12/8/2020

STAT 350 Final Project

Group 7 – Dataset 8/Wine Data

Simon Fraser University – STAT 350

Professor: Derek Bingham

Prepared by: Crystal Fan & Iker Guo

Table of Contents

Introduction 2

Data Description 2

Testing Models 4

Data cleaning 4

Model 1 – Linear Regression with Stepwise Selection 4

Procedure 4

Output 4

Model Check 5

Assumptions check for linear regression 6

Model 2 - Ordinal logistic regression (OLR) 7

Procedure 7

Assumptions in OLR 8

Output 8

Model Check 9

Model 3 - Random forest (one of the bootstrap aggregating) 10

Procedure 10

Assumptions in Random Forrest 10

Output 10

Model Check 10

Model comparison and Final selection 12

Cross Validation 13

Add Extra Point into Data 14

Conclusion 15

Appendix and Reference 16

Code 17

Stepwise 21

Abstract

Wine is one of the most popular brewed wines globally, and the quality of the wine is the core of the winery industry. The purpose of this paper is to study the statistical model for the rate of wine. The data fitting process goes through data structure analysis, model building, model, and assumption check. Three models we test are Linear Regression, Ordinal Linear Regression, and Random Forest. Based on the accurate rate and some error measurements, we select the random forest as the final model. Low interpretability is the most critical limitation in the random forest model, which can be partially solved by the importance of variables provided by the random forest. Considering our model user could be wineries and wine collectors, we think the high accuracy and ranking of variables brought by a random forest make it the most optimized model for these datasets.

Introduction

Wine is one type of alcoholic drink made from grape juice by fermentation. Grape s often used to make wine since it contains an excellent ratio of glucose and fructose. Wine is the world's largest and most popular monosaccharide brewed wine. In order to define the qualities, many countries issued legal names of wine. Several factors contribute to the rate of wine. Some essential wine characteristics are acidity, sweetness, alcohol, etc. In this report, we would like to test the relationship between different factors' quality according to a reliable data set. The result can help wineries test the qualities of batches of wine and reference the wine collector when they are deciding whether they should buy the wine.

Data Description

- The data is created by: Paulo Cortez (Univ. Minho), Antonio Cerdeira, Fernando Almeida, Telmo Matos and Jose Reis (CVRVV) @ 2009
- Two datasets in this study. They share same data structure but one for white wine and the other is for red wine. We use the red wine for this study
- One categorical ordinal dependent variable – rate for wine (Total 1599 observations)
- Eleven independent continuous variables – different factors potentially have impact on the rate of wine
    - fixed acidity: most acids involved with wine or fixed or non-volatile (do not evaporate readily)
    - volatile acidity: steam distillable acids present in wine, primarily acetic acid but also lactic, formic, butyric, and propionic acids.
    - citric acid: It occurs naturally in citrus fruits
    - residual sugar: from natural grape sugars leftover in a wine after the alcoholic fermentation finishes
    - chlorides: a chemical element

- free sulfur dioxide: a chemical compound is critical in wine
- total sulfur dioxide: a chemical compound is critical in wine
- density: the mass per unit volume of wine
- pH: a measure of the concentration of free hydrogen ions in solution
- sulphates: a chemical compound is critical in wine
- alcohol: the amount of ethanol in a given volume of liquid

- Details for the variables:

	Min	1st Quantile	Median	3rd Quantile	Max
quality	3.000	5.000	6.000	6.000	8.000
fixed acidity	4.60	7.10	7.90	9.20	15.90
volatile acidity	0.1200	0.3900	0.5200	0.6400	1.5800
citric acid	0.000	0.090	0.260	0.420	1.000
residual sugar	0.900	1.900	2.200	2.600	15.500
chlorides	0.01200	0.07000	0.07900	0.09000	0.61100
free sulfur dioxide	1.00	7.00	14.00	21.00	72.00
total sulfur dioxide	6.00	22.00	38.00	62.00	289.00
density	0.9901	0.9956	0.9968	0.9978	1.0037
pH	2.740	3.210	3.310	3.400	4.010
sulphates	0.3300	0.5500	0.6200	0.7300	2.0000
alcohol	8.40	9.50	10.20	11.10	14.90

Table 1. Data structure for red wine dataset

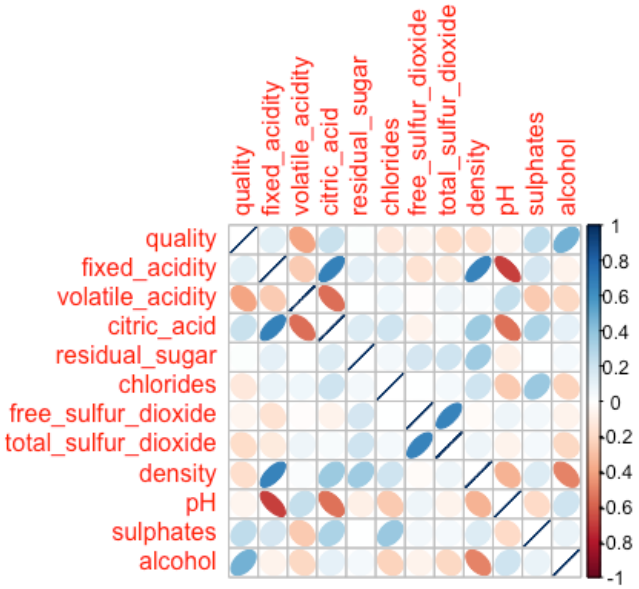


Figure 1. Correlation among variables for red wine

This graph is showing the relationship between variables. The color blue represents a positive

relationship, and red represents a negative relationship. The darker the color is, the closer the paired variables' relationship, and the darker color comes with a more elliptic shape displayed in the graph. By checking the shape for citric acid and fixed acidity, it shows dark blue and sharp oval. This represents they have a positive linear relationship. Conversely, the fixed acidity and pH have a negative correlation due to a dark red and sharp elliptic shape.

Testing Models

Data cleaning

- Remove NA records
- Remove unreasonable records
- Split the data into training and test data
- Training data is used for model fitting
- Test data is used to check the model accuracy

Model 1 – Linear Regression with Stepwise Selection

Procedure

- Use the stepwise selection to choose a subset of variables which minimizes the AIC of models, and we choose the predict variables: volatile acidity, total sulfur dioxide, chlorides, free sulfur dioxide, pH, sulphates and alcohol.
- End with the assumption check

Output

```
Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.4147750  0.4796417   9.204  < 2e-16 ***
volatile_acidity     -0.9943632  0.1238015  -8.032 2.54e-15 ***
total_sulfur_dioxide -0.0031637  0.0008344  -3.792 0.000158 ***
chlorides            -2.2489874  0.5074843  -4.432 1.03e-05 ***
free_sulfur_dioxide   0.0043384  0.0026079   1.664 0.096491 .
pH                   -0.4474661  0.1403266  -3.189 0.001471 **
sulphates             0.9314772  0.1331721   6.995 4.71e-12 ***
alcohol               0.2759894  0.0202163  13.652  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6414 on 1058 degrees of freedom
Multiple R-squared:  0.3533,    Adjusted R-squared:  0.349
F-statistic: 82.56 on 7 and 1058 DF,  p-value: < 2.2e-16
```

Important figures: AIC = 2088.405

Interpretation:
The regression equation is Quality = 4.415 – 0.994* volatile acidity – 0.00316* total sulfur dioxide – 2.249* chlorides + 0.00434 * free sulfur dioxide – 0.447*pH + 0.931*sulphates +0.276*alcohol

Take the alcohol as an example, every increased one unit of alcohol will lead to 0.2759894 unit increase in quality. Other variables have their coefficient for contributing to the quality of wine.

Model Check

By applying the model on the 1/3 test data, we can compare it with the actual value.
Here is the comparison between predicted value and actual value:

Actual\Predicted	1	2	3	4	5	6	7	8
1	**0**	0	0	0	0	0	0	0
2	0	**0**	0	0	0	0	0	0
3	0	0	**0**	1	3	0	0	0
4	0	0	0	**2**	8	7	0	0
5	0	0	0	0	**168**	53	2	0
6	1	0	0	0	60	**152**	8	0
7	0	0	0	0	2	47	**9**	0
8	0	0	0	0	0	7	2	**0**

Table 2. Actual response variable VS. Predicted response variable from Linear Regression

The diagonal highlighted in red displays the number of the correct prediction made by the linear regression model. This model underestimates the rate for those actual rated at six and overestimate those with the actual rate at 5. The main reason could be the majority of our test data gather around 5-6, which is hard for the linear model to predict a precise discrete output.

Here are some important figures we use to access the model:

Accurate rate	61.914%
MSPE/SD(y) [notes 1]	0.8862049
MAPE [notes 2]	0.4240150

The accurate rate is around 62% at a moderate level. The MSPE/Standard deviation of testing y is relatively higher than what we can accept. Lower MAPE will be more favorable since it is an indicator about how off the predication can be. 42.4% is a moderate figure for MAPE.

Assumptions check for linear regression

- Linearity assumption

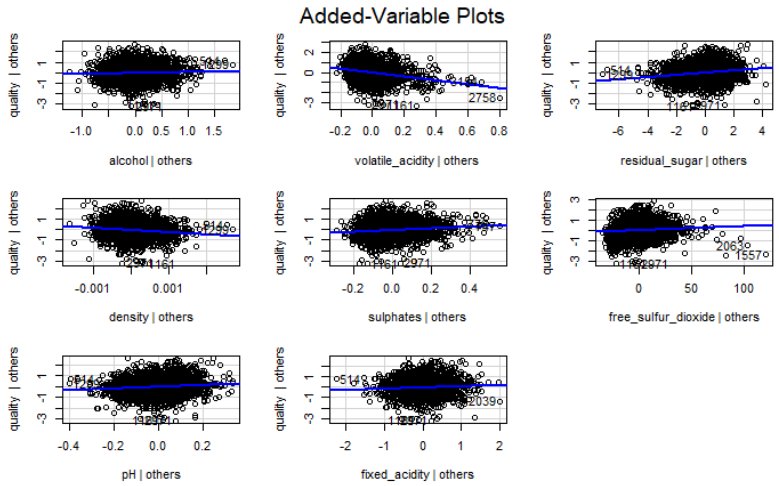


Figure 2. Added variable plot

By checking the plot, the linearity assumption holds.

- Residual plot – independent assumption

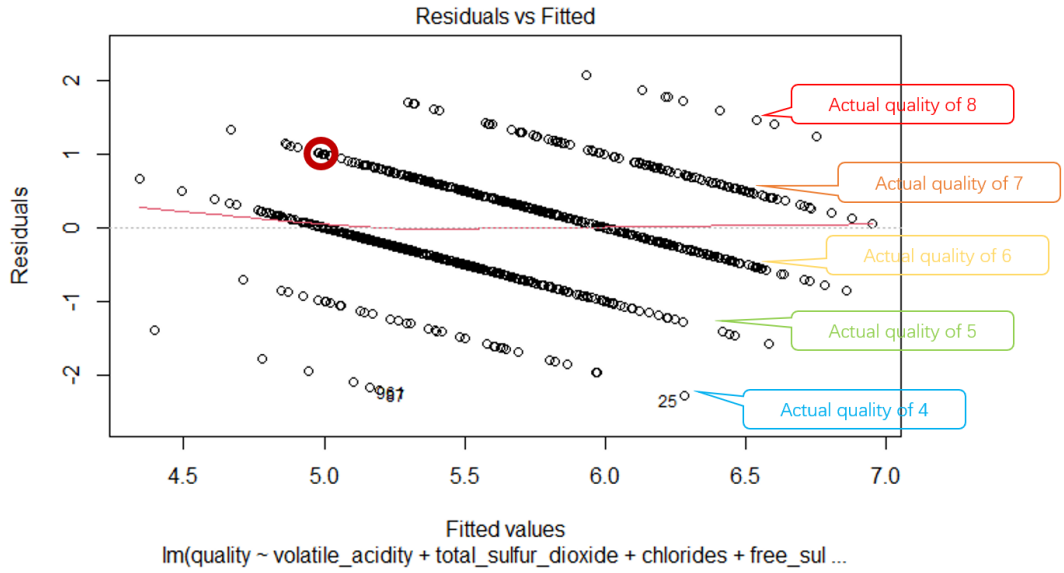


Figure 3. Residual VS. Fitted value for Linear Regression

As we can see that there are 3 obvious straight lines with negative slope. For instance, the line in the middle marked as orange has equation: residual + fitted value = 6. It represents the prediction values that have actual quality 6. The dark red point, it has a fitted value 5 and actual value 6. Therefore, the residual is 1. It is clear that the randomness assumption is not violated. And the majority of the fitted value are in the range of 5 to 7 which meets the origin data set.

- Residual plot – Normality assumption

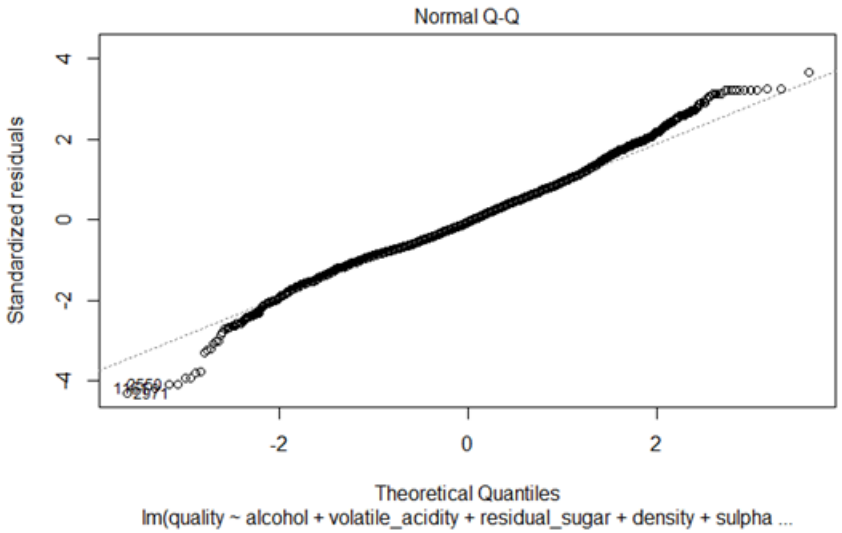


Figure 4. QQ plot for Linear Regression Model

Majority of points lay around the standard line, so the normality assumption holds.

- VIF for collinearity

	VIF
volatile acidity	1.243978
total sulfur dioxide	1.898842
chlorides	1.416292
free sulfur dioxide	1.834910
pH	1.279484
sulphates	1.362732
alcohol	1.218972

Table 3. VIF figures for Linear Regression Model

None of the variable has a VIF higher than 10, so absence of multicollinearity assumption holds.

Model 2 - Ordinal logistic regression (OLR)

Procedure

- Since the dependent variable is both categorical and ordered, OLR is a potential better candidate for the model fitting
- It is a regression process that make prediction on log (odds of an event), which can be seen as a transformation of the cumulative probability for each rate level.
- General procedure is similar to normal linear regression after transformation: fit the data first and remove insignificant variables by large p-value through hypothesis test
- Use the remained variables to do prediction and model check

Assumptions in OLR

1. The dependent variable is ordered.
2. One or more of the independent variables are either continuous, categorical or ordinal.
3. No multi-collinearity
4. Proportional odds

Output

I). Fitting without removal of any predictors

	Value	Std. Error	t value	p value
fixed_acidity	0.11392945	0.060442204	1.884932	0.0594390
volatile_acidity	-3.37256416	0.496254855	-6.796033	0.0000000
citric_acid	-0.61964600	0.556613802	-1.113242	0.2656044
residual_sugar	0.06236876	0.048733663	1.279788	0.2006197
chlorides	-5.82402743	1.725178119	-3.375899	0.0007357
free_sulfur_dioxide	0.01217429	0.008214645	1.482022	0.1383344
total_sulfur_dioxide	-0.01001665	0.002852729	-3.511253	0.0004460
density	-35.85335978	1.193836609	-30.032049	0.0000000
pH	-0.66164566	0.604343186	-1.094818	0.2735965
sulphates	2.88497627	0.423213591	6.816833	0.0000000
alcohol	0.83428526	0.071782070	11.622474	0.0000000
3|4	-35.12280313	1.221767518	-28.747534	0.0000000
4|5	-32.95504279	1.219785167	-27.017088	0.0000000
5|6	-29.21696611	1.225441124	-23.841999	0.0000000
6|7	-26.36599835	1.241495398	-21.237290	0.0000000
7|8	-23.22151769	1.284837718	-18.073502	0.0000000

Interpretation: The p-value of fixed acidity, citric acid, residual sugar, free sulfur dioxide and pH are all greater than 0.05. Therefore, we do not reject the Null hypothesis and remove the five independent variables before next move.

II). Fitting with removal of fixed acidity, citric acid, residual sugar, free sulfur dioxide and pH

	Value	Std. Error	t value	p value
volatile_acidity	-3.432570164	0.399755744	-8.586669	0.0000000
total_sulfur_dioxide	-0.007521269	0.002030923	-3.703374	0.0002128
chlorides	-5.632911970	1.616650265	-3.484311	0.0004934
density	37.394814305	0.455384616	82.116991	0.0000000
sulphates	2.838104518	0.418119293	6.787787	0.0000000
alcohol	0.857646572	0.067508088	12.704353	0.0000000
3|4	39.278492327	0.452458735	86.811215	0.0000000
4|5	41.445066100	0.455390648	91.009919	0.0000000
5|6	45.175144357	0.468896198	96.343593	0.0000000
6|7	47.993649646	0.514019165	93.369378	0.0000000
7|8	51.136900306	0.617776331	82.775752	0.0000000

Important figures: AIC = 2066.34 (Decreased from 2088.405 in Linear Regression)

Interpretation:
The formula used in proportional odds is shown below,

$$logit[\mathbb{P}(Y \leq j)] = \alpha_j - \sum_{i=1}^{M} \beta_i X_i$$

Where j = 1, 2, ..., J-1 and i = 1, ..., M. J is the total number of categories of dependent variable and M is the number of independent variables.
In this case, j = 1 refers to "Quality is 3", j = 2 refers to "Quality is 4" and so on. And M=1 refers to "fixed acidity", M = 2 refers to "volatile acidity" and so on.
For instance, the probability corresponding to "Quality is 3" can be calculated as

$$\begin{aligned} logit[\mathbb{P}(Quality \leq 3)] &= 39.27849 - [(-3.43257 * 0.7) + (-0.007521 * 34) \\ &\quad + (-5.632911 * 0.076) + (37.3948143 * 0.9978) \\ &\quad + (2.8381045 * 0.56) + (0.85764657 * 9.4)] = -4.59865 \\ \mathbb{P}(Quality = 3) &= \mathbb{P}(Quality \leq 3) = \frac{exp(-4.59865)}{1 + exp(-4.59865)} = 0.00996 \\ logit[\mathbb{P}(Quality \leq 4)] &= 41.445066 - [(-3.43257 * 0.7) + (-0.007521 * 34) \\ &\quad + (-5.632911 * 0.076) + (37.3948143 * 0.9978) \\ &\quad + (2.8381045 * 0.56) + (0.85764657 * 9.4)] = -2.43208 \\ \mathbb{P}(Quality \leq 4) &= \frac{exp(-2.43208)}{1 + exp(-2.43208)} = 0.08075 \\ &\quad \ldots\ldots \end{aligned}$$

$$\begin{aligned} &\mathbb{P}(Quality = 3) = 0.00996 \\ &\mathbb{P}(Quality = 4) = \mathbb{P}(Quality \leq 4) - \mathbb{P}(Quality \leq 3) = 0.08075 - 0.00996 = 0.07079 \\ &\mathbb{P}(Quality = 5) = 0.70474 \\ &\mathbb{P}(Quality = 6) = 0.19847 \\ &\quad \ldots \end{aligned}$$

We find that the probability of "Quality = 5" is the highest among all the probabilities. Therefore, we predict that the quality of this wine is 5.

Model Check

By applying the model on the 1/3 test data, we can compare it with the actual value.
Here is the comparison between predicted value and actual value:

Actual\Predicted	3	4	5	6	7	8
3	**0**	0	3	0	0	0
4	0	**0**	12	4	0	0
5	0	0	**177**	45	3	0
6	0	0	70	**112**	13	0
7	0	0	4	45	**11**	0
8	0	0	0	7	2	**0**

Table 4. Actual response variable VS. Predicted response variable from Ordinal Logistic Regression

This table's diagonal is where OLR predicts the correct value compared to what is stored in test data. This table looks similar compare to Table 2 in the linear regression model check section. This model has a close issue with linear regression since it is under the regression model. It has a hard time estimating the output correctly from rate 5-6.

Here are some important figures we use to evaluate the model:

Accurate rate	60.975%
MSPE/SD(y)	0.8845744
MAPE	0.4296435

This accurate rate is approximately the same what the linear regression has, and the magnitude of MSPE/Standard deviation of testing y and MAPE does not have a remarkable improvement compared to the linear regression.

Model 3 - Random forest (one of the bootstrap aggregating)

Procedure

- The random forest method first creates b bootstrap samples from the original training dataset (We select b=50).
- Then, we construct a decision tree for each bootstrap sample using recursive binary splitting. Moreover, a subset sampled randomly of k variables are considered at each split. We select k=4 by following the thumbs up rule (minimize the Out-Of-Bag error) that the best k for classification should be $\sim\sqrt{p}$ where p is 11 in our dataset.
- After that, the model predicts the response a new observation using the mode across all b trees.
- We run the random forest 50 times and select the most frequent response of each observation to represent our final prediction.

Assumptions in Random Forrest

And as we have no probabilistic model, but just binary split, there is any assumption.

Output

There is no explicit form generated from the random forest model fitting

Model Check

By applying the model on the 1/3 test data, we can compare it with the actual value.
Here is the comparison between predicted value and actual value:

Actual\Predicted	3	4	5	6	7	8
3	**0**	0	3	0	0	0
4	0	**0**	13	3	0	0
5	0	0	**189**	35	1	0
6	0	0	42	**163**	15	0
7	0	0	5	27	**28**	0
8	0	0	0	4	4	**1**

Table 5. Actual response variable VS. Predicted response variable from Random Forest

The diagonal in this table gets larger compare to the LR and OLR model even it still has some issues with identifying the correct rate between. However, since we set the prediction type in the random forest as a factor and list, the discrete output will increase our prediction accuracy, which can be proved by the bellowing figures.

Here are some important figures we use to access the model:

Accurate rate	71.482%
MSPE/SD(y)	0.7599145
MAPE	0.3151970

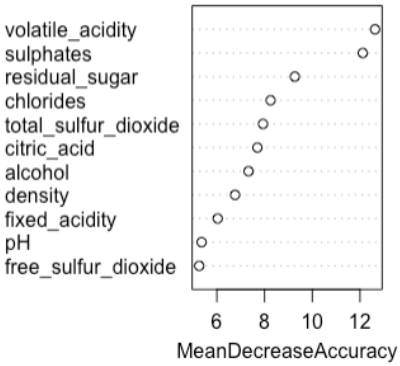
Recall the figures in the OLR and LR section, the accurate rate increases from ~61% to above 71%. The MSPE/Standard deviation of testing y decreases from ~88% to 76%, and MAPE falls around ~ 10% which is a big improvement regarding the error check

Model comparison and Final selection

	Linear Regression	Ordinal Logistic Regression	Random Forest
Accurate rate	61.914%	60.975%	71.482%
MSPE/SD(y)	0.8862049	0.8845744	0.7599145
MAPE	0.4240150	0.4296435	0.3151970
Advantages	a). Linear representation is easier to make interpretation b). High accuracy for qualitative response variable c). Relatively fast training speed	a). Relatively easy to make interpretation b). Handle the ordinal categorical response variable c). Relatively fast training speed	a). Great with high dimensionality b). Robust to outliers and non-linear data c). Handles unbalanced data d). Low bias and moderate variance due to average over all trees in use
Drawbacks	a). Limitation to qualitative response variable b). Assumption required for data	a). Assumption required for data b). Hard to handle complex relationship	a). Less interpretability compares to regression model b). Large tree will take up lots of memory c). Large number of tree will slow the speed of model run

Table 6. Model comparison

- Based on the structure of our data, the response variable is categorical.
- Ordinal Logistic Regression and Random Forest will outperform the normal regression model
- Due to the high dimension and complex relationship in our data, Random Forrest outperforms the OLR
- With a higher accurate rate and the lower error indication (MSPE/SD(y) and MAPE), Random Forest becomes our final model selected
- For actual model using purpose, the random forest cannot provide the exact interpretation for the relationship between the response variable and predictors. But it can evaluate the importance of each variable. See below for details,



The graph displays how much the model accuracy decrease if we drop that variable. From this figure, the volatile acidity plays a critical role in wine rating while free sulfur is the least important one.

Cross Validation

Cross validation without the extra point

	Accurate rate	MSPE/SD(y)	MAPE
Trail 1	68.105%	0.81372	0.3546
Trail 2	67.917%	0.80667	0.3546
Trail 3	68.105%	0.77419	0.34146
Trail 4	67.167%	0.79884	0.35835
Trail 5	66.229%	0.84037	0.37523
Trail 6	69.981%	0.81797	0.34146
Trail 7	66.417%	0.84067	0.36773
Trail 8	66.792%	0.78587	0.35272
Trail 9	69.231%	0.84005	0.34334
Trail 10	68.668%	0.77506	0.33959
Average	67.861%	0.80934	0.35291

Table 7. Cross validation result for Random Forest

We run the model 10 times with random select training data from our source file and return the statistics showed above in the table to evaluate the model in each trial. It is clear that the result is stable overall ten trails, and the average remains around 68% for the accurate rate, 0.81 for MSPE/SD(y) and 0.35 for MAPE. The consistency among these trails indicated the stability and reliability of the random forest model.

Add Extra Point into Data

Point Selected:

	Red
quality	1
fixed acidity	4
volatile acidity	0.9
citric acid	0.2
residual sugar	1.8
chlorides	0.8
free sulfur dioxide	14
total sulfur dioxide	48
density	0.99
pH	3
sulphates	0.4
alcohol	9.5

Reason:

- The quality selected for the extra point is 1 since there is no point in dataset has rate 1
- The direction for other variables depend on their relationship with quality (Figure1 and Figure 2)
- The values for other variables is calculated based on their 1st quantile, median and 3rd quantile (Table 1 and Table 2)

Cross validation with the extra point

	Accurate rate	MSPE/SD(y)	MAPE
Trail 1	67.167%	0.78272	0.35460
Trail 2	69.606%	0.79397	0.33396
Trail 3	63.977%	0.87386	0.41276
Trail 4	66.229%	0.84923	0.29024
Trail 5	69.418%	0.79608	0.34522
Trail 6	65.854%	0.82825	0.38274
Trail 7	71.107%	0.76454	0.32458
Trail 8	68.480%	0.77124	0.34897
Trail 9	69.043%	0.78814	0.34334
Trail 10	66.041%	0.82213	0.37711
Average	67.692%	0.80702	0.36135

Table 8. Cross validation result for Random Forest after adding the point

By comparing the Table 7 and Table 8, it is clear that the potential outliers we select does not have significant impact. Since random forest is robust to outliers and non-linear data, it is reasonable that there is no big impact brought the added point.

Conclusion

After we go through the data check, model testing, check, and comparison, Random Forest becomes the final model we select for this dataset. Cross-validation is performed on the final model to check the consistency and reliability approved to be valid. Because random forest does an excellent job handling the outliers and unbalanced data, the extra point we add does not significantly impact our analysis.

Rating the wine is the primary purpose we have for this dataset, which weakens the low interpretability of random forest. If the potential model user wants to know which factor plays a more critical role in wine rating, the random forest can provide the rank of importance for all variables in the model. Furthermore, the dataset for red wine is within an acceptable size range, so that the model running time is not a particular issue. Taking the accurate rate, error measurement, and limitations into consideration for this study, Random Forest is the best selection.

Appendix and Reference

Notes 1:
Mean Squared Prediction Error (MSPE)
The difference is that while MSE measures of an estimator's fit, the MSPE is a measure of a predictor's fit— or how well it predicts the true value.
SD(y)
Standard deviation of response variable in testing data

Notes 2:
The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a loss function for regression problems in machine learning. It usually expresses the accuracy as a ratio defined by the formula:

Reference:
https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706

https://christophm.github.io/interpretable-ml-book/limo.html

https://www.listendata.com/2014/11/random-forest-with-r.html#Popularity-of-Random-Forest-Algorithm

https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/time-series/how-to/trend-analysis/interpret-the-results/all-statistics-and-graphs/#:~:text=equals%20%CE%B2%202.-,MAPE,forecast%20is%20off%20by%205%25.

https://towardsdatascience.com/implementing-and-interpreting-ordinal-logistic-regression-1ee699274cf5

https://en.wikipedia.org/wiki/Wine

https://medium.com/evangelinelee/ordinal-logistic-regression-on-world-happiness-report-221372709095

Code

Project_STAT350_Red

Crystal Fan and Iker Guo

29/09/2020

```
rm(list=ls())
library(readr)
library(carData)
library(MASS)
library(randomForest)
library(caret)
library(tidyverse)
library(corrplot)
library(car)
```

#read the data

```
winequality_red <- read_csv("winequality-red.csv")
Data_red <- na.omit(winequality_red )
train_size_red <- floor(2/3*nrow(Data_red))
train_index_red <- sample(seq_len(nrow(Data_red)),size = train_size_r
ed)
Data_red_training <- Data_red[train_index_red,]
Data_red_test <- Data_red[-train_index_red,]
```

#draw correlation graph

```
summary(winequality_red)
head(winequality_red)

# Correlation panel
panel.cor <- function(x, y){
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- round(cor(x, y), digits=2)
    txt <- r
    cex.cor <- 0.8/strwidth(txt)
    text(0.5,0.5,txt)
}
```

```
# Customize upper panel
lower.panel<-function(x, y){
  points(x,y, pch = 10)
}

# Create the plots
pairs(winequality_red[,2:ncol(winequality_red)], upper.panel = pane
l.cor,lower.panel = lower.panel)

G = cor(winequality_red)
corrplot(G, method="ellipse")
```

#Start with the Ordinal Logistic Regression

```
Check_p <- function (model){
  summary_table <- coef(summary(model))
  pval <- pnorm(abs(summary_table[, "t value"]),lower.tail = FALSE)*
2
  summary_table <- cbind(summary_table, "p value" = round(pval,7))
  summary_table
}

Pred_OLR <- function (model,test_data) {
  predict_matrix <- as.matrix(round(predict(model,test_data[,2:ncol(t
est_data)],type = "p"), 3))
  Prediction <- NULL
  for (i in 1:nrow(Data_red_test)){
    Prediction[i] = which.max(predict_matrix[i,])+2
  }
  Prediction
}

Check_accurate_rate <- function (test_data,final_result) {
  difference_matrix = final_result - test_data[,1]
  count = 0
  for (j in 1:nrow(difference_matrix)){
    if (difference_matrix[j,] == 0){
      count = count + 1
    }
  }
  count/nrow(difference_matrix)
}

Data_red_test_OLR = Data_red_test
```

```r
Data_red_training_OLR = Data_red_training

#Factorize the dependent variable and scale the variable
Data_red_training_OLR$quality = factor(Data_red_training$quality, le
vels = c("3","4","5","6","7","8"),ordered = TRUE)
OLR_red <- polr(formula = quality ~ ., data = Data_red_training_OLR,
Hess = TRUE)
Check_p(OLR_red)
summary(OLR_red)

# Prediction before reduction
OLR_check_red <- Pred_OLR(OLR_red,Data_red_test_OLR)
paste("The accurate rate for Ordinal Logistic Regression Model is ",r
ound(Check_accurate_rate(Data_red_test,OLR_check_red),5))

R_sq_OLR = R2(Data_red_test$quality,OLR_check_red)
RMSPE_OLR = RMSE(Data_red_test$quality,OLR_check_red)
MAPE_OLR = MAE(Data_red_test$quality,OLR_check_red)
print(c(R_sq_OLR,RMSPE_OLR/sd(Data_red_test$quality),MAPE_OLR))

table(t(Data_red_test[,1]),OLR_check_red)

# Remove the predictor
# fixed acidity, citric acid, residual sugar and free sulfur dioxide
OLR_red_reduce <- polr(formula = quality ~ volatile_acidity+total_sul
fur_dioxide+chlorides+density+pH +sulphates+alcohol , data = Data_red
_training_OLR, Hess = TRUE)
Check_p(OLR_red_reduce)
summary(OLR_red_reduce)

# Prediction After reduction
OLR_check_red_reduced <- Pred_OLR(OLR_red_reduce,Data_red_test_OLR)
Accurate_rate_red = Check_accurate_rate(Data_red_test,OLR_check_red_
reduced)
paste("The accurate rate for Reduced Ordinal Logistic Regression Mode
l is ",round(Check_accurate_rate(Data_red_test,OLR_check_red_reduce
d),5))

RMSPE_OLR_redu = RMSE(Data_red_test$quality,OLR_check_red_reduced)
MAPE_OLR_redu = MAE(Data_red_test$quality,OLR_check_red_reduced)
print(c(RMSPE_OLR_redu/sd(Data_red_test$quality),MAPE_OLR_redu))

table(t(Data_red_test[,1]),OLR_check_red_reduced)
```

#Random forrest

```
Data_red_training_RF <- Data_red_training
Data_red_training_RF$quality <- as.factor(Data_red_training_RF$quali
ty)
Data_red_training_RF$quality = factor(Data_red_training_RF$quality,
levels = c("3","4","5","6","7","8"),ordered = TRUE)

Data_red_test_RF <- Data_red_test
Data_red_test_RF$quality  = NA
Data_red_test_RF$quality <- as.factor(Data_red_test_RF$quality)
Data_red_test_RF$quality <- factor(Data_red_test_RF$quality, levels =
 c("3","4","5","6","7","8"),ordered = TRUE)

#determine the number of loops
n = 50

#determine the ntree
n_tree = 50

for(i in 1:n)
{
  RF_red <- randomForest(quality~.,data = droplevels(Data_red_trainin
g_RF), ntree = n_tree, nPerm = 10,mtry = 4)
  pred_RF_red <- predict(RF_red,Data_red_test_RF)
  Data_red_test_RF <- cbind(Data_red_test_RF, as.data.frame(pred_RF_r
ed))
  colnames(Data_red_test_RF)[ncol(Data_red_test_RF)] <- paste0("Predi
ction",i)
}

#function to get mode of one row
getmode <- function(v)
{
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

#get the predicted value
RF_pred_red <- as.data.frame(Data_red_test_RF[,13:(13+n-1)])

# get the mode of the predicted data
RF_final_pred_red = NULL
for(i in 1:nrow(RF_pred_red))
```

```
{
  RF_final_pred_red = c(RF_final_pred_red,as.numeric(getmode(as.vecto
r(unlist(RF_pred_red[i,])))))
}

RF_check_red = as.matrix(RF_final_pred_red)

paste("The accurate rate for Random Forrest Model is ",round(Check_ac
curate_rate(Data_red_test,RF_check_red),5))

R_sq_RF = R2(Data_red_test$quality,RF_check_red)
RMSPE_RF = RMSE(Data_red_test$quality,RF_check_red)
MAPE_RF = MAE(Data_red_test$quality,RF_check_red)
print(c(R_sq_RF,RMSPE_RF/sd(Data_red_test$quality),MAPE_RF))

table(t(Data_red_test[,1]),RF_check_red)
```

Stepwise

```
#stepwise selection
LM_red_full = lm(formula = quality ~., data = Data_red_training)
LM_red_null = lm(formula = quality ~1, data = Data_red_training)

step(LM_red_null,data = Data_red_training,scope = list(upper = LM_red
_full),direction = "both")

# fixed acidity, citric acid, residual sugar and density
LM_step_red <- lm(formula = quality ~ volatile_acidity+total_sulfur_d
ioxide+chlorides+free_sulfur_dioxide+pH +sulphates+alcohol, data = Da
ta_red_training)

avPlots(LM_step_red)

#Assumption check for the col-linearity
vif(LM_step_red)
X_matrix_LM_step_red = as.matrix(Data_red_training[,-c(1,4,6,7)])
XX_LM_step_red=t(X_matrix_LM_step_red)%*%X_matrix_LM_step_red
lambda_LM_step_red = eigen(XX_LM_step_red)$values
cond_number_LM_step_red=max(lambda_LM_step_red)/min(lambda_LM_step_r
ed)
indices_LM_step_red=max(lambda_LM_step_red)/lambda_LM_step_red

plot(LM_step_red)
```

```r
#Prediction; Model check
LM_step_check_red = round(as.matrix(predict(LM_step_red,Data_red_tes
t)),0)
paste("The accurate rate for Step-Wise is ",round(Check_accurate_rate
(Data_red_test,LM_step_check_red),5))

R_sq_LM_red = R2(Data_red_test$quality,LM_step_check_red)
RMSPE_LM_red = RMSE(Data_red_test$quality,LM_step_check_red)
MAPE_LM_red = MAE(Data_red_test$quality,LM_step_check_red)
print(c(R_sq_LM_red,RMSPE_LM_red/sd(Data_red_test$quality),MAPE_LM_r
ed))

table(t(Data_red_test[,1]),LM_step_check_red)

rm(list=ls())
library(readr)
library(carData)
library(MASS)
library(randomForest)
library(caret)
library(tidyverse)
library(corrplot)
library(car)

winequality_red <- read_csv("winequality-red.csv")
Data_red <- na.omit(winequality_red )
train_size_red <- floor(2/3*nrow(Data_red))

getmode <- function(v)
{
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

Check_accurate_rate <- function (test_data,final_result) {
  difference_matrix = final_result - test_data[,1]
  count = 0
  for (j in 1:nrow(difference_matrix)){
    if (difference_matrix[j,] == 0){
      count = count + 1
    }
  }
  count/nrow(difference_matrix)
}
```

```r
#determine the number of loops
n = 50

#determine the ntree
n_tree = 50

#determine cross trail
n_cross = 10

Ave_accurate_red = 0
Ave_MSPE_red = 0
Ave_MAPE_red = 0

for (trail_red in 1:n_cross){
  train_index_red <- sample(seq_len(nrow(Data_red)),size = train_size
_red)
  Data_red_training <- Data_red[train_index_red,]
  Data_red_test <- Data_red[-train_index_red,]
  Data_red_training_RF <- Data_red_training
  Data_red_training_RF$quality <- as.factor(Data_red_training_RF$qual
ity)
  Data_red_training_RF$quality = factor(Data_red_training_RF$quality,
 levels = c("3","4","5","6","7","8","9"),ordered = TRUE)

  Data_red_test_RF <- Data_red_test
  Data_red_test_RF$quality  = NA
  Data_red_test_RF$quality <- as.factor(Data_red_test_RF$quality)
  Data_red_test_RF$quality <- factor(Data_red_test_RF$quality, levels
 = c("3","4","5","6","7","8","9"),ordered = TRUE)

  for(i in 1:n)
  {
    RF_red <- randomForest(quality~.,data = droplevels(Data_red_train
ing_RF), ntree = n_tree, nPerm = 10,mtry = 4)
    pred_RF_red <- predict(RF_red,Data_red_test_RF)
    Data_red_test_RF <- cbind(Data_red_test_RF, as.data.frame(pred_RF
_red))
    colnames(Data_red_test_RF)[ncol(Data_red_test_RF)] <- paste0("Pre
diction",i)
  }

  RF_pred_red <- as.data.frame(Data_red_test_RF[,13:(13+n-1)])

  # get the mode of the predicted data
  RF_final_pred_red = NULL
```

```
  for(j in 1:nrow(RF_pred_red))
  {
    RF_final_pred_red = c(RF_final_pred_red,as.numeric(getmode(as.vec
tor(unlist(RF_pred_red[j,])))))
  }

  RF_check_red = as.matrix(RF_final_pred_red)

  RMSPE_RF = RMSE(Data_red_test$quality,RF_check_red)
  MAPE_RF = MAE(Data_red_test$quality,RF_check_red)

  Ave_accurate_red = Ave_accurate_red + round(Check_accurate_rate(Dat
a_red_test,RF_check_red),5)
  Ave_MSPE_red = Ave_MSPE_red + RMSPE_RF/sd(Data_red_test$quality)
  Ave_MAPE_red = Ave_MAPE_red + MAPE_RF

  print(paste("Trail",trail_red,"The accurate rate for Random Forrest
 Model is ",round(Check_accurate_rate(Data_red_test,RF_check_red),
5),"The MSPE/Sd and MAPE are ",round(RMSPE_RF/sd(Data_red_test$qualit
y),5)," and ",round(MAPE_RF,5)))
}

print(paste("The average accurate rate, average MSPE/Sd and average M
APE are ",round(Ave_accurate_red/n_cross,5),round(Ave_MSPE_red/n_cro
ss,5)," and ",round(Ave_MAPE_red/n_cross,5)))

Add_point_red <- c(1, 4, 0.90, 0.20, 1.8, 0.8, 14, 48, 0.99, 3, 0.4,
9.50)
Ave_accurate_red_add = 0
Ave_MSPE_red_add = 0
Ave_MAPE_red_add = 0

for (trail_red_add in 1:10){
  train_index_red <- sample(seq_len(nrow(Data_red)),size = train_size
_red)
  Data_red_training_add <- data.frame(rbind(Data_red[train_index_re
d,],Add_point_red))

  Data_red_test <- Data_red[-train_index_red,]
  Data_red_training_RF_add <- Data_red_training_add
  Data_red_training_RF_add$quality <- as.factor(Data_red_training_RF_
add$quality)
  Data_red_training_RF_add$quality = factor(Data_red_training_RF_add
$quality, levels = c("1","2","3","4","5","6","7","8","9"),ordered = T
RUE)
```

```
  Data_red_test_RF <- Data_red_test
  Data_red_test_RF$quality  = NA
  Data_red_test_RF$quality <- as.factor(Data_red_test_RF$quality)
  Data_red_test_RF$quality <- factor(Data_red_test_RF$quality, levels
 = c("1","2","3","4","5","6","7","8","9"),ordered = TRUE)

  for(i in 1:n)
  {
    RF_red <- randomForest(quality~.,data = droplevels(Data_red_train
ing_RF_add), ntree = n_tree, nPerm = 10,mtry = 4)
    pred_RF_red <- predict(RF_red,Data_red_test_RF)
    Data_red_test_RF <- cbind(Data_red_test_RF, as.data.frame(pred_RF
_red))
    colnames(Data_red_test_RF)[ncol(Data_red_test_RF)] <- paste0("Pre
diction",i)
  }

  RF_pred_red <- as.data.frame(Data_red_test_RF[,13:(13+n-1)])

  # get the mode of the predicted data
  RF_final_pred_red = NULL
  for(j in 1:nrow(RF_pred_red))
  {
    RF_final_pred_red = c(RF_final_pred_red,as.numeric(getmode(as.vec
tor(unlist(RF_pred_red[j,])))))
  }

  RF_check_red = as.matrix(RF_final_pred_red)

  RMSPE_RF = RMSE(Data_red_test$quality,RF_check_red)
  MAPE_RF = MAE(Data_red_test$quality,RF_check_red)

  Ave_accurate_red_add = Ave_accurate_red_add + round(Check_accurate_
rate(Data_red_test,RF_check_red),5)
  Ave_MSPE_red_add = Ave_MSPE_red_add + RMSPE_RF/sd(Data_red_test$qua
lity)
  Ave_MAPE_red_add = Ave_MAPE_red_add + MAPE_RF

  print(paste("Trail",trail_red_add,"The accurate rate for Random For
rest Model after adding the point is ",round(Check_accurate_rate(Data
_red_test,RF_check_red),5),"The MSPE/Sd and MAPE are ",round(RMSPE_RF
/sd(Data_red_test$quality),5)," and ",round(MAPE_RF,5)))
}
```

```
print(paste("The average accurate rate, average MSPE/Sd and average MAPE after adding point are ",round(Ave_accurate_red_add/n_cross,5),round(Ave_MSPE_red_add/n_cross,5)," and ",round(Ave_MAPE_red_add/n_cross,5)))

winequality_red_rf <- winequality_red
winequality_red_rf$quality = factor(winequality_red_rf$quality, levels = c("3","4","5","6","7","8"),ordered = TRUE)

rf_import <- randomForest(quality~.,winequality_red_rf,mtry=4, importance=TRUE,ntree=10)

importance(rf_import)
varImpPlot(rf_import,2)
```