

Homework 4

Brady Miller

Table of contents

Question 1	4
Question 2	11
Question 3	15

[Link to the Github repository](#)

! Due: Sun, Apr 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

We will be using the following libraries:

```
packages <- c(
  "dplyr",
  "readr",
  "tidyr",
  "purrr",
  "stringr",
  "corrplot",
```

```
"car",  
"caret",  
"torch",  
"nnet",  
"broom"  
)  
  
renv::install(packages)
```

```
Installing dplyr [1.1.1] ...  
  OK [linked cache in 4.2 milliseconds]  
Installing readr [2.1.4] ...  
  OK [linked cache in 4.7 milliseconds]  
Installing purrr [1.0.1] ...  
  OK [linked cache in 4.3 milliseconds]  
Installing stringr [1.5.0] ...  
  OK [linked cache in 4.7 milliseconds]  
Installing tidyr [1.3.0] ...  
  OK [linked cache in 5.5 milliseconds]  
Installing corrplot [0.92] ...  
  OK [linked cache in 3.4 milliseconds]  
Installing nnet [7.3-18] ...  
  OK [linked cache in 4.6 milliseconds]  
Installing broom [1.0.4] ...  
  OK [linked cache in 6 milliseconds]  
Installing car [3.1-2] ...  
  OK [linked cache in 3.7 milliseconds]  
Installing caret [6.0-94] ...  
  OK [linked cache in 3.3 milliseconds]  
Installing torch [0.9.1] ...  
  OK [linked cache in 4.1 milliseconds]
```

```
sapply(packages, require, character.only=T)
```

Loading required package: 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

Loading required package: readr

Loading required package: tidyr

Loading required package: purrr

Loading required package: stringr

Loading required package: corrplot

corrplot 0.92 loaded

Loading required package: 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

Loading required package: caret

Loading required package: ggplot2

Loading required package: lattice

Attaching package: 'caret'

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

lift

Loading required package: torch

Loading required package: nnet

Loading required package: broom

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

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

$$\frac{d}{dx}g(x, y) = 2(x - 3), \quad \text{and} \quad \frac{d}{dy}g(x, y) = 2(y - 4).$$

Using your answer from above, what is the answer to

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

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?

```
library(numDeriv)
g <- function(x) {
  (x[1] - 3)^2 + (x[2] - 4)^2
}
grad(g, c(3,4))
```

```
[1] 0 0
```

The answer I got from the computed gradient in R is (0,0), which matches the values I got when I did the partial derivatives of the equation, so the answer I got is what I expected.

1.2 (10 points)

```
# command not working so commenting out in order to load PDF
$$$$\newcommand{\u}{\boldsymbol{u}}\newcommand{\v}{\boldsymbol{v}}$$$$
```

Consider $h(\cdot)$ given by

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

where \cdot 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

```
# command also not working for pdf
$$$$
#\begin{aligned}
#\frac{d}{du}\nabla_u h(\u, \v) &= \Bigg(\frac{d}{du_1}h(\u, \v), \frac{d}{du_2}h(\u, \v)
#\end{aligned}
$$$$
```

```
# command not working for pdf
$$$$
#\begin{aligned}
#\frac{d}{du}\nabla_u h(\u, \v) &= \Bigg(3(\u \cdot \v)^2 \times v_1, 3(\u \cdot \v)^2 \times v_2
#\end{aligned}
$$$$
```

Using your answer from above, what is the answer to $\frac{d}{du}h(\u, \v)$ when $n = 10$ and

```

# command not working as desired so commenting it out
###
#\begin{aligned}
#\u = (-1, +1, -1, +1, -1, +1, -1, +1, -1, +1)\\
#\v = (-1, -1, -1, -1, -1, +1, +1, +1, +1, +1)
#\end{aligned}
###

```

Define $h(\cdot)$ as a function in R, initialize the two vectors \tilde{u} and \tilde{v} as `torch_tensors`. Compute the gradient of $h(\cdot)$ with respect to \tilde{u} . Does the answer match what you expected?

```

h <- function(u,v){
  (torch_dot(u,v))^3
}

u <- torch_tensor(c(-1,1,-1,1,-1,1,-1,1,-1,1), requires_grad = TRUE)
v <- torch_tensor(c(-1,-1,-1,-1,-1,1,1,1,1,1), requires_grad = TRUE)

result <- h(u,v)

result$backward()

u$grad

```

```

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

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

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

$$f'(z_0) = \left. \frac{df}{dz} \right|_{z_0=-3.5} = 4(-3.5)^3 - 12(-3.5) - 3 = -132.5$$

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

```
func <- function(z){  
  (z^4) - 6*(z)^2 - 3*z + 4  
}  
z_0 <- torch_tensor(-3.5, requires_grad = TRUE)  
  
output <- func(z_0)  
  
output$backward()  
z_0$grad
```

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

The value I got using the `torch` library for $f'(-3.5)$ was -132.5, which matches the value I got when I calculated $f'(-3.5)$ by hand

1.4 (5 points)

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

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

```

# initializing parameters to be used for the function
z <- -3.5
n <- 100
eta <- 0.02
z_values <- c(z)

for (i in 1:n) {
  # updates derivative each iteration & finds new z value, adding it to array
  deriv <- 4*z^3 - 12*z - 3
  z <- z - eta * deriv
  z_values <- c(z_values, z)
}

```

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

```

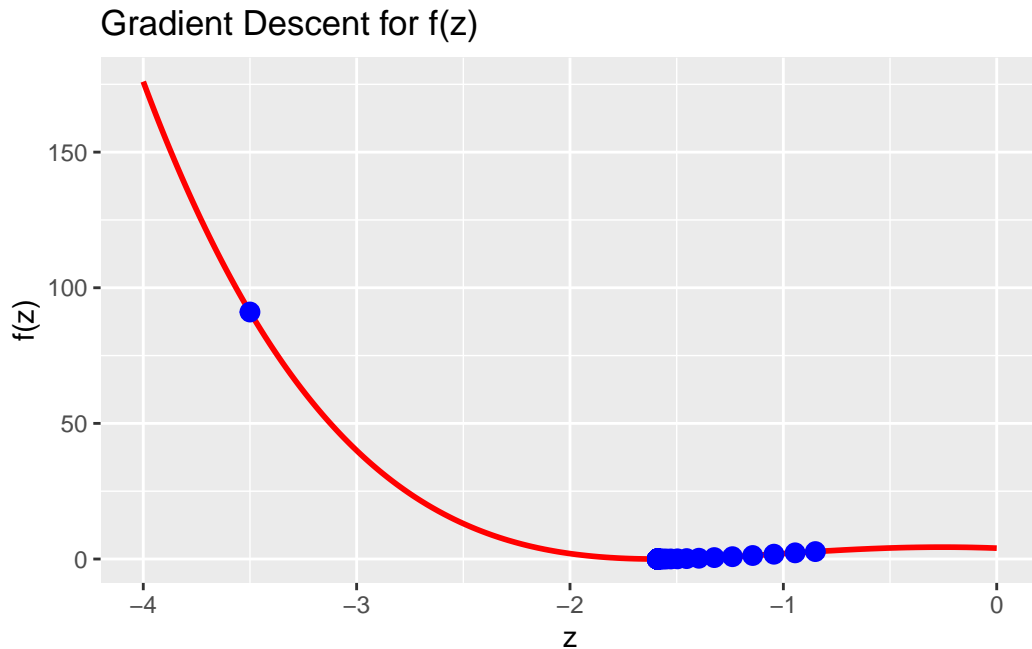
# creating a x-axis boundary
x_values <- seq(-4,0, by = 0.01)
# finding y-values for those x values (finding f(x) for function f above)
y_values <- func(x_values)

# creating data frames to store the function values and the gradient desc values
df_f <- data.frame(x = x_values, y = y_values)
df_z <- data.frame(x = z_values, y = func(z_values))

ggplot() +
  # red line showing the actual curve of the line
  geom_line(data = df_f, aes(x=x, y=y), color = 'red', size = 1) +
  # blue points showing the value after each iteration of gradient descent
  geom_point(data = df_z, aes(x=x, y=y), color = 'blue', size = 3) +
  xlab('z') +
  ylab('f(z)') +
  ggtitle('Gradient Descent for f(z)')

```

Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
 i Please use `linewidth` instead.



I observe that there is a quick decrease in the gradient descent after the first iteration. After that though, that it moves closer and closer to the local minimum, so the ‘gap’ or space from one iteration to the next gets smaller and smaller. This value only actually finds the local minimum of the function as after that first iteration, gradient descent ‘tells’ it to go left as it has a positive slope at that spot. Based on the fact that it started with a negative slope and ended up with a positive slope after the first iteration, it knows that there is some point in between those 2 points that has a slope of 0 (is a local minimum). Based on the gradient descent performed, I can conclude that the local minimum is at around a z value of -1.6 and has a $f(z)$ value of 0.

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

```
# initializing parameters to be used for the function
z <- -3.5
n <- 100
eta <- 0.03
zvals <- c(z)
```

```

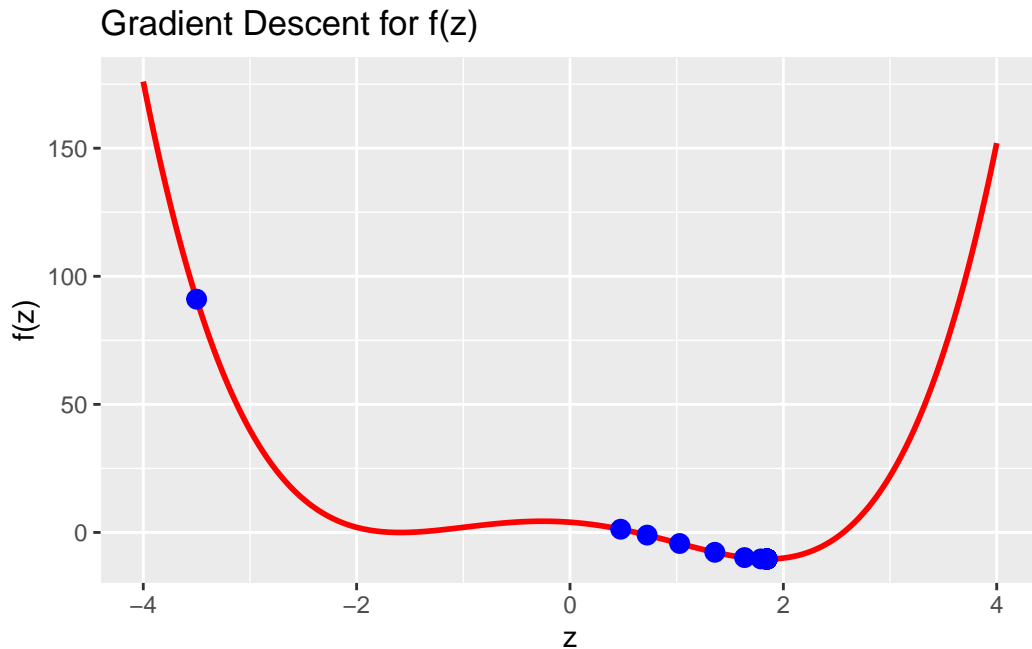
for (i in 1:n) {
  # updates derivative each iteration & finds new z value, adding it to array
  deriv <- 4*z^3 - 12*z - 3
  z <- z - eta * deriv
  zvals <- c(zvals, z)
}

# creating a x-axis boundary
xvals <- seq(-4,4, by = 0.01)
# finding y-values for those x values (finding f(x) for function f above)
yvals <- func(xvals)

# creating data frames to store the function values and the gradient desc values
df_f <- data.frame(x = xvals, y = yvals)
df_z <- data.frame(x = zvals, y = func(zvals))

ggplot() +
  # red line showing the actual curve of the line
  geom_line(data = df_f, aes(x=x, y=y), color = 'red', size = 1) +
  # blue points showing the value after each iteration of gradient descent
  geom_point(data = df_z, aes(x=x, y=y), color = 'blue', size = 3) +
  xlab('z') +
  ylab('f(z)') +
  ggtitle('Gradient Descent for f(z)')

```



When changing the value of eta from 0.02 to 0.03, we are then able to find the global minimum of the function. This is different from the last use of gradient descent with eta=0.02 because after the first iteration, there is still a negative slope, so it keeps moving to the right to see if there is a minimum. The distance between the z values gets smaller and smaller with each iteration of gradient descent. It converges to a global minimum at a z value of about 1.9, with a $f(z)$ value of about -10.

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)

# changing column name of 'Survived'
colnames(df)[colnames(df) == 'Survived'] <- 'y'

# getting rid of . in the column names
colnames(df)[colnames(df)=='Siblings.Spouses.Aboard']<-'Siblings Spouses Aboard'
colnames(df)[colnames(df)=='Parents.Children.Aboard']<-'Parents Children Aboard'

# converting all the column names to lower case
colnames(df) <- tolower(colnames(df))

# changing the y (survived) and sex to factors
df$y <- as.factor(df$y)
df$sex <- as.factor(ifelse(df$sex == 'male', 1, 0))

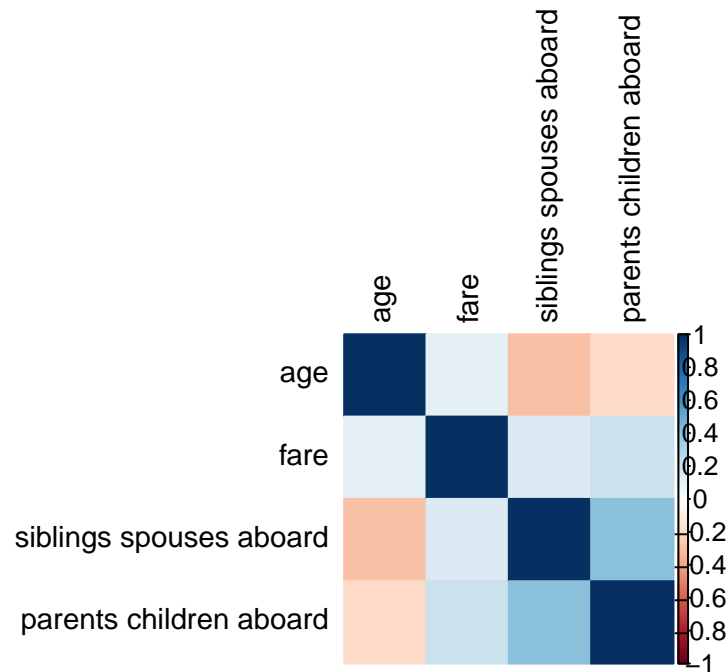
# converting pclass to factor too even though not a binary variable
# makes more sense when interpreting slope
df$pclass <- as.factor(df$pclass)
```

2.2 (5 points)

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

```
# keeping only numeric values
dfCorr <- df %>%
  keep(is.numeric) %>%
  cor()

# creating corrplot
corrplot(dfCorr, method = 'color', tl.cex = 0.9, tl.col = 'black',
  order = 'hclust')
```



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 + `siblings spouses aboard` +
                  `parents children aboard` + fare, data = df,
                  family = binomial())
summary(full_model)
```

Call:

```
glm(formula = y ~ pclass + sex + age + `siblings spouses aboard` +
    `parents children aboard` + fare, family = binomial(), data = df)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.7773	-0.5991	-0.3984	0.6131	2.4412

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	4.109777	0.463602	8.865	< 2e-16 ***
pclass2	-1.161491	0.300960	-3.859	0.000114 ***
pclass3	-2.350022	0.304666	-7.713	1.22e-14 ***
sex1	-2.756710	0.200642	-13.739	< 2e-16 ***
age	-0.043410	0.007790	-5.573	2.51e-08 ***
`siblings spouses aboard`	-0.401572	0.110795	-3.624	0.000290 ***
`parents children aboard`	-0.106884	0.118767	-0.900	0.368151
fare	0.002823	0.002468	1.144	0.252771

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1182.77 on 886 degrees of freedom
Residual deviance: 780.93 on 879 degrees of freedom
AIC: 796.93

Number of Fisher Scoring iterations: 5

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).

- The intercept term of 4.109777 means that if all the covariates are held constant, using log-odds, we can determine that the odds of survival is $\exp(4.109777)$ or about 60.93 times greater than the odds of not surviving when all the other variables are held constant

- The next covariates is pclass, which is categorical. Since its a categorical variable, we had to set a baseline value for this covariates, which in this case, we used pclass of 1 as the baseline, so the slopes for pclass2 and pclass3 are based off pclass1. For pclass2, the slope value is -1.161491, so plugging that into log-odds we get that the odds of survival is about 0.313 times greater for people of class 2 compared to class 1 (higher chance of surviving at pclass1 than pclass2). For pclass3, the slope value is -2.35, so the odds of survival for someone in class 3 is 0.095 times greater than someone in class 1, so someone in class 1 has a much better chance of survival than someone in class 3.
- The next covariate is sex, which is categorical. The baseline value for this covariate is female (as female is labeled as 0, and male 1 in the data frame). Using this, and the slope of sex1 being -2.7567, a male is 0.0635 times more likely to survive than a female, which makes sense as women were given a spot in the lifeboats first.
- The age covariate has a slope value of -0.043410. From this, we can determine that an increase in age by 1, while holding all other covariates constant, the odds of survival is about 0.95 times as likely. So, as age increases, the odds of survival go down slightly.
- For 'siblings spouses aboard', the slope value of -0.401572 indicates that for an increase in siblings spouses aboard by 1 (while holding other covariates constant), the odds of surviving is 0.669 times as likely, so as this value increases, your survival odds goes down.
- For 'parents children aboard', the slope value of -0.106884 indicates that for an increase in parents children aboard by 1 (while holding other covariates constant), the odds of surviving is about 0.9 times as likely, so once again, as the amount of parents children aboard goes up for an individual, the survival odds goes down.
- For the fare covariate, the slope value is 0.002823, so as the fare increases by 1, while holding other covariates constant, the odds of survival is about 0.99 times as likely, so as fare increases, an individuals odds of survival decrease slightly.

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){
  total_false_positives <- sum(expected != predicted & expected == 0)
  total_true_positives <- sum(expected == predicted & expected == 1)
  total_false_negatives <- sum(expected != predicted & expected == 1)
  total_true_negatives <- sum(expected == predicted & expected == 0)
  false_positive_rate <- total_false_positives / (total_false_positives +
                                                    total_true_negatives)
  false_negative_rate <- total_false_negatives / (total_false_negatives +
                                                    total_true_positives)
  accuracy <- (total_true_positives + total_true_negatives) / length(expected)
  error <- (total_false_positives + total_false_negatives) / length(expected)
  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$y, df$y)
```

	accuracy	error	false_positive_rate	false_negative_rate
1	1	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`

```
# getting predictions on the survival of Titanic passengers
predictions <- predict(full_model, type = 'response')

predictions <- ifelse(predictions >= 0.5, 1, 0)

# getting expected value for survival (survived = 1, died = 0)
expected <- df$y

# using overview func we created to see key performance metrics of full_model
fullModel_overview <- overview(predictions, expected)
fullModel_overview
```

	accuracy	error	false_positive_rate	false_negative_rate
1	0.8027057	0.1972943	0.1321101	0.3011696

3.3 (5 points)

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

```
# creating null model which backwards stepwise logistic regression will end at
null_model <- glm(y ~ 1, df, family = binomial())

# creating the backwards stepwise logistic regression model
step_model <- step(full_model, direction = 'backward',
                  scope = formula(null_model))
```

Start: AIC=796.93

`y ~ pclass + sex + age + `siblings spouses aboard` + `parents children aboard` + fare`

	Df	Deviance	AIC
- `parents children aboard`	1	781.75	795.75
- fare	1	782.37	796.37
<none>		780.93	796.93
- `siblings spouses aboard`	1	796.79	810.79

- age	1	815.20	829.20
- pclass	2	847.84	859.84
- sex	1	1020.26	1034.26

Step: AIC=795.75

y ~ pclass + sex + age + `siblings spouses aboard` + fare

	Df	Deviance	AIC
- fare	1	782.82	794.82
<none>		781.75	795.75
- `siblings spouses aboard`	1	801.56	813.56
- age	1	815.88	827.88
- pclass	2	852.19	862.19
- sex	1	1024.08	1036.08

Step: AIC=794.82

y ~ pclass + sex + age + `siblings spouses aboard`

	Df	Deviance	AIC
<none>		782.82	794.82
- `siblings spouses aboard`	1	801.59	811.59
- age	1	818.25	828.25
- pclass	2	900.80	908.80
- sex	1	1031.69	1041.69

```
summary(step_model)
```

Call:

```
glm(formula = y ~ pclass + sex + age + `siblings spouses aboard`,
     family = binomial(), data = df)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-2.7637	-0.5883	-0.3930	0.6136	2.4543

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	4.294169	0.417879	10.276	< 2e-16 ***
pclass2	-1.321703	0.268452	-4.923	8.5e-07 ***
pclass3	-2.541237	0.258324	-9.837	< 2e-16 ***
sex1	-2.738024	0.195796	-13.984	< 2e-16 ***

```
age -0.043918 0.007757 -5.662 1.5e-08 ***
`siblings spouses aboard` -0.409624 0.105495 -3.883 0.000103 ***
---
```

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

```
Null deviance: 1182.77 on 886 degrees of freedom
Residual deviance: 782.82 on 881 degrees of freedom
AIC: 794.82
```

Number of Fisher Scoring iterations: 5

```
# creating predictions based on the backward stepwise logistic regression model
step_predictions <- predict(step_model, type = 'response')

# ifelse statement to classify the predictions since they are currently decimals
step_predictions <- ifelse(step_predictions >= 0.5, 1, 0)

stepwise_overview <- overview(step_predictions, df$y)
stepwise_overview
```

```
accuracy error false_positive_rate false_negative_rate
1 0.8049605 0.1950395 0.133945 0.2923977
```

3.4 (15 points)

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

```
controls <- trainControl(method = 'cv', number = 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 $\{2^{-20}, 2^{-19.5}, 2^{-19}, \dots, 2^{-0.5}, 2^0\}$.

```
lasso_fit <- train(
  y ~ pclass + sex + age + `siblings spouses aboard` +
```

```

                                `parents children aboard` + fare,
data = df,
method = 'glmnet',
trControl = controls,
tuneGrid = expand.grid(
  alpha = 1,
  lambda = 2^seq(-20, 0, by = 0.5)
),
standardize = TRUE,
family = 'binomial'
)

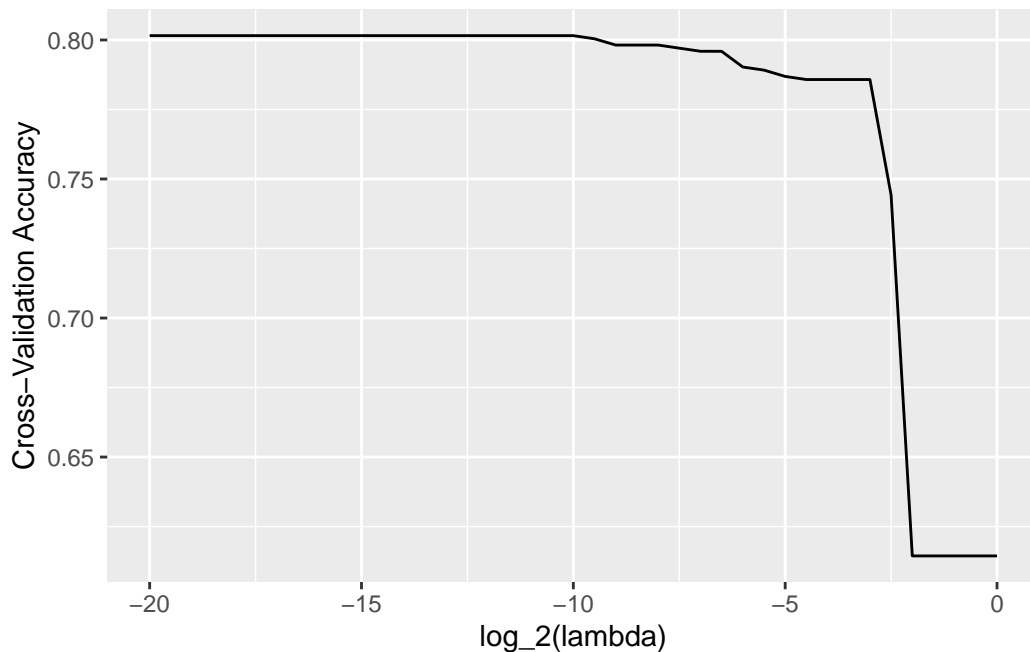
```

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^* .

```

# plotting the log2(lambda) values vs accuracy to find optimal lambda
lasso_fit$results %>%
  ggplot() +
  geom_line(aes(x = log2(lambda), y = Accuracy)) +
  xlab('log_2(lambda)') +
  ylab('Cross-Validation Accuracy')

```



```
# showing where max accuracy is & the associated lambda (the optimal lambda)
lasso_fit$results %>%
  filter(Accuracy == max(lasso_fit$results$Accuracy))
```

	alpha	lambda	Accuracy	Kappa	AccuracySD	KappaSD
1	1	9.536743e-07	0.8015489	0.5736119	0.02083189	0.04536339
2	1	1.348699e-06	0.8015489	0.5736119	0.02083189	0.04536339
3	1	1.907349e-06	0.8015489	0.5736119	0.02083189	0.04536339
4	1	2.697398e-06	0.8015489	0.5736119	0.02083189	0.04536339
5	1	3.814697e-06	0.8015489	0.5736119	0.02083189	0.04536339
6	1	5.394797e-06	0.8015489	0.5736119	0.02083189	0.04536339
7	1	7.629395e-06	0.8015489	0.5736119	0.02083189	0.04536339
8	1	1.078959e-05	0.8015489	0.5736119	0.02083189	0.04536339
9	1	1.525879e-05	0.8015489	0.5736119	0.02083189	0.04536339
10	1	2.157919e-05	0.8015489	0.5736119	0.02083189	0.04536339
11	1	3.051758e-05	0.8015489	0.5736119	0.02083189	0.04536339
12	1	4.315837e-05	0.8015489	0.5736119	0.02083189	0.04536339
13	1	6.103516e-05	0.8015489	0.5736119	0.02083189	0.04536339
14	1	8.631675e-05	0.8015489	0.5736119	0.02083189	0.04536339
15	1	1.220703e-04	0.8015489	0.5736119	0.02083189	0.04536339
16	1	1.726335e-04	0.8015489	0.5736119	0.02083189	0.04536339
17	1	2.441406e-04	0.8015489	0.5736119	0.02083189	0.04536339
18	1	3.452670e-04	0.8015489	0.5736119	0.02083189	0.04536339
19	1	4.882812e-04	0.8015489	0.5736119	0.02083189	0.04536339
20	1	6.905340e-04	0.8015489	0.5736119	0.02083189	0.04536339
21	1	9.765625e-04	0.8015489	0.5736119	0.02083189	0.04536339

Based on this plot, the value of $\log_2(\lambda)$ that gives the highest cross-validation accuracy is around -8. The table created, gives the row of data for the highest accuracy value in the `lasso_fit$results` table. Based on this we can see that the highest accuracy is about 79% and comes from a λ value of 0.00390625. This checks out with the graph since $\log_2(0.00390625)$ is about -8, so the optimal λ is 0.00390625.

```
# creating predictions based on lambda model
lasso_predictions <- predict(lasso_fit)

lasso_overview <- overview(lasso_predictions ,df$y)
lasso_overview
```

	accuracy	error	false_positive_rate	false_negative_rate
1	0.8015784	0.1984216	0.133945	0.3011696

3.5 (25 points)

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

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

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

```
X <- torch_tensor(covariate_matrix, dtype = torch_float())
y <- torch_tensor(df$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() {
    # need 7 in the linear layer because pclass is being used as a factor
    self$f <- nn_linear(7,1)
    self$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:

```
f(X)
```

```
torch_tensor
0.9993
1.0000
0.9999
1.0000
1.0000
```

```

0.9999
1.0000
0.9976
0.9999
1.0000
0.9970
1.0000
0.9992
1.0000
0.9974
1.0000
0.9998
0.9999
1.0000
0.9995
1.0000
1.0000
0.9981
1.0000
0.9996
1.0000
0.9998
1.0000
0.9998
0.9996
... [the output was truncated (use n=-1 to disable)]
[ CPUFloatType{887,1} ][ grad_fn = <SigmoidBackward0> ]

```

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, Fun){
  nn_bce_loss()(Fun(X), y)
}

```

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

```

f <- logistic()
optimizer <- optim_adam(f$parameters, lr = 0.01)

n <- 1000

```

```

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

  if (i %% 100 == 0) {
    cat(sprintf('Epoch: %d, Loss: %.6f\n', i, loss$item()))
  }
}

```

```

Epoch: 100, Loss: -27.801371
Epoch: 200, Loss: -32.865242
Epoch: 300, Loss: -35.233173
Epoch: 400, Loss: -36.509529
Epoch: 500, Loss: -36.911011
Epoch: 600, Loss: -37.583931
Epoch: 700, Loss: -37.684189
Epoch: 800, Loss: -37.878559
Epoch: 900, Loss: -37.976418
Epoch: 1000, Loss: -38.073086

```

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

```

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

torch_overview <- overview(torch_predictions, df$y)
torch_overview

```

	accuracy	error	false_positive_rate	false_negative_rate
1	0.3855693	0.6144307	1	0

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.


```
summary_table <- rbind(fullModel_overview, stepwise_overview, lasso_overview,
                        torch_overview) %>%
  mutate(Model = c('Full Model', 'Stepwise Regression', 'Lasso', 'Torch'))

summary_table <- summary_table[,c('Model', 'accuracy', 'error', 'false_positive_rate', 'false_negative_rate')]

summary_table
```

	Model	accuracy	error	false_positive_rate	false_negative_rate
1	Full Model	0.8027057	0.1972943	0.1321101	
2	Stepwise Regression	0.8049605	0.1950395	0.1339450	
3	Lasso	0.8015784	0.1984216	0.1339450	
4	Torch	0.3855693	0.6144307	1.0000000	
1					0.3011696
2					0.2923977
3					0.3011696
4					0.0000000

The overviews from the full model, stepwise regression formula and lasso regression are nearly identical in terms of their accuracy (~80%), error(~20%), false positive rate (~13%) and false negative rate (ranging from 29%-31%). So these models are decently accurate and what might be considered a good false positive rate, but it has a drawback of incorrectly predicting people to die when they actually survived (`false_negative_rate`). The torch model fared much worse in all performance metrics except false negative rate. The torch model is only predicting the survival of a passenger about 38% of the time, and 100% of the time it is incorrectly predicting that someone survived when they actually died. The only strength of the torch model is that it never predicted someone to die when they actually survived.

Session Information

Print your R session information using the following command

```
sessionInfo()
```

R version 4.2.3 (2023-03-15 ucrt)

Platform: x86_64-w64-mingw32/x64 (64-bit)

Running under: Windows 10 x64 (build 22621)

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] numDeriv_2016.8-1.1 broom_1.0.4 nnet_7.3-18

[4] torch_0.9.1 caret_6.0-94 lattice_0.20-45

[7] ggplot2_3.4.1 car_3.1-2 carData_3.0-5

[10] corrplot_0.92 stringr_1.5.0 purrr_1.0.1

[13] tidyr_1.3.0 readr_2.1.4 dplyr_1.1.1

loaded via a namespace (and not attached):

[1] nlme_3.1-162 lubridate_1.9.2 bit64_4.0.5

[4] tools_4.2.3 backports_1.4.1 utf8_1.2.3

[7] R6_2.5.1 rpart_4.1.19 colorspace_2.1-0

[10] withr_2.5.0 tidyselect_1.2.0 processx_3.8.0

[13] bit_4.0.5 compiler_4.2.3 glmnet_4.1-7

[16] cli_3.6.0 labeling_0.4.2 scales_1.2.1

[19] proxy_0.4-27 callr_3.7.3 digest_0.6.31

[22] rmarkdown_2.20 coro_1.0.3 pkgconfig_2.0.3

[25] htmltools_0.5.4 parallelly_1.35.0 fastmap_1.1.0

[28] rlang_1.1.0 rstudioapi_0.14 shape_1.4.6

[31]	farver_2.1.1	generics_0.1.3	jsonlite_1.8.4
[34]	ModelMetrics_1.2.2.2	magrittr_2.0.3	Matrix_1.5-3
[37]	Rcpp_1.0.10	munsell_0.5.0	fansi_1.0.4
[40]	abind_1.4-5	lifecycle_1.0.3	stringi_1.7.12
[43]	pROC_1.18.0	yaml_2.3.7	MASS_7.3-58.3
[46]	plyr_1.8.8	recipes_1.0.5	grid_4.2.3
[49]	parallel_4.2.3	listenv_0.9.0	splines_4.2.3
[52]	hms_1.1.2	knitr_1.42	ps_1.7.2
[55]	pillar_1.8.1	future.apply_1.10.0	reshape2_1.4.4
[58]	codetools_0.2-19	stats4_4.2.3	glue_1.6.2
[61]	evaluate_0.20	data.table_1.14.8	renv_0.16.0-53
[64]	vctrs_0.6.1	tzdb_0.3.0	foreach_1.5.2
[67]	gtable_0.3.1	future_1.32.0	xfun_0.37
[70]	gower_1.0.1	proclim_2019.11.13	e1071_1.7-13
[73]	class_7.3-21	survival_3.5-3	timeDate_4022.108
[76]	tibble_3.2.1	iterators_1.0.14	hardhat_1.2.0
[79]	lava_1.7.2.1	timechange_0.2.0	globals_0.16.2
[82]	ellipsis_0.3.2	ipred_0.9-14	