Homework 3

Gaurang Kakade

Table of contents

Link to the Github repository

❗ Due: Thu, Mar 2, 2023 @ 11:59pm

Please read the instructions carefully before submitting your assignment.

1. This assignment requires you to only upload a `PDF` file on Canvas
2. Don't collapse any code cells before submitting.
3. Remember to make sure all your code output is rendered properly before uploading your submission.

Please add your name to the author information in the frontmatter before submitting your assignment

For this assignment, we will be using the Wine Quality dataset from the UCI Machine Learning Repository. The dataset consists of red and white *vinho verde* wine samples, from the north of Portugal. The goal is to model wine quality based on physicochemical tests

We will be using the following libraries:

```r
library(readr)
library(tidyr)
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(purrr)
library(car)
```

```
Loading required package: carData

Attaching package: 'car'

The following object is masked from 'package:purrr':

    some

The following object is masked from 'package:dplyr':

    recode
```

```r
library(glmnet)
```

```
Loading required package: Matrix

Attaching package: 'Matrix'
```

```
The following objects are masked from 'package:tidyr':

    expand, pack, unpack

Loaded glmnet 4.1-6
```

Question 1

50 points

Regression with categorical covariate and t-Test

1.1 (5 points)

Read the wine quality datasets from the specified URLs and store them in data frames `df1` and `df2`.

```r
url1 <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequalit

url2 <- "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequalit


df1 <- read.csv(url1, header = TRUE, sep = ";")
df2 <- read.csv(url2, header = TRUE, sep = ";")
```

1.2 (5 points)

Perform the following tasks to prepare the data frame `df` for analysis:

1. Combine the two data frames into a single data frame `df`, adding a new column called `type` to indicate whether each row corresponds to white or red wine.
2. Rename the columns of `df` to replace spaces with underscores
3. Remove the columns `fixed_acidity` and `free_sulfur_dioxide`
4. Convert the `type` column to a factor
5. Remove rows (if any) with missing values.

```r
# Adding a new column called `type` to indicate each row corresponds to white
# or red
df1$type <- "white"
df2$type <- "red"

# Combine the two data frames into a single data frame "df"
df <- rbind(df1, df2)

# Renaming the columns of df to replace spaces with underscores
colnames(df) <- gsub("\\.", "_", colnames(df))

# Remove the columns `fixed_acidity` and `free_sulphur_dioxide`
df <- df %>%
  select(-fixed_acidity, -free_sulfur_dioxide)

# Converting the `type` column to a factor
df$type <- as.factor(df$type)

# Remove rows with missing values
df <- df %>%
  drop_na()
```

```r
dim(df)
```

```
[1] 6497   11
```

Your output to `R dim(df)` should be

```
[1] 6497   11
```

1.3 (20 points)

Recall from STAT 200, the method to compute the t statistic for the the difference in means (with the equal variance assumption)

1. Using `df` compute the mean of `quality` for red and white wine separately, and then store the difference in means as a variable called `diff_mean`.
2. Compute the pooled sample variance and store the value as a variable called `sp_squared`.

3. Using `sp_squared` and `diff_mean`, compute the t Statistic, and store its value in a variable called `t1`.

```
diff_mean <- df %>%
  group_by(type) %>%
  summarise(mean_quality = mean(quality)) %>%
  summarise(diff_mean = diff(mean_quality))

# Computing the sample size of the two groups
n_red <- sum(df$type == "red")
n_white <- sum(df$type == "white")

# Computing the sample variance of the two groups (red and white wine)
red_var <- var(df[df$type == "red", "quality"])
white_var <- var(df[df$type == "white", "quality"])

# Computing the pooled sample variance between the two groups
sp_squared <- ((n_red - 1) * red_var + (n_white - 1) * white_var) / (n_red + n_white - 2)



t1 <-  diff_mean / sqrt(sp_squared * (1/n_red + 1/n_white))
t1
```

```
  diff_mean
1   9.68565
```

1.4 (10 points)

Equivalently, R has a function called `t.test()` which enables you to perform a two-sample t-Test without having to compute the pooled variance and difference in means.

Perform a two-sample t-test to compare the quality of white and red wines using the `t.test()` function with the setting `var.equal=TRUE`. Store the t-statistic in `t2`.

```
t_test <- t.test(df$quality[df$type == "white"], df$quality[df$type == "red"], var.equal =
t2 <- t_test$statistic
t2
```

```
      t
9.68565
```

1.5 (5 points)

Fit a linear regression model to predict `quality` from `type` using the `lm()` function, and extract the t-statistic for the `type` coefficient from the model summary. Store this t-statistic in `t3`.

```r
fit <- lm(quality ~ type, data = df)
t3 <- summary(fit)$coefficient[2, "t value"]
t3
```

```
[1] 9.68565
```

1.6 (5 points)

Print a vector containing the values of `t1`, `t2`, and `t3`. What can you conclude from this? Why?

```r
c(t1, t2, t3) # Insert your code here
```

```
$diff_mean
[1] 9.68565

$t
[1] 9.68565

[[3]]
[1] 9.68565
```

Having interpreted the results of the t test (where all the test have the same values) t1 = 9.68565, t2 = 9.68565. t3 = 9.68565, we can conclude that there is a substantial difference between the quality of the white wine and red wine. The t-test statistic helps to determine the correlation between the response and the predictor variables where as dthe linear regression helps to determine the linear correlation between the predictor variable and the response variable. Hence, they are the same.

Question 2

25 points

Collinearity

2.1 (5 points)

Fit a linear regression model with all predictors against the response variable `quality`. Use the `broom::tidy()` function to print a summary of the fitted model. What can we conclude from the model summary?

```r
library(broom)
fit <- lm(quality ~., data = df)
summary(fit)
```

```
Call:
lm(formula = quality ~ ., data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.3415 -0.4725 -0.0405  0.4573  3.1140 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)           5.753e+01  9.331e+00   6.166 7.44e-10 ***
volatile_acidity     -1.609e+00  8.057e-02 -19.965  < 2e-16 ***
citric_acid           2.721e-02  7.833e-02   0.347  0.72827    
residual_sugar        4.509e-02  4.158e-03  10.844  < 2e-16 ***
chlorides            -9.639e-01  3.328e-01  -2.897  0.00378 ** 
total_sulfur_dioxide -3.289e-04  2.623e-04  -1.254  0.20995    
density              -5.520e+01  9.320e+00  -5.922 3.34e-09 ***
pH                    1.885e-01  6.613e-02   2.850  0.00438 ** 
sulphates             6.620e-01  7.584e-02   8.730  < 2e-16 ***
alcohol               2.767e-01  1.418e-02  19.514  < 2e-16 ***
typewhite            -3.858e-01  5.493e-02  -7.023 2.39e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.7371 on 6486 degrees of freedom
Multiple R-squared:  0.2887,    Adjusted R-squared:  0.2876
F-statistic: 263.3 on 10 and 6486 DF,  p-value: < 2.2e-16
```

```r
tidy(fit)
```

```
# A tibble: 11 x 5
   term                   estimate std.error statistic  p.value
   <chr>                     <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)            57.5       9.33         6.17  7.44e-10
 2 volatile_acidity       -1.61      0.0806     -20.0   4.07e-86
 3 citric_acid             0.0272    0.0783       0.347 7.28e- 1
 4 residual_sugar          0.0451    0.00416     10.8   3.64e-27
 5 chlorides              -0.964     0.333       -2.90  3.78e- 3
 6 total_sulfur_dioxide   -0.000329  0.000262    -1.25  2.10e- 1
 7 density               -55.2       9.32        -5.92  3.34e- 9
 8 pH                      0.188     0.0661       2.85  4.38e- 3
 9 sulphates               0.662     0.0758       8.73  3.21e-18
10 alcohol                 0.277     0.0142      19.5   1.87e-82
11 typewhite              -0.386     0.0549      -7.02  2.39e-12
```

The output shown above indicates that several predictors have statistically significant associations with quality, including volatile.acidity, citric.acid, residual.sugar, chlorides, total.sulfur.dioxide, density, pH, sulphates, alcohol, and type.The adjusted R-squared value of the model is 0.2876, indicating that the predictors explain about 29% of the variation in quality.

2.2 (10 points)

Fit two **simple** linear regression models using `lm()`: one with only `citric_acid` as the predictor, and another with only `total_sulfur_dioxide` as the predictor. In both models, use `quality` as the response variable. How does your model summary compare to the summary from the previous question?

```r
model_citric <- lm(quality ~ citric_acid, data = df)
summary(model_citric)
```

```
Call:
lm(formula = quality ~ citric_acid, data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.9938 -0.7831  0.1552  0.2426  3.1963 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  5.65461    0.02602 217.343   <2e-16 ***
citric_acid  0.51398    0.07429   6.918    5e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8701 on 6495 degrees of freedom
Multiple R-squared:  0.007316,	Adjusted R-squared:  0.007163 
F-statistic: 47.87 on 1 and 6495 DF,  p-value: 5.002e-12
```

```r
model_sulfur <- lm(quality ~ total_sulfur_dioxide, data = df)
summary(model_sulfur)
```

```
Call:
lm(formula = quality ~ total_sulfur_dioxide, data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-2.8866 -0.7971  0.1658  0.2227  3.1965 

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)    
(Intercept)           5.8923848  0.0246717 238.831  < 2e-16 ***
total_sulfur_dioxide -0.0006394  0.0001915  -3.338 0.000848 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8726 on 6495 degrees of freedom
Multiple R-squared:  0.001713,	Adjusted R-squared:  0.001559 
F-statistic: 11.14 on 1 and 6495 DF,  p-value: 0.000848
```

Comparing the model summaries (model_citric and model_sulfur_dioxide), we observe that the $P-$ value is much lower for both models (5e-12 and 0.000848 respectively) than for the

multiple linear regression model (7.282728e-01 and 2.099538e-0 respectively). However, their individual coefficients are smaller in magnitude (i.e. they have greater statistical significance of the obsevred difference) than those in the multiple linear regression model with all predictors. This suggests that other predictors in the multiple linear regression model may also be important in explaining the variation in quality.

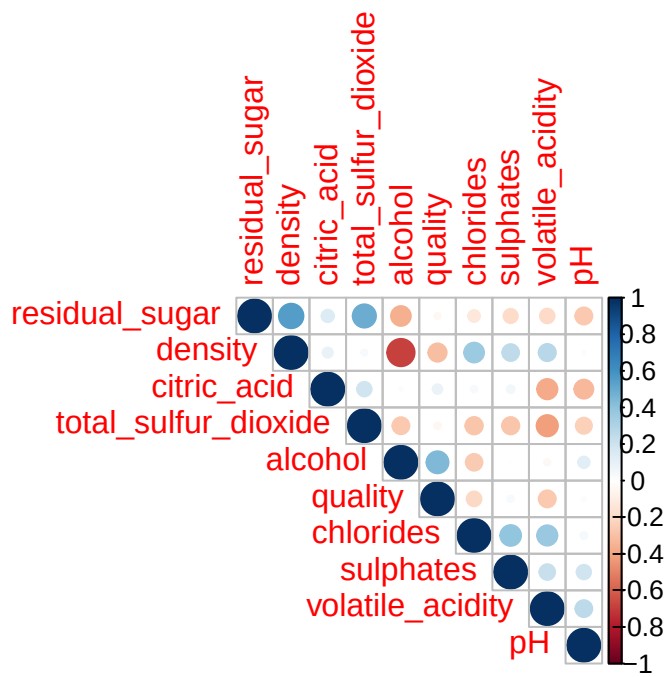
2.3 (5 points)

Visualize the correlation matrix of all numeric columns in `df` using `corrplot()`

```
library(corrplot)
```

```
corrplot 0.92 loaded
```

```
corr_matrix <- df %>%
  keep(is.numeric) %>%
  cor()
corrplot(corr_matrix, type="upper", order="hclust")
```



2.4 (5 points)

Compute the variance inflation factor (VIF) for each predictor in the full model using `vif()` function. What can we conclude from this?

```
library(car)
vif_model <- lm(quality ~ ., df)
vif(vif_model) %>% knitr::kable()
```

	x
volatile_acidity	2.103853
citric_acid	1.549248
residual_sugar	4.680035
chlorides	1.625065
total_sulfur_dioxide	2.628534
density	9.339357
pH	1.352005
sulphates	1.522809
alcohol	3.419849
type	6.694679

This code will output the VIF for each predictor in the full model. A VIF of 1 indicates no correlation with other predictors, while a VIF of greater than 1 indicates some degree of correlation (i.e. all the predictors in the full model have a value greater than 1). From the output of vif(), we can see that most of the predictors [volatile.acidity, citric.acid, residual.sugar, chlorides, total.sulfur.dioxide, pH, sulphates, alcohol]in the full model have relatively low VIF values, indicating low multicollinearity. However, the density predictor has a VIF value of over 9.339, which is quite high. This suggests that density may be highly correlated with other predictors in the model and may be contributing redundant information to the model.

Question 3

40 points

Variable selection

3.1 (5 points)

Run a backward stepwise regression using a `full_model` object as the starting model. Store the final formula in an object called `backward_formula` using the built-in `formula()` function in R

```
full_model <- lm(quality ~ ., df)
backward_formula <- step(full_model, direction = "backward", scope=formula(full_model))
```

```
Start:  AIC=-3953.43
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
    total_sulfur_dioxide + density + pH + sulphates + alcohol +
    type

                       Df Sum of Sq    RSS     AIC
- citric_acid           1     0.066 3523.6 -3955.3
- total_sulfur_dioxide  1     0.854 3524.4 -3953.9
<none>                              3523.5 -3953.4
- pH                    1     4.413 3527.9 -3947.3
- chlorides             1     4.559 3528.1 -3947.0
- density               1    19.054 3542.6 -3920.4
- type                  1    26.794 3550.3 -3906.2
- sulphates             1    41.399 3564.9 -3879.5
- residual_sugar        1    63.881 3587.4 -3838.7
- alcohol               1   206.860 3730.4 -3584.8
- volatile_acidity      1   216.549 3740.0 -3567.9

Step:  AIC=-3955.3
quality ~ volatile_acidity + residual_sugar + chlorides + total_sulfur_dioxide +
    density + pH + sulphates + alcohol + type

                       Df Sum of Sq    RSS     AIC
- total_sulfur_dioxide  1     0.818 3524.4 -3955.8
```

```
<none>                              3523.6 -3955.3
- chlorides             1     4.495 3528.1 -3949.0
- pH                    1     4.536 3528.1 -3948.9
- density               1    20.794 3544.4 -3919.1
- type                  1    26.943 3550.5 -3907.8
- sulphates             1    41.491 3565.1 -3881.2
- residual_sugar        1    67.371 3590.9 -3834.3
- alcohol               1   235.151 3758.7 -3537.6
- volatile_acidity      1   252.565 3776.1 -3507.5

Step:  AIC=-3955.8
quality ~ volatile_acidity + residual_sugar + chlorides + density +
    pH + sulphates + alcohol + type

                    Df Sum of Sq    RSS     AIC
<none>                           3524.4 -3955.8
- pH                 1     4.295 3528.7 -3949.9
- chlorides          1     4.523 3528.9 -3949.5
- density            1    21.540 3545.9 -3918.2
- sulphates          1    40.711 3565.1 -3883.2
- type               1    43.664 3568.0 -3877.8
- residual_sugar     1    66.572 3591.0 -3836.2
- alcohol            1   244.545 3768.9 -3521.9
- volatile_acidity   1   256.695 3781.1 -3501.0
```

3.2 (5 points)

Run a forward stepwise regression using a `null_model` object as the starting model. Store the final formula in an object called `forward_formula` using the built-in `formula()` function in R

```r
null_model <- lm(quality ~ 1, df)
forward_formula <- step(null_model, direction = "forward", scope = formula(full_model))
```

```
Start:  AIC=-1760.04
quality ~ 1

                        Df Sum of Sq    RSS     AIC
+ alcohol                1    977.95 3975.7 -3186.9
+ density                1    463.41 4490.3 -2396.2
```

```
+ volatile_acidity      1    349.71 4604.0 -2233.7
+ chlorides             1    199.47 4754.2 -2025.1
+ type                  1     70.53 4883.2 -1851.2
+ citric_acid           1     36.24 4917.4 -1805.7
+ total_sulfur_dioxide  1      8.48 4945.2 -1769.2
+ sulphates             1      7.34 4946.3 -1767.7
+ residual_sugar        1      6.77 4946.9 -1766.9
+ pH                    1      1.88 4951.8 -1760.5
<none>                              4953.7 -1760.0

Step:  AIC=-3186.88
quality ~ alcohol

                       Df Sum of Sq    RSS     AIC
+ volatile_acidity      1   307.508 3668.2 -3707.9
+ residual_sugar        1    85.662 3890.1 -3326.4
+ type                  1    54.335 3921.4 -3274.3
+ citric_acid           1    40.303 3935.4 -3251.1
+ chlorides             1    39.696 3936.0 -3250.1
+ total_sulfur_dioxide  1    31.346 3944.4 -3236.3
+ sulphates             1     7.859 3967.9 -3197.7
+ pH                    1     5.938 3969.8 -3194.6
<none>                              3975.7 -3186.9
+ density               1     0.005 3975.7 -3184.9

Step:  AIC=-3707.89
quality ~ alcohol + volatile_acidity

                       Df Sum of Sq    RSS     AIC
+ sulphates             1    48.259 3620.0 -3791.9
+ density               1    38.704 3629.5 -3774.8
+ residual_sugar        1    29.751 3638.5 -3758.8
+ type                  1    28.895 3639.3 -3757.3
+ total_sulfur_dioxide  1     5.619 3662.6 -3715.9
+ pH                    1     5.533 3662.7 -3715.7
<none>                              3668.2 -3707.9
+ chlorides             1     0.162 3668.1 -3706.2
+ citric_acid           1     0.099 3668.1 -3706.1

Step:  AIC=-3791.94
quality ~ alcohol + volatile_acidity + sulphates

                       Df Sum of Sq    RSS     AIC
```

```
+ residual_sugar        1    43.989 3576.0 -3869.4
+ density               1    18.661 3601.3 -3823.5
+ type                  1     6.012 3614.0 -3800.7
+ chlorides             1     4.988 3615.0 -3798.9
+ citric_acid           1     2.031 3617.9 -3793.6
+ pH                    1     1.903 3618.1 -3793.4
<none>                               3620.0 -3791.9
+ total_sulfur_dioxide  1     0.817 3619.2 -3791.4

Step:  AIC=-3869.37
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar

                       Df Sum of Sq    RSS     AIC
+ type                  1   20.7581 3555.2 -3905.2
+ total_sulfur_dioxide  1   13.3542 3562.6 -3891.7
+ pH                    1    6.6430 3569.3 -3879.5
+ citric_acid           1    4.3384 3571.6 -3875.3
+ chlorides             1    1.8907 3574.1 -3870.8
<none>                               3576.0 -3869.4
+ density               1    0.0071 3576.0 -3867.4

Step:  AIC=-3905.19
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar +
    type

                       Df Sum of Sq    RSS     AIC
+ density               1   20.4623 3534.8 -3940.7
+ chlorides             1    6.6602 3548.6 -3915.4
+ citric_acid           1    5.2242 3550.0 -3912.7
+ pH                    1    3.9477 3551.3 -3910.4
+ total_sulfur_dioxide  1    1.2539 3554.0 -3905.5
<none>                               3555.2 -3905.2

Step:  AIC=-3940.7
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar +
    type + density

                       Df Sum of Sq    RSS     AIC
+ chlorides             1    6.0826 3528.7 -3949.9
+ pH                    1    5.8541 3528.9 -3949.5
<none>                               3534.8 -3940.7
+ citric_acid           1    0.8471 3533.9 -3940.3
+ total_sulfur_dioxide  1    0.5646 3534.2 -3939.7
```

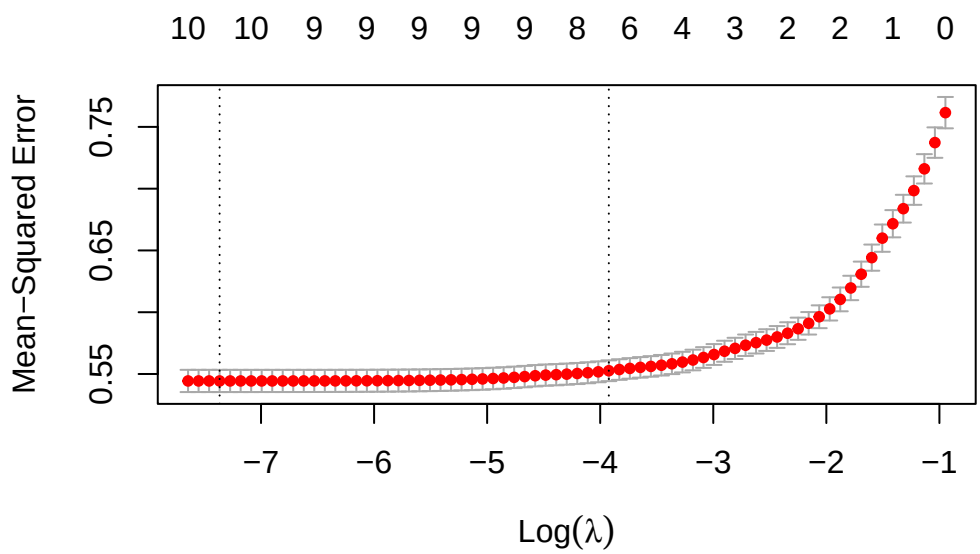
```
Step:  AIC=-3949.89
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar +
    type + density + chlorides

                       Df Sum of Sq    RSS     AIC
+ pH                    1    4.2945 3524.4 -3955.8
<none>                               3528.7 -3949.9
+ total_sulfur_dioxide  1    0.5765 3528.1 -3948.9
+ citric_acid           1    0.2338 3528.4 -3948.3

Step:  AIC=-3955.8
quality ~ alcohol + volatile_acidity + sulphates + residual_sugar +
    type + density + chlorides + pH

                       Df Sum of Sq    RSS     AIC
<none>                               3524.4 -3955.8
+ total_sulfur_dioxide  1   0.81762 3523.6 -3955.3
+ citric_acid           1   0.02919 3524.4 -3953.9
```

3.3 (10 points)

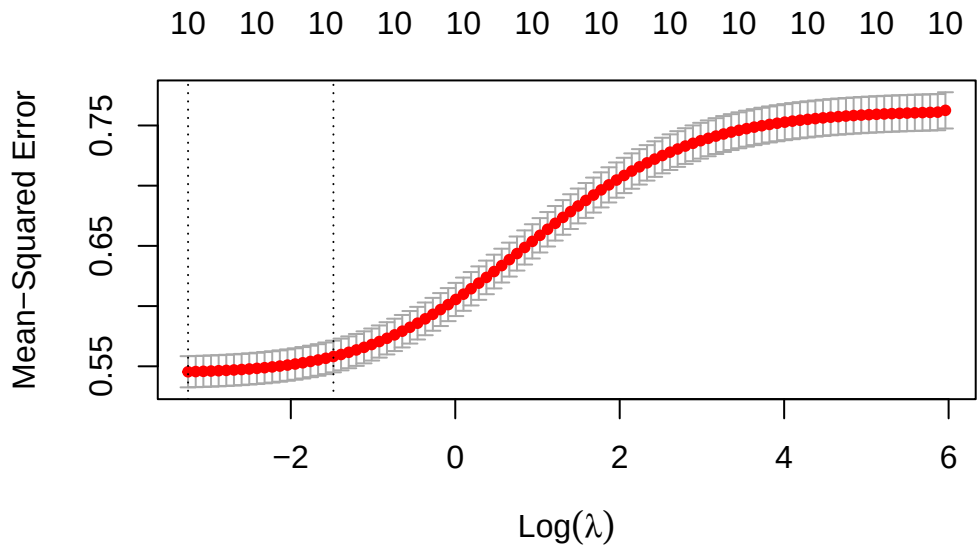
1. Create a `y` vector that contains the response variable (`quality`) from the `df` dataframe.
2. Create a design matrix `X` for the `full_model` object using the `make_model_matrix()` function provided in the Appendix.
3. Then, use the `cv.glmnet()` function to perform LASSO and Ridge regression with `X` and `y`.

```r
# Creating a `y` vector containing the response variable (`quality`)
y <- df$quality

# Creating a design matrix `X` for the `full_model`

make_model_matrix <- function(formula){
  X <- model.matrix(full_model, df)[, -1]
  cnames <- colnames(X)
  for(i in 1:ncol(X)){
    if(!cnames[i] == "typewhite"){
      X[, i] <- scale(X[, i])
```

```r
    } else {
      colnames(X)[i] <- "type"
    }
  }
  return(X)
}
# Perform LASSO and Ridge Regression with `X` and `y`
library(glmnet)
lasso <- cv.glmnet(make_model_matrix(forward_formula), y, alpha = 1)
plot(lasso)
```

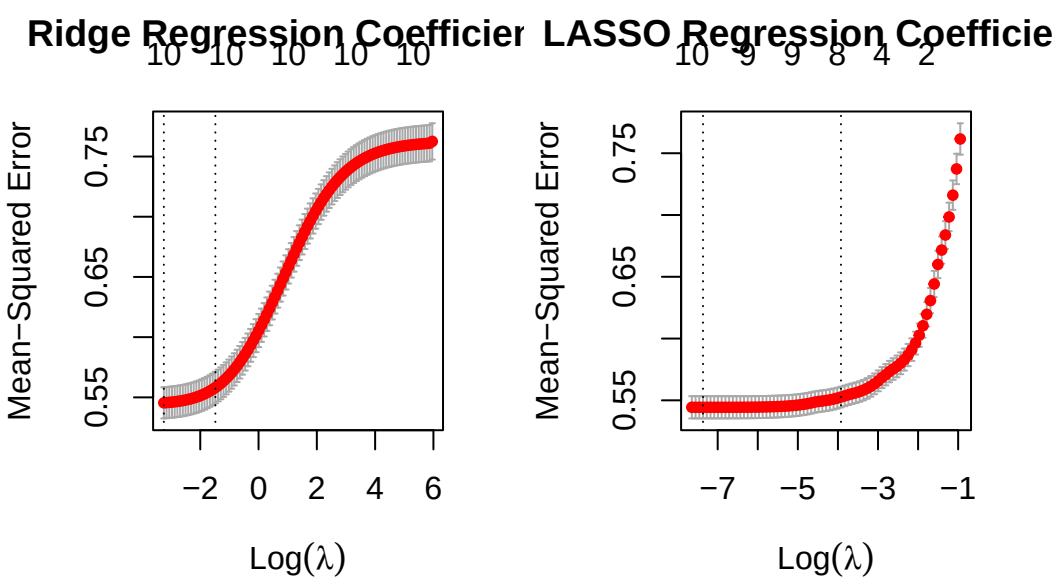



```r
library(glmnet)
ridge <- cv.glmnet(make_model_matrix(forward_formula), y, alpha = 0)
plot(ridge)
```



Create side-by-side plots of the ridge and LASSO regression results. Interpret your main findings.

```
par(mfrow=c(1, 2))
plot(ridge, main = "Ridge Regression Coefficients")
plot(lasso, main = "LASSO Regression Coefficients")
```



The regularization parameter lambda, which regulates the potency of the penalty term in the regression model, is represented by the x-axis in the generated figures. The coefficient values

for each predictor variable are shown on the y-axis. By examining the figures, we can see that as lambda increases, the coefficients for every predictor variable in both the ridge and LASSO regression models go closer and closer to zero. Yet, when lambda increases in LASSO regression, some predictors' coefficient estimates may be entirely zero, resulting in sparse models with few predictors. Ridge regression, on the other hand, never totally eliminates any predictors; rather, it gradually reduces each predictor's coefficient until it equals zero.

3.4 (5 points)

Print the coefficient values for LASSO regression at the `lambda.1se` value? What are the variables selected by LASSO?

Store the variable names with non-zero coefficients in `lasso_vars`, and create a formula object called `lasso_formula` using the `make_formula()` function provided in the Appendix.

```r
lasso_coef <- coef(lasso, s = "lambda.1se")
lasso_coef
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                               s1
(Intercept)           5.834205982
volatile_acidity     -0.210204144
citric_acid           .
residual_sugar        0.064638303
chlorides            -0.004937110
total_sulfur_dioxide -0.009652573
density               .
pH                    0.005328280
sulphates             0.068429778
alcohol               0.379580020
type                 -0.020995563
```

```r
make_formula <- function(x){
  as.formula(
    paste("quality ~ ", paste(x, collapse = " + "))
  )
}
lasso_vars <- rownames(lasso_coef)[which(abs(lasso_coef) > 0)][-1]
lasso_formula <- make_formula(lasso_vars)
```

```r
lasso_formula
```

```
quality ~ volatile_acidity + residual_sugar + chlorides + total_sulfur_dioxide +
    pH + sulphates + alcohol + type
<environment: 0x7fa37caa7df8>
```

3.5 (5 points)

Print the coefficient values for ridge regression at the `lambda.1se` value? What are the variables selected here?

Store the variable names with non-zero coefficients in `ridge_vars`, and create a formula object called `ridge_formula` using the `make_formula()` function provided in the Appendix.

```r
ridge_coef <- coef(ridge, s = "lambda.1se")
ridge_coef
```

```
11 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)           5.86565023
volatile_acidity     -0.16963262
citric_acid           0.02093447
residual_sugar        0.09332811
chlorides            -0.04701445
total_sulfur_dioxide -0.03990503
density              -0.08450948
pH                    0.02424940
sulphates             0.07985787
alcohol               0.26411918
type                 -0.06270510
```

```r
ridge_vars <- rownames(ridge_coef)[which(abs(ridge_coef) > 0)][-1]
ridge_formula <- make_formula(ridge_vars)
ridge_formula
```

```
quality ~ volatile_acidity + citric_acid + residual_sugar + chlorides +
    total_sulfur_dioxide + density + pH + sulphates + alcohol +
    type
<environment: 0x7fa37d3efbf0>
```

3.6 (10 points)

What is the difference between stepwise selection, LASSO and ridge based on you analyses above?

Stepwise selection adds and removes variables based on statistical criteria, such as p-values or AIC, in a sequential manner until a stopping criterion is reached. In our example, stepwise selection resulted in a model that included only `8` out of the 8 original predictor variables.LASSO regression, in particular, performs both variable selection and regularization by adding a penalty term to the objective function, which encourages sparse solutions where some coefficients are exactly zero. In our example, LASSO regression resulted in a model with only `4` predictor variables, all of which had non-zero coefficients.Ridge regression, on the other hand, shrinks the predictor variable coefficients in the direction of zero without actually setting any coefficients at zero. In our case, ridge regression produced a model with `ten` predictor variables, all of which had non-zero coefficients.

Question 4

70 points

Variable selection

4.1 (5 points)

Excluding `quality` from `df` we have 10 possible predictors as the covariates. How many different models can we create using any subset of these 10 coavriates as possible predictors? Justify your answer.

Each of the 10 potential predictors has two options: either it is incorporated into the model or it is not. Hence, there are two possibilities for each predictor. As a result, there are 2 x 10 = 1024 models that may be made utilizing any subset of these predictors as potential predictors for 10 predictors.To elaborate further, consider building a model with the first predictor to see why. There are two options: either we incorporate it or we don't. In a similar vein, we have two choices for the second predictor: include it or exclude it. As we proceed, we discover that there are two alternatives for each predictor, allowing us to build a total of 210 possible models.

4.2 (20 points)

Store the names of the predictor variables (all columns except `quality`) in an object called `x_vars`.

```
x_vars <- colnames(df %>% select(-quality))
```

Use:

- the `combn()` function (built-in R function) and
- the `make_formula()` (provided in the Appendix)

to **generate all possible linear regression formulas** using the variables in `x_vars`. This is most optimally achieved using the `map()` function from the `purrr` package.

```
formulas <- map(
  1:length(x_vars),
  function(x){
    vars <- combn(x_vars, x, simplify = FALSE) # Insert code here
    map(vars, make_formula) # Insert code here
  }
) %>% unlist()
```

If your code is right the following command should return something along the lines of:

```
sample(formulas, 4) %>% as.character()
```

```
[1] "quality ~ density + pH + sulphates"
[2] "quality ~ citric_acid + residual_sugar + density + type"
[3] "quality ~ chlorides + total_sulfur_dioxide + alcohol + type"
[4] "quality ~ volatile_acidity + chlorides + sulphates + alcohol"
```

```
# Output:
# [1] "quality ~ volatile_acidity + residual_sugar + density + pH + alcohol"
# [2] "quality ~ citric_acid"
# [3] "quality ~ volatile_acidity + citric_acid + residual_sugar + total_sulfur_dioxide +
# [4] "quality ~ citric_acid + chlorides + total_sulfur_dioxide + pH + alcohol + type"
```

4.3 (10 points)

Use `map()` and `lm()` to fit a linear regression model to each formula in `formulas`, using `df` as the data source. Use `broom::glance()` to extract the model summary statistics, and bind them together into a single tibble of summaries using the `bind_rows()` function from `dplyr`.

```
library(dplyr)
library(purrr)
library(broom)
models <- map(formulas, ~lm(.x, data = df)) # Insert your code here
summaries <- map(models, glance) %>% bind_rows # Insert your code here
summaries
```

```
# A tibble: 1,023 x 12
   r.squared adj.r.~1 sigma stati~2   p.value    df logLik    AIC    BIC devia~3
       <dbl>    <dbl> <dbl>   <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>   <dbl>
 1  0.0706   0.0705   0.842  493.   2.06e-105     1 -8100. 16206. 16226.   4604.
 2  0.00732  0.00716  0.870   47.9  5.00e- 12     1 -8314. 16634. 16654.   4917.
 3  0.00137  0.00121  0.873    8.89 2.87e-  3     1 -8333. 16673. 16693.   4947.
 4  0.0403   0.0401   0.856  273.   5.32e- 60     1 -8204. 16415. 16435.   4754.
 5  0.00171  0.00156  0.873   11.1  8.48e-  4     1 -8332. 16671. 16691.   4945.
 6  0.0935   0.0934   0.831  670.   9.66e-141     1 -8019. 16044. 16064.   4490.
 7  0.000380 0.000227 0.873    2.47 1.16e-  1     1 -8337. 16679. 16700.   4952.
 8  0.00148  0.00133  0.873    9.63 1.92e-  3     1 -8333. 16672. 16692.   4946.
 9  0.197    0.197    0.782 1598.   1.50e-312     1 -7623. 15253. 15273.   3976.
10  0.0142   0.0141   0.867   93.8  4.89e- 22     1 -8291. 16588. 16609.   4883.
# ... with 1,013 more rows, 2 more variables: df.residual <int>, nobs <int>,
#   and abbreviated variable names 1: adj.r.squared, 2: statistic, 3: deviance
```

4.4 (5 points)

Extract the `adj.r.squared` values from `summaries` and use them to identify the formula with the ***highest*** adjusted R-squared value.

```
# Extracting the adj_r_squared values from the summary table
adj_r_squared <- summaries$adj.r.squared

# Formula that gives the highest adjusted_r_squared_value
max_adj_r_index <- which.max(adj_r_squared)
```

Store resulting formula as a variable called `rsq_formula`.

```r
rsq_formula <- formulas[max_adj_r_index]
rsq_formula
```

```
[[1]]
quality ~ volatile_acidity + residual_sugar + chlorides + total_sulfur_dioxide +
    density + pH + sulphates + alcohol + type
<environment: 0x7fa380dce418>
```

4.5 (5 points)

Extract the `AIC` values from `summaries` and use them to identify the formula with the ***lowest*** AIC value.

```r
# Extracting the AIC values from the summary table
aic_val <- summaries$AIC

# Formula that gives the lowest AIC value
low_aic_index <- which.min(aic_val)
```

Store resulting formula as a variable called `aic_formula`.

```r
aic_formula <- formulas[low_aic_index]
aic_formula
```

```
[[1]]
quality ~ volatile_acidity + residual_sugar + chlorides + density +
    pH + sulphates + alcohol + type
<environment: 0x7fa380d312d0>
```

4.6 (15 points)

Combine all formulas shortlisted into a single vector called `final_formulas`.

```r
null_formula <- formula(null_model)
full_formula <- formula(full_model)
```

```r
final_formulas <- c(
  null_formula,
  full_formula,
  backward_formula,
  forward_formula,
  lasso_formula,
  ridge_formula,
  rsq_formula,
  aic_formula
)
```

- Are `aic_formula` and `rsq_formula` the same? How do they differ from the formulas shortlisted in question 3? > No, aic_formula and rsq_formula are not the same. aic_formula was selected based on minimizing the AIC value, while rsq_formula was selected based on maximizing the adjusted R-squared value. These two formulas will likely differ because they are optimized using different criteria.All of the formulae in question 3's shortlist are examples of linear regression formulas that could be produced using any subset of the 10 potential predictors. They were created exhaustively rather than being chosen based on any optimization criteria. As aic formula and rsq formula were selected from the shortlisted formulas, the shortlisted formulas in question 3 will be a superset of those two formulas.
- Which of these is more reliable? Why? I believe that the `aic_formula` is more reliable because the formula that was created is indistinguishable to the one which was created using the stepwise regression formulas.
- If we had a dataset with $10,000$ columns, which of these methods would you consider for your analyses? Why? Using stepwise selection would be computationally expensive and might take a long time to perform if we had a dataset with 10,000 columns. This is due to the fact that stepwise selection necessitates fitting a number of models with various variable combinations before choosing the best model based on a criterion, which can be time-consuming when there are a lot of variables. In this case, lasso and ridge regression might be more suitable. Lasso and ridge regression are computationally effective and are made to handle high-dimensional data. In instance, Lasso has the ability to execute variable selection by effectively deleting certain variables from the model by setting certain coefficients to zero. Hence, given a dataset with 10,000 columns, I would use Lasso or Ridge regression.

4.7 (10 points)

Use `map()` and `glance()` to extract the `sigma, adj.r.squared, AIC, df`, and `p.value` statistics for each model obtained from `final_formulas`. Bind them together into a single data frame `summary_table`. Summarize your main findings.

```
summary_table <- map_dfr(
  final_formulas, ~lm(.x, data = df)
    glance()%>%
    select(sigma, adj.r.squared, AIC, df, p.value)
) %>% bind_rows()

summary_table %>% knitr::kable()
```

We can see that each model has p-values that are statistically significant. The p-values were almost zero since they were so modest. Also, we can see that some of the methods produced the same model. The models produced by the backward, forward, and AIC approaches were all identical, while the entire model was produced by the ridge method. Besides the null model, we can also see that all of the sigma values, modified r-squard values, and AIC values were quite similar for all the models. The AIC values ranged from 144483.89 to 14520.61, the adjusted r-squared values from 0.283 to 0.288, and the sigma values from.737 to.739

Appendix

Convenience function for creating a formula object

The following function which takes as input a vector of column names `x` and outputs a `formula` object with `quality` as the response variable and the columns of `x` as the covariates.

```
make_formula <- function(x){
  as.formula(
    paste("quality ~ ", paste(x, collapse = " + "))
  )
}

# For example the following code will
# result in a formula object
# "quality ~ a + b + c"
make_formula(c("a", "b", "c"))
```

```
quality ~ a + b + c
<environment: 0x7fa35f549460>
```

Convenience function for `glmnet`

The `make_model_matrix` function below takes a `formula` as input and outputs a **rescaled** model matrix `X` in a format amenable for `glmnet()`

```
make_model_matrix <- function(formula){
  X <- model.matrix(formula, df)[, -1]
  cnames <- colnames(X)
  for(i in 1:ncol(X)){
    if(!cnames[i] == "typewhite"){
      X[, i] <- scale(X[, i])
    } else {
      colnames(X)[i] <- "type"
    }
  }
  return(X)
}
```

Session Information

Print your R session information using the following command

```r
sessionInfo()
```

```
R version 4.2.2 (2022-10-31)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur ... 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices datasets  utils     methods   base

other attached packages:
 [1] corrplot_0.92 broom_1.0.3   glmnet_4.1-6  Matrix_1.5-1  car_3.1-1
 [6] carData_3.0-5 purrr_1.0.1   dplyr_1.0.10  tidyr_1.2.1   readr_2.1.3

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.9       highr_0.10       pillar_1.8.1     compiler_4.2.2
 [5] iterators_1.0.14 tools_4.2.2      digest_0.6.31    lattice_0.20-45
 [9] jsonlite_1.8.4   evaluate_0.20    lifecycle_1.0.3  tibble_3.1.8
[13] pkgconfig_2.0.3  rlang_1.0.6      foreach_1.5.2    cli_3.6.0
[17] rstudioapi_0.14  yaml_2.3.6       xfun_0.36        fastmap_1.1.0
[21] withr_2.5.0      stringr_1.5.0    knitr_1.41       generics_0.1.3
[25] vctrs_0.5.1      hms_1.1.2        grid_4.2.2       tidyselect_1.2.0
[29] glue_1.6.2       R6_2.5.1         fansi_1.0.3      survival_3.4-0
[33] rmarkdown_2.20   tzdb_0.3.0       magrittr_2.0.3   backports_1.4.1
[37] splines_4.2.2    codetools_0.2-18 ellipsis_0.3.2   htmltools_0.5.4
[41] abind_1.4-5      shape_1.4.6      renv_0.16.0-53   utf8_1.2.2
[45] stringi_1.7.12
```