Name : Rishabh Kumar Rai

Reg No. : 20BKT0076

Q1) Implement and demonstrate the FIND-S algorithm for finding the most specific hypothesis based on a given set of training data samples. Read the training data from a .CSV file.

Code:

```
import pandas as pd

import numpy as np


#to read the data in the csv file

data = pd.read_csv("WS.csv")

print(data,"n")


#making an array of all the attributes

d = np.array(data)[:,:-1]

print("n The attributes are: ",d)


#segragating the target that has positive and negative examples

target = np.array(data)[:,-1]

print("n The target is: ",target)


#training function to implement find-s algorithm

def train(c,t):

    for i, val in enumerate(t):

        if val == "Yes":

            specific_hypothesis = c[i].copy()

            break


    for i, val in enumerate(c):

        if t[i] == "Yes":

            for x in range(len(specific_hypothesis)):

                if val[x] != specific_hypothesis[x]:

                    specific_hypothesis[x] = '?'
```

```
            else:
                pass

    return specific_hypothesis


#obtaining the final hypothesis
print("n The final hypothesis is:",train(d,target))
```

Output:

```
    return specific_hypothesis

#obtaining the final hypothesis
print("n The final hypothesis is:",train(d,target))
```

```
   Sunny  Warm Normal  Strong Warm.1    Same  Yes
0  Sunny  Warm   High  Strong   Warm    Same  Yes
1  Rainy  Cold   High  Strong   Warm  Change   No
2  Sunny  Warm   High  Strong   Cool  Change  Yes n
n The attributes are:  [['Sunny' 'Warm' 'High' 'Strong' 'Warm' 'Same']
 ['Rainy' 'Cold' 'High' 'Strong' 'Warm' 'Change']
 ['Sunny' 'Warm' 'High' 'Strong' 'Cool' 'Change']]
n The target is:  ['Yes' 'No' 'Yes']
n The final hypothesis is: ['Sunny' 'Warm' 'High' 'Strong' '?' '?']
```

Q2) For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent with the training examples.

Code:

```
import csv


with open("ws.csv") as f:
    csv_file = csv.reader(f)
    data = list(csv_file)


    specific = data[1][:-1]
    general = [['?' for i in range(len(specific))] for j in range(len(specific))]


    for i in data:
        if i[-1] == "Yes":
            for j in range(len(specific)):
                if i[j] != specific[j]:
```

```
                specific[j] = "?"
                general[j][j] = "?"

        elif i[-1] == "No":
            for j in range(len(specific)):
                if i[j] != specific[j]:
                    general[j][j] = specific[j]
                else:
                    general[j][j] = "?"

        print("\nStep " + str(data.index(i)+1) + " of Candidate Elimination Algorithm")
        print(specific)
        print(general)

    gh = [] # gh = general Hypothesis
    for i in general:
        for j in i:
            if j != '?':
                gh.append(i)
                break
    print("\nFinal Specific hypothesis:\n", specific)
    print("\nFinal General hypothesis:\n", gh)
```

Output:

```
Step 1 of Candidate Elimination Algorithm
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]

Step 2 of Candidate Elimination Algorithm
['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]

Step 3 of Candidate Elimination Algorithm
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]

Step 4 of Candidate Elimination Algorithm
['Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', 'Same']]

Step 5 of Candidate Elimination Algorithm
['Sunny', 'Warm', '?', 'Strong', '?', '?']
[['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Final Specific hypothesis:
 ['Sunny', 'Warm', '?', 'Strong', '?', '?']

Final General hypothesis:
 [['Sunny', '?', '?', '?', '?', '?'], ['?', 'Warm', '?', '?', '?', '?']]
```

Q4) A XYZ company has conducting the research for tracking the real estate investments carried out on last year to reveal the sales figures of new houses of different prices. Plot the data and check for the linear relationship between attributes if any? and find the least square regression line.

price $(xi) 160 280 180 200 260 240 220 170

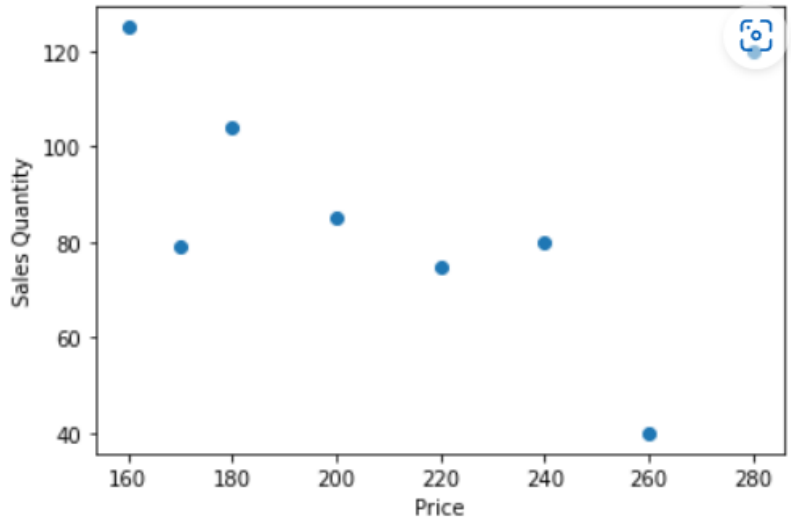
Sales quantity (yi) 125 120 104 85 40 80 75 79

Code:

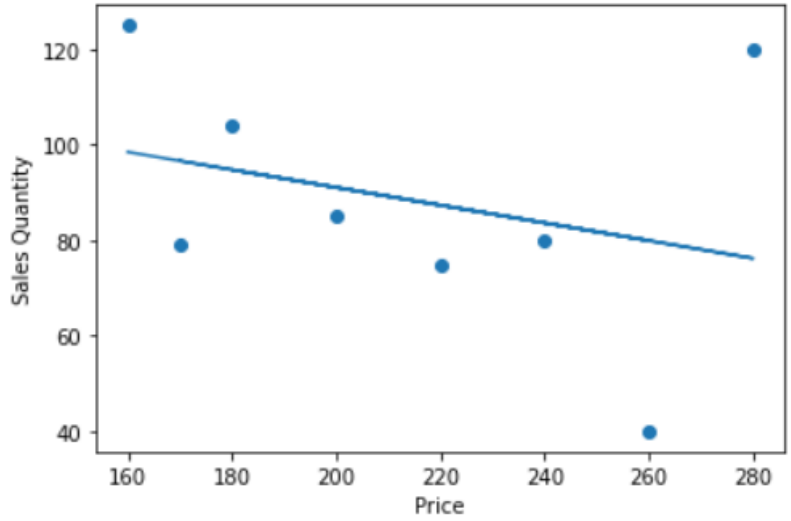
```
import matplotlib.pyplot as plt

import numpy as np

from scipy.stats import linregress


# Create the dataset

price = [160, 280, 180, 200, 260, 240, 220, 170]

sales_quantity = [125, 120, 104, 85, 40, 80, 75, 79]


# Plot the data

plt.scatter(price, sales_quantity)

plt.xlabel('Price')

plt.ylabel('Sales Quantity')

plt.show()
```

```
#Checking Linear Relationship
slope, intercept, r_value, p_value, std_err = linregress(price, sales_quantity)
print("slope: ", slope)
print("intercept: ", intercept)
print("r_value: ", r_value)
print("p_value: ", p_value)
print("std_err: ", std_err)

# Plotting the Least square Regression Line
line = [slope*xi + intercept for xi in price]
plt.scatter(price, sales_quantity)
plt.xlabel('Price')
plt.ylabel('Sales Quantity')
plt.plot(price, line)
plt.show()
```

Output:




```
slope:  -0.18562091503267975
intercept:  128.1764705882353
r_value:  -0.2957376301658849
p_value:  0.47697540679638883
std_err:  0.24477692420323022
```



Q5) Write a program to demonstrate the working of the decision tree based ID3 algorithm. Use an appropriate data set for building the decision tree and apply this knowledge to classify a new sample.

Code:

```
import pandas as pd

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score

from sklearn.preprocessing import OneHotEncoder


# Create a small synthetic dataset

data = {'Taste': ['Sweet', 'Sour', 'Spicy', 'Spicy', 'Sweet', 'Sour', 'Spicy', 'Sweet'],
```

```
    'Temperature': ['Hot', 'Hot', 'Hot', 'Cold', 'Cold', 'Cold', 'Cold', 'Hot'],
    'Texture': ['Soft', 'Soft', 'Hard', 'Hard', 'Soft', 'Soft', 'Hard', 'Hard'],
    'Class': ['Fruit', 'Fruit', 'Vegetable', 'Vegetable', 'Fruit', 'Fruit', 'Vegetable', 'Vegetable']}
df = pd.DataFrame(data)

# One-hot encode the categorical features
enc = OneHotEncoder(sparse=False, categories='auto')

# fit the encoder to the dataframe, and obtain the feature names
enc.fit(df[df.columns[:-1]])
enc_feature_names = enc.get_feature_names(df.columns[:-1])

# transform the data and put back the feature names
X = pd.DataFrame(enc.transform(df[df.columns[:-1]]), columns = enc_feature_names)
y = df['Class']

# Train a decision tree classifier using the ID3 algorithm
clf = DecisionTreeClassifier(criterion='entropy')
clf = clf.fit(X, y)

# Use the trained classifier to predict the class of a new sample
sample = pd.DataFrame([['Sweet', 'Hot', 'Soft']],columns = df.columns[:-1])
sample = pd.DataFrame(enc.transform(sample), columns = enc_feature_names)
prediction = clf.predict(sample)
print(prediction)
```

Output:

In [9]:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder

# Create a small synthetic dataset
data = {'Taste': ['Sweet', 'Sour', 'Spicy', 'Spicy', 'Sweet', 'Sour', 'Spicy', 'Sweet'],
        'Temperature': ['Hot', 'Hot', 'Hot', 'Cold', 'Cold', 'Cold', 'Cold', 'Hot'],
        'Texture': ['Soft', 'Soft', 'Hard', 'Hard', 'Soft', 'Soft', 'Hard', 'Hard'],
        'Class': ['Fruit', 'Fruit', 'Vegetable', 'Vegetable', 'Fruit', 'Fruit', 'Vegetable', 'Vegetable']}
df = pd.DataFrame(data)

# One-hot encode the categorical features
enc = OneHotEncoder(sparse=False, categories='auto')

# fit the encoder to the dataframe, and obtain the feature names
enc.fit(df[df.columns[:-1]])
enc_feature_names = enc.get_feature_names(df.columns[:-1])

# transform the data and put back the feature names
X = pd.DataFrame(enc.transform(df[df.columns[:-1]]), columns = enc_feature_names)
y = df['Class']

# Train a decision tree classifier using the ID3 algorithm
clf = DecisionTreeClassifier(criterion='entropy')
clf = clf.fit(X, y)

# Use the trained classifier to predict the class of a new sample
sample = pd.DataFrame([['Sweet', 'Hot', 'Soft']],columns = df.columns[:-1])
sample = pd.DataFrame(enc.transform(sample), columns = enc_feature_names)
prediction = clf.predict(sample)
print(prediction)
```

```
['Fruit']
```