Juice Capstone Project

Ravi Singh

03 January, 2021

1 Introduction

In this study we talk about a study on Juice which is done as a part of thstudy for a Data Science Certificate in edx [1]. A topic on “Juice” is selected as the topic of the project and the data set is attached within the GitHub reposority. The dataset is related to juice samples were created mixing random data sets from the internet. The dataset contains a lot of columns for a juice and an associated grade.

Juice is a commonly used beverage in every nook and corner of the world. Juice is produced in all sorts of way, preservatives, sugar, naturally are all ingredients used in its production. There can be small additions of acids or sulfates used to control to balance and stabilize the product. "No matter what it costs, most juices are made with water, preservatives and the real fruit juice. The remaining percent is a combination of acids, sugars, volatile flavour and aroma compounds, pigment compounds, etc. Amazingly, it’s this 2 per cent that makes all the difference, giving a wine its unique flavour, colour, and aroma.

The aim of this study is to use the juice dataset to explore the impact of preservatives on juice grade. There are features contained as a small subset within the dataset of a juice which is formed from the balance of sugars, acids, but the character of the juice is provided by the volatile aroma compounds. Thousands of flavors of juices are present throughout the world. They vary by grade and in this study we try to rate them with grade. The study tries to perform an analysis to predict the grade of the juice. Taking into consideration key components to differentiate based on grade and grade.

The following are the steps performed in this Study:-

1. Load/Verify Dataset
2. Understand and Analyze Dataset
3. Structure the Dataset
4. Dataset Analysis through methodology
5. Predict analysis
    - Numeric - regression models
    - Category - classification models
6. Estimation through the presence of prevalence
7. Conclusions through model evaluation
8. Results

Programming and modelling through R code are the most essential step. The codes are present in the R file and also within this body of this Report. All the codes have been used, the codes not seen on the report are seen in the R file and the ones in R would be present in pdf file and all is essential for better data visualization.

2 Data Exploration and Analysis

2.1 Setup

All libraries used are loaded at this step followed by turning off the scientific notation before the programming code can be drafted.

```
#set global options for code chunks to no messages or warnings
knitr::opts_chunk$set(
    message = FALSE,
    warning = FALSE,
    echo = TRUE
)
#install packages if needed
if(!require(tidyverse)) install.packages("tidyverse")
if(!require(caret)) install.packages("caret")
if(!require(knitr)) install.packages("knitr")
if(!require(kableExtra)) install.packages("kableExtra")
if(!require(rpart)) install.packages("rpart")
if(!require(rpart.plot)) install.packages("rpart.plot")
if(!require(AppliedPredictiveModeling)) install.packages("AppliedPredictiveModeling")
if(!require(corrplot)) install.packages("corrplot")
if(!require(randomForest)) install.packages("randomForest")
if(!require(FactoMineR)) install.packages("FactoMineR")
if(!require(factoextra)) install.packages("factoextra")
if(!require(broom)) install.packages("broom")
if(!require(corrplot)) install.packages("corrplot")
if(!require(kernlab)) install.packages("kernlab")
if(!require(mboost)) install.packages("mboost")
if(!require(import)) install.packages("import")
if(!require(gam)) install.packages("gam")
if(!require(kknn)) install.packages("kknn")
if(!require(Rborist)) install.packages("Rborist")
if(!require(mgcv)) install.packages("mgcv")
if(!require(nlme)) install.packages("nlme")
if(!require(RSNNS)) install.packages("RSNNS")
detach(package:RSNNS) #package masks key functions from caret package
#Load the libraries
library(tidyverse)
library(caret)
library(knitr)
library(kableExtra)
library(rpart)
library(rpart.plot)
library(AppliedPredictiveModeling)
library(corrplot)
library(randomForest)
library(rpart)
library(FactoMineR)
library(factoextra)
library(broom)
library(corrplot)
#turn off scientific notation and set number of digits to print
options(scipen = 999, digits = 5)
```

2.2 Load/Verify Dataset

The juice grade dataset is provided as a .csv file, separated with a semicolon. The agenda of the study is only limited to the juice dataset. While loading the dataset, column names are replaced for better understanding, clarity and analysis.

```
# Reading the juice Dataset
white <- read_delim("juice.csv", delim = ";")
# Removing unnecessary spaces in the Juice dataset
colnames(white)<- colnames(white) %>% str_replace_all(" ", "_")
```

2.3 Understand and Analyze Dataset

Loading the juice data set to make sure its loaded properly and there are not errors or problems within the dataset.

```
#Showcase the data structure
str(white)
```

```
## tibble [4,898 x 12] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ fixed_acidity       : num [1:4898] 7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile_acidity    : num [1:4898] 0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
##  $ citric_acid         : num [1:4898] 0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
##  $ residual_sugar      : num [1:4898] 20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num [1:4898] 0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
##  $ free_sulfur_dioxide : num [1:4898] 45 14 30 47 47 30 30 45 14 28 ...
##  $ total_sulfur_dioxide: num [1:4898] 170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num [1:4898] 1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num [1:4898] 3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num [1:4898] 0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
##  $ alcohol             : num [1:4898] 8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : num [1:4898] 6 6 6 6 6 6 6 6 6 6 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `fixed acidity` = col_double(),
##   ..   `volatile acidity` = col_double(),
##   ..   `citric acid` = col_double(),
##   ..   `residual sugar` = col_double(),
##   ..   chlorides = col_double(),
##   ..   `free sulfur dioxide` = col_double(),
##   ..   `total sulfur dioxide` = col_double(),
##   ..   density = col_double(),
##   ..   pH = col_double(),
##   ..   sulphates = col_double(),
##   ..   alcohol = col_double(),
##   ..   quality = col_double()
##   .. )
```

A search for Not Applicable values (NA), shows that there are none present

```
#Remove all NA values
which(is.na(white))
```

```
## integer(0)
```

Final view of the entire Juice dataset

```
# View the juice dataset
summary(white)
```

```
##  fixed_acidity   volatile_acidity  citric_acid    residual_sugar
##  Min.   : 3.80   Min.   :0.080    Min.   :0.000   Min.   : 0.60
##  1st Qu.: 6.30   1st Qu.:0.210    1st Qu.:0.270   1st Qu.: 1.70
##  Median : 6.80   Median :0.260    Median :0.320   Median : 5.20
##  Mean   : 6.85   Mean   :0.278    Mean   :0.334   Mean   : 6.39
##  3rd Qu.: 7.30   3rd Qu.:0.320    3rd Qu.:0.390   3rd Qu.: 9.90
##  Max.   :14.20   Max.   :1.100    Max.   :1.660   Max.   :65.80
##    chlorides      free_sulfur_dioxide total_sulfur_dioxide    density
##  Min.   :0.0090   Min.   :  2.0       Min.   :  9          Min.   :0.987
##  1st Qu.:0.0360   1st Qu.: 23.0       1st Qu.:108          1st Qu.:0.992
##  Median :0.0430   Median : 34.0       Median :134          Median :0.994
##  Mean   :0.0458   Mean   : 35.3       Mean   :138          Mean   :0.994
##  3rd Qu.:0.0500   3rd Qu.: 46.0       3rd Qu.:167          3rd Qu.:0.996
##  Max.   :0.3460   Max.   :289.0       Max.   :440          Max.   :1.039
##        pH         sulphates       alcohol        quality
##  Min.   :2.72   Min.   :0.22   Min.   : 8.0   Min.   :3.00
##  1st Qu.:3.09   1st Qu.:0.41   1st Qu.: 9.5   1st Qu.:5.00
##  Median :3.18   Median :0.47   Median :10.4   Median :6.00
##  Mean   :3.19   Mean   :0.49   Mean   :10.5   Mean   :5.88
##  3rd Qu.:3.28   3rd Qu.:0.55   3rd Qu.:11.4   3rd Qu.:6.00
##  Max.   :3.82   Max.   :1.08   Max.   :14.2   Max.   :9.00
```

The juice data set has 12 x 4898 observations and the type of dataset is numeric.

Grade ratings for juice are found to be ranged from 4 - 9, with a mean of 5.76, a median of six (6.00).

2.3.1 Result Verification

He data set is well structured; it is usable form with no missing values or bad grade grades were detected.

2.4 Create Training and Test model

The training set will be divided into two further sets namely, the training set and the other, the test set. No further analysis would be done in the test dataset. The train an test dataset would be used in the classification of models. A categorla version of the grade results would be created on the numeric dataset. A grade based category version would also be created so as to classify models In addition to this split, we will also be creating a test and train data set to use with our classification models. The minimum to maximum value of the grade grade would be converted into factors.

Set of following codes help perform the initial split between the train and test data set. About a certain percentage (20) would be used in the testing process.

```
# Repeatability set seed
set.seed(931, sample.kind = "Rounding")
# Train & Test dataset for Juice
test_index <- createDataPartition(white$quality, times = 1, p = 0.2, list = FALSE)
train_rating <- white %>% slice(-test_index)
test_rating <- white %>% slice(test_index)
```

Another data set for train and test is created where the grade vales are not numerical but are factors.

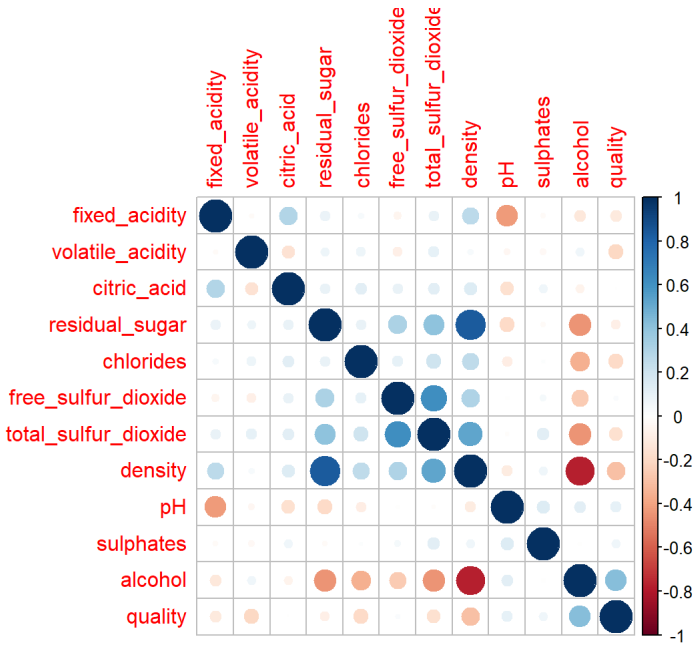
```
# Second test and train dataset is created
# The categories are acceptable, good, premium
# These are the same data sets as the rating sets, but with the quality factor
# Changed to a category
test_category <- test_rating %>%
  mutate(quality = as.factor(
    ifelse(quality >=8, "premium",
           ifelse(quality <= 4, "acceptable", "good") )))
train_category <- train_rating %>%
  mutate(quality = as.factor(
    ifelse(quality >=8, "premium",
           ifelse(quality <= 4, "acceptable", "good") )))
test_category <- test_rating %>%
  mutate(quality = as.factor(quality))
train_category <- train_rating %>%
  mutate(quality = as.factor(quality))
```

2.5 Correlation matrix of variables

A correlation matrix is created to better understand how features and outcomes might be related to better understand the correlation matrix. The following shoes the correlation matrix and its plot (Figure 1).

Correlation Matrix

	fixed_acidity	volatile_acidity	citric_acid	residual_sugar	chlorides	free_sulfur_dioxide	total_sulfur_dioxide	density
fixed_acidity	1.00000	-0.02067	0.29190	0.08983	0.03267	-0.05687	0.09203	0.26811
volatile_acidity	-0.02067	1.00000	-0.15703	0.07058	0.07570	-0.08624	0.10074	0.03270
citric_acid	0.29190	-0.15703	1.00000	0.09173	0.12167	0.09648	0.12045	0.14456
residual_sugar	0.08983	0.07058	0.09173	1.00000	0.09092	0.31632	0.40753	0.83787
chlorides	0.03267	0.07570	0.12167	0.09092	1.00000	0.10093	0.20065	0.25724
free_sulfur_dioxide	-0.05687	-0.08624	0.09648	0.31632	0.10093	1.00000	0.61778	0.30142
total_sulfur_dioxide	0.09203	0.10074	0.12045	0.40753	0.20065	0.61778	1.00000	0.52880
density	0.26811	0.03270	0.14456	0.83787	0.25724	0.30142	0.52880	1.00000
pH	-0.42553	-0.04229	-0.16804	-0.19779	-0.09595	-0.00861	-0.01335	-0.10057
sulphates	-0.02598	-0.03818	0.06559	-0.02658	0.01149	0.04553	0.12827	0.06718
alcohol	-0.12538	0.06492	-0.06751	-0.44545	-0.35828	-0.25106	-0.44534	-0.77586
quality	-0.11869	-0.20195	-0.00037	-0.08194	-0.19790	0.02006	-0.16383	-0.29151



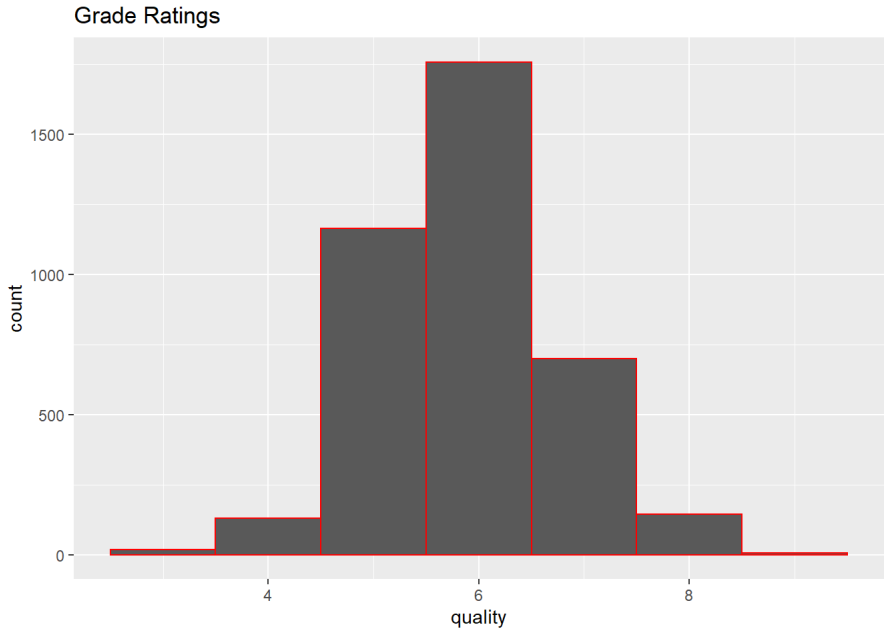
Correlation Plot

Grade outcome is most negatively impacted by the density and is strongly correlated with preservatives. The preservatives and density were also corelated with one another and this relation is not surprising. When preservatives are added to any food product, the density function is most certainly found to be affected, the density is a measure of sugar and preservatives in any beverage. Sugar has a higher density and most certainly rises the specific gravity of juice i.e., affects its density. Certain Preservatives are less denser than water and negatively affect the specific gravity i.e., it decreases the density. It is not surprising to find that all the parameters are correlated We do not eliminate any of the data set as it seems to be of a smaller size, elimination would me more practical if the dataset was extremely large so as to reduce computational power.

2.6 Data Visualization

The dependent variable in this study is the grade values. Predictions are based on grade values of all the 11 observations. Lab analysis from the basis of this study. Each wine only has one entry and one grade rating. Each juice has a single entry and a singular grade rating and each row seen in the data set is for the juice.

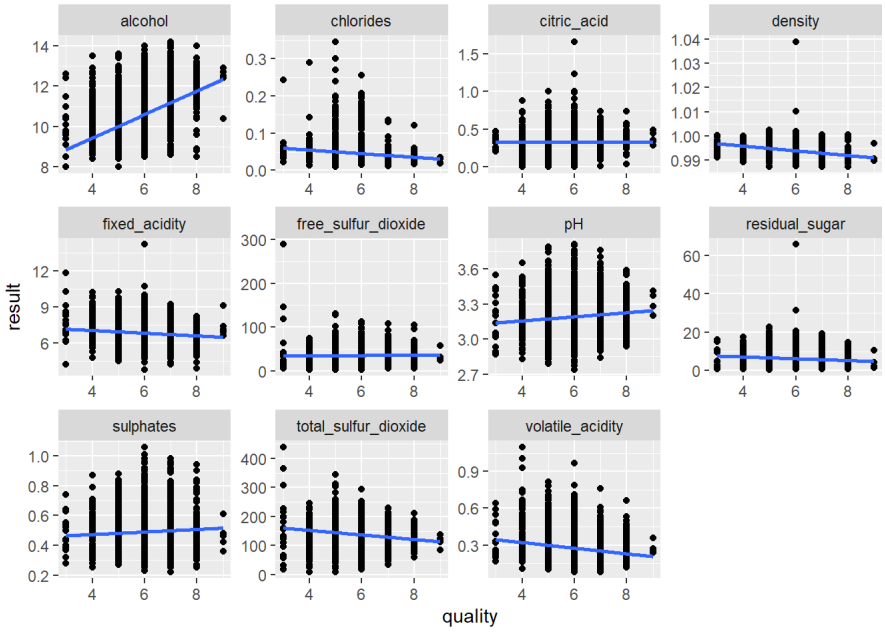
As the prediction is based on grade outcome. A histogram is shown for the grade ratings on the training dataset (Figure 2). The histogram shows a strong tendency for three of the data. High and low ratings are distinctly seen in the histogram.



Quality Rating Histogram

2.6.1 Linear Regression Relationship

A relationship between grade variable and features are useful in modeling. The plots display a feature vs grade (Figure 3). Upon looking at the nature of the graph almost all the variables have a flat line.



Features Test Results

To draw a conclusion, p-value is a factor that is calculated In order to see if there is any significance, the p value for each factor was calculated. A significantly high confidence interval of 95% will most certainly have a p-value below 0.05. The factors are shown below:

```
##        fixed_acidity     volatile_acidity       residual_sugar
##                    0                    0                    0
##            chlorides total_sulfur_dioxide              density
##                    0                    0                    0
##                   pH            sulphates              alcohol
##                    0                    0                    0
```

All values are significant, some have linear correlation, and some don't have the linear trend.

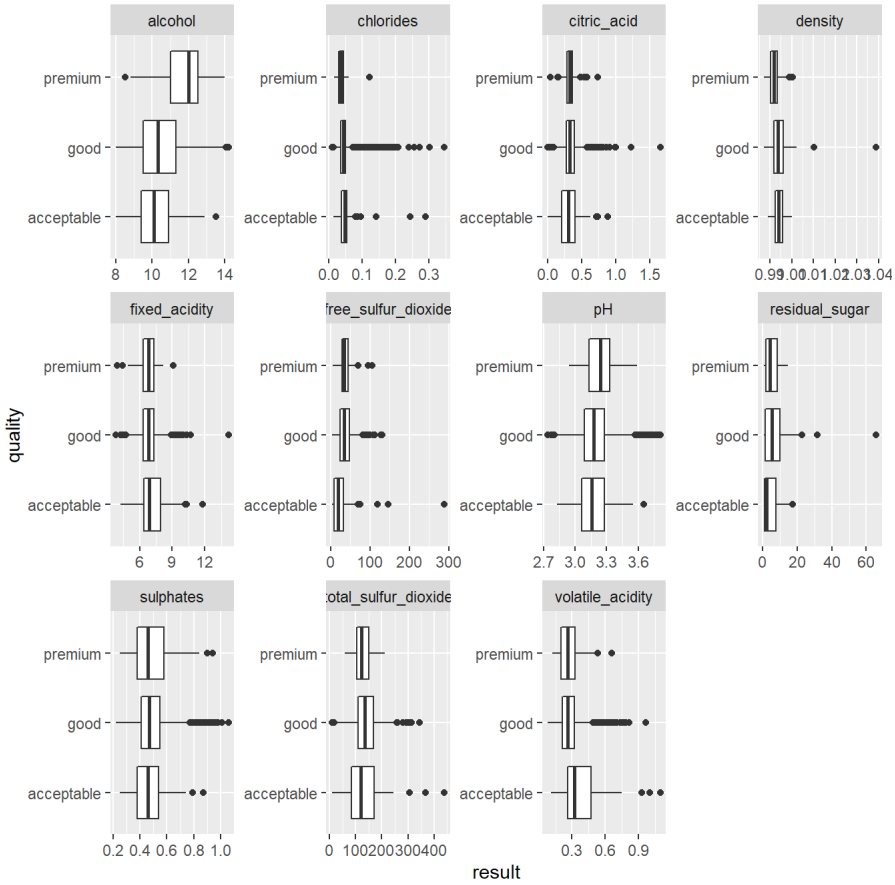
```
##         citric_acid free_sulfur_dioxide
##              0.9818              0.2094
```

2.6.2 Boxplots of Features vs Quality

Boxplots is another means of visualizing a relationship among all the grade values. For better visualization the date is grouped together in three categories as shown below: ws:

- Acceptable - quality ratings 3 and 4
- Good - quality ratings 5, 6, and 7
- Premium - quality ratings 8 and 9

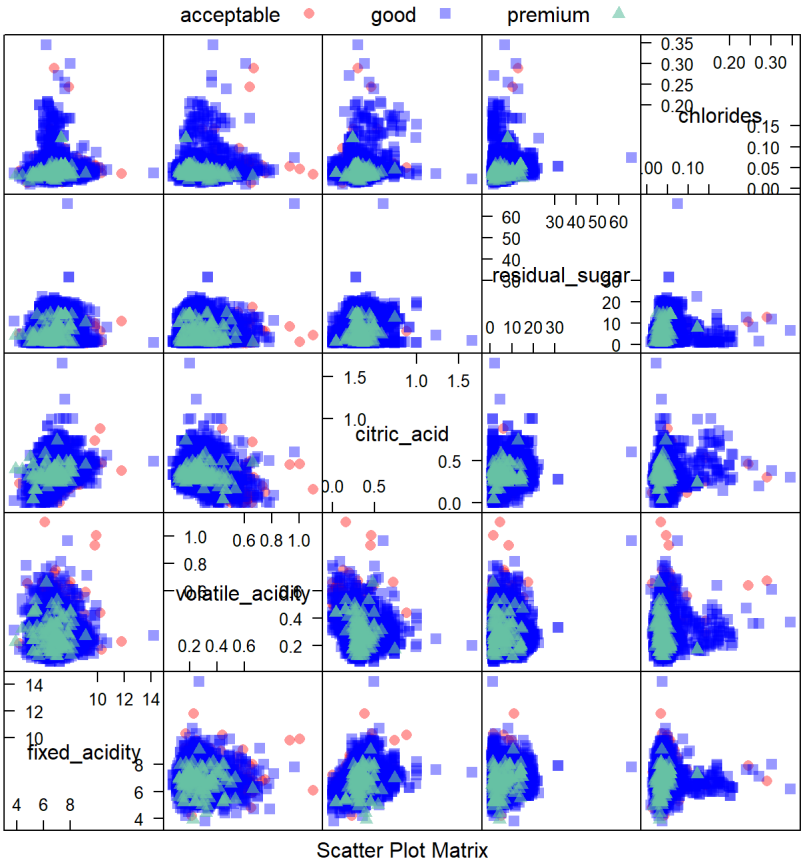
Reduction in categories allows in better visualization and better understanding of boxplots are shown in Figure 4.



Test Result vs Quality

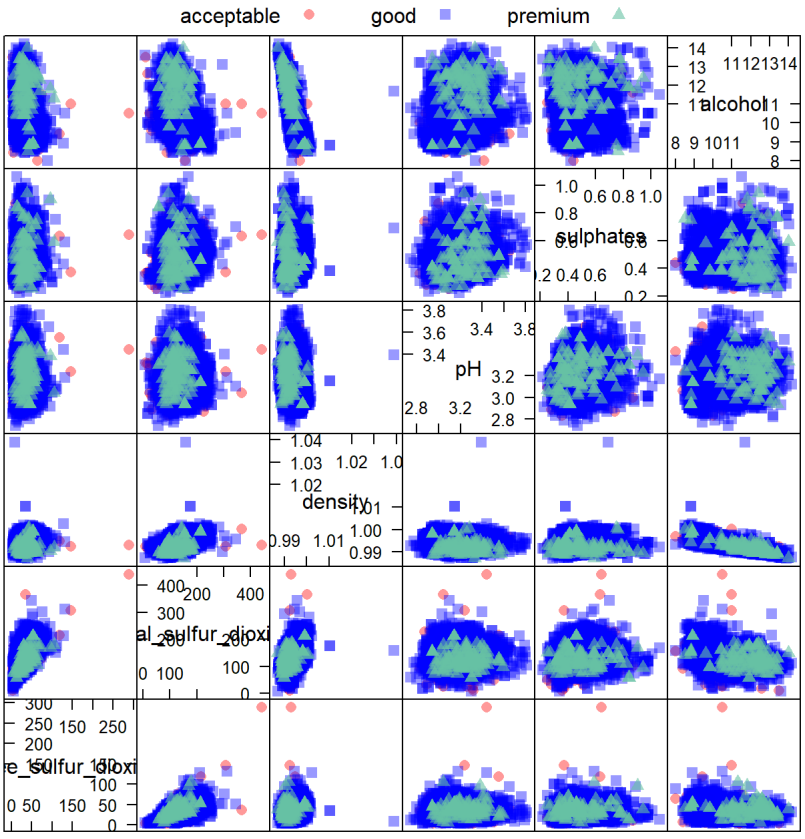
2.6.3 Plot for scattered matrix

Comparison of features vs grade show an interesting result. The ok, great and best ratings are introduced. A boxplot split into two groups were done so as to give a better insight (Figure 5 and Figure 6). On looking at the plots, all plots nearly show the same pattern, the grade rating seem to cluster around the premium rating which are targeted at the center or within center. The acceptable rating could be outlier or lie within the great range, making the prediction slightly difficult and a model that works for concentric circles might be difficult



Scatter Plot Matrix

Features Pairwise Comparison 1



Scatter Plot Matrix

Features Pairwise Comparison 2

2.7 Feature Standardization

Till this point, analysis has been conducted on the originally unstructured Raw data form. The dataset has varying measurements and scales. It can be concluded that standardized dataset is useful as centered and standardized data performed better in this model whereas few other models showed no impact with a few sets of test runs, concluding the usage of standardized data set going forward.

For standardizing the data, a pre-processing function from the caret package will be used. Furthermore, for categorial and numerical analysis, a single set of features are to be used. The grade predictor, y_cat (y_num), will be separate for classification (categorical) and regression(numerical) models.

```
# Function caret
# Subset feature creation
# features <- train_rating[ , -12]
y_num <- train_rating[ , 12]
y_cat <- train_category[ , 12]
y_test_num <- test_rating[ , 12]
y_test_cat <- test_category[ , 12]
# Pre-processing value
# One pre-processing value needed for one set of features.

preProcValues <- preProcess(train_rating[,-12], method = c("center", "scale"))
# training set feature calculation
train_feat <- predict(preProcValues, train_rating[ , -12])
# Test set transformation using mean and standard deviation
# stored in the pre-processing value data set

test_feat <- predict(preProcValues, test_rating[ , -12])
```

As seen in the new transformed dataset, all the features have a mean value of zero and a standard deviation of one. As it can be expected, the mean and standard deviation vales are slightly different since the pre-processing value is made in the train dataset.

```
## Mean and Standard Deviations for the training set:
```

```
##      fixed_acidity volatile_acidity citric_acid residual_sugar chlorides
## [1,]             0                0           0              0         0
## [2,]             1                1           1              1         1
##      free_sulfur_dioxide total_sulfur_dioxide density pH sulphates alcohol
## [1,]                   0                    0       0  0         0       0
## [2,]                   1                    1       1  1         1       1
```

```
##
##
## Mean and Standard Deviations for the test set:
```

```
##      fixed_acidity volatile_acidity citric_acid residual_sugar chlorides
## [1,]         0.074           -0.027      -0.005          0.051    -0.009
## [2,]         1.016            0.967       0.901          1.012     0.986
##      free_sulfur_dioxide total_sulfur_dioxide density     pH sulphates alcohol
## [1,]               0.020                0.001   0.065 -0.036     0.050  -0.046
## [2,]               0.991                0.990   1.007  1.018     1.048   1.006
```

2.8 Decision Tree for better understanding and Data Mining

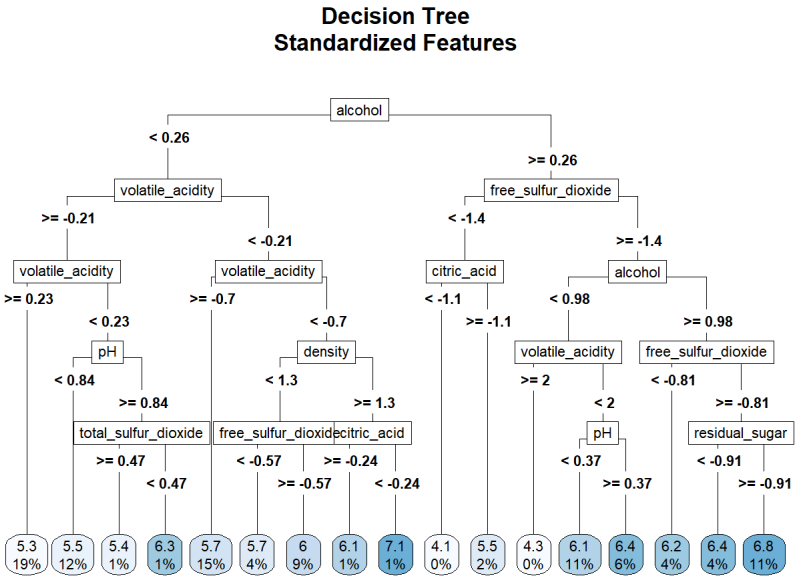
Aim of this study is to find out the features that can be helpful for juice making/creation. Decision tree is one such tool used for better understanding and Data Mining.

To meet our studies requirement, decision tree is the closest and best method, for data mining. Fast to construct and help is better introspection [@Hastie, Trevor, Robert Tibshirani, Jerome Friedman. 2017. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Second Edition. pg 352] First step is observing the decision tree for better understand and predicting the rating value of juice (Figure 7). The tree is based on standardizing data. The standardization process would be reversed if utilizing the figure for decision making.

```
# Training and Test data set for standardizing feature
train <- cbind(train_feat, y_num)
test <- cbind(test_feat, y_test_num )
# Training Rpart model
train_rpart2 <- train(quality ~ .,
                      method = "rpart",
                      tuneGrid = data.frame(cp = seq(0.0,0.1, len=25)),
                      data = train)
# RMSE check of the model
y_hat <- predict(train_rpart2, test)
rmse_rpart_tuned <- RMSE(test$quality, y_hat)
# Showcase RMSE result
cat("The RMSE for the model produced by the decision tree is", rmse_rpart_tuned )
```

```
## The RMSE for the model produced by the decision tree is 0.7114
```

```
# Rpart model prediction
rpart.plot(train_rpart2$finalModel,
           main = "Decision Tree\nStandardized Features",
           type = 5,
           yesno = 2,
           cex = .7)
```



Decision Tree for Numeric Ratings

Decision tree can help us understand and assign values in the model. In a decision tree, there are quite a few branches, with same features used on multiple branches like preservatives. There are unique features that appear on the decision tree..

```
## [1] "alcohol"              "volatile_acidity"     "pH"
## [4] "total_sulfur_dioxide" "density"              "free_sulfur_dioxide"
## [7] "citric_acid"          "residual_sugar"
```

Rpart model help produce variable important matrix, it can help provide an overall measure [1]. A few parameters have high importance and a few don't, ones who have high importance are chlorides and citric acid. Variable importance is shown as follows:-

```
# Importance variable
vi <- varImp(train_rpart2)$importance %>% arrange(desc(Overall))
# Saving variable importance for future calculation
vi_table <- as_tibble_col(rownames(vi)[1:4], column_name = "rating")
# Displaying the obtained variable importance
vi
```

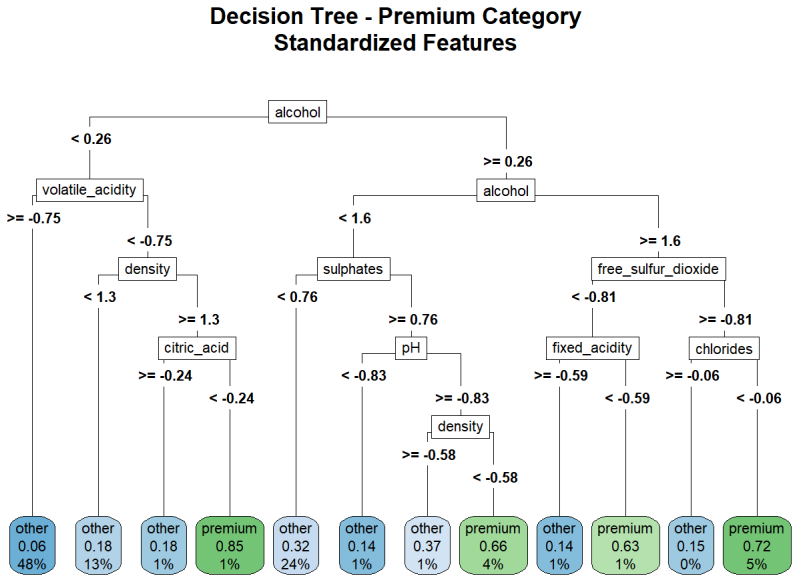
	Overall <dbl>
alcohol	100.000
chlorides	74.907
citric_acid	59.073
residual_sugar	56.714
pH	56.107
density	29.801
free_sulfur_dioxide	27.536
total_sulfur_dioxide	21.391
volatile_acidity	16.173
fixed_acidity	11.037

1-10 of 11 rows Previous **1** 2 Next

2.8.1 Decision Tree for the Best category

Next step is to look at the other categories, the best and the others (Figure 8). If a juice creator wants to be the best, he needs to have the best grade of juice with him. When focusing on the best category, the tree needs to be be very very different. Where the focus is to correctly pic the best grade around all the spectrum

```
# Set up category for the best or others
y_prem <- y_num %>% mutate(quality = ifelse(quality < 7, "other", "premium"))
# Training set creation
train <- cbind(train_feat, y_prem)
# Train the category model
train_rpart2 <- train(quality ~ .,
                      method = "rpart",
                      tuneGrid = data.frame(cp = seq(0.0,0.1, len=25)),
                      data = train)
# Reduce the number of branchess
pruned <- prune(train_rpart2$finalModel, cp = 0.01)
# Plotting the tree
rpart.plot(pruned,
           main = "Decision Tree - Premium Category\nStandardized Features",
           type = 5,
           yesno = 2,
           cex = .7)
```



Decision Tree for Premium Category

The Decision tree if different from the previously shown decision tree. I needs to reduce the branches i.e., it need to be pruned. With the change in the tree, features utilization greatly increased form the previous 7 to the new 11. The variable importance shows significant shifts.

```
# Return the three terms
ind <- !(train_rpart2$finalModel$frame$var == "<leaf>")
tree_terms <-
  train_rpart2$finalModel$frame$var[ind] %>%
  unique() %>%
  as.character()
# Showcase tree terms
tree_terms
```

```
##  [1] "alcohol"              "volatile_acidity"     "density"
##  [4] "citric_acid"          "sulphates"            "residual_sugar"
##  [7] "chlorides"            "total_sulfur_dioxide" "fixed_acidity"
## [10] "pH"                   "free_sulfur_dioxide"
```

```
# Importance Variable
vi <- varImp(train_rpart2)$importance %>% arrange(desc(Overall))
# Saving variable importance for future calculation
vi_table <- add_column(vi_table, premium = rownames(vi)[1:4])
# Display variable importance
vi
```

	Overall <dbl>
alcohol	100.0000

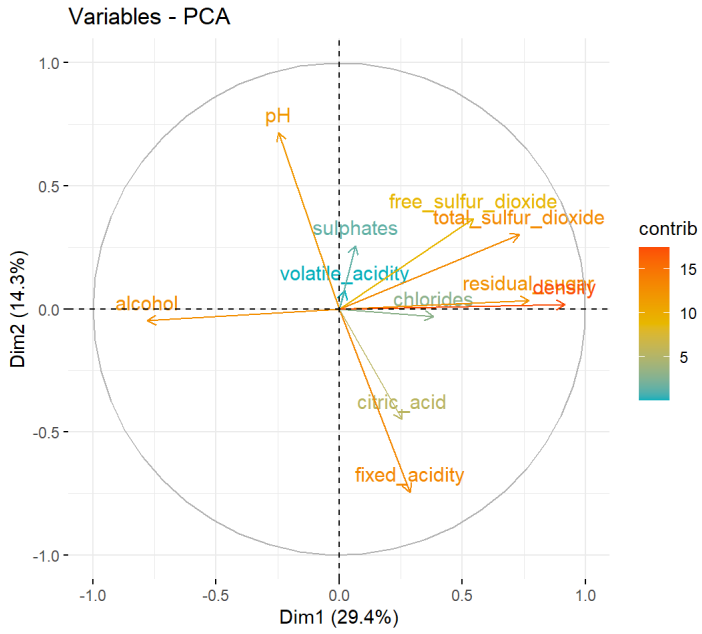
	Overall <dbl>
density	64.9570
chlorides	49.2895
pH	23.3611
citric_acid	19.0244
free_sulfur_dioxide	18.0296
total_sulfur_dioxide	14.6757
fixed_acidity	12.5542
residual_sugar	5.5581
volatile_acidity	1.6646

1-10 of 11 rows Previous **1** 2 Next

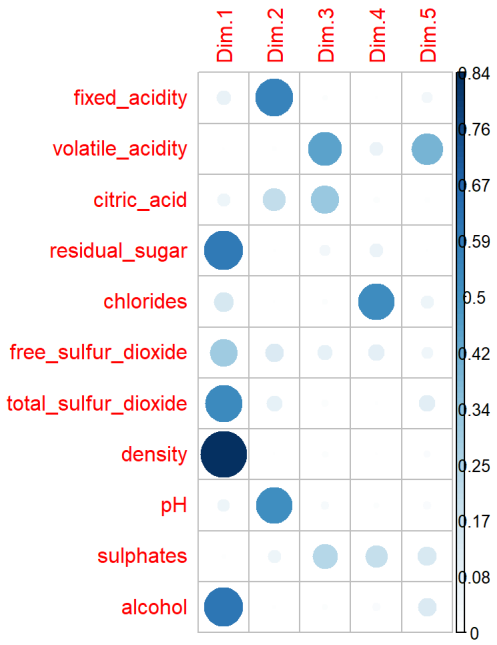
2.9 Analysis (Principal Component)

Another analysis used to understand the feature is the Principal Component Analysis (PCA). The impact of grade rating isn't included in the analysis as we are trying to find relationships. The two plots about PCA are shown below. PCA is a data analysis method which assists in visualization and better understanding of dataset values."[2]

Figure 9 displays such a visualization chart using the above stated PCA. Features that point a single direction are positively correlated (sugar, acid, density) and the ones pointing in the other direction are negatively correlated (pH value). The color code showcases the two dimensions. Another PCA analysis, Figure 10 displays correlation plot of five dimensions in PCA. The color and size of the dot represent the contribution of dimension shown in the right side of the plot. The graphs help us understand keep variables like sugar, preservatives, density, pH, acidity. Numerical result of the PCA analysis is shown below.



PCA Variables Plot



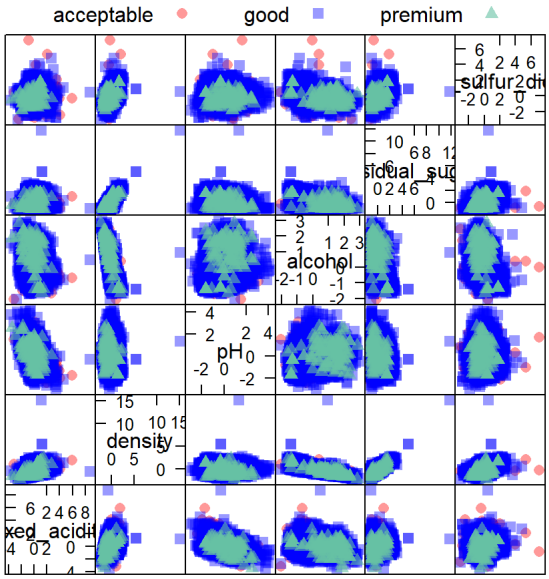
Correlation Plot

feature <chr>	Dim.1 <dbl>
density	25.979462
alcohol	18.779256
residual_sugar	18.369159
total_sulfur_dioxide	16.543640
free_sulfur_dioxide	9.220821
chlorides	4.530552
fixed_acidity	2.537572
citric_acid	2.012563
pH	1.882410
sulphates	0.128170

1-10 of 11 rows Previous **1** 2 Next

2.10 PCA based scatterplot matrix

On PCA analysis, the important feature preservatives, acidity, density, sugar, pH. Figure 11 shows the scatter plot, in the standardized format. The pattern is still existent with the display of a bulls eye.



Scatter Plot Matrix

Important Features from PCA

2.11 Analysis result

Three unsupervised model provide commonality of variable importance, solutions contain variability. As seen in Table, preservatives and density top variables in three models. Chlorides is the most important variable. Most important variable could be suger, dioxide in the order. Citirc acid is present in a single model.

Preservatives, density and sugar are factors that are correlated. Increase in preservatives content is key, with some variability not showing high levels of accuracy. Variations exist and there could be two causes. First being missing components or the features are correlated with wrong factors. Example preservatives, density and sugar are highly correlated, there could be even better predictors which are not in the database and hence are unrelated.

```
# Importance Variable
# Variable importance for future comparison and unsupervised models
vi_table <- add_column(vi_table, PCA = vc$feature[1:4])
# Displaying importance Table
kable(vi_table, caption = "Variable Importance", booktabs=T)%>%
  kable_styling(latex_options = "hold_position")
```

Variable Importance

rating	premium	PCA
alcohol	alcohol	density
chlorides	density	alcohol
citric_acid	chlorides	residual_sugar
residual_sugar	pH	total_sulfur_dioxide

3 Modeling

Numerical outcomes and categorical outcome, both can be used to conduct modelling. Modeling was conducted for both numeric and categorical outcomes. The modeling is performed by utilizing the caret function package. As can be seen from the code shown in this section, a “group_train” function was employed which allowed for the testing of multiple models in one run. Many more models were tested than the ones included in this report. Only a few example model outcomes are included as part of this report.

3.1 Regression Models

It provides prediction of results in a numerical manner. Evaluation of multiple models will help us determines which model works best. As it is regression, model that minimizes RMSE is utilized.

Summary of regression runs are shown in Table. It is not at all shocking to see the linear models did not perform well. Random forest model produced the lowest RMSE.

```r
# Creating train and test from features
train <- train_feat %>% cbind(y_num)
test <- test_feat %>% cbind(y_test_num)
# ML model list to evaluate
models <- c("glm", "svmLinear", "gamboost","gamLoess", "knn",
            "kknn", "gam","ranger", "rf", "Rborist", "mlp",
            "svmRadial", "svmRadialCost", "svmRadialSigma")
#Function to visualize function
#Runs training model list

group_train <- function(model_list, seed = 831){
  # using override user or set seed model
  set.seed(seed, sample.kind = "Rounding")
  # Model length
  l <- length(model_list)
  # initialize the train_list after traiing first model
  train_result <- train(quality ~ ., method = model_list[1], data = train)
  train_list <- list(train_result)
  # remaining models training
  for (i in 2:l){
    train_result <- train(quality ~ ., method = model_list[i], data = train)
    train_list[[i]] <- train_result
  }
  # Model Namet
  names(train_list) <- model_list
  # List of training model
  return(train_list)
}
# Models train
model_list <- group_train(models)
# Prediction matrix creation using ML Model
p <- sapply(model_list,predict, test)
# RMSE model calculation
a <- apply(p, 2, RMSE, test$quality) %>% enframe(name="Model", value = "RMSE")
# RMSE table displaying
kable(a, caption = "Regression Models", booktabs=T)%>%
  kable_styling(latex_options = "hold_position")
```

Regression Models

Model	RMSE
glm	0.72591
svmLinear	0.73068
gamboost	0.69499
gamLoess	0.68673
knn	0.67166
kknn	0.63441
gam	0.70082
ranger	0.55569
rf	0.55643
Rborist	0.55078
mlp	1.05568
svmRadial	0.65794
svmRadialCost	0.66596
svmRadialSigma	0.65757

3.1.1 Random Rborist

Good results for best Rborist regression models. Tuning the Rborist model is important to get result

```
# Test and train for numeric evaluation
train <- train_feat %>% cbind(y_num)
test <- test_feat %>% cbind(y_test_num)
# Model tuning grid
train_rborist <- train(quality ~ .,
             method = "Rborist",
             tuneGrid = data.frame(predFixed = 2,
                                   minNode = c(3, 50)),
             data = train)
# Value for the test set
y_hat <- predict(train_rborist, test)
# Calculate the model RMSE
rmse_rborist <- RMSE(test$quality, y_hat)
# Display the results
cat("The RMSE for the tuned Rborist model is", rmse_rborist)
```

```
## The RMSE for the tuned Rborist model is 0.55238
```

RMSE for the base caret model and tuned model are the same. Tuning parameter in caret package performed well.

3.2 Classification Models

Predictions such as categorical predictions are provided by the classification model. Multiple models are evaluated to determine which model performs well. As there is a classification model approach, we look to maximize accuracy. Accuracy simply means proportion of true correct value, the accuracy in R programming is calculated using this model.

Converting grade rating factors into numbers and accessing them can be done with RMSE. Not an interest to optimize models but to calculate RMSE using regression models analysis.

```
# Resetting train and test set
train <- train_feat %>% cbind(y_cat)
test <- test_feat %>% cbind(y_test_cat)
# ML model creation
models <- c("knn", "kknn", "Rborist", "ranger", "mlp","svmRadial")
# Evaluating each function in the supply list
# Returning list of test model
group_train <- function(model_list, seed = 831){
  # Model run and set seed model
  set.seed(seed, sample.kind = "Rounding")
  # Length of list
  l <- length(model_list)
  # initialize train_list after input
  train_result <- train(quality ~ ., method = model_list[1], data = train)
  train_list <- list(train_result)
  # Remaining model training
  for (i in 2:l){
    train_result <- train(quality ~ ., method = model_list[i], data = train)
    train_list[[i]] <- train_result
  }
  # Naming model in list
  names(train_list) <- model_list
  # List of training model
  return(train_list)
}
 # Model train
model_list <- group_train(models)
# RMSE calculation
RMSE2 <- function(predicted_ratings, true_ratings){
  # Addition of value for rating correction
  tr <- as.numeric(true_ratings) +2
  pr <- as.numeric(predicted_ratings)
  sqrt(mean((tr - pr)^2))
}
# ML Model column prediction matrix
p <- sapply(model_list, predict, test)
# RMSE for each model
a <- apply(p, 2, RMSE2, test$quality) %>% enframe(name="Model", value = "RMSE")
kable(a, caption = "Classification Models", booktabs=T) %>%
  kable_styling(latex_options = "hold_position")
```

Classification Models

Model	RMSE

Model	RMSE
knn	0.81316
kknn	0.75255
Rborist	0.60945
ranger	0.60945
mlp	0.78701
svmRadial	0.73886

```
# Hold accurate result
accuracy_df <- tibble(Model = character(),
                      Accuracy = numeric())
# Display accuracy for all model run
for(i in 1:ncol(p)){
  y_hat <- factor(p[,i], levels(test$quality))
  cm <- confusionMatrix(y_hat, test$quality)
  accuracy_df[i,1] <- models[i]
  accuracy_df[i,2] <- cm$overall["Accuracy"]
}
kable(accuracy_df, caption = "Classification Model Accuracy", booktabs=T )%>%
  kable_styling(latex_options = "hold_position")
```

Classification Model Accuracy

Model	Accuracy
knn	0.55000
kknn	0.65577
Rborist	0.69694
ranger	0.70306
mlp	0.54694
svmRadial	0.58367

```
# "ranger", a prediction for best model
y_hat <- factor(p[,"ranger"], levels(test$quality))
# Specificity and sensitivity print out
cm <- confusionMatrix(y_hat, test$quality)
cm$overall["Accuracy"]
```

```
## Accuracy
##  0.70306
```

```
# cm$byClass[,1:2]
# broom function to get the confusion matrix output in tidy format
tcm <- tidy(cm)
# drop rows to create a wide version
tcm <- tcm[3:nrow(tcm),] %>%
  select(term, class, estimate) %>%
  pivot_wider(names_from = term, values_from = estimate) %>%
  select(class, sensitivity, specificity, prevalence, detection_rate)
# Dsiplay prediction
tcm
```

class <chr>	sensitivity <dbl>	specificity <dbl>	prevalence <dbl>	detection_rate <dbl>
3	0.00000	1.00000	0.0020408	0.0000000
4	0.18750	1.00000	0.0326531	0.0061224
5	0.71429	0.90087	0.3000000	0.2142857
6	0.84318	0.64074	0.4489796	0.3785714
7	0.49171	0.96496	0.1846939	0.0908163

class <chr>	sensitivity <dbl>	specificity <dbl>	prevalence <dbl>	detection_rate <dbl>
8	0.41935	0.99895	0.0316327	0.0132653

6 rows

```
# save Prediction
y_hat_ranger <- y_hat
```

Good Result for accuracy and RSME is produced by Ranger. The results are shown above. Because of the issue of prevalence, sensitivity of predicting best and great ratings are low. The mode error is also a little high.

It is well understood to avoid any low rating target and to select the best of the lot. If its possible to improve sensitivity, it will improve the model as a whole. Knn model below could be a good substitute.

```
# knn model predictor
y_hat <- factor(p[,1], levels(test$quality))
# sensitivity and specialty
cm <- confusionMatrix(y_hat, test$quality)
cm$overall["Accuracy"]
```

```
## Accuracy
##     0.55
```

```
#cm$byClass[,1:2]
#broom function to get the confusion matrix.

tcm2 <- tidy(cm)
# Create a broader version by dropping two rows
tcm2 <- tcm2[3:nrow(tcm2),] %>%
  select(term, class, estimate) %>%
  pivot_wider(names_from = term, values_from = estimate) %>%
  select(class, sensitivity, specificity, prevalence, detection_rate)
# Showcase result
tcm2
```

class <chr>	sensitivity <dbl>	specificity <dbl>	prevalence <dbl>	detection_rate <dbl>
3	0.000000	1.00000	0.0020408	0.0000000
4	0.093750	0.99895	0.0326531	0.0030612
5	0.581633	0.81924	0.3000000	0.1744898
6	0.659091	0.58704	0.4489796	0.2959184
7	0.403315	0.89862	0.1846939	0.0744898
8	0.064516	0.98736	0.0316327	0.0020408

6 rows

3.3 Compensate for Prevalence

Leveling helps remove prevalence in the category. Leveling will create a dataset of observations. Process will start with base observations and sample replacement, similar to bootstrapping. This approach is new and can help compensate earlier models and can be extended to handle multiple outcomes as correctly reflected in the data set.

By utilizing the leveled dataset, the leveling process can improve sensitivity and can help make good predictions.[3]

`pub.va/leanpub.com/d

```
## Clear test and train set
train <- train_feat %>% cbind(y_cat)
test <- test_feat %>% cbind(y_test_cat)
# No of observations of ratings
obs_by_category <- train %>% group_by(quality) %>% summarize(obs = n())
# Leveled dataset
train_level_prev <- train
for(i in 1:7){
  obs <- obs_by_category[i,2] %>% .$obs
  qlty <- obs_by_category[i, 1] %>% .$quality
  sample_obs <- train %>% filter(quality == qlty)
  ind <- sample(1:nrow(sample_obs), 2000 - obs, replace = TRUE)
  new_obs <- sample_obs[ind,]
  train_level_prev <- rbind(train_level_prev, new_obs)
}
```

3.4 Leveled Models

Subsection repets the model run utilizing leveled function, with reduced number of runs. Table summarizes the RMSE results. This section repeats the multiple model run from above, but this time using the leveled features. For efficiency purposes, the number of models was also reduced. Table 6 summarizes the Accuracy results and Table 7 summarizes the RMSE results.

```
# Training set for the leveled model run
train <- train_level_prev
# Remove redundant object
rm(train_level_prev)
# Test set for modelling as categories
# train <- train_feat %>% cbind(y_cat)

test <- test_feat %>% cbind(y_test_cat)
# ML evaluation
models <- c("knn", "kknn", "ranger", "rf", "Rborist", "mlp")
# Evaluate each function
# A list of trained models

group_train <- function(model_list, seed = 831){
  # use user override function
  set.seed(seed, sample.kind = "Rounding")
  # Getting length
  l <- length(model_list)
  # train the model train_list
  train_result <- train(quality ~ ., method = model_list[1], data = train)
  train_list <- list(train_result)
    # Train remaining
  for (i in 2:l){
    # print(model_list[i])
    train_result <- train(quality ~ ., method = model_list[i], data = train)
    train_list[[i]] <- train_result
  }
  ## Name model in list
  names(train_list) <- model_list
  # Return list of models
  return(train_list)
}
# train_models
model_list <- group_train(models)
# column for each ML Model
# caret predict function

p <- sapply(model_list, predict, test)
# ranger model results
y_hat <- factor(p[,"ranger"], levels(test$quality))
# display the confusion matrix
cm <- confusionMatrix(y_hat, test$quality)
cm$overall["Accuracy"]
```

```
## Accuracy
##  0.71939
```

```
# broom function for the confusion matrix output
tcm_r <- tidy(cm)
# drop rows and create a broad version
tcm_r <- tcm_r[3:nrow(tcm_r),] %>%
  select(term, class, estimate) %>%
  pivot_wider(names_from = term, values_from = estimate) %>%
  select(class, sensitivity, specificity, prevalence, detection_rate)
# Result display
cat("\n\n\nThe confusion matrix results for the ranger model after leveling are:\n")
```

```
##
##
##
## The confusion matrix results for the ranger model after leveling are:
```

```
tcm_r
```

class	sensitivity	specificity	prevalence	detection_rate
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
3	0.00000	0.99898	0.0020408	0.0000000
4	0.28125	1.00000	0.0326531	0.0091837

class <chr>	sensitivity <dbl>	specificity <dbl>	prevalence <dbl>	detection_rate <dbl>
5	0.75510	0.89650	0.3000000	0.2265306
6	0.79773	0.71667	0.4489796	0.3581633
7	0.60221	0.93992	0.1846939	0.1112245
8	0.45161	0.99789	0.0316327	0.0142857

6 rows

```
# Data frame creation
accuracy_df <- tibble(Model = character(),
                      Accuracy = numeric())
# Accuracy printing of modelun
for(i in 1:ncol(p)){
  y_hat <- factor(p[,i], levels(test$quality))
  cm <- confusionMatrix(y_hat, test$quality)
  accuracy_df[i,1] <- models[i]
  accuracy_df[i,2] <- cm$overall["Accuracy"]
}
kable(accuracy_df, caption = "Leveled Model Accuracy", booktabs=T )%>%
  kable_styling(latex_options = "hold_position")
```

Leveled Model Accuracy

Model	Accuracy
knn	0.46319
kknn	0.65679
ranger	0.71939
rf	0.70582
Rborist	0.70143
mlp	0.42252

```
# Create RMSE values
RMSE2 <- function(predicted_ratings, true_ratings){
  #the min factor is 3, so add 2 to the conversion to get the rating correct
  tr <- as.numeric(true_ratings) +2
  pr <- as.numeric(predicted_ratings)
  sqrt(mean((tr - pr)^2))
}
# RMSE calculation
a <- apply(p, 2, RMSE2, test$quality) %>% enframe(name="Model", value = "RMSE")
# Display RMSE results
kable(a, caption = "Leveled Model RMSE", booktabs=T)%>%
  kable_styling(latex_options = "hold_position")
```

Leveled Model RMSE

Model	RMSE
knn	1.01318
kknn	0.72703
ranger	0.58902
rf	0.58902
Rborist	0.60187
mlp	1.18623

The accuracy decreases in the model, decrease from the original value. As Ranger has the best result, lets look at the specialty and sensitivity model. Knn model has a good sensitivity.

```
# Pre-leveled result
cat("The confusion matrix results for the knn model prior to leveling were:\n")
```

```
## The confusion matrix results for the knn model prior to leveling were:
```

```
tcm2
```

class <chr>	sensitivity <dbl>	specificity <dbl>	prevalence <dbl>	detection_rate <dbl>
3	0.000000	1.00000	0.0020408	0.0000000
4	0.093750	0.99895	0.0326531	0.0030612
5	0.581633	0.81924	0.3000000	0.1744898
6	0.659091	0.58704	0.4489796	0.2959184
7	0.403315	0.89862	0.1846939	0.0744898
8	0.064516	0.98736	0.0316327	0.0020408

6 rows

```
# knn model prediction
y_hat <- factor(p[,1], levels(test$quality))
# display confusion matrix
cm <- confusionMatrix(y_hat, test$quality)
cm$overall["Accuracy"]
```

```
## Accuracy
##  0.46319
```

```
# broom function to get confusion matrix
tcm <- tidy(cm)
# Create braod function
tcm <- tcm[3:nrow(tcm),] %>%
  select(term, class, estimate) %>%
  pivot_wider(names_from = term, values_from = estimate) %>%
  select(class, sensitivity, specificity, prevalence, detection_rate)
# Result display
cat("\n\n\nThe confusion matrix results for the knn model after leveling are:\n")
```

```
##
##
##
## The confusion matrix results for the knn model after leveling are:
```

```
tcm
```

class <chr>	sensitivity <dbl>	specificity <dbl>	prevalence <dbl>	detection_rate <dbl>
3	0.00000	0.99488	0.002045	0.000000
4	0.56250	0.93446	0.032720	0.018405
5	0.56463	0.79971	0.300613	0.169734
6	0.34624	0.82004	0.448875	0.155419
7	0.56667	0.81579	0.184049	0.104294
8	0.48387	0.91869	0.031697	0.015337

6 rows

As shown above, the sensitivity when predicting category 8 increased from 0.0541 in the original model to 0.541 in the leveled model. Although the overall model accuracy is lacking, this increase in sensitivity while still maintaining very good specificity, could be useful in some applications such as wine purchasing.

4 Result Summary

4.0.1 Unsupervised Models

Unsupervised models were explored, a decision tree made, Three models with similarities and accuracy measured either RMSE, the categories were stated and accuracy, sensitivity and fraction explained. Correlation of models were made, as and where possible keeping in mind all the columns in the data set and correlation among them ### Supervised models

Models were tested using caret package Best prediction for category and accuracy. No result is bad or precise.

Errors in the Rborist model primarily existed in the high and low grade ratings. It was suspected that prevalence might be impacting the results because of the low prevalence of both high and low grade ratings.

Leveling was introduced to test the impact of prevalence. Leveling had a significant impact on the knn model, but had no impact on the random forest models. The random forest models still performed significantly better than any other family of models. It appears that there is not a clear signal present in the features contained in the dataset to accurately predict highly and poorly rated wines.

A look at the confusion matrix when predicting categories showed that although the accuracy was poor, the specificity of the predictions was actually very good and nearly 100% accurate. The sensitivity was poor only picking up a small percentage of the premium grade ratings.

5 Conclusions

Several conclusions can be drawn from these results.

1. The data set contains a very limited set of features.
    - These products are based on physicochemical characteristics of the finished wine. Volatile compounds, such as esters, thiols, fatty acids, etc., were not measured and are key aroma components in finished wine.
    - Features associated with the wine making ingredients and process were not included in the dataset. These features might provide much more insight for the unsupervised models and increase the accuracy of the supervised models.
    - Grade ratings were expressed as a single final statistic. Typically these ratings have a number of subcategories and are complied by multiple individual tasters. Access to the grade scoring process data might be very useful
2. Accuracy and RMSE of models could be better
    - The results are marginally useable. They do not predict high and low grade ratings well, and that is what would be most useful.
    - Specificity is good, which would make the models useful for wine purchasing if the ratings were still unknown.
    - Wether the models could be useful for blending or wine making would need further testing.
3. Leveling did not improve the accuracy of the models but did improve the sensitivity of the knn model for predicting the premium category. If this were being used by a wine buyer, the increased sensitivity might be valuable.

Overall, the models produced have some limited usefulness. I suspect that expanding the features available in the data set, as described above, would produce improved results.

6 Further Study

The primary area for increased study would be to broaden the features data set and include key features that address the volatile aroma components, the wine making process, and the grade scoring process.

In addition, it would be interesting to see if establishing blending targets using the models could be leveraged to improve grade scores. This might be very useful to winemakers.

Extensibility would be another area of interest. This dataset was for juice vihno verde produced in Northern Portugal. I suspect that the characteristics for red wines from the same region would be quite different. Also, I would suspect that region and grape varieties have a major impact on the model. Studying the impact and variability over a variety of regions, grapes, and wine styles might lead to some interesting insights.

7 References

Australian Academy of Science. 2017. The chemistry of wine: Part 1. https://www.science.org.au/curious/earth-environment/chemistry-wine-part-1 (https://www.science.org.au/curious/earth-environment/chemistry-wine-part-1)

P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009. https://archive.ics.uci.edu/ml/datasets/Wine+Grade (https://archive.ics.uci.edu/ml/datasets/Wine+Grade)

Hastie, Trevor, Robert Tibshirani, Jerome Friedman. 2017. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition

Irizzary, Rafael, 2019. Introduction to Data Science. https://leanpub.com/datasciencebook (https://leanpub.com/datasciencebook)

Kassambra, Alboukadel. 2017. PCA in R Using FactoMineR: Quick Scripts and Videos. http://www.sthda.com/english/articles/22-principal-component-methods-videos/65-pca-in-r-using-factominer-quick-scripts-and-videos/ (http://www.sthda.com/english/articles/22-principal-component-methods-videos/65-pca-in-r-using-factominer-quick-scripts-and-videos/)

Kuhn, Max. 2019. The caret Package. https://topepo.github.io/caret/ (https://topepo.github.io/caret/)

Lê, S., Josse, J. & Husson, F. (2008). FactoMineR: An R Package for Multivariate Analysis. Journal of Statistical Software. 25(1). pp. 1-18.

Therneau, Terry, Elizabeth Atkinson. 2019. An Introduction to Recursive Partitioning Using the RPART Routines. https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf (https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf)

Wansbrough, Heather, Robert Sherlock, Maurice Barnes, Malcolm Reeves. 2017. Chemistry in Winemaking. pg 5. https://nzic.org.nz/app/uploads/2017/10/6B.pdf (https://nzic.org.nz/app/uploads/2017/10/6B.pdf)

---

1. Therneau, Terry, Elizabeth Atkinson. 2019. An Introduction to Recursive Partitioning Using the RPART Routines. p12. https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf (https://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf)↩

2. Kassambra, Alboukadel. 2017. PCA in R Using FactoMineR: Quick Scripts and Videos. http://www.sthda.com/english/articles/22-principal-component-methods-videos/65-pca-in-r-using-factominer-quick-scripts-and-videos/ (http://www.sthda.com/english/articles/22-principal-component-methods-videos/65-pca-in-r-using-factominer-quick-scripts-and-videos/)↩

3. Irizzary, Rafael, 2019. Introduction to Data Science.<[[https:/com/datascom/datasciencebook\\](https:/com/datascom/datasciencebook){.uri}(%5Bhttps:/com/datascom/datasciencebook%5D(https:/com/datascom/datasciencebook)%7B.uri%7D)>](https:/ (https:/)↩