Decision Tree Challenge

Feature Importance and Categorical Variable Encoding

Decision Tree Challenge: Feature Importance and Variable Encoding

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. When treated as numerical, the tree might split on "zipCode > 50012.5" - which has no meaningful interpretation for house prices.

Data Loading and Model Building

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 %>%
```

```r
  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 8 terminal nodes
```

Python

```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()
```

```python
# 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)
```

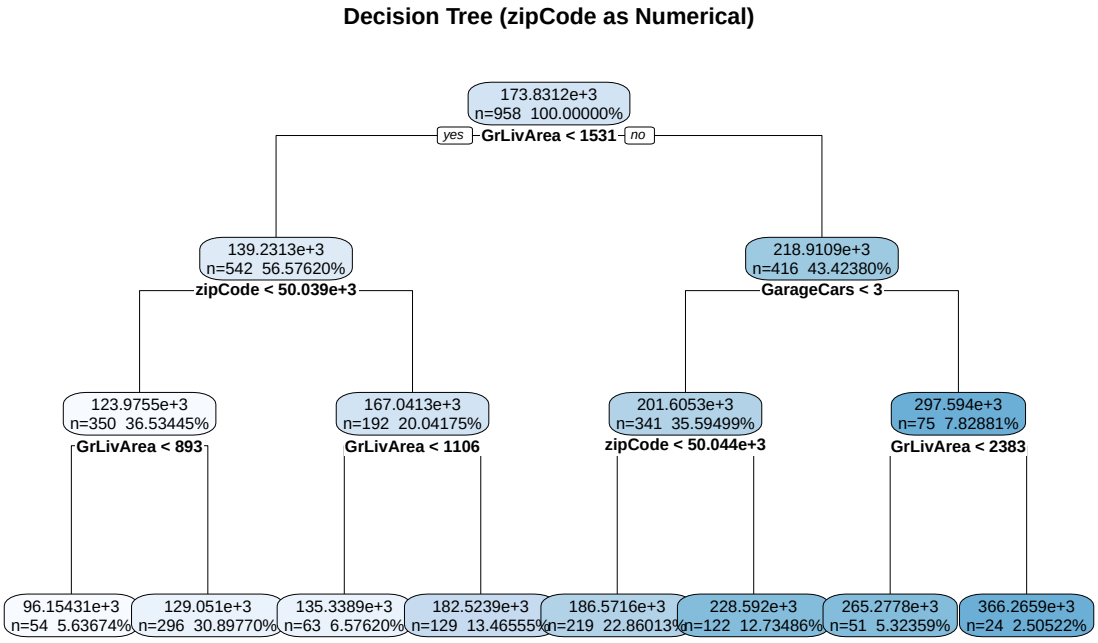
```python
print(f"Model built with {tree_model.get_n_leaves()} terminal nodes")
```

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

Tree Visualization

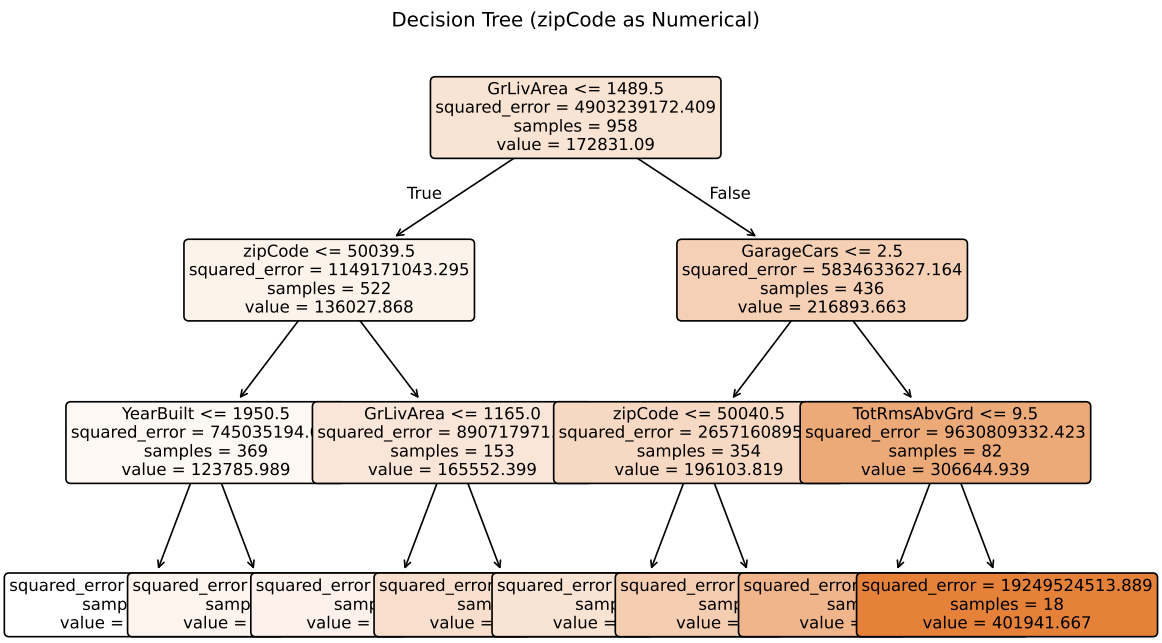
R

```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()
```



Feature Importance Analysis

R

```r
# Extract and display feature importance
importance_df <- data.frame(
  Feature = names(tree_model$variable.importance),
  Importance = as.numeric(tree_model$variable.importance)
) %>%
  arrange(desc(Importance)) %>%
  mutate(Importance_Percent = round(Importance / sum(Importance) * 100, 2))

cat("Feature Importance Rankings:\n")
```

```
Feature Importance Rankings:
```

```r
print(importance_df)
```

```
       Feature   Importance Importance_Percent
```

```
1    GrLivArea 1.864741e+12             28.65
2 TotRmsAbvGrd 1.003386e+12             15.42
3     FullBath 8.468023e+11             13.01
4    YearBuilt 7.219815e+11             11.09
5   GarageCars 6.209281e+11              9.54
6 BedroomAbvGr 5.387957e+11              8.28
7      zipCode 4.569801e+11              7.02
8     HalfBath 4.022969e+11              6.18
9      LotArea 5.178462e+10              0.80
```

```r
# Check zipCode ranking
zipcode_rank <- which(importance_df$Feature == "zipCode")
zipcode_importance <- importance_df$Importance_Percent[zipcode_rank]

cat("\nKey Finding:\n")
```

Key Finding:

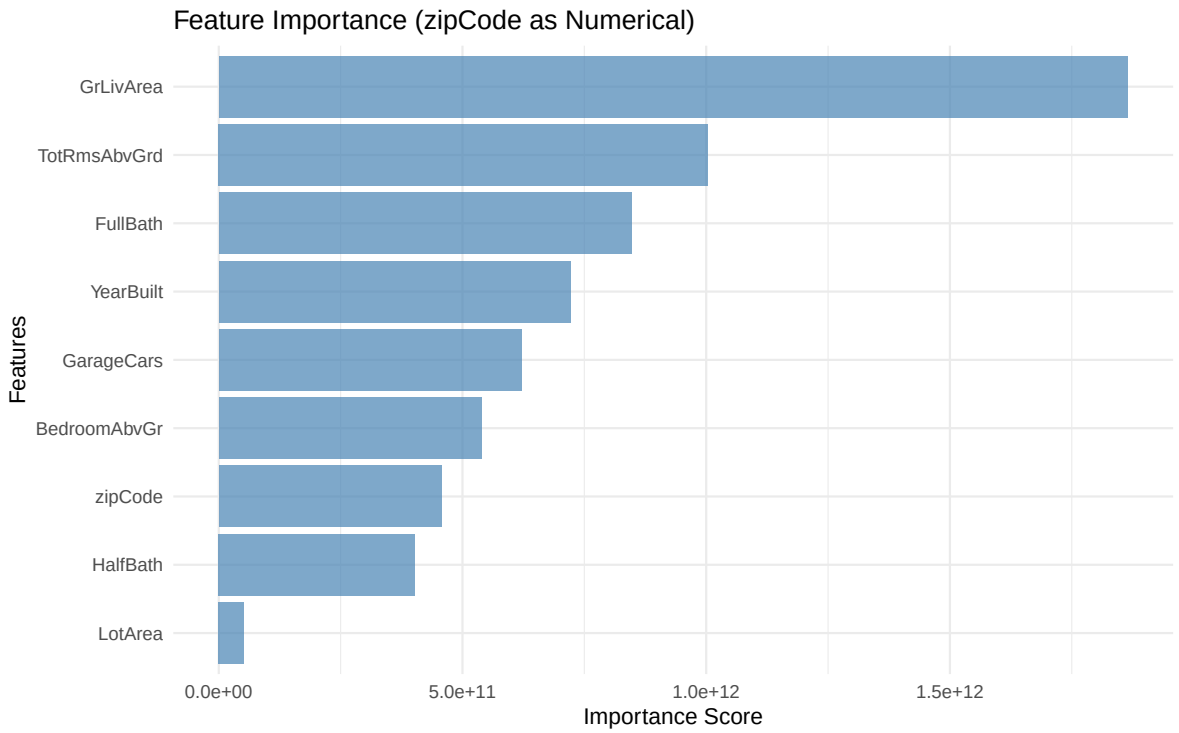
```r
cat("- zipCode ranks #", zipcode_rank, " with ", zipcode_importance, "% importance\n", sep =
```

- zipCode ranks #7 with 7.02% importance

```r
cat("- This is problematic: zipCode is treated as numerical but represents categorical areas
```

- This is problematic: zipCode is treated as numerical but represents categorical areas

```r
# Plot feature importance
library(ggplot2)
ggplot(importance_df, aes(x = reorder(Feature, Importance), y = Importance)) +
  geom_col(fill = "steelblue", alpha = 0.7) +
  coord_flip() +
  labs(title = "Feature Importance (zipCode as Numerical)",
       x = "Features", y = "Importance Score") +
  theme_minimal()
```



Python

```
# Extract and display feature importance
importance_df = pd.DataFrame({
    'Feature': X_train.columns,
    'Importance': tree_model.feature_importances_
}).sort_values('Importance', ascending=False)

importance_df['Importance_Percent'] = (importance_df['Importance'] * 100).round(2)

print("Feature Importance Rankings:")
```

```
Feature Importance Rankings:
```

```
print(importance_df)
```

```
        Feature  Importance  Importance_Percent
2     GrLivArea    0.519472               51.95
```

```
7    GarageCars    0.260808              26.08
8       zipCode    0.133463              13.35
6  TotRmsAbvGrd    0.067144               6.71
1     YearBuilt    0.019114               1.91
0       LotArea    0.000000               0.00
3      FullBath    0.000000               0.00
4      HalfBath    0.000000               0.00
5  BedroomAbvGr    0.000000               0.00
```

```
# Check zipCode ranking
zipcode_rank = importance_df[importance_df['Feature'] == 'zipCode'].index[0] + 1
zipcode_importance = importance_df[importance_df['Feature'] == 'zipCode']['Importance_Percent

print(f"\nKey Finding:")
```

```
Key Finding:
```

```
print(f"- zipCode ranks #{zipcode_rank} with {zipcode_importance}% importance")
```

```
- zipCode ranks #9 with 13.35% importance
```

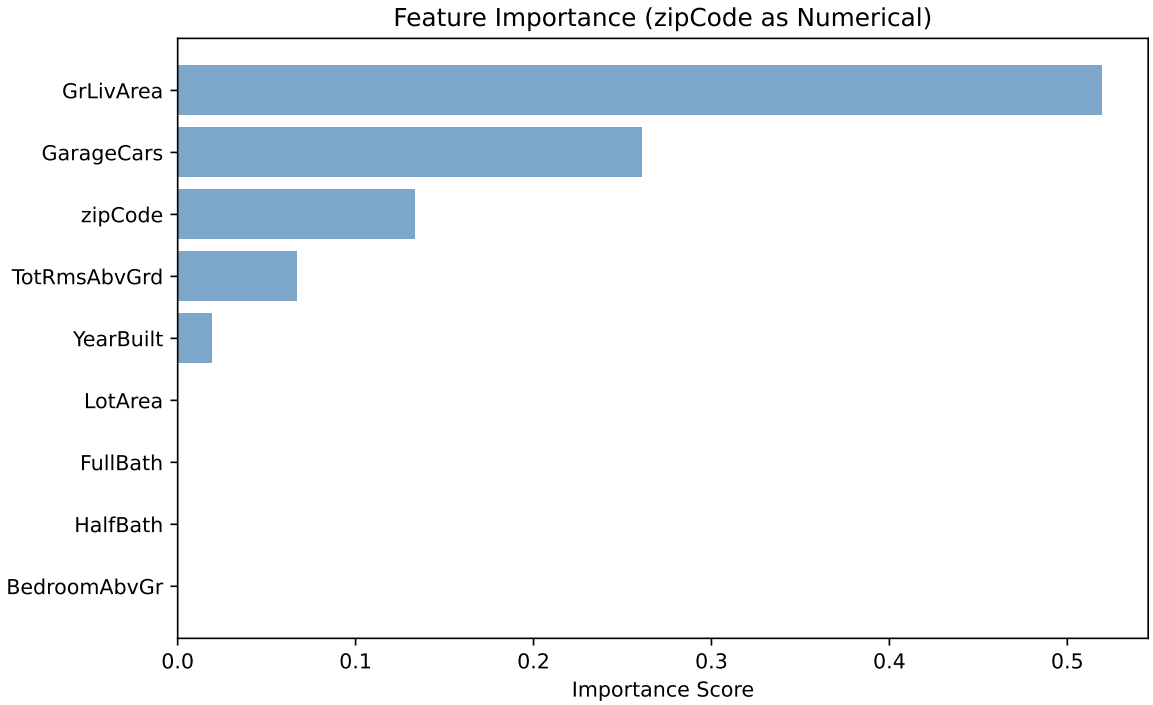
```
print("- This is problematic: zipCode is treated as numerical but represents categorical area
```

```
- This is problematic: zipCode is treated as numerical but represents categorical areas
```

```
# Plot feature importance
plt.figure(figsize=(8, 5))
plt.barh(range(len(importance_df)), importance_df['Importance'],
         color='steelblue', alpha=0.7)
plt.yticks(range(len(importance_df)), importance_df['Feature'])
```

```
([<matplotlib.axis.YTick object at 0x0000021E4A773CE0>, <matplotlib.axis.YTick object at 0x0
```

```
plt.xlabel('Importance Score')
plt.title('Feature Importance (zipCode as Numerical)')
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```



Critical Analysis: The Encoding Problem

⚠ The Problem Revealed

What we observed: Our decision tree treated `zipCode` as a numerical variable, allowing splits like "zipCode $>$ 50012.5". This creates several problems:

1. **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
3. **Misleading Interpretations:** We might conclude zipCode is important when it's really just an artifact of poor encoding

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.

Model Performance

R

```r
# Calculate performance metrics
train_pred <- predict(tree_model, train_data)
test_pred <- predict(tree_model, test_data)

train_rmse <- sqrt(mean((train_data$SalePrice - train_pred)^2))
test_rmse <- sqrt(mean((test_data$SalePrice - test_pred)^2))

train_r2 <- 1 - sum((train_data$SalePrice - train_pred)^2) /
            sum((train_data$SalePrice - mean(train_data$SalePrice))^2)
test_r2 <- 1 - sum((test_data$SalePrice - test_pred)^2) /
           sum((test_data$SalePrice - mean(test_data$SalePrice))^2)

cat("Model Performance:\n")
```

```
Model Performance:
```

```r
cat("Training RMSE: $", round(train_rmse, 2), " | R²: ", round(train_r2, 3), "\n", sep = "")
```

```
Training RMSE: $40132.24 | R²: 0.64
```

```r
cat("Testing RMSE: $", round(test_rmse, 2), " | R²: ", round(test_r2, 3), "\n", sep = "")
```

```
Testing RMSE: $45940.5 | R²: 0.668
```

Python

```python
# Calculate performance metrics
train_pred = tree_model.predict(X_train)
test_pred = tree_model.predict(X_test)

train_rmse = np.sqrt(mean_squared_error(y_train, train_pred))
test_rmse = np.sqrt(mean_squared_error(y_test, test_pred))

train_r2 = r2_score(y_train, train_pred)
```

```
test_r2 = r2_score(y_test, test_pred)

print("Model Performance:")
```

```
Model Performance:
```

```
print(f"Training RMSE: ${train_rmse:,.2f} | R²: {train_r2:.3f}")
```

```
Training RMSE: $40,585.49 | R²: 0.664
```

```
print(f"Testing RMSE: ${test_rmse:,.2f} | R²: {test_r2:.3f}")
```

```
Testing RMSE: $44,463.43 | R²: 0.566
```

Conclusion

Key Takeaway: Treating zipCode as a numerical variable in decision trees leads to:

1. **Misleading splits** that have no meaningful interpretation
2. **Distorted feature importance** rankings
3. **False insights** about which variables actually matter for house price prediction

Next Steps: Proper categorical encoding would treat zipCode as discrete categories, revealing the true importance of each variable and providing interpretable splits that make business sense.

The decision tree example demonstrates why data preprocessing and proper variable encoding are crucial for interpretable machine learning models.