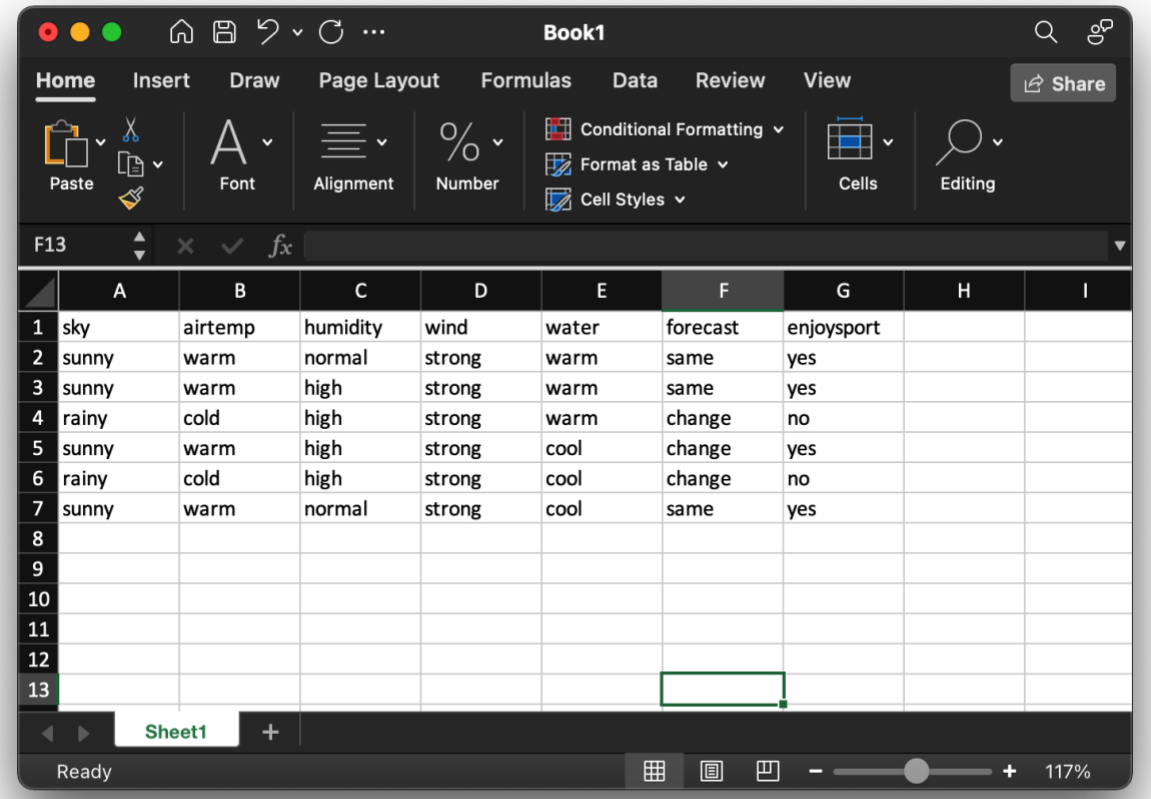
CSE4020

Lab-1

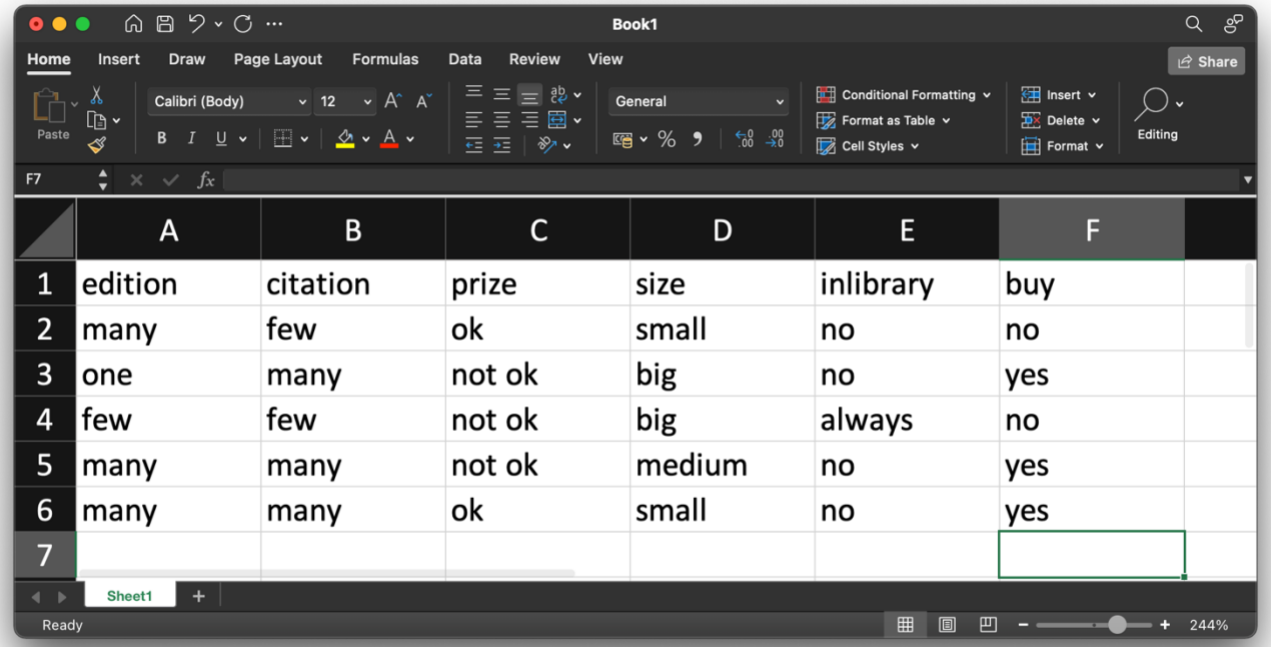
Routu Venkat Sai Sampath
19BCE0238

Data

The data used for the experiments will be attached below.



	A	B	C	D	E	F	G
1	sky	airtemp	humidity	wind	water	forecast	enjoysport
2	sunny	warm	normal	strong	warm	same	yes
3	sunny	warm	high	strong	warm	same	yes
4	rainy	cold	high	strong	warm	change	no
5	sunny	warm	high	strong	cool	change	yes
6	rainy	cold	high	strong	cool	change	no
7	sunny	warm	normal	strong	cool	same	yes



	A	B	C	D	E	F
1	edition	citation	prize	size	inlibrary	buy
2	many	few	ok	small	no	no
3	one	many	not ok	big	no	yes
4	few	few	not ok	big	always	no
5	many	many	not ok	medium	no	yes
6	many	many	ok	small	no	yes
7						

Experiment 1A

Aim: Implement Find S algorithm

Procedure:

1. Get the data ready
2. Form a general hypothesis as [“Φ”…]
3. Since Φ will not be matched with any row, put the first row as the general hypothesis
4. Now compare the general hypothesis to each positive row of the dataset.
5. If any column in the hypothesis does not match with the data on the row, replace it with “?”.
6. Finally, we will obtain the general hypothesis

<u>Code, Output and Results</u>

In [1]:
```
import pandas as pd
```

In [2]:
```
data = pd.read_csv("data_lab_1.csv")
```

In [3]:
```
data.head(10)
```

Out[3]:

	sky	airtemp	humidity	wind	water	forecast	enjoysport
0	sunny	warm	normal	strong	warm	same	yes
1	sunny	warm	high	strong	warm	same	yes
2	rainy	cold	high	strong	warm	change	no
3	sunny	warm	high	strong	cool	change	yes
4	rainy	cold	high	strong	cool	change	no
5	sunny	warm	normal	strong	cool	same	yes

In [4]:

```
from functools import reduce

def find_dataset_info(data):
    ncol = data.shape[1] - 1
    uniqueValues = list(map(lambda x: len(data[x].unique()),data))[:-1]
    nInstances = reduce((lambda x, y: x * y), uniqueValues)

    uniqueValues = list(map(lambda x: x+2, uniqueValues))
    print(uniqueValues)
    nSyntactial = reduce((lambda x, y: x * y), uniqueValues)

    uniqueValues = list(map(lambda x: x-1, uniqueValues))
    nSemantic = 1 + reduce((lambda x, y: (x) * (y)), uniqueValues)

    print("Number of Instances = " + str(nInstances))
    print("Number of Syntactial = " + str(nSyntactial))
    print("Number of Semantic = " + str(nSemantic))

    return [nInstances, nSyntactial, nSemantic]

def list_to_string(ln):
    temp = ""
    for x in ln:
        temp+=x
        temp+=" "
    return temp

def findS_algorithm(data, positive_value = "yes"):
    hypothesis = ["Φ" for _ in range(data.shape[1]-1)]
    columns = data.columns
    print("Initial value of hypothesis: "+ list_to_string(hypothesis))
    first_value = True
    i = 0
    while(i < data.shape[0]):
        if(data.iloc[i,-1] != positive_value):
            i+=1
            continue
        value = list(data.iloc[i,:-1])
        if(first_value):
            hypothesis = list(data.iloc[i,:-1])
            first_value = False
        else:
            for item in range(len(value)):
                if(value[item] != hypothesis[item]):
                    hypothesis[item] = "?"
        print("Hypothesis value after data example "+ str(i+1)+": "+list
_to_string(hypothesis))
        i+=1
    return hypothesis
```

In [7]:
```
find_dataset_info(data)
```

```
[4, 4, 4, 3, 4, 4]
Number of Instances = 32
Number of Syntactial = 3072
Number of Semantic = 487
```

Out[7]:
```
[32, 3072, 487]
```

In [8]:
```
findS_algorithm(data)
```

```
Initial value of hypothesis: Φ Φ Φ Φ Φ Φ
Hypothesis value after data example 1: sunny warm normal strong warm sa
me
Hypothesis value after data example 2: sunny warm ? strong warm same
Hypothesis value after data example 4: sunny warm ? strong ? ?
Hypothesis value after data example 6: sunny warm ? strong ? ?
```

Out[8]:
```
['sunny', 'warm', '?', 'strong', '?', '?']
```

Experiment 1B

Aim: Implement list then eliminate algorithm

Procedure:

1. Get the data ready
2. Get the hypothesis ready
3. Now run the function of item eliminate on the hypothesis
    a. First each hypothesis will get iterated over the data to find matches with same target value
    b. After matching, the hypothesis will be compared with values from the dataset.
    c. The hypothesis will be segregated into consistent and inconsistent sets.
4. The function will return the inconsistent hypothesis set first and then the consistent set.

<u>Code, Output and Results</u>

In [1]:
```
import pandas as pd
```

In [2]:
```
data = pd.read_csv("books.csv")
```

In [3]:
```
data.head(10)
```

Out[3]:

	edition	citation	prize	size	inlibrary	buy
0	many	few	ok	small	no	no
1	one	many	not ok	big	no	yes
2	few	few	not ok	big	always	no
3	many	many	not ok	medium	no	yes
4	many	many	ok	small	no	yes

In [4]:
```
def package_hypothesis(hypothesis, outcome):
    ln = dict()
    ln['hypothesis'] = hypothesis
    ln['outcome'] = outcome
    return ln

#Test hypothesises
h1 = package_hypothesis(["?","?","ok","?","?"],"no")
h2 = package_hypothesis(["few","few","?","?","?"],"no")
h3 = package_hypothesis(["many","?","ok","?","?"],"yes")
h4 = package_hypothesis(["many","few","not ok","?","?"],"yes")
h5 = package_hypothesis(["?","many","?","medium","?"],"yes")
h6 = package_hypothesis(["?","?","?","always","?"],"no")
h7 = package_hypothesis(["?","many","?","?","?"],"yes")
h8 = package_hypothesis(["?","few","?","?","?"],"no")
```

In [5]:
```
def compare(values, hypo):
    for i in range(len(values)):
        if(hypo[i] != "?"):
            if(values[i] != hypo[i]):
                return False
    return True

def list_then_eliminate(data, *hypothesis):
    consistent_space = []
    inconsistent_space = []

    for hyp in hypothesis:
        state = True
        for i in range(data.shape[0]):
            if(hyp['outcome'] == data.iloc[i,-1]):
                if(not compare(hypo = hyp['hypothesis'], values = list(d
ata.iloc[i,:-1])[:-1])):
                    inconsistent_space.append(hyp)
                    state = False
                    break
        if(state):
            consistent_space.append(hyp)

    return (inconsistent_space, consistent_space)
```

In [6]:
```
list_then_eliminate(data, h1,h2,h3,h4,h5,h6,h7,h8)
```

Out[6]:
```
([{'hypothesis': ['?', '?', 'ok', '?', '?'], 'outcome': 'no'},
  {'hypothesis': ['few', 'few', '?', '?', '?'], 'outcome': 'no'},
  {'hypothesis': ['many', '?', 'ok', '?', '?'], 'outcome': 'yes'},
  {'hypothesis': ['many', 'few', 'not ok', '?', '?'], 'outcome': 'ye
s'},
  {'hypothesis': ['?', 'many', '?', 'medium', '?'], 'outcome': 'yes'},
  {'hypothesis': ['?', '?', '?', 'always', '?'], 'outcome': 'no'}],
 [{'hypothesis': ['?', 'many', '?', '?', '?'], 'outcome': 'yes'},
  {'hypothesis': ['?', 'few', '?', '?', '?'], 'outcome': 'no'}])
```

In [ ]:

Experiment 1C

Aim: Implement Candidate algorithm

Procedure:

1. Get the data ready
2. Get the hypothesis ready
3. Iterate over the values of the dataset
    a. If the target variable is positive then iterate over the specific hypothesis such that if any inconsistency is found between the values and specific hyp then change the hypothesis value to “?” on both general and specific.
    b. If the target is negative then iterate over the specific hypothesis such that if any inconsistency is found between the values and specific hyp then change the general hypothesis value to specific hypothesis. Else change the general hypothesis value to “?”.
4. Print the general hypothesis bounds and the specific hypothesis.

Code, Output and Results

In [1]:
```
import pandas as pd
import numpy as np

data = pd.read_csv("data_lab_1.csv")
values = np.array(data.iloc[:,0:-1])
target = np.array(data.iloc[:,-1])
```

In [2]:
```
def candidate_algorithm(values, target):
    specific_hyp = values[0].copy()
    general_hyp = [["?" for i in range(len(specific_hyp))] for i in rang
e(len(specific_hyp))]

    for i, h in enumerate(values):
        if target[i] == "yes":
            for x in range(len(specific_hyp)):
                if h[x]!= specific_hyp[x]:
                    specific_hyp[x] ='?'
                    general_hyp[x][x] ='?'

        if target[i] == "no":
            for x in range(len(specific_hyp)):
                if h[x]!= specific_hyp[x]:
                    general_hyp[x][x] = specific_hyp[x]
                else:
                    general_hyp[x][x] = '?'

    return specific_hyp, [val for val in general_hyp if val != ['?', '?'
, '?', '?', '?', '?']]
```

In [4]:
```
candidate_algorithm(values, target)
```

Out[4]:
```
(array(['sunny', 'warm', '?', 'strong', '?', '?'], dtype=object),
 [['sunny', '?', '?', '?', '?', '?'], ['?', 'warm', '?', '?', '?',
'?']])
```

All the code is original and can be verified from the following repository
https://github.com/nyac-1/cse4020-ML