Wine Capstone Project 2022

Anaiz

February 2022

Introduction

The goal of this exercise is to create two predictive models for the wine data. Firstly, a data analysis will be performed in order to see what type of data and prediction can be made. Secondly, predictive models will be created with the help of a train and test set. At last, the models will be evaluated and compared to each other.

In order to reach the goal, machine learning algorithms are used.

1. Data Analysis

The data set for this exercise was downloaded from Kaggle and credit goes to: P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.

This data set includes a subset of wine quality parameters obtained only in red wine. There are different parameters included, like acidity, pH density and also a score for each sample. The goal of this data analysis is to investigate any possible correlations between the different parameters and in particularly in relationship with it's score.

1.1 Packages installation

Before the data analysis started, useful packages that could help with the analysis were installed (ex. ggplot2, dplyr,lubridate,validate,caret, etc..)

1.2 Downloading of data set and first analysis

The dataset was downloaded and imported to R studio.

Once ready, a copy of wine data was created:

```
wine1 <- wine
```

This step is not really needed, but I found it useful while doing the data analysis in case something did not work well.

The first analysis was to take a look at all the data and the class of each variable:

```
##        X          fixed.acidity   volatile.acidity  citric.acid
##  Min.   :   1.0   Min.   : 4.60   Min.   :0.1200   Min.   :0.000
##  1st Qu.: 400.5   1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090
##  Median : 800.0   Median : 7.90   Median :0.5200   Median :0.260
##  Mean   : 800.0   Mean   : 8.32   Mean   :0.5278   Mean   :0.271
##  3rd Qu.:1199.5   3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420
##  Max.   :1599.0   Max.   :15.90   Max.   :1.5800   Max.   :1.000
##  residual.sugar     chlorides       free.sulfur.dioxide total.sulfur.dioxide
##  Min.   : 0.900   Min.   :0.01200   Min.   : 1.00       Min.   :  6.00
##  1st Qu.: 1.900   1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00
##  Median : 2.200   Median :0.07900   Median :14.00       Median : 38.00
##  Mean   : 2.539   Mean   :0.08747   Mean   :15.87       Mean   : 46.47
```

```
##  3rd Qu.: 2.600   3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00
##  Max.   :15.500   Max.   :0.61100   Max.   :72.00       Max.   :289.00
##     density             pH          sulphates         alcohol
##  Min.   :0.9901   Min.   :2.740   Min.   :0.3300   Min.   : 8.40
##  1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50
##  Median :0.9968   Median :3.310   Median :0.6200   Median :10.20
##  Mean   :0.9967   Mean   :3.311   Mean   :0.6581   Mean   :10.42
##  3rd Qu.:0.9978   3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10
##  Max.   :1.0037   Max.   :4.010   Max.   :2.0000   Max.   :14.90
##     quality
##  Min.   :3.000
##  1st Qu.:5.000
##  Median :6.000
##  Mean   :5.636
##  3rd Qu.:6.000
##  Max.   :8.000
```

```
## 'data.frame':    1599 obs. of  13 variables:
##  $ X                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

The data set contains 13 variables and 1599 observations. The next step was to look if there were any NA present:

```
anyNA(wine1)
```

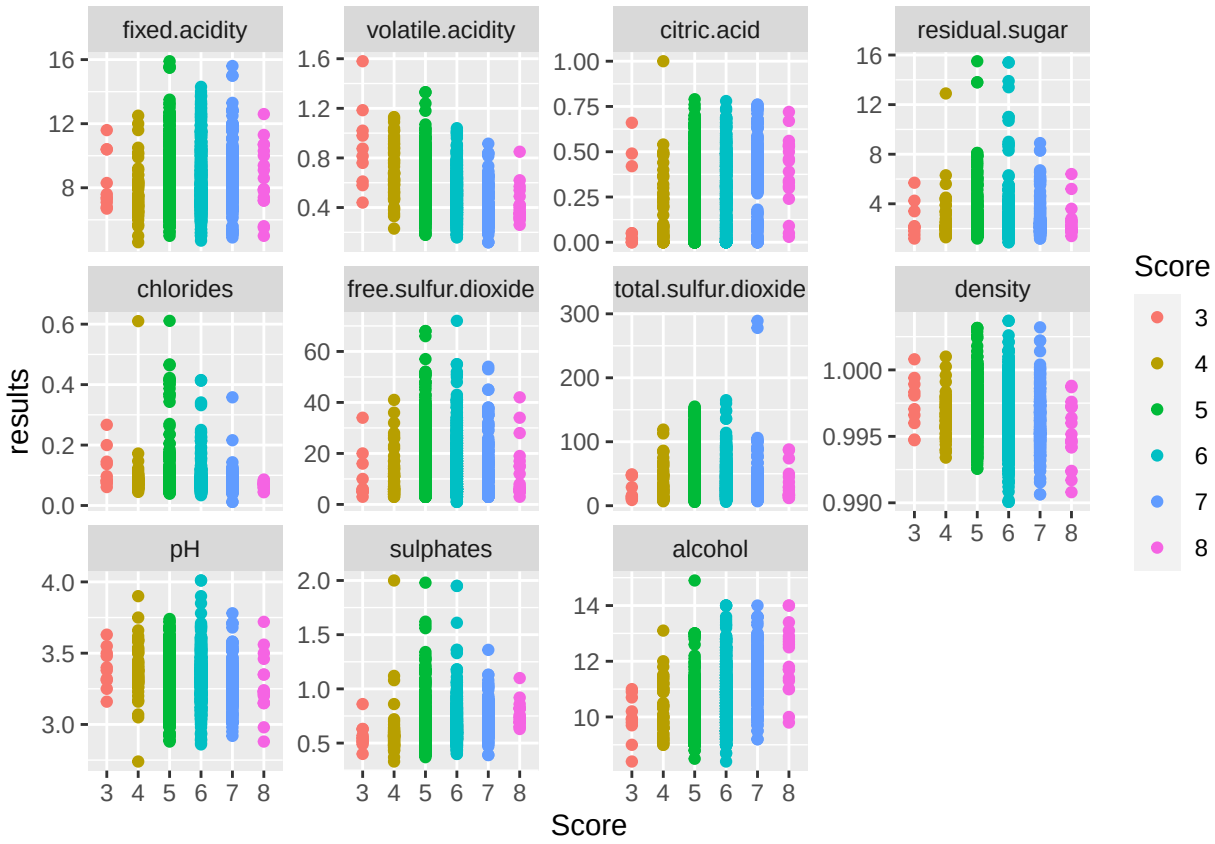
```
## [1] FALSE
```

What was observed is that the last variable is called quality. In order to make it clear that this corresponds to a grade, the name was changed to Score.

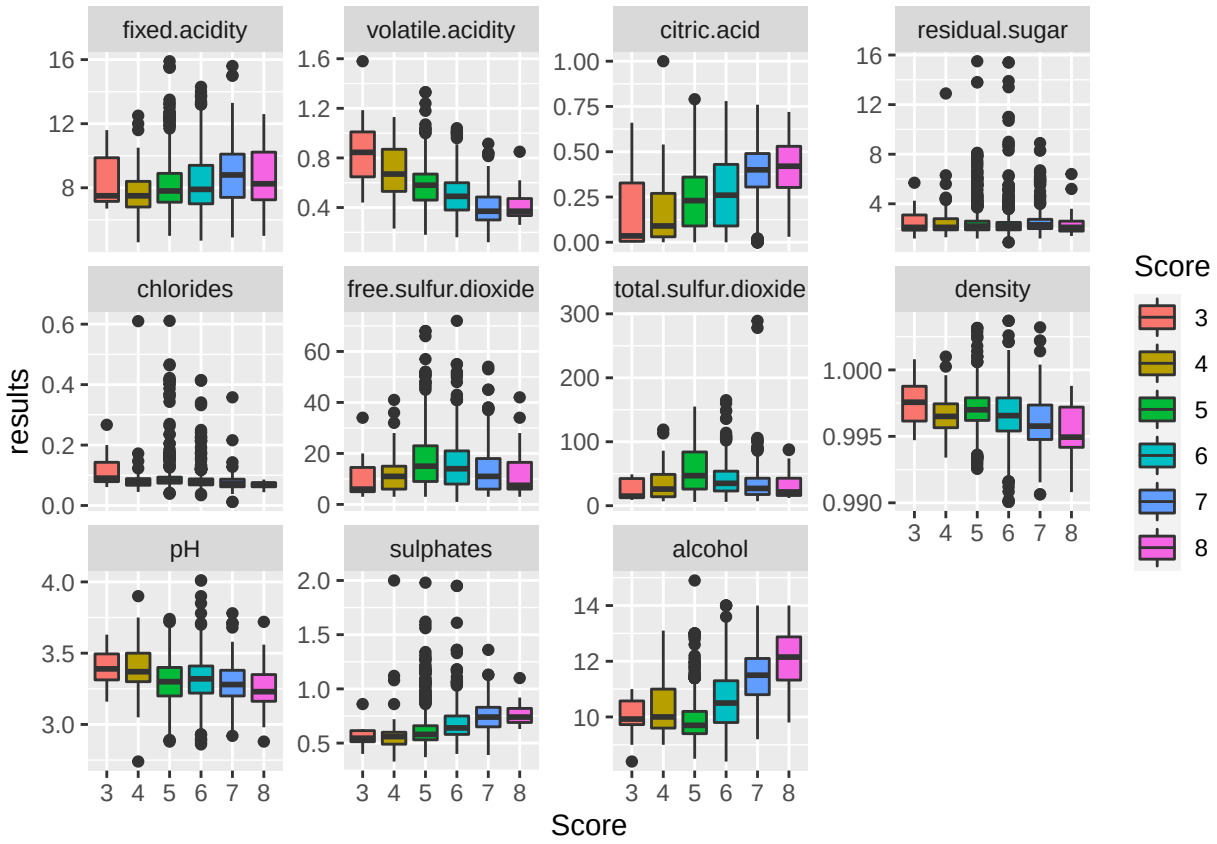
1.3 Data visualization and in depth analysis

In order to have an overview of each parameter versus the score, first the data format needed to be changed from wide to long.

Once this was performed, the first graph was achieved:



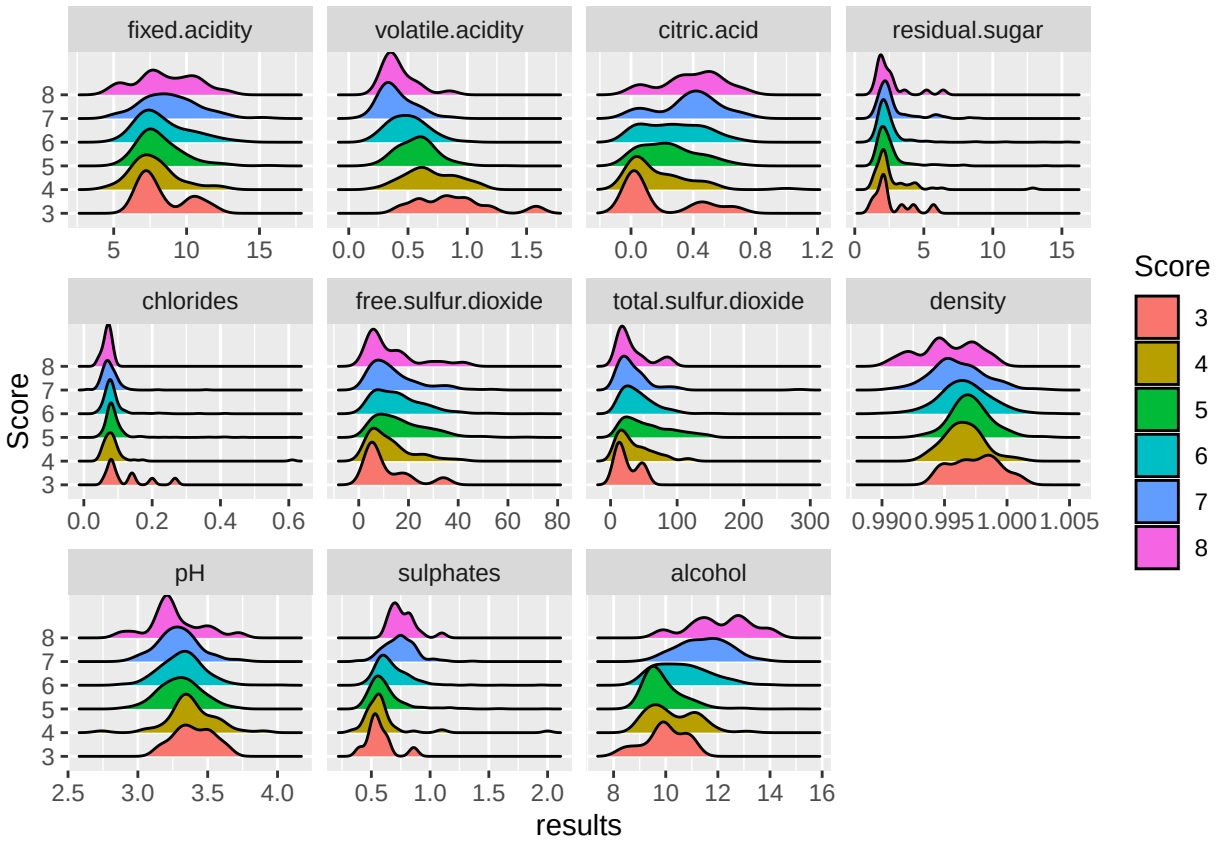
The results or this graph did not show a clear trend, therefore it was decided to create a boxplot:



In this graph is clearer that there is a trend for some of the variables. We can see that volatile acidity decrease while the score increase. The inverse result is seen with citric acid and alcohol. We need to be careful with these interpretations as for some variables the variability between the quartals is quite high and for some variables (ex. chlorides) there are quite some outliers not contributing to the bowplot. Therefore, more investigation is needed.

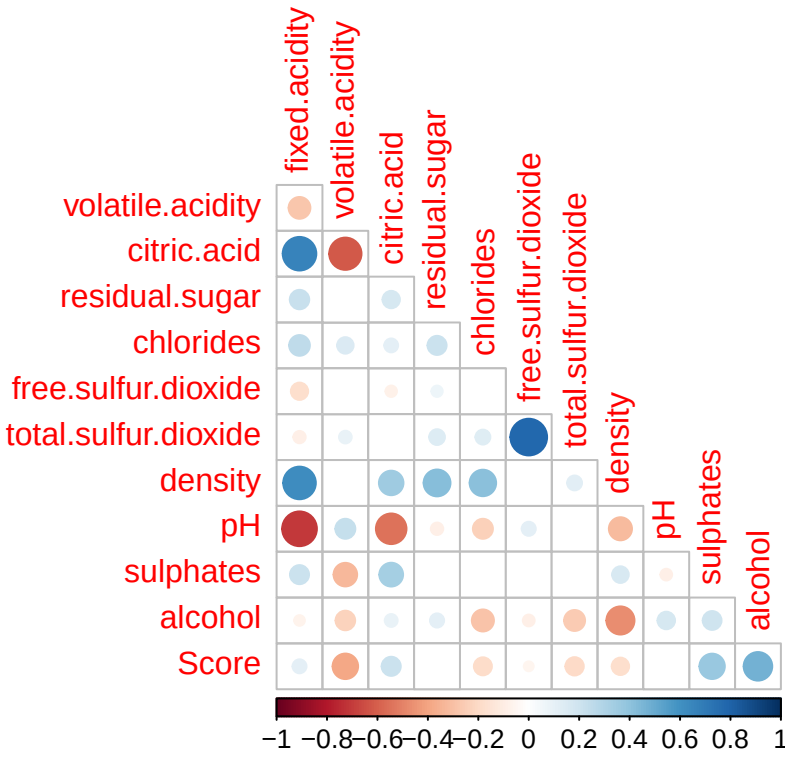
Boxplot has it's limitation when it handles large data sets. It was decided to created a ridge plot in order to see if we might have a bimodal distribution for the variables:

```
## Picking joint bandwidth of 0.659
## Picking joint bandwidth of 0.0694
## Picking joint bandwidth of 0.0718
## Picking joint bandwidth of 0.251
## Picking joint bandwidth of 0.00867
## Picking joint bandwidth of 3.08
## Picking joint bandwidth of 8.51
## Picking joint bandwidth of 0.000695
## Picking joint bandwidth of 0.0537
## Picking joint bandwidth of 0.0371
## Picking joint bandwidth of 0.338
```



We can see that for some variables a bimodal distribution is observed. This can be due to either the distribution that really is bimodal or that we are analysing a variable as one, without realizing that for example two or more different types of grapes were used to produce the wine and can have an influence on the variable itself. Unfortunately, with this data set we don't know the grapes of each wine.

To confirm if there is any correlation of variables with score, a correlation matrix was created.The significance level was set to 0.05.



We can see that fixed acidity correlates positively with citric acid and density, while it correlates negatively with pH. As we are only investigating the correlations of the variables with score, no further analysis in between variables were made.

For the correlations with score, it seems like the 3 variables having the most impact are alcohol, sulphates and volatile acidity. We see a positiv correlation between score and alcohol, score and sulphates and a negative correlation between score and volatile acidity. Meaning the higher the score, the higher the average alcohol and sulphates content and the lower the average volatile acidity.

It was decided to investigate these 3 parameters a bit further:

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

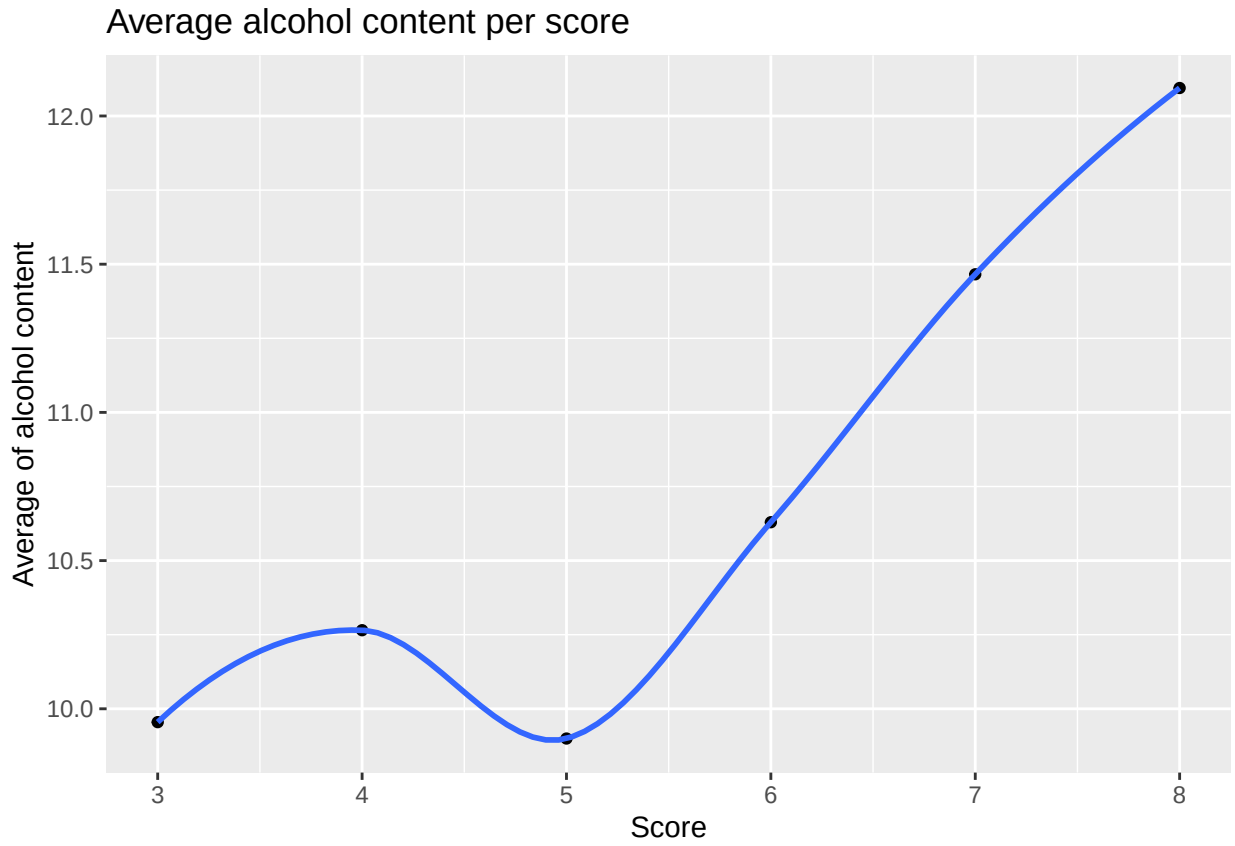
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : Chernobyl! trL>n 6
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : Chernobyl! trL>n 6
```

```
## Warning in sqrt(sum.squares/one.delta): NaNs wurden erzeugt
```

```
## Warning in stats::qt(level/2 + 0.5, pred$df): NaNs wurden erzeugt
```

```
## Warning in max(ids, na.rm = TRUE): kein nicht-fehlendes Argument für max; gebe
## -Inf zurück
```




```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

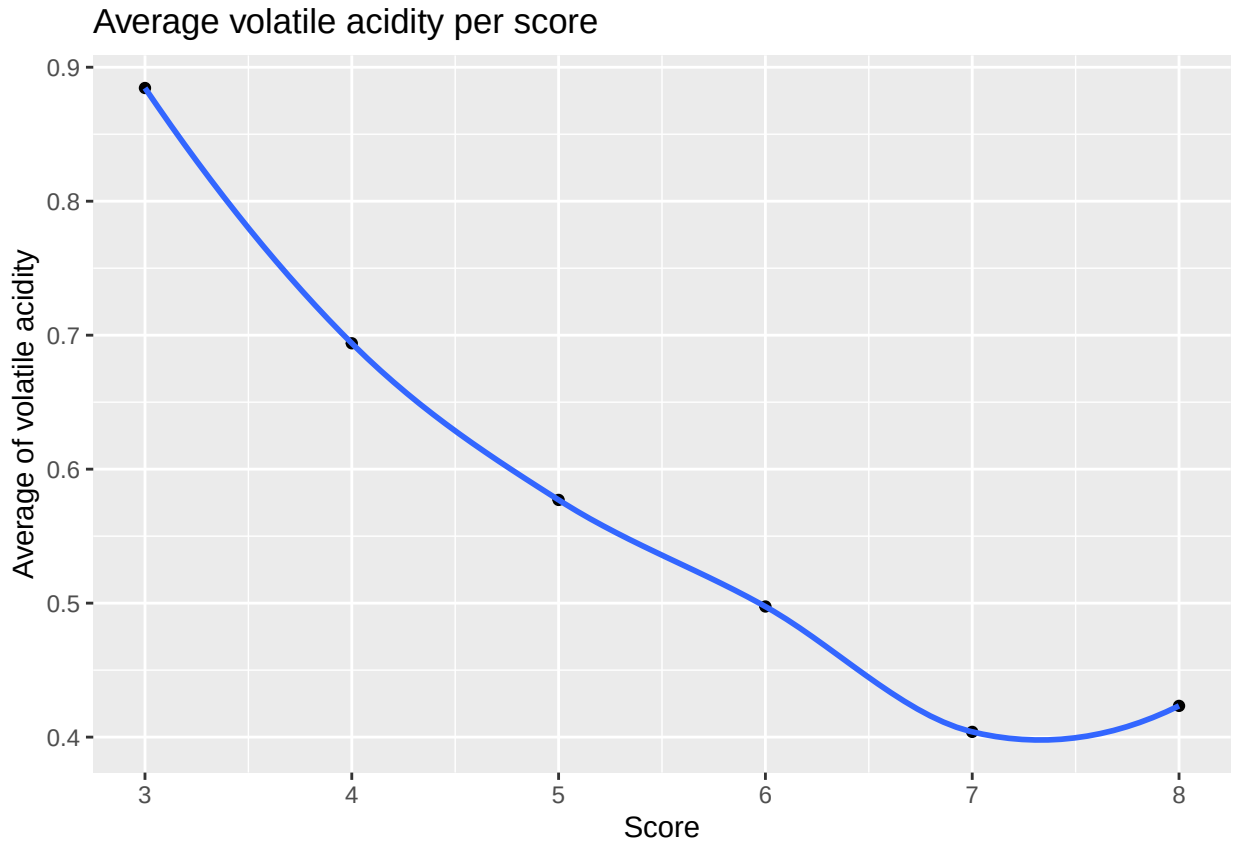
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : Chernobyl! trL>n 6
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : Chernobyl! trL>n 6
```

```
## Warning in sqrt(sum.squares/one.delta): NaNs wurden erzeugt
```

```
## Warning in stats::qt(level/2 + 0.5, pred$df): NaNs wurden erzeugt
```

```
## Warning in max(ids, na.rm = TRUE): kein nicht-fehlendes Argument für max; gebe
## -Inf zurück
```




```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

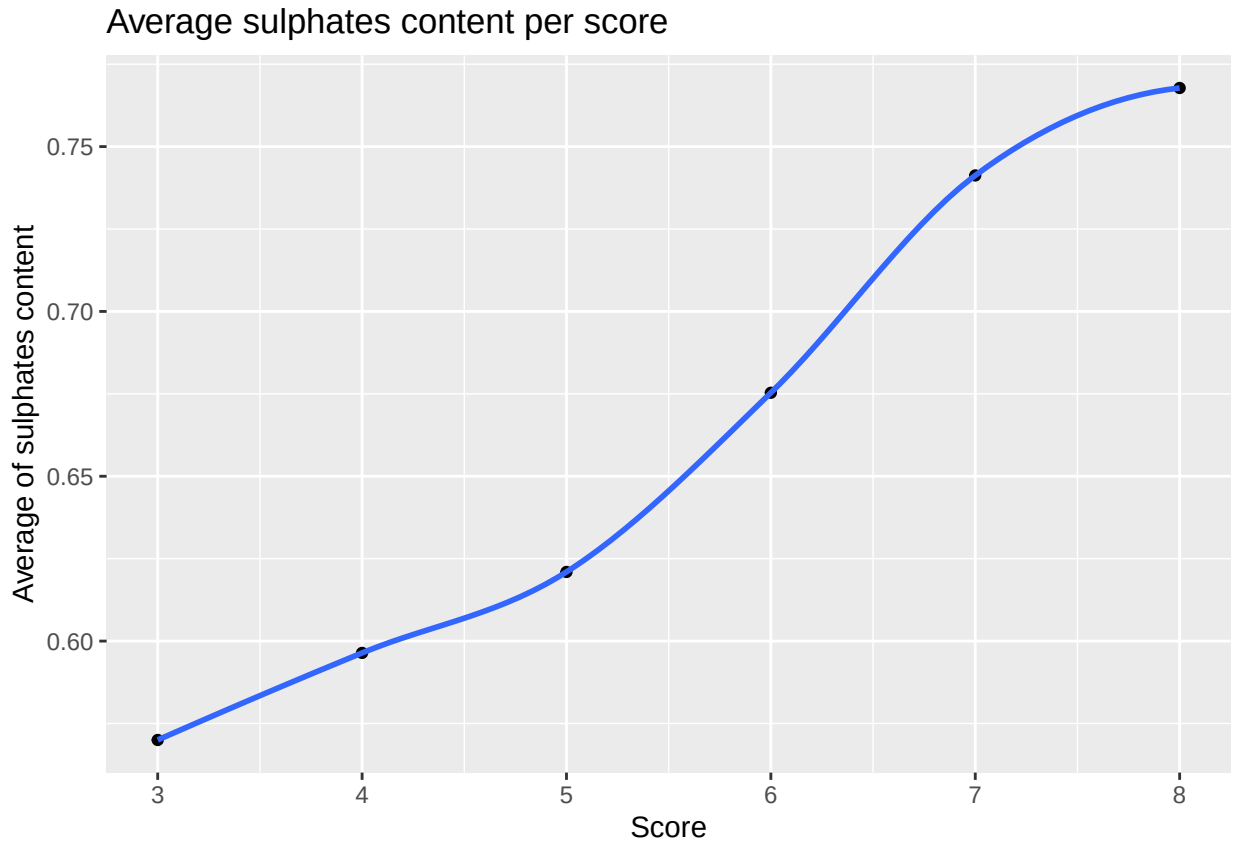
```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : Chernobyl! trL>n 6
```

```
## Warning in simpleLoess(y, x, w, span, degree = degree, parametric =
## parametric, : Chernobyl! trL>n 6
```

```
## Warning in sqrt(sum.squares/one.delta): NaNs wurden erzeugt
```

```
## Warning in stats::qt(level/2 + 0.5, pred$df): NaNs wurden erzeugt
```

```
## Warning in max(ids, na.rm = TRUE): kein nicht-fehlendes Argument für max; gebe
## -Inf zurück
```



The graphs above confirm the observations seen in the correlation matrix. These parameters will be used to create one of our models.

2. Modelling analysis and Results

In this section, two different models approach will be used. The first model approach is the recommendation system (like in the movielens exercise from EDX Capstone course). For this model the variables used are alcohol, sulphates and volatile acidity to predict the score of a wine. The second approach is the random forest model. For both models, the RMSE will be reported.

Please be aware, in the analysis "summarise" is written with "s" as with "z" the code did not work with my R and gave me an error each time I used it.

Before starting any anaylsis, a train and test data set were created with 50% partition.

```
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(wine1$Score, times = 1, p = 0.5, list = FALSE)

train <- wine1[test_index,]
test <- wine1[-test_index,]

test <- test %>%
  semi_join(train, by = "alcohol") %>%
  semi_join(train, by = "volatile.acidity") %>%
  semi_join(train, by = "sulphates")
```

2.1 Recommendation System Model

First, the Residual Mean Squared Error function was defined

```
RMSE <- function(true_prediction, predicted_prediction){
  sqrt(mean((true_prediction - predicted_prediction)^2))}
```

The first model used in this approach was to predict the score of the wine independent of the variables.

```
mu_score <- mean(train$Score)
mu_score
```

```
## [1] 5.64
```

```
first_rmse <- RMSE(test$Score, mu_score)
first_rmse
```

```
## [1] 0.7941076
```

model	RMSE
Baseline Model	0.7941076

This model was used as base and RMSE will be tried to improve.

The second model included the predicting the alcohol, volatile acidity and sulphates effect on the score.

```
alcohol_avg <- train %>%
  group_by(alcohol) %>%
  summarise(b_a = mean(Score - mu_score))

acidity_avg <- train %>%
  left_join(alcohol_avg, by='alcohol') %>%
  group_by(volatile.acidity) %>%
  summarise(b_v = mean(Score - mu_score - b_a))

sulphates_avg <- train %>%
  left_join(alcohol_avg, by='alcohol') %>%
  left_join(acidity_avg, by='volatile.acidity') %>%
  group_by(sulphates) %>%
  summarise(b_s = mean(Score - mu_score - b_a - b_v))

predicted_score <- test %>%
  left_join(alcohol_avg, by='alcohol') %>%
  left_join(acidity_avg, by='volatile.acidity') %>%
  left_join(sulphates_avg, by='sulphates') %>%
  mutate(pred = mu_score + b_a + b_v + b_s) %>%
  pull(pred)

Combined_effect <-RMSE(predicted_score, test$Score)
Combined_effect
```

```
## [1] 0.7427212
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

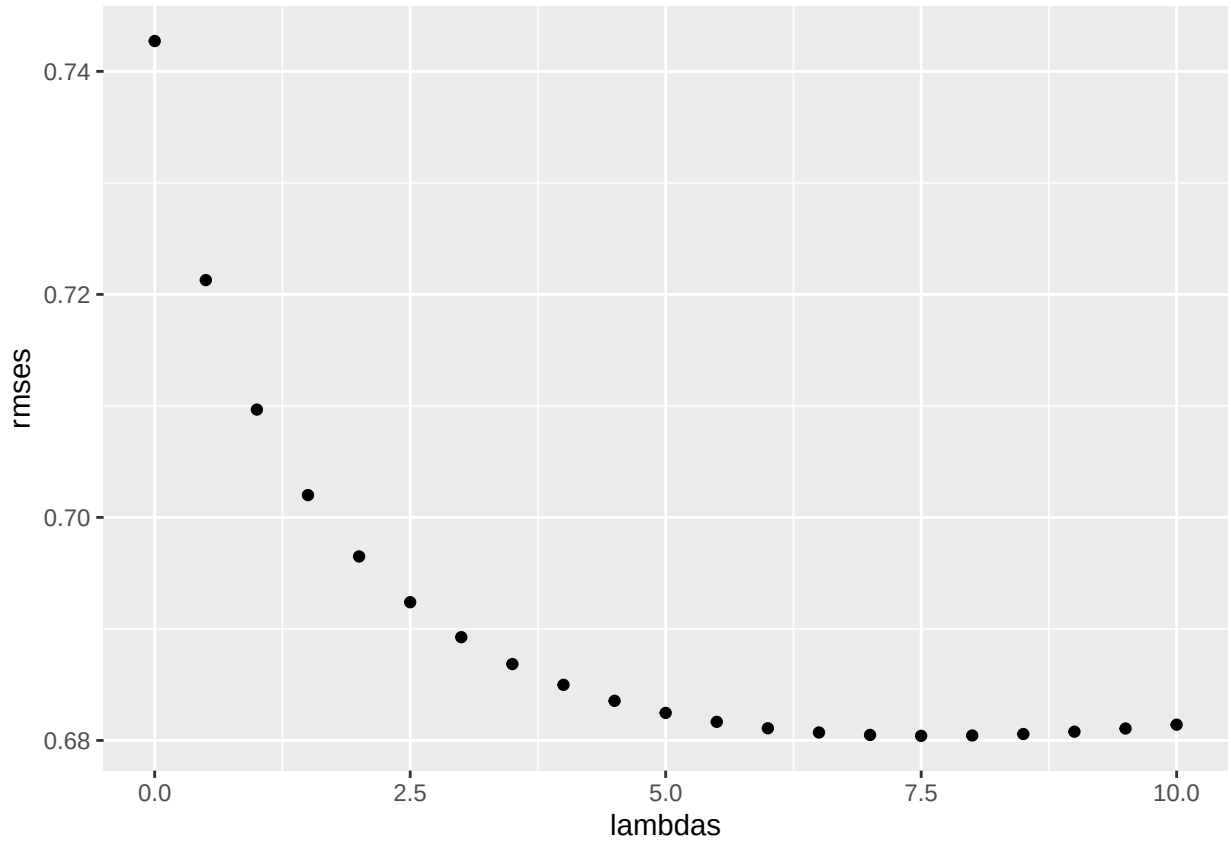
model	RMSE
Baseline Model	0.7941076

model	RMSE
Combined effect Model	0.7427212

The third model was the regularized model. For this model we chose lambda (tuning parameter) for alcohol, volatile acidity and sulphates effect in order to improve the RMSE.

```
lambdas <- seq(0, 10, 0.5)

rmses <- sapply(lambdas, function(l){
  mu_score <- mean(train$Score)

  b_a <- train %>%
    group_by(alcohol) %>%
    summarise(b_a = sum(Score - mu_score)/(n()+l))

  b_v <- train %>%
    left_join(b_a, by="alcohol") %>%
    group_by(volatile.acidity) %>%
    summarise(b_v = sum(Score - b_a - mu_score)/(n()+l))

  b_s <- train %>%
    left_join(b_a, by="alcohol") %>%
    left_join(b_v, by="volatile.acidity") %>%
    group_by(sulphates) %>%
    summarise(b_s = sum(Score - b_a - b_v - mu_score)/(n()+l))

  predicted_score <- test %>%
    left_join(b_a, by = "alcohol") %>%
    left_join(b_v, by = "volatile.acidity") %>%
    left_join(b_s, by = "sulphates") %>%
    mutate(pred = mu_score + b_a + b_v + b_s) %>%
    pull(pred)

  return(RMSE(predicted_score, test$Score))
})
```

The distribution of lambdas is the following:



By using following code, the optimal lambda was chosen:

```
lambda1 <- lambdas[which.min(rmses)]
lambda1
```

```
## [1] 7.5
```

Once the lambda was selected, it was added to the code in order to see if RMSE has improved.

```
alcohol_avg <- train %>%
  group_by(alcohol) %>%
  summarise(b_a = sum(Score - mu_score)/(n()+lambda1), n_a = n())

acidity_avg <- train %>%
  left_join(alcohol_avg, by='alcohol') %>%
  group_by(volatile.acidity) %>%
  summarise(b_v = sum(Score - mu_score - b_a)/(n()+lambda1), n_v = n())

sulphates_avg <- train %>%
  left_join(alcohol_avg, by='alcohol') %>%
  left_join(acidity_avg, by='volatile.acidity') %>%
  group_by(sulphates) %>%
  summarise(b_s = sum(Score - mu_score - b_a - b_v)/(n()+lambda1), n_s = n())

predicted_score <- test %>%
  left_join(alcohol_avg, by = "alcohol") %>%
  left_join(acidity_avg, by = "volatile.acidity") %>%
  left_join(sulphates_avg, by = "sulphates") %>%
  mutate(pred = mu_score + b_a + b_v + b_s) %>%
```

```
  pull(pred)

reg_m_u_effect <- RMSE(predicted_score, test$Score)
```

model	RMSE
Baseline Model	0.7941076
Combined effect Model	0.7427212
Reg. Model 3 parameters effect	0.6804121

The RMSE improved from 0.7941076 to 0.6804121 with the regularized model.

For this analysis/model we used only three variables. Therefore, another approach might help us to improve the RMSE even more.

2.2 Random Forest Model

The second approach used is the random forest model. This model helps to improve prediction performance and is based on creating multiple decision trees to improve prediction.

As we are working with a regression model and not a classification model, no accuracy can be predicted. The prediction will be made for RMSE (Root Mean Squared Error) and MAE (Mean Absolute Error).

The first thing we do is define the control measurements and evaluate the model with a grid search of 10 folder

```
trainControl <- trainControl(method = "cv",
                             number = 10,
                             search = "grid")
```

A first formula was created to implement the random forest (rf) model:

```
rf1 <- train(Score~., train, method = "rf", trControl = trainControl)
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
print(rf1)
```

```
## Random Forest
##
## 800 samples
##  12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 720, 721, 720, 720, 721, 719, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##    2    0.6029878  0.4363404  0.4692250
##    7    0.6043070  0.4270677  0.4655828
##   12    0.6076072  0.4199954  0.4655462
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 2.
```

This results i already better than what we obtained with the recommendation system model.

Let's try to optimize this further. To do so, we optimize our model by tuning the grid and search for a better mtry between 1 and 12.

```
set.seed(1234)
tuneGrid <- expand.grid(.mtry = c(1: 12))

rf2 <- train(Score~., train, method = "rf", trControl = trainControl, tuneGrid =tuneGrid, importance = '
```

```
## Warning in (function (kind = NULL, normal.kind = NULL, sample.kind = NULL) :
## non-uniform 'Rounding' sampler used
```

```
print(rf2)
```

```
## Random Forest
##
## 800 samples
##  12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 719, 720, 719, 720, 720, 721, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##    1    0.6143601  0.4257514  0.4817144
##    2    0.6026631  0.4358298  0.4681243
##    3    0.6017622  0.4333287  0.4666172
##    4    0.6019408  0.4303443  0.4662305
##    5    0.6015840  0.4307955  0.4645512
##    6    0.6054645  0.4212309  0.4672077
##    7    0.6035850  0.4249879  0.4646807
##    8    0.6047842  0.4223016  0.4661487
##    9    0.6095853  0.4130345  0.4687084
##   10    0.6085202  0.4150640  0.4683855
##   11    0.6074984  0.4166846  0.4666578
##   12    0.6083152  0.4151878  0.4668063
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 5.
```

The results show the best mtry for the best RMSE.

Now we can evaluate the model against our test data:

```
prediction <-predict(rf2, test)

RF_Model<- RMSE(prediction, test$Score)
```

model	RMSE
Baseline Model	0.7941076
Combined effect Model	0.7427212
Reg. Model 3 parameters effect	0.6804121
Random Forest Model	0.5898854

The RMSE obtained for the random forest model is 0.5898854.

#3. Conclusions

In general, we could see that using different models, the RMSE was improved:

model	RMSE
Baseline Model	0.7941076
Combined effect Model	0.7427212
Reg. Model 3 parameters effect	0.6804121
Random Forest Model	0.5898854

The random forest model showed significantly improvement in the RMSE.

There is still room for more improvement in both models. For the first model (recommendation system), more variables could be added in order to make the model more precise. For the random forest model, there is the possibility of not only improving the mtry, but also the maxnodes and the ntrees (which were not optimize for this exercise). This could be made in a future project.

What would be also interesting, is to have more information of the data, ex. what kind of grapes are used to produce the wine or if they come from a specific region in order to predict more specifically on the origin of the wine and not only the quality.

4. References

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553. ISSN: 0167-9236.

https://rafalab.github.io/dsbook/

https://www.guru99.com/r-random-forest-tutorial.html