1. GENERAL PREPARATION

Remove all objects from the workspace, clean console and collect garbage

1.1. Remove all objects from the workspace

```
rm(list = ls())
cat("\f")
```

```
gc()
```

```
##          used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
## Ncells 528375 28.3    1174930 62.8         NA   669425 35.8
## Vcells 971352  7.5    8388608 64.0      16384  1851712 14.2
```

1.2. Set a CRAN Mirror

The default behavior for install.packages() assumes a CRAN mirror is set. If no mirror is configured, R throws the error. Explicitly setting a mirror ensures the packages can be installed without interruption. A **CRAN mirror** is a server that hosts the Comprehensive R Archive Network (CRAN) repository, providing access to R packages and documentation. Mirrors are distributed globally to improve download speed and reliability by serving users from geographically closer locations. R requires a mirror to download packages effectively.

```
options(repos = c(CRAN = "https://cran.r-project.org"))
```

1.3 .Install packages

```
# Set CRAN Mirror
options(repos = c(CRAN = "https://cran.r-project.org"))

# List required packages
required_packages <- c("readr", "dplyr", "rpart", "caret", "Metrics", "rpart.plot")

# Install missing packages
new_packages <- required_packages[!(required_packages %in% installed.packages()[, "Packa
ge"])]
if (length(new_packages)) install.packages(new_packages)

# Load all libraries
lapply(required_packages, library, character.only = TRUE)
```

```
##
## Attaching package: 'dplyr'
```

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

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

```
## Warning: package 'caret' was built under R version 4.3.3
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'Metrics'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```
## [[1]]
## [1] "readr"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [7] "methods"   "base"
##
## [[2]]
## [1] "dplyr"     "readr"     "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
##
## [[3]]
##  [1] "rpart"     "dplyr"     "readr"     "stats"     "graphics"  "grDevices"
##  [7] "utils"     "datasets"  "methods"   "base"
##
## [[4]]
##  [1] "caret"     "lattice"   "ggplot2"   "rpart"     "dplyr"     "readr"
##  [7] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
## [13] "base"
##
## [[5]]
##  [1] "Metrics"   "caret"     "lattice"   "ggplot2"   "rpart"     "dplyr"
##  [7] "readr"     "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [13] "methods"   "base"
##
## [[6]]
##  [1] "rpart.plot" "Metrics"    "caret"      "lattice"    "ggplot2"
##  [6] "rpart"      "dplyr"      "readr"      "stats"      "graphics"
## [11] "grDevices"  "utils"      "datasets"   "methods"    "base"
```

2. Load and clean the data

```
setwd("~/Dropbox/ UCL PhD/5- Learning Agendas/16-Data Analytics/Kaggle/Kaggle Housing Mo
del/R version of mode")

home_data <- read.csv("melb_data.csv", header = TRUE)

home_data <- na.omit(home_data)
```

3. Creat a simple model with no split data

3.1 Select target and features

```
y <- home_data$Price

feature_names <- c("Rooms", "Bathroom", "Landsize", "Lattitude", "Longtitude")
X <- home_data[feature_names]

summary(X)
```

```
##      Rooms          Bathroom        Landsize         Lattitude
##  Min.   :1.000   Min.   :1.000   Min.   :    0.0   Min.   :-38.16
##  1st Qu.:2.000   1st Qu.:1.000   1st Qu.:  167.0   1st Qu.:-37.86
##  Median :3.000   Median :1.000   Median :  404.0   Median :-37.80
##  Mean   :2.978   Mean   :1.594   Mean   :  487.5   Mean   :-37.81
##  3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:  641.0   3rd Qu.:-37.76
##  Max.   :8.000   Max.   :8.000   Max.   :37000.0   Max.   :-37.41
##    Longtitude
##  Min.   :144.5
##  1st Qu.:144.9
##  Median :145.0
##  Mean   :145.0
##  3rd Qu.:145.1
##  Max.   :145.5
```

```
head(X)
```

```
##    Rooms Bathroom Landsize Lattitude Longtitude
## 2      2        1      156  -37.8079   144.9934
## 3      3        2      134  -37.8093   144.9944
## 5      4        1      120  -37.8072   144.9941
## 7      3        2      245  -37.8024   144.9993
## 8      2        1      256  -37.8060   144.9954
## 10     2        1      220  -37.8010   144.9989
```

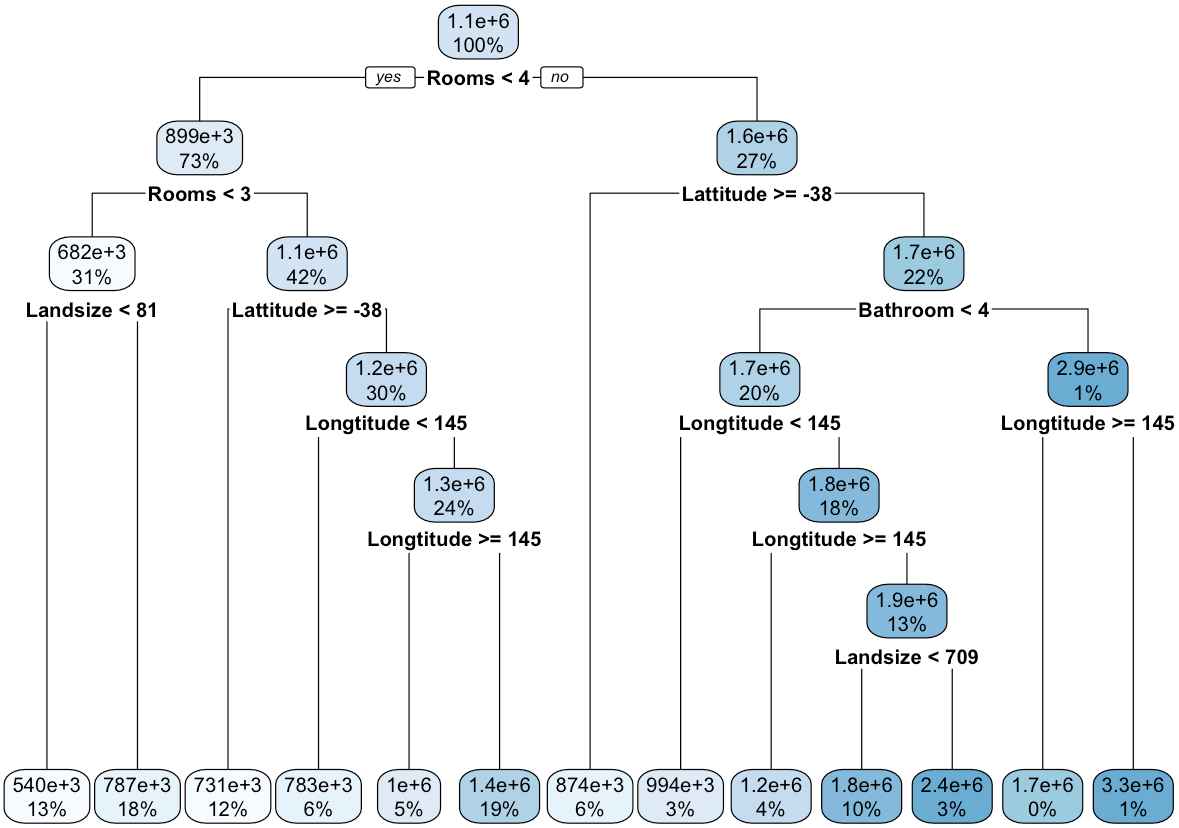
3.2. Specify and visualize the model

```
melbmod1 <- rpart(Price ~ Rooms + Bathroom + Landsize + Lattitude + Longtitude,
                          data = home_data,
                          method = "anova",
                          control = rpart.control(cp = 0.01))

print(melbmod1)
```

```
## n= 6830
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##   1) root 6830 3.095997e+15 1077604.0
##     2) Rooms< 3.5 4958 1.145311e+15  899206.0
##       4) Rooms< 2.5 2118 2.090175e+14  681874.5
##         8) Landsize< 80.5 900 3.580626e+13  539863.5 *
##         9) Landsize>=80.5 1218 1.416492e+14  786808.8 *
##       5) Rooms>=2.5 2840 7.616476e+14 1061286.0
##        10) Lattitude>=-37.75219 820 4.064196e+13  731093.4 *
##        11) Lattitude< -37.75219 2020 5.953111e+14 1195325.0
##          22) Longtitude< 144.891 413 2.419750e+13  782866.1 *
##          23) Longtitude>=144.891 1607 4.827961e+14 1301327.0
##            46) Longtitude>=145.0988 321 1.115497e+14 1006823.0 *
##            47) Longtitude< 145.0988 1286 3.364556e+14 1374839.0 *
##     3) Rooms>=3.5 1872 1.374980e+15 1550091.0
##       6) Lattitude>=-37.74835 402 3.585598e+13  873789.7 *
##       7) Lattitude< -37.74835 1470 1.104973e+15 1735039.0
##        14) Bathroom< 3.5 1389 8.572895e+14 1665717.0
##          28) Longtitude< 144.8884 177 2.732685e+13  993590.4 *
##          29) Longtitude>=144.8884 1212 7.383249e+14 1763874.0
##            58) Longtitude>=145.1021 299 5.559960e+13 1223525.0 *
##            59) Longtitude< 145.1021 913 5.668337e+14 1940834.0
##             118) Landsize< 708.5 697 3.202128e+14 1787659.0 *
##             119) Landsize>=708.5 216 1.774975e+14 2435106.0 *
##        15) Bathroom>=3.5 81 1.265466e+14 2923784.0
##          30) Longtitude>=145.1037 18 4.960362e+12 1743972.0 *
##          31) Longtitude< 145.1037 63 8.937242e+13 3260873.0 *
```

```
library(rpart.plot)
rpart.plot(melbmod1)
```



3.3. Make and view predictions, and calculate MAE

```
predictions <- predict(melbmod1, X)

print("Top Predictions:")
```

```
## [1] "Top Predictions:"
```

```
print(head(predictions, 5))
```

```
##         2         3         5         7         8
##  786808.8 1374838.8 1787658.6 1374838.8  786808.8
```

```
print("Actual Values:")
```

```
## [1] "Actual Values:"
```

```
print(head(y, 5))
```

```
## [1] 1035000 1465000 1600000 1876000 1636000
```

```
mae1 <- mae(y, predictions)
print(paste("Mean Absolute Error:", mae1))
```

```
## [1] "Mean Absolute Error: 298045.610140932"
```

4. Split Train Test model

4.1. Partition data

```
set.seed(1)
trainIndex <- createDataPartition(y, p = 0.8, list = FALSE)
train_data <- home_data[trainIndex, ]
test_data <- home_data[-trainIndex, ]
```

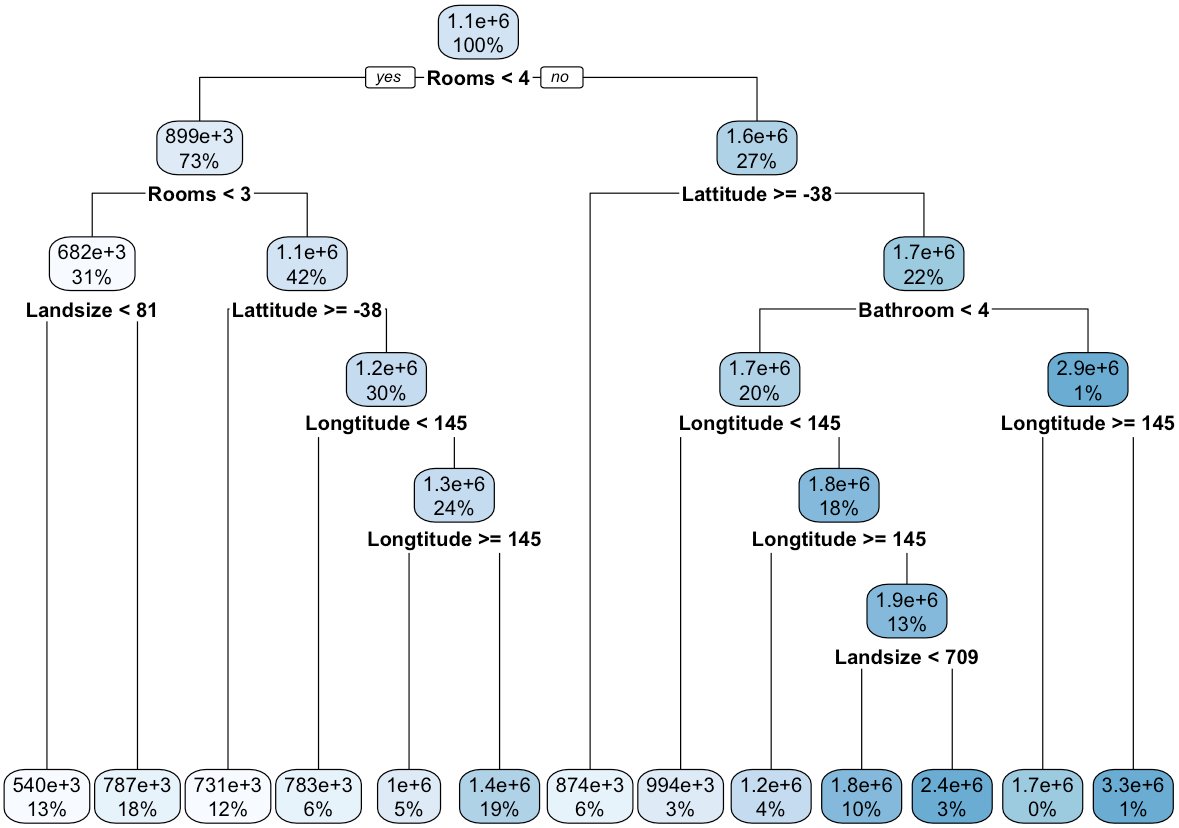
4.2. Specify and visualize model on training dataset

```
train_model <- rpart(Price ~ Rooms + Bathroom + Landsize + Lattitude + Longtitude,
                     data = home_data,
                     method = "anova",
                     control = rpart.control(cp = 0.01))

print(train_model)
```

```
## n= 6830
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##   1) root 6830 3.095997e+15 1077604.0
##     2) Rooms< 3.5 4958 1.145311e+15  899206.0
##       4) Rooms< 2.5 2118 2.090175e+14  681874.5
##         8) Landsize< 80.5 900 3.580626e+13  539863.5 *
##         9) Landsize>=80.5 1218 1.416492e+14  786808.8 *
##       5) Rooms>=2.5 2840 7.616476e+14 1061286.0
##        10) Lattitude>=-37.75219 820 4.064196e+13  731093.4 *
##        11) Lattitude< -37.75219 2020 5.953111e+14 1195325.0
##          22) Longtitude< 144.891 413 2.419750e+13  782866.1 *
##          23) Longtitude>=144.891 1607 4.827961e+14 1301327.0
##            46) Longtitude>=145.0988 321 1.115497e+14 1006823.0 *
##            47) Longtitude< 145.0988 1286 3.364556e+14 1374839.0 *
##     3) Rooms>=3.5 1872 1.374980e+15 1550091.0
##       6) Lattitude>=-37.74835 402 3.585598e+13  873789.7 *
##       7) Lattitude< -37.74835 1470 1.104973e+15 1735039.0
##        14) Bathroom< 3.5 1389 8.572895e+14 1665717.0
##          28) Longtitude< 144.8884 177 2.732685e+13  993590.4 *
##          29) Longtitude>=144.8884 1212 7.383249e+14 1763874.0
##            58) Longtitude>=145.1021 299 5.559960e+13 1223525.0 *
##            59) Longtitude< 145.1021 913 5.668337e+14 1940834.0
##             118) Landsize< 708.5 697 3.202128e+14 1787659.0 *
##             119) Landsize>=708.5 216 1.774975e+14 2435106.0 *
##        15) Bathroom>=3.5 81 1.265466e+14 2923784.0
##          30) Longtitude>=145.1037 18 4.960362e+12 1743972.0 *
##          31) Longtitude< 145.1037 63 8.937242e+13 3260873.0 *
```

```
library(rpart.plot)
rpart.plot(train_model)
```



##

4.3. Make and view predictions, and calculate MAE

```
test_predictions <- predict(train_model, test_data[feature_names])

test_mae <- mae(test_data$Price, test_predictions)
print(paste("Test Set MAE:", test_mae))
```

```
## [1] "Test Set MAE: 299096.453449816"
```