title: “Homework 4” author: “Arjun Laxman” toc: true title-block-banner: true title-block-style: default format: pdf # format: html

> **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:

```
options(repos = c(CRAN = "https://cloud.r-project.org"))

# Load the 'torch' package and install it if it's not fully set up
if (!requireNamespace("torch", quietly = TRUE)) {
  # Install the torch package from CRAN
  install.packages("torch")
  # Install system dependencies and set up the package
  torch::install_torch()
}

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

# Load the packages using sapply
sapply(packages, require, character.only=TRUE)
```

```
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

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

Using your answer from above, what is the answer to

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

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?

```
# Load the numDeric package
library(numDeriv)

# Define the function g(x, y) where (x, y) are  the inputs
g <- function(vec) {
  x <- vec[1]
  y <- vec[2]
  return((x - 3)^2 + (y - 4)^2)
}

# Point which gradient is to be computed
point <- c(3, 4)
```

```
# Compute the gradient of g at the specified point
gradient <- grad(g, point)

# Extract and print the gradient components
gradient_x <- gradient[1]
gradient_y <- gradient[2]

print(paste("Gradient of g with respect to x at (x=3, y=4):", gradient_x))
```

```
[1] "Gradient of g with respect to x at (x=3, y=4): 0"
```

```
print(paste("Gradient of g with respect to y at (x=3, y=4):", gradient_y))
```

```
[1] "Gradient of g with respect to y at (x=3, y=4): 0"
```

1.2 (10 points)

Consider $h(\boldsymbol{u}, \boldsymbol{v})$ given by

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

where $\boldsymbol{u} \cdot \boldsymbol{v}$ denotes the dot product of two vectors, i.e., $\boldsymbol{u} \cdot \boldsymbol{v} = \sum_{i=1}^{n} u_i v_i$.

Using elementary calculus derive the expressions for the gradients

$$\nabla_{\boldsymbol{u}} h(\boldsymbol{u}, \boldsymbol{v}) = \left(\frac{d}{du_1} h(\boldsymbol{u}, \boldsymbol{v}), \frac{d}{du_2} h(\boldsymbol{u}, \boldsymbol{v}), \ldots, \frac{d}{du_n} h(\boldsymbol{u}, \boldsymbol{v}) \right)$$

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

$$\boldsymbol{u} = (-1, +1, -1, +1, -1, +1, -1, +1, -1, +1)$$
$$\boldsymbol{v} = (-1, -1, -1, -1, -1, +1, +1, +1, +1, +1)$$

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

```
# Install and load the torch package
if (!requireNamespace("torch", quietly = TRUE)) {
    install.packages("torch")
    torch::install_torch()
}

# Load the torch library
library(torch)


# Initialize vector u and v with specific values
```

```
u <- torch_tensor(c(-1, 1, -1, 1, -1, 1, -1, 1, -1, 1), dtype = torch_double())
v <- torch_tensor(c(-1, -1, -1, -1, -1, 1, 1, 1, 1, 1), dtype = torch_double())

# Calculate the dot product u and v
uv <- torch_dot(u, v)

# Compute the gradient of h withrespect to u using dot product
gradient_u <- 3 * uv^2 * v

# Display the computed gradient
print(gradient_u)
```

```
torch_tensor
-12
-12
-12
-12
-12
 12
 12
 12
 12
 12
[ CPUDoubleType{10} ]
```

1.3 (5 points)

Consider the following function

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

Derive the expression for

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

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

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

```
# Load the torch library
library(torch)

# Initialize a tensor for z with a specific value, enable gradient tracking
z_tensor <- torch_tensor(-3.5, dtype = torch_double(), requires_grad = TRUE)

# Define the polynomial function f(z)
f <- function(z) {
  return(z^4 - 6 * z^2 - 3 * z + 4)
}
```

```
##
# Evaluate the function at the given tensor z
f_value <- f(z_tensor)

# Perform backpropagation to compute the gradient of f with respect to z
f_value$backward()
#
# Retrieve the computed gradient
gradient <- z_tensor$grad
#
# Output the gradient
print(gradient)
```

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

```
#
```

1.4 (5 points)

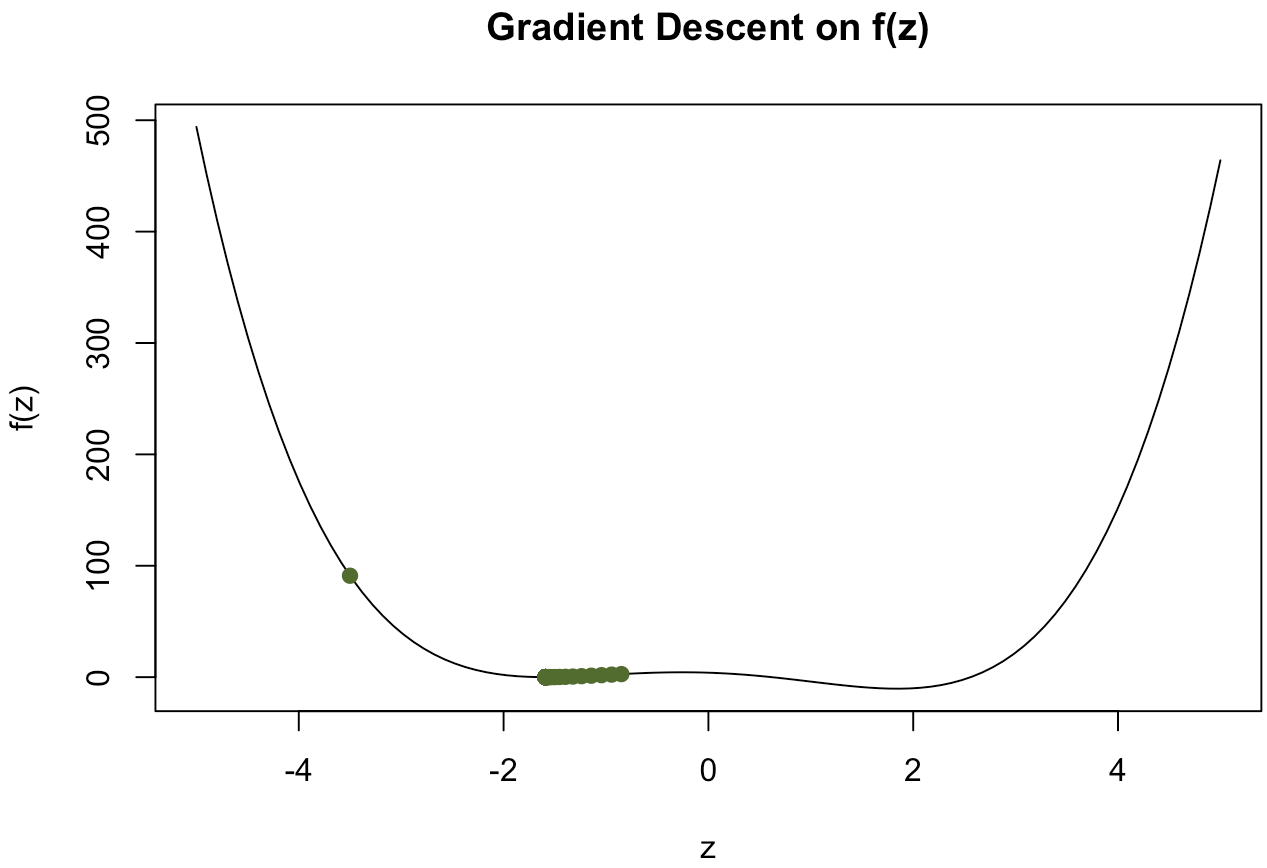
For the same function f, initialize $z[1] = -3.5$, and perform $n = 100$ iterations of **gradient descent**, i.e.,

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

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?

```
# Define the function f(z)
f <- function(z) {
  return(z^4 - 6 * z^2 - 3 * z + 4)
}

# Gradient function derived from f(z)
f_prime <- function(z) {
  return(4 * z^3 - 12 * z - 3)
}

# Gradient descent initialization
z <- -3.5
eta <- 0.02
n <- 100
z_values <- numeric(n)

# Gradient descent iterations
for (i in 1:n) {
  z_values[i] <- z
  z <- z - eta * f_prime(z)
```

```
}

# Plot the function and points
z_plot <- seq(-5, 5, length.out = 100)
plot(z_plot, f(z_plot), type = 'l', main = "Gradient Descent on f(z)", ylab = "f(z)", xla
points(z_values, f(z_values), col = 'darkolivegreen', pch = 19)
```

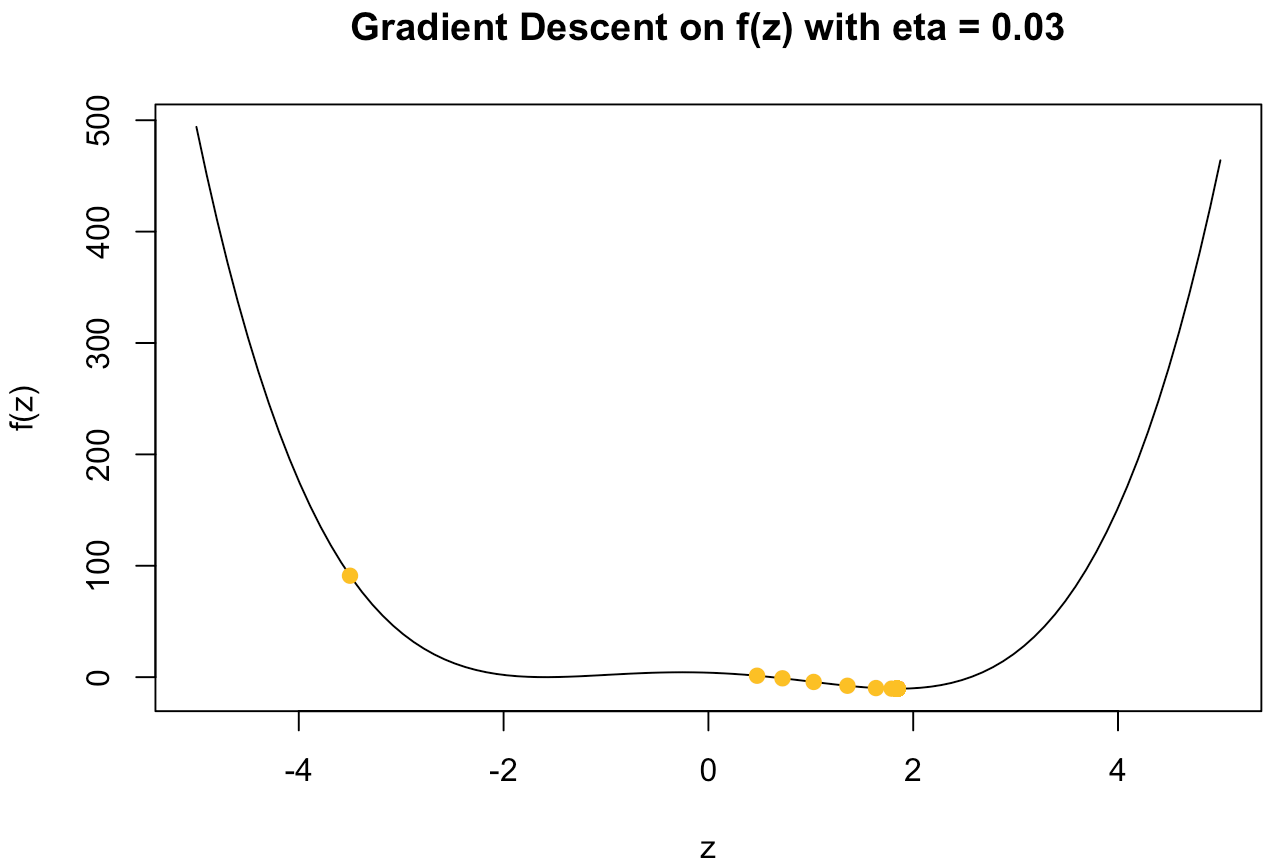


1.5 (5 points)

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

```
#redo
z <- -3.5
eta <- 0.03
z_values_eta3 <- numeric(n)

for (i in 1:n) {
  z_values_eta3[i] <- z
  z <- z - eta * f_prime(z)
}

# Plot the function and points with new learning rate
```

```
plot(z_plot, f(z_plot), type = 'l', main = "Gradient Descent on f(z) with eta = 0.03", yl
points(z_values_eta3, f(z_values_eta3), col = 'goldenrod1', pch = 19)
```



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.

```
# Load required packages
library(dplyr)
library(readr)


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

# Read the dataset from the URL and preprocess it
df <- read_csv(url) %>%
  # Rename columns for consistency and clarity
  rename(
    y = Survived,
    Pclass = Pclass,
    Sex = Sex,
    Age = Age,
    `Siblings/Spouses Aboard` = `Siblings/Spouses Aboard`,
    `Parents/Children Aboard` = `Parents/Children Aboard`,
    Fare = Fare
  ) %>%
  # Convert all character columns to factors for categorical analysis
  mutate(across(where(is.character), as.factor))
```

```
Rows: 887 Columns: 8
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): Name, Sex
dbl (6): Survived, Pclass, Age, Siblings/Spouses Aboard, Parents/Children Ab...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Display the first few rows of the preprocessed dataset to verify changes
head(df)
```

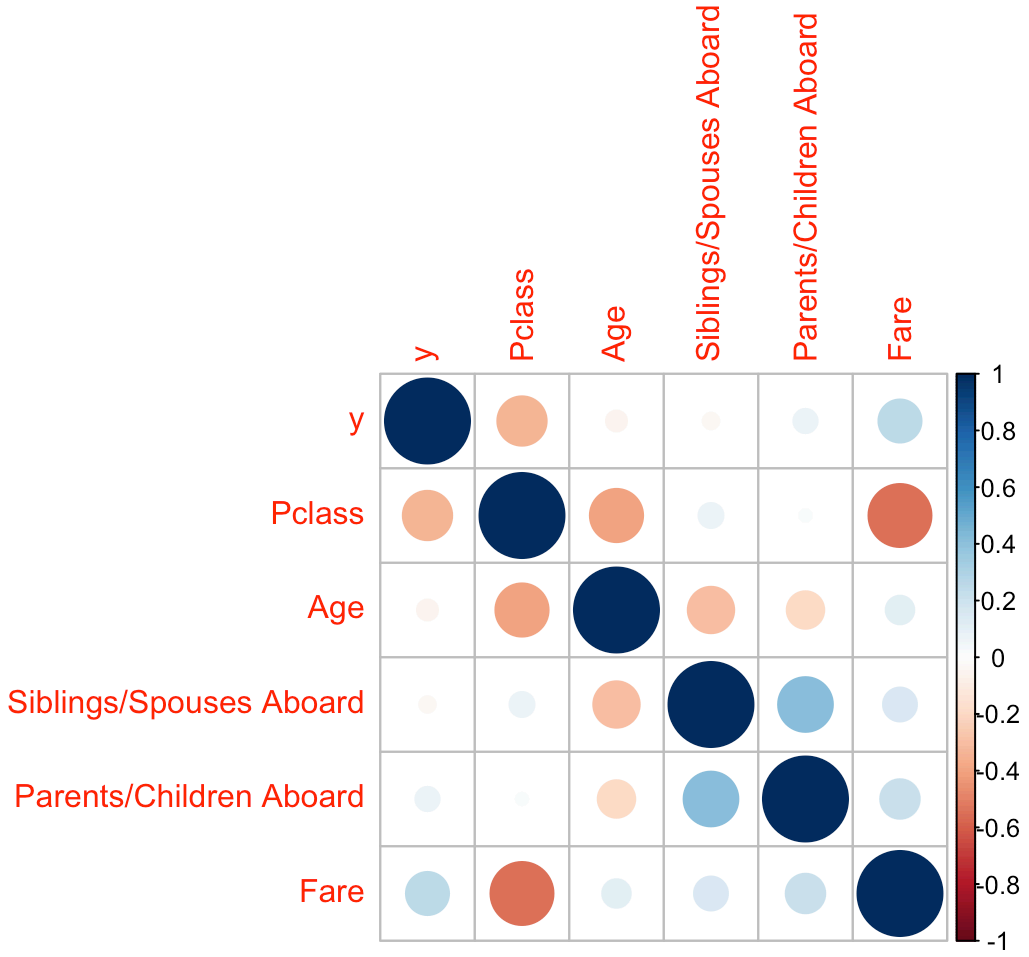
```
# A tibble: 6 × 8
      y Pclass Name    Sex     Age Siblings/Spouses Abo…¹ Parents/Children Abo…²
  <dbl>  <dbl> <fct>   <fct> <dbl>                  <dbl>                  <dbl>
1     0      3 Mr. Ow… male     22                      1                      0
2     1      1 Mrs. J… fema…    38                      1                      0
3     1      3 Miss. … fema…    26                      0                      0
4     1      1 Mrs. J… fema…    35                      1                      0
5     0      3 Mr. Wi… male     35                      0                      0
6     0      3 Mr. Ja… male     27                      0                      0
```

```
# ℹ abbreviated names: ¹`Siblings/Spouses Aboard`, ²`Parents/Children Aboard`
# ℹ 1 more variable: Fare <dbl>
```

2.2 (5 points)

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

```
# Load necessary library for visualization
library(corrplot)

# elect only numeric columns from the dataset
numeric_df <- df %>% select(where(is.numeric))

# Compute the correlation matrix for numeric columns
correlation_matrix <- cor(numeric_df)

# Visualize correlation matrix
corrplot(correlation_matrix, method = "circle")
```



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

```
# Load d library for statistical modeling
library(stats)


# Predict variable 'y' based on passenger class, sex, age, fare, number of siblings/spous
full_model <- glm(y ~ Pclass + Sex + Age + Fare + `Siblings/Spouses Aboard` + `Parents/Ch
                  data = df,
                  family = binomial)  # Specify 'binomial' family for logistic regression

# Display a detailed summary of the fitted model
# This includes coefficients, standard errors, z-values, and p-values, which help assess
summary(full_model)
```

```
Call:
glm(formula = y ~ Pclass + Sex + Age + Fare + `Siblings/Spouses Aboard` +
    `Parents/Children Aboard`, family = binomial, data = df)

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)                5.297252   0.557409   9.503  < 2e-16 ***
Pclass                    -1.177659   0.146079  -8.062 7.52e-16 ***
Sexmale                   -2.757282   0.200416 -13.758  < 2e-16 ***
Age                       -0.043474   0.007723  -5.629 1.81e-08 ***
Fare                       0.002786   0.002389   1.166 0.243680
`Siblings/Spouses Aboard` -0.401831   0.110712  -3.630 0.000284 ***
`Parents/Children Aboard` -0.106505   0.118588  -0.898 0.369127
---
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 880  degrees of freedom
AIC: 794.93

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 log-odds of survival are estimated using the logistic regression model according to the predictor factors. Now let's analyze the words for slope and intercept:

Term of Intercept (β_0): The estimated log-odds of survival when all other predictors are 0 are represented by the intercept term. It might not have a clear-cut application in this situation, but it does offer a starting point for survival probabilities.

Slope Terms (β_i): Keeping all other predictors fixed, each slope term shows the change in the log-odds of survival corresponding to a one-unit change in the relevant predictor variable.

for the continuous and categorical variables:

Pclass: A Category: The estimated coefficient value is the amount that the log-odds of survival drop with each unit rise in pclass (e.g., shifting from lower to higher class). This coefficient may be exponentiated to provide the odds ratio, which shows how the probabilities of survival vary between class levels.

Category: Sex For the sex variable, if it is encoded as "female" = 1 and "male" = 0, the slope term represents the change in the log-odds of survival associated with being female compared to being male. The odds ratio of female survival relative to male survival may be obtained by exponentiating this coefficient.

Age (Continuous): The log-odds of surviving fall by the predicted coefficient value for every unit increase in age. The multiplicative change in the probabilities of survival associated with an age increase of one year may be obtained by exponentiating this coefficient.

Fare (Continuous): Likewise, the log-odds of survival rise by the anticipated coefficient value for every unit increase in fare. The multiplicative change in the probability of survival associated with a one-unit increase in fare would be obtained by exponentiating this coefficient.

Sibsp and Parch (Continuous): These variables show how many parents/children and siblings/spouses are traveling with you. The interpretation would be the same for age and fare, but for every extra parent/child or sibling/spouse.

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){
    # Calculate the accuracy: proportion of correct predictions
    accuracy <- sum(predicted == expected) / length(predicted)

    # Calculate the error: proportion of incorrect predictions
    error <- 1 - accuracy

    # Calculate the total number of false positives: predicted as positive but actually n
    total_false_positives <- sum(predicted == 1 & expected == 0)

    # Calculate the total number of true positives: predicted as positive and actually po
    total_true_positives <- sum(predicted == 1 & expected == 1)

    # Calculate the total number of false negatives: predicted as negative but actually p
    total_false_negatives <- sum(predicted == 0 & expected == 1)

    # Calculate the total number of true negatives: predicted as negative and actually ne
    total_true_negatives <- sum(predicted == 0 & expected == 0)

    # Calculate the false positive rate: proportion of negative cases incorrectly classif
    false_positive_rate <- total_false_positives / (total_false_positives + total_true_ne

    # Calculate the false negative rate: proportion of positive cases incorrectly classif
    false_negative_rate <- total_false_negatives / (total_false_negatives + total_true_po

    # Return a data frame with all the calculated metrics
    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`

```
# A threshold of 0.5 is used to classify predictions into binary outcomes (0 or 1)
# `df$y` contains the actual binary outcomes from the dataset

performance_metrics <- overview(
  predicted = predict(full_model, type = "response") > 0.5,
# Convert the   probabilities to 0 or 1 based on threshold
  expected = df$y # Actual values from the dataset
)

# Print the computed performance metrics
print(performance_metrics)
```

```
   accuracy     error false_positive_rate false_negative_rate
1 0.8015784 0.1984216            0.133945           0.3011696
```

3.3 (5 points)

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

```
# Perform backward stepwise regression to simplify the logistic regression model
step_model <- step(full_model, direction = "backward")
```

```
Start:  AIC=794.93
y ~ Pclass + Sex + Age + Fare + `Siblings/Spouses Aboard` + `Parents/Children Aboard`

                           Df Deviance     AIC
- `Parents/Children Aboard`  1   781.75  793.75
- Fare                       1   782.43  794.43
<none>                           780.93  794.93
- `Siblings/Spouses Aboard`  1   796.85  808.85
- Age                        1   815.81  827.81
- Pclass                     1   847.84  859.84
- Sex                        1  1021.33 1033.33

Step:  AIC=793.75
y ~ Pclass + Sex + Age + Fare + `Siblings/Spouses Aboard`

                           Df Deviance     AIC
- Fare                       1   782.88  792.88
<none>                           781.75  793.75
- `Siblings/Spouses Aboard`  1   801.59  811.59
```

```
- Age                        1   816.44  826.44
- Pclass                     1   852.19  862.19
- Sex                        1  1025.55 1035.55

Step:  AIC=792.88
y ~ Pclass + Sex + Age + `Siblings/Spouses Aboard`

                           Df Deviance     AIC
<none>                          782.88  792.88
- `Siblings/Spouses Aboard`  1   801.61  809.61
- Age                        1   818.41  826.41
- Pclass                     1   900.80  908.80
- Sex                        1  1031.86 1039.86
```

```r
# Output a summary of the reduced model to evaluate its coefficients and significance
summary(step_model)
```

```
Call:
glm(formula = y ~ Pclass + Sex + Age + `Siblings/Spouses Aboard`,
    family = binomial, data = df)

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)                5.532066   0.504750  10.960  < 2e-16 ***
Pclass                    -1.265129   0.127021  -9.960  < 2e-16 ***
Sexmale                   -2.736487   0.195730 -13.981  < 2e-16 ***
Age                       -0.043697   0.007695  -5.679 1.36e-08 ***
`Siblings/Spouses Aboard` -0.407770   0.105197  -3.876 0.000106 ***
---
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:  782.88  on 882  degrees of freedom
AIC: 792.88

Number of Fisher Scoring iterations: 5
```

```r
# Generatse binary predictions from simplified model for evaluation
step_predictions <- predict(step_model, type = "response") > 0.5

# Calculates and display performance metric for simplified model
overview(step_predictions, df$y)
```

```
   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::trainConrol()` function

```
#install caret
if (!requireNamespace("caret", quietly = TRUE)) {
  install.packages("caret")
}
```

```
library(caret)

# Set up 5-fold cross-validation
controls <- trainControl(
  method = "cv",        # Use cross-validation
  number = 5,           # Number of folds
  savePredictions = "final",
  classProbs = TRUE     # Save class probabilities
)

# Define the range for lambda using a sequence from -20 to 0 by 0.5
lambda_values <- 2^seq(-20, 0, by = 0.5)

# Prepare the training data for LASSO logistic regression
train_data <- df[, c("Pclass", "Sex", "Age", "Fare", "Siblings/Spouses Aboard", "Parents/
train_labels <- df$y
```

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

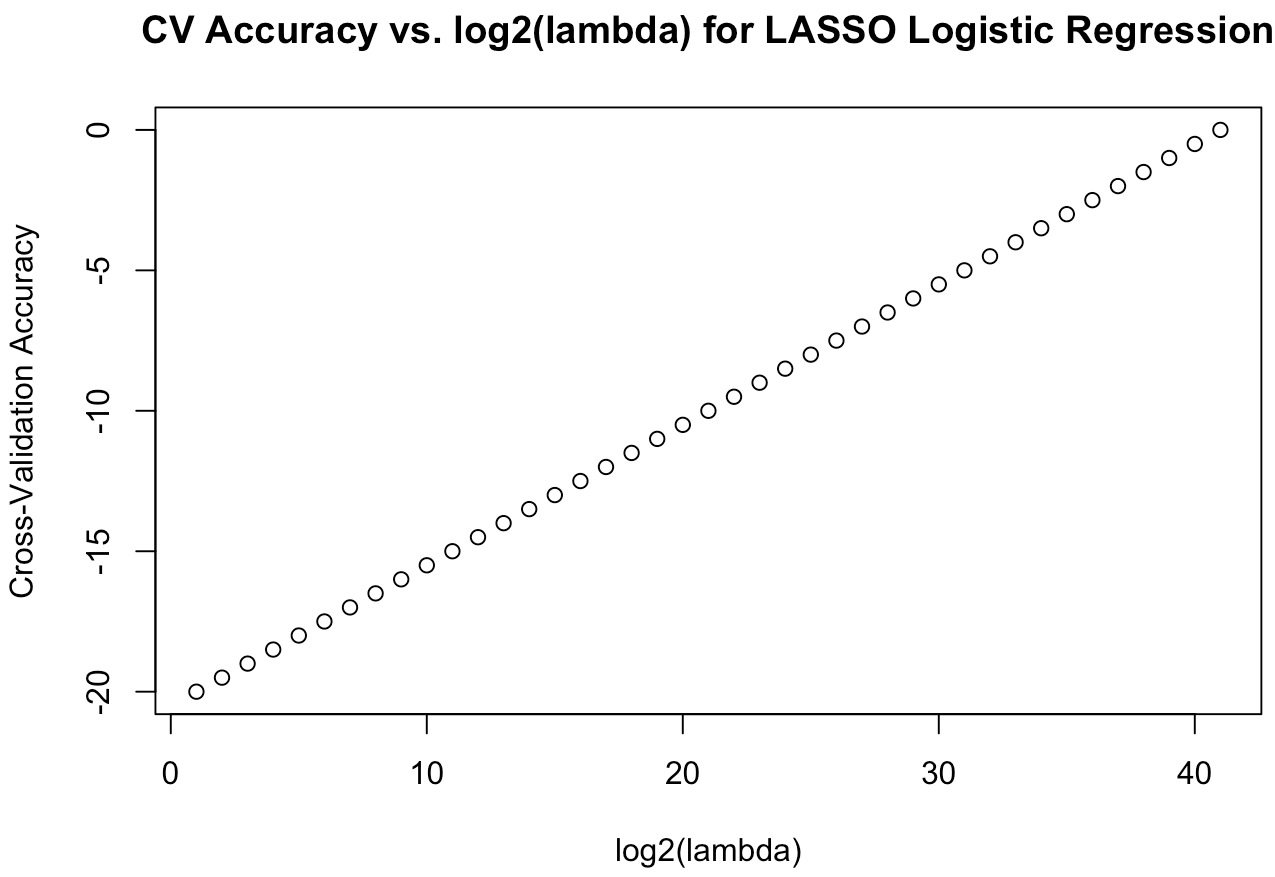
Take the search grid for λ to be in $\{2^{-20}, 2^{-19.5}, 2^{-19}, \ldots, 2^{-0.5}, 2^{0}\}$.

```
# Train the LASSO logistic regression model
lasso_fit <- train(
  x = train_data,
  y = train_labels,
  method = "glmnet",
  trControl = controls,
  tuneGrid = expand.grid(
    alpha = 1,         # LASSO penalty
    lambda = lambda_values
  ),
  family = "binomial"
)
```

```
Warning in train.default(x = train_data, y = train_labels, method = "glmnet", :
You are trying to do regression and your outcome only has two possible values
Are you trying to do classification? If so, use a 2 level factor as your
outcome column.
```

```
Warning in train.default(x = train_data, y = train_labels, method = "glmnet", :
cannnot compute class probabilities for regression

Warning: Setting row names on a tibble is deprecated.

Warning in storage.mode(xd) <- "double": NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning: Setting row names on a tibble is deprecated.

Warning in storage.mode(xd) <- "double": NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning: Setting row names on a tibble is deprecated.

Warning in storage.mode(xd) <- "double": NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning: Setting row names on a tibble is deprecated.

Warning in storage.mode(xd) <- "double": NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning: Setting row names on a tibble is deprecated.

Warning in storage.mode(xd) <- "double": NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning in cbind2(1, newx) %*% nbeta: NAs introduced by coercion

Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
: There were missing values in resampled performance measures.

Warning: Setting row names on a tibble is deprecated.

Warning in storage.mode(xd) <- "double": NAs introduced by coercion
```

```
# Plot cross-validation results for accuracy vs. log2(lambda)
plot(log2(lambda_values), lasso_fit$results$Accuracy, type = 'b',
     xlab = "log2(lambda)", ylab = "Cross-Validation Accuracy",
     main = "CV Accuracy vs. log2(lambda) for LASSO Logistic Regression")
```




```
# Report the optimal lambda
optimal_lambda <- lasso_fit$bestTune$lambda
cat("The optimal lambda is:", optimal_lambda, "\n")
```

```
The optimal lambda is: 1
```

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

```
# Convert data frame to model matrix for logistic regression
covariate_matrix <- model.matrix(~ Pclass + Sex + Age + Fare + `Siblings/Spouses Aboard`
```

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

```
# Convert to torch tensors
X <- torch_tensor(as.matrix(covariate_matrix), dtype = torch_float32())
y <- torch_tensor(as.numeric(df$y), dtype = torch_float32())
```

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$linear <- nn_linear(ncol(covariate_matrix), 1)
  },
  forward = function(x) {
    output <- self$linear(x)
    torch_sigmoid(output)
  }
)
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
 0.9794
 1.0000
 0.9897
 1.0000
 0.9988
 0.9941
 1.0000
 0.7632
 0.9961
 0.9904
 0.7481
 1.0000
 0.9751
 1.0000
 0.8930
 1.0000
 0.8648
 0.9943
 0.9985
 0.9754
 0.9999
 0.9994
 0.9124
 0.9999
 0.8888
```

```
 1.0000
 0.9918
 1.0000
 0.9845
 0.9861
... [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, model) {
  # Define the loss function as binary cross-entropy
  criterion <- nn_binary_cross_entropy_with_logits()
  loss <- criterion(model(x), y)
  loss
}
```

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

```
f <- logistic()

# Get parameters of the neural network
parameters <- f$parameters


# Set up the optimizer with a learning rate of 0.01
optimizer <- optim_adam(parameters, lr = 0.01)
```

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

```
# Convert model outputs to binary predictions and compute performance metrics
predicted_probabilities <- f(X)$detach() %>% as_array()
predicted <- ifelse(predicted_probabilities > 0.5, 1, 0)
overview(predicted, df$y)
```

```
   accuracy     error false_positive_rate false_negative_rate
1 0.6448703 0.3551297           0.0440367           0.8508772
```

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.

```
# Evaluates tne Full Model performance
overview_full <- overview(predict(full_model, type = "response") > 0.5, df$y)

# Evaluates the Stepwise Model performance
overview_step <- overview(predict(step_model, type = "response") > 0.5, df$y)
```

```
# Gather LASSO Model CV result and best lambda values
cv_results <- lasso_fit$results
best_lambda <- lasso_fit$bestTune$lambda

# Evaluate PyTorch Model performanc
overview_torch <- overview(predicted, df$y)

# Compile model performance summary into a table
summary_table <- data.frame(
  Model = c("Full Model", "Stepwise Model", "LASSO with Logistic Regression", "PyTorch Lo
  Accuracy = c(overview_full$accuracy, overview_step$accuracy, max(cv_results$Accuracy),
  Error = c(overview_full$error, overview_step$error, 1 - max(cv_results$Accuracy), overv
  False_Positive_Rate = c(overview_full$false_positive_rate, overview_step$false_positive
  False_Negative_Rate = c(overview_full$false_negative_rate, overview_step$false_negative
)
```

```
Warning in max(cv_results$Accuracy): no non-missing arguments to max; returning
-Inf

Warning in max(cv_results$Accuracy): no non-missing arguments to max; returning
-Inf
```

```
# Print summary table
print(summary_table)
```

```
                         Model  Accuracy     Error False_Positive_Rate
1                   Full Model 0.8015784 0.1984216           0.1339450
2                Stepwise Model 0.8049605 0.1950395           0.1339450
3 LASSO with Logistic Regression      -Inf       Inf                  NA
4    PyTorch Logistic Regression 0.6448703 0.3551297           0.0440367
  False_Negative_Rate
1           0.3011696
2           0.2923977
3                  NA
4           0.8508772
```

The Full Model may include overfitting and increased complexity, but it offers a thorough understanding of the connections between all predictors and the response variable.

The Stepwise Model seeks to minimize complexity and enhance prediction accuracy by choosing a subset of predictors; nevertheless, because of the variable selection processes, it may generate instability or ignore significant factors.

In order to promote sparsity in the model, LASSO with Logistic Regression penalizes the absolute magnitude of the regression coefficients. This strikes a compromise between model complexity and predictive accuracy. It can be necessary to fine-tune the choice of the ideal regularization parameter (lambda).

For training complicated models, PyTorch Logistic Regression offers flexibility and scalability, but it may be computationally demanding and necessitates careful hyperparameter optimization. In addition, in contrast to conventional statistical models, the interpretation of the data could be more difficult.

Session Information