Problem understanding

write about what features can help to make the task is solvable(for example how many parks aroud the house, quality of life at region, medecine, also can write some hypotesis). Write just text without coding something

In the project we need predict a housing price. The housing price depends on four general fueatures:

- quality of life at region (medecine, life security, quality of schools, air pollution, etc.)
- quality of house (area, number of rooms, area of kitchen, floor number, soundproofing, etc)
- economy of contry (price can rise from time to time due to inflation, crisis)
- country and region laws (taxes, some specific laws that affect the price)

Data understadning

About this Dataset

Context

The dataset consists of lists of unique objects of popular portals for the sale of real estate in Russia. More than 5477 thousand objects. The dataset contains 5477 real estate objects in Russia.

Dataset The dataset has 13 fields.

- date - Date of publication of the announcement;
- time - The time when the ad was published;
- geo_lat - Latitude coordinate
- geo_lon - Longitude coordinate
- region - Region of Russia. There are 85 subjects in the country in total.
- building_type - Facade type. 0 - Other. 1 - Panel. 2 - Monolithic. 3 - Brick. 4 - Blocky. 5 - Wooden
- object_type - Apartment type. 1 - Secondary real estate market; 2 - New building;
- level - Apartment floor
- levels - Number of storeys
- rooms - The number of living rooms. If the value is "-1", then it means "studio apartment"
- area - The total area of the apartment
- kitchen_area - Kitchen area
- price - Price. in rubles

Content

The dataset cointains information about date of publication of the announcement, as we do not have any inforamtion about the inflation of country we can not find some really good correlation beetween price and economy features. The same situation is for quality of life at region, country laws. To minimize mistakes from quality of life at region and region laws we use a model only for one chosen region to predict prices.

Data preparation

The dataset consist x numbers of samples, does not have any Nan values and have only 13 duplicates.

We choose only x region because calculation process takes too much time and diffrent region has different quality of life and laws. Also we choose some intervals for atributes to avoid outliers:

- date ["2020-01-01", "2021-01-01"]
- price [1000000, 1.5e7]
- area [20, 160]
- kitchen_area [4, 30]

We choose intervals of them by boxplots of attributes. After all filters we have N samples.

```
library(tidyverse) # tidy data cleaning
library(tidymodels) # tidy machine learning
library(tsibble)
library(leaflet)
library(ggplot2)
library(vctrs)

raw <- read.csv("../input/russia-real-estate-20182021/all_v2.csv")

raw %>% summarise_all(~ sum(is.na(.))) # Nan values

raw_1 = raw %>% distinct()
head(raw)

housing_raw <- raw %>%  # choose only 7896 region
  filter(
    region == 7896
  )
length(housing_raw$price)

housing_raw <- housing_raw %>% # apply some filters
  filter(
    date > "2020-01-01",
    date < "2021-01-01"
  ) %>%
  mutate(
```

```r
    year_month = yearmonth(date)
  ) %>%
  distinct_at(
    vars(c(everything(), -price, -date ),
         ),
    .keep_all = T
  ) %>%
  filter(
    price > 1000000,
    price < 1.5e7
  ) %>%
  filter(
    area > 20,
    area < 160
  ) %>%
  filter(
    kitchen_area > 4,
    kitchen_area < 30
  )

b <- boxplot(housing_raw$price) # make boxplot
b1 <- boxplot(housing_raw$area)
b2 <- boxplot(housing_raw$kitchen_area)

length(housing_raw$price) # amount of examples after filters

housing_raw %>% #distribution of announcements by date
  mutate(
    year_month = yearmonth(date)
  ) %>%
  count(year_month) %>%
  ggplot(aes(year_month, n)) +
  geom_col()

keeps <- c("price","area", "rooms", "kitchen_area",
"object_type","levels", "level", "building_type") # summary of
attributes
tmp = housing_raw[keeps]
summary(tmp)

count = vec_count(raw$region)
count = count[count$count > 20000,] # choose region only with
announsments more than 20000

#plt =
ggplot(data = count, mapping = aes(x = reorder(key, count), count)) +
  geom_bar(stat = "identity") + coord_flip() + xlab("Region") +
ylab("Frequency")+ theme(plot.margin = margin(,,,, "cm"),
        plot.background = element_rect(fill = "white"))

ggsave(("test.png"), plot = plt, dpi = 300)
```

```r
library(leaflet) #opent street map
library(dplyr)
leaflet()%>%addTiles()%>%addCircleMarkers(data=housing_raw, lat =
~geo_lat, lng = ~geo_lon, radius = ~0.7)

# Function to add correlation coefficients
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    Cor <- abs(cor(x, y)) # Remove abs function if desired
    txt <- paste0(prefix, format(c(Cor, 0.123456789), digits = digits)
[1])
    if(missing(cex.cor)) {
        cex.cor <- 0.4 / strwidth(txt)
    }
    text(0.5, 0.5, txt,
         cex = 1 + cex.cor * Cor) # Resize the text by level of
correlation
}

pairs(~ price + building_type + level + levels + rooms + area +
kitchen_area + object_type, data = housing_raw, main = "Housing Data",

              upper.panel = panel.cor)

keeps <- c("price","area", "rooms", "kitchen_area",
"object_type","levels", "level", "building_type")
int_df = housing_raw[keeps]
nrow(int_df)

install.packages('fastDummies')

library('fastDummies')
dataf <- dummy_cols(int_df, select_columns = c('object_type',
'building_type'),
           remove_selected_columns = TRUE)

dataf <- dataf %>% mutate_at(c("area", "rooms", "kitchen_area",
"levels","level"), ~(scale(.) %>% as.vector))

# set.seed for pseudorandom number generation. In simple words, for
reproducibility
set.seed(123)

#store rows for partition
rows1 <- sample(1:nrow(dataf), nrow(dataf)*0.8, replace = F)

#training data set
train_base <- dataf[rows1,]
#testing data set
```

```
test_base <- dataf[-rows1,]

# structure of the Training dataset
str(train_base)
```

Modeling

```
library(caret)

model_lm <- caret::train(price ~., data = train_base,
                         method = "lm")
model_lm

model_forest <- caret::train(price ~., data = train_base,
                         method = "ranger",
                         )
model_forest

model_gbm <- caret::train(price ~., data = train_base,
                         method = "gbm",
                         trControl = trainControl(method =
"repeatedCV",
                                                  number = 5, repeats
= 2))

model_gbm

model_lm1 <- caret::train(price ~., data = train_base,
                         method = "lm",
                         trControl = trainControl(method =
"repeatedCV", number = 5, repeats = 2),
                         preProcess = c("center","scale","pca"))

model_lm1

model_forest1 <- caret::train(price ~., data = train_base,
                         method = "ranger",
                         trControl = trainControl(method =
"repeatedCV", number = 5, repeats = 2),
                         preProcess = c("center","scale","pca"))
model_forest1

model_gbm1 <- caret::train(price ~., data = train_base,
                         method = "gbm",
                         trControl = trainControl(method =
"repeatedCV", number = 5, repeats = 2),
                         preProcess = c("center","scale","pca"))
model_gbm1
```

Evaluating

```r
library(Metrics)
library(MLmetrics)

pred_lm <- stats::predict(object = model_lm, test_base)

#RMSE
error_lm <- pred_lm - test_base$price
rmse_lm <- sqrt(mean(error_lm^2))

paste("RMSE for Linear Regression is:",round(rmse_lm,3))
mae(actual = test_base$price, predicted = pred_lm)
```

- RMSE: 1166090
- MAE: 706938
- R2: 0.65

```r
# predicting using GBM
pred_gbm <- stats::predict(object = model_gbm, test_base)

#RMSE
error_gbm <- pred_gbm - test_base$price
rmse_gbm <- sqrt(mean(error_gbm^2))
paste("RMSE for Gradient Boosting Machine is:",round(rmse_gbm,3))
mae(actual = test_base$price, predicted = pred_gbm)
R2_Score(test_base$price,pred_gbm)
```

- RMSE: 350659
- MAE: 635302
- R2: 0.89

```r
# predicting using  Model
pred_rf <- stats::predict(object = model_forest, test_base)

#RMSE
error_rf <- pred_rf - test_base$price
rmse_rf <- sqrt(mean(error_rf^2))
paste("RMSE for Random Forest is:",round(rmse_rf,3))
mae(actual = test_base$price, predicted = pred_rf)
R2_Score(test_base$price, pred_rf)
```

- RMSE: 965076
- MAE: 620402
- R2: 0.75