```
from google.colab import files
uploaded =files.upload()
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving dataset.csv to dataset.csv
```

```
import pandas as pd
import numpy as np


df=pd.read_csv('dataset.csv')
```

```
df
```

| | sky | temp | humidity | wind | isplay |
|---|---|---|---|---|---|
| **0** | sunny | warm | high | strong | yes |
| **1** | sunny | warm | normal | strong | yes |
| **2** | rainy | cold | high | strong | no |
| **3** | sunny | warm | high | less | yes |

```
df.describe()
```

| | sky | temp | humidity | wind | isplay |
|---|---|---|---|---|---|
| **count** | 4 | 4 | 4 | 4 | 4 |
| **unique** | 2 | 2 | 2 | 2 | 2 |
| **top** | sunny | warm | high | strong | yes |
| **freq** | 3 | 3 | 3 | 3 | 3 |

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   sky       4 non-null      object
 1   temp      4 non-null      object
 2   humidity  4 non-null      object
 3   wind      4 non-null      object
 4   isplay    4 non-null      object
dtypes: object(5)
memory usage: 292.0+ bytes
```

```
df.shape
```

```
(4, 5)
```

```
y=df["isplay"]


X=df.drop(columns=["isplay"])
```

```
X
```

| | sky | temp | humidity | wind |
|---|---|---|---|---|
| **0** | sunny | warm | high | strong |
| **1** | sunny | warm | normal | strong |
| **2** | rainy | cold | high | strong |
| **3** | sunny | warm | high | less |

y

| | isplay |
|---|---|
| **0** | yes |
| **1** | yes |
| **2** | no |
| **3** | yes |

**dtype:** object

type(X)

```
pandas.core.frame.DataFrame
def __init__(data=None, index: Axes | None=None, columns: Axes | None=None, dtype: Dtype |
None=None, copy: bool | None=None) -> None
```

```
Two-dimensional, size-mutable, potentially heterogeneous tabular data.

Data structure also contains labeled axes (rows and columns).
Arithmetic operations align on both row and column labels. Can be
thought of as a dict-like container for Series objects. The primary
pandas data structure.
```

type(y)

```
pandas.core.series.Series
def __init__(data=None, index=None, dtype: Dtype | None=None, name=None, copy: bool | None=None,
fastpath: bool | lib.NoDefault=lib.no_default) -> None
```

```
One-dimensional ndarray with axis labels (including time series).

Labels need not be unique but must be a hashable type. The object
supports both integer- and label-based indexing and provides a host of
methods for performing operations involving the index. Statistical
methods from ndarray have been overridden to automatically exclude
```

```
X=np.array(X)
```

```
y=np.array(y)
```

```
def candidate_elimination(X, y):
    # Initialize the specific hypothesis (S) and general hypothesis (G)
    num_attributes = len(X[0])
    S = ['φ'] * num_attributes  # Most specific hypothesis
    G = [['?'] * num_attributes]  # Most general hypothesis

    # Iterate through the dataset
    for i, instance in enumerate(X):
        if y[i] == 'yes':  # Positive example
            # Update S to be consistent with the instance
            for j in range(num_attributes):
                if S[j] == 'φ':  # Replace 'φ' with the attribute value
                    S[j] = instance[j]
                elif S[j] != instance[j]:  # Generalize to '?'
                    S[j] = '?'

            # Remove inconsistent hypotheses from G
            G = [g for g in G if all(
                g[k] == '?' or S[k] == '?' or g[k] == S[k]
                for k in range(num_attributes)
            )]
```

```
        elif y[i] == 'no':  # Negative example
            # Specialize G to exclude the negative instance
            new_G = []
            for g in G:
                for j in range(num_attributes):
                    if g[j] == '?':  # Specialize this attribute
                        if S[j] != '?':
                            new_hypothesis = g.copy()
                            new_hypothesis[j] = S[j]
                            new_G.append(new_hypothesis)
            G = new_G

            # Remove inconsistent hypotheses from G
            G = [g for g in G if any(
                g[k] != instance[k] and g[k] != '?' for k in range(num_attributes)
            )]

    return S, G



S_final, G_final = candidate_elimination(X, y)

# Output results
print("Final Specific Hypothesis:", S_final)
print("Final General Hypothesis:", G_final)
```

```
Final Specific Hypothesis: ['sunny', 'warm', '?', '?']
Final General Hypothesis: [['sunny', '?', '?', '?'], ['?', 'warm', '?', '?']]
```