Project 4

Adam Goran & Willie Langenberg

2022-01-07

Introduction

In this project we will analyse the wine quality data set provided by Cortez et al. (2009). It includes samples of red and white wines from the north of Portugal and is one of the most popular data sets from the UCI Machine learning repository. Even though the samples were gathered to try and predict the wine quality, other aspects could be interesting to investigate as well, for example if it is possible to predict whether the wine is red or white using the same variables. We will treat the data set using methods of classification, since the natural response variable wine quality takes categorical values on the ordinal scale. Also, the colour of the wine is a nominal variable taking two different values. The aim of this report is separately to explain what variables has the most effect on the wine quality and to predict the wine colour.

Data preparation and exploratory analysis

The data set consists of two files, one file for red wines and one for white. We merge the two files and create a new binary variable indicating what wine colour the sample has. There are 6497 samples in total of which 1599 are on red wines and 4898 on white ones. The intended response variable wine quality is measured on a scale from 0 to 10. This is problematic since the cell counts looks as follows.

	0	1	2	3	4	5	6	7	8	9	10
counts	0	0	0	30	216	2138	2836	1079	193	5	0

The data set is heavily unbalanced since most of the samples are of medium quality. Also, there are no observations at all of wines having a very poor quality of 0, 1 or 2 and the best quality 10. We have to combine some levels to make the analysis plausible whilst keeping inference meaningful. We choose to combine the levels 0 to 4 into “low” quality and levels 7 to 10 into “high”. We treat level 5 as medium-low and level 6 as medium-high. This leaves us with 4 categories. Although we still have relatively few samples (246) of low quality wines, we believe that the explanatory variables should be able to characterize these. All 11 explanatory variables (excluding the variable for wine colour) are continuous and they consist of physicochemical measurements, such as alcohol level and pH value.

In figure 1 we can see boxplots for the explanatory variables. It might be the case that the median value and quartiles for some variables differ significantly for red and white wines so that the relevance of these values in the plot become questionable. Still, we can spot some extreme values. The variable measuring the amount of free sulfur dioxide does for example contain an apparent outlier far from the others and many variables have observations above the maximum lines. Meanwhile the density variable only has two outliers and the samples on alcohol level quite concentrated too. It is hard to assess whether the data set is really noisy or not from this plot only. By examination of the data frame of explanatory variables we know that some levels are especially common for the variables and this data frame does indeed contain over 1000 duplicated rows,

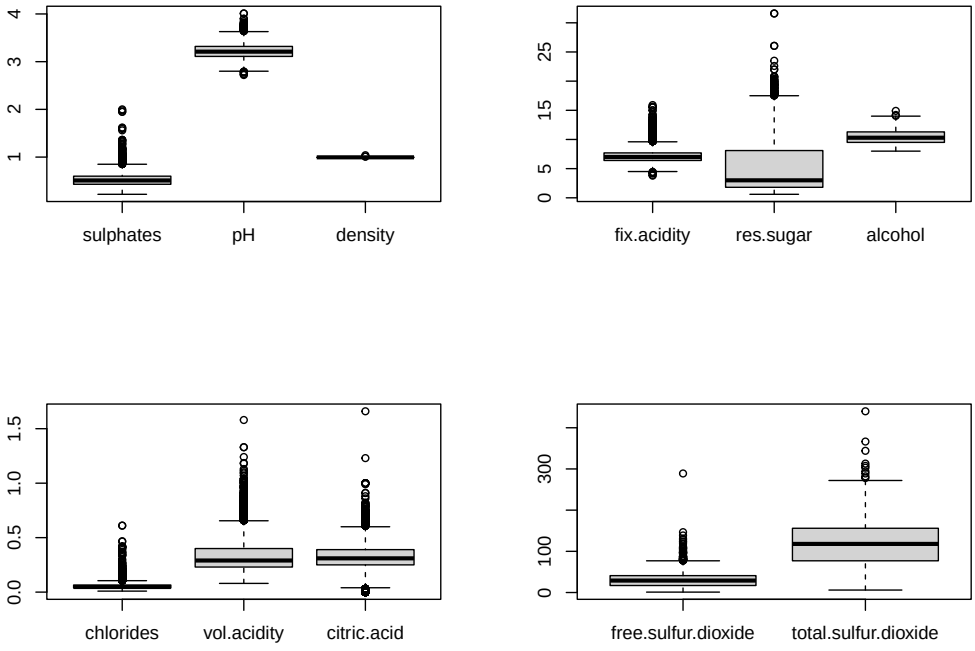

Figure 1: Boxplots of explanatory variables

i.e. non distinct observations. This does not dispute the choice of treating the variables as continuous since all of them take many distinct values. Fortunately, all samples having common levels of inputs also has the same output, so the data set is not noisy in this regard.

Recall that the first aim of the report is to determine what variables affects the wine quality the most. To investigate this we want to use random forests, since it is possible to obtain variable importance measures from the method. Random forests requires little tuning and performs well in general, unless the fraction of relevant variables when there are many variables is small (Hastie et al., 2009, p. 596). We do not have so many variables here so this should not be a problem. The second aim is to predict wine type. To deal with this task we compare the performance of random forests to linear discriminant analysis. As we for this task have 11 continuous explanatory variables, it is natural to consider LDA. By definition, LDA assumes normality of the explanatory variables. We quickly check whether this is a reasonable assumption by displaying histograms of the explanatory variables in figure 2. There, we can see that the assumption seems reasonable for some variables, e.g. the pH variable, whilst several other variables seem to follow right skewed distributions. It is seldom the case that all explanatory variables follow the normal distribution, so altogether LDA should be applicable.

In terms of software, the `R` package `caret` is used to perform random forests, which in turn uses the `randomForest` package as back-end. We tune `m`, the random number of variables to consider in each split, whilst keeping the other settings at default. This means that 500 trees are fully grown. For LDA we use the `lda` function in the `MASS` package. Default settings are used here as well and there are no hyperparameters to tune. When wine quality is the response variable the whole data set is used for training. When predicting wine type, we divide the data set and use 80% for training and 20% for testing.

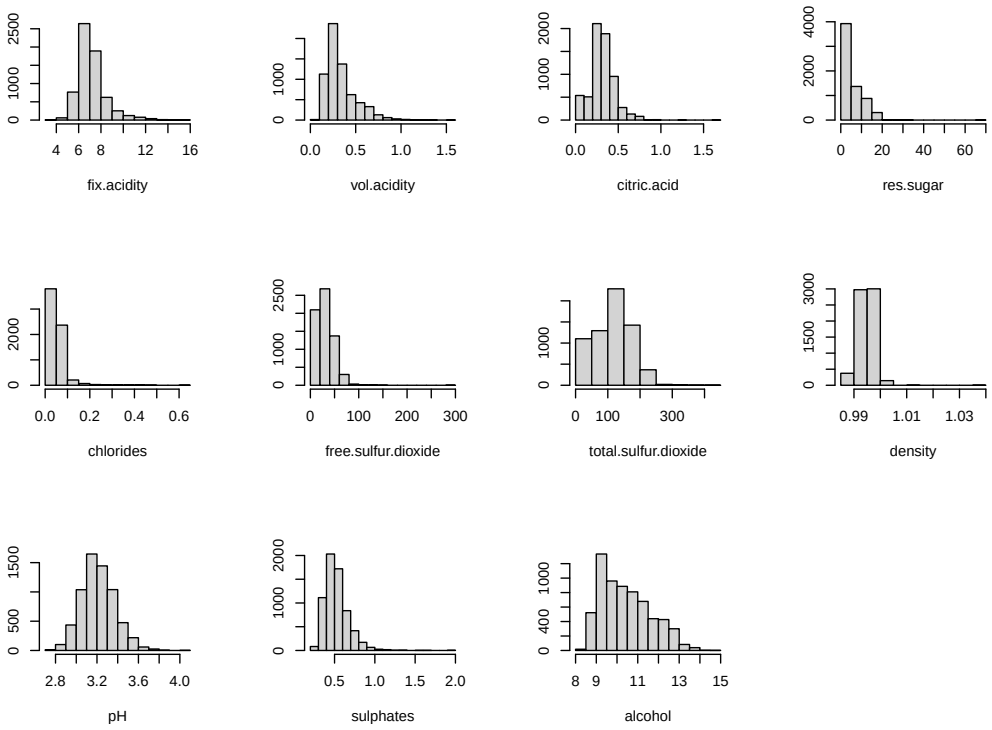

Figure 2: Histograms of explanatory variables

Main analysis

Wine quality prediction and importance of variables

For the random forest model we have one hyperparamater to choose, the m amount of variables to randomly sample at each split. To pick this we will use a five-fold cross validation on the data, picking the m that gives the lowest estimate of the misclassification rate. The suggested value of m is $\sqrt{12}$ since we have 12 predictors. So, we will go ahead and test the values $1, 2, 3, 4, 5, 6$ and 7. In the table below we can see that the model with $m = 2$ gives the highest average accuracy at $70,6\%$ on the five folds and thus we will use that model. The accuracy might seem low, but considering we have four classes it should be sufficiently good for the results to be reliable.

Table 2: Cross-validation Accuracy

m	Accuracy
1	0.6766203
2	0.7058698
3	0.7000190
4	0.7007881
5	0.6983260
6	0.6990941
7	0.6975567

The minimising value of m is quite low, indicating that there are few irrelevant variables in the model. The accuracy estimates are however close to each other, so there might be many variables having a similar degree of importance.

To further check the adequacy of the model consider the confusion matrix below, which is based on out-of-bag data. We can see that the model is worse at classifying the first class, corresponding to wine with quality 3 and 4. This makes sense since those classes contains the lowest amount of data points. We would probably be able to improve the model accuracy if we were to get more data of bad wines. It does a decent job of predicting good wines on the other hand, since most of the incorrect classifications were made on the adjacent class medium-high quality.

Table 3: Confusion Matrix

1	2	3	4	class.error
36	127	78	5	0.8536585
5	1589	523	21	0.2567820
2	370	2264	200	0.2016925
0	22	424	831	0.3492561

Using this model we can now find out the variables' importance. However, our conclusions should be drawn with care. This is because higher test accuracy means that the model fits better and we can be more certain of results, such as variable importance. Although it is difficult to evaluate the model fit when the Bayes error rate is unknown. The variable importance plot is given in figure 3. Alcohol seem to be the most important feature, density next, while we can see that the wine type (wine being red or white) have the lowest importance. Other than this we can see that the remaining variables have relatively equal importance. The results are reasonable since we do not expect that red wines are of higher quality than white wines for example.

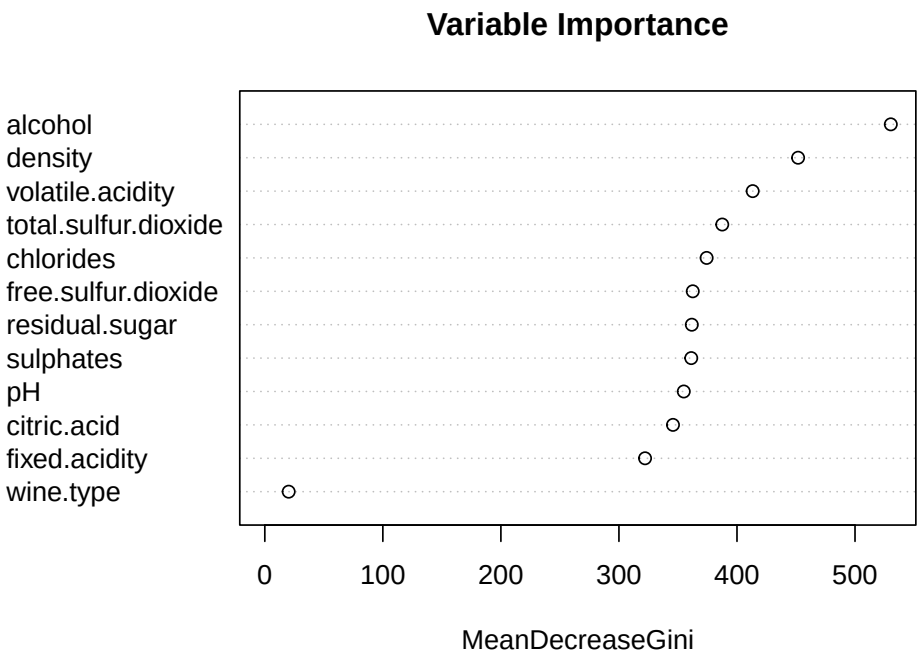

Figure 3: Variable importance measures for random forests model

Wine type prediction

When using random forests to predict whether the wine is red or white we go trough almost the same process again, by choosing the m by a five-fold cross-validation. However, since we first splitted the data into an 80/20 training-/test dataset, cross-validation was only performed on the training set and once again it was $m = 2$ that gave the highest accuracy. Furthermore, since we are using a random forest model we can easily access the variable importance plot again as a remark. For this new task we got different results. Now, alcohol was the least important variable, while chlorides, sulfur dioxide and acidity were most important. Using this model we achieved an test accuracy of 99.615% on the test set. LDA achieved an accuracy of 99.769%, which is two less missclassifications than the random forests model on the test set. Hence, both models are able to classify wine type almost perfectly.

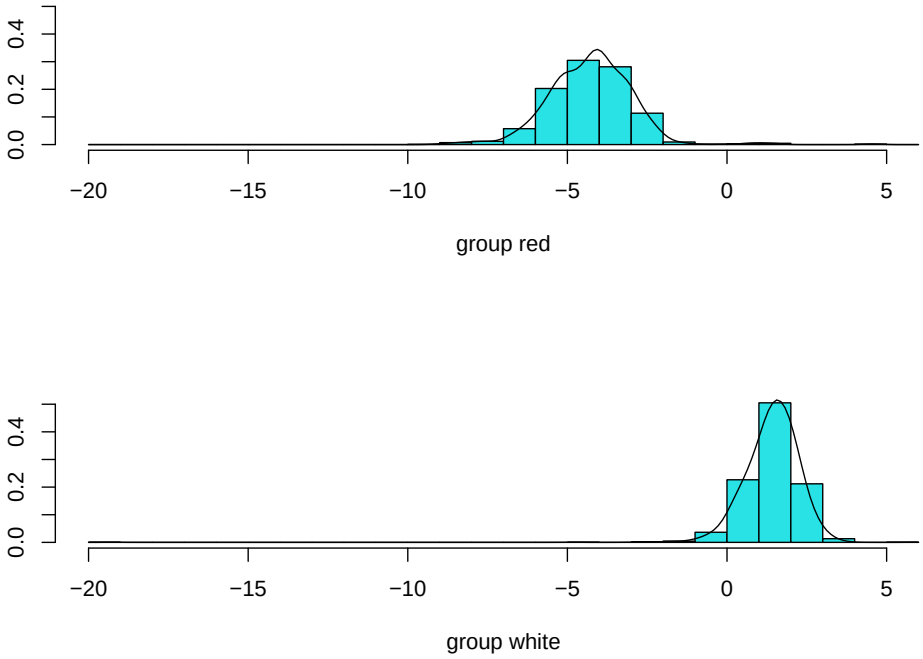

Figure 4: posterior densities for red and white wines

Let us study more in detail the high accuracy of LDA. Figure 4 shows histograms and density plots on the training data that the methods produces. These are calculated from the formula $x^T\hat{\Sigma}^{-1}(\hat{\mu}_1 - \hat{\mu}_2)$, where $\hat{\mu}_1, \hat{\mu}_2$ are the estimated means for the first and second group respectively, and $\hat{\Sigma}$ is the estimated common covariance matrix. A sample contributes to the upper plot if the corresponding input x belongs to the first class, which is red wines in this case. We can see from the plot that the two groups are almost separable for the training data, although some overlapping occur from -1 to -2 and at certain positive values. This explains the high accuracy. Note however that the plot says nothing about the decision boundary. The output from the model, which is not showed here, provides sufficient information to calculate this. The output also gives indications of the importance of the variables. Similarly as for random forests the most important variables seem to be fixed and free sulfur dioxide and volatile acidity. This can be realized by comparing group means and coefficients of linear discriminants.

In conclusion, we were able to answer the questions that we set for this report. Regarding determination of variables having impact on the wine quality, inference focused methods such as logistic regression would probably be superior to random forests. By contrast, one has to be careful with logistic regression to check

that the model assumptions are fulfilled in order for the inference to be valid. Regarding LDA for predicting wine type, we could have taken the logarithm of variables that did not seem to follow the normal distribution. As we saw in figure 2, this is especially motivated for the variables residual sugar and chlorides since these cannot take negative values, but takes many values close to zero. We did not mind to do this as the accuracy was so high anyway. Finally, we saw that LDA can outperform random forests, even though the margin probably was not statistically significant.

References

Cortez, P., Cerdeira, A., Almeida, F., Matos T. & Reis, J. (2009). *Modeling wine preferences by data mining from physicochemical properties.* In Decision Support Systems, Elsevier, 47(4):547-553.

Hastie, T., Tibshirani, R. & Friedman, J. H. (2009). *The elements of statistical learning: data mining, inference, and prediction.* 2nd ed. New York: Springer.

Appendix

Source code

```
library(tidyverse)
library(randomForest)
library(MASS)
library(knitr)
library(caret)

# reading and combining data and creating new variable for wine type
red_wine_df <- read.csv2("winequality-red.csv")
red_wine_df['wine_type'] = 1
white_wine_df <- read.csv2("winequality-white.csv")
white_wine_df['wine_type'] = 0

# converting to numeric vales and recoding the output variable
combined_df <- rbind(red_wine_df, white_wine_df)
combined_df_num <- as.data.frame(apply(combined_df, 2, FUN = as.numeric))
combined_df_num$quality <- ifelse(combined_df_num$quality < 5, 1,
                                  ifelse(combined_df_num$quality < 6, 2,
                                  ifelse(combined_df_num$quality < 7, 3, 4)))
combined_df_num$quality <- as.factor(combined_df_num$quality)
combined_df_num$wine_type <- as.factor(combined_df_num$wine_type)

## Shuffling the data
set.seed(9907)
shuffled_df <- combined_df_num[sample(nrow(combined_df_num)), ]

######## Applying Random Forest to predict wine quality

## Using Caret To Find m Using 5-fold CV
control <- trainControl(method = "cv", number = 5, search = 'grid')
tunegrid <- expand.grid(.mtry = (1:7))
rf_gridsearch <- train(quality ~.,
```

```
                        data = shuffled_df,
                        method = 'rf',
                        metric = 'Accuracy',
                        tuneGrid = tunegrid,
                        trControl = control)

# To get the two tables
kable(rf_gridsearch$finalModel$confusion, caption = "Confusion Matrix")
kable(rf_gridsearch$results[,c("mtry", "Accuracy")],
      col.names=c("m", "Accuracy"), caption = "Cross-validation Accuracy")

# To get variable importance plot
varImpPlot(rf_gridsearch$finalModel, main = "Variable Importance")


######## Applying Random Forest to predict wine type

##Splitting the data
df_winetype <- subset(shuffled_df, select = -quality)
smp_size <- floor(0.80 * nrow(df_winetype))
set.seed(9907)
train_ind <- sample(seq_len(nrow(df_winetype)), size = smp_size)
training_df <- df_winetype[train_ind, ]
testing_df <- df_winetype[-train_ind, ]

## Using Caret To Find m Using 5-fold CV on train data
control2 <- trainControl(method = 'cv', number = 5, search = 'grid')
tunegrid2 <- expand.grid(.mtry = (1:7))
rf_winetype <- train(wine.type ~.,
                        data = training_df,
                        method = 'rf',
                        metric = 'Accuracy',
                        tuneGrid = tunegrid2,
                        trControl = control2)

# accuracy on test data
mean(predict(rf_winetype$finalModel, testing_df) == testing_df$wine_type)


####### Applying LDA to predict wine type

ld <- lda(wine_type ~., data = training_df)

# accuracy on test data
preds <- predict(ld, testing_df)$class
mean(preds == testing_df$wine_type)

# To obtain the density plot
p <- predict(ld, training_df)
ldahist(p$x[,1], training_df$wine_type, type = "both")
```