Mathematics in Machine Learning

Vittorio Zampinetti

Contents

1 Introduction

Many real-world problems require estimation of the relationship between a dependent variable (often called *response*) and one or more independent explanatory variables (often called *predictors*). This task is known as *regression analysis*. In this work we apply this statistical processing, focusing in particular on *general linear models*, to estimate the extent of forest fires according to a set of attributes like weather conditions, time period, location etc. Fast detection is a key element for successful firefighting and the aim of this analysis is not only to accurately predict the burned area of a forest fire, but also to understand which predictors are decisive for this purpose, in order to keep the cost of sensors low while being able to readily react and prevent large environmental disasters.

Here a summary of the topics touched by this document. After briefly introducing the software environment in which the whole project has been developed, we describe the dataset attributes and main characteristics. Then we dive into the linear regression analysis, defining the metrics used for performance evaluation and selecting the features to be included in the models. After that, a peculiar approach for skewed data with many zeros is tested, given the unusual distribution of the response variable. This method, called the *"Two-part model"*, has been inspired by the research paper of Fletcher, MacKenzie, and Villouta (2005). Finally, after having analyzed the regression task under a probabilistic approach, we try to overcome the limitations of a linear model with the adoption of a widely used decision-rule based regression algorithm, namely Random Forest.

2 Dev Environment

The project is fully developed in the **R** language, being it particularly suitable for probabilistic modelling and data science procedures. The entire code is embedded in an R Markdown notebook which is then knit into this document.

The code follows the tidyverse standards as much as possible, using ggplot2 library as the main engine for plots and figures. In the end, concerning the random forest test, the quite new machine learning library called mlr3 has been exploited.

```r
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(gridExtra) # arrange plots in a grid

# ml libs for random forest section
library(mlr3)
library(mlr3viz)
library(mlr3learners)
library(mlr3tuning)
library(ranger) # random forest implementation
library(data.table)
library(paradox)
```

3 Dataset

The dataset has been created by Cortez and Morais (2007) and retrieved from the *UCI Machine Learning Repository* (2017) at this link. It was collected from January 2000 to December 2003. It consists of 517 records each of which is composed by 12 predictors and a ground-truth outcome value, *i.e.* the area of the forest fire in hectares (*ha*).

3.1 Attributes

The following is a brief description of the attributes:

Attribute	Description
X	x-axis spatial coordinate within the Montesinho park map: 1 to 9
Y	y-axis spatial coordinate within the Montesinho park map: 2 to 9
month	month of the year: 'jan' to 'dec'

Attribute	Description
day	day of the week: ‘mon’ to ‘sun’
FFMC	FFMC index from the FWI system: 18.7 to 96.20
DMC	DMC index from the FWI system: 1.1 to 291.3
DC	DC index from the FWI system: 7.9 to 860.6
ISI	ISI index from the FWI system: 0.0 to 56.10
temp	temperature in Celsius degrees: 2.2 to 33.30
RH	relative humidity in %: 15.0 to 100
wind	wind speed in km/h: 0.40 to 9.40
rain	outside rain in mm/m2 : 0.0 to 6.4
area	the burned area of the forest (in ha): 0.00 to 1090.84

The four indices, Fine Fuel Moisture Code (FFMC), Duff Moisture Code (DMC), Drought Code (DC) and Initial Spread Index (ISI) are specific measures for rating fire danger according to the Fire Weather Index (FWI) Canadian system. For more details about the meaning of these measures, please refer to the reference paper (2007).

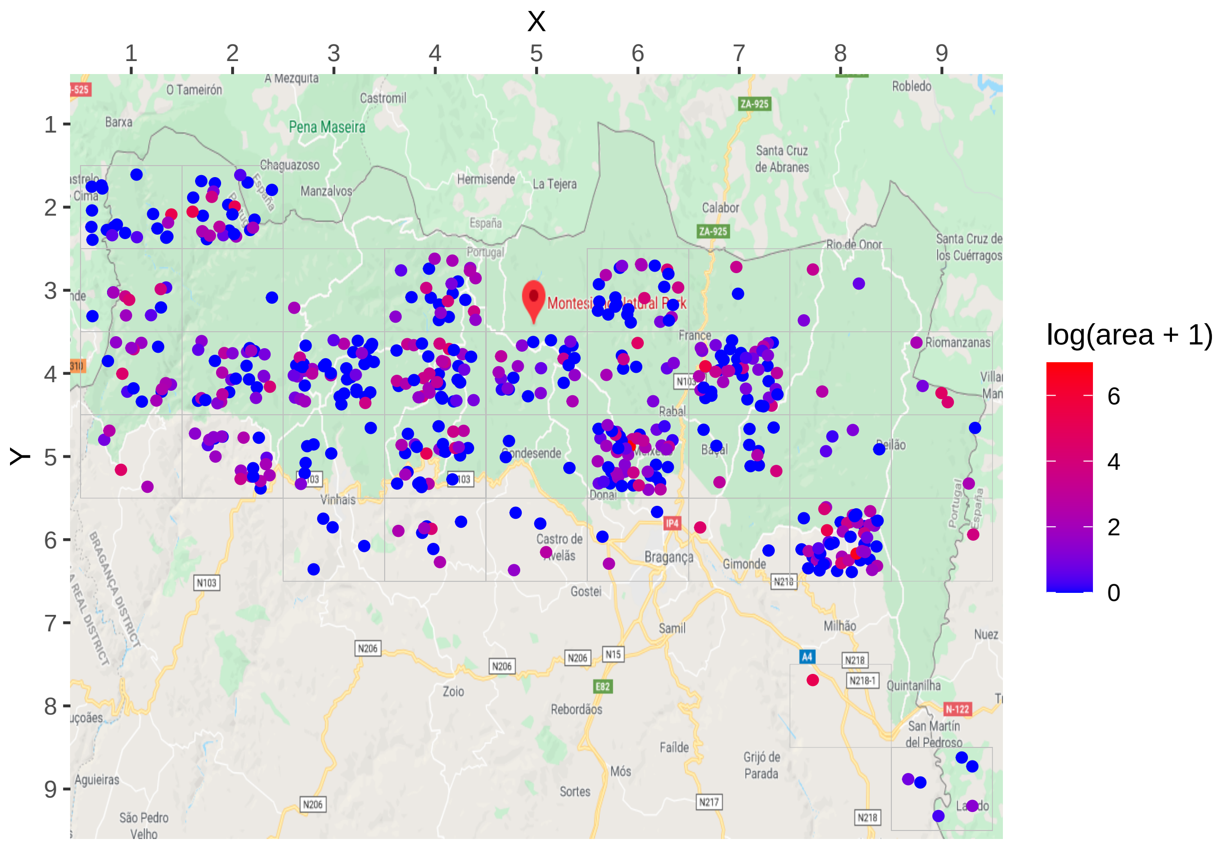
```
# read data
fires.raw <- read_csv("data/forestfires.csv", col_types = cols(
  X = col_factor(levels = 1:9),
  Y = col_factor(levels = 2:9),
  month = col_factor(levels = c("jan", "feb", "mar", "apr", "may", "jun", "jul", "aug", "sep", "oct", "
  day = col_factor(levels = c("mon", "tue", "wed", "thu", "fri", "sat", "sun")),
  FFMC = col_double(),
  DMC = col_double(),
  DC = col_double(),
  ISI = col_double(),
  temp = col_double(),
  RH = col_double(),
  wind = col_double(),
  rain = col_double(),
  area = col_double()
))

# merge X, Y in one single factor xy
fires <- fires.raw %>%
  mutate(X = factor(paste(X,Y, sep = ""))) %>%
  rename(xy = X) %>%
  select(-c(Y))

fires
```

```
## # A tibble: 517 x 12
##    xy    month day    FFMC   DMC    DC   ISI  temp    RH  wind  rain  area
##    <fct> <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 75    mar   fri    86.2  26.2  94.3   5.1   8.2    51   6.7   0       0
##  2 74    oct   tue    90.6  35.4 669.    6.7  18      33   0.9   0       0
##  3 74    oct   sat    90.6  43.7 687.    6.7  14.6    33   1.3   0       0
##  4 86    mar   fri    91.7  33.3  77.5   9     8.3    97   4     0.2     0
##  5 86    mar   sun    89.3  51.3 102.    9.6  11.4    99   1.8   0       0
##  6 86    aug   sun    92.3  85.3 488    14.7  22.2    29   5.4   0       0
##  7 86    aug   mon    92.3  88.9 496.    8.5  24.1    27   3.1   0       0
##  8 86    aug   mon    91.5 145.  608.   10.7   8      86   2.2   0       0
##  9 86    sep   tue    91   130.  693.    7    13.1    63   5.4   0       0
## 10 75     sep    sat     92.5   88    699.     7.1   22.8     40    4      0        0
## # ... with 507 more rows
```

3.2 Exploratory Data Analysis (EDA)

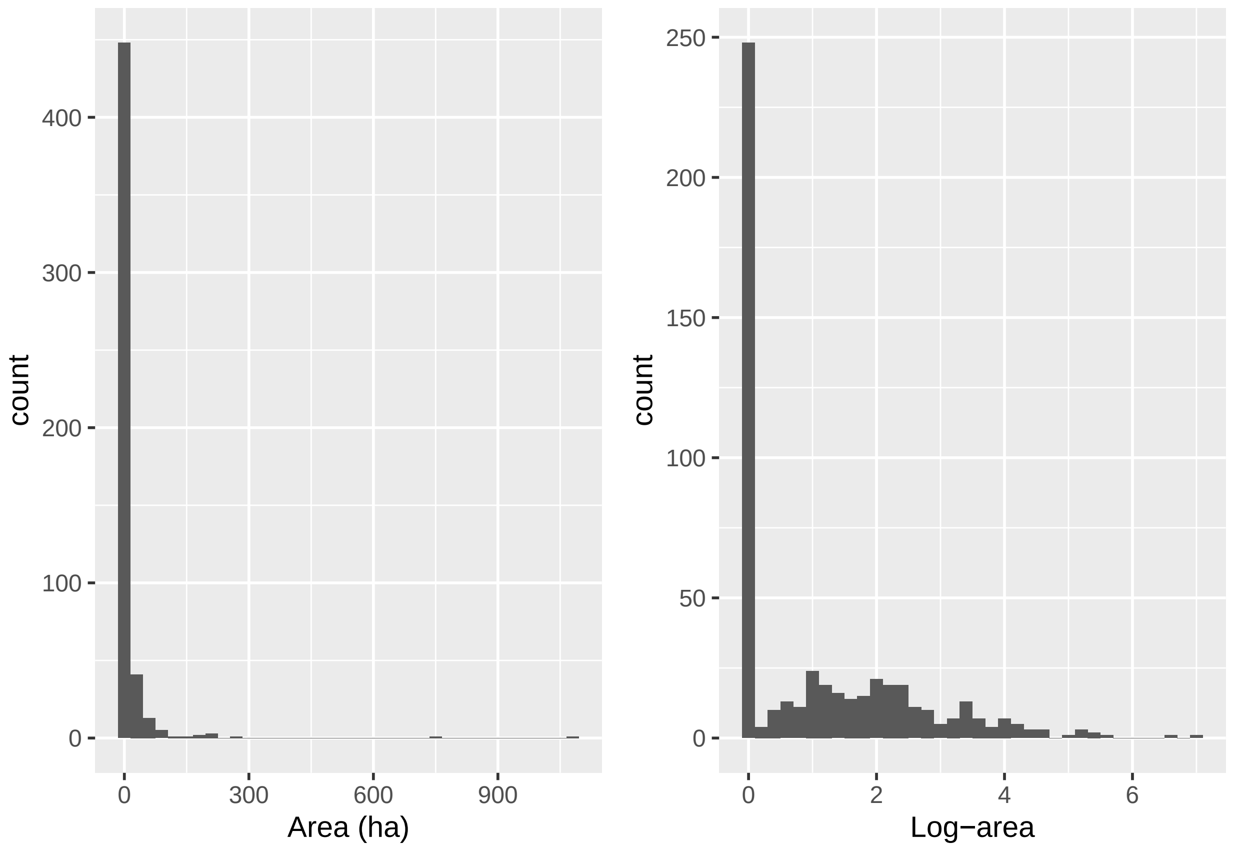
Apart from location (X and Y), *month* and *day* attributes, which have been encoded as categorical predictors (or *factors* in the R jargon), all the others are real-valued numbers. It is worth mentioning that the coordinates have been merged together as the key point is the zone in which the fire has spread and not the x or y coordinates alone. Here a map showing the distribution of the forest fires in *Montesinho Natural Park*, Portugal.

```r
library(png)
library(ggpubr)
img <- readPNG("data/map.png")

fires.raw %>%
  ggplot(aes(X, y = reorder(Y, desc(Y)))) +
  background_image(img) +
  geom_jitter(aes(color = log(area+1))) +
  geom_tile(color = "grey", alpha = 0) +
  scale_x_discrete(position = "top") +
  scale_y_discrete(limits = factor(9:1)) +
  coord_cartesian(ylim = c(1,9)) +
  scale_color_gradient(low = "blue", high = "red") +
  ylab("Y")
```



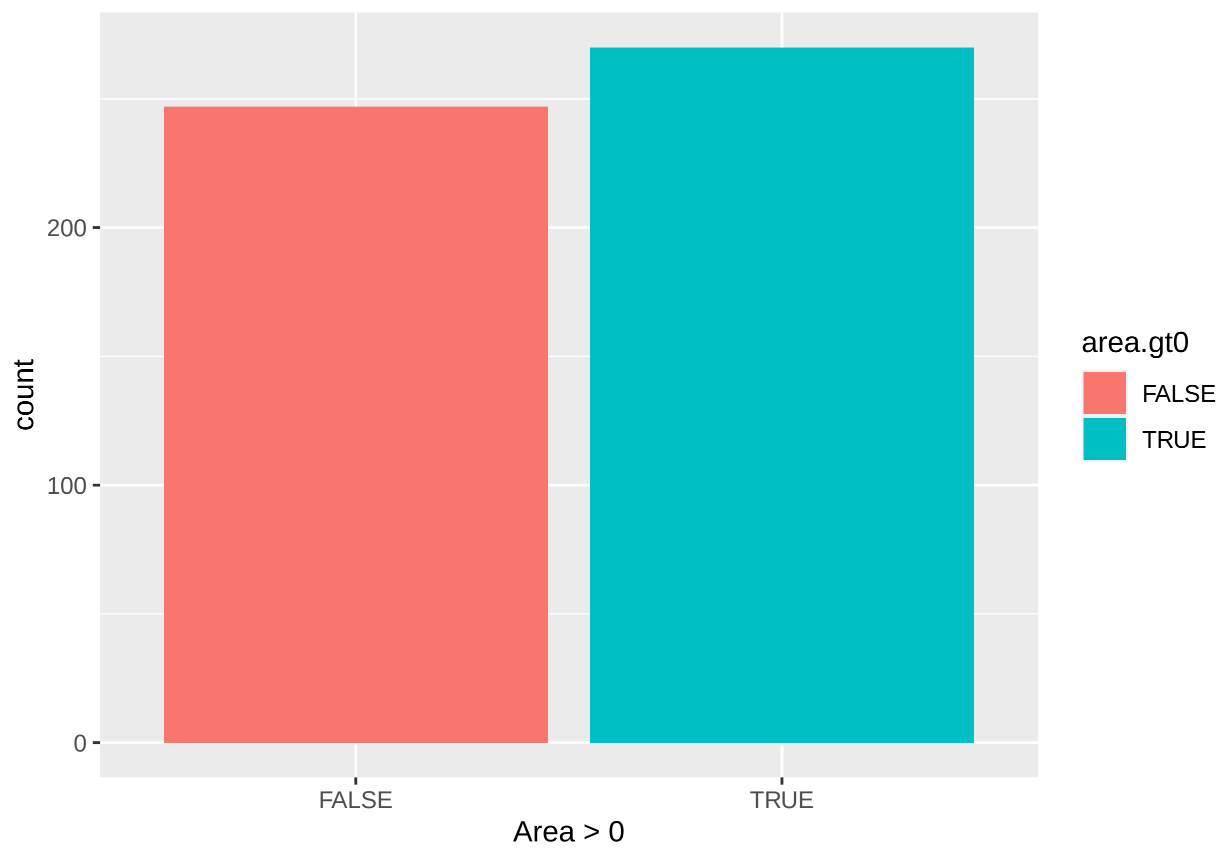
One important aspect of this dataset, is the distribution of the outcome variable: it appears to be highly positively skewed and therefore it is advisable, in order to perform linear regression, to log-transform the values. In the next plots we see that after applying $log(x + 1)$ function, the response variable presents a more gaussian-shaped distribution.

```r
parea <- ggplot(data = fires) +
  geom_histogram(aes(area), binwidth = 30) +
  xlab("Area (ha)")
  # coord_cartesian(ylim = c(0, 50))

parea_log <- ggplot(data = fires) +
  geom_histogram(aes(log(area + 1)), binwidth = 0.2) +
  xlab("Log-area")
  # coord_cartesian(xlim = c(0, 10))

grid.arrange(parea, parea_log, ncol = 2)
```



Of course we can also notice that the occurrence of zero values is very high, constituting almost half of the entire dataset. According to the reference paper, zero value means that an area lower than $1ha/100 = 100m^2$ was burned.

```r
fires %>%
  mutate(area.gt0 = area > 0) %>% # greater than 0 binary variable
  ggplot(aes(area.gt0)) +
  geom_bar(aes(fill = area.gt0)) +
  xlab("Area > 0")
```



This issue will be covered - and tackled - in one of the next sections, experimenting the *two-part model* proposed by Fletcher, MacKenzie, and Villouta (2005). For the time being, we will consider the normal assumption as in (2007).

4 Linear Regression

As the first and main regression model, we introduce the general linear model.

Given a matrix of explanatory variables $X = (X_1, ...X_n,)$ where X_i represents the row vector of predictors for the i^{th} observation, we want to explain the probabilistic behavior of the quantitative responses $y_1, ..., y_n$, considered as realizations of normal random variables $Y = Y_1, ..., Y_n$, in terms of X.

More specifically, the goal is to estimate the coefficients of the following linear combination.

$$Y = X\beta + \epsilon$$

where X is a matrix $n \times p$, with $p = \#predictors + 1$ since a column of 1's is commonly added to the matrix in order to concisely represent the intercept of the regression line. The $\beta = (\beta_0, ..., \beta_{p-1})'$ column vector is a set of parameters, usually unknown, main objective of the statistical inference, whereas the $\epsilon = (\epsilon_1, ..., \epsilon_n)'$ column vector is a set of unobservable random variables (errors) which account for natural variability, measurement error and other sources of uncertainty. In general, such vector is under the normal assumption, meaning that it is considered as drawn from a multivariate normal distribution. Formally:

$$\epsilon \sim \mathcal{N}_n(\underline{0}, \sigma^2 I_{n\times n})$$

which represents a situation of i.i.d. errors added to the signal $X\beta$. It then follows that also Y is a normal random vector, with each element having the same variance (homoscedasticity).

$$Y = X\beta + \epsilon \sim \mathcal{N}_n(X\beta, \sigma^2 I_{n\times n})$$

4.1 Ordinary Least Square Estimates (OLS)

One way to estimate the values of the β coefficients, is to adopt the Ordinary Least Square (OLS) estimate. It is a closed-form solution to the following minimization problem:

$$\min_{\beta} ||y - \mu||^2$$

This comes from the fact that we want to maximize the likelihood $\mathcal{L}(\theta|Y)$ which is, in our case, a function of β, i.e.

$$\mathcal{L}(\beta|Y) \propto \exp(-\frac{1}{2\sigma^2}||Y - X\beta||^2)$$

Therefore, assuming that $X'X$ is an invertible matrix, the formula for the OLS estimator is

$$\hat{\beta} = (X'X)^{-1}X'Y$$

It is immediate to notice that, since $\hat{\beta}$ is a linear transformation of Y, it is a normal random variable itself, of mean β and var-cov matrix $\sigma^2(X'X)^{-1}$.

In the following sections, having ascertained that the y vector is the vector of the outcome variables (area), realizations of Y, we compute the OLS estimate selecting different subsets of variables as the X matrix, evaluating the performances and then making inference on the validity of the model.

4.2 The null and complete models

In the first place, it is reasonable to test the null model, just to have a benchmark for the performance of the models. This means that we consider the X matrix as a single column vector of 1's.

If we compute the least square estimate, we find:

$$\hat{\beta} = (X'X)^{-1}X'y = \frac{1}{n}\sum_{i=1}^{n} y_i$$

This is also known as *naive predictor* since for each observation, the outcome will be the sample mean of the response variables, regardless of the values of the explanatory variables (if any).

Recall that in order to have a normal-like distribution of the area, we first have to log-transform it.

```r
naive.lm <- lm(log(area + 1) ~ 1, fires)
summary(naive.lm)
```

```
##
## Call:
## lm(formula = log(area + 1) ~ 1, data = fires)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.1110 -1.1110 -0.6923  0.9132  5.8846
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.1110     0.0615   18.07   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.398 on 516 degrees of freedom
```

The `summary()` extractor gives us some insights on this model. First of all we can see some points of the residuals distribution. The residual vector is $e = Y - \hat{Y}$ and measures the deviation of the predictions from the ground-truth values. Then we see that the value of the estimated parameter $\hat{\beta}_0$ (the intercept) is 1.1110, which is, as mentioned before, the sample mean of all the area values (in the log space). We can quickly check it:

```
near(naive.lm$coefficients[[1]], mean(log(1 + fires$area)))
```

```
## [1] TRUE
```

The standard error is computed as $SE(\hat{\beta}_i) = \hat{\sigma}\sqrt{(X'X)^{-1}_{i+1,i+1}}$ where $\hat{\sigma}$ is the residual standard error (in R, `sd(naive.lm$residuals)`, also shown at the bottom of the summary output). It allows us to construct confidence intervals on the true parameter β_i, but this holds only if we can assume that β is normally distributed. Having this in mind, we can easily compute 95% confidence interval, obtaining the following.

```
confint(naive.lm)
```

```
##                 2.5 %   97.5 %
## (Intercept) 0.9901984 1.231853
```

Finally, the t-value is the ratio $\hat{\beta}_i / SE(\hat{\beta}_i)$ and, along with the corresponding p-value, represents the outcome of an hypothesis test where we want to know if we can reject the null hypothesis $H_0 : \beta_i = 0$ in favor of the alternative hypothesis $H_1 : \beta_i \neq 0$. Therefore a large p-value would tell us that, being the true value β_i equal to 0 with a given confidence level, the i^{th} predictor has no effect on the response variable, and, presumably, we should drop it from the model. Of course, in this trivial example (null model) we have no reason to speculate on the confidence intervals being the intercept the only parameter considered.

On the opposite side of the null model, we can consider the complete model in which we use all the provided attributes.

```
complete.lm <- lm(log(area + 1) ~ ., fires)
complete.lm$call
```

```
## lm(formula = log(area + 1) ~ ., data = fires)
```

We omit the summary of this model since it will display too many coefficients all at once and, moreover, it could be very difficult to interpret.

Beware that when using all the predictors, the `lm()` call is implicitly performing some operations not explained before. This time, since we also have qualitative predictors such as month, day, location etc., the number of parameters to be estimated is different: the factors are split into separate dummy variables which account for the presence of a certain factor level. However the number of coefficients to be estimated for each factor is equal to the number of levels *minus one* since otherwise the columns related to that factor would be linearly dependent and, as a consequence, the $X'X$ matrix would not be invertible. To make an example, the following are the coefficients estimated by the complete model related to the factor *day*. Monday, the first level, is used as reference.

```
complete.lm$coefficients[grepl("day", names(coef(complete.lm)))]
```

```
##      daytue      daywed      daythu      dayfri      daysat      daysun
##  0.148855217  0.003965977 -0.042352484 -0.062776216  0.114421424  0.170850707
```

4.3 Metrics

Before generating other models, we define two metrics based on which we will compare the models. The first one is the *root mean squared error* (RMSE).

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Notice that, although sometimes the term mean squared error is used to refer to the unbiased estimate of error variance, for this work we consider the biased, yet consistent version just defined. They only differ in the denominator: instead of the cardinality of the observations, the residual sum of squares (RSS) is divided by the number of degrees of freedom, which is $n - p$ with p being the number of estimated parameters.

```r
# notice, MSE is implemented with exact mean and not sum/(n-p)
mse <- function(predicted, actual) {
  mse <- mean((actual - predicted) ^ 2)
  return(mse)
}

rmse <- function(predicted, actual) {
  mse <- mse(predicted, actual)
  rmse <- sqrt(mse)
  return(rmse)
}
```

The second metric considered is the *mean absolute deviation* (MAD), defined as follows:

$$MAD = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

```r
mad <- function(predicted, actual) {
  mad <- mean(abs(actual - predicted))
  return(mad)
}
```

In both metrics, lower values result in better predictive models. However, the RMSE is more sensitive to high residuals.

In the examples above, null and complete models, the computation of these metrics gives us the following results (after applying the inverse of the log transformation and setting to 0 all the negative values that may occur in the prediction):

```r
naive.predicted <- exp(predict(naive.lm, fires, type = "response")) - 1
naive.predicted[naive.predicted < 0] <- 0.
naive.rmse <- round(rmse(naive.predicted, fires$area), digits = 2)
naive.mad <- round(mad(naive.predicted, fires$area), digits = 2)

complete.predicted <- exp(predict(complete.lm, fires, type = "response")) - 1
complete.predicted[complete.predicted < 0] <- 0.
complete.rmse <- round(rmse(complete.predicted, fires$area), digits = 2)
complete.mad <- round(mad(complete.predicted, fires$area), digits = 2)

res.df <- data.frame("RMSE" = c(naive.rmse, complete.rmse),
           "MAD" = c(naive.mad, complete.mad))
row.names(res.df) <- c("naive.lm", "complete.lm")
as.matrix(res.df)
```

```
##                RMSE   MAD
## naive.lm     64.51 12.98
## complete.lm 63.59 12.11
```

The complete model performs better than the null one, although the scores are quite close. These results are analysed more in detail in the next section.

4.4 Model selection

As already anticipated in the introduction, the focus of the analysis is also to select the most relevant subset of features, finding smaller models which can still perform well while generalizing to unseen data.

4.4.1 ANOVA, Global F-test and AIC

Apart from comparing some statistics on the responses of the models (like, indeed, RMSE and MAD) we could also perform some tests. One is the *ANOVA test*: it allows to compare a big model to a smaller nested model (as, for instance, our complete and null models).

```
print(anova(naive.lm, complete.lm))
```

```
## Analysis of Variance Table
##
## Model 1: log(area + 1) ~ 1
## Model 2: log(area + 1) ~ xy + month + day + FFMC + DMC + DC + ISI + temp +
##     RH + wind + rain
##   Res.Df     RSS Df Sum of Sq     F  Pr(>F)
## 1    516 1009.10
## 2    456  842.03 60    167.07 1.508 0.01147 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A low p-value for the F statistic means that we can reject the hypothesis that the smaller model explains the data well enough. In this case the p-value is low, but probably not quite as much as we would hope. In addition to that, we would point out that the F statistic here coincides with the statistic computed with an F-test on the complete model. In fact, the global F-test (*all-or-nothing* test), which is in general one of the outputs of a `summary(lm)` call, always compares one "big" model to the null model. It does so by testing the null hypothesis $H_0 : \beta_1 = \beta_2 = ... = \beta_{p-1} = 0$ (null model holds) which would say that the true mean of $Y = X\beta$ actually belongs to the smaller subspace $span((1, ..., 1)')$. Briefly, the test is performed knowing that, based on Cochran's theorem, under the null hypothesis, a specific computable statistic of the residuals, belongs to a Fisher-Snedecor distribution of known parameters. The value of such statistic here is 1.508 which coincides, indeed, with the F-statistic computed with the following R code.

```
sum <- summary(complete.lm)
sum$fstatistic["value"]
```

```
##   value
## 1.50795
```

Up to now it seems that the complete model better explains the response, but it is also true that the predictors used are many more than the null model's. Another criterion used to compare two models, is the Akaike's Information Criterion (AIC) which also takes into account the number of estimated parameters, penalyzing models with many coefficients. Higher AIC score means better model. In fact, the AIC computed for the two models is

```
as.matrix(AIC(naive.lm, complete.lm))
```

```
##              df      AIC
## naive.lm      2 1816.938
## complete.lm 62 1843.361
```

showing that the better performances cannot completely justify the high number of estimated parameters.

4.5 Smaller models

We now investigate on the selection of subsets of attributes on which to train new models. Based on the study made by Cortez and Morais (2007), we select four subsets of predictors:

- **STFWI**: spatial, temporal and the four FWI indices
- **STM**: spatial, temporal and the four weather variables
- **FWI**: only the four FWI indices
- **M**: only the four weather variables

For each subset, we fit a new linear model, and compute the two metrics defined before and the AIC score. We include also the null and complete models to have a global overview.

```
stfwi.lm <- lm(log(area + 1) ~ xy + month + day + FFMC + DMC + DC + ISI, fires)
stfwi.predicted <- exp(predict(stfwi.lm, fires, type = "response")) - 1
stfwi.predicted[stfwi.predicted < 0] <- 0.
stfwi.rmse <- round(rmse(stfwi.predicted, fires$area), digits = 2)
stfwi.mad <- round(mad(stfwi.predicted, fires$area), digits = 2)

stm.lm <- lm(log(area + 1) ~ xy + month + day + temp + RH + wind + rain, fires)
stm.predicted <- exp(predict(stm.lm, fires, type = "response")) - 1
stm.predicted[stm.predicted < 0] <- 0.
stm.rmse <- round(rmse(stm.predicted, fires$area), digits = 2)
stm.mad <- round(mad(stm.predicted, fires$area), digits = 2)

fwi.lm <- lm(log(area + 1) ~ FFMC + DMC + DC + ISI, fires)
fwi.predicted <- exp(predict(fwi.lm, fires, type = "response")) - 1
fwi.predicted[fwi.predicted < 0] <- 0.
fwi.rmse <- round(rmse(fwi.predicted, fires$area), digits = 2)
fwi.mad <- round(mad(fwi.predicted, fires$area), digits = 2)

m.lm <- lm(log(area + 1) ~ temp + RH + wind + rain, fires)
m.predicted <- exp(predict(m.lm, fires, type = "response")) - 1
m.predicted[m.predicted < 0] <- 0.
m.rmse <- round(rmse(m.predicted, fires$area), digits = 2)
m.mad <- round(mad(m.predicted, fires$area), digits = 2)

res.df <- AIC(naive.lm, complete.lm, stfwi.lm, stm.lm, fwi.lm, m.lm)
res.df["RMSE"] <- c(naive.rmse, complete.rmse, stfwi.rmse, stm.rmse, fwi.rmse, m.rmse)
res.df["MAD"] <- c(naive.mad, complete.mad, stfwi.mad, stm.mad, fwi.mad, m.mad)
as.matrix(res.df)
```

```
##             df      AIC  RMSE   MAD
## naive.lm     2 1816.938 64.51 12.98
## complete.lm 62 1843.361 63.59 12.11
## stfwi.lm    58 1840.663 63.66 12.12
## stm.lm      58 1840.897 63.65 12.11
## fwi.lm       6 1820.762 64.48 12.96
## m.lm         6 1819.532 64.47 12.97
```

The first two new subsets show scores which are comparable to the complete model ones, while the FWI and M models are slightly better than the naive predictor. Probably, the spatio-temporal predictors indirectly provide useful information about the flora present in a certain location and the period in which tourism is more intense, and this could certainly be of some relevance when analyzing forest fires.

Finally, for one last confirmation, we run the ANOVA test, this time comparing the complete model to the STM model.

```
print(anova(stm.lm, complete.lm))
```

```
## Analysis of Variance Table
##
## Model 1: log(area + 1) ~ xy + month + day + temp + RH + wind + rain
## Model 2: log(area + 1) ~ xy + month + day + FFMC + DMC + DC + ISI + temp +
##     RH + wind + rain
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    460 851.09
## 2    456 842.03  4     9.064 1.2271 0.2985
```

The F-statistic here is quite high, therefore we consider the STM predictors enough with respect to all the predictors.

4.6 Two-part model

With the analysis carried out in the previous sections we were able to get some improvement on the null model. However, it is clear that those models are not able to explain well the response variable. Inspired by the work of Fletcher, MacKenzie, and Villouta (2005), here we implement and test the so-called “Two-part model”, which is particularly suitable when the response variable has a positively skewed distribution with many zeros, like our case.

The idea is to create two new datasets: in the first one, the outcome variable will be a binary value representing the presence of an area greater than zero (referred to as “presence data”), whereas in the second one, the outcome will be the logarithm of the original area value (referred to as “log-abundance”). Notice that the second dataset will be of a smaller size since it will contain only the observations where area is greater than zero.

Following this rationale, assuming that the abundance is lognormally distributed, the regression task is performed by a conditional model, composed by a logistic regression model and an ordinary linear regression model.

Before further specifications on this approach, we first briefly introduce logistic regression.

4.6.1 Logistic Regression

After general linear models were discovered (quantitative normal response with qualitative predictors), further generalization was reached with not-normal response. The aim was to keep the basis of a linear regression model while predicting a response which is not drawn from a normal distribution but, for instance, from a Binomial distribution family.

Similarly to the ordinary linear regression model, we still have a response vector $Y = (Y_1, ..., Y_n)$ of independent random variables with means $\mu_i = \mathbb{E}(Y_i)$, whose realizations y_i represent our data. The linear combination of predictors remains the same: $\eta = X\beta$. The difference with respect to linear regression is the presence of a link function which connects the means μ_i to the linear combination:

$$g(\mu_i) = \eta_i$$

The choice of the g function depends on the family of the Y distribution. In our case, if we denote with Z_i the presence of an area greater than zero for the i^{th} observation, then it clearly belongs to a Bernoulli distribution: $Z_i \sim Bernoulli(p_i)$.

For this distribution (and, more generally, for the Binomial family), the most common choice is the *logit link* which "stretches" the interval $(0,1)$ into the whole real line, so that it makes sense to apply the linear combination:

$$logit(\mu_i) = \log \frac{\mu_i}{1-\mu_i} = \sum_{j=0}^{p-1} \beta_i x_{i+1j}$$

If we then apply the inverse of the logit function (called, indeed, the *logistic function*), we obtain a value in $(0,1)$ which will be the objective of our prediction, *i.e.* $\mu_i = \mathbb{P}(Z_i = 1) = g^{-1}(\eta_i)$

```r
# code for plotting the logistic function
lin.comb <- seq(-15,15, by = 0.001)
ggplot() +
  geom_line(mapping = aes(lin.comb, exp(lin.comb)/(1+exp(lin.comb)))) +
  geom_line(aes(seq(-15, 15, by = 0.01), 1), linetype = "dashed") +
  ylim(c(0, 1.3))
```

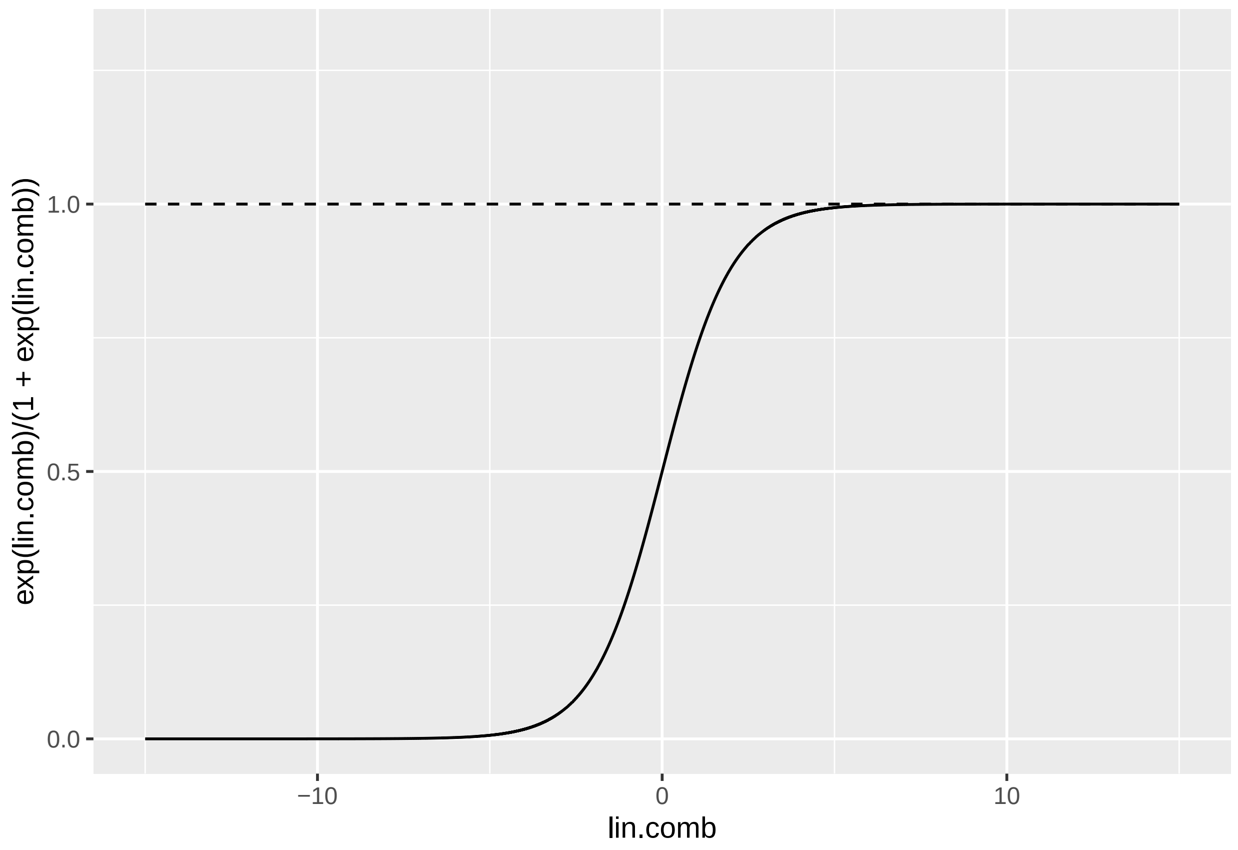


Figure 1: Plot of logistic function

4.6.2 Applying the two-part model

If we denote with Y the log-abundance random variable previously mentioned, and with Z the presence r.v., then according to the conditional two-part model, the expected value $\mathbb{E}(Y)$ is given by:

$$\begin{aligned}\mathbb{E}(Y_i) &= \mathbb{P}(Z_i = 1)\mathbb{E}(Y_i|Z_i = 1) + \mathbb{P}(Z_i = 0)\mathbb{E}(Y_i|Z_i = 0)\\ &= \mathbb{P}(Z_i = 1)\mathbb{E}(Y_i|Z_i = 1)\\ &= \pi_i\mu_i\end{aligned}$$

We obtain the estimates $\hat{\pi}_i$ and $\hat{\mu}_i$ after fitting, respectively, the logistic regression and the ordinary linear regression models.

For this purpose, however, we have to point out that categorical predictors can be used only if each level to them associated is present in the second (smaller) dataset, otherwise it would not be possible, for the linear regression model, to predict a response for categories not seen in the fitted data. For this reason, here we apply the two-phase model with only two subsets of features FWI and M, defined in section Smaller models.

To proceed, we first create the two datasets.

```r
# presence dataset
fires.bin <- fires %>%
  mutate(area.gt0 = factor(area > 0)) %>%
  select(-area)
# area abundance dataset
fires.ab <- fires %>%
  filter(area > 0)

print("Presence data (area.gt0)")
summary(fires.bin$area.gt0)
print("Log-abundance data (log-area)")
summary(log(fires.ab$area + 1))
```

```
## [1] "Presence data (area.gt0)"
## FALSE  TRUE
##   247   270
## [1] "Log-abundance data (log-area)"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.08618 1.14422 1.99742 2.12741 2.79865 6.99562
```

Then we fit the R `lm()` as before and also the `glm()` specifying the logit link function

```r
m.bin.glm <- glm(area.gt0 ~ temp + RH + wind + rain, family = binomial(link = "logit"), data = fires.bin

m.ab.lm <- lm(log(area +1) ~ temp + RH + wind + rain, data = fires.ab)

fwi.bin.glm <- glm(area.gt0 ~ FFMC + DMC + DC + ISI, family = binomial(link = "logit"), fires.bin)

fwi.ab.lm <- lm(log(area +1) ~ FFMC + DMC + DC + ISI, fires.ab)
```

Finally we compute the predictions in the way described at the beginning of the section.

```r
# compute E(area) = Pr(Z = 1)E(Y | Z = 1)
pred.fwi.bin <- predict(fwi.bin.glm, newdata = fires, type = "response")
pred.fwi.ab <- exp(predict(fwi.ab.lm, newdata = fires, type = "response")) - 1

pred.m.bin <- predict(m.bin.glm, newdata = fires, type = "response")
pred.m.ab <- exp(predict(m.ab.lm, newdata = fires, type = "response")) - 1

pred.fwi.cond <- pred.fwi.bin * pred.fwi.ab
pred.fwi.cond[pred.fwi.cond < 0] <- 0.
fwi.cond.rmse <- round(rmse(pred.fwi.cond, fires$area), digits = 2)
fwi.cond.mad <- round(mad(pred.fwi.cond, fires$area), digits = 2)

pred.m.cond <- pred.m.bin * pred.m.ab
pred.m.cond[pred.m.cond < 0] <- 0.
m.cond.rmse <- round(rmse(pred.m.cond, fires$area), digits = 2)
m.cond.mad <- round(mad(pred.m.cond, fires$area), digits = 2)

res.df <- data.frame("RMSE" = c(fwi.cond.rmse, m.cond.rmse),
           "MAD" = c(fwi.cond.mad, m.cond.mad))
row.names(res.df) <- c("FWI", "M")
as.matrix(res.df)
```

```
##      RMSE   MAD
## FWI 64.17 13.52
## M   64.17 13.49
```

Unfortunately, this method seems to worsen the performance. This may be due to various factors, as for instance the small size of the dataset and the fact that a value of 0 doesn't actually mean that the burned area is null, but rather that the area is smaller than a predefined threshold. Therefore this conditional model could be hardly suitable for this dataset, even though the area distribution made us think, in the first place, that this could have been the case.

Without going into more experiments on that (since we already know we won't reach outstanding results anyway) it could be interesting to notice that the two-part model allows different sets of explanatory variables to be used to model the two components, thereby leading to a better understanding of the system under study. E.g. we may be induced to think that the M subset of predictors could be useful to predict the presence of a medium-large forest fire, but not much its abundance, while the FWI indices could help better in predicting the extent of the burned area.

5 Random Forest regression

To conclude the analysis, since we could not reach satisfactory results with almost any of the previous probabilistic models, we briefly attempt to improve the results with a decision-rule based regression algorithm, namely *Random Forest* - ironically, on a forest fires dataset - .

In few words, a random forest model is an ensemble model composed by T regression trees, each of which predicts an outcome based on a subset of the provided explanatory variables. The final prediction of the random forest is then simply the average over all T predictions.

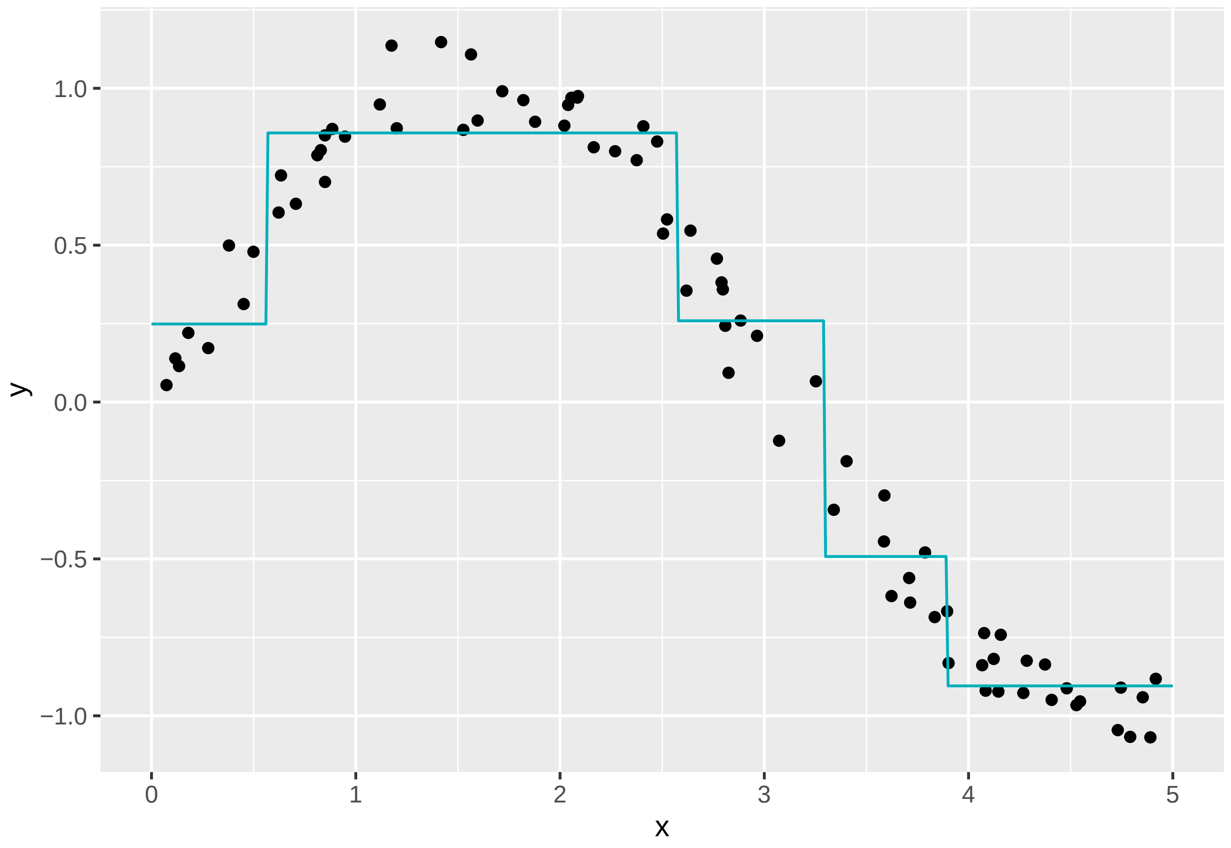
5.1 Decision tree as regressor

A regression tree works similarly to a decision tree used for classification. The main difference is that, instead of finding a step function separating two or more classes of observations, it finds the step function that better fits the datapoints.

As an example, here we train a regression tree to fit the sine function.

```
set.seed(404)

x <- runif(80) * 5
y <- sin(x) + rnorm(length(x), sd = 0.1)
ex.data <- data.frame(x = x, y = y)
ex.task <- TaskRegr$new(id = "example", backend = ex.data, target = "y")

rt <- lrn("regr.rpart")
rt$train(ex.task)
```

```r
test.x <- seq(0, 5, by = 0.01)
test.y <- sin(test.x)
new.data <- data.frame(x = test.x, y = test.y)
test.task <- TaskRegr$new(id = "example.test", backend = new.data, target = "y")
ggplot() +
  geom_point(aes(x, y)) +
  geom_line(aes(new.data$x, rt$predict(test.task)$response), color = "#00AFBB")
```



Instead of splitting the nodes based on an impurity index, such as the GINI index, the regression tree aims at minimizing the residual sum of squares (RSS) which accounts for the variance of the data under a certain splitting rule. The lower the variance, the better the split since the data points are closer to the predicted value (average).

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

The construction of the tree is made in the following way:

1. For each attribute and for each split of the data on that attribute, predict the target value as the average of the target data in the split and, with those predictions, compute the residual sum of squares. Lower RSS means more accurate predictions after that split.
2. Choose the attribute with the lowest minimum RSS value and split the data based on this rule.
3. Repeat the process starting from (1) until a predefined maximum depth is reached or the RSS for each node is under a certain threshold (other termination criterions could be followed).

5.1.1 Applying random forest regression model

Now, in a random forest algorithm, T decision trees are trained, each one on a bootstrap sample and, at each node, a random subset of m candidate predictors from all the available ones.

The idea behind the random forest algorithm, is that many "weak learners" will produce a "strong learner" by averaging their outcomes (in the classification setting, rather than an average, a major voting approach is adopted).

To proceed with our forest fire dataset, we first create a `mlr3` regression task on the dataset (to which we add the column of log-transformed area) and we train a random forest learner (code-named `ranger` in R) and we compute the metrics score on the predictions. Regarding the hyperparameters, we use

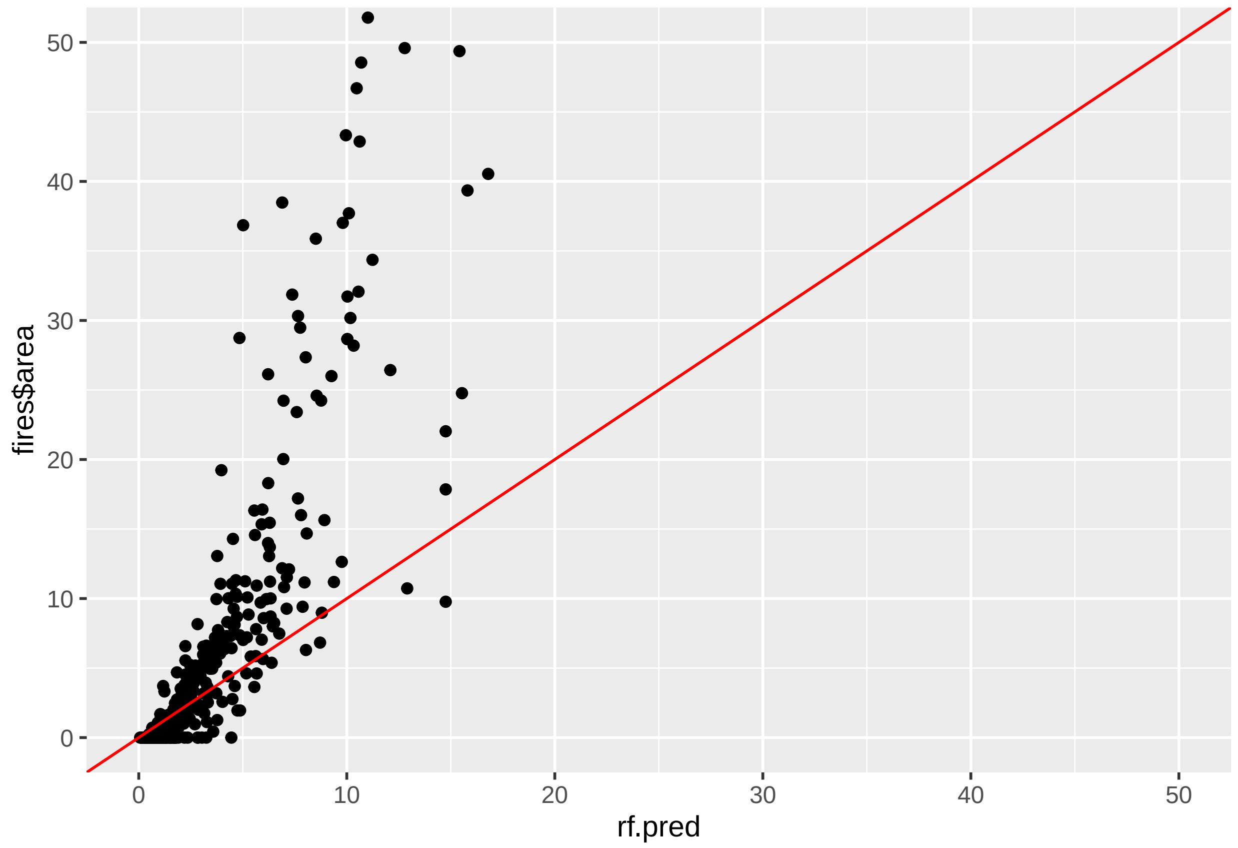
- $T = 500$, default parameter
- splitrule = `"variance"`

As for the other parameters, m and maximum depth are initially set, respectively, to 4 and 20 (a common choice for m in regression with random forest is, in fact, $p/3$ where p is the total number of predictors).

```r
fires <- fires.raw %>%
  mutate(X = factor(paste(X,Y, sep = "")), log_area = log(area + 1)) %>%
  rename(xy = X) %>%
  select(-c(Y))

# set up the task using log_area as target variable
task <- TaskRegr$new(id = "fires", backend = fires[, -12], target = "log_area")

rf <- lrn("regr.ranger", num.trees = 500, mtry = 4, splitrule = "variance", max.depth = 20)
rf$train(task)
rf.log.pred <- rf$predict(task)

rf.pred <- exp(rf.log.pred$response) - 1
rf.pred[rf.pred < 0] <- 0.

paste("RMSE:", round(rmse(rf.pred, fires$area), digits = 2))
paste("MAD:", round(mad(rf.pred, fires$area), digits = 2))
```

```
## [1] "RMSE: 59.43"
## [1] "MAD: 10.18"
```

```r
ggplot() +
  geom_point(mapping = aes(rf.pred, fires$area)) +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  coord_cartesian(xlim = c(0, 50), ylim = c(0, 50))
```



The results are quite higher than those obtained with linear regression. However it is very likely that the decision trees are overfitting the data. In fact, with a maximum depth of 20, the trees are capable of generating many splits, resulting in models with poor generalization to unseen data.

To overcome this issue, we tune the m and maximum depth parameters with a 10-fold cross validation approach, finding the configuration of the two parameters that generates lowest RMSE. After that, we re-train a random forest ensemble on the entire dataset with those parameters and print the RMSE and MAD scores.

```
learner <- lrn("regr.ranger", num.trees = 500, splitrule = "variance")

design = data.table(expand.grid(max.depth = seq(2, 15, by = 2),
                                mtry = 3:11))

tune_ps <- ParamSet$new(list(
  ParamInt$new("max.depth", lower = 1, upper = 20),
  ParamDbl$new("mtry", lower = 1, upper = 11)
))

tuner = tnr("design_points", design = design)

instance = TuningInstance$new(
  task = task,
  learner =  learner,
  resampling = rsmp("cv", folds = 10),
  measures = msr("regr.rmse"),
  param_set = tune_ps,
  terminator = term("none")
)

tuner$tune(instance)
```

```
print("Best params")
instance$result$params
```

```
## [1] "Best params"
## $num.trees
## [1] 500
##
## $splitrule
## [1] "variance"
##
## $max.depth
## [1] 2
##
## $mtry
## [1] 3
```

```
learner$param_set$values <- instance$result$params
learner$train(task)
rf.log.pred <- learner$predict(task)

rf.pred <- exp(rf.log.pred$response) - 1
rf.pred[rf.pred < 0] <- 0.

paste("RMSE:", round(rmse(rf.pred, fires$area), digits = 2))
paste("MAD:", round(mad(rf.pred, fires$area), digits = 2))
```

```
## [1] "RMSE: 64.42"
## [1] "MAD: 12.84"
```

The results are now more reliable but comparable with those of the linear regression analysis, showing that, even though decision trees can better fit the data, they cannot generalize well with the dataset under analysis.

6 Conclusion

There are many ways to perform regression tasks, and sometimes linear regression could not be enough to reach high prediction accuracy. However, with linear models, we can extract not only predictions, but also, and most importantly, information about the probabilistic behavior of the response variable when certain explanatory variables are available.

Regarding the results obtained on the dataset under analysis, as argued in MALARZ, KACZANOWSKA, and KUŁAKOWSKI (2002), predicting the size of forest fires is a challenging task. To improve it, the authors Cortez and Morais (2007) believe that *"additional information (not available in this study) is required, such as the type of vegetation and firefighting intervention"*. Despite that, as can also be seen from the *predicted vs truth* scatterplot, the last proposed solution is able to predict quite accurately small fires which constitute the majority of the occurrences, and this could still be of some help since, for instance, when small fires are predicted, then air tankers could be spared and small ground crews could be sent.

References

Cortez, Paulo, and A. Morais. 2007. "A Data Mining Approach to Predict Forest Fires Using Meteorological Data," January.

Dua, Dheeru, and Casey Graff. 2017. "UCI Machine Learning Repository." University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml.

Fletcher, David, Darryl MacKenzie, and Eduardo Villouta. 2005. "Modelling Skewed Data with Many Zeros: A Simple Approach Combining Ordinary and Logistic Regression." *Environmental and Ecological Statistics* 12 (March): 45–54. https://doi.org/10.1007/s10651-005-6817-1.

MALARZ, K., S. KACZANOWSKA, and K. KUŁAKOWSKI. 2002. "Are Forest Fires Predictable?" *International Journal of Modern Physics C* 13 (08): 1017–31. https://doi.org/10.1142/S0129183102003760.