Machine Learning

| Regression using DT

Prof. Dr. Mostafa Elhosseini

https://youtube.com/ElhosseiniAcademy

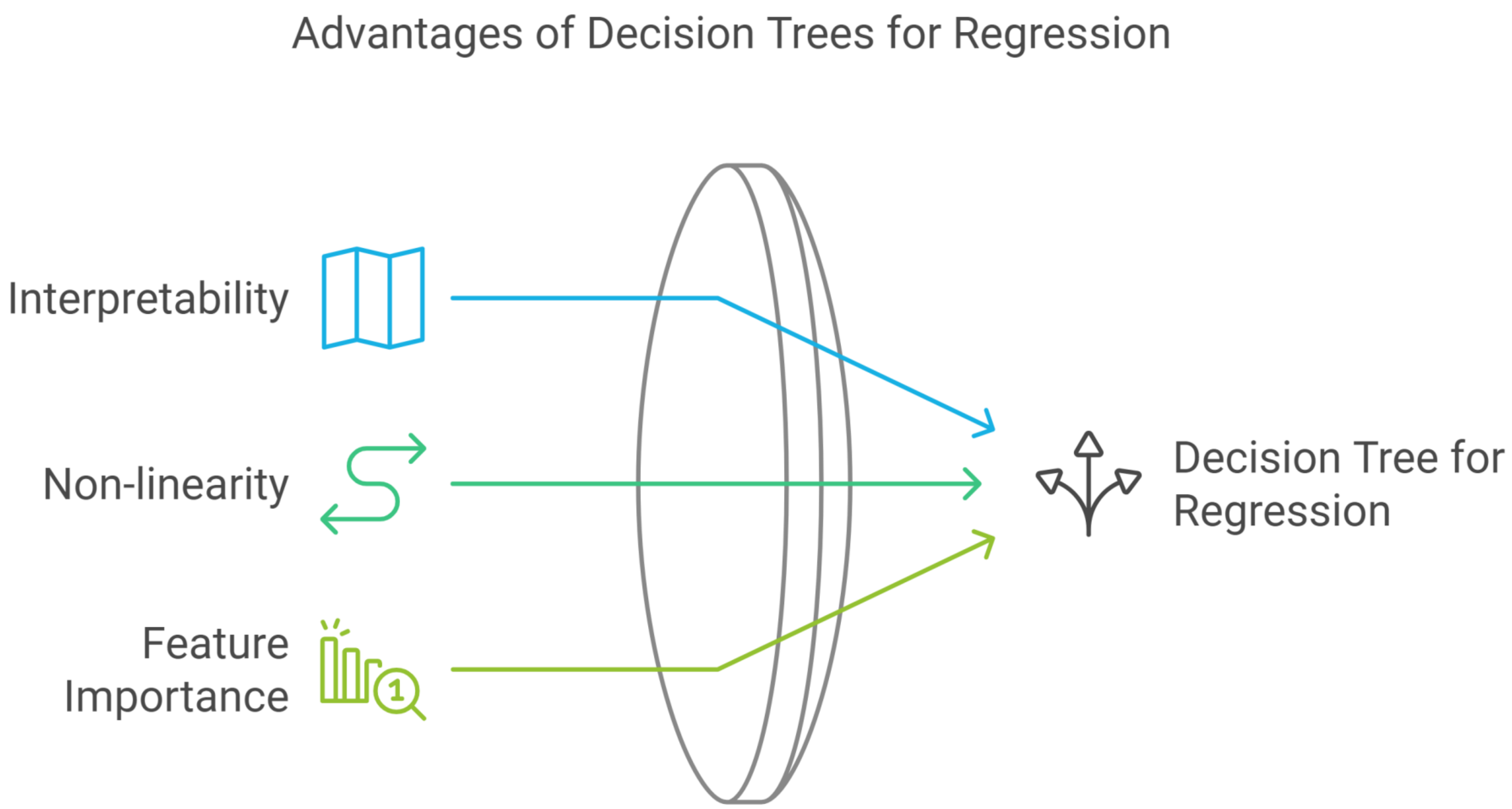
Agenda

- Regression using DT

Decision tree for regression

- Predicting continuous outcomes.
- It works by splitting the data into subsets based on feature values, forming a tree-like structure where each internal node represents a feature, each branch represents a decision rule, and each leaf node represents the **predicted outcome**.

Decision tree for regression



How Decision Trees Work for Regression

- **Splitting**: The algorithm begins with the entire dataset and **selects the best feature** to split the data into subsets. The goal is to minimize the variance (or mean squared error) within each subset.
- **Recursive Partitioning**: This process of splitting **continues** recursively for each subset **until a stopping criterion** is met, such as a **maximum depth** of the tree or a minimum number of samples in a leaf node.
- **Prediction**: For a new input, the decision tree traverses the nodes according to the feature values of the input until it **reaches a leaf node**, which provides the predicted continuous value.

How Decision Trees Work for Regression

Example 01

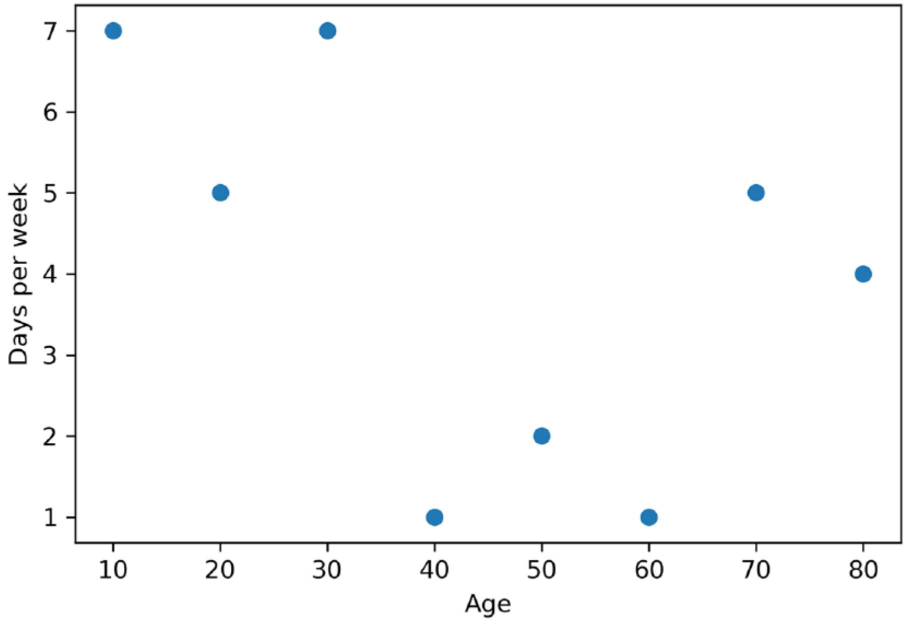
- we have an app, and we want to predict the level of engagement of the users in terms of how many days per week they used it. The only feature we have is the user’s age

Age	Engagement
10	7
20	5
30	7
40	1
50	2
60	1
70	5
80	4

How Decision Trees Work for Regression

Example 01

- we have an app, and we want to predict the level of engagement of the users in terms of how many days per week they used it. The only feature we have is the user’s age

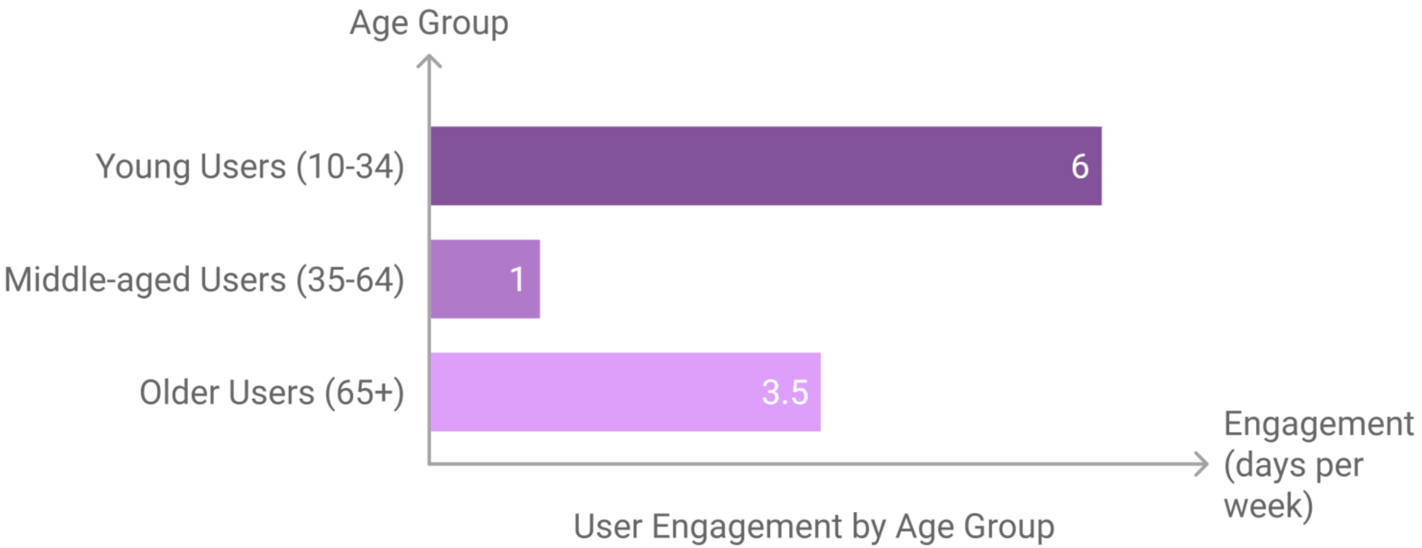


Age	Engagement
10	7
20	5
30	7
40	1
50	2
60	1
70	5
80	4

How Decision Trees Work for Regression

Example 01

- We have three clusters of users

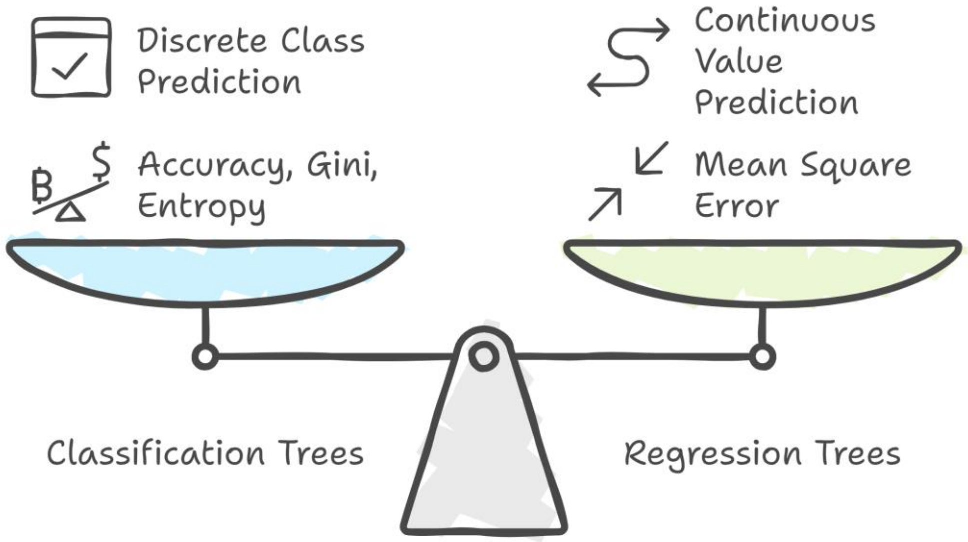


Age	Engagement
10	7
20	5
30	7
40	1
50	2
60	1
70	5
80	4

How Decision Trees Work for Regression

Example 01

- Use the mean square error (MSE)

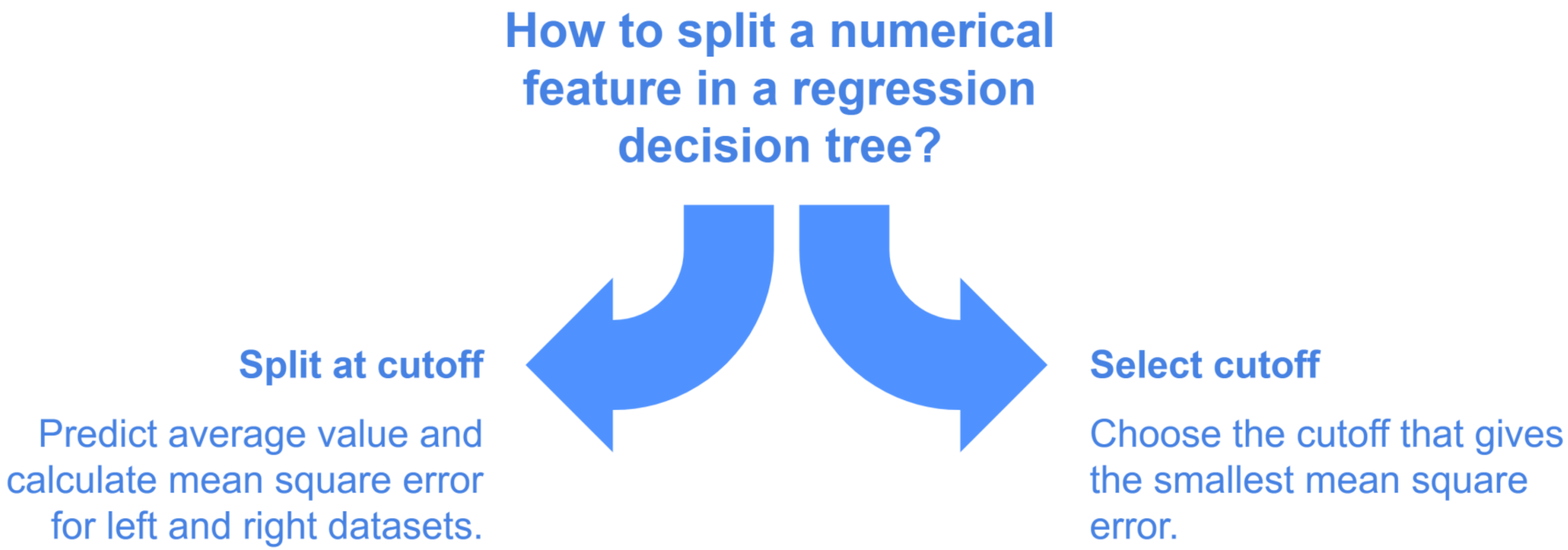


Comparing decision tree algorithms for classification and regression.

How Decision Trees Work for Regression

Example 01

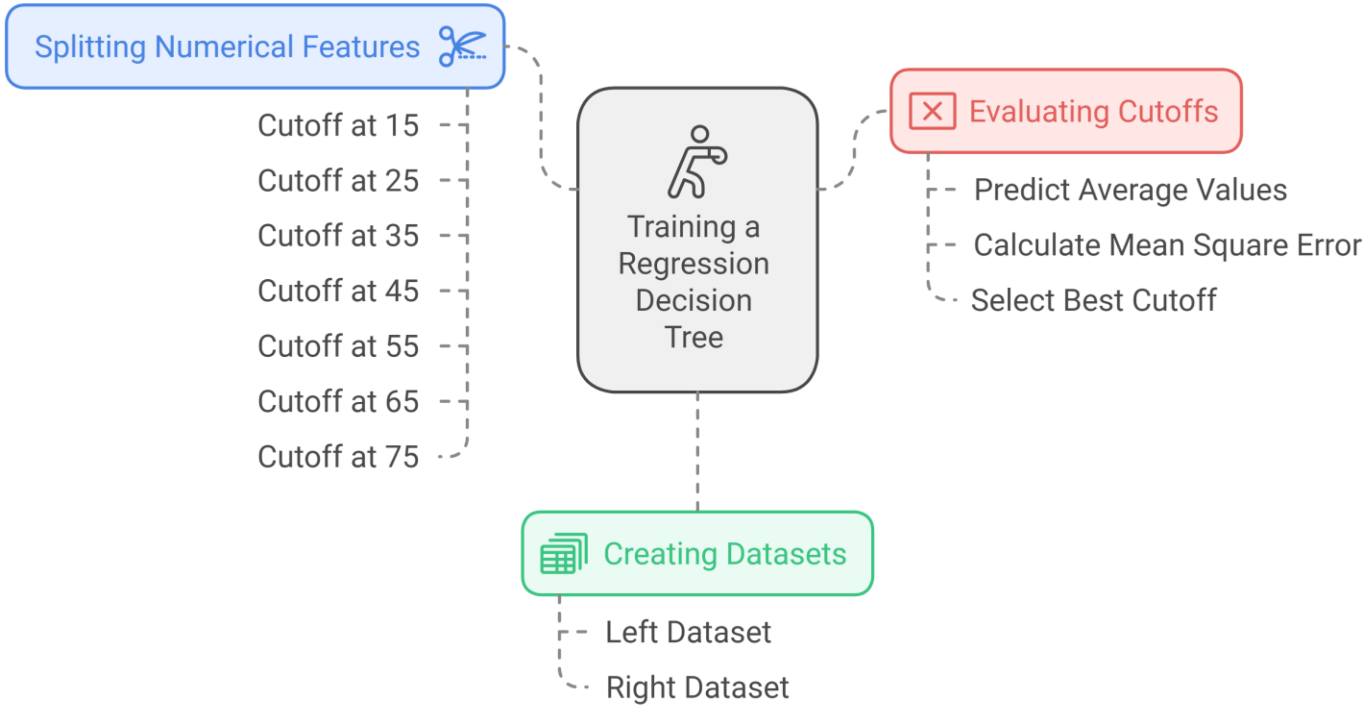
- Use the mean square error (MSE)



How Decision Trees Work for Regression

Example 01

- when a feature is numerical, we consider all the possible ways to split

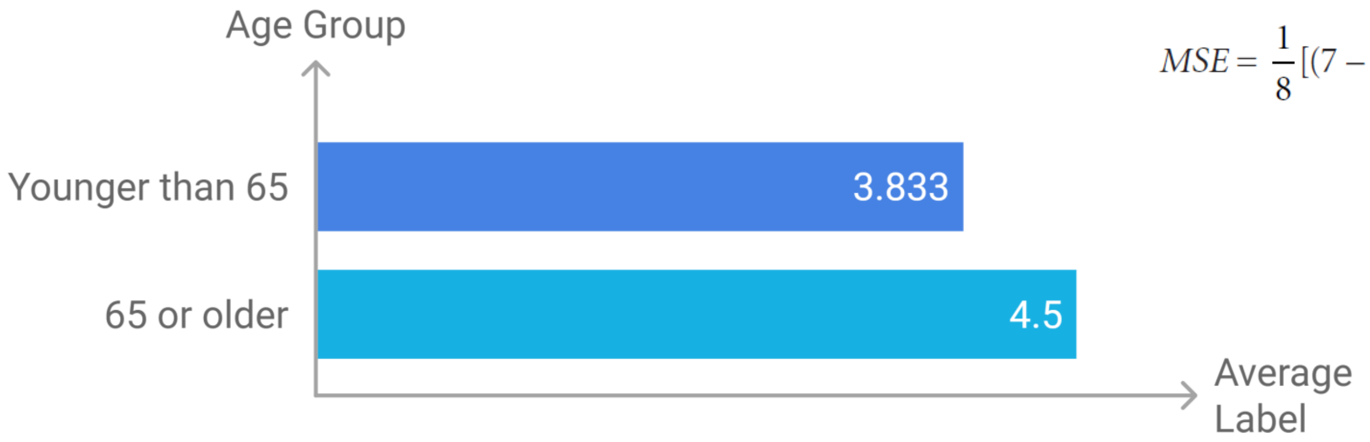


Age	Engagement
10	7
20	5
30	7
40	1
50	2
60	1
70	5
80	4

How Decision Trees Work for Regression

Example 01

- if our cutoff is 65, then the two datasets are the following:
  - Left dataset: users younger than 65. The labels are {7, 5, 7, 1, 2, 1}.
  - Right dataset: users 65 or older. The labels are {5,4}.



Average Labels for User Age Groups

$$MSE = \frac{1}{8}[(7-3.833)^2 + (5-3.833)^2 + (7-3.833)^2 + (1-3.833)^2 + (2-3.833)^2$$

$$+ (1-3.833)^2 + (5-4.5)^2 + (4-4.5)^2]$$

$$= 5.167$$

How Decision Trees Work for Regression

Example 01

- The **best cutoff** is at 35 years old. we’ve built the first decision node in our regression decision tree. The next steps are to continue splitting the left and right datasets recursively in the same fashion

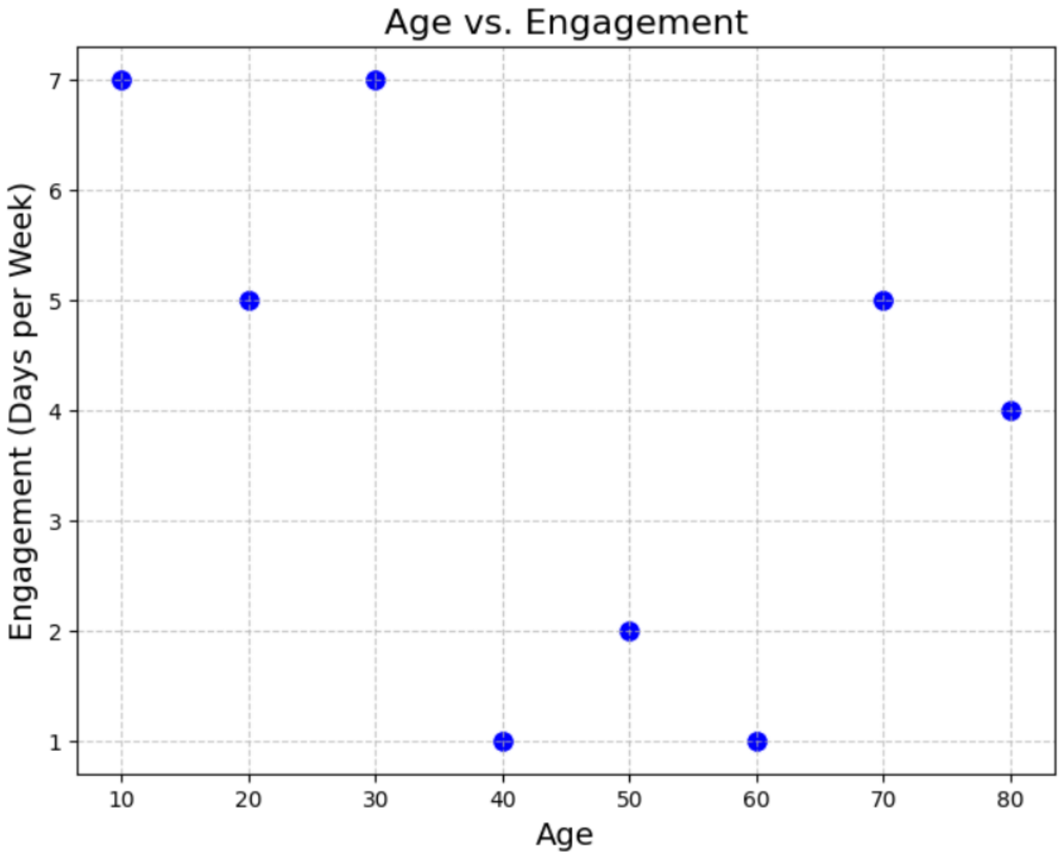
Cutoff	Labels left set	Labels right set	Prediction left set	Prediction right set	MSE
0	{}	{7,5,7,1,2,1,5,4}	None	4.0	5.25
15	{7}	{5,7,1,2,1,5,4}	7.0	3.571	3.964
25	{7,5}	{7,1,2,1,5,4}	6.0	3.333	3.917
35	**{7,5,7}**	**{1,2,1,5,4}**	**6.333**	**2.6**	**1.983**
45	{7,5,7,1}	{2,1,5,4}	5.0	3.0	4.25
55	{7,5,7,1,2}	{1,5,4}	4.4	3.333	4.983
65	{7,5,7,1,2,1}	{5,4}	3.833	4.5	5.167
75	{7,5,7,1,2,1,5}	{4}	4.0	4.0	5.25
100	{7,5,7,1,2,1,5,4}	{}	4.0	none	5.25

Decision Tree Regression Coding

- Creating Dataset

```python
# Creating the dataset
data = {
    'Age': [10, 20, 30, 40, 50, 60, 70, 80],
    'Engagement': [7, 5, 7, 1, 2, 1, 5, 4]
}

df = pd.DataFrame(data)
```

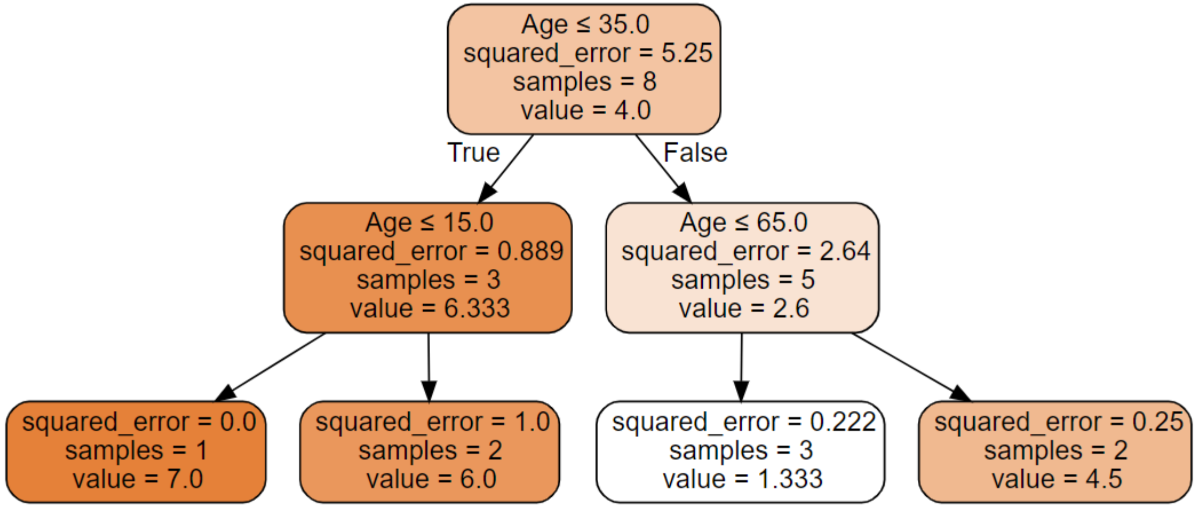
Age	Engagement
10	7
20	5
30	7
40	1
50	2
60	1
70	5
80	4



Decision Tree Regression Coding

- Training the Decision Tree Regressor

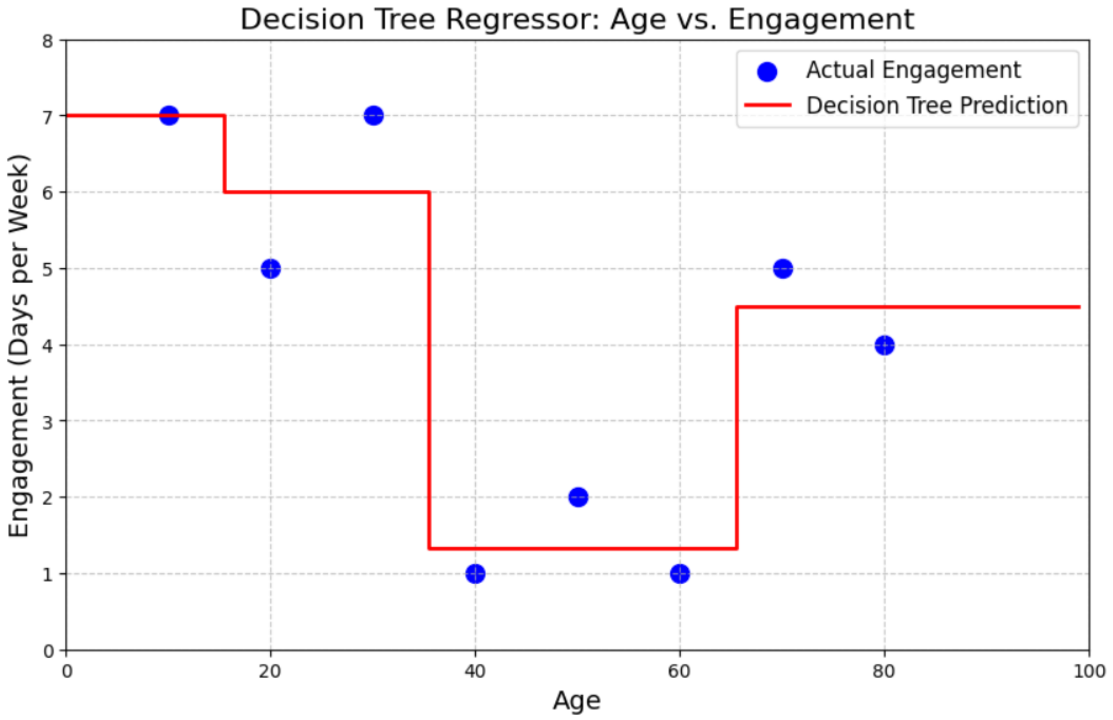
```python
# Initialize the regressor
regressor = DecisionTreeRegressor(random_state=42,max_depth=2)

# Train the model
regressor.fit(X, y)
```



Decision Tree Regression Coding

- Training the Decision Tree Regressor

```
# Initialize the regressor
regressor = DecisionTreeRegressor(random_state=42,max_depth=2)

# Train the model
regressor.fit(X, y)
```



Decision Tree Regression Coding

Example 02

- Let's build a regression tree using Scikit-Learn's **DecisionTreeRegressor** class, training it on a **noisy quadratic** dataset with **max_depth=2**

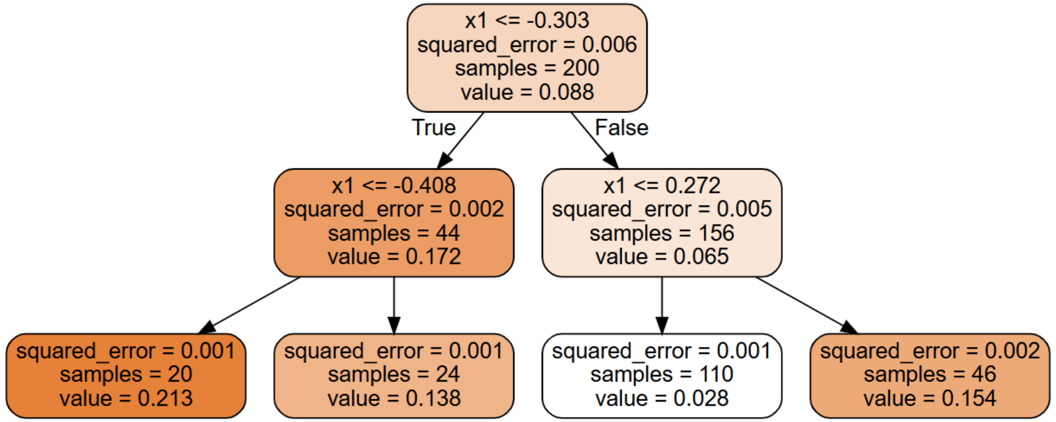
```
from sklearn.tree import DecisionTreeRegressor

np.random.seed(42)
X_quad = np.random.rand(200, 1) - 0.5  # a single random input feature
y_quad = X_quad ** 2 + 0.025 * np.random.randn(200, 1)

tree_reg = DecisionTreeRegressor(max_depth=2, random_state=42)
tree_reg.fit(X_quad, y_quad)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=2, random_state=42)
```

Decision Tree Regression Coding

Example 02

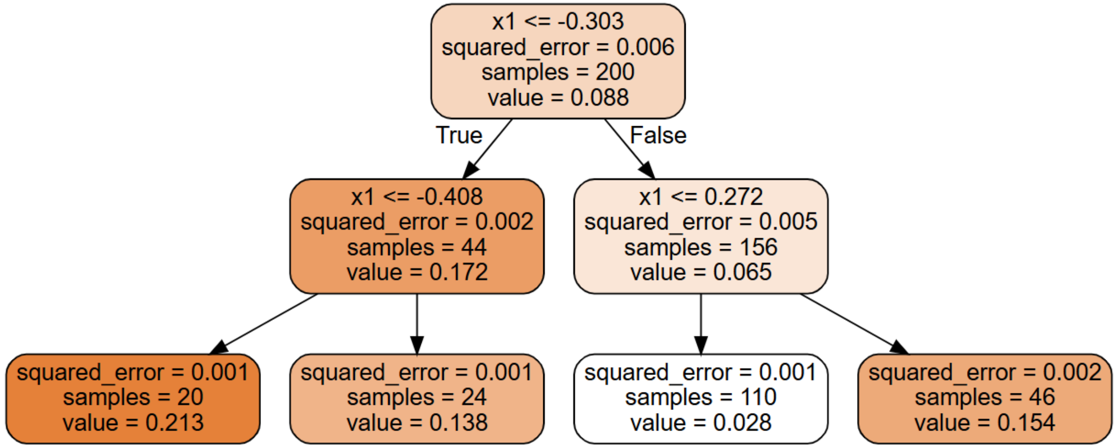
- suppose you want to **make a prediction** for a new instance with $x_1 = 0.2$. The root node asks whether $x_1 \leq -0.303$. Since it is not, the algorithm goes to the right child node, which asks whether $x_1 \leq 0.272$.
- Since it is, the algorithm goes to the left child node. This is a leaf node, and it predicts value=0.028



Decision Tree Regression Coding

Example 02

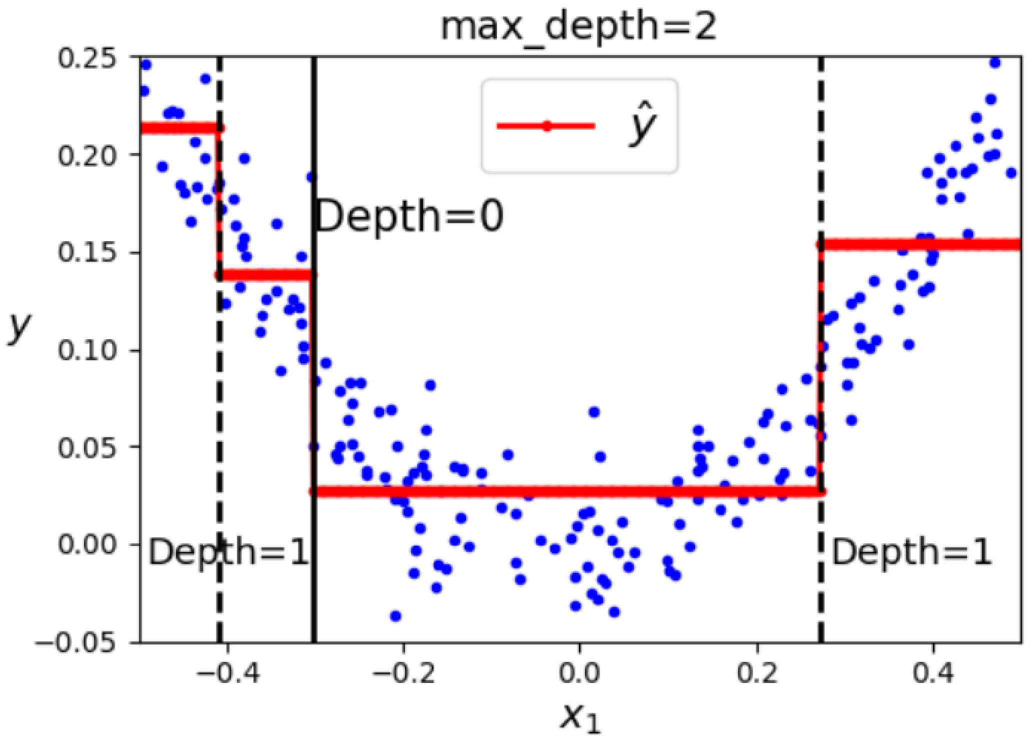
- This prediction is the average target value of the 110 training instances associated with this leaf node, and it results in a mean squared error equal to 0.001 over these 110 instances.

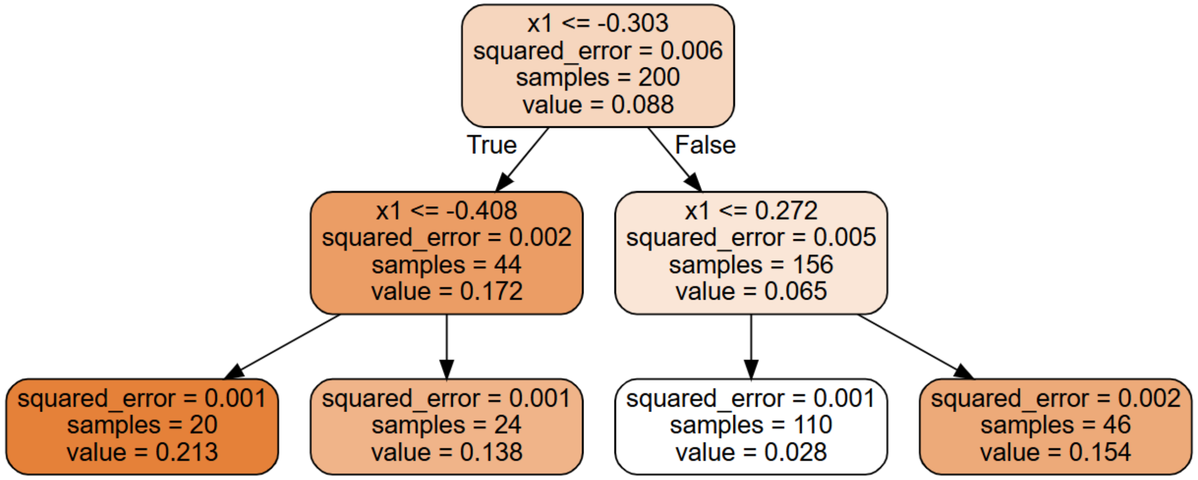


Decision Tree Regression Coding

Example 02

- The predicted value for each region is always the average target value of the instances in that region

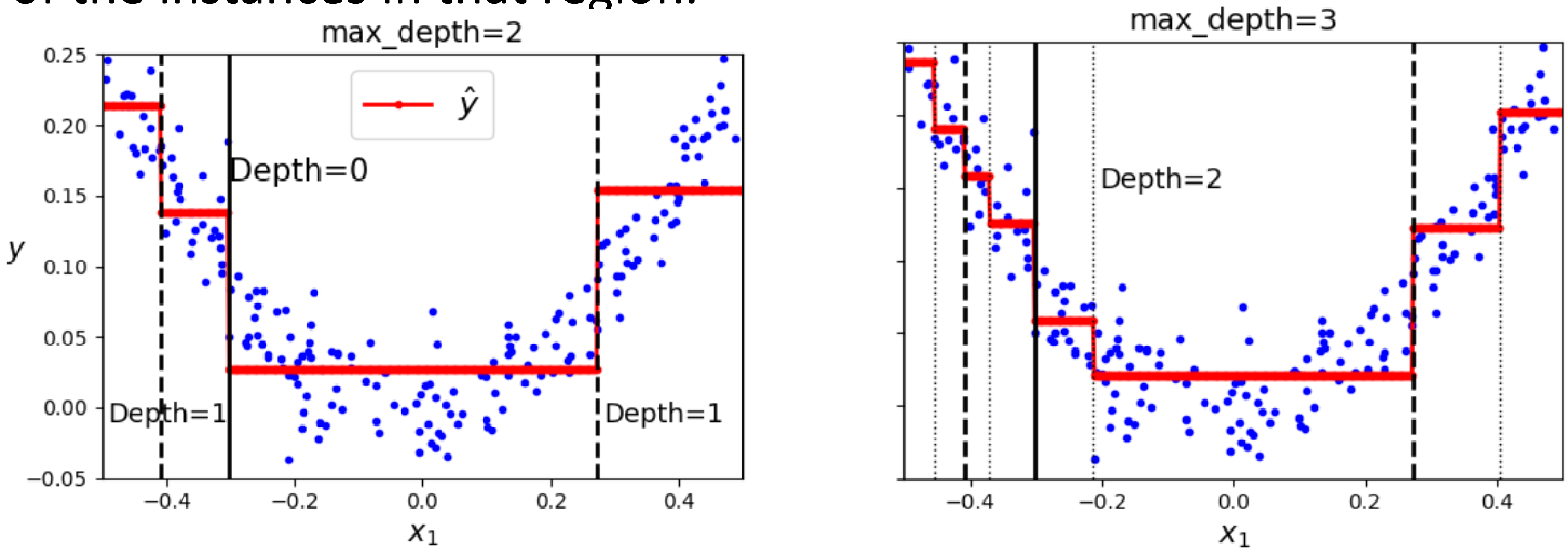




Decision Tree Regression Coding

Example 02

- The predicted value for each region is always the average target value of the instances in that region.



Regression Overfitting

- Just like for classification tasks, decision trees are prone to overfitting when dealing with regression tasks.