1.Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.axes as ax

from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
```

2.Loading Data

```
In [2]: from sklearn.datasets import load_boston

        boston_data = load_boston()
        boston_data
```

```
Out[2]: {'data': array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
                 4.9800e+00],
                [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
                 9.1400e+00],
                [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
                 4.0300e+00],
                ...,
                [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
                 5.6400e+00],
                [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
                 6.4800e+00],
                [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
                 7.8800e+00]]),
         'target': array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15. ,
                18.9, 21.7, 20.4, 18.2, 19.9, 23.1, 17.5, 20.2, 18.2, 13.6, 19.6,
                15.2, 14.5, 15.6, 13.9, 16.6, 14.8, 18.4, 21. , 12.7, 14.5, 13.2,
                13.1, 13.5, 18.9, 20. , 21. , 24.7, 30.8, 34.9, 26.6, 25.3, 24.7,
                21.2, 19.3, 20. , 16.6, 14.4, 19.4, 19.7, 20.5, 25. , 23.4, 18.9,
                35.4, 24.7, 31.6, 23.3, 19.6, 18.7, 16. , 22.2, 25. , 33. , 23.5,
                19.4, 22. , 17.4, 20.9, 24.2, 21.7, 22.8, 23.4, 24.1, 21.4, 20. ,
                20.8, 21.2, 20.3, 28. , 23.9, 24.8, 22.9, 23.9, 26.6, 22.5, 22.2,
                23.6, 28.7, 22.6, 22. , 22.9, 25. , 20.6, 28.4, 21.4, 38.7, 43.8,
                33.2, 27.5, 26.5, 18.6, 19.3, 20.1, 19.5, 19.5, 20.4, 19.8, 19.4,
                21.7, 22.8, 18.8, 18.7, 18.5, 18.3, 21.2, 19.2, 20.4, 19.3, 22. ,
                20.3, 20.5, 17.3, 18.8, 21.4, 15.7, 16.2, 18. , 14.3, 19.2, 19.6,
                23. , 18.4, 15.6, 18.1, 17.4, 17.1, 13.3, 17.8, 14. , 14.4, 13.4,
                15.6, 11.8, 13.8, 15.6, 14.6, 17.8, 15.4, 21.5, 19.6, 15.3, 19.4,
                17. , 15.6, 13.1, 41.3, 24.3, 23.3, 27. , 50. , 50. , 50. , 22.7,
                25. , 50. , 23.8, 23.8, 22.3, 17.4, 19.1, 23.1, 23.6, 22.6, 29.4,
                23.2, 24.6, 29.9, 37.2, 39.8, 36.2, 37.9, 32.5, 26.4, 29.6, 50. ,
                32. , 29.8, 34.9, 37. , 30.5, 36.4, 31.1, 29.1, 50. , 33.3, 30.3,
                34.6, 34.9, 32.9, 24.1, 42.3, 48.5, 50. , 22.6, 24.4, 22.5, 24.4,
                20. , 21.7, 19.3, 22.4, 28.1, 23.7, 25. , 23.3, 28.7, 21.5, 23. ,
                26.7, 21.7, 27.5, 30.1, 44.8, 50. , 37.6, 31.6, 46.7, 31.5, 24.3,
                31.7, 41.7, 48.3, 29. , 24. , 25.1, 31.5, 23.7, 23.3, 22. , 20.1,
                22.2, 23.7, 17.6, 18.5, 24.3, 20.5, 24.5, 26.2, 24.4, 24.8, 29.6,
                42.8, 21.9, 20.9, 44. , 50. , 36. , 30.1, 33.8, 43.1, 48.8, 31. ,
                36.5, 22.8, 30.7, 50. , 43.5, 20.7, 21.1, 25.2, 24.4, 35.2, 32.4,
                32. , 33.2, 33.1, 29.1, 35.1, 45.4, 35.4, 46. , 50. , 32.2, 22. ,
                20.1, 23.2, 22.3, 24.8, 28.5, 37.3, 27.9, 23.9, 21.7, 28.6, 27.1,
                20.3, 22.5, 29. , 24.8, 22. , 26.4, 33.1, 36.1, 28.4, 33.4, 28.2,
                22.8, 20.3, 16.1, 22.1, 19.4, 21.6, 23.8, 16.2, 17.8, 19.8, 23.1,
                21. , 23.8, 23.1, 20.4, 18.5, 25. , 24.6, 23. , 22.2, 19.3, 22.6,
                19.8, 17.1, 19.4, 22.2, 20.7, 21.1, 19.5, 18.5, 20.6, 19. , 18.7,
                32.7, 16.5, 23.9, 31.2, 17.5, 17.2, 23.1, 24.5, 26.6, 22.9, 24.1,
                18.6, 30.1, 18.2, 20.6, 17.8, 21.7, 22.7, 22.6, 25. , 19.9, 20.8,
                16.8, 21.9, 27.5, 21.9, 23.1, 50. , 50. , 50. , 50. , 50. , 13.8,
                13.8, 15. , 13.9, 13.3, 13.1, 10.2, 10.4, 10.9, 11.3, 12.3,  8.8,
                 7.2, 10.5,  7.4, 10.2, 11.5, 15.1, 23.2,  9.7, 13.8, 12.7, 13.1,
                12.5,  8.5,  5. ,  6.3,  5.6,  7.2, 12.1,  8.3,  8.5,  5. , 11.9,
                27.9, 17.2, 27.5, 15. , 17.2, 17.9, 16.3,  7. ,  7.2,  7.5, 10.4,
                 8.8,  8.4, 16.7, 14.2, 20.8, 13.4, 11.7,  8.3, 10.2, 10.9, 11. ,
                 9.5, 14.5, 14.1, 16.1, 14.3, 11.7, 13.4,  9.6,  8.7,  8.4, 12.8,
                10.5, 17.1, 18.4, 15.4, 10.8, 11.8, 14.9, 12.6, 14.1, 13. , 13.4,
```

```
        15.2, 16.1, 17.8, 14.9, 14.1, 12.7, 13.5, 14.9, 20. , 16.4, 17.7,
        19.5, 20.2, 21.4, 19.9, 19. , 19.1, 19.1, 20.1, 19.9, 19.6, 23.2,
        29.8, 13.8, 13.3, 16.7, 12. , 14.6, 21.4, 23. , 23.7, 25. , 21.8,
        20.6, 21.2, 19.1, 20.6, 15.2,  7. ,  8.1, 13.6, 20.1, 21.8, 24.5,
        23.1, 19.7, 18.3, 21.2, 17.5, 16.8, 22.4, 20.6, 23.9, 22. , 11.9]),
 'feature_names': array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RA
D',
        'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7'),
 'DESCR': ".. _boston_dataset:\n\nBoston house prices dataset\n-----------------------
---\n\n**Data Set Characteristics:**  \n\n    :Number of Instances: 506 \n\n    :Number
of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usuall
y the target.\n\n    :Attribute Information (in order):\n        - CRIM     per capita
crime rate by town\n        - ZN       proportion of residential land zoned for lots ov
er 25,000 sq.ft.\n        - INDUS    proportion of non-retail business acres per town\n
- CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)\n
- NOX      nitric oxides concentration (parts per 10 million)\n        - RM       avera
ge number of rooms per dwelling\n        - AGE      proportion of owner-occupied units
built prior to 1940\n        - DIS      weighted distances to five Boston employment ce
ntres\n        - RAD      index of accessibility to radial highways\n        - TAX
full-value property-tax rate per $10,000\n        - PTRATIO  pupil-teacher ratio by tow
n\n        - B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town\n
- LSTAT    % lower status of the population\n        - MEDV     Median value of owner-o
ccupied homes in $1000's\n\n    :Missing Attribute Values: None\n\n    :Creator: Harris
on, D. and Rubinfeld, D.L.\n\nThis is a copy of UCI ML housing dataset.\nhttps://archiv
e.ics.uci.edu/ml/machine-learning-databases/housing/\n\n\nThis dataset was taken from t
he StatLib library which is maintained at Carnegie Mellon University.\n\nThe Boston hou
se-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic\nprices and the demand for c
lean air', J. Environ. Economics & Management,\nvol.5, 81-102, 1978.   Used in Belsley,
Kuh & Welsch, 'Regression diagnostics\n...', Wiley, 1980.   N.B. Various transformation
s are used in the table on\npages 244-261 of the latter.\n\nThe Boston house-price data
has been used in many machine learning papers that address regression\nproblems.   \n
\n.. topic:: References\n\n   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identif
ying Influential Data and Sources of Collinearity', Wiley, 1980. 244-261.\n   - Quinla
n,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the T
enth International Conference of Machine Learning, 236-243, University of Massachusett
s, Amherst. Morgan Kaufmann.\n",
 'filename': 'C:\\Users\\G.SAI KRISHNA\\anaconda3\\lib\\site-packages\\sklearn\\dataset
s\\data\\boston_house_prices.csv'}
```

In [3]:
```
data=pd.DataFrame(data=boston_data.data,columns=boston_data.feature_names)
```

In [4]:
```
data['MEDV']=boston_data.target
```

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

In [6]:
```
data.head(10)
```

Out[6]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0.0	0.458	6.430	58.7	6.0622	3.0	222.0	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0.0	0.524	6.012	66.6	5.5605	5.0	311.0	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0.0	0.524	6.172	96.1	5.9505	5.0	311.0	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0.0	0.524	5.631	100.0	6.0821	5.0	311.0	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0.0	0.524	6.004	85.9	6.5921	5.0	311.0	15.2	386.71	17.10	18.9

3.Data Preprocessing

In [7]:
```
#Handling Null Values
data.isnull().sum()
```

Out[7]:
```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
MEDV       0
dtype: int64
```

In [8]:
```
data.describe()
```

Out[8]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.00
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.54
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.70
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.00
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.00
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.00
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.00
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.00

4.Data Splitting

In [9]:
```
x=data.drop(['MEDV'],axis=1)
y=data['MEDV']
```

In [10]:
```
x.head()
```

Out[10]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33

In [11]:
```
y.head()
```

Out[11]:
```
0    24.0
1    21.6
2    34.7
3    33.4
4    36.2
Name: MEDV, dtype: float64
```

5.Data Scaling

In [12]:
```
scaler_x = StandardScaler()
x = scaler_x.fit_transform(x)
x
```

Out[12]:
```
array([[-0.41978194,  0.28482986, -1.2879095 , ..., -1.45900038,
         0.44105193, -1.0755623 ],
       [-0.41733926, -0.48772236, -0.59338101, ..., -0.30309415,
         0.44105193, -0.49243937],
       [-0.41734159, -0.48772236, -0.59338101, ..., -0.30309415,
         0.39642699, -1.2087274 ],
       ...,
       [-0.41344658, -0.48772236,  0.11573841, ...,  1.17646583,
         0.44105193, -0.98304761],
       [-0.40776407, -0.48772236,  0.11573841, ...,  1.17646583,
         0.4032249 , -0.86530163],
       [-0.41500016, -0.48772236,  0.11573841, ...,  1.17646583,
         0.44105193, -0.66905833]])
```

6.Training & Testing Data

In [13]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=0)
```

In [14]:
```
x_train.shape
```

Out[14]:
```
(354, 13)
```

In [15]:
```
x_test.shape
```

Out[15]:
```
(152, 13)
```

In [16]:
```
y_train.shape
```

Out[16]:
```
(354,)
```

In [17]:
```
y_test.shape
```

Out[17]:
```
(152,)
```

7.Random Forest Regression

Training the Model

In [18]:
```
rfr = RandomForestRegressor(1000)

rfr.fit(x_train,y_train)
```

Out[18]:
```
RandomForestRegressor(n_estimators=1000)
```

Predicting Test Values

In [19]:
```
y_pred = rfr.predict(x_test)

y_pred
```

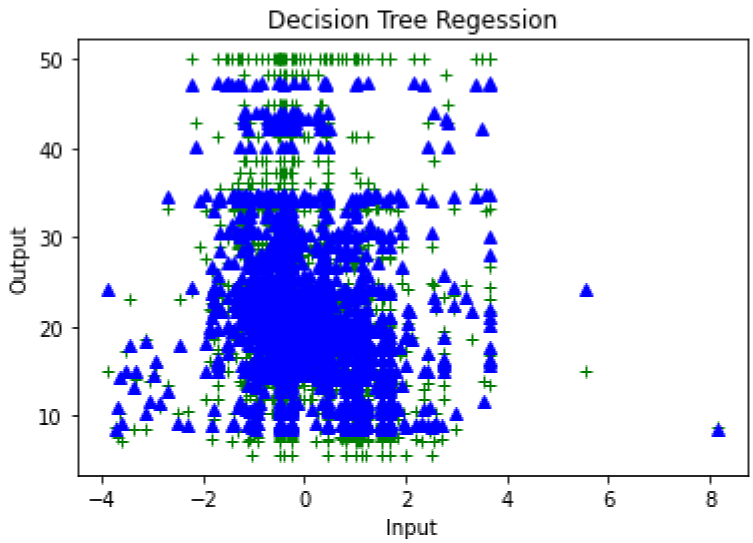
Out[19]:
```
array([23.8649, 30.313 , 22.0392, 10.8742, 20.5112, 20.9013, 21.1548,
       20.2765, 20.0401, 18.4683,  8.5282, 15.1039, 14.8617,  8.8998,
       47.3892, 33.9982, 20.9845, 34.6679, 25.4874, 21.1749, 23.7837,
       21.9821, 19.7088, 24.1614, 20.3698, 18.0315, 18.6928, 15.9858,
       43.9416, 18.6259, 14.774 , 17.4063, 20.4513, 20.9547, 22.9134,
       17.7156,  8.6921, 30.1649, 14.454 , 14.6563, 22.8025, 20.7983,
       22.3097, 15.0248, 23.2536, 22.491 , 20.5788, 16.3919, 14.9816,
       25.2653, 16.2098, 19.7328, 20.142 , 40.1847, 14.8773, 19.9407,
       19.5109, 18.9039, 24.1756, 20.1048, 21.6727, 20.9496, 33.0194,
       28.604 , 18.2651, 27.2308, 16.3462, 18.0122, 18.1677, 21.8204,
       20.0745, 22.7805, 24.3898, 31.512 , 29.0344,  9.0521, 43.3805,
       21.9193, 22.715 , 19.4919, 26.9377, 18.0274, 23.7381, 42.8931,
       42.2137, 24.2931, 23.0369, 14.4315, 25.738 , 16.4235, 18.8312,
       12.7482, 22.4625, 30.4702, 21.3455, 22.0251, 11.6095, 23.2518,
       14.9602, 18.9726, 23.9216, 19.9358, 28.1455, 21.1187, 28.1366,
       20.2765,  9.1435, 19.4267, 21.7074, 23.0575, 34.3257, 13.5123,
       18.2186, 18.478 , 17.1325, 21.0765, 10.1944, 19.722 , 10.7995,
       47.1577, 30.5152, 10.5338, 19.3249, 20.4651, 20.5921, 18.579 ,
       34.1507, 19.101 , 20.5561, 34.8378, 14.0063, 10.9678, 14.9268,
       19.6606, 13.2385, 34.5584, 20.6294, 15.5886, 26.7282,  9.2076,
       10.7778, 20.813 , 32.2243, 24.2117, 24.4897, 17.0145, 34.7306,
       34.2757, 11.4451,  8.9918, 29.1161, 24.3784])
```

Visualizing Model Performance

In [20]:
```
error=mean_squared_error(y_pred,y_test)
error
```

Out[20]:
```
14.102739131381625
```

In [21]:
```
plt.figure()
plt.plot(x_test,y_test,'+',color="green")
plt.plot(x_test,y_pred,'^',color="blue")
plt.title("Decision Tree Regession")
plt.xlabel("Input")
plt.ylabel("Output")
plt.show()
```



In [22]:
```
print("Accuracy : "+str(100 - error)+"%")
```

```
Accuracy : 85.89726086861837%
```