LAB-1

Q1) Write a python program to import and export data using Pandas library functions

```
[36] import pandas as pd
```

```
[37] airbnb_data = pd.read_csv("/content/austinHousingData.csv")
```

```
airbnb_data.to_csv("/content/austinHousingData.csv")
```

```
[ ] airbnb_data.head()
```

	Unnamed: 0.1	Unnamed: 0	zpid	city	streetAddress	zipcode	description	latitude	longitude	propertyTaxRate	...	numOfMiddleSchools	numOfHighSchools	avgSchoolDistance	avgSchoolRating	avgSchoolSize	MedianStudentsPerTeacher	numOfBathrooms
0	0	0	111373431	pflugerville	14424 Lake Victor Dr	78660	14424 Lake Victor Dr. Pflugerville, TX 78660 I...	30.430632	-97.663078	1.98	...	1	1	1.266667	2.666667	1063	14	3.0
1	1	1	120900430	pflugerville	1104 Strickling Dr	78660	Absolutely GORGEOUS 4 Bedroom home with 2 full...	30.432673	-97.661697	1.98	...	1	1	1.400000	2.666667	1063	14	2.0
2	2	2	2084491383	pflugerville	1408 Fort Dessau Rd	78660	Under construction - estimated completion in A...	30.409748	-97.639771	1.98	...	1	1	1.200000	3.000000	1108	14	2.0
3	3	3	120901374	pflugerville	1025 Strickling Dr	78660	Absolutely darling one story home in charming ...	30.432112	-97.661659	1.98	...	1	1	1.400000	2.666667	1063	14	2.0
4	4	4	60134862	pflugerville	15005 Donna Jane Loop	78660	Brimming with appeal & warm livability! Sleek ...	30.437368	-97.656860	1.98	...	1	1	1.133333	4.000000	1223	14	3.0

```
[40] import pandas as pd
```

```
[41] iris_data = pd.read_csv("/content/iris.data")
```

```
[42] iris_data.head()
```

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

Next steps: View recommended plots

```
[43] # Webpage URL
     url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"

     # Define the column names
     col_names = ["sepal_length_in_cm",
                  "sepal_width_in_cm",
                  "petal_length_in_cm",
                  "petal_width_in_cm",
                  "class"]

     # Read data from URL
     iris_data = pd.read_csv(url, names=col_names)

     iris_data.head()
```

	sepal_length_in_cm	sepal_width_in_cm	petal_length_in_cm	petal_width_in_cm	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Austin & iris

import

```
import pandas as pd
airbnb_data = pd.read_csv("/content/austinHousingData.csv")
airbnb_data.head()
```

~~output~~

Export:-

```
airbnb_data.to_csv("/content/austinHousingData.csv")
```

austinHousingData.csv

Reading Data from url:-

~~url = "https~~

```
import pandas as pd
iris_data = pd.read_csv("/content/iris.data")
iris_data.head()

url = "https://archive.ics.uci.edu/ml/
       machine_learning_databases/iris/iris.data"

colnames = ["sepal_length_in_cm",
            "sepal_width_in_cm",
            "petal_length_in_cm",
            "petal_width_in_cm",
            "class"]

iris_data = pd.read_csv(url, names=col_names)
iris_data.head()
```

LAB-2

Use appropriate dataset to building the decision tree (ID3) and apply this knowledge to classify a new sample.

1.) importing data set

```
import numpy as np
import pandas as pd
eps = np.finfo(float).eps
from numpy import log2 as log
import matplotlib.pyplot as plt
```

```
[2] outlook = 'overcast,overcast,overcast,overcast,rainy,rainy,rainy,rainy,rainy,sunny,sunny,sunny,sunny,sunny'.split(',')
temp = 'hot,cool,mild,hot,mild,cool,cool,mild,mild,hot,hot,mild,cool,mild'.split(',')
humidity = 'high,normal,high,normal,high,normal,normal,normal,high,high,high,high,normal,normal'.split(',')
windy = 'FALSE,TRUE,TRUE,FALSE,FALSE,FALSE,TRUE,FALSE,TRUE,FALSE,TRUE,FALSE,FALSE,TRUE'.split(',')
play = 'yes,yes,yes,yes,yes,yes,no,yes,no,no,no,no,yes,yes'.split(',')
```

```
dataset ={'outlook':outlook,'temp':temp,'humidity':humidity,'windy':windy,'play':play}
df = pd.DataFrame(dataset,columns=['outlook','temp','humidity','windy','play'])
df
```

	outlook	temp	humidity	windy	play
0	overcast	hot	high	FALSE	yes
1	overcast	cool	normal	TRUE	yes
2	overcast	mild	high	TRUE	yes
3	overcast	hot	normal	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	rainy	mild	normal	FALSE	yes
8	rainy	mild	high	TRUE	no
9	sunny	hot	high	FALSE	no
10	sunny	hot	high	TRUE	no
11	sunny	mild	high	FALSE	no
12	sunny	cool	normal	FALSE	yes
13	sunny	mild	normal	TRUE	yes

2)find the entropy

```
[4] ##1. claculate entropy o the whole dataset

    entropy_node = 0  #Initialize Entropy
    values = df.play.unique()  #Unique objects - 'Yes', 'No'
    for value in values:
        fraction = df.play.value_counts()[value]/len(df.play)
        entropy_node += -fraction*np.log2(fraction)

    print(f'Values: {values}')
    print(f'entropy_node: {entropy_node}')
```

```
Values: ['yes' 'no']
entropy_node: 0.9402859586706311
```

```
[5] def ent(df,attribute):
        target_variables = df.play.unique()  #This gives all 'Yes' and 'No'
        variables = df[attribute].unique()    #This gives different features in that attribute (like 'Sweet')


        entropy_attribute = 0
        for variable in variables:
            entropy_each_feature = 0
            for target_variable in target_variables:
                num = len(df[attribute][df[attribute]==variable][df.play ==target_variable]) #numerator
                den = len(df[attribute][df[attribute]==variable])  #denominator
                fraction = num/(den+eps)  #pi
                entropy_each_feature += -fraction*log(fraction+eps) #This calculates entropy for one feature like 'Sweet'
            fraction2 = den/len(df)
            entropy_attribute += -fraction2*entropy_each_feature   #Sums up all the entropy ETaste

        return(abs(entropy_attribute))
    a_entropy = {k:ent(df,k) for k in df.keys()[:-1]}
    a_entropy
```

```
{'outlook': 0.6935361388961914,
 'temp': 0.9110633930116756,
 'humidity': 0.7884504573082889,
 'windy': 0.892158928262361}
```

3)find the information gain

```
[6] def ig(e_dataset,e_attr):
        return(e_dataset-e_attr)
    #entropy_node = entropy of dataset
    #a_entropy[k] = entropy of k(th) attr
    IG = {k:ig(entropy_node,a_entropy[k]) for k in a_entropy}
    IG
```

```
{'outlook': 0.24674981977443977,
 'temp': 0.029222565658955535,
 'humidity': 0.15183550136234225,
 'windy': 0.048127030408270155}
```

4)find the attribute with the max information gain

```
def find_entropy(df):
    Class = df.keys()[-1]   #To make the code generic, changing target variable class name
    entropy = 0
    values = df[Class].unique()
    for value in values:
        fraction = df[Class].value_counts()[value]/len(df[Class])
        entropy += -fraction*np.log2(fraction)
    return entropy


def find_entropy_attribute(df,attribute):
  Class = df.keys()[-1]   #To make the code generic, changing target variable class name
  target_variables = df[Class].unique()  #This gives all 'Yes' and 'No'
  variables = df[attribute].unique()    #This gives different features in that attribute (like 'Hot','Cold' in Temperature)
  entropy2 = 0
  for variable in variables:
      entropy = 0
      for target_variable in target_variables:
          num = len(df[attribute][df[attribute]==variable][df[Class] ==target_variable])
          den = len(df[attribute][df[attribute]==variable])
          fraction = num/(den+eps)
          entropy += -fraction*log(fraction+eps)
      fraction2 = den/len(df)
      entropy2 += -fraction2*entropy
  return abs(entropy2)


def find_winner(df):
    Entropy_att = []
    IG = []
    for key in df.keys()[:-1]:
#         Entropy_att.append(find_entropy_attribute(df,key))
        IG.append(find_entropy(df)-find_entropy_attribute(df,key))
    return df.keys()[:-1][np.argmax(IG)]


def get_subtable(df, node,value):
  return df[df[node] == value].reset_index(drop=True)
```

5)build the tree

```
def buildTree(df,tree=None):
    Class = df.keys()[-1]   #To make the code generic, changing target variable class name

    #Here we build our decision tree

    #Get attribute with maximum information gain
    node = find_winner(df)

    #Get distinct value of that attribute e.g Salary is node and Low,Med and High are values
    attValue = np.unique(df[node])

    #Create an empty dictionary to create tree
    if tree is None:
        tree={}
        tree[node] = {}

   #We make loop to construct a tree by calling this function recursively.
    #In this we check if the subset is pure and stops if it is pure.

    for value in attValue:

        subtable = get_subtable(df,node,value)
        clValue,counts = np.unique(subtable[Class],return_counts=True)

        if len(counts)==1:#Checking purity of subset
            tree[node][value] = clValue[0]
        else:
            tree[node][value] = buildTree(subtable) #Calling the function recursively

    return tree

t  = buildTree(df)
import pprint
pprint.pprint(t)
```

Output:-

```
{'outlook': {'overcast': 'yes',
             'rainy': {'windy': {'FALSE': 'yes', 'TRUE': 'no'}},
             'sunny': {'humidity': {'high': 'no', 'normal': 'yes'}}}}
```