# Screencast and Discussion 14.1

## What is the main difference between linear regression and a regression tree?

Linear regression assumes a linear relationship and fits a straight line across all data. A regression tree, on the other hand, divides the input space into regions and fits constant values, enabling it to model non-linear patterns.

## Why is it that the initial regression tree generated from the training data should not be used?

The initial tree is usually unpruned, which can overfit the training data and lead to poor generalization on new data.

## If you get an R² value of close to 100% on your training data, what does that mean?

It suggests overfitting, where the model captures noise in addition to patterns, and is likely to perform poorly on unseen data.

## What can be done to optimize the initial regression tree from the training data?

Use cost complexity pruning to reduce overfitting and improve generalization.

## What are ccp alphas, and how are they used?

Cost Complexity Pruning Alpha (ccp\_alpha) is a regularization parameter that controls tree pruning. A higher value results in a simpler tree. A range of trees is built using different alphas, and cross-validation is used to find the best one.

## What is different about the final regression tree with an R² of 0.82 from the initial regression tree with an R² of 0.78?

The final tree is pruned and smaller, improving test performance and reducing overfitting.

## How would you further optimize the regression tree on the Bluebikes rentals dataset? What is the highest R² value you can get on the test data?

By encoding categorical variables (e.g., day\_of\_week), pruning the tree, and performing cross-validation. The highest R² obtained on the test data is 0.867.

To achieve this, I included one cell in the notebook.

# One-hot encode 'day\_of\_week' to convert it into numeric format

df\_encoded = pd.get\_dummies(df, columns=['day\_of\_week'], drop\_first=True)

# Define features and target

X = df\_encoded.drop('rentals', axis=1)

y = df\_encoded['rentals']

# Split into train and test sets

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state=42)

# Fit a DecisionTreeRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import r2\_score

tree = DecisionTreeRegressor(random\_state=42)

tree.fit(X\_train, y\_train)

y\_pred = tree.predict(X\_test)

r2\_test = r2\_score(y\_test, y\_pred)

print(f'Improved R^2 on test data: {r2\_test:.4f}')

**Improved R^2 on test data: 0.8669**

## What technique was successful at optimizing the classification model of loan worthiness and why was this technique needed?

Pruning via cost complexity (ccp\_alpha) simplified the model, reduced overfitting, and improved accuracy on new data.