Machine Learning

Experiment - 1

Implement and demonstrate 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.

Program:-

~~In[7]: import~~

```
In[7] : import pandas as pd
        import numpy as np

In[8] : data = pd.read_csv(\\172.16.17.3 \All data \ Enjoy sport.csv)
        Print (data)
```

	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
0	Sunny	warm	Normal	strong	Warm	Same	1
1	Sunny	warm	high	strong	Warm	same	1
2	Rainy	cold	high	Strong	Warm	change	0
3	Sunny	Warm	high	Strong	Warm	change	1

```
In[9] : d = np.array(data)[:, :-1]
        Print ("The attributes are:", d)
```

The attributes are: [['Sunny', 'Warm', 'Normal', 'Strong', 'warm', 'same']
['Sunny' 'warm' 'high' 'strong' 'warm' 'same']

['Rainy' 'cold' 'High' 'strong' 'warm' 'change']

['sunny' 'warm' 'High' 'strong' 'cool' 'change']]

```
In [10]: target = np.array(data)[:, -1]
         print ("the target is :", target)
    the target is: [1 1 0 1]

In [11]: def train(c,t):
             specific_hypothesis = [None]* len (c[0])

             for i in range (len(c)):
                 if t[i] == 1;
                     for j in range (len (c[i])):
                         if specific_hypothesis [j] is none:
                             specific_hypothesis [j] = c[i][j]
                     else specific_hypothesis [j] != c[i][j] :
                             specific hypothesis [j] = '?'

             return specific_hypothesis

In [12]: print(" the final hypothesis is :", train (d,target))
    The final hypothesis is : ['sunny', 'warm', '?', 'strong',
                               '?', '?']
```

10/11/23 5

EXPERIMENT-2

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 hypothesis consistent with the training examples.

Program:-

```
import numpy as np
import pandas as pd
data = pd.read_csv (Path + '\enjoysport.csv')
concepts = np. array (data.iloc [:, 0:-1])
Print ("\n Instances are :\n", concepts)
target = np. array (data.iloc [:,-1])
Print ("\n Target values are : ", target)
def learn (concepts, target):
    specific_h = concepts [0].copy()
    Print ("\n Initialization of specific_h and general_h")
    Print ("\n Specific Boundary : ", specific_h)
    general_h = [["?" for i in range (len (specific_h))]
    for i in range (len (specific_h))
     Print ("\n Generic Boundary : ", general_h)
    for i, h in enumerate (concepts):
        Print ("\n Instance ", i+1, "is", h)
        if target [i] = = "yes":
            Print ("Instance is positive ")
            for x in range (len (specific_h)):
```

```
if h[x] != specific_h[x]:
        specific_h[x] = '?'
        general_h[x][x] = '?'
target [i] == "no":
Print ("Instance is negative ")
for x in range (len (specific_h)):
    if h[x] != specific_h[x]:
        general_h[x][x] = specific_h[x]
    else:
        general_h[x][x] ='?'
Print ("specific Boundary after ", i+1, "Instance is", specific_h)
Print ("Generic Boundary after", i+1, "Instance is", general_h)
Print("\n")
indices = [i for i, val in enumerate (general_h) if
                    val == ['?','?','?','?','?','?']]
for i in indices:
    general_h.remove (['?','?','?','?','?','?'])
return specific_h, general_h
S_final, g_final = learn (concepts, targets)
Print ("Final specific_h :", S_final, sep = "\n")
Print ("Final General_h:", g_final, sep = "\n")
```

Initialization of specific_h and general_h

Specific Boundary: ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

Generic Boundary: [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']]

Instance 1 is ['sunny' 'warm' 'normal' 'strong' 'warm' 'same'] instance is positive.

Specific Boundary after 1 instance is ['sunny' 'warm' 'normal' 'strong' 'warm' 'same']

Generic Boundary after 1 instance is [['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?']].

G1 = G

S1 = ['sunny', 'warm', 'normal', 'strong', 'warm', 'same']

For instance 2: positive output.

$G_2 = G$

S_2 = ['sunny', 'warm', ?, 'strong', 'warm', 'same']

For instance 3: <'rainy', 'cold', 'high', 'strong', 'warm', 'change'> and negative output.

G_3 = [['sunny', ?, ?, ?, ?, ?], [?, 'warm', ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, ?], [?, ?, ?, ?, ?, 'same']]

$S_3 = S_2$

For instance 4: <'sunny', 'warm', 'high', 'strong', 'cool', 'change'> and positive output.

$G_4 = G_3$

S_4 = ['sunny', 'warm', ?, 'strong', ?, ?]

```
Print ("Final specific_h: ", s_final, sep="\n")
Print ("Final General_h: ", g_final, sep="\n").
```

Final Specific_h: ['sunny' 'warm' '?' 'strong' '?' '?']

Final General_h: [['sunny', '?', '?', '?', '?', '?'],

['?', 'warm', '?', '?', '?', '?']].

EXPERIMENT-3

Write a program to demonstrate the working of Decision tree regressor. Use appropriate dataset for decision tree regressor.

```
import pandas as pd
from pandas import DataFrame
df_tennis = DataFrame
df_tennis = DataFrame.from_csv (Path)
Print ("\n Given play Tennis Data set: \n\n" df_tennis)
```

Given play Tennis Data set:

	PlayTennis	outlook	Temperature	Humidity	wind
0	No	Sunny	Hot	High	
1	No	Sunny	Hot	High	
2	Yes	overcast	Hot	High	
3	Yes	Rain	Mild	High	
4	Yes	Rain	cool	Normal	
5	No	Rain	cool	Normal	
6	Yes	overcast	cool	High	
7	No	sunny	Mild	Normal	
8	Yes	sunny	cool	Normal	
9	Yes	Rain	mild	Normal	
10	Yes	Sunny	mild	High	
11	Yes	Overcast	mild	Normal	
12	Yes	overcast	Hot	High	
13	No	Rain	mild		

```
df_tennis.key() [0]
'play Tennis'
```

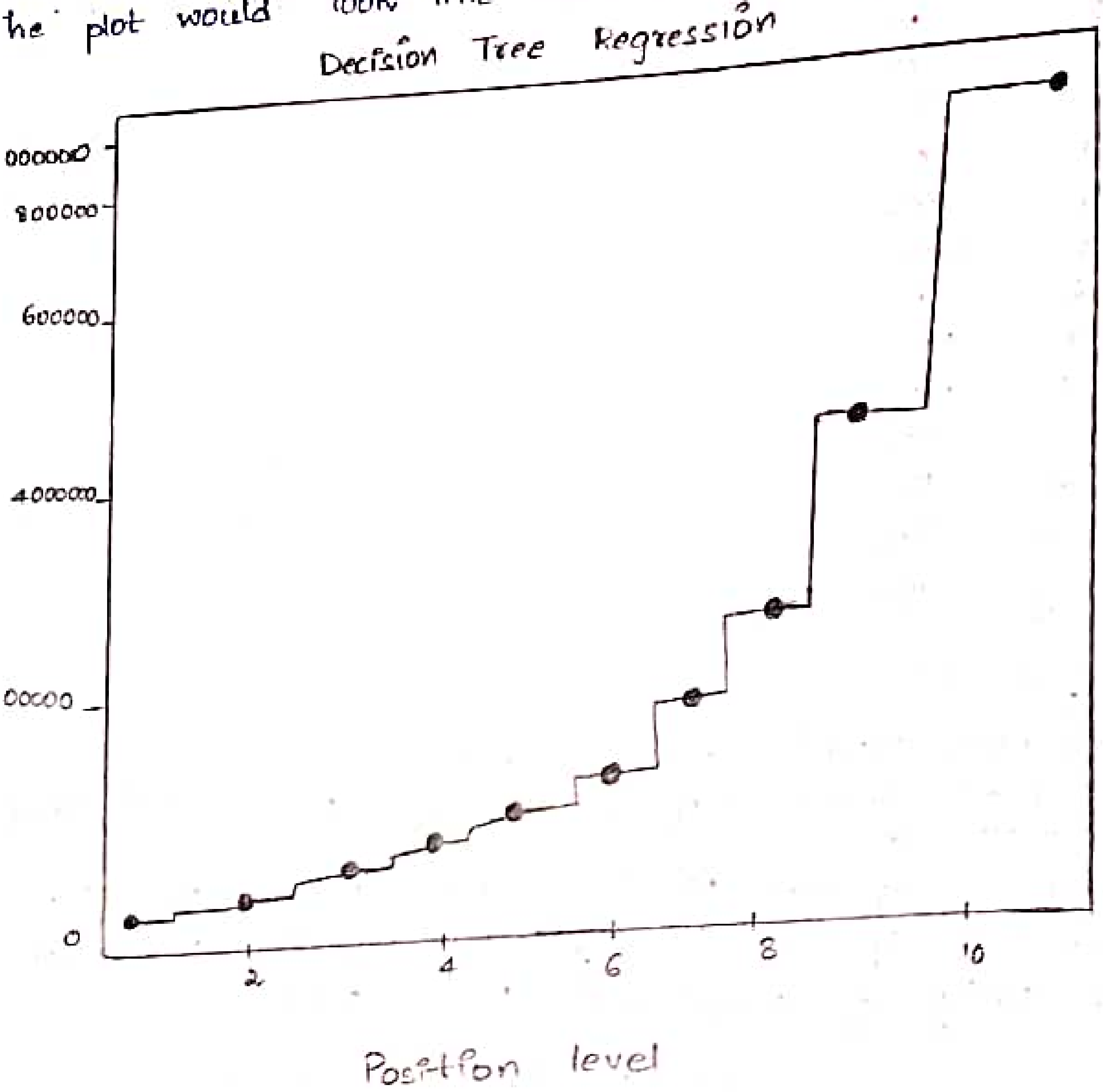
```
def entropy (Probs):
    import math
    return sum([-prob * math.log (Prob, 2) for prob in Probs])

def entropy_of_list (a list):
    from collections import counter
    cnt = counter (x for x in a_list)
    num_instances = len (a_list) * 1.0
    Print ("\n Number of Instances of the current sub
                class is {0}:" .format (num))
    Probs = [x / num_instances for x in cnt.values ()]
    Print ("\n classes : ", min (cnt) )
    Print ("\n Probabilities of class {0} is {1} : ". format
                    (max(cnt) , max (probs))

    return entropy (Probs)

Print ("\n INPUT DATA SET FOR ENTROPY CALCULATION :
            \n ", df_tennis ['play Tennis']
total_entropy = entropy_of_list (df_tennis ('play
                                        Tennis'])
Print ("\n Total Entropy of Play Tennis Data set:",
                                    total_entropy)
```

INPUT DATASET FOR ENTROPY CALCULATION:

0	NO
1	Yes
2	Yes
3	Yes
4	Yes
5	NO
6	Yes
7	NO
8	Yes
9	Yes
10	Yes
11	Yes
12	Yes
13	NO

Name: play Tennis, dtype: object
Number of Instances of the current sub class is 14.0:
classes: NO Yes
Probabilities of class No is 0.35714285714285715:
Probabilities of class Yes is 0.6428571428571429:
Total Entropy of play Tennis Data set:
0.9402859586706309.

he plot would look like this:



EXPERIMENT - 4

Write a program to demonstrate the working of Decision tree regressor. Use appropriate dataset for Random Forest classifier decision tree regressor.

```
import pandas as pd
import matplotlib.pyplot as plt
dataset = pd.read_csv("position_salaries.csv")
x = dataset.iloc[:, 1:2] values
y = dataset.iloc[:, 2].values
regressor = DecisionTreeRegressor(random_state=0)
regressor.fit(x,y)
y_pred = regressor.predict([6.5])
x_grid = np.arrange(min(x), max(x), 0.01)
x_grid = x_grid.reshape((len(x_grid), 1))
plt.scatter(x, y, color='red')
plt.plot(x_grid, regressor.predict(x_grid), color='blue')
plt.title('Decision Tree Regression')
plt.xlabel('Position level)
plt.ylabel('salary')
plt.show()
```

EXPERIMENT - 5

Write a program to demonstrate the working of Random Forest classifier. Use appropriate dataset for Random Forest classifier.

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,
        precision_score, recall_score, ConfusionMatrixDisplay

from sklearn.model_selection import RandomizedSearchCV,
        train_test_split

from scipy.stats import randint
from sklearn.tree import export_graphviz
from Ipython.display import Image
import graphviz
bank_data['default'] = bank_data['default'].map({'no':0,
        'yes':1,'unknown:0})

bank_data['y'] = bank_data['y'].map({'no':0,'yes':1})

x = bank_data.drop('y', axis=1)
y = bank_data['y']
x_train, x_test, y_train, y_test = train_test_split(x, y,
        test_size=0.2)

rf = RandomForestClassifier()
rf.fit(x_train, y_train)
```

```
y_Pred = rf.predict(x_test)
accuracy = accuracy_score(y_test, y_Pred)
Print("Accuracy:", accuracy)
```

Output:-

Accuracy : 0.888

EXPERIMENT - 6

Write a program to demonstrate the working of logistic Regression classifier. Use appropriate dataset for logistic Regression.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot
as plt
dataset = pd.read_csv ("user_Data.csv)
x = dataset.iloc [:, [2,3]].values
y = dataset.iloc [:, 4].values
from sklearn.model_selection import train_test_split

x train, x test, y train, y test = train_test_split (x, y,
            test_size = 0.25, random_state = 0)

from sklearn.preprocessing import standard scalar.
sc_x = standardScaler()
x train = sc_x.fit_transform (x train)
x test = sc_x.transform (x test)

Print (x train [0: 10, :])
```

Output:

```
[[0.58164944.
- 0.88670699]
[-0.60673761
 1.46173768]
[-0.01254409 - 0.5677824]
```

[-0.60673761
1.89663484]
[1.37390747
-1.40858358]
[1.47293972
0.99784738]
[0. 08648817
-0.79972756]
[-0.01254409
-0.24885782]
[-0.21060859 - 0.56777284]
[-0.21060859
- 0.19087153]]