Jacob Adams: HomeWork 4

❗ 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

We will be using the following libraries:

```
packages <- c(
  "dplyr",
  "readr",
  "tidyr",
  "purrr",
  "stringr",
  "corrplot",
  "car",
  "caret",
  "torch",
  "nnet",
  "broom"
)

#renv::install(packages)
sapply(packages, require, character.only=T)
```

```
Loading required package: dplyr
```

```
Warning: package 'dplyr' was built under R version 4.3.3


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

Loading required package: readr

Warning: package 'readr' was built under R version 4.3.3

Loading required package: tidyr

Warning: package 'tidyr' was built under R version 4.3.3

Loading required package: purrr

Warning: package 'purrr' was built under R version 4.3.3

Loading required package: stringr

Loading required package: corrplot

Warning: package 'corrplot' was built under R version 4.3.3

corrplot 0.92 loaded

Loading required package: car

Warning: package 'car' was built under R version 4.3.3

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

Loading required package: caret

Warning: package 'caret' was built under R version 4.3.3

Loading required package: ggplot2

Loading required package: lattice


Attaching package: 'caret'

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

    lift

Loading required package: torch

Warning: package 'torch' was built under R version 4.3.3

Loading required package: nnet

Warning: package 'nnet' was built under R version 4.3.3

Loading required package: broom

Warning: package 'broom' was built under R version 4.3.3
```

```
   dplyr    readr    tidyr    purrr  stringr corrplot      car    caret 
    TRUE     TRUE     TRUE     TRUE     TRUE     TRUE     TRUE     TRUE 
   torch     nnet    broom 
    TRUE     TRUE     TRUE 
```

```
library(torch)
library(dplyr)
library(tidyr)
library(purrr)
library(stringr)
library(corrplot)
library(car)
library(caret)
library(nnet)
library(broom)
#torch :: install_torch()
```

Question 1

30 points

Automatic differentiation using `torch`

1.1 (5 points)

Consider $g(x, y)$ given by

$$g(x, y) = (x - 3)^2 + (y - 4)^2.$$

Using elementary calculus derive the expressions for

$$\frac{d}{dx} g(x, y), \quad \text{and} \quad \frac{d}{dy} g(x, y).$$

Using elementary calculus to solve:

$$\frac{d}{dx} g(x, y) = 2x - 6, \quad \text{and} \quad \frac{d}{dy} g(x, y) = 2y - 8.$$

Using your answer from above, what is the answer to

$$\frac{d}{dx} g(x, y) \Bigg|_{(x=3, y=4)} \quad \text{and} \quad \frac{d}{dy} g(x, y) \Bigg|_{(x=3, y=4)} \quad ?$$

Solutions plugging in $x = 3$, $y = 4$ for partial derivative with respect to x and y. As we can see, both expressions equal zero.

$$\frac{d}{dx} g(x,y)\bigg|_{(x=3,y=4)} = 0 \quad \text{and} \quad \frac{d}{dy} g(x,y)\bigg|_{(x=3,y=4)} = 0$$

Define $g(x,y)$ as a function in R, compute the gradient of $g(x,y)$ with respect to $x = 3$ and $y = 4$. Does the answer match what you expected?

Yes the answer matches my answer from above

```r
library(torch)
#torch::install_torch(reinstall=TRUE)
g <- function(x,y){
  return((x-3)^2 +(y-4)^2)

}
x <- torch_tensor(3,  requires_grad = TRUE)
y <- torch_tensor(4,  requires_grad = TRUE)
z <- g(x,y)
z$backward()

part_x <- x$grad
part_y <- y$grad

print(part_x$item())
```

```
[1] 0
```

```r
print(part_y$item())
```

```
[1] 0
```

1.2 (10 points)

Consider $h(u,v)$ given by

$$h(u,v) = (u \cdot v)^3,$$

where ˇ· denotes the dot product of two vectors, i.e.,

$$\cdot = \sum_{i=1}^{n} u_i v_i.$$

Using elementary calculus derive the expressions for the gradients

The expression of the gradient is below

3(u dot v)^2 dot u

Using your answer from above, what is the answer to the gradient of h(u,v) when n=10 and

$$\begin{aligned} &= (-1, +1, -1, +1, -1, +1, -1, +1, -1, +1) \\ &= (-1, -1, -1, -1, -1, +1, +1, +1, +1, +1) \end{aligned}$$

When n = 10 and vectors u and v have varying values of 1 and -1, the gradient is a vector of length 10 where the first half is -12 and the second half is positive 12.

Define $h(,)$ as a function in R, and initialize the two vectors˘ and˘ as `torch_tensor`s. Compute the gradient of $h(,)$ with respect to˘. Does the answer match what you expected? *Yes*

```r
h <- function(u,v){
  dot <- sum( u * v)
  return(dot^3)
}

u <- c(-1, 1, -1, 1, -1, 1, -1, 1, -1, 1)
v <- c(-1,-1,-1,-1,-1,1,1,1,1,1)

u <- torch_tensor(u, requires_grad = TRUE)
v <- torch_tensor(v, requires_grad = TRUE)
z <- h(u,v)


z$backward()
grad_u <- u$grad
print(grad_u)
```

```
torch_tensor
-12
-12
-12
-12
```

```
-12
 12
 12
 12
 12
 12
[ CPUFloatType{10} ]
```

1.3 (5 points)

Consider the following function

$$f(z) = z^4 - 6z^2 - 3z + 4$$

Derive the expression for

$$f'(z_0) = \left.\frac{df}{dz}\right|_{z=z_0}$$

and evaluate $f'(z_0)$ when $z_0 = -3.5$.

$$f'(z_0) = \left.\frac{df}{dz}\right|_{z=z_0} = 4(z_0)^3 - 12z_0 - 3$$

Define $f(z)$ as a function in R, and using the `torch` library compute $f'(-3.5)$.

```r
f_z <- function(z_0){
  return_value = (z_0^4) - 6 *(z_0^2) - 3*(z_0) + 4
  return(return_value)
}

z <- torch_tensor(-3.5, requires_grad = TRUE)
ans <- f_z(z)
ans$backward()
gradient <- z$grad

print(gradient)
```

```
torch_tensor
-132.5000
[ CPUFloatType{1} ]
```

I could not get this gradient descent function to work.

1.4 (5 points)

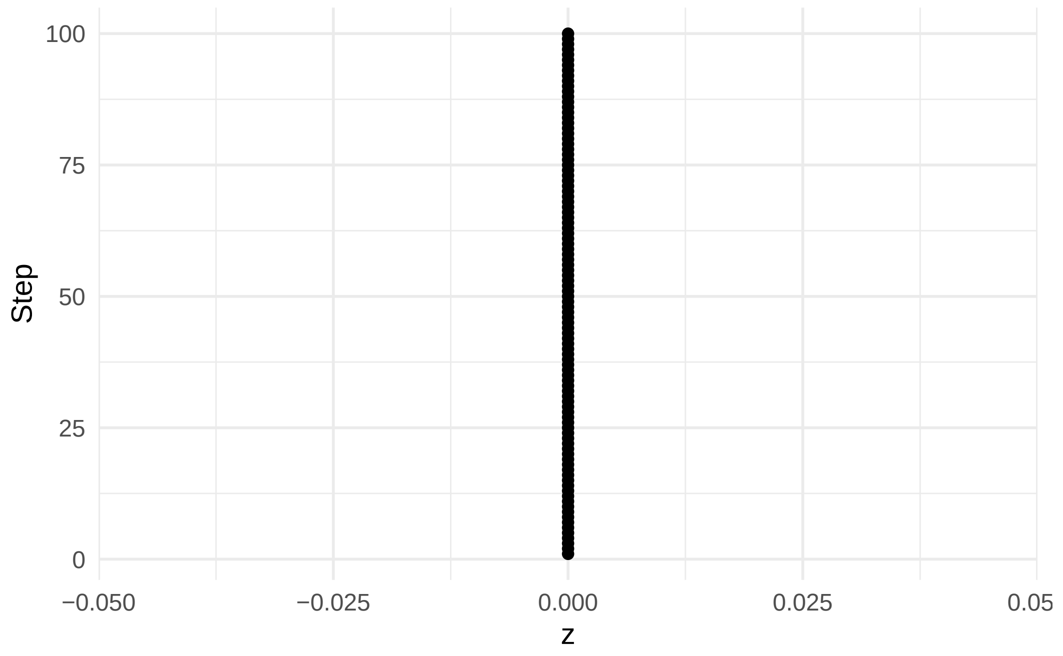
For the same function f, initialize $z[1] = -3.5$, and perform $n = 100$ iterations of **gradient descent**, i.e.,

$z[{k+1}] = z[k] - f'(z[k]) $ for $k = 1, 2, \ldots, 100$

Plot the curve f and add taking $\eta = 0.02$, add the points $\{z_0, z_1, z_2, \ldots z_{100}\}$ obtained using gradient descent to the plot. What do you observe?

```r
f_z <- function(z_0){
  return_value = (z_0^4) - 6 *(z_0^2) - 3*(z_0) + 4
  return(return_value)
}
eta <- 0.02
z <- torch_tensor(-3.5, requires_grad = TRUE)

steps <- 100
gradient_vector <- numeric(100)
# Perform gradient descent
#for (k in 1:steps) {
  #output <- f_z(z)

  #output$backward()

  #gradient <- z$grad
  #gradient<- gradient$to(device = torch_device("cpu"))

  #z <- z[k] - eta * gradient
  #gradient_vector[k] <- z$item()
  #z$zero_grad()
#}
```

```r
df_grad <- data.frame(z = gradient_vector, k = 1:steps)
ggplot(df_grad, aes(x = z, y = k)) +
  geom_line(stat = "function", fun = f_z, color = "blue") +
  geom_point() +
  labs(x = "z", y = "Step") +
  theme_minimal()
```

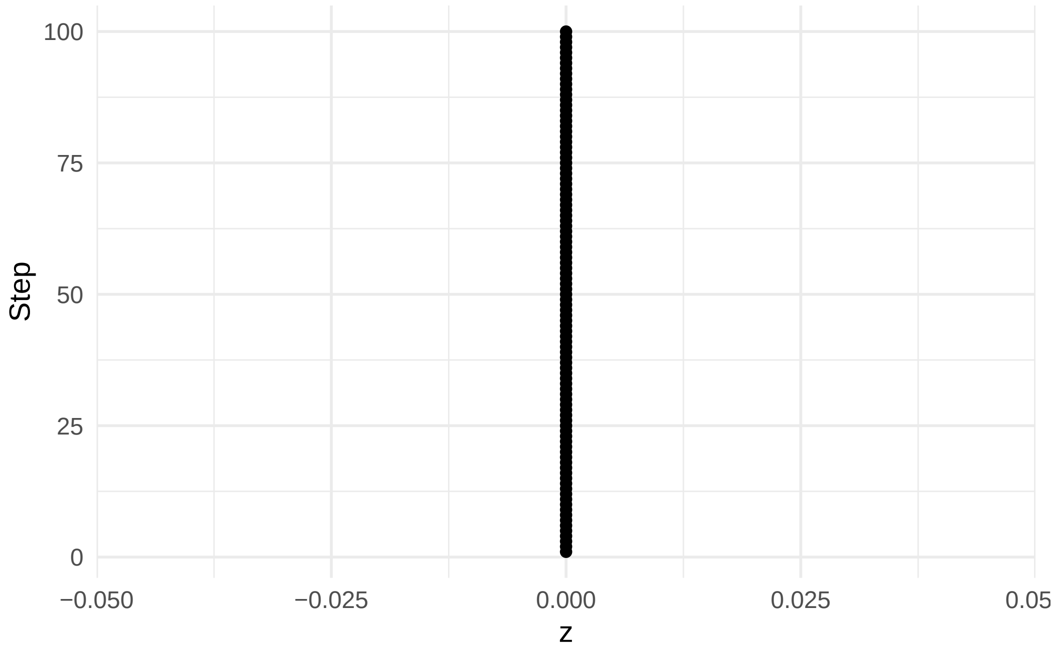


1.5 (5 points)

Redo the same analysis as **Question 1.4**, but this time using $\eta = 0.03$. What do you observe? What can you conclude from this analysis.

```r
f_z <- function(z_0){
  return_value = (z_0^4) - 6 *(z_0^2) - 3*(z_0) + 4
  return(return_value)
}
eta <- 0.03
z <- torch_tensor(-3.5, requires_grad = TRUE)

steps <- 100
gradient_vector2 <- numeric(100)
# Perform gradient descent
#for (k in 1:steps) {
  #output <- f_z(z)

  #output <- output$backward()

  #gradient <- z$grad
  #gradient<- gradient$to(device = torch_device("cpu"))

  #z <- z - eta * gradient
  #gradient_vector2[k] <- z$item()
  #z$zero_grad()
#}
```

```r
df_grad2 <- data.frame(z = gradient_vector2, k = 1:steps)
ggplot(df_grad2, aes(x = z, y = k)) +
  geom_line(stat = "function", fun = f_z, color = "blue") +
  geom_point() +
  labs(x = "z", y = "Step") +
  theme_minimal()
```



The gradient's are relatively similary in how quickly they descend to zero, but eta = .03 would be slightly faster.

Question 2

50 points

Logistic regression and interpretation of effect sizes

For this question we will use the **Titanic** dataset from the Stanford data archive. This dataset contains information about passengers aboard the Titanic and whether or not they survived.

2.1 (5 points)

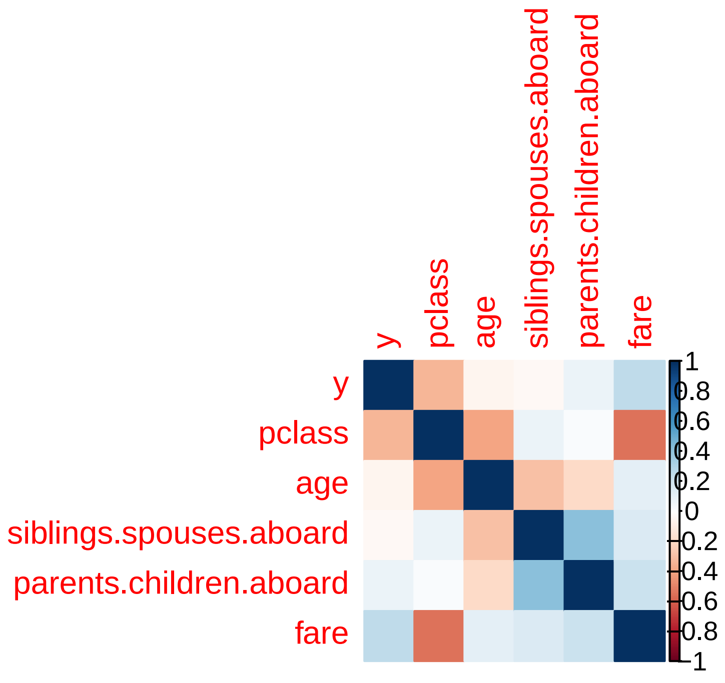
Read the data from the following URL as a tibble in R. Preprocess the data such that the variables are of the right data type, e.g., binary variables are encoded as factors, and convert all column names to lower case for consistency. Let's also rename the response variable `Survival` to `y` for convenience.

```
url <- "https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv"

df <- read.csv(url)
df_tidy <- df %>%
  mutate_if(is.logical, as.factor) %>%
  rename_all(tolower) %>%
  rename(y = survived)
```

2.2 (5 points)

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

```
numeric_df <- df_tidy[sapply(df_tidy, is.numeric)]
numeric_df %>%
  cor()%>%
  corrplot(method = "color")
```



2.3 (10 points)

Fit a logistic regression model to predict the probability of surviving the titanic as a function of:

- pclass
- sex
- age
- fare
- # siblings
- # parents

```
full_model <- glm(y ~ pclass + sex + age + fare + siblings.spouses.aboard + parents.children
summary(full_model)
```

```
Call:
glm(formula = y ~ pclass + sex + age + fare + siblings.spouses.aboard +
    parents.children.aboard, data = df_tidy)

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              1.3320583  0.0711052  18.734  < 2e-16 ***
pclass                  -0.1800338  0.0200993  -8.957  < 2e-16 ***
sexmale                 -0.5077356  0.0279609 -18.159  < 2e-16 ***
age                     -0.0061972  0.0010359  -5.983 3.19e-09 ***
fare                     0.0004034  0.0003230   1.249 0.212020
siblings.spouses.aboard -0.0502913  0.0131963  -3.811 0.000148 ***
parents.children.aboard -0.0192834  0.0180771  -1.067 0.286387
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.1438727)

    Null deviance: 210.14  on 886  degrees of freedom
Residual deviance: 126.61  on 880  degrees of freedom
AIC: 806.43

Number of Fisher Scoring iterations: 2
```

2.4 (30 points)

Provide an interpretation for the slope and intercept terms estimated in `full_model` in terms of the log-odds of survival in the titanic and in terms of the odds-ratio (if the covariate is also categorical).

Recall the definition of logistic regression from the lecture notes, and also recall how we interpreted the slope in the linear regression model (particularly when the covariate was categorical).

Interpretations:

(Intercept) 1.3320583 : The estimated log-odds of survival(y) is 1.33 when all predictors are zero, but logistic regression only allows for values y in [-1,1]. Thus, this intercept may not be of value

Pclass -0.1800338: The estimated log-odds of survival decreased by .18 for each unit increase of class when all other covariates are held constant. sexmale -0.5077356: The estimated log-odds of survival decreased by .5 when a case was male and all other variables were held constant.

age -0.0061972: The estimated log-odds of survival decreased by .006 when age increased by one unit and all other variables were held constant.

fare 1.249: The estimated log-odds of survival increased by 1.2 when there was a one unit increase in fare and all other covariates were held constant.

Siblings and Spouses aboard -3.811: The estimated log-odds of survival decreased by 3.811 when there was a one unit increase in siblings aboard and all other covariates were held constant.

Parents with children aboard -1.067 : The estimated log-odds of survival decreased by 1.067 when there was a one unit increase in parents with children aboard and all other covariates were held constant.

Question 3

70 points

Variable selection and logistic regression in `torch`

3.1 (15 points)

Complete the following function `overview` which takes in two categorical vectors (`predicted` and `expected`) and outputs:

- The prediction accuracy

- The prediction error
- The false positive rate, and
- The false negative rate

```
overview <- function(predicted, expected){
    confusion_matrix <- table(predicted,expected)
    accuracy <- (confusion_matrix[1] + confusion_matrix[4]) / length(expected) * 100 # Insert
    error <- 100 - accuracy # Insert your code here

    total_false_positives <- confusion_matrix[3]
    total_true_positives <- confusion_matrix[4]
    total_false_negatives <- confusion_matrix[2]
    total_true_negatives <- confusion_matrix[1]

    false_positive_rate <- total_false_positives / (total_false_positives + total_true_negati
    false_negative_rate <- total_false_negatives / (total_false_negatives + total_true_positi
    return(
        data.frame(
            accuracy = accuracy,
            error=error,
            false_positive_rate = false_positive_rate,
            false_negative_rate = false_negative_rate
        )
    )
}
```

You can check if your function is doing what it's supposed to do by evaluating

```
overview(df_tidy$y, df_tidy$y)
```

```
  accuracy error false_positive_rate false_negative_rate
1      100     0                   0                   0
```

and making sure that the accuracy is 100% while the errors are 0%.

3.2 (5 points)

display an overview of the key performance metrics of `full_model`

```
residual <- predict(full_model, type = "response")
residual_adjusted <- ifelse(residual < 0.5, 0, 1)
overview(residual_adjusted, df_tidy$y)
```

```
  accuracy   error false_positive_rate false_negative_rate
1  80.0451 19.9549           0.1816609           0.2330097
```

3.3 (5 points)

Using backward-stepwise logistic regression, find a parsimonious altenative to `full_model`, and print its `overview`

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

```
Start:  AIC=806.43
y ~ pclass + sex + age + fare + siblings.spouses.aboard + parents.children.aboard

                          Df Deviance     AIC
- parents.children.aboard  1   126.77  805.58
- fare                     1   126.83  806.00
<none>                         126.61  806.43
- siblings.spouses.aboard  1   128.70  818.95
- age                      1   131.76  839.79
- pclass                   1   138.15  881.82
- sex                      1   174.05 1086.71

Step:  AIC=805.58
y ~ pclass + sex + age + fare + siblings.spouses.aboard

                          Df Deviance     AIC
- fare                     1   126.94  804.76
<none>                         126.77  805.58
- siblings.spouses.aboard  1   129.60  823.15
- age                      1   131.83  838.27
- pclass                   1   138.62  882.82
- sex                      1   174.96 1089.35

Step:  AIC=804.76
y ~ pclass + sex + age + siblings.spouses.aboard

                          Df Deviance     AIC
<none>                         126.94  804.76
- siblings.spouses.aboard  1   129.60  821.17
- age                      1   132.11  838.19
- pclass                   1   145.96  926.61
- sex                      1   176.17 1093.45
```

```r
summary(step_model)
```

```
Call:
glm(formula = y ~ pclass + sex + age + siblings.spouses.aboard,
    data = df_tidy)

Coefficients:
                         Estimate Std. Error t value Pr(>|t|)
(Intercept)              1.367793   0.060360  22.661  < 2e-16 ***
pclass                  -0.193543   0.016835 -11.496  < 2e-16 ***
sexmale                 -0.504846   0.027297 -18.495  < 2e-16 ***
age                     -0.006191   0.001033  -5.995 2.96e-09 ***
siblings.spouses.aboard -0.052207   0.012138  -4.301 1.89e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.1439234)

    Null deviance: 210.14  on 886  degrees of freedom
Residual deviance: 126.94  on 882  degrees of freedom
AIC: 804.76

Number of Fisher Scoring iterations: 2
```

```r
step_predictions <- predict(step_model, type = "response")
step_predictions_adjusted <- ifelse(step_predictions < 0.5, 0, 1)
overview(step_predictions_adjusted, df_tidy$y)
```

```
  accuracy    error false_positive_rate false_negative_rate
1 79.59414 20.40586           0.1829268           0.2428115
```

3.4 (15 points)

Using the `caret` package, setup a 5-fold cross-validation training method using the `caret::trainConrol()` function

```r
controls <- trainControl(method = "cv", 5)
```

Now, using `control`, perform 5-fold cross validation using `caret::train()` to select the optimal λ parameter for LASSO with logistic regression.

Take the search grid for λ to be in $\{2^{-20}, 2^{-19.5}, 2^{-19}, \dots, 2^{-0.5}, 2^{0}\}$.

```r
library(glmnet)
```

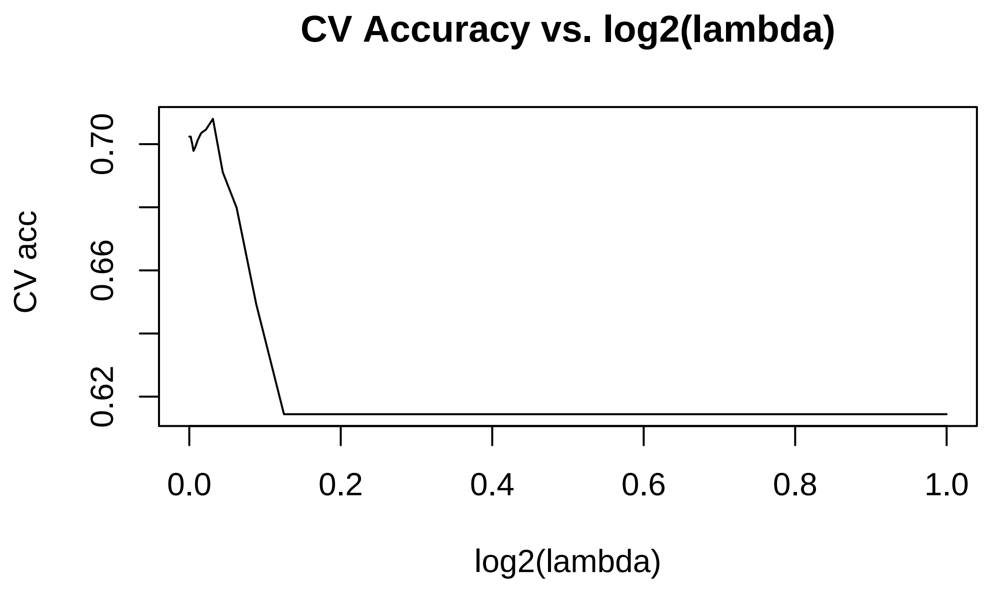
```
Loading required package: Matrix
```

```

Attaching package: 'Matrix'
```

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

    expand, pack, unpack
```

```
Loaded glmnet 4.1-8
```

```r
df_filtered <- df_tidy %>%
  select(-which(sapply(df_tidy,is.character)))
# Insert your code in the ... region
lasso_fit <- train(
  x = df_filtered[, -which(names(df_tidy) == "y")],
  y = as.factor(df_tidy$y),
  method = "glmnet",
  trControl = controls,
  tuneGrid = expand.grid(
    alpha = 1,
    lambda = 2^seq(-20, 0, by = 0.5)
    ),
  family = "binomial"
)

plot(lasso_fit$results$lambda, lasso_fit$results$Accuracy, type = "l",
     xlab = "log2(lambda)", ylab = " CV acc",
     main = "CV Accuracy vs. log2(lambda)")
```



```r
optimal_lambda <- lasso_fit$results$lambda[which.max(lasso_fit$results$Accuracy)]

cat("Lambda", optimal_lambda)
```

```
Lambda 0.03125
```

Using the information stored in `lasso_fit$results`, plot the results for cross-validation accuracy vs. $log_2(\lambda)$. Choose the optimal λ^*, and report your results for this value of λ^*.

3.5 (25 points)

First, use the `model.matrix()` function to convert the covariates of `df` to a matrix format

```r
covariate_matrix <- model.matrix(full_model)[, -1]
```

Now, initialize the covariates X and the response y as `torch` tensors

```r
X <- torch_tensor(covariate_matrix, dtype = torch_float())
y <- torch_tensor(df_tidy$y, dtype = torch_float())
```

Using the `torch` library, initialize an `nn_module` which performs logistic regression for this dataset. (Remember that we have 6 different covariates)

```
logistic <- nn_module(
  initialize = function() {
    self$f <- nn_linear(6, 1)
    self$g <- nn_sigmoid()
  },
  forward = function(x) {
    x %>%
      self$f() %>%
      self$g()

  }
)

f <- logistic()
```

You can verify that your code is right by checking that the output to the following code is a vector of probabilities:

```
head(f(X))
```

```
torch_tensor
 0.9985
 0.7784
 0.9996
 0.9569
 1.0000
 0.9996
[ CPUFloatType{6,1} ][ grad_fn = <SliceBackward0> ]
```

Now, define the loss function `Loss()` which takes in two tensors `X` and `y` and a function `Fun`, and outputs the **Binary cross Entropy loss** between `Fun(X)` and `y`.

```
Loss <- function(X, y, model){
   nn_bce_loss()(model(X), y)
}
f <- logistic()
Loss(X, y, f)
```

```
torch_tensor
3.15937
[ CPUFloatType{} ][ grad_fn = <BinaryCrossEntropyBackward0> ]
```

Initialize an optimizer using `optim_adam()` and perform $n = 1000$ steps of gradient descent in order to fit logistic regression using `torch`.

```r
f <- logistic()
optimizer <- optim_adam(f$parameters, lr = .01) # Insert your code here

n <- 1000
for(k in 1: n){
  loss <- Loss(X,y, f)
  optimizer$zero_grad()
  loss$backward()
  optimizer$step()

  if (k %% 20 == 0) {
        cat(sprintf("Epoch: %d, Loss: %.4f\n", k, loss$item()))
    }




}
```

```
Epoch: 20, Loss: 1.1967
Epoch: 40, Loss: 0.6569
Epoch: 60, Loss: 0.6377
Epoch: 80, Loss: 0.6055
Epoch: 100, Loss: 0.5921
Epoch: 120, Loss: 0.5819
Epoch: 140, Loss: 0.5734
Epoch: 160, Loss: 0.5661
Epoch: 180, Loss: 0.5596
Epoch: 200, Loss: 0.5536
Epoch: 220, Loss: 0.5481
Epoch: 240, Loss: 0.5431
Epoch: 260, Loss: 0.5385
Epoch: 280, Loss: 0.5342
Epoch: 300, Loss: 0.5303
Epoch: 320, Loss: 0.5266
Epoch: 340, Loss: 0.5233
Epoch: 360, Loss: 0.5201
Epoch: 380, Loss: 0.5172
Epoch: 400, Loss: 0.5146
Epoch: 420, Loss: 0.5121
```

```
Epoch: 440, Loss: 0.5098
Epoch: 460, Loss: 0.5076
Epoch: 480, Loss: 0.5056
Epoch: 500, Loss: 0.5038
Epoch: 520, Loss: 0.5020
Epoch: 540, Loss: 0.5004
Epoch: 560, Loss: 0.4988
Epoch: 580, Loss: 0.4974
Epoch: 600, Loss: 0.4960
Epoch: 620, Loss: 0.4947
Epoch: 640, Loss: 0.4935
Epoch: 660, Loss: 0.4923
Epoch: 680, Loss: 0.4911
Epoch: 700, Loss: 0.4900
Epoch: 720, Loss: 0.4890
Epoch: 740, Loss: 0.4879
Epoch: 760, Loss: 0.4869
Epoch: 780, Loss: 0.4859
Epoch: 800, Loss: 0.4850
Epoch: 820, Loss: 0.4840
Epoch: 840, Loss: 0.4831
Epoch: 860, Loss: 0.4822
Epoch: 880, Loss: 0.4813
Epoch: 900, Loss: 0.4804
Epoch: 920, Loss: 0.4795
Epoch: 940, Loss: 0.4786
Epoch: 960, Loss: 0.4778
Epoch: 980, Loss: 0.4769
Epoch: 1000, Loss: 0.4761
```

Using the final, optimized parameters of `f`, compute the compute the predicted results on `X`

```r
predicted_probabilities <- f(X) %>% as_array()
torch_predictions <- ifelse(predicted_probabilities <  0.5, 0, 1)

overview(torch_predictions, df_tidy$y)
```

```
  accuracy    error false_positive_rate false_negative_rate
1 79.36866 20.63134           0.1890034           0.2393443
```

3.6 (5 points)

Create a summary table of the `overview()` summary statistics for each of the 4 models we have looked at in this assignment, and comment on their relative strengths and drawbacks.

Session Information

Print your `R` session information using the following command

```
sessionInfo()
```

```
R version 4.3.1 (2023-06-16 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 datasets  utils     methods   base

other attached packages:
 [1] glmnet_4.1-8   Matrix_1.5-4.1 broom_1.0.5    nnet_7.3-19    torch_0.12.0
 [6] caret_6.0-94   lattice_0.21-8 ggplot2_3.5.0  car_3.1-2      carData_3.0-5
[11] corrplot_0.92  stringr_1.5.1  purrr_1.0.2    tidyr_1.3.1    readr_2.1.5
[16] dplyr_1.1.4

loaded via a namespace (and not attached):
 [1] tidyselect_1.2.1     timeDate_4032.109    farver_2.1.1
 [4] fastmap_1.1.1        pROC_1.18.5          digest_0.6.35
 [7] rpart_4.1.19         timechange_0.3.0     lifecycle_1.0.4
[10] survival_3.5-5       processx_3.8.4       magrittr_2.0.3
[13] compiler_4.3.1       rlang_1.1.3          tools_4.3.1
[16] utf8_1.2.4           yaml_2.3.8           data.table_1.15.2
[19] knitr_1.45           labeling_0.4.3       bit_4.0.5
[22] plyr_1.8.9           abind_1.4-5          withr_3.0.0
[25] grid_4.3.1           stats4_4.3.1         fansi_1.0.6
```

```
[28] e1071_1.7-14         colorspace_2.1-0     future_1.33.1
[31] globals_0.16.3       scales_1.3.0         iterators_1.0.14
[34] MASS_7.3-60          cli_3.6.2            rmarkdown_2.26
[37] generics_0.1.3       future.apply_1.11.1  reshape2_1.4.4
[40] tzdb_0.4.0           proxy_0.4-27         splines_4.3.1
[43] parallel_4.3.1       coro_1.0.4           vctrs_0.6.5
[46] hardhat_1.3.1        jsonlite_1.8.8       callr_3.7.5
[49] hms_1.1.3            bit64_4.0.5          listenv_0.9.1
[52] foreach_1.5.2        gower_1.0.1          recipes_1.0.10
[55] glue_1.7.0           parallelly_1.37.1    ps_1.7.6
[58] codetools_0.2-19     shape_1.4.6.1        lubridate_1.9.3
[61] stringi_1.8.3        gtable_0.3.4         munsell_0.5.0
[64] tibble_3.2.1         pillar_1.9.0         htmltools_0.5.7
[67] ipred_0.9-14         lava_1.8.0           R6_2.5.1
[70] evaluate_0.23        backports_1.4.1      renv_1.0.3
[73] class_7.3-22         Rcpp_1.0.12          nlme_3.1-162
[76] prodlim_2023.08.28   xfun_0.42            pkgconfig_2.0.3
[79] ModelMetrics_1.2.2.2
```