

# Plot the confusion matrix
class_names = iris.target_names
sns.heatmap(confusion_mat, annot=True, cmap='Blues', fmt='d', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix for Iris Decision Tree Classifier')
plt.show()
```



3.



4. Installed in python

5.

In [1]:
```
###Question 5

import os
import numpy as np
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split

# Root folder directory for images
imagefolder = 'D:/MS/ML Algorithms/data/caltech/caltech-101/101_ObjectCategoriestar/101_ObjectCategories/101_ObjectCategories'
imagesetpath = os.path.join(imagefolder)

# Create an ImageDataGenerator for image augmentation
data_generator = ImageDataGenerator(preprocessing_function=lambda x: x)  # No color preprocessing

# Create an ImageDataGenerator for training set
train_datagen = data_generator.flow_from_directory(
    imagesetpath,
    target_size=(224, 224),  # Resize images to match inputSize of ResNet50
    batch_size=32,  # Batch size for training
    class_mode='categorical',  # Categorical class mode for multiclass classification
    subset='training',  # Subset to training set
    shuffle=True,  # Shuffle the data
    seed=42  # Set random seed for reproducibility
)

# Create an ImageDataGenerator for test set
test_datagen = data_generator.flow_from_directory(
    imagesetpath,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation',  # Subset to test set
```

```
# Create an ImageDataGenerator for test set
test_datagen = data_generator.flow_from_directory(
    imagesetpath,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation',  # Subset to test set
    shuffle=False,  # Do not shuffle the data for evaluation
    seed=42
)

# Load a pre-trained ResNet50 model
net = ResNet50(weights='imagenet', include_top=False, pooling='avg')
inputSize = net.layers[0].input_shape[1:3]  # Size of input data for ResNet50
layer = net.get_layer('avg_pool').output  # Features are extracted after avg_pool layer

# Feature extraction of training set and test set images using resized images
featuresTrain = net.predict(train_datagen)
featuresTest = net.predict(test_datagen)

# Labels for each training case
labels = train_datagen.classes
# Unique class labels
lclasses = list(train_datagen.class_indices.keys())

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(featuresTrain, labels, test_size=0.2, random_state=42)
```

```
Found 9144 images belonging to 102 classes.
Found 0 images belonging to 102 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
```

```
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Train a decision tree classifier
clf = DecisionTreeClassifier(max_depth=5, criterion='gini')
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Evaluate the performance of the decision tree
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')
f1 = f1_score(y_test, y_pred, average='macro')

# Print the evaluation metrics
print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1 Score:', f1)
```

6.

```
###Question 6

import os
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Root folder for images
imagefolder = 'D:/MS/ML Algorithms/data/caltech/caltech-101/101_ObjectCategoriestar/101_ObjectCategories/101_ObjectCategories'

# Create a list to store image data and labels
images = []
labels = []

# Load images and labels
for foldername in os.listdir(imagefolder):
    folderpath = os.path.join(imagefolder, foldername)
    for filename in os.listdir(folderpath):
        filepath = os.path.join(folderpath, filename)
        images.append(filepath)
        labels.append(foldername)

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(images, labels, test_size=0.2, random_state=42)

# Create a support vector machine classifier
svm = SVC()
```

```
# Feature extraction of training set images
featuresTrain = []
for image_path in X_train:
    # Perform feature extraction on the image (e.g., resize, convert to grayscale, etc.)
    # Add the extracted features to the featuresTrain list
    featuresTrain.append(extract_features(image_path))

# Convert featuresTrain list to a numpy array
featuresTrain = np.array(featuresTrain)

# Train the SVM classifier
svm.fit(featuresTrain, y_train)

# Feature extraction of test set images
featuresTest = []
for image_path in X_test:
    # Perform feature extraction on the image (e.g., resize, convert to grayscale, etc.)
    # Add the extracted features to the featuresTest list
    featuresTest.append(extract_features(image_path))

# Convert featuresTest list to a numpy array
featuresTest = np.array(featuresTest)

# Predict labels for test set images
y_pred = svm.predict(featuresTest)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```