Assignment 7

Maksim Nikiforov

Read in the data set

Since we have a comma-separated file, we can read in our training data using the function `read_csv()`.

```
# Read in CSV train file, subset data
wineTrainingData <- as_tibble(read.csv(file = "./wineQualityTrain.csv"))
names(wineTrainingData)
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
## [13] "type"
```

Plan our models

When planning our initial models, we can either start with the single best variable and add more or consider all variables and remove ones which are the least significant. When considering a subset, we can remove certain predictors if they are colinear (highly correlated within the same model).

In this case, we can start with a model that incorporates all terms.

```
# Create model with all terms
linearModel1 <- lm(quality ~ ., data = wineTrainingData)
summary(linearModel1)
```

```
##
## Call:
## lm(formula = quality ~ ., data = wineTrainingData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3559 -0.4734 -0.0486  0.4474  2.7786
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.511e+02  3.733e+01   4.048 5.47e-05 ***
## fixed.acidity         1.010e-01  3.855e-02   2.620 0.008909 **
## volatile.acidity     -1.529e+00  1.849e-01  -8.267 3.40e-16 ***
## citric.acid           7.198e-02  1.888e-01   0.381 0.703133
## residual.sugar        8.246e-02  1.475e-02   5.592 2.74e-08 ***
## chlorides            -1.375e+00  8.027e-01  -1.713 0.087039 .
## free.sulfur.dioxide   3.337e-03  1.626e-03   2.053 0.040320 *
## total.sulfur.dioxide -1.538e-03  7.115e-04  -2.161 0.030867 *
## density              -1.494e+02  3.780e+01  -3.953 8.15e-05 ***
## pH                    2.889e-01  2.105e-01   1.373 0.170142
```

```
## sulphates             7.750e-01  1.804e-01   4.296 1.87e-05 ***
## alcohol               1.749e-01  4.796e-02   3.648 0.000275 ***
## typeWhite            -5.307e-01  1.332e-01  -3.985 7.13e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7499 on 1287 degrees of freedom
## Multiple R-squared:  0.3089, Adjusted R-squared:  0.3025
## F-statistic: 47.95 on 12 and 1287 DF,  p-value: < 2.2e-16
```

We can then see which predictor variable we may want to delete.

```
# Compute AIC values to drop the lowest one(s)
drop1(linearModel1)
```

```
## Single term deletions
##
## Model:
## quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol + type
##                      Df Sum of Sq    RSS     AIC
## <none>                             723.81 -735.27
## fixed.acidity         1     3.859 727.67 -730.36
## volatile.acidity      1    38.433 762.24 -670.01
## citric.acid           1     0.082 723.89 -737.12
## residual.sugar        1    17.588 741.40 -706.06
## chlorides             1     1.649 725.46 -734.31
## free.sulfur.dioxide   1     2.369 726.18 -733.02
## total.sulfur.dioxide  1     2.627 726.44 -732.56
## density               1     8.787 732.60 -721.58
## pH                    1     1.059 724.87 -735.37
## sulphates             1    10.378 734.19 -718.76
## alcohol               1     7.484 731.29 -723.90
## type                  1     8.930 732.74 -721.33
```

Here, the Akaike information criterion (AIC) is lowest for `citric.acid` (-737.12), so we remove this term from our model.

```
# Get model formula
formula(linearModel1)
```

```
## quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol + type
```

```
# Create model without citric.acid
linearModel2 <- lm(quality ~ fixed.acidity + volatile.acidity + residual.sugar +
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + pH + sulphates + alcohol + type, data = wineTrainingData)
summary(linearModel2)
```

```
##
## Call:
## lm(formula = quality ~ fixed.acidity + volatile.acidity + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol + type, data = wineTrainingData)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3462 -0.4735 -0.0507  0.4449  2.7738
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.501e+02  3.723e+01   4.033 5.83e-05 ***
## fixed.acidity         1.038e-01  3.784e-02   2.742 0.006195 **
## volatile.acidity     -1.556e+00  1.708e-01  -9.105  < 2e-16 ***
## residual.sugar        8.223e-02  1.473e-02   5.583 2.88e-08 ***
## chlorides            -1.339e+00  7.970e-01  -1.680 0.093180 .
## free.sulfur.dioxide   3.314e-03  1.624e-03   2.040 0.041538 *
## total.sulfur.dioxide -1.500e-03  7.044e-04  -2.130 0.033394 *
## density              -1.484e+02  3.770e+01  -3.937 8.69e-05 ***
## pH                    2.807e-01  2.093e-01   1.341 0.180202
## sulphates             7.760e-01  1.803e-01   4.303 1.81e-05 ***
## alcohol               1.777e-01  4.741e-02   3.748 0.000186 ***
## typeWhite            -5.276e-01  1.329e-01  -3.970 7.58e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7497 on 1288 degrees of freedom
## Multiple R-squared:  0.3089, Adjusted R-squared:  0.303
## F-statistic: 52.33 on 11 and 1288 DF,  p-value: < 2.2e-16
```

We can then see which predictor variable we may want to delete next.

```
# Compute AIC values to drop the lowest one(s)
drop1(linearModel2)
```

```
## Single term deletions
##
## Model:
## quality ~ fixed.acidity + volatile.acidity + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol + type
##                      Df Sum of Sq    RSS     AIC
## <none>                             723.89 -737.12
## fixed.acidity         1     4.225 728.12 -731.56
## volatile.acidity      1    46.588 770.48 -658.04
## residual.sugar        1    17.519 741.41 -708.04
## chlorides             1     1.586 725.48 -736.28
## free.sulfur.dioxide   1     2.339 726.23 -734.93
## total.sulfur.dioxide  1     2.549 726.44 -734.55
## density               1     8.712 732.60 -723.57
## pH                    1     1.010 724.90 -737.31
## sulphates             1    10.408 734.30 -720.56
## alcohol               1     7.894 731.79 -725.02
## type                  1     8.859 732.75 -723.31
```

Here, the AIC is lowest for `pH` (-737.31), so we remove this term from our model.

```
# Create model without citric.acid, pH
linearModel3 <- lm(quality ~ fixed.acidity + volatile.acidity + residual.sugar +
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
```

```
    density + sulphates + alcohol + type, data = wineTrainingData)
summary(linearModel3)
```

```
##
## Call:
## lm(formula = quality ~ fixed.acidity + volatile.acidity + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + sulphates + alcohol + type, data = wineTrainingData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3398 -0.4771 -0.0360  0.4510  2.7295
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.186e+02  2.888e+01   4.107 4.26e-05 ***
## fixed.acidity         6.542e-02  2.480e-02   2.638  0.00843 **
## volatile.acidity     -1.569e+00  1.706e-01  -9.197  < 2e-16 ***
## residual.sugar        6.975e-02  1.142e-02   6.109 1.32e-09 ***
## chlorides            -1.458e+00  7.922e-01  -1.841  0.06591 .
## free.sulfur.dioxide   3.446e-03  1.622e-03   2.125  0.03379 *
## total.sulfur.dioxide -1.584e-03  7.018e-04  -2.257  0.02417 *
## density              -1.159e+02  2.885e+01  -4.016 6.26e-05 ***
## sulphates             7.400e-01  1.784e-01   4.148 3.57e-05 ***
## alcohol               2.149e-01  3.845e-02   5.589 2.79e-08 ***
## typeWhite            -4.947e-01  1.307e-01  -3.787  0.00016 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7499 on 1289 degrees of freedom
## Multiple R-squared:  0.3079, Adjusted R-squared:  0.3025
## F-statistic: 57.35 on 10 and 1289 DF,  p-value: < 2.2e-16
```

We repeat this process several more times.

```
# Compute AIC values to drop the lowest one(s)
drop1(linearModel3)
```

```
## Single term deletions
##
## Model:
## quality ~ fixed.acidity + volatile.acidity + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + sulphates + alcohol + type
##                       Df Sum of Sq    RSS     AIC
## <none>                             724.90 -737.31
## fixed.acidity          1     3.914 728.82 -732.31
## volatile.acidity       1    47.567 772.47 -656.69
## residual.sugar         1    20.989 745.89 -702.20
## chlorides              1     1.905 726.81 -735.90
## free.sulfur.dioxide    1     2.539 727.44 -734.76
## total.sulfur.dioxide   1     2.865 727.77 -734.18
## density                1     9.071 733.97 -723.14
## sulphates              1     9.678 734.58 -722.07
## alcohol                1    17.565 742.47 -708.18
```

```
## type                 1     8.064 732.97 -724.93
```

```
# Create model without citric.acid, pH, chlorides
linearModel4 <- lm(quality ~ fixed.acidity + volatile.acidity + residual.sugar +
                     free.sulfur.dioxide + total.sulfur.dioxide +
                     density + sulphates + alcohol + type, data = wineTrainingData)
summary(linearModel4)
```

```
##
## Call:
## lm(formula = quality ~ fixed.acidity + volatile.acidity + residual.sugar +
##     free.sulfur.dioxide + total.sulfur.dioxide + density + sulphates +
##     alcohol + type, data = wineTrainingData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3249 -0.4731 -0.0409  0.4464  2.7390
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.213e+02  2.887e+01   4.200 2.85e-05 ***
## fixed.acidity         6.485e-02  2.482e-02   2.613  0.00907 **
## volatile.acidity     -1.600e+00  1.699e-01  -9.416  < 2e-16 ***
## residual.sugar        7.178e-02  1.137e-02   6.311 3.82e-10 ***
## free.sulfur.dioxide   3.299e-03  1.621e-03   2.035  0.04210 *
## total.sulfur.dioxide -1.546e-03  7.022e-04  -2.202  0.02785 *
## density              -1.187e+02  2.884e+01  -4.117 4.09e-05 ***
## sulphates             7.180e-01  1.781e-01   4.031 5.89e-05 ***
## alcohol               2.237e-01  3.818e-02   5.860 5.87e-09 ***
## typeWhite            -4.679e-01  1.300e-01  -3.600  0.00033 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7506 on 1290 degrees of freedom
## Multiple R-squared:  0.3061, Adjusted R-squared:  0.3012
## F-statistic: 63.22 on 9 and 1290 DF,  p-value: < 2.2e-16
```

```
# Compute AIC values to drop the lowest one(s)
drop1(linearModel4)
```

```
## Single term deletions
##
## Model:
## quality ~ fixed.acidity + volatile.acidity + residual.sugar +
##     free.sulfur.dioxide + total.sulfur.dioxide + density + sulphates +
##     alcohol + type
##                       Df Sum of Sq    RSS     AIC
## <none>                             726.81 -735.90
## fixed.acidity          1     3.848 730.65 -731.03
## volatile.acidity       1    49.958 776.77 -651.48
## residual.sugar         1    22.437 749.24 -698.37
## free.sulfur.dioxide    1     2.332 729.14 -733.73
## total.sulfur.dioxide   1     2.732 729.54 -733.02
## density                1     9.548 736.35 -720.93
## sulphates              1     9.154 735.96 -721.63
```

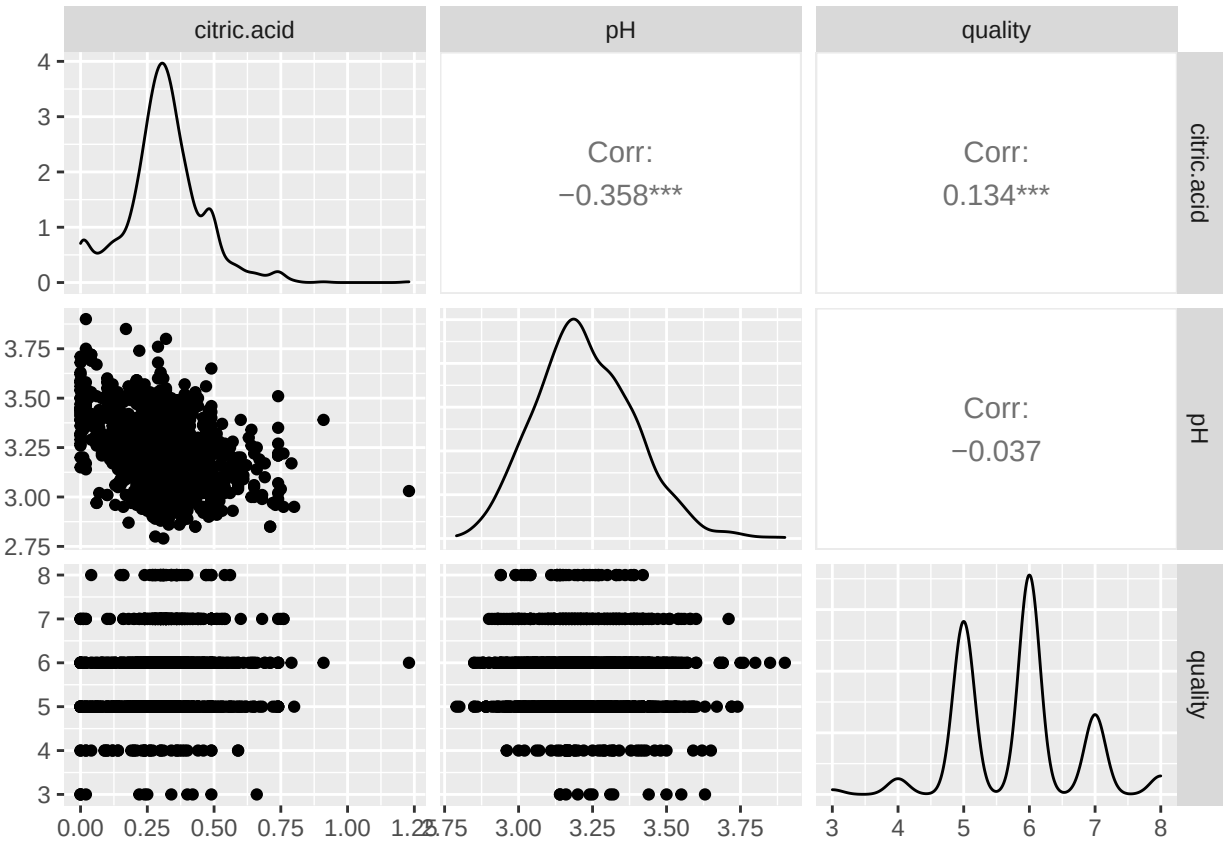
```
## alcohol               1    19.348 746.15 -703.74
## type                  1     7.304 734.11 -724.90
```

```
# Create model without citric.acid, pH, chlorides, free.sulfur.dioxide
linearModel5 <- lm(quality ~ fixed.acidity + volatile.acidity + residual.sugar +
                    total.sulfur.dioxide + density + sulphates + alcohol +
                     type, data = wineTrainingData)
summary(linearModel5)
```

```
##
## Call:
## lm(formula = quality ~ fixed.acidity + volatile.acidity + residual.sugar +
##     total.sulfur.dioxide + density + sulphates + alcohol + type,
##     data = wineTrainingData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3197 -0.4834 -0.0477  0.4459  2.7752
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           1.274e+02  2.875e+01   4.429 1.03e-05 ***
## fixed.acidity         6.348e-02  2.484e-02   2.556   0.0107 *
## volatile.acidity     -1.645e+00  1.687e-01  -9.752  < 2e-16 ***
## residual.sugar        7.497e-02  1.128e-02   6.647 4.41e-11 ***
## total.sulfur.dioxide -6.991e-04  5.661e-04  -1.235   0.2171
## density              -1.248e+02  2.872e+01  -4.344 1.51e-05 ***
## sulphates             7.353e-01  1.782e-01   4.127 3.90e-05 ***
## alcohol               2.210e-01  3.820e-02   5.785 9.06e-09 ***
## typeWhite            -5.212e-01  1.274e-01  -4.090 4.59e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7515 on 1291 degrees of freedom
## Multiple R-squared:  0.3039, Adjusted R-squared:  0.2995
## F-statistic: 70.44 on 8 and 1291 DF,  p-value: < 2.2e-16
```

By this point, our multiple R-squared values are decreasing and we may wish to stop. Since there was little difference in models using all predictors and models without `citric.acid` and `pH`, we may wish to check if these two variables are colinear.

```
# Visualizations to look at relationships
# income is our response
varAnalysis <- wineTrainingData %>% select(citric.acid, pH, quality)
GGally::ggpairs(varAnalysis)
```



There appears to be a weak relationship between the two. Therefore, we may wish to consider a model that excludes `citric.acid` and `pH`.

Using the `caret` package

The `preProcess()` function in the `caret` package ensures that, when we center and scale our training data, the set's mean and standard deviation values are saved and applied to the test data to validate our predictions.

```
# Standardize all numeric columns,
# save mean and standard deviation from train set for
# future application to test set
preProcValues <- preProcess(wineTrainingData, method = c("center", "scale"))
trainTransformed <- predict(preProcValues, wineTrainingData)
```

Train our models

We can use the `trainControl()` function in the `caret` package to specify how we will fit our model to the training set. In this case, we specify ten-fold cross-validation, which splits our training set into ten unique partitions and uses nine as the "training" data. The tenth partition then behaves as the "test" data. This partitioning is repeated for all possible iterations and the errors are averaged to help estimate performance.

```
# Example of out-of-band trainControl use
trainControl(method = "cv", number = 10)
```

If we wish, we may also perform the steps above in one sequence.

```
# Set seed for repeatability
set.seed(11)

# Fit a linear model
```

```
fit1 <- train(quality ~ ., data = wineTrainingData,
              method = "lm",
              preProcess = c("center", "scale"),
              trControl = trainControl(method = "cv", number = 10))
```

The returned linear regression fit specifies a root mean squared error (RMSE) of ~0.755 on the training set.

```
# View RMSE
fit1
```

```
## Linear Regression
##
## 1300 samples
##   12 predictor
##
## Pre-processing: centered (12), scaled (12)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1170, 1169, 1170, 1170, 1170, 1170, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.7548357  0.2953867  0.5787664
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

We can fit four additional models for comparison.

```
set.seed(11)

# Fit a model with quadratics
fit2 <- train(quality ~ .^2, data = wineTrainingData,
              method = "lm",
              preProcess = c("center", "scale"),
              trControl = trainControl(method = "cv", number = 10))

# View RMSE
fit2
```

```
## Linear Regression
##
## 1300 samples
##   12 predictor
##
## Pre-processing: centered (78), scaled (78)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1169, 1170, 1170, 1171, 1170, 1170, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.7694926  0.3047192  0.5812276
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
set.seed(11)

# Exclude citric.acid and pH as discussed in the planning section
fit3 <- train(quality ~ fixed.acidity + volatile.acidity + residual.sugar +
```

```
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + sulphates + alcohol + type, data = wineTrainingData,
             method = "lm",
             preProcess = c("center", "scale"),
             trControl = trainControl(method = "cv", number = 10))

# View RMSE
fit3
```

```
## Linear Regression
##
## 1300 samples
##   10 predictor
##
## Pre-processing: centered (10), scaled (10)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1169, 1170, 1170, 1171, 1170, 1170, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.7542811  0.2961162  0.5806917
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

We can look at all interactions by wine type.

```
set.seed(11)

# Consider interactions of all variables with type
fit4 <- train(quality ~ (fixed.acidity + volatile.acidity + residual.sugar +
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + sulphates + alcohol):type, data = wineTrainingData,
             method = "lm",
             preProcess = c("center", "scale"),
             trControl = trainControl(method = "cv", number = 10))

# View RMSE
fit4
```

```
## Linear Regression
##
## 1300 samples
##   10 predictor
##
## Pre-processing: centered (18), scaled (18)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1169, 1170, 1170, 1171, 1170, 1170, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.7529892  0.2992554  0.5776987
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
set.seed(11)
```

```
# Consider all possible combinations with type
fit5 <- train(quality ~ (fixed.acidity*volatile.acidity + residual.sugar*alcohol +
    chlorides + free.sulfur.dioxide*total.sulfur.dioxide +
    density + sulphates + type), data = wineTrainingData,
              method = "lm",
              preProcess = c("center", "scale"),
              trControl = trainControl(method = "cv", number = 10))

# View RMSE
fit5
```

```
## Linear Regression
##
## 1300 samples
##   10 predictor
##
## Pre-processing: centered (13), scaled (13)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1169, 1170, 1170, 1171, 1170, 1170, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.7414448  0.3184701  0.5749434
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

We can compare the results for all five models.

```
# Side-by-side comparison of results
data.frame(t(fit1$results), t(fit2$results), t(fit3$results), t(fit4$results), t(fit5$results))
```

```
##                    X1       X1.1       X1.2       X1.3       X1.4
## intercept  1.00000000 1.00000000 1.00000000 1.00000000 1.00000000
## RMSE       0.75483575 0.78628957 0.75599295 0.75536727 0.75103524
## Rsquared   0.29538672 0.30468316 0.29282642 0.29668559 0.30075840
## MAE        0.57876635 0.58312247 0.58223730 0.57509329 0.57622981
## RMSESD     0.05131182 0.11477904 0.06151863 0.03951855 0.06687901
## RsquaredSD 0.05896453 0.05854389 0.06302578 0.04932166 0.05377221
## MAESD      0.03258859 0.04638444 0.03876398 0.03405945 0.04100039
```

Now that we have fitted our models, we wish to do predictions. We can use the `predict()` function on the `fit` object that was returned by the `caret` package above and view useful metrics using the `postResample()` function.

```
set.seed(11)

# Do predictions using fit object
pred1 <- predict(fit1, newdata = wineTrainingData)
pred4 <- predict(fit4, newdata = wineTrainingData)

# View useful metrics on these predictions
postResample(pred1, obs = wineTrainingData$quality)
```

```
##      RMSE  Rsquared       MAE
## 0.7461747 0.3089458 0.5728856
```

```
postResample(pred4, obs = wineTrainingData$quality)
```

```
##      RMSE  Rsquared       MAE
## 0.7400542 0.3202360 0.5677883
```

Apply models to test set

Now, we can use our selected model to perform predictions on the test set.

```
# Read in CSV test file, subset data
wineTest <- tbl_df(read.csv(file = "./wineQualityTest.csv"))
```

```
set.seed(11)

# Apply final model to test set
pred1 <- predict(fit1, newdata = wineTest)
pred5 <- predict(fit5, newdata = wineTest)

postResample(pred1, obs = wineTest$quality)
```

```
##      RMSE  Rsquared       MAE
## 0.7335588 0.2860098 0.5719915
```

```
postResample(pred5, obs = wineTest$quality)
```

```
##      RMSE  Rsquared       MAE
## 0.7316239 0.2899658 0.5726234
```

The best model had an RMSE of 0.732. The best model was one which removed two variables with mild correlation (citric.acid and pH) after gauging their Akaike information criterion (AIC). The model also included all possible interactions between `fixed.acidity` and `volatile.acidity`, `residual.sugar and alcohol`, and `free.sulfur.dioxide` and `total.sulfur.dioxide`. These interactions were selected as an educated guess (acidities and sulfur dioxides are generally to one another and sugar is a key component of fermentation) and all five RMSE values were relatively similar, suggesting that our models do a relatively poor job of predicting quality.