6.DECISON TREE:

- Data scientist Interview preparation
- Practical
- June2024
- II.Decison Tree Regressor

Supervised ml :classifcation & regression task

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
##Boston House Pricing Dataset :sklearn # dont run this cell
from sklearn.datasets import load_boston
boston_df=load_boston()
#Note: The Boston housing dataset has been deprecated in scikit-lear
#ethical concerns and is no longer available in the latest version
```

```
from sklearn.datasets import fetch_openml

# Load the Boston housing dataset from OpenML
boston = fetch_openml(data_id=531)
```

```
boston
```

```
print(boston.feature_names)
```

```
['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD'
```

```
print(boston.data.shape)
print(boston.target.shape)
```

```
(506, 13)
(506,)
```

```
# Extract features and target
X = boston.data
y = boston.target
```

```
X
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	29
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	24
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	24
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	22
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	22
...	...	...	...	...	...	...	...	...	...	
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	21
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	21
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	21
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	21
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	21

506 rows × 13 columns

```
y
```

```
0      24.0
1      21.6
2      34.7
```

```
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: MEDV, Length: 506, dtype: float64
```

```
 #or you canwrite #independent features
X=pd.DataFrame(boston_df.data,columns=boston_df.feature_names)
#dependent features
y=boston_df.target
```

```
### train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size=0.20, random_state=42)
```

II.DECISION TREE REGRESSOR:

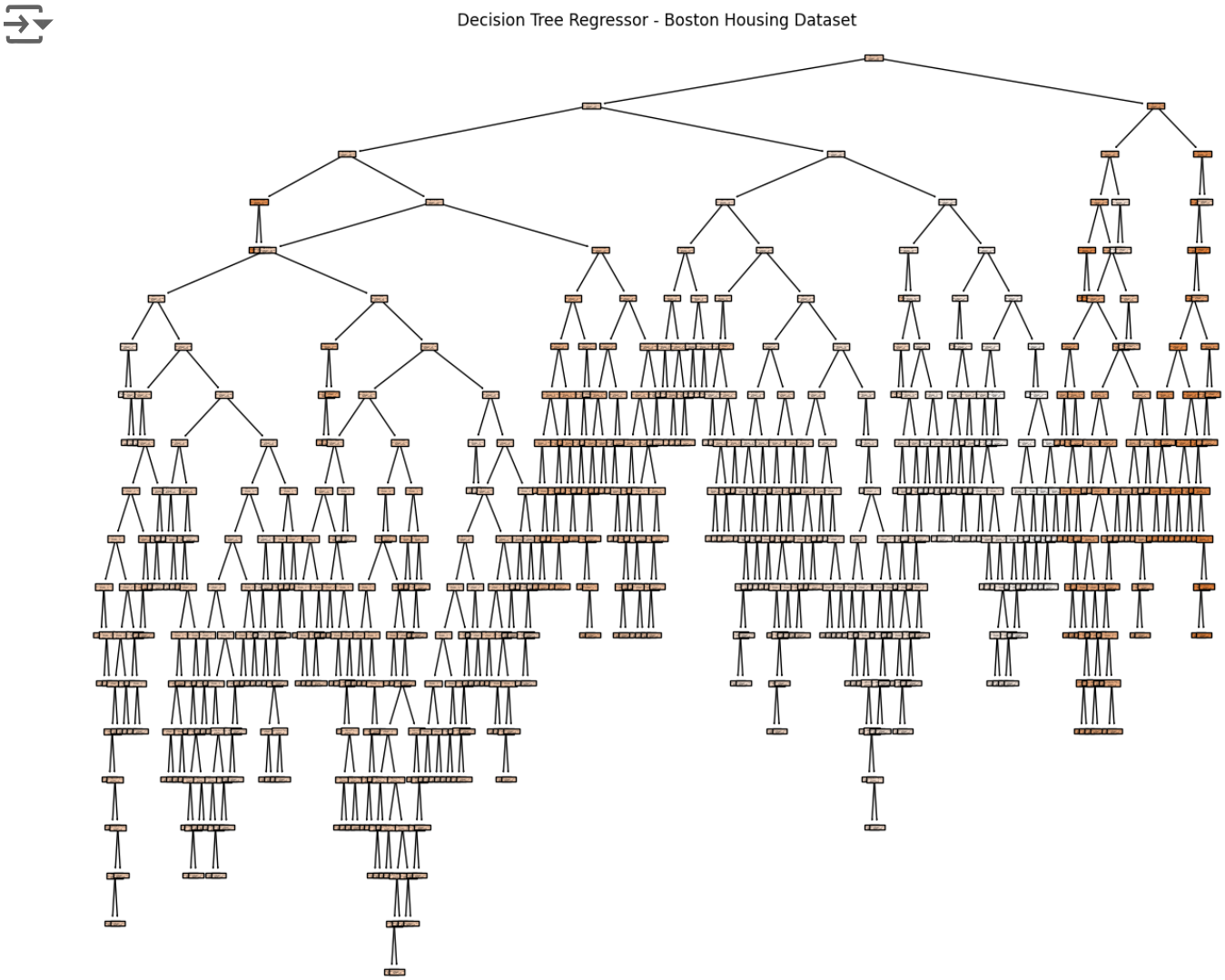
- Post pruning
- Pre pruning

```
from sklearn.tree import DecisionTreeRegressor
regressor=DecisionTreeRegressor()
regressor.fit(X_train,y_train)
```

▾ DecisionTreeRegressor

DecisionTreeRegressor()

```
from sklearn.tree import plot_tree
plt.figure(figsize=(15, 13))
plot_tree(regressor, filled=True, feature_names=boston.feature_names)
plt.title("Decision Tree Regressor - Boston Housing Dataset")
plt.show()
```


Decision Tree Regressor - Boston Housing Dataset

```
regressor.fit(X_train,y_train)
```

```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

```
y_pred=regressor.predict(X_test)
```

```
from sklearn.metrics import r2_score
score=r2_score(y_pred,y_test)
```

```
score
```

```
0.7125309163545431
```

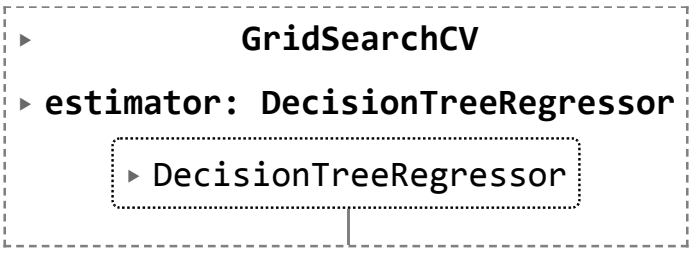
```
# Hyperparameter Tunning
parameter={
 'criterion':['squared_error','friedman_mse','absolute_error','poiss
  'splitter':['best','random'],
  'max_depth':[1,2,3,4,5,6,7,8,10,11,12],
  'max_features':['auto', 'sqrt', 'log2']

}
```

```
regressor=DecisionTreeRegressor()
```

```
from sklearn.model_selection import GridSearchCV
regressorcv=GridSearchCV(regressor,param_grid=parameter,cv=5,scoring
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
regressorcv.fit(X_train,y_train)
```




```
regressorcv.best_params_
```

```
{'criterion': 'poisson',
 'max_depth': 5,
 'max_features': 'auto',
 'splitter': 'best'}
```

```
y_pred=regressorcv.predict(X_test)
```

```
r2_score(y_pred,y_test)
```

```
0.6906700687236822
```

Post pruning

- post pruning :cutting down the specific nodes after building the Decision Tree

```
#post pruning :cutting down the specific nodes after building the De
```

II.Pre pruning:

- It is used before the construction of decision tree wih hyperparameter tuning and crsoss validation

- - criterion: Determines how the quality of a split is measured.
- - splitter: Determines the strategy used to choose the split at each node.