The Chemistry of Quality

White Wine of North Portugal

Alana Tauala, Leif Watkins, Sang Xing

Table of Contents

01 Introduction

02 Data Analysis

03 Data Modeling

- K-nearest Neighbor (KNN) Classifier
- Logistic Regression
- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)

04 Conclusion

05 Further Modeling

Part.01

Introduction

Overview

- The wine quality dataset is available in the UCI Machine Learning Repository [1].
- It contains information about the physicochemical properties and quality ratings of red and white wine samples from the north of Portugal.

Predictors	Meaning
`Fixed acidity`	the amount of non-volatile acids in the wine.
`Volatile acidity`	the amount of acetic acid in the wine.
`Citric acid`	the amount of citric acid in the wine.
`Residual sugar`	the amount of sugar left in the wine after fermentation.
`Chlorides`	the amount of salt in the wine.
`Free sulfur dioxide`	the amount of free sulfur dioxide in the wine.
`Total sulfur dioxide`	the amount of total sulfur dioxide in the wine.
`Density`	the density of the ethanol.
`pH`	the acidity level of the wine.
`Sulphates`	the acidity level of the wine.
`Alcohol`	the percentage of alcohol in the wine.
`Quality`	the quality rating of the wine on a scale from 0 to 10.

Task

Consider white wine quality dataset only, mutate `quality` as `good` for `quality` ≥ 7, `good` $= 1$, else 0. Drop `quality`, use `good` as our response. Use 10-fold cross-validation to construct the following method, and justify which one you recommend?

- `caret::`KNN
- `MASS::` Logistic Regression
- `MASS::` LDA
- `MASS::` QDA

Part.02

Data Analysis

Data Description and Distribution

```
wine.data <- read.csv("dataset\\winequality-white.csv", sep=";", header = T)

str(wine.data)
```

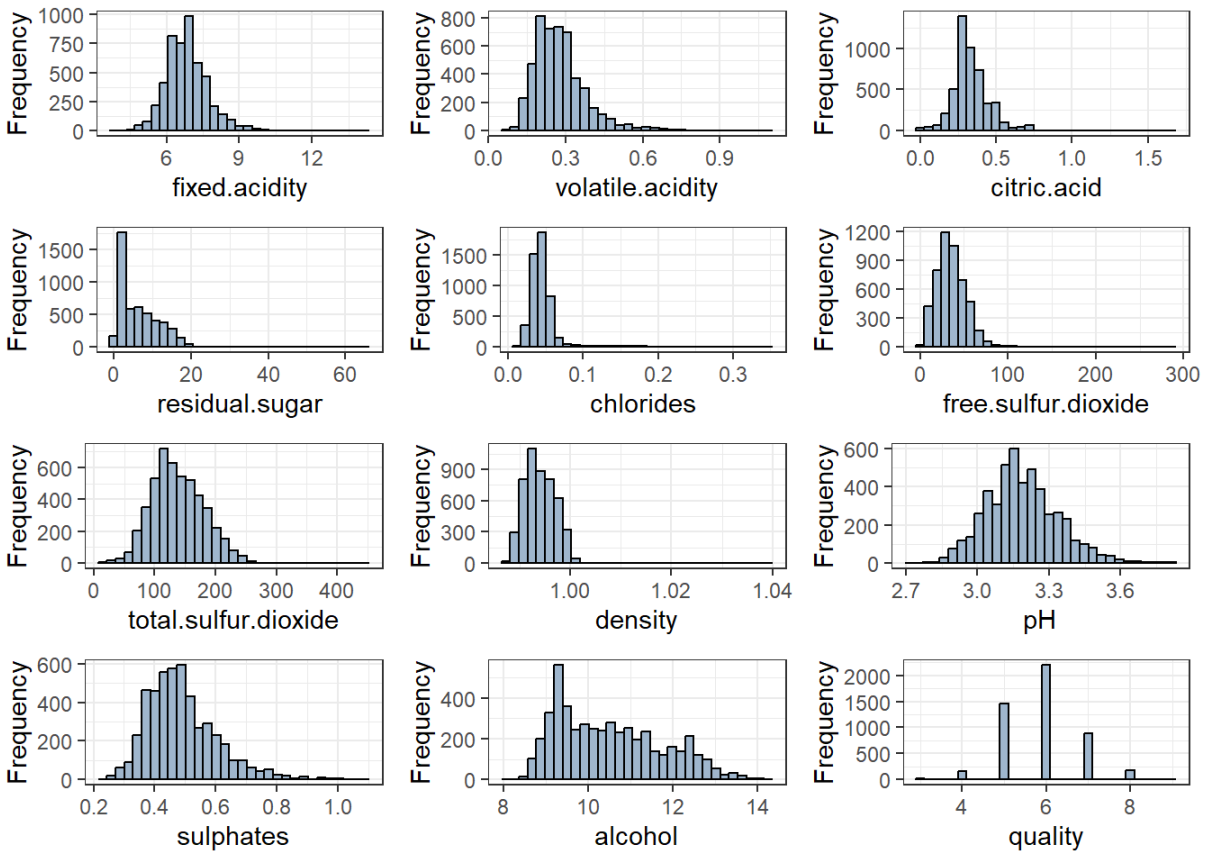
```
'data.frame':    4898 obs. of  12 variables:
 $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
 $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
 $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
```

```
# Check for NAs in dataset
sum(is.na(wine.data))
```

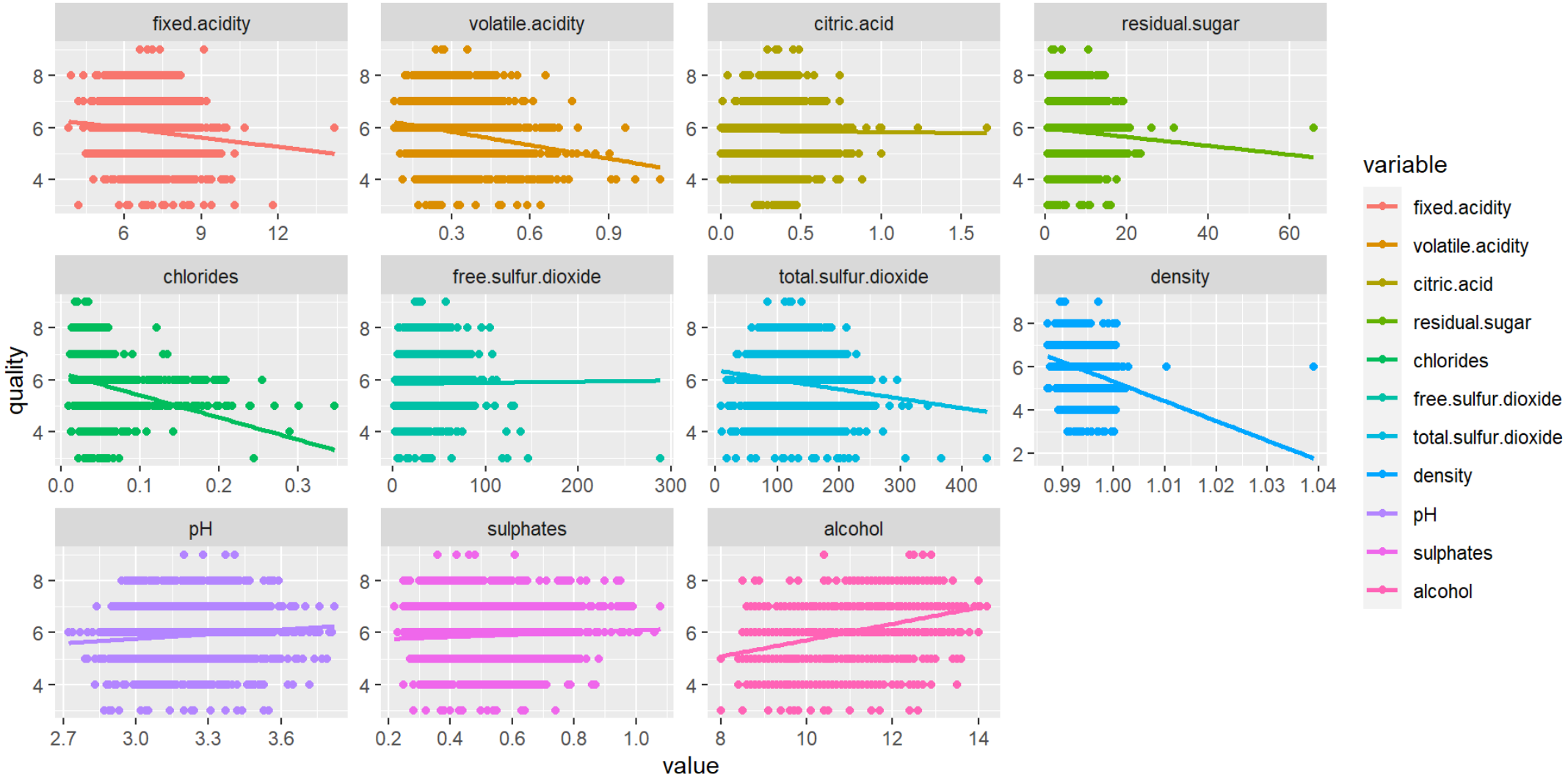
```
[1] 0
```

```
# Counts at each combination of response's factor levels
table(wine.data$quality)
```

```
  3   4   5   6   7   8   9
 20 163 1457 2198 880 175   5
```



Data Relationship



Data Correlations and Splits

Data Mutation

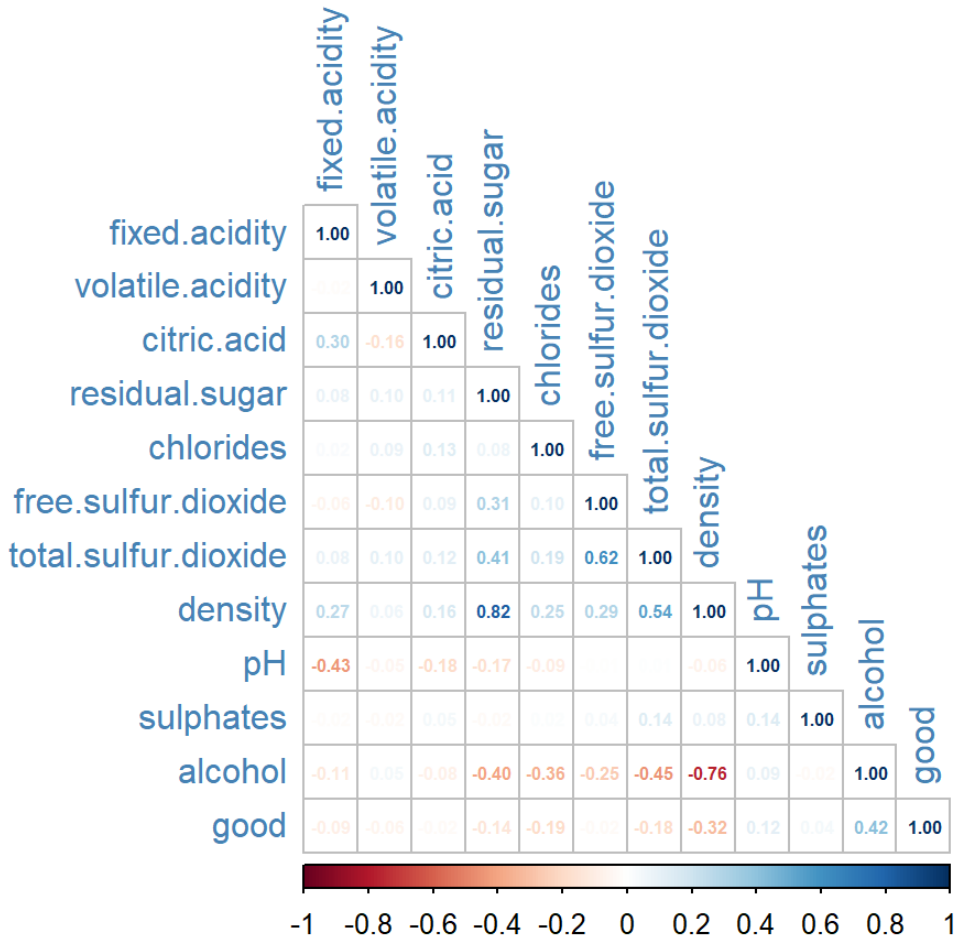
```
wine.data_cleaned <-  wine.data %>%
  mutate(good = ifelse(quality>=7, 1, 0)) %>%
  distinct() %>%
  select(-quality)
```

```
dim(wine.data_cleaned)
```

```
[1] 3961   12
```

Data Split

```
set.seed(1234)
# Splitting the dataset into train and test (7/10th for train remaining for test)
inTrain <- caret::createDataPartition(wine.data_cleaned$good, p = 7/10, list = F)
train <- wine.data_cleaned[inTrain,]
test  <- wine.data_cleaned[-inTrain,]


# Convert the outcome variable to a factor with two levels
train$good <- as.factor(train$good)
test$good <- as.factor(test$good)
```



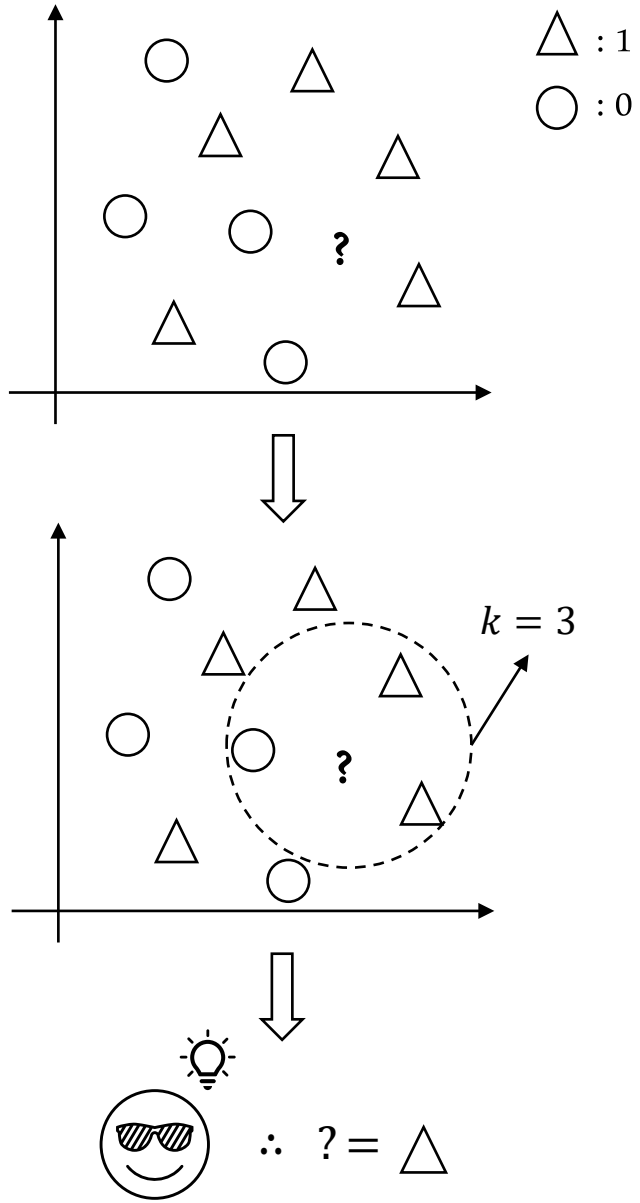
Part.03

Data Modeling

K-nearest Neighbors (KNN) Classifier

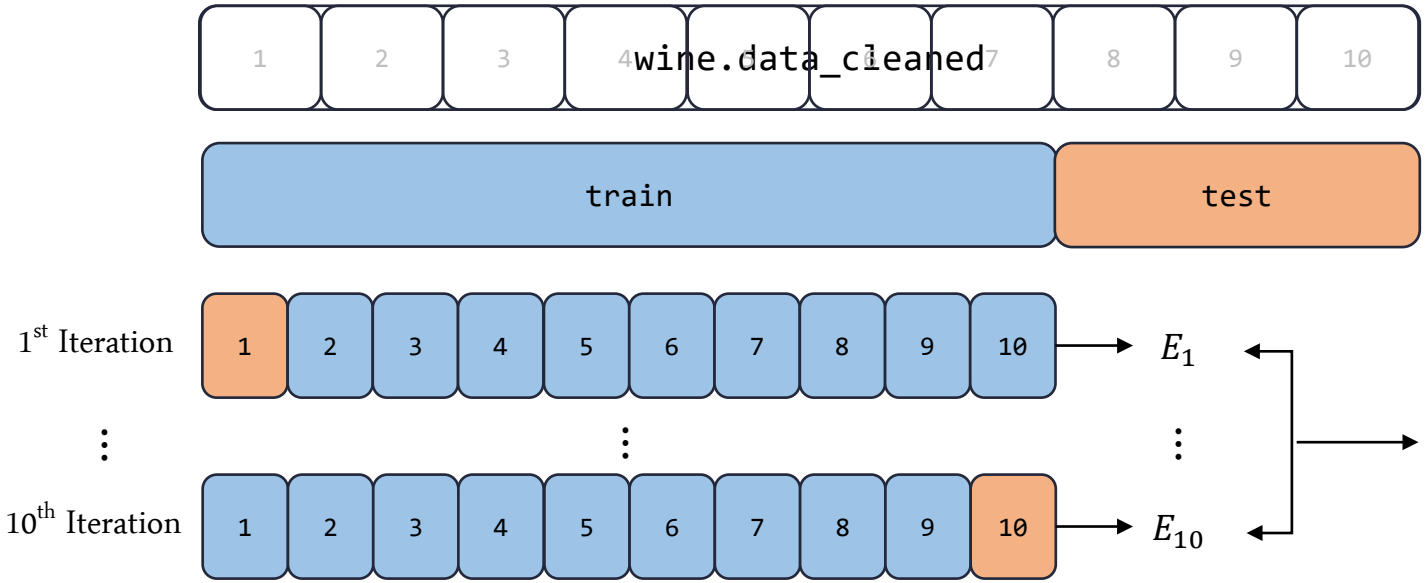
To predict the class of a new observation using KNN, we need to follow these steps:

1. Choose the number of nearest neighbors K.
2. Calculate the distance between the new observation and all observations in the training data.
3. Select the K nearest neighbors based on the calculated distances.
4. Determine the majority class of the K nearest neighbors.
5. Assign the new observation to the majority class.



10-fold Cross Validation (CV)

- However, the choice of K can have a significant impact on the performance of the algorithm.
- A small value of K can result in overfitting, while a large value of K can result in underfitting.
- Hence, we need to choose an optimal value of K that minimizes the test error rate.




```r
set.seed(1234)
knn_model <- train(good ~ .,
                   data = train,
                   method = "knn",
                   trControl = trainControl(method = "cv", number = 10),
                   tuneGrid = data.frame(k = 1:10))
```

$$OER_{(k=1)} = \frac{1}{10}\sum_{i=1}^{10} E_i \implies k = \min(OER_{k=1,\,2,\,\dots,\,10})$$

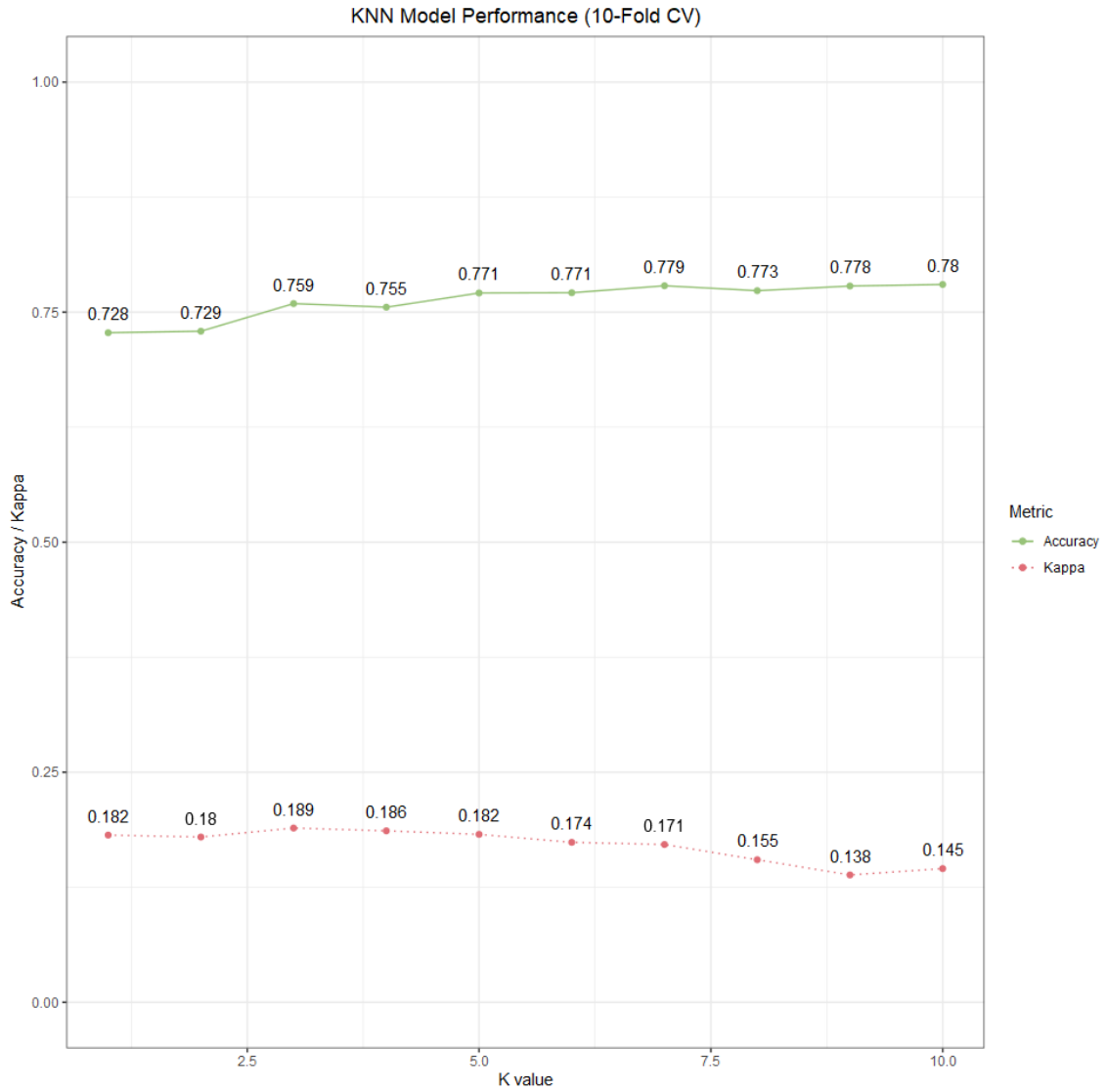
KNN Accuracy

```
knn_model$bestTune
```

```
    k
10 10
```

```
knn.predicted_quality <- predict(knn_model, newdata = test)

confusionMatrix(knn.predicted_quality, test$good)
```

```
          Reference
Prediction   0   1
         0 918 169
         1  43  48

               Accuracy : 0.8215
                 95% CI : (0.7986, 0.8429)
    No Information Rate : 0.8089
    P-Value [Acc > NIR] : 0.142

                  Kappa : 0.2675

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9553
            Specificity : 0.2555
         Pos Pred Value : 0.8445
         Neg Pred Value : 0.5743
             Prevalence : 0.8089
         Detection Rate : 0.7727
   Detection Prevalence : 0.9150
      Balanced Accuracy : 0.6054

       'Positive' Class : 0
```



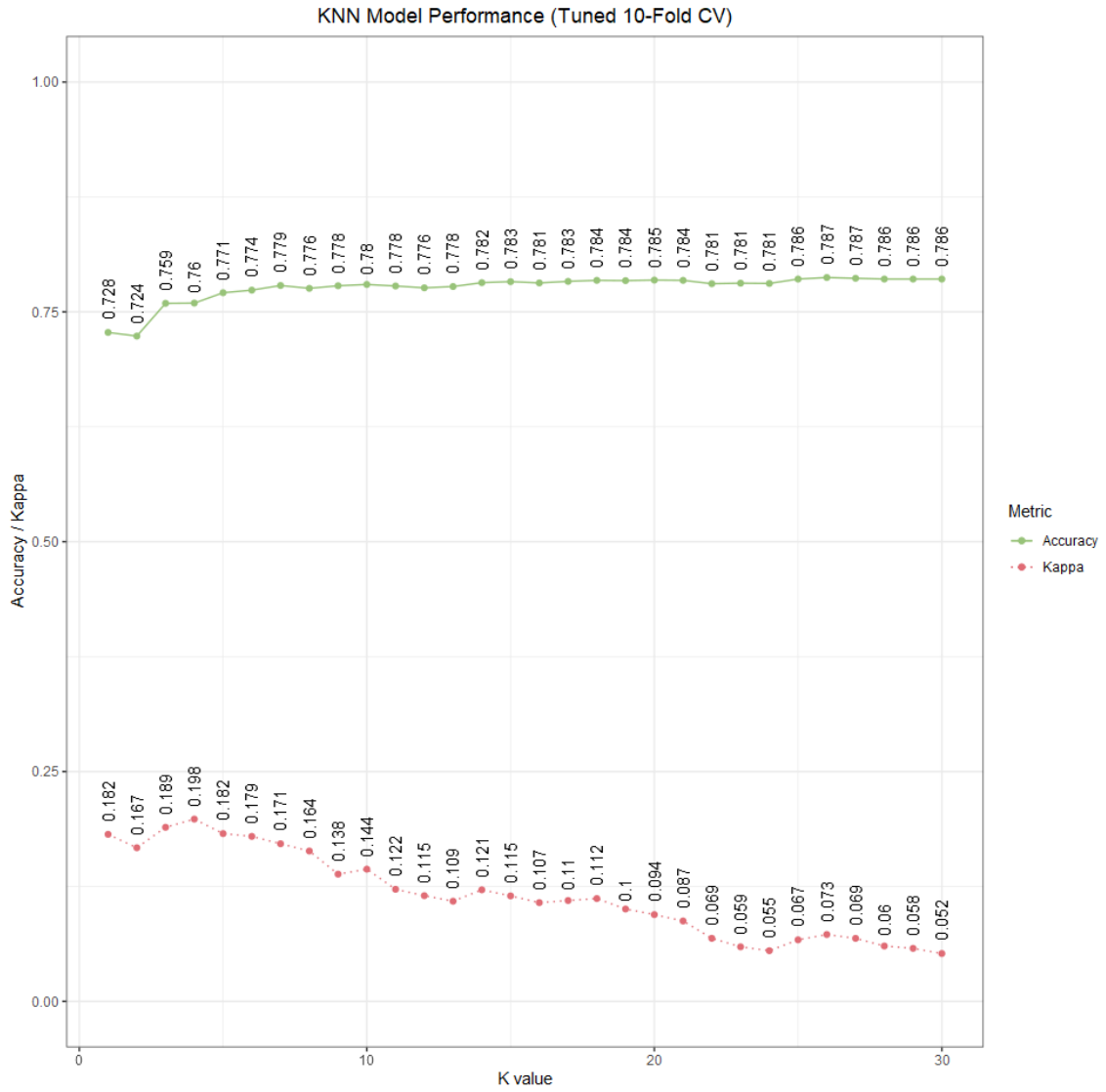
KNN Accuracy

```
knn_model$bestTune
```

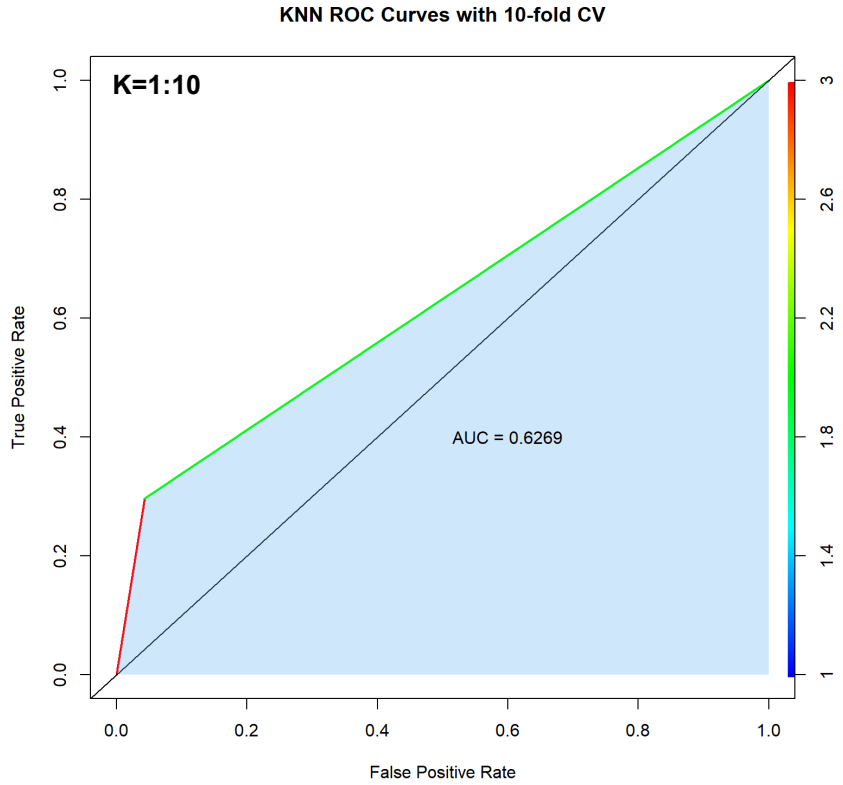
```
    k
26 26
```

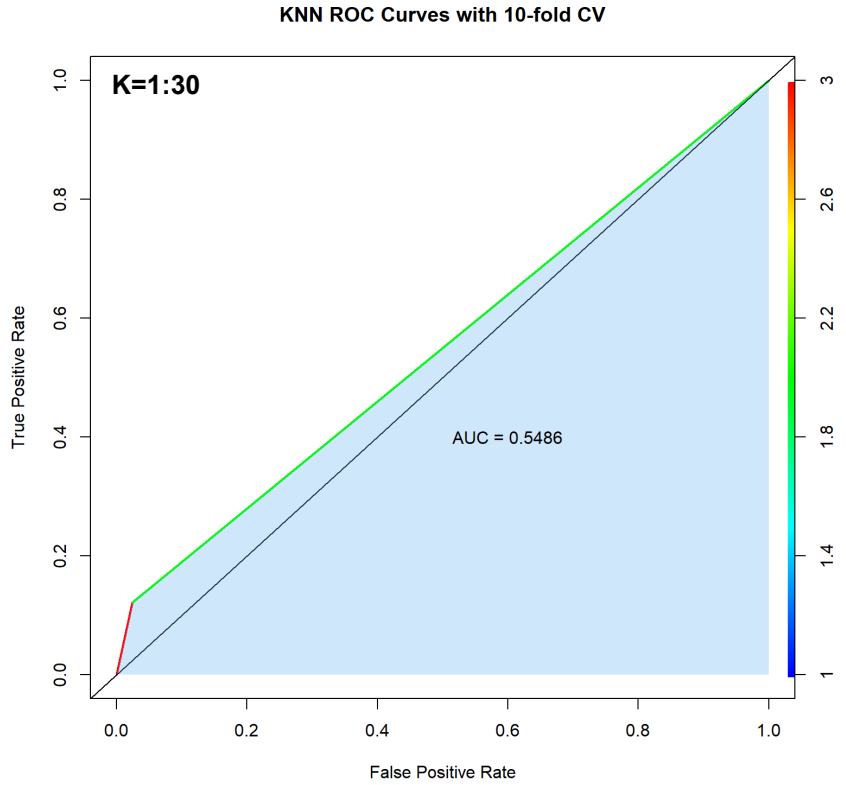
```
knn.predicted_quality <- predict(knn_model, newdata = test)

confusionMatrix(knn.predicted_quality, test$good)
```

```
          Reference
Prediction   0   1
         0 934 209
         1  27  18

               Accuracy : 0.8013
                 95% CI : (0.7775, 0.8237)
    No Information Rate : 0.8089
    P-Value [Acc > NIR] : 0.7596

                  Kappa : 0.0738

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9719
            Specificity : 0.0793
         Pos Pred Value : 0.8171
         Neg Pred Value : 0.4000
             Prevalence : 0.8089
         Detection Rate : 0.7862
   Detection Prevalence : 0.9621
      Balanced Accuracy : 0.5256

       'Positive' Class : 0
```



KNN Performance





Method	Test Error Rate	Sensitivity	Specificity	AUC
K-Fold CV (k=1:10)	0.1785	0.9568	0.2971	0.6269339
K-Fold CV (k=1:30)	0.1961	0.9758	0.1213	0.5485514

Logistic Regression

Goal: To fit a logistic regression model with estimated values of the coefficients β_0 to β_k that best fit the data to make predictions on new data..

```
logit_model <- glm(good ~ ., data = train, family = binomial)
logit_model$coefficients
```

(Intercept)	fixed.acidity	volatile.acidity
6.796587e+02	6.260591e-01	-3.217817e+00
citric.acid	**residual.sugar**	**chlorides**
-4.094836e-01	2.933474e-01	-1.565951e+01
free.sulfur.dioxide	**total.sulfur.dioxide**	**density**
1.547773e-02	-3.055192e-03	-7.066102e+02
pH	**sulphates**	**alcohol**
4.277068e+00	2.098402e+00	1.535664e-01

$$P(\hat{Y}) = \frac{1}{1 + \exp\left(-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_{11} X_{11})\right)}$$

$P(\hat{Y})$: The probability of the response variable being 1.

β_p : Intercept and Predictors' Coefficients Estimates.

10-fold CV steps

1. Split the original dataset (in our case, the 70% training dataset) into 10 roughly equal-sized folds.
2. For each fold, fit the logistic regression model using the other 9 folds as training data.
3. Use the fitted model to predict the response variable for the observations in the held-out fold.
4. Calculate the accuracy, sensitivity, specificity, and AUC for the predicted values of the held-out fold.
5. Repeat steps 2-4 for each of the 10 folds, using a different fold as the held-out set each time.
6. Average the accuracy, sensitivity, specificity, and AUC values across the 10 folds to obtain estimates of the model's performance on new data.

Linear Discriminant Analysis (LDA)

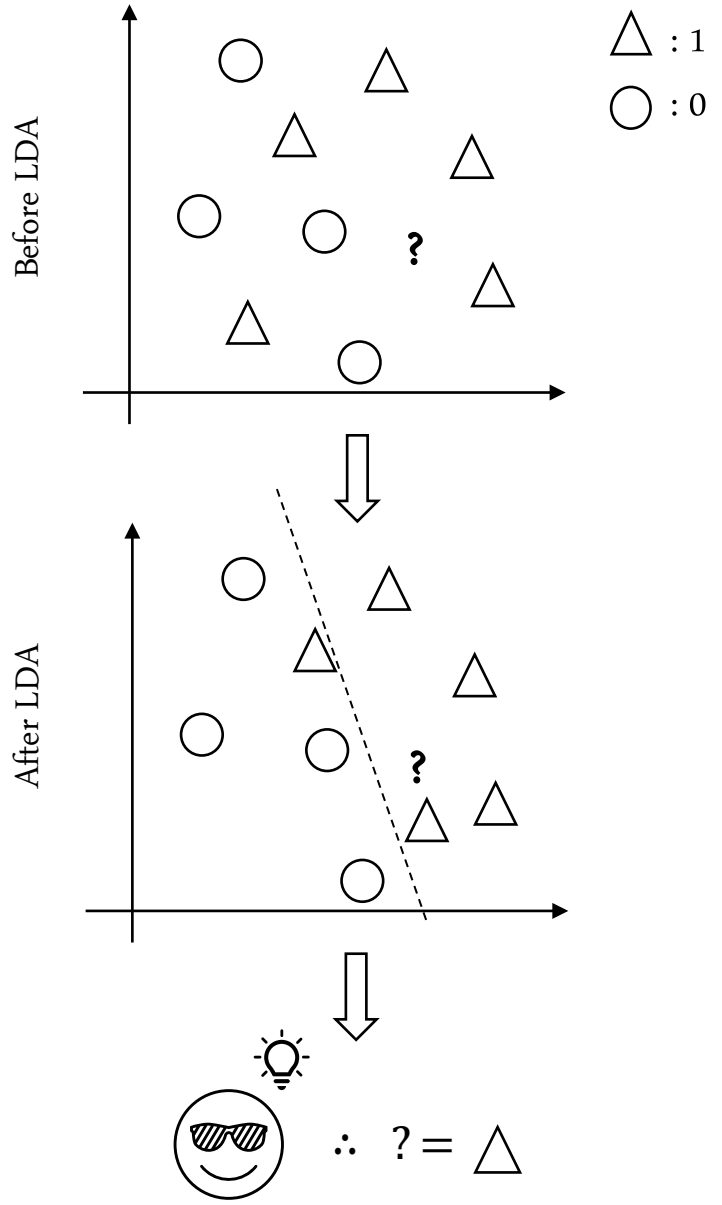
1. During the LDA model fitting process, the algorithm will estimate the means and covariances of each input variable for the two classes (0 and 1).

2. Use these estimates to construct a linear boundary between the two classes.

3. The LDA algorithm assumes that the input variables are normally distributed within each class and that the covariance matrix is the same for both classes.

4. Once the LDA model is fitted, it can be used to make predictions on new data by using Bayes' theorem to calculate the posterior probability of the input data belonging to each class and selecting the class with the highest probability.

The decision boundary between the two classes is a linear function of the input features, which can be visualized as a line in two-dimensional space or a plane in higher-dimensional space.



```
lda_model
```

```
Call:
lda(good ~ ., data = train, family = binomial)

Prior probabilities of groups:
        0         1
0.7901823 0.2098177

Coefficients of linear discriminants:
                              LD1
fixed.acidity          2.526843e-01
volatile.acidity      -9.983086e-01
citric.acid            4.710615e-01
residual.sugar         1.533646e-01
chlorides             -2.197128e+00
free.sulfur.dioxide    1.460021e-02
total.sulfur.dioxide -3.841712e-03
density               -3.454840e+02
pH                     2.544330e+00
sulphates              1.638637e+00
alcohol                4.311189e-01
```

The LDA formula for classifying a new observation based on the coefficients of linear discriminants is:

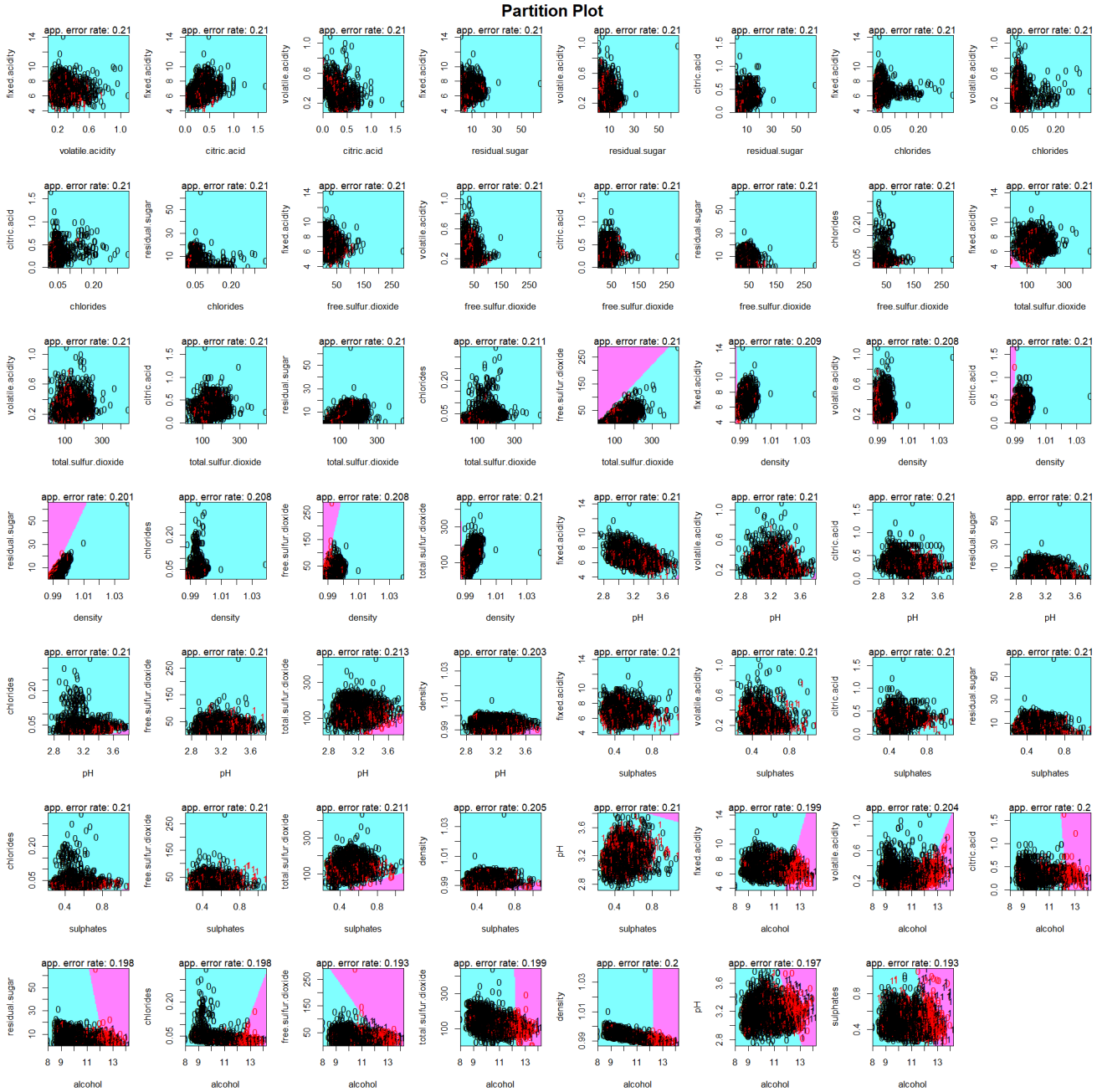
$$Score = LD1(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$$

where:

- $LD_1(X)$ is the linear discriminant score for the observation X
- $X_1, X_2, \ldots, X_p$ are the predictor variables of the new observation
- $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ are the coefficients of the linear discriminant function, which are estimated from the training data using the LDA method.

The predicted class for the new observation is based on the sign of the score. If the score is positive, the new observation is classified as belonging to group 1, and if the score is negative, the new observation is classified as belonging to group 0.

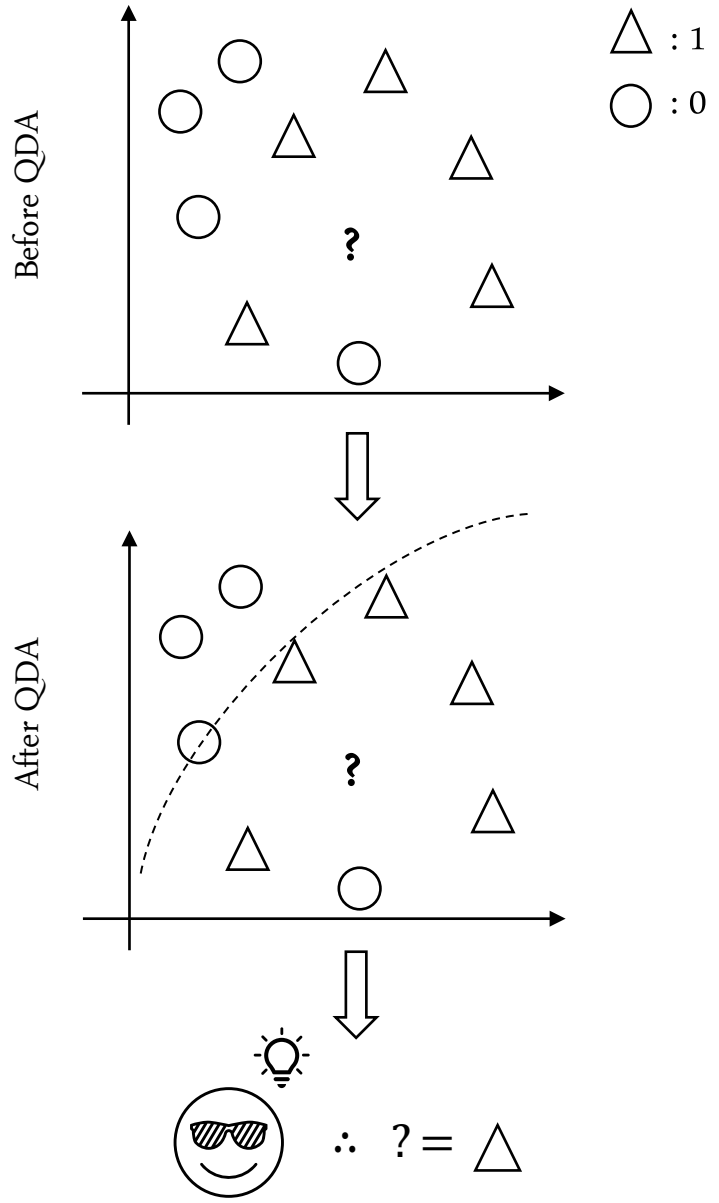
Decision Boundary Plot



Quadratic Discriminant Analysis (QDA)

QDA method is very similar to LDA, except it assumes that the covariance matrix for each class is different and estimates the mean and covariance matrix separately for each class.

It then uses these estimates to calculate the likelihood and posterior probability of each class for new observations. The observation is classified as belonging to the class with the highest posterior probability.



```
qda_model
```

```
Call:
qda(good ~ ., data = train, family = binomial)
c
Prior probabilities of groups:
        0         1
0.7901823 0.2098177

Group means:
  fixed.acidity    volatile.acidity      citric.acid     residual.sugar      chlorides
0      6.877423          0.2834594          0.3358644           6.239439     0.04806922
1      6.683356          0.2690976          0.3288102           4.580949     0.03724465

  free.sulfur.dioxide     total.sulfur.dioxide      density           pH     sulphates
0            35.11164              141.1303        0.9942356     3.185588     0.4871494
1            34.25334              122.1698        0.9920065     3.231016     0.5014305

      sulphates      alcohol
0     0.4871494     10.33893
1     0.5014305     11.56941
```

The formula for QDA is similar to LDA, but it allows for different covariance matrices for each group. The formula for the quadratic discriminant function is:

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i) - \frac{1}{2}\log(|\Sigma_i|) + \log(\pi_i)$$

where:

$g_i(x)$ is the discriminant score for group i

x is a vector of predictor variable values for a specific observation

$\mu_i$is the vector of mean values for the predictor variables for group i

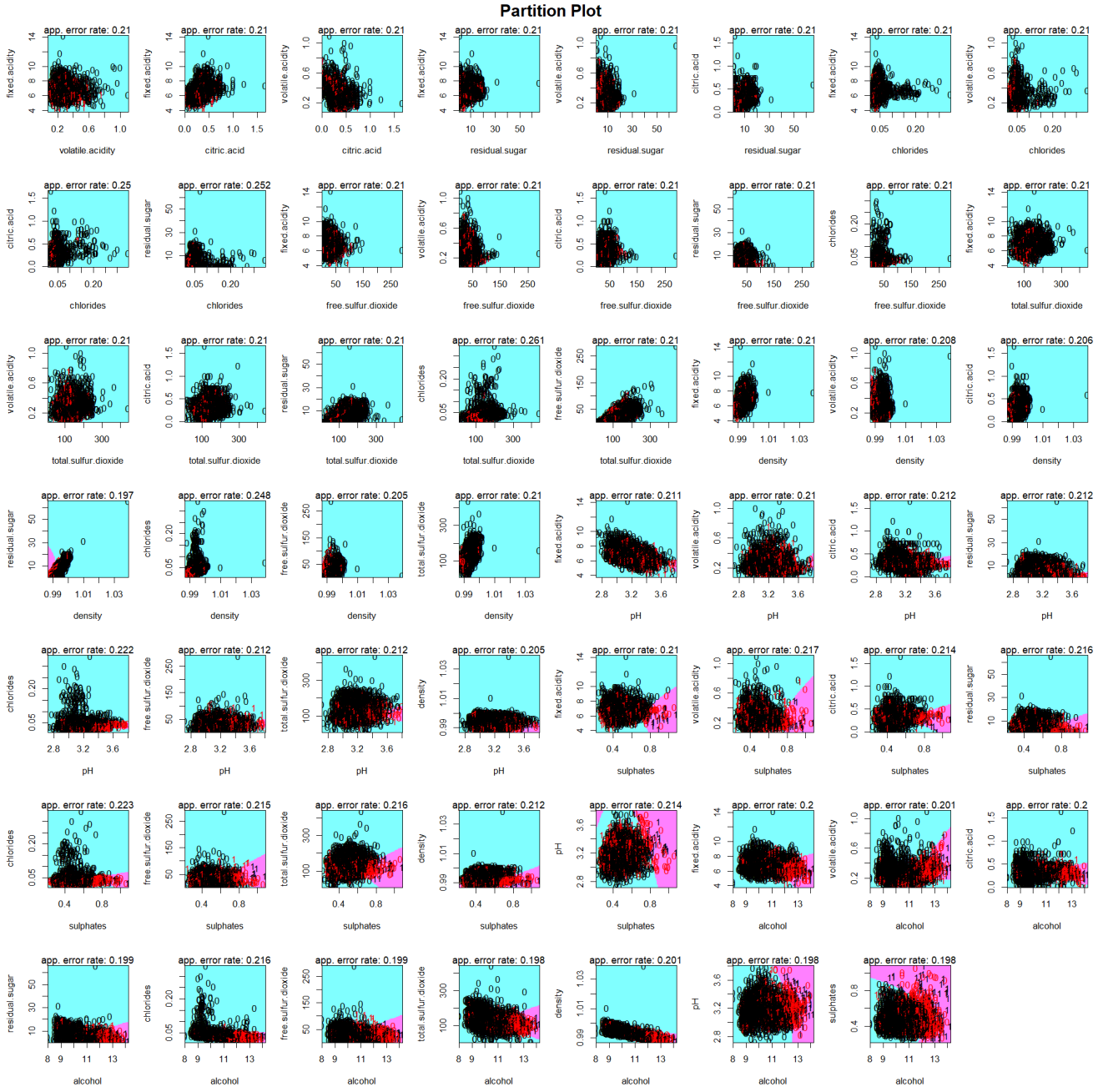
Σ_i is the covariance matrix for group i

$|\Sigma_i|$ is the determinant of the covariance matrix for group i

π_i is the prior probability of group i

The observation is classified into the group with the highest discriminant score.

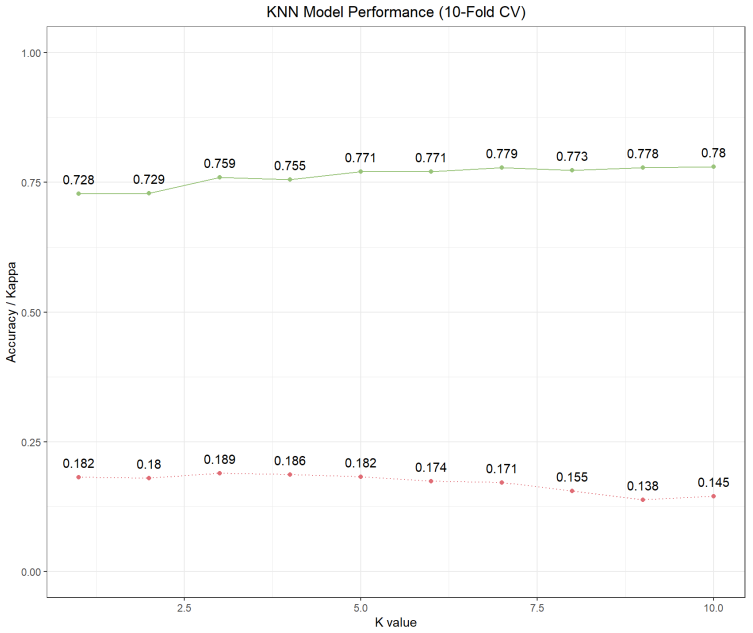
Decision Boundary Plot

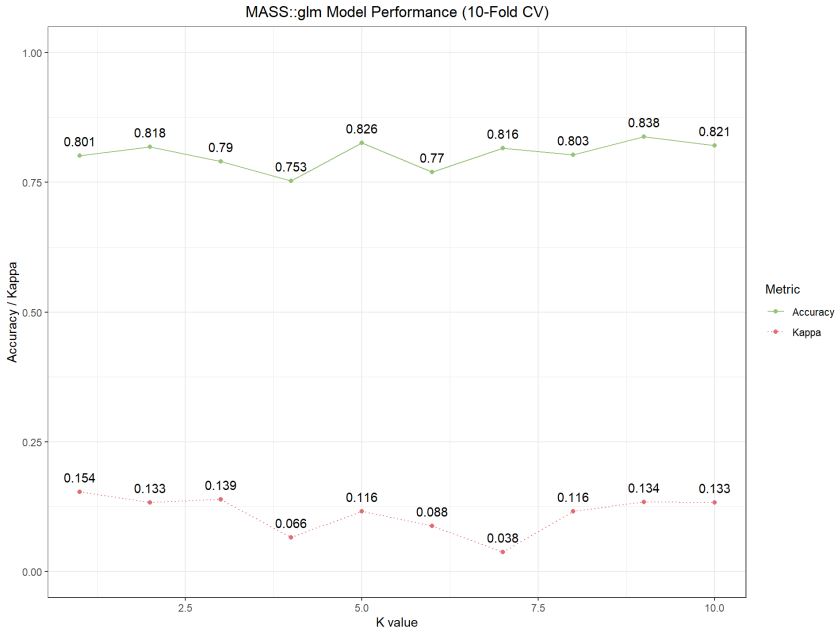


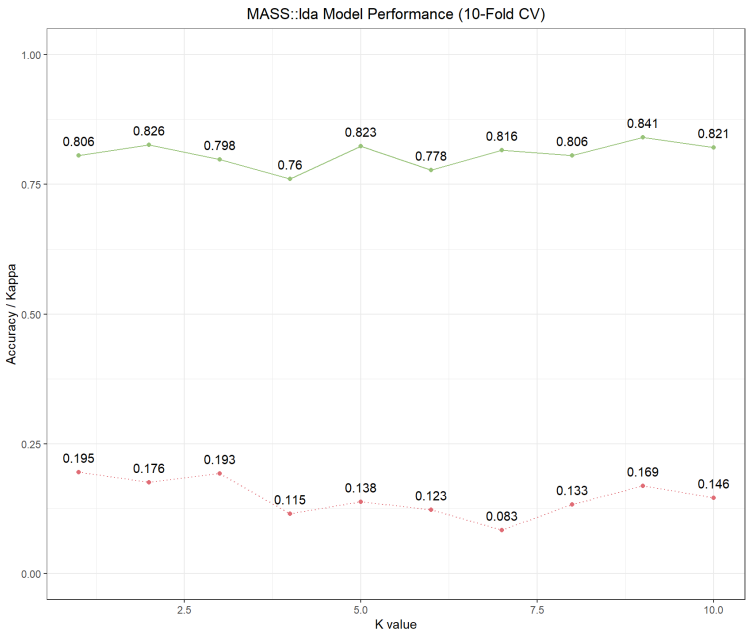
Part.04

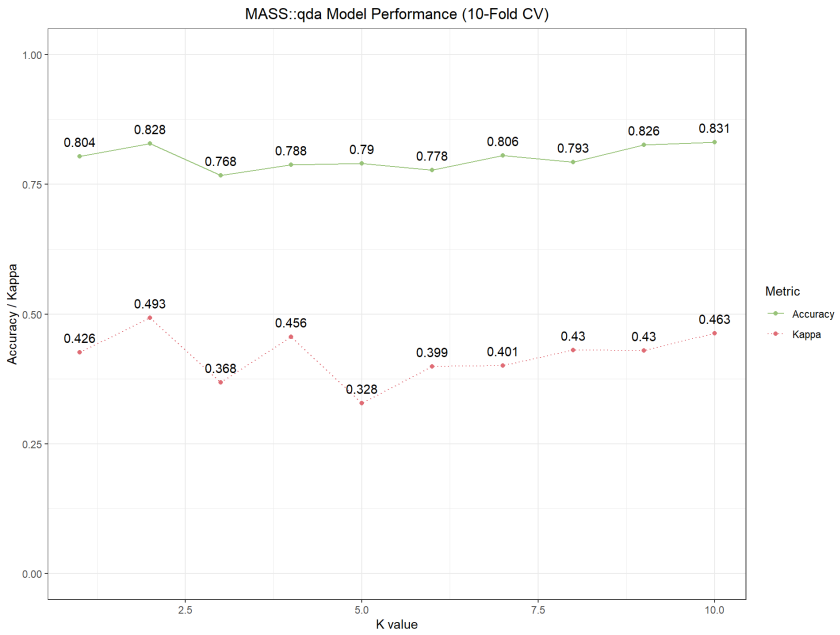
Conclusion

Summary

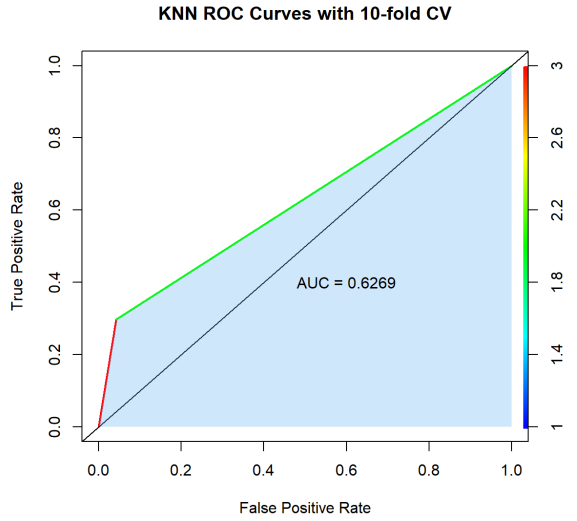

KNN Model Performance (10-Fold CV)
Accuracy / Kappa
K value
Metric
Accuracy
Kappa
0.728 0.729 0.759 0.755 0.771 0.771 0.779 0.773 0.778 0.78
0.182 0.18 0.189 0.186 0.182 0.174 0.171 0.155 0.138 0.145

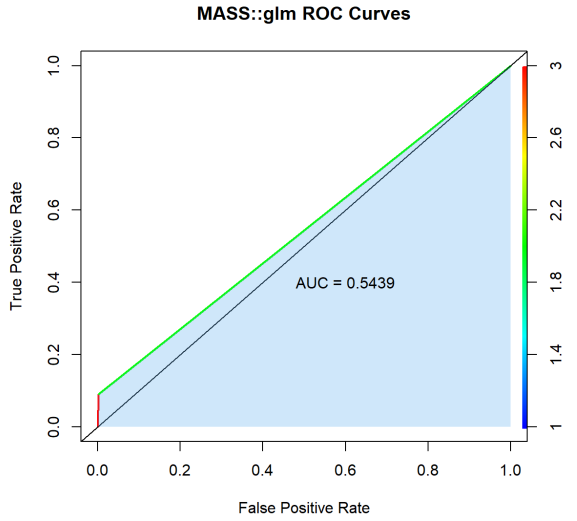

MASS::glm Model Performance (10-Fold CV)
Accuracy / Kappa
K value
Metric
Accuracy
Kappa
0.801 0.818 0.79 0.753 0.826 0.77 0.816 0.803 0.838 0.821
0.154 0.133 0.139 0.066 0.116 0.088 0.038 0.116 0.134 0.133

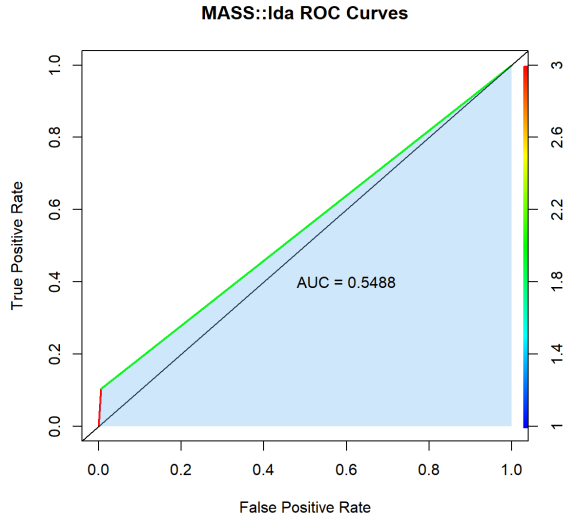

MASS::lda Model Performance (10-Fold CV)
Accuracy / Kappa
K value
Metric
Accuracy
Kappa
0.806 0.826 0.798 0.76 0.823 0.778 0.816 0.806 0.841 0.821
0.195 0.176 0.193 0.115 0.138 0.123 0.083 0.133 0.169 0.146

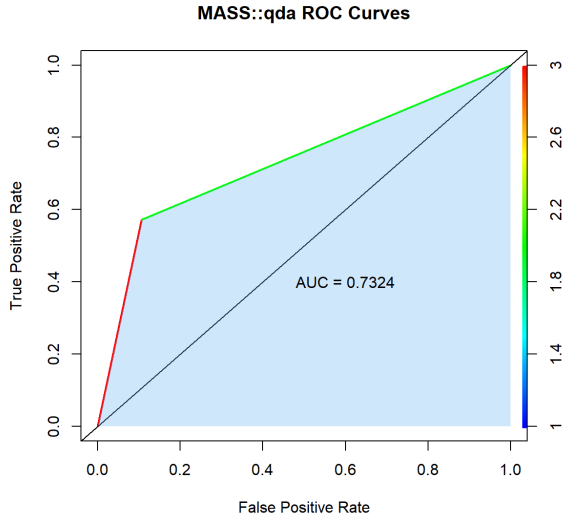

MASS::qda Model Performance (10-Fold CV)
Accuracy / Kappa
K value
Metric
Accuracy
Kappa
0.804 0.828 0.768 0.788 0.79 0.778 0.806 0.793 0.826 0.831
0.426 0.493 0.368 0.456 0.328 0.399 0.401 0.43 0.43 0.463

Summary


KNN ROC Curves with 10-fold CV
True Positive Rate
False Positive Rate
AUC = 0.6269


MASS::glm ROC Curves
True Positive Rate
False Positive Rate
AUC = 0.5439


MASS::lda ROC Curves
True Positive Rate
False Positive Rate
AUC = 0.5488


MASS::qda ROC Curves
True Positive Rate
False Positive Rate
AUC = 0.7324

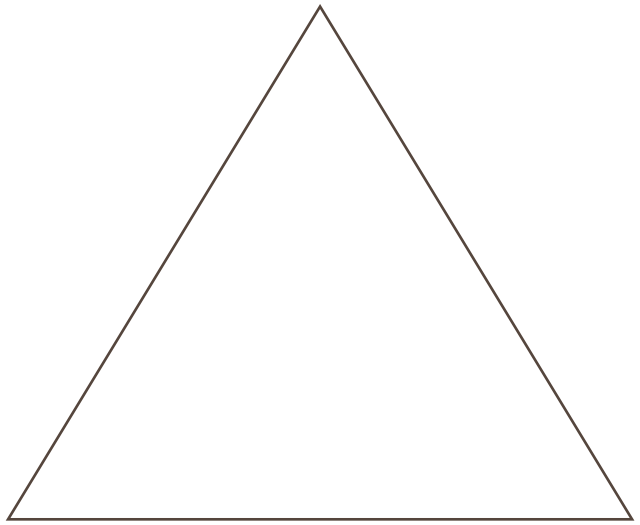
Summary

Model	Resampling Method	Error Rate	Sensitivity	Specificity	AUC
KNN	10-fold CV (k=1:10)	0.1759	0.9568	0.2971	0.6269339
Logit	10-fold CV	0.1616	1.0000	0.0857	0.5438871
LDA	10-fold CV	0.1591	0.9969	0.1143	0.5488133
QDA	10-fold CV	0.1692	0.8934	0.5714	0.7324227

Review

- Imbalance Class Size
- Overfitting Possibility
- Data correlations

Issues



Insight

- Supervised Modeling
- Exposure of `caret` and `MASS` packages

Improvements

- L1 and L2 regularization
- Dropout
- Early stopping
- Dimensionality reduction

Part.05

Further Modeling

Summary

Model	Resampling Method	Error Rate	Sensitivity	Specificity	AUC
KNN	10-fold CV (k=1:10)	0.1759	0.9568	0.2971	0.6269339
KNN	10-fold CV (k=1:30)	0.1961	0.9758	0.1213	0.5485514
KNN	Hold-out CV (k=1:10)	0.1759	0.9568	0.2971	0.6269339
KNN	Hold-out CV (k=1:30)	0.1987	0.0053	0.0335	0.5141020
KNN	LOOCV (k=1:10)	0.1717	0.9621	0.2971	0.6295682
KNN	LOOCV (k=1:20)	0.1995	0.9768	0.1004	0.5386181
KNN	Repeated CV (5×10-fold CV)	0.1776	0.9104	0.4728	0.6916177
KNN	Repeated (rectangular, gaussian, and cosine)	0.1120	0.9547	0.6234	0.7890601
Logit	10-fold CV	0.1616	1.0000	0.0857	0.5438871
Logit	10-fold CV (`caret`)	0.1902	0.9336	0.3180	0.6258030
Logit	10-fold CV (`caret`) with Step AIC	0.1919	0.9895	0.0879	0.5386644
Logit	Hold-out CV	0.1894	0.9884	0.1046	0.5465057
LDA	10-fold CV	0.1591	0.9969	0.1143	0.5488133
LDA	10-fold CV (`caret`)	0.1919	0.9283	0.3305	0.6294448
QDA	10-fold CV	0.1692	0.8934	0.5714	0.7324227
QDA	10-fold CV (`caret`)	0.2559	0.7418	0.7531	0.7474858

Summary

Model	Error Rate	Sensitivity	Specificity	AUC
Naive Bayes	0.2466	0.7829	0.6360	0.7094563
Decision Tree (CART)	0.1827	0.9062	0.4644	0.6853261
Random Forest	0.0471	0.9789	0.8494	0.9141488
Bagging	0.0497	0.9694	0.8745	0.9219593
Boosting	0.1633	0.9389	0.4310	0.6849227
XGBoost	0.1338	0.9589	0.4979	0.7284060
SVM	0.1641	0.9726	0.2929	0.6327449
Neural Network	0.1818	0.9378	0.3431	0.6404628

The End

Questions?

Reference

[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.