STATS5241, Summer 2021
Final Project Report, Group 1

FINAL PROJECT REPORT: TITANIC AND WINE QUALITY

0.INTRODUCTION

In our final project, we choose to use datasets, Titanic and Wine Quality. We were interested in predicting if people could survive in Titanic, and predicting the wine quality in the score from 0 to 10. For Titanic, we applied Bagging, Random Forest, Logistic regression, Linear discriminant analysis (LDA), K-nearest neighbors (KNN) and Support vector machine (SVM). For Wine Quality, we applied Regression tree and Principal component analysis (PCA).

1.DATA ANALYSIS

1.1 Titanic

1.1 Data Preprocessing

Data Summary: There are 3 tables for the Titanic dataset, train (train.csv), test (test.csv), solutions (gender_submission.csv). Training set has 891 observations, while the test set has 418.

Missing Data: Around 77% of Cabin data are missing, it is quite difficult to insert estimated values in these empty slots, so we decided to remove this variable. As for the Age variable, there are 177 missing values in the column, so it is possible to insert estimated values to make the column complete. We found that there are titles(Mr, Mrs, Miss) in the name column for each passenger. Instead of assigning average or median values of all ages, we decided to fill out the missing values by the median age of its title. The two observations missing the 'Embarked' value in the training dataset were removed. The one observation missing the 'Fare' value in the test dataset was removed. The only one observation missing the 'Fare' value was removed.

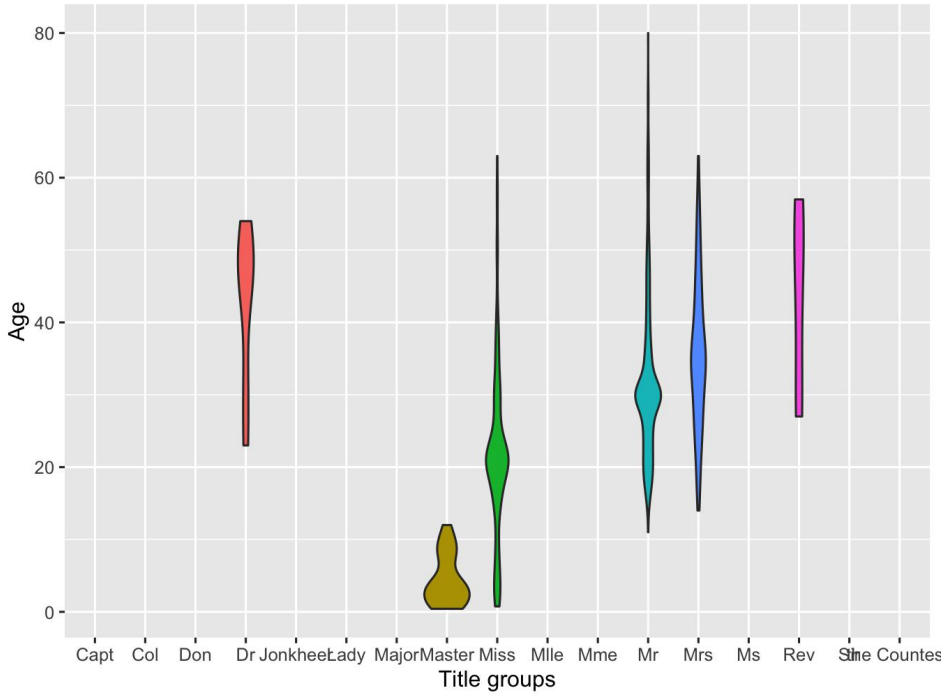
```
> colSums(is.na(titanicTrain))
PassengerId    Survived      Pclass        Name
          0           0           0           0
        Sex         Age       SibSp       Parch
          0         177           0           0
     Ticket        Fare       Cabin    Embarked
          0           0         687           2
```

Other:We replaced names with titles (Mr., Mrs., Miss, etc).

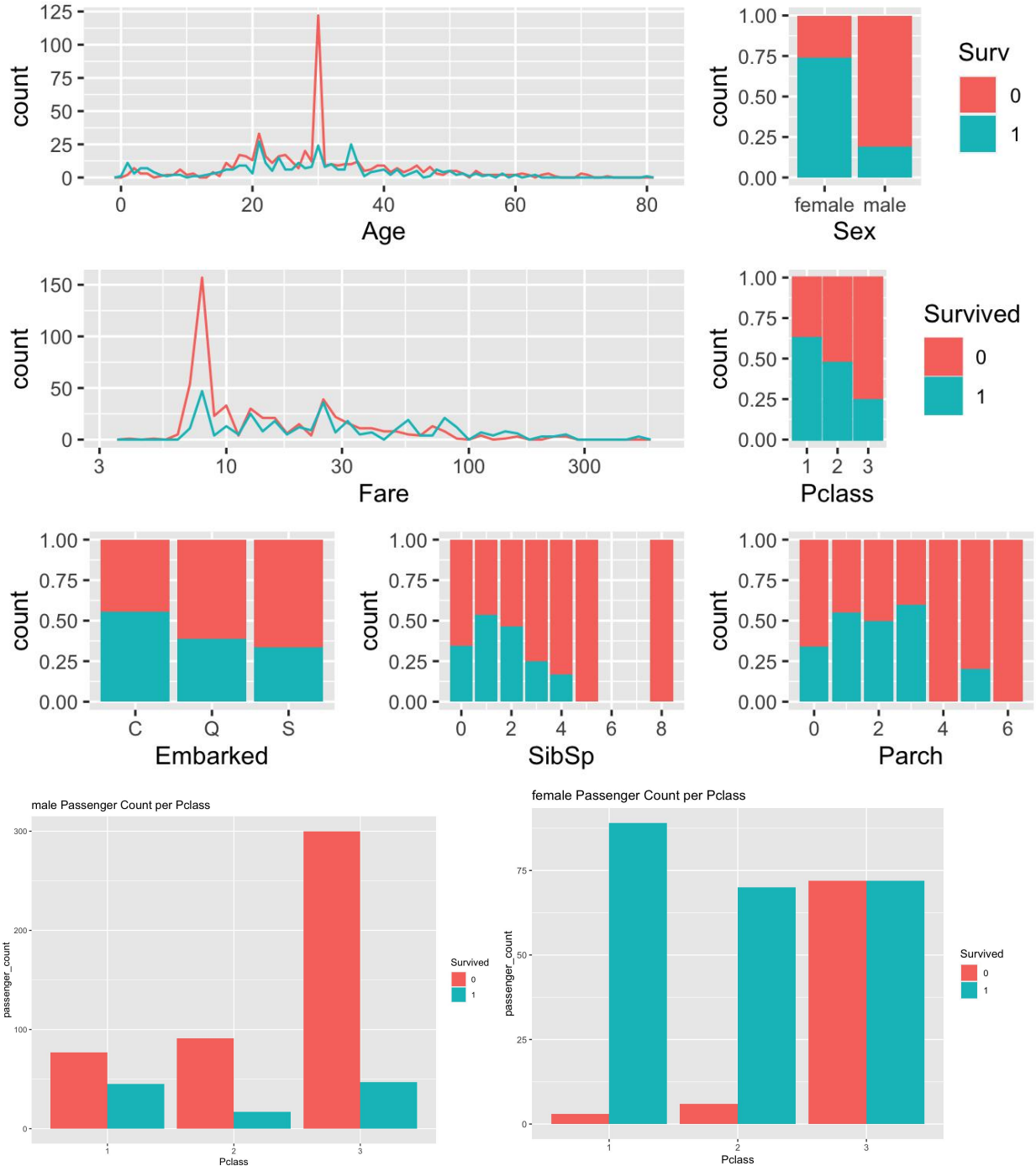
```
> unique(train$Title)
 [1] "Mr"           "Mrs"          "Miss"         "Master"       "Don"          "Rev"          "Dr"
 [8] "Mme"          "Ms"           "Major"        "Lady"         "Sir"          "Mlle"         "Col"
[15] "Capt"         "the Countess" "Jonkheer"
```

We also removed ‘PassengerId’ and ‘Ticket’. ‘Solutions’ that contain ‘Survived’ were merged with the test set.

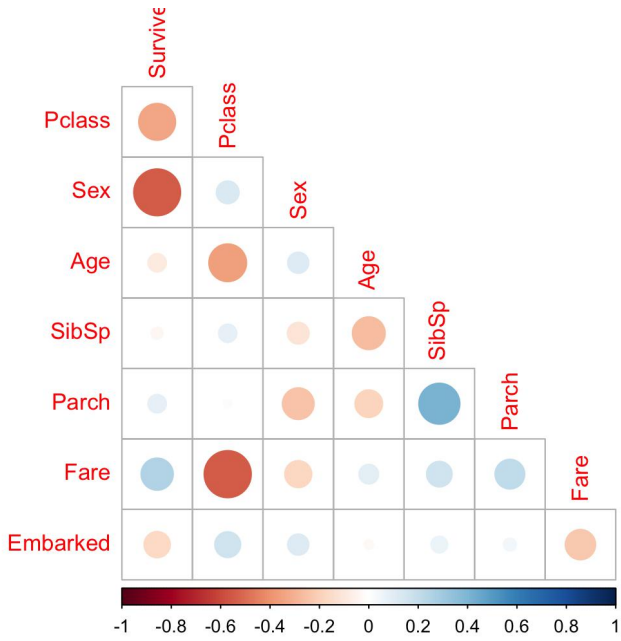
1.2 EDA



The violin age density plot grouped by title shows that the median age indeed varies a lot by different name titles, so our method to assign missing age value is reasonable.



There's a peak of people not-survived at age 25. The age of people survived concentrates on the range of 15 to 40. Male survival rate is generally low in the lower class ('Pclass'=3), where it has the largest number of people compared to upper and middle class. Males survived from the upper and lower class are both slightly less than 50, while the one in the middle class is around 20. Female survival rate is generally higher than male's among all the 3 classes. There's a tendency that the survival rate grows as the class goes higher. At the lower class, the amount of people surviving and not-surviving are equal.



'Survived' is correlated with 'Sex' most , and then with 'Pclass', which are proved by previous plots as well. The Pclass and Age are correlated, the richer the passengers, the older they are. The Pclass and Fare are correlated, because better classes are more expensive.

1.3 Modeling

We took 'Survived' as the response variable. There are 8 predictive variables.

1). Bagging

```
baggingModel = randomForest(Survived ~ ., data = train, mtry = 8, importance = TRUE)
test = rbind(train[1,], test)
test = test[-1,]
yhatBagging = predict(baggingModel, test)
```

Bagging is quite similar to the random forest method, except it uses all predictors when growing branches, so we set mtry = 8 in the function to ensure it is a bagging method.

```
> confusionMatrix(yhatBagging, test$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 223  33
         1  42 118

               Accuracy : 0.8197
                 95% CI : (0.7793, 0.8555)
    No Information Rate : 0.637
    P-Value [Acc > NIR] : 2.321e-16
```

Bagging method' s accuracy is 0.8197, which is acceptable. Because all trees in the bagging method are not strictly independent, we expect the random forest method to perform better.

2). Random Forest

```r
# random forest
rfModel = randomForest(Survived ~ ., data = train, importance = TRUE)
yhatRF = predict(rfModel, test)
```

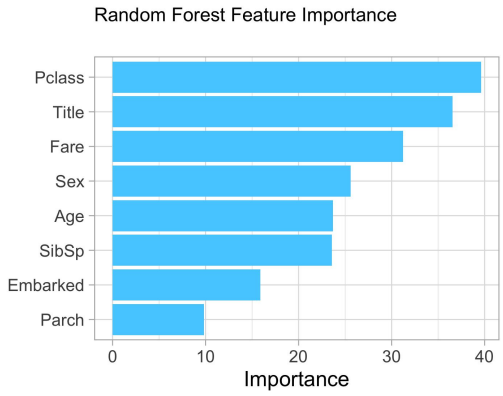
The only difference between bagging method and random forest is that we do not specify mtry value in the function. So the function will use the default value of mtry = sqrt(p), that means when growing trees' branches, only sqrt(p) number of predictors will be chosen from.

```
> confusionMatrix(yhatRF, test$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 245  28
         1  20 123

               Accuracy : 0.8846
                 95% CI : (0.8499, 0.9137)
    No Information Rate : 0.637
    P-Value [Acc > NIR] : <2e-16
```

The accuracy of the random forest method is 0.8846 which is higher than bagging. Since they choose predictors only among sqrt(p) of all predictors, tresses in random forest are more independent, thus the results are more accurate.

```
> importance(rfModel)
                0         1 MeanDecreaseAccuracy MeanDecreaseGini
Title    33.069237 16.966955             36.542160         73.37812
Pclass   20.387643 34.034564             39.610850         32.36246
Sex      24.547380  9.321530             25.606509         56.37896
Age      12.991731 19.617477             23.686793         41.77163
SibSp    24.535421  2.527753             23.562966         17.70048
Parch     7.154369  5.449963              9.817868         10.58716
Fare     19.248883 24.142681             31.210946         54.15363
Embarked  6.267404 14.426380             15.877919         10.21514
```

We can also output the predictors' importance from the random forest model. To make it easy to analyse, we create an important plot.



We can see that when using random forest to predict the survived response, 'Pclass' has the most important role followed by Title. And titles are also relevant to passengers' genders and status, so the feature importance results also consist with previous analysis.

3) Logistic Regression

```
#logistic regression
fullLogModel = glm(Survived ~ ., train, family = binomial)
summary(fullLogModel)
library(MASS)
logModel = stepAIC(fullLogModel, direction = "both", trace = TRUE)
library(car)
vif(logModel)
yhatLog = ifelse(predict(logModel, test, type = "response") > 0.5, 1, 0)
#logistic accuracy
mean(test$Survived == yhatLog)
#logistic confusion matrix
confusionMatrix(as.factor(yhatLog), test$Survived)
```

We also want to use logistic regression to predict the survived response variable. We use general linear function (glm) here and set family = binomial to specify this is a logistic regression whose actual response will be log(odd ratio). When predicting survived status we want to get survived status instead of odd ratio, so we set type = "response" to get Pr(Y=1), and use ifelse to set survived being 1 when Pr(Y=1)>0.5 or else being 0. Since there are 8 variables in the dataset, we also need to proceed model selection.

```
Step:  AIC=766.48
Survived ~ Title + Pclass + Age + SibSp + Parch + Fare

           Df Deviance     AIC
<none>          720.48  766.48
+ Sex       1   718.96  766.96
- Fare      1   723.24  767.24
+ Embarked  2   717.88  767.88
- Parch     1   727.94  771.94
- Age       1   730.89  774.89
- SibSp     1   748.63  792.63
- Pclass    2   771.35  813.35
- Title    16  1019.50 1033.50
```

```
> vif(logModel)
            GVIF Df GVIF^(1/(2*Df))
Title   2.863891 16        1.033427
Pclass  2.045297  2        1.195884
Age     1.967064  1        1.402520
SibSp   1.593721  1        1.262426
Parch   1.450469  1        1.204354
Fare    1.588468  1        1.260345
```

We use AIC as selection criteria and proceed with both forward and backward selection. Finally we select 6 of 8 predictors, which are less correlated with each other according to the vif table.

```
> confusionMatrix(as.factor(yhatLog), test$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 244   8
         1  21 143

               Accuracy : 0.9303
                 95% CI : (0.9014, 0.9528)
    No Information Rate : 0.637
    P-Value [Acc > NIR] : < 2e-16
```

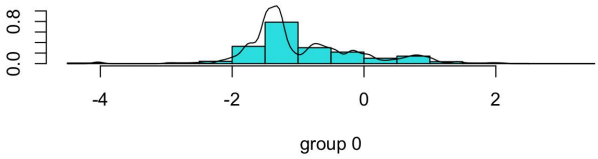
```
Pclass2          -1.026e+00  3.288e-01  -3.120  0.00181 **
Pclass3          -2.139e+00  3.214e-01  -6.653 2.87e-11 ***
Sexmale          -1.712e+01  2.400e+03  -0.007  0.99431
Age              -2.966e-02  9.695e-03  -3.059  0.00222 **
SibSp            -5.635e-01  1.265e-01  -4.454 8.41e-06 ***
Parch            -3.444e-01  1.344e-01  -2.562  0.01040 *
```

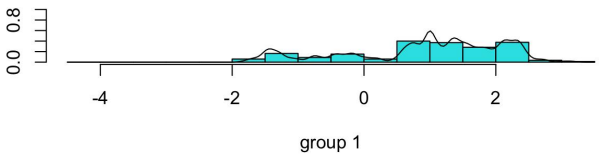
The accuracy of the fitted model on the test dataset is 0.9303, which is quite satisfactory. Also we can use the model summary to check significance of each predictor, which indicates that Pclass , Age and SibSp are significant predictors. In a word, logistic regression is quite good in this case since it not only can provide high accuracy but also can provide significant information for predictors.

4) LDA

```
#LDA
library(MASS)
temp = train %>% preProcess(method = c("center", "scale"))
normalTrain = temp %>% predict(train)
temp = test %>% preProcess(method = c("center", "scale"))
normalTest = temp %>% predict(test)
ldaModel = lda(Survived ~ ., normalTrain)
yhatLDA = predict(ldaModel, normalTest)$class
#LDA accuracy
mean(yhatLDA == test$Survived)
#LDA confusion matrix
confusionMatrix(yhatLDA, test$Survived)
```

We use the LDA method to build the model, before that we have to normalize both training and test dataset.





After building the LDA model, we can plot the density line of two classes. There are some overlapping parts between two density functions and not very 'normal' as expected.

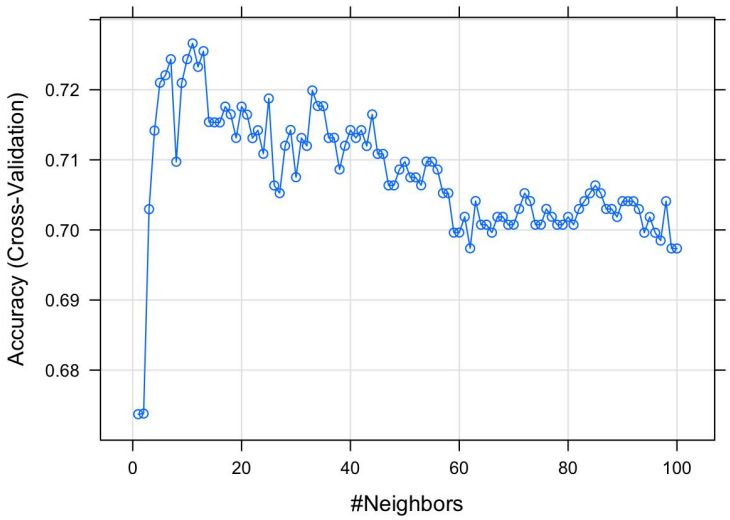
```
> confusionMatrix(yhatLDA, test$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 246   5
         1  19 146

               Accuracy : 0.9423
                 95% CI : (0.9154, 0.9627)
    No Information Rate : 0.637
    P-Value [Acc > NIR] : < 2.2e-16
```

Even though the density plots seem not ideal, the accuracy is quite satisfactory, which is 0.9627. So the LDA can provide really accurate results in this case.

5) KNN

```
#KNN
#using cv to select K
library(caret)
trainScaled = train %>%
  select(c('Age', 'SibSp', 'Parch', 'Fare')) %>%
  scale() %>%
  as_tibble() %>%
  cbind(Survived = train$Survived)
trControl = trainControl(method = "cv", number = 10)
cvKNN = train(Survived ~ .,
      method     = "knn",
      tuneGrid   = expand.grid(k = 1:100),
      trControl  = trControl,
      metric     = "Accuracy",
      data       = trainScaled)
```

When using KNN, we have to choose a tuning parameter k. Cross validation is used to find out the best k, which can provide the highest accuracy.




```
> cvKNN$bestTune
    k
11 11
```

By comparing the cv accuracy plot, we can find out the best k is 11 from the training dataset. So we use k=11 as the tuning parameter value to build the KNN model and predict the result.

```
> confusionMatrix(yhatKNN, test$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 245   6
         1  20 145

               Accuracy : 0.9375
                 95% CI : (0.9098, 0.9588)
    No Information Rate : 0.637
    P-Value [Acc > NIR] : < 2e-16
```

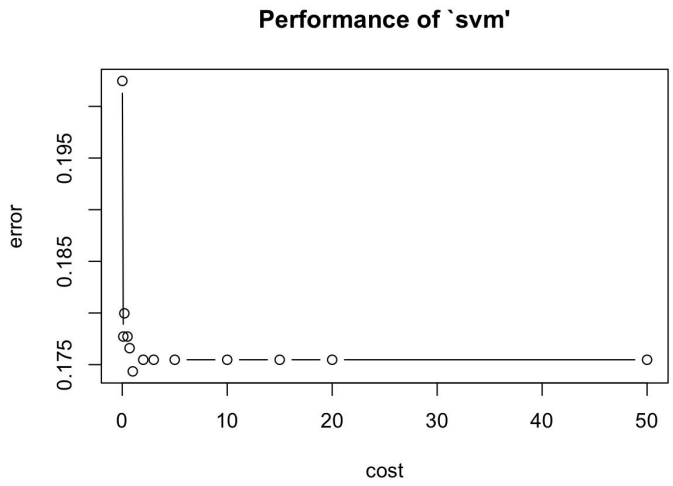
The accuracy is 0.9375, which is also of high performance.

6) SVM

Linear kernel

```
#linear
svmLinearModel=tune.svm(Survived~.,data=train,kernel="linear",cost=c(0.01,0.1,0.2,0.5,0.7,1,2,3,5,10,15,20,50))
```

We can choose different kernels in the SVM method and will try the linear kernel first. There is a tuning parameter cost in the linear kernel and we still use CV to choose that.




```
> svmLinearModel$best.parameters
  cost
6    1
```

According to CV results, cost = 1 is the best tuning value. So we use cost = 1 to build the linear SVM model and make predictions.

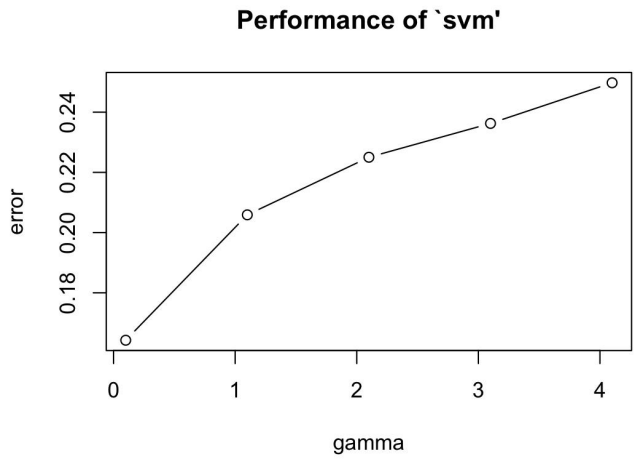
```
> confusionMatrix(yhatSVM.linear, test$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 248   4
         1  17 147

               Accuracy : 0.9495
                 95% CI : (0.9239, 0.9685)
    No Information Rate : 0.637
    P-Value [Acc > NIR] : < 2.2e-16
```

The accuracy is 0.9495, which also provides high performance.

Radial kernel

```
#radial
svmRadialModel = tune.svm(Survived~.,data=train,kernel="radial",gamma=seq(0.01,5))
```

Next we will try the radial kernel in the SVM method. The tuning parameter gamma also need to be chosen by CV.




```
> svmRadialModel$best.parameters
  gamma
1  0.01
```

According to the CV results, the smallest gamma has the best result. So we use gamma = 0.01 to build the model and make predictions.

```
> confusionMatrix(yhatSVM.radial, test$Survived)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 254   4
         1  11 147

               Accuracy : 0.9639
                 95% CI : (0.9412, 0.9797)
    No Information Rate : 0.637
    P-Value [Acc > NIR] : <2e-16
```

The accuracy is 0.9639, which is higher than the linear kernel. So it seems that using a more complicated kernel indeed improves the accuracy in this case

1.5 Conclusion

According to the EDA, random forest model and logistic regression model, we can conclude that gender and cabin class are key factors to determine survival. Women are more likely to survive than men, so the "women first" is indeed true in emergency situations. The higher the cabin class is, the passengers are more likely to survive. So the rich people having more privilege also tend to survive compared to passengers in lower cabin classes. The accuracy of the model we used are all higher than 0.8. Among them logistic regression, LDA, KNN and SVM have accuracy that are more than 0.9, which indicates we did a really good job of predicting survival status.

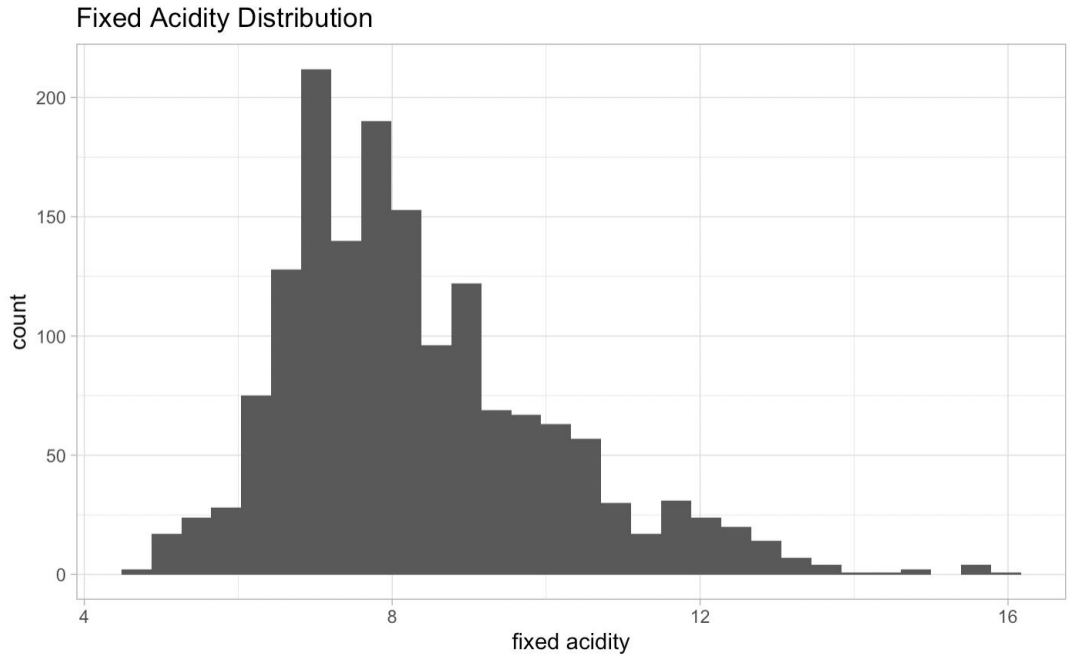
1.2 Wine Quality

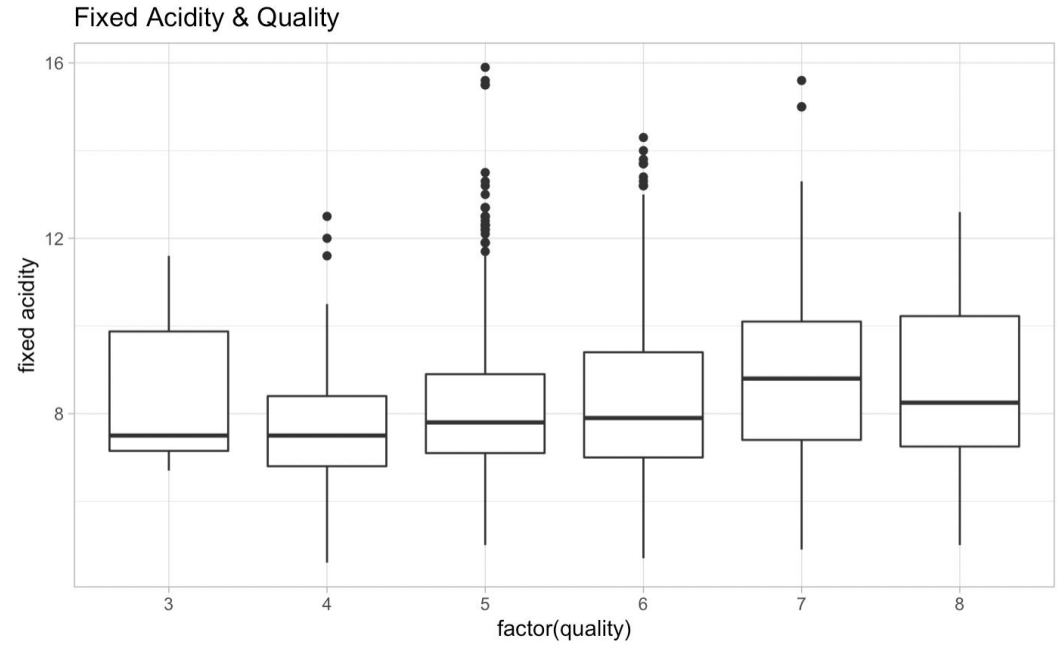
1.1.1 Data Preprocessing

Data Summary: We used table winequality-red.csv, which contains 12 columns initially. We choose quality to be the response variable. There are 1599 observations in this dataset, and we sampled 1066 to be the training set. There is no need for data cleaning.

1.1.2 EDA

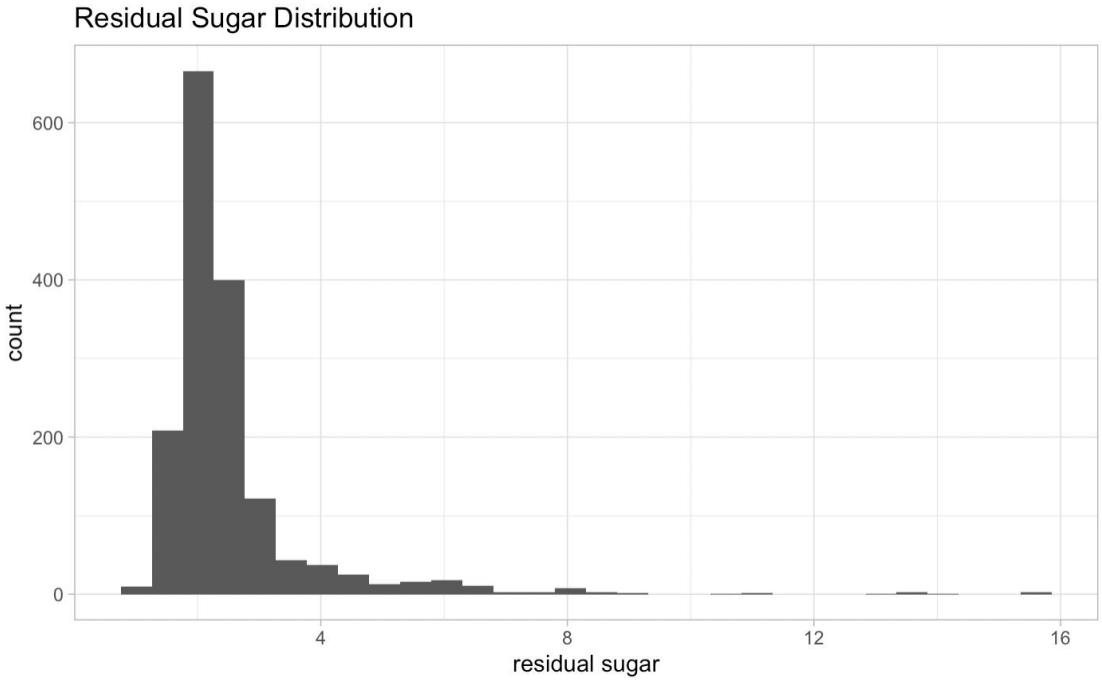
By checking the correlation of each feature with the quality using histogram, boxplot and median table of each quality, we found there are 9 features that have potential impacts on red wine quality. They are fixed acidity, volatile acidity, citric acid, chlorides, free SO2, total SO2, density, pH, sulphates & alcohol. Take fixed acidity and Residual Sugar as examples.

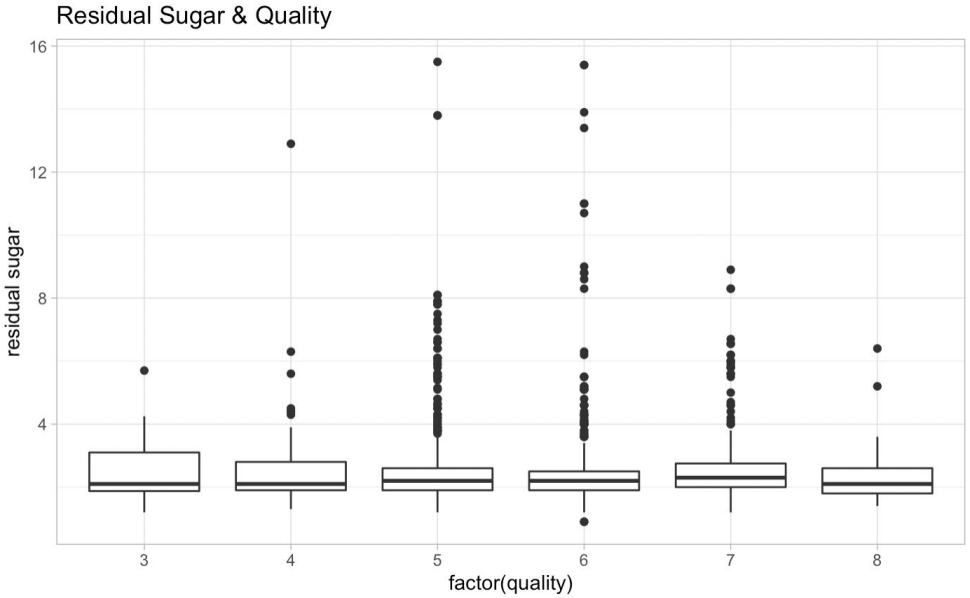




quality <dbl>	median(`fixed acidity`) <dbl>
3	7.50
4	7.50
5	7.80
6	7.90
7	8.80
8	8.25

It seems fixed acidity has a tendency of growing as quality increases.





quality <dbl>	median(`residual sugar`) <dbl>
3	2.1
4	2.1
5	2.2
6	2.2
7	2.3
8	2.1

It looks like residual sugar has no correlation with quality.

1.2.3 Modeling

0)Split data

We use the inbuilt sample() function with fixed seed in ISLR to separate the dataset. The size of the training set is 1599 and that of the test set is 1066.

1)Linear Model

We use the glm() function instead of lm() to perform the model to match our coding structure. The result is as follows:

```
[1] "Training error: 0.426181278306232"
[1] "Test error: 0.402411922216154"
```

By checking p-value of variables, we find that volatile acidity, chlorides, alcohol, total sulfur dioxide and sulphates are significant ones.

2) Stepwise Regression

Judged by AIC, we perform the forward regression. The model is as follows:

```
Call:
lm(formula = quality ~ alcohol + `volatile acidity` + sulphates +
    chlorides + pH + `total sulfur dioxide` + `free sulfur dioxide`,
    data = wine, subset = train)

Coefficients:
          (Intercept)                alcohol      `volatile acidity`                sulphates
             5.048094               0.276941               -0.932469                0.862492
           chlorides                      pH  `total sulfur dioxide`   `free sulfur dioxide`
            -2.409495              -0.630424               -0.003808                0.005184
```

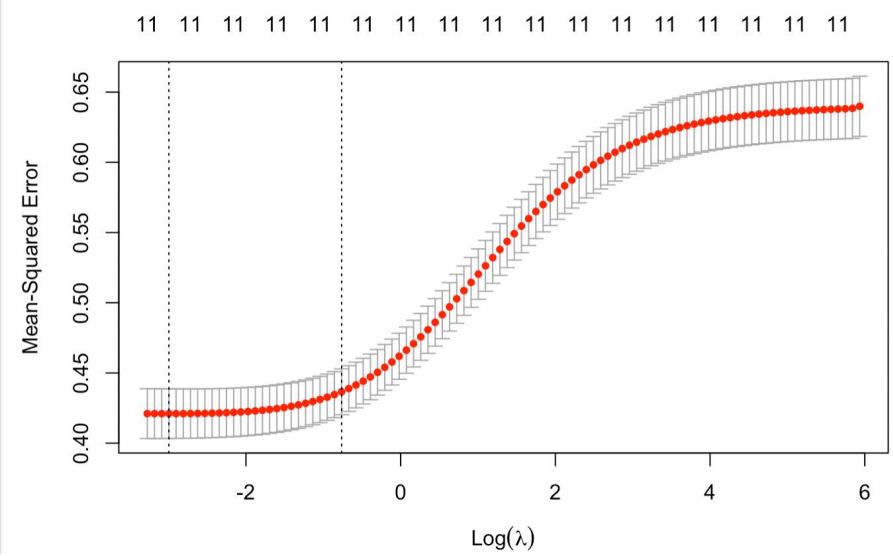
The following is the result:

```
[1] "Training error: 0.42658253524676"
[1] "Test error: 0.403272567709995"
```

It seems not better than the full model. But the coefficients are certainly more relevant.

3) Ridge Regression

First, we use cross validation to pick proper λ. We found that the λ which minimizes MSE is 0.04388198.



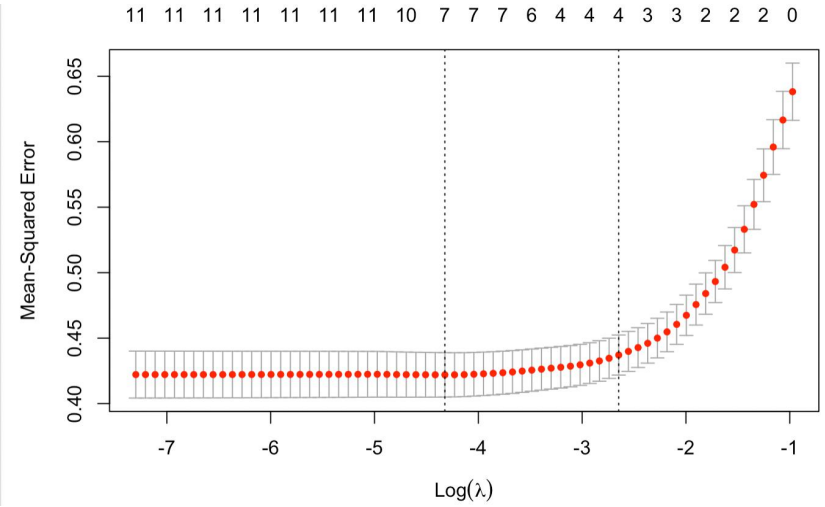
MSE of different λ in ridge regression

We use this λ to make predictions and get the following results.

```
[1] "Training error: 0.426734968486393"
[1] "Test error: 0.404402494017777"
```

4) LASSO regression

First, we use cross validation to pick proper λ. We found that the λ which minimizes MSE is 0.008033642.



MSE of different λ in LASSO regression

We use this λ to make predictions and get the following results.

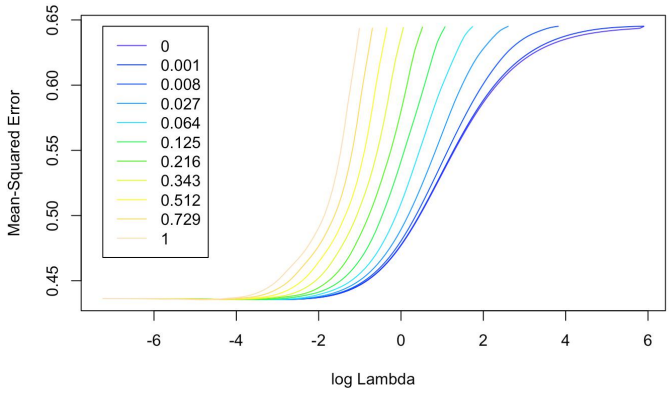
```
[1] "Training error: 0.429143813530265"
[1] "Test error: 0.396227919983233"
```

Below are the coefficients, by which we can find that fixed acidity, citric acid and density are not used in the lasso model.

(Intercept)	fixed acidity	volatile acidity	citric acid
4.175491045	0.000000000	-1.023622631	0.000000000
residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide
0.001672095	-1.727042306	0.002575543	-0.002720052
density	pH	sulphates	alcohol
0.000000000	-0.387343293	0.823234299	0.285318396

5) Elastic Net Regression

We use cross validation to pick the best α. The following are the plot and table of mean-squared errors of different α-λ pairs.



alpha <dbl>	min_mse <dbl>	min_lambda <dbl>
0.000	0.4355608	0.043881982
0.001	0.4355540	0.043881982
0.008	0.4355398	0.046611082
0.027	0.4355177	0.042175854
0.064	0.4355082	0.037452430
0.125	0.4355731	0.030533048
0.216	0.4357416	0.023358156
0.343	0.4357435	0.017717656
0.512	0.4356263	0.014296789
0.729	0.4354976	0.010041092
1.000	0.4353926	0.008033642

11 rows

It is clear that when α=1, the elastic net reaches its most optimal MSE, which is exactly LASSO.

6)principal components regression (PCR)

We choose to scale the data and turn on cross validation in the function while getting the following result.

```
[1] "Training error: 0.633817035400934"
[1] "Test error: 0.451693160294395"
```

It does not give out a satisfying result.

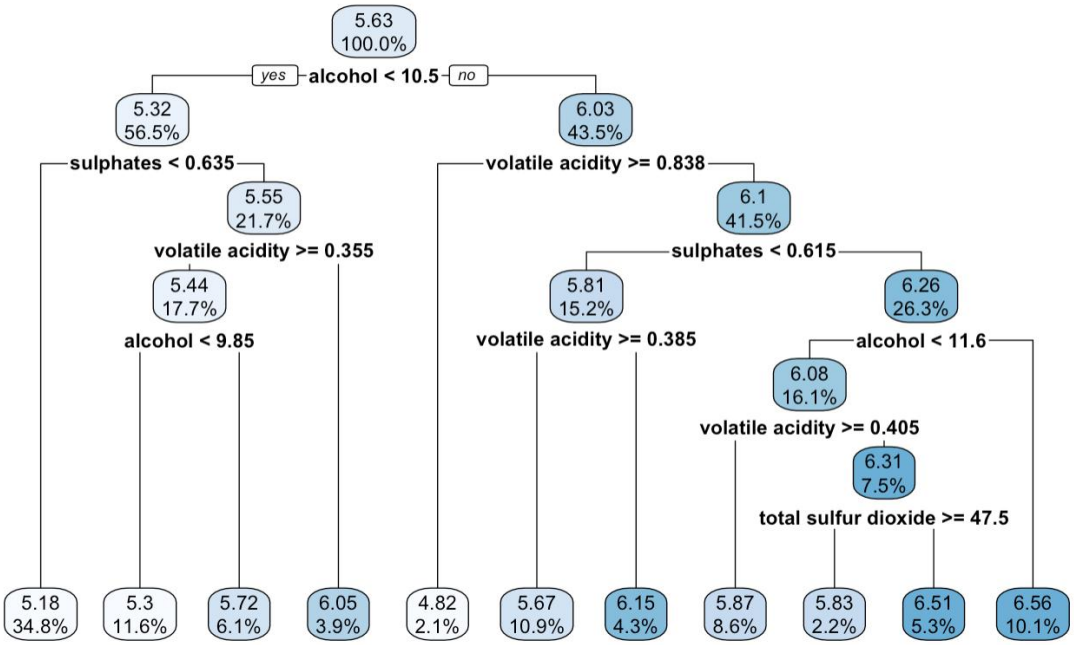
7)Partial Least Squares and Principal Component Regression (PLSR)

We choose to scale the data and turn on cross validation in the function while getting the following result.

```
[1] "Training error: 0.447010779027326"
[1] "Test error: 0.419918935598852"
```

8) Regression Tree

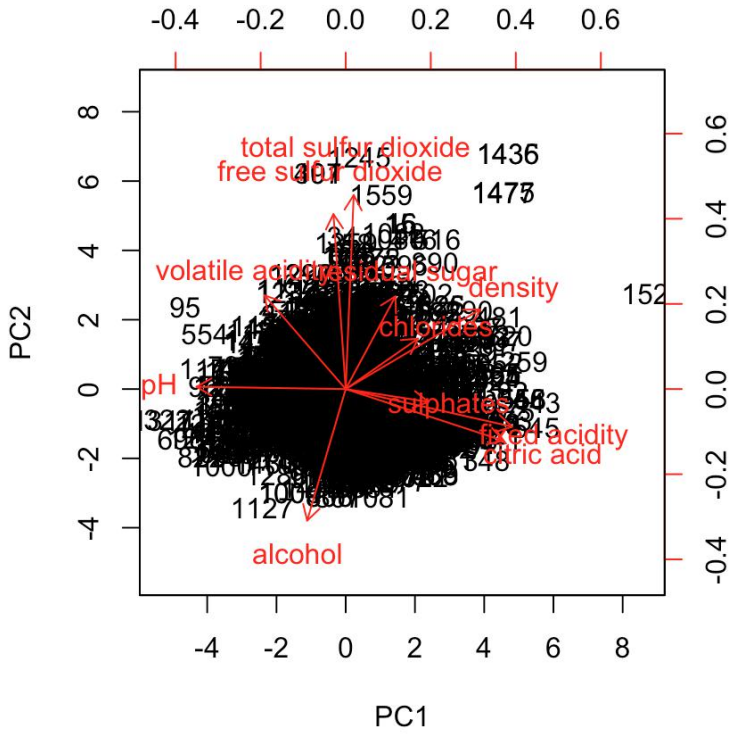
In this part we chose to use the rpart package to get a regression tree. The tree model is as follows.




```
[1] "Training error: 0.407716497944043"
[1] "Test error: 0.472685397083195"
```

According to the above result, there seems to be a little bit of overfitting.

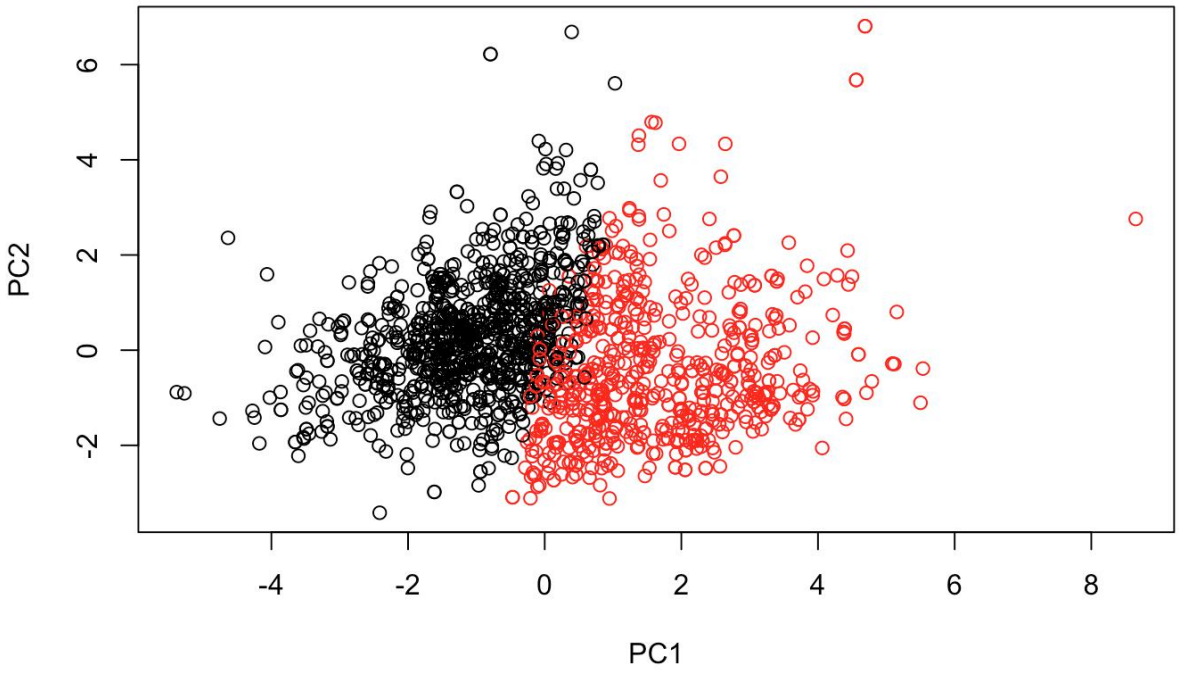
9) PCA

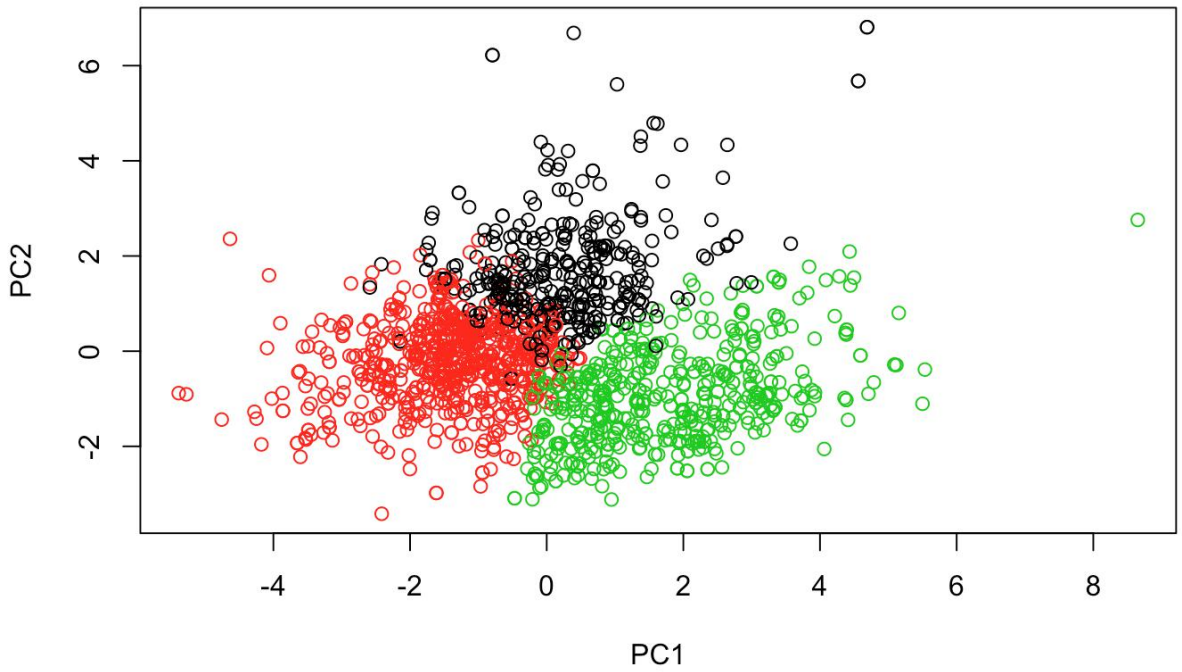


After PCA, we can see that fixed acidity, citric acid and pH are among the most influential variables, which is not the same as the significance we found in the above models.

8) K-means clustering

We use the output matrix of PCA and try to do clustering. The following are two-cluster and three-cluster results.





By checking the new labels we can find that it does not have a strong relationship with the response variable.

2.CONCLUSIONS

For the Titanic dataset, radial SVM performed the best with highest accuracy 0.9639. For the Wine Quality dataset, LASSO regression gave the best result with the least test error 0.396 of the methods we have examined. For the titanic dataset, we can conclude that gender and cabin class are key factors to determine survival. Women are more likely to survive than men, so the "women first" is indeed true in emergency situations. The higher the cabin class is, the passengers are more likely to survive. So the rich people having more privilege also tend to survive compared to passengers in lower cabin classes. For the wine dataset, we find that volatile acidity, chlorides, alcohol, total sulfur dioxide and sulfites are significant ones. The performance of methods that can be easier to interpret, such as the regression tree, is relatively worse. Clustering methods do not work well for the Wine Quality dataset.

From the results, we can conclude that statistical modelling methods indeed have a better interpretability of modeling structure and procedure while machine learning methods have a moderate high modeling accuracy, which fits the statement of reference[1].

3.REFLECTIONS

For the titanic dataset, we try different classification methods, but there are still some methods that we have not covered yet. We should do the QDA model after LDA, but there are several bugs that we still can not solve yet. So we can not present this part in the report, but we will still try to apply this method after the deadline.For the Wine Quality dataset, we can combine wine-quality-red.csv with wine-quality-white.csv to find out what the types of wine matter and how they differ from each other. We have done

some EDA on wine-quality-white.csv and found out not only the distribution of data centers in a much smaller area but the range of the response variable shrink as well. While we are running models, the methods of tree in ISLR cannot work and we cannot make it out. So we use another package instead.
In general, we can try some “statistically reinforced machine learning (SML)” methods which take nonlinear associations and higher-order interactions into account automatically, while testing statistical significance and thereby conducting variable selection.[1] It keeps moderate interpretability of modeling structure while showing moderate high modeling accuracy. However, methods combining these fascinating features are relatively low in numbers and not embedded in our common packages. It might be appropriate to try these methods afterwards.

REFERENCE

[1]Statistically reinforced machine learning for nonlinear patterns and variable interactions
MASAHIRO RYO AND MATTHIAS C. RILLIG

APPENDIX

In the final project, the group work distributions are as follows. Dawei He and Deshu Chen wrote the code for analyzing Titanic and Wine Quality. Wuyin Zhou, Dawei He and Deshu Chen wrote the written report together.

Data used for this final project can be found in these two websites, https://www.kaggle.com/c/titanic/data, https://archive.ics.uci.edu/ml/datasets/wine+quality.

```
#Titanic Code
library(tidyverse)
train = read_csv("/Users/david/Documents/5241/Titanic/titanic/train.csv")
test = read_csv("/Users/david/Documents/5241/Titanic/titanic/test.csv")
solutions = read_csv("/Users/david/Documents/5241/Titanic/titanic/gender_submission.csv")
colSums(is.na(train))
train = train %>% select(-Cabin)
colSums(is.na(test))
test = test %>% select(-Cabin)

titleAndFirstName = sapply(str_split(train$Name, ','), '[', 2) %>% str_trim()
train$Title = sapply(str_split(titleAndFirstName, '\\.'), '[', 1) %>% str_trim()
train = train %>% select(-Name)
```

```
titleAndFirstName = sapply(str_split(test$Name, ','), '[', 2) %>% str_trim()
test$Title = sapply(str_split(titleAndFirstName, '\\.'), '[', 1) %>% str_trim()
test = test %>% select(-Name)

medianAgeByTitle = train %>%
  group_by(Title) %>%
  summarise(medianAge = median(Age, na.rm = TRUE))

train = merge(train, medianAgeByTitle, by = "Title")
train[is.na(train$Age), 'Age'] = train[is.na(train$Age), 'medianAge']
train = train %>% select(-medianAge, -PassengerId, -Ticket)

colSums(is.na(train))
train[is.na(train$Embarked),]
train = train %>% filter(!is.na(train$Embarked))
colSums(is.na(train))

train = train %>% mutate_at(c("Title", "Pclass", "Sex", "Embarked", "Survived"), as.factor)

test = merge(test, medianAgeByTitle, by = "Title")
test[is.na(test$Age), 'Age'] = test[is.na(test$Age), 'medianAge']

colSums(is.na(test))
test[is.na(test$Fare),]
test = test %>% filter(!is.na(test$Fare))
colSums(is.na(train))

test = merge(test, solutions)
test = test %>%
  select(-medianAge, -PassengerId, -Ticket) %>%
  mutate_at(c("Title", "Pclass", "Sex", "Embarked", "Survived"), as.factor)



#EDA
ggplot(train,
       aes(x = Age, group = Survived, fill = Survived)) +
  geom_density(alpha = .5) +
  annotate('text', x = 5, y = .02, label = "Master") +
  ggtitle("Age Distribution of Passengers")

ggplot(train,
       aes(x = Age, fill = Pclass)) +
```

```
  geom_density(alpha = .5) +
  annotate('text', x = 5, y = .02, label = "Master") +
  ggtitle("Age Distribution of Passengers")

p_age = ggplot(train) +
  geom_freqpoly(mapping = aes(x = Age, color = Survived), binwidth = 1) +
  guides(fill=FALSE) +
  theme(legend.position = "none")

p_sex = ggplot(train, mapping = aes(x = Sex, fill = Survived)) +
  geom_bar(stat='count', position='fill') +
  labs(x = 'Sex') +
  scale_fill_discrete(name="Surv") +


p_class = ggplot(train, mapping = aes(x = Pclass, fill = Survived, colour = Survived)) +
  geom_bar(stat='count', position='fill') +
  labs(x = 'Pclass') +
  theme(legend.position = "none")

p_emb = ggplot(train, aes(Embarked, fill = Survived)) +
  geom_bar(stat='count', position='fill') +
  labs(x = 'Embarked') +
  theme(legend.position = "none")

p_sib = ggplot(train, aes(SibSp, fill = Survived)) +
  geom_bar(stat='count', position='fill') +
  labs(x = 'SibSp') +
  theme(legend.position = "none")

p_par = ggplot(train, aes(Parch, fill = Survived)) +
  geom_bar(stat='count', position='fill') +
  labs(x = 'Parch') +
  theme(legend.position = "none")

p_fare = ggplot(train) +
  geom_freqpoly(mapping = aes(Fare, color = Survived), binwidth = 0.05) +
  scale_x_log10() +
  theme(legend.position = "none")

library(Rmisc)
layout <- matrix(c(1,1,2,3,3,4,5,6,7),3,3,byrow=TRUE)
multiplot(p_age, p_sex, p_fare, p_class, p_emb, p_sib, p_par, layout=layout)
```

```
train %>%
  select(-Title) %>%
  mutate(Sex = fct_recode(Sex,

                          "0" = "male",

                          "1" = "female")

  ) %>%

  mutate(Sex = as.integer(Sex),

         Pclass = as.integer(Pclass),

         Survived = as.integer(Survived),

         Embarked = as.integer(Embarked)) %>%

  cor(use="complete.obs") %>%

  corrplot(type="lower", diag=FALSE)


for (sex in c('male', 'female')) {
  print(
    ggplot(train %>%
             filter(Sex == sex) %>%
             group_by(Pclass, Survived) %>%
             count(name = 'passenger_count'),
           aes(x = Pclass, y = passenger_count, fill = Survived)) +
      geom_bar(stat = 'identity', position = 'dodge') +
      ggtitle(glue::glue("{sex} Passenger Count per Pclass"))
  )
}


# bagging
library(randomForest)
set.seed(1)
baggingModel = randomForest(Survived ~ ., data = train, mtry = 8, importance = TRUE)
test = rbind(train[1,], test)
test = test[-1,]
yhatBagging = predict(baggingModel, test)
# bagging accuracy
```

```
mean(test$Survived == yhatBagging)
#bagging confusion matrix
library(caret)
confusionMatrix(yhatBagging, test$Survived)

plot(baggingModel$importance)

# random forest
rfModel = randomForest(Survived ~ ., data = train, importance = TRUE)
yhatRF = predict(rfModel, test)
#random forest accuracy
mean(test$Survived == yhatRF)
#random forest confusion matrix
confusionMatrix(yhatRF, test$Survived)
# rf feature importance
imp = importance(rfModel, type = 1)
featureImportance <- data.frame(Feature=row.names(imp), Importance=imp[,1])
p = ggplot(featureImportance, aes(x=reorder(Feature, Importance), y=Importance)) +
  geom_bar(stat="identity", fill="#53cfff") +
  coord_flip() +
  theme_light(base_size=20) +
  xlab("") +
  ylab("Importance") +
  ggtitle("Random Forest Feature Importance\n") +
  theme(plot.title=element_text(size=18))


#logistic regression
fullLogModel = glm(Survived ~ ., train, family = binomial)
summary(fullLogModel)
library(MASS)
logModel = stepAIC(fullLogModel, direction = "both", trace = TRUE)
library(car)
vif(logModel)
yhatLog = ifelse(predict(logModel, test, type = "response") > 0.5, 1, 0)
#logistic accuracy
mean(test$Survived == yhatLog)
#logistic confusion matrix
confusionMatrix(as.factor(yhatLog), test$Survived)

#LDA
library(MASS)
temp = train %>% preProcess(method = c("center", "scale"))
normalTrain = temp %>% predict(train)
```

```
temp = test %>% preProcess(method = c("center", "scale"))
normalTest = temp %>% predict(test)
ldaModel = lda(Survived ~ ., normalTrain)
yhatLDA = predict(ldaModel, normalTest)$class
#LDA accuracy
mean(yhatLDA == test$Survived)
#LDA confusion matrix
confusionMatrix(yhatLDA, test$Survived)

library(ggplot2)
lda.data <- cbind(normalTrain, predict(ldaModel)$x)
ggplot(lda.data, aes(LD1)) +
  geom_point(aes(color = Survived))
library(klaR)

#QDA
#qdaModel = qda(Survived ~ ., train)

#KNN
#using cv to select K
library(caret)
trainScaled = train %>%
  select(c('Age', 'SibSp', 'Parch', 'Fare')) %>%
  scale() %>%
  as_tibble() %>%
  cbind(Survived = train$Survived)
trControl = trainControl(method = "cv", number = 10)
cvKNN = train(Survived ~ .,
      method     = "knn",
      tuneGrid   = expand.grid(k = 1:100),
      trControl  = trControl,
      metric     = "Accuracy",
      data       = trainScaled)

plot(cvKNN)
abline(v = cvKNN$bestTune)
#building knn model
library(class)
testScaled = test %>%
  select(c('Age', 'SibSp', 'Parch', 'Fare')) %>%
  scale() %>%
  as_tibble() %>%
  cbind(test$Survived)
yhatKNN = knn(trainScaled, testScaled, trainScaled$Survived, k = cvKNN$bestTune)
```

```
#knn accuracy
mean(yhatKNN == test$Survived)
#knn confusionMatrix
confusionMatrix(yhatKNN, test$Survived)

plot.df = data.frame(testScaled, predicted = yhatKNN)

#svm
#using cv to choose tuning parameters
#linear
svmLinearModel=tune.svm(Survived~.,data=train,kernel="linear",cost=c(0.01,0.1,0.2,0.5,0.7,1,2,3,5,10,1
5,20,50))
yhatSVM.linear = predict(svmLinearModel$best.model, test)
#linear svm accuracy
mean(yhatSVM.linear == test$Survived)
#linear svm confusion matrix
confusionMatrix(yhatSVM.linear, test$Survived)
#radial
svmRadialModel = tune.svm(Survived~.,data=train,kernel="radial",gamma=seq(0.01,5))
yhatSVM.radial = predict(svmRadialModel$best.model, test)
#radial svm accuracy
mean(yhatSVM.radial == test$Survived)
#radial svm confusion matrix
confusionMatrix(yhatSVM.radial, test$Survived)



#Wine-Quality Code

### Exploratory Data Analysis

{r,message=FALSE,warning=FALSE}
#Libraries needed
library(ggplot2)
library(rpart)
library(tidyverse)
# library(GGally)
library(dplyr)
library(randomForest)
library(boot)
#Load in our dataset
#From the data, there are 11 features that will impact the wine quality
wine<-read_csv("./winequality-red.csv")
```

```
summary(wine)


# Let's do some EDA before we make any predictions
#Quality Distribution
#From the plot we can see that the majority of the quality is around 5-6. Around 82%

wine%>%ggplot(aes(factor(quality)))+geom_bar()+ggtitle("Quality Distribution")+theme_light()
prop.table(table(wine$quality))


#Now let's check how those 11 features impacted the quality
#Fixed Acidity/Quality

wine%>%ggplot(aes(`fixed acidity`))+geom_histogram()+ggtitle("Fixed Acidity
Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), `fixed acidity`, group=quality))+geom_boxplot()+ggtitle("Fixed
Acidity & Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(`fixed acidity`))


# Volatile Acidity/Quality

wine%>%ggplot(aes(`volatile acidity`))+geom_histogram()+ggtitle("Volatile Acidity
Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), `volatile acidity`, group=quality))+geom_boxplot()+ggtitle("Volatile
Acidity by Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(`volatile acidity`))


# Citric Acid/Quality

wine%>%ggplot(aes(`citric acid`))+geom_histogram()+ggtitle("Citric Acid Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), `citric acid`, group=quality))+geom_boxplot()+ggtitle("Citric Acid
& Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(`citric acid`))
```

Residual Sugar/Quality

```
wine%>%ggplot(aes(`residual sugar`))+geom_histogram()+ggtitle("Residual Sugar Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), `residual sugar`, group=quality))+geom_boxplot()+ggtitle("Residual Sugar & Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(`residual sugar`))
```

Chrolides/Quality

```
wine%>%ggplot(aes(`chlorides`))+geom_histogram()+ggtitle("Chlorides Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), `chlorides`, group=quality))+geom_boxplot()+ggtitle("Chrolides & Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(chlorides))
```

Free SO2/Quality

```
wine%>%ggplot(aes(`free sulfur dioxide`))+geom_histogram()+ggtitle("Fixed Acidity Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), `free sulfur dioxide`, group=quality))+geom_boxplot()+ggtitle("Free SO2 & Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(`free sulfur dioxide`))
```

Total SO2/Quality

```
wine%>%ggplot(aes(`total sulfur dioxide`))+geom_histogram()+ggtitle("Total sulfur dioxide Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), `total sulfur dioxide`,
group=quality))+geom_boxplot()+ggtitle("Total SO2 & Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(`total sulfur dioxide`))
```

Density/Quality

```
wine%>%ggplot(aes(density))+geom_histogram()+ggtitle("Density Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), density, group=quality))+geom_boxplot()+ggtitle("Density & Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(density))
```

```
# pH/Quality

wine%>%ggplot(aes(pH))+geom_histogram()+ggtitle("pH Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), pH, group=quality))+geom_boxplot()+ggtitle("pH &
Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(pH))



# Sulphates/Quality

wine%>%ggplot(aes(sulphates))+geom_histogram()+ggtitle("Sulphates Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), sulphates, group=quality))+geom_boxplot()+ggtitle("Sulphates &
Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(sulphates))


# Alcohol/Quality

wine%>%ggplot(aes(alcohol))+geom_histogram()+ggtitle("Alcohol Distribution")+theme_light()
wine%>%ggplot(aes(factor(quality), alcohol, group=quality))+geom_boxplot()+ggtitle("Alcohol &
Quality")+theme_light()
wine%>%group_by(quality)%>%summarise(median(alcohol))



# From the EDA it looks like these features will have an impact on the quality. T
# They are : fixed acidity, volatile acidity, citric acid, chlorides,free SO2, total SO2, density, pH,
sulphates & alcohol.
# While for residuals it seems like no impact. From the graph, the median for residual for wine quality 3-8
it is similar.


library(corrplot)
corrplot(cor(wine), method = 'color', order = "AOE", type="upper", tl.pos = 'l')
corrplot(cor(wine), add=TRUE, type="lower", method="number",order="AOE",col =
'black',diag=FALSE,tl.pos="n", cl.pos="n")


# There seems not much relation between quality and other variables.



#Data split
```

```
library(ISLR)
set.seed(1)
train = sample(1599, 1066)


#Linear Regression

# cv.error.10=rep(0,10)
# for (i in 1:10){
#   glm.fit=glm(quality ~., data = wine)
#   cv.error.10[i]=cv.glm(wine,glm.fit,K=10)$delta[1]
# }
# cv.error.10
m.lm = glm(quality ~., data = wine, subset = train)
p.lm = predict(m.lm, wine)
paste('Training error:', mean((wine$quality - p.lm)[train]^2))
paste('Test error:', mean((wine$quality - p.lm)[-train]^2))
# mean(abs(wine$quality - as.integer(p.lm+0.5))^2)


summary(m.lm)


#Stepwise Regression

null = lm(quality~1, wine, subset = train)
full = lm(quality~., wine, subset = train)
step(null, scope=list(lower=null, upper=full), direction="forward")



m.forward = lm(quality ~ alcohol + `volatile acidity` + sulphates +
    `total sulfur dioxide` + chlorides + pH + `free sulfur dioxide`, data = wine, subset = train)
p.forward = predict(m.forward, wine)
paste('Training error:', mean((wine$quality - p.forward)[train]^2))
paste('Test error:', mean((wine$quality - p.forward)[-train]^2))
# mean(abs(wine$quality - as.integer(p.forward+0.5))^2)


Not better than the full model.

#Ridge Regression
```

```
library(glmnet)
x = as.matrix(wine[, 1:11])
y = as.matrix(wine[, 12])
set.seed (1)
cv.out=cv.glmnet(x[train ,],y[train],alpha=0)
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam


m.ridge <- glmnet(x[train,],y[train],alpha=0)
p.ridge <- predict(m.ridge,x,s = bestlam)
paste('Training error:', mean((wine$quality - p.ridge)[train]^2))
paste('Test error:', mean((wine$quality - p.ridge)[-train]^2))
# mean(abs(wine$quality - as.integer(p.ridge+0.5)))
```

Not much improvement.

```
#LASSO regression

x = as.matrix(wine[, 1:11])
y = as.matrix(wine[, 12])
set.seed(1)
cv.out=cv.glmnet(x[train,],y[train],alpha=1)
plot(cv.out)
bestlam=cv.out$lambda.min
bestlam


m.lasso <- glmnet(x,y,alpha=1)
p.lasso <- predict(m.lasso,x,s = bestlam)
paste('Training error:', mean((wine$quality - p.lasso)[train]^2))
paste('Test error:', mean((wine$quality - p.lasso)[-train]^2))



out=glmnet(x,y,alpha=1,lambda=bestlam)
lasso.coef=predict(out,type="coefficients",s=bestlam)[1:12,]
lasso.coef
```

```
#Elastic Net Regression

library(glmnetUtils)
library(data.table)
# alpha is best when it is 0.995
x = as.matrix(wine[, 1:11])
y = as.matrix(wine[, 12])
set.seed(1)
cva.out=cva.glmnet(x[train,],y[train])
plot(cva.out)
number.of.alphas.tested <- length(cva.out$alpha)

cv.glmnet.dt <- data.table()

for (i in 1:number.of.alphas.tested){
  glmnet.model <- cva.out$modlist[[i]]
  min.mse <-  min(glmnet.model$cvm)
  min.lambda <- glmnet.model$lambda.min
  alpha.value <- cva.out$alpha[i]
  new.cv.glmnet.dt <- data.table(alpha=alpha.value,min_mse=min.mse,min_lambda=min.lambda)
  cv.glmnet.dt <- rbind(cv.glmnet.dt,new.cv.glmnet.dt)
}

best.params <- cv.glmnet.dt[which.min(cv.glmnet.dt$min_mse)]

# bestlam=cv.out$lambda.min
# bestlam
# m.en <- glmnet(x,y,alpha=0.5)
# p.en <- predict(m.en,x,s = bestlam)
# paste('Training error:', mean((wine$quality - p.en)[train]^2))
# paste('Test error:', mean((wine$quality - p.en)[-train]^2))
cv.glmnet.dt


#PCR

library(pls)
set.seed(1)
m.pcr = pcr(quality ~., data = wine, scale = T, validation = 'CV',subset = train)
p.pcr = predict(m.pcr, wine)
paste('Training error:', mean((wine$quality - p.pcr)[train]^2))
paste('Test error:', mean((wine$quality - p.pcr)[-train]^2))
```

```
#PLSR

library(pls)
set.seed(1)
m.pls = plsr(quality ~., data = wine, scale = T, validation = 'CV', subset = train)
p.pls = predict(m.pls, wine)
paste('Training error:', mean((wine$quality - p.pls)[train]^2))
paste('Test error:', mean((wine$quality - p.pls)[-train]^2))


#Regression Tree


library(rpart)
m.rpart <- rpart(quality ~ ., data = wine, subset = train)
m.rpart



library(rpart.plot)
rpart.plot(m.rpart, digits = 3)



p.rpart <- predict(m.rpart, wine)
paste('Training error:', mean((wine$quality - p.rpart)[train]^2))
paste('Test error:', mean((wine$quality - p.rpart)[-train]^2))


Seems to have overfitted.

#PCA

pr.out = prcomp(x, scale=TRUE)
biplot(pr.out, scale=0)


Indeed, fixed acidity and pH matter a lot.

#K-means clustering

km.out = kmeans(pr.out$x,2)
plot(pr.out$x, col = (km.out$cluster))
km.out
```

```
km.out = kmeans(pr.out$x,3)
plot(pr.out$x, col = (km.out$cluster))
km.out
```