Generalised Additive Models

Luca Casuscelli

28/05/2020

#Intro In this chapter we will take a look at our data with the help of generalised additive models.

#Import Data As a first step, we must import the data.

```
#required packages
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
library(dplyr)
```

```
##
## 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
```

```
#import data
house_data <- read.csv("C:/Users/lucac/OneDrive/Dokumente/GitHub/ML-Group-Work/01_data/house_data.csv")
attach(house_data)

#We work for every model with a separate data set
house_data_gam <- house_data
attach(house_data_gam)
```

```
## The following objects are masked from house_data:
##
##     build_year, lat, living_area, long, municipality_name, num_rooms,
##     number_of_apartments_in_hectare, number_of_buildings_in_hectare,
##     number_of_workplaces_in_hectare,
##     number_of_workplaces_sector_1_in_hectare,
##     number_of_workplaces_sector_2_in_hectare,
##     number_of_workplaces_sector_3_in_hectare, object_type_name,
##     population_in_hectare, price, travel_time_private_transport,
##     travel_time_public_transport, water_percentage_1000, zipcode
```

```
#correct the price skewness and add to data frame
house_data_gam$price.log <- log(house_data_gam$price)

#define which variables have to be seen as factors
zipcode.fac <- factor(zipcode)
```

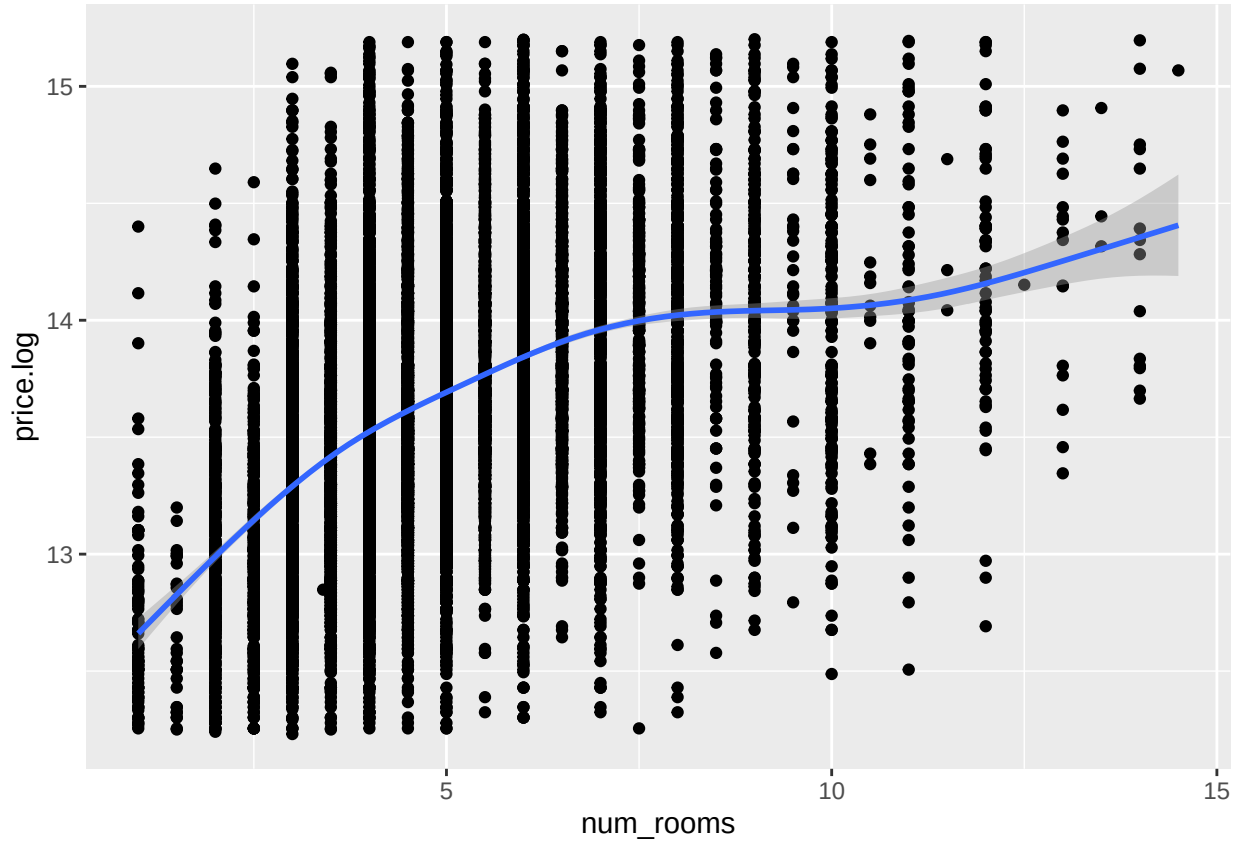
```
#limit the number of rooms
hd_limited_rooms <- filter(house_data_gam, num_rooms < 15 & num_rooms > 0)
```

#Graphical analysis So far, we have assumed all effects of continuous predictors to be linear. In this chapter we will leave linearity and look for non-linear relationships in the data. As an example let's compare the dependency of price from number of rooms. In order to achieve a better graphical overview, we will limit the number of rooms to be considered to 15 and delete all zeros, as it doesn't make sense that any real estate has zero rooms.

Example price to number of rooms:

```
library(ggplot2)

gg.price.rooms <- ggplot(data = hd_limited_rooms,
                         mapping = aes(y = price.log,
                                       x = num_rooms)) +
 geom_point()

gg.price.rooms +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



We must consider, that the variance for each category in the number of rooms is fairly high. Indicating, that much more variable do influence the price. At the same time, the curve indicates, that the effect of the price may be not linear.

Quadratic effects

In this chapter we try to fit a model by adding a qudratic term and compare it to a model with the assumption of linearity. As we have seen in the chapter about Linear Regression, we do not have to consider all variables. So, for simplicity and limitations in computational power, we avoid interactions and only consider the main variables.

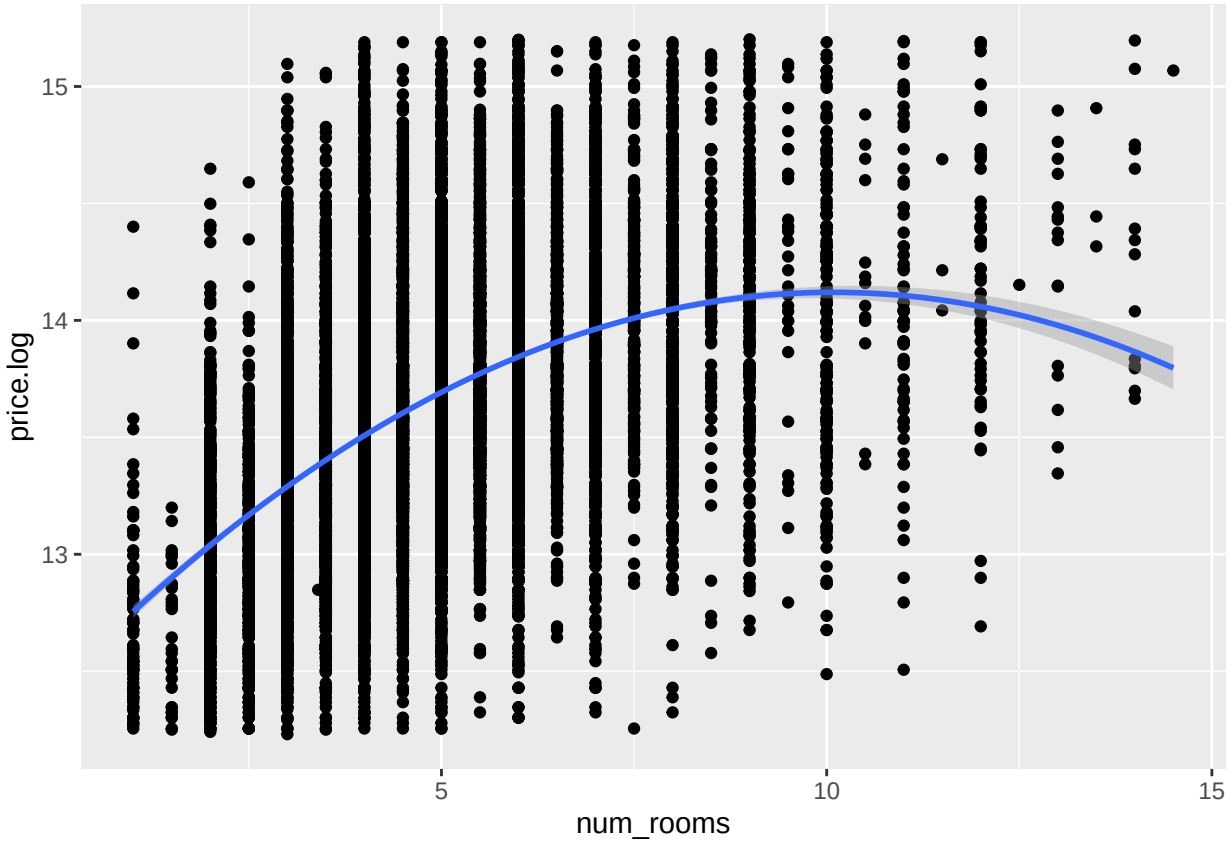
```
# 1) model with linear effect for price.log
lm.price.1 <- lm(price.log ~ object_type_name + build_year + living_area + num_rooms + number_of_apartme
+ number_of_workplaces_in_hectare + population_in_hectare,
data = house_data_gam)

# 2) model with quadratic effect for price.log
lm.price.2 <- update(lm.price.1, . ~ . + I(num_rooms^2))

# We test the quadratic term with a F-Test:
anova(lm.price.1, lm.price.2)
```

```
## Analysis of Variance Table
##
## Model 1: price.log ~ object_type_name + build_year + living_area + num_rooms +
##     number_of_apartments_in_hectare + number_of_workplaces_in_hectare +
##     population_in_hectare
## Model 2: price.log ~ object_type_name + build_year + living_area + num_rooms +
##     number_of_apartments_in_hectare + number_of_workplaces_in_hectare +
##     population_in_hectare + I(num_rooms^2)
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1  22560 4435.3
## 2  22559 4295.6  1    139.66 733.42 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is a strong evidence, that num_rooms needs a quadratic term. As we could already see visually.

We can now visualize our model on the data:

```
gg.price.rooms +
  geom_smooth(method = "lm",
              formula = y ~ poly(x, degree = 2))
```

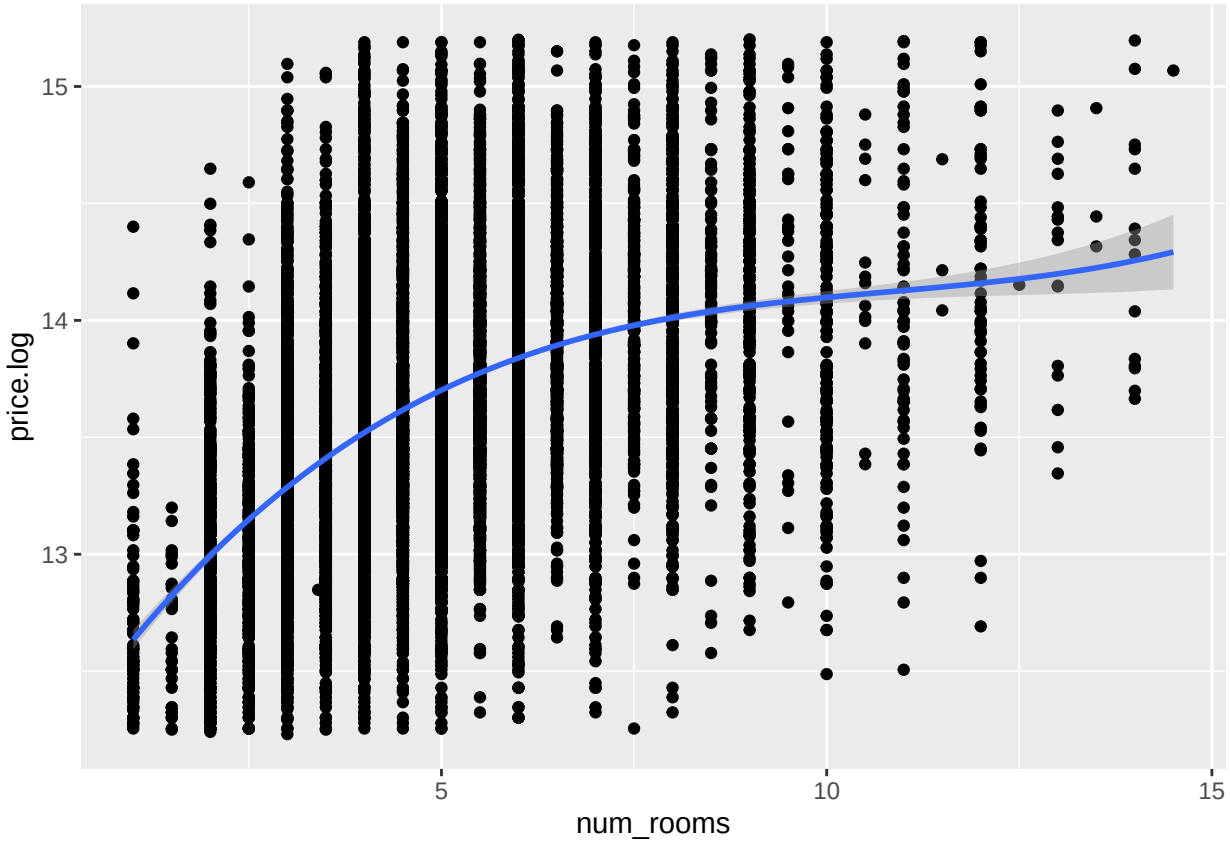


The value increase seems to be decaying with the numbers of rooms, which we already did see in the first visualisations. Nonetheless, it does not make sense that the value decreases with more than 10 rooms. The model obviously underestimates the price increase for a higher number of rooms. We can therefore try a more complex model.

More complex non-linear relationships

In this chapter we will have a look at more complex polynomials. Let's try a cubic polynomial to the data before:

```
gg.price.rooms +
  geom_smooth(method = "lm",
              formula = y ~ poly(x, degree = 3))
```



We can see that with a cubic polynomial we can reduce the error we had with quadratic polynomials in the example before. Now we can model our data with the following model:

```
lm.cubic <- lm(price.log ~ poly(num_rooms, degree = 3), data = house_data_gam )
summary(lm.cubic)
```

```
##
## Call:
## lm(formula = price.log ~ poly(num_rooms, degree = 3), data = house_data_gam)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.06614 -0.29384 -0.02404  0.27776  2.39466
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   13.641726   0.003023 4513.35   <2e-16 ***
## poly(num_rooms, degree = 3)1  29.833576   0.454084   65.70   <2e-16 ***
## poly(num_rooms, degree = 3)2 -16.721700   0.454084  -36.83   <2e-16 ***
## poly(num_rooms, degree = 3)3   8.802973   0.454084   19.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4541 on 22566 degrees of freedom
## Multiple R-squared:  0.2114, Adjusted R-squared:  0.2113
## F-statistic:  2016 on 3 and 22566 DF,  p-value: < 2.2e-16
```

We have a R-quared of 0.21, which tells us, that around 21% of the variation is explained by our variable

"num_rooms". As we already know, we have many more variables and therefore this is not surprising. Give the number of variables, we think that 21% is fairly good result.

Generalised Additive Models (GAMs)

As a next step we finally reach Generalised Additive Models. These are a very powerfull models to fit non-linear relations with multiple predictiors, as this the case with our data set.

As first step can allow all most relevant variable to have non-linear, smooth, effect. And select on which ones there is significance out of the result.

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

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

```
## This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
```

```
gam.price.1 <- gam(price.log ~ object_type_name + s(build_year) + s(living_area) + s(num_rooms) + s(numb
+ s(number_of_workplaces_in_hectare) + s(population_in_hectare),data = house_data_gam)
summary(gam.price.1)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## price.log ~ object_type_name + s(build_year) + s(living_area) +
##     s(num_rooms) + s(number_of_apartments_in_hectare) + s(number_of_workplaces_in_hectare) +
##     s(population_in_hectare)
##
## Parametric coefficients:
##                                  Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                     13.662430   0.004804 2844.130  < 2e-16 ***
## object_type_nameMehrfamilienhaus -0.045180   0.008143   -5.548 2.92e-08 ***
## object_type_nameSonstiges        -0.097643   0.010151   -9.619  < 2e-16 ***
## object_type_nameWohnung          -0.017694   0.007925   -2.233   0.0256 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                                    edf Ref.df       F  p-value
## s(build_year)                    8.674  8.957 139.951  < 2e-16 ***
## s(living_area)                   8.858  8.992 793.661  < 2e-16 ***
## s(num_rooms)                     6.486  7.423   4.207 9.55e-05 ***
## s(number_of_apartments_in_hectare) 3.733  4.639  23.366  < 2e-16 ***
## s(number_of_workplaces_in_hectare) 4.079  4.912  24.647  < 2e-16 ***
## s(population_in_hectare)         7.252  8.059  13.186  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## R-sq.(adj) =  0.474   Deviance explained = 47.5%
## GCV = 0.13772  Scale est. = 0.13746   n = 22570
```

As we can see out of the summary, all variables seem to have a fairly high edf (estimated degree of freedom) and do therefore not stand in linear relationship to the price.

We can try to optimize our model by removing the least relevant smoothing factors and see whether the model does suffer a lot from it:

```
gam.price.2 <- gam(price.log ~ object_type_name + s(build_year) + s(living_area) + s(num_rooms) + numbe
+ number_of_workplaces_in_hectare + population_in_hectare,data = house_data_gam)
summary(gam.price.2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## price.log ~ object_type_name + s(build_year) + s(living_area) +
##     s(num_rooms) + number_of_apartments_in_hectare + number_of_workplaces_in_hectare +
##     population_in_hectare
##
## Parametric coefficients:
##                                   Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                     13.6208768  0.0050862 2677.992  < 2e-16 ***
## object_type_nameMehrfamilienhaus -0.0489488  0.0080994   -6.044 1.53e-09 ***
## object_type_nameSonstiges        -0.0931867  0.0101309   -9.198  < 2e-16 ***
## object_type_nameWohnung          -0.0112565  0.0078424   -1.435    0.151
## number_of_apartments_in_hectare   0.0021635  0.0002679    8.074 7.12e-16 ***
## number_of_workplaces_in_hectare   0.0050612  0.0005640    8.974  < 2e-16 ***
## population_in_hectare            -0.0005134  0.0001125   -4.562 5.09e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                  edf Ref.df       F  p-value
## s(build_year)  8.763  8.977 133.693  < 2e-16 ***
## s(living_area) 8.862  8.993 786.806  < 2e-16 ***
## s(num_rooms)   6.422  7.363   4.402 5.44e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.472   Deviance explained = 47.2%
## GCV = 0.13832  Scale est. = 0.13813   n = 22570
```

We do realize, that the smoothing splines are only relevant for 3 variables: build_year, living_area, num_rooms. So we were able to save some computational power without loosing any quality on the model.

Collinearity of the model

Last but not least we must check the model on collinearity in order to make sure we do not take any wrong conclusions about which variables are of real relevance. For that we can use the vif() function from the {car} package:

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.6.3
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 3.6.3
```

```
##
## Attaching package: 'car'
```

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

```
vif(gam.price.2)
```

```
##                                       GVIF Df GVIF^(1/(2*Df))
## object_type_name                1.331786e+42  0             Inf
## number_of_apartments_in_hectare 1.331786e+42  0             Inf
## number_of_workplaces_in_hectare 1.331786e+42  0             Inf
## population_in_hectare           1.331786e+42  0             Inf
## build_year                      1.331786e+42  0             Inf
## living_area                     1.331786e+42  0             Inf
## num_rooms                       1.331786e+42  0             Inf
```

The Generalised Variance Inflation Error (GVIF) tell us whether we have to remove a variable. In respect to our model, we do not have any factor above 5. Therefore the model is not affected by collinearity.