Modeling to Predict Wine Quality

Final Project

YoungRi Lee & Jihyun Lee

Last compiled on 2020-11-15

1 Introduction

1.1 Data

This wine quality dataset is from [1]. The dataset includes *vinho verde*, a unique product from the Minho (northwest) region of Portugal. The data were collected from May/2004 to February/2007 using only protected designation of origin samples that were tested at the official certification entity (CVRVV). The CVRVV is an inter-professional organization with the goal of improving the quality and marketing of vinho verde. The data were recorded by a computerized system (iLab), which automatically manages the process of wine sample testing from producer requests to laboratory and sensory analysis.

The outcome variable is `wine quality`, which was measured by a minimum score of three sensory assessors using blind tastes in a scale that ranges from 0 (very bad) to 10 (excellent). There are 11 attributes of the wine based on physicochemical tests: fixed acidity (g(tartaric acid)/dm^3), volatile acidity (g(acetic acid)/dm^3), citric acid (g/dm^3), residual sugar (g/dm^3), chlorides (g(sodium chloride)/dm^3), free sulfur dioxide (mg/dm^3), total sulfur dioxide (mg/dm^3), density (g/dm^3), pH, sulphates (g(potassium sulphate)/dm^3), and alcohol (vol.%). Originally, two datasets were created separately, one for red wine ($n = 1599$) and another for white wine ($n = 4988$). In this report, we use a merged dataset and create a dummy variable to indicate the wine type, `red`. Thus, in total, the dataset includes 11 numerical attributes (covariates), one dummy variable, and one numerical outcome. There is no missing value in this dataset.

Table 1 shows the descriptive statistics of 11 attributes by wine type. [ADD SOME DESCRIPTION]

Figure 1 shows the distribution of wine quality by wine type. Generally, it shows a normal shape distribution and centered around the middle point of the scale. Red wine has fewer observations than

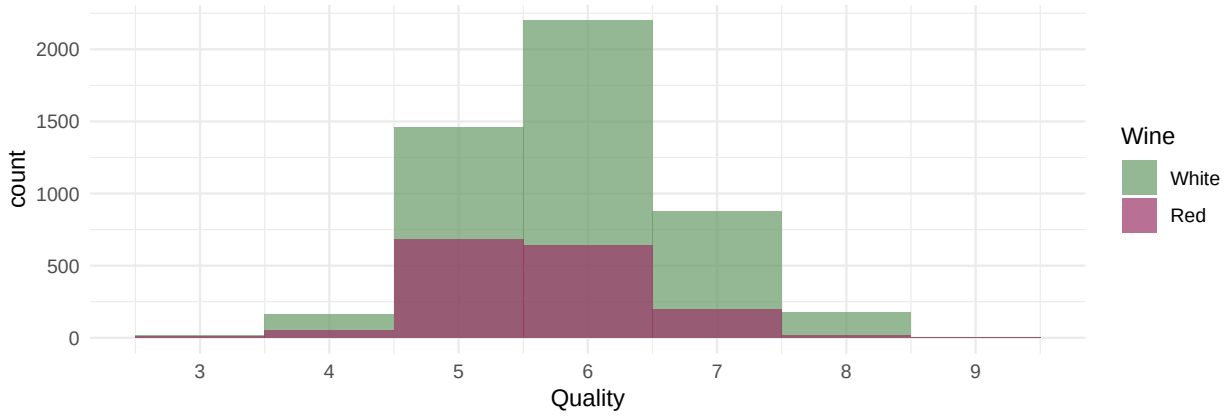
Table 1: Descriptive statistics of 11 attributes

	White			Red		
	Min	Max	Mean	Min	Max	Mean
Fixed acidity	3.8	14.2	6.855	4.6	15.9	8.320
Volatile acidity	0.08	1.10	0.278	0.12	1.58	0.528
Citric acid	0	1.66	0.334	0	1.00	0.271
Residual sugar	0.6	65.8	6.391	0.9	15.5	2.539
Chlorides	0.009	0.346	0.046	0.012	0.611	0.087
Free sulfur dioxide	2	289	35.308	1	72	15.875
Total sulfur dioxide	9	440	138.361	6	289	46.468
Density	0.987	1.039	0.994	0.990	1.004	0.997
pH	2.72	3.82	3.188	2.74	4.01	3.311
Sulphates	0.22	1.08	0.490	0.33	2.00	0.658
Alcohol	8.0	14.2	10.514	8.4	14.9	10.423

Note:
The scale of each attribute is given in the text.

white wine.

Figure 1. Histrogram of wine quality by wine type



1.2 Research Questions

Using this dataset, we would like to (1) build the best model to predict wine quality, (2) examine the most influential set of attributes to predict wine quality.

To achieve this goal, we implemented two classes of techniques: shrinkage approaches (i.e., Ridge and Lasso regressions) and tree-based approaches (i.e., regression tree). This report includes the comparisons of two approaches, model selection procedures (e.g., cross-validation), and proposal of the best model.

[Possible Implications]

- Wine producers will use this information to produce the better quality of wine considering the se-

lected attributes.

- Wine consumers will be able to choose a good quality of wine without tasting (e.g., via online shopping) if the physicochemical information of wine is available.

2 Methods

2.1 Shrinkage approaches

The shrinkage method uses all possible covariates while shrinking the covariates' coefficient that does not associate with the outcome as strong as other predictors. The advantage of this method is to reduce the variance by reducing the relatively unimportant attributes. Also, important covariates in the model can be emphasized. However, it leads to the bias of the model by reducing the variance. There are two types of shrinkage methods, ridge regression, and lasso regression.

2.1.1 Ridge regression

The purpose of ridge regression is to minimize the residual sum of squares (RSS) of the model as well as the shrinkage penalty:

$$RSS + \lambda \sum_{j=1}^{p} \beta_j^2,$$

where $RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$ and the second term indicates the shrinkage penalty. As the coefficients of the attributes become close to zero, the model fit of ridge regression will be desirable. The value of λ controls the shrinkage penalty. As the value of λ increases, the more coefficients will be close to zero. In other words, the ridge regression uses all covariates in the data since the coefficient becomes not equal to zero. Generally, the value of λ is selected using cross-validation to find the optimal value. However, ridge regression has a drawback in using all variables in the data and cannot select or subset the important variable.

2.1.2 Lasso regression

Lasso regression is similar to Ridge regression as it shrinks the coefficients toward zero but has a more stringent shrinkage penalty:

$$RSS + \lambda \sum_{j=1}^{p} |\beta_j^2|.$$

The shrinkage penalty in the second term forces the coefficients to be equal to zero. By pushing the coefficient to be zero, lasso regression can identify the variables that substantially impact the outcome. When there are a small number of variables with strong effect, lasso regression has a benefit over Ridge regression by selecting only those variables. Thus, the model using lasso regression might be more parsimonious than that using ridge regression, making the model results more interpretable. Like ridge regression, the value of λ will be determined by the cross-validation.

2.2 Tree-based approaches

The tree-based approaches can be classified as regression trees and classification trees. We will focus on regression trees because the outcome variable is continuous in our data. Tree-based approaches segment the predictor space into several simple regions. In a tree, each rule that split the segment will be summarized, called decision-tree methods. After splitting the predictor space into a number of regions (a leaf node; $R_1, R_2, ...R_J$) using the selected predictors X_j, the average of the outcome within each region ($\hat{y}_{R_j}$) will be calculated. The model fit of the regression tree aims to minimize the RSS:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

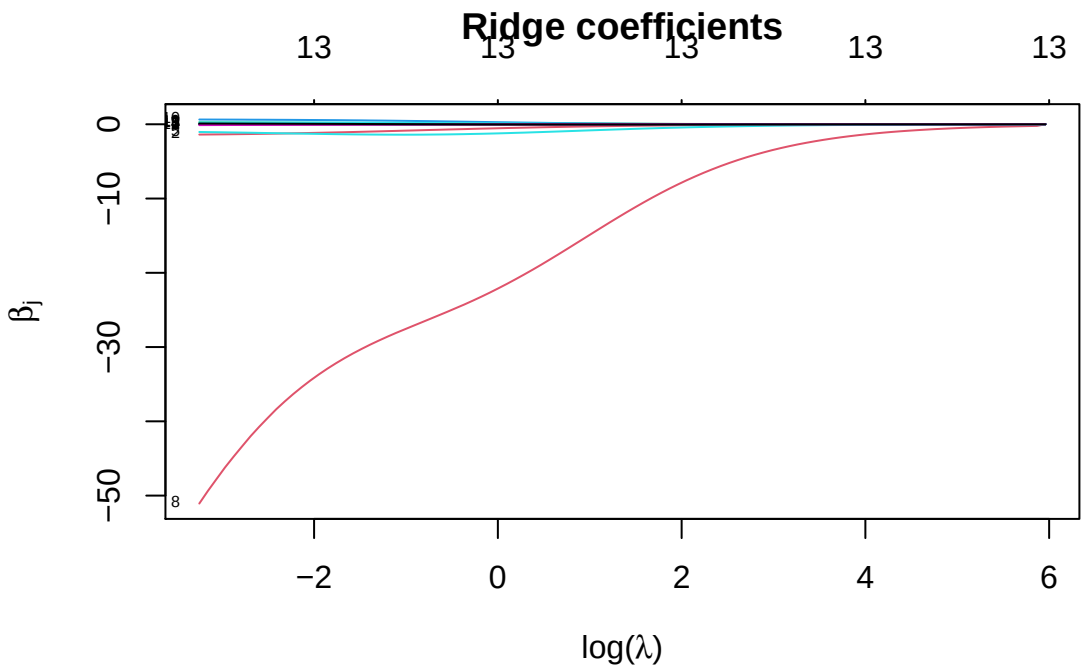
Thus, the regression trees split the predictor space and create the two new branches only if the new split decreases RSS. Since the regression trees use this top-down approach to select the model, the final model might not be the true optimal model. However, the regression trees have the advantage of taking into account the nonlinearity or interaction of covariates. Also, the regression tree is useful in interpreting the results.

3 Results

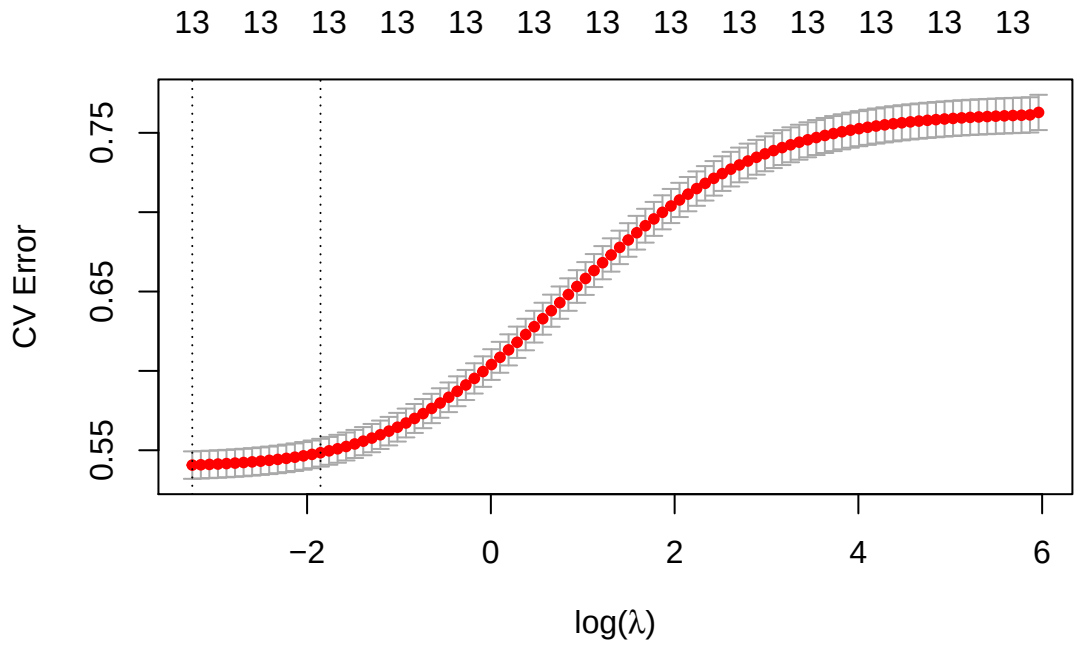
3.1 Ridge regression

```
# data
X <- model.matrix(quality ~ . -1, data = data)
y <- data$quality
p <- ncol(X)


# fit ridge
```

```r
fit_ridge <- glmnet(X, y, alpha = 0)
plot(fit_ridge, xvar = "lambda", label = TRUE, xlab = bquote('log('*lambda*')'), ylab = bquote(beta[j]))
```

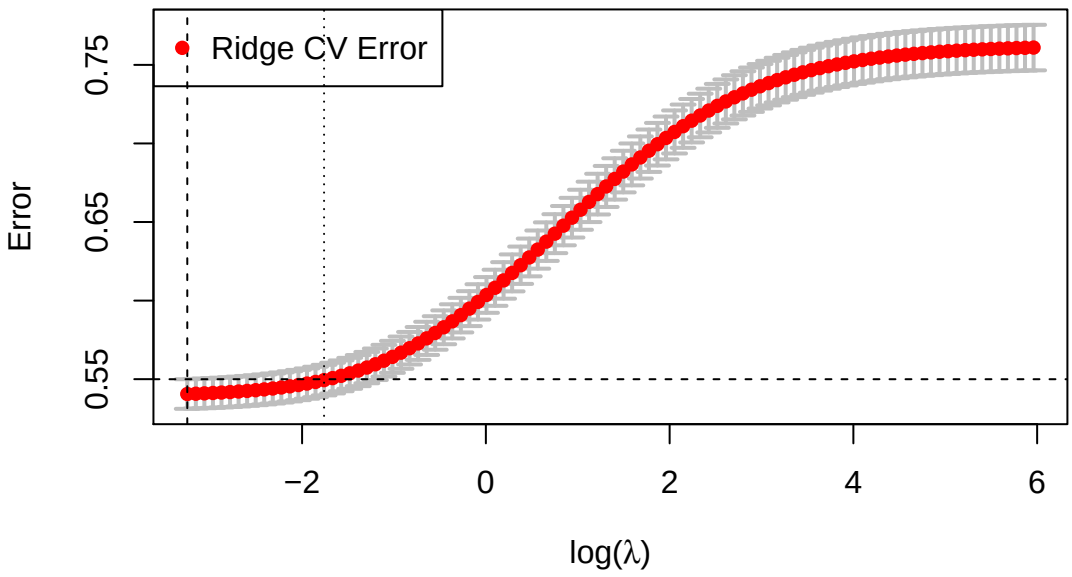



```r
lambda_grid <- fit_ridge$lambda # get lambda_grid for later


# cv
cv_ridge <- cv.glmnet(X, y, alpha = 0)
plot(cv_ridge, xlab = TeX("$\\log (\\lambda)$"), ylab = 'CV Error')
```




```r
# hand-written cv

# cv function
cv_penalized_reg <- function(X, y, lambda_grid, K = floor(sqrt(nrow(X)))){


  n <- nrow(X)
  p <- ncol(X)
  folds <- cut(sample(1:n), breaks = K, labels = FALSE)


  train_err <- cv_err <- array(NA, dim = c(K, length(lambda_grid)))


  for(k in 1:K){
    idx_tr <- which(folds != k)
    idx_ts <- which(folds == k)


    fit_glm <- glmnet(X[idx_tr,], y[idx_tr], alpha = 0, lambda = lambda_grid)


    if (length(idx_ts) > 1){
```

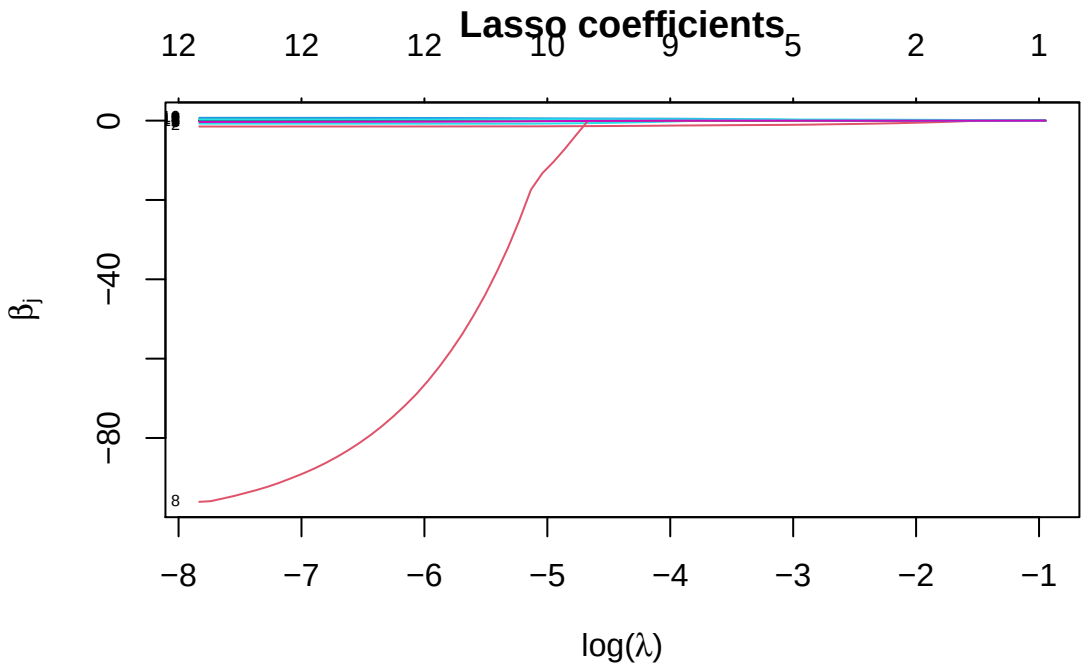
```
      cv_err[k,] <- as.numeric(colMeans((predict(fit_glm, X[idx_ts,],
                                lambda = lambda_grid) - y[idx_ts])^2))
    }
    else{
      cv_err[k,] <- as.numeric(colMeans((predict(fit_glm,
                    matrix(X[idx_ts,],1,p),
                    lambda = lambda_grid) - y[idx_ts])^2))
    }
  }
  cv_MSE <- colMeans(cv_err)
  sd_cv_MSE <- apply(cv_err, 2, sd)/sqrt(K)

  return(list('cv_MSE' = cv_MSE, 'sd_cv_MSE' = sd_cv_MSE, 'cv_err' = cv_err))
}

# find the best model
fit_err <- cv_penalized_reg(X, y, lambda_grid, K = 10)

# plot and see which one is the optimal & conservative one.
plotCI(x = log(lambda_grid), y = fit_err$cv_MSE, uiw = fit_err$sd_cv_MSE,
       col = 'red', scol = 'gray', pch = 16, lwd = 2,
       xlab = TeX("$\\log (\\lambda)$"), ylab = 'Error')
abline(v = log(lambda_grid)[which.min(fit_err$cv_MSE)], lwd = 1, col = 1, lty = 2)
abline(h = min(fit_err$cv_MSE) + fit_err$sd_cv_MSE[which.min(fit_err$cv_MSE)],
       lwd = 1, lty = 2)
idx <- which( min(fit_err$cv_MSE) + fit_err$sd_cv_MSE[which.min(fit_err$cv_MSE)] >= fit_err$cv_MSE)
idx <- idx[1]
abline(v = log(lambda_grid[idx]), lwd = 1, col = 1, lty = 3)
legend('topleft', legend = 'Ridge CV Error', pch = 16, col = 'red')
```



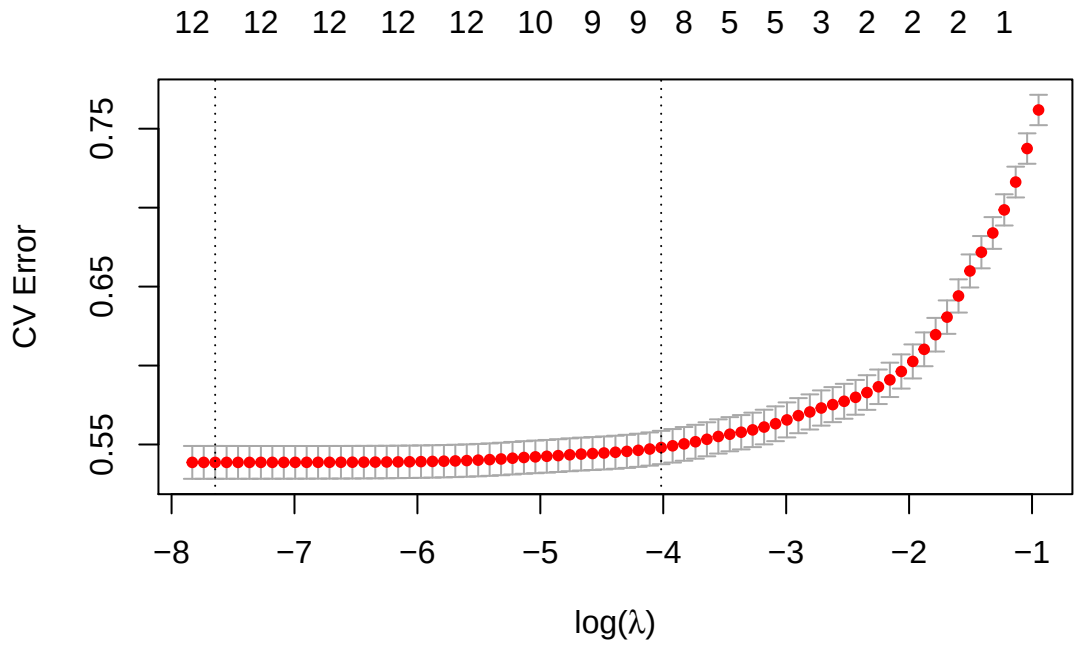

```
cv10_ridge <- fit_err$cv_err[,which.min(fit_err$cv_MSE)] # optimal
compare <- data.frame(cv10_ridge) %>% mutate(model = "Ridge regression") %>%
  rename(value = cv10_ridge) %>%
  select(model, value)
```

3.2 Lasso regression

```
# fit lasso
fit_lasso <- glmnet(X, y)
plot(fit_lasso, xvar = "lambda", label = TRUE,
     xlab = bquote('log('*lambda*')'), ylab = bquote(beta[j]),
     main = 'Lasso coefficients')
```

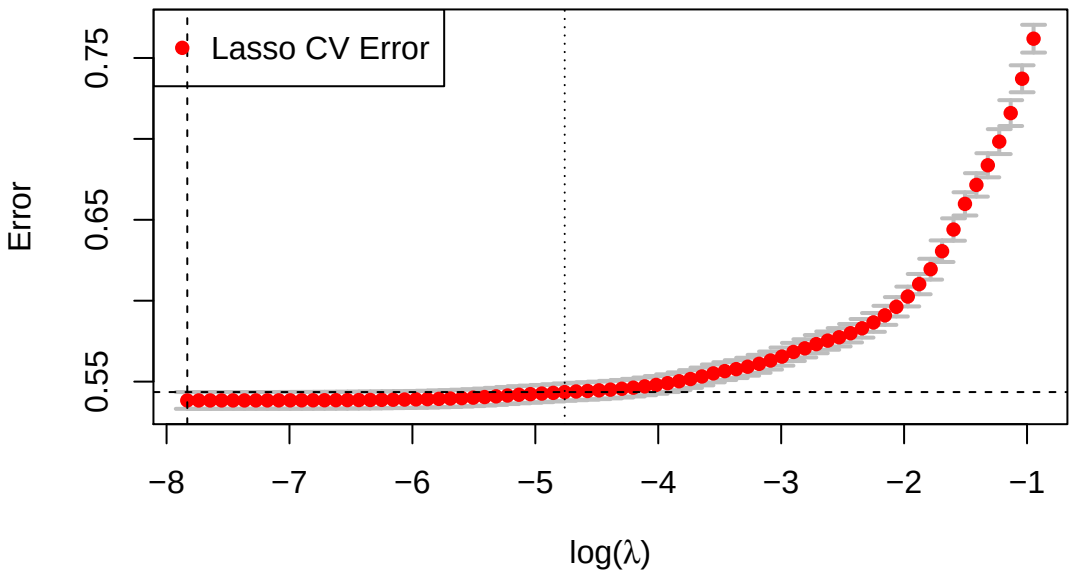



```
lambda_grid <- fit_lasso$lambda # get lambda_grid for later


# cv
cv_lasso <- cv.glmnet(X, y)
plot(cv_lasso, xlab = TeX("$\\log (\\lambda)$"), ylab = 'CV Error')
```




```r
# hand-written cv

# cv function
cv_penalized_reg <- function(X, y, lambda_grid, K = floor(sqrt(nrow(X)))){


  n <- nrow(X)
  p <- ncol(X)
  folds <- cut(sample(1:n), breaks = K, labels = FALSE)


  train_err <- cv_err <- array(NA, dim = c(K, length(lambda_grid)))


  for(k in 1:K){
    idx_tr <- which(folds != k)
    idx_ts <- which(folds == k)


    fit_glm <- glmnet(X[idx_tr,], y[idx_tr], lambda = lambda_grid)


    if (length(idx_ts) > 1){
```

```
      cv_err[k,] <- as.numeric(colMeans((predict(fit_glm, X[idx_ts,],
                                lambda = lambda_grid) - y[idx_ts])^2))
    }
    else{
      cv_err[k,] <- as.numeric(colMeans((predict(fit_glm,
                    matrix(X[idx_ts,],1,p),
                    lambda = lambda_grid) - y[idx_ts])^2))
    }
  }
  cv_MSE <- colMeans(cv_err)
  sd_cv_MSE <- apply(cv_err, 2, sd)/sqrt(K)

  return(list('cv_MSE' = cv_MSE, 'sd_cv_MSE' = sd_cv_MSE, 'cv_err' = cv_err))
}

# find the best model
fit_err <- cv_penalized_reg(X, y, lambda_grid, K = 10)

# plot and see which one is the optimal & conservative one.
plotCI(x = log(lambda_grid), y = fit_err$cv_MSE, uiw = fit_err$sd_cv_MSE,
       col = 'red', scol = 'gray', pch = 16, lwd = 2,
       xlab = TeX("$\\log (\\lambda)$"), ylab = 'Error')
abline(v = log(lambda_grid)[which.min(fit_err$cv_MSE)], lwd = 1, col = 1, lty = 2)
abline(h = min(fit_err$cv_MSE) + fit_err$sd_cv_MSE[which.min(fit_err$cv_MSE)],
       lwd = 1, lty = 2)
idx <- which( min(fit_err$cv_MSE) + fit_err$sd_cv_MSE[which.min(fit_err$cv_MSE)] >= fit_err$cv_MSE)
idx <- idx[1]
abline(v = log(lambda_grid[idx]), lwd = 1, col = 1, lty = 3)
legend('topleft', legend = 'Lasso CV Error', pch = 16, col = 'red')
```



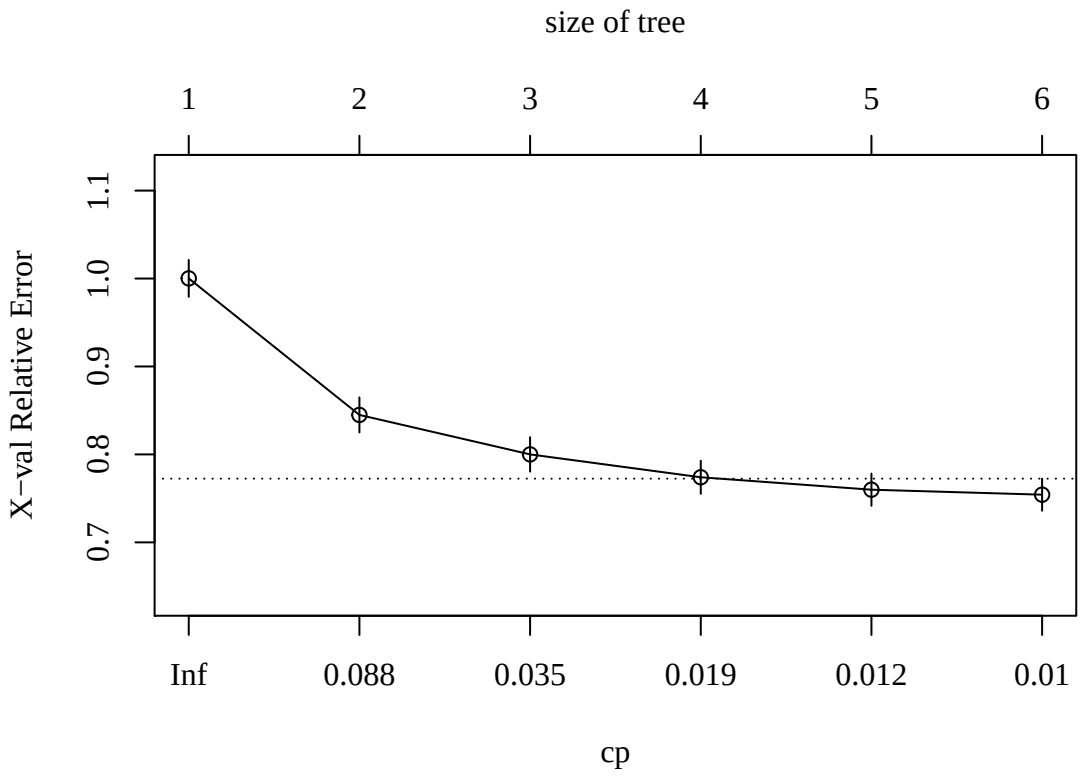

```
cv10_lasso <- fit_err$cv_err[,which.min(fit_err$cv_MSE)] # optimal
compare <- data.frame(cv10_lasso) %>% mutate(model = "Lasso regression") %>%
  rename(value = cv10_lasso) %>%
  select(model, value) %>% bind_rows(compare)

# comparison
compare$model <- factor(compare$model , levels=c("Ridge regression", "Lasso regression"))
ggplot(compare, aes(x = model, y = value, fill = model)) +
  geom_boxplot(width = 0.5, alpha = 0.7) +
  labs(title="Test error by Ridge and Lasso regressions",
       x ="Model", y = "Test Error") +
  scale_fill_manual(values=c("#E41A1C", "#377EB8")) + # "#4DAF4A", "#984EA3"
  theme_minimal() +
  theme(legend.title = element_blank())
```

3.3 Tree-based approaches

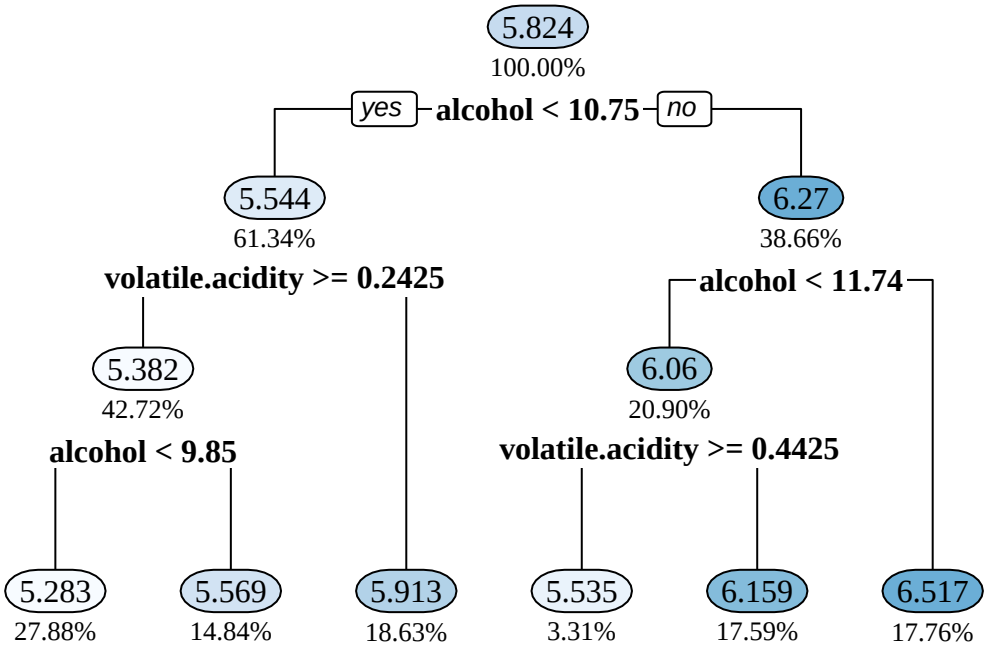
```
# Split the data in training and test
n <- nrow(data)
n_train <- floor(0.8 * n)
n_test <- n - n_train

set.seed(123)
idx_test <- sample(1:n, n_test, replace = F)
data_ts <- data[idx_test,]
data <- data[-idx_test,]

# fit tree
best_tree <- rpart(quality ~ ., method = "anova", data = data)
cat('Size of optimal tree:', length(unique(best_tree$where)), '\n')
```

```
## Size of optimal tree: 6
```

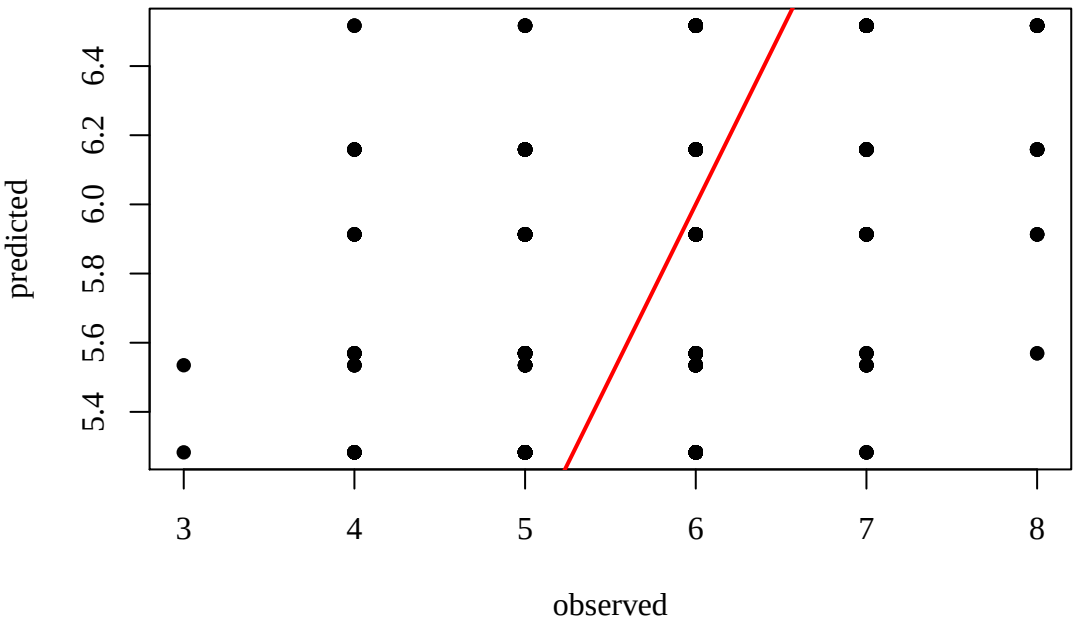
```r
# Plot the cv error curve for the tree
par(mar=c(4,4,4,2), family = 'serif')
plotcp(best_tree)
```




```r
# Show the optimal tree
rpart.plot(best_tree, clip.right.labs = FALSE, under = TRUE, digits = 4)
```




```
# Prediction on the test set

pred_y <- as.numeric(predict(best_tree, newdata = data_ts))

err_all <- (pred_y - data_ts$quality)^2


plot(data_ts$quality, pred_y, pch = 16, xlab = 'observed', ylab = 'predicted')

abline(0, 1, lwd = 2, col = 'red')
```



4 Discussion

4.1 Limitations

- Only uses the wine data from *vinho verde* region of Portugal. If a test dataset outside of this region is available, we would be able to test to generalize the results to a broader range of wine.

References

[1] P. Cortez et al. "Modeling wine preferences by data mining from physicochemical properties". In: *Decision Support Systems* 47.4 (2009), pp. 547–553.