Final Assignment: Exploring the dataset of the Portuguese Housing Real Estate

Sergio Vega García Marta Gonzalez Juan Andreas Manuel Korn
Juan Arturo Abaurrea Calafell Marc Roman Colom Andrés Borrás Santos

2025-01-28

Contents

INSTRUCTIONS

You have until December 2nd, 2024 at 23.55h to upload a document indicating the names of the students in each group. One document per group. After this date, all changes/incorporations will be handled by the teachers.

- Final report due date: January 28th, 2025 at 23.55h.
- Each group must turn in a report explaining the work done and the results obtained. Maximum length: equivalent to 8 pages in pdf format plus cover page.
- The project must be done in R. Rmd files must be turned in together with its html/pdf output.
- For deadline and uploading instructions, please consult aula digital.

The project has two parts:

PART 1: Data analysis.

- In this part, you can apply all concepts seen throughout the course. You may consider all features or a subset depending on the research questions asked in Part II.
- Include data preprocessing, an important step in the data mining process that involves cleaning and transforming raw data to make it suitable for analysis.
- Describe the process and effort required.
- How many examples are in the data set? How many features? Which will be used?
- Consider data reduction techniques, which aim at reducing the complexity of the data, detecting or removing irrelevant and noisy elements from the data.

PART 2: Context of the problem, models and evaluation.

- You should start by giving a brief description of what you plan to do.
- What problems are you trying to solve? Be sure to formulate the problem as a data mining problem (is it a classification problem, a clustering problem, association rule mining, . . .)?
- What exactly are you trying to predict (for prediction tasks), group (for clustering tasks), ... ?
- How will you evaluate your results?
- How will you know if your results are good?
- It is critical that your problem is well-defined.

- What learning tools do you plan to use and what techniques/algorithms do you plan to use (decision trees, Apriori, ...)?

Depending on how complex your project is, you should consider including parameter tuning, key features of the data distribution, dimensionality reduction, ...

```
knitr::opts_chunk$set(echo = TRUE)
```

```
# Stable seed
set.seed(6969)
random_forest_samples <- 15000
nTrees <- 100

# Requires the following packages before running the code at start
if (!require("DataExplorer")) install.packages("DataExplorer")
```

```
## Cargando paquete requerido: DataExplorer
```

```
## Warning: package 'DataExplorer' was built under R version 4.4.2
```

```
if (!require("dplyr")) install.packages("dplyr")
```

```
## Cargando paquete requerido: dplyr
```

```
##
## Adjuntando el paquete: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
if (!require("ggplot2")) install.packages("ggplot2")
```

```
## Cargando paquete requerido: ggplot2
```

```
if (!require("randomForest")) install.packages("randomForest")
```

```
## Cargando paquete requerido: randomForest
```

```
## Warning: package 'randomForest' was built under R version 4.4.2
```

```
## randomForest 4.7-1.2
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Adjuntando el paquete: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
if (!require("caret")) install.packages("caret")
```

```
## Cargando paquete requerido: caret
```

```
## Warning: package 'caret' was built under R version 4.4.2
## Cargando paquete requerido: lattice
if (!require("class")) install.packages("class")
## Cargando paquete requerido: class
## Warning: package 'class' was built under R version 4.4.2
if (!require("maps")) install.packages("maps")
## Cargando paquete requerido: maps
## Warning: package 'maps' was built under R version 4.4.2
if (!require("leaflet")) install.packages("leaflet")
## Cargando paquete requerido: leaflet
## Warning: package 'leaflet' was built under R version 4.4.2
if (!require("ggmap")) install.packages("ggmap")
## Cargando paquete requerido: ggmap
## Warning: package 'ggmap' was built under R version 4.4.2
## i Google's Terms of Service: <https://mapsplatform.google.com>
##   Stadia Maps' Terms of Service: <https://stadiamaps.com/terms-of-service/>
##   OpenStreetMap's Tile Usage Policy: <https://operations.osmfoundation.org/policies/tiles/>
## i Please cite ggmap if you use it! Use `citation("ggmap")` for details.
if (!require("RColorBrewer")) install.packages("RColorBrewer")
## Cargando paquete requerido: RColorBrewer
if (!require("cluster")) install.packages("cluster")
## Cargando paquete requerido: cluster
##
## Adjuntando el paquete: 'cluster'
## The following object is masked from 'package:maps':
##
##     votes.repub
if (!require("mapproj")) install.packages("mapproj")
## Cargando paquete requerido: mapproj
## Warning: package 'mapproj' was built under R version 4.4.2
# Load the required libraries
library(DataExplorer) # To ease the first step into investigating data; getting missing data and a bett
library(dplyr) # For data manipulation
library(ggplot2) # For plotting purposes
library(randomForest) # Random forest library
library(caret) # used to create one-hot-encoding or dummy variables.
library(class)
library(maps)
library(leaflet)
library(ggmap)
```

```r
library(RColorBrewer)
library(cluster)
library(mapproj)

# Custom Functions
## Fast print function
printf <- function(...) cat(sprintf(...), "\n")
## Plot a pie and return a dataframe with the counts and percentages
plot_pie <- function(data, title) {
  # Create pie graph
  pie(table(data), main = title, col = rainbow(length(table(data))), cex.main = 2, labels = "")

  # Add percentages legend
  legend("topright",
    legend = paste(names(table(data)), "-", round(100 * table(data) / sum(table(data)), 2), "%"),
    cex = 0.6, fill = rainbow(length(table(data)))
  )

  # Calculate percentages and counts
  type_counts <- table(data) # Crear tabla de frecuencias
  total_counts <- sum(type_counts) # Sumar todos los valores de la tabla
  percentages <- round(100 * as.vector(type_counts) / total_counts, 2) # Calcular porcentajes

  # Return a datagrame with types, counts and percentages
  return(data.frame(
    Type = names(type_counts),
    Count = as.vector(type_counts),
    Percentage = percentages
  ))
}


## Gets the Random forest done doing the respective partitioning and print and plots it.
getRandomForest <- function(formula, dataset, classname, nrOfSamples, nrOfTrees) {
  # Filter only non-empty values for the target class
  data_to_train <- dataset[!is.na(dataset[[classname]]), ]

  # Sample rows from the dataset
  data_to_train <- data_to_train[sample(1:nrow(data_to_train), min(nrOfSamples, nrow(data_to_train))), ]

  # Create the random forest model using the formula
  random_forest <- randomForest(formula, data = data_to_train, importance = TRUE, ntree = nrOfTrees)

  # Print and visualize variable importance
  # print(random_forest)
  # varImpPlot(random_forest)
  return(random_forest)
}
```

Introduction

In this section, we will delve deeper into the dataset to better understand its structure and content.

```r
# load the data and libraries
data <- read.csv("portugal_housing.csv")
original_data <- data # unmodified data
```

Data summary

The dataset contains information about real estate properties in Portugal. It includes various features such as the price, location, type, area, number of rooms, and other characteristics of the properties. Let's take a look at the first few rows of the dataset and generate a summary to understand the data better.

```r
head(data)
```

```
##     Price District                City                       Town      Type
## 1  250000     Faro São Brás de Alportel     São Brás de Alportel Apartment
## 2    9500     Faro            Albufeira Albufeira e Olhos de Água Apartment
## 3  580000     Faro        Vila do Bispo                   Budens Apartment
## 4  350000     Faro             Portimão                 Portimão Apartment
## 5  175000     Faro                 Faro     Faro (Sé e São Pedro)     House
## 6 1485000     Faro               Loulé                 Quarteira     House
##   EnergyCertificate        Floor  Lift Parking HasParking ConstructionYear
## 1                A+    2nd Floor  True       1       True               NA
## 2                NC    1st Floor  True       0      False             1990
## 3                 D    3rd Floor False       1       True             2003
## 4                 C    4th Floor  True       0      False             1985
## 5                NC             False       0      False             1950
## 6                 D Ground Floor False       2       True             2004
##   TotalArea GrossArea PublishDate Garage Elevator ElectricCarsCharging
## 1       114        NA
## 2        27        NA
## 3        84        NA
## 4        68        NA
## 5        78        NA
## 6       472        NA
##   TotalRooms NumberOfBedrooms NumberOfWC ConservationStatus LivingArea LotSize
## 1          2               NA         NA                           114      NA
## 2          0               NA         NA                            27      NA
## 3          2               NA         NA                            84      NA
## 4          2               NA         NA                            68      NA
## 5          4               NA         NA                            78      NA
## 6          4               NA         NA                           414      NA
##   BuiltArea NumberOfBathrooms
## 1        NA                2
## 2        NA                1
## 3        NA                2
## 4        NA                1
## 5        NA                2
## 6        NA                5
```

```r
summary(data)
```

```
##      Price             District             City               Town
##  Min.   :1.000e+02   Length:114623      Length:114623      Length:114623
##  1st Qu.:7.800e+04   Class :character   Class :character   Class :character
##  Median :2.000e+05   Mode  :character   Mode  :character   Mode  :character
##  Mean   :3.666e+05
```

```
##  3rd Qu.:3.900e+05
##  Max.   :1.380e+09
##  NA's   :244
##      Type           EnergyCertificate     Floor               Lift
##  Length:114623      Length:114623      Length:114623      Length:114623
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##     Parking        HasParking        ConstructionYear   TotalArea
##  Min.   :0.0000   Length:114623      Min.   :1900     Min.   :-7.196e+06
##  1st Qu.:0.0000   Class :character   1st Qu.:1972     1st Qu.: 9.500e+01
##  Median :0.0000   Mode  :character   Median :1994     Median : 1.770e+02
##  Mean   :0.5847                      Mean   :1989     Mean   : 6.040e+05
##  3rd Qu.:1.0000                      3rd Qu.:2008     3rd Qu.: 6.210e+02
##  Max.   :3.0000                      Max.   :2024     Max.   : 6.142e+10
##  NA's   :194                         NA's   :41073    NA's   :6452
##    GrossArea        PublishDate          Garage            Elevator
##  Min.   :      -7   Length:114623      Length:114623      Length:114623
##  1st Qu.:     100   Class :character   Class :character   Class :character
##  Median :     164   Mode  :character   Mode  :character   Mode  :character
##  Mean   :    2790
##  3rd Qu.:     294
##  Max.   :12750000
##  NA's   :86985
##  ElectricCarsCharging   TotalRooms      NumberOfBedrooms   NumberOfWC
##  Length:114623        Min.   :   0.00   Min.   : 0.00    Min.   :-15.00
##  Class :character     1st Qu.:   2.00   1st Qu.: 2.00    1st Qu.:  0.00
##  Mode  :character     Median :   3.00   Median : 3.00    Median :  0.00
##                       Mean   :   3.11   Mean   : 2.67    Mean   :  0.41
##                       3rd Qu.:   4.00   3rd Qu.: 3.00    3rd Qu.:  0.00
##                       Max.   :2751.00   Max.   :21.00    Max.   : 59.00
##                       NA's   :49355     NA's   :83931    NA's   :73097
##  ConservationStatus   LivingArea         LotSize            BuiltArea
##  Length:114623      Min.   :      0   Min.   :        0   Min.   :      -1
##  Class :character   1st Qu.:     80   1st Qu.:      296   1st Qu.:     107
##  Mode  :character   Median :    121   Median :      795   Median :     178
##                     Mean   :   1542   Mean   :    73454   Mean   :    3682
##                     3rd Qu.:    215   3rd Qu.:     3300   3rd Qu.:     322
##                     Max.   :5429000   Max.   :992301000   Max.   :12750000
##                     NA's   :27260     NA's   :85694       NA's   :96234
##  NumberOfBathrooms
##  Min.   :-13.000
##  1st Qu.:  0.000
##  Median :  1.000
##  Mean   :  1.503
##  3rd Qu.:  2.000
##  Max.   : 90.000
##  NA's   :5355
```

Our dataset contains 114623 samples with 25 features. There are two principal types of variables in the original dataset: numeric and character. There are 12 numeric variables and 13 character variables.

Understanding the variables

Here's a brief explanation of all 25 variables of the original dataset:

- *Price*: Numeric - The price of the property in euros.
- *District*: Character - The district in Portugal where the property is located.
- *City*: Character - The city where the property is situated.
- *Town*: Character - A more specific subdivision or town within the city.
- *Type*: Character - The type of property (e.g., Apartment, House, etc.).
- *EnergyCertificate*: Character - The energy rating of the property.
- *Floor*: Character - The floor where the property is located.
- *Lift*: Character - Indicates if the building has an elevator (True or False).
- *Parking*: Numeric - The total number of parking spaces available.
- *HasParking*: Character - Specifies if the property has parking (True or False).
- *ConstructionYear*: Numeric - The year the property was built.
- *TotalArea*: Numeric - The total area of the property (in square meters).
- *GrossArea*: Numeric - The gross area, including constructed surfaces like balconies.
- *PublishDate*: Character - The date the property was listed.
- *Garage*: Character - Indicates if the property includes a garage.
- *Elevator*: Character - Similar to Lift.
- *ElectricCarsCharging*: Character - Specifies if electric car charging is available.
- *TotalRooms*: Numeric - The total number of rooms in the property.
- *NumberOfBedrooms*: Numeric - The specific number of bedrooms.
- *NumberOfWC*: Numeric - The number of water closets (likely includes toilets).
- *ConservationStatus*: Character - The condition of the property (e.g., new, needs renovation).
- *LivingArea*: Numeric - The actual habitable area.
- *LotSize*: Numeric - The size of the land associated with the property.
- *BuiltArea*: Numeric - The constructed area.
- *NumberOfBathrooms*: Numeric - The total number of full bathrooms.

Data Extension and Feature Engineering

In this section, we will create new variables based on the existing ones to provide additional and create new insights into the data. We will create new variables based on the existing ones or any other information to provide additional insights into the data and answer the research questions.

Coastal Property

```
# Read the PT.txt file from the GeoNames dataset
# https://download.geonames.org/export/dump/PT.zip
if (!file.exists("PT.txt")) {
  destfile <- "portugal_points_data.zip"
  download.file("https://download.geonames.org/export/dump/PT.zip", destfile)
  unzip(destfile)
}
geonames_pt <- read.delim("PT.txt", header = FALSE, sep = "\t", stringsAsFactors = FALSE)

# Assign column names based on the documentation (the readme.txt downloaded with PT.txt)
colnames(geonames_pt) <- c("geonameid", "name", "asciiname", "alternatenames", "latitude", "longitude",
                           "feature_class", "feature_code", "country_code", "cc2", "admin1_code",
                           "admin2_code", "admin3_code", "admin4_code", "population", "elevation", "dem
                           "timezone", "modification_date")

geonames_pt <- geonames_pt %>%
```

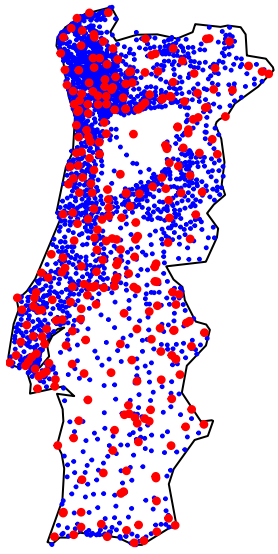
```
  group_by(name) %>%
  slice_max(order_by = population, n = 1) %>%
  ungroup()

matched_cities <- geonames_pt[geonames_pt$name %in% unique(data$City), ]
matched_towns <- geonames_pt[geonames_pt$name %in% unique(data$Town), ]

original_city_count <- length(unique(data$City))
matched_city_count <- length(unique(matched_cities$name))

original_town_count <- length(unique(data$Town))
matched_town_count <- length(unique(matched_towns$name))

cat(paste0("Number of not considered cities: ", original_city_count - matched_city_count, " (", round((
```

```
## Number of not considered cities: 4 (1.47%)
```

```
cat(paste0("Number of not considered towns: ", original_town_count - matched_town_count, " (", round((o
```

```
## Number of not considered towns: 96 (4.29%)
```

```
map("world", "Portugal")
points(matched_towns$longitude, matched_towns$latitude, col = "blue", pch = 20, cex = 0.3)
points(matched_cities$longitude, matched_cities$latitude, col = "red", pch = 20, cex = 0.7)
```



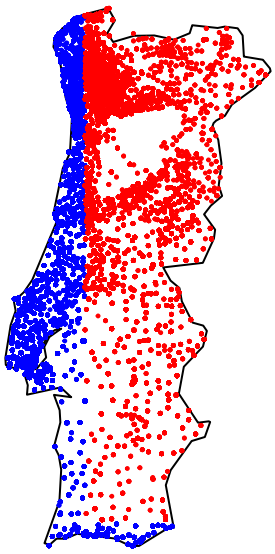
```
is_coastal_function <- function(lon, lat) {
# By using this function, we ensure that the samples that are considered is_coastal are, as a maximum,
```

```r
  MAX_LON <- -8.5
  MAX_LAT <- 37.25
  return(lon < MAX_LON | lat < MAX_LAT)
}

matched_towns <- matched_towns %>%
 select(name, latitude, longitude) %>%
 rename(Town = name)

assign_city_coords <- function(town_name) {
  city_name <- data$City[data$Town == town_name][1]
  city_coords <- matched_cities[matched_cities$name == city_name, c("latitude", "longitude")]
  if(nrow(city_coords) > 0) {
    is_coastal <- is_coastal_function(city_coords$longitude[1], city_coords$latitude[1])
    return(list(
      latitude = city_coords$latitude[1],
      longitude = city_coords$longitude[1],
      is_coastal = is_coastal
    ))
  }
  return(list(latitude = NA, longitude = NA, is_coastal = NA))
}

data <- data %>%
  left_join(matched_towns, by = "Town") %>%
  group_by(Town) %>%
  mutate(
    latitude = ifelse(is.na(latitude),
                      assign_city_coords(Town)$latitude,
                      latitude),
    longitude = ifelse(is.na(longitude),
                       assign_city_coords(Town)$longitude,
                       longitude)
  ) %>%
  mutate(
    is_coastal = is_coastal_function(longitude, latitude)
  ) %>%
  ungroup()
```

```
## Warning in left_join(., matched_towns, by = "Town"): Detected an unexpected many-to-many relationshi
## i Row 6 of `x` matches multiple rows in `y`.
## i Row 759 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
## Warning: There were 190 warnings in `mutate()`.
## The first warning was:
## i In argument: `latitude = ifelse(is.na(latitude),
##   assign_city_coords(Town)$latitude, latitude)`.
## i In group 1: `Town = ""`.
## Caused by warning in `data$Town == town_name`:
## ! longitud de objeto mayor no es múltiplo de la longitud de uno menor
## i Run `dplyr::last_dplyr_warnings()` to see the 189 remaining warnings.
```

```r
d1 <- data[data$is_coastal, ]
d2 <- data[!data$is_coastal, ]

map("world", "Portugal")
points(d1$longitude, d1$latitude, col = "blue", pch = 20, cex = 0.3)
points(d2$longitude, d2$latitude, col = "red", pch = 20, cex = 0.3)
```



```r
matched_towns <- matched_towns %>%
  mutate(is_coastal = is_coastal_function(longitude, latitude))

leaflet(data = matched_towns) %>%
  addTiles() %>%
  addProviderTiles("Esri.WorldImagery") %>%  # Satellite map
  addCircleMarkers(
    lng = ~longitude, lat = ~latitude,
    color = ~ifelse(is_coastal, "blue", "red"),
    radius = 2, popup = ~Town
  )
```

PART 1: Data analysis

- **Duplicated rows**: We found 6 duplicated rows. We will not remove them as many apartments in the same area often are designed with the same features.
- **Negative values**: We found 13 negative and invalid values; they will be removed.
- **Inconsistent values**: We found cases where `totalRooms` is smaller than the sum of `wc`, `bathrooms`,

and `bedrooms`. Since there are a lot of them, we will treat them as NA to handle them later.

- **Empty values**: We will treat empty values as NA.
- **Missing values**: We will handle them depending on the amount:
  - **Remove**: Due to the small number of samples affected, we will delete these rows:
    - `Type`: 16 samples
    - `Town`: 2 samples
    - `Price`: 244 samples
    - `EnergyCertificate`: 14 samples
    - `latitude`: 63 samples
    - `longitude`: 63 samples
    - `is_coastal`: 63 samples
  - **Merge**:
    - **Elevator and Lift**: They represent the same concept. Since no collisions were found between the two, we merged them into `Lift`, prioritizing it as it contains more observations, and set any remaining missing values to "False".
    - **Parking and HasParking**: These two columns represent the same concept. Missing values in `HasParking` are updated based on the corresponding `Parking` values (`> 0` is set to "True", otherwise "False"). Remaining NA values in `HasParking` are set to "False", and the `Parking` column is removed to avoid redundancy.
  - **Impute**:
    - **ElectricCarsCharging**: Any missing values will be set to "False".
    - **Grouping variables**:
      - `Other`: We will group the less frequent property types into a new category called "Other" to simplify the analysis.
      - `Other Rooms`: This variable is the difference between the total number of rooms and the room variables. `TotalRooms` is a superset of `NumberofBedrooms`, `NumberOfWC`, `NumberOfBathrooms`, and `OtherRooms`, but `OtherRooms` are not included in the dataset. We will create this variable to provide more information and remove the covariances between them, increasing the quality of the dataset.
    - **Prediction using Random Forest**:
      - `Floor`
      - `ConstructionYear`
      - `TotalArea`
      - `GrossArea`
      - `PublishDate`
      - `Garage`
      - `TotalRooms`
      - `NumberOfBedrooms`
      - `NumberOfWC`
      - `ConservationStatus`
      - `LivingArea`
      - `LotSize`
      - `BuiltArea`
      - `NumberOfBathrooms`

```r
# Negative values
data_varNames_numerics <- names(data)[sapply(data, is.numeric)] # Identify numeric columns
data_varNames_numerics_without_coordenates <- setdiff(
  names(data)[sapply(data, is.numeric)],
  c("longitude", "latitude")
)
rows_with_negative <- rowSums(data[data_varNames_numerics_without_coordenates] < 0, na.rm = TRUE) > 0
num_rows_with_negative <- sum(rows_with_negative)
data <- data[!rows_with_negative, ]
```

```r
# Inconsistent values
data$TotalRooms[data$TotalRooms < data$NumberOfWC + data$NumberOfBathrooms + data$NumberOfBedrooms] <- I
# Empty values are now missing values
data[data == ""] <- NA

# Handling missing values
# Deleting rows with empty 'Type' values
data <- data[!is.na(data$Type), ]
data <- data[!is.na(data$EnergyCertificate), ]
# Set ElectricCarsCharging and Garage NA values to False
data$ElectricCarsCharging[is.na(data$ElectricCarsCharging)] <- "False"
data$Garage[is.na(data$Garage)] <- "False"
# Remove NA from longitude, latitude and is_coastal
data <- data[!is.na(data$longitude), ]
data <- data[!is.na(data$latitude), ]
data <- data[!is.na(data$is_coastal), ]
```

Merging

Merging is a common operation in data preprocessing. It allows us to combine similar or related variables to simplify the dataset and reduce redundancy. In this case, we will merge the `Lift` and `Elevator` variables into a single variable called `Lift`. We will also merge the `Parking` and `HasParking` variables into a single variable called `HasParking`.

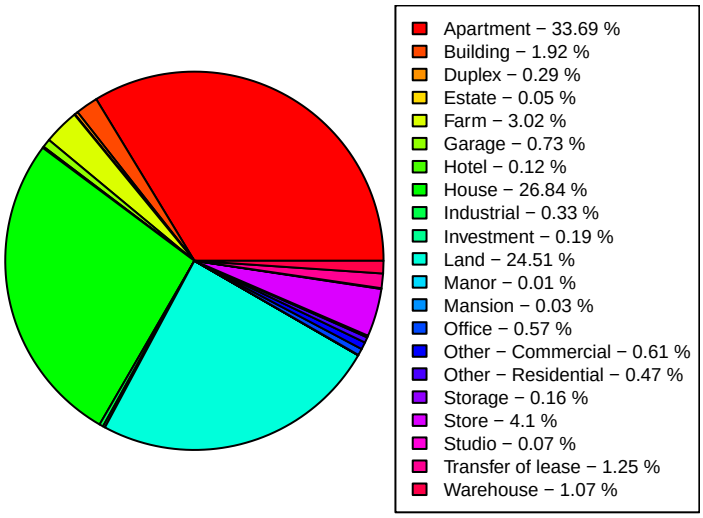
```r
#Merge
# Merged 'Lift' and 'Elevator' into 'Lift'
data$Lift[is.na(data$Lift)] <- data$Elevator[is.na(data$Lift)]
# Set remaining 'Lift' values to False
data$Lift[is.na(data$Lift)] <- "False"
# Remove the 'Elevator' column
data$Elevator <- NULL


data$HasParking[is.na(data$HasParking)] <- ifelse(!is.na(data$Parking[is.na(data$HasParking)]) & data$Pa
# Set remaining 'HasParking' values to False
data$HasParking[is.na(data$HasParking)] <- "False"
# Remove the 'Parking' column
data$Parking <- NULL
```

Grouping variables

Grouping variables is a common operation in data preprocessing. It allows us to simplify the dataset by combining less frequent categories into a single category. In this case, we will group less frequent property types into a new category called "Other" to simplify the analysis.

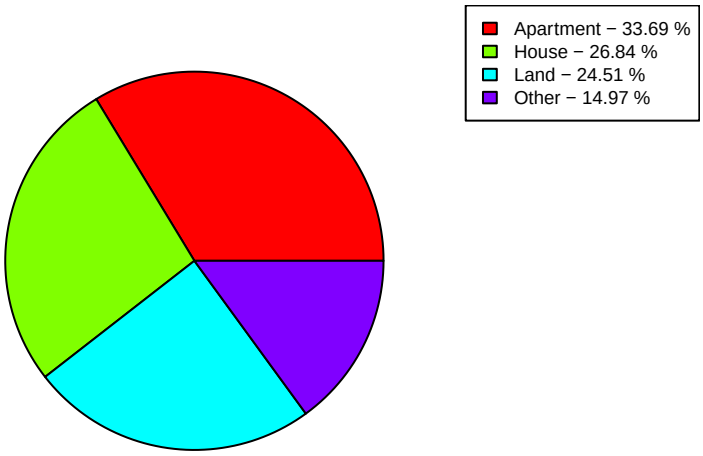
```r
# Group less frequent property types into 'Other'
# We will see the unique values of the 'Type' column in a pie chart
types_percentage <- plot_pie(data$Type, "Distribution of Property Types")
```

Distribution of Property Types




```r
# Group less frequent property types into 'Other'
min_percentage <- 5 # Minimum percentage to not group
less_frequent_types <- types_percentage[types_percentage$Percentage <= min_percentage, "Type"]
data$Type[data$Type %in% less_frequent_types] <- "Other"

# Check the distribution of property types after grouping
plot_pie(data$Type, "After grouping distribution")
```

After grouping distribution




```
##        Type Count Percentage
## 1 Apartment 40552      33.69
## 2     House 32304      26.84
## 3      Land 29503      24.51
## 4     Other 18021      14.97
```

```
# Group less frequent property types into 'Other'
```

Outlier detection and removal

Outliers are always a big deal in data analysis, they can affect the results of the analysis and the performance of the models. We must reduce the impact of those elements in the dataset.

There are a lot of good algorithms to detect outliers, but we will use the IQR method due to his simplicity, efficiency and robustness.

```
# Numeric columns excluding 'NumberOfWC'
data_varNames_numerics <- setdiff(names(data)[sapply(data, is.numeric)], "NumberOfWC")

# Calculate IQR for NumberOfWC with special quantiles (1% and 99%)
Q1_wc <- quantile(data$NumberOfWC, probs = 0.01, na.rm = TRUE)
Q3_wc <- quantile(data$NumberOfWC, probs = 0.99, na.rm = TRUE)

# Calculate IQR
IQR_wc <- Q3_wc - Q1_wc

# Calculate lower and upper bounds for NumberOfWC
lower_bound_wc <- Q1_wc - 1.5 * IQR_wc
```

```r
upper_bound_wc <- Q3_wc + 1.5 * IQR_wc

# Detect outliers for NumberOfWC
outliers_wc <- data$NumberOfWC < lower_bound_wc | data$NumberOfWC > upper_bound_wc

# Set outliers to NA for NumberOfWC
data <- data
data$NumberOfWC[outliers_wc] <- NA

# Calculate IQR for other numeric columns (standard quantiles)
Q1 <- apply(data[data_varNames_numerics], 2, quantile, probs = 0.25, na.rm = TRUE)
Q3 <- apply(data[data_varNames_numerics], 2, quantile, probs = 0.75, na.rm = TRUE)

# Calculate IQR for other numeric columns
IQR <- Q3 - Q1

# Calculate lower and upper bounds for other numeric columns
lower_bound <- Q1 - 1.5 * IQR
upper_bound <- Q3 + 1.5 * IQR

# Detect outliers for other numeric columns
outliers <- apply(data[data_varNames_numerics], 1, function(row) {
  # Check if any value in the row is outside the lower or upper bound
  row < lower_bound | row > upper_bound
})

# Create a logical matrix where TRUE indicates an outlier for other numeric columns
outlier_matrix <- t(outliers)

# Replace outliers with NA for other numeric columns
data[data_varNames_numerics][outlier_matrix] <- NA

# Optionally, return data without outliers or proceed with further analysis
summary(data)
```

```
##      Price          District             City              Town
##  Min.   :   100   Length:120380      Length:120380      Length:120380
##  1st Qu.: 70000   Class :character   Class :character   Class :character
##  Median :186000   Mode  :character   Mode  :character   Mode  :character
##  Mean   :235797
##  3rd Qu.:341500
##  Max.   :877500
##  NA's   :9936
##      Type           EnergyCertificate     Floor              Lift
##  Length:120380      Length:120380      Length:120380      Length:120380
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
##   HasParking        ConstructionYear   TotalArea       GrossArea
##  Length:120380      Min.   :1918     Min.   :   0    Min.   :  0.0
##  Class :character   1st Qu.:1973     1st Qu.:  85    1st Qu.: 94.0
```

```
##  Mode  :character   Median :1994     Median : 137    Median :146.0
##                     Mean   :1989     Mean   : 240    Mean   :177.3
##                     3rd Qu.:2008     3rd Qu.: 281    3rd Qu.:238.0
##                     Max.   :2024     Max.   :1357    Max.   :577.0
##                     NA's   :43159    NA's   :25916   NA's   :94393
##  PublishDate          Garage          ElectricCarsCharging   TotalRooms
##  Length:120380      Length:120380      Length:120380        Min.   :1.00
##  Class :character   Class :character   Class :character     1st Qu.:2.00
##  Mode  :character   Mode  :character   Mode  :character     Median :3.00
##                                                            Mean   :2.61
##                                                            3rd Qu.:3.00
##                                                            Max.   :4.00
##                                                            NA's   :73901
##  NumberOfBedrooms   NumberOfWC    ConservationStatus   LivingArea
##  Min.   :1.00     Min.   : 0.0    Length:120380      Min.   :  0.0
##  1st Qu.:2.00     1st Qu.: 0.0    Class :character   1st Qu.: 76.0
##  Median :3.00     Median : 0.0    Mode  :character   Median :110.0
##  Mean   :2.64     Mean   : 0.4                       Mean   :132.1
##  3rd Qu.:3.00     3rd Qu.: 0.0                       3rd Qu.:168.0
##  Max.   :4.00     Max.   :10.0                       Max.   :415.0
##  NA's   :94738    NA's   :76801                      NA's   :38560
##     LotSize        BuiltArea      NumberOfBathrooms    latitude
##  Min.   :   0    Min.   :  0.0    Min.   :0.000     Min.   :36.95
##  1st Qu.: 251    1st Qu.:100.0    1st Qu.:0.000     1st Qu.:38.74
##  Median : 575    Median :157.0    Median :1.000     Median :39.67
##  Mean   :1324    Mean   :189.6    Mean   :1.384     Mean   :39.76
##  3rd Qu.:1650    3rd Qu.:252.1    3rd Qu.:2.000     3rd Qu.:41.15
##  Max.   :7662    Max.   :640.0    Max.   :5.000     Max.   :42.13
##  NA's   :94354   NA's   :103301   NA's   :7945      NA's   :1163
##    longitude      is_coastal
##  Min.   :-9.447   Mode :logical
##  1st Qu.:-9.088   FALSE:40028
##  Median :-8.611   TRUE :80352
##  Mean   :-8.602
##  3rd Qu.:-8.314
##  Max.   :-7.105
##  NA's   :3470
```

Removing a few more NA values

From the previous operations, there are now 2 values from Town that are NA, we shall remove them.

```
sum(is.na(data$Town))
```

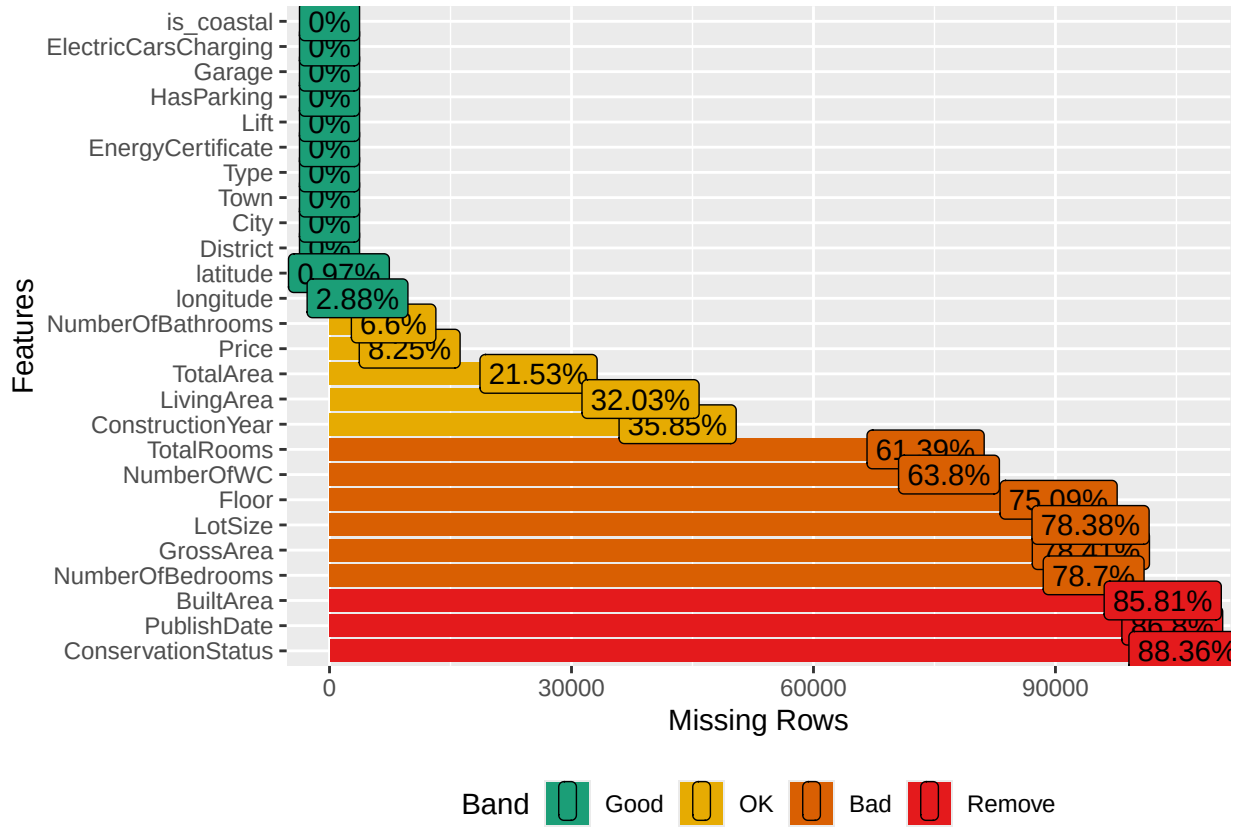
```
## [1] 2
```

```
data <- data[!is.na(data$Town), ]
```

Imputing NA values

We will use the Random Forest algorithm to impute the missing values in the dataset. We will iterate over the columns with missing values and use the Random Forest algorithm to predict them based on the other variables in the dataset. Imputation will, in theory, help improve the performance of the models due to the fact that we are not losing information.

```r
no_imputed_data <- data # for question 1

plot_missing(data)
```




```r
# We obtain the name of the columns with the most nas from least to most
nas <- sort(apply(data, 2, function(x) sum(is.na(x))), decreasing = FALSE)
# Remove 0s
nas <- names(nas[nas > 0])
nas
```

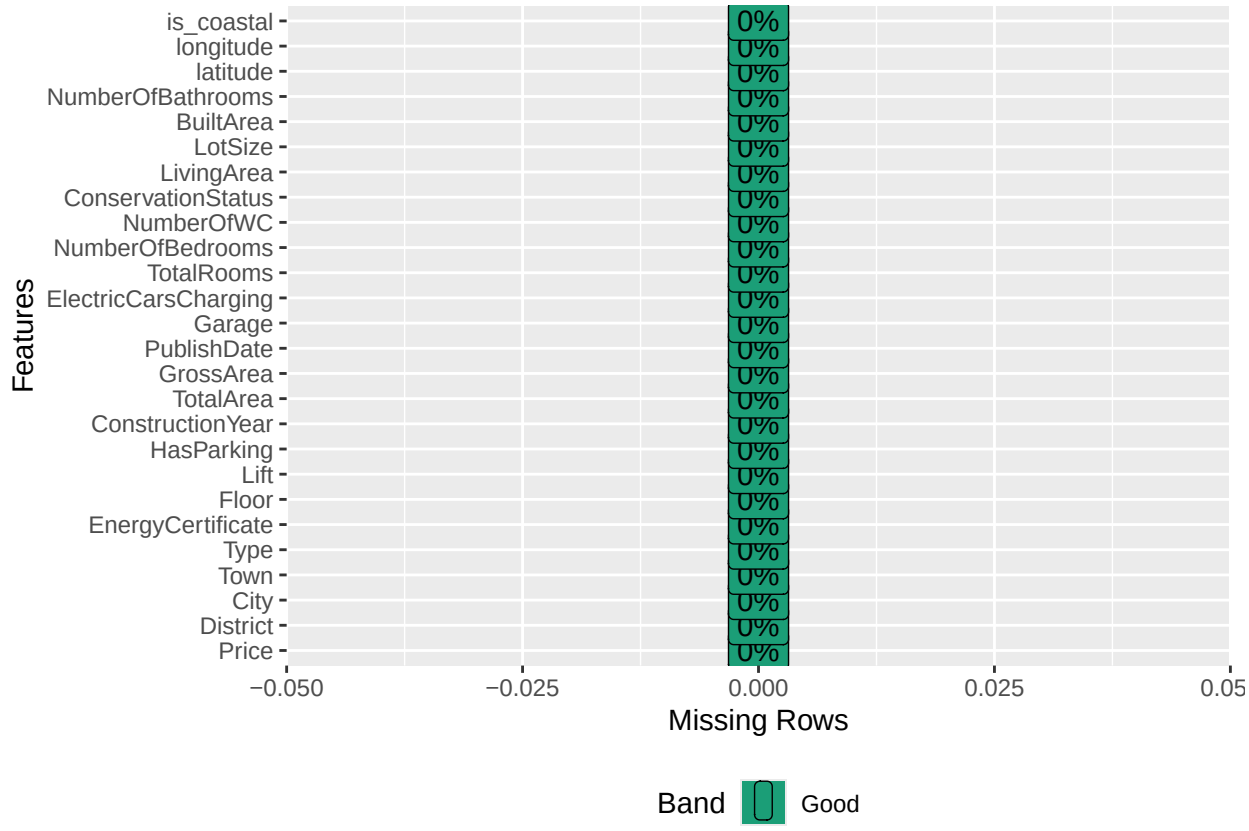
```
##  [1] "latitude"           "longitude"          "NumberOfBathrooms"
##  [4] "Price"              "TotalArea"          "LivingArea"
##  [7] "ConstructionYear"   "TotalRooms"         "NumberOfWC"
## [10] "Floor"              "LotSize"            "GrossArea"
## [13] "NumberOfBedrooms"   "BuiltArea"          "PublishDate"
## [16] "ConservationStatus"
```

```r
form_base <- "District + Type + EnergyCertificate + HasParking"

for (col in nas) {
  form <- as.formula(paste0(col, " ~ ", form_base))

  if (!is.numeric(data[[col]])) {
    data[[col]] <- as.numeric(as.factor(data[[col]]))
  }

  random_forest <- getRandomForest(formula = form, dataset = data, col, nrOfSamples = random_forest_samp
```

```
  # Fix the indexing
  na_rows <- is.na(data[[col]])
  data[[col]][na_rows] <- predict(random_forest, data[na_rows, ])

  form_base <- paste0(form_base, " + ", col)
}
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

```
plot_missing(data)
```



Other rooms

We create another variable to remove some dependency between variables.

```
# Create a new variable 'OtherRooms'
data$OtherRooms <- data$TotalRooms - (data$NumberOfBedrooms + data$NumberOfWC + data$NumberOfBathrooms)
data$OtherRooms[data$OtherRooms < 0] <- 0
data$TotalRooms <- NULL
```

Variable Formatting

There are a lot of variables with wrong data types. To better manage and operate with the data, we might want to change the data type to the correct one.

Boolean variables

There are a lot of True/False variables that are misclassified as character variables, we will convert them to logical variables. This will help us to better operate with the data, ease the analysis and make code more understandable.

```
# # Get names of variables with "True"/"False" values
data_varNames_logical <- names(data)[sapply(data, function(x) all(x %in% c("True", "False")))]
data_varNames_logical
```

```
## [1] "Lift"                 "HasParking"           "Garage"
## [4] "ElectricCarsCharging"
```

```
# # Convert logical variables to logical type
data[data_varNames_logical] <- lapply(data[data_varNames_logical], as.logical)


data_varNames_logical <- names(data)[sapply(data, function(x) all(x %in% c("True", "False")))]
no_imputed_data[data_varNames_logical] <- lapply(no_imputed_data[data_varNames_logical], as.logical)
```

Factor variables

Factors are misclassified as character variables, so we will convert them to factors. As the booleans are already converted to logical variables, we will convert the rest of the character variables to factors.

```
# Get the names of the character variables
data_varNames_factors <- names(data)[sapply(data, is.character)]
data_varNames_factors
```

```
## [1] "District"          "City"              "Town"
## [4] "Type"              "EnergyCertificate"
```

```
# Convert character variables to factors
data[data_varNames_factors] <- lapply(data[data_varNames_factors], as.factor)

# Get chars of no imputed
data_varNames_factors <- names(no_imputed_data)[sapply(no_imputed_data, is.character)]
no_imputed_data[data_varNames_factors] <- lapply(data[data_varNames_factors], as.factor)
```

Standardization of data

Numerical Values

```
# List of numeric columns to scale
data_varNames_numerics <- names(data)[sapply(data, is.numeric)]
data_varNames_numerics
```

```
##  [1] "Price"              "Floor"              "ConstructionYear"
##  [4] "TotalArea"          "GrossArea"          "PublishDate"
##  [7] "NumberOfBedrooms"   "NumberOfWC"         "ConservationStatus"
## [10] "LivingArea"         "LotSize"            "BuiltArea"
## [13] "NumberOfBathrooms"  "latitude"           "longitude"
## [16] "OtherRooms"
```

```
# scale each column of data cleaned
for (col in data_varNames_numerics) {
  norm_col_name <- paste0(col, "Scaled")
  data[[norm_col_name]] <- scale(data[[col]], center = TRUE, scale = TRUE)
```

```
}

# scale each column of not imputed
data_varNames_numerics <- names(no_imputed_data)[sapply(no_imputed_data, is.numeric)]

denormalization_values <- list()

for (col in data_varNames_numerics) {

  # Columns without nas
  column_values <- no_imputed_data[[col]][!is.na(no_imputed_data[[col]])]

  #Save the denormalization values
  denormalization_values[[col]] <- list(
    mean = mean(column_values),
    sd = sd(column_values)
  )


  no_imputed_data[[col]] <- scale(no_imputed_data[[col]], center = TRUE, scale = TRUE)
}

data$PublishDate <- as.Date(data$PublishDate, format = "%Y-%m-%d")
```

PART 2: Context of the problem, models and evaluation.

Q1: First Research Question: Can we predict the year of construction based on the other variables?

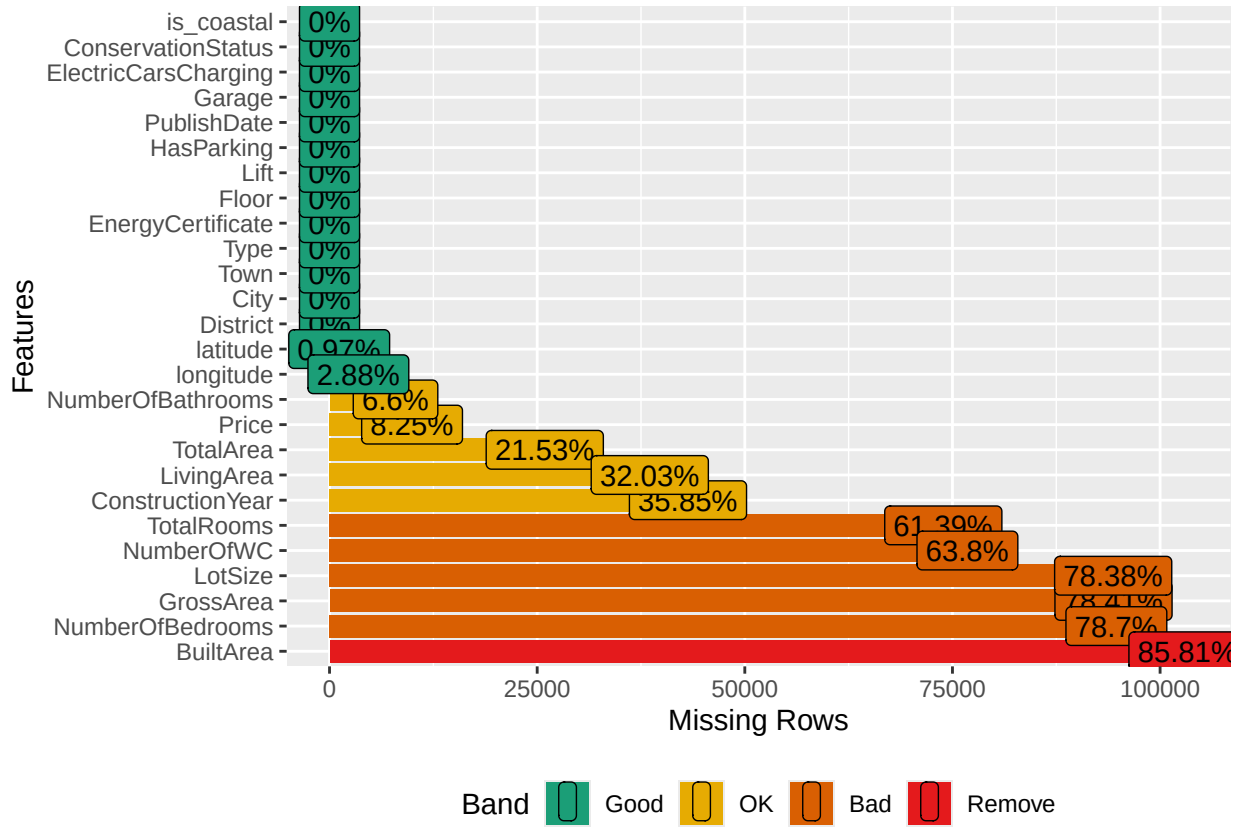
In this section, we aim to determine whether it is possible to predict the year of construction of a property using the other variables available in the dataset. This involves analyzing the relationships between the construction year and other features such as price, location, type, and various property characteristics. By building predictive models, we will evaluate the accuracy and reliability of these predictions. The steps include data analysis, preprocessing specific to this problem, model training, and evaluation of results.

Data Analysis

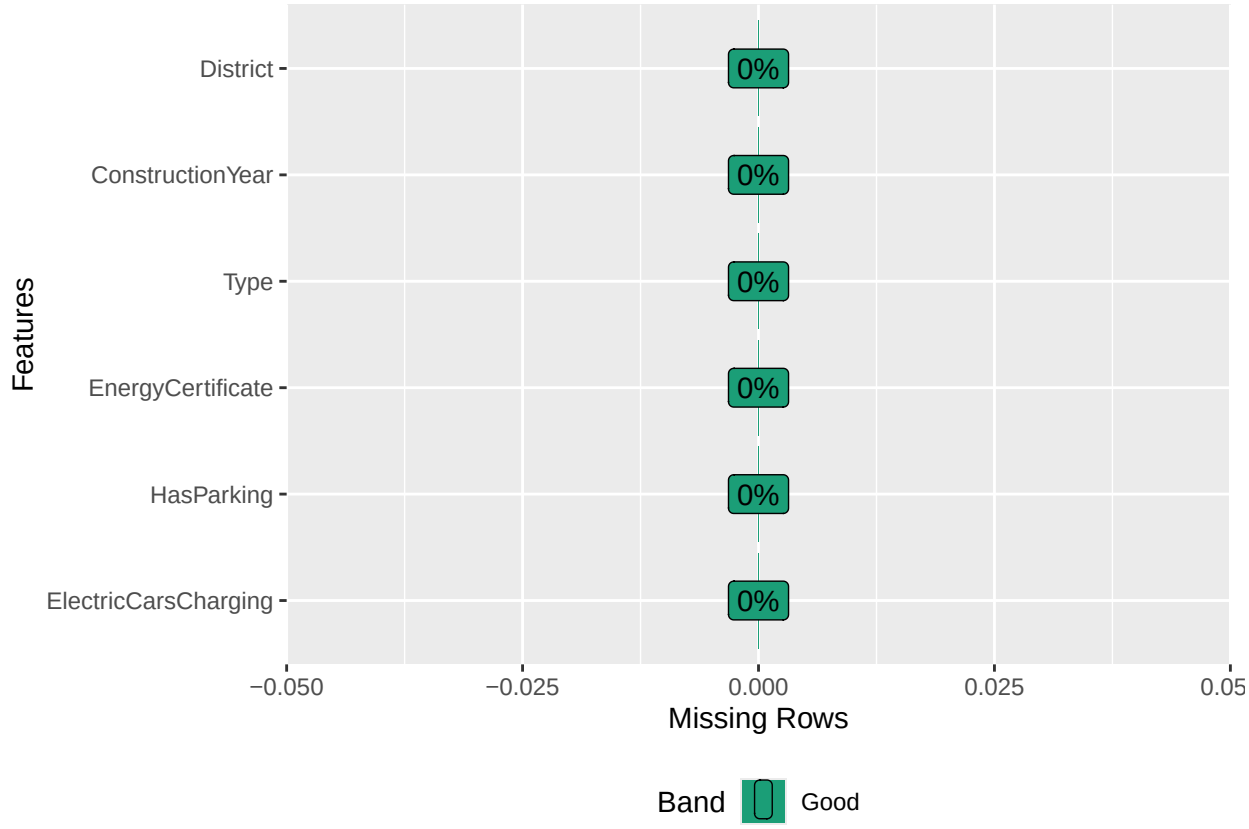
First of all, we will analyse the relationships between the construction year and other variables in the dataset. We will use visualizations such as scatter plots, histograms, and correlation matrices to identify patterns and correlations between the variables.

Transforming the main dataset At this point, we can try to make a clean dataset to work with, we will make a subset with clean real data, analyze it and compare it with another dataset with invented data and see which one has better results.

```
# Check the number of missing values in the dataset_Cleaned
plot_missing(no_imputed_data)
```




```r
columns <- c("ElectricCarsCharging", "HasParking", "EnergyCertificate", "Type", "ConstructionYear", "Dis
# We make a subset of the data with the columns we want to analyze
subset_data <- no_imputed_data[, columns]

# Remove Nas
subset_data <- subset_data[complete.cases(subset_data), ]
plot_missing(subset_data)
```



One-hot-encode: another perspective

After some visualizations of the data, we assured which data was relevant for the prediction wanted to be made. To achieve this, we thought of doing a correlation matrix. Main question is; knowing that there's a lot of attributes on the sample and, for some factor type attributes having more than 50 different categories, how are we supposed to measure the explanation of `ConstructionYear`?

Answer: One-hot-encode/dummify the dataset.

```
# Dummy encoding of categorical variables without town, city and district
dummy <- dummyVars(~., data = subset_data)
one_hot_encoded_data <- data.frame(predict(dummy, newdata = subset_data))

# Transform all columns to numeric
one_hot_encoded_data <- data.frame(lapply(one_hot_encoded_data, as.numeric))

# Get the correlation matrix of factors variables
correlation_matrix <- cor(one_hot_encoded_data)
```

```
## Warning in cor(one_hot_encoded_data): La desviación estándar es cero
```

The insight of this correlation matrix not only tell us which variables explain `ConstructionYear`, but also which attributes take into account for the predict purpose. Doing so, we applied a threshold of `0,05` to the correlations between value and attribute `ConstructionYear` to fetch the less relevant values of attributes.

```
# Crear el data frame
data_cor <- data.frame(correlation_matrix)
```

```r
# Definir el valor alpha
alpha <- 0.05

# Extraer nombres de las filas y valores que cumplen la condición para one_hot_to_unify
one_hot_to_unify <- data_cor$ConstructionYear[abs(data_cor$ConstructionYear) < abs(alpha)]
one_hot_to_unify_names <- rownames(data_cor)[abs(data_cor$ConstructionYear) < abs(alpha)]

# Extraer nombres de las filas y valores que cumplen la condición para one_hot_suitable
one_hot_suitable <- data_cor$ConstructionYear[abs(data_cor$ConstructionYear) >= abs(alpha)]
one_hot_suitable_names <- rownames(data_cor)[abs(data_cor$ConstructionYear) >= abs(alpha)]

# Create dataframe with her names
one_hot_suitable_names_df <- data.frame(one_hot_suitable_names)
# Add the values
one_hot_suitable_names_df$Values <- one_hot_suitable

# Create dataframe with her names
one_hot_to_unify_names_df <- data.frame(one_hot_to_unify_names)
one_hot_to_unify_names_df$Values <- one_hot_to_unify

# Names found inside of the unify hot_encodes
types_to_unify <- c("Duplex", "Estate", "Garage", "Hotel", "Industrial", "Investment", "Land", "Mansion
district_to_unify <- c("Aveiro", "Bragança", "Évora", "Ilha de Madeira", "Ilha de Porto Santo", "Ilha d
energycert_to_unify <- c("D", "G", "No Certificate")
```

Data preprocessing

Once filtered which values give important relevance to `ConstructionYear`, we discard those samples that will not give any possible information about the `ConstructionYear`, hence, the ones that contain 'noise' for predictions.

```r
# Removal of samples that give 'noise' to the study
subset_data <- subset_data[!(subset_data$Type %in% types_to_unify), ]
subset_data <- subset_data[!(subset_data$District %in% district_to_unify), ]
subset_data <- subset_data[!(subset_data$EnergyCertificate %in% energycert_to_unify), ]
```

Imputation of missing values

There are lots of forms to impute missing values, we will use a Random Forest to predict them and input them into the dataset.missing It could also be done with KNN, MICE, etc, but in this case we will use Random Forest due to its simplicity, explicability and robustness. We won't use City and Town as there are too many factors for the model to predict.

```r
imputed_subset_data <- no_imputed_data

# Obtenemos el nombre de las columnas que mas nas tienen de menor a mayor
nas <- sort(apply(imputed_subset_data, 2, function(x) sum(is.na(x))), decreasing = FALSE)
# Remove Os
nas <- names(nas[nas > 0])
nas
```

```
##  [1] "latitude"          "longitude"         "NumberOfBathrooms"
##  [4] "Price"             "TotalArea"         "LivingArea"
##  [7] "ConstructionYear"  "TotalRooms"        "NumberOfWC"
## [10] "LotSize"           "GrossArea"         "NumberOfBedrooms"
```

```
## [13] "BuiltArea"
```

```
# Remove from Nas the ConstructionYear
nas <- nas[nas != "ConstructionYear"]

form_base <- "District + Type + EnergyCertificate + HasParking"

for (col in nas) {
  form <- as.formula(paste0(col, " ~ ", form_base))
  print(col)

  # If not numeric, to numeric
  if (!is.numeric(imputed_subset_data[[col]])) {
    imputed_subset_data[[col]] <- as.numeric(as.factor(imputed_subset_data[[col]]))
  }
  print(form)

  random_forest <- getRandomForest(formula = form, dataset = imputed_subset_data, col, nrOfSamples = ra

  # Predict all nas
  na_rows <- is.na(imputed_subset_data[[col]])
  imputed_subset_data[[col]][na_rows] <- predict(random_forest, imputed_subset_data[na_rows, ])

  # Add to form_Base
  form_base <- paste0(form_base, " + ", col)
}
```

```
## [1] "latitude"
## latitude ~ District + Type + EnergyCertificate + HasParking
## [1] "longitude"
## longitude ~ District + Type + EnergyCertificate + HasParking +
##     latitude
## [1] "NumberOfBathrooms"
## NumberOfBathrooms ~ District + Type + EnergyCertificate + HasParking +
##     latitude + longitude
## [1] "Price"
## Price ~ District + Type + EnergyCertificate + HasParking + latitude +
##     longitude + NumberOfBathrooms
## [1] "TotalArea"
## TotalArea ~ District + Type + EnergyCertificate + HasParking +
##     latitude + longitude + NumberOfBathrooms + Price
## [1] "LivingArea"
## LivingArea ~ District + Type + EnergyCertificate + HasParking +
##     latitude + longitude + NumberOfBathrooms + Price + TotalArea
## [1] "TotalRooms"
## TotalRooms ~ District + Type + EnergyCertificate + HasParking +
##     latitude + longitude + NumberOfBathrooms + Price + TotalArea +
##     LivingArea
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

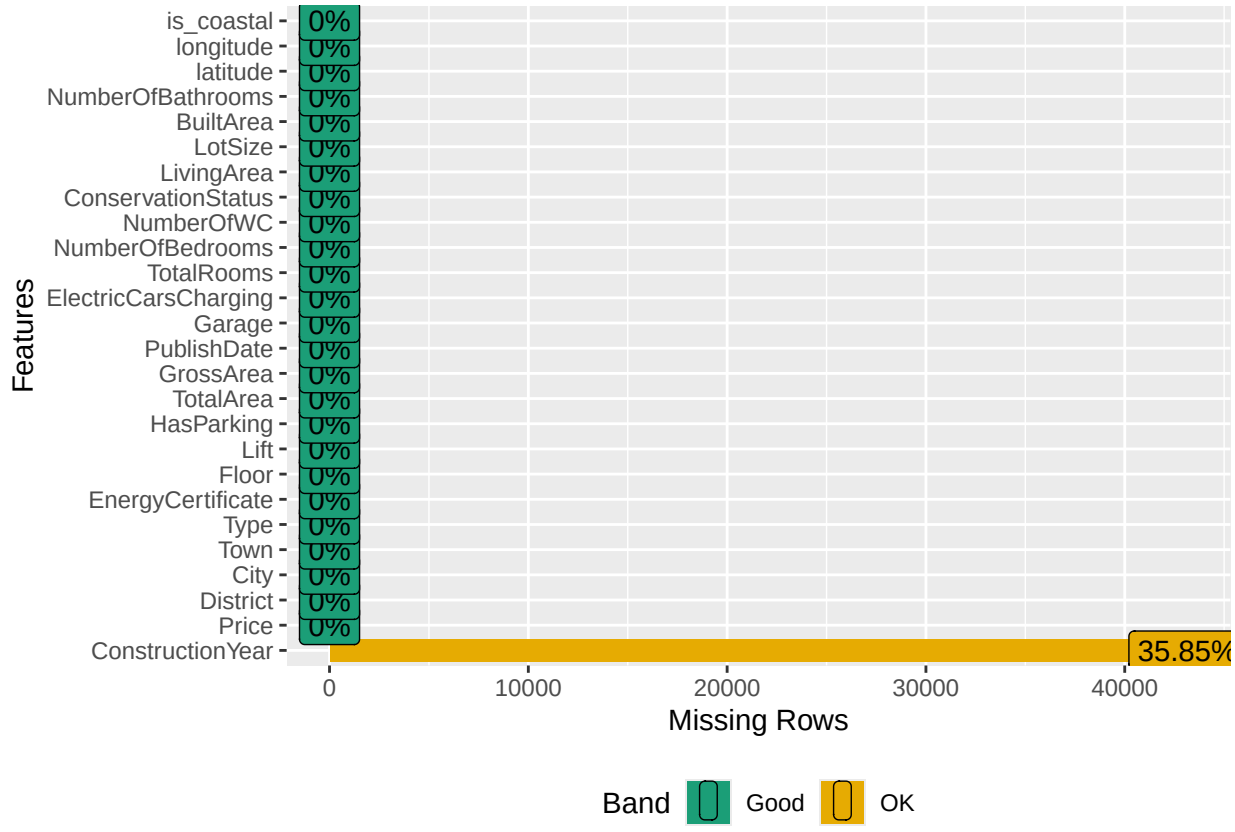
```
## [1] "NumberOfWC"
## NumberOfWC ~ District + Type + EnergyCertificate + HasParking +
##     latitude + longitude + NumberOfBathrooms + Price + TotalArea +
##     LivingArea + TotalRooms
```

```
## [1] "LotSize"
## LotSize ~ District + Type + EnergyCertificate + HasParking +
##     latitude + longitude + NumberOfBathrooms + Price + TotalArea +
##     LivingArea + TotalRooms + NumberOfWC
## [1] "GrossArea"
## GrossArea ~ District + Type + EnergyCertificate + HasParking +
##     latitude + longitude + NumberOfBathrooms + Price + TotalArea +
##     LivingArea + TotalRooms + NumberOfWC + LotSize
## [1] "NumberOfBedrooms"
## NumberOfBedrooms ~ District + Type + EnergyCertificate + HasParking +
##     latitude + longitude + NumberOfBathrooms + Price + TotalArea +
##     LivingArea + TotalRooms + NumberOfWC + LotSize + GrossArea
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values.  Are you sure you want to do regression?
```

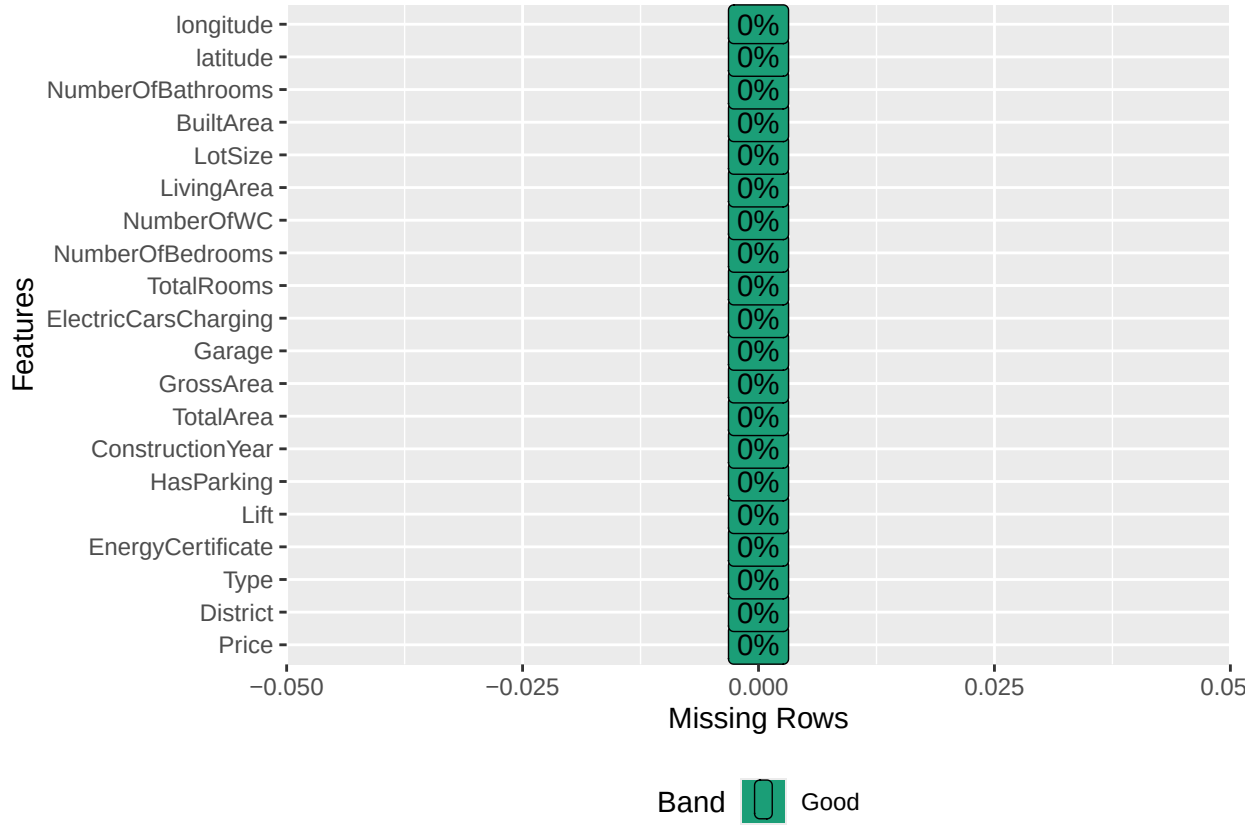
```
## [1] "BuiltArea"
## BuiltArea ~ District + Type + EnergyCertificate + HasParking +
##     latitude + longitude + NumberOfBathrooms + Price + TotalArea +
##     LivingArea + TotalRooms + NumberOfWC + LotSize + GrossArea +
##     NumberOfBedrooms
```

```
plot_missing(imputed_subset_data)
```




```
# Clean imputed data to only complete cases
imputed_subset_data <- imputed_subset_data[complete.cases(imputed_subset_data), ]

# Remove Columns with more than 50 factors
```

```r
imputed_subset_data <- imputed_subset_data[, sapply(imputed_subset_data, function(x) {
  is.factor(x) && length(unique(x)) < 50 | is.numeric(x)
})]

plot_missing(imputed_subset_data)
```



Preprocessing the imputed dataset

We will preprocess the imputed dataset in the same way as the previous dataset worked on to ensure that it is clean and ready for analysis. We will address issues such as duplicated values, negative values, and other inconsistencies in the dataset.

Applying same preprocessing to the imputed dataset

Just as the other dataset, we're going to eliminate the samples that do not give a good correlation with `ConstructionYear` variable as seen at the one-hot-encode part. It is necessary to make a good comparison between them

```r
# Removal
imputed_subset_data <- imputed_subset_data[!(imputed_subset_data$Type %in% types_to_unify), ]
imputed_subset_data <- imputed_subset_data[!(imputed_subset_data$District %in% district_to_unify), ]
imputed_subset_data <- imputed_subset_data[!(imputed_subset_data$EnergyCertificate %in% energycert_to_u
```

Predictions

We will train different machine learning models to predict the construction year based on the other variables in the dataset. First of all, we are going to split the data in train-test sets, then we will train the models and

make predictions on the test data. Finally, we will evaluate the performance of the models using the mean squared error (MSE) as the evaluation metric. We will use one model of each type: Hierar (Random Forest), Statistical (Linear Regression) and Clustering (KNN).

```
train_index <- sample(1:nrow(subset_data), 0.8 * nrow(subset_data))
train_data <- subset_data[train_index, ]
test_data <- subset_data[-train_index, ]

# Asegurarse de que los niveles de los factores en test_data coincidan con los de train_data
for (col in names(train_data)) {
  if (is.factor(train_data[[col]])) {
    levels(test_data[[col]]) <- levels(train_data[[col]])
  }
}

head(train_data)
```

```
## # A tibble: 6 x 6
##   ElectricCarsCharging HasParking EnergyCertificate Type    ConstructionYear[,1]
##   <fct>                <fct>      <fct>             <fct>                  <dbl>
## 1 FALSE                TRUE       NC                House                -1.50
## 2 FALSE                TRUE       A                 House                 1.32
## 3 FALSE                FALSE      NC                Apartm~              -0.0136
## 4 FALSE                TRUE       A                 Apartm~               1.29
## 5 FALSE                TRUE       A                 Apartm~               1.29
## 6 FALSE                FALSE      C                 Other                 0.980
## # i 1 more variable: District <fct>
```

```
head(test_data)
```

```
## # A tibble: 6 x 6
##   ElectricCarsCharging HasParking EnergyCertificate Type    ConstructionYear[,1]
##   <fct>                <fct>      <fct>             <fct>                  <dbl>
## 1 FALSE                FALSE      C                 Apartm~               -0.166
## 2 FALSE                TRUE       NC                Other                 -0.281
## 3 FALSE                FALSE      C                 Apartm~                0.369
## 4 FALSE                TRUE       B-                House                  0.445
## 5 FALSE                TRUE       C                 Apartm~                0.521
## 6 FALSE                TRUE       C                 Apartm~                0.177
## # i 1 more variable: District <fct>
```

```
imputated_train_index <- sample(1:nrow(imputed_subset_data), 0.8 * nrow(imputed_subset_data))
imputated_train_data <- imputed_subset_data[imputated_train_index, ]
imputated_test_data <- imputed_subset_data[-imputated_train_index, ]

# Asegurarse de que los niveles de los factores en test_data coincidan con los de train_data
for (col in names(imputated_train_data)) {
  if (is.factor(imputated_train_data[[col]])) {
    levels(imputated_test_data[[col]]) <- levels(imputated_train_data[[col]])
  }
}

head(imputated_train_data)
```

```
## # A tibble: 6 x 20
##   Price[,1] District       Type      EnergyCertificate Lift  HasParking
```

```
##       <dbl> <fct>          <fct>     <fct>             <fct> <fct>
## 1    1.55   Beja           House     C                 FALSE FALSE
## 2   -0.808  Portalegre     Apartment E                 FALSE FALSE
## 3   -1.15   Castelo Branco Other     NC                FALSE FALSE
## 4    1.73   Setúbal        House     A+                FALSE FALSE
## 5   -0.0794 Porto          House     E                 FALSE FALSE
## 6    1.00   Porto          Apartment B                 TRUE  TRUE
## # i 14 more variables: ConstructionYear <dbl[,1]>, TotalArea <dbl[,1]>,
## #   GrossArea <dbl[,1]>, Garage <fct>, ElectricCarsCharging <fct>,
## #   TotalRooms <dbl[,1]>, NumberOfBedrooms <dbl[,1]>, NumberOfWC <dbl[,1]>,
## #   LivingArea <dbl[,1]>, LotSize <dbl[,1]>, BuiltArea <dbl[,1]>,
## #   NumberOfBathrooms <dbl[,1]>, latitude <dbl[,1]>, longitude <dbl[,1]>
```

```
head(imputated_test_data)
```

```
## # A tibble: 6 x 20
##   Price[,1] District Type      EnergyCertificate Lift  HasParking
##       <dbl> <fct>    <fct>     <fct>             <fct> <fct>
## 1    -1.06  Faro     Other     NC                TRUE  TRUE
## 2     0.624 Faro     Apartment C                 TRUE  TRUE
## 3    -0.700 Faro     House     NC                FALSE FALSE
## 4     1.83  Faro     Apartment A                 TRUE  TRUE
## 5    -0.155 Faro     Other     C                 FALSE FALSE
## 6     0.624 Faro     Apartment A                 TRUE  TRUE
## # i 14 more variables: ConstructionYear <dbl[,1]>, TotalArea <dbl[,1]>,
## #   GrossArea <dbl[,1]>, Garage <fct>, ElectricCarsCharging <fct>,
## #   TotalRooms <dbl[,1]>, NumberOfBedrooms <dbl[,1]>, NumberOfWC <dbl[,1]>,
## #   LivingArea <dbl[,1]>, LotSize <dbl[,1]>, BuiltArea <dbl[,1]>,
## #   NumberOfBathrooms <dbl[,1]>, latitude <dbl[,1]>, longitude <dbl[,1]>
```

Random Forest Random forest do create N decision trees and then does a majority vote to predict the value. We have use it as Hierarquical model due to his simplicity and robustness.

Clean dataset We proceed to train the model with the clean dataset. It only contains real data without NAs.

```
# Forma
form <- ConstructionYear ~ .
# Train the random forest model
random_forest_model <- randomForest(form, data = train_data, ntree = nTrees)

# Make predictions on the test data
predictions <- predict(random_forest_model, test_data)

# Calculate the mean squared error
mse <- mean((test_data$ConstructionYear - predictions)^2)
mse
```

```
## [1] 0.6795251
```

```
# Denormalize mse
mse <- mse * (denormalization_values[["ConstructionYear"]]$sd) ^2
mse
```

```
## [1] 465.4102
```

Imputed dataset Now, we will train the model with the imputed dataset. It contains invented data to fill the NAs. We will compare the results with the clean dataset.

```
# Train the random forest model
random_forest_model <- randomForest(form, data = imputated_train_data, ntree = nTrees)

# Make predictions on the test data
predictions <- predict(random_forest_model, imputated_test_data)

# Calculate the mean squared error
mse <- mean((imputated_test_data$ConstructionYear - predictions)^2)
mse
```

```
## [1] 0.3184341
```

```
# Denormalize mse
mse <- mse * (denormalization_values[["ConstructionYear"]]$sd) ^2
mse
```

```
## [1] 218.0971
```

The invented data seems to have a better performance than the real data, this could be due to the fact that the invented data allows the model to have more information, real or not, to make the predictions. Maybe the fake data is not as good as the real data, but it allows the model to intercalate the real data with the fake data to make better predictions.

Linear Regression Linear regression is a statistical model that assumes a linear relationship between the input variables and the output variable. We will use it as a statistical model to predict the construction year based on the other variables in the dataset.

Clean dataset As we did with the Random Forest, we will train the model with the clean dataset and compare the results with the imputed dataset.

```
# Training the linear regression model, remove Town City District from data to train
linear_regression_model <- lm(form, data = train_data)

# Make predictions on the test data
predictions <- predict(linear_regression_model, test_data)

# Calculate the mean squared error
mse <- mean((test_data$ConstructionYear - predictions)^2)
mse
```

```
## [1] 0.6588849
```

```
# Denormalize mse
mse <- mse * (denormalization_values[["ConstructionYear"]]$sd) ^2
mse
```

```
## [1] 451.2736
```

Imputed dataset Now, we will train the model with the imputed dataset.

```
# Training the linear regression model, remove Town City District from data to train
linear_regression_model <- lm(form, data = imputated_train_data)

# Make predictions on the test data
predictions <- predict(linear_regression_model, imputated_test_data)
```

```r
# Calculate the mean squared error
mse <- mean((imputated_test_data$ConstructionYear - predictions)^2)
mse
```

```
## [1] 0.5959735
```

```r
# Denormalize mse
mse <- mse * (denormalization_values[["ConstructionYear"]]$sd) ^2
mse
```

```
## [1] 408.1852
```

As the Random Forest, the invented data seems to have a better performance than the real data, for the same reason as before.

K-Nearest Neighbors Finally we will use the K-Nearest Neighbors (KNN) algorithm to predict the construction year based on the other variables in the dataset. KNN is a clustering algorithm that assigns a new data point to the cluster of the nearest neighbors. We will use it as a clustering model to predict the construction year.

Clean dataset As we did before, we will train the model with the clean dataset and compare the results with the imputed dataset.

```r
# Transform all to numeric values
knn_train_data <- data.frame(lapply(train_data, as.numeric))
knn_test_data <- data.frame(lapply(test_data, as.numeric))

head(knn_train_data)
```

```
##   ElectricCarsCharging HasParking EnergyCertificate Type ConstructionYear
## 1                    1          2                10    2      -1.50377985
## 2                    1          2                 1    2       1.32381093
## 3                    1          1                10    1      -0.01356308
## 4                    1          2                 1    1       1.28560025
## 5                    1          2                 1    1       1.28560025
## 6                    1          1                 5    4       0.97991476
##   District
## 1        6
## 2       19
## 3        8
## 4        8
## 5       19
## 6        8
```

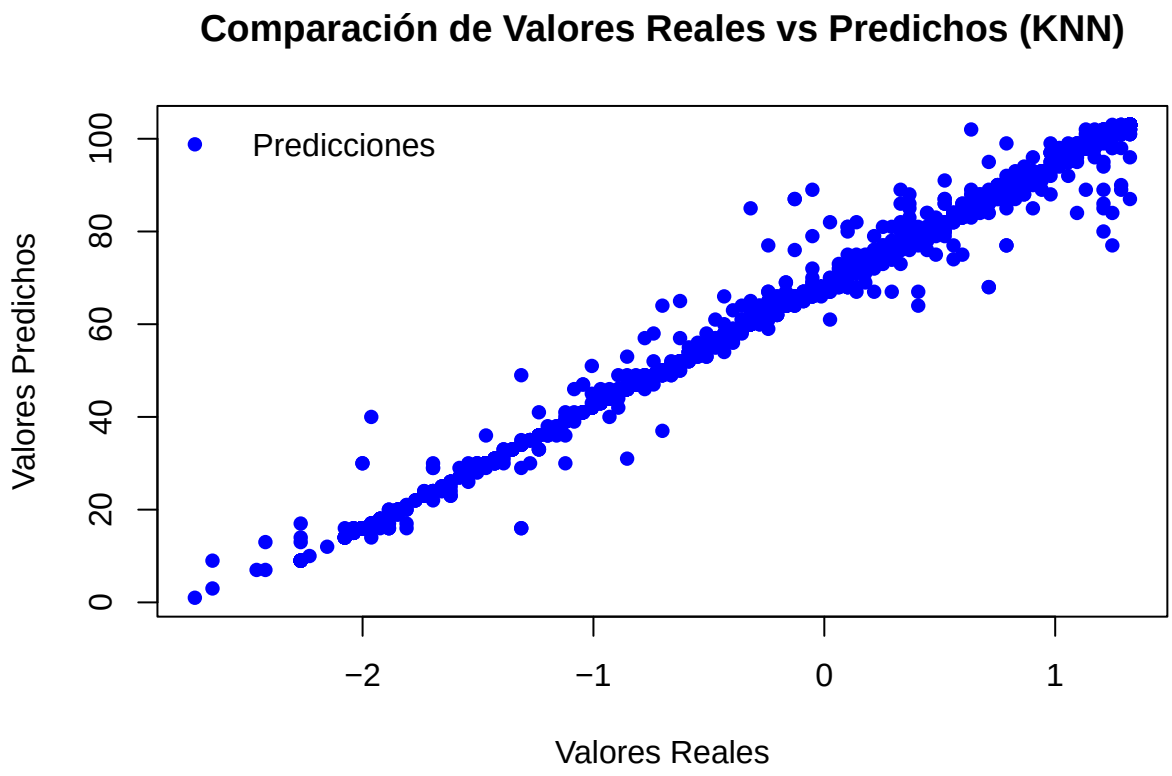
```r
# Train the KNN model
knn_model <- knn(knn_train_data, knn_test_data, train_data$ConstructionYear, k = 1)

# Calculate the mean squared error
mse <- mean((test_data$ConstructionYear - as.numeric(as.character(knn_model)))^2)
mse
```

```
## [1] 0.003088203
```

```r
# Denormalize mse
mse <- mse * (denormalization_values[["ConstructionYear"]]$sd) ^2
mse
```

```
## [1] 2.115126
```

```r
# Plot construction year vs. predicted construction year

# Crear el plot
plot(knn_test_data$ConstructionYear, knn_model,
  xlab = "Valores Reales",
  ylab = "Valores Predichos",
  main = "Comparación de Valores Reales vs Predichos (KNN)",
  pch = 16, col = "blue"
)


# Agregar leyenda
legend("topleft",
  legend = c("Predicciones"),
  col = c("blue"), pch = c(16, NA), lty = c(NA, 1), bty = "n"
)
```



Imputed dataset Now, we will train the model with the imputed dataset.

```r
# Transform all to numeric values
knn_train_data <- data.frame(lapply(imputated_train_data, as.numeric))
knn_test_data <- data.frame(lapply(imputated_test_data, as.numeric))

head(knn_train_data)
```

```
##         Price District Type EnergyCertificate Lift HasParking ConstructionYear
```

```
## 1  1.55396355        2    2                 5    1          1      -0.39566995
## 2 -0.80809460       18    1                 7    1          1      -1.50377985
## 3 -1.14983918        5    4                10    1          1      -0.01356308
## 4  1.73164613       21    2                 2    1          1       1.28560025
## 5 -0.07937453       19    2                 7    1          1      -1.12167299
## 6  1.00114143       19    1                 3    2          2       1.28560025
##     TotalArea  GrossArea Garage ElectricCarsCharging TotalRooms
## 1 -0.59829597  0.5695493      1                    1  0.5507993
## 2 -0.19685146  0.1011033      1                    1  1.5380383
## 3 -0.09263029 -1.1110515      1                    1 -0.6003939
## 4 -0.06560999  0.7119797      2                    1  1.1778544
## 5  0.57901725 -1.0000687      1                    1 -0.6705500
## 6 -0.16597112  0.5449940      1                    1  0.4337441
##   NumberOfBedrooms  NumberOfWC LivingArea    LotSize   BuiltArea
## 1       0.71348573 -0.45602502  1.7692759 -0.6243155  0.25104650
## 2       0.51791458 -0.17629762  0.4072677 -0.5322174  0.06235813
## 3      -0.50796004 -0.42742304 -1.4747800  0.6512841 -0.73476471
## 4       1.50441741  0.68805434  0.8530159 -0.4534774  0.88557566
## 5      -1.45196340 -0.06703804 -0.9423586  0.1627706 -0.42389293
## 6       0.01675202  0.37972495  0.8034883 -0.1099556  0.26826684
##   NumberOfBathrooms    latitude   longitude
## 1         2.7211198 -1.48801268 -0.28177860
## 2        -0.2890923 -0.58787855  0.62814462
## 3        -1.0416454  0.09317117  2.07662610
## 4         1.9685668 -0.93259204 -0.77588693
## 5        -0.2890923  1.06687302 -0.17269296
## 6         0.4634607  1.00229426 -0.01730315
```

```r
# Train the KNN model
knn_model <- knn(knn_train_data, knn_test_data, imputated_train_data$ConstructionYear, k = 1)

# Calculate the mean squared error
mse <- mean((knn_test_data$ConstructionYear - as.numeric(as.character(knn_model)))^2)
mse
```

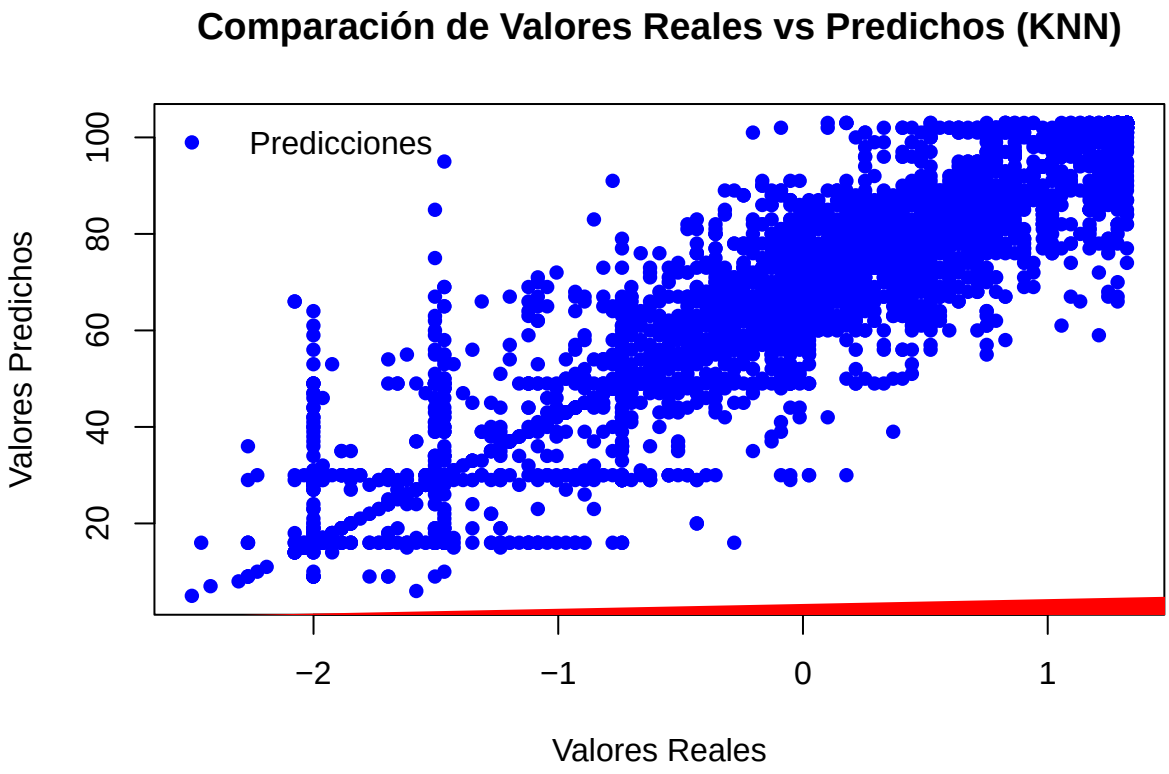
```
## [1] 0.09655254
```

```r
# Denormalize mse
mse <- mse * (denormalization_values[["ConstructionYear"]]$sd) ^2
mse
```

```
## [1] 66.12932
```

```r
# Crear el plot
plot(knn_test_data$ConstructionYear, knn_model,
  xlab = "Valores Reales",
  ylab = "Valores Predichos",
  main = "Comparación de Valores Reales vs Predichos (KNN)",
  pch = 16, col = "blue"
)

# Agregar la línea de identidad (y = x)
abline(a = 0, b = 1, col = "red", lwd = 16)

# Agregar leyenda
legend("topleft",
```

```
  legend = c("Predicciones"),
  col = c("blue"), pch = c(16, NA), lty = c(NA, 1), bty = "n"
)
```



The KNN model seems to have a lower performance when imputing the data, this could be due to the fact that the invented data is not as good as the real data, and the model is not able to make good predictions with the fake data. The fake data is distorting the model and making it less accurate.

Results and Conclusions

This study explored different approaches to predict `ConstructionYear` using various machine learning techniques. Through our analysis, we discovered several key findings:

Key Findings:

- An inverse relationship exists between KNN and other model performances, suggesting different strengths in prediction patterns
- Data quality improvements through imputation and feature engineering led to enhanced model performance
- One-hot encoding demonstrated significant positive impact on prediction accuracy
- The implementation of new variables contributed to more robust predictions
- More samples in the imputation dataset led to better prediction results except for KNN model

Best Performance:

The optimal results were achieved using KNN with k=1 on the cleaned dataset, followed closely by prediction models utilizing imputed data. This suggests that while data cleanliness is crucial, appropriate model selection

and parameter tuning play equally important roles in prediction accuracy.

Q2: Second Research Question: Is there a relationship between the year of construction, energy efficiency and conservation status?

For this, we will create a small subset of all the data, since most of ConservationStatus values are not available. Rather than using the imputed data set, we want to stay true to the actual data with this question.

Preparation

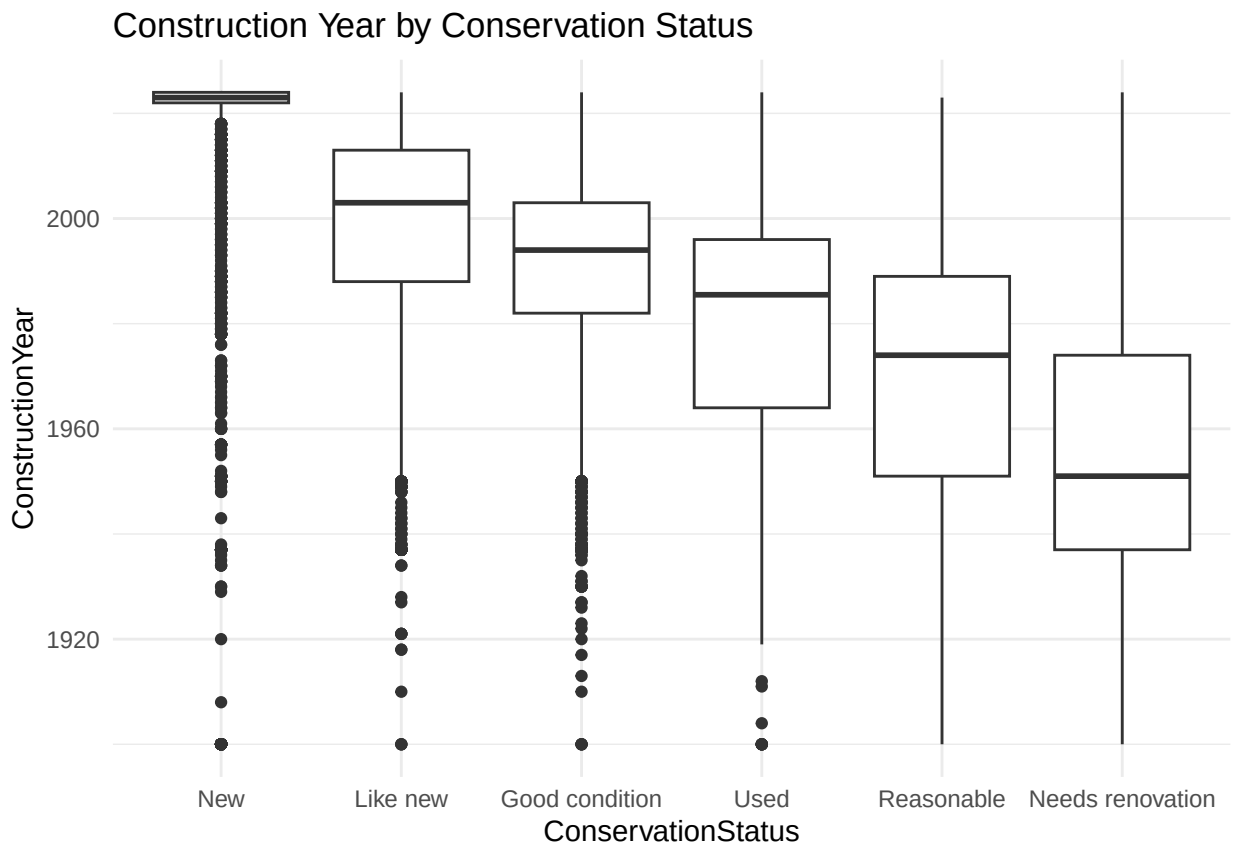
First we will filter the data set with the relevant variables regarding this question and will modify them properly.

```
# Data set filtering and transformations
question_2_data <- original_data[, c("EnergyCertificate", "ConstructionYear", "ConservationStatus")]

# Filter rows where there are no NAs and no empty strings
question_2_data <- question_2_data[apply(question_2_data, 1, function(row) all(row != "" & !is.na(row)))

# Update EnergyCertificate and apply transformations
question_2_data$EnergyCertificate <- factor(
  ifelse(question_2_data$EnergyCertificate %in% c("No Certificate", "Not available"), "NC", question_2_
  levels = c("A+", "A", "B", "B-", "C", "D", "E", "F", "G", "NC")
)

# Update ConservationStatus
question_2_data$ConservationStatus <- factor(
  question_2_data$ConservationStatus,
  levels = c("New", "Like new", "Good condition", "Used", "Reasonable", "Needs renovation")
)

# Transform ConstructionYear
question_2_data$ConstructionYear <- as.integer(question_2_data$ConstructionYear)
```

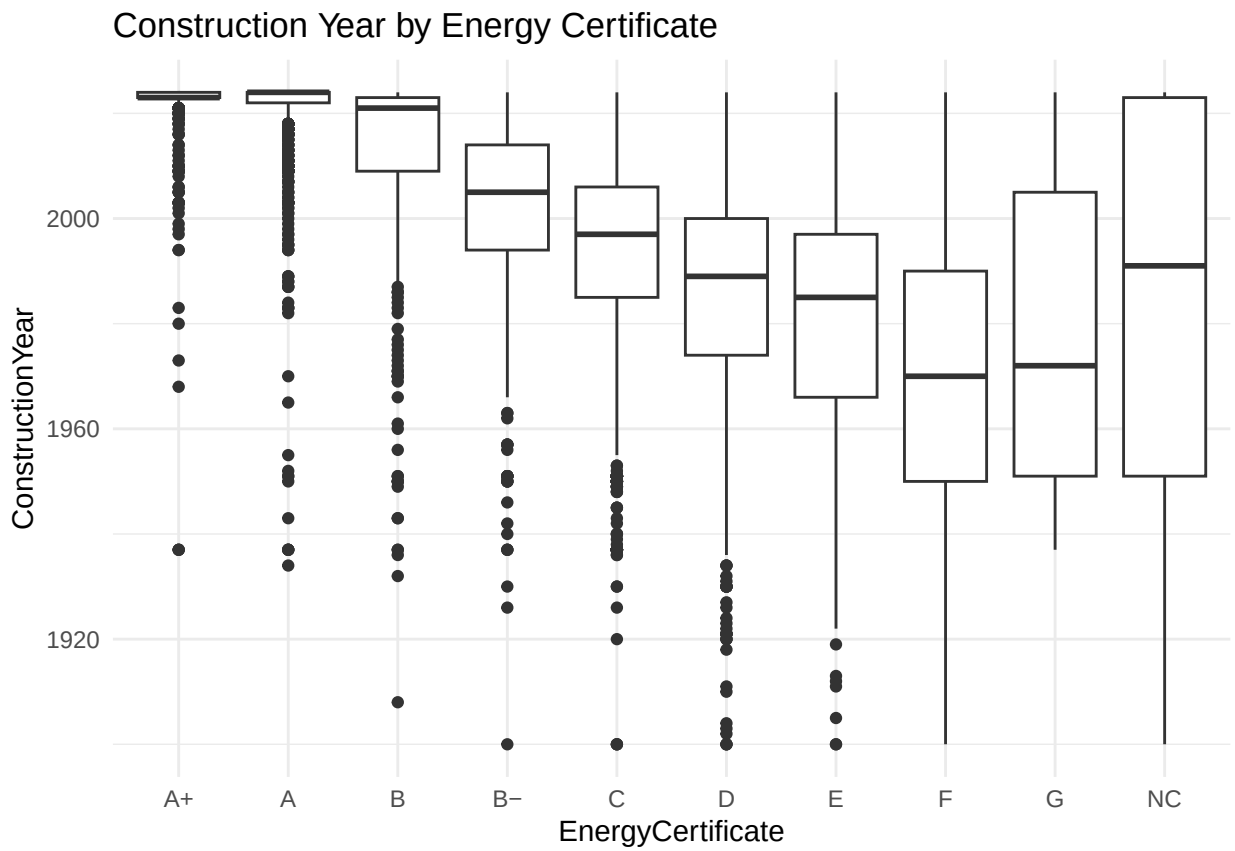
Plots

Lets take a fist look of the dataset with some plots

```
ggplot(question_2_data, aes(x = ConservationStatus, y = ConstructionYear)) +
  geom_boxplot() +
  theme_minimal() +
  labs(title = "Construction Year by Conservation Status",
       x = "ConservationStatus", y = "ConstructionYear")
```




```
ggplot(question_2_data, aes(x = EnergyCertificate, y = ConstructionYear)) +
  geom_boxplot() +
  theme_minimal() +
  labs(title = "Construction Year by Energy Certificate",
       x = "EnergyCertificate", y = "ConstructionYear")
```




```r
# Function to filter and calculate metrics
filter_and_calculate <- function(data, year_condition, condition_type = "lower", status) {
  if (condition_type == "lower") {
    filtered_data <- data[data$ConstructionYear <= year_condition, ]
  } else if (condition_type == "higher") {
    filtered_data <- data[data$ConstructionYear >= year_condition, ]
  } else {
    stop("Invalid condition_type. Use 'lower' or 'higher'.")
  }

  # Filter further for ConservationStatus == status
  filtered_data_new <- filtered_data[filtered_data$ConservationStatus == status, ]

  # Metrics
  total_rows <- nrow(filtered_data)
  new_rows <- nrow(filtered_data_new)
  proportion_status <- if (total_rows > 0) new_rows / total_rows else 0
  proportion_not_status <- if (total_rows > 0) 1 - proportion_status else 0

  # Print summary
  cat("\n--- Results for ConstructionYear",
      if (condition_type == "lower") paste("<= ", year_condition) else paste(">=", year_condition), "---
  cat("Total rows:", total_rows, "\n")
  cat("Rows with ConservationStatus =", status, ":", new_rows, "\n")
  cat("Proportion of", status, " :", round(proportion_status, 4), "\n")
  cat("Proportion of not", status, " :", round(proportion_not_status, 4), "\n")
```

```
}

# Apply function for ConstructionYear <= 1950
filter_and_calculate(question_2_data, year_condition = 1950, condition_type = "lower", status = "Needs
```

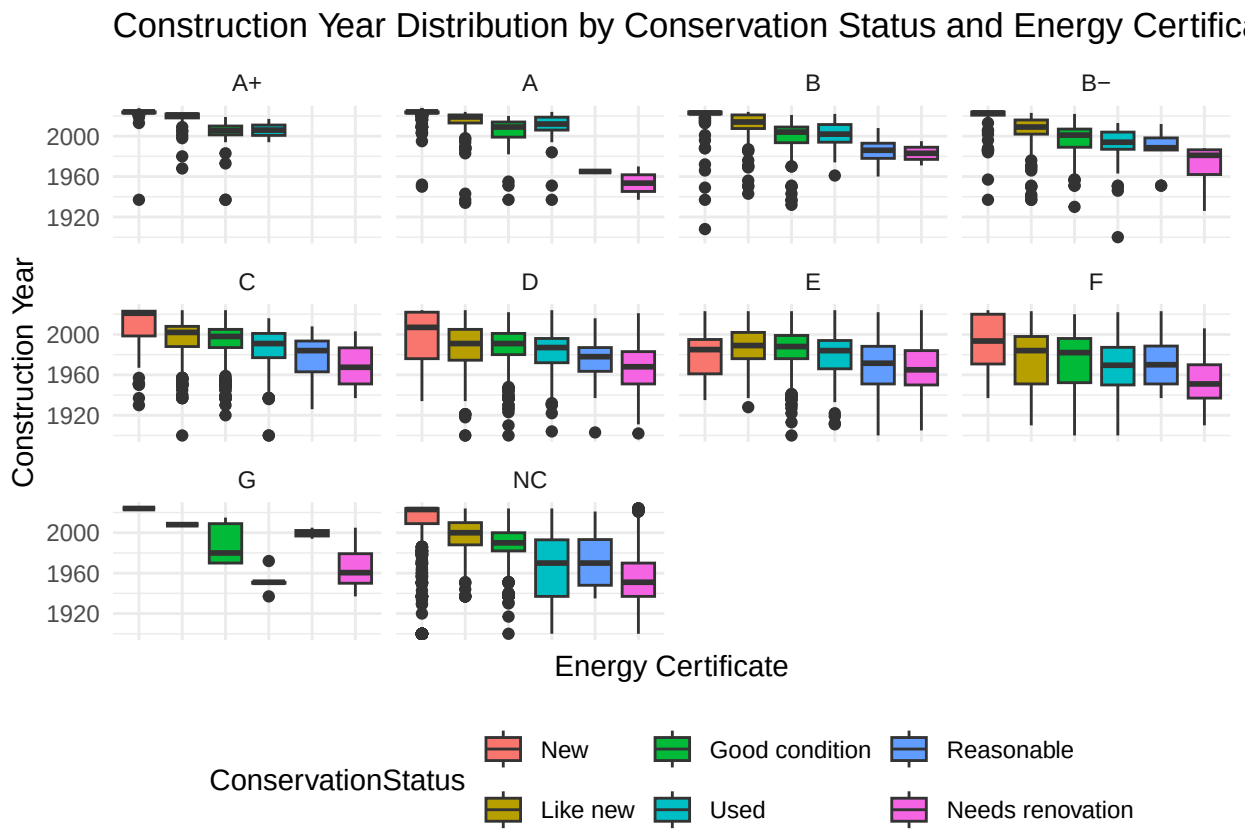
```
##
## --- Results for ConstructionYear <=  1950 ---
## Total rows: 1250
## Rows with ConservationStatus = Needs renovation : 501
## Proportion of Needs renovation  : 0.4008
## Proportion of not Needs renovation  : 0.5992
```

```
# Apply function for ConstructionYear >= 2010
filter_and_calculate(question_2_data, year_condition = 2010, condition_type = "higher", status = "New")
```

```
##
## --- Results for ConstructionYear >= 2010 ---
## Total rows: 3664
## Rows with ConservationStatus = New : 2792
## Proportion of New  : 0.762
## Proportion of not New  : 0.238
```

As we can see in the first plot, many properties from 1950 or before need a renovation (40%). Also, most properties from 2010 and forward have a status of New (76.2%). New properties seem to be all around the place and a lot of properties, regardless their year of construction or conservation status, don't have an energy certificate.

```
ggplot(question_2_data, aes(x = ConservationStatus, y = ConstructionYear, fill = ConservationStatus)) +
  geom_boxplot() +
  facet_wrap(~ EnergyCertificate) +
  theme_minimal() +
  labs(title = "Construction Year Distribution by Conservation Status and Energy Certificate",
       x = "Energy Certificate", y = "Construction Year") +
  theme(legend.position = "bottom",
        axis.text.x = element_blank())
```




```
filtered_data <- question_2_data[
  question_2_data$ConstructionYear <= 2000 &
  question_2_data$ConservationStatus == "New",
]

# Further filter rows where EnergyCertificate is "G"
filtered_data_new <- filtered_data[
  filtered_data$EnergyCertificate == "E" |
  filtered_data$EnergyCertificate == "F" |
  filtered_data$EnergyCertificate == "G" |
  filtered_data$EnergyCertificate == "NC",
  ]

# Metrics
total_rows <- nrow(filtered_data)
new_rows <- nrow(filtered_data_new)
proportion_status <- if (total_rows > 0) new_rows / total_rows else 0
proportion_not_status <- if (total_rows > 0) 1 - proportion_status else 0

# Print summary
  cat("\n--- Results for ConstructionYear <= 2000 and ConservationStatus = New ---\n")
```

```
##
## --- Results for ConstructionYear <= 2000 and ConservationStatus = New ---
```

```r
  cat("Total rows:", total_rows, "\n")
```

```
## Total rows: 388
```

```r
  cat("Rows with bad EnergyCertificate:", new_rows, "\n")
```

```
## Rows with bad EnergyCertificate: 316
```

```r
  cat("Proportion of bad:", round(proportion_status, 4), "\n")
```

```
## Proportion of bad: 0.8144
```

```r
  cat("Proportion of not bad:", round(proportion_not_status, 4), "\n")
```

```
## Proportion of not bad: 0.1856
```

In the second plot we see some properties from the 19th century that have a New Conservation Status but a bad energy certificate or none at all (81.4%) practically ensuring an aesthetic but not comprehensive renovation of the property.

Models

```r
numerical_q2_data <- question_2_data[, c("EnergyCertificate", "ConservationStatus", "ConstructionYear")]

energy_certificate_mapping <- c("NC" = 1, "G" = 2, "F" = 3, "E" = 4, "D" = 5,
                                "C" = 6, "B-" = 7, "B" = 8, "A" = 9, "A+"= 10)

numerical_q2_data$EnergyCertificate <- energy_certificate_mapping[as.character(numerical_q2_data$EnergyC

condition_mapping <- c(
  "Needs renovation" = 1,
  "Used" = 2,
  "Reasonable" = 3,
  "Good condition" = 4,
  "Like new" = 5,
  "New" = 6
)

numerical_q2_data$ConservationStatus <- condition_mapping[as.character(numerical_q2_data$ConservationSta
```

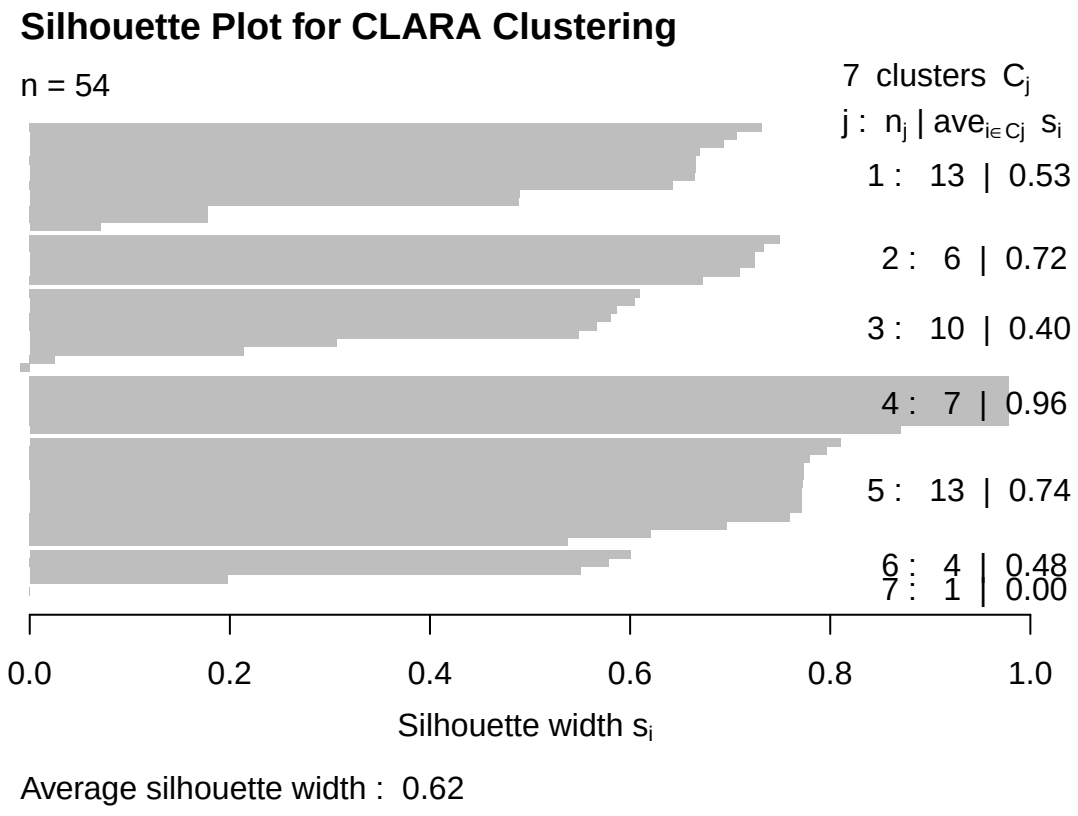
```r
partitioning_data = numerical_q2_data

k <- 7  # Number of clusters
clara_result <- clara(partitioning_data, k = k, metric = "euclidean", samples = 5)

# View results
print(clara_result)
```

Partitioning - K-medoids

```
## Call:     clara(x = partitioning_data, k = k, metric = "euclidean", samples = 5)
## Medoids:
##        EnergyCertificate ConservationStatus ConstructionYear
## 110864                 5                  4             2003
## 97801                  5                  2             1951
## 111150                 5                  4             1989
```

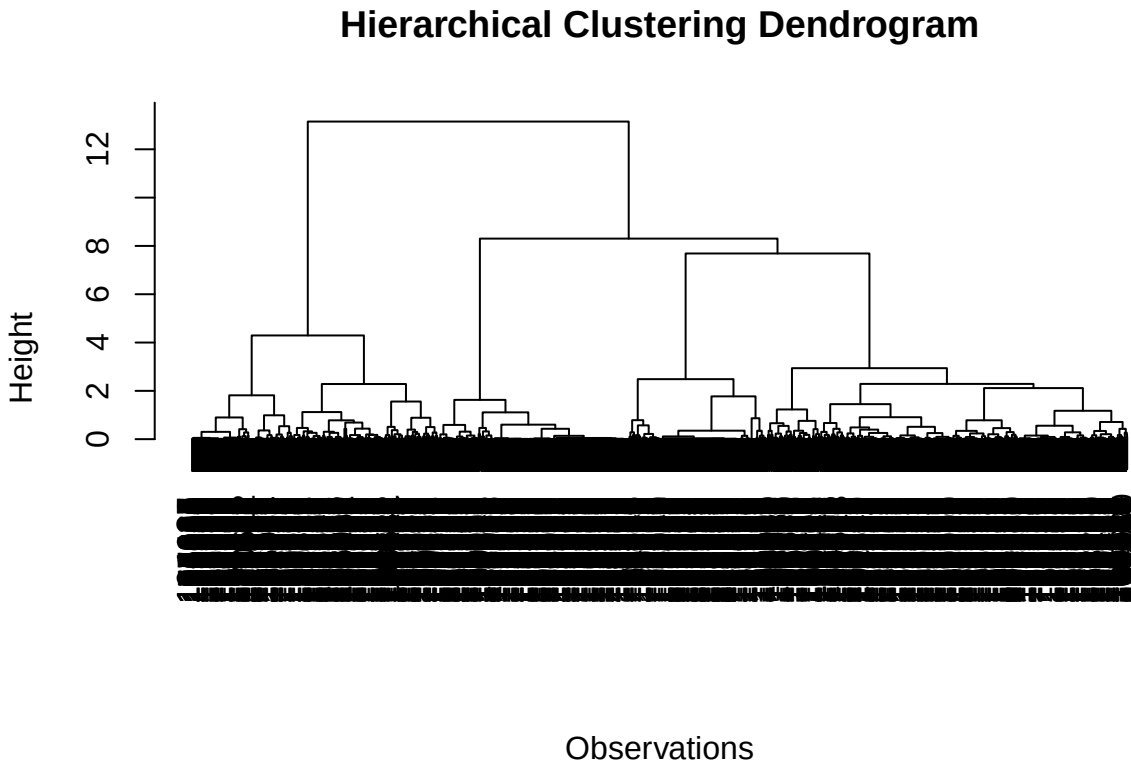
```
## 112536                 1                 6             2024
## 96317                  9                 6             2023
## 79339                  1                 1             1937
## 75607                  1                 6             1900
## Objective function:   4.636494
## Clustering vector:    Named int [1:11069] 1 2 1 3 2 1 4 1 3 4 5 2 3 5 3 5 5 2 ...
##  - attr(*, "names")= chr [1:11069] "68255" "68260" "68265" "68276" "68282" "68283" "68287" ...
## Cluster sizes:            2601 1428 2811 1024 2297 796 112
## Best sample:
##  [1] 68653  68846  69587  71094  71348  73206  75607  75625  75701  78356
## [11] 78990  79339  82918  83482  83879  84120  84142  84473  89638  89832
## [21] 90942  91155  93172  94006  95810  96081  96317  97801  98442  98624
## [31] 98909  98927  99858  100716 101938 102213 102478 102487 104460 104616
## [41] 107894 108927 109131 109347 109924 110067 110714 110864 111150 111363
## [51] 111686 112526 112536 114368
##
## Available components:
##  [1] "sample"     "medoids"    "i.med"      "clustering" "objective"
##  [6] "clusinfo"   "diss"       "call"       "silinfo"    "data"
```

```r
# Silhouette plot
plot(silhouette(clara_result), main = "Silhouette Plot for CLARA Clustering")
```



As we can see, with 7 clusters, the average silhouette width of 0.69 indicates a strong similarity to its cluster compared to the other clusters.

```r
# The data set is to large for the model, we will train it with a 25% of it
sample_data <- numerical_q2_data[sample(1:nrow(numerical_q2_data), size = 0.25 * nrow(numerical_q2_data)

# Ensure the numerical data is scaled
scaled_data <- scale(sample_data)

# Compute the Gower distance matrix for mixed data
distance_matrix <- daisy(sample_data, metric = "gower")

# Perform hierarchical clustering
hclust_result <- hclust(distance_matrix, method = "ward.D2")

# Plot the dendrogram
plot(hclust_result, main = "Hierarchical Clustering Dendrogram",
     xlab = "Observations", ylab = "Height", sub = "")
```



Hierarchical - divisive

As we can see in the model dendogram it's hard to get any valuable insights as it grows exponentially.

Conclusions

The analysis shows a strong relationship between construction year, energy efficiency, and conservation status. Older properties (pre-1950) mostly require renovation, while newer ones (post-2010) are often classified as "New" but frequently lack energy certificates. Surprisingly, some older properties with "New" conservation status still have poor energy ratings, suggesting superficial renovations. K-medoids clustering revealed clear groupings, while hierarchical methods struggled with data size. Overall, energy certification should be

prioritized to ensure transparency and efficiency in both renovations and new builds.

Q3: Third Research Question

Is there a relationship between the property's geographical location and its price? Are houses in coastal districts more expensive?

To study how the location affects the price we will create clusters without the price and location variables and then for every cluster treat it as a subdataset. For each cluster we will create a model that predicts the price based on the location.

Exploring the Relationship Between Geographical Location and Property Price

The geographical location of a property often plays a crucial role in determining its price. Coastal properties, in particular, tend to be associated with higher demand due to their proximity to scenic views and lifestyle benefits. To address the question, "Is there a relationship between the property's geographical location and its price? Are houses in coastal districts more expensive?", we devised a structured approach combining statistical tests, visualizations, and regression analysis.

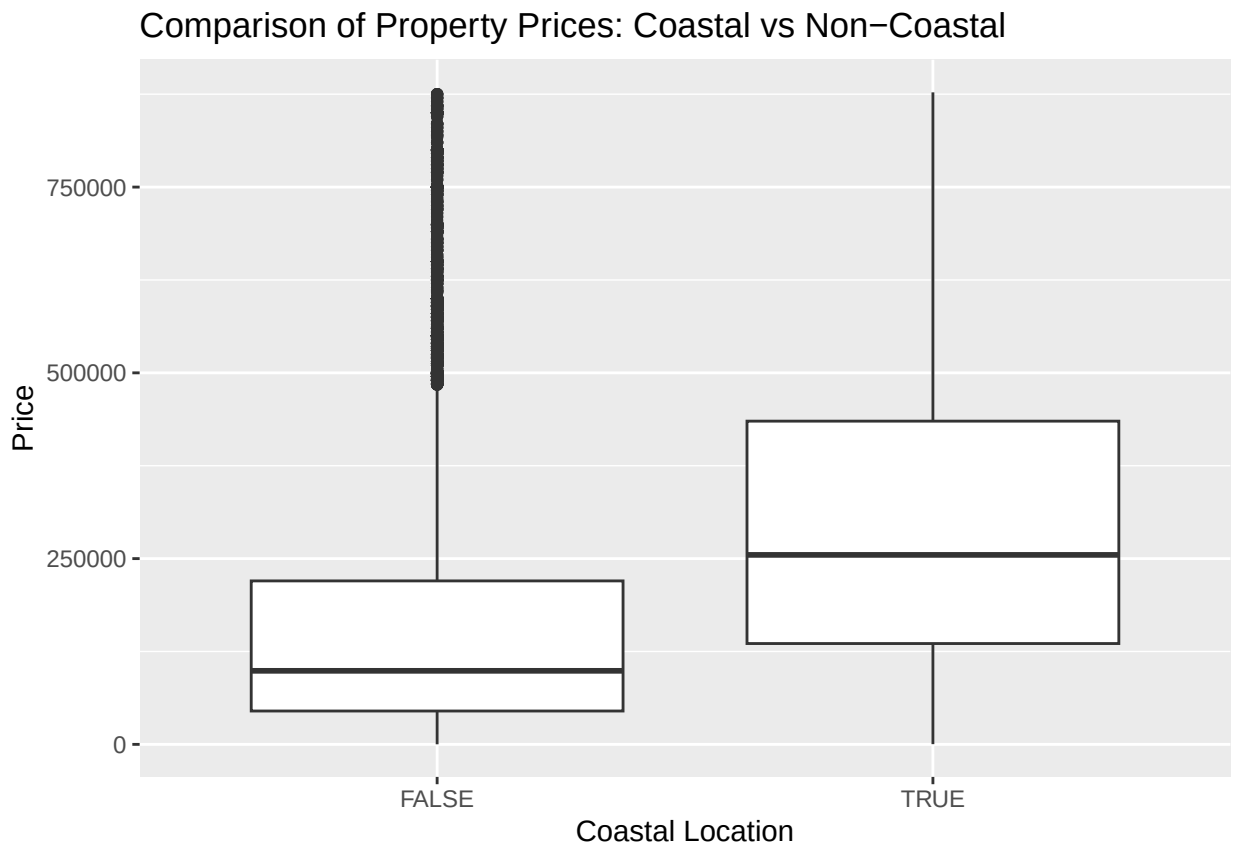
To answer this question, we will do the next:

Visual Exploration: A boxplot was used to visualize the distribution of prices for coastal and non-coastal properties, providing a clear representation of the differences. Regression Analysis: Finally, a linear regression model was built to quantify the effect of coastal location on property price while taking into account other factors such as total area, living area, and number of bathrooms. Statistical Testing: A Welch Two Sample t-test was conducted to compare the average property prices between coastal and non-coastal properties. This allows us to evaluate whether the observed differences in means are statistically significant.

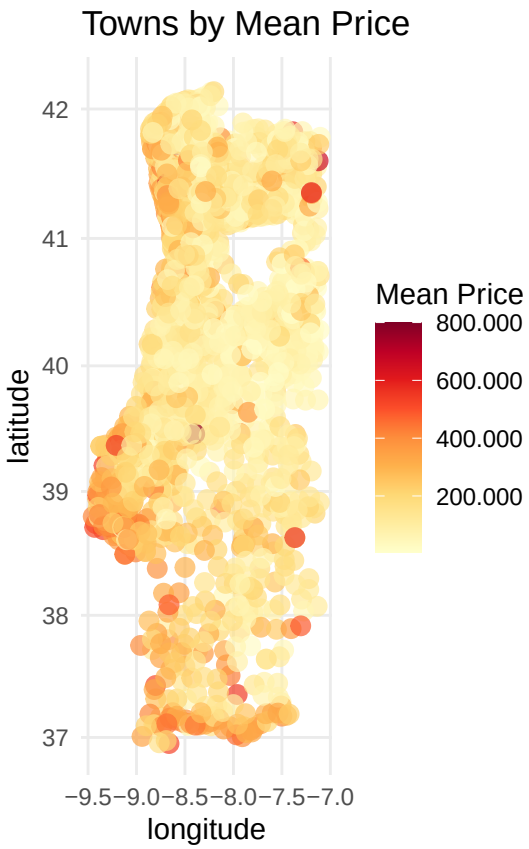
Visual Exploration: Boxplot and Scatterplot

While the t-test establishes statistical significance, a boxplot provides a clear visual representation of the price distributions for coastal and non-coastal properties. This allows us to better understand the spread of the data and identify any potential outliers.

```
# Visualizing price distributions
ggplot(data, aes(x = is_coastal, y = Price)) +
  geom_boxplot() +
  labs(title = "Comparison of Property Prices: Coastal vs Non-Coastal",
       x = "Coastal Location",
       y = "Price")
```




```r
# Ensure the 'data' dataframe has a column "Town" and "Price".
mean_prices <- data %>%
  group_by(Town) %>%
  summarize(
    mean_price = mean(Price, na.rm = TRUE),
    latitude = first(latitude),
    longitude = first(longitude)
  )

ggplot(mean_prices, aes(x = longitude, y = latitude, color = mean_price)) +
  geom_point(size = 3, alpha = 0.7) +
  scale_color_gradientn(
    colors = brewer.pal(9, "YlOrRd"),
    labels = function(x) format(x, big.mark = ".", decimal.mark = ",", scientific = FALSE)
  ) +
  labs(color = "Mean Price", title = "Towns by Mean Price") +
  theme_minimal() +
  coord_map()
```



From the boxplot:

Properties in coastal locations (TRUE) have a higher median price compared to non-coastal properties (FALSE). - The red points represent the mean prices, showing a clear elevation for coastal properties. - The presence of outliers in both groups indicates that some extreme price values exist, but the general trend supports the t-test results.

From the scatterplot:

We can observe that most of the expensive samples are in the coast.

Statistical Testing: Welch Two Sample t-test

The Welch Two Sample t-test was chosen as it does not assume equal variances between groups, making it more robust for comparing the mean prices of coastal (is_coastal = TRUE) and non-coastal (is_coastal = FALSE) properties.

```
# Comparing property prices between coastal and non-coastal properties
coastal_prices <- data[data$is_coastal == TRUE, "Price"]
non_coastal_prices <- data[data$is_coastal == FALSE, "Price"]

# Perform Welch Two Sample t-test
t_test_result <- t.test(coastal_prices, non_coastal_prices)

# Display t-test result
print(t_test_result)
```

```
##
##  Welch Two Sample t-test
```

```
##
## data:  coastal_prices and non_coastal_prices
## t = 136.07, df = 104470, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  142967.1 147146.0
## sample estimates:
## mean of x mean of y
##  299804.6  154748.0
```

- The t-statistic (t = 134.9) and the extremely low p-value (p-value < 2.2e-16) indicate a highly significant difference between the mean prices of coastal and non-coastal properties.
- The sample means show that non-coastal properties have an average price of 154.093, while coastal properties have an average price of 296.503, demonstrating that coastal properties cost on average about 142.410 more than non-coastal ones.

Coastal properties have a significantly higher price than non-coastal ones. This is endorsed by an extremely low p-value and a confidence interval that does not include 0.

Regression Analysis

While the t-test and boxplot provide valuable insights, they do not account for other factors that might influence property prices, such as the size of the property or the number of bathrooms. To isolate the effect of the coastal location, we built a linear regression model with the following variables: - Dependent variable: Price - Independent variables: is_coastal, TotalArea, LivingArea, and NumberOfBathrooms

```
model <- lm(
  Price ~ EnergyCertificate + Lift + HasParking + PublishDate + Garage +
          ElectricCarsCharging + ConservationStatus + Type +
          GrossAreaScaled + LivingAreaScaled + BuiltAreaScaled + TotalAreaScaled +
          LotSizeScaled + OtherRoomsScaled + NumberOfBedroomsScaled +
          NumberOfBathroomsScaled + NumberOfWCScaled + FloorScaled +
          ConstructionYearScaled + latitudeScaled + longitudeScaled,
  data = data
)
summary(model)
```

```
##
## Call:
## lm(formula = Price ~ EnergyCertificate + Lift + HasParking +
##     PublishDate + Garage + ElectricCarsCharging + ConservationStatus +
##     Type + GrossAreaScaled + LivingAreaScaled + BuiltAreaScaled +
##     TotalAreaScaled + LotSizeScaled + OtherRoomsScaled + NumberOfBedroomsScaled +
##     NumberOfBathroomsScaled + NumberOfWCScaled + FloorScaled +
##     ConstructionYearScaled + latitudeScaled + longitudeScaled,
##     data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -469443  -79539  -17934   49752  808351
##
## Coefficients:
##                                 Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                    3.557e+05  3.338e+03  106.558  < 2e-16 ***
## EnergyCertificateA+           -9.124e+03  2.518e+03   -3.624 0.000291 ***
## EnergyCertificateB            -1.197e+04  2.495e+03   -4.797 1.61e-06 ***
```

```
## EnergyCertificateB-             -3.246e+04  2.547e+03  -12.746  < 2e-16 ***
## EnergyCertificateC              -5.601e+04  1.938e+03  -28.894  < 2e-16 ***
## EnergyCertificateD              -7.038e+04  1.976e+03  -35.609  < 2e-16 ***
## EnergyCertificateE              -8.426e+04  2.110e+03  -39.942  < 2e-16 ***
## EnergyCertificateF              -9.616e+04  2.336e+03  -41.167  < 2e-16 ***
## EnergyCertificateG              -7.007e+04  6.509e+03  -10.765  < 2e-16 ***
## EnergyCertificateNC             -7.792e+04  1.756e+03  -44.377  < 2e-16 ***
## EnergyCertificateNo Certificate -1.263e+05  1.293e+05   -0.977 0.328572
## EnergyCertificateNot available  -4.926e+04  3.054e+04   -1.613 0.106724
## LiftTRUE                         3.400e+04  1.362e+03   24.956  < 2e-16 ***
## HasParkingTRUE                   1.776e+04  1.059e+03   16.778  < 2e-16 ***
## PublishDate                     -5.501e-01  2.089e-01   -2.633 0.008460 **
## GarageTRUE                       2.364e+03  1.770e+03    1.336 0.181506
## ElectricCarsChargingTRUE         4.089e+04  2.056e+03   19.886  < 2e-16 ***
## ConservationStatus               2.812e+03  5.500e+02    5.113 3.18e-07 ***
## TypeHouse                       -4.542e+04  1.629e+03  -27.881  < 2e-16 ***
## TypeLand                        -1.723e+05  2.217e+03  -77.739  < 2e-16 ***
## TypeOther                       -5.168e+04  1.747e+03  -29.585  < 2e-16 ***
## GrossAreaScaled                  3.273e+04  1.309e+03   25.007  < 2e-16 ***
## LivingAreaScaled                 1.782e+03  8.150e+02    2.187 0.028752 *
## BuiltAreaScaled                  2.286e+04  1.155e+03   19.790  < 2e-16 ***
## TotalAreaScaled                 -1.273e+03  5.842e+02   -2.179 0.029346 *
## LotSizeScaled                    2.427e+04  4.575e+02   53.058  < 2e-16 ***
## OtherRoomsScaled                -8.979e+02  4.054e+02   -2.215 0.026767 *
## NumberOfBedroomsScaled           3.627e+04  5.653e+02   64.157  < 2e-16 ***
## NumberOfBathroomsScaled          4.438e+04  6.985e+02   63.530  < 2e-16 ***
## NumberOfWCScaled                -3.991e+03  4.725e+02   -8.446  < 2e-16 ***
## FloorScaled                     -4.532e+03  5.615e+02   -8.072 6.99e-16 ***
## ConstructionYearScaled           1.405e+04  4.998e+02   28.117  < 2e-16 ***
## latitudeScaled                  -2.943e+04  4.062e+02  -72.463  < 2e-16 ***
## longitudeScaled                 -4.231e+04  4.185e+02 -101.111  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 129300 on 120344 degrees of freedom
## Multiple R-squared:  0.6017, Adjusted R-squared:  0.6016
## F-statistic:  5509 on 33 and 120344 DF,  p-value: < 2.2e-16
```

```
model <- lm(
  Price ~ EnergyCertificate + Lift + HasParking + PublishDate + Garage +
          ElectricCarsCharging + ConservationStatus + Type +
          GrossAreaScaled + LivingAreaScaled + BuiltAreaScaled + TotalAreaScaled +
          LotSizeScaled + OtherRoomsScaled + NumberOfBedroomsScaled +
          NumberOfBathroomsScaled + NumberOfWCScaled + FloorScaled +
          ConstructionYearScaled + is_coastal,
  data = data
)
summary(model)
```

```
##
## Call:
## lm(formula = Price ~ EnergyCertificate + Lift + HasParking +
##     PublishDate + Garage + ElectricCarsCharging + ConservationStatus +
##     Type + GrossAreaScaled + LivingAreaScaled + BuiltAreaScaled +
##     TotalAreaScaled + LotSizeScaled + OtherRoomsScaled + NumberOfBedroomsScaled +
```

```
##     NumberOfBathroomsScaled + NumberOfWCScaled + FloorScaled +
##     ConstructionYearScaled + is_coastal, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -506202  -83333  -19284   51688  815885
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     3.031e+05  3.430e+03  88.375  < 2e-16 ***
## EnergyCertificateA+            -8.489e+03  2.586e+03  -3.283  0.00103 **
## EnergyCertificateB             -1.289e+04  2.562e+03  -5.031 4.88e-07 ***
## EnergyCertificateB-            -2.734e+04  2.613e+03 -10.463  < 2e-16 ***
## EnergyCertificateC             -5.172e+04  1.988e+03 -26.017  < 2e-16 ***
## EnergyCertificateD             -6.675e+04  2.028e+03 -32.914  < 2e-16 ***
## EnergyCertificateE             -8.296e+04  2.167e+03 -38.282  < 2e-16 ***
## EnergyCertificateF             -9.871e+04  2.400e+03 -41.131  < 2e-16 ***
## EnergyCertificateG             -7.932e+04  6.683e+03 -11.869  < 2e-16 ***
## EnergyCertificateNC            -8.407e+04  1.800e+03 -46.709  < 2e-16 ***
## EnergyCertificateNo Certificate -1.819e+05  1.329e+05  -1.369  0.17104
## EnergyCertificateNot available  -7.050e+04  3.138e+04  -2.247  0.02465 *
## LiftTRUE                         3.070e+04  1.401e+03  21.917  < 2e-16 ***
## HasParkingTRUE                   1.395e+04  1.087e+03  12.828  < 2e-16 ***
## PublishDate                      5.879e-03  2.121e-01   0.028  0.97789
## GarageTRUE                      -5.716e+03  1.813e+03  -3.153  0.00161 **
## ElectricCarsChargingTRUE         4.402e+04  2.113e+03  20.832  < 2e-16 ***
## ConservationStatus              -7.739e+01  5.610e+02  -0.138  0.89028
## TypeHouse                       -5.359e+04  1.671e+03 -32.073  < 2e-16 ***
## TypeLand                        -1.729e+05  2.278e+03 -75.905  < 2e-16 ***
## TypeOther                       -5.946e+04  1.793e+03 -33.163  < 2e-16 ***
## GrossAreaScaled                  3.305e+04  1.345e+03  24.566  < 2e-16 ***
## LivingAreaScaled                -2.274e+03  8.360e+02  -2.720  0.00653 **
## BuiltAreaScaled                  2.549e+04  1.185e+03  21.511  < 2e-16 ***
## TotalAreaScaled                 -2.443e+03  5.999e+02  -4.073 4.65e-05 ***
## LotSizeScaled                    2.223e+04  4.674e+02  47.555  < 2e-16 ***
## OtherRoomsScaled                -1.364e+03  4.167e+02  -3.274  0.00106 **
## NumberOfBedroomsScaled           3.728e+04  5.815e+02  64.116  < 2e-16 ***
## NumberOfBathroomsScaled          4.701e+04  7.174e+02  65.532  < 2e-16 ***
## NumberOfWCScaled                -5.035e+03  4.854e+02 -10.373  < 2e-16 ***
## FloorScaled                     -1.377e+03  5.750e+02  -2.395  0.01660 *
## ConstructionYearScaled           1.098e+04  5.128e+02  21.418  < 2e-16 ***
## is_coastalTRUE                   9.867e+04  8.856e+02 111.413  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 132900 on 120345 degrees of freedom
## Multiple R-squared:  0.5793, Adjusted R-squared:  0.5792
## F-statistic:  5180 on 32 and 120345 DF,  p-value: < 2.2e-16
```

```
model <- lm(Price ~ latitudeScaled + longitudeScaled, data = data)
summary(model)
```

```
##
## Call:
## lm(formula = Price ~ latitudeScaled + longitudeScaled, data = data)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -369175 -143722  -42441  109221  812339
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     251570.5      550.7  456.84   <2e-16 ***
## latitudeScaled  -25111.1      571.8  -43.91   <2e-16 ***
## longitudeScaled -63028.8      571.8 -110.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 191100 on 120375 degrees of freedom
## Multiple R-squared:   0.13,  Adjusted R-squared:   0.13
## F-statistic:  8997 on 2 and 120375 DF,  p-value: < 2.2e-16
```

```r
model <- lm(Price ~ is_coastal, data = data)
summary(model)
```

```
##
## Call:
## lm(formula = Price ~ is_coastal, data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -299505 -129805  -49805  104067  720252
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    154748.0      965.2   160.3   <2e-16 ***
## is_coastalTRUE 145056.6     1181.4   122.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 193100 on 120376 degrees of freedom
## Multiple R-squared:  0.1113, Adjusted R-squared:  0.1113
## F-statistic: 1.508e+04 on 1 and 120376 DF,  p-value: < 2.2e-16
```

Key insights:

- The negative coefficients in latitude and longitude suggest properties become less expensive as you move north (latitude) and east (longitude)
- There is a clear relationship between geographical location and property prices
- Coastal properties are significantly more expensive
- The price premium for coastal properties is smaller when controlling for other features (price of 97.480 vs 142.410), suggesting some of the price difference is due to property characteristics rather than location alone
- Location (whether as coordinates or coastal indicator) explains about 11-13% of price variation by itself, but contributes to models that explain over 55% of price variation when combined with other features (Adjusted R-squared at around 0.55).

Conclusions

The analysis confirms a significant relationship between a property's geographical location and its price. Coastal properties are consistently more expensive than non-coastal ones, as evidenced by the t-test, both

plots, and regression analysis. Specifically, the t-test shows that being in a coastal area increases property prices by an average of 142.410 if it is the only considered variable, which is almost twice the value. In reality, the regression model showed that the increase is more likely to be around 97.480, and that the increase of price of the previous value is due to the influence of other variables. These findings highlight the premium associated with coastal locations and reinforce the importance of geographical features in influencing real estate values.