Prediction of Titratable Acidity

Vincent van der Berg

24/10/2022

Introduction:

This workbook covers a modelling workflow to identify and build the highest performing model to predict titratable acidity from commonly analysed soil attributes.

This project was executed on behalf of Vivian White from Citrus Research International (CRI).

The following packages were used throughout this analysis.

```
pacman::p_load(
  tidyverse,
  tidymodels,
  GGally, # For additional visualisation ability
  earth, # MARS model
  glmnet, # Generalised linear model
  rpart, # Decision tree model
  ranger, # Random forest model
  mixOmics,
  tidytext,
  skimr,
  corrr,
  ggforce
)
```

Data Import:

```
data <- readr::read_csv("data/tit_acid_full_raw")
```

```
## Rows: 5916 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): texture
## dbl (22): sample, pH, resistance, H, stone_v_v, p_ambic_i, K, ec_Na, ec_K, e...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(data)
```

```
## Rows: 5,916
## Columns: 23
## $ sample     <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ~
## $ texture    <chr> "Loam", "Loam", "Loam", "Loam", "Loam", "Loam", "Loam", "Lo~
## $ pH         <dbl> 4.41, 4.89, 4.12, 3.92, 5.07, 5.07, 4.97, 4.55, 4.64, 4.65,~
## $ resistance <dbl> 1808.1088, 2158.6728, 2362.6912, 2362.6912, 1978.9950, 1860~
## $ H          <dbl> 1.605, 0.975, 1.570, 1.650, 0.685, 0.855, 0.880, 1.000, 0.9~
## $ stone_v_v  <dbl> 5.697395, 17.009494, 21.719133, 13.200000, 11.770907, 7.212~
## $ p_ambic_i  <dbl> 18.249821, 11.084400, 11.814932, 22.510775, 9.761629, 17.00~
## $ K          <dbl> 134.65253, 58.37692, 67.36806, 106.74411, 35.82490, 52.7166~
## $ ec_Na      <dbl> 0.10518443, 0.09209445, 0.08057108, 0.06419033, 0.09914975,~
## $ ec_K       <dbl> 0.34526289, 0.14968442, 0.17273861, 0.27370284, 0.09185871,~
## $ ec_Ca      <dbl> 1.5040232, 1.6016999, 1.0755850, 0.3420924, 3.1894703, 3.17~
## $ ec_Mg      <dbl> 0.4524739, 0.8750030, 0.5821628, 0.2430769, 0.4751200, 0.50~
## $ bs_Na      <dbl> 4.370040, 3.387716, 4.216047, 6.954062, 2.183627, 2.187111,~
## $ bs_K       <dbl> 14.344449, 5.506177, 9.038902, 29.651605, 2.023053, 2.83346~
## $ bs_Ca      <dbl> 62.48683, 58.91891, 56.28219, 37.06059, 70.24339, 66.56338,~
## $ bs_Mg      <dbl> 18.798683, 32.187195, 30.462861, 26.333743, 10.463818, 10.4~
## $ bs_C       <dbl> 0.7394366, 0.6408451, 0.7676056, 0.5000000, 0.5000000, 0.64~
## $ s_value    <dbl> 2.4069444, 2.7184819, 1.9110574, 0.9230625, 3.8555988, 3.91~
## $ EC         <dbl> 0.088, 0.090, 0.093, 0.066, 0.082, 0.187, 0.057, 0.055, 0.0~
## $ r_eksteen  <dbl> 1.721461, 4.055556, 1.175870, 1.084400, 5.421965, 5.421965,~
## $ r_observed <dbl> 1.2190013, 2.5402082, 1.0558903, 0.3546481, 5.3497669, 4.29~
## $ h_eksteen  <dbl> 1.1365327, 0.6106938, 1.4098055, 0.5396251, 0.6758786, 0.67~
## $ h_observed <dbl> 1.605, 0.975, 1.570, 1.650, 0.685, 0.855, 0.880, 1.000, 0.9~
```

```
data <- data %>%
  # Select and rename columns
  dplyr::select(
    sample,
    texture,
    pH,
    resistance,
    stone = stone_v_v,
    ambic_p = p_ambic_i,
    K,
    ec_Na:bs_Mg,
    carbon = bs_C,
    s_value,
    EC,
    r_eksteen:r_observed,
    ta_eksteen = h_eksteen,
    ta_true = H
  )
# Check for correctness and missing values
skim(data)
```

Table 1: Data summary

Name	data
Number of rows	5916
Number of columns	22

Table 1: Data summary

Column type frequency:	
character	1
numeric	21
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
texture	549	0.91	4	9	0	5	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
sample	0	1.00	2958.50	1707.95	1.00	1479.75	2958.50	4437.25	5916.00	
pH	0	1.00	5.07	0.65	2.68	4.60	5.19	5.61	6.92	
resistance	0	1.00	1408.43	773.34	146.72	900.22	1246.30	1713.75	7574.86	
stone	15	1.00	12.13	11.08	1.00	2.78	9.04	18.84	96.00	
ambic_p	4	1.00	34.41	42.31	0.50	8.80	20.25	42.99	552.60	
K	0	1.00	109.87	99.23	3.00	49.40	82.30	135.26	1902.00	
ec_Na	0	1.00	0.27	0.32	0.04	0.14	0.20	0.27	11.70	
ec_K	1	1.00	0.28	0.25	0.01	0.13	0.21	0.35	4.88	
ec_Ca	1	1.00	3.99	3.08	0.32	1.92	3.19	5.10	58.39	
ec_Mg	1	1.00	1.14	1.07	0.02	0.51	0.85	1.39	17.79	
bs_Na	1	1.00	4.77	3.34	0.43	2.53	3.95	5.95	40.97	
bs_K	1	1.00	4.61	2.88	0.25	2.59	4.05	5.96	29.65	
bs_Ca	1	1.00	59.04	13.39	9.04	50.81	61.11	68.86	97.60	
bs_Mg	1	1.00	17.09	7.61	0.31	12.16	15.82	20.23	73.61	
carbon	1	1.00	0.59	0.50	0.01	0.25	0.43	0.74	4.07	
s_value	1	1.00	5.68	4.00	0.57	2.89	4.62	7.30	59.13	
EC	834	0.86	0.23	0.36	0.01	0.07	0.14	0.25	6.37	
r_eksteen	0	1.00	6.88	4.52	-1.64	2.42	6.47	10.82	21.08	
r_observed	1	1.00	10.41	10.72	0.18	3.81	7.80	13.56	181.20	
ta_eksteen	1	1.00	1.16	1.37	-0.85	0.43	0.77	1.39	52.57	
ta_true	0	1.00	0.75	0.66	0.12	0.36	0.54	0.88	7.76	

Data Cleaning:

Lots of missing values for texture, EC, and some for stone and ambic_p.

```
data_clean <- data %>%
  # Drop sample 1117 - most vars are NA's #
  filter(!(sample == 1117)) %>%
  # Drop EC and stone - too many NA's (834, 15) #
  dplyr::select(-c(EC, stone)) %>%
  # Fill missing carbon and phosphorous value with column mean
  mutate(carbon = replace(carbon, is.na(carbon), mean(carbon, na.rm = TRUE)),
```

```
        ambic_p = replace(ambic_p, is.na(ambic_p), mean(ambic_p, na.rm = TRUE)),
        # Fill missing texture values with "unknown"
        texture = replace(texture, is.na(texture), "unknown"))

# Check for correctness and missing values
skim(data = data_clean)
```

Table 4: Data summary

Name	data_clean
Number of rows	5915
Number of columns	20
Column type frequency:	
character	1
numeric	19
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
texture	0	1	4	9	0	6	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
sample	0	1	2958.81	1707.92	1.00	1480.50	2959.00	4437.50	5916.00	
pH	0	1	5.07	0.65	2.68	4.60	5.19	5.61	6.92	
resistance	0	1	1408.51	773.39	146.72	900.22	1246.30	1713.75	7574.86	
ambic_p	0	1	34.41	42.30	0.50	8.80	20.28	42.94	552.60	
K	0	1	109.87	99.24	3.00	49.39	82.25	135.27	1902.00	
ec_Na	0	1	0.27	0.32	0.04	0.14	0.20	0.27	11.70	
ec_K	0	1	0.28	0.25	0.01	0.13	0.21	0.35	4.88	
ec_Ca	0	1	3.99	3.08	0.32	1.92	3.19	5.10	58.39	
ec_Mg	0	1	1.14	1.07	0.02	0.51	0.85	1.39	17.79	
bs_Na	0	1	4.77	3.34	0.43	2.53	3.95	5.95	40.97	
bs_K	0	1	4.61	2.88	0.25	2.59	4.05	5.96	29.65	
bs_Ca	0	1	59.04	13.39	9.04	50.81	61.11	68.86	97.60	
bs_Mg	0	1	17.09	7.61	0.31	12.16	15.82	20.23	73.61	
carbon	0	1	0.59	0.50	0.01	0.25	0.43	0.74	4.07	
s_value	0	1	5.68	4.00	0.57	2.89	4.62	7.30	59.13	
r_eksteen	0	1	6.88	4.52	-1.64	2.42	6.47	10.82	21.08	
r_observed	0	1	10.41	10.72	0.18	3.81	7.80	13.56	181.20	
ta_eksteen	0	1	1.16	1.37	-0.85	0.43	0.77	1.39	52.57	
ta_true	0	1	0.75	0.66	0.12	0.36	0.54	0.88	7.76	

```
write_csv(data_clean, file = "./data/data_clean.csv")
```

Train/Test Split:

The target **ta_true** was normalised by taking its log. This is done to ensure a normal distribution in the target, and to ensure positive predictions. It also limits the effect of skewness on model error, thus resulting in better model fit during training.

```
## Load cleaned data set
main_data <- read_csv(file = "./data/data_clean.csv")
```

```
## Rows: 5915 Columns: 20
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): texture
## dbl (19): sample, pH, resistance, ambic_p, K, ec_Na, ec_K, ec_Ca, ec_Mg, bs_...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
set.seed(42)
soil_split <- main_data %>%
  dplyr::select(-c(texture, sample)) %>%
  mutate(ta_true = log10(ta_true)) %>%
  initial_split(strata = ta_true)

soil_split
```

```
## <Training/Testing/Total>
## <4436/1479/5915>
```

```
soil_train <- training(soil_split)
soil_test <- testing(soil_split)
```

Exploratory Data Analysis:

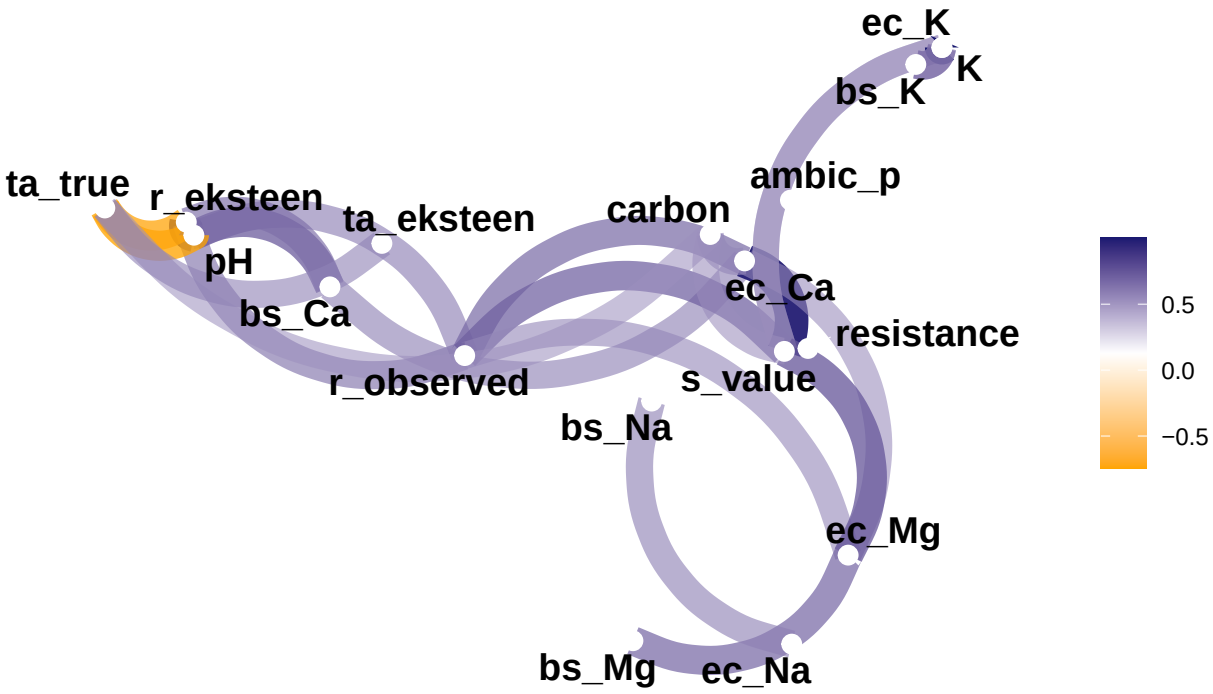
To explore the training set further, a correlation analysis, principal component analysis (PCA) and a partial least squares (PLS) analysis was performed.

Correlation:

```
soil_train %>%
  correlate() %>%
  rearrange() %>%
  network_plot(colours = c("orange", "white", "midnightblue"),
               min_cor = 0.5, legend = "range") +
  labs(title = "Correlation Network")
```

```
## Correlation computed with
## * Method: 'pearson'
## * Missing treated using: 'pairwise.complete.obs'
```

Correlation Network



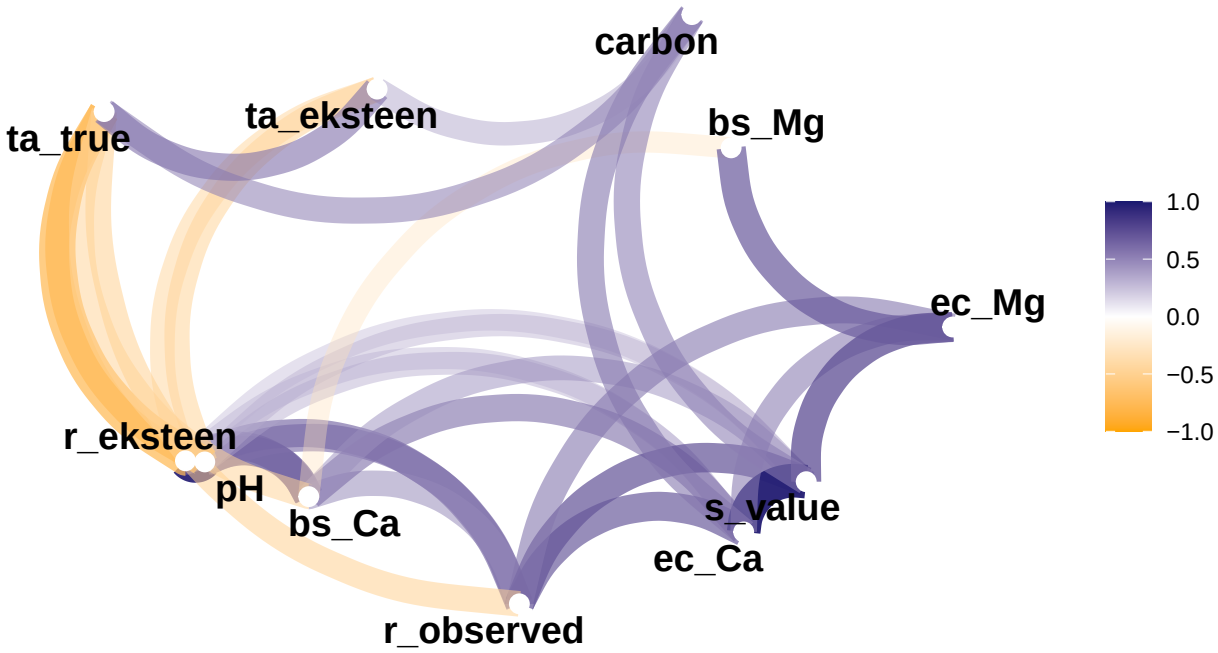
There is a significant direct negative correlation between variables **r_eksteen** and **pH** and the target **ta_true**. This makes sense as both these variables are inextricably linked with soil acidity. **carbon** and **ta_eksteen** also show to be sensibly positvely correlated with **ta_true**.

There are numerous variables not associated with the target **ta_true**. Variables associated with Na, K, and Mg show no direct correlation with **ta_true**. This includes the **s_value** and **resistance** variables. **ambic_p** further shows no significant correlation.

Insignificantly correlated were removed, and the resulting data set was correlated again, to tease out finer relationships.

```
soil_train %>%
  dplyr::select(-c(resistance, ambic_p, ec_Na, bs_Na, bs_K, ec_K, K)) %>%
  correlate() %>%
  rearrange() %>%
  network_plot(colours = c("orange", "white", "midnightblue")) +
  labs(title = "Correlation Network on Data-subset")
```

```
## Correlation computed with
## * Method: 'pearson'
## * Missing treated using: 'pairwise.complete.obs'
```



The variables with significant direct correlation to **ta__true** are **r__eksteen**, **pH**, **bs__Ca** and **r__observed** (*All negative*). **carbon** and **ta__eksteen** show significant positive correlation.

All these variables are however correlate with other variables that show no significant direct correlation with **ta__true**. It can be concluded that these variables with significant correlation should be considered as candidates for modelling **ta__true**

Principal Component Analysis:

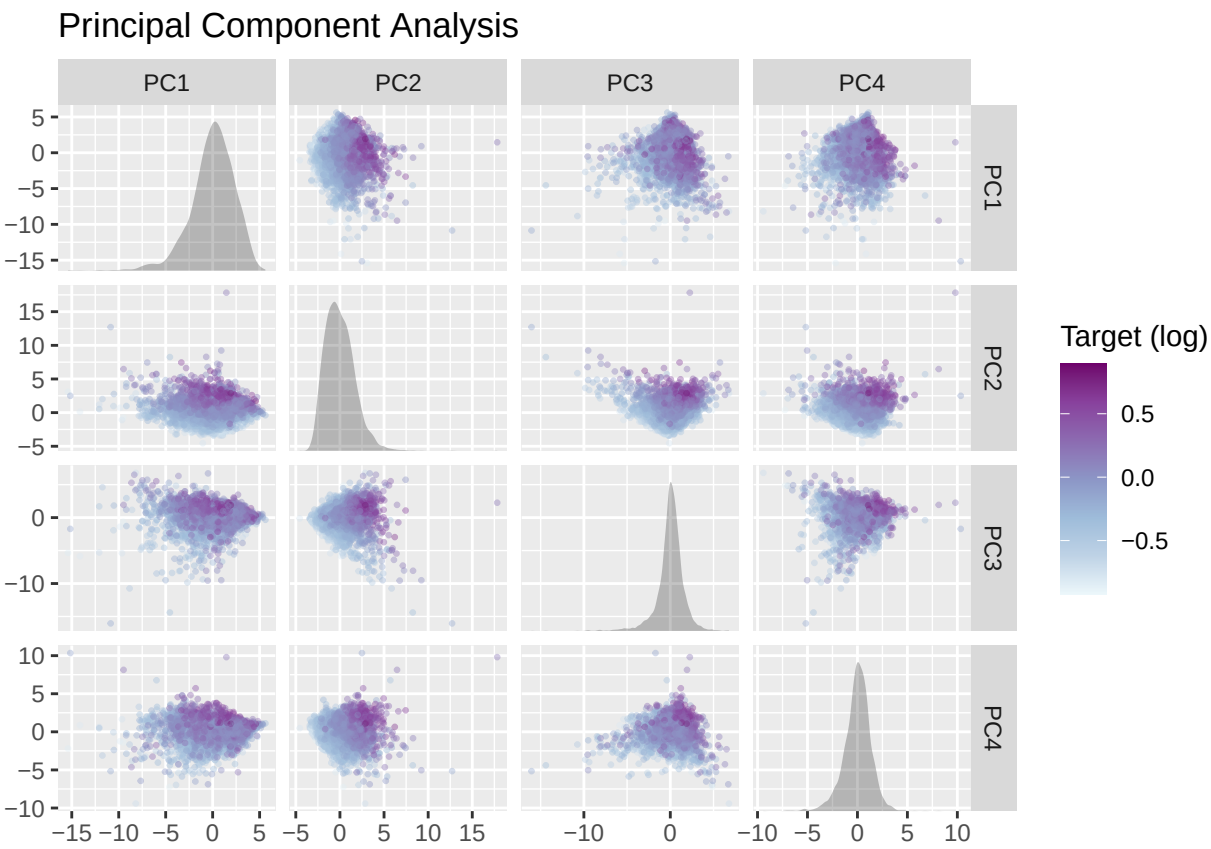
PCA is one of the most straghtforward dimensionality reduction approaches. It is a linear, unsupervised technique that makes new features to try and account for as much variation in the data as possible.

```
soil_rec <- recipe(ta_true ~ ., data = soil_train) %>%
  step_zv(all_numeric_predictors()) %>%
  step_normalize(all_numeric_predictors())

rec_trained <- prep(soil_rec)
rec_trained
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor         17
##
## Training data contained 4436 data points and no missing data.
```

```
##
## Operations:
##
## Zero variance filter removed <none> [trained]
## Centering and scaling for pH, resistance, ambic_p, K, ec_Na, ec_K, ec_Ca,... [trained]
```

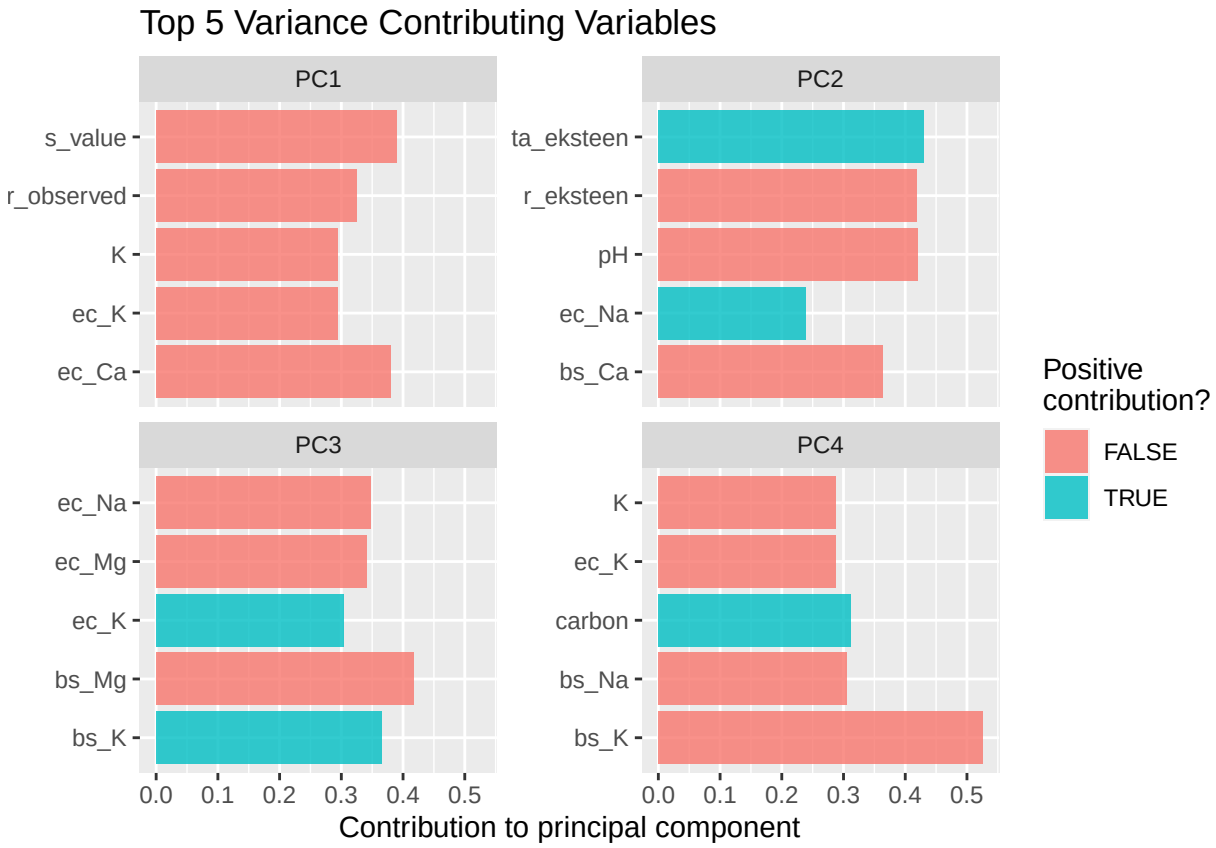
```
rec_trained %>%
  step_pca(all_numeric_predictors(), num_comp = 4) %>%
  prep() %>%
  juice() %>%
  ggplot() +
  geom_autopoint(aes(colour = ta_true), alpha = 0.4, size = 0.5) +
  geom_autodensity(alpha = 0.3) +
  facet_matrix(vars(-ta_true), layer.diag = 2) +
  scale_color_distiller(palette = "BuPu", direction = 1) +
  labs(colour = "Target (log)") +
  ggtitle("Principal Component Analysis")
```



From the above it is clear that there is somewhat of a relationship between the components and the target variable. There is a noticeable grouping and resulting gradient in the distribution of target values. However, the relationships are not adequate enough to suggest that this data set is strictly linearly separable.

These components was further investigated to find the features that contribute the most to these principal components. Thus, variables that have the greatest linear contribution to the variance observed within the data set were identified.

```
rec_trained %>%
  step_pca(all_numeric_predictors(), num_comp = 4) %>%
  prep() %>%
  tidy(number = 3) %>%
  filter(component %in% paste0("PC", 1:4)) %>%
  group_by(component) %>%
  slice_max(abs(value), n = 5) %>%
  ungroup() %>%
  ggplot(mapping = aes(x = abs(value), y = terms, fill = value > 0)) +
  geom_col(alpha = 0.8) +
  facet_wrap(vars(component), scales = "free_y") +
  labs(x = "Contribution to principal component",
       y = NULL, fill = "Positive\ncontribution?",
       title = "Top 5 Variance Contributing Variables")
```

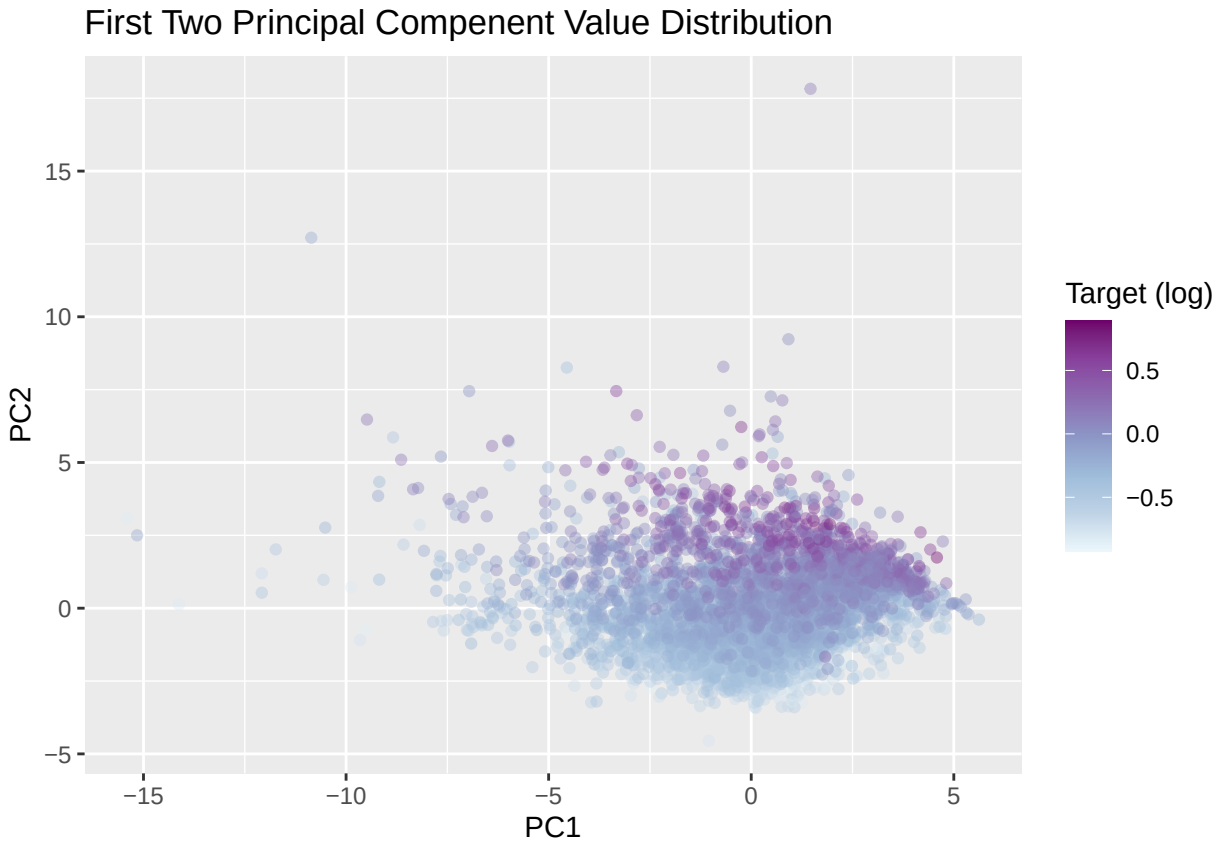


PC1 is mostly about a soil's exchangeable cation component, particularly **s__value** and **ec__Ca**. The contribution to observed variance by these components change in unison.

PC2 is mostly about acidity and soil reaction. **r__eksteen**, **pH** and **bs__Ca** are strongly associated, in contrast to **ta__eksteen**. Thus, this component suggests that as **ta__eksteen** increases, decreases in **r__eksteen** and **pH** are to be expected.

PC3 and PC4 are mostly concerned with exchangeable cations, particularly **bs__Mg** and **bs__K**. There are thus two groupings of soils that contribute to the variance of soils. Soils with a high exchangeable cation component and a low soil acidity and reaction component.

```
rec_trained %>%
  step_pca(all_numeric_predictors(), num_comp = 4) %>%
  prep() %>%
  juice() %>%
  ggplot(mapping = aes(x = PC1, y = PC2, colour = ta_true)) +
  geom_point(alpha = 0.4,) +
  scale_color_distiller(palette = "BuPu", direction = 1) +
  labs(colour = "Target (log)",
       title = "First Two Principal Compenent Value Distribution")
```



The diagram above illustrate the point stated previously. Soils with high exchangeable cation content, low acidity and high pH are at the lower left of the data cloud. Soils with lower exchangeable cation content, higher acidity, and lower pH are clustered towards the top and top-right of the data cloud.

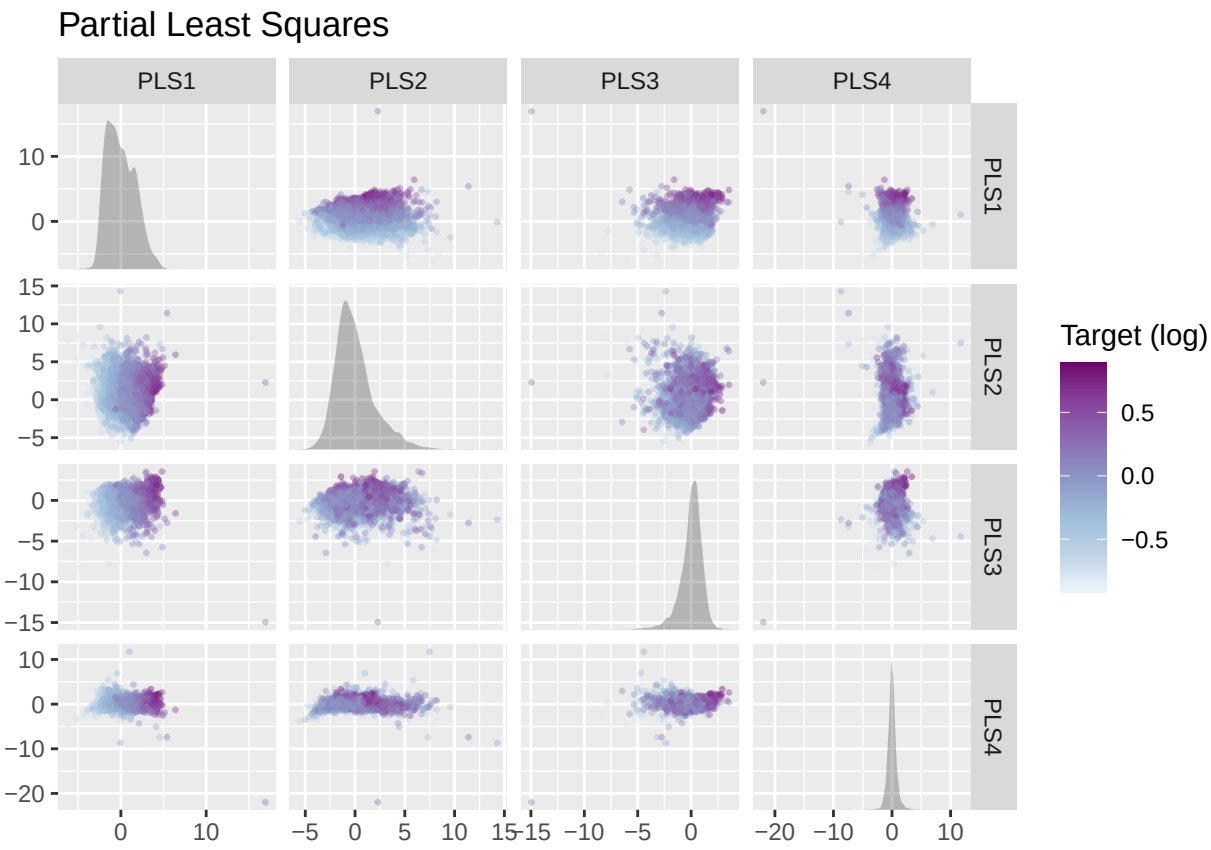
Partial Least Squares Analysis:

PLS is similar to PCA, but it is a supervised technique. The technique creates components that aim to account for as much variation as possible, whilst being related to the outcome.

```
pls_rec <- recipe(ta_true ~ ., data = soil_train) %>%
  step_log(ta_true) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_pls(all_numeric_predictors(), outcome = "ta_true")

pls_prep <- prep(pls_rec)
```

```
## Warning in bake.step_log(x$steps[[i]], new_data = training): NaNs produced

## Warning: `step_pls()` failed: Error in if (diff.value < tol | iter > max.iter) break :
##   missing value where TRUE/FALSE needed
```

```
rec_trained %>%
  step_pls(all_numeric_predictors(), outcome = "ta_true", num_comp = 4) %>%
  prep() %>%
  juice() %>%
  ggplot() +
  geom_autopoint(aes(colour = ta_true), alpha = 0.4, size = 0.5) +
  geom_autodensity(alpha = 0.3) +
  facet_matrix(vars(-ta_true), layer.diag = 2) +
  scale_color_distiller(palette = "BuPu", direction = 1) +
  labs(colour = "Target (log)") +
  ggtitle("Partial Least Squares")
```

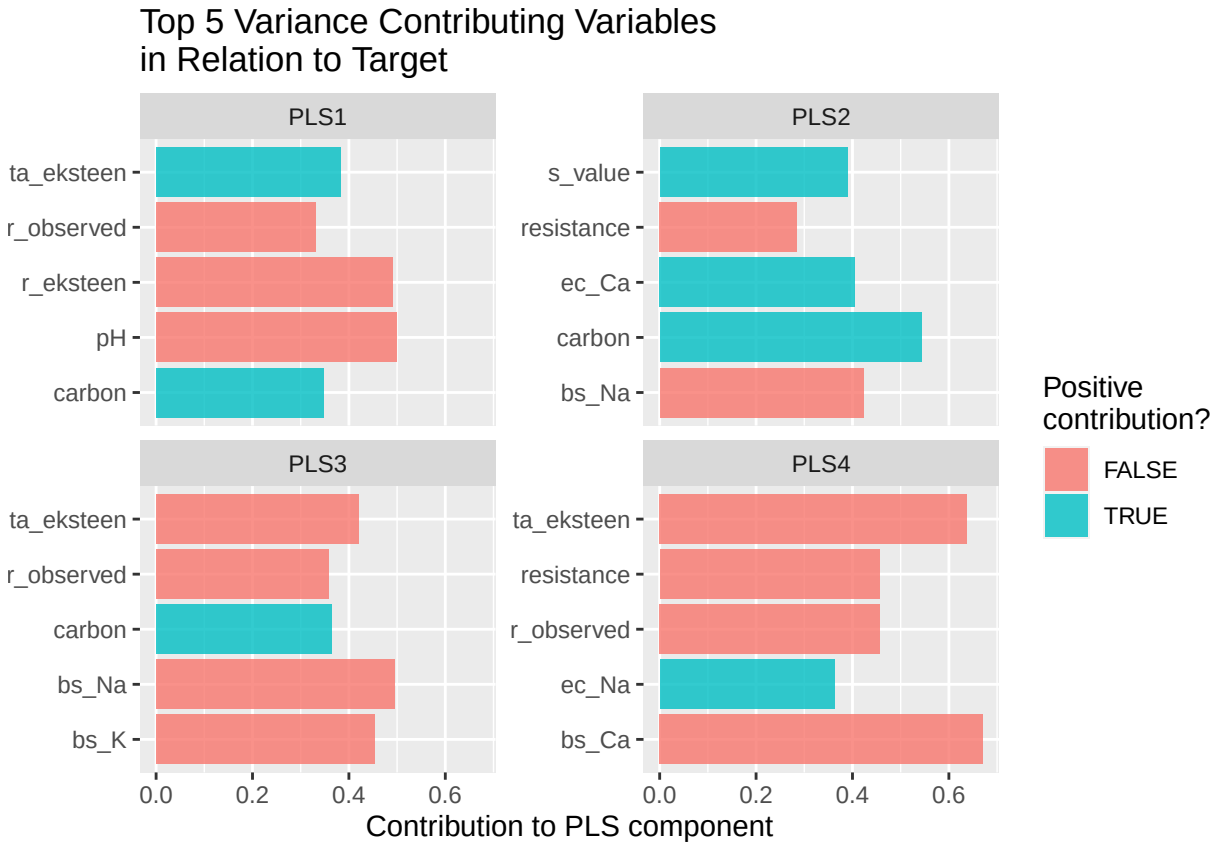


The diagram above illustrates the relationships between the components. There is a stronger relationship present between the components and the target. This is expected from the direct relation to **ta_true**.

These components was further investigated to find the features that contribute the most to the components, similar to that of PCA.

```
rec_trained %>%
  step_pls(all_numeric_predictors(), outcome = "ta_true", num_comp = 4) %>%
  prep() %>%
  tidy(number = 3) %>%
```

```
  filter(component %in% paste0("PLS", 1:4)) %>%
  group_by(component) %>%
  slice_max(abs(value), n = 5) %>%
  ungroup() %>%
  ggplot(mapping = aes(x = abs(value), y = terms, fill = value > 0)) +
  geom_col(alpha = 0.8) +
  facet_wrap(vars(component), scales = "free_y") +
  labs(x = "Contribution to PLS component",
       y = NULL, fill = "Positive\ncontribution?",
       title = "Top 5 Variance Contributing Variables\nin Relation to Target")
```



The diagram above illustrates that **ta_eksteen**, **r_observed**, **r_eksteen**, **pH**, and **carbon** most significantly contribute to the first couple of components.

EDA Conclusion

It can be concluded from this exploration that only six variables are required to effectively model titratable acidity. These variables include **ta_eksteen**, **r_observed**, **r_eksteen**, **pH**, **carbon**. Components related to calcium are also of interest. The selected variables will serve as the main parameters for model development.

Both **ec_Ca** and **bs_Ca** showed relationships to **ta_true**. However, since the value of **bs_Ca** relies on acidity itself, its interaction could be misleading. Going forward then, **ec_Ca** was included in the six variables used for modelling of the current data set.

Modelling Workflow

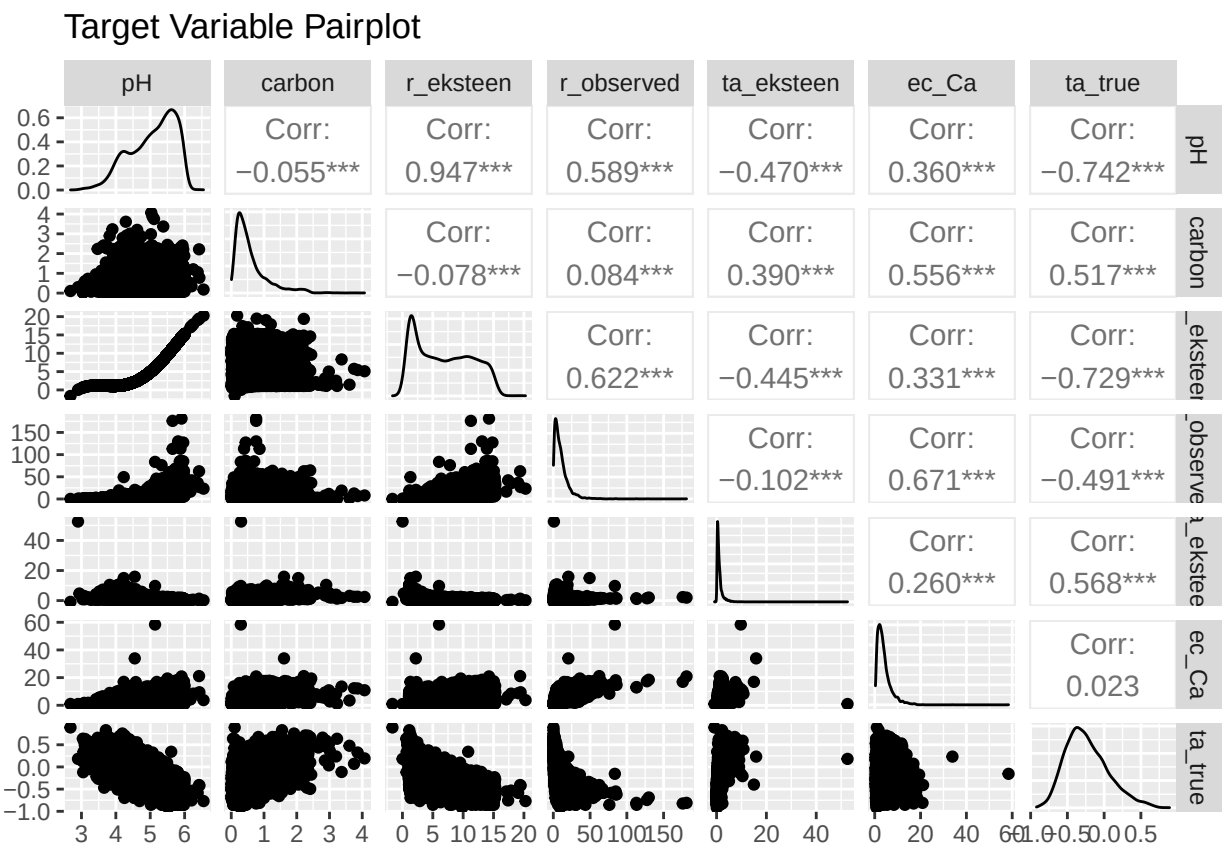
Model evaluation

Four models was selected for evaluation. They include a penalised general linear model, multi-adaptive regression splines, decision trees and an ensemble of trees, the random forest.

The linear model was included as reference model, whilst the other three are strictly non-linear. The reliance on non-linear models is due to the poor linear separability of the data set.

The models were also selected due to their ease of use and their robustness to predictor skewness. Thus, pre-processing of the data, such as transformation, normalisation, feature scaling, etc is not required.

```
train_data <- soil_train %>%
  dplyr::select(pH, carbon, r_eksteen, r_observed, ta_eksteen, ec_Ca, ta_true)

test_data <- soil_test %>%
  dplyr::select(pH, carbon, r_eksteen, r_observed, ta_eksteen, ec_Ca, ta_true)

train_data %>%
  GGally::ggpairs(progress = FALSE) +
  labs(title = "Target Variable Pairplot")
```



This study evaluated model performance during training by implementing repeated K-fold cross-validation. Training data was split into 15 folds. Models were trained on 14 folds and evaluated on the holdout fold. There were thus 15 unique iterations of the holdout fold. This entire process repeated 3 times.

The approach described above allows for a detailed assessment of model performance. The allows for the

generation of performance values such as rmse or mae with statistical properties. The statistical properties of interest are the mean (n = 45) and standard error of the mean, and was focussed on during this evaluation.

Each of the models' hyper-parameters were tuned during training to find the most optimal combination. To achieve this, grid search was employed on a grid of 10 iterations of each hyper-parameter. Thus, for the penalised GLM, a 10 x 10 hyper-parameter grid was searched for the combination of parameters that yielded the best performing model. To avoid overfitting and reduce training time, the random forest was allowed to only utilise 750 trees.

```
## Create validation folds
cv_folds <- vfold_cv(data = train_data,
                     strata = ta_true,
                     repeats = 3,
                     v = 15)

## Create recipes for modelling
normal_rec <- recipe(ta_true ~ pH + carbon + r_eksteen + r_observed + ta_eksteen + ec_Ca, data = train_

## Recipe with interactions
interact_rec <- normal_rec %>%
  step_interact(~ all_predictors():all_predictors()) %>%
  step_corr(threshold = 0.75) # Remove highly correlated variables

## Specify models
# Penalised GLM
linear_model <- linear_reg(penalty = tune(),
                           mixture = tune()) %>%
  set_engine("glmnet") %>%
  set_mode("regression")
# MARS model
mars_model <- mars(prod_degree = tune()) %>%
  set_engine("earth") %>%
  set_mode("regression")
# Decision tree
cart_model <- decision_tree(cost_complexity = tune(),
                            min_n = tune()) %>%
  set_engine("rpart") %>%
  set_mode("regression")
# Random forest
rf_model <- rand_forest(mtry = tune(),
                        min_n = tune(),
                        trees = 750) %>%
  set_engine("ranger") %>%
  set_mode("regression")

## Compile recipes and model specifications into workflow
all_workflows <- workflow_set(
  preproc = list(normal = normal_rec,
                 interactions = interact_rec),
  models = list(linear = linear_model, mars = mars_model,
                cart = cart_model, forest = rf_model)
)

## Specify control grid
grid_ctrl <- control_grid(
```

```
  save_pred = TRUE,
  parallel_over = "everything"
)
```

Evaluation results

```
## Collect evaluation results
grid_results <- all_workflows %>%
  workflow_map(
    fn = "tune_grid",
    seed = 42,
    resamples = cv_folds,
    grid = 10,
    control = grid_ctrl,
    metrics = metric_set(rmse, mae, rsq),
    verbose = TRUE
  )
```

```
## i 1 of 8 tuning:     normal_linear

## v 1 of 8 tuning:     normal_linear (1m 33.5s)

## i 2 of 8 tuning:     normal_mars

## v 2 of 8 tuning:     normal_mars (23.1s)

## i 3 of 8 tuning:     normal_cart

## v 3 of 8 tuning:     normal_cart (2m 10.4s)

## i 4 of 8 tuning:     normal_forest

## i Creating pre-processing data to finalize unknown parameter: mtry

## v 4 of 8 tuning:     normal_forest (32m 8.3s)

## i 5 of 8 tuning:     interactions_linear

## v 5 of 8 tuning:     interactions_linear (2m 56.3s)

## i 6 of 8 tuning:     interactions_mars

## v 6 of 8 tuning:     interactions_mars (57.8s)

## i 7 of 8 tuning:     interactions_cart

## v 7 of 8 tuning:     interactions_cart (5m 33.4s)
```

```
## i 8 of 8 tuning:     interactions_forest

## i Creating pre-processing data to finalize unknown parameter: mtry

## v 8 of 8 tuning:     interactions_forest (1h 41m 10s)
```

```
## Plot results
autoplot(grid_results,
         metric = c("rsq", "rmse", "mae"),
         select_best = TRUE) +
  ggtitle("Model Training Results\n15-Fold Cross Validation\n3 Repeats")
```


```
training_metrics <- grid_results %>%
  rank_results(select_best = TRUE) %>%
  dplyr::select(
    Model = wflow_id,
    Metric = .metric,
    Mean = mean,
    Std_err = std_err,
    n,
    Rank = rank
  ) %>%
  mutate(Mean = round(Mean, 4),
         Std_err = round(Std_err, 4))
```

```
# write_csv(training_metrics, file = "./ouputs/model_training_metrics.csv")

training_metrics
```

```
## # A tibble: 24 x 6
##    Model               Metric   Mean Std_err     n  Rank
##    <chr>               <chr>   <dbl>   <dbl> <int> <int>
##  1 interactions_mars   mae    0.0114  0.0004    45     1
##  2 interactions_mars   rmse   0.0193  0.0017    45     1
##  3 interactions_mars   rsq    0.994   0.0013    45     1
##  4 interactions_forest mae    0.0283  0.0003    45     2
##  5 interactions_forest rmse   0.0439  0.0011    45     2
##  6 interactions_forest rsq    0.977   0.0011    45     2
##  7 normal_forest       mae    0.0344  0.0004    45     3
##  8 normal_forest       rmse   0.0513  0.0013    45     3
##  9 normal_forest       rsq    0.969   0.0016    45     3
## 10 interactions_cart   mae    0.0424  0.0004    45     4
## # ... with 14 more rows
```

The top three performing workflows were the MARS and random forest models with interaction features. The random forest without interaction components came in third, yet nevertheless performed adequately.

Below are each of the top three performing model's hyperparameter settings.

```
grid_results %>%
  extract_workflow_set_result(id = "interactions_mars") %>%
  select_best(metric = "mae")
```

```
## # A tibble: 1 x 2
##   prod_degree .config
##         <int> <chr>
## 1           1 Preprocessor1_Model1
```

```
grid_results %>%
  extract_workflow_set_result(id = "interactions_forest") %>%
  select_best(metric = "mae")
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1    13     8 Preprocessor1_Model07
```

```
grid_results %>%
  extract_workflow_set_result(id = "normal_forest") %>%
  select_best(metric = "mae")
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     4     8 Preprocessor1_Model07
```

Train Best Models

Below each of the models were trained with the hyper parameter settings found to be optimal during model evaluation (table above).

```
mars_spec <- mars(prod_degree = 1) %>%
  set_engine("earth") %>%
  set_mode("regression")

mars_wflow <- workflow() %>%
  add_recipe(interact_rec) %>%
  add_model(mars_spec)

rf_spec <- rand_forest(mtry = 13,
                       min_n = 8,
                       trees = 750) %>%
  set_engine("ranger") %>%
  set_mode("regression")

rf_wflow <- workflow() %>%
  add_recipe(interact_rec) %>%
  add_model(rf_spec)
```

Models were trained on all training set data, and then fit on the test data. The results from training and testing were collected to evaluate model performance and identify if over fitting occurred.

```
mars_fit <- fit(mars_wflow, data = soil_train)
rf_fit <- fit(rf_wflow, data = soil_train)
```

```
results_train <- mars_fit %>%
  predict(new_data = soil_train) %>%
  mutate(truth = soil_train$ta_true,
         model = "MARS") %>%
  bind_rows(rf_fit %>%
    predict(new_data = soil_train) %>%
    mutate(truth = soil_train$ta_true,
           model = "RF"))

results_test <- mars_fit %>%
  predict(new_data = soil_test) %>%
  mutate(truth = soil_test$ta_true,
         model = "MARS") %>%
  bind_rows(rf_fit %>%
    predict(new_data = soil_test) %>%
    mutate(truth = soil_test$ta_true,
           model = "RF"))
```

```
results_train %>%
  group_by(model) %>%
  rmse(truth = 10^truth, estimate = 10^.pred) %>%
  bind_rows(
    results_train %>%
      group_by(model) %>%
```

```
    mae(truth = 10^truth, estimate = 10^.pred)
) %>%
bind_rows(
  results_train %>%
    group_by(model) %>%
    rsq(truth = 10^truth, estimate = 10^.pred)
)
```

```
## # A tibble: 6 x 4
##   model .metric .estimator .estimate
##   <chr> <chr>   <chr>          <dbl>
## 1 MARS  rmse    standard      0.0742
## 2 RF    rmse    standard      0.0956
## 3 MARS  mae     standard      0.0211
## 4 RF    mae     standard      0.0266
## 5 MARS  rsq     standard      0.988
## 6 RF    rsq     standard      0.981
```

The models performed similarly during training, with the MARS model marginally outperforming the random forest model. The R^2 value is noticeably high for both models.

Expected model performance

The values below identify model performance on unseen data. These values are to be reported as an estimate of each model's true expected performance. This allows us to make an estimate of each models ability to generalise to new, unseen data.

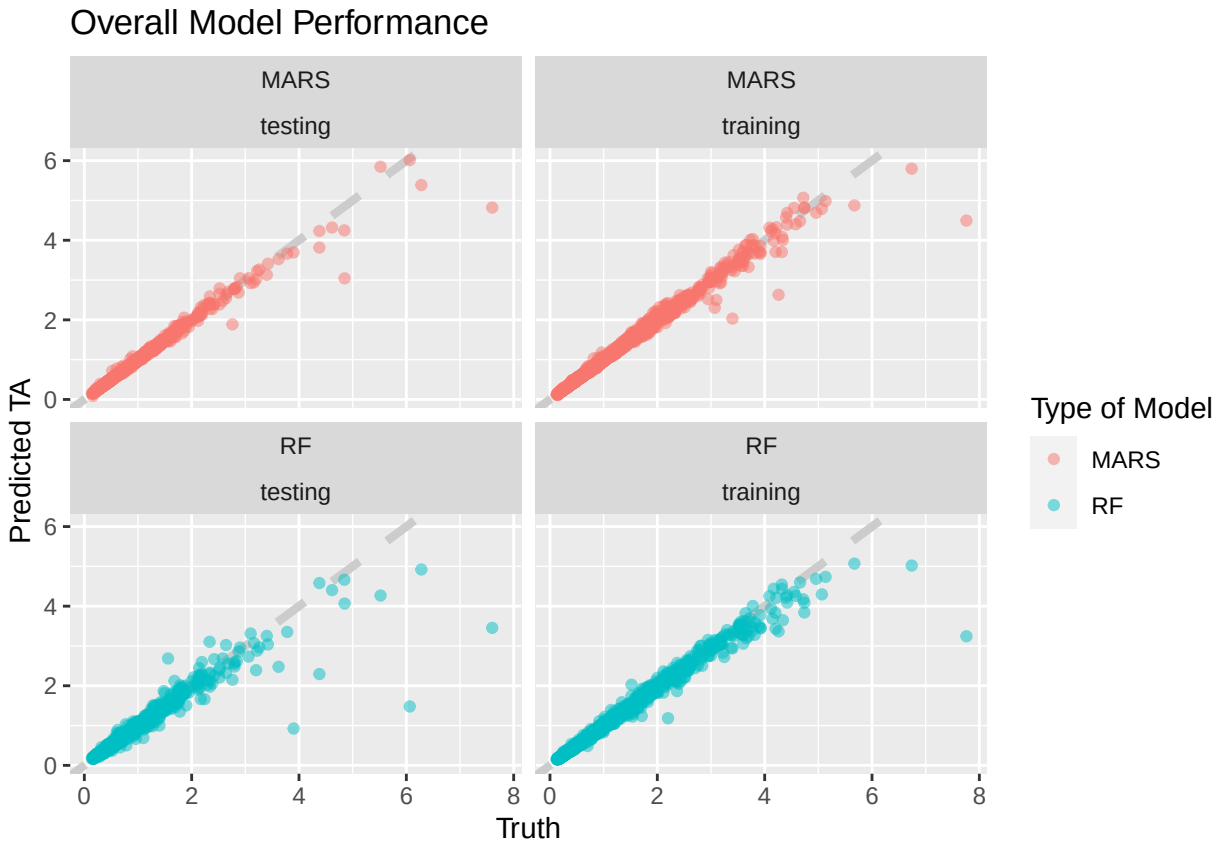
```
results_test %>%
  group_by(model) %>%
  rmse(truth = 10^truth, estimate = 10^.pred) %>%
  bind_rows(
    results_test %>%
      group_by(model) %>%
      mae(truth = 10^truth, estimate = 10^.pred)
  ) %>%
  bind_rows(
    results_test %>%
      group_by(model) %>%
      rsq(truth = 10^truth, estimate = 10^.pred)
  )
```

```
## # A tibble: 6 x 4
##   model .metric .estimator .estimate
##   <chr> <chr>   <chr>          <dbl>
## 1 MARS  rmse    standard       0.101
## 2 RF    rmse    standard       0.215
## 3 MARS  mae     standard      0.0233
## 4 RF    mae     standard      0.0574
## 5 MARS  rsq     standard       0.978
## 6 RF    rsq     standard       0.896
```

The values in the table above are to be the reported values for final model performance. It is clear from these results that the random forest slightly over fitted on the training set.

Were a model to be developed with perhaps a smaller number of trees in the ensemble, generalisation is expected to improve. It could also lead to a model that trains faster, due to the decrease in model complexity.

```
results_test %>%
  mutate(train = "testing",
         .pred = 10^.pred,
         truth = 10^truth) %>%
  bind_rows(results_train %>%
    mutate(train = "training",
           .pred = 10^.pred,
           truth = 10^truth)) %>%
  ggplot(mapping = aes(x = truth, y = .pred, colour = model)) +
  geom_abline(lty = 2, colour = "gray80", size = 1.5) +
  geom_point(alpha = 0.5) +
  facet_wrap(model ~ train) +
  labs(
    x = "Truth",
    y = "Predicted TA",
    colour = "Type of Model",
    title = "Overall Model Performance"
  )
```



From the graph above both models start becoming erratic in their predictions at high titratable acidity values. This was attributed to a lower number of values present in that titratable acidity range.

Nevertheless, the MARS model performed much better in those sparse regions and showed acceptable generalisation capacity.

Conclusion

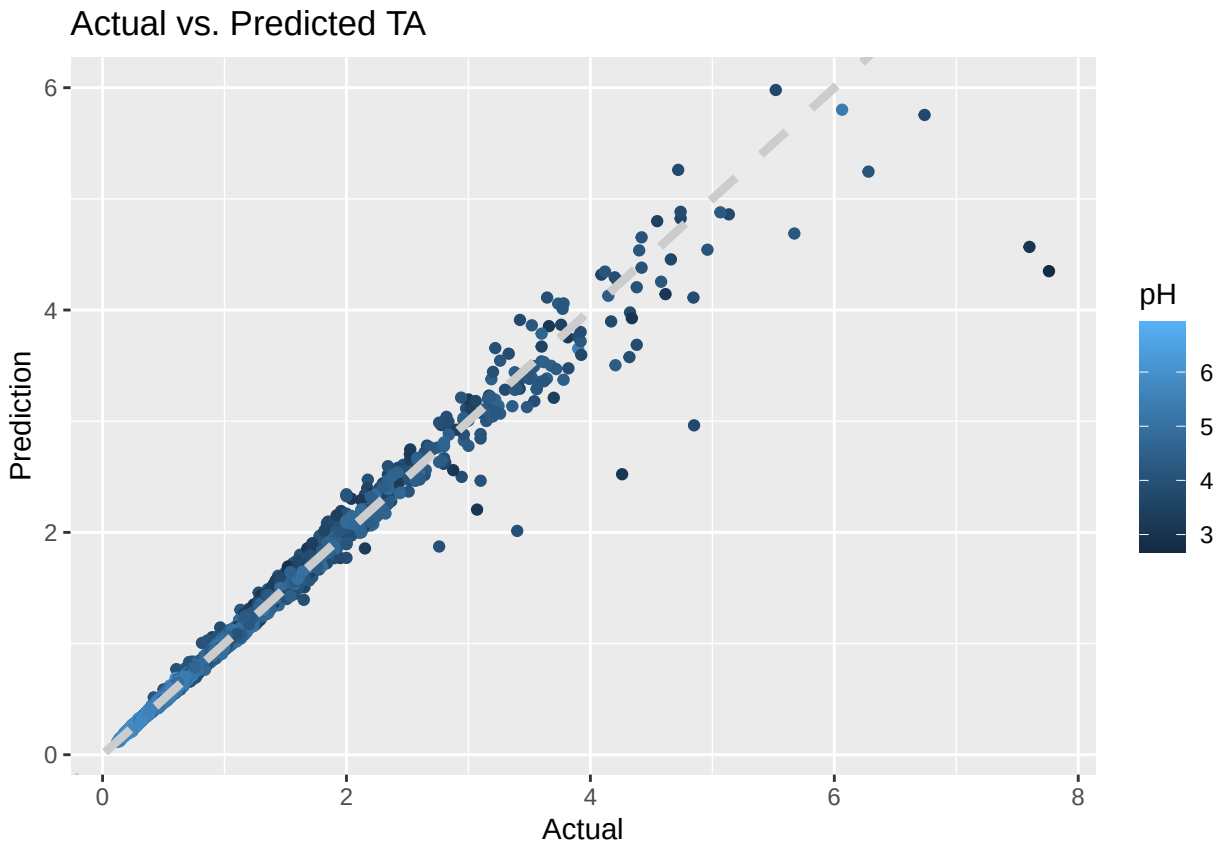
The MARS model with a highest possible interaction degree of one has shown to be the best fitting model for this data set. Exploratory analysis found that only 6 variables have the highest predictive ability, and include **ta_eksteen**, **r_observed**, **r_eksteen**, **pH**, **carbon** and **ec_Ca**. Creating interaction terms between these variables and removing terms with correlations greater than 0.75 proved to be a sufficient preprocessing technique. The model performed well on the testing set, showing very little sign of over fitting.

The random forest model performed well during training, however fared considerably poorly on the test set. It is clear that the random forest model over fit to the training data. It is thus recommended that the total number of trees relied upon in the ensemble be lessened to avoid over fitting. It should be noted that the MARS model be used for predictive purposes due to its ability to be fit quickly, its simplicity and robustness to noise in this data set.

Below, the MARS model is trained on all data and predicted values are appended to the complete data set, to be used for deeper analysis.

```
# Define data preprocessing recipe
final_build_data <- main_data %>%
  mutate(ta_true_log = log10(ta_true))

main_rec <- recipe(ta_true_log ~ ta_eksteen + r_observed + r_eksteen +
                   pH + carbon + bs_Ca, data = final_build_data) %>%
  step_interact(~ all_numeric_predictors():all_numeric_predictors()) %>%
  step_corr(threshold = 0.75)
# Define model
target_model <- mars(prod_degree = 1) %>%
  set_engine("earth") %>%
  set_mode("regression")

model_wflow <- workflow() %>%
  add_recipe(main_rec) %>%
  add_model(target_model)

model_fit <- fit(model_wflow, data = final_build_data)
```

```
final_data <- augment(model_fit, new_data = final_build_data) %>%
  mutate(.pred = 10^(.pred))

final_data %>%
  ggplot(mapping = aes(x = ta_true, y = .pred, colour = pH)) +
  geom_point() +
  geom_abline(lty = 2, colour = "gray80", size = 1.5) +
  labs(title = "Actual vs. Predicted TA",
       x = "Actual", y = "Prediction")
```




```
# write_csv(final_data, file = "./data/final_data.csv")
```

Final model results

```
final_data %>%
  rsq(truth = ta_true, estimate = .pred) %>%
  bind_rows(
    final_data %>%
      rmse(truth = ta_true, estimate = .pred)) %>%
      bind_rows(
        final_data %>%
          mae(truth = ta_true, estimate = .pred))
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard      0.981
## 2 rmse    standard      0.0913
## 3 mae     standard      0.0275
```