MachinelearningonWine

Harrie

22/05/2021

```
library(tidyverse)
```

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

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.1     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

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

```
library(finalfit)
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 0.1.2 --

## v broom     0.7.3      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.8
## v infer     0.5.3      v tune      0.1.2
## v modeldata 0.1.0      v workflows 0.2.1
## v parsnip   0.1.4      v yardstick 0.0.7

## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x scales::discard() masks purrr::discard()
## x dplyr::filter()   masks stats::filter()
## x recipes::fixed()  masks stringr::fixed()
## x dplyr::lag()      masks stats::lag()
## x yardstick::spec() masks readr::spec()
## x recipes::step()   masks stats::step()
```

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'
```

```
## The following objects are masked from 'package:yardstick':
##
##     precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(rpart)
```

```
##
## Attaching package: 'rpart'

## The following object is masked from 'package:dials':
##
##     prune
```

```
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

INTRODUCTION

`Tidymodels` is the successor to the `caret` package which is used during the *Introduction to Data Science*-course of the Harvard University (Kuhn & Johnson, 2013; Irizarry, 2020). `Tidymodels` is a collection of modeling packages that, like the `tidyverse`, have consistent API and are designed to work together specifically to support predictive analytics and machine learning. Different books (Kuhn & Silge, 2021; Kuhn en Johnson, 2019) blogs (Lendway, 2020; Roamiar (2021); Ruiz (2019), Barter (2019; Seyedia (2021) and couses/video's (Lewis, 2020; Silge, 2021; Silge 2020) I followed and looked at. I tried to learn this new system and wrote different blogs in Dutch on this (Jonkman, 2021).

Core tidymodel packages include: parsnip, recipes, rsample and tune. Collectively, these packages provide a grammar for modeling that makes things a lot easier and provide a unified modeling and analysis interface to seamlessly access several model varieteis in R.

- `tidymodels` is a meta-package that installs and load the core packages listed below that you need for modeling and machine learning;

- `recipes` is tidy interface for to data pre-processing tools for feature/variables engineering;
- `rsample` provides infrastucture for efficient data splitting and resampling;
- `parsnip` is a tidy, unified interface to models that can be used to try a range of models without getting bagged down in the syntactical minutae of the underlying packages;
- `tune` helps you optimize the hyperparameters of your model and chose pre-processing steps;
- `yardstick` measures the effectiveness of models during performance metrics;
- `workflow` bundles your pre-processing modeling and post-processing together;
- `dials` creates and manages tuning parameters and parameters grids;
- `brooms` convert the information in common statistical R objects into user-friendly predictable formats.

Last months I tried to learn working with `tidymodels`. I tried to finish the Capstone course with the use of this packages.

PROBLEM DEFINITION

Data mining approaches are use to predict human wine taste preferences that are based on easily available analytical tests at certification step. A dataset on red wine from Portugal is used here to research quality of the wine and different predictors (Cortez et al., 2009) Supervised machine learning support us in this. In this world two kind of algorithms are often used. One is called regression (see also Attalides, 2020 for regression, which I expand in this articel) and the other is called classification (which I worked out also).

In this study we use regression for predicting quality of wine based on the predictors.The wine data used here contains the following independent and dependent variables

Independent variables: (symbol I) - I1 *Fixed acidity* (g(tartaric acid/dm3) - I2 *Volatile acidity* (g(acetic acid)/dm3) - I3 *Citric acid* (g/dm3) - I4 *Residual sugar* (g/dm3) - I5 *Chlorides* (g(sodium chloride)/dm3) - I6 *Free sulfar dioxide* (mg/dm3) - I7 *Total sulfar dioxide* (mg/dm3) - I8 *Density* (g/cm3) - I9 *pH* - I10 *Sulphates* (g(potassium sulphate)/dm3) - I11 *Alcohol* (vol%)

Dependent variable: (symbol D) - D1 *Quality*

DATA LOADING AND PREPROCESSING

Let us first load the dataset.

```
wf<-readRDS("wine.rds")
```

Let us give a summary of the data frame.

```
glimpse(wf)
```

```
## Rows: 1,599
## Columns: 12
## $ ‘fixed acidity‘        <dbl> 7.4, 7.8, 7.8, 11.2, 7.4, 7.4, 7.9, 7.3, 7.8, 7~
```

```
## $ `volatile acidity`     <dbl> 0.700, NA, 0.760, 0.280, 0.700, NA, 0.600, 0.65~
## $ `citric acid`          <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.06, 0.00,~
## $ `residual sugar`       <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2, 2.0, 6.~
## $ chlorides              <dbl> 0.076, 0.098, 0.092, 0.075, 0.076, 0.075, 0.069~
## $ `free sulfur dioxide`  <dbl> 11, 25, 15, 17, 11, 13, 15, 15, 9, 17, 15, 17, ~
## $ `total sulfur dioxide` <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 102, 65, 10~
## $ density                <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978, 0.9978,~
## $ pH                     <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.30, 3.39,~
## $ sulphates              <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.46, 0.47,~
## $ alcohol                <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, NA, 9.5, 10.~
## $ quality                <dbl> 5, 5, 5, 6, 5, 5, 5, 7, 7, 5, 5, 5, 5, 5, 5, 5,~
```

Do we see missings?

```
missing_glimpse(wf)
```

```
##                                   label var_type    n missing_n
## fixed.acidity              fixed.acidity    <dbl> 1599         0
## volatile.acidity        volatile.acidity    <dbl> 1596         3
## citric.acid                  citric.acid    <dbl> 1599         0
## residual.sugar            residual.sugar    <dbl> 1599         0
## chlorides                      chlorides    <dbl> 1599         0
## free.sulfur.dioxide   free.sulfur.dioxide    <dbl> 1599         0
## total.sulfur.dioxide total.sulfur.dioxide    <dbl> 1599         0
## density                         density    <dbl> 1599         0
## pH                                   pH    <dbl> 1599         0
## sulphates                     sulphates    <dbl> 1599         0
## alcohol                         alcohol    <dbl> 1594         5
## quality                         quality    <dbl> 1599         0
##                       missing_percent
## fixed.acidity                     0.0
## volatile.acidity                  0.2
## citric.acid                       0.0
## residual.sugar                    0.0
## chlorides                         0.0
## free.sulfur.dioxide               0.0
## total.sulfur.dioxide              0.0
## density                           0.0
## pH                                0.0
## sulphates                         0.0
## alcohol                           0.3
## quality                           0.0
```

What kind of variables do we have and what are their scores?

```
ff_glimpse(wf)
```

```
## $Continuous
##                                   label var_type    n missing_n
## fixed.acidity              fixed.acidity    <dbl> 1599         0
## volatile.acidity        volatile.acidity    <dbl> 1596         3
## citric.acid                  citric.acid    <dbl> 1599         0
```

```
## residual.sugar             residual.sugar    <dbl> 1599         0
## chlorides                       chlorides    <dbl> 1599         0
## free.sulfur.dioxide   free.sulfur.dioxide    <dbl> 1599         0
## total.sulfur.dioxide total.sulfur.dioxide    <dbl> 1599         0
## density                           density    <dbl> 1599         0
## pH                                     pH    <dbl> 1599         0
## sulphates                     sulphates    <dbl> 1599         0
## alcohol                         alcohol    <dbl> 1594         5
## quality                         quality    <dbl> 1599         0
##                       missing_percent mean   sd min quartile_25 median
## fixed.acidity                     0.0  8.3  1.7 4.6         7.1    7.9
## volatile.acidity                  0.2  0.5  0.2 0.1         0.4    0.5
## citric.acid                       0.0  0.3  0.2 0.0         0.1    0.3
## residual.sugar                    0.0  2.5  1.4 0.9         1.9    2.2
## chlorides                         0.0  0.1  0.0 0.0         0.1    0.1
## free.sulfur.dioxide               0.0 15.9 10.5 1.0         7.0   14.0
## total.sulfur.dioxide              0.0 46.5 32.9 6.0        22.0   38.0
## density                           0.0  1.0  0.0 1.0         1.0    1.0
## pH                                0.0  3.3  0.2 2.7         3.2    3.3
## sulphates                         0.0  0.7  0.2 0.3         0.6    0.6
## alcohol                           0.3 10.4  1.1 8.4         9.5   10.2
## quality                           0.0  5.6  0.8 3.0         5.0    6.0
##                       quartile_75   max
## fixed.acidity                 9.2  15.9
## volatile.acidity              0.6   1.6
## citric.acid                   0.4   1.0
## residual.sugar                2.6  15.5
## chlorides                     0.1   0.6
## free.sulfur.dioxide          21.0  72.0
## total.sulfur.dioxide         62.0 289.0
## density                       1.0   1.0
## pH                            3.4   4.0
## sulphates                     0.7   2.0
## alcohol                      11.1  14.9
## quality                       6.0   8.0
##
## $Categorical
## # A tibble: 1,599 x 0
```

Let us make correlation matrix of the data.

```
cor(wf)
```

```
##                       fixed acidity volatile acidity citric acid residual sugar
## fixed acidity           1.00000000               NA  0.67170343    0.114776724
## volatile acidity                NA                1          NA             NA
## citric acid             0.67170343               NA  1.00000000    0.143577162
## residual sugar          0.11477672               NA  0.14357716    1.000000000
## chlorides               0.09370519               NA  0.20382291    0.055609535
## free sulfur dioxide    -0.15379419               NA -0.06097813    0.187048995
## total sulfur dioxide   -0.11318144               NA  0.03553302    0.203027882
## density                 0.66804729               NA  0.36494718    0.355283371
## pH                     -0.68297819               NA -0.54190414   -0.085652422
```

```
## sulphates               0.18300566               NA  0.31277004    0.005527121
## alcohol                        NA               NA          NA             NA
## quality                 0.12405165               NA  0.22637251    0.013731637
##                         chlorides free sulfur dioxide total sulfur dioxide
## fixed acidity         0.093705186        -0.153794193          -0.11318144
## volatile acidity               NA                  NA                   NA
## citric acid           0.203822914        -0.060978129           0.03553302
## residual sugar        0.055609535         0.187048995           0.20302788
## chlorides             1.000000000         0.005562147           0.04740047
## free sulfur dioxide   0.005562147         1.000000000           0.66766645
## total sulfur dioxide  0.047400468         0.667666450           1.00000000
## density               0.200632327        -0.021945831           0.07126948
## pH                   -0.265026131         0.070377499          -0.06649456
## sulphates             0.371260481         0.051657572           0.04294684
## alcohol                        NA                  NA                   NA
## quality              -0.128906560        -0.050656057          -0.18510029
##                           density          pH    sulphates alcohol     quality
## fixed acidity          0.66804729 -0.68297819  0.183005664      NA  0.12405165
## volatile acidity              NA          NA           NA      NA          NA
## citric acid            0.36494718 -0.54190414  0.312770044      NA  0.22637251
## residual sugar         0.35528337 -0.08565242  0.005527121      NA  0.01373164
## chlorides              0.20063233 -0.26502613  0.371260481      NA -0.12890656
## free sulfur dioxide   -0.02194583  0.07037750  0.051657572      NA -0.05065606
## total sulfur dioxide   0.07126948 -0.06649456  0.042946836      NA -0.18510029
## density                1.00000000 -0.34169933  0.148506412      NA -0.17491923
## pH                    -0.34169933  1.00000000 -0.196647602      NA -0.05773139
## sulphates              0.14850641 -0.19664760  1.000000000      NA  0.25139708
## alcohol                       NA          NA           NA       1          NA
## quality               -0.17491923 -0.05773139  0.251397079      NA  1.00000000
```

Let us define the column names on a consistent way.

```
colnames(wf) <- wf %>%
  colnames() %>% str_replace_all(pattern = " ", replacement = "_")
colnames(wf)
```

```
##  [1] "fixed_acidity"        "volatile_acidity"     "citric_acid"
##  [4] "residual_sugar"       "chlorides"            "free_sulfur_dioxide"
##  [7] "total_sulfur_dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
```

Let us also remove any missing values.

```
wf <- na.omit(wf)
```

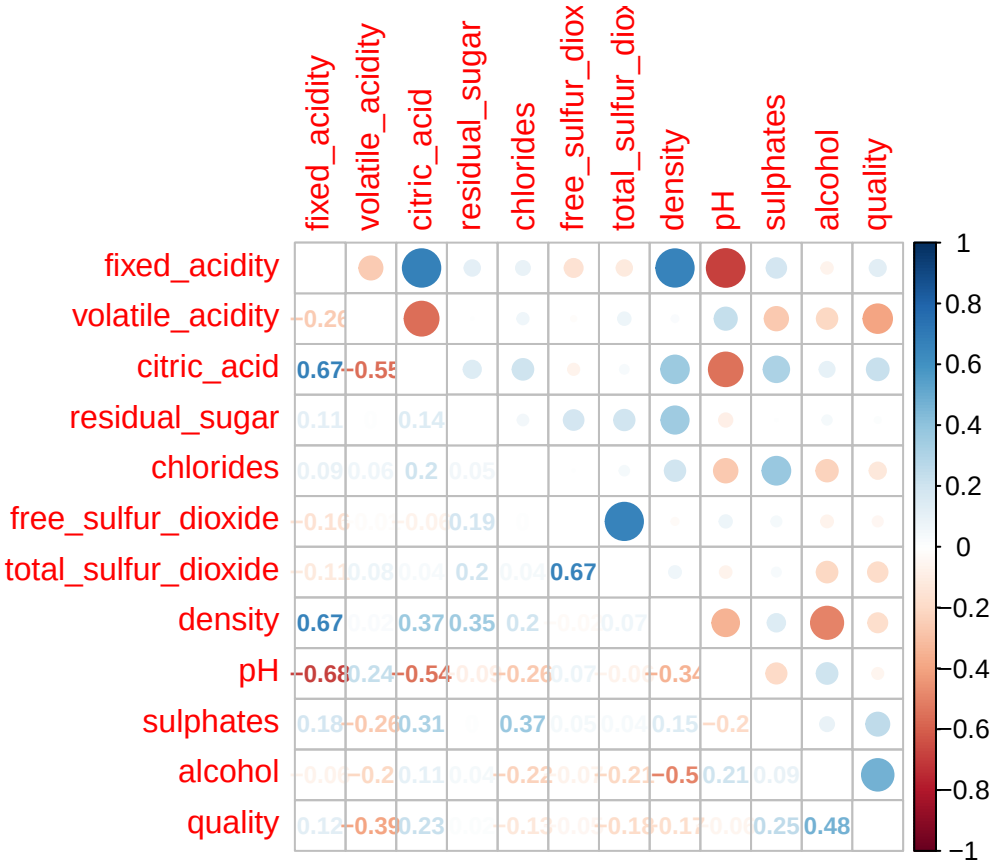
EXPLORATIVE DATA ANALYSIS (EDA).

Now we have preprocessed the data we explore them. let us first visualise correlations within the dataset. For this you need the package `corrplot`

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
wf %>% cor() %>%
  corrplot.mixed(upper = "circle",
                 tl.cex = 1,
                 tl.pos = 'lt',
                 number.cex = 0.75)
```



SPLITTING THE DATA

Now we split the data into: a) Train set, b) Test set. Here we work on the last pre-model analysis. All functions below come from the `rsample` package, which is part of `tidymodels`. First we set the seed to fix the randomisation and to make reproducable possible. We use 80% of the dataset for the trainingset. We split it and than make a training- and test-dataset

```
set.seed(12345)

data_split <- initial_split(wf, prop = 0.8)

train_data <- training(data_split)

test_data  <- testing(data_split)
```

MODELING AND DATA ANALYSIS

1. Lineair modelling

For the continues outcome or target variable `quality` we first research some different lineair regression models and choose the best one. For the below tasks, we store each formula in a different R object.

We have to define the data: - The target variable. `quality` is the target variable and it is numeric - The features of the model (predictors) are the other variables here and they are numeric also.

Futhermore, we esign a simple formula to predict the target variable. In this formula (f1) all the available 11 predictors are used.

```
formula <- formula(quality ~ fixed_acidity + volatile_acidity + citric_acid +
                   residual_sugar + chlorides + free_sulfur_dioxide +
                   total_sulfur_dioxide + density + pH + sulphates + alcohol)
```

Let us fit a linear regression model to the data. What we do: - We created an object that will store the model fit.
- We specify the model.
- We specify also that we work with regression because of the continue target variable (`quality`) - We specify also the `lm` package to train the model - We add the formula and the training data to fit the model

```
lm_fit <-
  linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm") %>%
  fit(formula, data = train_data)
```

Show the results

```
print(lm_fit$fit)
```

```
##
## Call:
## stats::lm(formula = quality ~ fixed_acidity + volatile_acidity +
##     citric_acid + residual_sugar + chlorides + free_sulfur_dioxide +
##     total_sulfur_dioxide + density + pH + sulphates + alcohol,
##     data = data)
##
## Coefficients:
##          (Intercept)         fixed_acidity      volatile_acidity
##            18.274260              0.010860             -1.027444
##          citric_acid        residual_sugar             chlorides
##            -0.078096              0.010779             -1.791911
##  free_sulfur_dioxide  total_sulfur_dioxide              density
##             0.006088             -0.003397            -14.013720
##                   pH             sulphates               alcohol
##            -0.433262              0.971165              0.267353
```

Show the results in another way.

```r
summary(lm_fit$fit)
```

```
##
## Call:
## stats::lm(formula = quality ~ fixed_acidity + volatile_acidity +
##     citric_acid + residual_sugar + chlorides + free_sulfur_dioxide +
##     total_sulfur_dioxide + density + pH + sulphates + alcohol,
##     data = data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7300 -0.3503 -0.0440  0.4596  2.0380
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           18.274260  23.733238   0.770 0.441452
## fixed_acidity          0.010860   0.028947   0.375 0.707601
## volatile_acidity      -1.027444   0.133217  -7.713 2.49e-14 ***
## citric_acid           -0.078096   0.161267  -0.484 0.628280
## residual_sugar         0.010779   0.016606   0.649 0.516372
## chlorides             -1.791911   0.496353  -3.610 0.000318 ***
## free_sulfur_dioxide    0.006088   0.002409   2.527 0.011620 *
## total_sulfur_dioxide  -0.003397   0.000804  -4.225 2.56e-05 ***
## density              -14.013720  24.214779  -0.579 0.562877
## pH                    -0.433262   0.210939  -2.054 0.040185 *
## sulphates              0.971165   0.127820   7.598 5.84e-14 ***
## alcohol                0.267353   0.029843   8.959  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6482 on 1261 degrees of freedom
## Multiple R-squared:  0.3512, Adjusted R-squared:  0.3456
## F-statistic: 62.06 on 11 and 1261 DF,  p-value: < 2.2e-16
```

We can also visualise the fit summary by using the `broom` package which is inside `tidymodels`.

```r
tidy(lm_fit$fit) %>% mutate_if(is.numeric, round, 3)
```

```
## # A tibble: 12 x 5
##    term                estimate std.error statistic p.value
##    <chr>                  <dbl>     <dbl>     <dbl>   <dbl>
##  1 (Intercept)             18.3      23.7      0.77    0.441
##  2 fixed_acidity            0.011     0.029    0.375   0.708
##  3 volatile_acidity        -1.03      0.133   -7.71    0
##  4 citric_acid             -0.078     0.161   -0.484   0.628
##  5 residual_sugar           0.011     0.017    0.649   0.516
##  6 chlorides               -1.79      0.496   -3.61    0
##  7 free_sulfur_dioxide      0.006     0.002    2.53    0.012
##  8 total_sulfur_dioxide    -0.003     0.001   -4.22    0
##  9 density                -14.0      24.2     -0.579   0.563
## 10 pH                      -0.433     0.211   -2.05    0.04
## 11 sulphates                0.971     0.128    7.60    0
## 12 alcohol                  0.267     0.03     8.96    0
```

2. Decision tree

After we worked with lineair regression, it is possible to work with other models which maybe give us better results. Let us first look at decision tree. For this you need decision tree and for this you have to install and open the library of `rpart`. We see similar steps here - defining an object `dt_fit`
- telling that we work with decision tree
- once again set the mode on regression
- once again set the engine on `rpart`
- fit the formula on the training data-set

```
dt_fit <-
  decision_tree() %>%
  set_mode("regression") %>%
  set_engine("rpart") %>%
  fit(formula, data = train_data)
```

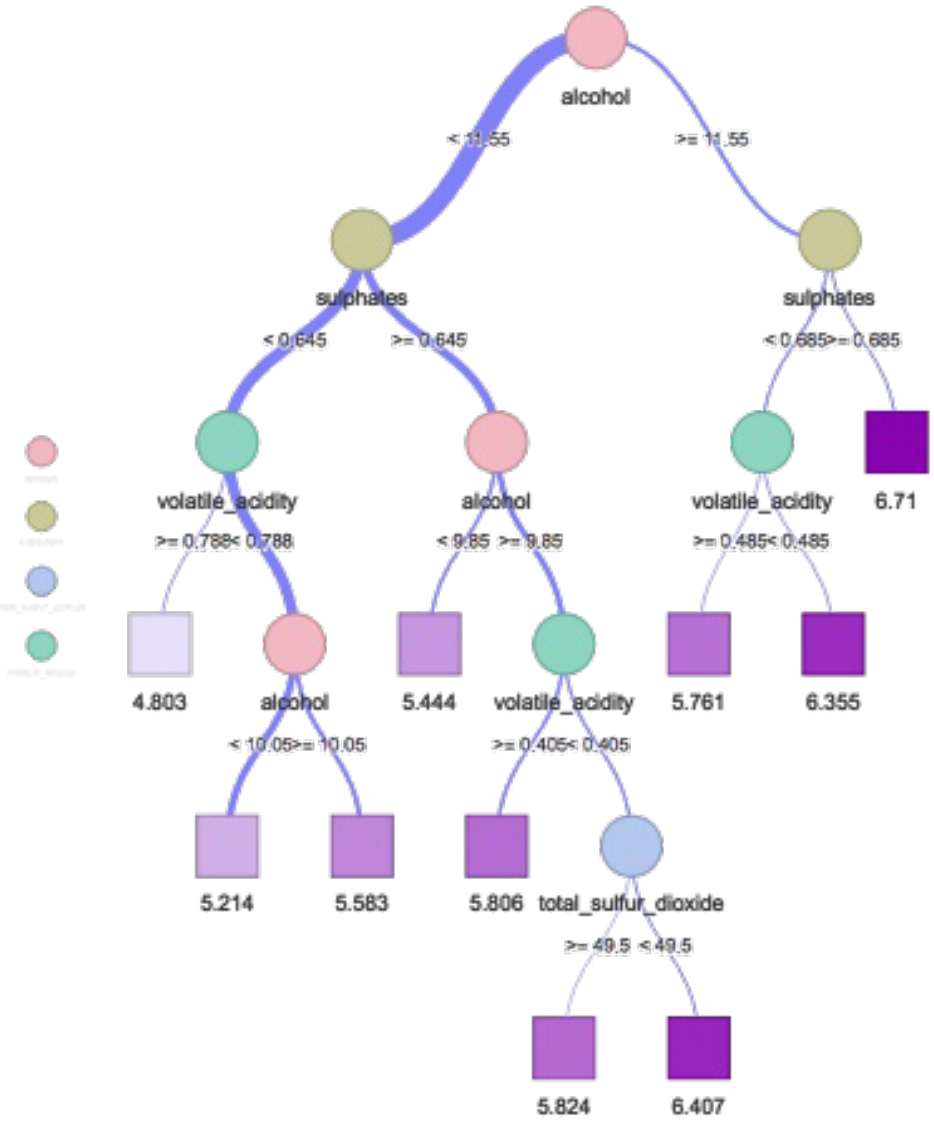
Print the results

```
print(dt_fit$fit)
```

```
## n= 1273
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 1273 816.65670 5.641791
##    2) alcohol< 11.55 1072 573.99160 5.502799
##      4) sulphates< 0.645 636 270.01730 5.294025
##        8) volatile_acidity>=0.7875 66  40.43939 4.803030 *
##        9) volatile_acidity< 0.7875 570 211.82460 5.350877
##         18) alcohol< 10.05 359 100.48470 5.214485 *
##         19) alcohol>=10.05 211  93.29858 5.582938 *
##      5) sulphates>=0.645 436 235.81650 5.807339
##       10) alcohol< 9.85 151  51.27152 5.443709 *
##       11) alcohol>=9.85 285 154.00000 6.000000
##         22) volatile_acidity>=0.405 160  74.99375 5.806250 *
##         23) volatile_acidity< 0.405 125  65.31200 6.248000
##           46) total_sulfur_dioxide>=49.5 34  14.94118 5.823529 *
##           47) total_sulfur_dioxide< 49.5 91  41.95604 6.406593 *
##    3) alcohol>=11.55 201 111.50250 6.383085
##      6) sulphates< 0.685 108  53.87963 6.101852
##       12) volatile_acidity>=0.485 46  22.36957 5.760870 *
##       13) volatile_acidity< 0.485 62  22.19355 6.354839 *
##      7) sulphates>=0.685 93  39.16129 6.709677 *
```

As a sidestep we can visualise this, but than we have to install and open the `visNetwork` and `sparkline` packages. Then we see this.

```
library(visNetwork)
library(sparkline)
visTree(dt_fit$fit)
```



Export as png

3. Random forest

A third model we use here is RandomForest.You need to install `randomForest` and open the library `randomForest`. And again the same steps: - define object `rf_fit`
- tell we want to use randomforst
- set the mode again on regression
- set the engine here on randomForest
- fit the model on the training_set

```
rf_fit <-
  rand_forest() %>%
  set_mode("regression") %>%
  set_engine("randomForest") %>%
  fit(formula, data = train_data)
```

Print these results.

```
print(rf_fit$fit)
```

```
##
## Call:
##  randomForest(x = maybe_data_frame(x), y = y)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.3392329
##                     % Var explained: 47.12
```

EVALUATION AND PREDICTION

Now we have three objects which we have to evaluate. We do this on the test-set. We compare three models (`lm_fit`, `dt_fit` and `rf_fit`) on the MSE score.

Accuracy of the lm-model

```
lm_pred <- test_data %>%
  bind_cols(predict(object = lm_fit, new_data = test_data))
```

Now we see a new column, `.pred`, with a predicted scores for each row.

```
View(lm_pred)
```

```
lm_pred <- test_data %>%
  bind_cols(predict(object = lm_fit, new_data = test_data)) %>%
  mutate(pred = round(.pred, 0))
```

```r
lm_mse <- lm_pred %>%
  summarise(type = "lm",
            MSE = round(mean((pred - quality)^2), 4))
```

```r
head(lm_mse)
```

```
## # A tibble: 1 x 2
##   type    MSE
##   <chr> <dbl>
## 1 lm    0.484
```

Accuracy of the Decision Tree Model

```r
dt_pred <- test_data %>%
  bind_cols(predict(object = dt_fit, new_data = test_data)) %>%
  rename(pred = .pred) %>%
  mutate(pred = round(pred, 0))
```

```r
dt_mse <- dt_pred %>%
  summarise(type = "dt",
            MSE = round(mean((pred - quality)^2), 4))
```

```r
head(dt_mse)
```

```
## # A tibble: 1 x 2
##   type    MSE
##   <chr> <dbl>
## 1 dt    0.531
```

Accuracy of the Random Forest Model

```r
rf_pred <- test_data %>%
  bind_cols(predict(object = rf_fit, new_data = test_data)) %>%
  rename(pred = .pred) %>%
  mutate(pred = round(pred, 0))
```

```r
rf_mse <- rf_pred %>%
  summarise(type = "rf",
            MSE = round(mean((pred - quality)^2), 4))
```

```r
head(rf_mse)
```

```
## # A tibble: 1 x 2
##   type    MSE
##   <chr> <dbl>
## 1 rf    0.380
```

All results together

Let us put all the results together.

```
res <- bind_rows(lm_mse, dt_mse, rf_mse)
```

Let us show these results.

```
head(res)
```

```
## # A tibble: 3 x 2
##   type    MSE
##   <chr> <dbl>
## 1 lm    0.484
## 2 dt    0.531
## 3 rf    0.380
```

We choose the random_forest model as the best opportunity here. Let us look at it once again.

```
View(rf_pred)
```

```
head(rf_pred)
```

```
## # A tibble: 6 x 13
##   fixed_acidity volatile_acidity citric_acid residual_sugar chlorides
##           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
## 1           7.4             0.7         0               1.9     0.076
## 2           8.5             0.28        0.56            1.8     0.092
## 3           8.1             0.56        0.28            1.7     0.368
## 4           7.4             0.59        0.08            4.4     0.086
## 5           7.9             0.43        0.21            1.6     0.106
## 6           6.3             0.39        0.16            1.4     0.08
## # ... with 8 more variables: free_sulfur_dioxide <dbl>,
## #   total_sulfur_dioxide <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
## #   alcohol <dbl>, quality <dbl>, pred <dbl>
```

RESULTS SUMMARIZED

In this simple scenario, we may be interested in seeing how the model performs on the testing data that was left out. The code below will fit the model to the training data and apply it to the testing data. There are other ways we could have done this, but the way we do it here will be useful when we start using more complex models where we need to tune model parameters.

After the model is fit and applied, we collect the performance metrics and display them and show the predictions from the testing data.

```
head(rf_pred)
```

```
## # A tibble: 6 x 13
##   fixed_acidity volatile_acidity citric_acid residual_sugar chlorides
```

```
##           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
## 1           7.4             0.7         0              1.9     0.076
## 2           8.5             0.28        0.56           1.8     0.092
## 3           8.1             0.56        0.28           1.7     0.368
## 4           7.4             0.59        0.08           4.4     0.086
## 5           7.9             0.43        0.21           1.6     0.106
## 6           6.3             0.39        0.16           1.4     0.08
## # ... with 8 more variables: free_sulfur_dioxide <dbl>,
## #   total_sulfur_dioxide <dbl>, density <dbl>, pH <dbl>, sulphates <dbl>,
## #   alcohol <dbl>, quality <dbl>, pred <dbl>
```

```
metrics(rf_pred, quality, pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.617
## 2 rsq     standard       0.460
## 3 mae     standard       0.355
```

References

Attalides, N. (2020). Introduction to machine learning. Barcelona. Presentation

Barter, R.(2020). Tidymodels: tidy machine learning in R

Baumer, B. Kaplan, D.T., Horton, N.J. (2017). *Modern data science with R.* CRCPress: Boca Raton.

Boehmke, B. & Greenwell, B. (2020). *Hands on machine learning with R.* Bookdown version

Cortez, P., Cerdeira, A., Almeida, F., Matos, T. & Reis, J. (2009). Modeling wine prefernces by data mining from physicochemical properties. *Decision Support Systems, 47*, 547-533.

Hartie, T., Tibskirani, R. & Friedmann, J. (2009). *The elements of statistical learning. Data mining, inference and prediction.* 2nd edition. Springer: New York.

Irizarry, R.A. (2020). *Introduction to data science. Data analysis and prediction algorithms with R.* CRC Press: Boca Raton.

James, S., Witten, D., Hastie, T.. & Tibskirani, R. (2013). *An introduction to statistical learning with application in R.*

Jonkman, H. (2019-2021). Harrie's hoekje, his website with different blogs on machine learning in Dutch

Kuhn, M. 7 Johnson, K. (2013). *Applied predictive modeling.* Springer: New York.

Kuhn, M. & Johnson, K. (2019). Feature engineering and selection: A practical approach for predictive models

Kuhn, M. & Silge, J. (2021). Tydy modeling with R. Bookdown version

Lendway, L. (2020). 2020_north-tidymodels.Introduction on github

Lewis, J.E. (2020). Coding machine learning models

Raoniar, R. (2021). Modeling binary logistic regression using tidymodels library in R (Part 1). Towards data science

Ruiz, E. (2019). A gentle introduction to tidymodels

Seyedian, A. (2021). Medical cost personal datasets. Insurance forcast by using linear regression

Silge, J. (2020). Get started with tidymodels and #TidyTuesdag Palmer penguins

Silge, J. (2021). Supervised machine learning case studies in R

Tidymodels. Tidymodels website