Decision Tree Challenge

Feature Importance and Categorical Variable Encoding

Decision Tree Challenge - Feature Importance and Variable Encoding

Challenge Overview

Your Mission: Create a simple GitHub Pages site that demonstrates how decision trees measure feature importance and analyzes the critical differences between categorical and numerical variable encoding. You'll answer two key discussion questions by adding narrative to a pre-built analysis and posting those answers to your GitHub Pages site as a rendered HTML document.

> ⚠ AI Partnership Required
>
> This challenge pushes boundaries intentionally. You'll tackle problems that normally require weeks of study, but with Cursor AI as your partner (and your brain keeping it honest), you can accomplish more than you thought possible.
> **The new reality:** The four stages of competence are Ignorance → Awareness → Learning → Mastery. AI lets us produce Mastery-level work while operating primarily in the Awareness stage. I focus on awareness training, you leverage AI for execution, and together we create outputs that used to require years of dedicated study.

The Decision Tree Problem

> "The most important thing in communication is hearing what isn't said." - Peter Drucker

The Core Problem: Decision trees are often praised for their interpretability and ability to handle both numerical and categorical variables. But what happens when we encode categorical variables as numbers? How does this affect our understanding of feature importance?

What is Feature Importance? In decision trees, feature importance measures how much each variable contributes to reducing impurity (or improving prediction accuracy) across all splits in the tree. It's a key metric for understanding which variables matter most for your predictions.

> **!** The Key Insight: Encoding Matters for Interpretability
>
> **The problem:** When we encode categorical variables as numerical values (like 1, 2, 3, 4…), decision trees treat them as if they have a meaningful numerical order. This can completely distort our analysis.
> **The Real-World Context:** In real estate, we know that neighborhood quality, house style, and other categorical factors are crucial for predicting home prices. But if we encode these as numbers, we might get misleading insights about which features actually matter most.
> **The Devastating Reality:** Even sophisticated machine learning models can give us completely wrong insights about feature importance if we don't properly encode our variables. A categorical variable that should be among the most important might appear irrelevant, while a numerical variable might appear artificially important.

Let's assume we want to predict house prices and understand which features matter most. The key question is: **How does encoding categorical variables as numbers affect our understanding of feature importance?**

The Ames Housing Dataset

We are analyzing the Ames Housing dataset which contains detailed information about residential properties sold in Ames, Iowa from 2006 to 2010. This dataset is perfect for our analysis because it contains a categorical variable (like zip code) and numerical variables (like square footage, year built, number of bedrooms).

The Problem: ZipCode as Numerical vs Categorical

Key Question: What happens when we treat zipCode as a numerical variable in a decision tree? How does this affect feature importance interpretation?

The Issue: Zip codes (50010, 50011, 50012, 50013) are categorical variables representing discrete geographic areas, i.e. neighborhoods. When treated as numerical, the tree might split on "zipCode $>$ 50012.5" - which has no meaningful interpretation for house prices. Zip codes are non-ordinal categorical variables meaning they have no inherent order that aids house price prediction (i.e. zip code 99999 is not the priceiest zip code).

Data Loading and Model Building

> **!** Note on Python Usage
>
> You have not been coached through setting up a Python environment. You will need to set up a Python environment and install the necessary packages to run this code - takes about 15 minutes; see https://quarto.org/docs/projects/virtual-environments.html. Alternatively, delete the Python code and only leave the remaining R code that is provided. You can see the executed Python output at my GitHub pages site: https://flyaflya.github.io/decTreeChallenge/.

R

```
# Load libraries
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(rpart))
if (!require(rpart.plot, quietly = TRUE)) {
  install.packages("rpart.plot", repos = "https://cran.rstudio.com/")
  library(rpart.plot)
}

# Load data
sales_data <- read.csv("https://raw.githubusercontent.com/flyaflya/buad442Fall2025/refs/heads

# Prepare model data (treating zipCode as numerical)
model_data <- sales_data %>%
  select(SalePrice, LotArea, YearBuilt, GrLivArea, FullBath, HalfBath,
         BedroomAbvGr, TotRmsAbvGrd, GarageCars, zipCode) %>%
  na.omit()

# Split data
set.seed(123)
train_indices <- sample(1:nrow(model_data), 0.8 * nrow(model_data))
train_data <- model_data[train_indices, ]
test_data <- model_data[-train_indices, ]

# Build decision tree
tree_model <- rpart(SalePrice ~ .,
                    data = train_data,
                    method = "anova",
                    control = rpart.control(maxdepth = 3,
```

```
                                          minsplit = 20,
                                          minbucket = 10))

cat("Model built with", sum(tree_model$frame$var == "<leaf>"), "terminal nodes\n")
```

```
Model built with 7 terminal nodes
```

Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
import warnings
warnings.filterwarnings('ignore')

# Load data
sales_data = pd.read_csv("https://raw.githubusercontent.com/flyaflya/buad442Fall2025/refs/hea

# Prepare model data (treating zipCode as numerical)
model_vars = ['SalePrice', 'LotArea', 'YearBuilt', 'GrLivArea', 'FullBath',
              'HalfBath', 'BedroomAbvGr', 'TotRmsAbvGrd', 'GarageCars', 'zipCode']
model_data = sales_data[model_vars].dropna()

# Split data
X = model_data.drop('SalePrice', axis=1)
y = model_data['SalePrice']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)

# Build decision tree
tree_model = DecisionTreeRegressor(max_depth=3,
                                   min_samples_split=20,
                                   min_samples_leaf=10,
                                   random_state=123)
tree_model.fit(X_train, y_train)
```

```
DecisionTreeRegressor(max_depth=3, min_samples_leaf=10, min_samples_split=20,
                      random_state=123)
```

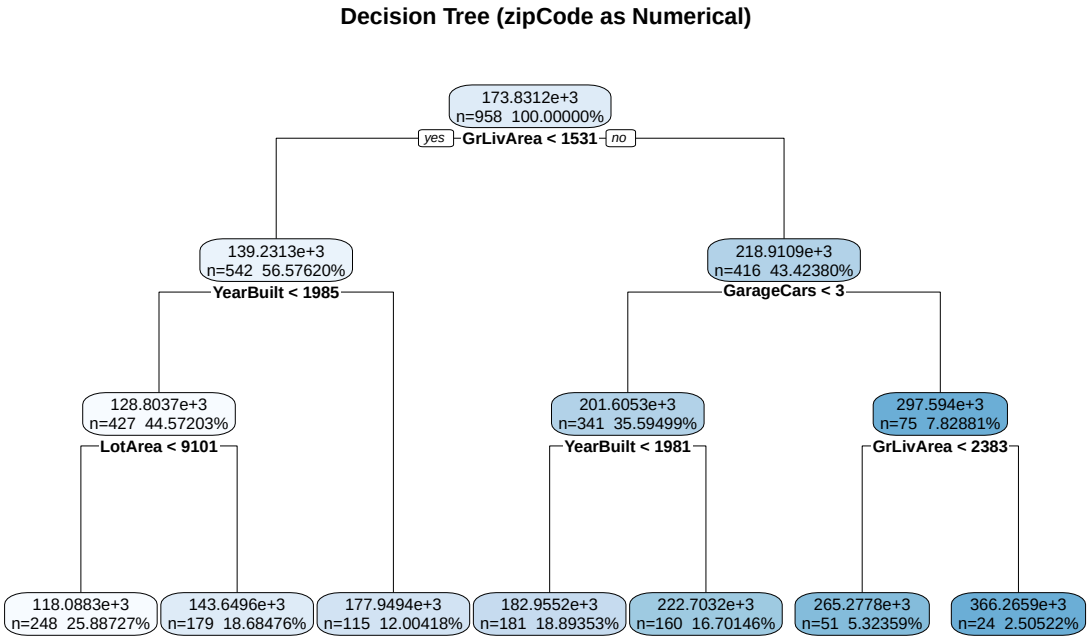
```python
print(f"Model built with {tree_model.get_n_leaves()} terminal nodes")
```

```
Model built with 8 terminal nodes
```

Tree Visualization

R

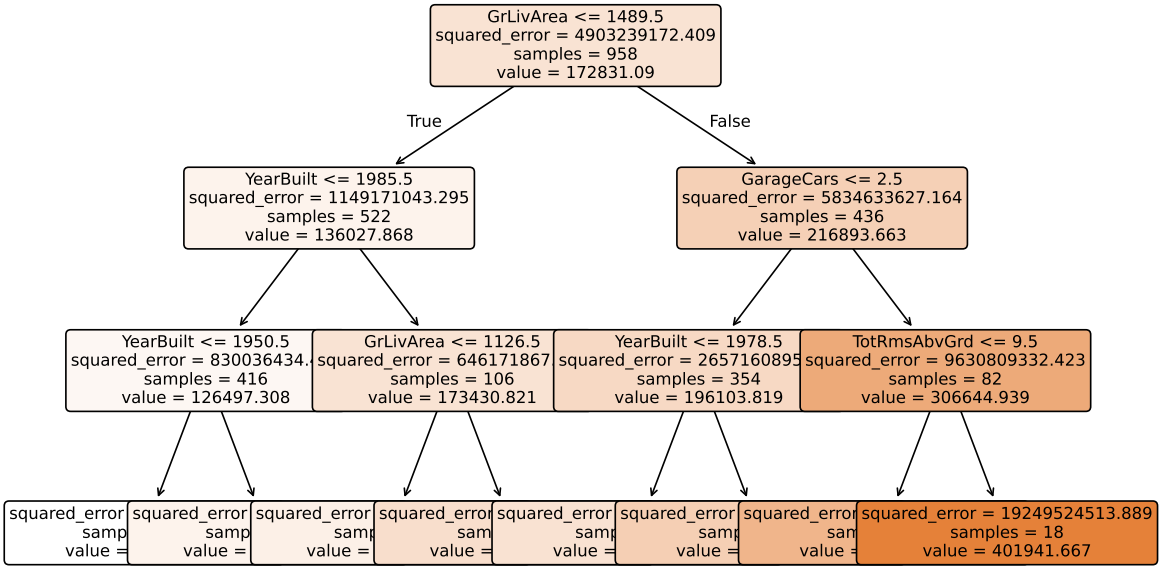
```r
# Visualize tree
if (require(rpart.plot, quietly = TRUE)) {
  rpart.plot(tree_model,
             type = 2,
             extra = 101,
             fallen.leaves = TRUE,
             digits = 0,
             cex = 0.8,
             main = "Decision Tree (zipCode as Numerical)")
} else {
  plot(tree_model, uniform = TRUE, main = "Decision Tree (zipCode as Numerical)")
  text(tree_model, use.n = TRUE, all = TRUE, cex = 0.8)
}
```



Python

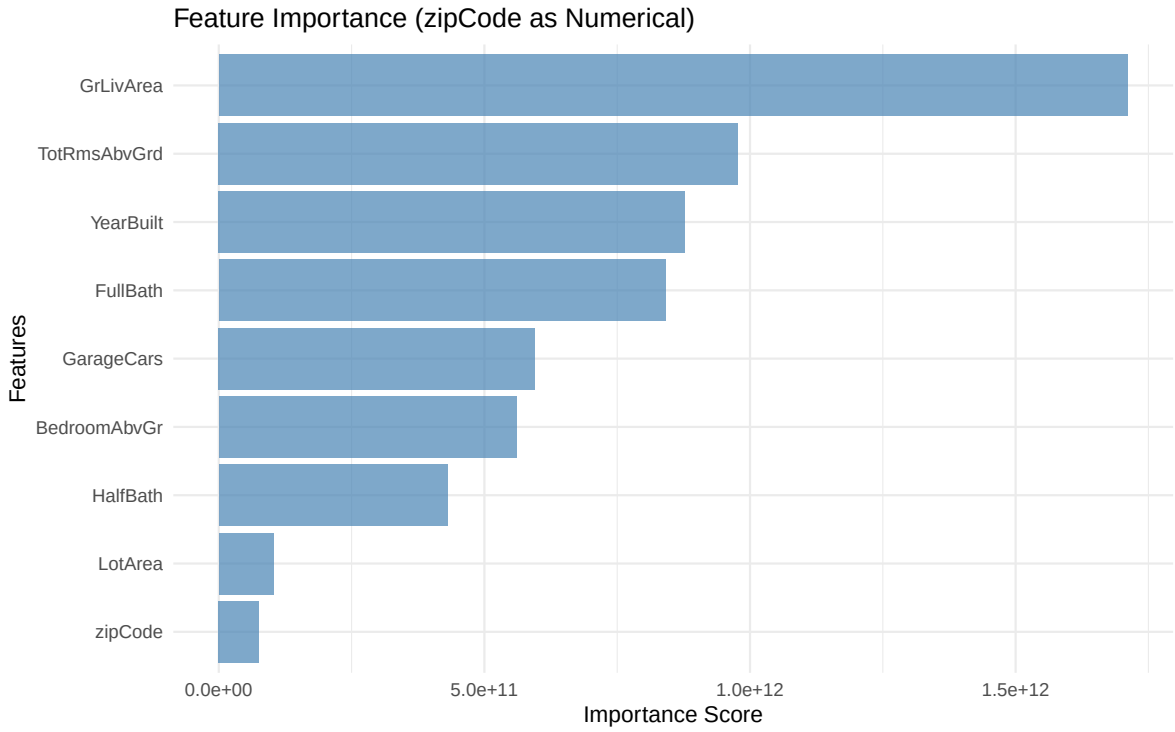
```
# Visualize tree
plt.figure(figsize=(10, 6))
plot_tree(tree_model,
          feature_names=X_train.columns,
          filled=True,
          rounded=True,
          fontsize=10,
          max_depth=3)
plt.title("Decision Tree (zipCode as Numerical)")
plt.tight_layout()
plt.show()
```

Decision Tree (zipCode as Numerical)



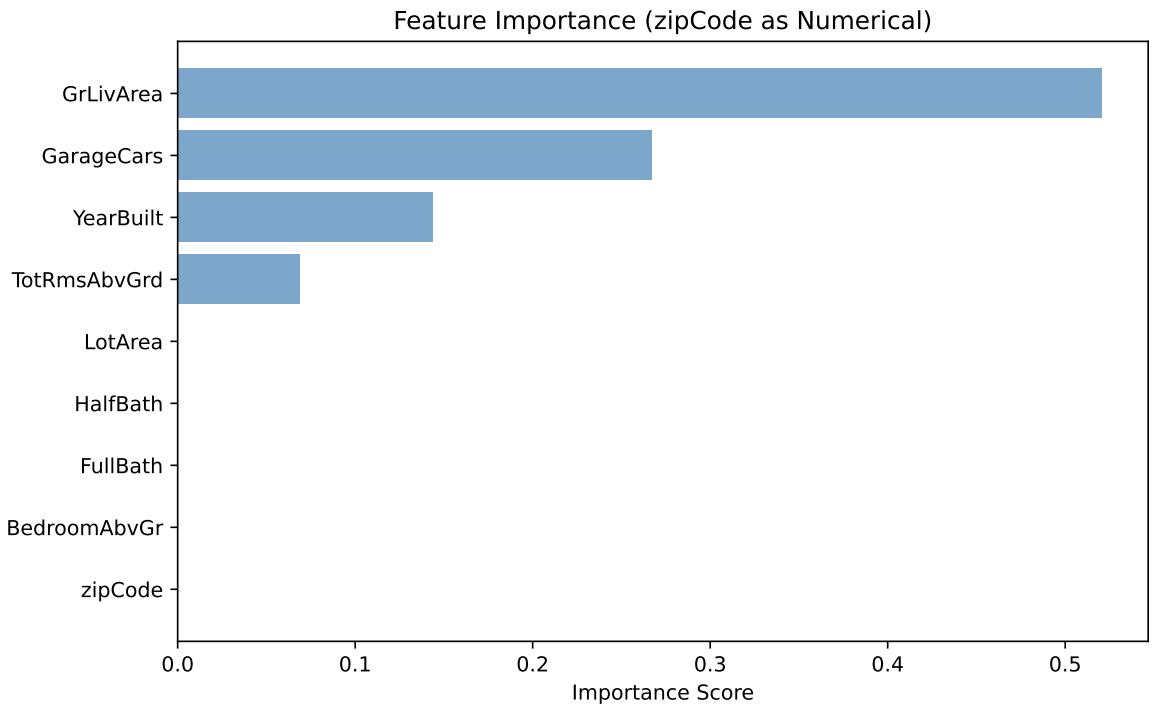
Feature Importance Analysis

R



Python

```
([<matplotlib.axis.YTick object at 0x000001EA11499910>, <matplotlib.axis.YTick object at 0x0
```



Critical Analysis: The Encoding Problem

The Problem Revealed

What to note: Our decision tree treated `zipCode` as a numerical variable. This leads to zip code being unimportant. Not surprisingly, because there is no reason to believe allowing splits like "zipCode $<$ 50012.5" should be beneficial for house price prediction. This false coding of a variable creates several problems:

1. **Potentially Meaningless Splits:** A zip code of 50013 is not "greater than" 50012 in any meaningful way for house prices
2. **False Importance:** The algorithm assigns importance to zipCode based on numerical splits rather than categorical distinctions OR the importance of zip code is completely missed as numerical ordering has no inherent relationship to house prices.
3. **Misleading Interpretations:** We might conclude zipCode is not important when our intuition tells us it should be important (listen to your intuition).

The Real Issue: Zip codes are categorical variables representing discrete geographic areas. The numerical values have no inherent order or magnitude relationship to house

prices. These must be modelled as categorical variables.

Proper Categorical Encoding: The Solution

Now let's repeat the analysis with zipCode properly encoded as categorical variables to see the difference.

R Approach: Convert zipCode to a factor (categorical variable)
Python Approach: One-hot encode zipCode (create dummy variables for each zip code)

Categorical Encoding Analysis

R

```
# Convert zipCode to factor (categorical)
model_data_cat <- model_data %>%
  mutate(zipCode = as.factor(zipCode))

# Split data
set.seed(123)
train_indices_cat <- sample(1:nrow(model_data_cat), 0.8 * nrow(model_data_cat))
train_data_cat <- model_data_cat[train_indices_cat, ]
test_data_cat <- model_data_cat[-train_indices_cat, ]

# Build decision tree with categorical zipCode
tree_model_cat <- rpart(SalePrice ~ .,
                        data = train_data_cat,
                        method = "anova",
                        control = rpart.control(maxdepth = 3,
                                                minsplit = 20,
                                                minbucket = 10))

# Feature importance with categorical zipCode
importance_cat <- data.frame(
  Feature = names(tree_model_cat$variable.importance),
  Importance = as.numeric(tree_model_cat$variable.importance)
) %>%
  arrange(desc(Importance)) %>%
  mutate(Importance_Percent = round(Importance / sum(Importance) * 100, 2))
```

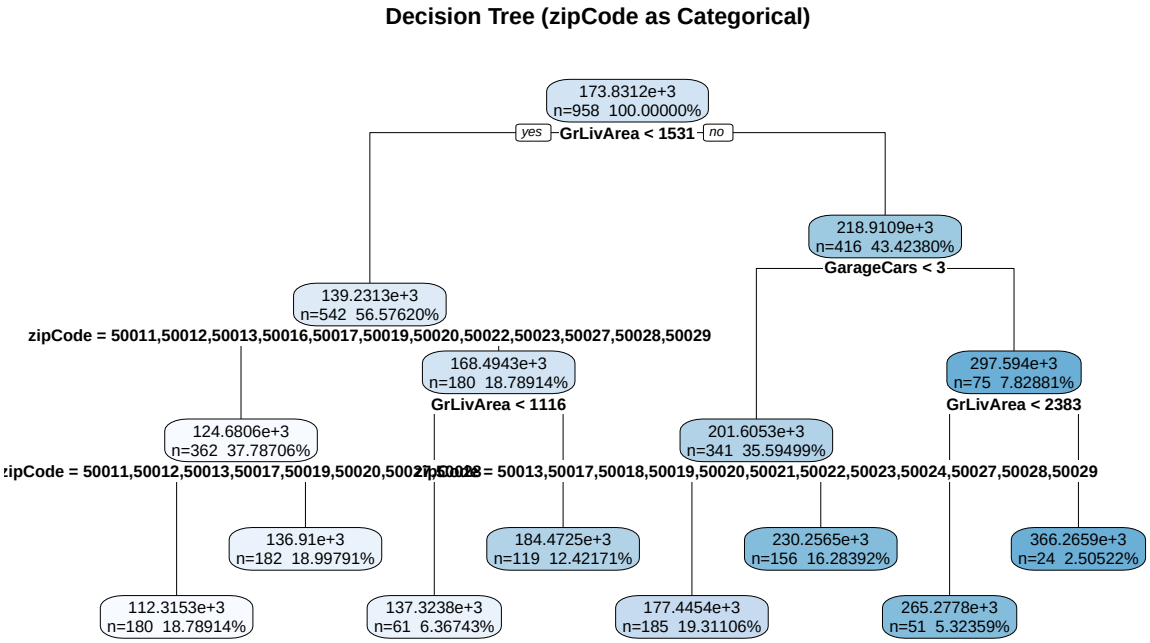
```r
# Check if zipCode appears in tree
zipcode_in_tree <- "zipCode" %in% names(tree_model_cat$variable.importance)
if(zipcode_in_tree) {
  zipcode_rank_cat <- which(importance_cat$Feature == "zipCode")
}
```

Python

Tree Visualization: Categorical zipCode

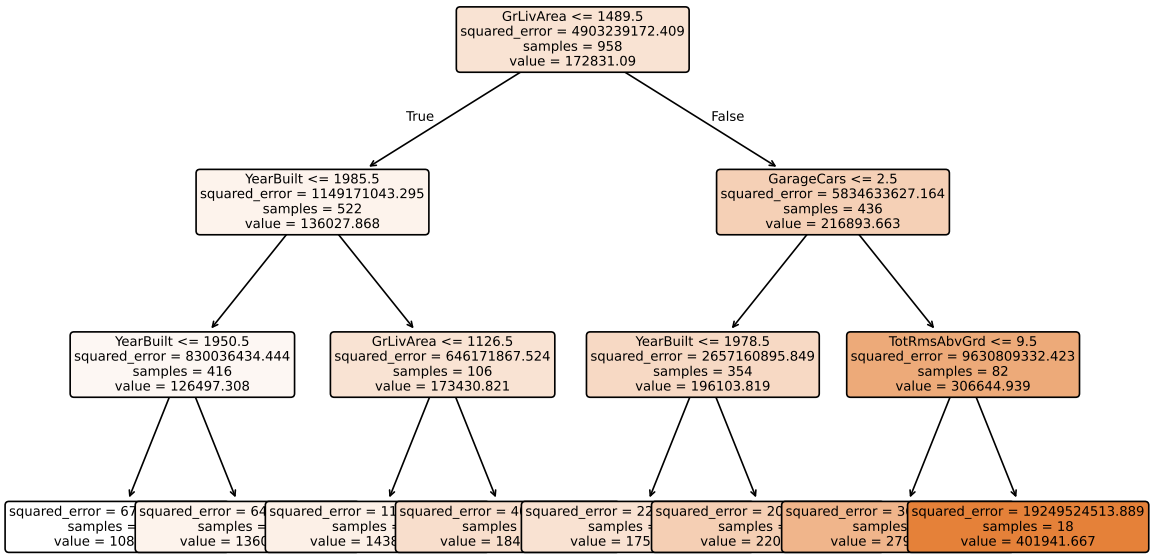
R

```r
# Visualize tree with categorical zipCode
if (require(rpart.plot, quietly = TRUE)) {
  rpart.plot(tree_model_cat,
             type = 2,
             extra = 101,
             fallen.leaves = TRUE,
             digits = 0,
             cex = 0.8,
             main = "Decision Tree (zipCode as Categorical)")
} else {
  plot(tree_model_cat, uniform = TRUE, main = "Decision Tree (zipCode as Categorical)")
  text(tree_model_cat, use.n = TRUE, all = TRUE, cex = 0.8)
}
```



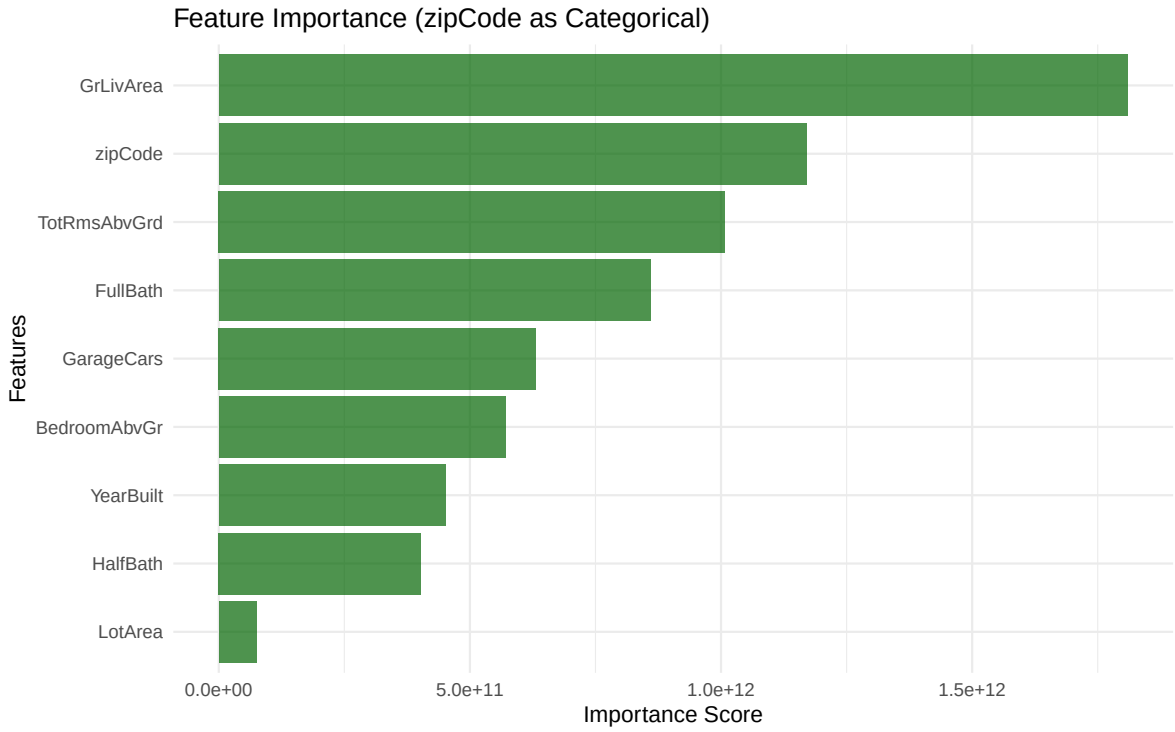
Python

```
# Visualize tree with one-hot encoded zipCode
plt.figure(figsize=(10, 6))
plot_tree(tree_model_cat,
          feature_names=X_train_cat.columns,
          filled=True,
          rounded=True,
          fontsize=8,
          max_depth=4)
plt.title("Decision Tree (zipCode One-Hot Encoded)")
plt.tight_layout()
plt.show()
```


Decision Tree (zipCode One-Hot Encoded)
GrLivArea <= 1489.5
squared_error = 4903239172.409
samples = 958
value = 172831.09
True
False
YearBuilt <= 1985.5
squared_error = 1149171043.295
samples = 522
value = 136027.868
GarageCars <= 2.5
squared_error = 5834633627.164
samples = 436
value = 216893.663
YearBuilt <= 1950.5
squared_error = 830036434.444
samples = 416
value = 126497.308
GrLivArea <= 1126.5
squared_error = 646171867.524
samples = 106
value = 173430.821
YearBuilt <= 1978.5
squared_error = 2657160895.849
samples = 354
value = 196103.819
TotRmsAbvGrd <= 9.5
squared_error = 9630809332.423
samples = 82
value = 306644.939
squared_error = 19249524513.889
samples = 18
value = 401941.667

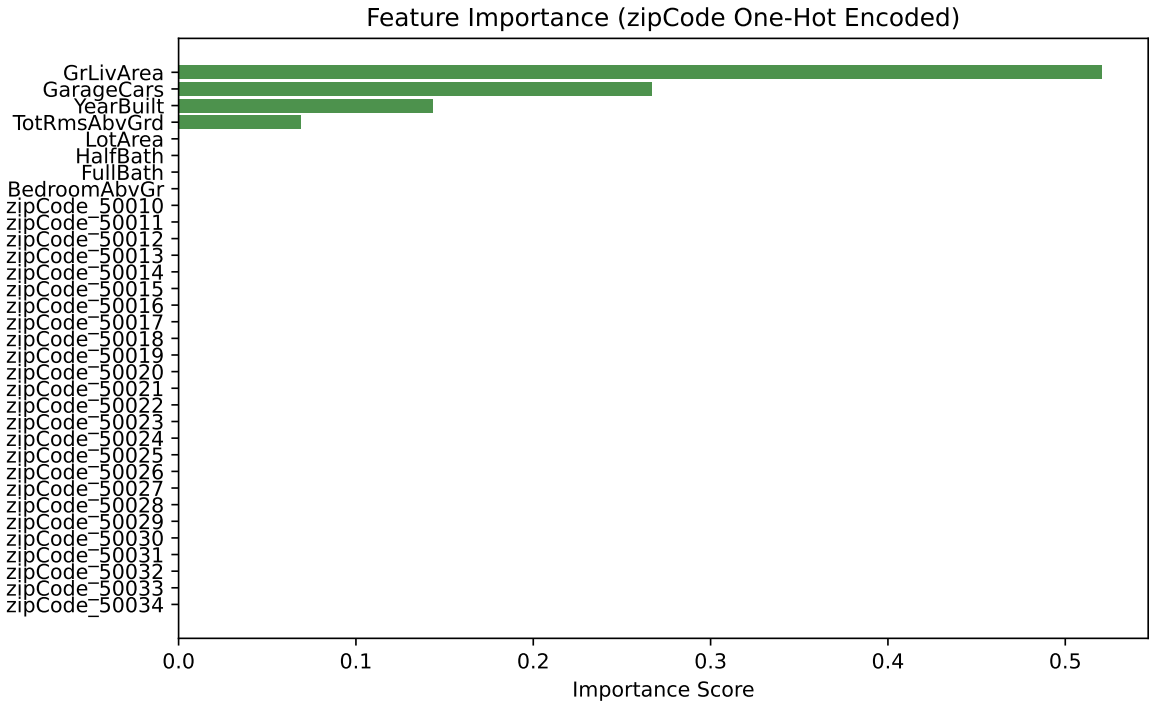
Feature Importance: Categorical zipCode

R



Python

```
([<matplotlib.axis.YTick object at 0x000001EA115539B0>, <matplotlib.axis.YTick object at 0x0
```



Challenge Requirements

Minimum Requirements for Any Points on Challenge

1. **Create a GitHub Pages Site:** Use the starter repository (see Repository Setup section below) to begin with a working template. The repository includes all the analysis code and visualizations above.
2. **Add Discussion Narrative:** Add your answers to the two discussion questions below in the Discussion Questions section of the rendered HTML.
3. **GitHub Repository:** Use your forked repository (from the starter repository) named "decTreeChallenge" in your GitHub account.
4. **GitHub Pages Setup:** The repository should be made the source of your github pages:
    - Go to your repository settings (click the "Settings" tab in your GitHub repository)
    - Scroll down to the "Pages" section in the left sidebar
    - Under "Source", select "Deploy from a branch"
    - Choose "main" branch and "/ (root)" folder
    - Click "Save"
    - Your site will be available at: `https://[your-username].github.io/decTreeChallenge/`

- **Note:** It may take a few minutes for the site to become available after enabling Pages

Getting Started: Repository Setup

Quick Start with Starter Repository

Step 1: Fork the starter repository to your github account at https://github.com/flyaflya/decTreeChallenge.git
Step 2: Clone your fork locally using Cursor (or VS Code)
Step 3: You're ready to start! The repository includes pre-loaded data and a working template with all the analysis above.

Why Use the Starter Repository?

Benefits:

- **Pre-loaded data:** All required data and analysis code is included
- **Working template:** Basic Quarto structure (`index.qmd`) is ready
- **No setup errors:** Avoid common data loading issues
- **Focus on analysis:** Spend time on the discussion questions, not data preparation

Getting Started Tips

Navy SEALs Motto

"Slow is Smooth and Smooth is Fast"

Take your time to understand the decision tree mechanics, plan your approach carefully, and execute with precision. Rushing through this challenge will only lead to errors and confusion.

Important: Save Your Work Frequently!

Before you start: Make sure to commit your work often using the Source Control panel in Cursor (Ctrl+Shift+G or Cmd+Shift+G). This prevents the AI from overwriting your progress and ensures you don't lose your work.
Commit after each major step:

- After adding your discussion answers

- After rendering to HTML
- Before asking the AI for help with new code

How to commit:

1. Open Source Control panel (Ctrl+Shift+G)
2. Stage your changes (+ button)
3. Write a descriptive commit message
4. Click the checkmark to commit

Remember: Frequent commits are your safety net!

Discussion Questions for Challenge

Your Task: Add thoughtful narrative answers to these two questions in the Discussion Questions section of your rendered HTML site.

1. **Numerical vs Categorical Encoding:** There are four models above, two in R and two in Python. For each language, the models differ by how zip code is modelled, either as a numerical variable or as a categorical variable. Given what you know about zip codes and real estate prices, how should zip code be modelled, numerically or categorically?
2. **R vs Python Implementation Differences:** When modelling zip code as a categorical variable, the output tree and feature importance differs quite significantly between R and Python. Investigate why this is the case. Which language would you say does a better job of modelling zip code as a categorical variable? Why is this the case? Do you see any documentation suggesting the other language does a better job? If so, please provide a quote from the documentation.

! Important Note on AI Usage

No coding assistance expected: This challenge is designed for independent analysis and critical thinking. You are **not** expected to use AI for coding help or to write code for you. You are **not** expected to code at all unless curious about any ideas you may have.
AI as unreliable thought partner: If you choose to use AI tools, treat them as an unreliable thought partner. AI responses should be verified and cross-checked rather than accepted at face value.
Documentation verification required: For question 2, you must investigate the official documentation for both `rpart` (R) and `sklearn.tree.DecisionTreeRegressor` (Python) to understand why the two implementations yield vastly different results when handling categorical variables. Your analysis should be grounded in the actual documentation and technical specifications of these libraries, not AI-generated explanations. AI

can be helpful in digesting the documentation, but you must verify the information is correct.

Grading Rubric

! What You're Really Being Graded On

This is an investigative report, not a coding exercise. You're analyzing decision tree models and reporting your findings like a professional analyst would. Think of this as a brief you'd write for a client or manager about why proper variable encoding matters in machine learning.
What makes a great report:

- **Clear narrative:** Tell the story of what you discovered about decision tree feature importance
- **Insightful analysis:** Focus on the most interesting differences between numerical and categorical encoding
- **Professional presentation:** Clean, readable, and engaging
- **Concise conclusions:** No AI babble or unnecessary technical jargon
- **Human insights:** Your interpretation of what the feature importance rankings actually mean (or don't mean)
- **Documentation-based analysis:** For question 2, ground your analysis in actual library documentation

What we're looking for: A compelling 1-2 minute read that demonstrates both the power of decision trees for interpretability and the critical importance of proper variable encoding.

Questions to Answer for 75% Grade on Challenge

1. **Numerical vs Categorical Analysis:** Provide a clear, well-reasoned answer to question 1 about how zip codes should be modelled. Your answer should demonstrate understanding of why categorical variables need special treatment in decision trees.

Questions to Answer for 85% Grade on Challenge

2. **R vs Python Implementation Analysis:** Provide a thorough analysis of question 2, including investigation of the official documentation for both `rpart` (R) and `sklearn.tree.DecisionTreeRegressor` (Python). Your analysis should explain the

technical differences and provide a reasoned opinion about which implementation handles categorical variables better.

Questions to Answer for 95% Grade on Challenge

3. **Professional Presentation:** Your discussion answers should be written in a professional, engaging style that would be appropriate for a business audience. Avoid technical jargon and focus on practical implications. Use Quarto markdown linking to create a link to the discussion section from the top of the page (see https://quarto.org/docs/authoring/cross-references.html#sections).

Questions to Answer for 100% Grade on Challenge

4. **Documentation Integration:** For question 2, include a specific quote from the official documentation of `sklearn.tree.DecisionTreeRegressor` that supports your analysis.

Submission Checklist

Minimum Requirements (Required for Any Points):

- ☐ Forked starter repository from https://github.com/flyaflya/decTreeChallenge.git
- ☐ Cloned repository locally using Cursor (or VS Code)
- ☐ Added thoughtful narrative answers to both discussion questions
- ☐ Document rendered to HTML successfully
- ☐ HTML files uploaded to your forked repository
- ☐ GitHub Pages enabled and working
- ☐ Site accessible at `https://[your-username].github.io/decTreeChallenge/`

75% Grade Requirements:

- ☐ Clear, well-reasoned answer to question 1 about numerical vs categorical encoding

85% Grade Requirements:

- ☐ Thorough analysis of question 2 with investigation of official documentation

95% Grade Requirements:

- ☐ Professional presentation style appropriate for business audience with links to the discussion section from the top of the page (see https://quarto.org/docs/authoring/cross-references.html#sections).

100% Grade Requirements:

- ☐ Specific quote from official documentation of `sklearn.tree.DecisionTreeRegressor` supporting your analysis

Report Quality (Critical for Higher Grades):

- ☐ Clear, engaging narrative that tells a story
- ☐ Focus on the most interesting findings about decision tree feature importance
- ☐ Professional writing style (no AI-generated fluff)
- ☐ Concise analysis that gets to the point
- ☐ Practical insights that would help a real data scientist
- ☐ Documentation-based analysis for technical questions