Name - **Adarsh Upadhyay** Reg no.- **23MCA0237**

```
def candidate_elimination(examples, treat_yes_as):
    specific_h = examples[0][:-1]
    general_h = [['?' for _ in range(len(specific_h))] for _ in range(len(specific_h))]

    for example in examples:
        if example[-1] == treat_yes_as:
            for i in range(len(specific_h)):
                if example[i] != specific_h[i]:
                    specific_h[i] = '?'
                    general_h[i][i] = '?'
        else:
            for i in range(len(specific_h)):
                if example[i] != specific_h[i]:
                    general_h[i][i] = specific_h[i]
                else:
                    general_h[i][i] = '?'
    return specific_h, general_h


dataset = [
 ['Overcast', 'Hot', 'High', 'False', 'Yes'],
 ['Rainy', 'Mild', 'High', 'False', 'Yes'],
 ['Rainy', 'Cool', 'Normal', 'False', 'Yes'],
 ['Rainy', 'Cool', 'Normal', 'True', 'No'],
 ['Overcast', 'Cool', 'Normal', 'True', 'Yes'],
 ['Sunny', 'Mild', 'High', 'False', 'No'],
 ['Sunny', 'Cool', 'Normal', 'False', 'Yes'],
 ['Rainy', 'Mild', 'Normal', 'False', 'Yes'],
 ['Sunny', 'Mild', 'Normal', 'True', 'Yes'],
 ['Rainy', 'Mild', 'High', 'True', 'No']
]


#Result origin
specific_yes, general_yes = candidate_elimination(dataset, 'Yes')


#Result toggle
specific_no, general_no = candidate_elimination(dataset, 'No')


count_specific_yes = 0
for i in specific_yes:
    if(i!='?'):
        count_specific_yes+=1


count_specific_no = 0
for i in specific_no:
    if(i!='?'):
        count_specific_no+=1


if(count_specific_yes >= count_specific_no):
    print("Specific Hypothesis:", specific_yes)
    print("General Hypothesis:")
    for hypothesis in general_yes:
        print(hypothesis)
else:
    print("Specific Hypothesis:", specific_no)
    print("General Hypothesis:")
    for hypothesis in general_no:
        print(hypothesis)
```

```
 Specific Hypothesis: ['?', '?', '?', '?']
 General Hypothesis:
 ['?', '?', '?', '?']
 ['?', '?', '?', '?']
 ['?', '?', '?', '?']
 ['?', '?', '?', '?']
```

-

Multiple linear regression (MLR).

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt


# Load your dataset from a CSV file
# Replace 'your_dataset.csv' with the actual filename
dataset = pd.read_csv('car_data.csv')


from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

# Apply label encoding to categorical variables
dataset['Fuel_Type'] = label_encoder.fit_transform(dataset['Fuel_Type'])
dataset['Seller_Type'] = label_encoder.fit_transform(dataset['Seller_Type'])
dataset['Transmission'] = label_encoder.fit_transform(dataset['Transmission'])


dataset.to_csv('car_data_processed.csv', index=False)


# Assuming the dataset has a column named 'feature' and a column named 'target'
X = dataset[['Present_Price','Kms_Driven', 'Fuel_Type', 'Seller_Type', 'Transmission']]
y = dataset['Selling_Price']


# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Create a multiple linear regression model
model = LinearRegression()


# Train the model on the training set
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

```
# Make predictions on the test set
y_pred = model.predict(X_test)


# Print the coefficients and intercept
print('Coefficients:', model.coef_)
print('Intercept:', model.intercept_)
```

```
Coefficients: [ 4.29082212e-01 -1.94213590e-05 -1.55460748e+00 -1.54958273e+00
 -1.90073022e+00]
Intercept: 7.157709924479171
```

```
# Calculate Mean Absolute Error (MAE)
from sklearn.metrics import mean_absolute_error


mae = mean_absolute_error(y_test, y_pred)
print('Mean Absolute Error (MAE):', mae)
```

```
Mean Absolute Error (MAE): 1.428752196642739
```

```
# Calculate Mean Squared Error (MSE)
from sklearn.metrics import mean_squared_error


mse = mean_squared_error(y_test, y_pred)
print('Mean Squared Error (MSE):', mse)
```

```
Mean Squared Error (MSE): 5.0803527711401735
```

```
# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)
print('Root Mean Squared Error (RMSE):', rmse)
```

```
Root Mean Squared Error (RMSE): 2.25396379100024
```