Wine Quality

Piri Yebra

2023-03-11

Executive summary

The purpose of this project is to predict the quality of wine based on eleven measurements. The data set was taken from a University of California at Irvine Machine Learning Repository (https://archive.ics.uci.edu/ml/index.php).

I used several machine learning algorithms. The code was written in R, using the Caret Package, that conveniently includes many machine learning algorithms using a standardized nomenclature. RMSE was chosen as the metric to evaluate the models. I also kept track of the model's Accuracy and used some graphs to better understand the models' performance.

Among the used algorithms, GLM (Generalized Linear Model) had the best performance, in both metrics.

Preparation

To start the project, libraries are installed and loaded. In this project, caret package is used.

```
# Install all needed libraries if not present
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(caret)) install.packages("caret")
if(!require(gam)) install.packages("gam")

# Loading all needed libraries
library(tidyverse)
library(caret)
library(gam)
```

I chose a wine quality data set available at https://archive.ics.uci.edu/ml/datasets/Wine.

```
# Retriving the Wine Quality dataset previously downloaded
setwd("C:/Users/Piri Yebra/projects/ejemplo/final_project2")
filename <- "winequality-red.csv"
dat = read.csv(filename)
```

Data set is analyzed

```
# Analizing dataset
str(dat)
```

```
## 'data.frame':    1599 obs. of  12 variables:
##  $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
##  $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
##  $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
##  $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
##  $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
##  $ free.sulfur.dioxide : int  11 25 15 17 11 13 15 15 9 17 ...
##  $ total.sulfur.dioxide: int  34 67 54 60 34 40 59 21 18 102 ...
##  $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
##  $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
##  $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
##  $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
##  $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

Data set has 12 variables. The first 11 are wine features. The last one, quality, should be the result of the previous features. Quality is in integer number.

I divided the original data set in train and test set. Train set will be used to develop the prediction algorithm, and test set to predict quality.

```
# Dividing the original data set in train / test sets
set.seed(42, sample.kind = "Rounding")
test_index <- createDataPartition(dat$quality,times=1, p=.1,  list = FALSE)
wine_test <- dat[test_index, ]
wine_train <- dat[-test_index, ]
```

Before trying machine learning models, I decided to use a metric to evaluate them. In this case, Root Mean Square Error (RMSE). The model that minimizes RMSE will be the best one.

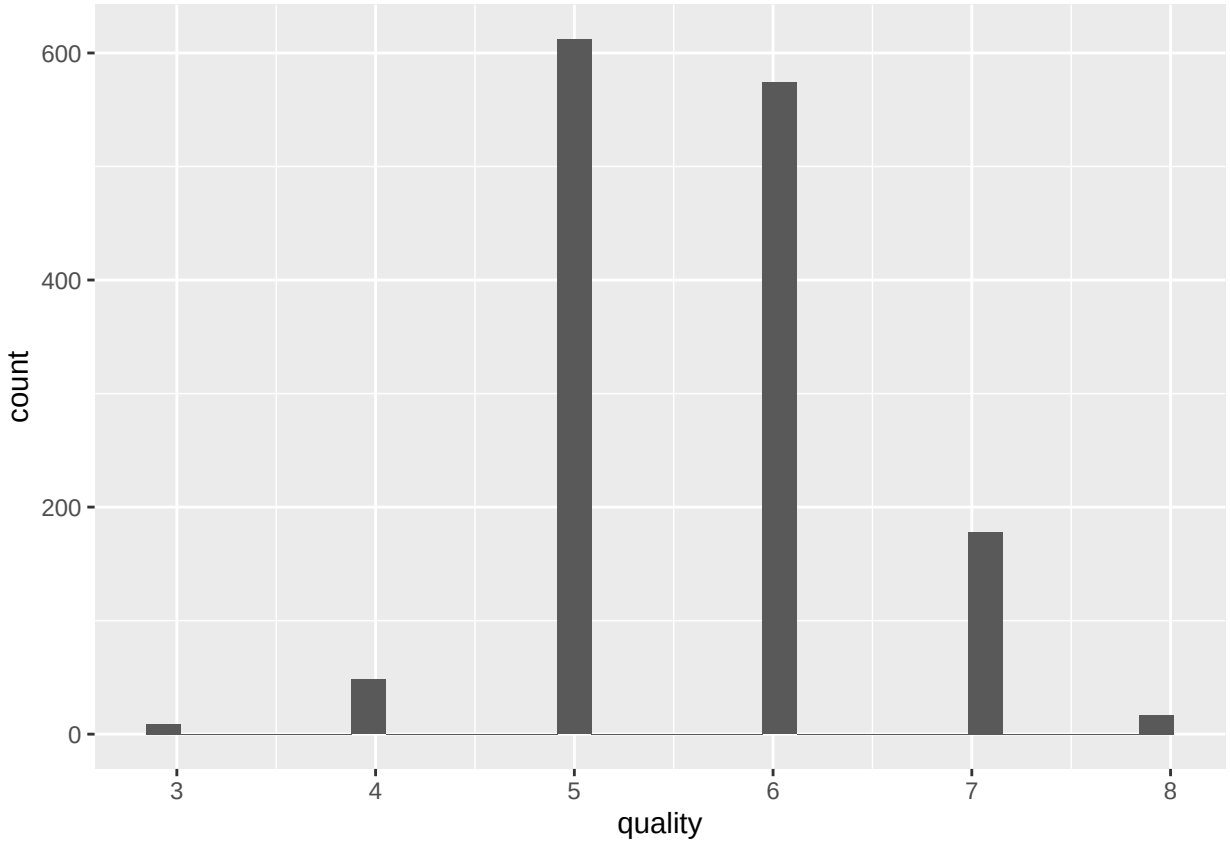
```
#RMSE function
RMSE <- function(true_ratings = NULL, predicted_ratings = NULL) {
  sqrt(mean((true_ratings - predicted_ratings)^2))
}
```

In addition, I decided to keep track of the model's accuracy, as a reference. Accuracy is calculated as part of the Confusion Matrix.

To have a visual understanding of the data, the following graphs were used:

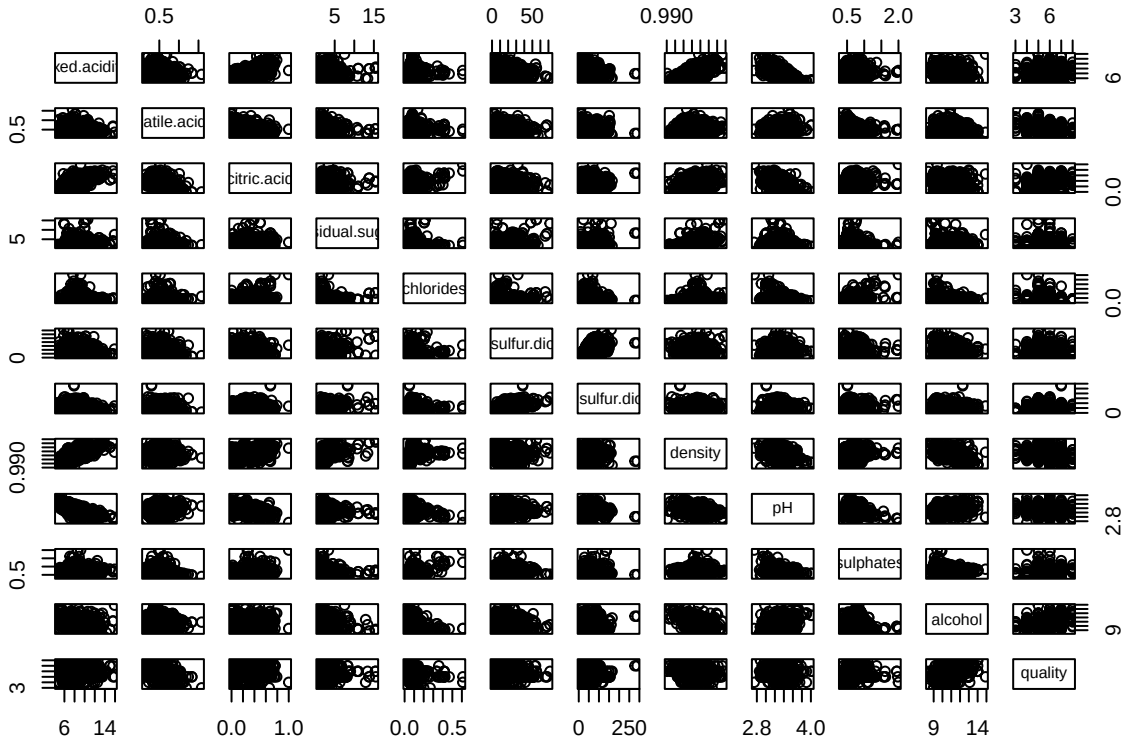
First, an histogram of quality distribution

```
wine_train %>% ggplot(aes(x = quality)) + geom_histogram()
```



Almost all records within the train set have a quality of 5 or 6. In this set of data, quality goes from 3 to 8. Then, I decided to plot all variables against each other.

```
plot(wine_train)
```



This graph shows that some variables are related. For example, density seems to be positively related to fixed acidity.And PH negatively related to fixed acidity.

The last row plots all variables against quality. No variable alone is clearly determining quality. In fact, some variables seem to have no effect in quality, since plots look like ovals, with no clear change in quality when the variable changes. For example, PH or fixed acidity. Neither of these variables alone seem to have an effect on quality.

Machine learning models

As previously said, in this project I used the caret package. I used the train function with different methods. Then, the predict function to test the model in the test set. With this result, I calculated the Confusion Matrix, which provides several metrics such as: a Summary of the prediction vs reference quality. I used this table to build a model comparison graph at the end of the analysis. Other important metrics are accuracy, which I also compared among models. Sensitivity and Specificity is also automatically calculated.

1. GLM - all variables

The fist model to try is Generalized Linear Model (GLM) by using all 11 variables in the data set.

```
# Generalized Linear Model - GLM algorith
train <- train(quality ~ ., method= "glm", data = wine_train)
y_hat <- round(predict(train, wine_test))
confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
##          4  0  0  0  0  0  0
##          5  1  3 49 19  0  0
##          6  0  2 20 40 17  0
##          7  0  0  0  5  4  1
##          8  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5776
##                  95% CI : (0.4974, 0.655)
##     No Information Rate : 0.4286
##     P-Value [Acc > NIR] : 0.0001001
##
##                   Kappa : 0.3021
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000  0.00000   0.7101   0.6250  0.19048 0.000000
## Specificity          1.000000  1.00000   0.7500   0.5979  0.95714 1.000000
## Pos Pred Value            NaN      NaN   0.6806   0.5063  0.40000      NaN
## Neg Pred Value       0.993789  0.96894   0.7753   0.7073  0.88742 0.993789
## Prevalence           0.006211  0.03106   0.4286   0.3975  0.13043 0.006211
## Detection Rate       0.000000  0.00000   0.3043   0.2484  0.02484 0.000000
## Detection Prevalence 0.000000  0.00000   0.4472   0.4907  0.06211 0.000000
## Balanced Accuracy    0.500000  0.50000   0.7301   0.6115  0.57381 0.500000
```

```
accuracy <- confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$overall[
RMSE <- RMSE(wine_test$quality, y_hat)
results <- data.frame(model="GLM", RMSE=RMSE, Accuracy=accuracy)
# Keep track of a table of reference vs prediction to later on build a comparative graph of all models
t <- as.data.frame(confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$ta
graphs <- data.frame(model="GLM", graph = t)
results
```

```
##   model      RMSE  Accuracy
## 1   GLM 0.6915641 0.5776398
```

In the results data frame I kept track of RMSE and Accuracy of all models

2. GLM - 5 wine variables

When I looked at the plot between the 11 variables and quality, I noticed that several variables seemed to have no relationship at all with quality since graphs looked like ovals. Therefore I decided to use GLM model with just 5 variables which visual relationship seemed somehow stronger: volatile acidity, chlorides, free sulfer dioxide, total sulfur dioxide and alhocol.

```r
# Generalized Linear Model - GLM algorith - just 5 variables
train <- train(quality ~ volatile.acidity + chlorides + free.sulfur.dioxide + total.sulfur.dioxide + al
y_hat <- round(predict(train, wine_test))
confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
##          4  0  0  0  0  0  0
##          5  1  3 49 21  1  0
##          6  0  2 20 39 16  0
##          7  0  0  0  4  4  1
##          8  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5714
##                  95% CI : (0.4912, 0.649)
##     No Information Rate : 0.4286
##     P-Value [Acc > NIR] : 0.000185
##
##                   Kappa : 0.2892
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000  0.00000   0.7101   0.6094  0.19048 0.000000
## Specificity          1.000000  1.00000   0.7174   0.6082  0.96429 1.000000
## Pos Pred Value            NaN      NaN   0.6533   0.5065  0.44444      NaN
## Neg Pred Value       0.993789  0.96894   0.7674   0.7024  0.88816 0.993789
## Prevalence           0.006211  0.03106   0.4286   0.3975  0.13043 0.006211
## Detection Rate       0.000000  0.00000   0.3043   0.2422  0.02484 0.000000
## Detection Prevalence 0.000000  0.00000   0.4658   0.4783  0.05590 0.000000
## Balanced Accuracy    0.500000  0.50000   0.7138   0.6088  0.57738 0.500000
```

```r
accuracy <- confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$overall[
RMSE <- RMSE(wine_test$quality, y_hat)
results <- results %>% add_row(model="GLM 5 vars", RMSE=RMSE, Accuracy=accuracy)
t <- as.data.frame(confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$ta
graphs <- graphs %>% add_row(model = "GLM 5 vars", graph.Prediction = t$Prediction, graph.Reference = t$
results
```

```
##        model      RMSE  Accuracy
## 1        GLM 0.6915641 0.5776398
## 2 GLM 5 vars 0.7092994 0.5714286
```

However, this model wasn't better than GLM using all variables.

3. KNN - No tunning

Next, I tried K-Nearest Neighbor Algorithm (KNN) with no tuning parameters.

```
# KNN algorithm - default
train <- train(quality ~ ., method = "knn", data=wine_train)
y_hat <- round(predict(train, wine_test))
confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
##          4  0  0  0  0  0  0
##          5  0  2 41 22  1  0
##          6  1  3 28 42 18  1
##          7  0  0  0  0  2  0
##          8  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.528
##                  95% CI : (0.4478, 0.607)
##     No Information Rate : 0.4286
##     P-Value [Acc > NIR] : 0.007028
##
##                   Kappa : 0.2041
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000  0.00000   0.5942   0.6562  0.09524 0.000000
## Specificity          1.000000  1.00000   0.7283   0.4742  1.00000 1.000000
## Pos Pred Value            NaN      NaN   0.6212   0.4516  1.00000      NaN
## Neg Pred Value       0.993789  0.96894   0.7053   0.6765  0.88050 0.993789
## Prevalence           0.006211  0.03106   0.4286   0.3975  0.13043 0.006211
## Detection Rate       0.000000  0.00000   0.2547   0.2609  0.01242 0.000000
## Detection Prevalence 0.000000  0.00000   0.4099   0.5776  0.01242 0.000000
## Balanced Accuracy    0.500000  0.50000   0.6612   0.5652  0.54762 0.500000
```

```
accuracy <- confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$overall[
RMSE <- RMSE(wine_test$quality, y_hat)
results <- results %>% add_row(model="KNN", RMSE=RMSE, Accuracy=accuracy)
t <- as.data.frame(confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$ta
graphs <- graphs %>% add_row(model = "KNN", graph.Prediction = t$Prediction, graph.Reference = t$Referer
results
```

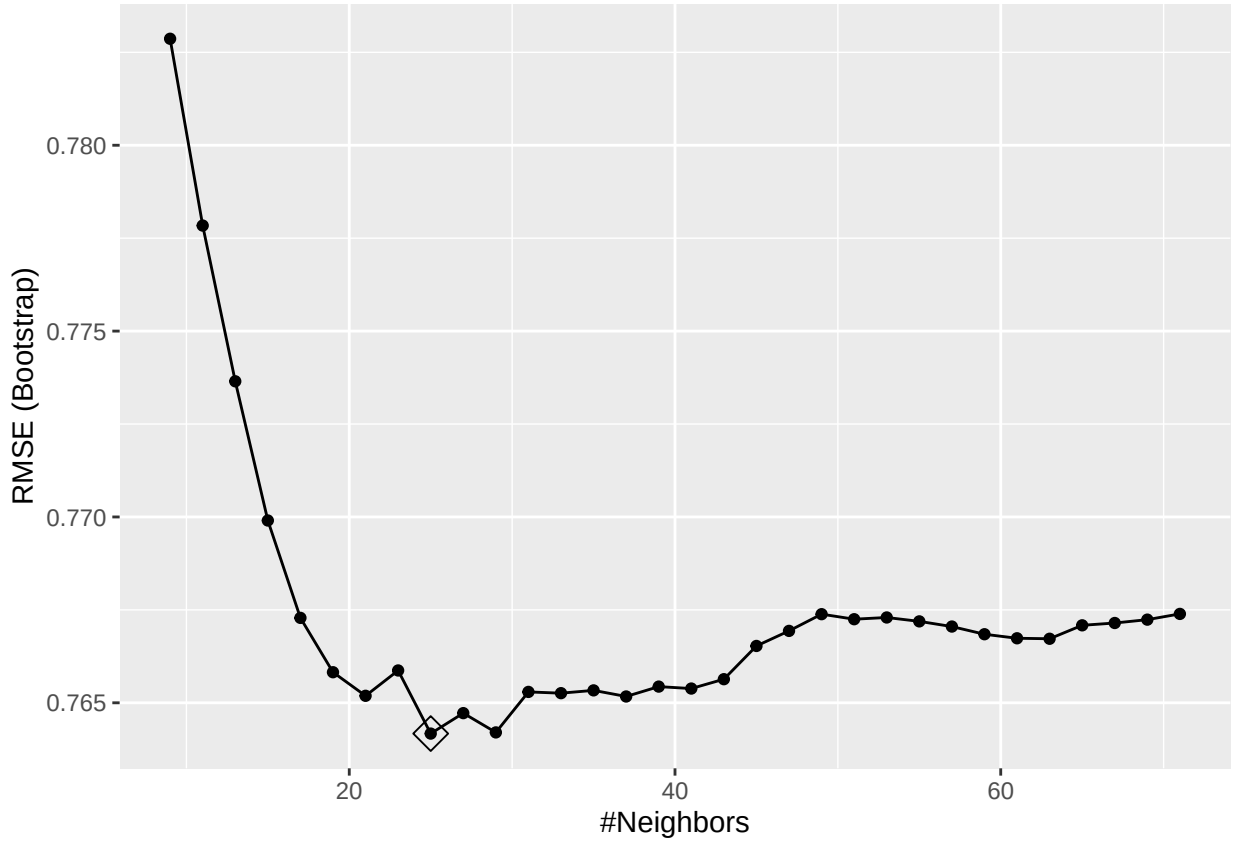
```
##        model      RMSE  Accuracy
## 1        GLM 0.6915641 0.5776398
## 2 GLM 5 vars 0.7092994 0.5714286
## 3        KNN 0.7841600 0.5279503
```

KNN have a worse RMSE than GLM

4. KNN - tuned

Next model was KNN but with tuning parameters.

```
# KNN algorithm - tuned
train <- train(quality ~ ., method = "knn", data=wine_train, tuneGrid = data.frame(k=seq(9, 71,2)))
ggplot(train, highlight = TRUE)
```




```
train$bestTune
```

```
##    k
## 9 25
```

```
y_hat <- round(predict(train, wine_test))
confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
```

```
##          4  0  0  0  0  0  0
##          5  0  1 43 21  0  0
##          6  1  4 26 43 21  1
##          7  0  0  0  0  0  0
##          8  0  0  0  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.5342
##                  95% CI : (0.454, 0.613)
##     No Information Rate : 0.4286
##     P-Value [Acc > NIR] : 0.00448
##
##                   Kappa : 0.2104
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000  0.00000   0.6232   0.6719   0.0000 0.000000
## Specificity          1.000000  1.00000   0.7609   0.4536   1.0000 1.000000
## Pos Pred Value            NaN      NaN   0.6615   0.4479      NaN      NaN
## Neg Pred Value       0.993789  0.96894   0.7292   0.6769   0.8696 0.993789
## Prevalence           0.006211  0.03106   0.4286   0.3975   0.1304 0.006211
## Detection Rate       0.000000  0.00000   0.2671   0.2671   0.0000 0.000000
## Detection Prevalence 0.000000  0.00000   0.4037   0.5963   0.0000 0.000000
## Balanced Accuracy    0.500000  0.50000   0.6920   0.5627   0.5000 0.500000
```

```
accuracy <- confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$overall[
RMSE <- RMSE(wine_test$quality, y_hat)
results <- results %>% add_row(model="KNN_T", RMSE=RMSE, Accuracy=accuracy)
t <- as.data.frame(confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$ta
graphs <- graphs %>% add_row(model = "KNN_T", graph.Prediction = t$Prediction, graph.Reference = t$Refe
results
```

```
##        model      RMSE  Accuracy
## 1        GLM 0.6915641 0.5776398
## 2 GLM 5 vars 0.7092994 0.5714286
## 3        KNN 0.7841600 0.5279503
## 4      KNN_T 0.7801895 0.5341615
```

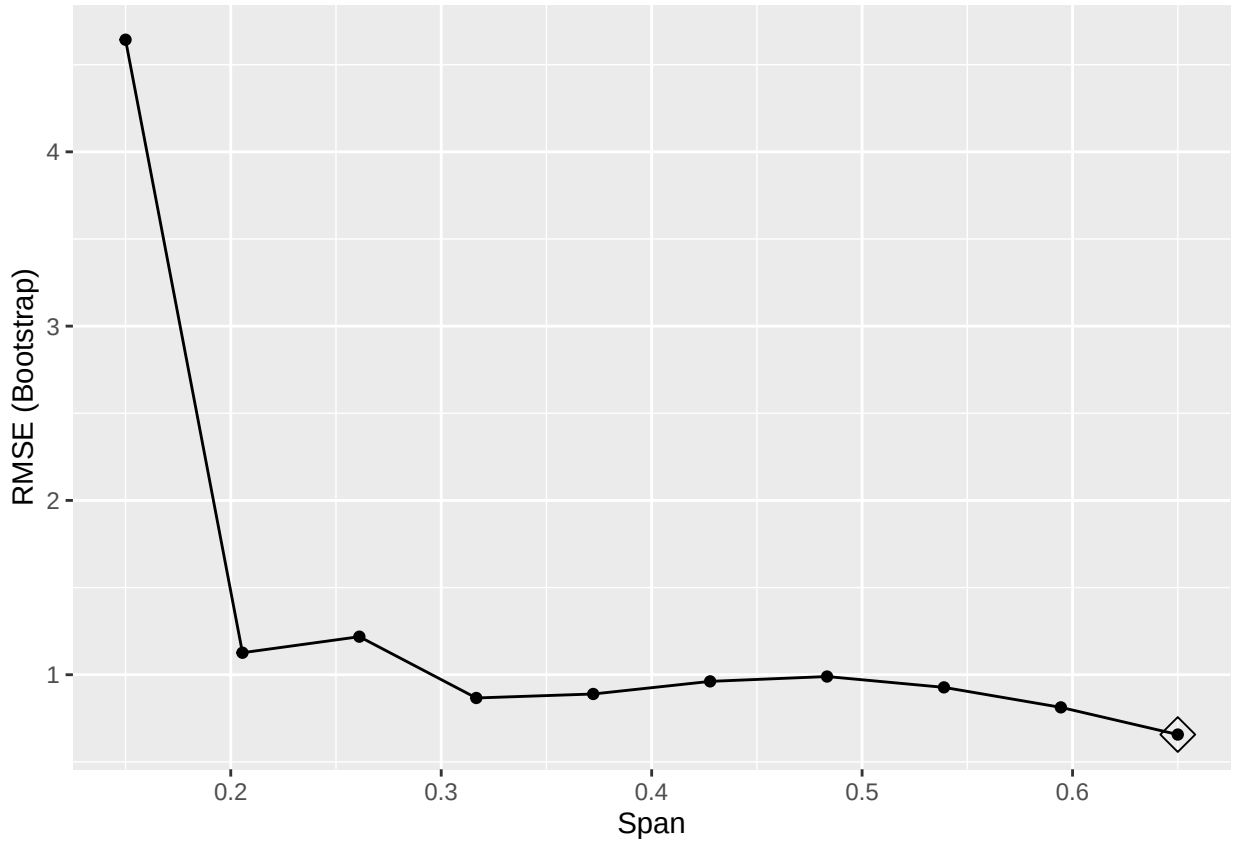
As expected, when using tuning paramaters, KNN algorithm was better than KK without tuning parameters. But no better than GLM.

5. Loess

The last model used was Loess.

```
# Loess algorithm
grid <- expand.grid(span = seq(0.15, 0.65, len = 10), degree = 1)
train <- train(quality ~ .,
```

```
                      method = "gamLoess",
                      tuneGrid=grid,
                      data = wine_train)
ggplot(train, highlight = TRUE)
```




```
train$bestTune
```

```
##    span degree
## 10 0.65      1
```

```
y_hat <- round(predict(train, wine_test))
confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  3  4  5  6  7  8
##          3  0  0  0  0  0  0
##          4  0  0  0  0  0  0
##          5  1  3 44 17  0  0
##          6  0  2 25 42 15  0
##          7  0  0  0  5  6  1
##          8  0  0  0  0  0  0
##
```

```
## Overall Statistics
##
##                Accuracy : 0.5714
##                  95% CI : (0.4912, 0.649)
##     No Information Rate : 0.4286
##     P-Value [Acc > NIR] : 0.000185
##
##                   Kappa : 0.2973
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 3 Class: 4 Class: 5 Class: 6 Class: 7 Class: 8
## Sensitivity          0.000000  0.00000   0.6377   0.6562  0.28571 0.000000
## Specificity          1.000000  1.00000   0.7717   0.5670  0.95714 1.000000
## Pos Pred Value            NaN      NaN   0.6769   0.5000  0.50000      NaN
## Neg Pred Value       0.993789  0.96894   0.7396   0.7143  0.89933 0.993789
## Prevalence           0.006211  0.03106   0.4286   0.3975  0.13043 0.006211
## Detection Rate       0.000000  0.00000   0.2733   0.2609  0.03727 0.000000
## Detection Prevalence 0.000000  0.00000   0.4037   0.5217  0.07453 0.000000
## Balanced Accuracy    0.500000  0.50000   0.7047   0.6116  0.62143 0.500000
```

```
accuracy <- confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$overall[
RMSE <- RMSE(wine_test$quality, y_hat)
results <- results %>% add_row(model="Loess", RMSE=RMSE, Accuracy=accuracy)
t <- as.data.frame(confusionMatrix(data = as.factor(y_hat), reference = as.factor(wine_test$quality))$ta
graphs <- graphs %>% add_row(model = "Loess", graph.Prediction = t$Prediction, graph.Reference = t$Refe
results
```

```
##        model      RMSE  Accuracy
## 1        GLM 0.6915641 0.5776398
## 2 GLM 5 vars 0.7092994 0.5714286
## 3        KNN 0.7841600 0.5279503
## 4      KNN_T 0.7801895 0.5341615
## 5      Loess 0.6960403 0.5714286
```

Loess model performed better than KNN, but worse than GLM.

Results

After trying 3 different machine learning algorithms plus two slight variations, the results were the following:

```
#Analysis
results
```

```
##        model      RMSE  Accuracy
## 1        GLM 0.6915641 0.5776398
## 2 GLM 5 vars 0.7092994 0.5714286
## 3        KNN 0.7841600 0.5279503
## 4      KNN_T 0.7801895 0.5341615
## 5      Loess 0.6960403 0.5714286
```

The model with the lowest RMSE and therefore the one selected for predicting wine quality is:

```
results$model[which.min(results$RMSE)]
```

```
## [1] "GLM"
```

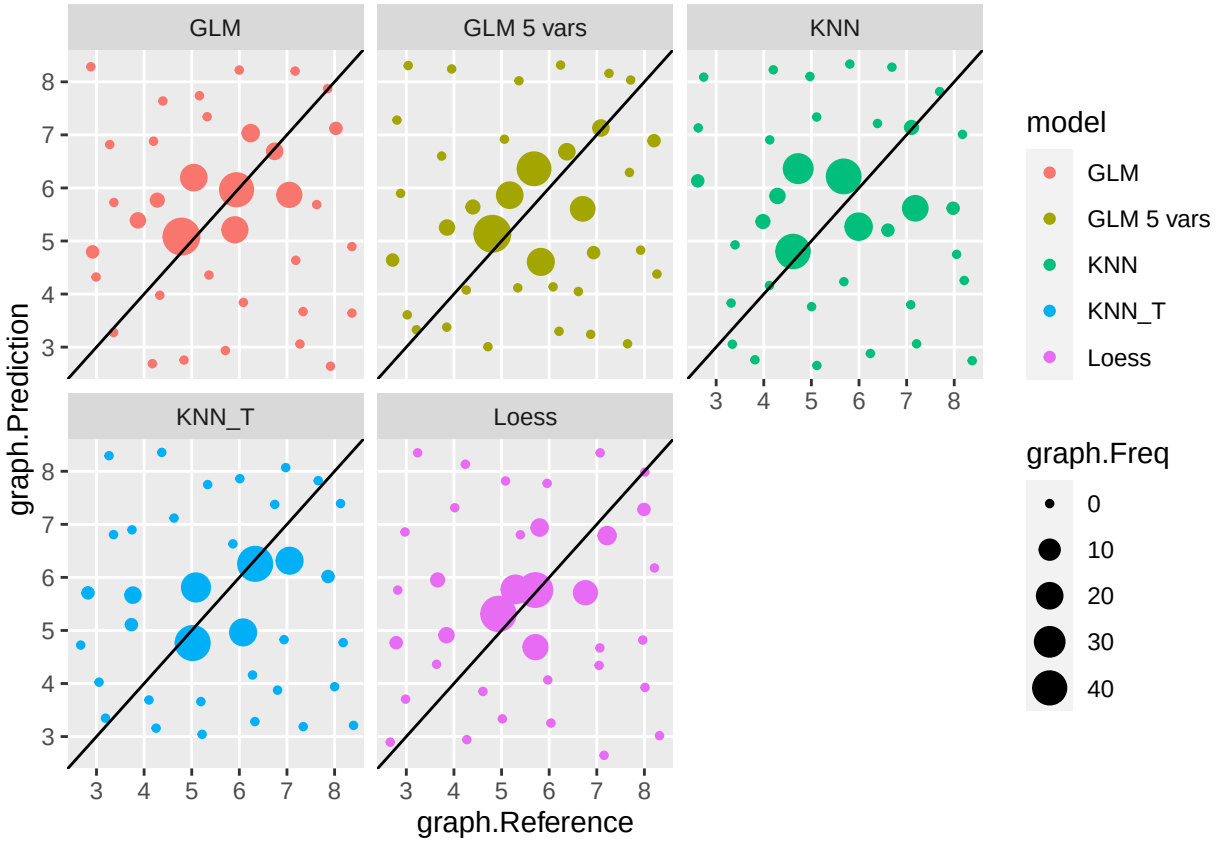
As stated earlier, I kept track of the model's accuracy. The model that performed best in accuracy is:

```
results$model[which.max(results$Accuracy)]
```

```
## [1] "GLM"
```

Finally, to have a visual reference on how the models performed, I plotted Reference vs Prediction. The size of the dots mean the times of such prediction. A perfect model would predict quality equals to reference quality, which will show all dots over the line.

```
graphs %>% ggplot(aes(x=graph.Reference, y=graph.Prediction, size=graph.Freq, color=model)) + geom_jitt
```



Analyzing these graphs, models don't seem to perform very different. However, GLM shows more frequent predictions near the line than the other models. 5 models have some predictions far from the reference.

Conclusion

In this project, I had the opportunity to practice what has been taught in this course series. I used a wine data set provided in https://archive.ics.uci.edu/ml/datasets/Wine. I created a train set to try different algorithms and then make predictions on a test set.

I used RMSE to quantify the best algorithm. The best model would be the minimum RMSE. But I also wanted to take Accuracy into account for reference. Since wine quality is an integer, the predictions had to be as well integers. I decided to round the predictions in order to have an integer as a result.

The algorithms used were: GLM, KNN, and Loess. For GLM, first I used all variables in the data set. Then I used just 5 variables that by looking at the plot, seemed to have more effect on quality. The model with all variables had slightly better results than the one that used just 5 variables. For KNN, I also tried two variations. The first with no parameters in the caret package. In the second KNN model, I used the parameters to tune the model. The second KNN model had better results than the first. Last, I used the Loess algorithm. This model had better results than KNN, but not as good as GLM.

GLM using all 11 variables had the least RMSE. Therefore, this is the best model among the ones used. It also had the best accuracy. To complement with a graphical view of this analysis, I decided to make a graph that shows prediction vs reference data. Although there is a significant high frequency of good predictions (reference equals or near predicted data), there are also several predictions off reference data.

Although RMSE of the best model is 0.6915, accuracy is not ideal 0.5776. For future work, a better model would improve accuracy.

References

1. Forina, M. et al, PARVUS - An Extendible Package for Data Exploration, Classification and Correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy. https://archive.ics.uci.edu/ml/index.php

2. Irizarry, R., (2019). Introduction to Data Science. http://rafalab.dfci.harvard.edu/dsbook/