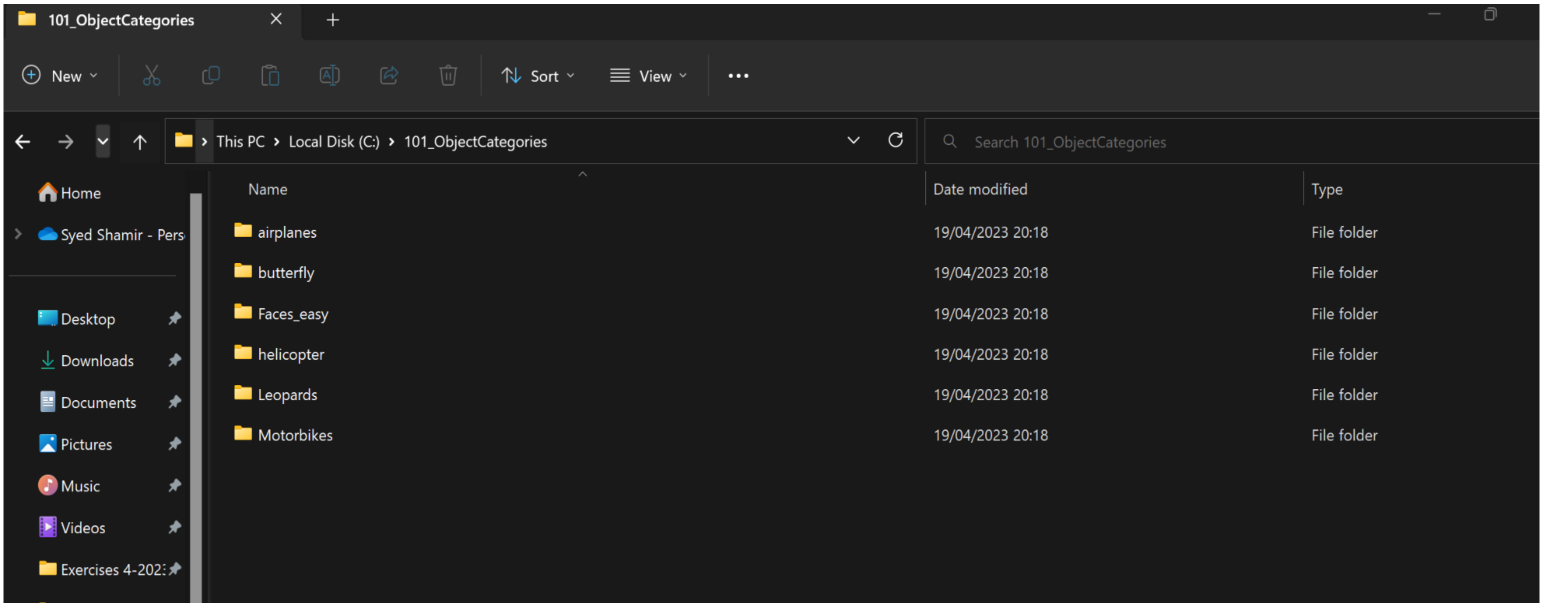
```
From Question 1 to Question 2, solution is in PYTHON
%Question 1:

import pandas as pd
import numpy as np
data = pd.DataFrame({
    'Outlook': ['Sunny', 'Sunny', 'Overcast', 'Rainy', 'Rainy', 'Rainy', 'Overcast', 'Sunny',
'Sunny', 'Rainy', 'Sunny', 'Overcast', 'Overcast', 'Rainy'],
    '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', 'Normal', 'High'],
    'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak',
'Weak', 'Strong', 'Strong', 'Weak', 'Strong'],
    'PlayTennis': ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes',
'Yes', 'Yes', 'No']
})

col = data.columns.tolist()
gain = []
for j in range(len(col)-1):
    feature_data = data[[col[j],col[4]]]
    feat_no  = len(feature_data[feature_data['PlayTennis'] == 'No'])
    feat_yes = len(feature_data[feature_data['PlayTennis'] == 'Yes'])
    P_pos = feat_yes/len(feature_data)
    P_neg = feat_no/len(feature_data)
    feature_entropy = -(P_pos)*(np.log2(P_pos))-(P_neg)*(np.log2(P_neg))
    print(feature_entropy)
    cat=data[col[j]].unique().tolist()
    print(cat)
    ent = []
    for i in range(len(cat)):
            p_yes = data[(data[col[j]] == cat[i]) & (data['PlayTennis'] == 'No')]
            p_no  = data[(data[col[j]] == cat[i]) & (data['PlayTennis'] == 'Yes')]
            p_pos = len(p_yes)/(len(p_yes)+len(p_no))
            p_neg = len(p_no)/(len(p_yes)+len(p_no))
            if p_neg == 0:
                Entropy_Sv =((len(p_yes)+len(p_no))/len(data))*(-(p_pos)*np.log2(p_pos) -
(p_neg))
                Entropy_Sv = abs(Entropy_Sv)
            else:
                Entropy_Sv =((len(p_yes)+len(p_no))/len(data))*(-(p_pos)*np.log2(p_pos) -
(p_neg)*np.log2(p_neg))
                Entropy_Sv = abs(Entropy_Sv)
            ent.append(Entropy_Sv)
            import math
            ent = [0 if math.isnan(x) else x for x in ent]
    print(ent)
    gain.append(feature_entropy - sum(ent))
gain

%[0.24674981977443933, 0.02922256565895487, 0.15183550136234159, 0.04812703040826949]
%max gain is of OUTLOOK feature so we will split from there
```

```
#Replacing Categorical Values with Numerical Values

#Use One Hot Encoding for OutlookTemperature  Humidity    Wind  PlayTennis

from sklearn.preprocessing import OneHotEncoder
ohe_outlook     = OneHotEncoder(sparse = False, handle_unknown = 'ignore')
ohe_temperature = OneHotEncoder(sparse = False, handle_unknown = 'ignore')
ohe_humidity    = OneHotEncoder(sparse = False, handle_unknown = 'ignore')
ohe_wind        = OneHotEncoder(sparse = False, handle_unknown = 'ignore')
ohe_playtennis  = OneHotEncoder(sparse = False, handle_unknown = 'ignore')


data_outlook    = ohe_outlook.fit_transform(data[['Outlook']])
data_temperature= ohe_temperature.fit_transform(data[['Temperature']])
data_humidity   = ohe_humidity.fit_transform(data[['Humidity']])
data_wind       = ohe_wind.fit_transform(data[['Wind']])
data_playtennis = ohe_playtennis.fit_transform(data[['PlayTennis']])

X_data_transformed = np.concatenate((data_outlook, data_temperature, data_humidity, data_wind),
axis = 1)
Y_data_transformed = data_playtennis

## Apply model Decision Tree
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_data_transformed, Y_data_transformed)

from sklearn import tree
tree.plot_tree(clf)
```

```
%Question 2

with open('iris.data', 'r') as file:
    data2 = file.read()

df = pd.read_csv('iris.data', delimiter=',', header=None)
df = df.reset_index(drop = True)

#Transform output feature to numerical values
df[4]=df[4].replace({'Iris-setosa':0, 'Iris-versicolor':1, 'Iris-virginica':2})


#Select the first 25 rows of each class as training data
train_data = pd.DataFrame(np.concatenate((df.iloc[:25, :],df.iloc[50:75, :], df.iloc[100:125,
:]), axis = 0))

#Select the remaining 25 rows of each class as test data
test_data = pd.DataFrame(np.concatenate((df.iloc[25:50, :],df.iloc[75:100, :], df.iloc[125:150,
:]), axis = 0))

X_train_data = train_data.iloc[:,0:4]
Y_train_data = train_data.iloc[:,-1]
X_test_data = test_data.iloc[:,0:4]
Y_test_data = test_data.iloc[:,-1]


from sklearn.tree import DecisionTreeClassifier

clf = DecisionTreeClassifier()
clf.fit(X_train_data, Y_train_data)
y_pred=clf.predict(X_test_data)

from sklearn.metrics import accuracy_score
accuracy_score(Y_test_data, y_pred)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test_data, y_pred)

%OUTPUT
array([[25,  0,  0],
       [ 0, 23,  2],
       [ 0,  2, 23]],
From Question 3 to Question 6, solution is in MATLAB
%%QUESTION 3
```

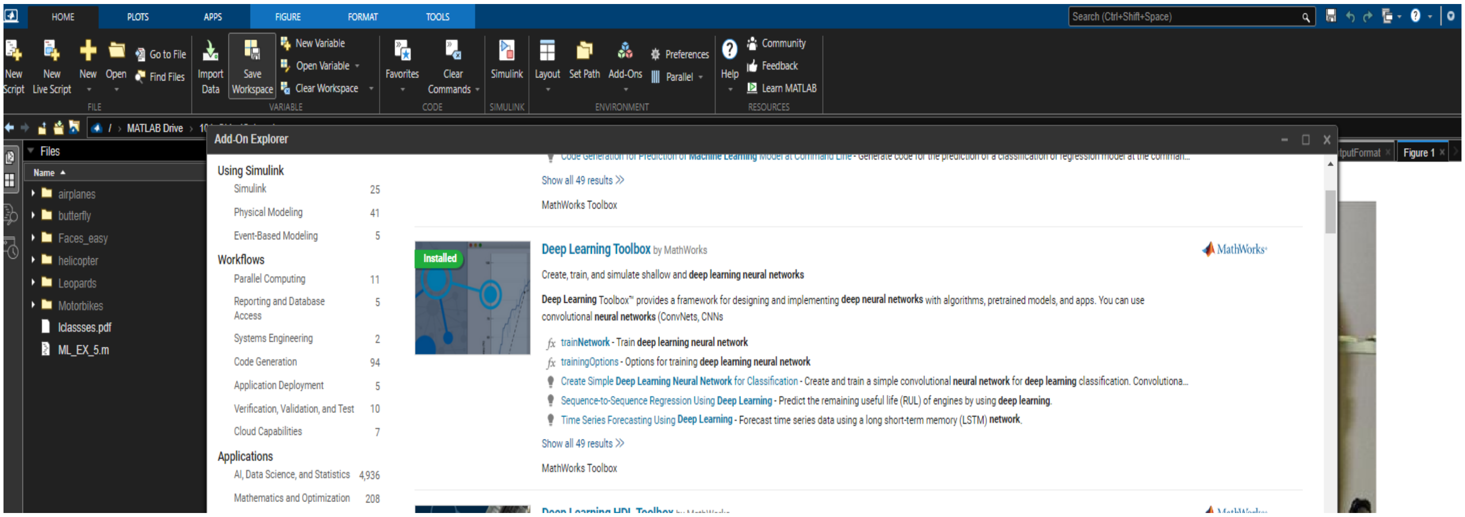


Name	Date modified	Type
airplanes	19/04/2023 20:18	File folder
butterfly	19/04/2023 20:18	File folder
Faces_easy	19/04/2023 20:18	File folder
helicopter	19/04/2023 20:18	File folder
Leopards	19/04/2023 20:18	File folder
Motorbikes	19/04/2023 20:18	File folder

```
%Root folder "directory" for images.
imagefolder='/MATLAB Drive/101_ObjectCategories';
imagesetpath = fullfile(imagefolder);
%Create storage for image data.
images = imageDatastore(imagesetpath, 'IncludeSubfolders',true,'LabelSource','foldernames');
n=1;
img = readimage(images,n);
imshow(img);

%Split data into training and test sets. About 90% of cases belong to training
%set.
[trainingset,testset] = splitEachLabel(images,0.8,'randomized');

%Load in a pre trained deep convolutional neural network.
net = resnet50; %Has 177 layers.
inputSize = net.Layers(1).InputSize; %Size of input data for resnet50.
layer = 'avg_pool'; %Features are extracted after avg_pool layer. Layer 174.

%Resizing of trainingset and testset images for resnet50 network.
augimdsTrain =
augmentedImageDatastore(inputSize(1:2),trainingset,'ColorPreprocessing','gray2rgb');
augimdsTest = augmentedImageDatastore(inputSize(1:2),testset,'ColorPreprocessing','gray2rgb');

%Feature extraction of trainingset and testset images using resized images.
%This stage may take some time even with fast computer.
featuresTrain = activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest = activations(net,augimdsTest,layer,'OutputAs','rows');

%Labels for each training cases.
labels=trainingset.Labels;


%Unique class labels.
lclasses=unique(labels);
```

```
fprintf(lclasses)
```

```
%%QUESTION 4
Deep Learning toolBox Installed
```



Using Simulink

Simulink	25
Physical Modeling	41
Event-Based Modeling	5

Workflows

Parallel Computing	11
Reporting and Database Access	5
Systems Engineering	2
Code Generation	94
Application Deployment	5
Verification, Validation, and Test	10
Cloud Capabilities	7

Applications

AI, Data Science, and Statistics	4,936
Mathematics and Optimization	208

Code Generation for Prediction of Machine Learning Model at Command Line - Generate code for the prediction of a classification or regression model at the comman...

Show all 49 results »

MathWorks Toolbox

Deep Learning Toolbox by MathWorks

Create, train, and simulate shallow and **deep learning neural networks**

Deep Learning Toolbox™ provides a framework for designing and implementing **deep neural networks** with algorithms, pretrained models, and apps. You can use convolutional **neural networks** (ConvNets, CNNs

- *fx* trainNetwork - Train **deep learning neural network**
- *fx* trainingOptions - Options for training **deep learning neural network**
- Create Simple Deep Learning Neural Network for Classification - Create and train a simple convolutional **neural network** for **deep learning** classification. Convolutiona...
- Sequence-to-Sequence Regression Using Deep Learning - Predict the remaining useful life (RUL) of engines by using **deep learning**.
- Time Series Forecasting Using Deep Learning - Forecast time series data using a long short-term memory (LSTM) **network**.

Show all 49 results »

MathWorks Toolbox

Deep Learning HDL Toolbox by MathWorks

```matlab
%% QUESTION 5


%Root folder "directory" for images.
imagefolder='/MATLAB Drive/101_ObjectCategories';
imagesetpath = fullfile(imagefolder);

%Create storage for image data.
images = imageDatastore(imagesetpath, 'IncludeSubfolders',true,'LabelSource','foldernames');

n=1;
img = readimage(images,n);
imshow(img);

%Split data into training and test sets. About 90% of cases belong to training
%set.
[trainingset,testset] = splitEachLabel(images,0.8,'randomized');

%Load in a pre trained deep convolutional neural network.
net = resnet50; %Has 177 layers.
inputSize = net.Layers(1).InputSize; %Size of input data for resnet50.
layer = 'avg_pool'; %Features are extracted after avg_pool layer. Layer 174.

%Resizing of trainingset and testset images for resnet50 network.
augimdsTrain =
augmentedImageDatastore(inputSize(1:2),trainingset,'ColorPreprocessing','gray2rgb');
augimdsTest = augmentedImageDatastore(inputSize(1:2),testset,'ColorPreprocessing','gray2rgb');

%Feature extraction of trainingset and testset images using resized images.
%This stage may take some time even with fast computer.
featuresTrain = activations(net,augimdsTrain,layer,'OutputAs','rows');
featuresTest = activations(net,augimdsTest,layer,'OutputAs','rows');

%Labels for each training cases.
labels=trainingset.Labels;
%%
%Unique class labels.

lclasses=unique(labels);
fprintf(lclasses)
%%applying decision Tree

augimdsTrain;
augimdsTest;
treemodel = fitctree(featuresTrain, labels);
predicted_labels = predict(treemodel, featuresTest);
model_accuracy = sum(predicted_labels == testset.Labels)/numel(testset.Labels)

%% accuracy = 94%
```

```matlab
%QUESTION 6

%fitcoec for multi class svm model
svmmodel = fitcecoc(featuresTrain, labels);
predicted_labels_svm = predict(svmmodel, featuresTest);
model_accuracy_svm = sum(predicted_labels == testset.Labels)/numel(testset.Labels)

%accuracy = 94.5
```