Variable importance using random forests

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 us to rely on the results.

Table 1: Cross-validation Accuracy

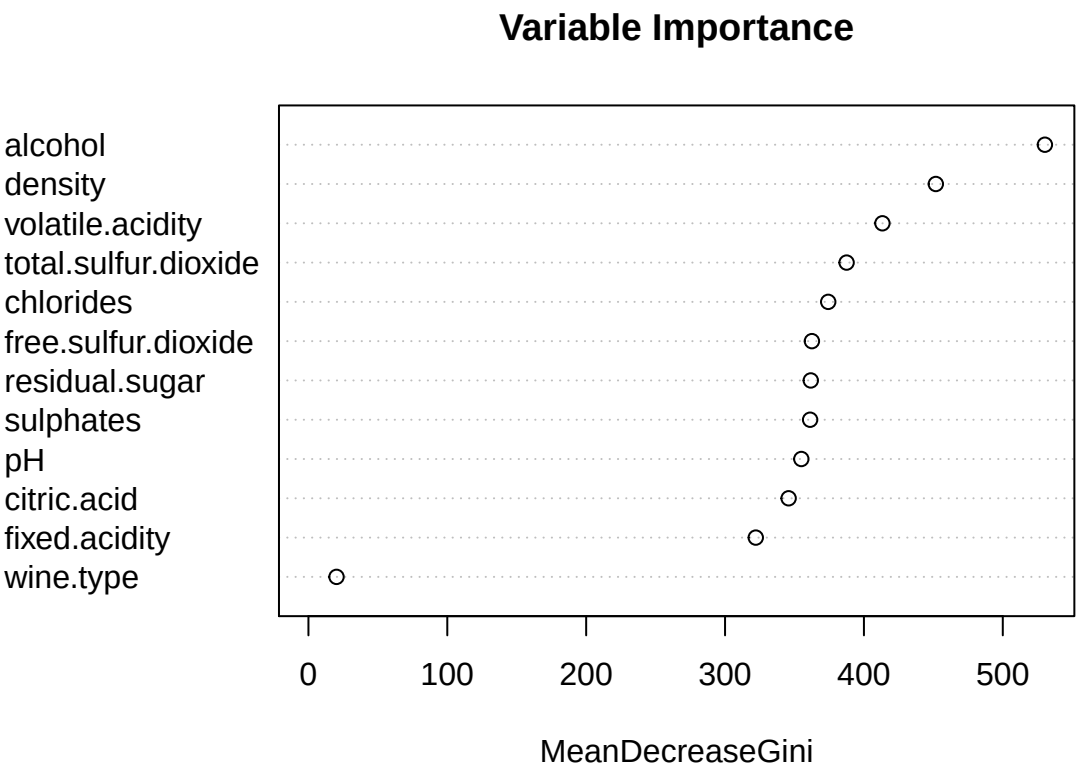
m	Accuracy
1	0.6766203
2	0.7058698
3	0.7000190
4	0.7007881
5	0.6983260
6	0.6990941
7	0.6975567

To further check the adequacy of the model consider the confusion matrix below. 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. We will skip this in this paper.

Table 2: 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. Given the variable importance plot below we can draw some conclusions. 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.



Using Random Forest to predict wine type

When using random forests to predict wether 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, we first splitted the data into an 80/20 training-/test dataset. By performing the cross-validation on the training set it once again was $m = 2$ that gave the highest accuracy. Using this model we achieved an test accuracy of 99.615% on the test set. Furthermore, since we are using a random forest model we can easily access the variable importance plot again. 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 Random Forest to predict wine quality

## Combining data from red and white wine into one dataframe
set.seed(9907)
red_wine_df <- read.csv2("../Data/winequality-red.csv")
red_wine_df['wine.type'] = 1
white_wine_df <- read.csv2("../Data/winequality-white.csv")
white_wine_df['wine.type'] = 0
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$quality < 6, 2
combined_df_num$quality <- as.factor(combined_df_num$quality)

## Shuffling the data
set.seed(9907)
```

```r
shuffled_df <- combined_df_num[sample(nrow(combined_df_num)), ]

## 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)
save(rf_gridsearch, file="rf_model_gridsearch.RData")


######## Using Random Forest to predict wine type

##Splitting the data
df_winetype <- subset(shuffled_df, select= -quality)
df_winetype$wine.type <- as.factor(df_winetype$wine.type)
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, ]

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)
save(rf_winetype, file="rf_winetype.RData")

sum(predict(rf_winetype$finalModel, testing_df)==testing_df$wine.type)/nrow(testing_df)
varImpPlot(rf_winetype$finalModel)
```