# Final Project for pstat131

Haocheng Zhang

## Contents

## Introduction

### Part 1: Overview and Context

The real estate market is a dynamic ecosystem where many different elements interact to affect how much a property costs. And understanding these factors can not only assist those who are looking to buy or sell in making informed decisions, but it can also offer valuable information to investors, politicians, and urban planners. The project seeks to investigate and predict the price of real estate in California, a state famous for its diverse housing markets, which range from some of the most low-cost to the most pricey in the United States.

### Part 2: Dataset Description

The dataset used in this project was sourced from Kaggle (https://www.kaggle.com/datasets/yellowj4acket/real-estate-california), specifically the "Real Estate California" dataset. This dataset contains listings for the first six months of 2021 and includes a variety of features:

Observations (rows): 35,389 Predictors (columns): 39

Types of variables included:

Numerical Variables: Price, living area, number of bathrooms, etc. Categorical Variables: City, home type Binary Variables: Presence of a garage, new construction status

We will particularly focus on 'price,' 'livingArea,' 'zipcode, 'bedrooms,' 'bathrooms,' 'homeType,' and 'isNewConstruction' in our analyses."

**Sub-part 1: Strategy of Handling Missing Values** It is worth noting that the dataset does contain some missing values. For instance, ‘zipcode’ has 25 missing values, and these rows with missing zipcode values should be removed since the zipcode is crucial for geographical analysis and is significant in predicting property prices. Rows where the living area is 0 are removed. A zero in this field is likely to be an error, as it is implausible for a property to have zero living area, similarly we remove observations of properties with a price of zero, and also rows where both bedrooms and bathrooms are zero. These steps can be found in Step 3 of section 1.

### Part 3: Research Questions

The primary research question is: Can we predict property prices in California based on factors/variables such as living area, number of bedrooms and bathrooms, and location (area and property type)?

Secondary questions include: How do various features affect property prices? Are there location-based trends in property pricing?

### Part 4: Methodology Overview

Given that the primary objective is to predict property prices, a regression approach will be adopted for this project. Various machine learning techniques (introduced in Pstat 131 course such as linear regression, K-NN, and etc.) will be evaluated to find the most accurate predictive model. The model’s performance will be assessed using k-fold cross-validation to ensure its robustness and reliability. However, because of the size of the dataset (more than 35k observations), to reduce the time on running/training models, there will be a smaller dataset (with around 4k observations). In this way, we have both full EDA on the entire dataset, which gives a better understanding of the data’s characteristics, outliers, and potential issues, and also the models will train much faster on a smaller dataset to save time.

### Part 5: Goal of the Model

The overarching goal of this project is predictive. We aim to build a model that can accurately forecast property prices in California based on a set of predictor variables. While the focus is on prediction, the model can also offer descriptive insights into the factors most significantly affecting property prices.

### Part 6: Data Citation

This dataset is a co-production, with special thanks to the contributor and their colleague Jordan for compiling this comprehensive dataset. Using this dataset requires citing the contributor as per the terms listed on the Kaggle page.

## Section 1: Data Preparation.

### Step 1: Data Loading.

```r
data <- read.csv("RealEstate_California.csv")
```

### Step 2: Data Cleaning. Keep only the columns that are necessary for the analysis

```r
data_cleaned <- data %>%
  select(price, zipcode, livingArea, bathrooms, bedrooms,
  garageSpaces, isNewConstruction, homeType, yearBuilt)
```

**Step 3: Handle Missing Values and Compute New Useful Variables.**

**Sub-step 1: Remove invalid observations.** Rows with missing zipcode values should be removed since the zipcode is crucial for geographical analysis and is significant in predicting property prices. Rows where the living area is 0 are removed. A zero in this field is likely to be an error, as it is implausible for a property to have zero living area, similarly we remove observations of properties with a price of zero, and also rows where both bedrooms and bathrooms are zero.

```r
# Remove rows where 'zipcode' is NA, 'livingArea' is 0, 'yearBuilt' is 0, 'price' is
# 0, and both 'bedrooms' and 'bathrooms' are 0, per the plan.
data_cleaned <- data_cleaned %>%
  filter(!is.na(zipcode) &
         livingArea != 0  &
         yearBuilt != 0 &
         price != 0 &
         !(bedrooms == 0 & bathrooms == 0))
```

```r
# Handle the case where bathrooms is zero to avoid division by zero
data_cleaned$bed_bath_ratio <- ifelse(data_cleaned$bathrooms == 0, 0,
                      data_cleaned$bedrooms / data_cleaned$bathrooms)
```

**Sub-step 2: Create a new feature that is the ratio of bedrooms to bathrooms**

```r
data_cleaned$price_per_area <- data_cleaned$price / data_cleaned$livingArea
```

**Sub-step 3: Create a new variable 'price_per_area'**

**Step 4: Convert Data Types.**

```r
# Convert 'yearBuilt' and 'isNewConstruction' to integer type.
data_cleaned$yearBuilt <- as.integer(data_cleaned$yearBuilt)
data_cleaned$isNewConstruction <- as.integer(data_cleaned$isNewConstruction)

# Ensure 'price', 'livingArea', 'bathrooms', 'bedrooms', 'garageSpaces', 'bed_bath_ratioare', and 'pric
data_cleaned$price <- as.numeric(data_cleaned$price)
data_cleaned$livingArea <- as.numeric(data_cleaned$livingArea)
data_cleaned$bathrooms <- as.numeric(data_cleaned$bathrooms)
data_cleaned$bedrooms <- as.numeric(data_cleaned$bedrooms)
data_cleaned$garageSpaces <- as.numeric(data_cleaned$garageSpaces)
data_cleaned$bed_bath_ratio <- as.numeric(data_cleaned$bed_bath_ratio)
```

```r
data_cleaned$price_per_area <- as.numeric(data_cleaned$price_per_area)
data_cleaned$homeType <- as.numeric(factor(data_cleaned$homeType))
data_cleaned$zipcode <- as.numeric(factor(data_cleaned$zipcode))
```

**Step 5: Print a summary of the cleaned data.**

```r
summary(data_cleaned)
```

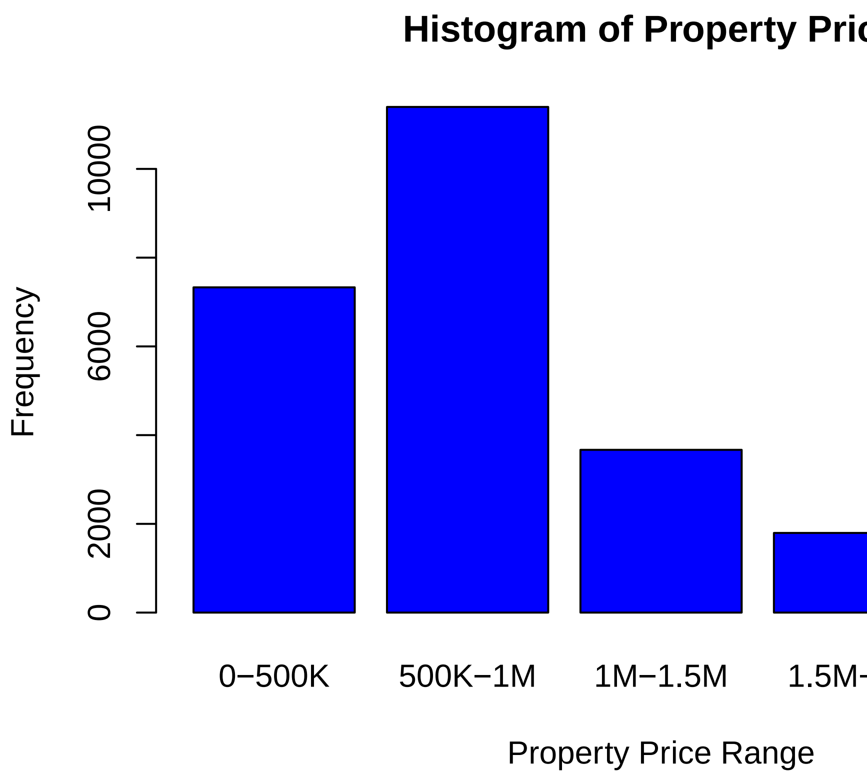
```
##      price              zipcode         livingArea      bathrooms
##  Min.   :      650   Min.   :   1.0   Min.   :    1   Min.   : 0.000
##  1st Qu.:   490000   1st Qu.: 306.0   1st Qu.: 1260   1st Qu.: 2.000
##  Median :   749000   Median : 607.0   Median : 1720   Median : 2.000
##  Mean   : 1202667   Mean   : 668.6   Mean   : 2075   Mean   : 2.581
##  3rd Qu.: 1200000   3rd Qu.:1005.0   3rd Qu.: 2414   3rd Qu.: 3.000
##  Max.   :90000000   Max.   :1500.0   Max.   :55284   Max.   :48.000
##     bedrooms       garageSpaces    isNewConstruction    homeType
##  Min.   : 0.000   Min.   : 0.000   Min.   :0.00000   Min.   :1.000
##  1st Qu.: 3.000   1st Qu.: 0.000   1st Qu.:0.00000   1st Qu.:4.000
##  Median : 3.000   Median : 1.000   Median :0.00000   Median :4.000
##  Mean   : 3.344   Mean   : 1.206   Mean   :0.01469   Mean   :3.632
##  3rd Qu.: 4.000   3rd Qu.: 2.000   3rd Qu.:0.00000   3rd Qu.:4.000
##  Max.   :82.000   Max.   :22.000   Max.   :1.00000   Max.   :5.000
##    yearBuilt    bed_bath_ratio   price_per_area
##  Min.   :1850   Min.   : 0.000   Min.   :      0.3
##  1st Qu.:1955   1st Qu.: 1.000   1st Qu.:    298.6
##  Median :1976   Median : 1.333   Median :    450.6
##  Mean   :1973   Mean   : 1.373   Mean   :    657.3
##  3rd Qu.:1995   3rd Qu.: 1.500   3rd Qu.:    657.4
##  Max.   :2022   Max.   :12.000   Max.   :1775000.0
```

## Section 2: Exploratory Data Analysis

**Step 1: Univariate Analysis**

```r
# Create a table of price categories directly
price_count <- table(cut(data_cleaned$price,
                         breaks = c(0, 500000, 1000000, 1500000, 2000000, Inf),
                         labels = c("0-500K", "500K-1M", "1M-1.5M", "1.5M-2M", "2M+")))

# Create a bar plot based on the price categories
barplot(price_count,
        main = "Histogram of Property Prices",
        xlab = "Property Price Range",
        ylab = "Frequency",
        col = "blue",
        border = "black",
        legend.text = paste(names(price_count), "\n", round(price_count/sum(price_count) * 100, 2), "%"
```



**Sub-step 1: Histogram of Property Prices**
From the bar chart above, we can make the follwoing conslusions: The largest portion of properties falls within the "500K-1M" range, making up approximately 42.17% of the total listings.

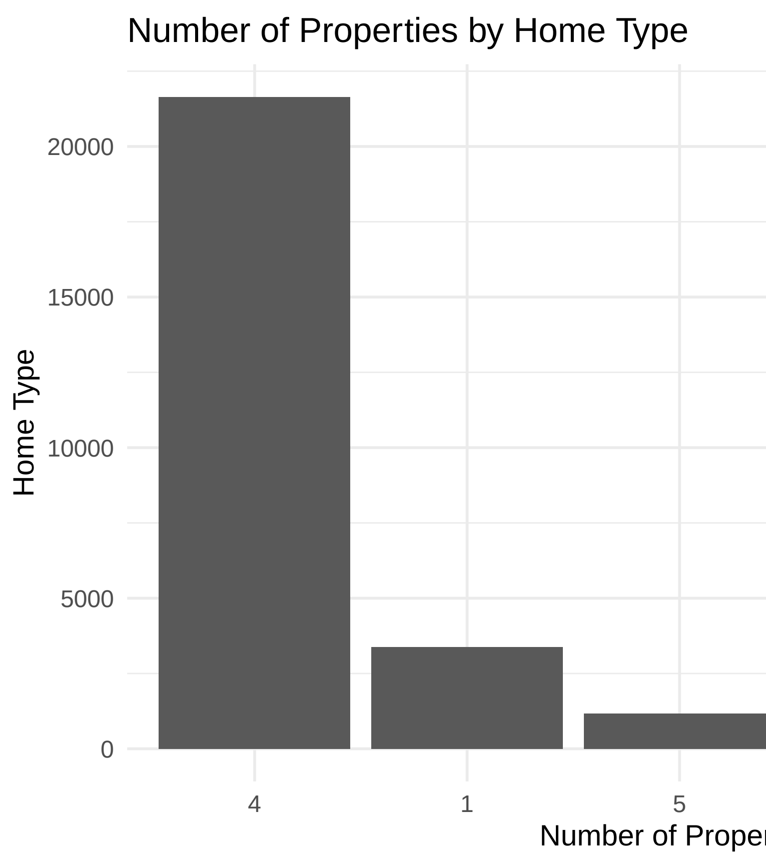
The next largest category is the "0-500K" range, which accounts for about 27.12%.

Properties priced between "1M-1.5M" represent around 13.57%.

The "1.5M-2M" and "2M+" categories make up smaller portions of the data, approximately 6.64% and 10.49% respectively.

This distribution highlights the concentration of property prices within the middle ranges, with fewer properties at the extreme low or high ends of the price spectrum, which does make sense because it follows the common sense in reality.

```r
data_cleaned %>%
  count(homeType) %>%
  ggplot(aes(x = reorder(homeType, -n), y = n)) +
  geom_bar(stat = "identity") +
  labs(title = "Number of Properties by Home Type",
       y = "Home Type",
       x = "Number of Properties") +
  theme_minimal()
```



**Sub-step 2: Number of Properties by Home Type**
The most common home type appears to be "Single Family," followed by "Condo" and "Townhouse" This visualization can provide valuable insights into the types of properties that are most commonly available in the market, which can be useful for both buyers and sellers.

```
agg_by_zipcode <- data_cleaned %>%
  group_by(zipcode) %>%
  summarise(avg_price = mean(price, na.rm = TRUE),
            median_price = median(price, na.rm = TRUE))

# Sort the dataset by the average price
agg_by_zipcode_sorted <- agg_by_zipcode %>%
  arrange(desc(avg_price))

# View the top rows of the sorted dataset
head(agg_by_zipcode_sorted)
```

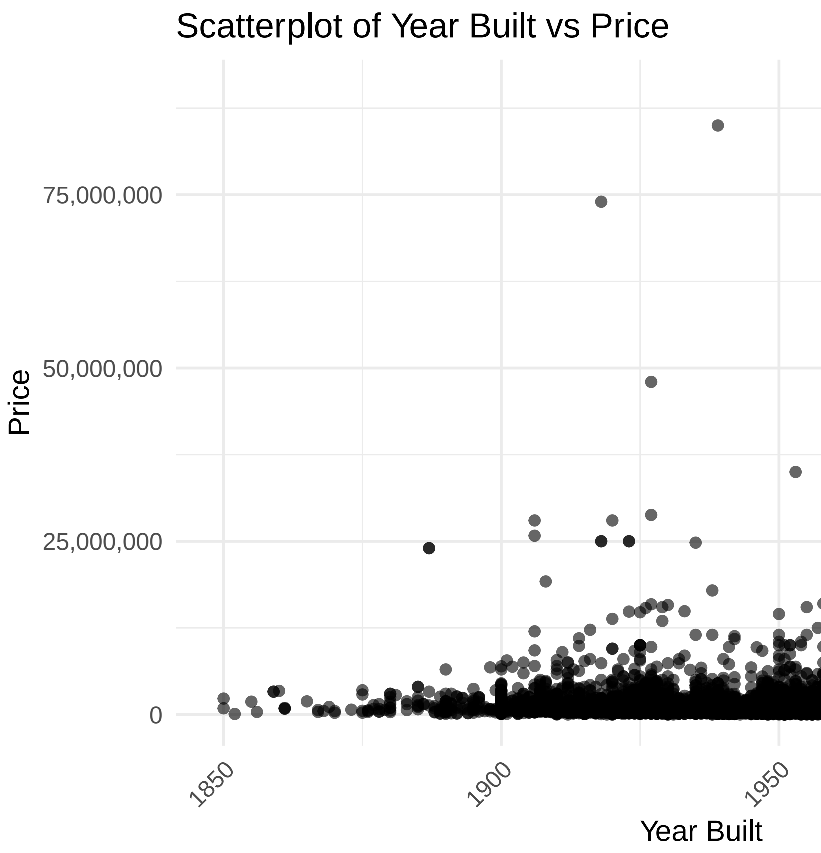
**Sub-step 3: Aggregate Prices by Zipcode**

```
## # A tibble: 6 x 3
##   zipcode avg_price median_price
##     <dbl>     <dbl>        <dbl>
## 1     883 53888000      53888000
## 2     612 19858800       9200000
## 3      60 17128875.      6669000
## 4      835 14671538.      12500000
## 5     1011 12149000        3895000
## 6     1167 11917000      11917000
```

Zip code 94304 has the highest average price of approximately \$53,888,000 and a median price of \$53,888,000, which is incredibly expresnsive. After doing some research, the zip code belongs to Palo Alto, which is a beautiful city, located in Santa Clara county. And the reason why it is being so expensive is that developers and business men in Silicon Valley live there.

**Step 2: Bivariate Analysis**

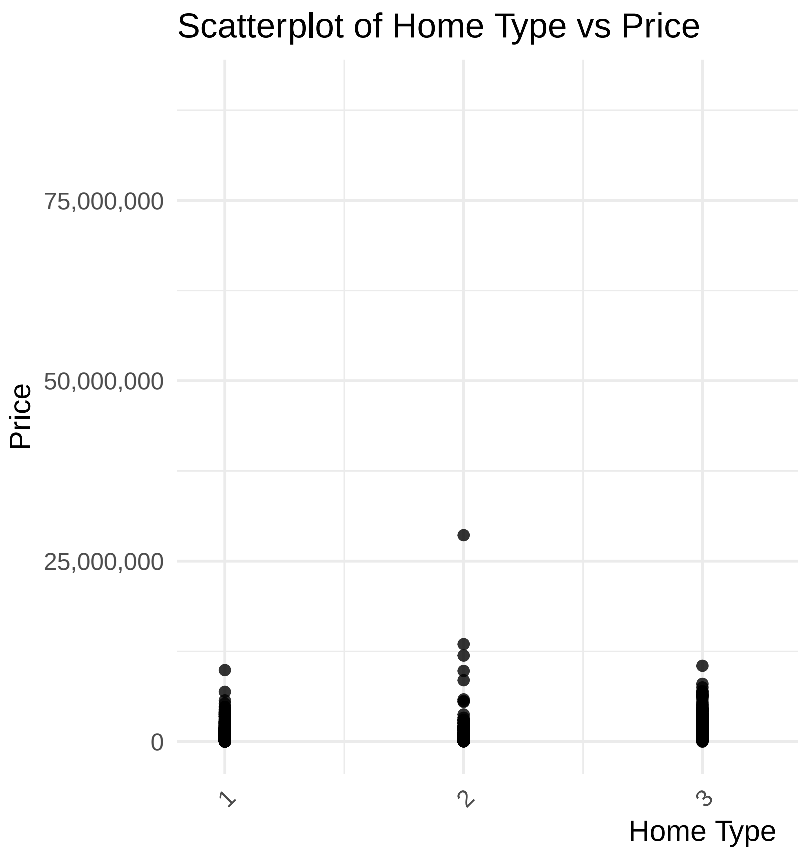
```r
ggplot(data_cleaned, aes(x = yearBuilt, y = price)) +
  geom_point(alpha = 0.6) +
  labs(title = "Scatterplot of Year Built vs Price",
       x = "Year Built",
       y = "Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_y_continuous(labels = scales::comma)
```



**Sub-step 1: Scatterplot of YearBuilt vs Price**
By observing the plot above, there seems to be a general upward trend in property prices for more recently built properties, especially as we approach the year 2010s. This trend could be indicative of a variety of factors such as inflation, increasing land costs, or simply that newer homes might have more modern

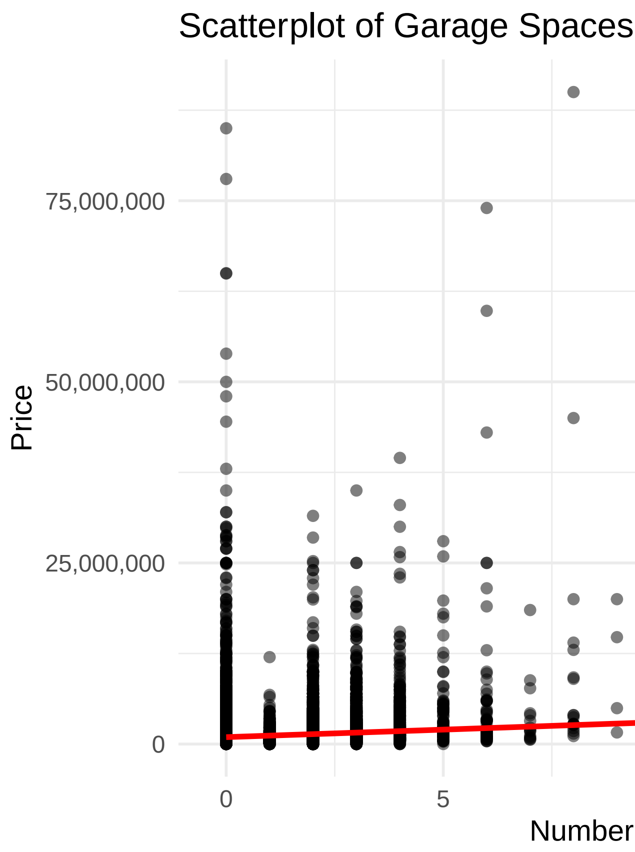
amenities that could make them more expensive. Also, there are some outliers with extremely high prices in around 1940s.

```r
ggplot(data_cleaned, aes(x = homeType, y = price)) +
  geom_point(alpha = 0.8) +
  labs(title = "Scatterplot of Home Type vs Price",
       x = "Home Type",
       y = "Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_y_continuous(labels = scales::comma)
```



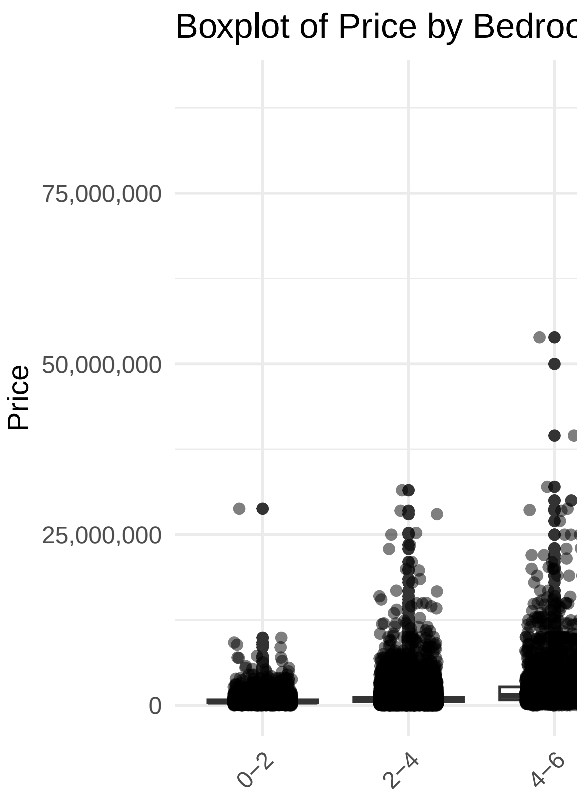
**Sub-step 2: Scatterplot of Home Type vs Price**
The scatterplot above illustrates the relationship between the type of home and its corresponding price. It is clear that different types of homes have different price ranges. While certain categories like 'Single Family' and 'Lot' appear to have a broad range of prices, other categories like 'MultiFamily' and 'Townhouse' seem to cluster around specific and lower price points.

```r
ggplot(data_cleaned, aes(x = garageSpaces, y = price)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = 'lm', se = FALSE, color = 'red') +
  labs(title = "Scatterplot of Garage Spaces vs Price",
       x = "Number of Garage Spaces",
       y = "Price") +
  theme_minimal() +
  scale_y_continuous(labels = scales::comma)
```



**Sub-step 3: Scatterplot of Garage Spaces vs Property Price**
By observing the scatterplot of garage spaces versus property price and linear regression line (in red), which is to indicate the trend. It seems that as the number of garage spaces increases, the property price also tends to rise, although the relationship doesn't appear to be very strong. Which does make sense because more garage spaces usually means bigger space/area of the property, which lead to higher price.

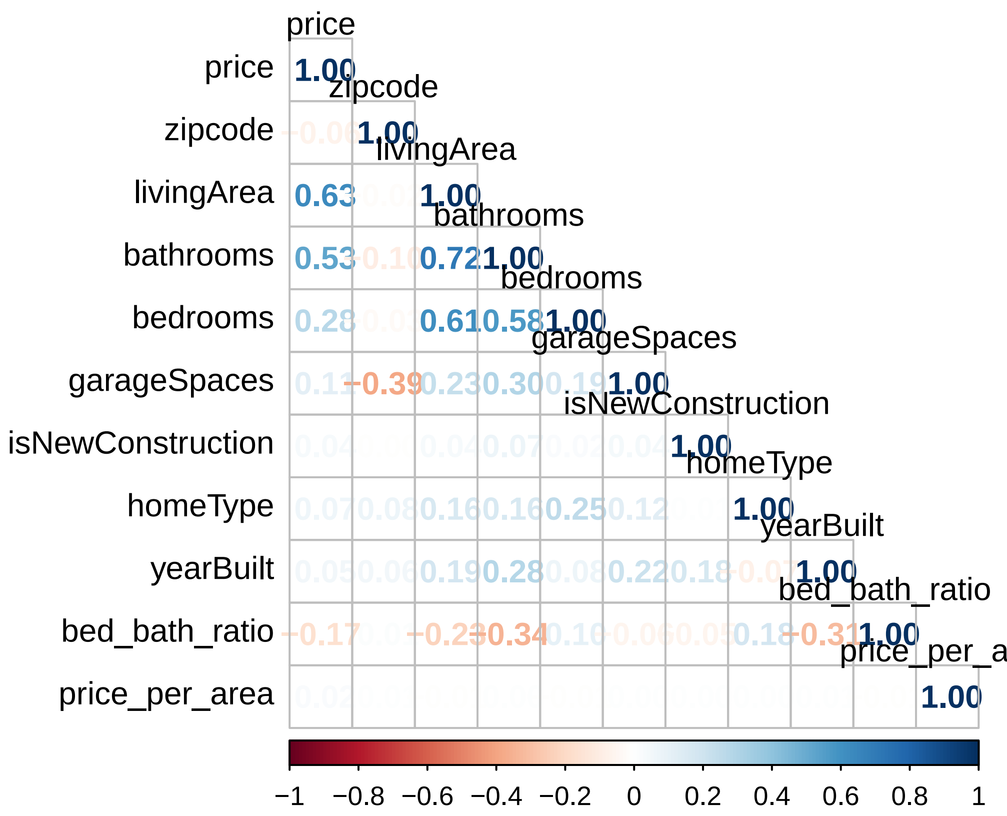
```r
ggplot(data = data_cleaned %>% filter(!is.na(bedrooms)),
       aes(x = cut(bedrooms, breaks = c(0, 2, 4, 6, 8, 10, 12, Inf),
                   include.lowest = TRUE, labels = c("0-2", "2-4", "4-6", "6-8",
                                                     "8-10", "10-12", "12+")),
           y = price)) +
  geom_boxplot() +
  geom_jitter(alpha = 0.5, width = 0.2) +
  labs(title = "Boxplot of Price by Bedroom Category",
       x = "Bedroom Category",
       y = "Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  scale_y_continuous(labels = scales::comma)
```



**Sub-step 4: Create a box plot of Price by categorized Bedrooms**
After observing the boxplot above, it is evident that properties with more bedrooms tend to have higher prices, although there is a significant amount of variation within each category. Specifically, the median price appears to increase with the number of bedrooms, especially when going from "0-2" to "2-4" bedrooms.

```r
# Define the correlation matrix of data_cleaned
cor_matrix <- cor(select_if(data_cleaned, is.numeric))

# Create the correplot
corrplot(cor_matrix, method = "number", type = "lower", tl.col = "black", tl.srt = 0)
```



**Sub-step 5: Correlation Matrix**
From the correlation matrix above, ‘price’ and ‘livingArea’ have a positive correlation of 0.63, suggesting that as the living area increases, the price generally tends to increase as well.

‘garageSpaces’ and ‘bedrooms’ have a positive correlation of 0.3, indicating that properties with more bedrooms are probably to have more garage spaces.

## Section 3: Data Splitting and Cross-Validation.

### Step 1: Setting seed. And make a smaller dataset.

Also, I would remove ‘price_per_area’ column here, because it is calculated by price/livingArea, which is strongly correlated with price.

```r
# Set the seed to ensure reproducibility.
set.seed(123)

# Create a smaller dataset containing 4000 observations.
small_data <- sample_n(data_cleaned, size = 4000) %>% select(-price_per_area, -bed_bath_ratio)
```

### Step 2: Perform Stratified Sampling to Split the Data.

```r
# 70% of the data will go into the training set, and 30% will go into the test set.
# The split is stratified by price.
data_split <- initial_split(small_data, prop = 0.7, strata = "price")
```

```
train_data <- training(data_split)
test_data <- testing(data_split)

#Check the First Few Rows of Both Sets:
head(train_data)
```

```
##    price zipcode livingArea bathrooms bedrooms garageSpaces isNewConstruction
## 1 349000     527       1040         2        2            0                 0
## 2 325000     326        959         2        2            1                 0
## 3 399000     356        676         1        1            1                 0
## 4 320000    1141        870         1        2            0                 0
## 5 370000    1443       2000         3        4            0                 0
## 6 435000    1367       1158         2        3            0                 0
##   homeType yearBuilt
## 1        1      1965
## 2        1      1981
## 3        1      1970
## 4        4      1950
## 5        4      1978
## 6        4      1993
```

```
head(test_data)
```

```
##     price zipcode livingArea bathrooms bedrooms garageSpaces isNewConstruction
## 1 2650000      86       2701         4        4            0                 0
## 2  505000      22       1014         1        2            0                 0
## 3  725000     389       2814         3        4            2                 0
## 4 1089000    1063       1502         2        4            0                 0
## 5  888000     276       2800         4        6            2                 0
## 6  899000    1484       3461         5        5            3                 0
##   homeType yearBuilt
## 1        4      1957
## 2        4      1919
## 3        4      2005
## 4        4      1967
## 5        4      1908
## 6        4      1997
```

**Step 3: Create k-fold cross-validation.**

```
# Create 5 Folds from the Training Set for Cross-Validation.
folds <- vfold_cv(train_data, v = 5, strata = "price")
```

Now the training set is further divided into 5 folds using stratified sampling based on the target variable. This results in 5 different sets or “folds” of data, which is helpful for model evaluation.

## Section 4: Model Building and Evaluation.

### Step 1: Create the Recipe.

```r
# Create the recipe
real_estate_recipe <- recipe(price ~ .,
                             data = train_data) %>%
  step_novel(all_nominal_predictors()) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_zv(all_predictors()) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

# Print the recipe to check it
print(real_estate_recipe)
```

### Step 2: Model Specification and Workflow

```r
# k-NN Model Specification
knn_spec <- nearest_neighbor(
  neighbors = tune(),
  engine = "kknn") %>%
  set_mode("regression")

# k-NN Workflow
knn_wf <- workflow() %>%
  add_model(knn_spec) %>%
  add_recipe(real_estate_recipe)

# Linear Regression Model Specification
lm_spec <- linear_reg() %>%
  set_engine("lm") %>%
  set_mode("regression")

# Linear Regression Workflow
lm_wf <- workflow() %>%
  add_model(lm_spec) %>%
  add_recipe(real_estate_recipe)

# Random Forest Model Specification
rf_spec <- rand_forest(
  mtry = tune(),
  trees = tune(),
  min_n = tune(),
  mode = "regression") %>%
  set_engine("ranger", importance = "impurity")

# Random Forest Workflow
rf_wf <- workflow() %>%
  add_model(rf_spec) %>%
  add_recipe(real_estate_recipe)
```

```r
# Elastic Net Model Specification
enet_spec <- linear_reg(
  penalty = tune(),
  mixture = tune()) %>%
  set_engine("glmnet") %>%
  set_mode("regression")

# Elastic Net Workflow
enet_wf <- workflow() %>%
  add_model(enet_spec) %>%
  add_recipe(real_estate_recipe)
```

**Setp 3: Grid Search Setup. Set up the grid of hyperparameters to be tuned for each model.**

```r
# k-NN Grid: Trying neighbors from 1 to 10, in 6 levels
knn_grid <- grid_regular(
  neighbors(range = c(1, 10)),
  levels = 6
)

# Random Forest Grid
rf_grid <- grid_regular(
  mtry(range = c(1, 7)),
  trees(range = c(50, 500)),
  min_n(range = c(2, 20)),
  levels = 8
)

# Elastic Net Grid
enet_grid <- grid_regular(
  penalty(),
  mixture(range = c(0, 1)),
  levels = 10
)

# Check all grids
print(knn_grid)
```

```
## # A tibble: 6 x 1
##   neighbors
##       <int>
## 1         1
## 2         2
## 3         4
## 4         6
## 5         8
## 6        10
```

```r
print(rf_grid)
```

```
## # A tibble: 448 x 3
##     mtry trees min_n
##    <int> <int> <int>
##  1     1    50     2
##  2     2    50     2
##  3     3    50     2
##  4     4    50     2
##  5     5    50     2
##  6     6    50     2
##  7     7    50     2
##  8     1   114     2
##  9     2   114     2
## 10     3   114     2
## # i 438 more rows
```

```
print(enet_grid)
```

```
## # A tibble: 100 x 2
##          penalty mixture
##            <dbl>   <dbl>
##  1 0.0000000001        0
##  2 0.00000000129       0
##  3 0.0000000167        0
##  4 0.000000215         0
##  5 0.00000278          0
##  6 0.0000359           0
##  7 0.000464            0
##  8 0.00599             0
##  9 0.0774              0
## 10 1                   0
## # i 90 more rows
```

#### Setp 4: Fit All Initial Models Using Cross-Validation

```
# Fit k-NN Model
knn_results <- tune_grid(
  object = knn_wf,
  resamples = folds,
  grid = knn_grid
)

# Check notes/warnings
show_notes(knn_results)

# Save the knn_results object to a .rds file
saveRDS(knn_results, "knn_results.rds")
```

```
# Fit Linear Regression Model
lm_results <- tune_grid(
  object = lm_wf,
  resamples = folds
```

```r
)

# Check notes/warnings
show_notes(lm_results)

# Save the lm_results object to a .rds file
saveRDS(lm_results, "lm_results.rds")
```

```r
# Fit Random Forest Model
rf_results <- tune_grid(
  object = rf_wf,
  resamples = folds,
  grid = rf_grid
)

# Check notes/warnings
show_notes(rf_results)

# Save the rf_results object to a .rds file
saveRDS(rf_results, "rf_results.rds")
```

```r
# Fit Elastic Net Model
enet_results <- tune_grid(
  object = enet_wf,
  resamples = folds,
  grid = enet_grid
)

# Check notes/warnings
show_notes(enet_results)

# Save the enet_results object to a .rds file
saveRDS(enet_results, "enet_results.rds")
```

```r


# Load the knn_results object from the .rds file
knn_results <- readRDS("knn_results.rds")

# Load the lm_results object from the .rds file
lm_results <- readRDS("lm_results.rds")

# Load the rf_results object from the .rds file
rf_results <- readRDS("rf_results.rds")

# Load the enet_results object from the .rds file
enet_results <- readRDS("enet_results.rds")
```

**Sub-step 1: Load Pre-trained Model.**

**Step 5: Collect Metrics for all results.**

```r
# Collect Metrics
knn_metrics <- collect_metrics(knn_results)
lm_metrics <- collect_metrics(lm_results)
rf_metrics <- collect_metrics(rf_results)
enet_metrics <- collect_metrics(enet_results)

# Print a summary of each model
summary(knn_metrics)
```

```
##    neighbors        .metric           .estimator             mean         
##  Min.   : 1.000   Length:12          Length:12          Min.   :      0.2  
##  1st Qu.: 2.000   Class :character   Class :character   1st Qu.:      0.3  
##  Median : 5.000   Mode  :character   Mode  :character   Median : 701308.7  
##  Mean   : 5.167                                         Mean   : 798117.5  
##  3rd Qu.: 8.000                                         3rd Qu.:1497391.5  
##  Max.   :10.000                                         Max.   :1937139.2  
##        n        std_err            .config         
##  Min.   :5   Min.   :     0.02   Length:12         
##  1st Qu.:5   1st Qu.:     0.03   Class :character  
##  Median :5   Median : 50013.47   Mode  :character  
##  Mean   :5   Mean   : 55643.08                     
##  3rd Qu.:5   3rd Qu.:102384.75                     
##  Max.   :5   Max.   :144322.14                     
```

```r
summary(lm_metrics)
```

```
##    .metric           .estimator             mean                 n    
##  Length:2           Length:2           Min.   :      0.4   Min.   :5  
##  Class :character   Class :character   1st Qu.: 338042.6   1st Qu.:5  
##  Mode  :character   Mode  :character   Median : 676084.7   Median :5  
##                                        Mean   : 676084.7   Mean   :5  
##                                        3rd Qu.:1014126.8   3rd Qu.:5  
##                                        Max.   :1352169.0   Max.   :5  
##     std_err            .config         
##  Min.   :     0.04   Length:2          
##  1st Qu.: 30921.09   Class :character  
##  Median : 61842.13   Mode  :character  
##  Mean   : 61842.13                     
##  3rd Qu.: 92763.18                     
##  Max.   :123684.22                     
```

```r
summary(rf_metrics)
```

```
##       mtry       trees            min_n          .metric         
##  Min.   :1   Min.   : 50.0   Min.   : 2.00   Length:896        
##  1st Qu.:2   1st Qu.:162.0   1st Qu.: 6.25   Class :character  
##  Median :4   Median :274.5   Median :10.50   Mode  :character  
##  Mean   :4   Mean   :274.6   Mean   :10.62                     
##  3rd Qu.:6   3rd Qu.:387.0   3rd Qu.:14.75                     
##  Max.   :7   Max.   :500.0   Max.   :20.00
##   .estimator             mean                 n        std_err
##  Length:896         Min.   :      0.5   Min.   :5   Min.   :     0.02
##  Class :character   1st Qu.:      0.5   1st Qu.:5   1st Qu.:     0.03
##  Mode  :character   Median : 616307.2   Median :5   Median : 43110.27
##                     Mean   : 633935.0   Mean   :5   Mean   : 53592.28
##                     3rd Qu.:1259246.4   3rd Qu.:5   3rd Qu.:107473.20
##                     Max.   :1344890.7   Max.   :5   Max.   :133331.98
##    .config
##  Length:896
##  Class :character
##  Mode  :character
##
##
##
```

```
summary(enet_metrics)
```

```
##     penalty            mixture         .metric           .estimator
##  Min.   :0.0000000   Min.   :0.0000   Length:200         Length:200
##  1st Qu.:0.0000000   1st Qu.:0.2222   Class :character   Class :character
##  Median :0.0000194   Median :0.5000   Mode  :character   Mode  :character
##  Mean   :0.1083924   Mean   :0.5000
##  3rd Qu.:0.0059948   3rd Qu.:0.7778
##  Max.   :1.0000000   Max.   :1.0000
##       mean                 n        std_err            .config
##  Min.   :      0.4   Min.   :5   Min.   :     0.04   Length:200
##  1st Qu.:      0.4   1st Qu.:5   1st Qu.:     0.04   Class :character
##  Median : 676028.2   Median :5   Median : 61297.47   Mode  :character
##  Mean   : 676137.3   Mean   :5   Mean   : 61674.79
##  3rd Qu.:1352185.2   3rd Qu.:5   3rd Qu.:123408.50
##  Max.   :1353325.8   Max.   :5   Max.   :123531.16
```

**Step 6: Use RMSE for model evaluation.**

```
# Identifying the Best Model of each type:
best_knn <- knn_results %>% show_best("rmse", n = 1)
best_lm <- lm_results %>% show_best("rmse", n = 1)
best_rf <- rf_results %>% show_best("rmse", n = 1)
best_enet <- enet_results %>% show_best("rmse", n = 1)

# Print the best models.
print(best_knn)
```

```
## # A tibble: 1 x 7
##   neighbors .metric .estimator     mean     n std_err .config
##       <int> <chr>   <chr>         <dbl> <int>   <dbl> <chr>
## 1        10 rmse    standard   1402617.     5 102448. Preprocessor1_Model6
```

```r
print(best_lm)
```

```
## # A tibble: 1 x 6
##   .metric .estimator     mean     n std_err .config
##   <chr>   <chr>         <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   1352169.     5 123684. Preprocessor1_Model1
```

```r
print(best_rf)
```

```
## # A tibble: 1 x 9
##    mtry trees min_n .metric .estimator     mean     n std_err .config
##   <int> <int> <int> <chr>   <chr>         <dbl> <int>   <dbl> <chr>
## 1     4    50     9 rmse    standard   1232614.     5 105118. Preprocessor1_Mod~
```

```r
print(best_enet)
```

```
## # A tibble: 1 x 8
##        penalty mixture .metric .estimator     mean     n std_err .config
##          <dbl>   <dbl> <chr>   <chr>         <dbl> <int>   <dbl> <chr>
## 1 0.0000000001   0.111 rmse    standard   1352056.     5 123531. Preprocessor1_~
```

From the output above, we know that for the best knn model we have Neighbor Count = 10, RMSE = 1402617, and Standard Error = 102447.6.

For the best Linear regression model we have Linear Regression RMSE = 1352169 and Standard Error = 123684.2.

For the best Random Forest model we have mtry = 4, Trees = 50, Min_n = 9, RMSE = 1232614 and Standard Error = 105118.3.

Lastly for the best ENET model, we have Penalty = 1e-10, Mixture = 0.1111111, RMSE = 1352056 and Standard Error: 123531.2.

#### Step 7: Model Comparison.

From the conclusion above, we may conclude that the model with the lowest RMSE is the Random Forest model with an RMSE of 1232614, followed by the ENET model and linear regression model with an RMSE of 1352056 and 1352169 respectively. The KNN model has the highest RMSE value of 1402617.

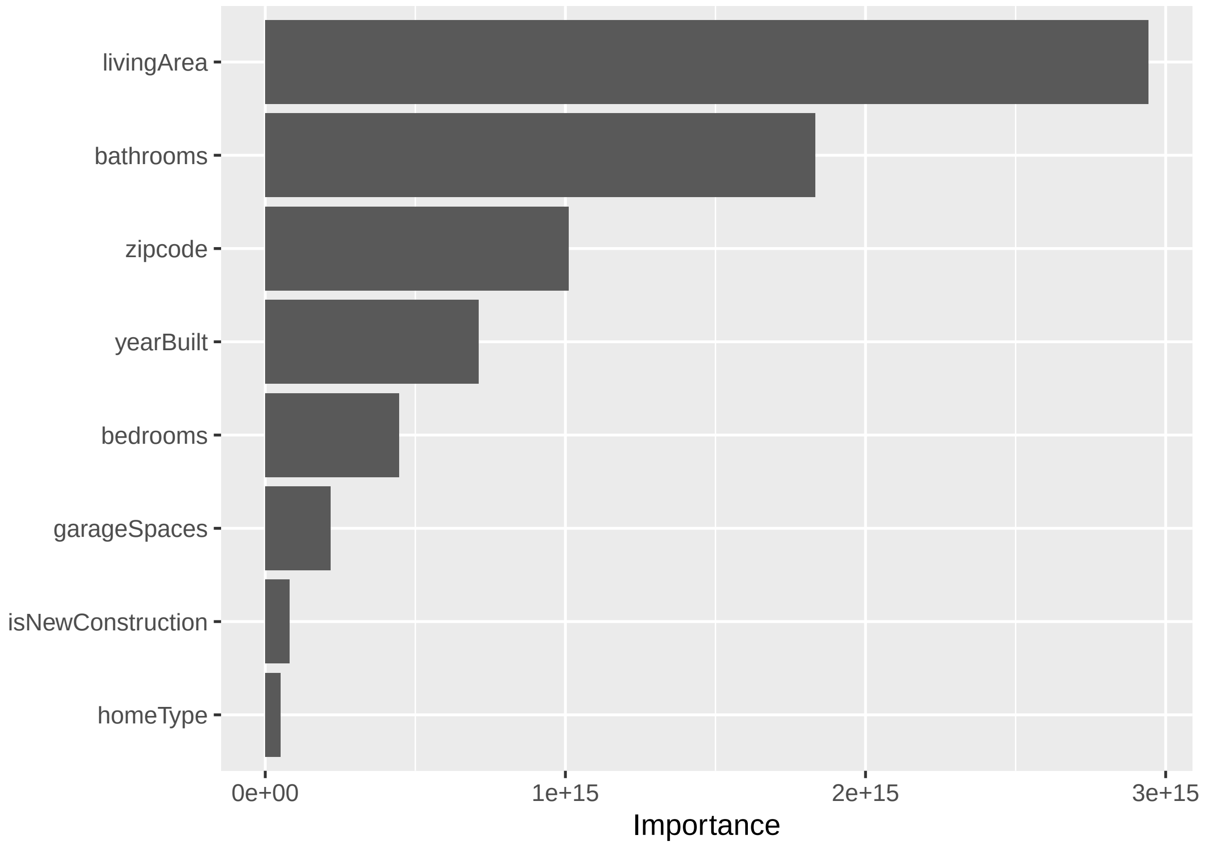
Thus, with respect to the RMSE, the Random Forest model performs the best in predicting real estate prices among the four models.

## Section 5: Final Model Fitting and Testing

#### Step 1: Model Fitting, fit the best model to the entire training set.

```r
# Fit the optimal Random Forest model to the training set
best_rf_model <- rand_forest(
  mtry = 4,
  trees = 50,
  min_n = 9,
  mode = "regression") %>%
  set_engine("ranger", importance = "impurity") %>%
  fit(price ~ ., data = train_data)

# Create the variable importance plot
vip(best_rf_model)
```



By observing the plot, 'livingArea' seems to be the most useful, while 'homeType' seems to be the least useful for the model. And it is surprising that 'bathroom' is more significant than 'zipcode'.

**Step 2: Model Testing, evaluate its performance on the testing set.**

```
# Augment the model to get the predictions in test_data
best_rf_model_test <- augment(best_rf_model, test_data) %>%
  select(price, starts_with(".pred"))

# Calculate RMSE
rmse_test <- sqrt(mean((best_rf_model_test$price - best_rf_model_test$.pred)^2))

# Print the RMSE calculated above
print(rmse_test)
```

```
## [1] 1572970
```

Given that the RMSE for test_data is 1,559,293, and from previous part we have the RMSE for train_data, which was 1,232,614. And the RMSE on the test set is higher than on the training set, which suggests that the model may be overfitting to the training data.

## Conclusion

### Part 1: Summary

In this project, we delved into the intricacies of the real estate market in California, aiming to understand the various factors affecting property prices. Starting with an extensive data cleaning and preparation process, we removed irrelevant columns and handled missing values to create a dataset conducive for analysis.

### Part 2: Key Findings from Exploratory Data Analysis.

Our univariate and bivariate analyses revealed important trends and relationships in the data. Notably, the year a property was built and its living area to significantly affect its price.

Also, one of the most striking findings was the significant variability in property prices across different zip codes and cities. This underscores the importance of geographical location as a determining factor in real estate valuation.

### Part 3: Implications

Understanding these factors can help in making informed decisions, potentially saving or making extra money.

The model can serve as a preliminary tool for investment decisions, although further refinement is advisable.

### Part 4: Future Work

Data Adding: Now the RMSE value of the models are still very high, making the reliability low. I would consider using more data to train the model, given that we selected only 4000 observations when training the model. Also, adding more data in the future may help train the models even better.

Data Enrichment: Incorporating more variables like proximity to amenities, rate of surrounded schools or crime rates could improve our model.

Model Tuning: Further fine-tuning of the model parameters can likely improve its predictive power. For example, in the variable importance plot we may notice that 'homeType' and 'isNewConstruction' provide little help on predicting the price.

Temporal Analysis: Analyzing how property prices change over time could provide additional insights. And if we find any unusual price change at some time, we may also

External Factors and Unusual Price Changes: Future studies should investigate the influence of external factors such as policy changes or international incidents when there's an unusual change in property prices. This would add another layer of complexity and realism to our model, enabling it to be more responsive to real-world scenarios. Also the model could be used as a tool for policy makers and country leaers when making real estate related decisions.