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. 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

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
```

```r
# 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

```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)
```

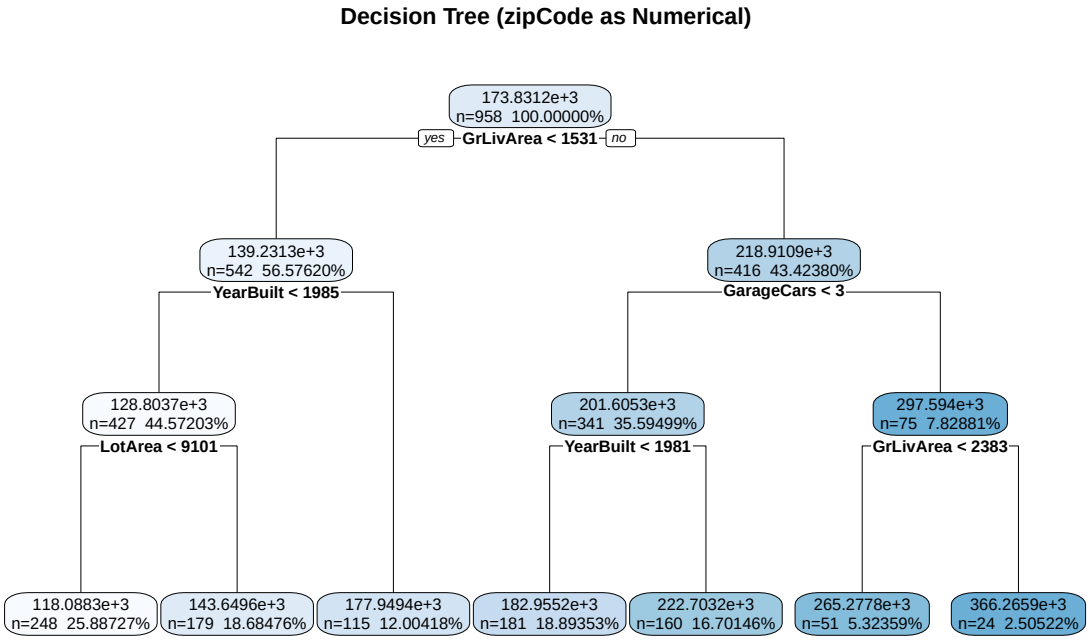
```
print(f"Model built with {tree_model.get_n_leaves()} terminal nodes")
```

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

Tree Visualization

R

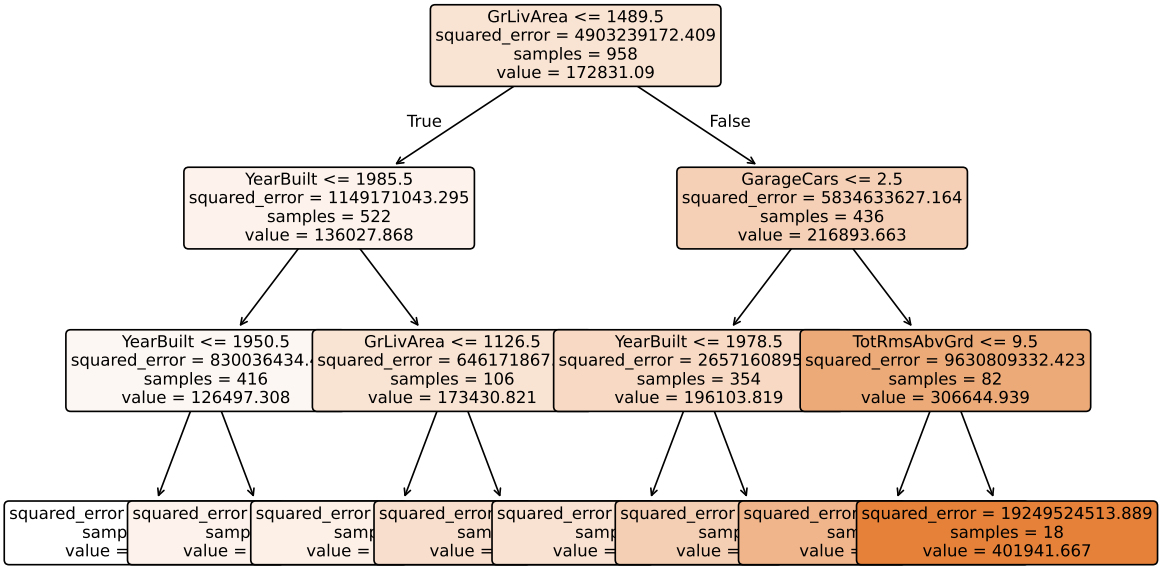
```
# Visualize tree
if (require(rpart.plot, quietly = TRUE)) {
  rpart.plot(tree_model,
             type = 2,
             extra = 101,
             fallen.leaves = TRUE,
             digits = 0,
             cex = 0.8,
             main = "Decision Tree (zipCode as Numerical)")
} else {
  plot(tree_model,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



Python

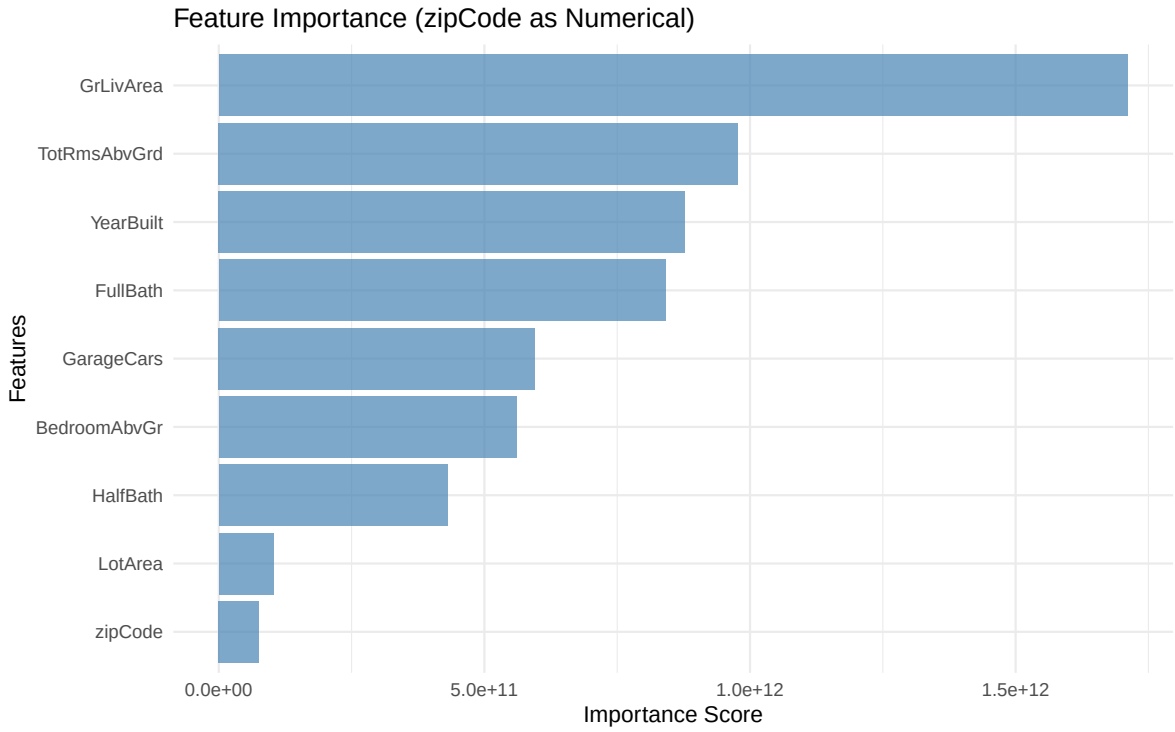
```
# Visualize tree
plt.figure(figsize=(10, 6))
plot_tree(tree_model,
          feature_names=X_train.columns,
          filled=True,
          rounded=True,
          fontsize=10,
          max_depth=3)
plt.title("Decision Tree (zipCode as Numerical)")
plt.tight_layout()
plt.show()
```

Decision Tree (zipCode as Numerical)



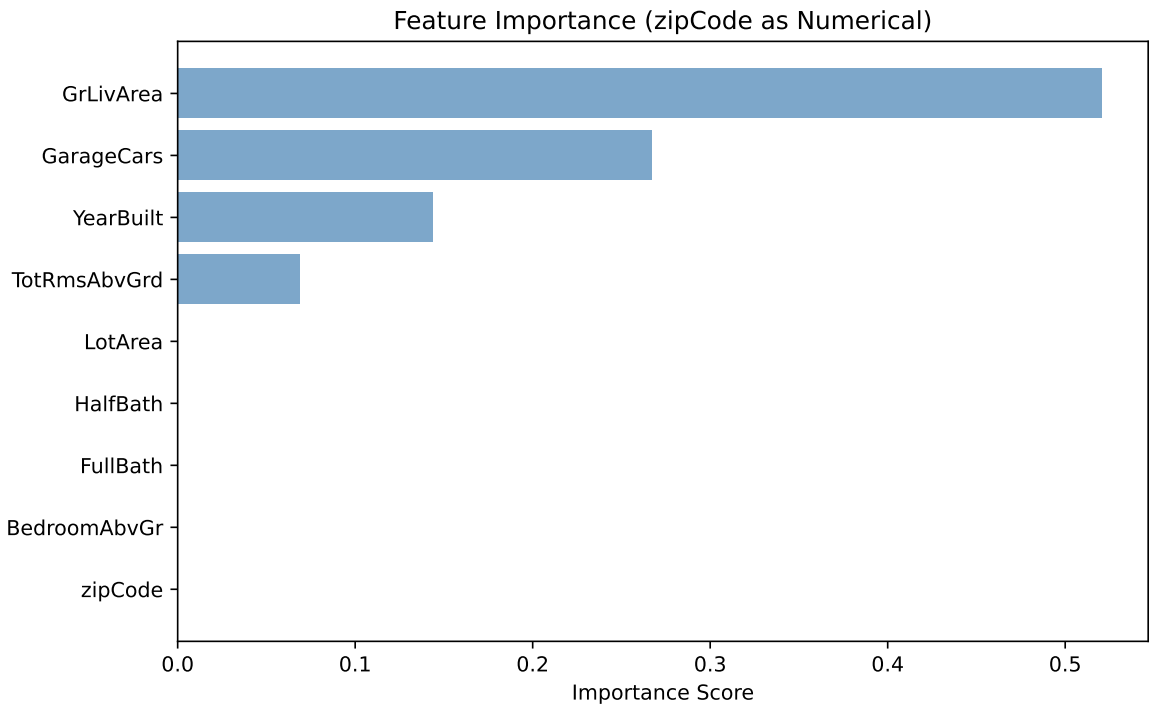
Feature Importance Analysis

R



Python

```
([<matplotlib.axis.YTick object at 0x0000018783F565A0>, <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 nor 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

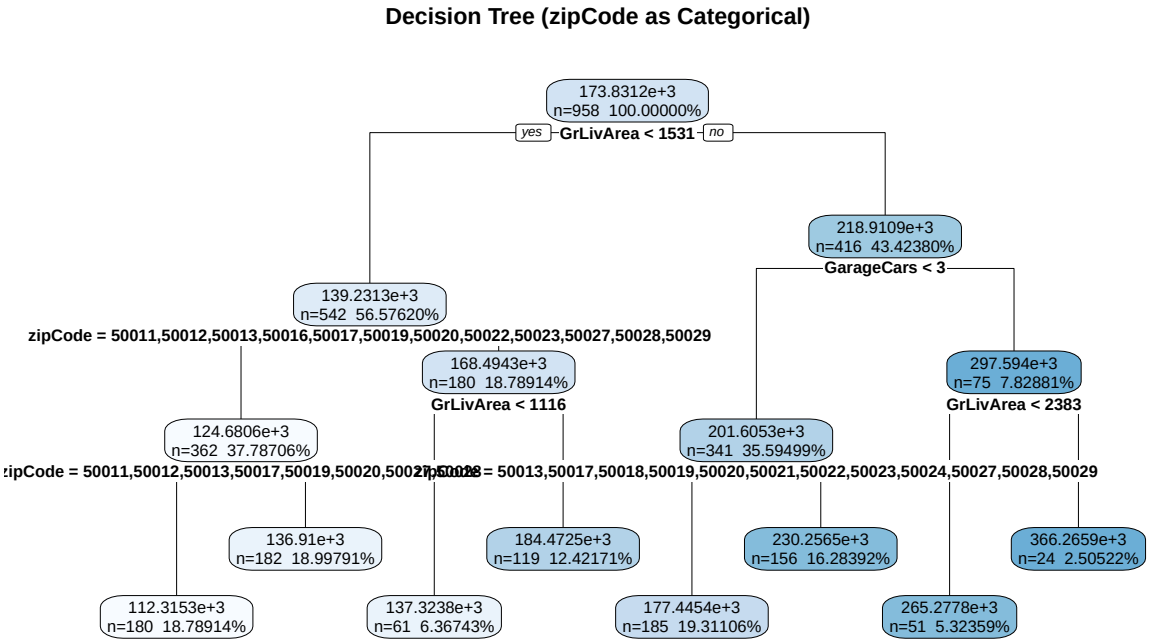
R

Python

Tree Visualization: Categorical zipCode

R

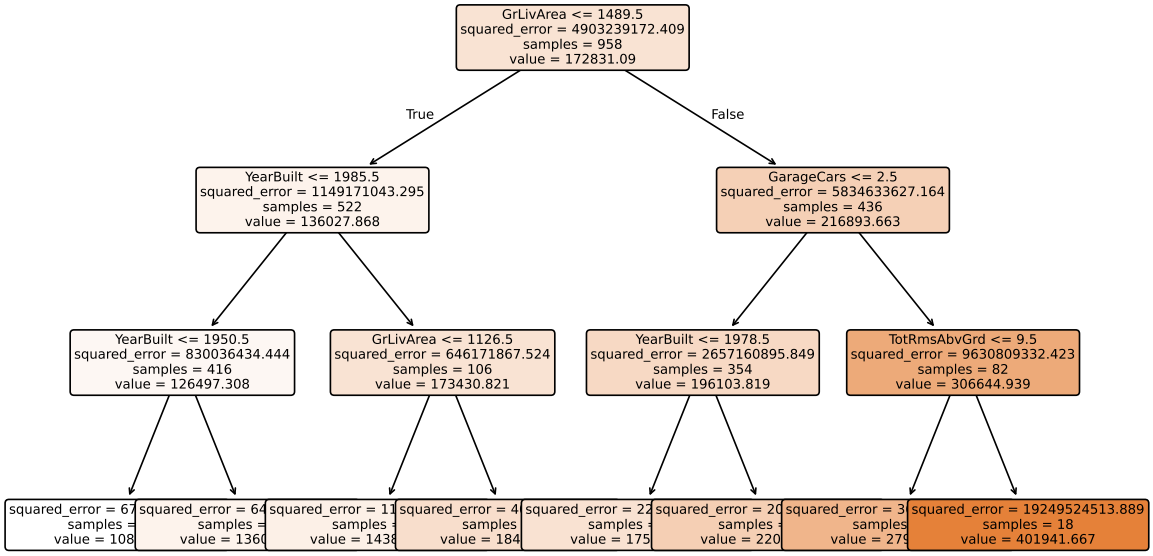
```
# Visualize tree with categorical zipCode
if (require(rpart.plot, quietly = TRUE)) {
  rpart.plot(tree_model_cat,
             type = 2,
             extra = 101,
             fallen.leaves = TRUE,
             digits = 0,
             cex = 0.8,
             main = "Decision Tree (zipCode as Categorical)")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

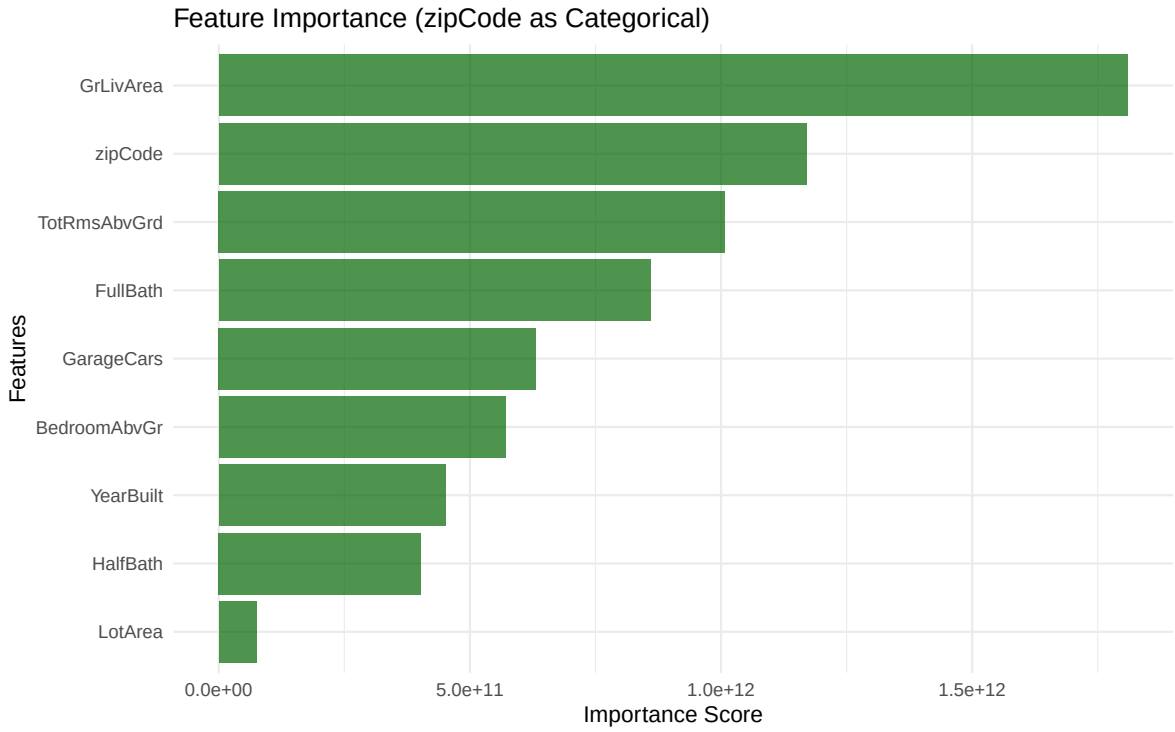
```
# Visualize tree with one-hot encoded zipCode
plt.figure(figsize=(10, 6))
plot_tree(tree_model_cat,
          feature_names=X_train_cat.columns,
          filled=True,
          rounded=True,
          fontsize=8,
          max_depth=4)
plt.title("Decision Tree (zipCode One-Hot Encoded)")
plt.tight_layout()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

Decision Tree (zipCode One-Hot Encoded)



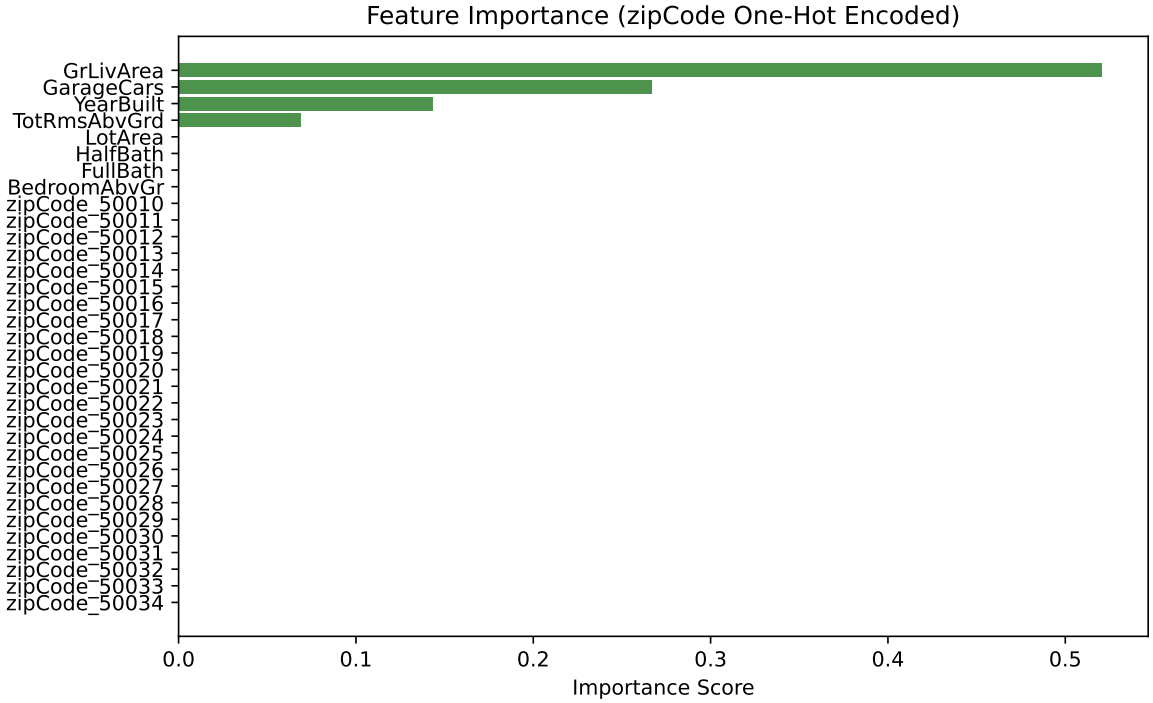
Feature Importance: Categorical zipCode

R



Python

```
([<matplotlib.axis.YTick object at 0x0000018784148650>, <matplotlib.axis.YTick object at 0x0
```


Feature Importance (zipCode One-Hot Encoded)
GrLivArea
GarageCars
YearBuilt
TotRmsAbvGrd
LotArea
HalfBath
FullBath
BedroomAbvGr
zipCode_50010
zipCode_50011
zipCode_50012
zipCode_50013
zipCode_50014
zipCode_50015
zipCode_50016
zipCode_50017
zipCode_50018
zipCode_50019
zipCode_50020
zipCode_50021
zipCode_50022
zipCode_50023
zipCode_50024
zipCode_50025
zipCode_50026
zipCode_50027
zipCode_50028
zipCode_50029
zipCode_50030
zipCode_50031
zipCode_50032
zipCode_50033
zipCode_50034
0.0
0.1
0.2
0.3
0.4
0.5
Importance Score