Final Report

Data Mining - DNSC 6279
Leqi Yin, Qunzhe Ding, Seungheon Han

Introduction

The dataset we are using for the project is called wine quality dataset. It is related to the red and white Portuguese "Vinho Verde" wine, which we found from the UCI machine learning lab. Due to the privacy issue, the grape types, wine brand and selling price are not being provided.

Our data totally has 6497 observations and 14 columns; 12 variables are the basic attributes of the wine, they are fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, PH, sulphates, alcohol degree and color.

Beside these objective attributes, we also have two subjective attributes, which are quality and goodness of the wine. since Goodness is repeating with quality theoretically, so we can't use it as a predictive variable, and we decided to remove it. As a result, our dependent variable is the quality.

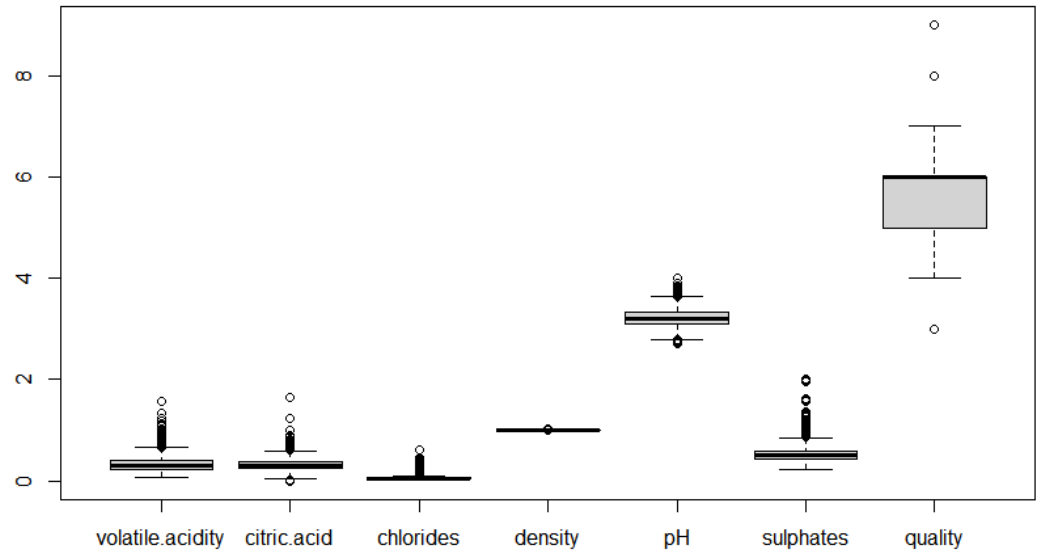
Data preparation

we have three processes in this part: first, it's data cleaning, we must make sure There is no null and missing value in our dataset, after checking and there is no any NA values in our data: after testing, we found our data doesn't contain this problem.

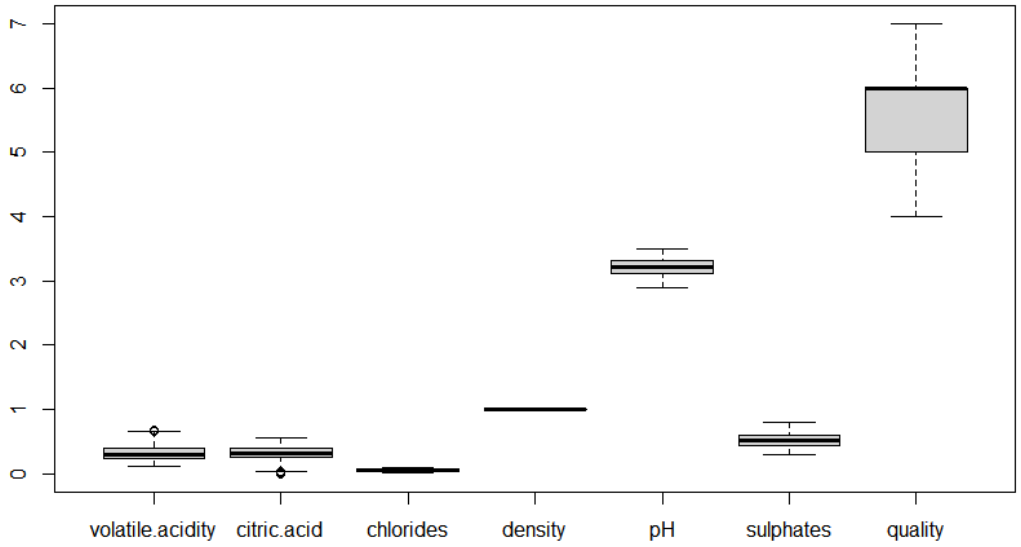
```
> colSums(is.na(wine))
      fixed.acidity    volatile.acidity         citric.acid      residual.sugar           chlorides
                  0                   0                   0                   0                   0
 free.sulfur.dioxide total.sulfur.dioxide             density                  pH           sulphates
                  0                   0                   0                   0                   0
            alcohol             quality                good               color
                  0                   0                   0                   0
> sum(is.na(wine))
[1] 0
> wine <- na.omit(wine)
```

Then We rescaled all the continuous predictors to make sure all of the predictors have the equal weight on the prediction of the dependent variable.

The last part of data preparation is the outlier replacement, We select some variable and using boxplot to display their observations' distribution.



According to the graph below, we can see except density, other variables all have many outliers which we thought it will have the negative impact on our modeling process, so we decide to use the 95% upper and 1% lower quantile to replace the extreme numbers and then we the draw the boxplot again.



we can see the outlier situation looks better and more appropriate.
At the end, we split our data into training and testing parts by 80% and 20% respectively, which we will utilize in the data modeling process.

Data Exploration

We first explored our data by using the summary function;

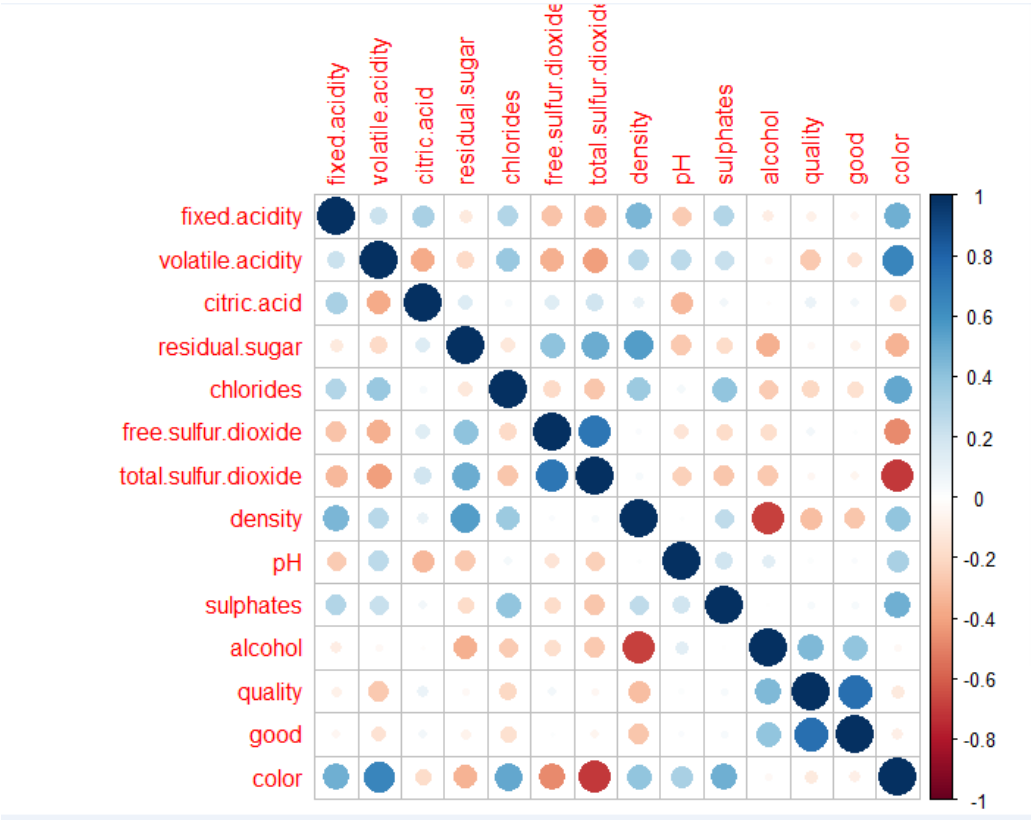
```
> summary(wine)
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar     chlorides       free.sulfur.dioxide
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.   :0.00900   Min.   :  1.00
 1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.2500   1st Qu.: 1.800   1st Qu.:0.03800   1st Qu.: 17.00
 Median : 7.000   Median :0.2900   Median :0.3100   Median : 3.000   Median :0.04700   Median : 29.00
 Mean   : 7.215   Mean   :0.3397   Mean   :0.3186   Mean   : 5.443   Mean   :0.05603   Mean   : 30.53
 3rd Qu.: 7.700   3rd Qu.:0.4000   3rd Qu.:0.3900   3rd Qu.: 8.100   3rd Qu.:0.06500   3rd Qu.: 41.00
 Max.   :15.900   Max.   :1.5800   Max.   :1.6600   Max.   :65.800   Max.   :0.61100   Max.   :289.00

 total.sulfur.dioxide    density             pH          sulphates         alcohol      quality  good
 Min.   :  6.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200   Min.   : 8.00   3:  30   0:5220
 1st Qu.: 77.0        1st Qu.:0.9923   1st Qu.:3.110   1st Qu.:0.4300   1st Qu.: 9.50   4: 216   1:1277
 Median :118.0        Median :0.9949   Median :3.210   Median :0.5100   Median :10.30   5:2138
 Mean   :115.7        Mean   :0.9947   Mean   :3.219   Mean   :0.5313   Mean   :10.49   6:2836
 3rd Qu.:156.0        3rd Qu.:0.9970   3rd Qu.:3.320   3rd Qu.:0.6000   3rd Qu.:11.30   7:1079
 Max.   :440.0        Max.   :1.0390   Max.   :4.010   Max.   :2.0000   Max.   :14.90   8: 193
                                                                                       9:   5

   color
 red  :1599
 white:4898
```

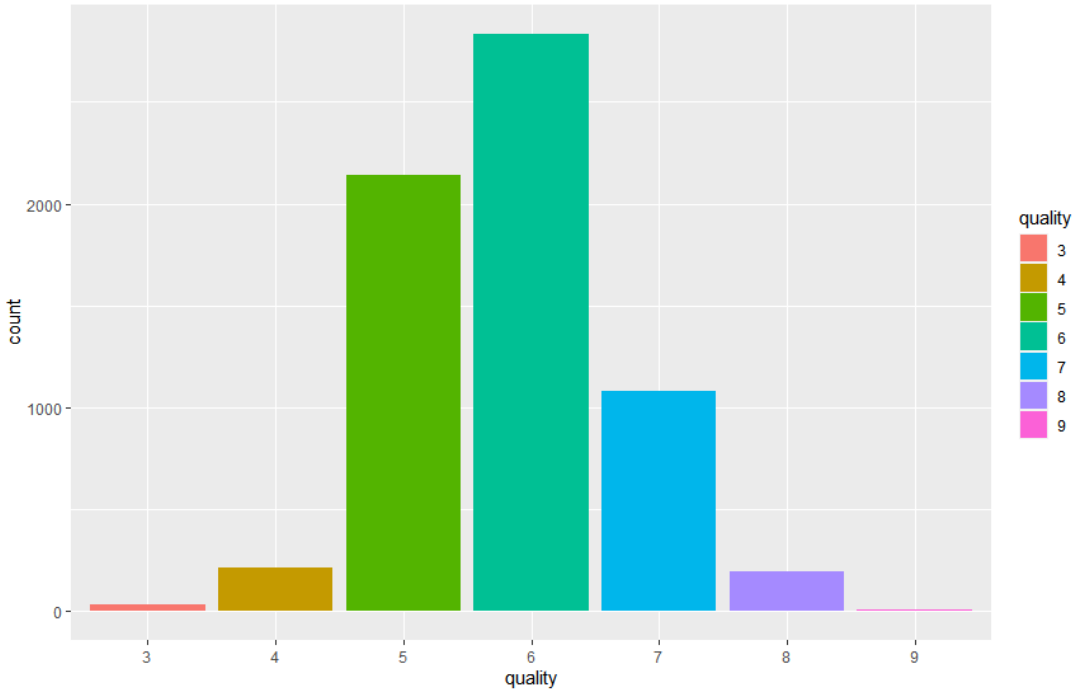
The output makes us have a basic understanding of the features in the dataset: the maximum score of the quality is 9 and average is 5.818, the mean of color is equal to 0.2461 which means only 24.61% wine is red wine…

The heatmap is a great way for data exploration since it can help us to understand the correlation between each variable, so then we draw the heatmap.



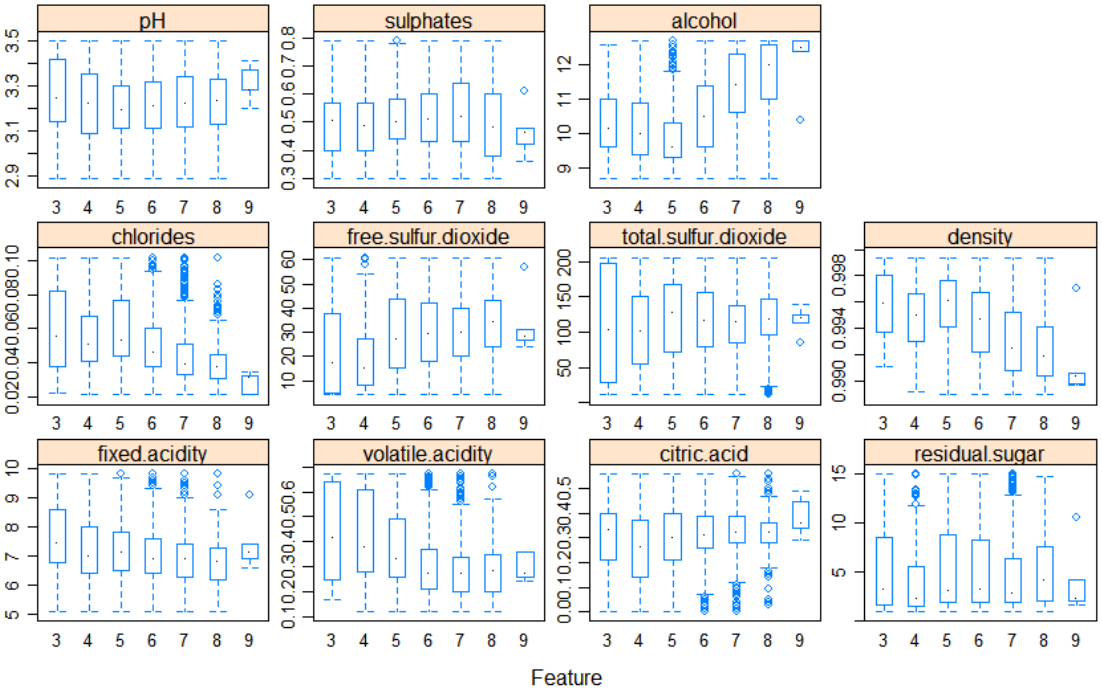
From the heatmap, we can see, alcohol, density and volatile acidity are the top 3 independent variables that correlated with wine

Then we draw we draw the histogram to show the distribution of our objective: quality



According to the histogram, the quality with 6 and 5 are two of the most frequency quality, and wine with best and the worst quality are least.

Then we draw the box plot between the quality and most of our independent variables, which could provide us a basic view of how the quality is related to them.



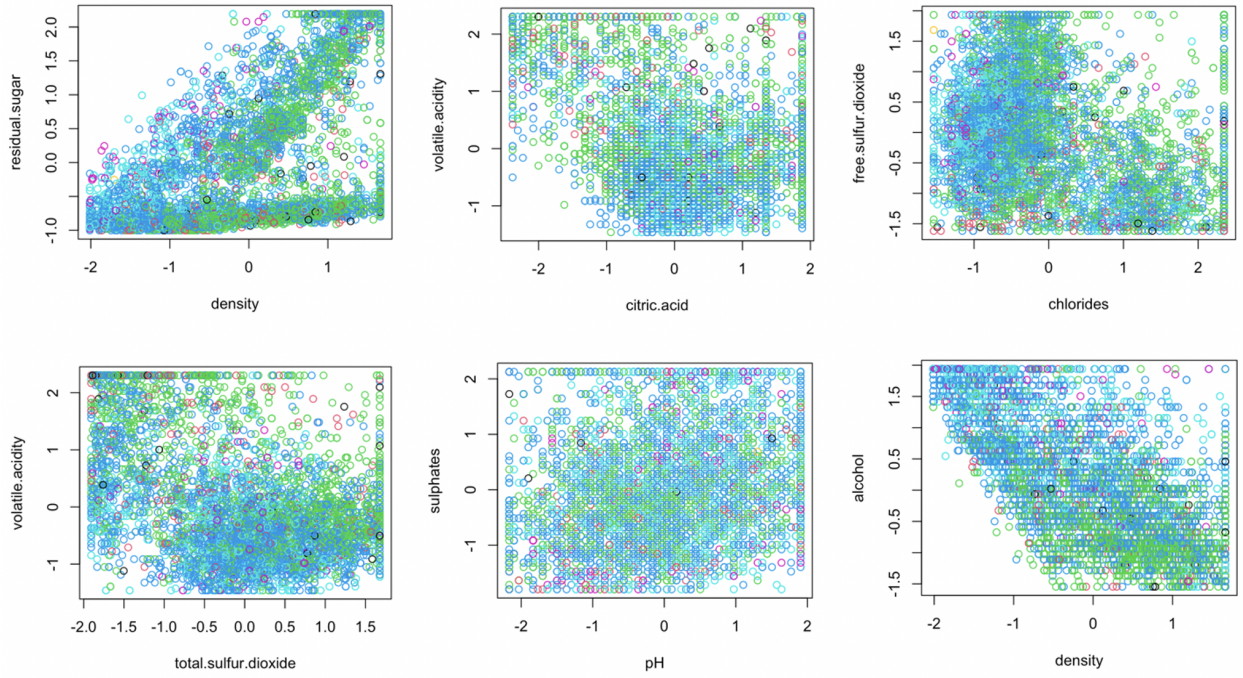
We found some interesting thing by looking at the boxplot, like the distribution of the wine with quality of nine is very different with others, and they typically have a comparative smaller range.

Until now, we finished that data exploration part and then we will focus on the modeling process.

Modeling & Validation

(1) Support Vector Machine (SVM)

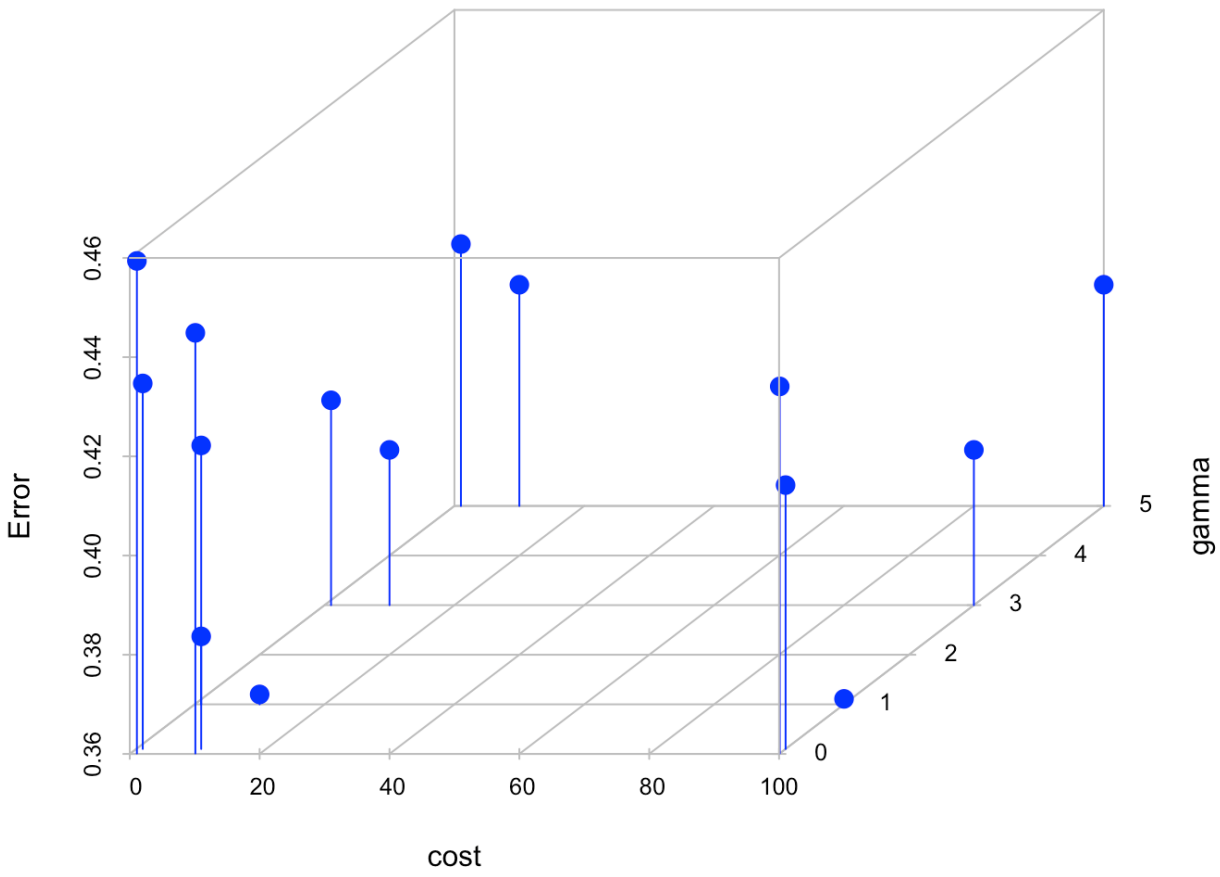
* Pairs of variables with the seven classes of “quality”



- X-axis and Y-axis correspond to different pairs of the independent variables and the seven distinct quality scores are differentiated with different point colors. The distributions of the seven quality labels between different pairs of predictors seem hard to be dissected by linear decision boundary through the one-versus-one approach.
  Under the circumstances, the quality labels should be better to be classified by SVM models with radial and polynomial kernels.

(1-1) SVM with radial kernel

* Finding the optimal tuning parameters (cost & gamma)



- By using 5-Folds Cross-Validation on the training set with the cost (1, 10, 100) and gamma (0.01, 0.1, 1, 3, 5), the parameters minimizing the validation errors are chosen:

Cost: 100
Gamma: 1

```
[1] "cost = 1 gamma = 0.01 cv-error: 0.4593"
[1] "cost = 1 gamma = 0.1 cv-error: 0.4337"
[1] "cost = 1 gamma = 1 cv-error: 0.3737"
[1] "cost = 1 gamma = 3 cv-error: 0.4013"
[1] "cost = 1 gamma = 5 cv-error: 0.4128"
[1] "cost = 10 gamma = 0.01 cv-error: 0.4448"
[1] "cost = 10 gamma = 0.1 cv-error: 0.4212"
[1] "cost = 10 gamma = 1 cv-error: 0.362"
[1] "cost = 10 gamma = 3 cv-error: 0.3913"
[1] "cost = 10 gamma = 5 cv-error: 0.4046"
[1] "cost = 100 gamma = 0.01 cv-error: 0.434"
[1] "cost = 100 gamma = 0.1 cv-error: 0.4132"
[1] "cost = 100 gamma = 1 cv-error: 0.3611"
[1] "cost = 100 gamma = 3 cv-error: 0.3913"
[1] "cost = 100 gamma = 5 cv-error: 0.4046"
```

- The chosen two tuning parameters control the model's complexity optimizing the bias – variance tradeoff.

*** Validation of the fitted model**

Confusion Matrix

	test.y						
pred.rad	3	4	5	6	7	8	9
3	0	1	0	0	0	0	0
4	0	6	2	4	0	0	0
5	2	17	287	100	10	3	0
6	0	20	106	434	67	12	0
7	0	1	7	41	127	8	0
8	0	0	0	3	2	14	0
9	0	0	0	0	0	0	0

- For the purpose of evaluating the train model on the test set, the confusion matrix created with the predicted values of the test set can be utilized. The correctly classified values by labels are placed on the diagonal line.

Sensitivity test

3	4	5	6	7	8	9
0.0000000	0.1333333	0.7139303	0.7457045	0.6165049	0.3783784	NaN

- Sensitivity represents the proportion of the number of the correctly classified values by class to the total number of the values classified into the corresponding class. It can measure the correctness of the classification label by label.

Benchmark

3	4	5	6	7	8	9
0.001569859	0.035321821	0.315541601	0.456828885	0.161695447	0.029042386	0.000000000

- Benchmark is the proportion of the number of observations of test set response by each class to the total number of the observations of the test set response.
- As the model is trained with the unbalanced number of observations by class, the performance of the model should be evaluated by relative criteria. In this sense, the benchmark can be utilized to independently assess the accuracy of each class.

- According to the sensitivity, the quality score 4, 5, 6, 7, 8 are relatively well classified when compared with the benchmark. The class 3 which has been trained with very small number of training observations demonstrates unsatisfactory accuracy. In the case of the class 9, the number of its observations is too small to be assigned to the test set.

Accuracy

Accuracy = 0.6813

- Accuracy is calculated by dividing the total number of the correctly classified values by the total number of the observations of the test response variable.
- The overall accuracy of the SVM model with radial kernel is about **68.13%**.

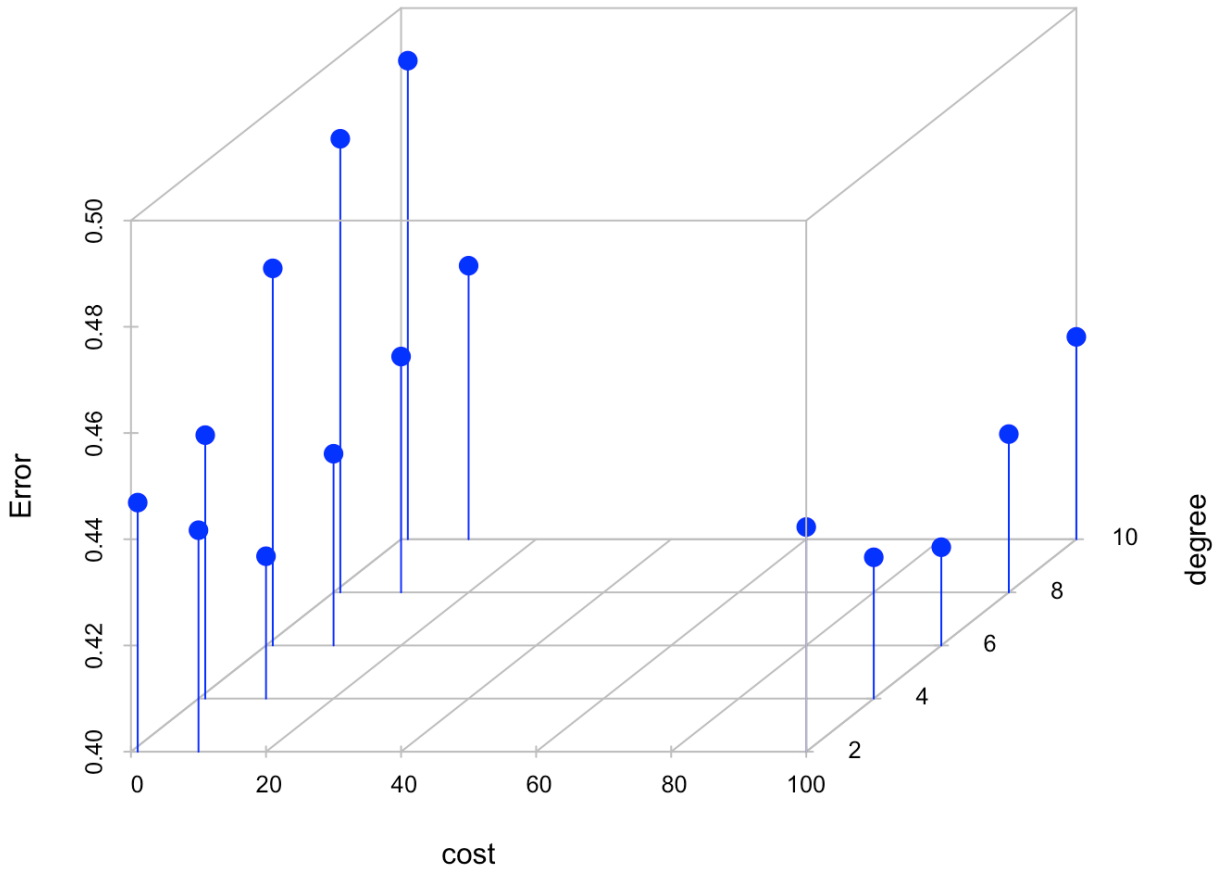
Test Error

Test Error = 0.3187

- Test error can be calculated by dividing the sum of the incorrectly classified values by the total number of the observations of the test response variable.
- The test error produced by the fitted SVM model with radial kernel is approximately **31.87%.**

(1-2) SVM with polynomial kernel

* Finding the optimal tuning parameters (cost & degree)



- By using a 5-Folds Cross-Validation on the training set with the cost (1, 10, 100) and degree (2, 4, 6, 8, 10), the parameters minimizing the validation errors are chosen:

```
[1] "cost = 1 degree = 2 cv-error: 0.4469"
[1] "cost = 1 degree = 4 cv-error: 0.4496"
[1] "cost = 1 degree = 6 cv-error: 0.471"
[1] "cost = 1 degree = 8 cv-error: 0.4854"
[1] "cost = 1 degree = 10 cv-error: 0.4901"
[1] "cost = 10 degree = 2 cv-error: 0.4417"
[1] "cost = 10 degree = 4 cv-error: 0.4268"
[1] "cost = 10 degree = 6 cv-error: 0.4361"
[1] "cost = 10 degree = 8 cv-error: 0.4444"
[1] "cost = 10 degree = 10 cv-error: 0.4515"
[1] "cost = 100 degree = 2 cv-error: 0.4423"
[1] "cost = 100 degree = 4 cv-error: 0.4266"
[1] "cost = 100 degree = 6 cv-error: 0.4185"
[1] "cost = 100 degree = 8 cv-error: 0.4298"
[1] "cost = 100 degree = 10 cv-error: 0.4381"
```

Cost: 100
Degree: 6

- Fitting the SVM model with polynomial kernel with cost = 100 and degree = 6 for the optimal bias-variance tradeoff

*** Validation of the fitted model**

Confusion Matrix

pred.pol	test.y 3	4	5	6	7	8	9
3	0	1	0	0	0	0	0
4	1	11	12	6	0	1	0
5	1	22	267	120	10	3	0
6	0	8	116	396	101	14	0
7	0	1	7	52	92	10	0
8	0	2	0	8	3	9	0
9	0	0	0	0	0	0	0

- This is the confusion matrix created with the predicted values of the test set. The correctly classified values by labels are placed on the diagonal line.

Sensitivity test

3	4	5	6	7	8	9
0.0000000	0.2444444	0.6641791	0.6804124	0.4466019	0.2432432	NaN

Benchmark

3	4	5	6	7	8	9
0.001569859	0.035321821	0.315541601	0.456828885	0.161695447	0.029042386	0.000000000

- As the SVM model with radial kernel did above, the model with polynomial basis kernel also has the sensitivity exceeding the benchmark values in the classes ranging from 4 to 8. However, when being compared to the SVM with radial model above, the sensitivity of the polynomial-based model shows worse performance by class.

Accuracy

Accuracy = 0.6083

- The overall accuracy of the SVM model with polynomial kernel is about **60.83%**.

Test Error

Test Error = 0.3917

- The test error rate produced by the fitted SVM model with polynomial kernel is approximately **39.17%.**

- When it comes to the performance of the model's classification, the SVM model with polynomial is worse than the radial-based SVM model based on the higher test error rate.

(2) Multinomial Logistic Regression

* train model with all the 12 predictors

Coefficients

```
Call:
multinom(formula = quality ~ ., data = train)

Coefficients:
  (Intercept) fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides free.sulfur.dioxide total.sulfur.dioxide
4    1.656015    -0.8162073      -0.03375891 -0.05830879    -0.55876400 -0.2202091         -0.74486015           0.10615076
5    6.712247    -1.0480044      -0.50883121  0.11030923    -0.37809005 -0.2572648         -0.01139097           0.40748140
6    7.499323    -1.0030137      -1.21971720  0.03122322     0.06964121 -0.2405065          0.24219984          -0.02341059
7    6.680357    -0.5727066      -1.50649164  0.10150194     0.78968771 -0.4596201          0.29194472          -0.07440237
8    4.515389    -0.4912574      -1.47358621  0.22438092     1.43282764 -0.3419641          0.55687233          -0.16113626
9  -19.421593     8.6720825      -1.93162252  2.02779715     8.19850938 -9.2995095          0.53423563           0.58350206
      density         pH   sulphates     alcohol  colorwhite
4   0.6302440 -0.5192996 -0.06573283  0.08070913 -0.02426877
5   0.4032547 -0.6595053  0.06787864 -0.29893659 -3.15063576
6   0.1426088 -0.5805904  0.36534863  0.55553361 -3.45381884
7  -0.7935651 -0.2798402  0.61195121  0.90460473 -4.26611648
8  -1.4211146 -0.1171161  0.58578657  1.02317183 -4.26891114
9 -12.3340127  6.8260959  1.00242277 -1.82232787 -4.18695614
```

- As we have 7 groups, I set the label 3 as the reference group. So, we can interpret each coefficient comparing with the reference group. For example, the coefficient of the "Fixed.acidity" whose quality group is 4 can be interpreted as one unit increase in the Fixed.acidity reduces the log odds of the quality 4 by 0.816 from the value of quality 3 holding the other predictors constant.

P-values

	(Intercept)	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide
4	1.483227e-01	0.039438374	8.994891e-01	0.81359880	0.31903001	0.48919179	0.02865032
5	4.591245e-10	0.004531677	4.286724e-02	0.63204118	0.46839053	0.38749687	0.97105138
6	3.693268e-12	0.006599340	1.381369e-06	0.89245538	0.89376216	0.42051478	0.44065869
7	1.015611e-09	0.127108843	6.172470e-09	0.66741725	0.13723546	0.13424758	0.35915801
8	1.116807e-04	0.232263537	3.879480e-07	0.39202030	0.01418261	0.32439375	0.10125043
9	3.976106e-01	0.011149415	1.378347e-01	0.09784321	0.02031640	0.01379591	0.60787894

	total.sulfur.dioxide	density	pH	sulphates	alcohol	colorwhite
4	0.7999188	0.47765573	0.100444281	0.81925165	0.85982268	0.985704970
5	0.3034088	0.62561660	0.024733674	0.79978725	0.48147309	0.013543320
6	0.9529233	0.86301225	0.047952692	0.17181226	0.19072647	0.006943120
7	0.8535346	0.34628591	0.347428001	0.02329503	0.03670222	0.001035315
8	0.7115576	0.12752224	0.715334653	0.03745945	0.03373819	0.002293168
9	0.7363410	0.03087611	0.008184778	0.24994924	0.44339334	0.849600933

- The covariates of which p-values greater than 0.05 should be sifted out from the regression model as the statistically non-significant predictors.

- The p-values can be derived from the z-score of each label in each variable:

    - Z-score: I first divided the 7 coefficients of each variable by their standard errors to obtain their z-scores.

    - P-value: I can draw the probability corresponding to the critical value from the standard normal distribution and subtracted the probability from 1 to get the extreme probability at the tails.

- All the p-values of the "Citric-acid" and "Total.Sulfur.Dioxide" are bigger than 0.05 meaning that none of the 7 classes can be explained by these two variables. So, the two variables should be removed from the train model.

*** Validation of the fitted model**

Confusion Matrix

pred.test \ test.y	3	4	5	6	7	8	9
3	0	0	0	0	0	0	0
4	0	1	2	1	0	0	0
5	2	28	247	149	11	1	0
6	0	16	152	386	145	26	0
7	0	0	1	45	49	9	0
8	0	0	0	0	0	0	0
9	0	0	0	1	1	1	0

Labels	Benchmark	Sensitivity	Accuracy
3	0.00157	0.00000	
4	0.03532	0.02222	**0.5361**
5	0.31554	0.61443	
6	0.45683	0.66323	**Test Error**
7	0.16170	0.23786	
8	0.02904	0.00000	**0.4639**
9	0.00000	NaN	

- The sensitivity, overall accuracy, and the corresponding test error rate can be computed from the confusion matrix created on the test set. For the case of the logistic model, the only labels of which sensitivity exceeding the benchmark are 5, 6, and 7. And the overall performance proved to be not considerable due to its poor accuracy score which is 54%.

(3) K-Nearest Neighbor (KNN)

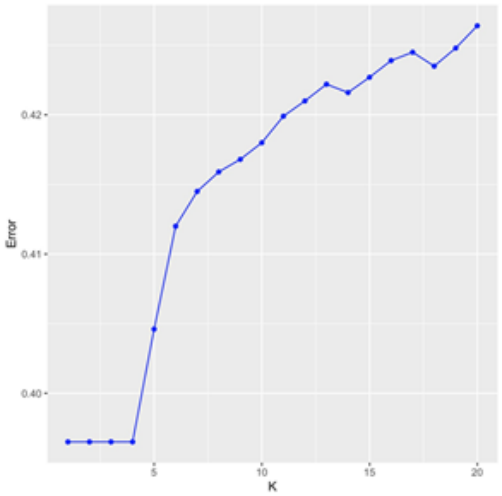
Combined with the data set, the KNN model we constructed has the following characteristics.

1. Since our dataset has 7 categories (3, 4, 5, 6, 7, 8, 9 respectively), it is a multi-category KNN model.

2. The independent variable used for classification in the data set contains the factor type, so the KKNN package is better to solve the problem instead of the class package.

3. We find the best optimal K is 1 since it has the smallest cv error. Here is the picture of the process of finding the optimal k.

```
[1] "K = 1 cv-error: 0.3965"
[1] "K = 2 cv-error: 0.3965"
[1] "K = 3 cv-error: 0.3965"
[1] "K = 4 cv-error: 0.3965"
[1] "K = 5 cv-error: 0.4046"
[1] "K = 6 cv-error: 0.412"
[1] "K = 7 cv-error: 0.4145"
[1] "K = 8 cv-error: 0.4159"
[1] "K = 9 cv-error: 0.4168"
[1] "K = 10 cv-error: 0.418"
[1] "K = 11 cv-error: 0.4199"
[1] "K = 12 cv-error: 0.421"
[1] "K = 13 cv-error: 0.4222"
[1] "K = 14 cv-error: 0.4216"
[1] "K = 15 cv-error: 0.4227"
[1] "K = 16 cv-error: 0.4239"
[1] "K = 17 cv-error: 0.4245"
[1] "K = 18 cv-error: 0.4235"
[1] "K = 19 cv-error: 0.4248"
[1] "K = 20 cv-error: 0.4264"
```



Through the establishment of the model, the following results are obtained.

```
          test.y
pred.knn    3   4   5   6   7   8   9
       3    0   4   2   0   0   0   0
       4    0   9  10  20   1   0   0
       5    2  17 280 110  10   4   0
       6    0  14  95 378  55   7   0
       7    0   1  14  65 133  11   0
       8    0   0   1   9   7  15   0
       9    0   0   0   0   0   0   0
>
> ## test errors
> 1-sum(diag(t.knn))/sum(t.knn)
[1] 0.3602826
>
> ## accuracy
> sum(diag(t.knn))/sum(t.knn)
[1] 0.6397174
>
> ## sensitivity
> diag(t.knn/colSums(t.knn))
        3         4         5         6         7         8         9
0.0000000 0.2000000 0.6965174 0.6494845 0.6456311 0.4054054       NaN
>
> ## benchmark
> n <- table(test.y)
> n/sum(n)
test.y
          3           4           5           6           7           8           9
0.001569859 0.035321821 0.315541601 0.456828885 0.161695447 0.029042386 0.000000000
```

Labels	Benchmark	Sensitivity	Accuracy
3	0.00157	0.00000	
4	0.03532	0.20000	0.63971
5	0.31554	0.69651	
6	0.45683	0.64948	Test Error
7	0.16170	0.64563	
8	0.02904	0.40540	0.36028
9	0.00000	NaN	

1. From the perspective of the confusion matrix, the overall prediction effect of the model is good. The accuracy of the model is 0.63.

2. Finally, the sensitivity of each category is output. It can be seen from the results that the prediction results of the model in categories 5, 6 and 7 are good, while the prediction results in categories 3 and 9 are poor. The reason for this phenomenon is that there are more samples in categories 5, 6 and 7, and fewer samples in categories 3 and 9. It is easy to cause this phenomenon in the KNN model.

(4) Quadratic Discriminant Analysis (QDA)

In the process of using the data set for research, we established the QDA model according to it. Because this data set has some characteristics, the QDA model is explained as follows.

1. QDA model has certain requirements on the number of categories. Categories 3 and 9 are few in number, so they are removed when modeling.

2. We use QDA functions in the MASS package for modeling.

The results of the QDA model are as follows,

```
pred_test_qda   4   5   6   7   8
            4   4   5   6   1   0
            5  25 248 140  10   0
            6  10 147 322  96  21
            7   3  11 102 109  21
            8   0   0   5   5   2
>
>
> ### test error
> 1-sum(diag(confusion_matrix_test))/sum(confusion_matrix_test)
[1] 0.4702243
>
>
> ### accuracy
> sum(diag(confusion_matrix_test))/sum(confusion_matrix_test)
[1] 0.5297757
>
>
> ### sensitivity
> diag(confusion_matrix_test/colSums(confusion_matrix_test))
         4          5          6          7          8
0.09523810 0.60340633 0.56000000 0.49321267 0.04545455
>
>
> ### benchmark
> n <- table(test_wine$quality)
> n/sum(n)

         4          5          6          7          8
0.03248260 0.31786543 0.44470224 0.17092034 0.03402939
```

Labels	Benchmark	Sensitivity	Accuracy
4	0.03248	0.09523	**0.52977**
5	0.31786	0.60340	
6	0.44470	0.56000	**Test Error**
7	0.17092	0.49321	
8	0.03402	0.04545	**0.47022**

1. It can be seen from the results that the accuracy of the model is 0.5297757. Overall, the accuracy of the model is lower than that of the KNN model. This is also because the prediction effect of the QDA model is generally not very good in the process of multi-classification tasks.

2. From the perspective of sensitivity of each category, the QDA model has a better prediction effect in categories 5, 6 and 7. However, the prediction effect on Model 4 and 8 is not good, because the prediction effect of the categories with a small number is often inferior to that of the categories with a large number in the prediction process.

Based on all the test error results, we can assume that the SVM Model with radial Kernel performs the best with 68% accuracy.

Model	Test Error Rate
SVM model with radial kernel	0.3187
SVM model with polynomial kernel	0.3917
Logistic Regression model	0.4639
KNN model	0.3603
QDA model	0.4702