Homework 5

Advait Ashtikar

Table of contents

Link to the Github repository

❗ Due: Wed, Apr 19, 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

In this assignment, we will explore decision trees, support vector machines and neural networks for classification and regression. The assignment is designed to test your ability to fit and analyze these models with different configurations and compare their performance.

We will need the following packages:

```r
library(dplyr)
library(readr)
library(tidyr)
library(purrr)
library(broom)
library(magrittr)
library(corrplot)
library(caret)
library(rpart)
library(rpart.plot)
library(e1071)
library(torch)
library(luz)
```

Question 1

60 points

Prediction of Median House prices

1.1 (2.5 points)

The `data` folder contains the `housing.csv` dataset which contains housing prices in California from the 1990 California census. The objective is to predict the median house price for California districts based on various features.

Read the data file as a tibble in R. Preprocess the data such that:

1. the variables are of the right data type, e.g., categorical variables are encoded as factors
2. all column names to lower case for consistency
3. Any observations with missing values are dropped

```r
path <- "data/housing.csv"

df <- read_csv(path) %>%
  mutate_if(is.character, as.factor) %>%
  rename_all(tolower) %>%
  na.omit()
```

```
Rows: 20640 Columns: 10
-- Column specification --------------------------------------------------------
Delimiter: ","
```

```
chr (1): ocean_proximity
dbl (9): longitude, latitude, housing_median_age, total_rooms, total_bedroom...

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.
```

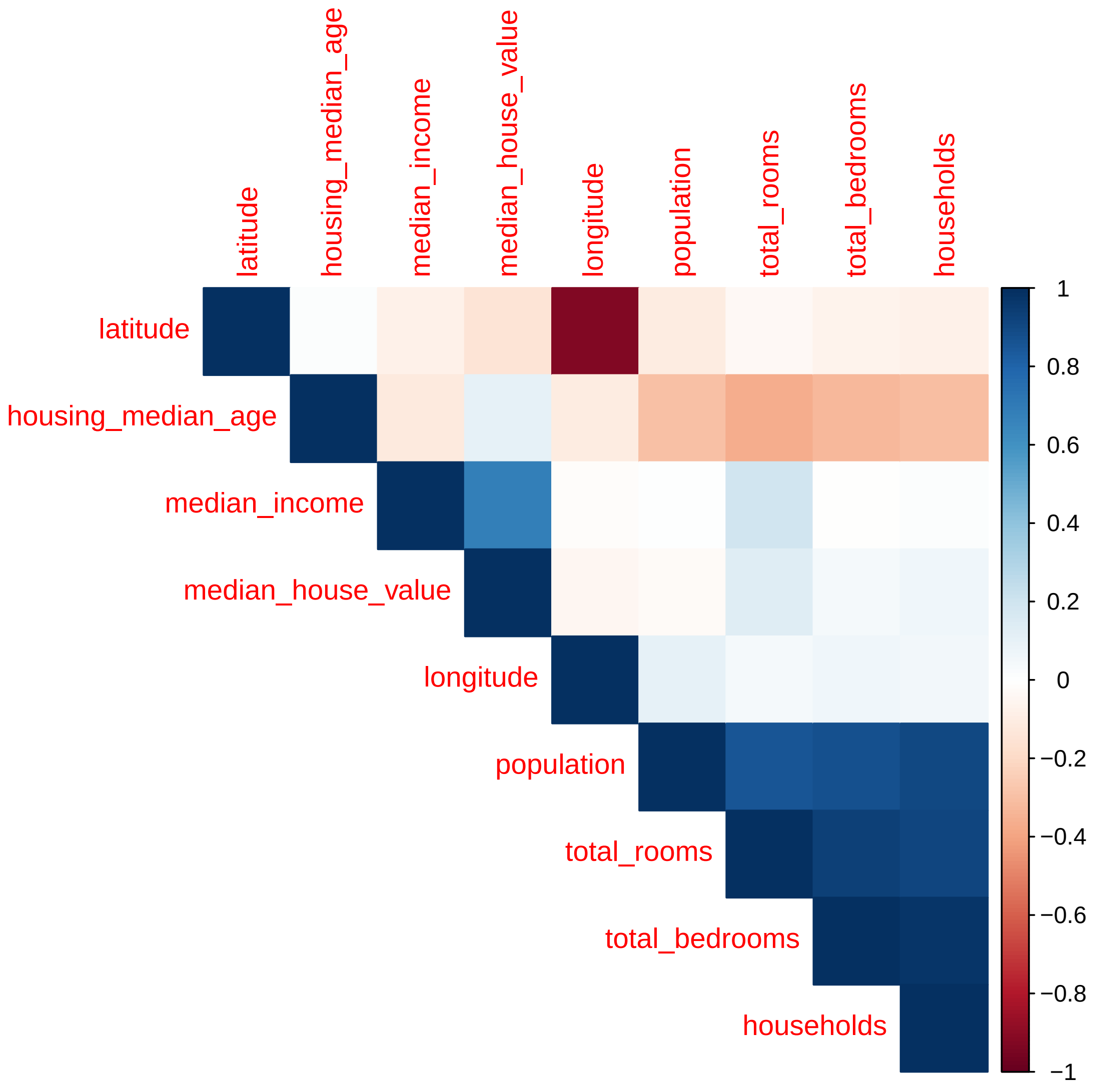
```
df %>% head
```

```
# A tibble: 6 x 10
  longitude latitude housing_median_age total_rooms total_bedrooms population
      <dbl>    <dbl>              <dbl>       <dbl>          <dbl>      <dbl>
1     -122.     37.9                 41         880            129        322
2     -122.     37.9                 21        7099           1106       2401
3     -122.     37.8                 52        1467            190        496
4     -122.     37.8                 52        1274            235        558
5     -122.     37.8                 52        1627            280        565
6     -122.     37.8                 52         919            213        413
# i 4 more variables: households <dbl>, median_income <dbl>,
#   median_house_value <dbl>, ocean_proximity <fct>
```

1.2 (2.5 points)

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

```
df %>%
  select_if(is.numeric) %>%
  cor() %>%
  corrplot(type = "upper", method = "color", order = "hclust")
```



1.3 (5 points)

Split the data `df` into `df_train` and `df_split` using `test_ind` in the code below:

```
set.seed(42)
test_ind <- sample(
```

```r
  1:nrow(df),
  floor( nrow(df)/10 ),
  replace=FALSE
)

df_train <- df[-test_ind, ]
df_test  <- df[test_ind, ]
```

1.4 (5 points)

Fit a linear regression model to predict the `median_house_value` :

- `latitude`
- `longitude`
- `housing_median_age`
- `total_rooms`
- `total_bedrooms`
- `population`
- `median_income`
- `ocean_proximity`

Interpret the coefficients and summarize your results.

```r
lm_fit <- lm(median_house_value ~ . -households, data = df_train)

summary(lm_fit)
```

```
Call:
lm(formula = median_house_value ~ . - households, data = df_train)

Residuals:
    Min      1Q  Median      3Q     Max
-559024  -42322  -10389   28743  710215

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)               -2.273e+06  9.138e+04 -24.873  < 2e-16 ***
longitude                 -2.681e+04  1.060e+03 -25.305  < 2e-16 ***
latitude                  -2.539e+04  1.047e+03 -24.244  < 2e-16 ***
```

```
housing_median_age         1.074e+03  4.616e+01  23.261  < 2e-16 ***
total_rooms               -6.159e+00  8.431e-01  -7.306 2.87e-13 ***
total_bedrooms             1.353e+02  4.254e+00  31.804  < 2e-16 ***
population                -3.413e+01  9.838e-01 -34.694  < 2e-16 ***
median_income              3.936e+04  3.573e+02 110.154  < 2e-16 ***
ocean_proximityINLAND     -4.018e+04  1.836e+03 -21.891  < 2e-16 ***
ocean_proximityISLAND      1.324e+05  3.442e+04   3.847  0.00012 ***
ocean_proximityNEAR BAY   -2.522e+03  2.022e+03  -1.247  0.21226
ocean_proximityNEAR OCEAN  4.349e+03  1.658e+03   2.622  0.00875 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 68780 on 18378 degrees of freedom
Multiple R-squared:  0.643, Adjusted R-squared:  0.6428
F-statistic:  3009 on 11 and 18378 DF,  p-value: < 2.2e-16
```

Based on the summary , we can see that …

1.5 (5 points)

Complete the `rmse` function for computing the Root Mean-Squared Error between the true `y` and the predicted `yhat`, and use it to compute the RMSE for the regression model on `df_test`

```r
rmse <- function(y, yhat) {
  sqrt(mean((y - yhat)^2))
}

lm_predictions <- predict(lm_fit, newdata = df_test)

lm_RMSE <- rmse(df_test$median_house_value, lm_predictions)
lm_RMSE
```

```
[1] 68339.82
```

1.6 (5 points)

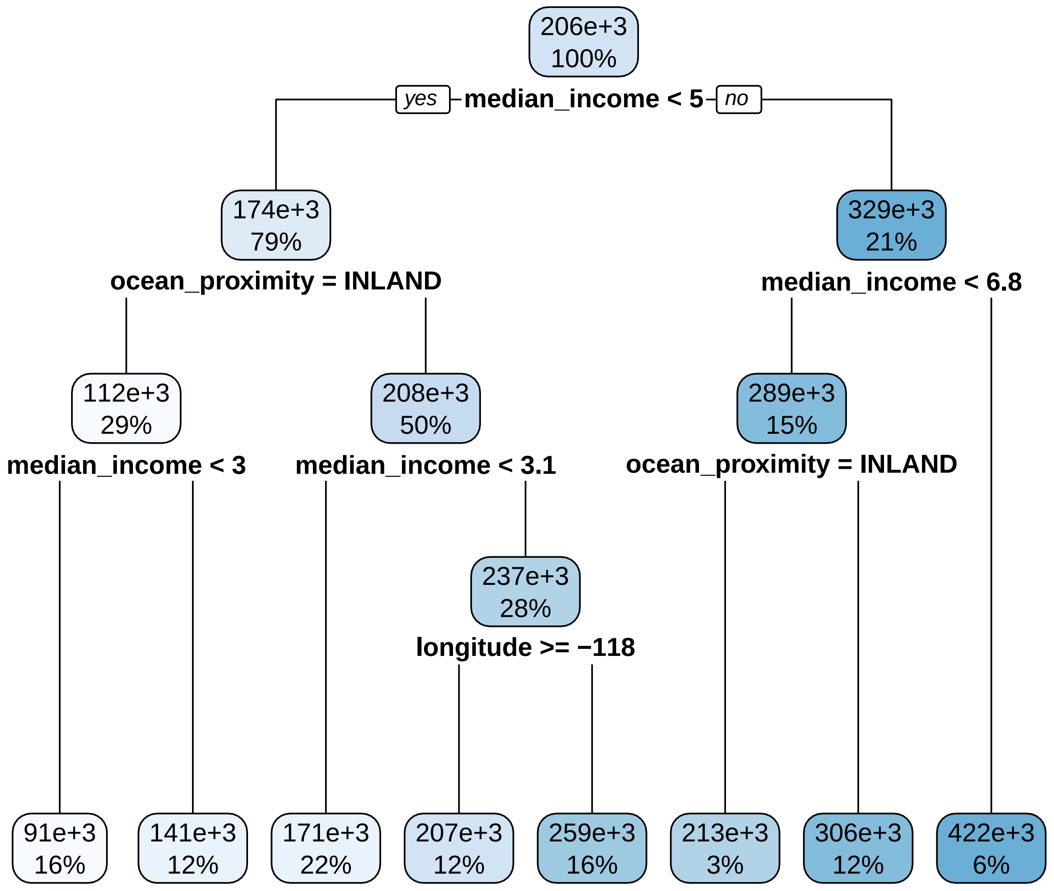
Fit a decision tree model to predict the `median_house_value` using the same predictors as in 1.4. Use the `rpart()` function.

```r
rpart_fit <- rpart(median_house_value ~ . -households, data = df_train)

rpart_predictions <- predict(rpart_fit, newdata = df_test)
```

Visualize the decision tree using the `rpart.plot()` function.

```r
rpart.plot(rpart_fit)
```



Report the root mean squared error on the test set.

```r
rpart_RMSE <- rmse(df_test$median_house_value, rpart_predictions)
rpart_RMSE
```

```
[1] 75876.87
```

1.7 (5 points)

Fit a support vector machine model to predict the `median_house_value` using the same predictors as in 1.4. Use the `svm()` function and use any kernel of your choice. Report the root mean squared error on the test set.

```r
svm_fit <- svm(median_house_value ~ . -households, data = df_train, kernel = "radial")

svm_predictions <- predict(svm_fit, newdata = df_test)

svm_RMSE <- rmse(df_test$median_house_value, svm_predictions)
svm_RMSE
```

```
[1] 56678.84
```

1.8 (25 points)

Initialize a neural network model architecture:

```r
NNet <- nn_module(
    initialize = function(p, q1, q2, q3){
      self$hidden1 <- nn_linear(p, q1)
      self$hidden2 <- nn_linear(q1, q2)
      self$hidden3 <- nn_linear(q2, q3)
      self$output <- nn_linear(q3, 1)
      self$activation <- nn_relu()
      self$sigmoid <- nn_sigmoid()
    },
    forward = function(x){
      x %>%
        self$hidden1() %>% self$activation() %>%
        self$hidden2() %>% self$activation() %>%
        self$hidden3() %>% self$activation() %>%
        self$output() %>% self$sigmoid()
    }
)
```

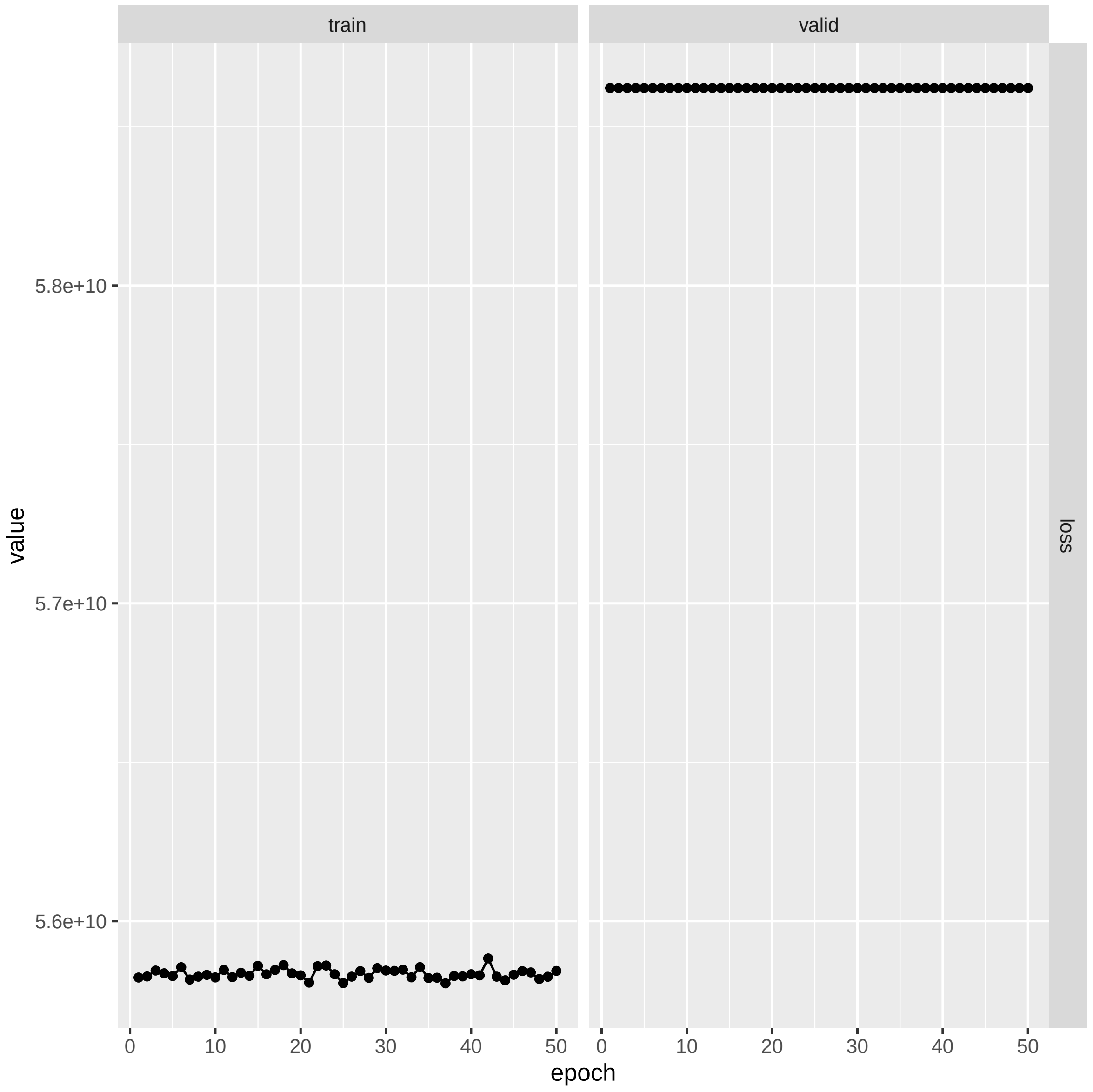
Fit a neural network model to predict the `median_house_value` using the same predictors as in 1.4. Use the `model.matrix` function to create the covariate matrix and `luz` package for fitting the network with $32, 16, 8$ nodes in each of the three hidden layers.

```r
M <- model.matrix(median_house_value ~ 0 + . -households, data = df_train)

nnet_fit <- NNet %>%
  setup(
    loss = nn_mse_loss(),
    optimizer = optim_adam,
  ) %>%
  set_hparams(
    p = ncol(M), q1 = 32, q2 = 16, q3 = 8
  ) %>%
  set_opt_hparams(
    lr = 0.1
  ) %>%
  fit(
    data = list(
      model.matrix(median_house_value ~ 0 + . -households, data = df_train),
      df_train %>% select(median_house_value) %>% as.matrix
    ),
    valid_data = list(
      model.matrix(median_house_value ~ 0 + . -households, data = df_test),
      df_test %>% select(median_house_value) %>% as.matrix
    ),
    epochs = 50,
    dataloader_options = list(batch_size = 128, shuffle = TRUE),
    verbose = FALSE # Change to TRUE while tuning. But, set to FALSE before submitting
  )
```

Plot the results of the training and validation loss and accuracy.

```r
plot(nnet_fit)
```



Report the root mean squared error on the test set.

```
nnet_predictions <- predict(nnet_fit,
                            model.matrix(median_house_value ~ 0 + . -households,
                                         data = df_test)) %>% as.array()

nn_RMSE <- rmse(df_test$median_house_value, nnet_predictions)
```

```r
nn_RMSE
```

```
[1] 242118.3
```

Warning

Remember to use the `as_array()` function to convert the predictions to a vector of numbers before computing the RMSE with `rmse()`

1.9 (5 points)

Summarize your results in a table comparing the RMSE for the different models. Which model performed best? Why do you think that is?

```r
summaryTable <- data.frame(
  Model = c('Linear Regression', 'Decison Tree', 'SVM', 'Newral Network'),
  RMSE = c(lm_RMSE, rpart_RMSE, svm_RMSE, nn_RMSE)
)

summaryTable
```

```
              Model      RMSE
1 Linear Regression  68339.82
2      Decison Tree  75876.87
3               SVM  56678.84
4    Newral Network 242118.32
```

> The model that performed best was the SVM model. This can because the data is non-linear, and SVM is good at handling non-linear data through the different kernels that can be used for the SVM model.

—

Question 2

50 points

Spam email classification

The `data` folder contains the `spam.csv` dataset. This dataset contains features extracted from a collection of spam and non-spam emails. The objective is to classify the emails as spam or non-spam.

2.1 (2.5 points)

Read the data file as a tibble in R. Preprocess the data such that:

1. the variables are of the right data type, e.g., categorical variables are encoded as factors
2. all column names to lower case for consistency
3. Any observations with missing values are dropped

```r
path <- "data/spambase.csv"

df <- read_csv(path) %>%
  select(-contains("Unnamed")) %>%
  mutate_if(is.character, as.factor) %>%
  rename_all(tolower) %>%
  na.omit()
```

```
Rows: 4601 Columns: 58
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (58): word_freq_1, word_freq_2, word_freq_3, word_freq_4, word_freq_5, w...

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.
```

```r
df %>% head
```

```
# A tibble: 6 x 58
  word_freq_1 word_freq_2 word_freq_3 word_freq_4 word_freq_5 word_freq_6
        <dbl>       <dbl>       <dbl>       <dbl>       <dbl>       <dbl>
1        0           0.64        0.64           0        0.32        0
2        0.21        0.28        0.5            0        0.14        0.28
3        0.06        0           0.71           0        1.23        0.19
4        0           0           0              0        0.63        0
5        0           0           0              0        0.63        0
6        0           0           0              0        1.85        0
# i 52 more variables: word_freq_7 <dbl>, word_freq_8 <dbl>, word_freq_9 <dbl>,
#   word_freq_10 <dbl>, word_freq_11 <dbl>, word_freq_12 <dbl>,
#   word_freq_13 <dbl>, word_freq_14 <dbl>, word_freq_15 <dbl>,
#   word_freq_16 <dbl>, word_freq_17 <dbl>, word_freq_18 <dbl>,
#   word_freq_19 <dbl>, word_freq_20 <dbl>, word_freq_21 <dbl>,
#   word_freq_22 <dbl>, word_freq_23 <dbl>, word_freq_24 <dbl>,
#   word_freq_25 <dbl>, word_freq_26 <dbl>, word_freq_27 <dbl>, ...
```

2.2 (2.5 points)

Split the data `df` into `df_train` and `df_split` using `test_ind` in the code below:

```
set.seed(42)
test_ind <- sample(
  1:nrow(df),
  floor( nrow(df)/10 ),
  replace=FALSE
)

dfTrain <- df[-test_ind, ]
dfTest  <- df[test_ind, ]
```

Complete the `overview` function which returns a data frame with the following columns: `accuracy`, `error`, `false positive rate`, `true positive rate`, between the true `true_class` and the predicted `pred_class` for any classification model.

```
overview <- function(pred_class, true_class) {
  accuracy <- mean(pred_class == true_class)
  error <- 1 - accuracy
  true_positives <- sum(pred_class == "1" & true_class == "1")
  true_negatives <- sum(pred_class == "0" & true_class == "0")
```

```r
  false_positives <- sum(pred_class == "1" & true_class == "0")
  false_negatives <- sum(pred_class == "0" & true_class == "1")
  true_positive_rate <- true_positives / (true_positives + false_negatives)
  false_positive_rate <- false_positives / (true_negatives + false_positives)
  return(
    data.frame(
      accuracy = accuracy,
      error = error,
      true_positive_rate = true_positive_rate,
      false_positive_rate = false_positive_rate
    )
  )
}
```

2.3 (5 points)

Fit a logistic regression model to predict the `spam` variable using the remaining predictors. Report the prediction accuracy on the test set.

```r
glm_fit <- glm(
  spam ~ .,
  dfTrain %>% mutate_at("spam", factor),
  family = binomial()
)
```

```
Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
glm_test <- predict(glm_fit, dfTest, output = "response")
glm_classes <- ifelse(glm_test > 0.5, 1, 0)

overview(glm_classes, dfTest$spam)
```

```
   accuracy      error true_positive_rate false_positive_rate
1 0.9108696 0.08913043          0.8244681          0.02941176
```

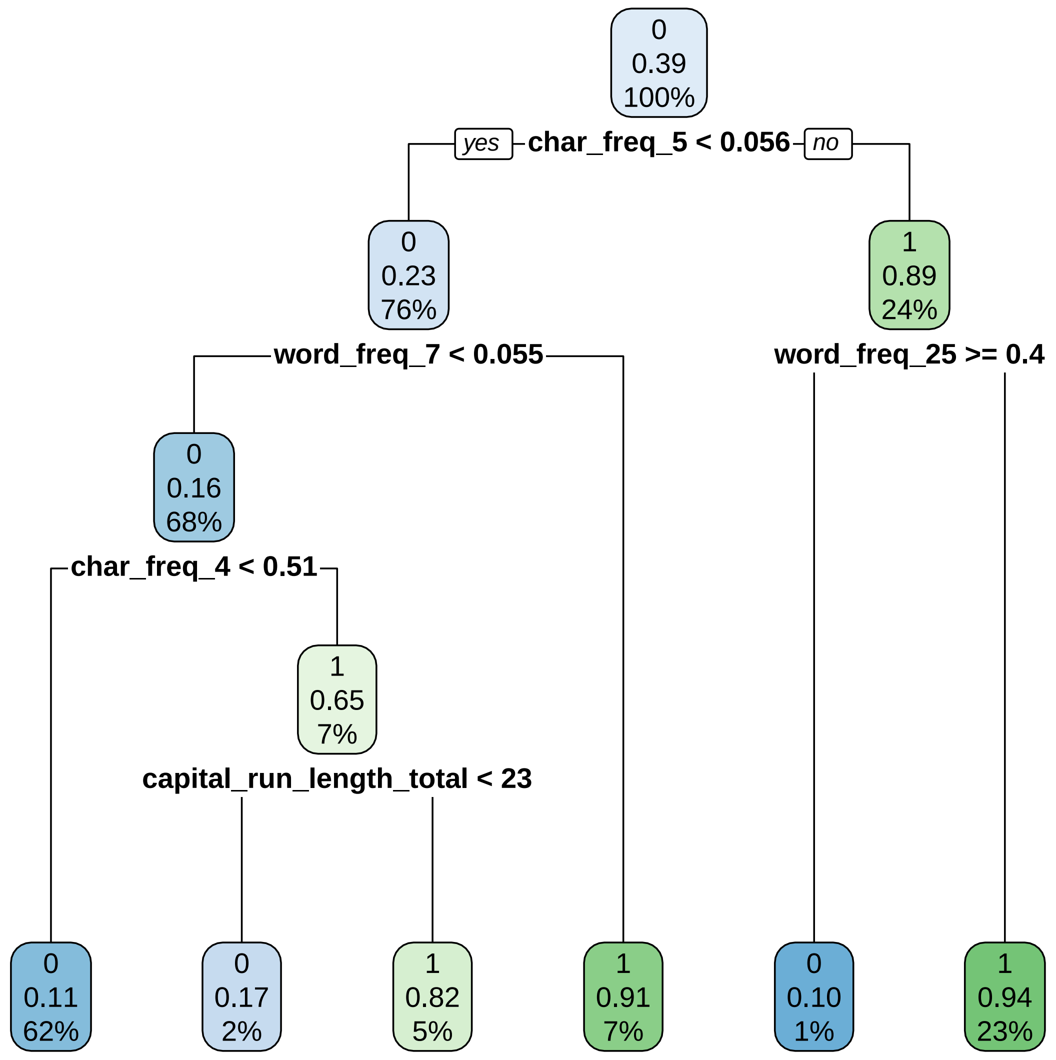
2.4 (5 points)

Fit a decision tree model to predict the `spam` variable using the remaining predictors. Use the `rpart()` function and set the `method` argument to `"class"`.

```r
rpart_fit <- rpart(spam ~ ., dfTrain, method = "class")
```

Visualize the decision tree using the `rpart.plot()` function.

```r
rpart.plot(rpart_fit)
```



Report the prediction accuracy on the test set.

```
rpart_classes <- ifelse(predict(rpart_fit, newdata = dfTest) > 0.5, 1, 0)

overview(rpart_classes, dfTest$spam)
```

```
  accuracy error true_positive_rate false_positive_rate
1      0.5   0.5                0.5                 0.5
```

2.5 (5 points)

Fit a support vector machine model to predict the `spam` variable using the remaining predictors. Use the `svm()` function and use any kernel of your choice. Remember to set the `type` argument to `"C-classification"` **if you haven't** already converted `spam` to be of type `factor`.

```
svm_fit <- svm(spam ~ ., dfTrain, kernel = "radial", type = "C-classification")
```

Report the prediction accuracy on the test set.

```
svm_classes <- predict(svm_fit, newdata = dfTest)
overview(svm_classes, dfTest$spam)
```

```
  accuracy      error true_positive_rate false_positive_rate
1 0.923913 0.07608696          0.8776596          0.04411765
```

2.6 (25 points)

Using the same neural network architecture as in 1.9, fit a neural network model to predict the `spam` variable using the remaining predictors.

> ⚠ Classification vs. Regression
>
> Note that the neural network in **Q 1.9** was a regression model. You will need to modify the neural network architecture to be a classification model by changing the output layer to have a single node with a sigmoid activation function.

Use the `model.matrix` function to create the covariate matrix and `luz` package for fitting the network with $32, 16, 8$ nodes in each of the three hidden layers.

```
NNet <- nn_module(
    initialize = function(p, q1, q2, q3){
        self$hidden1 <- nn_linear(p, q1)
        self$hidden2 <- nn_linear(q1, q2)
        self$hidden3 <- nn_linear(q2, q3)
        self$output <- nn_linear(q3, 1)
        self$activation <- nn_relu()
        self$sigmoid <- nn_sigmoid()
```

```r
    },
    forward = function(x){
      x %>%
        self$hidden1() %>% self$activation() %>%
        self$hidden2() %>% self$activation() %>%
        self$hidden3() %>% self$activation() %>%
        self$output() %>% self$sigmoid()
    }
)
```

```r
M1 <- model.matrix(spam ~ 0 + ., data = dfTrain)

nnet_fit <- NNet %>%
  setup(
    loss = nn_bce_loss(),
    optimizer = optim_adam,
  ) %>%
  set_hparams(
    p = ncol(M1), q1 = 32, q2 = 16, q3 = 8
  ) %>%
  set_opt_hparams(
    lr = 0.1
  ) %>%
  fit(
    data = list(
      model.matrix(spam ~ 0 + ., data = dfTrain),
      (dfTrain[["spam"]] %>% as.numeric() - 1) %>% as.matrix()
    ),
    valid_data = list(
      model.matrix(spam ~ 0 + ., data = dfTest),
      (dfTest[["spam"]] %>% as.numeric() - 1) %>% as.matrix()
    ),
    epochs = 100,
    dataloader_options = list(batch_size = 128, shuffle = TRUE),
    verbose = FALSE # Change to TRUE while tuning. But, set to FALSE before submitting
  )
```

```r
nnet_predictions <- ifelse(predict(nnet_fit,
                                   model.matrix(spam ~ 0 + ., data = dfTest)) > 0.5, 1, 0)
```

2.7 (5 points)

Summarize your results in a table comparing the accuracy metrics for the different models.

```r
list(glm_classes, rpart_classes, svm_classes, nnet_predictions) %>%
    lapply(\(x) overview(x, dfTest$spam)) %>%
    bind_rows() %>%
  cbind(Model = c('Logistic Regression', 'Decision Tree', 'SVM', 'Neural Network')) %>%
  select(Model, accuracy, error, true_positive_rate, false_positive_rate)
```

```
                Model  accuracy      error true_positive_rate
1 Logistic Regression 0.9108696 0.08913043          0.8244681
2       Decision Tree 0.5000000 0.50000000          0.5000000
3                 SVM 0.9239130 0.07608696          0.8776596
4      Neural Network 0.5913043 0.40869565          0.0000000
  false_positive_rate
1          0.02941176
2          0.50000000
3          0.04411765
4          0.00000000
```

If you were to choose a model to classify spam emails, which model would you choose? Think about the context of the problem and the cost of false positives and false negatives.

—

Question 3

60 points

Three spirals classification

To better illustrate the power of depth in neural networks, we will use a toy dataset called the "Three Spirals" data. This dataset consists of two intertwined spirals, making it challenging for shallow models to classify the data accurately.

This is a multi-class classification problem

The dataset can be generated using the provided R code below:

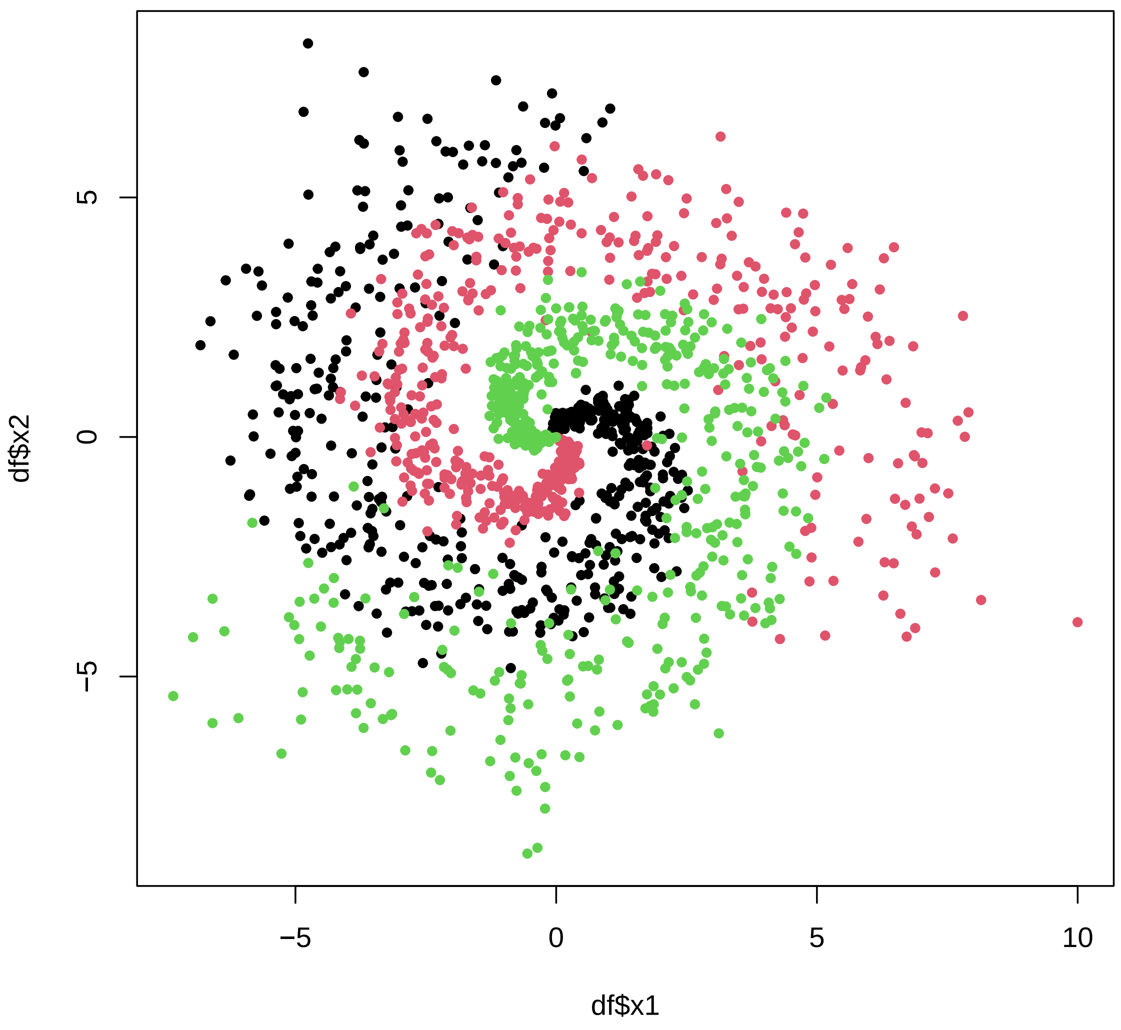
```r
generate_three_spirals <- function(){
  set.seed(42)
  n <- 500
  noise <- 0.2
  t <- (1:n) / n * 2 * pi
  x1 <- c(
      t * (sin(t) + rnorm(n, 0, noise)),
      t * (sin(t + 2 * pi/3) + rnorm(n, 0, noise)),
      t * (sin(t + 4 * pi/3) + rnorm(n, 0, noise))
    )
  x2 <- c(
      t * (cos(t) + rnorm(n, 0, noise)),
      t * (cos(t + 2 * pi/3) + rnorm(n, 0, noise)),
      t * (cos(t + 4 * pi/3) + rnorm(n, 0, noise))
    )
  y <- as.factor(
    c(
      rep(0, n),
      rep(1, n),
      rep(2, n)
    )
  )
  return(tibble(x1=x1, x2=x2, y=y))
}
```

3.1 (5 points)

Generate the three spirals dataset using the code above. Plot x_1 vs x_2 and use the `y` variable to color the points.

```r
df <- generate_three_spirals()

plot(
  df$x1, df$x2,
  col = df$y,
  pch = 20
)
```



Define a grid of 100 points from -10 to 10 in both x_1 and x_2 using the `expand.grid()`. Save it as a tibble called `df_test`.

```
grid <- expand.grid(
  x1 = seq(-10, 10, length.out = 100),
  x2 = seq(-10, 10, length.out = 100)
)
```

```r
df_test <- as_tibble(grid)
```

3.2 (10 points)

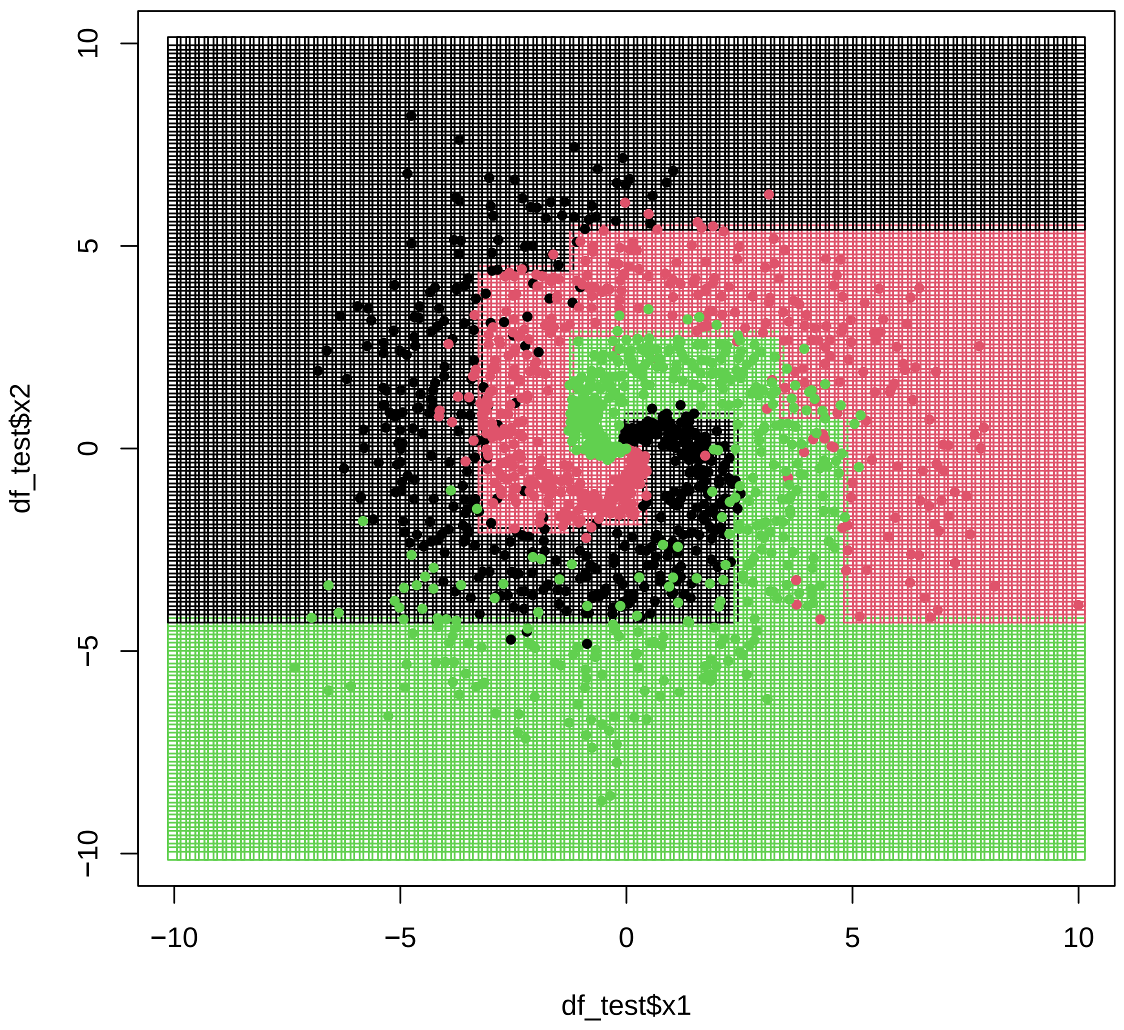
Fit a classification tree model to predict the `y` variable using the `x1` and `x2` predictors, and plot the decision boundary.

```r
rpart_fit <- rpart(y ~ x1 + x2, df, method = "class")
rpart_classes <- predict(rpart_fit, newdata = df_test, type = "class")
```

Plot the decision boundary using the following function:

```r
plot_decision_boundary <- function(predictions){
  plot(
    df_test$x1, df_test$x2,
    col = predictions,
    pch = 0
  )
  points(
    df$x1, df$x2,
    col = df$y,
    pch = 20
  )
}
```

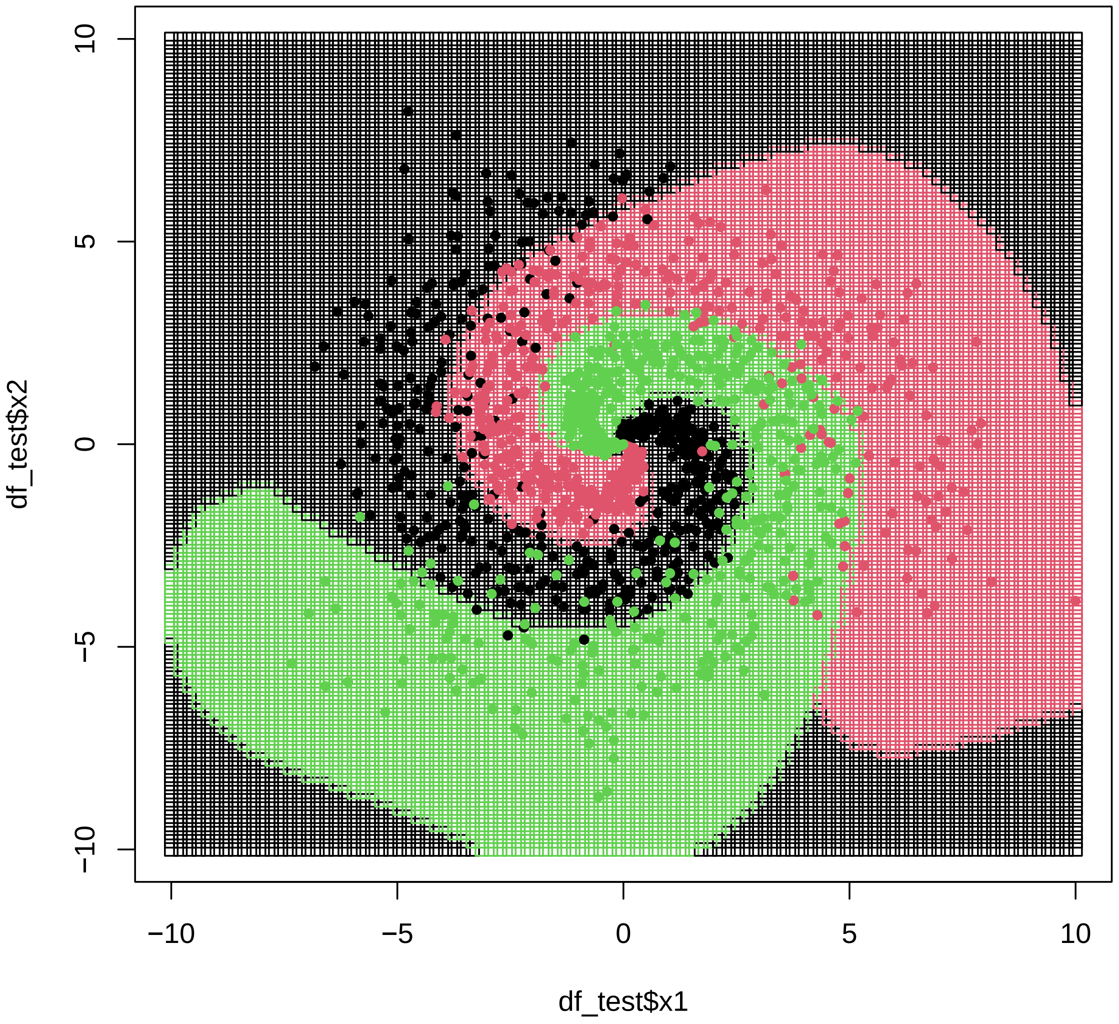
```r
plot_decision_boundary(rpart_classes)
```



3.3 (10 points)

Fit a support vector machine model to predict the `y` variable using the `x1` and `x2` predictors. Use the `svm()` function and use any kernel of your choice. Remember to set the `type` argument to `"C-classification"` **if you haven't** converted `y` to be of type `factor`.

```r
svm_fit <- svm(y ~ x1 + x2, df, kernel = "radial", type = "C-classification")
svm_classes <- predict(svm_fit, newdata = df_test)
plot_decision_boundary(svm_classes)
```



Instructions

For the next questions, you will need to fit a series of neural networks. In all cases, you can:

- set the number of units in each hidden layer to 10
- set the output dimension `o` to 3 (remember this is multinomial classification)
- use the appropriate loss function for the problem (**not `nn_bce_loss`**)
- set the number of epochs to 50
- fit the model using the `luz` package

You can use any optimizer of your choice, but you **will need to tune the learning rate for each problem**.

3.4 (10 points)

Fit a neural network with **1 hidden layer** to predict the `y` variable using the `x1` and `x2` predictors.

```
NN1 <- nn_module(
  initialize = function(p, q1, o){
    self$hidden1 <- nn_linear(p, q1)
    self$output <- nn_linear(q1, o)
    self$activation <- nn_relu()
  },
  forward = function(x){
    x %>%
      self$hidden1() %>%
      self$activation() %>%
      self$output()
  }
)

fit_1 <- NN1 %>%
  setup(
    loss = nn_cross_entropy_loss(),
    optimizer = optim_adam
  ) %>%
  set_hparams(
    p = ncol(df_test), q1 = 10, o = 3
  ) %>%
  set_opt_hparams(
    lr = 0.001
  ) %>%
  fit(
    data = list(
      df %>% select(x1, x2) %>% as.matrix,
      df$y %>% as.integer
    ),
    epochs = 50,
    dataloader_options = list(batch_size = 200, shuffle = TRUE),
    verbose = FALSE
  )
```

In order to generate the class predictions, you will need to use the `predict()` function as follows

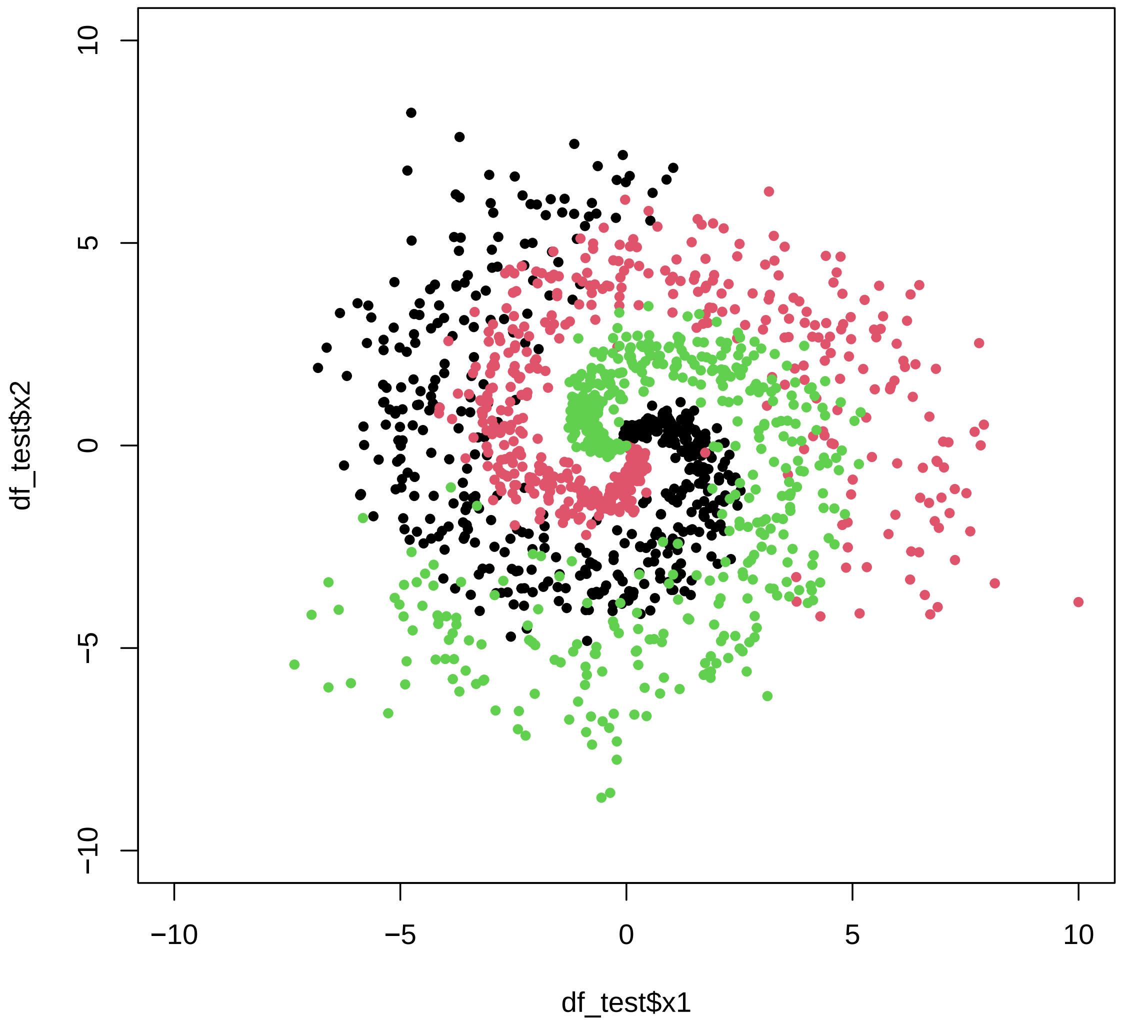
```
test_matrix <- df_test %>% select(x1, x2) %>% as.matrix

fit_1_predictions <- predict(fit_1, test_matrix) %>%
  torch_argmax(2) %>%
  as.integer()
```

Plot the results using the `plot_decision_boundary()` function.

```
plot_decision_boundary(NN1)
```

```
Warning in plot.xy(xy, type, ...): supplied color is neither numeric nor
character
```



3.5 (10 points)

Fit a neural network with **0 hidden layers** to predict the `y` variable using the `x1` and `x2` predictors.

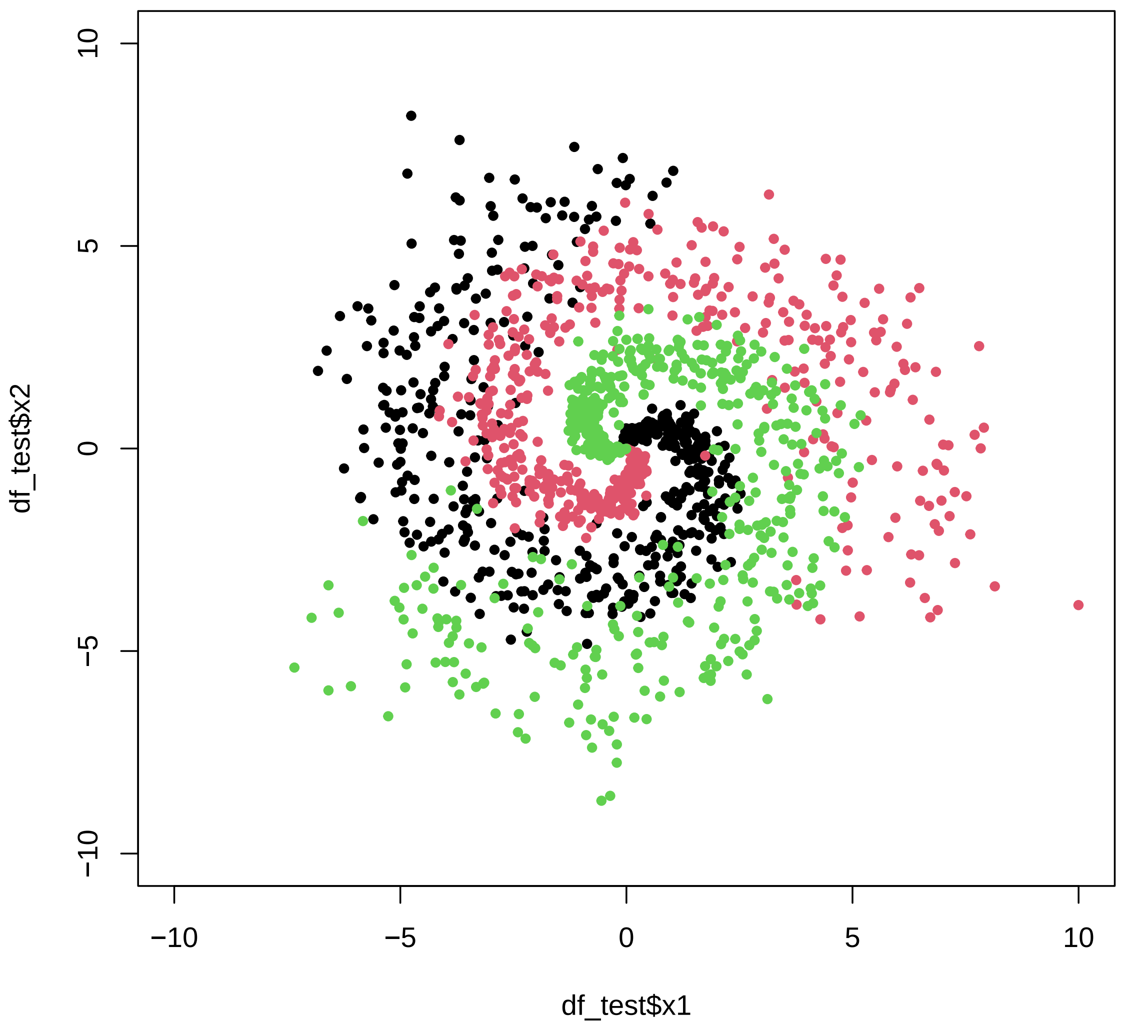
```r
NN0 <- nn_module(
  initialize = function(p, o){
    self$hidden1 <- nn_linear(p, o)
  },
  forward = function(x){
    x %>%
      self$hidden1()
  }
)

fit_0 <- NN0 %>%
  setup(
    loss = nn_cross_entropy_loss(),
    optimizer = optim_adam
  ) %>%
  set_hparams(
    p = ncol(df_test), o = 3
  ) %>%
  set_opt_hparams(
    lr = 0.001
  ) %>%
  fit(
    data = list(
      df %>% select(x1 , x2) %>% as.matrix,
      df$y %>% as.integer
    ),
    epochs = 50,
    dataloader_options = list(batch_size = 128, shuffle = TRUE),
    verbose = FALSE
  )
```

Plot the results using the `plot_decision_boundary()` function.

```r
plot_decision_boundary(NN0)
```

```
Warning in plot.xy(xy, type, ...): supplied color is neither numeric nor
character
```



3.6 (10 points)

Fit a neural network with **3 hidden layers** to predict the `y` variable using the `x1` and `x2` predictors.

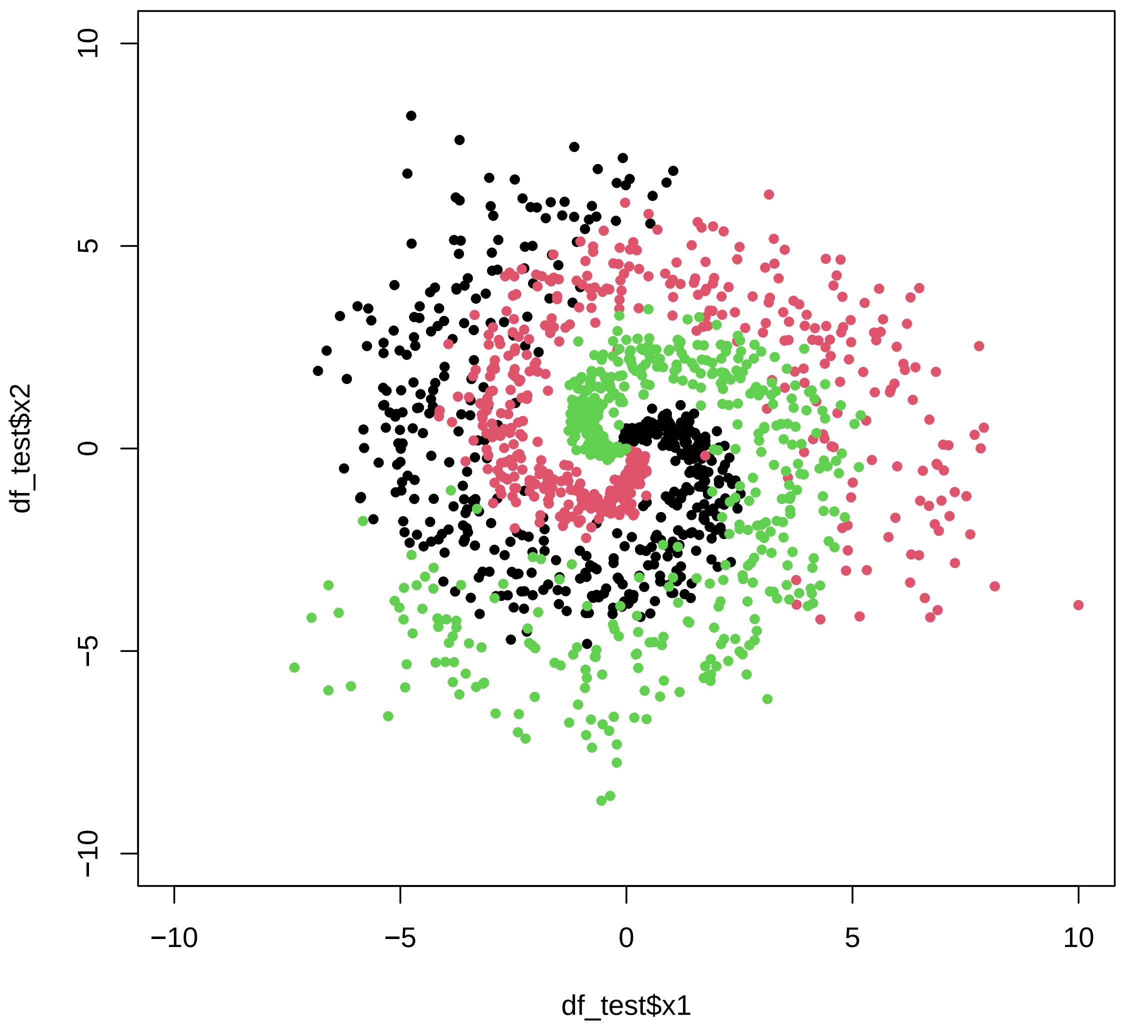
```r
NN3 <- nn_module(
  initialize = function(p , q1, q2, o){
    self$hidden1 <- nn_linear(p, q1)
    self$hidden2 <- nn_linear(q1, q2)
    self$hidden3 <- nn_linear(q2, o)
    self$activation <- nn_relu()
  },
  forward = function(x){
    x %>%
      self$hidden1() %>% self$activation() %>%
      self$hidden2() %>% self$activation() %>%
      self$hidden3()
  }
)

fit_3 <- NN3 %>%
  setup(
    loss = nn_cross_entropy_loss(),
    optimizer = optim_adam
  ) %>%
  set_hparams(
    p = ncol(df_test), q1 = 10, q2 = 10, o = 3
  ) %>%
  set_opt_hparams(
    lr = 0.001
  ) %>%
  fit(
    data = list(
      df %>% select(x1 , x2) %>% as.matrix,
      df$y %>% as.integer
    ),
    epochs = 50,
    dataloader_options = list(batch_size = 128, shuffle = TRUE),
    verbose = FALSE
  )
```

Plot the results using the `plot_decision_boundary()` function.

```r
plot_decision_boundary(NN3)
```

```
Warning in plot.xy(xy, type, ...): supplied color is neither numeric nor
character
```



3.7 (5 points)

What are the differences between the models? How do the decision boundaries change as the number of hidden layers increases?

> The differences between the models are …

\pagebreak

—

Session Information

Print your R session information using the following command

```
sessionInfo()
```

```
R version 4.2.2 (2022-10-31)
Platform: x86_64-apple-darwin17.0 (64-bit)
Running under: macOS Big Sur ... 10.16

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.2/Resources/lib/libRlapack.dylib

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

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

other attached packages:
 [1] luz_0.3.1        torch_0.9.1      e1071_1.7-13     rpart.plot_3.1.1
 [5] rpart_4.1.19     caret_6.0-94     lattice_0.20-45  ggplot2_3.4.2
 [9] corrplot_0.92    magrittr_2.0.3   broom_1.0.4      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_3.1-160         fs_1.6.1             lubridate_1.9.2
 [4] bit64_4.0.5          progress_1.2.2       tools_4.2.2
 [7] backports_1.4.1      utf8_1.2.3           R6_2.5.1
[10] colorspace_2.1-0     nnet_7.3-18          withr_2.5.0
[13] tidyselect_1.2.0     prettyunits_1.1.1    processx_3.8.0
[16] bit_4.0.5            compiler_4.2.2       cli_3.6.1
[19] scales_1.2.1         callr_3.7.3          proxy_0.4-27
[22] stringr_1.5.0        digest_0.6.31        rmarkdown_2.21
[25] coro_1.0.3           pkgconfig_2.0.3      htmltools_0.5.5
[28] parallelly_1.35.0    fastmap_1.1.1        rlang_1.1.0
```

```
[31] generics_0.1.3       jsonlite_1.8.4       ModelMetrics_1.2.2.2
[34] Matrix_1.5-1         Rcpp_1.0.10          munsell_0.5.0
[37] fansi_1.0.4          lifecycle_1.0.3      stringi_1.7.12
[40] pROC_1.18.0          yaml_2.3.7           MASS_7.3-58.1
[43] plyr_1.8.8           recipes_1.0.5        grid_4.2.2
[46] parallel_4.2.2       listenv_0.9.0        crayon_1.5.2
[49] splines_4.2.2        hms_1.1.3            zeallot_0.1.0
[52] knitr_1.42           ps_1.7.4             pillar_1.9.0
[55] future.apply_1.10.0  reshape2_1.4.4       codetools_0.2-18
[58] stats4_4.2.2         glue_1.6.2           evaluate_0.20
[61] data.table_1.14.8    renv_0.16.0-53       vctrs_0.6.1
[64] tzdb_0.3.0           foreach_1.5.2        gtable_0.3.3
[67] future_1.32.0        xfun_0.38            gower_1.0.1
[70] prodlim_2023.03.31   class_7.3-20         survival_3.4-0
[73] timeDate_4022.108    tibble_3.2.1         iterators_1.0.14
[76] hardhat_1.3.0        lava_1.7.2.1         timechange_0.2.0
[79] globals_0.16.2       ellipsis_0.3.2       ipred_0.9-14
```