Decision-Tree-Regression

January 14, 2025

[ ]:
```
'''
   Decision Tree Regression -->

   Decision Tree Regression is a type of regression model that uses a decision␣
↪tree structure
   to predict a continuous target variable. It splits the data into subsets␣
↪based on feature values
   and assigns a prediction value (typically the mean of target values in that␣
↪subset) at each leaf node.

   Decision Tree can be used for both Regression and Classification !
'''
```

[ ]:
```
'''
   How It Works -->

   Splitting the Data :

   At each node, the algorithm finds the best feature and split point␣
↪(threshold)
   that minimizes the error in the prediction.

   Common metrics for splitting :

   Mean Squared Error (MSE) : Measures the average squared difference between␣
↪actual and predicted values.
   Mean Absolute Error (MAE): Measures the average absolute difference between␣
↪actual and predicted values.

   Recursive Splitting :

   The tree continues splitting the dataset into smaller subsets until a␣
↪stopping condition is met
   (e.g., max depth, minimum samples per leaf, or no further improvement in␣
↪error reduction).

   Prediction :
```

```
    The predicted value for a leaf node is usually the mean of the target␣
 ↪values in that leaf.
'''
```

[ ]:
```
'''
    Advantages of Decision Tree Regression -->

    Non-Linear Relationships: Can model non-linear patterns without requiring␣
 ↪explicit transformations.
    Feature Importance: Identifies the most significant features for␣
 ↪predictions.
    No Scaling Required: No need for normalization or standardization of␣
 ↪features.
    Easy to Interpret: The tree structure is intuitive and easy to visualize.

    Disadvantages of Decision Tree Regression -->

    Overfitting: Deep trees can overfit the training data, leading to poor␣
 ↪generalization on unseen data.
    Instability: Small changes in data can result in a completely different␣
 ↪tree.
    Piecewise Predictions: The model creates a step-like prediction function,␣
 ↪which may not always capture smooth relationships.
'''
```

[ ]:
```
'''
    Key Parameters -->

    max_depth: Maximum depth of the tree to control overfitting.
    min_samples_split: Minimum number of samples required to split an internal␣
 ↪node.
    min_samples_leaf: Minimum number of samples required to be in a leaf node.
    max_features: Number of features to consider when looking for the best␣
 ↪split.
'''
```

[ ]:
```
'''
    When to Use Decision Tree Regression -->

    When your data has a non-linear relationship between features and the␣
 ↪target variable.
    When you need a model that's interpretable and doesn't require scaling.
    When you suspect interaction effects between features.
'''
```

[ ]:
```
#   Importing Libraries -->

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
```

[ ]:
```
#   Importing Dataset -->

data = pd.read_csv('Data/Position_Salaries.csv')
data
```

[ ]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000
5	Region Manager	6	150000
6	Partner	7	200000
7	Senior Partner	8	300000
8	C-level	9	500000
9	CEO	10	1000000

[ ]:
```
#   Extracting Features -->

x_data = data.iloc[:, 1:-1].values
y_data = data.iloc[:, -1].values
```

[5]:
```
#   Building Model -->

model = DecisionTreeRegressor(random_state = 0)
model.fit(x_data, y_data)
```

[5]:
```
DecisionTreeRegressor(random_state=0)
```

[8]:
```
#   Predicting Result -->

y_pred = model.predict([[6.5]])
y_pred
```

[8]:
```
array([150000.])
```

[9]:
```
#   Visualizing Results [High Resolution] -->

x_grid = np.arange(min(x_data), max(x_data), 0.1)
x_grid = x_grid.reshape((len(x_grid), 1))
plt.scatter(x_data, y_data, color = 'red')
```

```python
plt.plot(x_grid, model.predict(x_grid), color = 'blue')
plt.title("Decision Tree Regression")
plt.show()
```

```
C:\Users\krish\AppData\Local\Temp\ipykernel_6776\3695832824.py:3:
DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is
deprecated, and will error in future. Ensure you extract a single element from
your array before performing this operation. (Deprecated NumPy 1.25.)
  x_grid = np.arange(min(x_data), max(x_data), 0.1)
```