

Co 1 :Unit 1

1.Implement Concept Learning using Find-S Algorithm
program code

```
data = [['Sunny','Warm'], ['Sunny','Hot'], ['Rainy','Cold']]
```

```
target = ['Yes', 'Yes', 'No']
```

```
hypo = data[0].copy() # Start with the first example
```

```
for i in range(len(data)):
```

```
    if target[i] == 'Yes': # Only look at positive cases
```

```
        for j in range(len(hypo)):
```

```
            if hypo[j] != data[i][j]: hypo[j] = '?' # Generalize if  
different
```

```
print(hypo) #
```

output

```
... ['Sunny', '?']
```

2. Implement Candidate Elimination Algorithm

program code

```
S, G = ['Sunny','Warm'], [['?','?']] # Initial Specific and General  
guesses
```

```
data, target = [['Sunny','Warm'], ['Rainy','Cold']], ['Yes', 'No']
```

```

for i, row in enumerate(data):
    if target[i] == 'Yes': # If Yes: Make S more general (like
Find-S)
        S = [S[j] if S[j] == row[j] else '?' for j in range(len(S))]
    else: # If No: Make G more specific to exclude this bad
example
        G = [[S[j] if j==k else '?' for j in range(len(S))] for k in
range(len(S)) if row[k] != S[k]]
print("S:", S, "\nG:", G)

```

output

```

*** S: ['Sunny', 'Warm']
      G: [['Sunny', '?'], ['?', 'Warm']]

```

3.Study Inductive Bias in Learning Algorithms

program code

```

# A new situation the computer hasn't seen: Sunny but Cold.
new_case = ['Sunny', 'Cold']

```

```

# Find-S Bias: Only cares about its one specific rule
find_s_rule = ['Sunny', 'Warm']
result_fs = "Yes" if all(find_s_rule[i] == '?' or find_s_rule[i] ==
new_case[i] for i in range(2)) else "No"

```

```
# Candidate Elimination Bias: Only says "Yes" if the case fits  
WITHIN the boundaries
```

```
S, G = ['Sunny', 'Warm'], [['Sunny', '?'], ['?', 'Warm']]
```

```
fits_S = all(S[i] == '?' or S[i] == new_case[i] for i in range(2))
```

```
fits_G = any(all(g[i] == '?' or g[i] == new_case[i] for i in  
range(2)) for g in G)
```

```
print(f"Find-S says: {result_fs}") # Output: No (too strict)
```

```
print(f"Candidate Elimination says: {'Yes' if fits_S or fits_G else  
'No'}") #
```

output

```
** Find-S says: No  
Candidate Elimination says: Yes
```

4. Decision Tree Learning using ID3

program code

```
from sklearn.tree import DecisionTreeClassifier, export_text  
import pandas as pd
```

```
# 1. Simple Data: [Temp, Humidity] (0=Cold/Normal,  
1=Hot/High)
```

```
X = [[1, 1], [1, 0], [0, 1], [0, 0]]
```

```
y = ['No', 'Yes', 'No', 'Yes'] # Target: Enjoy Sport?
```

```
# 2. Initialize ID3 (using 'entropy' as the criterion)
```

```
clf = DecisionTreeClassifier(criterion='entropy')
```

```
clf.fit(X, y)
```

```
# 3. Show the "Tree" logic
print(export_text(clf, feature_names=['Temp', 'Humidity']))
```

Output

```
...
... |--- Humidity <= 0.50
|   |--- class: Yes
|--- Humidity >  0.50
|   |--- class: No
```

5. Heuristic Search for Hypothesis Space

program code

```
import random
```

```
X, y = [['Sunny','Warm'], ['Rainy','Cold'], ['Sunny','Cold']], [1, 0, 1]
```

```
h = ['Sunny', 'Warm']
```

```
for _ in range(100):
```

```
    new_h = [random.choice([h[i], '?']) for i in range(2)]
```

```
# Score: How many predictions match the target 'y'
```

```
score = lambda hyp: sum(all(hyp[j] in [X[i][j], '?'] for j in range(2)) == y[i] for i in range(3))
```

```
    if score(new_h) > score(h): h = new_h
```

```
print("Final Rule:", h)
```

output

```
.. Final Rule: ['Sunny', '?']
```
