ML 25 - Decision Tree Regressor By Virat Tiwari

December 14, 2023

1 Decision Tree Regressor By Virat Tiwari

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```
# For this implementation of Decision Tree Regressor we have to " california␣
 ↪house pricing dataset " from sklearn
# Dataset name - California house pricing
# Source - sklearn

from sklearn.datasets import fetch_california_housing
```

NOTE - In this " California House Pricing Dataset " we have features of house and on the behalf of that features we PREDICT the PRICE of the house

[3]:
```
california_df=fetch_california_housing()
```

[4]:
```
# This is how our data look like

california_df
```

[4]:
```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, …,
2.55555556,
           37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, …,    2.10984183,
           37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, …,    2.80225989,
           37.85      , -122.24      ],
        …,
        [   1.7       ,   17.        ,    5.20554273, …,    2.3256351 ,
           39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, …,    2.12320917,
           39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, …,    2.61698113,
           39.37      , -121.24      ]]),
```

```
 'target': array([4.526, 3.585, 3.521, …, 0.923, 0.847, 0.894]),
 'frame': None,
 'target_names': ['MedHouseVal'],
 'feature_names': ['MedInc',
  'HouseAge',
  'AveRooms',
  'AveBedrms',
  'Population',
  'AveOccup',
  'Latitude',
  'Longitude'],
 'DESCR': '.. _california_housing_dataset:\n\nCalifornia Housing
dataset\n--------------------------\n\n**Data Set Characteristics:**\n\n
:Number of Instances: 20640\n\n    :Number of Attributes: 8 numeric, predictive
attributes and the target\n\n    :Attribute Information:\n        - MedInc
median income in block group\n        - HouseAge      median house age in block
group\n        - AveRooms      average number of rooms per household\n        -
AveBedrms     average number of bedrooms per household\n        - Population
block group population\n        - AveOccup      average number of household
members\n        - Latitude      block group latitude\n        - Longitude
block group longitude\n\n    :Missing Attribute Values: None\n\nThis dataset was
obtained from the StatLib
repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housing.html\n\nThe
target variable is the median house value for California districts,\nexpressed
in hundreds of thousands of dollars ($100,000).\n\nThis dataset was derived from
the 1990 U.S. census, using one row per census\nblock group. A block group is
the smallest geographical unit for which the U.S.\nCensus Bureau publishes
sample data (a block group typically has a population\nof 600 to 3,000
people).\n\nAn household is a group of people residing within a home. Since the
average\nnumber of rooms and bedrooms in this dataset are provided per
household, these\ncolumns may take surpinsingly large values for block groups
with few households\nand many empty houses, such as vacation resorts.\n\nIt can
be downloaded/loaded using
the\n:func:`sklearn.datasets.fetch_california_housing` function.\n\n.. topic::
References\n\n    - Pace, R. Kelley and Ronald Barry, Sparse Spatial
Autoregressions,\n      Statistics and Probability Letters, 33 (1997)
291-297\n'}
```

[5]:
```
# Independent Features

x=pd.DataFrame(california_df.data,columns=california_df.feature_names)

# Dependent Features

y=california_df.target
```

[6]:
```
x.head()
```

[6]:
```
   MedInc  HouseAge  AveRooms  AveBedrms  Population  AveOccup  Latitude  \
0  8.3252      41.0  6.984127   1.023810       322.0  2.555556     37.88
1  8.3014      21.0  6.238137   0.971880      2401.0  2.109842     37.86
2  7.2574      52.0  8.288136   1.073446       496.0  2.802260     37.85
3  5.6431      52.0  5.817352   1.073059       558.0  2.547945     37.85
4  3.8462      52.0  6.281853   1.081081       565.0  2.181467     37.85

   Longitude
0    -122.23
1    -122.22
2    -122.24
3    -122.25
4    -122.25
```

[7]:
```
y
```

[7]:
```
array([4.526, 3.585, 3.521, …, 0.923, 0.847, 0.894])
```

[8]:
```
# Train and 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.
 ↪33,random_state=42)
```

[9]:
```
from sklearn.tree import DecisionTreeRegressor
```

[10]:
```
regressor=DecisionTreeRegressor()
```

[11]:
```
regressor.fit(x_train,y_train)
```

[11]:
```
DecisionTreeRegressor()
```

[12]:
```
y_pred=regressor.predict(x_test)
```

[13]:
```
y_pred
```

[13]:
```
array([0.487  , 0.521  , 5.00001, …, 1.33   , 1.389  , 5.00001])
```

r2 score - it is used for evaluating the performance of regression model

[14]:
```
# Here we import r2_ score for evaluating the performance

from sklearn.metrics import r2_score
```

[15]:
```
score=r2_score(y_pred,y_test)
```

[16]:
```
score
```

[16]:
```
0.6037063484892174
```

r2 score = 59 %

HYPERPARAMETER TUNING -

[17]:
```
## Hyperparameter Tunning
parameter={
 'criterion':['squared_error','friedman_mse','absolute_error','poisson'],
  'splitter':['best','random'],
  'max_depth':[1,2,3,4,5,6,7,8,10,11,12],
  'max_features':['auto', 'sqrt', 'log2']

}
regressor=DecisionTreeRegressor()
```

[18]:
```
from sklearn.model_selection import GridSearchCV
regressorcv=GridSearchCV(regressor,param_grid=parameter,cv=5,scoring='neg_mean_squared_error')
```

[ ]:
```
regressorcv.fit(x_train,y_train)
```

[ ]:
```
GridSearchCV(cv=5, estimator=DecisionTreeRegressor(),
             param_grid={'criterion': ['squared_error', 'friedman_mse',
                                       'absolute_error', 'poisson'],
                         'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'splitter': ['best', 'random']},
             scoring='neg_mean_squared_error')
```

[ ]:
```
regressorcv.best_params_
```

[ ]:
```
y_pred=regressorcv.predict(x_test)
```

[ ]:
```
r2_score(y_pred,y_test)
```

THANK YOU SO MUCH !1

YOURS VIRAT TIWARI :)