# Predicting Organic Carbon in European Soils

Marcus Schmidt

23/02/2021

# (I) INTRODUCTION

## (I a) Background, goal & data set

Soils are the basis of all agriculture. Organic carbon and its dynamics play a large role carbon sequestration and therefore help regulate our climate. The goal of this project is to predict soil organic carbon from a large set of European soil data. The data set is from the LUCAS initiative 2015 (https://esdac.jrc.ec.europa.eu/projects/lucas) and includes over 20,000 sampling points from the upper soil on land of various land-use types.

## (I b) Data set download

The data set was requested online from the LUCAS initiative and the author got permission to use it for this project. Out of the data set, I created a *.Rdata file to be downloaded here:

```
load(url("https://github.com/ms-soil/ds-submission-soil/raw/main/eusoil.Rdata"))
```

## (I c) Data set structure

Let's take a peak at the datset, showing the variables included, the dimension of the data set and a first look at the numbers and their variable type:

```
names(eusoil) # variable names
```

```
##  [1] "Clay"        "Sand"        "Silt"        "pH.CaCl2."   "pH.H2O."
##  [6] "EC"          "OC"          "CaCO3"       "P"           "N"
## [11] "K"           "Elevation"   "Soil_Stones" "LC1"         "LC1_Desc"
```

```
dim(eusoil) # data set dimensions
```

```
## [1] 21859    15
```

```r
as_tibble(eusoil) # overview of data structure
```

```
## # A tibble: 21,859 x 15
##     Clay  Sand  Silt pH.CaCl2. pH.H2O.    EC    OC CaCO3     P     N     K
##    <int> <int> <int>     <dbl>   <dbl> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1    NA    NA    NA       3.9    3.91  44.2  25.5     0  42.9   2.8  24.6
##  2    NA    NA    NA       3.1    3.91  46.4 504.      0 165.   19.9 460.
##  3    NA    NA    NA       4.9    5.48  15.8  51.4     0  26.9   4.3 173.
##  4    NA    NA    NA       3      3.76  26.9 470.      0 103.   16.1 313
##  5    10    46    44       3.9    4.04  28.4  43.1     1   6.3   2.3  38.6
##  6    14    36    50       4.2    4.41  41.8  32.4     0   7.5   3.3  48
##  7    18    35    46       4.9    5.13  32    21.1     1  12.4   2.1  36
##  8    14    36    50       4      4.16  72.4  53.2     0  52.1   4.2 158.
##  9    19    48    34       3.7    3.87  11.6  16       1   3.7   1    24.4
## 10     8    71    20       4      3.99  22.2  16       1   8.3   1.1  30
## # ... with 21,849 more rows, and 4 more variables: Elevation <int>,
## #   Soil_Stones <int>, LC1 <chr>, LC1_Desc <chr>
```

# (II) METHODS & ANALYSIS

## (II a) Data preparation

Analysis was done with R version 4.0.3.
Important variables for this data set to keep an eye on are:
- OC = organic carbon in mg C per gram soil
- N = total nitrogen in mg N per gram soil &
- Elevation in m.

Land use as a variable in the data set is not so straightforward. It is decoded in the LC1-variable, so we detect the capital letter in the variable character string and assign the respective land use from the data-set documentation, which can be found at https://github.com/ms-soil/ds-submission-soil (data-info.pdf). After this we remove the LC1-variable. We further check whether there are any observations where organic carbon (OC) is missing and take a look at how many observations we have for each land use. For further analysis, land use will be turned into a factor.

```r
eusoil <- eusoil %>% mutate(landuse = case_when(str_detect(LC1, "A") ~ "artif_land",
                                                str_detect(LC1, "B") ~ "cropland",
                                                str_detect(LC1, "C") ~ "woodland",
                                                str_detect(LC1, "D") ~ "shrubland",
                                                str_detect(LC1, "E") ~ "grassland",
                                                str_detect(LC1, "F") ~ "bare_land",
                                                str_detect(LC1, "G") ~ "water",
                                                str_detect(LC1, "H") ~ "wetland")) %>%
  select(-LC1)

table(is.na(eusoil$OC)) # organic carbon data is available in all observations
```

```
##
## FALSE
## 21859
```

Let's take a look at the distribution of samples on different land uses.

```r
# distribution of landuses
table(eusoil$landuse)
```

```
##
## artif_land  bare_land   cropland  grassland  shrubland      water    wetland
##         50        604       8972       4751        846          6         49
##   woodland
##       6581
```

```r
# factorize landuse
eusoil <- eusoil %>% mutate(landuse = as.factor(landuse))
```

## (II b) Variable selection

We want to use a complete data set for our predictions, so we exclude variables that are often missing in the data set. We can see it with the following code. It yields that we are missing the texture variables (sand, silt, clay) in over 80% of cases.

```r
# print availability for every variable
res <- for (i in 1:length(names(eusoil))) {
  print(table(!is.na(eusoil[,i])))
  print(names(eusoil[i]))
}
# texture (the silt, sand and clay variables) are only available in a fraction of cases
# see which fraction of the dataset misses texture
condition <- !is.na(eusoil$Clay)
table(condition)[[1]]/(table(condition)[[1]] + table(condition)[[2]])
# texture is missing in 80.5% of cases
```

Here is a bit of the result:

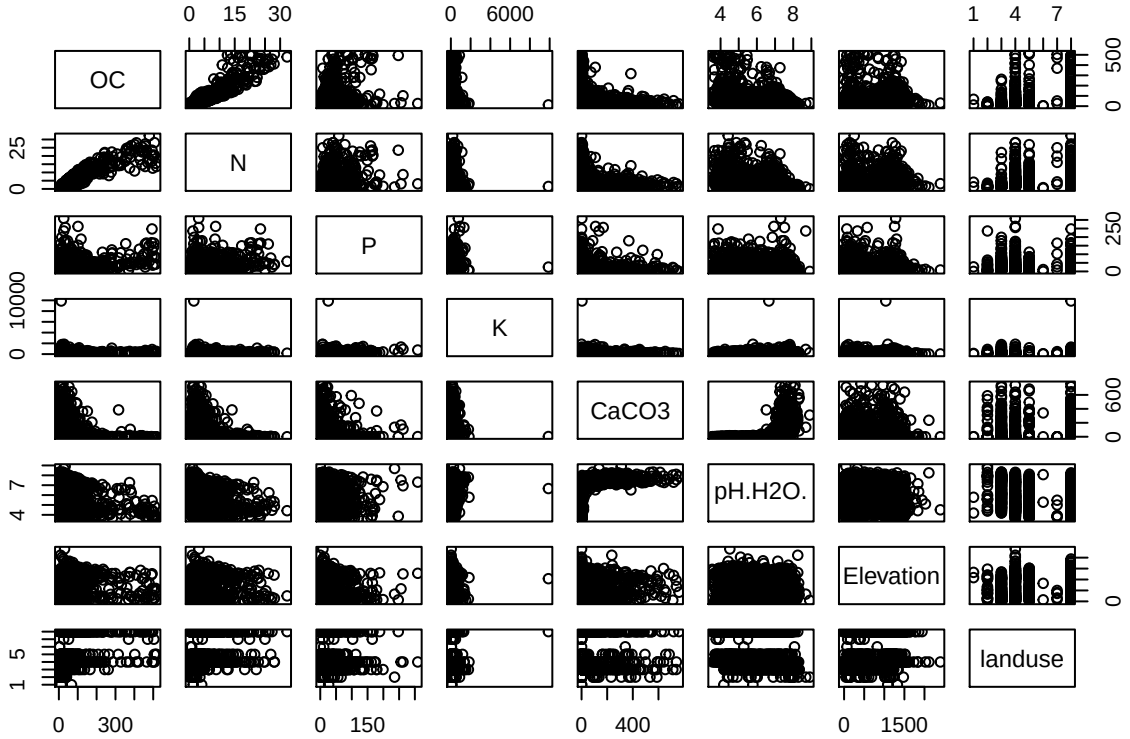
```r
# FALSE  TRUE
# 17599  4260
# [1] "Clay"
#
# FALSE  TRUE
# 17599  4260
# [1] "Sand"
#
# FALSE  TRUE
# 17599  4260
# [1] "Silt"
```

The texture variables clay, sand and silt are not often measured in this data set so we exclude them. We also exclude other variables that are not commonly measured by soil scientists or not clearly defined for this data set:

```r
eusoil <- eusoil %>% select(-Clay, -Sand, -Silt, -EC, -pH.CaCl2., -Soil_Stones, -LC1_Desc) %>%
  select(OC, N, P, K, CaCO3, pH.H2O., Elevation, landuse)
```

Which variables are left and how do they relate? We want to draw some conclusions of what to include by looking at a correlation plot.

```
#### variable selection III: influential variables ####
names(eusoil)
## a) numerical variables that correlate with organic carbon but not with each other
relationships <- plot(eusoil[1:2000,]) # plotting the first 2000 observations
```

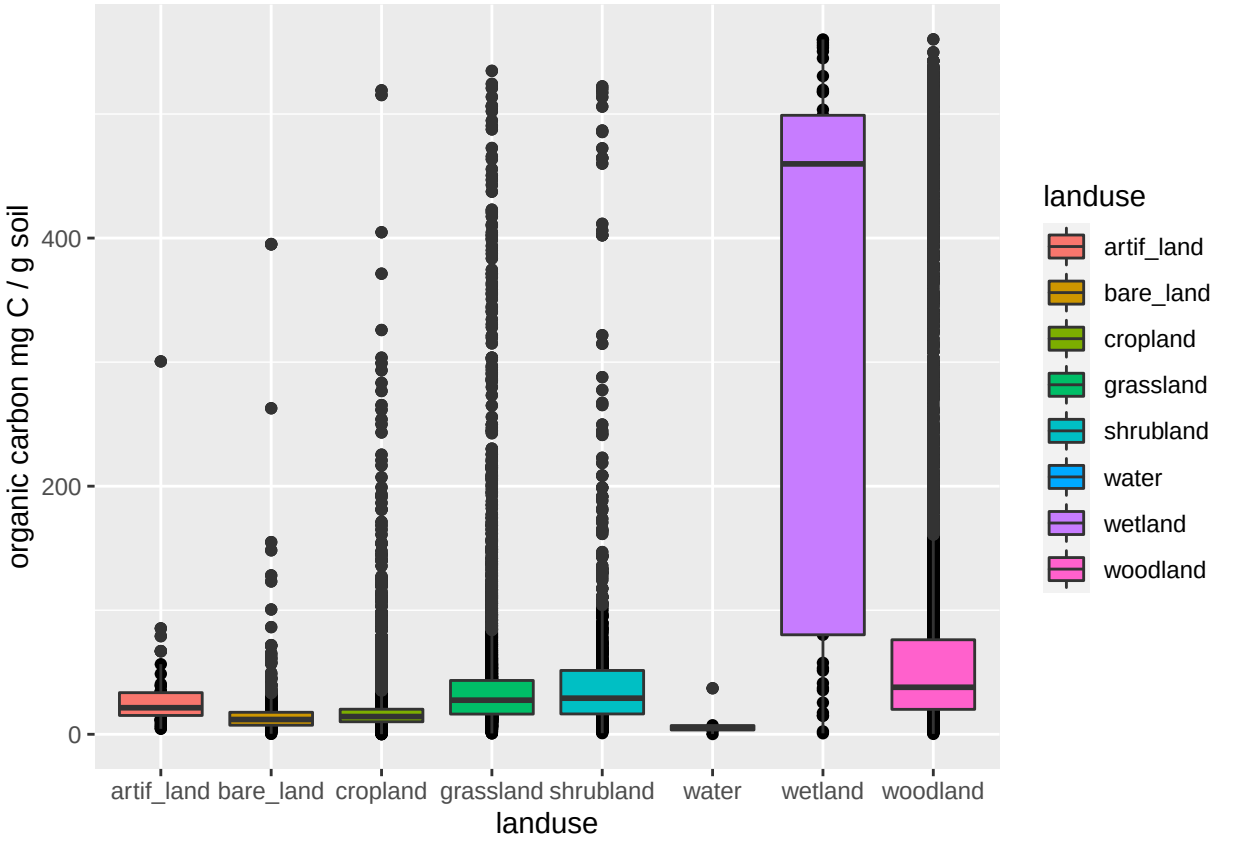



```
relationships
```

P (phosphorus) and K (potassium) show no clear relationship to organic carbon so we will exclude them soon in the code below. CaCO3 and pH are co-dependend so it will be enough to include one of them to represent both. At this point, we are keeping N (Nitrogen), pH and elevation in our analysis

There is also a categorical variable, land use, so we check whether it influences organic carbon (OC):

```
qplot(landuse, OC, data = eusoil, fill = landuse) + geom_boxplot() +
  ylab("organic carbon mg C / g soil")
```




Organic carbon differs with land use so we keep it as a factor. Now that the decision on the variables has been made, we give out some easier-to-type variable names and select the chosen variables from the data set.

```
eusoil <- eusoil %>% mutate(pH = pH.H2O., elev = Elevation) %>%
  select(OC, N, P, pH, elev, landuse)
```

## (II c) Data set division into validation, train and test set

First, from our data set, a validation set of 10% is taken which will only be used after deciding on a model. The validation set is important so that our model will not be fit to a specific data set. We keep a large proportion of the data for model training. The training set will be 80% and the test set 20% of the remaining data. We should choose a split that allows lots of training but still a representative test set. For reproducible results, we use the set.seed() function.

```
set.seed(1, sample.kind = "Rounding")
valindex <- createDataPartition(eusoil$OC, p = 0.1, list = F)
temp <- eusoil[-valindex,]
validation <- eusoil[valindex,]

set.seed(1, sample.kind = "Rounding")
testindex <- createDataPartition(temp$OC, p = 0.2, list = F)
trainset <- temp[-testindex,]
testset <- temp[testindex,]
```

```r
# see how many observations are in each
nrow(trainset); nrow(testset); nrow(validation)
```

```
## [1] 15735
```

```
## [1] 3937
```

```
## [1] 2187
```

The split retains 2000 obervations for the final validation set which should be plenty.

## (II d) RMSE function

A rooted mean square error (RMSE) function is used to evaluate the performance of the models that will be set up. It can be viewed as the typical error we make when predicting from a given model.

```r
RMSE <- function(observed_values, predicted_values){
  sqrt(mean((observed_values - predicted_values)^2))
}
```

## (II e) Reference model

A reference model can be useful to evaluate how different models perform compared to simply guessing the outcome. In guessing, we would take the average of the observed carbon values.

```r
#### guessing model ####
mu <- mean(trainset$OC)

rmse_mu <- RMSE(testset$OC, mu)
rmse_mu
```

```
## [1] 77.78776
```

```r
# create an RMSE table to always add the RMSEs
rmses <- data.frame(model = "mean", rmse = rmse_mu)
rmses
```

```
##   model     rmse
## 1  mean 77.78776
```

We see that if we guess, we are typically ~78 mg C / g soil off the observed value. We will improve this prediction, first with linear models but also with KNN and random forest algorithms.

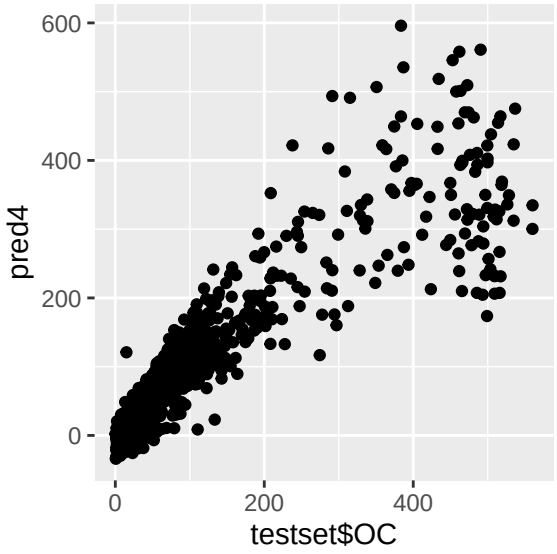
## (II f) Linear models

In the following, different linear models (LM) are set up, first with N (nitrogen) as predictor, then adding pH, elevation and landuse. The full code is present in the R script and not shown here as to not overload the report, However, we show the LM which includes all variables and the table of how RMSE impoved step by step when adding more variables.

```
#### lm with N, pH, elevation, landuse ####

m_all <- lm(OC ~ ., data = trainset)

base <- data.frame(testset)

pred4 <- predict(m_all, base)
head(pred4)
```

```
##         7        12        16        28        29        38
##  12.67697  72.41147  55.78709 188.79738  66.74575 168.52116
```

```
qplot(testset$OC, pred4)
```




```
rmse4 <- RMSE(testset$OC, pred4)
rmse4
```

```
## [1] 30.81398
```

```
rmses <- rbind(rmses,
               data.frame(model = "lm with all variables", rmse = rmse4))
rmses
```

```
##                    model     rmse
## 1                   mean 77.78776
## 2             lm with N 33.24694
## 3         lm with N + pH 32.60644
## 4 lm with N + pH + elev. 32.54013
## 5  lm with all variables 30.81398
```

## (II g) KNN models

Since all variables appeared to improve the LM model above, we also include these in a KNN model, which looks multidimensionally at the nearest neighbors of an observation and predicts from there.

A first try with KNN including elevation yields a less-than-optimal result (see RMSE table below). Another model which excludes elevation worked well. I could not immediately detect the reason for this, but my conclusion is that not every variable improves every type of model in the same way. Generally, KNN performed well after improving on the number of neighbors with a sapply() function.

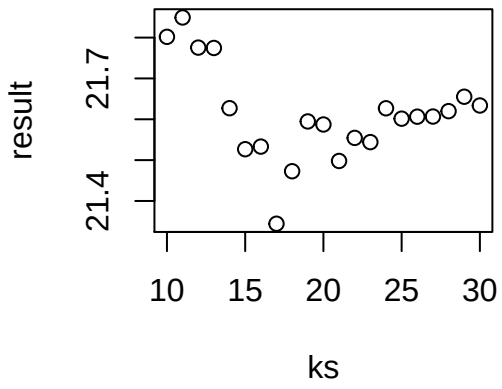
```
#### KNN [caret] all variables / excl. elevation ####
sqrt(nrow(trainset))
```

```
## [1] 125.4392
```

```
ks <- seq(10,30,1)
ks
```

```
##  [1] 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
```

```
# checking which is the optimal k
result <- lapply(ks, function(i){
  fit_knn2 <- knnreg(OC ~ N + pH + landuse, data = trainset, k = i)
  pred <-predict(fit_knn2, base)
  RMSE(testset$OC, pred)
})

plot(ks, result)
```

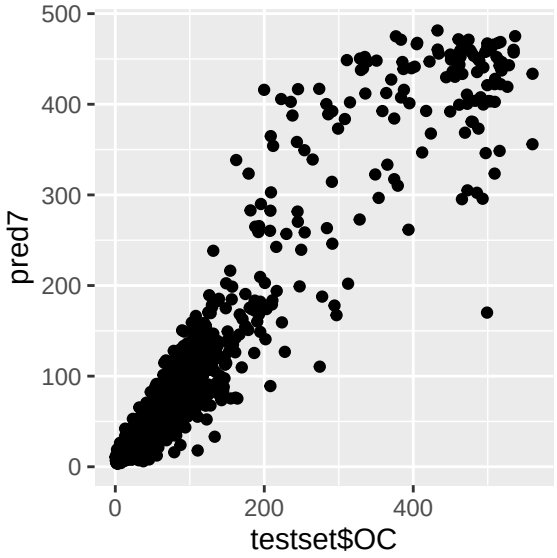



The graph above shows different k's (numbers of neighbors considered) and how the result (the RMSE) behaves with this. A k of 17 was used because it yields the best result.

```
best_k <- ks[which.min(data.frame(result)[1,])] #31
best_k # this is 17
```

```
## [1] 17
```

```
## using best k

fit_knn2 <- knnreg(OC ~ N + pH + landuse, data = trainset, k = best_k)
pred7 <-predict(fit_knn2, base)

qplot(testset$OC, pred7)
```




```
rmse7 <- RMSE(testset$OC, pred7)
rmse7
```

```
## [1] 21.34432
```

```
rmses <- rbind(rmses,
               data.frame(model = "knn excluding elev.", rmse = rmse7))
rmses[5:7,]
```

```
##                     model     rmse
## 5  lm with all variables 30.81398
## 6 knn with all variables 50.09455
## 7    knn excluding elev. 21.34432
```

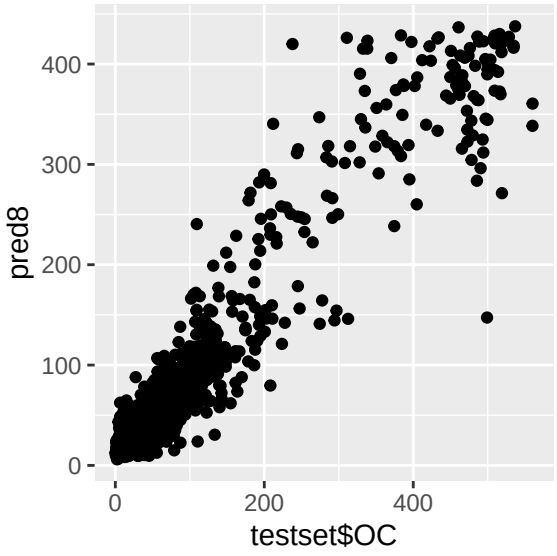
## (II h) Random forest model

A random forest model consists of many models (trees) that are combined (to a forest) so the number of trees plays a role in the performance of the model. The more trees the more exact the model usually is, but with more trees it also takes more time to calculate. Here, I started with 10 trees, then tried 100, then 250 and then 500. The number of trees (ntree = x) of 250 appeared to yield a good compromise between computing time and performance. The best model includes all selected variables, unlike KNN, which performed best without elevation. However, the random forest model is not quite as good as the KNN was.

```
set.seed(1)

fit_rf <- randomForest(
  OC ~ ., data = trainset,
  ntree = 250
)

# going from 10 to 100 trees in the only-N model improved the RMSE by ~0.4
# and another 0.2 for 250 trees, hardly then anymore for 500 trees

pred8 <- predict(fit_rf, base)
head(pred8)
```

```
##         7        12        16        28        29        38
##  21.22866  50.31972  65.58160 165.00924  46.67700 148.49321
```

```
qplot(testset$OC, pred8)
```




```
rmse8 <- RMSE(testset$OC, pred8)
rmse8
```

```
## [1] 23.14774
```

```
rmses <- rbind(rmses,
               data.frame(model = "rf with all variables", rmse = rmse8))
rmses[7:8,]
```

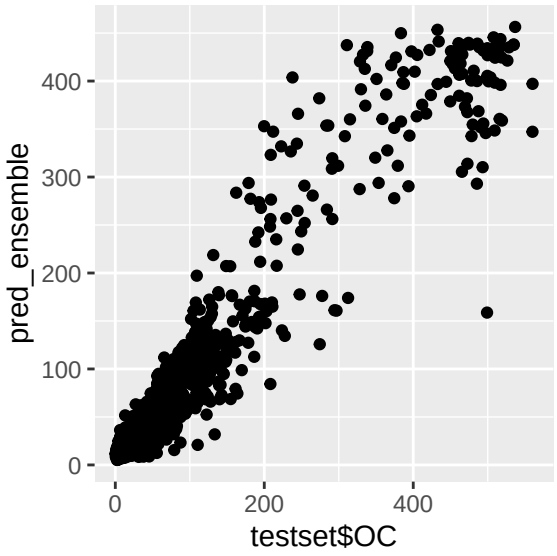
```
##                    model     rmse
## 7   knn excluding elev. 21.34432
## 8 rf with all variables 23.14774
```

```
## in this case all variables improve the model
```

## (II i) Ensemble model (KNN and random forest)

The two models with the best performance (lowest RMSE), were combined into an ensemble, where the mean is taken from the prediction of both models. This worked well as it further improved the RMSE:

```
pred_ensemble <- (pred7 + pred8)/2 # knn & random forest

qplot(testset$OC, pred_ensemble)
```




```
rmse_ens1 <- RMSE(testset$OC, pred_ensemble)
rmse_ens1
```

```
## [1] 20.86007
```

```
rmses <- rbind(rmses,
               data.frame(model = "ensemble of knn & rf", rmse = rmse_ens1))

rmses[7:9,]
```

```
##                    model     rmse
## 7   knn excluding elev. 21.34432
## 8 rf with all variables 23.14774
## 9  ensemble of knn & rf 20.86007
```

# (III) RESULTS

## (III a) RMSEs of tested models

As a result, we see an order of performance from best to least performing model that is Ensemble (KNN & random forest) > KNN > random forest > linear model.

Here's the overview table:

```
rmses
```

```
##                     model     rmse
## 1                    mean 77.78776
## 2              lm with N 33.24694
## 3          lm with N + pH 32.60644
## 4 lm with N + pH + elev. 32.54013
## 5  lm with all variables 30.81398
## 6 knn with all variables 50.09455
## 7    knn excluding elev. 21.34432
## 8  rf with all variables 23.14774
## 9   ensemble of knn & rf 20.86007
```

## (III b) Evaluation of best model on validation set

To finally evaluate performance of our model on an unseen dataset, we now predict organic carbon in the validation set with the best two models: KNN without elevation and random forest with all variables. It is done as an ensemble.

```
# incorporate the mean prediction of knn and rf since they are the best ones

fin_predict_a <- predict(fit_knn2, validation)
fin_predict_b <- predict(fit_rf, validation)

fin_predict <- (fin_predict_a + fin_predict_b) / 2

qplot(validation$OC, fin_predict, col = validation$landuse) +
  geom_vline(xintercept = 150, lty = 2) + geom_vline(xintercept = 400, lty = 2)
```


```
fin_rmse <- RMSE(validation$OC, fin_predict)

rmses <- rbind(rmses,
               data.frame(model = "FINAL RMSE OVERALL", rmse = fin_rmse))
```

The dashed lines represent areas where the model performs differently. We will come back to this soon, but let's first calculate the final RMSE overall.

```
fin_rmse
```

**The final RMSE, our typical prediction error, is:**

```
## [1] 19.01878
```

## (III b) Model performance in the most common range of organic carbon

In the observation-prediction-graph above, we see that there are ranges of organic carbon in which the model performs better than in other ranges. We can divide the validation set into three ranges that appear to differ and calculate model performance for each. This can be interesting in practical regards as the user may be interested in certain soils that lay within a specific range of organic carbon. I show here how it wsa done for the 0-150 mg C / g soil range and then performance in each range as a table:

```
# I put the observations, predictions and land uses in a data frame
df <- data.frame(landuse = validation$landuse,
                 observation = validation$OC, prediction = fin_predict)

# in the following, I divide this set and observe the model performance on
# data ranges
```

```
# range 0-150
obs  <- df$observation[df$observation < 150]
pred <- df$prediction[df$observation < 150]

fin_rmse_0_150 <- RMSE(obs, pred)
fin_rmse_0_150
```

```
## [1] 10.21561
```

It is of interest which fraction of observations within this range that yields the best result. It turns out to be 95% of all observations.

```
length(eusoil$OC[eusoil$OC < 150]) /
  length(eusoil$OC[eusoil$OC])
```

```
## [1] 0.9514061
```

```
# it is 95% of observations
```

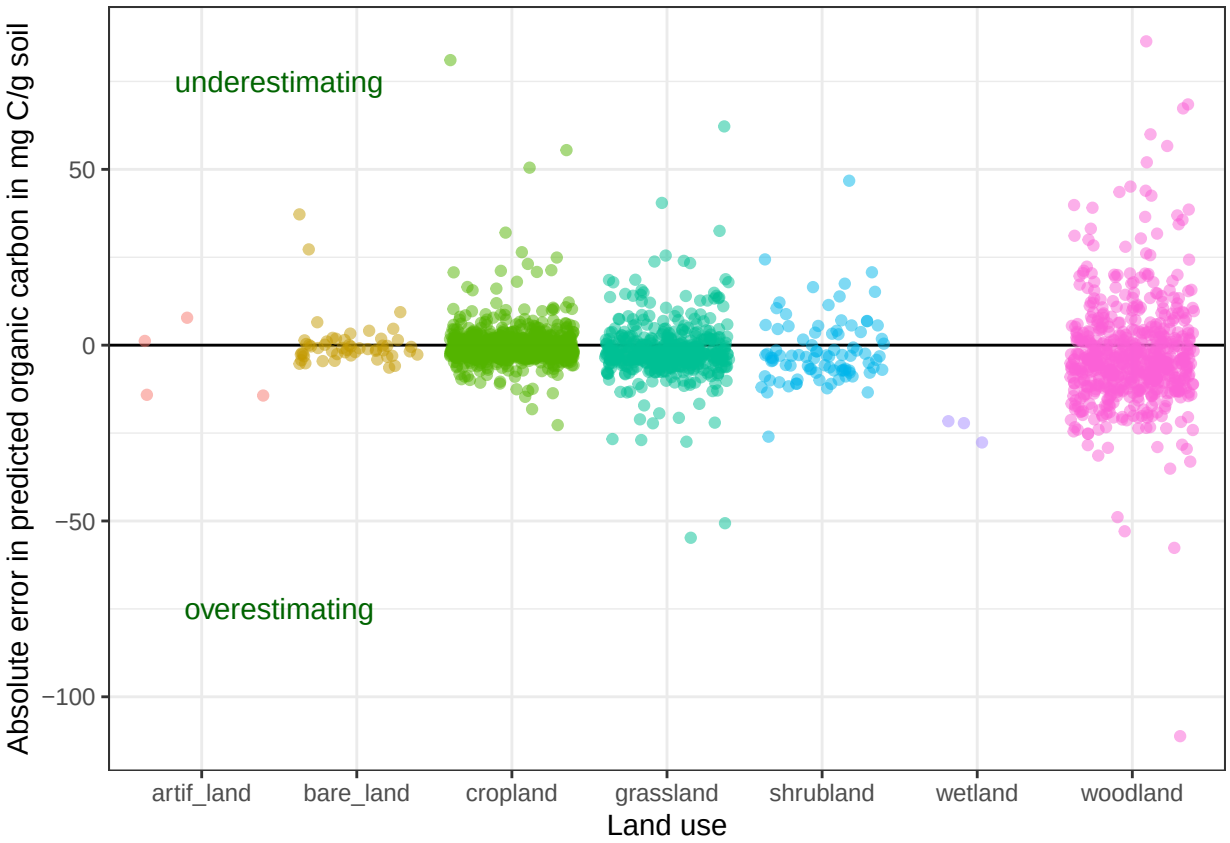
For the selected ranges, the model performs as follows:

```
rmses
```

```
##                       model     rmse
## 1    range <150 mg/g org. C 10.21561
## 2 range 150-400 mg/g org. C 80.21301
## 3    range 400+ mg/g org. C 75.19328
```

We see that performance is best in a range lower than 150 mg C / g soil with a typical error of just 10.2 mg C / g soil. How does this error spread over the different land uses?

```
df %>% filter(observation < 150) %>% mutate(error = observation - prediction) %>%
  ggplot(aes(landuse, error, col = landuse)) +
  geom_hline(yintercept = 0)+ geom_jitter(alpha = 0.5) +
  theme_bw() + ylab("Absolute error in predicted organic carbon in mg C/g soil") +
  xlab("Land use") + theme(legend.position = "none") +
  annotate(geom = "text", label = "underestimating", x = 1.5, y = 75, col = "darkgreen") +
  annotate(geom = "text", label = "overestimating", x = 1.5, y = -75, col = "darkgreen")
```




# (IV) CONCLUSION

## (IV a) Project summary

Using the soil parameters total P (phosphorus), pH, elevation and landuse, we were able to predict soil organic carbon with a typical error of 10.2 mg C/g soil in the range of organic carbon below 150 mg C/g soil. The predictions were relatively consistent for all land uses. It paid off to use different modelling approaches and do some fine-tuning of these models as well. In the end, a combination of KNN and random forest performed best.

## (IV b) Limitations

There are of course several more approaches that can be done and an ensemble of many more models is very likely to further improve the result. Also more environmental variables like precipitation, temperature and consistent availability of soil texture would likely improve the model.

## (IV c) Future applications

Predicting soil organic carbon is just one example of what can be done. Within large datasets, often some soil variables are missing and those could, in a certain range of accuracy, be predicted from those variables that are present to fill in the gaps. Another application is the large-scale prediction of a variable that is complicated to measure and only available for a fraction of the area of interest. Machine learning algorithms

can be used to predict those variables. When we use such algorithms in a transparent way and are open about their strength, limiations and range of error, they can be very useful in giving insights that we otherwise would not obtain - which is what data science is all about.

This report together with the *.R analysis file, the *.Rmd file and the *Rdata file used can be found at my github repository at: https://github.com/ms-soil/ds-submission-soil