ASSIGNMENT 1

Name: Kulsoom Khurshid

Reg #: Sp20-BCS-044

Course: Machine Learning

Question 1)

Adjustment of S and G:

According to the Candidate elimination algorithm, it begins with S being the null set whereas G has the entire input space.

S = < ϕ, ϕ, ϕ… ϕ > and G = <?,?,?,… ?>

S and G are updated at each instance x as follows;

- If x is positive, we generalize by removing any $g \in G$ and expand $s \in S$ that does not have x.
- If x is negative, we specialize by removing any $s \in S$ and restrict $g \in G$ that does not have x.

There are more than one way for generalization and specialization due to which we can have multiple hypothesis in S or G.

Question 2)

Parameters:

In case of a circle, the parameters that will be considered are center and the radius of that circle.

Parameters of circle hypothesis calculation:

To get the accurate calculations of the parameters, we need to find the tightest circle which consist of all positive examples as S known as Specific hypothesis. We also need the largest circle that consist of all positive examples but not the negative ones as G known as Generic hypothesis.

Generalization for K > 2 classes:

If K > 2 classes, one circle is not enough, we need to have circles for each class. For every class C_i we need to have the hypothesis that consist of all the elements of C_i as positive examples and elements of C_j as negative examples where $j \neq i$.

Question 3)

Code:

```python
import numpy as np
import pandas as pd
import random
import csv

import os
for dirname, _, filenames in os.walk('C:/Users/sa/Desktop/6th sem/Machine Learning/ASSIGNMENTS/data.csv'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

def g_0(n):
    return ("?",)*n

def s_0(n):
    return ('0',)*n

def more_general(h1, h2):
    more_general_parts = []
    for x, y in zip(h1, h2):
        mg = x == "?" or (x != "0" and (x == y or y == "0"))
        more_general_parts.append(mg)
    return all(more_general_parts)

l1 = [1, 2, 3]
l2 = [3, 4, 5]

list(zip(l1, l2))

# min_generalizations
def fulfills(example, hypothesis):
    ### the implementation is the same as for hypotheses:
    return more_general(hypothesis, example)
```

```python
def min_generalizations(h, x):
    h_new = list(h)
    for i in range(len(h)):
        if not fulfills(x[i:i+1], h[i:i+1]):
            h_new[i] = '?' if h[i] != '0' else x[i]
    return [tuple(h_new)]

min_generalizations(h=('0', '0'  , 'sunny'),
                    x=('rainy', 'windy', 'cloudy'))

def min_specializations(h, domains, x):
    results = []
    for i in range(len(h)):
        if h[i] == "?":
            for val in domains[i]:
                if x[i] != val:
                    h_new = h[:i] + (val,) + h[i+1:]
                    results.append(h_new)
        elif h[i] != "0":
            h_new = h[:i] + ('0',) + h[i+1:]
            results.append(h_new)
    return results

min_specializations(h=('?', 'x',),
                    domains=[['a', 'b', 'c'], ['x', 'y']],
                    x=('b', 'x'))

with open('C:/Users/sa/Desktop/6th sem/Machine Learning/ASSIGNMENTS/data.csv')  as csvFile:
        examples = [tuple(line) for line in csv.reader(csvFile)]
```

```
def generalize_S(x, G, S):
    S_prev = list(S)
    for s in S_prev:
        if s not in S:
            continue
        if not fulfills(x, s):
            S.remove(s)
            Splus = min_generalizations(s, x)
            ## keep only generalizations that have a counterpart in G
            S.update([h for h in Splus if any([more_general(g,h)
                                               for g in G])])
            ## remove hypotheses less specific than any other in S
            S.difference_update([h for h in S if
                                 any([more_general(h, h1)
                                      for h1 in S if h != h1])])
    return S

def specialize_G(x, domains, G, S):
    G_prev = list(G)
    for g in G_prev:
        if g not in G:
            continue
        if fulfills(x, g):
            G.remove(g)
            Gminus = min_specializations(g, domains, x)
            ## keep only specializations that have a conuterpart in S
            G.update([h for h in Gminus if any([more_general(h, s)
                                                for s in S])])
            ## remove hypotheses less general than any other in G
            G.difference_update([h for h in G if
                                 any([more_general(g1, h)
                                      for g1 in G if h != g1])])
    return G

candidate_elimination(examples)
```

```python
examples

def get_domains(examples):
    d = [set() for i in examples[0]]
    for x in examples:
        for i, xi in enumerate(x):
            d[i].add(xi)
    return [list(sorted(x)) for x in d]

get_domains(examples)

def candidate_elimination(examples):
    domains = get_domains(examples)[:-1]

    G = set([g_0(len(domains))])
    S = set([s_0(len(domains))])
    i=0
    print("\n G[{0}]:".format(i),G)
    print("\n S[{0}]:".format(i),S)
    for xcx in examples:
        i=i+1
        x, cx = xcx[:-1], xcx[-1]  # Splitting data into attributes and decisions
        if cx=='Y': # x is positive example
            G = {g for g in G if fulfills(x, g)}
            S = generalize_S(x, G, S)
        else: # x is negative example
            S = {s for s in S if not fulfills(x, s)}
            G = specialize_G(x, domains, G, S)
        print("\n G[{0}]:".format(i),G)
        print("\n S[{0}]:".format(i),S)
    return
```

Output:

```
PS C:\Users\sa\Desktop\6th sem\Machine Learning\ASSIGNMENTS> & 'C:\Program Files\Python310\python.exe' 'c:\Users\sa\.vscode\extensions\ms-python.python-2022.14.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '58842' '--' 'c:\Users\sa\Desktop\6th sem\Machine Learning\ASSIGNMENTS\CandidateElimination.py'

 G[0]: {('?', '?', '?', '?', '?', '?')}

 S[0]: {('0', '0', '0', '0', '0', '0')}

 G[1]: {('?', '?', '?', '?', '?', '?')}

 S[1]: {('Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same')}

 G[2]: {('?', '?', '?', '?', '?', '?')}

 S[2]: {('Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same')}

 G[3]: {('?', '?', '?', '?', '?', 'Same'), ('?', 'Warm', '?', '?', '?', '?'), ('Sunny', '?', '?', '?', '?', '?')}

 S[3]: {('Sunny', 'Warm', '?', 'Strong', 'Warm', 'Same')}

 G[4]: {('?', 'Warm', '?', '?', '?', '?'), ('Sunny', '?', '?', '?', '?', '?')}

 S[4]: {('Sunny', 'Warm', '?', 'Strong', '?', '?')}
PS C:\Users\sa\Desktop\6th sem\Machine Learning\ASSIGNMENTS>
```