individual_capstone

Ekundayo Azubuike

2024-12-02

Introduction

Case and Data Overview

Wine producers are interested in creating products that consumers rate highly. The quality of wine has implications for sales and ultimately revenue. The following analysis aims to predict the quality of wine given 11 features: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, and alcohol. For more information on the data set, please review the references at the end of this document.

Summary

The following steps were performed the the course of the analysis:

- environmental setup
- loading data
- data pre-processing and exploratory analysis
- model construction and evaluation
  - base ("naive") model
  - generalized linear model
  - k-nearest neighbors model
  - random tree model
  - random forest model

Ultimately, a random forest model with overall accuracy of `0.6972` was selected.

Method/Analysis

Environment Setup

I first initialized the environment by downloading relevant packages for data analysis and machine learning tasks as follows:

```
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(Hmisc)) install.packages("Hmisc", repos = "http://cran.us.r-project.org")
if(!require(dslabs)) install.packages("dlabs", repos = "http://cran.us.r-project.org")
```

```
if(!require(GGally)) install.packages("GGally", repos = "http://cran.us.r-project.org")
if(!require(ggthemes)) install.packages("ggthemes", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
if(!require(moments)) install.packages("moments", repos = "http://cran.us.r-project.org")

ds_theme_set()

options(timeout = 120)
```

Download and Load Data

Next, I downloaded the relevant files and created the `data.frame` objects I would be working with. As the red and white wines were contained in separate `.csv` files, I merged them, and created a new predictor called `color`.

```
# download and load data set
dl <- "wine_quality.zip"
if(!file.exists(dl))
  download.file("https://archive.ics.uci.edu/static/public/186/wine+quality.zip", dl)

red_file <- "winequality-red.csv"
if(!file.exists(red_file))
  unzip(dl, red_file)

white_file <- "winequality-white.csv"
if(!file.exists(white_file))
  unzip(dl, white_file)

red.df <- read_delim(file = red_file,
                     delim = ";",
                     col_types = list(
                       quality = col_integer(),
                       .default = col_double()))
red.df$color <- "red"

white.df <- read_delim(file = white_file,
                       delim = ";",
                       col_types = list(
                         quality = col_integer(),
                         .default = col_double()))
white.df$color = "white"

names(red.df) <- make.names(colnames(red.df))
names(white.df) <- make.names(colnames(white.df))

wine.df <- rbind(red.df, white.df)
wine.df <- wine.df %>%
  mutate(color = as_factor(color))

wine.df %>% head()
```

```
## # A tibble: 6 x 13
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
##           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
## 1           7.4             0.7         0               1.9     0.076
## 2           7.8             0.88        0               2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.7         0               1.9     0.076
## 6           7.4             0.66        0               1.8     0.075
## # i 8 more variables: free.sulfur.dioxide <dbl>, total.sulfur.dioxide <dbl>,
## #   density <dbl>, pH <dbl>, sulphates <dbl>, alcohol <dbl>, quality <int>,
## #   color <fct>
```

Data Pre-processing and Exploratory Data Analysis

I split the data into training and testing sets using a 20% split. I scaled the data in order to make it more interpretable during exploratory data analysis.

```
wine.x <- wine.df %>%
  select(-quality)
wine.y <- wine.df %>%
  select(quality)

# data pre-processing
set.seed(2024)

test.index <- createDataPartition(wine.y$quality,
                                  p = 0.2,
                                  list = FALSE)
wine.x.train <- wine.x[-test.index,]
wine.x.test <- wine.x[test.index,]
wine.y.train <- wine.y[-test.index,]
wine.y.test <- wine.y[test.index,]

head(wine.x.train)
```

```
## # A tibble: 6 x 12
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
##           <dbl>            <dbl>       <dbl>          <dbl>     <dbl>
## 1           7.4             0.7         0               1.9     0.076
## 2           7.8             0.88        0               2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.7         0               1.9     0.076
## 6           7.4             0.66        0               1.8     0.075
## # i 7 more variables: free.sulfur.dioxide <dbl>, total.sulfur.dioxide <dbl>,
## #   density <dbl>, pH <dbl>, sulphates <dbl>, alcohol <dbl>, color <fct>
```

```
summary(wine.x.train)
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 4.200   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
```

```
##  1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.2500   1st Qu.: 1.800
##  Median : 7.000   Median :0.2900   Median :0.3100   Median : 3.000
##  Mean   : 7.227   Mean   :0.3388   Mean   :0.3208   Mean   : 5.452
##  3rd Qu.: 7.700   3rd Qu.:0.4000   3rd Qu.:0.3900   3rd Qu.: 8.100
##  Max.   :15.900   Max.   :1.5800   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.00900   Min.   :  1.00      Min.   :  6.0        Min.   :0.9871
##  1st Qu.:0.03800   1st Qu.: 17.00      1st Qu.: 77.0        1st Qu.:0.9923
##  Median :0.04700   Median : 29.00      Median :118.0        Median :0.9949
##  Mean   :0.05586   Mean   : 30.62      Mean   :115.8        Mean   :0.9947
##  3rd Qu.:0.06500   3rd Qu.: 41.00      3rd Qu.:156.0        3rd Qu.:0.9969
##  Max.   :0.61000   Max.   :146.50      Max.   :366.5        Max.   :1.0390
##        pH          sulphates         alcohol        color
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   red  :1277
##  1st Qu.:3.110   1st Qu.:0.4300   1st Qu.: 9.50   white:3919
##  Median :3.210   Median :0.5000   Median :10.30
##  Mean   :3.217   Mean   :0.5302   Mean   :10.49
##  3rd Qu.:3.320   3rd Qu.:0.6000   3rd Qu.:11.30
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90
```

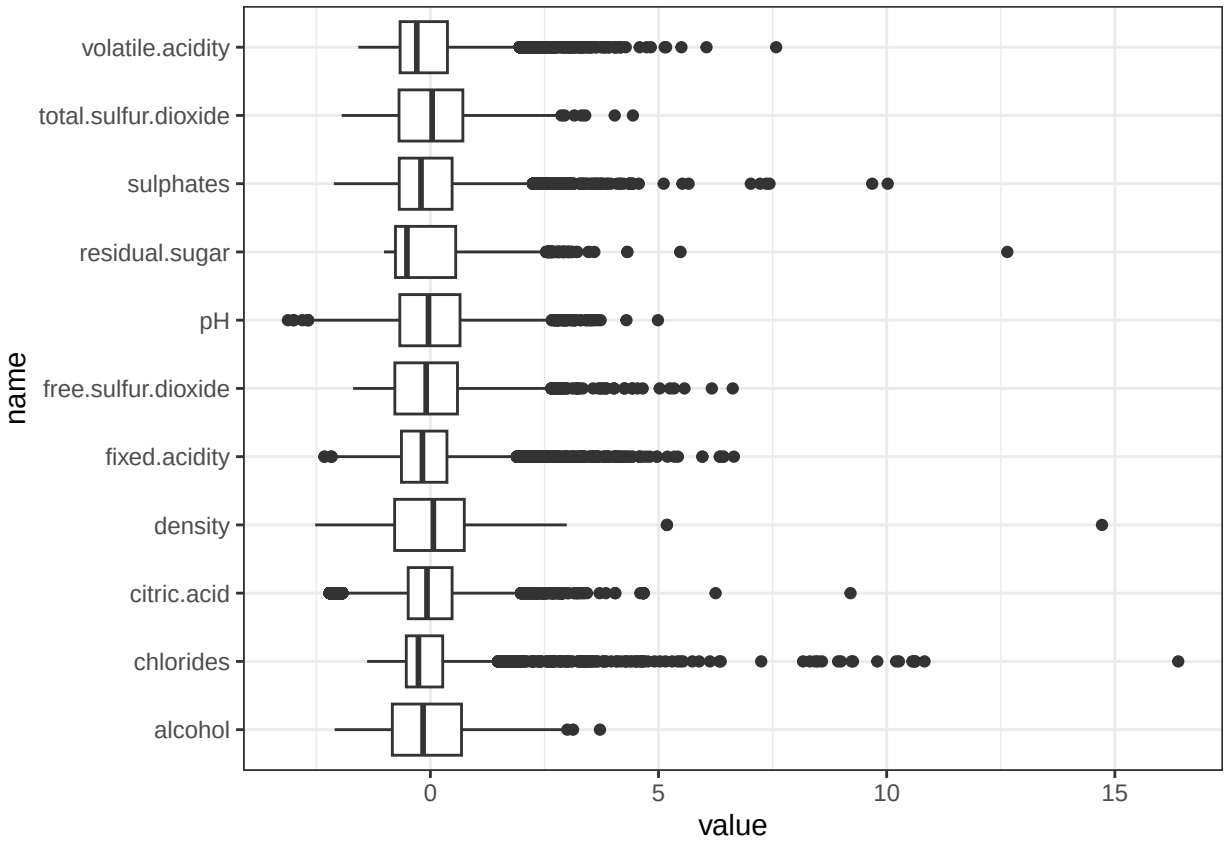
```
wine.x.train.scaled <- wine.x.train %>%
  mutate_if(is.numeric, function(x) { (x - mean(x)) / sd(x) })

wine.x.test.scaled <- wine.x.test %>%
  mutate_if(is.numeric, function(x) { (x - mean(x)) / sd(x) })
```

Distribution and Transformation of Predictors

Examination of the following box plot reveals that most of the variables have positive skew, with the six variables in the following table having the most significant skew.

```
# boxplots of variables
wine.x.train %>%
  select(-color) %>%
  mutate_all(scale) %>%
  pivot_longer(cols = names(wine.x.train)[-12]) %>%
  group_by(name) %>%
  ggplot(aes(x = name, y = value)) +
  geom_boxplot() +
  coord_flip()
```




```
# skew of variables
wine.x.train %>%
  select(-color) %>%
  pivot_longer(cols = names(wine.x.train)[-12]) %>%
  group_by(name) %>%
  summarise(skew = skewness(value)) %>%
  filter(skew > 1 | skew < -1)
```

```
## # A tibble: 5 x 2
##   name              skew
##   <chr>            <dbl>
## 1 chlorides         5.06
## 2 fixed.acidity     1.76
## 3 residual.sugar    1.51
## 4 sulphates         1.61
## 5 volatile.acidity  1.52
```

I performed a log transformation on the numeric predictors to remove skew in order to improve model performance down the line, as normalcy of predictors is preferred.

```
# log transformation of data to remove skewness
high.skew.ind <- c("chlorides", "fixed.acidity", "free.sulfur.dioxide",
                   "residual.sugar", "sulphates", "volatile.acidity")
low.skew.train <- wine.x.train[names(wine.x.train)[!names(wine.x.train) %in% high.skew.ind]]
low.skew.test <- wine.x.test[names(wine.x.test)[!names(wine.x.test) %in% high.skew.ind]]
```

```
wine.x.train %>%
  select(all_of(high.skew.ind)) %>%
  pivot_longer(cols = high.skew.ind) %>%
  mutate(value = ifelse(value == 0, 0, log(value))) %>%
  group_by(name) %>%
  summarise(skew = skewness(value))
```

```
## # A tibble: 6 x 2
##   name                  skew
##   <chr>                <dbl>
## 1 chlorides            0.815
## 2 fixed.acidity        0.918
## 3 free.sulfur.dioxide -0.852
## 4 residual.sugar       0.241
## 5 sulphates            0.357
## 6 volatile.acidity     0.338
```

```
# log transform data
wine.x.train.scaled <- sapply(wine.x.train[high.skew.ind], function(x){
  ifelse(x == 0, 0, log(x)) }) %>%
  cbind(low.skew.train) %>%
  mutate(color = factor(color)) %>%
  data.frame()

wine.x.test.scaled <- sapply(wine.x.test[high.skew.ind], function(x){
  ifelse(x == 0, 0, log(x)) }) %>%
  cbind(low.skew.test) %>%
  mutate(color = factor(color)) %>%
  data.frame()
```

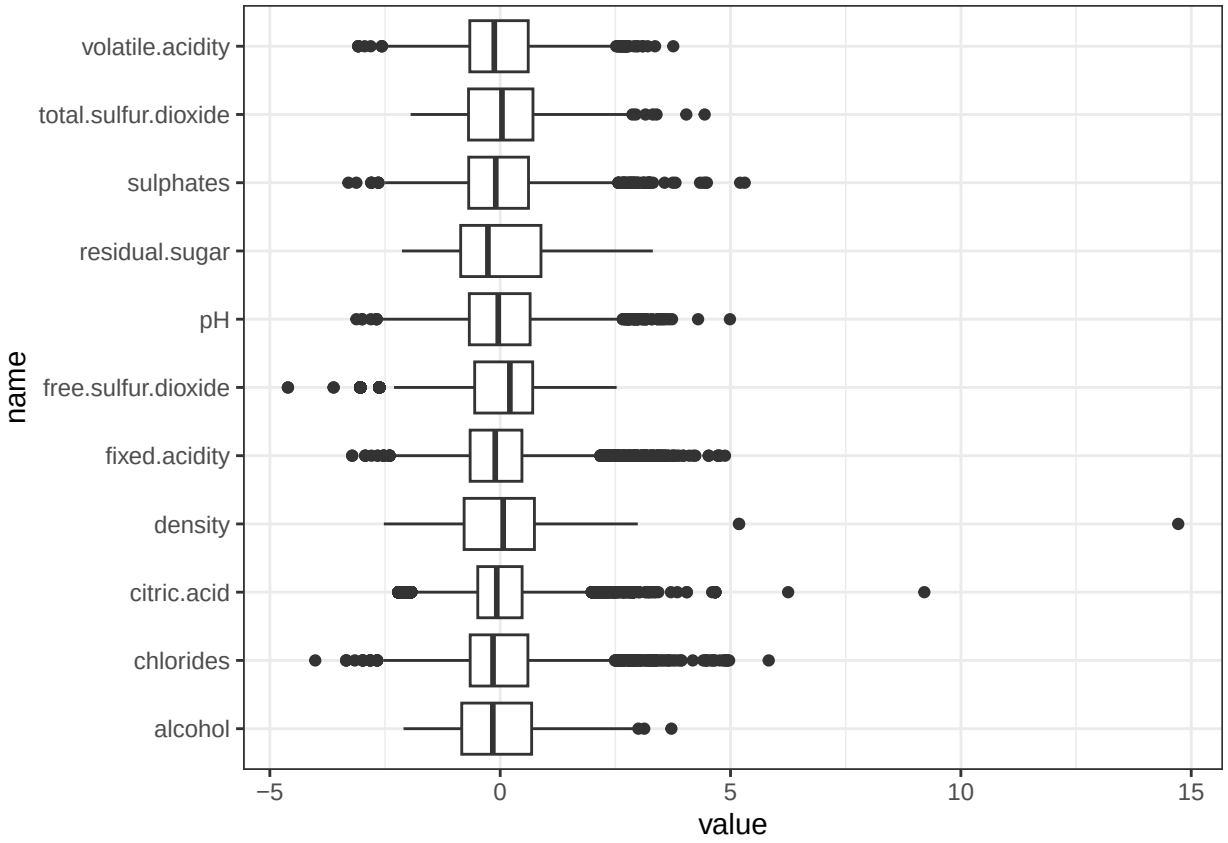
The skewness of all variables has improved.

```
# skew of log-transformed variables
wine.x.train.scaled %>%
  select(-color) %>%
  pivot_longer(cols = names(wine.x.train.scaled)[-12]) %>%
  group_by(name) %>%
  summarise(skew = skewness(value))
```

```
## # A tibble: 11 x 2
##    name                   skew
##    <chr>                 <dbl>
##  1 alcohol               0.573
##  2 chlorides             0.815
##  3 citric.acid           0.555
##  4 density               0.627
##  5 fixed.acidity         0.918
##  6 free.sulfur.dioxide  -0.852
##  7 pH                    0.339
##  8 residual.sugar        0.241
##  9 sulphates             0.357
## 10 total.sulfur.dioxide -0.0366
## 11 volatile.acidity      0.338
```

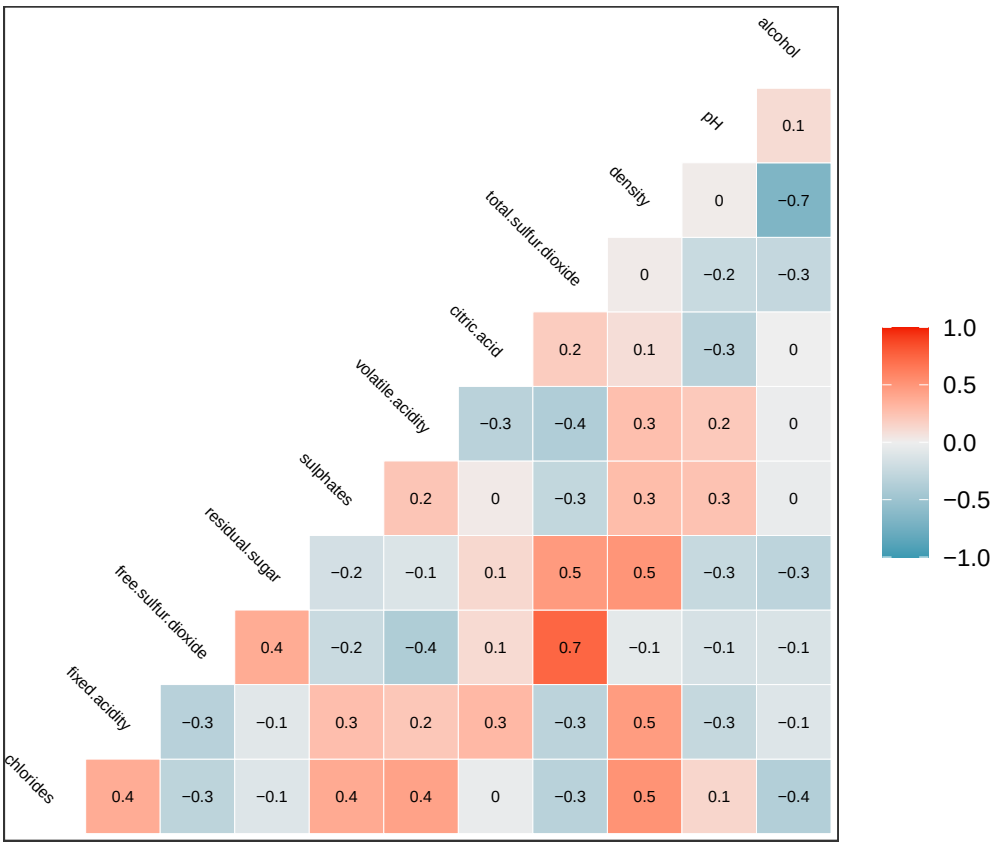
Examination of a box plot of scaled and log transformed predictors demonstrates that the distribution of values for each is normalized. Several outliers (± 3 standard deviations from mean $= 0$) exist for each variable. For now, I will carry on with analysis without addressing them.

```
# box plots of scaled and log-transformed variables
wine.x.train.scaled %>%
  select(-color) %>%
  mutate_all(scale) %>%
  pivot_longer(cols = names(wine.x.train.scaled)[-12]) %>%
  group_by(name) %>%
  ggplot(aes(x = name, y = value)) +
  geom_boxplot() +
  coord_flip()
```



Correlation and Variable Selection

A correlation matrix reveals that two set of variables are significantly correlated (Spearman's rank coefficient > 0.6 or < -0.6): `alcohol` and `density` (-0.7) as well as `sulfur.dioxide` and `free.sulfur.dioxide` (0.7).



In order to remove confounders from modeling techniques that will be sensitive to the correlations, I created a new variable `alcohol.density` that is a linear combination of `alcohol` and `density` in order to remove correlation but preserve the data from each column. Then, because `free.sulfur.dioxide` is the difference between `total.sulfur.dioxide` and bound sulfur dioxide, I removed the `free.sulfur.dioxide` variable to avoid redundant data.

```
wine.train <- tibble(cbind(wine.x.train.scaled, wine.y.train$quality)) %>%
  rename(quality = `wine.y.train$quality`)
wine.train %>% head()
```

```
## # A tibble: 6 x 13
##   chlorides fixed.acidity free.sulfur.dioxide residual.sugar sulphates
##       <dbl>         <dbl>                <dbl>          <dbl>     <dbl>
## 1     -2.58          2.00                 2.40          0.642    -0.580
## 2     -2.32          2.05                 3.22          0.956    -0.386
## 3     -2.39          2.05                 2.71          0.833    -0.431
## 4     -2.59          2.42                 2.83          0.642    -0.545
## 5     -2.58          2.00                 2.40          0.642    -0.580
## 6     -2.59          2.00                 2.56          0.588    -0.580
## # i 8 more variables: volatile.acidity <dbl>, citric.acid <dbl>,
## #   total.sulfur.dioxide <dbl>, density <dbl>, pH <dbl>, alcohol <dbl>,
## #   color <fct>, quality <int>
```

```
wine.train.selected <- wine.train %>%
  mutate(alcohol.density = alcohol * density) %>%
  select(-c(alcohol, density, free.sulfur.dioxide))
wine.train.selected %>% head()
```

```
## # A tibble: 6 x 11
##   chlorides fixed.acidity residual.sugar sulphates volatile.acidity citric.acid
##       <dbl>         <dbl>          <dbl>     <dbl>            <dbl>       <dbl>
## 1     -2.58          2.00          0.642    -0.580           -0.357        0
## 2     -2.32          2.05          0.956    -0.386           -0.128        0
## 3     -2.39          2.05          0.833    -0.431           -0.274        0.04
## 4     -2.59          2.42          0.642    -0.545           -1.27         0.56
## 5     -2.58          2.00          0.642    -0.580           -0.357        0
## 6     -2.59          2.00          0.588    -0.580           -0.416        0
## # i 5 more variables: total.sulfur.dioxide <dbl>, pH <dbl>, color <fct>,
## #   quality <int>, alcohol.density <dbl>
```

```
wine.test <- tibble(cbind(wine.x.test.scaled, wine.y.test$quality)) %>%
  rename(quality = `wine.y.test$quality`)
wine.test %>% head()
```

```
## # A tibble: 6 x 13
##   chlorides fixed.acidity free.sulfur.dioxide residual.sugar sulphates
##       <dbl>         <dbl>               <dbl>          <dbl>     <dbl>
## 1     -2.33          1.90                2.71          0.588    -0.616
## 2     -2.39          2.14                3.56          0.588    -0.288
## 3     -2.50          2.03                3.14          0.833    -0.431
## 4     -2.53          1.84                2.40          0.336    -0.580
## 5     -2.50          2.05                2.08          0.693    -0.528
## 6     -2.27          1.65                2.56          0.588    -0.598
## # i 8 more variables: volatile.acidity <dbl>, citric.acid <dbl>,
## #   total.sulfur.dioxide <dbl>, density <dbl>, pH <dbl>, alcohol <dbl>,
## #   color <fct>, quality <int>
```

```
wine.test.selected <-  wine.test %>%
  mutate(alcohol.density = alcohol * density) %>%
  select(-c(alcohol, density, free.sulfur.dioxide))
wine.test.selected %>% head()
```

```
## # A tibble: 6 x 11
##   chlorides fixed.acidity residual.sugar sulphates volatile.acidity citric.acid
##       <dbl>         <dbl>          <dbl>     <dbl>            <dbl>       <dbl>
## 1     -2.33          1.90          0.588    -0.616           -0.545        0.08
## 2     -2.39          2.14          0.588    -0.288           -1.27         0.56
## 3     -2.50          2.03          0.833    -0.431           -0.942        0.31
## 4     -2.53          1.84          0.336    -0.580           -0.942        0.16
## 5     -2.50          2.05          0.693    -0.528           -0.439        0
## 6     -2.27          1.65          0.588    -0.598           -1.14         0.25
## # i 5 more variables: total.sulfur.dioxide <dbl>, pH <dbl>, color <fct>,
## #   quality <int>, alcohol.density <dbl>
```

Results

Base Model

I constructed a base model that uses the mean (`mu`) as the prediction for the target variable `quality`. This approach yields a baseline RMSE of `0.8750728`.

```r
# base model
mu <- mean(as.numeric(wine.train$quality))

base.pred <- rep(mu,
                 times = length(wine.test$quality))
base.rmse <- RMSE(base.pred, as.numeric(wine.test$quality))
base.rmse %>% kable()
```

x
0.8750728

Linear Model

I then constructed a generalized linear model of `quality` as a function of the previously selected variables. I also performed a pre-processing step to center the data for analysis. This approach yielded an improved RMSE of `0.7323501` as compared to the base model.

```r
# linear model
wine.lm <- train(quality ~ .,
                 data = wine.train.selected,
                 preProcess = "center",
                 method = "glm")

lm.rmse <- RMSE(predict(wine.lm, wine.test.selected),
                wine.test.selected$quality)
lm.rmse %>% kable()
```

x
0.7323501

K-Nearest Neighbors Model

I next constructed a k-nearest neighbors classification model with 10-fold cross-validation, a scaling pre-process step, and a tuning grid comprised of a range of values for `k` between 5 and 25. The ideal value of `k` is `7`. Using this value, the model achieves an overall accuracy of `0.5327`, with the highest balanced accuracy for wines with a `quality` of 5: `0.7123`.

```r
# knn
wine.train <- wine.train %>%
  mutate(quality = factor(quality))

wine.test <- wine.test %>%
  mutate(quality = factor(quality))

quality.levels <- levels(wine.test$quality)
controls <- trainControl(method = "cv", p = 0.8, number = 10)
grid = data.frame(k = seq(5, 25, 2))
```

```
wine.knn <- train(quality ~ .,
                  method = "knn",
                  data = wine.train,
                  trControl = controls,
                  tuneGrid = grid)
wine.knn$bestTune %>% kable()
```

	k
2	7

```
knn.preds <- factor(predict(wine.knn, wine.test),
                    levels = levels(wine.test$quality))
knn.cm <- confusionMatrix(knn.preds, wine.test$quality)
knn.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
##          4   0   1   2   3   0   0   0
##          5   5  18 267 131  25   4   0
##          6   2  23 133 340 114   9   0
##          7   0   5  18  85  78  14   1
##          8   0   1   2   9   4   7   0
##          9   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5327
##                  95% CI : (0.5051, 0.5601)
##     No Information Rate : 0.4366
##     P-Value [Acc > NIR] : 2.18e-12
##
##                   Kappa : 0.2838
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3  Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.00000 0.0208333   0.6327   0.5986  0.35294  0.20588
## Specificity           1.00000 0.9960096   0.7918   0.6166  0.88611  0.98737
## Pos Pred Value            NaN 0.1666667   0.5933   0.5475  0.38806  0.30435
## Neg Pred Value        0.99462 0.9637066   0.8179   0.6647  0.87000  0.97887
## Prevalence            0.00538 0.0368947   0.3244   0.4366  0.16987  0.02613
## Detection Rate        0.00000 0.0007686   0.2052   0.2613  0.05995  0.00538
## Detection Prevalence  0.00000 0.0046118   0.3459   0.4773  0.15450  0.01768
## Balanced Accuracy     0.50000 0.5084215   0.7123   0.6076  0.61953  0.59663
##                       Class: 9
## Sensitivity          0.0000000
```

```
## Specificity          1.0000000
## Pos Pred Value             NaN
## Neg Pred Value       0.9992314
## Prevalence           0.0007686
## Detection Rate       0.0000000
## Detection Prevalence 0.0000000
## Balanced Accuracy    0.5000000
```

Random Tree Model

I next constructed a random tree and tested a range of values for `cp`. The selected value for `cp` was `0.004166667`. This model yielded an overall accuracy of `0.5519`, a decrease from the KNN model.

```
# Random Tree
wine.rt <- train(quality ~ .,
                 data = wine.train,
                 tuneGrid = data.frame(cp = seq(0.0, 0.1, len = 25)),
                 method = "rpart")

wine.rt$bestTune %>% kable()
```

	cp
2	0.0041667

```
rt.preds <- factor(predict(wine.rt, wine.test),
                   levels = levels(wine.test$quality))

rt.cm <- confusionMatrix(rt.preds, wine.test$quality)
rt.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
##          4   0   0   0   0   0   0   0
##          5   4  23 265 121  13   2   0
##          6   2  24 157 421 176  24   0
##          7   1   1   0  26  32   8   1
##          8   0   0   0   0   0   0   0
##          9   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.5519
##                  95% CI : (0.5244, 0.5791)
##     No Information Rate : 0.4366
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2707
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.00000  0.00000   0.6280   0.7412  0.14480  0.00000
## Specificity           1.00000  1.00000   0.8146   0.4775  0.96574  1.00000
## Pos Pred Value            NaN      NaN   0.6192   0.5236  0.46377      NaN
## Neg Pred Value        0.99462  0.96311   0.8202   0.7042  0.84659  0.97387
## Prevalence            0.00538  0.03689   0.3244   0.4366  0.16987  0.02613
## Detection Rate        0.00000  0.00000   0.2037   0.3236  0.02460  0.00000
## Detection Prevalence  0.00000  0.00000   0.3290   0.6180  0.05304  0.00000
## Balanced Accuracy     0.50000  0.50000   0.7213   0.6093  0.55527  0.50000
##                      Class: 9
## Sensitivity          0.0000000
## Specificity          1.0000000
## Pos Pred Value             NaN
## Neg Pred Value       0.9992314
## Prevalence           0.0007686
## Detection Rate       0.0000000
## Detection Prevalence 0.0000000
## Balanced Accuracy    0.5000000
```

Random Forest

Lastly, I constructed a random forest model of 150 trees with 5-fold cross validation, a range of minimum node sizes between 3 and 50, and 3 variables randomly sampled as candidates at each split. This model yielded the highest accuracy measure: `0.6972`. The ideal hyperparameter values are `predFixed = 3` and `minNode = 3`.

```
# Random Forest
controls <- trainControl(method="cv", p = 0.8, number = 5)

wine.rf <- train(quality ~ .,
                 data = wine.train,
                 method = "Rborist",
                 trControl = controls,
                 tuneGrid = data.frame(predFixed = 3,
                                       minNode = c(3, 50)),
                 preProcess = "scale",
                 nTree = 150)

summary(wine.rf)
```

```
##             Length Class      Mode
## sampler      6     Sampler    list
## leaf         2     Leaf       list
## forest       5     Forest     list
## predMap     12     -none-     numeric
## signature    5     Signature  list
## training     6     -none-     list
## prediction   4     PredictCtg list
## validation   3     ValidCtg   list
```

```
## xNames      12     -none-     character
## problemType  1     -none-     character
## tuneValue    2     data.frame list
## obsLevels    7     -none-     character
## param        1     -none-     list
```

```
wine.rf$bestTune
```

```
##   predFixed minNode
## 1         3       3
```

```
rf.preds <- factor(predict(wine.rf, wine.test),
                   levels = levels(wine.test$quality))

rf.cm <- confusionMatrix(rf.preds,
                         wine.test$quality)
rf.cm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   3   4   5   6   7   8   9
##          3   0   0   0   0   0   0   0
##          4   0   5   0   0   0   0   0
##          5   3  24 327  99   5   1   0
##          6   4  19  93 432  86   8   1
##          7   0   0   2  37 129  11   0
##          8   0   0   0   0   1  14   0
##          9   0   0   0   0   0   0   0
##
## Overall Statistics
##
##                Accuracy : 0.6972
##                  95% CI : (0.6714, 0.722)
##     No Information Rate : 0.4366
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5312
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity           0.00000 0.104167   0.7749   0.7606  0.58371  0.41176
## Specificity           1.00000 1.000000   0.8498   0.7121  0.95370  0.99921
## Pos Pred Value            NaN 1.000000   0.7124   0.6719  0.72067  0.93333
## Neg Pred Value        0.99462 0.966821   0.8872   0.7933  0.91800  0.98445
## Prevalence            0.00538 0.036895   0.3244   0.4366  0.16987  0.02613
## Detection Rate        0.00000 0.003843   0.2513   0.3321  0.09915  0.01076
## Detection Prevalence  0.00000 0.003843   0.3528   0.4942  0.13759  0.01153
## Balanced Accuracy     0.50000 0.552083   0.8124   0.7364  0.76871  0.70549
##                      Class: 9
```

```
## Sensitivity          0.0000000
## Specificity          1.0000000
## Pos Pred Value             NaN
## Neg Pred Value       0.9992314
## Prevalence           0.0007686
## Detection Rate       0.0000000
## Detection Prevalence 0.0000000
## Balanced Accuracy    0.5000000
```

Conclusion

Summary

After data pre-processing, variable selection, and hyperparameter tuning, the best-performing model was the random forest with an overall accuracy of `0.6972`.

Limitations

In future investigations, I would like to explore the impact of removing outliers from the data set on predictive power of the models. Moreover, there are other machine learning algorithms that may provide even greater accuracy. The authors of the referenced paper decided on a support vector machine (SVM) model. I would also be interested in understanding how well a boosted tree would perform for this analysis.

References

- Wine Quality Data
- Modeling wine preferences by data mining from physicochemical properties
  - By P. Cortez, A. Cerdeira, Fernando Almeida, Telmo Matos, J. Reis. 2009 (Published in Decision Support Systems)