Homework 3

Marc Hughes

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)
```

```
Warning: package 'tidyr' was built under R version 4.2.2
```

```r
library(dplyr)
```

```
Warning: package 'dplyr' was built under R version 4.2.2

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)
```

```
Warning: package 'purrr' was built under R version 4.2.2
```

```r
library(car)
```

```
Warning: package 'car' was built under R version 4.2.2

Loading required package: carData

Warning: package 'carData' was built under R version 4.2.2

Attaching package: 'car'
```

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

    some

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

    recode
```

```r
library(glmnet)
```

```
Warning: package 'glmnet' was built under R version 4.2.2

Loading required package: Matrix

Attaching package: 'Matrix'

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

    expand, pack, unpack

Loaded glmnet 4.1-6
```

```r
library(corrplot)
```

```
Warning: package 'corrplot' was built under R version 4.2.2

corrplot 0.92 loaded
```

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 = ';')
df2 <- read.csv(url2, 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 new column to both data frames that will distinguish the two types of wine when b
df1$type = "white"
df2$type = "red"

# binding both data frames
df <- rbind(df1, df2)

# replacing the periods with and underscore
colnames(df) <- gsub('\\.', '_', colnames(df))

# removing columns 'fixed_acidity' and 'free_sulfur_dioxide' from the data frame
df <- df %>%
  select(!fixed_acidity & !free_sulfur_dioxide)

# changing the 'type' column to a factor
df$type <- factor(df$type)

# dropping any missing values
df <- df %>%
  drop_na()
```

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
# creating temporary df to calculate mean
temp_df <-
  df %>%
  group_by(type) %>%
  summarise("quality_mean" = mean(quality))

# calculating mean
diff_mean <- abs(temp_df$quality_mean[temp_df$type == "white"] - temp_df$quality_mean[temp

# finding lengths
n1 <- length(df$quality[df$type == "white"])
n2 <- length(df$quality[df$type == "red"])

var1 <- var(df$quality[df$type == "white"])
var2 <- var(df$quality[df$type == "red"])

# manually calculating sp_squared
sp_squared <- ((n1-1)*var1 + (n2-1)*var2) / (n1+n2-2)

# calculating standard deviation
sd1 <- sd(df$quality[df$type == "white"])
sd2 <- sd(df$quality[df$type == "red"])

# calculating the t-statistic
```

```r
t1 <-  diff_mean / sqrt(sp_squared*(1/n1 + 1/n2))
```

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
# using 't.test()' function to calculate the t-statistic
t_test <- t.test(df$quality[df$type == "white"], df$quality[df$type == "red"], var.equal =
t2 <- t_test$statistic
```

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)$coefficients[2, "t value"]
```

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)
```

```
               t
9.68565 9.68565 9.68565
```

You can conclude that all three of the methods are valid ways of extracting the t-statistic due to all t values being the exact same regardless of the method. In addition, this shows that the t-statistic is significant which allows us to reject the null hypothesis.

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?

```
fit_all <- lm(quality ~ ., data = df)
broom::tidy(summary(fit_all))
```

```
# 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
```

We can conclude that all of the p-values are significant which means we reject the null hypothesis and accept the alternative. In addition, the summary shows, evident by the "statistic" column, that only a select few of the t values are statistically significant.

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
```

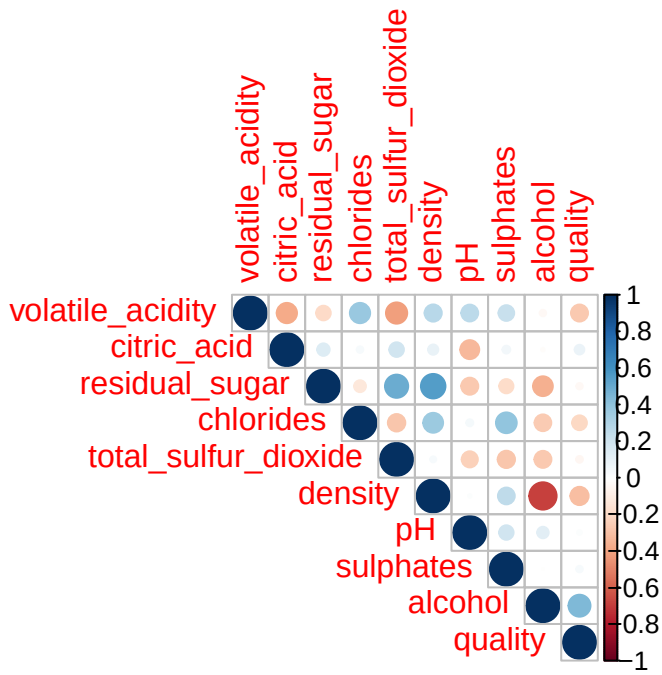
The t value of the "citric_acid" predictor under the model_citric model shows a large increase to the t value shown in the model with all predictors. The increase was so much so that "citric_acid" actually has a significant t value in "model_critic". On the other hand, the t value of the "total_sulfur_dioxide" predictor is even smaller and remains insignificant.

2.3 (5 points)

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

```r
# creating data frame with only numeric columns
df_numeric <- df %>%
  keep(is.numeric)

# creating correlation matrix
cor_mat <- cor(df_numeric)
# visualizing the correlation matrix
corrplot(cor_mat, type = 'upper')
```



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(fit_all)
```

```
  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
```

We can conclude that predictors like; 'volatile_acidity', 'residual_sugar', 'total_sulfur_dioxide', 'density', 'alcohol', and 'type' all of relatively high variance inflation factors. Having a high variance inflation means that the predictors listed are highly correlated with other variables in the model. This means that values like the t-statistic and p-value vastly different for variables

with a high VIF depending on the inclusion of other highly correlated variables within 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

```
... # Insert your code here
```

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

```
... # Insert your code here
```

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`.

```
... # Insert your code here.
```

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

```
par(mfrow=c(1, 2))
... # 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.

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.

3.6 (10 points)

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

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.

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),
  \(x){
    vars <- combn(...) # Insert code here
    map(vars, ...) # Insert code here
  }
) %>% unlist()
```

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

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

```
models <- map(formulas, ...) # Insert your code here
summaries <- map(models, ...) # Insert your code here
```

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.

```
... # Insert your code here
```

Store resulting formula as a variable called `rsq_formula`.

```
rsq_formula <- ... # Insert your code
```

4.5 (5 points)

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

```
... # Insert your code here
```

Store resulting formula as a variable called `aic_formula`.

```
aic_formula <- ... # Insert your code
```

4.6 (15 points)

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

```
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) ... # Insert your code here
) %>% 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"))
```

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 4.2.1 (2022-06-23 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 22000)

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

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

other attached packages:
[1] corrplot_0.92 glmnet_4.1-6  Matrix_1.4-1  car_3.1-1     carData_3.0-5
[6] purrr_1.0.1   dplyr_1.1.0   tidyr_1.3.0   readr_2.1.4

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.10      pillar_1.8.1     compiler_4.2.1   iterators_1.0.14
 [5] tools_4.2.1      digest_0.6.31    jsonlite_1.8.4   evaluate_0.20
 [9] lifecycle_1.0.3  tibble_3.1.8     lattice_0.20-45  pkgconfig_2.0.3
[13] rlang_1.0.6      foreach_1.5.2    cli_3.6.0        yaml_2.3.7
[17] xfun_0.37        fastmap_1.1.1    withr_2.5.0      knitr_1.42
[21] generics_0.1.3   vctrs_0.5.2      hms_1.1.2        grid_4.2.1
[25] tidyselect_1.2.0 glue_1.6.2       R6_2.5.1         fansi_1.0.4
[29] survival_3.3-1   rmarkdown_2.20   tzdb_0.3.0       magrittr_2.0.3
[33] backports_1.4.1  splines_4.2.1    codetools_0.2-18 ellipsis_0.3.2
[37] htmltools_0.5.4  abind_1.4-5      shape_1.4.6      renv_0.16.0-53
[41] utf8_1.2.3       broom_1.0.3
```