MachinelearningonWine

Harrie

10/12/2021

INTRODUCTION

`Tidymodels` is the relatively new package for machine learning with `R`. It 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`, has consistent API and are designed to work together specifically to support predictive analytics and machine learning. I followed and looked at 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 tried to learn this new system and wrote different blogs in Dutch on this (Jonkman, 2021).

The `tidymodel` package provides a suite of packages for machine learnig as does `tidyverse` does for data wrangling. `Tidymodels` is a grammar for modeling that makes things a lot easier and provides a unified modeling and analysis interface to seamlessly access several model varities 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.

This year I learned working with `tidymodels`. I tried to finish the Capstone course with the use of this packages. I will show you the different steps in the working proces. Let us first open the packages used in this article (`tidymodels`, but also `tidyverse`, `finalfit`, `caret`, `rpart` and `randomforest`):

PROBLEM DEFINITION

Data mining approaches are used in this article to predict human wine taste preferences that are based on easily available analytical tests at certification steps. A data set on red wine from Portugal is used here to research quality of the wine and different predictors for the quality (Cortez et al., 2009) Supervised machine learning supports us in this. In this world two kind of algorithms are often used. One is called regression (see also Attalides, 2020) and the other is called classification (not used here).

In this study we use regression for predicting quality of wine based on several predictors.The wine data used here contains the following eleven independent variables (predictors, I1-I11) and one dependent variable (outcome, D1)

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, PREPROCESSING, EXPLORING

Let us load the data set (`wine.rds`) first.

Then, we look at the column names:

And defined them on a consistent way.

Let us make an overview and summary now of this 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, 7.5~
## $ volatile_acidity     <dbl> 0.700, NA, 0.760, 0.280, 0.700, NA, 0.600, 0.650,~
## $ citric_acid          <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.06, 0.00, 0~
## $ residual_sugar       <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2, 2.0, 6.1,~
## $ 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, 16~
## $ total_sulfur_dioxide <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 102, 65, 102,~
## $ density              <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978, 0.9978, 0~
## $ pH                   <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.30, 3.39, 3~
## $ sulphates            <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.46, 0.47, 0~
## $ alcohol              <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, NA, 9.5, 10.5,~
## $ quality              <dbl> 5, 5, 5, 6, 5, 5, 5, 7, 7, 5, 5, 5, 5, 5, 5, 5, 7~
```

```
summary(wf)
```

```
##  fixed_acidity   volatile_acidity  citric_acid    residual_sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5275   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##                  NA's   :3
##    chlorides       free_sulfur_dioxide total_sulfur_dioxide    density
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00       Min.   :0.9901
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00       1st Qu.:0.9956
##  Median :0.07900   Median :14.00       Median : 38.00       Median :0.9968
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47       Mean   :0.9967
##  3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00       3rd Qu.:0.9978
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00       Max.   :1.0037
##
##        pH          sulphates         alcohol         quality
##  Min.   :2.740   Min.   :0.3300   Min.   : 8.40   Min.   :3.000
##  1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.310   Median :0.6200   Median :10.20   Median :6.000
##  Mean   :3.311   Mean   :0.6581   Mean   :10.43   Mean   :5.636
##  3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10   3rd Qu.:6.000
```

```
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :8.000
##                                   NA's   :5
```

We have twelve variables inside this data set which are all continuous variables.

At this moment we want to know also something about the missings?

We have eight missings (three on volatile.acidity and five on alcohol). We remove any missing values and kept 1591 cases. Let us show it here.

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

missing_glimpse(wf)
```

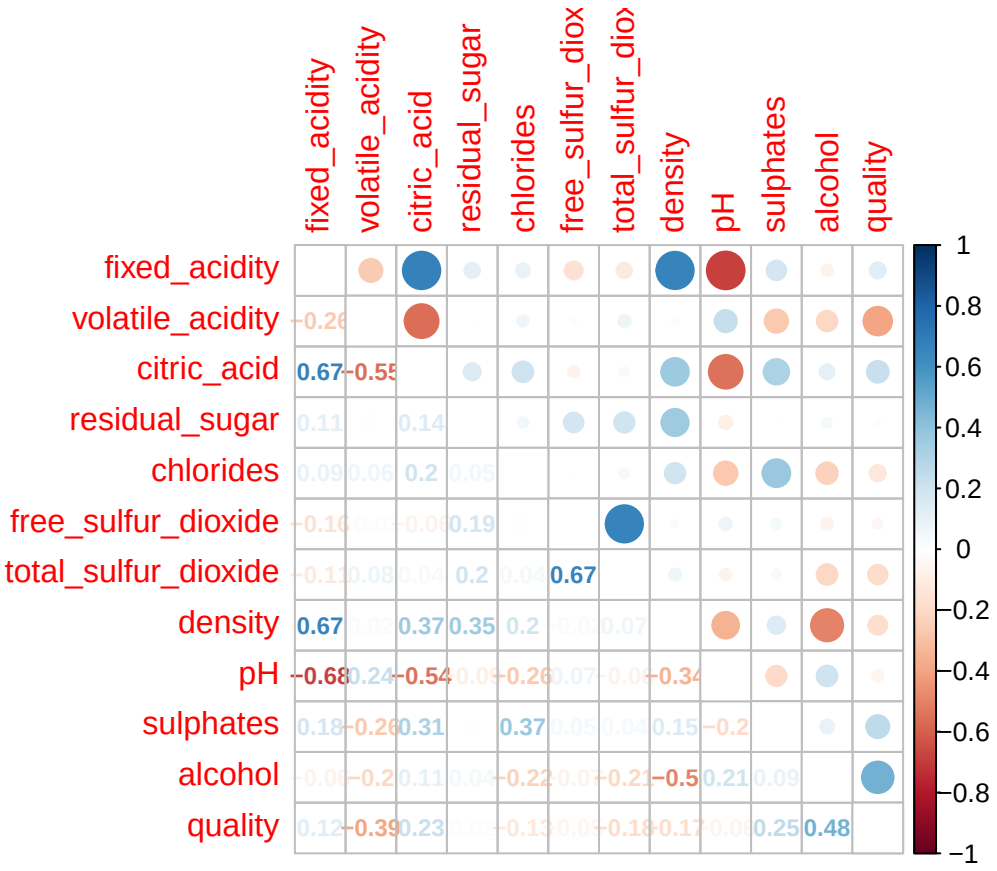
```
##                                    label var_type    n missing_n
## fixed_acidity               fixed_acidity    <dbl> 1591         0
## volatile_acidity         volatile_acidity    <dbl> 1591         0
## citric_acid                   citric_acid    <dbl> 1591         0
## residual_sugar             residual_sugar    <dbl> 1591         0
## chlorides                       chlorides    <dbl> 1591         0
## free_sulfur_dioxide   free_sulfur_dioxide    <dbl> 1591         0
## total_sulfur_dioxide total_sulfur_dioxide    <dbl> 1591         0
## density                           density    <dbl> 1591         0
## pH                                     pH    <dbl> 1591         0
## sulphates                     sulphates    <dbl> 1591         0
## alcohol                         alcohol    <dbl> 1591         0
## quality                         quality    <dbl> 1591         0
##                       missing_percent
## fixed_acidity                     0.0
## volatile_acidity                  0.0
## 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                           0.0
## quality                           0.0
```

Now we have wrangeled and preprocessed the data, we can 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

In this phase we understand the data and have to 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. For a big dataset as this winedata set with 1591 observations, 80:20 works well. We split it and than make a training- and test-dataset

MODELING AND DATA ANALYSIS

Now we compare different models with each other and we want to know which one works the best for this data set, with this dependent and these independent variables. This part of machine learning is called **supervised learning** of which the basic goal is to find a function that accurately describes how different measured explanatory variables can be combined to make a prediction about the target variable. We start with **lineair modellling**. Regression models can help us quantify the magnitude and direction of relationships among variables.

1. Linear modelling

For the outcome or target variable `quality`, we first research some different linear regression models and choose the best one based on indices. For these 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 (independent) 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: - First, we created an object that will store the model fit.
- Then, we specify the model.
- Then, We specify also that we work with regression because of the continue target variable (`quality`) - Then, we specify also the `lm` package to train the model - And we finish in this chunck by adding the formula and the training data to fit the model.

Let us see how this works.

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

We present the results on different ways

But this is probably the best and clearest way to show the results.

```
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 visualize the fit summary by using the `broom` package which is inside `tidymodels`.

```
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 linear regression, it is possible to work with other models which maybe give us better results for predicting the outcome. Let us first look at decision tree modeling . A decision tree is tree-like flowchart that assigns labels to individual observations. It splits it into homogeneous subsets, which share the same class labels. For this you need decision tree package and for this you have to install and open the library of `rpart`. We see similar steps here in the machine learning workflow. Once again: - define an object `dt_fit`
- tell that we work with decision tree
- set the mode on regression
- set the engine on `rpart`
- fit the formula on the training data-set

Print the results

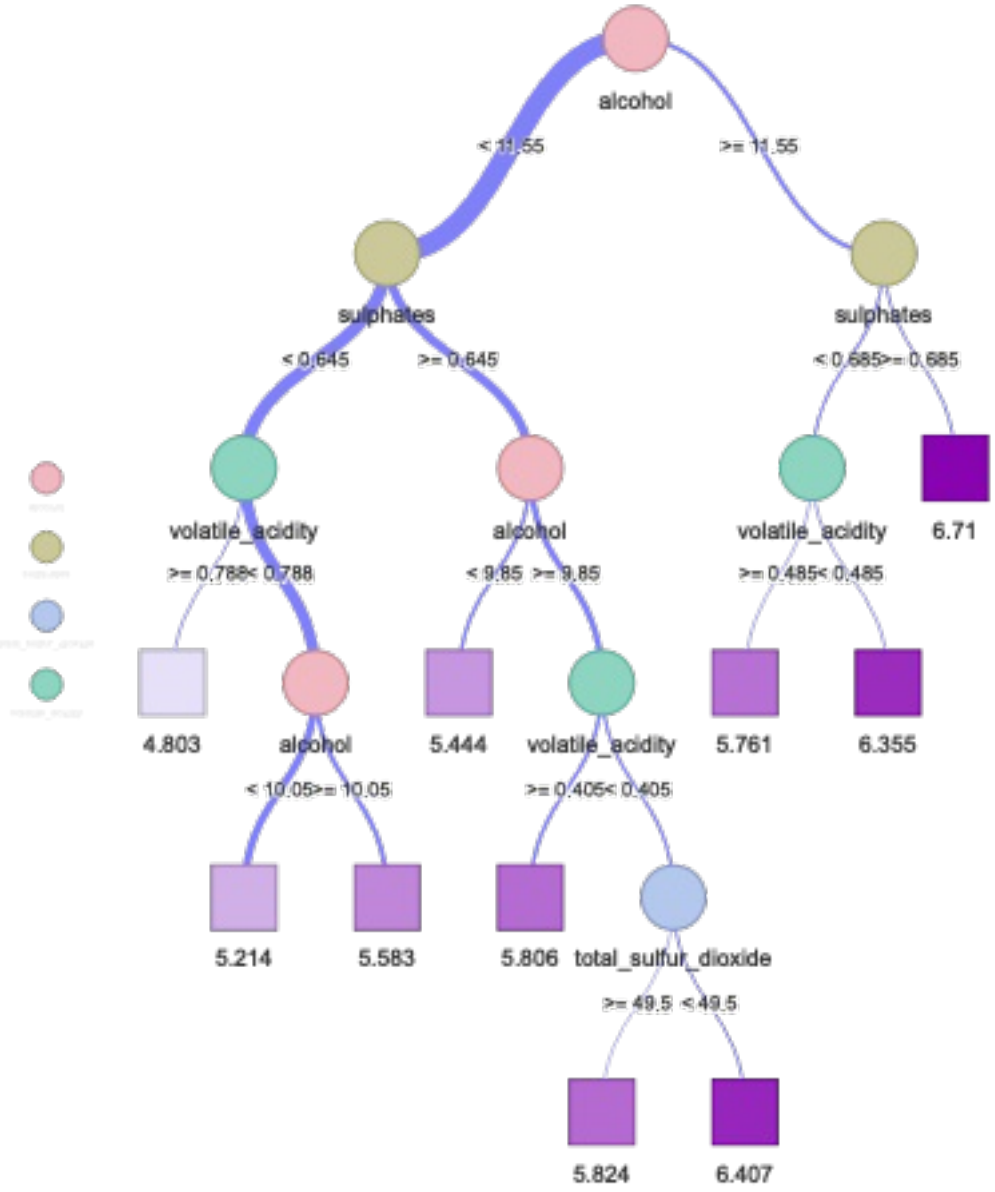
```
print(dt_fit$fit)
```

```
## 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 visualize 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.RandomForest is a natural extension of DecisionTree. A RandomForest is a collection of Deciontrees that are aggregated by majority rule, and is in essence a collection 'bootstrapped' decision trees. You need to install `randomForest` package and open the library `randomForest`. And also here, once 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

Print these results (not shown here).

```
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 of the three models we ran and which we have to compare and evaluate. We do this on the test-set. We compare the three models (`lm_fit`, `dt_fit` and `rf_fit`) on the Men Square Score (MSE) score. We need to find a model algoritm that produces predictors for the outcome (`quality`) that minimizes the MSE-score. So, the lower the mse-score of the model, the better.

1. Accuracy of the lm-model

Let us first look at the accuracy of the linear-model.

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

It gives here the following mse-score for lineair modeling, which we show here

```
head(lm_mse)
```

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

2. Accuracy of the Decision Tree Model

Then we look at the accuracy of the DecisionTree Model

The decision model gives the following mse-score:

```
head(dt_mse)
```

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

3. Accuracy of the Random Forest Model

And then ofcourse we also have to look at the accuracy of the RandomForest-model.

The Random Forest Model gives us the following mse-score:

```
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 and compare them with each other.

Let us show these results together.

```
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.

```
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>
```

CONCLUSION

In this simple scenario, we were interested in seeing how the model performs on the testing data that were 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. Root Mean Square Error (RMSE) is a standard way to measure the error of a model in predicting quantitative data. RMSE is a good measure to use if we want to estimate the standard deviation of a typical observed value from our model's prediction, R-squared is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted. In Machine Learning, MAE is a model evaluation metric often used with regression models. After the model is fitted and applied, we collected the performance metrics and display them and show the predictions from the testing data. Altogether the prediction scores don't look very well, but we know that RandomForest is the best model for prediction.

```r
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
```

$105/307 * 100 = 34,2$ is wrong, which is a the end a bit disappointing after all the work. But we know what the best model is for this dataset. This work has some **strengths** We found a uniform and consistent way to compare models with each and to choose the best one out of them. One of the big advantages of the random forest model which is choosen her is the versality and flexibility. It can be used for both regression and classification problems. But this work has also some **limitations**. Random forest is good for predictions and not that good for prediction, so this could not be used by the researcher for interpretation. He found himself restricted here. A limitation of random fores is that this algorithm is fast to train, but quit slow to create predictions once they are trained: a more accurate prediction needs more trees, which results in a slower model. And a last limitation which we have to mention here is, that we used only three models and maybe other models were better for these data.

Lesson learned is that we found a consitent workflow for analyzing data as presented here on quality of wine.The next step would be now to work on increasing the predictive power of the model and start with tuning on the hyperparameters.

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