Homework 3

jiaxi li

Table of contents

! Important

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(Matrix)
library(readr)
library(tidyr)
```

```
Attaching package: 'tidyr'

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

    expand, pack, unpack
```

```r
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)
```

```
Loaded glmnet 4.1-8
```

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, sep=';') # Insert your code here
df2 <- read.csv(url2, sep=';') # Insert your code here


head(df1)
```

```
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1           7.0             0.27        0.36           20.7     0.045
2           6.3             0.30        0.34            1.6     0.049
3           8.1             0.28        0.40            6.9     0.050
4           7.2             0.23        0.32            8.5     0.058
5           7.2             0.23        0.32            8.5     0.058
6           8.1             0.28        0.40            6.9     0.050
```

```
  free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
1                  45                  170  1.0010 3.00      0.45     8.8
2                  14                  132  0.9940 3.30      0.49     9.5
3                  30                   97  0.9951 3.26      0.44    10.1
4                  47                  186  0.9956 3.19      0.40     9.9
5                  47                  186  0.9956 3.19      0.40     9.9
6                  30                   97  0.9951 3.26      0.44    10.1
  quality
1       6
2       6
3       6
4       6
5       6
6       6
```

```r
head(df2)
```

```
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1           7.4             0.70        0.00            1.9     0.076
2           7.8             0.88        0.00            2.6     0.098
3           7.8             0.76        0.04            2.3     0.092
4          11.2             0.28        0.56            1.9     0.075
5           7.4             0.70        0.00            1.9     0.076
6           7.4             0.66        0.00            1.8     0.075
  free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
1                  11                   34  0.9978 3.51      0.56     9.4
2                  25                   67  0.9968 3.20      0.68     9.8
3                  15                   54  0.9970 3.26      0.65     9.8
4                  17                   60  0.9980 3.16      0.58     9.8
5                  11                   34  0.9978 3.51      0.56     9.4
6                  13                   40  0.9978 3.51      0.56     9.4
  quality
1       5
2       5
3       5
4       6
5       5
6       5
```

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.

1

```r
df1 <- mutate(df1,type= "white")
df2 <- mutate(df2,type = "red")
df_list <- list(df1, df2)
combined_df <- Reduce(function(x, y) merge(x, y, all=TRUE), df_list)
head(combined_df)
```

```
  fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
1           3.8            0.310        0.02           11.1     0.036
2           3.9            0.225        0.40            4.2     0.030
3           4.2            0.170        0.36            1.8     0.029
4           4.2            0.215        0.23            5.1     0.041
5           4.4            0.320        0.39            4.3     0.030
6           4.4            0.460        0.10            2.8     0.024
  free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
1                  20                  114 0.99248 3.75      0.44    12.4
2                  29                  118 0.98900 3.57      0.36    12.8
3                  93                  161 0.98999 3.65      0.89    12.0
4                  64                  157 0.99688 3.42      0.44     8.0
5                  31                  127 0.98904 3.46      0.36    12.8
6                  31                  111 0.98816 3.48      0.34    13.1
  quality  type
1       6 white
2       8 white
3       7 white
4       3 white
5       8 white
6       6 white
```

```r
df <- gsub(" ", "_", combined_df)
head(df)
```

```
[1] "c(3.8,_3.9,_4.2,_4.2,_4.4,_4.4,_4.4,_4.5,_4.6,_4.6,_4.7,_4.7,_4.7,_4.7,_4.7,_4.7,_4.8,_4
[2] "c(0.31,_0.225,_0.17,_0.215,_0.32,_0.46,_0.54,_0.19,_0.445,_0.52,_0.145,_0.335,_0.455,_0
[3] "c(0.02,_0.4,_0.36,_0.23,_0.39,_0.1,_0.09,_0.21,_0,_0.15,_0.29,_0.14,_0.18,_0.17,_0.09,_0
[4] "c(11.1,_4.2,_1.8,_5.1,_4.3,_2.8,_5.1,_0.95,_1.4,_2.1,_1,_1.3,_1.9,_2.3,_1,_3.4,_1.2,_2.9
[5] "c(0.036,_0.03,_0.029,_0.041,_0.03,_0.024,_0.038,_0.033,_0.053,_0.054,_0.042,_0.036,_0.03
[6] "c(20,_29,_93,_64,_31,_31,_52,_89,_11,_8,_35,_69,_33,_17,_5,_23,_40,_22,_17,_47,_23,_38,_
```

```r
new_df <- combined_df %>% select(-fixed.acidity, -free.sulfur.dioxide)
head(new_df)
```

```
  volatile.acidity citric.acid residual.sugar chlorides total.sulfur.dioxide
1            0.310        0.02           11.1     0.036                  114
2            0.225        0.40            4.2     0.030                  118
3            0.170        0.36            1.8     0.029                  161
4            0.215        0.23            5.1     0.041                  157
5            0.320        0.39            4.3     0.030                  127
6            0.460        0.10            2.8     0.024                  111
  density   pH sulphates alcohol quality  type
1 0.99248 3.75      0.44    12.4       6 white
2 0.98900 3.57      0.36    12.8       8 white
3 0.98999 3.65      0.89    12.0       7 white
4 0.99688 3.42      0.44     8.0       3 white
5 0.98904 3.46      0.36    12.8       8 white
6 0.98816 3.48      0.34    13.1       6 white
```

```r
new_df <- new_df %>%
  na.omit() %>%
  mutate(type = as.factor(type))

head(new_df)
```

```
  volatile.acidity citric.acid residual.sugar chlorides total.sulfur.dioxide
1            0.310        0.02           11.1     0.036                  114
2            0.225        0.40            4.2     0.030                  118
3            0.170        0.36            1.8     0.029                  161
4            0.215        0.23            5.1     0.041                  157
```

```
5            0.320        0.39            4.3     0.030                    127
6            0.460        0.10            2.8     0.024                    111
  density   pH sulphates alcohol quality  type
1 0.99248 3.75      0.44    12.4       6 white
2 0.98900 3.57      0.36    12.8       8 white
3 0.98999 3.65      0.89    12.0       7 white
4 0.99688 3.42      0.44     8.0       3 white
5 0.98904 3.46      0.36    12.8       8 white
6 0.98816 3.48      0.34    13.1       6 white
```

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`.

```r
df_white <- new_df %>%
  filter(type == "white") %>%
  summarise(mean(quality))
df_red <- new_df %>%
  filter(type == "red") %>%
  summarise(mean(quality))
```

```r
n_white <- new_df %>%
  filter(type == "white") %>%
  nrow()

n_red <- new_df %>%
  filter(type == "red") %>%
  nrow()
```

```r
var_white <- new_df %>%
  filter(type == "white") %>%
  summarise(var(quality))

var_red <- new_df %>%
  filter(type == "red") %>%
  summarise(var(quality))
```

```r
diff_mean <- df_white - df_red # Insert your code here
sp_squred <- (((n_white - 1) * var_white) + ((n_red - 1) * var_red)) / (n_white + n_red -
t1 <- diff_mean / sqrt(sp_squred * (1/n_white + 1/n_red)) # Insert your code here

diff_mean
```

```
  mean(quality)
1     0.2418868
```

```r
sp_squred
```

```
  var(quality)
1    0.7518329
```

```r
t1
```

```
  mean(quality)
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`.

```r
white_wine_data <- new_df %>%
  filter(type == "white")
red_wine_data <- new_df %>%
  filter(type == "red")
```

```r
t_test <- t.test(quality ~ type,data = new_df,var.equal = TRUE) # Insert your code here
t2 <- t_test$statistic # Insert your code here
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 = new_df) # Insert your here
model_summary <- summary(fit)
t3 <- model_summary$coefficients["typewhite", "t value"] # Insert your here
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
print(c(t1, t2, t3)) # Insert your code here)
```

```
$`mean(quality)`
[1] 9.68565
```

```
$t
[1] -9.68565

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

both the manual calculation and the automated methods (t.test() and lm()) are in agreement about the effect of type on quality. but from the function which from t2, the vector was negative sign.

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
full_model <- lm(quality ~ . ,data = new_df) # Insert your code here
tidy_fullmodel <- broom::tidy(full_model)
print(tidy_fullmodel)
```

```
# 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
```

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 = new_df)    # Insert your code here
```

```r
model_sulfur <- lm(quality ~ total.sulfur.dioxide, data = new_df)# Insert your code here
```

```r
summary_citric <- summary(model_citric)
summary_sulfur <- summary(model_sulfur)
print(summary_citric)
```

```
Call:
lm(formula = quality ~ citric.acid, data = new_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
print(summary_sulfur)
```

```
Call:
lm(formula = quality ~ total.sulfur.dioxide, data = new_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
```

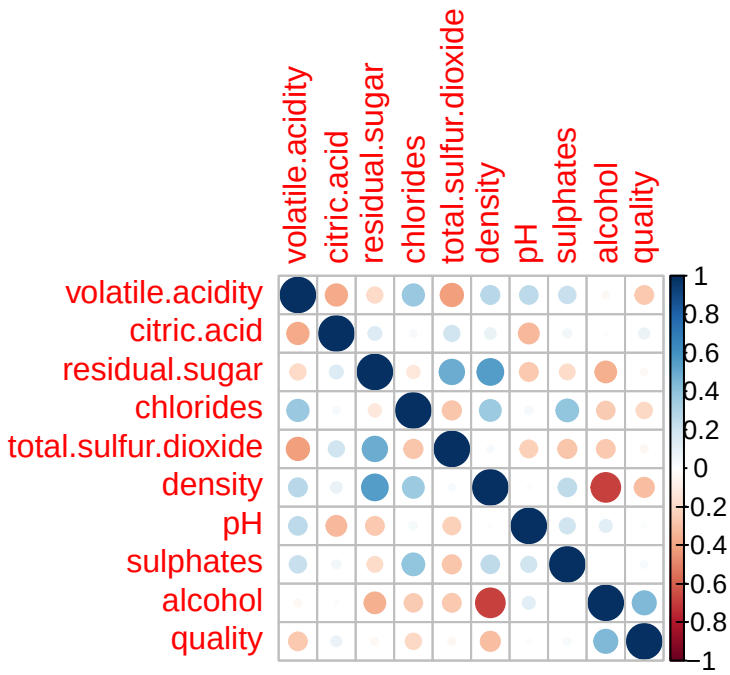
2.3 (5 points)

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

```r
library(corrplot)
```

```
corrplot 0.92 loaded
```

```r
numeric_df <- new_df[sapply(new_df, is.numeric)]
cor_matrix <- cor(numeric_df)
corrplot(cor_matrix, method = "circle")
```



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?

```
vif_values <- vif(full_model)
print(vif_values)
```

```
  volatile.acidity           citric.acid        residual.sugar
          2.103853              1.549248              4.680035
         chlorides total.sulfur.dioxide               density
          1.625065              2.628534              9.339357
                pH             sulphates               alcohol
          1.352005              1.522809              3.419849
              type
          6.694679
```

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

```
backward_formula <- step(full_model,direction = "backward")
```

```
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
```

```r
print(backward_formula)
```

```
Call:
lm(formula = quality ~ volatile.acidity + residual.sugar + chlorides +
    density + pH + sulphates + alcohol + type, data = new_df)

Coefficients:
     (Intercept)  volatile.acidity    residual.sugar         chlorides
        57.22518          -1.62632           0.04425          -0.95067
         density                pH         sulphates           alcohol
       -54.87625           0.17589           0.65234           0.28073
       typewhite
        -0.41760
```

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

```
null_model <- lm(quality ~ 1, data = new_df)
forward_formula <- step(null_model,direction = "forward")
```

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

```
print(backward_formula)
```

```
Call:
lm(formula = quality ~ volatile.acidity + residual.sugar + chlorides +
    density + pH + sulphates + alcohol + type, data = new_df)

Coefficients:
     (Intercept)  volatile.acidity    residual.sugar         chlorides
        57.22518          -1.62632           0.04425          -0.95067
         density                pH         sulphates           alcohol
       -54.87625           0.17589           0.65234           0.28073
       typewhite
        -0.41760
```

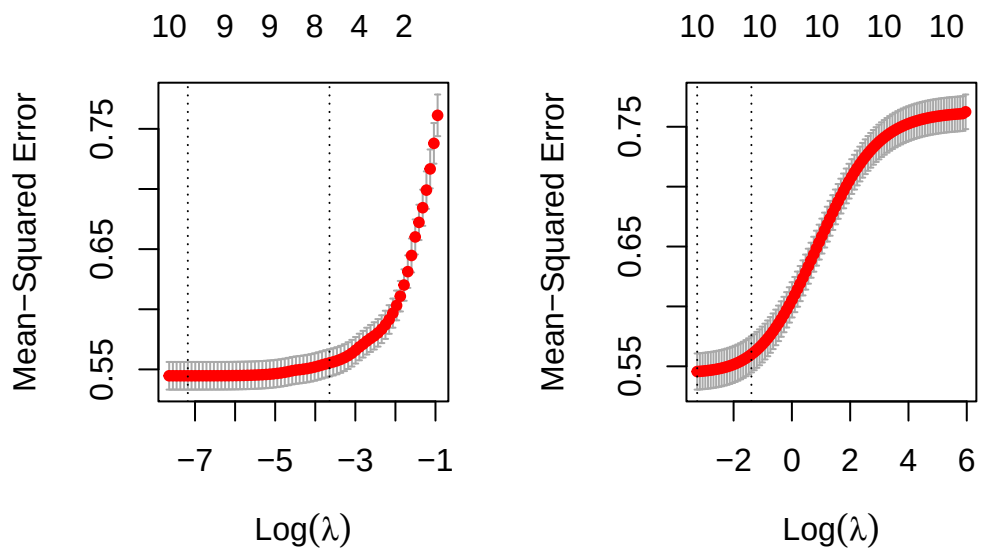
3.3 (10 points)

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`.

```
y <- new_df$quality
X <- model.matrix(~ . - quality, data = new_df)
lasso_model <- cv.glmnet(X, y, alpha = 1)
ridge_model <- cv.glmnet(X, y, alpha = 0)
```

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

```
par(mfrow = c(1, 2))
plot(lasso_model)
plot(ridge_model) # Insert your code here.
```



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.

```
lasso_coefs <- coef(lasso_model, s = "lambda.1se")
print(lasso_coefs)
```

```
12 x 1 sparse Matrix of class "dgCMatrix"
                                s1
```

```
(Intercept)           2.67453965
(Intercept)           .
volatile.acidity     -1.19309408
citric.acid           .
residual.sugar        0.01015105
chlorides             .
total.sulfur.dioxide  .
density               .
pH                    .
sulphates             0.41647996
alcohol               0.31191732
typewhite             .
```

```r
non_zero_lasso <- lasso_coefs != 0



lasso_vars <- row.names(non_zero_lasso)
lasso_formula <- as.formula(paste("quality ~", paste(lasso_vars[-1], collapse = "+")))
print(lasso_formula)
```

```
quality ~ (Intercept) + volatile.acidity + citric.acid + residual.sugar +
    chlorides + total.sulfur.dioxide + density + pH + sulphates +
    alcohol + typewhite
```

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_coefs <- coef(ridge_model, s = "lambda.1se")
print(ridge_coefs)
```

```
12 x 1 sparse Matrix of class "dgCMatrix"
                                 s1
(Intercept)           30.893009885
```

```
(Intercept)            .
volatile.acidity      -1.000495440
citric.acid            0.149886471
residual.sugar         0.018618685
chlorides             -1.354686257
total.sulfur.dioxide  -0.000699345
density              -27.859683911
pH                     0.147945418
sulphates              0.524707405
alcohol                0.216471090
typewhite             -0.054662741
```

```r
non_zero_ridge <- ridge_coefs != 0

ridge_vars <- row.names(non_zero_ridge)
ridge_formula <- as.formula(paste("quality ~", paste(ridge_vars[-1], collapse = "+")))
print(ridge_formula)
```

```
quality ~ (Intercept) + volatile.acidity + citric.acid + residual.sugar +
    chlorides + total.sulfur.dioxide + density + pH + sulphates +
    alcohol + typewhite
```

3.6 (10 points)

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

Stepwise selection is simple but lacks automatic selection and can overfit.LASSO automatically selects variables and encourages sparsity.Ridge regression stabilizes the model without eliminating predictors.

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.

```
number_of_models <- 2^10
print(number_of_models)
```

```
[1] 1024
```

4.2 (20 points)

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

```
x_vars <- colnames(new_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.

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

```r
# 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: 0x000001d6f3089298>
```

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

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

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

```
[1] "quality ~ volatile.acidity + citric.acid + residual.sugar + density"
[2] "quality ~ residual.sugar + chlorides + pH + sulphates"
[3] "quality ~ volatile.acidity + density + pH"
[4] "quality ~ citric.acid + residual.sugar + chlorides + total.sulfur.dioxide + pH + sulpha
```

```r
# 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`.

```r
models <- map(formulas, ~lm(., data = new_df))
summaries <- map(models, broom::glance) %>% bind_rows()
```

```r
head(summaries)
```

```
# A tibble: 6 x 12
  r.squared adj.r.squared sigma statistic   p.value    df logLik    AIC    BIC
      <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl>  <dbl>  <dbl>
1   0.0706        0.0705  0.842    493.   2.06e-105     1 -8100. 16206. 16226.
2   0.00732       0.00716 0.870     47.9  5.00e- 12     1 -8314. 16634. 16654.
3   0.00137       0.00121 0.873      8.89 2.87e-  3     1 -8333. 16673. 16693.
4   0.0403        0.0401  0.856    273.   5.32e- 60     1 -8204. 16415. 16435.
5   0.00171       0.00156 0.873     11.1  8.48e-  4     1 -8332. 16671. 16691.
6   0.0935        0.0934  0.831    670.   9.66e-141     1 -8019. 16044. 16064.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

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.

```r
max_rsq_formula <- which.max(summaries$adj.r.squared) # Insert your code here
```

```r
max_rsq_formula
```

```
[1] 1021
```

Store resulting formula as a variable called `rsq_formula`.

```r
rsq_formula <- summaries$r.squared[max_rsq_formula]
rsq_formula
```

```
[1] 0.2886986
```

4.5 (5 points)

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

```r
min_aic_formula <- which.min(summaries$AIC)
min_aic_formula
```

```
[1] 1001
```

Store resulting formula as a variable called `aic_formula`.

```r
aic_formula <- summaries$AIC[min_aic_formula]
aic_formula
```

```
[1] 14483.89
```

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)

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?
- Which of these is more reliable? Why?
- If we had a dataset with $10,000$ columns, which of these methods would you consider for your analyses? Why?

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(
#  final_formulas,
#  \(x)  lm(x, data = new_df) %>%
#  glance()) %>%
#  bind_rows()

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

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: 0x000001d6e7e7d5c8>
```

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 4.3.2 (2023-10-31 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 11 x64 (build 22631)

Matrix products: default


locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

time zone: America/New_York
tzcode source: internal

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

other attached packages:
[1] corrplot_0.92  glmnet_4.1-8   car_3.1-2      carData_3.0-5  purrr_1.0.2
[6] dplyr_1.1.4    tidyr_1.3.1    readr_2.1.5    Matrix_1.6-1.1

loaded via a namespace (and not attached):
 [1] jsonlite_1.8.8    compiler_4.3.2    tidyselect_1.2.0  Rcpp_1.0.12
 [5] splines_4.3.2     yaml_2.3.8        fastmap_1.1.1     lattice_0.21-9
 [9] R6_2.5.1          generics_0.1.3    shape_1.4.6       knitr_1.45
[13] backports_1.4.1   iterators_1.0.14  tibble_3.2.1      pillar_1.9.0
[17] tzdb_0.4.0        rlang_1.1.3       utf8_1.2.4        broom_1.0.5
[21] xfun_0.41         cli_3.6.2         withr_3.0.0       magrittr_2.0.3
[25] digest_0.6.34     foreach_1.5.2     grid_4.3.2        rstudioapi_0.15.0
[29] hms_1.1.3         lifecycle_1.0.4   vctrs_0.6.5       evaluate_0.23
[33] glue_1.7.0        codetools_0.2-19  survival_3.5-7    abind_1.4-5
[37] fansi_1.0.6       rmarkdown_2.25    tools_4.3.2       pkgconfig_2.0.3
[41] htmltools_0.5.7
```