**Name: Abdul Moiz Asif**

**CMS: 263802**

import pandas as pd

import numpy as np

data = pd.read\_csv('./DataSet.csv')

def data\_split(data):

train = data.sample(n = 10)

train = train.iloc[:].reset\_index(drop=True)

test = data.sample(n = 2)

test = test.iloc[:].reset\_index(drop=True)

return train, test

def disorder(target):

elements, counts = np.unique(target, return\_counts=True)

return np.sum([(-counts[i]/np.sum(counts))\*np.log2(counts[i] / np.sum(counts)) for i in range(len(elements))])

def pred(query, id\_tree, default=1):

for key in list(query.keys()):

if key in list(id\_tree.keys()):

try:

result = id\_tree[key][query[key]]

except:

return default

result = id\_tree[key][query[key]]

if isinstance(result, dict):

return pred(query, result)

else:

return result

def check(data, id\_tree):

queries = data.iloc[:, :-1].to\_dict(orient="records")

predicted = pd.DataFrame(columns=["WillWait"])

for i in range(len(data)):

predicted.loc[i, "WillWait"] = pred(queries[i], id\_tree, 1.0)

print("\nPrediction for testing dataSet:")

print(predicted)

print('\nAccuracy is: ',(np.sum(predicted["WillWait"] == data["WillWait"])/len(data))\*100,'%')

def validity(data, split\_attribute\_name, target="WillWait"):

total\_disorder = disorder(data[target])

vals, counts = np.unique(data[split\_attribute\_name], return\_counts=True)

set\_disorder = np.sum([(counts[i]/np.sum(counts))\*disorder(data.where(data[split\_attribute\_name] == vals[i]).dropna()[target]) for i in range(len(vals))])

return total\_disorder - set\_disorder

def id\_tree(data, originaldata, attr, target="WillWait", parent\_node\_class=None):

if len(np.unique(data[target])) <= 1:

return np.unique(data[target])[0]

elif len(data) == 0:

return np.unique(originaldata[target])[np.argmax(np.unique(originaldata[target], return\_counts=True)[1])]

elif len(attr) == 0:

return parent\_node\_class

else:

parent\_node\_class = np.unique(data[target])[np.argmax(np.unique(data[target], return\_counts=True)[1])]

item\_values = [validity(data, feature, target) for feature in attr]

best\_feature\_index = np.argmax(item\_values)

best\_feature = attr[best\_feature\_index]

tree = {best\_feature: {}}

attr = [i for i in attr if i != best\_feature]

for value in np.unique(data[best\_feature]):

value = value

sub\_data = data.where(data[best\_feature] == value).dropna()

subtree = id\_tree(sub\_data, data, attr, target, parent\_node\_class)

tree[best\_feature][value] = subtree

return(tree)

train, test = data\_split(data)

print("Testing Data:")

print(test)

print("\nTraining Data:")

print(train)

tree = id\_tree(train, train, train.columns[:-1])

check(test, tree)

Output:

