Apartment Rental Prediction System

Dr. Ivan S. Zapreev

2020-01-07

Contents

Introduction

Dataset overview

As stated on the webpage of the 'Apartment rental offers in Germany' dataset, it contains `198,379` rental offers scraped from the Germany's biggest real estate online platform ß ImmobilienScout24.

The data set consists of a single CSV file: *immo_data.csv* which only contains offers for rental properties. The data features important rental property attributes, such as the living area size, the rent (both base rent as well as total rent), the location, type of energy, and etc. The `date` column present in the data set defines the time of scraping, which was done on three distinct dates: *2018-09-22*, *2019-05-10* and *2019-10-08*.

The complete list of data set columns is extensive[1] and thus in this study we will use the following subset:

```
##  [1] "hasKitchen"             "heatingType"            "balcony"
##  [4] "lift"                   "garden"                 "cellar"
##  [7] "noParkSpaces"           "livingSpace"            "typeOfFlat"
## [10] "noRooms"                "floor"                  "numberOfFloors"
## [13] "condition"              "newlyConst"             "interiorQual"
## [16] "yearConstructed"        "energyEfficiencyClass" "regio1"
## [19] "regio2"                 "regio3"                 "baseRent"
## [22] "electricityBasePrice"  "heatingCosts"           "serviceCharge"
## [25] "totalRent"              "date"
```

This sub-selection reduces the number of considered data set columns[2] from 48 to 26 and is motivated by the personal preferences of the report's author and has no scientifically proven motivation. On the contrary, this column selection shall be seen as a part of problem statement. In other words, the task is to build an accurate[3] rental price prediction model based on the predictors from this set of columns.

The additional data preparation steps will be described in the *"Data wrangling"* section of this document.

Project goal

Execution plan

Let us now briefly outline the main steps to be performed to reach the previously formalized project goal:

1. **Prepare the data** – see the *"Data wrangling"* section:
   - Select, clean, and reshape relevant data; split it into training and validation sets; and etc.
2. **Analyze the dataset** – see the *"Dataset analysis"* section:
   - Perform data exploration and visualization; summarize insights on the data.
3. **Describe the modeling approach** – see the *"Modeling approach"* section:
   - Consider the insights of the data analysis; suggest the way for building the prediction model.
4. **Present modeling results** – see the *"Results"* section:
   - Train the model on the `modeling` set; analyze the training results; evaluate on the `validation` set.
5. **Provide concluding remarks** – see the *"Conclusions"* section:
   - Summarize the results; mention any approach limitations; outline possible future improvements.

Data wrangling

In this section we present cleaning, enriching, and restructuring the raw data taken from the 'Apartment rental offers in Germany' dataset.

This section will be organized as follows: First we explain how we cleaned the data and solved some of its inconsistencies, by enriching the data. Then we identify some structural changes done to the data. Further, we provide a summary of the wrangled data set. In the end, we explain how we split the entire data set into the `validation` and `modeling` sub-sets[4].

[1] Please consider reading *"Appendix A"* for the complete list of the data set columns.
[2] Please consider reading *"Appendix B"* for the column descriptions.
[3] Please consider reading the *"Project goal"* section for an exact goal formulation.
[4] The latter will also be split into the `training` and `testing` set for the sake of model cross-validation.

Data cleaning & enriching

Let us note that the number of data entries in the original data set is equal to 198332. This data is however not ready to be worked with as, for instance, it contains multiple N/A values and there are also other inconsistencies present.

Consider the next table summarizing the number of N/A values per data set column:

```
## # A tibble: 26 x 3
##    `Column name`         `N/A count` `N/A percent`
##    <chr>                       <int>         <dbl>
##  1 electricityBasePrice       151158         76.2
##  2 energyEfficiencyClass      143315         72.3
##  3 heatingCosts               135154         68.2
##  4 noParkSpaces               130405         65.8
##  5 interiorQual                83001         41.8
##  6 numberOfFloors              71792         36.2
##  7 condition                   50317         25.4
##  8 yearConstructed             42293         21.3
##  9 floor                       37612         19.0
## 10 heatingType                 32605         16.4
## 11 totalRent                   29762         15.0
## 12 typeOfFlat                  27571         13.9
## 13 serviceCharge                5110          2.58
## 14 hasKitchen                      1          0
## 15 lift                            1          0
## 16 garden                          1          0
## 17 cellar                          1          0
## 18 livingSpace                     1          0
## 19 noRooms                         1          0
## 20 baseRent                        1          0
## 21 balcony                         0          0
## 22 newlyConst                      0          0
## 23 regio1                          0          0
## 24 regio2                          0          0
## 25 regio3                          0          0
## 26 date                            0          0
```

As one can see, about $\frac{1}{2}$ of the columns has 10–80% N/A^s, whereas the other half has (almost) no N/A^s.

The data cleaning and enriching will be explained in the next steps:

1. We begin with the totalRent column as this is the value that we want to predict;
2. We proceed with the columns with the marginal ($< 1\%$) of N/A values;
3. We cover the remaining columns in the descending order of the number of N/A values.
4. We consider and sole some other data inconsistencies.

The first steps

The totalRent column contains data that we want to predict. Therefore, the rows with totalRent == N/A are useless to us and shall be removed. Unfortunately, this will reduce the data set by 15.01%. There are also 13 columns with a marginal (0 to 1) number of N/A values. The latter can be seamlessly removed as even if all of these N/A^s appear in different rows, we will remove at most 13 entries which is just 0.0066% of data.

The main columns

Let us consider the columns one by one. Note that, some modifications we will do to the data to remove the `N/A` values may introduce bias. To for test that we would need a clean data set with no `N/A` values initially present and then to use such a data set for the trained model(s) validation. Due to the lack of time this will not be done in the case study.

Column: `electricityBasePrice` - 76.2% `N/A` values

We will set the electricity base price for the `N/A` values to zero. The motivation is that, since the number of `N/A` values is almost 80% and no other zero values are present:

```
x <- arog_data$selected_data %>% filter(!is.na(electricityBasePrice))
sum(x$electricityBasePrice == 0)
```

```
## [1] 0
```

it is likely that the `N/A` values were used to determine the fact that there is no electricity base price.

Column: `energyEfficiencyClass` - 72.3% `N/A` values

The energy efficiency factor levels are:

```
levels(arog_data$selected_data$energyEfficiencyClass)
```

```
##  [1] ""               "A"              "A_PLUS"         "B"
##  [5] "C"              "D"              "E"              "F"
##  [9] "G"              "H"              "NO_INFORMATION"
```

So we shall naturally set all the `N/A` and `""` energy efficiency levels to `"NO_INFORMATION"`.

Column: `heatingCosts` - 68.2% `N/A` values

We will set the heating costs for the `N/A` values to zero as there are already 1989 zero-valued heating cost entries. It is unlikely that there are non-heated accommodations in Germany so *we assume that the* 0 *values, the same as `N/A`^{s} mean - "unknown"*.

Column: `noParkSpaces` - 65.8% `N/A` values

We will set the number of parking places for the `N/A` values to zero as there is already 2850 zero-valued entries. By this step we assume that, `N/A` is interpreted as *"not applicable"* or *"no are available"*.

Column: `interiorQual` - 38.8% `N/A` values

The interior quality factor levels are:

```
levels(arog_data$selected_data$interiorQual)
```

```
## [1] ""              "luxury"        "normal"        "simple"
## [5] "sophisticated"
```

So we shall introduce a new level for the `N/A` and `""` values, called `"unknown"`.

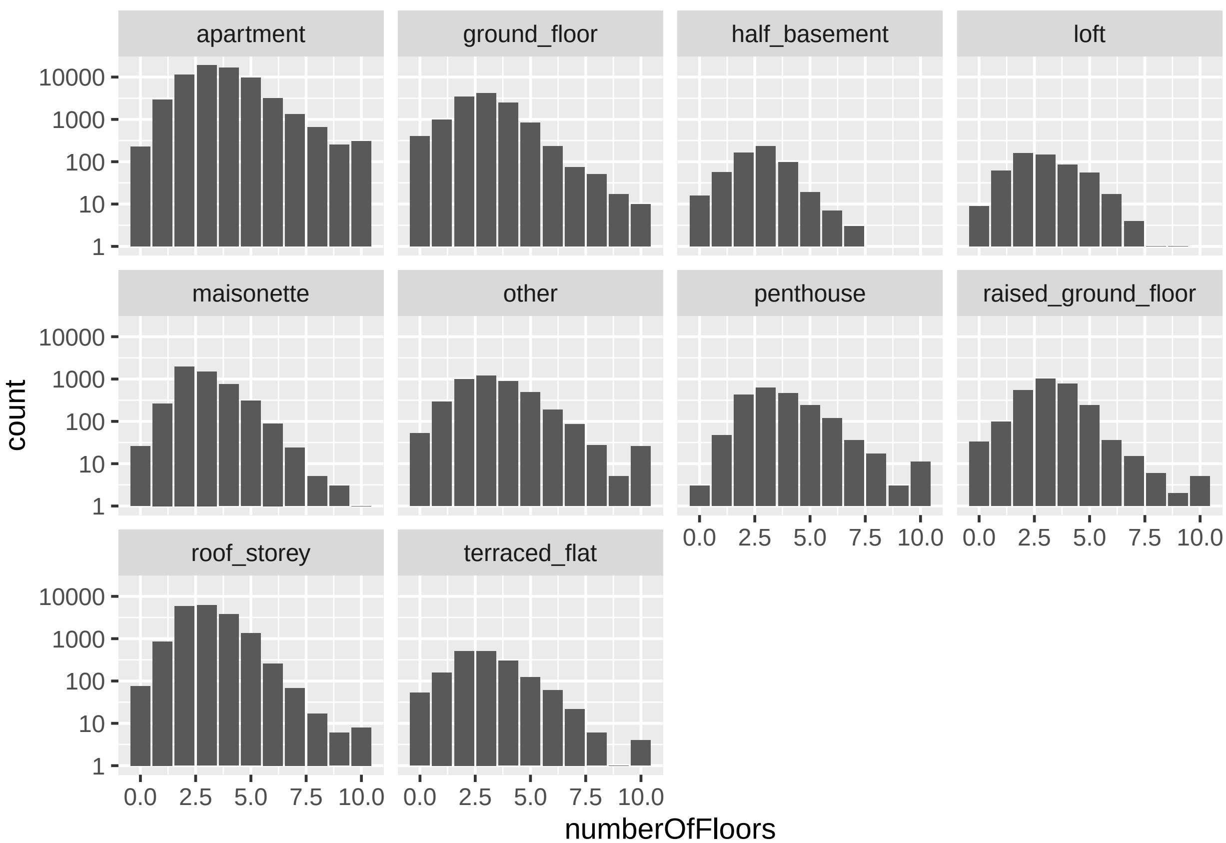
Column: `numberOfFloors` - 36.2% `N/A` values

Setting the `N/A` values for the floors shall be agreed with the apartment type, if we gather some number of floors statistics for each available apartment type we get the following:

Table 1: Type of flat vs. number of floors statistics

	numberOfFloors				
typeOfFlat	Count	Average	Standard error	Minimum	Maximum
apartment	66844	3.9	6.6	0	999
roof_storey	18450	3.1	6.7	0	800
ground_floor	12802	3	8.9	0	999
maisonette	4979	2.9	1.5	0	43
other	4284	3.6	5	0	301
raised_ground_floor	2779	3.4	7.3	0	370
penthouse	2011	3.7	2.2	0	33
terraced_flat	1760	3	1.5	0	14
half_basement	598	2.7	1.1	0	7
loft	542	3	1.5	0	15
""	0	NaN	NA	Inf	-Inf

From where we conclude that the data we have is very polluted. Clearly, one can not expect apartments with 99 floors and alike. See also on the large average (all `+/-` around 3 floors) and the huge standard error values. If we visualize the results (filtering out 1662 flats with more than 10 floors), we see that:



The data seems to be normally distributed (except for the `apartment` type) with the mean values within 2.5 - 4.0 range. This makes us believe that this data is too much biased and polluted. So we will not rely on this column in our analysis.

Column: `condition` - 25.4% `N/A` values

The condition factor levels are:

```
##  [1] ""                                   "first_time_use"
##  [3] "first_time_use_after_refurbishment" "fully_renovated"
##  [5] "mint_condition"                     "modernized"
##  [7] "need_of_renovation"                 "negotiable"
##  [9] "refurbished"                        "ripe_for_demolition"
## [11] "well_kept"
```

So we shall introduce a new level for the `N/A` and `""` values, called `"unknown"`.

Column: `yearConstructed` - 21.3% `N/A` values

There is no good default to replace the `N/A` values here. Yet, it is a significant amount of data which we do not want to exclude. Therefore drop this column from the analysis and just use the `newlyConst` flag column.

Column: `floor` - 19.0% `N/A` values

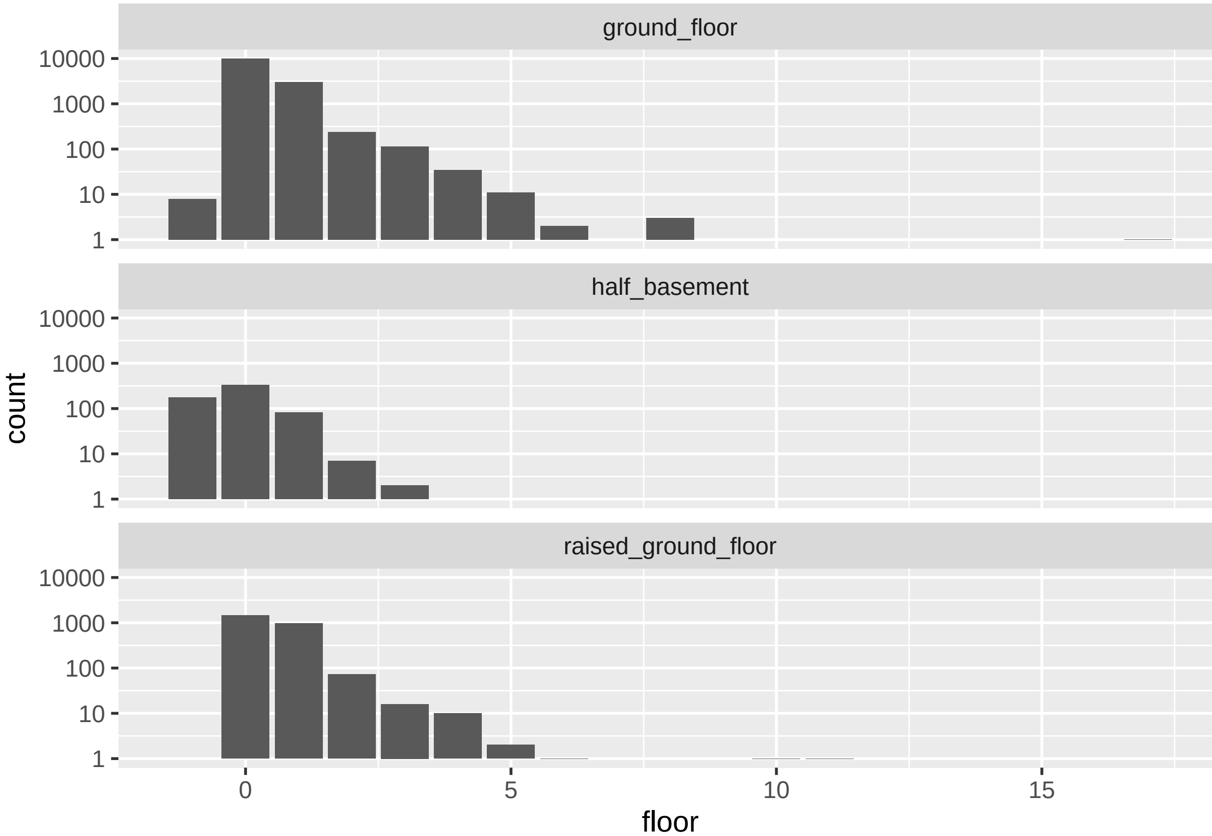
We could assign some `floor` values based on the flat types:

```
##  [1] ""                "apartment"         "ground_floor"
##  [4] "half_basement"   "loft"              "maisonette"
##  [7] "other"           "penthouse"         "raised_ground_floor"
## [10] "roof_storey"     "terraced_flat"
```

For example, we could consider assigning:

- `half_basement` – the average `floor` for the half-basement
- `ground_floor` – `floor = 0`
- `raised_ground_floor` – the average `floor` for the raised ground floor

but, let us look at the floor values (filtering out 10 flats higher that at the 100'th floor), for these flat types:



From the data above we see that we shall not only correct the `N/A` values but set all of the floor values for the considered flat types as follows:

- `half_basement` – `floor = -1`
- `ground_floor` – `floor = 0`
- `raised_ground_floor` – `floor = 0`

If we do that then there will still be 20050 (10.1% of data) `N/A` floor values for the flat types for which we can not give any exact value. So we will just assign those to the mean floor value in the category.

Column: `heatingType` - 16.4% `N/A` values

The heating type factor levels are:

```
##  [1] "central_heating"                "combined_heat_and_power_plant"
##  [3] "district_heating"               "electric_heating"
##  [5] "floor_heating"                  "gas_heating"
##  [7] "heat_pump"                      "night_storage_heater"
##  [9] "oil_heating"                    "self_contained_central_heating"
## [11] "solar_heating"                  "stove_heating"
## [13] "wood_pellet_heating"
```

So we shall introduce a new level for the `N/A`, `""`, and `"H"` values, called `"unknown"`.

Column: `typeOfFlat` - 13.9% `N/A` values

The type of flat factor levels are:

```
##  [1] ""                    "apartment"          "ground_floor"
##  [4] "half_basement"       "loft"               "maisonette"
```

```
##  [7] "other"               "penthouse"           "raised_ground_floor"
## [10] "roof_storey"         "terraced_flat"
```

So we shall introduce a new level for the `N/A` and `""` values, called `"unknown"`. Note that, we do not use the pre-defined level `"other"` here as we interpret it as known flat type which is just not on the list of available choices.

Column `serviceCharge` - 2.58% `N/A` values

We will set the service charges for the `N/A` values to zero. The motivation is that, there are:

```r
x <- arog_data$selected_data %>% filter(!is.na(serviceCharge))
sum(x$serviceCharge == 0)
```

```
## [1] 2496
```

zero values present, so we interpret the `N/A` values as defining the fact of no additional service charges.

Additional steps

In addition to the data alternations done above we have also done the following:

- Re-setting the number of floors:
  - `half_basement` $-$`floor` $= -1$
  - `ground_floor` $-$ `floor` $= 0$
  - `raised_ground_floor` $-$`floor` $= 0$
- Filter out flats:
  - With `floor` > 100
  - Other than `"half_basement"`, `"other"`, and `"unknown"`; but with `floor` < 0

Restructuring data

There data set at hand does not have any complex structure. However, because we want to be able to do predictions per city and avoid cities with the same names within different lands and regions we shall combine the `regio` columns into a new single one, as follows:

```r
  clean_arog_data <- clean_arog_data %>%
    unite("location", c("regio1", "regio2", "regio3"), remove=FALSE) %>%
    select(-regio1, -regio2, -regio3)
```

The resulting columns have values constructed according to the following pattern:

`location = regio1 + "_" + regio2 + "_" + regio3`

For example:

```r
arog_data$wrangled_data$location[1:5]
```

```
## [1] "Nordrhein_Westfalen_Essen_Karnap"
## [2] "Nordrhein_Westfalen_Steinfurt_Kreis_Emsdetten"
## [3] "Nordrhein_Westfalen_Bottrop_Lehmkuhle"
## [4] "Sachsen_Anhalt_Salzlandkreis_Schönebeck_Elbe"
## [5] "Sachsen_Chemnitz_Bernsdorf"
```

Wrangled data set

Let us now summarize the resulting clean data:

```
## # A tibble: 22 x 3
##    `Column name`          `N/A count` `N/A percent`
##    <chr>                        <int>         <dbl>
##  1 hasKitchen                       0             0
##  2 heatingType                      0             0
##  3 balcony                          0             0
##  4 lift                             0             0
##  5 garden                           0             0
##  6 cellar                           0             0
##  7 noParkSpaces                     0             0
##  8 livingSpace                      0             0
##  9 typeOfFlat                       0             0
## 10 noRooms                          0             0
## 11 floor                            0             0
## 12 condition                        0             0
## 13 newlyConst                       0             0
## 14 interiorQual                     0             0
## 15 energyEfficiencyClass            0             0
## 16 location                         0             0
## 17 baseRent                         0             0
## 18 electricityBasePrice             0             0
## 19 heatingCosts                     0             0
## 20 serviceCharge                    0             0
## 21 totalRent                        0             0
## 22 date                             0             0
```

As one can notice, the dat set size has been reduced from 198332 to 164637 . The major reason for that is excluding the rows with the `N/A` values of the `totalRent` column. Let us recall that the number of such raws was 15.01% of the data set, e.g. 29770 rows. It now remains to notice that $198332 - 29770 = 168562 \approx 164637$. The remaining 2% delta is explained by cleaning the `floor`/`typeOfFlat` columns and etc.

The data has been cleaned but we can expect that there is some noise in the data which we have not addressed. We might get more data-quality insights when we perform data analysis in the subsequent sections.

Splitting data

To facilitate supervised learning, the wrangled data is split into the `modeling`, 90% of data, and `validation`, 10% of data, sets. The former will be used for training statistical model(s) and the latter for the model(s) validation. Note that, for the sake of subsequent cross validation during modeling part, we further split the `modeling` set into the `training`, 80% thereof, and `testing`, 20% thereof, sets.

We split the data in the following steps:

1. The data is randomly split into to parts according to the specified ratio:

```
test_index <- createDataPartition(y = data_set$totalRent, times = 1,
                                  p = ratio, list = FALSE)
```

2. The `test_index` rows are the candidates for the `testing`/`validation` set rows
3. The factorized column values of the `testing`/`validation` set are considered:

```
str_data <- capture.output(str(arog_data$wrangled_data))
str_replace_all(str_subset(str_data, "Factor"), "\",..:.*","\",..")
```

```
## [1] " $ heatingType          : Factor w/ 14 levels \"central_heating\",.."
## [2] " $ typeOfFlat           : Factor w/ 11 levels \"apartment\",\"ground_floor\",.."
## [3] " $ condition            : Factor w/ 11 levels \"first_time_use\",.."
## [4] " $ interiorQual         : Factor w/ 5 levels \"luxury\",\"normal\",.."
## [5] " $ energyEfficiencyClass: Factor w/ 10 levels \"A\",\"A_PLUS\",\"B\",.."
## [6] " $ date                 : Factor w/ 4 levels \"\",\"May19\",\"Oct19\",.."
```

4. The rows with the values not present in the `testing/modeling` set are dropped
5. The `testing/modeling` set consists of rows absent in the `testing/validation` set

The procedure above ensures that the `testing/validation` set can always be evaluated on a model trained on the `testing/modeling` set. For more details, see the `create_arog_data` and `split_train_test_sets` functions located in the `apartment_rental_project.R` script.

The resulting set sizes are as follows:

- `modeling` – 135485 rows, 82.3% of data
  - `training` – 119219 rows, 88% of `modeling` set
  - `testing` – 16266 rows, 12% of `modeling` set
- `validation` – 16266 rows, 9.9% of data

As expected, due to returning `testing/validation` set rows to the `testing/modeling` set for consistency, the desired set ratios are biased. The `validation` set size is almost as prescribed (10% of data), but the `testing` set size is affected more significantly[5]. Yet, we see no issue as the `testing` set is still > 10% of the `modeling` set, which should be enough for performing cross validation.

Data analysis

Modeling approach

Results

Conclusions

Future work

Check for introducing any bias by data wrangling.

Appendix A: The complete list of data set columns

Hereby we present the list of columns from the original data set:

```
##  [1] "regio1"                    "serviceCharge"
##  [3] "heatingType"               "telekomTvOffer"
##  [5] "telekomHybridUploadSpeed" "newlyConst"
##  [7] "balcony"                   "electricityBasePrice"
##  [9] "picturecount"              "pricetrend"
## [11] "telekomUploadSpeed"        "totalRent"
```

[5] The requested `testing` set size was 20% of the `modeling` set.

```
## [13] "yearConstructed"          "electricityKwhPrice"
## [15] "scoutId"                  "noParkSpaces"
## [17] "firingTypes"              "hasKitchen"
## [19] "geo_bln"                  "cellar"
## [21] "yearConstructedRange"     "baseRent"
## [23] "houseNumber"              "livingSpace"
## [25] "geo_krs"                  "condition"
## [27] "interiorQual"             "petsAllowed"
## [29] "streetPlain"              "lift"
## [31] "baseRentRange"            "typeOfFlat"
## [33] "geo_plz"                  "noRooms"
## [35] "thermalChar"              "floor"
## [37] "numberOfFloors"           "noRoomsRange"
## [39] "garden"                   "livingSpaceRange"
## [41] "regio2"                   "regio3"
## [43] "description"              "facilities"
## [45] "heatingCosts"             "energyEfficiencyClass"
## [47] "lastRefurbish"            "date"
```

Appendix B: Data set column descriptions

Here is the list of the initially considered data set columns with the descriptions thereof:

1. `hasKitchen` – has a kitchen
2. `balcony` – does the object have a balcony
3. `cellar` – has a cellar
4. `lift` – is elevator available
5. `floor` – which floor is the flat on
6. `garden` – has a garden
7. `noParkSpaces` – number of parking spaces
8. `livingSpace` – living space in sqm
9. `condition` – condition of the flat
10. `interiorQual` – interior quality
11. `regio1` – Bundesland
12. `regio2` - District or Kreis, same as geo krs
13. `regio3` – City/town
14. `noRooms` – number of rooms
15. `numberOfFloors` – number of floors in the building
16. `typeOfFlat` – type of flat
17. `yearConstructed` – construction year
18. `newlyConst` – is the building newly constructed
19. `heatingType` – Type of heating
20. `energyEfficiencyClass` – energy efficiency class

21. `heatingCosts` – monthly heating costs in €
22. `serviceCharge` – auxiliary costs such as electricity or Internet in €
23. `electricityBasePrice` – monthly base price for electricity in €
24. `baseRent` – base rent without electricity and heating
25. `totalRent` – total rent (usually a sum of base rent, service charge and heating cost)
26. `date` – time of scraping