Experiment 8: Implement Decision Tree by using Diabetes Dataset

In [4]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
```

In [5]:

```
diabetes.feature_names
#diabetes.DESCR
```

Out[5]:

```
['age', 'sex', 'bmi', 'bp', 's1', 's2', 's3', 's4', 's5', 's6']
```

In [6]:

```
df = pd.DataFrame(diabetes.data, columns=diabetes.feature_names)
df.head()
```

Out[6]:

	age	sex	bmi	bp	s1	s2	s3	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0

In [7]:

```
df['target'] = diabetes.target
df.head()
```

Out[7]:

	age	sex	bmi	bp	s1	s2	s3	
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0

In [8]:

```
diabetes.target_filename
```

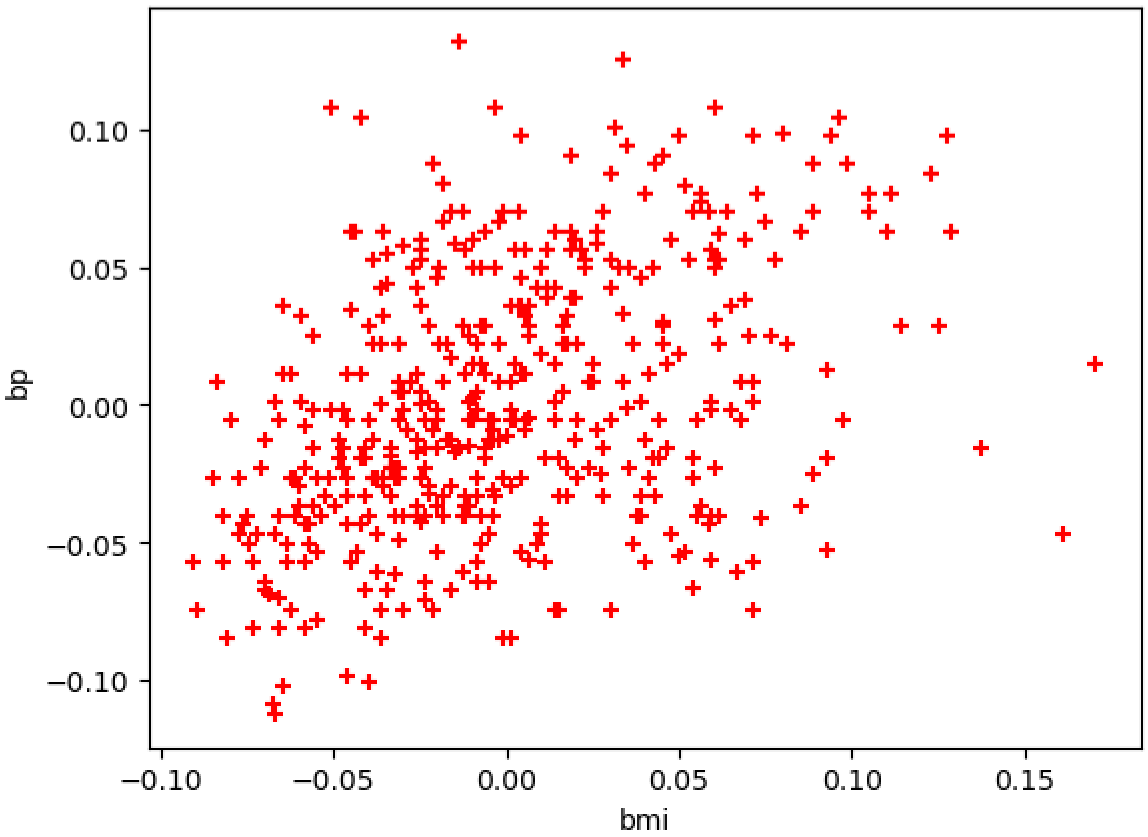
Out[8]:

```
'diabetes_target.csv.gz'
```

In [9]:

```
plt.xlabel('bmi')
plt.ylabel('bp')
plt.scatter(df['bmi'],df['bp'],color='red', marker='+')
```

```
Out[9]:  <matplotlib.collections.PathCollection at 0x20dad0115b0>
```



```
In [10]:  from sklearn.model_selection import train_test_split

In [11]:  x = df.iloc[:,:-1]

In [12]:  y = df.iloc[:,:-1]

In [13]:  x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2)

In [14]:  from sklearn import tree
          model = tree.DecisionTreeRegressor()

In [15]:  model.fit(x_train,y_train)

Out[15]:  ▾ DecisionTreeRegressor ⓘ ?
          DecisionTreeRegressor()

In [16]:  model.score(x_train,y_train)

Out[16]:  1.0

In [17]:  model.score(x_test,y_test)

Out[17]:  0.46798286239277226
```

In [18]:
```
model.predict(x_test[:6])
```

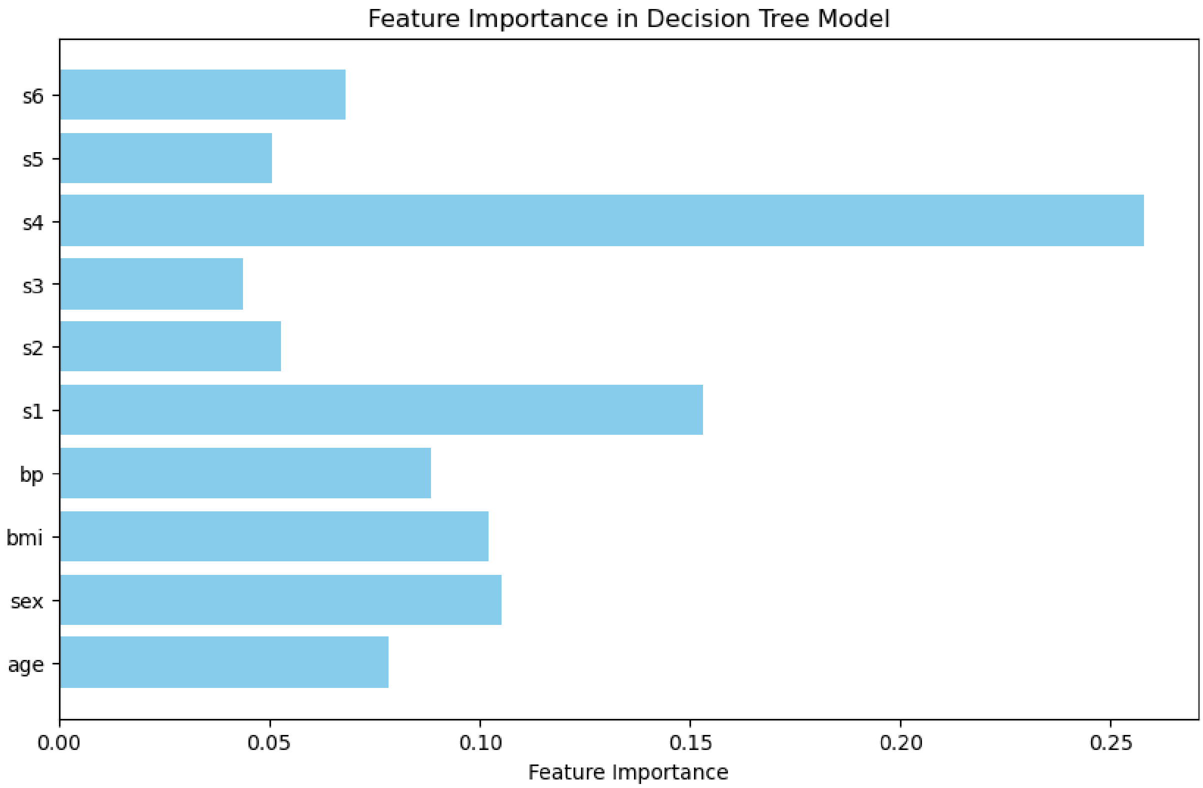
Out[18]:
```
array([[-0.02004471, -0.04464164,  0.00457217,  0.09761511,  0.0053108 ,
        -0.02072908,  0.06336665, -0.03949338,  0.01255119,  0.01134862],
       [ 0.02717829,  0.05068012,  0.09403057,  0.09761511, -0.03459183,
        -0.03200243, -0.04340085, -0.00259226,  0.03664373,  0.10661708],
       [-0.00914709,  0.05068012, -0.03961813, -0.04009893, -0.00844872,
         0.01622244, -0.06549067,  0.07120998,  0.01776532, -0.06735141],
       [ 0.05987114, -0.04464164, -0.00081689, -0.08485599,  0.0754844 ,
         0.07947843,  0.00446045,  0.03430886,  0.02337142,  0.02791705],
       [-0.07453279, -0.04464164,  0.0433734 , -0.03321323,  0.01219057,
         0.00025186,  0.06336665, -0.03949338, -0.02712902, -0.04664087],
       [-0.00551455,  0.05068012, -0.00836158, -0.00222757, -0.03321588,
        -0.06363042, -0.03603757, -0.00259226,  0.08059005,  0.00720652]])
```

In [19]:
```
x_test[:6]
```

Out[19]:

	age	sex	bmi	bp	s1	s2	s3
343	-0.020045	-0.044642	0.018584	0.090729	0.003935	0.008707	0.037595
163	0.016281	0.050680	0.072474	0.076958	-0.008449	0.005575	-0.006584
66	-0.009147	0.050680	-0.018062	-0.033213	-0.020832	0.012152	-0.072854
282	0.070769	-0.044642	-0.005128	-0.005670	0.087868	0.102965	0.011824
143	-0.030942	-0.044642	0.005650	-0.009113	0.019070	0.006828	0.074412
275	-0.005515	0.050680	-0.011595	0.011544	-0.022208	-0.015406	-0.021311

In [20]:
```
# Feature Importance
importances = model.feature_importances_
plt.figure(figsize=(10, 6))
plt.barh(df.columns[:-1], importances, color="skyblue")
plt.xlabel("Feature Importance")
plt.title("Feature Importance in Decision Tree Model")
plt.show()
```



In [40]:

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Predictions with the best model
y_pred = best_model.predict(x_test)

# Evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
r2_score = best_model.score(x_test, y_test)

print("Mean Absolute Error (MAE):", mae)
print("Root Mean Squared Error (RMSE):", rmse)
print("R^2 Score:", r2_score)
```

```
Mean Absolute Error (MAE): 0.021066464678941133
Root Mean Squared Error (RMSE): 0.02900287696752611
R^2 Score: 0.6095040401103815
```

This notebook was converted with convert.ploomber.io