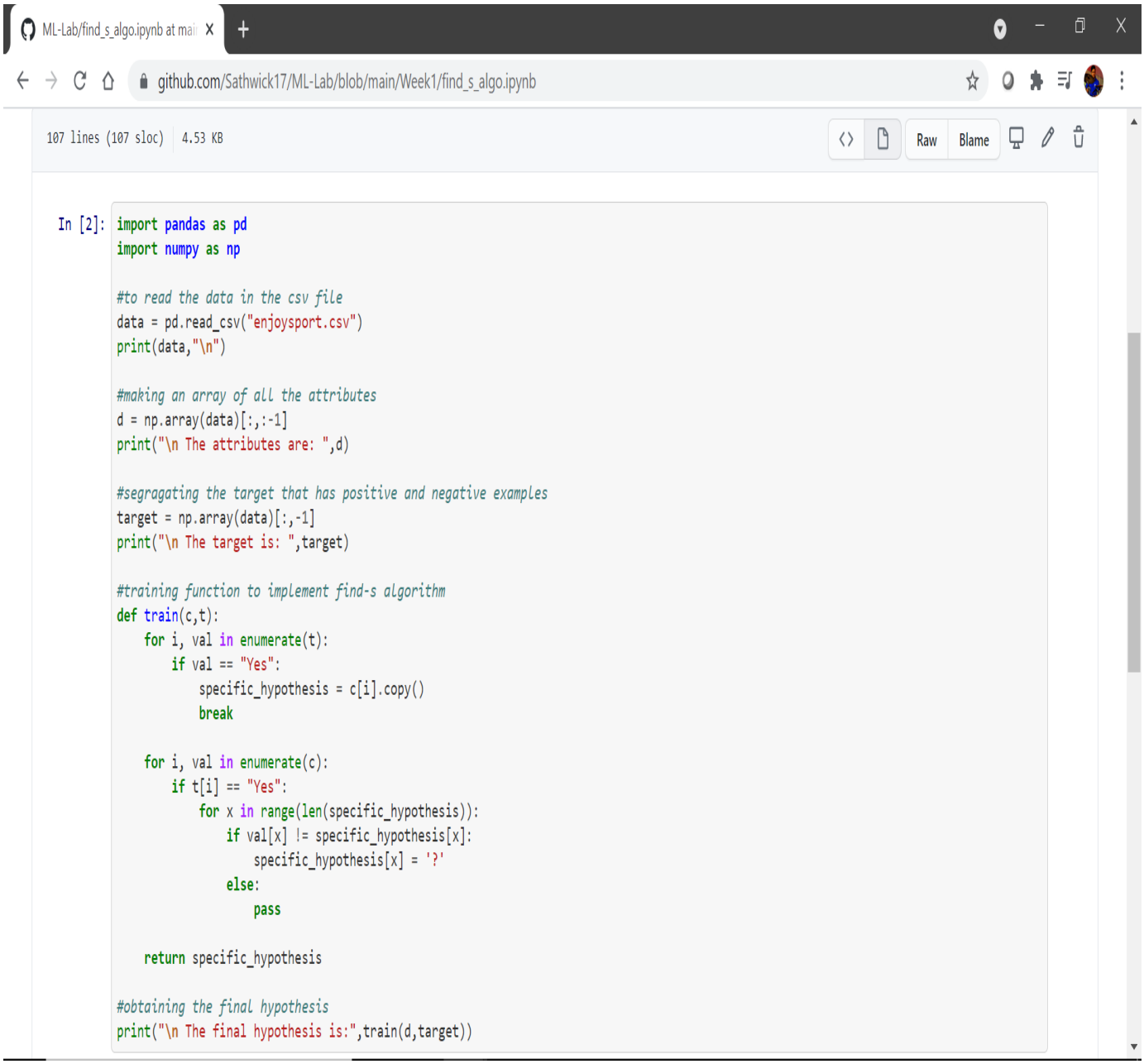
ML-Lab-01 Report

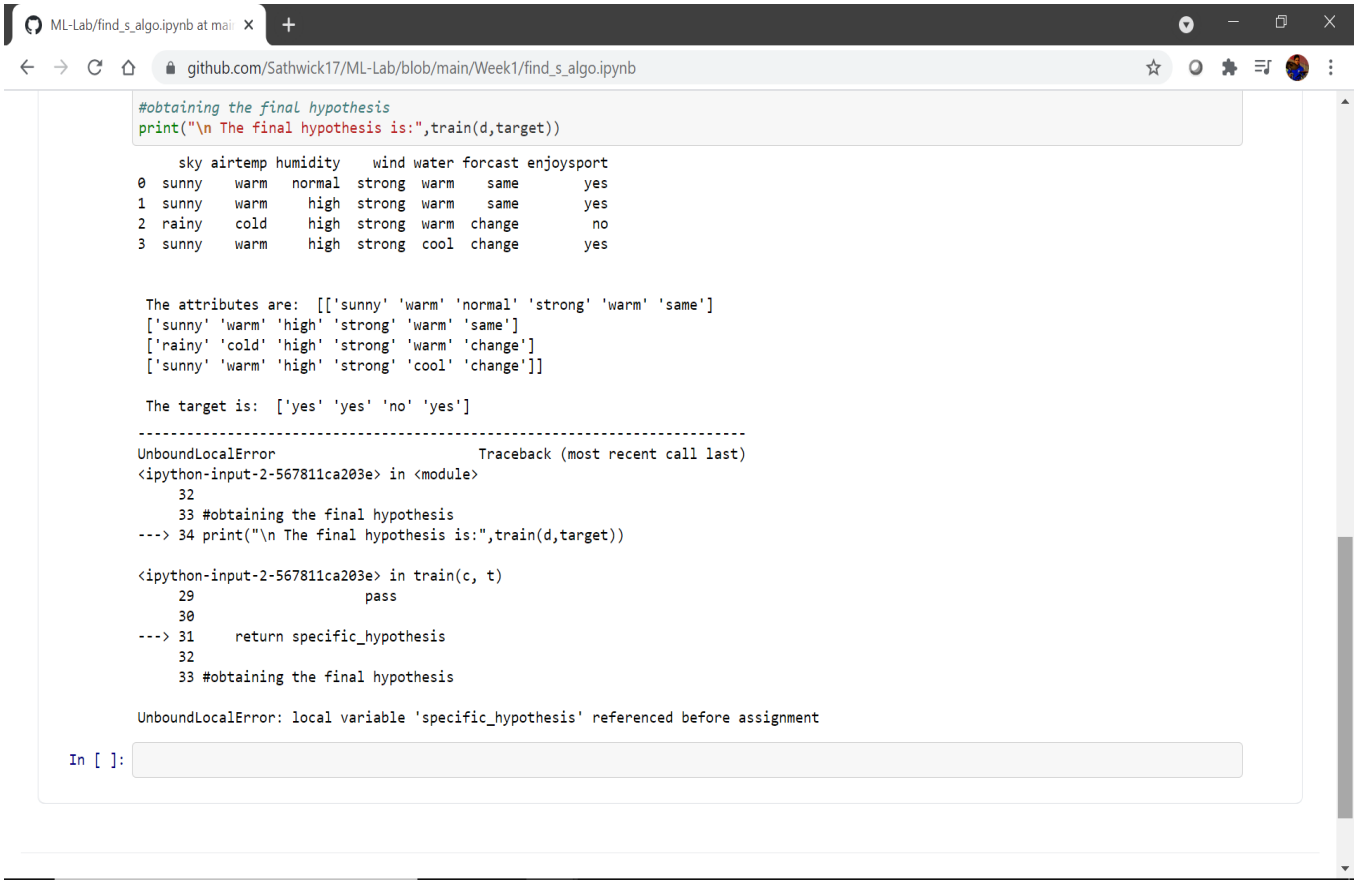
Lab-01

Program:-



In [2]:

```
import pandas as pd
import numpy as np

#to read the data in the csv file
data = pd.read_csv("enjoysport.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))
```



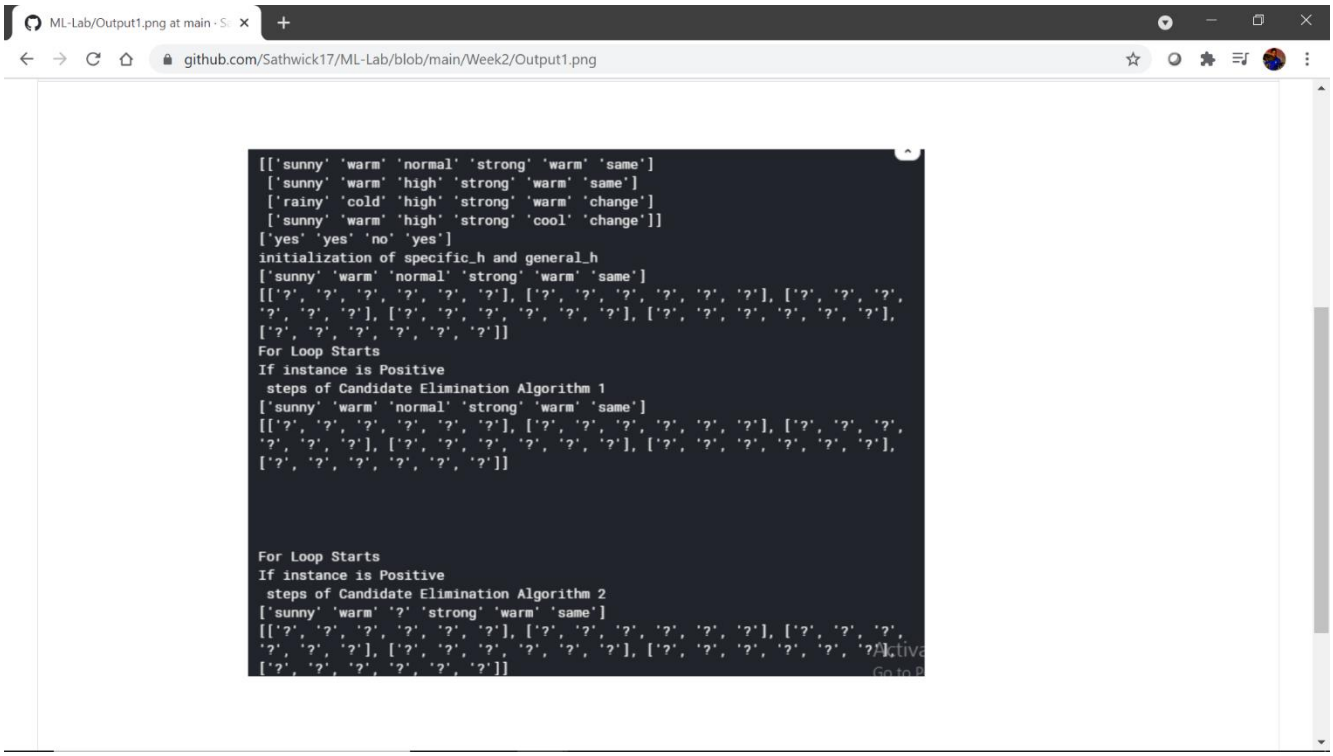
Lab-02:-

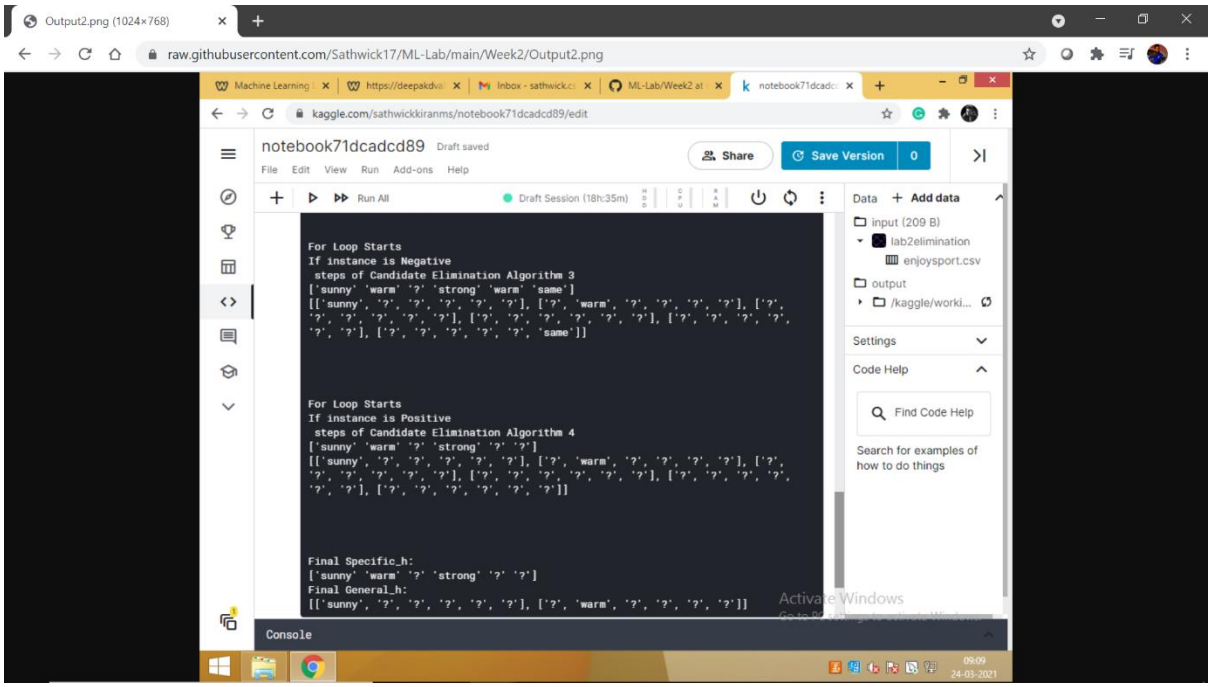
Program:-

```
import
numpy
as np
        import pandas as pd

        data = pd.read_csv('../input/lab2elimination/enjoysport.csv')
        concepts = np.array(data.iloc[:,0:-1])
        print(concepts)
        target = np.array(data.iloc[:,-1])
        print(target)
        def learn(concepts, target):
            specific_h = concepts[0].copy()
```

```
    print("initialization of specific_h and general_h")
    print(specific_h)
    general_h = [["?" for i in range(len(specific_h))] for i in
range(len(specific_h))]
    print(general_h)

    for i, h in enumerate(concepts):
        print("For Loop Starts")
        if target[i] == "yes":
            print("If instance is Positive ")
            for x in range(len(specific_h)):
                if h[x]!= specific_h[x]:
                    specific_h[x] ='?'
                    general_h[x][x] ='?'

        if target[i] == "no":
            print("If 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(" steps of Candidate Elimination Algorithm",i+1)
        print(specific_h)
        print(general_h)
        print("\n")
        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, target)

print("Final Specific_h:", s_final, sep="\n")
print("Final General_h:", g_final, sep="\n")
```

Output:-


ML-Lab/Output1.png at main
github.com/Sathwick17/ML-Lab/blob/main/Week2/Output1.png
[['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
['sunny' 'warm' 'high' 'strong' 'warm' 'same']
['rainy' 'cold' 'high' 'strong' 'warm' 'change']
['sunny' 'warm' 'high' 'strong' 'cool' 'change']]
['yes' 'yes' 'no' 'yes']
initialization of specific_h and general_h
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?']]
For Loop Starts
If instance is Positive
steps of Candidate Elimination Algorithm 1
['sunny' 'warm' 'normal' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?']]
For Loop Starts
If instance is Positive
steps of Candidate Elimination Algorithm 2
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?',
'?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'],
['?', '?', '?', '?', '?', '?']]


Output2.png (1024×768)
raw.githubusercontent.com/Sathwick17/ML-Lab/main/Week2/Output2.png
kaggle.com/sathwickkiranms/notebook71dcadcd89/edit
notebook71dcadcd89
Draft saved
File Edit View Run Add-ons Help
Share
Save Version 0
Run All
Draft Session (18h:35m)
Data
Add data
input (209 B)
lab2elimination
enjoysport.csv
output
/kaggle/worki...
Settings
Code Help
Find Code Help
Search for examples of how to do things
For Loop Starts
If instance is Negative
steps of Candidate Elimination Algorithm 3
['sunny' 'warm' '?' 'strong' 'warm' 'same']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', 'same']]
For Loop Starts
If instance is Positive
steps of Candidate Elimination Algorithm 4
['sunny' 'warm' '?' 'strong' '?' '?']
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?'], ['?',
'?', '?', '?', '?', '?'], ['?', '?', '?', '?', '?', '?'], ['?', '?', '?', '?',
'?', '?'], ['?', '?', '?', '?', '?', '?']]
Final Specific_h:
['sunny' 'warm' '?' 'strong' '?' '?']
Final General_h:
[['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?', '?']]
Console

Lab-03:-

Program and Output:-


```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "\n",
      " Input Data Set is:\n",
      "      Outlook Temperature Humidity    Wind Answer\n",
      "0      sunny         hot     high    weak     no\n",
      "1      sunny         hot     high  strong     no\n",
      "2   overcast         hot     high    weak    yes\n",
      "3       rain        mild     high    weak    yes\n",
      "4       rain        cool   normal    weak    yes\n",
      "5       rain        cool   normal  strong     no\n",
      "6   overcast        cool   normal  strong    yes\n",
      "7      sunny        mild     high    weak     no\n",
      "8      sunny        cool   normal    weak    yes\n",
      "9       rain        mild   normal    weak    yes\n",
      "10     sunny        mild   normal  strong    yes\n",
      "11  overcast        mild     high  strong    yes\n",
      "12  overcast         hot   normal    weak    yes\n",
      "13      rain        mild     high  strong     no\n"
     ]
    }
   ],
   "source": [
    "\n",
    "import pandas as pd\n",
    "df = pd.read_csv('id3.csv')\n",
    "print(\"\\n Input Data Set is:\\n\", df)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 4,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Target Attribute is:  Answer\n",
      "Predicting Attributes:  ['Outlook', 'Temperature', 'Humidity',
'Wind']\n"
```

```
     ]
    }
   ],
   "source": [
    "\n",
    "t = df.keys()[-1]\n",
    "print('Target Attribute is: ', t)\n",
    "# Get the attribute names from input dataset\n",
    "attribute_names = list(df.keys())\n",
    "#Remove the target attribute from the attribute names list\n",
    "attribute_names.remove(t) \n",
    "print('Predicting Attributes: ', attribute_names)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 5,
   "metadata": {},
   "outputs": [],
   "source": [
    "#Function to calculate the entropy of collection S\n",
    "import math\n",
    "def entropy(probs):  \n",
    "    return sum( [-prob*math.log(prob, 2) for prob in probs])\n",
    "\n",
    "#Function to calulate the entropy of the given Data Sets/List with \n\n",
    "#respect to target attributes\n",
    "def entropy_of_list(ls,value):  \n",
    "    from collections import Counter\n",
    "    cnt = Counter(x for x in ls)# Counter calculates the propotion of \nclass\n",
    "    print('Target attribute class count(Yes/No)=',dict(cnt))\n",
    "    total_instances = len(ls)  \n",
    "    print(\"Total no of instances/records associated with {0} is: \n{1}\".format(value,total_instances ))\n",
    "    probs = [x / total_instances for x in cnt.values()]  # x means no \nof YES/NO\n",
    "    print(\"Probability of Class {0} is: \n{1:.4f}\".format(min(cnt),min(probs)))\n",
    "    print(\"Probability of Class {0} is: \n{1:.4f}\".format(max(cnt),max(probs)))\n",
    "    return entropy(probs) # Call Entropy"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "metadata": {},
   "outputs": [],
   "source": [
    "def information_gain(df, split_attribute, target_attribute,battr):\n",
    "    print(\"\\n\\n-----Information Gain Calculation of \n\",split_attribute, \" --------\") \n",
    "    df_split = df.groupby(split_attribute) # group the data based on \nattribute values\n",
    "    glist=[]\n",
    "    for gname,group in df_split:\n",
    "        print('Grouped Attribute Values \\n',group)\n",
    "        glist.append(gname) \n",
    "    \n",
```

```
    "    glist.reverse()\n",
    "    nobs = len(df.index) * 1.0   \n",
    "    df_agg1=df_split.agg({target_attribute:lambda x:entropy_of_list(x,
glist.pop())})\n",
    "    df_agg2=df_split.agg({target_attribute :lambda x:len(x)/nobs})\n",
    "    \n",
    "    df_agg1.columns=['Entropy']\n",
    "    df_agg2.columns=['Proportion']\n",
    "    \n",
    "    # Calculate Information Gain:\n",
    "    new_entropy = sum( df_agg1['Entropy'] * df_agg2['Proportion'])\n",
    "    if battr !='S':\n",
    "        old_entropy = entropy_of_list(df[target_attribute],'S-
'+df.iloc[0][df.columns.get_loc(battr)])\n",
    "    else:\n",
    "        old_entropy = entropy_of_list(df[target_attribute],battr)\n",
    "    return old_entropy - new_entropy"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "metadata": {},
   "outputs": [],
   "source": [
    "def id3(df, target_attribute, attribute_names,
default_class=None,default_attr='S'):\n",
    "    \n",
    "    from collections import Counter\n",
    "    cnt = Counter(x for x in df[target_attribute])# class of YES
/NO\n",
    "    \n",
    "    ## First check: Is this split of the dataset homogeneous?\n",
    "    if len(cnt) == 1:\n",
    "        return next(iter(cnt))  # next input data set, or raises
StopIteration when EOF is hit.\n",
    "    \n",
    "    ## Second check: Is this split of the dataset empty? if yes,
return a default value\n",
    "    elif df.empty or (not attribute_names):\n",
    "        return default_class  # Return None for Empty Data Set\n",
    "    \n",
    "    ## Otherwise: This dataset is ready to be devied up!\n",
    "    else:\n",
    "        # Get Default Value for next recursive call of this
function:\n",
    "        default_class = max(cnt.keys()) #No of YES and NO Class\n",
    "        # Compute the Information Gain of the attributes:\n",
    "        gainz=[]\n",
    "        for attr in attribute_names:\n",
    "            ig= information_gain(df, attr,
target_attribute,default_attr)\n",
    "            gainz.append(ig)\n",
    "            print('Information gain of ',attr,' is : ',ig)\n",
    "        \n",
    "        index_of_max = gainz.index(max(gainz))               # Index
of Best Attribute\n",
    "        best_attr = attribute_names[index_of_max]            # Choose
Best Attribute to split on\n",
    "        print(\"\\nAttribute with the maximum gain is: \",
best_attr)\n",
```

```
    "        # Create an empty tree, to be populated in a moment\n",
    "        tree = {best_attr:{}} # Initiate the tree with best attribute
as a node \n",
    "        remaining_attribute_names =[i for i in attribute_names if i !=
best_attr]\n",
    "        \n",
    "        # Split dataset-On each split, recursively call this
algorithm.Populate the empty tree with subtrees, which\n",
    "        # are the result of the recursive call\n",
    "        for attr_val, data_subset in df.groupby(best_attr):\n",
    "            subtree = id3(data_subset,target_attribute,
remaining_attribute_names,default_class,best_attr)\n",
    "            tree[best_attr][attr_val] = subtree\n",
    "        return tree"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 8,
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "\n",
      "\n",
      "-----Information Gain Calculation of  Outlook  --------\n",
      "Grouped Attribute Values \n",
      "      Outlook Temperature Humidity    Wind Answer\n",
      "2   overcast         hot     high    weak    yes\n",
      "6   overcast        cool   normal  strong    yes\n",
      "11  overcast        mild     high  strong    yes\n",
      "12  overcast         hot   normal    weak    yes\n",
      "Grouped Attribute Values \n",
      "    Outlook Temperature Humidity    Wind Answer\n",
      "3     rain        mild     high    weak    yes\n",
      "4     rain        cool   normal    weak    yes\n",
      "5     rain        cool   normal  strong     no\n",
      "9     rain        mild   normal    weak    yes\n",
      "13    rain        mild     high  strong     no\n",
      "Grouped Attribute Values \n",
      "    Outlook Temperature Humidity    Wind Answer\n",
      "0    sunny         hot     high    weak     no\n",
      "1    sunny         hot     high  strong     no\n",
      "7    sunny        mild     high    weak     no\n",
      "8    sunny        cool   normal    weak    yes\n",
      "10   sunny        mild   normal  strong    yes\n",
      "Target attribute class count(Yes/No)= {'yes': 4}\n",
      "Total no of instances/records associated with overcast is: 4\n",
      "Probability of Class yes is: 1.0000\n",
      "Probability of Class yes is: 1.0000\n",
      "Target attribute class count(Yes/No)= {'yes': 3, 'no': 2}\n",
      "Total no of instances/records associated with rain is: 5\n",
      "Probability of Class no is: 0.4000\n",
      "Probability of Class yes is: 0.6000\n",
      "Target attribute class count(Yes/No)= {'no': 3, 'yes': 2}\n",
      "Total no of instances/records associated with sunny is: 5\n",
      "Probability of Class no is: 0.4000\n",
      "Probability of Class yes is: 0.6000\n",
      "Target attribute class count(Yes/No)= {'no': 5, 'yes': 9}\n",
```

```
"Total no of instances/records associated with S is: 14\n",
"Probability of Class no is: 0.3571\n",
"Probability of Class yes is: 0.6429\n",
"Information gain of  Outlook  is :  0.2467498197744391\n",
"\n",
"\n",
"-----Information Gain Calculation of  Temperature  --------\n",
"Grouped Attribute Values \n",
"     Outlook Temperature Humidity    Wind Answer\n",
"4      rain        cool   normal    weak    yes\n",
"5      rain        cool   normal  strong     no\n",
"6  overcast        cool   normal  strong    yes\n",
"8     sunny        cool   normal    weak    yes\n",
"Grouped Attribute Values \n",
"      Outlook Temperature Humidity    Wind Answer\n",
"0      sunny         hot     high    weak     no\n",
"1      sunny         hot     high  strong     no\n",
"2   overcast         hot     high    weak    yes\n",
"12  overcast         hot   normal    weak    yes\n",
"Grouped Attribute Values \n",
"      Outlook Temperature Humidity    Wind Answer\n",
"3       rain        mild     high    weak    yes\n",
"7      sunny        mild     high    weak     no\n",
"9       rain        mild   normal    weak    yes\n",
"10     sunny        mild   normal  strong    yes\n",
"11  overcast        mild     high  strong    yes\n",
"13      rain        mild     high  strong     no\n",
"Target attribute class count(Yes/No)= {'yes': 3, 'no': 1}\n",
"Total no of instances/records associated with cool is: 4\n",
"Probability of Class no is: 0.2500\n",
"Probability of Class yes is: 0.7500\n",
"Target attribute class count(Yes/No)= {'no': 2, 'yes': 2}\n",
"Total no of instances/records associated with hot is: 4\n",
"Probability of Class no is: 0.5000\n",
"Probability of Class yes is: 0.5000\n",
"Target attribute class count(Yes/No)= {'yes': 4, 'no': 2}\n",
"Total no of instances/records associated with mild is: 6\n",
"Probability of Class no is: 0.3333\n",
"Probability of Class yes is: 0.6667\n",
"Target attribute class count(Yes/No)= {'no': 5, 'yes': 9}\n",
"Total no of instances/records associated with S is: 14\n",
"Probability of Class no is: 0.3571\n",
"Probability of Class yes is: 0.6429\n",
"Information gain of  Temperature  is :  0.029222565658954647\n",
"\n",
"\n",
"-----Information Gain Calculation of  Humidity  --------\n",
"Grouped Attribute Values \n",
"      Outlook Temperature Humidity    Wind Answer\n",
"0      sunny         hot     high    weak     no\n",
"1      sunny         hot     high  strong     no\n",
"2   overcast         hot     high    weak    yes\n",
"3       rain        mild     high    weak    yes\n",
"7      sunny        mild     high    weak     no\n",
"11  overcast        mild     high  strong    yes\n",
"13      rain        mild     high  strong     no\n",
"Grouped Attribute Values \n",
"      Outlook Temperature Humidity    Wind Answer\n",
"4       rain        cool   normal    weak    yes\n",
"5       rain        cool   normal  strong     no\n",
"6   overcast        cool   normal  strong    yes\n",
```

```
"8      sunny        cool   normal    weak    yes\n",
"9       rain        mild   normal    weak    yes\n",
"10     sunny        mild   normal  strong    yes\n",
"12  overcast         hot   normal    weak    yes\n",
"Target attribute class count(Yes/No)= {'no': 4, 'yes': 3}\n",
"Total no of instances/records associated with high is: 7\n",
"Probability of Class no is: 0.4286\n",
"Probability of Class yes is: 0.5714\n",
"Target attribute class count(Yes/No)= {'yes': 6, 'no': 1}\n",
"Total no of instances/records associated with normal is: 7\n",
"Probability of Class no is: 0.1429\n",
"Probability of Class yes is: 0.8571\n",
"Target attribute class count(Yes/No)= {'no': 5, 'yes': 9}\n",
"Total no of instances/records associated with S is: 14\n",
"Probability of Class no is: 0.3571\n",
"Probability of Class yes is: 0.6429\n",
"Information gain of  Humidity  is :  0.15183550136234136\n",
"\n",
"\n",
"-----Information Gain Calculation of  Wind  --------\n",
"Grouped Attribute Values \n",
"      Outlook Temperature Humidity    Wind Answer\n",
"1      sunny         hot     high  strong     no\n",
"5       rain        cool   normal  strong     no\n",
"6   overcast        cool   normal  strong    yes\n",
"10     sunny        mild   normal  strong    yes\n",
"11  overcast        mild     high  strong    yes\n",
"13      rain        mild     high  strong     no\n",
"Grouped Attribute Values \n",
"      Outlook Temperature Humidity  Wind Answer\n",
"0      sunny         hot     high  weak     no\n",
"2   overcast         hot     high  weak    yes\n",
"3       rain        mild     high  weak    yes\n",
"4       rain        cool   normal  weak    yes\n",
"7      sunny        mild     high  weak     no\n",
"8      sunny        cool   normal  weak    yes\n",
"9       rain        mild   normal  weak    yes\n",
"12  overcast         hot   normal  weak    yes\n",
"Target attribute class count(Yes/No)= {'no': 3, 'yes': 3}\n",
"Total no of instances/records associated with strong is: 6\n",
"Probability of Class no is: 0.5000\n",
"Probability of Class yes is: 0.5000\n",
"Target attribute class count(Yes/No)= {'no': 2, 'yes': 6}\n",
"Total no of instances/records associated with weak is: 8\n",
"Probability of Class no is: 0.2500\n",
"Probability of Class yes is: 0.7500\n",
"Target attribute class count(Yes/No)= {'no': 5, 'yes': 9}\n",
"Total no of instances/records associated with S is: 14\n",
"Probability of Class no is: 0.3571\n",
"Probability of Class yes is: 0.6429\n",
"Information gain of  Wind  is :  0.04812703040826927\n",
"\n",
"Attribute with the maximum gain is:  Outlook\n",
"\n",
"\n",
"-----Information Gain Calculation of  Temperature  --------\n",
"Grouped Attribute Values \n",
"   Outlook Temperature Humidity    Wind Answer\n",
"4    rain        cool   normal    weak    yes\n",
"5    rain        cool   normal  strong     no\n",
"Grouped Attribute Values \n",
```

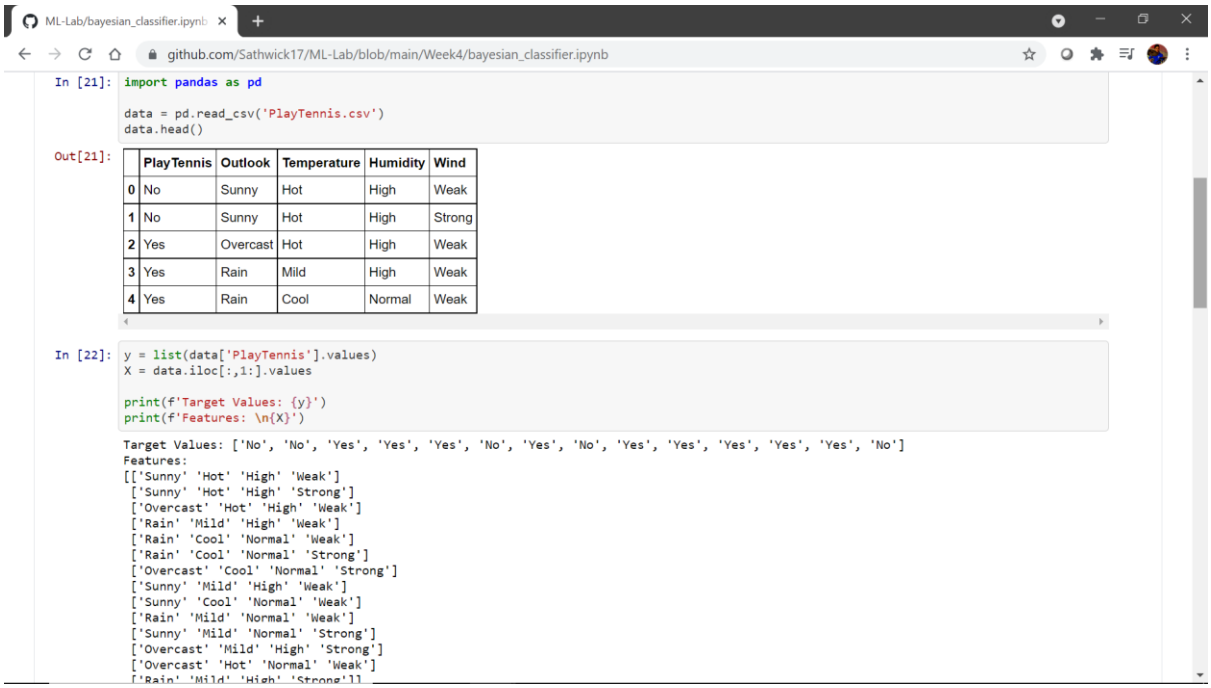
```
"    Outlook Temperature Humidity    Wind Answer\n",
"3     rain        mild     high    weak    yes\n",
"9     rain        mild   normal    weak    yes\n",
"13    rain        mild     high  strong     no\n",
"Target attribute class count(Yes/No)= {'yes': 1, 'no': 1}\n",
"Total no of instances/records associated with cool is: 2\n",
"Probability of Class no is: 0.5000\n",
"Probability of Class yes is: 0.5000\n",
"Target attribute class count(Yes/No)= {'yes': 2, 'no': 1}\n",
"Total no of instances/records associated with mild is: 3\n",
"Probability of Class no is: 0.3333\n",
"Probability of Class yes is: 0.6667\n",
"Target attribute class count(Yes/No)= {'yes': 3, 'no': 2}\n",
"Total no of instances/records associated with S-rain is: 5\n",
"Probability of Class no is: 0.4000\n",
"Probability of Class yes is: 0.6000\n",
"Information gain of  Temperature  is :  0.01997309402197489\n",
"\n",
"\n",
"-----Information Gain Calculation of  Humidity  --------\n",
"Grouped Attribute Values \n",
"    Outlook Temperature Humidity    Wind Answer\n",
"3     rain        mild     high    weak    yes\n",
"13    rain        mild     high  strong     no\n",
"Grouped Attribute Values \n",
"   Outlook Temperature Humidity    Wind Answer\n",
"4    rain        cool   normal    weak    yes\n",
"5    rain        cool   normal  strong     no\n",
"9    rain        mild   normal    weak    yes\n",
"Target attribute class count(Yes/No)= {'yes': 1, 'no': 1}\n",
"Total no of instances/records associated with high is: 2\n",
"Probability of Class no is: 0.5000\n",
"Probability of Class yes is: 0.5000\n",
"Target attribute class count(Yes/No)= {'yes': 2, 'no': 1}\n",
"Total no of instances/records associated with normal is: 3\n",
"Probability of Class no is: 0.3333\n",
"Probability of Class yes is: 0.6667\n",
"Target attribute class count(Yes/No)= {'yes': 3, 'no': 2}\n",
"Total no of instances/records associated with S-rain is: 5\n",
"Probability of Class no is: 0.4000\n",
"Probability of Class yes is: 0.6000\n",
"Information gain of  Humidity  is :  0.01997309402197489\n",
"\n",
"\n",
"-----Information Gain Calculation of  Wind  --------\n",
"Grouped Attribute Values \n",
"    Outlook Temperature Humidity    Wind Answer\n",
"5     rain        cool   normal  strong     no\n",
"13    rain        mild     high  strong     no\n",
"Grouped Attribute Values \n",
"   Outlook Temperature Humidity  Wind Answer\n",
"3    rain        mild     high  weak    yes\n",
"4    rain        cool   normal  weak    yes\n",
"9    rain        mild   normal  weak    yes\n",
"Target attribute class count(Yes/No)= {'no': 2}\n",
"Total no of instances/records associated with strong is: 2\n",
"Probability of Class no is: 1.0000\n",
"Probability of Class no is: 1.0000\n",
"Target attribute class count(Yes/No)= {'yes': 3}\n",
"Total no of instances/records associated with weak is: 3\n",
"Probability of Class yes is: 1.0000\n",
```

```
"Probability of Class yes is: 1.0000\n",
"Target attribute class count(Yes/No)= {'yes': 3, 'no': 2}\n",
"Total no of instances/records associated with S-rain is: 5\n",
"Probability of Class no is: 0.4000\n",
"Probability of Class yes is: 0.6000\n",
"Information gain of  Wind  is :  0.9709505944546686\n",
"\n",
"Attribute with the maximum gain is:  Wind\n",
"\n",
"\n",
"-----Information Gain Calculation of  Temperature  --------\n",
"Grouped Attribute Values \n",
"   Outlook Temperature Humidity  Wind Answer\n",
"8   sunny        cool   normal  weak    yes\n",
"Grouped Attribute Values \n",
"   Outlook Temperature Humidity    Wind Answer\n",
"0   sunny         hot     high    weak     no\n",
"1   sunny         hot     high  strong     no\n",
"Grouped Attribute Values \n",
"    Outlook Temperature Humidity    Wind Answer\n",
"7    sunny        mild     high    weak     no\n",
"10   sunny        mild   normal  strong    yes\n",
"Target attribute class count(Yes/No)= {'yes': 1}\n",
"Total no of instances/records associated with cool is: 1\n",
"Probability of Class yes is: 1.0000\n",
"Probability of Class yes is: 1.0000\n",
"Target attribute class count(Yes/No)= {'no': 2}\n",
"Total no of instances/records associated with hot is: 2\n",
"Probability of Class no is: 1.0000\n",
"Probability of Class no is: 1.0000\n",
"Target attribute class count(Yes/No)= {'no': 1, 'yes': 1}\n",
"Total no of instances/records associated with mild is: 2\n",
"Probability of Class no is: 0.5000\n",
"Probability of Class yes is: 0.5000\n",
"Target attribute class count(Yes/No)= {'no': 3, 'yes': 2}\n",
"Total no of instances/records associated with S-sunny is: 5\n",
"Probability of Class no is: 0.4000\n",
"Probability of Class yes is: 0.6000\n",
"Information gain of  Temperature  is :  0.5709505944546686\n",
"\n",
"\n",
"-----Information Gain Calculation of  Humidity  --------\n",
"Grouped Attribute Values \n",
"   Outlook Temperature Humidity    Wind Answer\n",
"0   sunny         hot     high    weak     no\n",
"1   sunny         hot     high  strong     no\n",
"7   sunny        mild     high    weak     no\n",
"Grouped Attribute Values \n",
"    Outlook Temperature Humidity    Wind Answer\n",
"8    sunny        cool   normal    weak    yes\n",
"10   sunny        mild   normal  strong    yes\n",
"Target attribute class count(Yes/No)= {'no': 3}\n",
"Total no of instances/records associated with high is: 3\n",
"Probability of Class no is: 1.0000\n",
"Probability of Class no is: 1.0000\n",
"Target attribute class count(Yes/No)= {'yes': 2}\n",
"Total no of instances/records associated with normal is: 2\n",
"Probability of Class yes is: 1.0000\n",
"Probability of Class yes is: 1.0000\n",
"Target attribute class count(Yes/No)= {'no': 3, 'yes': 2}\n",
"Total no of instances/records associated with S-sunny is: 5\n",
```

```
      "Probability of Class no is: 0.4000\n",
      "Probability of Class yes is: 0.6000\n",
      "Information gain of  Humidity  is :  0.9709505944546686\n",
      "\n",
      "\n",
      "-----Information Gain Calculation of  Wind  --------\n",
      "Grouped Attribute Values \n",
      "    Outlook Temperature Humidity    Wind Answer\n",
      "1    sunny         hot     high  strong     no\n",
      "10   sunny        mild   normal  strong    yes\n",
      "Grouped Attribute Values \n",
      "   Outlook Temperature Humidity  Wind Answer\n",
      "0   sunny         hot     high  weak     no\n",
      "7   sunny        mild     high  weak     no\n",
      "8   sunny        cool   normal  weak    yes\n",
      "Target attribute class count(Yes/No)= {'no': 1, 'yes': 1}\n",
      "Total no of instances/records associated with strong is: 2\n",
      "Probability of Class no is: 0.5000\n",
      "Probability of Class yes is: 0.5000\n",
      "Target attribute class count(Yes/No)= {'no': 2, 'yes': 1}\n",
      "Total no of instances/records associated with weak is: 3\n",
      "Probability of Class no is: 0.3333\n",
      "Probability of Class yes is: 0.6667\n",
      "Target attribute class count(Yes/No)= {'no': 3, 'yes': 2}\n",
      "Total no of instances/records associated with S-sunny is: 5\n",
      "Probability of Class no is: 0.4000\n",
      "Probability of Class yes is: 0.6000\n",
      "Information gain of  Wind  is :  0.01997309402197489\n",
      "\n",
      "Attribute with the maximum gain is:  Humidity\n",
      "\n",
      "The Resultant Decision Tree is:\n",
      "{'Outlook': {'overcast': 'yes',\n",
      "             'rain': {'Wind': {'strong': 'no', 'weak': 'yes'}},\n",
      "             'sunny': {'Humidity': {'high': 'no', 'normal':
'yes'}}}}\n"
     ]
    }
   ],
   "source": [
    "\n",
    "from pprint import pprint\n",
    "tree = id3(df,t,attribute_names)\n",
    "print(\"\\nThe Resultant Decision Tree is:\")\n",
    "pprint(tree)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
```

```
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.8.3"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 4
}
```

Lab-04:-



ML-Lab/bayesian_classifier.ipynb

github.com/Sathwick17/ML-Lab/blob/main/Week4/bayesian_classifier.ipynb

In [21]:
```
import pandas as pd

data = pd.read_csv('PlayTennis.csv')
data.head()
```

Out[21]:

	PlayTennis	Outlook	Temperature	Humidity	Wind
0	No	Sunny	Hot	High	Weak
1	No	Sunny	Hot	High	Strong
2	Yes	Overcast	Hot	High	Weak
3	Yes	Rain	Mild	High	Weak
4	Yes	Rain	Cool	Normal	Weak

In [22]:
```
y = list(data['PlayTennis'].values)
X = data.iloc[:,1:].values

print(f'Target Values: {y}')
print(f'Features: \n{X}')
```

```
Target Values: ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
Features:
[['Sunny' 'Hot' 'High' 'Weak']
 ['Sunny' 'Hot' 'High' 'Strong']
 ['Overcast' 'Hot' 'High' 'Weak']
 ['Rain' 'Mild' 'High' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Cool' 'Normal' 'Strong']
 ['Overcast' 'Cool' 'Normal' 'Strong']
 ['Sunny' 'Mild' 'High' 'Weak']
 ['Sunny' 'Cool' 'Normal' 'Weak']
 ['Rain' 'Mild' 'Normal' 'Weak']
 ['Sunny' 'Mild' 'Normal' 'Strong']
 ['Overcast' 'Mild' 'High' 'Strong']
 ['Overcast' 'Hot' 'Normal' 'Weak']
 ['Rain' 'Mild' 'High' 'Strong']]
```

ML-Lab/bayesian_classifier.ipynb

github.com/Sathwick17/ML-Lab/blob/main/Week4/bayesian_classifier.ipynb

In [23]:
```
y_train = y[:8]
y_val = y[8:]

X_train = X[:8]
X_val = X[8:]

print(f"Number of instances in training set: {len(X_train)}")
print(f"Number of instances in testing set: {len(X_val)}")
```

```
Number of instances in training set: 8
Number of instances in testing set: 6
```

In [24]:
```
class NaiveBayesClassifier:


    def __init__(self, X, y):

        self.X, self.y = X, y

        self.N = len(self.X)

        self.dim = len(self.X[0])

        self.attrs = [[] for _ in range(self.dim)]

        self.output_dom = {}

        self.data = []

        for i in range(len(self.X)):
            for j in range(self.dim):
                if not self.X[i][j] in self.attrs[j]:
                    self.attrs[j].append(self.X[i][j])

            if not self.y[i] in self.output_dom.keys():
                self.output_dom[self.y[i]] = 1

            else:
```

ML-Lab/bayesian_classifier.ipynb

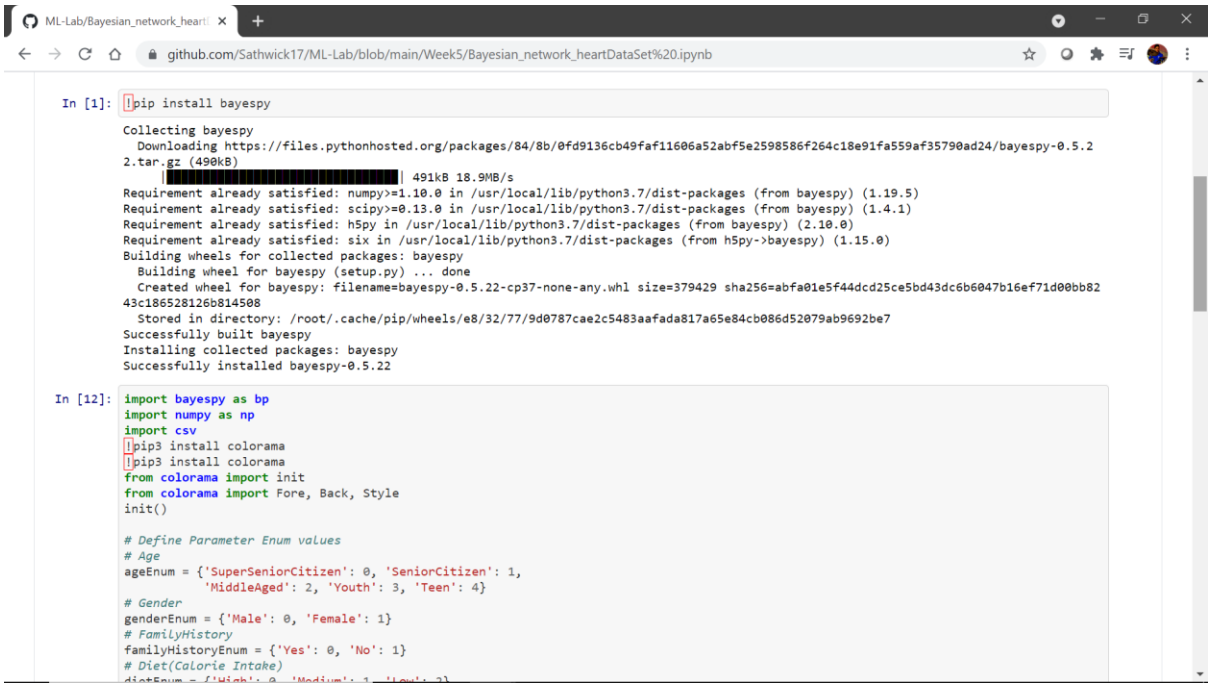
github.com/Sathwick17/ML-Lab/blob/main/Week4/bayesian_classifier.ipynb

```
                self.data.append([self.X[i], self.y[i]])
    def classify(self, entry):

        solve = None
        max_arg = -1

        for y in self.output_dom.keys():

            prob = self.output_dom[y]/self.N

            for i in range(self.dim):
                cases = [x for x in self.data if x[0][i] == entry[i] and x[1] == y]
                n = len(cases)
                prob *= n/self.N

            if prob > max_arg:
                max_arg = prob
                solve = y

        return solve
```

In [25]:
```
nbc = NaiveBayesClassifier(X_train, y_train)

total_cases = len(y_val)

good = 0
bad = 0
predictions = []

for i in range(total_cases):
    predict = nbc.classify(X_val[i])
    predictions.append(predict)

    if y_val[i] == predict:
        good += 1
    else:
        bad += 1
```


ML-Lab/bayesian_classifier.ipynb
github.com/Sathwick17/ML-Lab/blob/main/Week4/bayesian_classifier.ipynb
In [25]:
nbc = NaiveBayesClassifier(X_train, y_train)
total_cases = len(y_val)
good = 0
bad = 0
predictions = []
for i in range(total_cases):
predict = nbc.classify(X_val[i])
predictions.append(predict)
if y_val[i] == predict:
good += 1
else:
bad += 1
print('Predicted values:', predictions)
print('Actual values:', y_val)
print()
print('Total number of testing instances in the dataset:', total_cases)
print('Number of correct predictions:', good)
print('Number of wrong predictions:', bad)
print()
print('Accuracy of Bayes Classifier:', good/total_cases)
Predicted values: ['No', 'Yes', 'No', 'Yes', 'Yes', 'No']
Actual values: ['Yes', 'Yes', 'Yes', 'Yes', 'Yes', 'No']
Total number of testing instances in the dataset: 6
Number of correct predictions: 4
Number of wrong predictions: 2
Accuracy of Bayes Classifier: 0.6666666666666666

Lab-05:-


ML-Lab/Bayesian_network_heart
github.com/Sathwick17/ML-Lab/blob/main/Week5/Bayesian_network_heartDataSet%20.ipynb
In [1]:
!pip install bayespy
Collecting bayespy
Downloading https://files.pythonhosted.org/packages/84/8b/0fd9136cb49faf11606a52abf5e2598586f264c18e91fa559af35790ad24/bayespy-0.5.22.tar.gz (490kB)
491kB 18.9MB/s
Requirement already satisfied: numpy>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from bayespy) (1.19.5)
Requirement already satisfied: scipy>=0.13.0 in /usr/local/lib/python3.7/dist-packages (from bayespy) (1.4.1)
Requirement already satisfied: h5py in /usr/local/lib/python3.7/dist-packages (from bayespy) (2.10.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from h5py->bayespy) (1.15.0)
Building wheels for collected packages: bayespy
Building wheel for bayespy (setup.py) ... done
Created wheel for bayespy: filename=bayespy-0.5.22-cp37-none-any.whl size=379429 sha256=abfa01e5f44dcd25ce5bd43dc6b6047b16ef71d00bb8243c186528126b814508
Stored in directory: /root/.cache/pip/wheels/e8/32/77/9d0787cae2c5483aafada817a65e84cb086d52079ab9692be7
Successfully built bayespy
Installing collected packages: bayespy
Successfully installed bayespy-0.5.22
In [12]:
import bayespy as bp
import numpy as np
import csv
!pip3 install colorama
!pip3 install colorama
from colorama import init
from colorama import Fore, Back, Style
init()
# Define Parameter Enum values
# Age
ageEnum = {'SuperSeniorCitizen': 0, 'SeniorCitizen': 1,
'MiddleAged': 2, 'Youth': 3, 'Teen': 4}
# Gender
genderEnum = {'Male': 0, 'Female': 1}
# FamilyHistory
familyHistoryEnum = {'Yes': 0, 'No': 1}
# Diet(Calorie Intake)

```
dietEnum = {'High': 0, 'Medium': 1, 'Low': 2}
# LifeStyle
lifeStyleEnum = {'Athlete': 0, 'Active': 1, 'Moderate': 2, 'Sedetary': 3}
# Cholesterol
cholesterolEnum = {'High': 0, 'BorderLine': 1, 'Normal': 2}
# HeartDisease
heartDiseaseEnum = {'Yes': 0, 'No': 1}
```

```
Requirement already satisfied: colorama in /usr/local/lib/python3.7/dist-packages (0.4.4)
Requirement already satisfied: colorama in /usr/local/lib/python3.7/dist-packages (0.4.4)
```

In [4]:
```
import pandas as pd
```

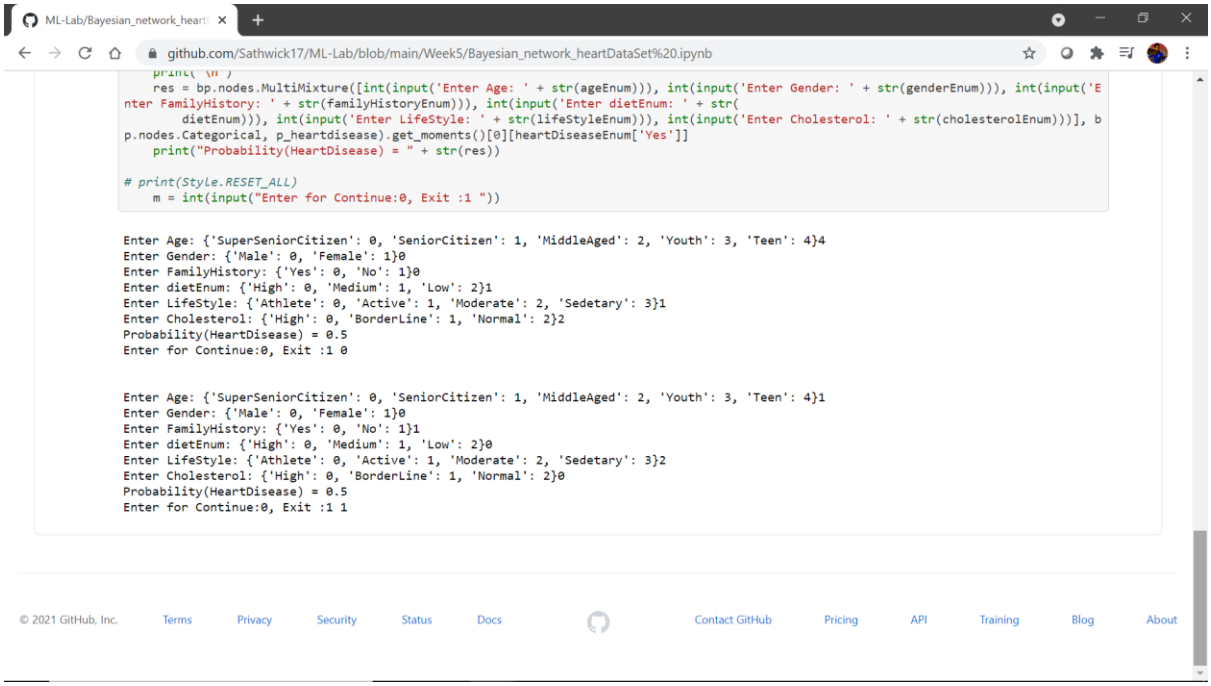
In [5]:
```
data = pd.read_csv("/content/heart_disease_data.csv")
```

In [7]:
```
data =np.array(data, dtype='int8')
N = len(data)
```

In [8]:
```
# Input data column assignment
p_age = bp.nodes.Dirichlet(1.0*np.ones(5))
age = bp.nodes.Categorical(p_age, plates=(N,))
age.observe(data[:, 0])

p_gender = bp.nodes.Dirichlet(1.0*np.ones(2))
gender = bp.nodes.Categorical(p_gender, plates=(N,))
gender.observe(data[:, 1])

p_familyhistory = bp.nodes.Dirichlet(1.0*np.ones(2))
familyhistory = bp.nodes.Categorical(p_familyhistory, plates=(N,))
familyhistory.observe(data[:, 2])

p_diet = bp.nodes.Dirichlet(1.0*np.ones(3))
diet = bp.nodes.Categorical(p_diet, plates=(N,))
diet.observe(data[:, 3])

p_lifestyle = bp.nodes.Dirichlet(1.0*np.ones(4))
lifestyle = bp.nodes.Categorical(p_lifestyle, plates=(N,))
lifestyle.observe(data[:, 4])

p_cholesterol = bp.nodes.Dirichlet(1.0*np.ones(3))
cholesterol = bp.nodes.Categorical(p_cholesterol, plates=(N,))
cholesterol.observe(data[:, 5])
```

In [9]:
```
# Prepare nodes and establish edges
# np.ones(2) -> HeartDisease has 2 options Yes/No
# plates(5, 2, 2, 3, 4, 3) -> corresponds to options present for domain values
p_heartdisease = bp.nodes.Dirichlet(np.ones(2), plates=(5, 2, 2, 3, 4, 3))
heartdisease = bp.nodes.MultiMixture(
    [age, gender, familyhistory, diet, lifestyle, cholesterol], bp.nodes.Categorical, p_heartdisease)
heartdisease.observe(data[:, 6])
p_heartdisease.update()
```

In [11]:
```
#print("Sample Probability")
#print("Probability(HeartDisease|Age=SuperSeniorCitizen, Gender=Female, FamilyHistory=Yes, DietIntake=Medium, LifeStyle=Sedetary, Cholesterol=High)")
#print(bp.nodes.MultiMixture([ageEnum['SuperSeniorCitizen'], genderEnum['Female'], familyHistoryEnum['Yes'], dietEnum['Medium'], lifeStyleEnum['Sedetary'], choLesterolEnum['High']], bp.nodes.Categorical, p_heartdisease).get_moments()[0] [heartDiseaseEnum['Yes']])

# Interactive Test
m = 0
while m == 0:
    print("\n")
    res = bp.nodes.MultiMixture([int(input('Enter Age: ' + str(ageEnum))), int(input('Enter Gender: ' + str(genderEnum))), int(input('Enter FamilyHistory: ' + str(familyHistoryEnum))), int(input('Enter dietEnum: ' + str(
        dietEnum))), int(input('Enter LifeStyle: ' + str(lifeStyleEnum))), int(input('Enter Cholesterol: ' + str(cholesterolEnum)))], bp.nodes.Categorical, p_heartdisease).get_moments()[0][heartDiseaseEnum['Yes']]
    print("Probability(HeartDisease) = " + str(res))

# print(Style.RESET_ALL)
    m = int(input("Enter for Continue:0, Exit :1 "))
```



```
    print('\n')
    res = bp.nodes.MultiMixture([int(input('Enter Age: ' + str(ageEnum))), int(input('Enter Gender: ' + str(genderEnum))), int(input('E
nter FamilyHistory: ' + str(familyHistoryEnum))), int(input('Enter dietEnum: ' + str(
        dietEnum))), int(input('Enter LifeStyle: ' + str(lifeStyleEnum))), int(input('Enter Cholesterol: ' + str(cholesterolEnum)))], b
p.nodes.Categorical, p_heartdisease).get_moments()[0][heartDiseaseEnum['Yes']]
    print("Probability(HeartDisease) = " + str(res))

# print(Style.RESET_ALL)
    m = int(input("Enter for Continue:0, Exit :1 "))
```

```
Enter Age: {'SuperSeniorCitizen': 0, 'SeniorCitizen': 1, 'MiddleAged': 2, 'Youth': 3, 'Teen': 4}4
Enter Gender: {'Male': 0, 'Female': 1}0
Enter FamilyHistory: {'Yes': 0, 'No': 1}0
Enter dietEnum: {'High': 0, 'Medium': 1, 'Low': 2}1
Enter LifeStyle: {'Athlete': 0, 'Active': 1, 'Moderate': 2, 'Sedetary': 3}1
Enter Cholesterol: {'High': 0, 'BorderLine': 1, 'Normal': 2}2
Probability(HeartDisease) = 0.5
Enter for Continue:0, Exit :1 0


Enter Age: {'SuperSeniorCitizen': 0, 'SeniorCitizen': 1, 'MiddleAged': 2, 'Youth': 3, 'Teen': 4}1
Enter Gender: {'Male': 0, 'Female': 1}0
Enter FamilyHistory: {'Yes': 0, 'No': 1}1
Enter dietEnum: {'High': 0, 'Medium': 1, 'Low': 2}0
Enter LifeStyle: {'Athlete': 0, 'Active': 1, 'Moderate': 2, 'Sedetary': 3}2
Enter Cholesterol: {'High': 0, 'BorderLine': 1, 'Normal': 2}0
Probability(HeartDisease) = 0.5
Enter for Continue:0, Exit :1 1
```