1. Using EconomyCar.csv dataset

Ishan Gupta - 19BCE7467 - Candidate Algorithm

```
[5] import csv
    with open('/content/EconomyCar.csv') as csvFile:
        examples = [tuple(line) for line in csv.reader(csvFile)]
    print(examples)
```

```
[('Japan ', 'Honda', 'Blue ', '1980', 'Economy', 'Yes'), ('Japan ', 'Toyota', 'Green', '1970', 'Sports', 'No'), ('Japan ', 'Toyota', 'Blue ', '1990', 'Economy', 'Yes'),
```

```
[6] 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)
```

```
[['Japan ', 'USA'],
 ['Chrysler', 'Honda', 'Toyota'],
 ['Blue ', 'Green', 'Red', 'White'],
 ['1970', '1980', '1990'],
 ['Economy', 'Sports'],
 ['No', 'Yes']]
```

```
[7] def g_0(n):
        return ('?',)*n

    def s_0(n):
        return ('Phi',)*n
    def more_general(h1, h2):
        more_general_parts = []
        for x, y in zip(h1, h2):
            mg = x == '?' or (x != 'Phi' and (x == y or y == 'Phi'))
            more_general_parts.append(mg)
        return all(more_general_parts)

    def consistent(hypothesis,example):
        return more_general(hypothesis, example)

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

    def generalize_S(x, G, S):
        S_prev = list(S)
        for s in S_prev:
            if s not in S:
                continue
```

```
[7]         if not consistent(s,x):
                S.remove(s)
                Splus = min_generalizations(s, x)
                S.update([h for h in Splus if any([more_general(g,h)
                                                   for g in G])])

                S.difference_update([h for h in S if
                                     any([more_general(h, h1)
                                          for h1 in S if h != h1])])
        return S

    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] != 'Phi':
                h_new = h[:i] + ('Phi',) + h[i+1:]
                results.append(h_new)
        return results

    def specialize_G(x, domains, G, S):
        G_prev = list(G)
        for g in G_prev:
            if g not in G:
                continue
            if consistent(g,x):
```

```
[7]             G.remove(g)
                Gminus = min_specializations(g, domains, x)
                G.update([h for h in Gminus if any([more_general(h, s)
                                                    for s in S])])
                G.difference_update([h for h in G if
                                     any([more_general(g1, h)
                                          for g1 in G if h != g1])])
        return G

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

        G = set([g_0(len(domains))])
        S = set([s_0(len(domains))])
        i=0
        print('All the hypotheses in General and Specific boundary are:\n')
        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]
            if cx=='Yes':
                G = {g for g in G if consistent(g,x)}
                S = generalize_S(x, G, S)
            else:
                S = {s for s in S if not consistent(s,x)}
                G = specialize_G(x, domains, G, S)
            print('\n G[{0}]:'.format(i),G)
            print('\n S[{0}]:'.format(i),S)
        return
[7] candidate_elimination(examples)
```

```
All the hypotheses in General and Specific boundary are:

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

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

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

 S[1]: {('Japan ', 'Honda', 'Blue ', '1980', 'Economy')}

 G[2]: {('?', '?', '?', '1980', '?'), ('?', 'Honda', '?', '?', '?'), ('?', '?', '?', '?', 'Economy'), ('?', '?', 'Blue ', '?', '?')}

 S[2]: {('Japan ', 'Honda', 'Blue ', '1980', 'Economy')}

 G[3]: {('?', '?', '?', '?', 'Economy'), ('?', '?', 'Blue ', '?', '?')}

 S[3]: {('Japan ', '?', 'Blue ', '?', 'Economy')}

 G[4]: {('Japan ', '?', '?', '?', 'Economy'), ('?', '?', 'Blue ', '?', '?')}

 S[4]: {('Japan ', '?', 'Blue ', '?', 'Economy')}

 G[5]: {('Japan ', '?', '?', '?', 'Economy')}

 S[5]: {('Japan ', '?', '?', '?', 'Economy')}
```

2. Using EnjoySport.csv dataset

Ishan Gupta - 19BCE7467 - Candidate Algorithm

```
[ ] import csv
    with open('/content/EnjoySport.csv') as csvFile:
        examples = [tuple(line) for line in csv.reader(csvFile)]
    print(examples)
```

```
[('Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes'), ('Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes'), ('Rainy', 'Cold', 'High', 'Strong', 'Warm', '
```

```
[ ] 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)
```

```
[['Rainy', 'Sunny'],
 ['Cold', 'Warm'],
 ['High', 'Normal'],
 ['Strong'],
 ['Cool', 'Warm'],
 ['Change', 'Same'],
 ['No', 'Yes']]
```

```
[7] def g_0(n):
        return ('?',)*n

    def s_0(n):
        return ('Phi',)*n
    def more_general(h1, h2):
        more_general_parts = []
        for x, y in zip(h1, h2):
            mg = x == '?' or (x != 'Phi' and (x == y or y == 'Phi'))
            more_general_parts.append(mg)
        return all(more_general_parts)

    def consistent(hypothesis,example):
        return more_general(hypothesis, example)

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

    def generalize_S(x, G, S):
        S_prev = list(S)
        for s in S_prev:
            if s not in S:
                continue
```

```
[7]         if not consistent(s,x):
                S.remove(s)
                Splus = min_generalizations(s, x)
                S.update([h for h in Splus if any([more_general(g,h)
                                                   for g in G])])

                S.difference_update([h for h in S if
                                     any([more_general(h, h1)
                                          for h1 in S if h != h1])])
        return S

    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] != 'Phi':
                h_new = h[:i] + ('Phi',) + h[i+1:]
                results.append(h_new)
        return results

    def specialize_G(x, domains, G, S):
        G_prev = list(G)
        for g in G_prev:
            if g not in G:
                continue
            if consistent(g,x):
```

```
[7]             G.remove(g)
                Gminus = min_specializations(g, domains, x)
                G.update([h for h in Gminus if any([more_general(h, s)
                                                    for s in S])])
                G.difference_update([h for h in G if
                                     any([more_general(g1, h)
                                          for g1 in G if h != g1])])
        return G

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

        G = set([g_0(len(domains))])
        S = set([s_0(len(domains))])
        i=0
        print('All the hypotheses in General and Specific boundary are:\n')
        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]
            if cx=='Yes':
                G = {g for g in G if consistent(g,x)}
                S = generalize_S(x, G, S)
            else:
                S = {s for s in S if not consistent(s,x)}
                G = specialize_G(x, domains, G, S)
            print('\n G[{0}]:'.format(i),G)
            print('\n S[{0}]:'.format(i),S)
        return
```

```
candidate_elimination(examples)
```

```
All the hypotheses in General and Specific boundary are:


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

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

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

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

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

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

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

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

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

 S[4]: {('Sunny', 'Warm', '?', 'Strong', '?', '?')}
```