Prediction of Wine Quality from Chemical Markers

matt_173

2022-07-19

Introduction

This analysis develops two wine quality prediction models (white and red) based on two data sets of Portuguese wines. The wine quality is classified on an integer scale of 1-10 with the median rating of expert wine tasters. Each wine is also analyzed chemically, with data provided for the following numeric characteristics:

Features
fixed_acidity
volatile_acidity
citric_acid
residual_sugar
chlorides
free_sulfur_dioxide
total_sulfur_dioxide
density
pH
sulphates
alcohol

The analysis will begin with data cleansing and some general exploration of the data sets. The end goal is to arrive at a model that can predict a wine quality rating from the chemical markers above. A primary consumer of wine ratings are non-experts who rely on the ratings to avoid bringing a terrible bottle of wine to a dinner party. Therefore our metric for fit will go beyond overall accuracy and consider class-specific measures. For example, an explicit goal of the rating prediction model will be to avoid a particularly high rating for a wine that is actually a very low rating.

Analysis

The approach will follow a fairly standard sequence, starting with data cleansing, exploratory data analysis. The next phase will evaluate a range of machine learning models to find one or more candidate methods for further tuning. The final model will then be presented and discussed in a results section and the study will conclude with some potential next steps for improving the model.

Data Cleansing

The data is provided as complete as we confirm below. No further cleansing or wrangling is required.

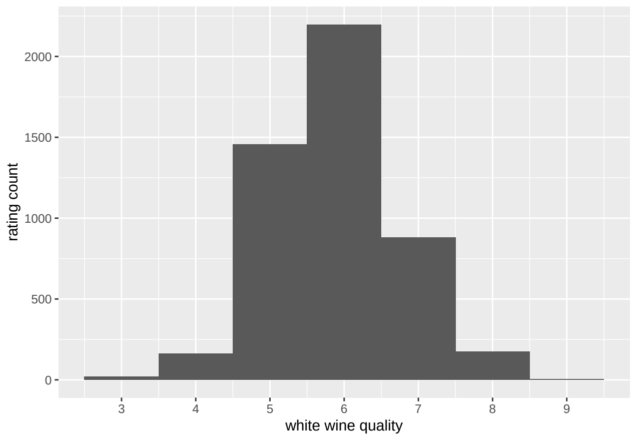
```r
#calculate summary statistics
white_stats <- stat.desc(white_wines)
red_stats <- stat.desc(red_wines)
```

```r
#check for na and null values.
str_c("white wine data missing or null: ",(sum(white_stats$nbr.na) +
                                             sum(white_stats$nbr.null)))
str_c("red wine data missing or null: ",(sum(red_stats$nbr.na) +
                                           sum(red_stats$nbr.null)))
```

```
## [1] "white wine data missing or null: 0"
## [1] "red wine data missing or null: 0"
```

Data Exploration and Visualization

Wine Ratings First we take a look at the distribution of ratings for each type of wine. The data is approximately normal, but we can see from the q-q plot that extreme ratings are over-represented in the data. For ordinal categorical data, the normal distribution would spread points symmetrically about the identity line. This creates a prevalence issue as most ML metrics will optimize for the heavily populated center and neglect the less-populated tails.



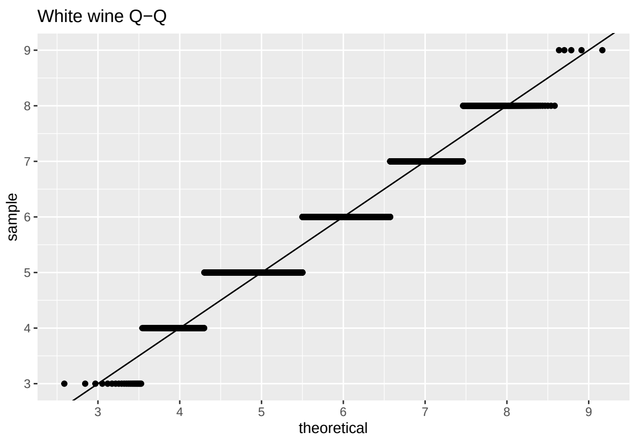
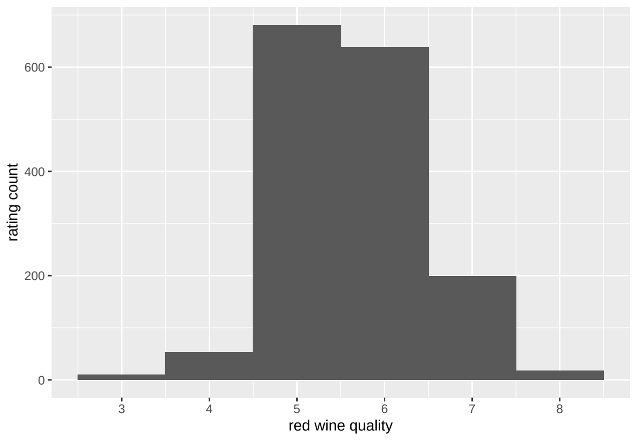


Table 1: White Wine Ratings Frequency

quality	n__w
3	20
4	163
5	1457
6	2198
7	880
8	175
9	5



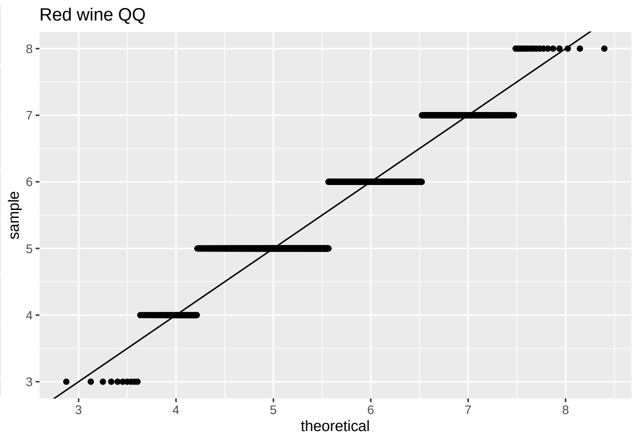
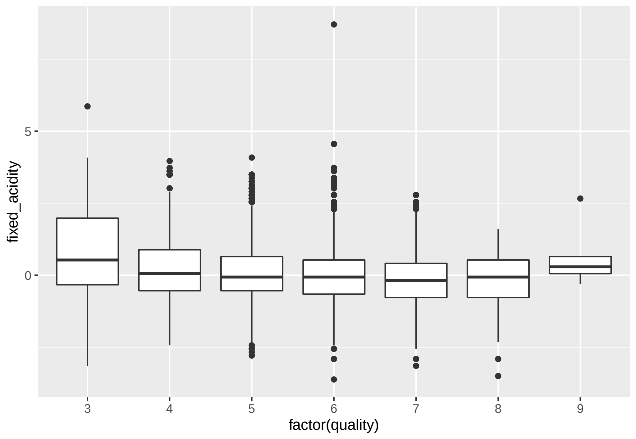


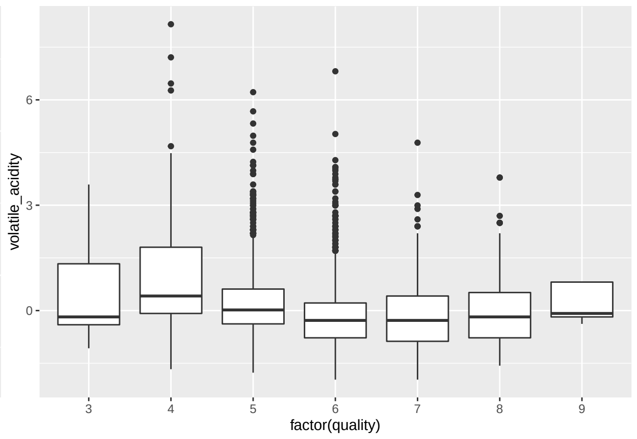
Table 2: Red Wine Ratings Frequency

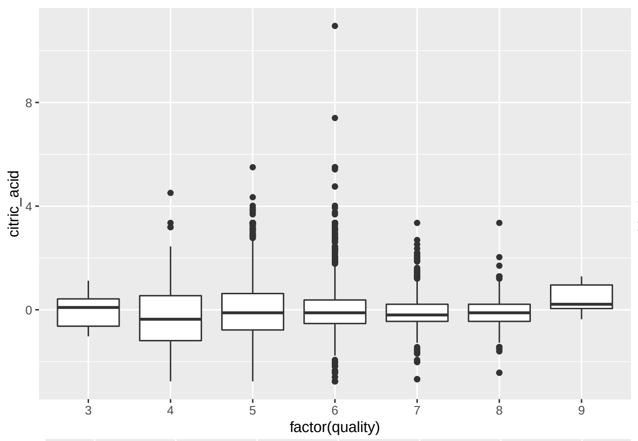
quality	n_r
3	10
4	53
5	681
6	638
7	199
8	18

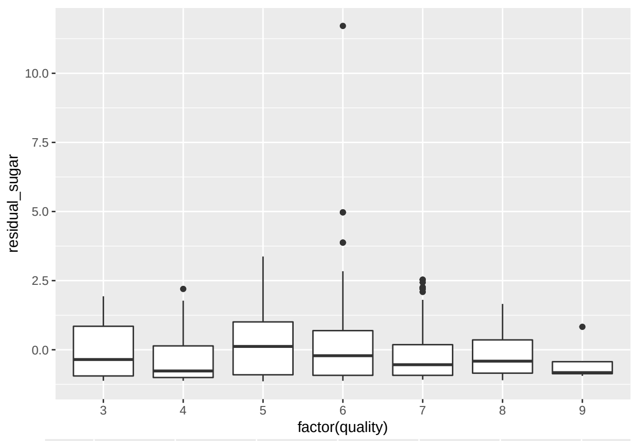
White Wine Features Here we explore the relationship of wine quality with each of the chemical markers provided. This can give a sense of any direct relationships we expect or potential non-linear relationships. We standardize the independent variables as some ML techniques are sensitive to variables of different scales.

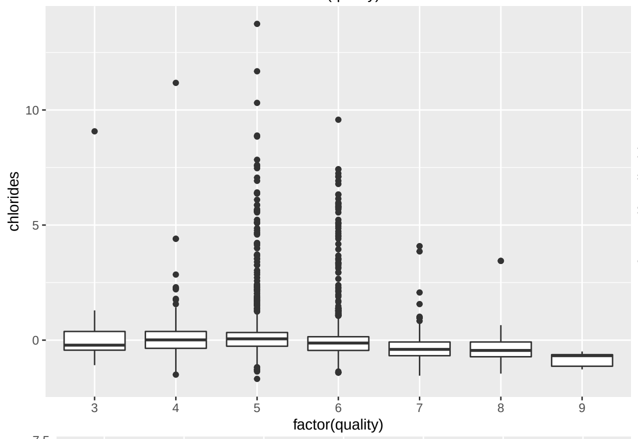
- quality generally improves with decreasing fixed_acidity and chlorides
- quality generally improves with increasing pH
- quality generally improves with increasing alcohol (and decreasing density)

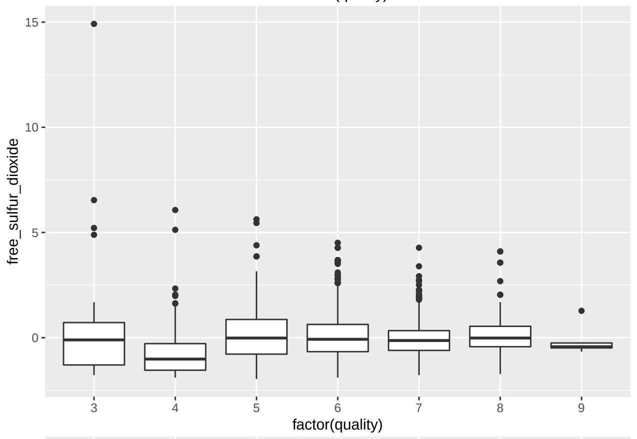


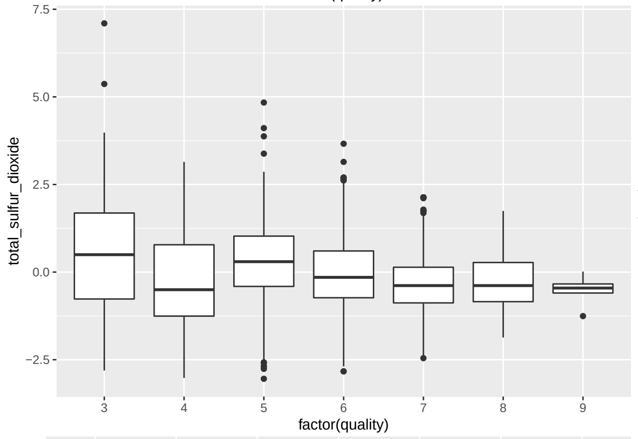


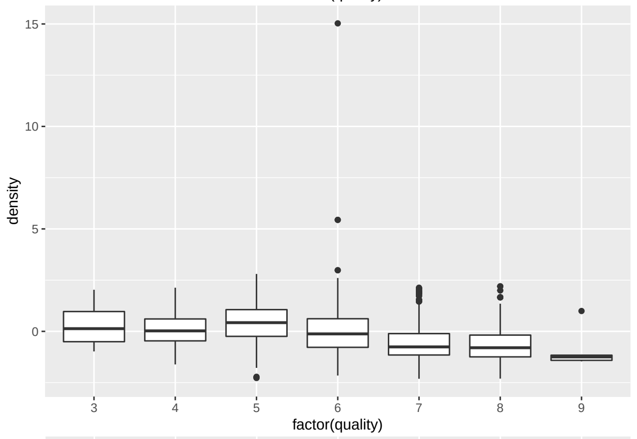

citric_acid
8
4
0
3 4 5 6 7 8 9
factor(quality)

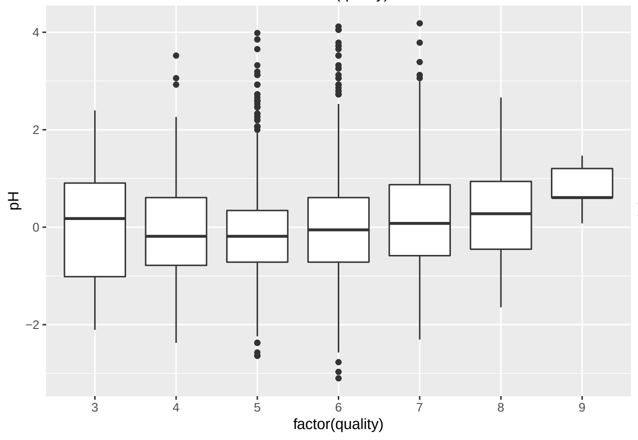

residual_sugar
10.0
7.5
5.0
2.5
0.0
3 4 5 6 7 8 9
factor(quality)

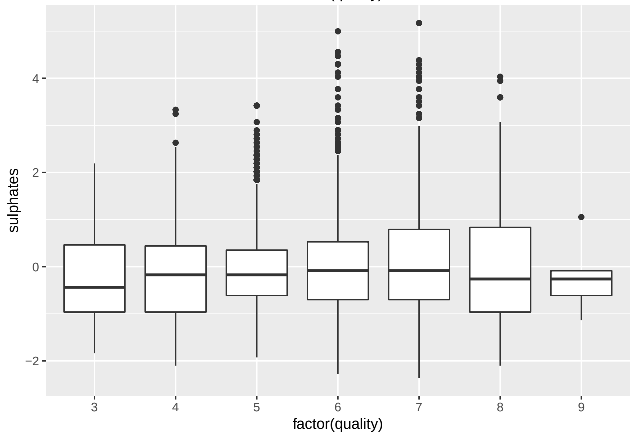

chlorides
10
5
0
3 4 5 6 7 8 9
factor(quality)

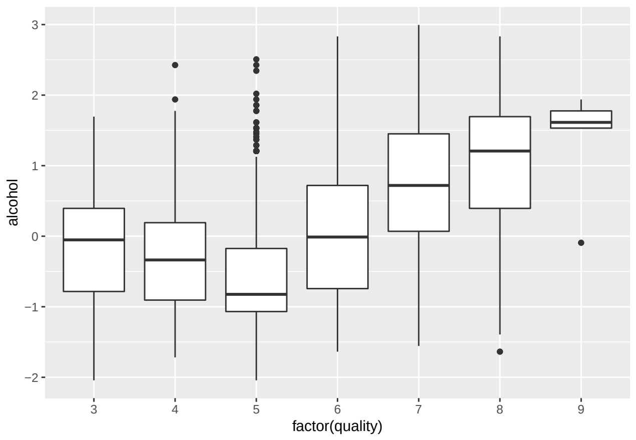

free_sulfur_dioxide
15
10
5
0
3 4 5 6 7 8 9
factor(quality)


total_sulfur_dioxide
7.5
5.0
2.5
0.0
−2.5
3 4 5 6 7 8 9
factor(quality)


density
15
10
5
0
3 4 5 6 7 8 9
factor(quality)


pH
4
2
0
−2
3 4 5 6 7 8 9
factor(quality)


sulphates
4
2
0
−2
3 4 5 6 7 8 9
factor(quality)



White Wine Feature Correlations Though not expected to be impactful for regression predictions, we take a look at the correlation structure and multicollinearity of the independent variables. The high VIF for density and associated factors (alcohol, sugar concentration) makes sense chemically. We can see that removing density as a feature might make sense, but it's not believed to be necessary in this context.

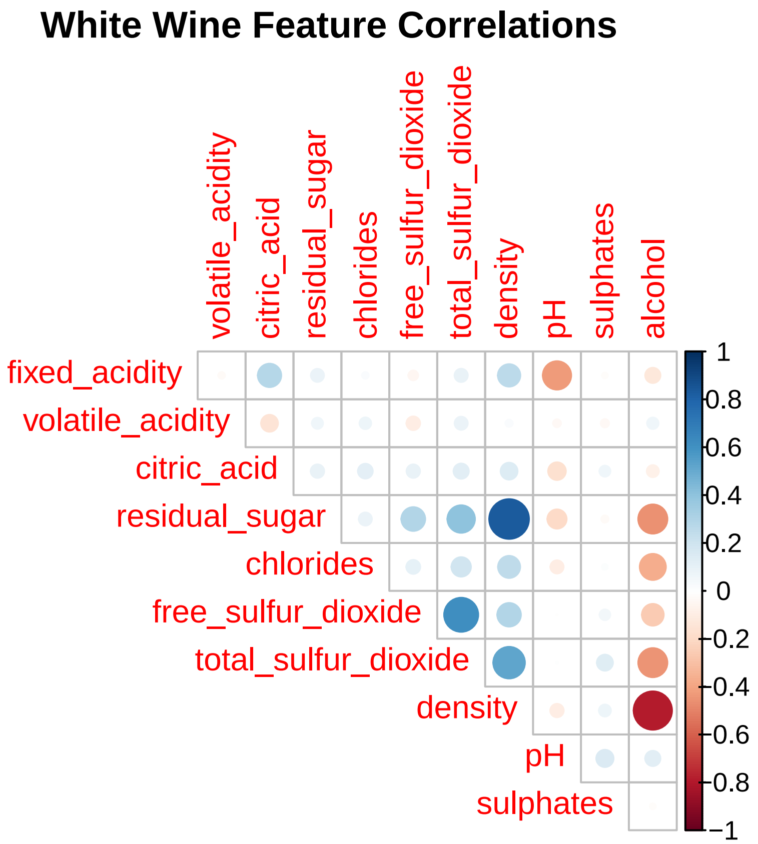


Table 3: White Wine VIF

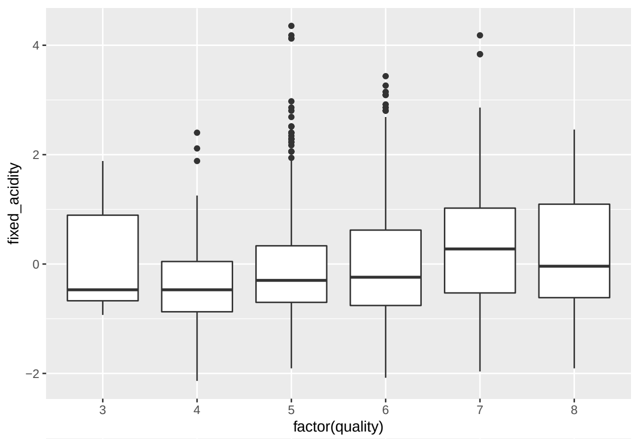
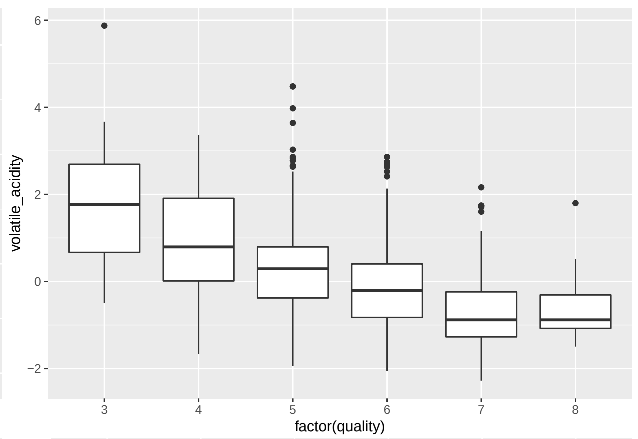
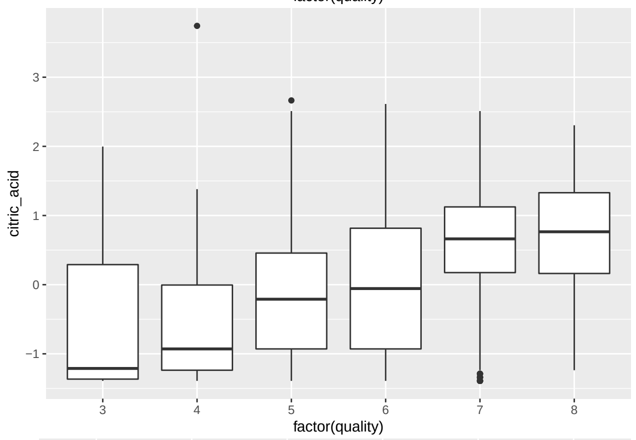
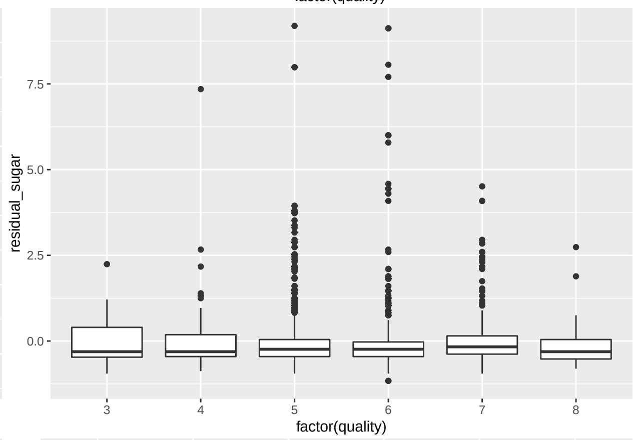
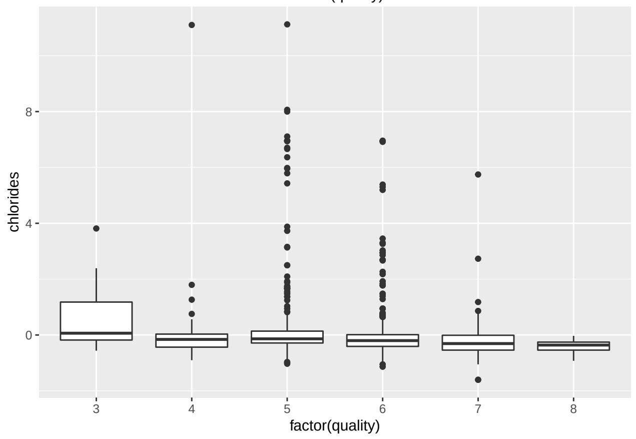
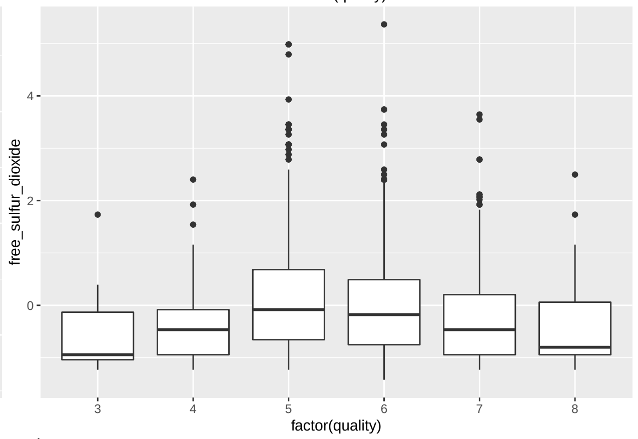
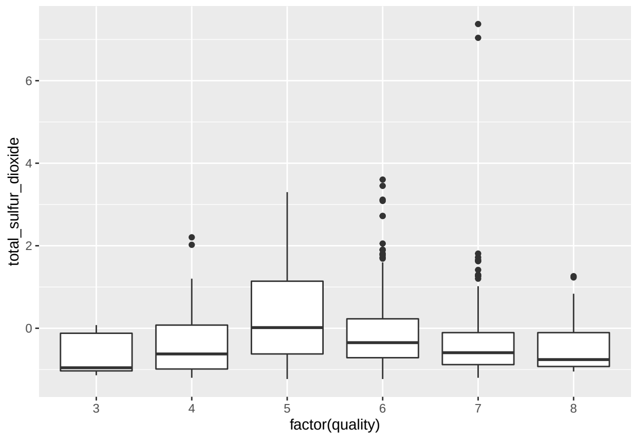
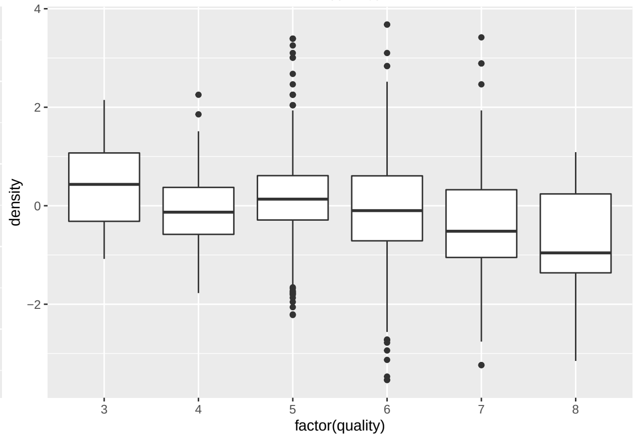
	x
fixed_acidity	2.69
volatile_acidity	1.14
citric_acid	1.17
residual_sugar	12.64
chlorides	1.24
free_sulfur_dioxide	1.79
total_sulfur_dioxide	2.24
density	28.23
pH	2.20
sulphates	1.14
alcohol	7.71

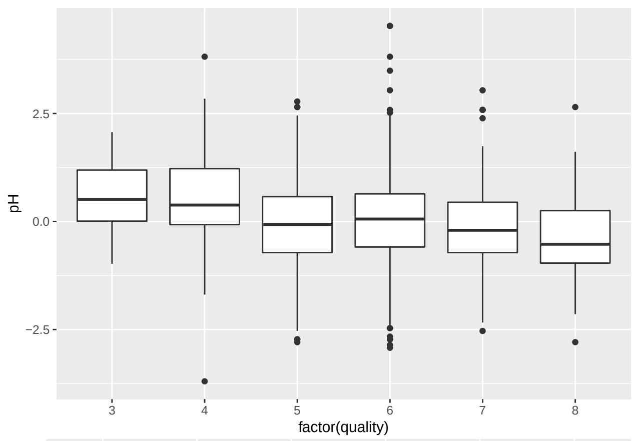
Table 4: White Wine VIF (no density)

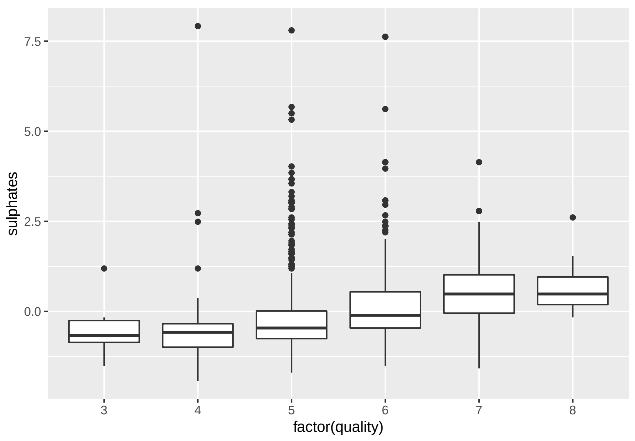
	x
fixed_acidity	1.36
volatile_acidity	1.13
citric_acid	1.16
residual_sugar	1.44
chlorides	1.20
free_sulfur_dioxide	1.75
total_sulfur_dioxide	2.15
pH	1.33
sulphates	1.06
alcohol	1.65

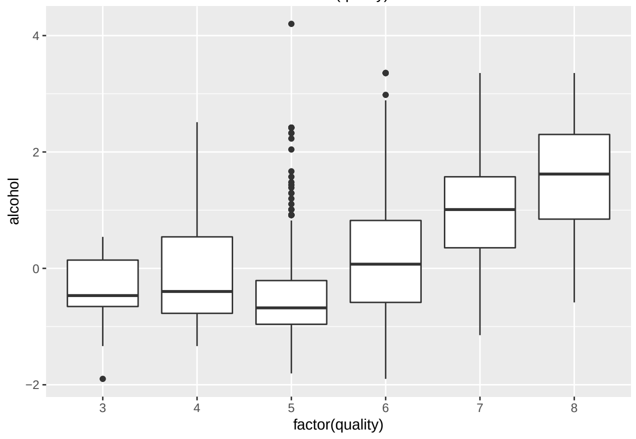
Red Wine Features We follow the same steps to standardize red wine features and examine the relationship between quality categories and chemical markers.

- quality improves with decreasing volatile acidity, chlorides and pH
- quality improves with increasing citric acid and sulphates
- quality improves with decreasing density / increasing alcohol


fixed_acidity
4
2
0
−2
3 4 5 6 7 8
factor(quality)

volatile_acidity
6
4
2
0
−2
3 4 5 6 7 8
factor(quality)

citric_acid
3
2
1
0
−1
3 4 5 6 7 8
factor(quality)

residual_sugar
7.5
5.0
2.5
0.0
3 4 5 6 7 8
factor(quality)

chlorides
8
4
0
3 4 5 6 7 8
factor(quality)

free_sulfur_dioxide
4
2
0
3 4 5 6 7 8
factor(quality)

total_sulfur_dioxide
6
4
2
0
3 4 5 6 7 8
factor(quality)

density
4
2
0
−2
3 4 5 6 7 8
factor(quality)







Red Wine Feature Correlations The various acidity measures are positively correlated as are the sulfur dioxide measures. As with white wines, there is a negative correlation between density and alcohol. Fixed acidity is positively correlated with density and negatively correlated with pH. We see a certain degree of multicollinearity, but dispersed across more variables than in the case of white wines.

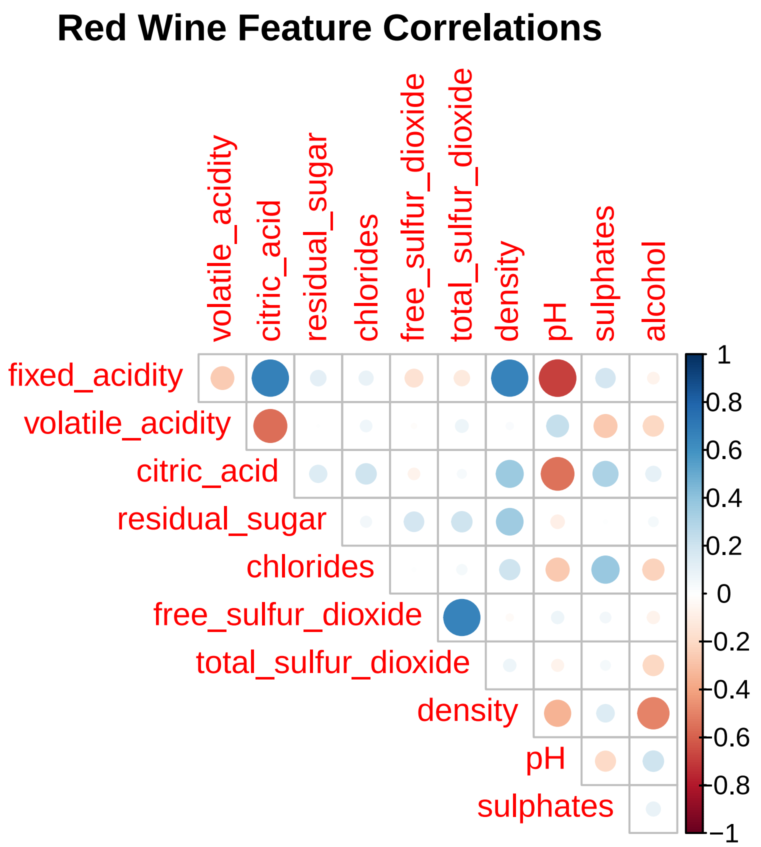


Table 5: Red Wine VIF

	x
fixed_acidity	7.77
volatile_acidity	1.79
citric_acid	3.13
residual_sugar	1.70
chlorides	1.48
free_sulfur_dioxide	1.96
total_sulfur_dioxide	2.19
density	6.34
pH	3.33
sulphates	1.43
alcohol	3.03

Modeling Approach

There are some tradeoffs in selecting a metric for "best" model for predicting wine quality. Given the bell curve of ratings, we immediately see a prevalence issue where the most important ratings, e.g very good and very bad, are under-represented relative to "average" wines.

There are very few ratings in the "3" and "9" categories. There is also little practical use to a rating system this granular. Consequently we shall collapse to three rating classes as follows:

Bad (0): Ratings 3 & 4
Ok (1): Ratings 5 & 6

Good (2): Ratings 7-9

We will consider a linear regression as a baseline prediction and compare a number of other regression and classification models for potential improvement. Regression model results will be rounded to allow for consistent treatment of scores as factors and usage of a confusion matrix to generate prediction quality statistics.

```r
white_wines <- white_wines %>%
  mutate(quality = ifelse(quality < 6, 0, ifelse(quality > 6, 2 , 1)))
red_wines <- red_wines %>%
  mutate(quality = ifelse(quality < 6, 0, ifelse(quality > 6, 2 , 1)))
```

Data Partitioning For both white and red wines, we'll create an 80/20 split for training and testing models. This allows for a significant and representative sample in the test set while maximizing the data available for training. Note that we convert the ratings to a factor to ensure that the partitioning matches the proportion of class prevalence across the train/test sets.

```r
y_white <- white_wines$quality
y_red <- red_wines$quality
w_test_index <- createDataPartition(factor(y_white), p = 0.2, times = 1, list = FALSE)
r_test_index <- createDataPartition(factor(y_red), p = 0.2, times = 1, list = FALSE)
w_train <- white_wines[-w_test_index,]
w_test <- white_wines[w_test_index,]
r_train <- red_wines[-r_test_index,]
r_test <- red_wines[r_test_index,]
options(digits = 4)
w_train %>% group_by(quality) %>% summarize(n_trn=n() / nrow(w_train)) %>%
  cbind(w_test %>% group_by(quality) %>% summarize(n_tst=n() / nrow(w_test)) %>%
          select(n_tst)) %>%
          kbl(caption = "White Wine Partition Proportions", booktabs = T) %>%
          row_spec(0,bold=TRUE) %>%
          kable_styling(latex_options = c("striped","HOLD_position"))
```

Table 6: White Wine Partition Proportions

quality	**n_trn**	**n_tst**
0	0.3349	0.3347
1	0.4487	0.4490
2	0.2164	0.2163

Regression Models In addition to running a standard linear regression, there are several model types that can run as regression or classification type models. The caret package simply looks at the dependent variable type (factor / non-factor) to determine which mode to execute. The quality ratings are still in numeric form so caret will employ the regression approach for the selected models.

- "lm" for linear regression
- "knn" for k-nearest neighbors
- "gamLoess" for localized least squares fitting on different regions
- "rf" for random forest algorithm

```r
reg_models <- c("lm", "knn", "gamLoess", "rf")
w_reg_fits <- lapply(reg_models, function(model){
  train(quality ~ ., method = model, data = w_train)
})
r_reg_fits <- lapply(reg_models, function(model){
  train(quality ~ ., method = model, data = r_train)
})
names(w_reg_fits) <- reg_models
names(r_reg_fits) <- reg_models

#Define levels of classification for confusion matrix
q_levels <- factor(c("0","1","2"),ordered = T)
```

The section below defines a function that will iterate through a set of model fits and store key results from a confusion matrix calculation in a specified tibble.

```
#create a reusable function to process model fits with test data.
parse_fits <- function(fits, test_set, m_summary, m_class = TRUE) {
  for (i in 1:length(fits)) {
    m_predicted <-predict(fits[[i]], test_set)
    y_test <- test_set$quality
    m_suff <- "_cls"
    #coerce to factors if regression results
    if (m_class != TRUE) {
      m_predicted <- factor(round(m_predicted, 0), levels = q_levels)
      y_test <- factor(round(y_test, 0), levels = q_levels)
      m_suff <- "_reg"
    }
    #generate the confusion matrix with predicted values
    m_cm <-confusionMatrix(m_predicted, y_test)
    #add new row to model summary
    m_summary <-m_summary %>%
      rbind(tibble(model = str_c(fits[[i]]$method, m_suff),
          acc = m_cm$overall["Accuracy"],
          c0_sens = m_cm$byClass[1, 1],
          c0_spec = m_cm$byClass[1, 2],
          c0_f1 = m_cm$byClass[1, 7],
          c1_sens = m_cm$byClass[2, 1],
          c1_spec = m_cm$byClass[2, 2],
          c1_f1 = m_cm$byClass[2, 7],
          c2_sens = m_cm$byClass[3, 1],
          c2_spec = m_cm$byClass[3, 2],
          c2_f1 = m_cm$byClass[3, 7]
        )
      )
  }
  return(m_summary)
}
```

Now we run the function for the white and red wine fits to store the results in the respective summary tibbles.

```
#Create a shell tibble to populate with model results
w_model_summary <- tibble(model = "none", acc = 0, c0_sens = 0, c0_spec = 0,  c0_f1 = 0,
                c1_sens = 0, c1_spec = 0, c1_f1 = 0, c2_sens = 0, c2_spec = 0, c2_f1 = 0)
r_model_summary <- copy(w_model_summary)
w_model_summary <- parse_fits(w_reg_fits, w_test, w_model_summary, m_class = FALSE) %>%
  filter(acc > 0)
```

```
r_model_summary <- parse_fits(r_reg_fits, r_test, r_model_summary, m_class = FALSE) %>%
  filter(acc > 0)
```

Classification Models Next we convert the quality rating to a factor and repeat the process of model training with a set of candidate classification models.

- naive_bayes : applies Baye's theorem with posterior probability
- svmLinear : linear support vector machine
- knn : k-nearest neighbors
- gamLoess : generalized additive model; locally weighted lest squares
- multinom : neural net based classification
- rf : random forest
- polr : proportional ordered logistic regression
- xgboost: Extreme gradient boosted tree

```
white_wines <- white_wines %>% mutate(quality = factor(quality, levels = q_levels))
w_train <- w_train %>% mutate(quality = factor(quality, levels = q_levels))
w_test <- w_test %>% mutate(quality = factor(quality, levels = q_levels))
red_wines <- red_wines %>% mutate(quality = factor(quality, levels = q_levels))
r_train <- r_train %>% mutate(quality = factor(quality, levels = q_levels))
r_test <- r_test %>% mutate(quality = factor(quality, levels = q_levels))
```

```
class_models <- c("naive_bayes", "svmLinear", "knn", "gamLoess", "multinom", "rf", "polr")
w_class_fits <- lapply(class_models, function(model){
  train(quality ~ ., method = model, data = w_train)
})
r_class_fits <- lapply(class_models, function(model){
  train(quality ~ ., method = model, data = r_train)
})
names(w_class_fits) <- class_models
names(r_class_fits) <- class_models
```

```
w_model_summary <- parse_fits(w_class_fits, w_test, w_model_summary)
r_model_summary <- parse_fits(r_class_fits, r_test, r_model_summary)
```

```
w_train_x <- w_train %>% select(-quality) %>% data.matrix()
w_train_y <- as.numeric(as.character(w_train$quality))
w_test_x <- w_test %>% select(-quality) %>% data.matrix()
w_test_y <- as.numeric(as.character(w_test$quality))
w_xgb_train <- xgb.DMatrix(data = w_train_x, label = w_train_y)
w_xgb_test <- xgb.DMatrix(data = w_test_x, label = w_test_y)

r_train_x <- r_train %>% select(-quality) %>% data.matrix()
r_train_y <- as.numeric(as.character(r_train$quality))
r_test_x <- r_test %>% select(-quality) %>% data.matrix()
r_test_y <- as.numeric(as.character(r_test$quality))
r_xgb_train <- xgb.DMatrix(data = r_train_x, label = r_train_y)
r_xgb_test <- xgb.DMatrix(data = r_test_x, label = r_test_y)

w_watch <- list(train=w_xgb_train, test=w_xgb_test)
r_watch <- list(train=r_xgb_train, test=r_xgb_test)
```

```r
xgb_params = list(booster = "gbtree",
               objective = "multi:softmax",
               num_class = 3)
w_xgb_fit <- xgb.train(data = w_xgb_train, max.depth = 3, watchlist=w_watch,
                       params = xgb_params, nrounds = 70)
r_xgb_fit <- xgb.train(data = r_xgb_train, max.depth = 3, watchlist=r_watch,
                       params = xgb_params, nrounds = 70)

#generate same measures as with other models trained in caret
w_xgb_cm <- confusionMatrix(factor(predict(w_xgb_fit, w_xgb_test),
                                   levels = q_levels), w_test$quality)
w_model_summary <- w_model_summary %>% rbind(tibble(model = "xgbtree_cls",
                    acc = w_xgb_cm$overall["Accuracy"], c0_sens = w_xgb_cm$byClass[1, 1],
                    c0_spec = w_xgb_cm$byClass[1, 2], c0_f1 = w_xgb_cm$byClass[1, 7],
                    c1_sens = w_xgb_cm$byClass[2, 1], c1_spec = w_xgb_cm$byClass[2, 2],
                    c1_f1 = w_xgb_cm$byClass[2, 7], c2_sens = w_xgb_cm$byClass[3, 1],
                    c2_spec = w_xgb_cm$byClass[3, 2], c2_f1 = w_xgb_cm$byClass[3, 7]))

r_xgb_cm <- confusionMatrix(factor(predict(r_xgb_fit, r_xgb_test),
                                   levels = q_levels), r_test$quality)
r_model_summary <- r_model_summary %>% rbind(tibble(model = "xgbtree_cls",
                    acc = w_xgb_cm$overall["Accuracy"], c0_sens = w_xgb_cm$byClass[1, 1],
                    c0_spec = w_xgb_cm$byClass[1, 2], c0_f1 = w_xgb_cm$byClass[1, 7],
                    c1_sens = w_xgb_cm$byClass[2, 1], c1_spec = w_xgb_cm$byClass[2, 2],
                    c1_f1 = w_xgb_cm$byClass[2, 7], c2_sens = w_xgb_cm$byClass[3, 1],
                    c2_spec = w_xgb_cm$byClass[3, 2], c2_f1 = w_xgb_cm$byClass[3, 7]))
```

Results

At this point we have trained a range prediction models on both white and red wine quality and generated summary tables of the prediction results against a held-out test set that was not available to the model training process. Although we will consider a range of confusion matrix statistics in selecting our best model for each wine type, we consider it especially important to predict "Bad" wines correctly. This translates to the "sensitivity" measure for class 0 in our summary tables. Our initial sort of the summaries will reflect this.

White Wine Model Results The random forest classification model provides the best overall accuracy as well as the best balanced performance across the classes. Compared to the linear regression model, we see significant improvement in the overall accuracy (74% vs. 55%) and the sensitivity on Bad wine predictions rises from 37.5% to 74.4%.

Table 7: White Wine Results

model	acc	c0_sens	c0_spec	c0_f1	c1_sens	c1_spec	c1_f1	c2_sens	c2_spec	c2_f1
lm_reg	0.551	0.375	0.922	0.490	0.856	0.315	0.635	0.189	0.975	0.295
gamLoess_reg	0.567	0.473	0.905	0.569	0.797	0.387	0.625	0.236	0.960	0.341
knn_reg	0.593	0.558	0.857	0.606	0.691	0.526	0.608	0.443	0.935	0.528
multinom_cls	0.588	0.607	0.839	0.630	0.691	0.544	0.614	0.344	0.931	0.432
gamLoess_cls	0.559	0.616	0.854	0.646	0.786	0.376	0.616	0.000	1.000	NA
svmLinear_cls	0.559	0.619	0.831	0.633	0.784	0.404	0.623	0.000	1.000	NA
knn_cls	0.595	0.628	0.817	0.631	0.591	0.644	0.583	0.552	0.888	0.564
rf_reg	0.701	0.640	0.928	0.718	0.855	0.578	0.720	0.476	0.977	0.610
naive_bayes_cls	0.563	0.677	0.771	0.635	0.473	0.748	0.531	0.575	0.814	0.512
polr_cls	0.574	0.683	0.768	0.637	0.675	0.550	0.606	0.198	0.970	0.303
xgbtree_cls	0.655	0.683	0.862	0.698	0.702	0.652	0.660	0.514	0.922	0.572
rf_cls	0.740	0.744	0.883	0.753	0.780	0.720	0.734	0.651	0.964	0.730

White Wine Random Forest Confusion Matrix We can look at the exact nature of hits and misses with the confusion matrix for the random forest classification prediction model. Of particular note, extremely bad predictions are rare. Of 328 white wines rated as "Bad" in the test set, only 4 are predicted to be "Good" wines. Likewise, out of 212 white wines rated as "Good", only 3 are predicted to be "Bad".

Table 8: White Wine Predictions: Random Forest Classifier

Predicted	Reference		
	Bad	**Ok**	**Good**
Bad	244	73	3
Ok	80	343	71
Good	4	24	138

White Wine Variable Importance There is an important opportunity for feedback between the prediction model and the actual data collection process. For instance, a variable that is not useful for prediction may reflect an unnecessary lab cost while chemical markers known to be useful may indicate other areas of data collection.

Alcohol and density appear as the most important features in the random forest model, while sulphates and fixed acidity are the least important.

Table 9: White Wine RF Variable Importance

	Overall
alcohol	100.000
density	89.342
volatile_acidity	55.085
free_sulfur_dioxide	43.499
total_sulfur_dioxide	33.938
chlorides	31.545
residual_sugar	31.533
pH	20.163
citric_acid	15.341
sulphates	3.559
fixed_acidity	0.000

Red Wine Model Results Although not the very best at detecting bad wines, the random forest classification model is again the best overall prediction model. Compared to the linear regression model, we see significant improvement in the overall accuracy (73% vs. 65%) and the sensitivity on Bad wine predictions rises from 73% to 83%.

Table 10: Red Wine Results

model	**acc**	**c0_sens**	**c0_spec**	**c0_f1**	**c1_sens**	**c1_spec**	**c1_f1**	**c2_sens**	**c2_spec**	**c2_f1**
gamLoess_reg	0.611	0.671	0.785	0.699	0.680	0.565	0.582	0.205	0.986	0.316
xgbtree_cls	0.655	0.683	0.862	0.698	0.702	0.652	0.660	0.514	0.922	0.572
knn_reg	0.620	0.691	0.779	0.710	0.641	0.622	0.580	0.318	0.960	0.406
lm_reg	0.648	0.725	0.802	0.742	0.734	0.596	0.627	0.136	0.996	0.235
knn_cls	0.604	0.732	0.709	0.708	0.531	0.689	0.531	0.386	0.939	0.436
rf_reg	0.717	0.738	0.866	0.780	0.812	0.653	0.696	0.364	0.996	0.525
gamLoess_cls	0.607	0.745	0.744	0.730	0.656	0.575	0.571	0.000	1.000	NA
naive_bayes_cls	0.620	0.799	0.686	0.739	0.422	0.793	0.486	0.591	0.899	0.531
multinom_cls	0.654	0.826	0.715	0.766	0.562	0.725	0.569	0.341	0.968	0.441
rf_cls	0.729	0.826	0.814	0.809	0.672	0.767	0.664	0.568	0.964	0.633
polr_cls	0.648	0.826	0.709	0.764	0.578	0.699	0.569	0.250	0.982	0.367
svmLinear_cls	0.632	0.839	0.692	0.765	0.547	0.699	0.547	0.182	0.975	0.271

Red Wine Random Forest Confusion Matrix We can look at the exact nature of hits and misses with the confusion matrix for the random forest classification prediction model. Of particular note, extremely bad predictions are rare. Of 149 red wines rated as "Bad" in the test set, none are predicted to be "Good" wines. Likewise, out of 44 red wines rated as "Good", none are predicted to be "Bad".

Table 11: Red Wine Predictions: Random Forest Classifier

Predicted	Reference		
	Bad	**Ok**	**Good**
Bad	123	31	0
Ok	26	87	19
Good	0	10	25

Red Wine Variable Importance Alcohol and sulphates appear as the most important features in the random forest model, while residual sugar and free sulphur dioxide are the least important.

Table 12: Red Wine RF Variable Importance

	Overall
alcohol	100.000
sulphates	65.848
volatile_acidity	51.854
total_sulfur_dioxide	51.541
density	36.742
chlorides	20.396
citric_acid	19.030
fixed_acidity	11.297
pH	9.423
residual_sugar	3.069
free_sulfur_dioxide	0.000

Conclusion

This study constructed two wine quality prediction models based on existing quality ratings and an associated set of chemical markers for the wine. (e.g. residual sugar) In addition to linear regression, a variety of machine learning models were trained and tested for prediction capabilities. In the case of both white and red wines, the random forest model provided the best predictions.

A clear understanding of the relationship between wine chemistry and subjective quality ratings has a large potential impact on commercial wine-making efforts. If one were to work backwards, this could impact selection and blending of grape stocks at the crush stage and the subsequent choices about time, temperature and other variables during fermentation. Looking forward from the point of obtaining the final wine, there are choices about how to market the wine and whether to even undergo a rating process that could have material economic impact to winemakers.

This modeling effort was limited with respect to what could potentially be done with the available data. Generally speaking, default training and tuning parameters from caret were utilized. Future work might involve deeper feature analysis, potentially adding ratios and dropping features without material impact on quality predictions. Each model could be subjected to a greater range of tuning grid parameters to further explore their capabilities. There are other machine learning models to test as well for improvement on the random forest approach. More broadly, a data scientist could engage with chemists to uncover novel chemical features to collect alongside the quality ratings for each wine.