Name: **Shehab Magdy**

ID: 22011558

Assignment (1): Backward Selection

Objective

Backward Selection (also known as Backward Elimination) is a Feature Selection Technique, one of the wrapper-based methods that are iterative and depends on the machine learning model performance to evaluate and assess dataset features to decide whether to include or discard that feature based on it's impact on the model performance. it starts with all features and removes the least useful one at each step. in this notebook i will be implementing the Backward Selection from scratch

Steps

---

1. Begin with all available features in the dataset
2. train the model on the availabe features
3. Measure the model's performance using an appropriate metric
4. Use a criterion to identify the least important feature
5. Drop the least significant feature from the dataset
6. Go to Step 2 and Repeat till removing features starts to degrade the model performance

Dataset : **Sklearn Diabetes Dataset**

Number of Instances: 442
Number of Attributes: The first 10 columns are numeric predictive values.
Target: Column 11 represents a quantitative measure of disease progression one year after baseline

Columns:
age: Age in years
sex: Gender of the patient
bmi: Body mass index
bp: Average blood pressure
s1: Total serum cholesterol (tc)
s2: Low-density lipoproteins (ldl)
s3: High-density lipoproteins (hdl)
s4: Total cholesterol / HDL (tch)
s5: Possibly log of serum triglycerides level (ltg)
s6: Blood sugar level (glu)

Machine Learning Model : **Linear Regression**

---

the objective behind using this data is to predict disease progression based on patient characteristics.
Target Variabe: a quantitative measure of disease progression one year after baseline for each patient.
Model Evaluation: Mean Absolute Error (MAE)

Feature Evaluation Creteria: **Model Performance measurement with Mean Absolute Error (MAE)**

- using MAEe as criteria to assess the importance of a feature in the model will help in making the decision whether to discard or keep that feature if it met the predefined threshold stated
- MAE calculates the error between the test target variable and the predicted target variable

Implementation

In [8]:
```python
# import dataset from sklearn
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
```

In [10]:
```python
# importing pandas to put the Data in a Pandas DataFrame
import pandas as pd
data = pd.DataFrame(diabetes.data, columns = diabetes.feature_names)

# Add target variable to the DataFrame
data['target'] = diabetes.target

data.head()
```

Out[10]:

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641	135.0

In [12]:
```python
from sklearn.linear_model import LinearRegression # import the regression model
from sklearn.model_selection import train_test_split # splitting data for modeling
from sklearn.metrics import mean_absolute_error  # model evaluation metric
```

In [152…
```python
def backward_selection(df, threshold = 0.03):
    '''
    Implements the backward selection technique using MAE (Mean Absolute Error) as the evaluation metric.

    Parameters:
    - df: DataFrame containing the features plus the target variable.
    - threshold: The MAE change threshold for feature elimination.

    Returns:
    - Reduced DataFrame with the selected features.
    '''

    X = df.drop(columns=['target'])  # Features
    y = df['target']  # Target

    # Split 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=43)

    features = X.columns.tolist() # a list of columns name

    # Evaluate model using MAE
    def evaluate_model(X_train, X_test, y_train, y_test):
        model = LinearRegression().fit(X_train, y_train)
        y_pred = model.predict(X_test)
        return mean_absolute_error(y_test, y_pred)

    # the model performance with full features before any feature removal
    mae_main = evaluate_model(X_train, X_test, y_train, y_test)

    while len(features) > 1:
        feature_to_drop = None

        # Iterate over features
        for feature in features:
            X_temp = X_train.drop(columns=[feature])  # Drop feature from train set
            X_test_temp = X_test.drop(columns=[feature])  # Drop feature from test set

            # Evaluate performance after the feature dropping
            mae_temp = evaluate_model(X_temp, X_test_temp, y_train, y_test)

            # compare the change between the older mae and the new mae with the threshold value
            if mae_main - mae_temp >= threshold:
                feature_to_drop = feature
```

```
                break  # Exit loop after selecting a feature to drop

        if feature_to_drop:  # drop the feature and check the model performance again
            X_train = X_train.drop(columns=[feature_to_drop])
            X_test = X_test.drop(columns=[feature_to_drop])
            features.remove(feature_to_drop)
            mae_main = evaluate_model(X_train, X_test, y_train, y_test)
        else:
            break  # Stop if no feature meets the threshold

    df = X.copy()[features]
    df['target'] = y  # Add back the target variable

    return df
```

In [144…

```
print(f"dataset shape before applying Backward Elimination:{data.drop(columns = 'target').shape}")
```

```
dataset shape before applying Backward Elimination:(442, 10)
```

In [154…

```
reduced_df = backward_selection(data)
print(f"dataset shape after applying Backward Elimination:{reduced_df.shape}")
```

```
dataset shape after applying Backward Elimination:(442, 8)
```

In [146…

```
data.drop(columns = 'target').head()
```

Out[146…

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641

In [150…

```
reduced_df.head()
```

Out[150…

	sex	bmi	bp	s1	s2	s5	s6	target
0	0.050680	0.061696	0.021872	-0.044223	-0.034821	0.019907	-0.017646	151.0
1	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	-0.068332	-0.092204	75.0
2	0.050680	0.044451	-0.005670	-0.045599	-0.034194	0.002861	-0.025930	141.0
3	-0.044642	-0.011595	-0.036656	0.012191	0.024991	0.022688	-0.009362	206.0
4	-0.044642	-0.036385	0.021872	0.003935	0.015596	-0.031988	-0.046641	135.0

Assumptions

- After applying back elimination on the dataset with threshold value `0.03` we discarded 3 features including `age`, `s3` and `s4` whom are the least effective features in the data set and their removal increased the model performance
- calibrating the threshold again we get different outcomes