Homework 5

Insert your name here

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:

```
packages <- c(
  "dplyr",
  "readr",
  "tidyr",
  "purrr",
  "broom",
  "magrittr",
  "corrplot",
  "caret",
  "rpart",
  "rpart.plot",
  "e1071",
  "torch",
  "luz"
)

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

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

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

df <- ... # Insert your code here
```

1.2 (2.5 points)

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

```
df %>% ... # Insert your code here
```

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

df_train <- ... # Insert your code here
df_test  <- ... # Insert your code here
```

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.

```
lm_fit <- ... # Insert your code here
... # Insert your code here
```

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`

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

lm_predictions <- ... # Insert your code here
```

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.

```
rpart_fit <- ... # Insert your code here
rpart_predictions <- ... # Insert your code here
```

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

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

Report the root mean squared error on the test set.

```
rpart_predictions <- ... # Insert your code here
```

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.

```
svm_fit <- ... # Insert your code here
svm_predictions <- ... # Insert your code here
```

1.8 (25 points)

Initialize a neural network model architecture:

```
NNet <- nn_module(
    initialize = function(p, q1, q2, q3){
      ... # Insert your code here
    },
    forward = function(x){
      ... # Insert your code here
    }
)
```

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.

```
nnet_fit <- NNet %>%
  setup(
    ... # Insert your code here
  ) %>%
  set_hparams(
    ... # Insert your code here
  ) %>%
  set_opt_params(
    ... # Insert your code here
  ) %>%
  fit(
    ... # Insert your code here
    dataloader_options = ... # Insert your code here
    verbose = FALSE # Change to TRUE while tuning. But, set to FALSE before submitting

  )
```

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

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

Report the root mean squared error on the test set.

```
nnet_predictions <- ... # Insert your code here
```

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?

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

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

```
df <- ... # Insert your code here
```

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
)

df_train <- ... # Insert your code here
df_test  <- ... # Insert your code here
```

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 <- ... # Insert your code here
  error <- ... # Insert your code here
  true_positives <- ... # Insert your code here
  true_negatives <- ... # Insert your code here
  false_positives <- ... # Insert your code here
  false_negatives <- ... # Insert your code here
  true_positive_rate <- ... # Insert your code here
  false_positive_rate <- ...  # Insert your code here
  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 <- ... # Insert your code here
glm_classes <- ... # Insert your code here
```

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_classes <- ... # Insert your code here
```

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

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

Report the prediction accuracy on the test set.

```r
rpart_classes <- ... # Insert your code here
```

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`.

```r
svm_fit <- ... # Insert your code here
```

Report the prediction accuracy on the test set.

```r
svm_classes <- ... # Insert your code here
```

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.

```r
nnet_fit <- NNet %>%
  setup(
    ... # Insert your code here
  ) %>%
  set_hparams(
    ... # Insert your code here
  ) %>%
  set_opt_params(
    ... # Insert your code here
  ) %>%
  fit(
    ... # Insert your code here
    dataloader_options = ... # Insert your code here
    verbose = FALSE # Change to TRUE while tuning. But, set to FALSE before submitting

  )
```

2.7 (5 points)

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

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

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:

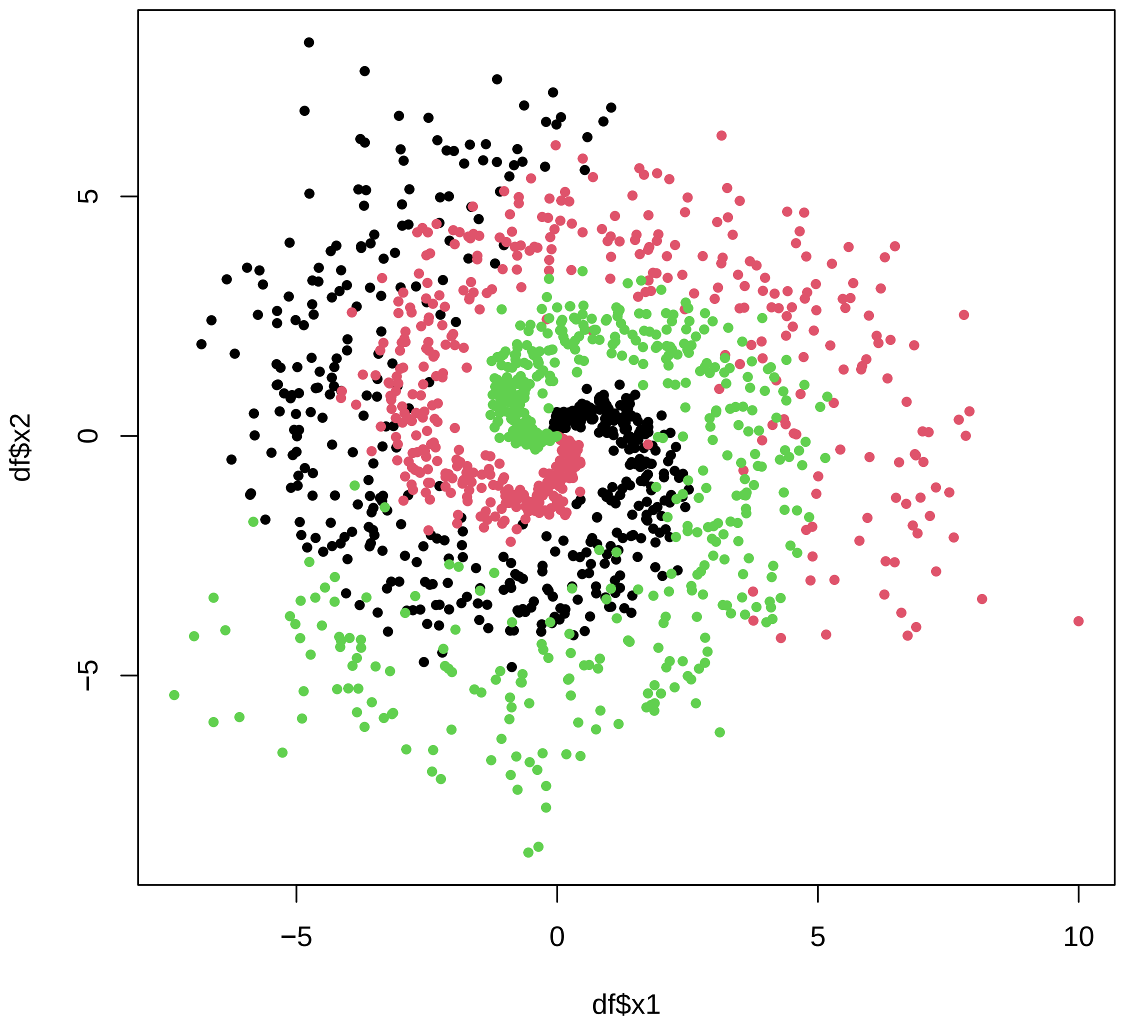
```r
generate_three_spirals <- function(){
  set.seed(42)
  n <- 500
  noise <- 0.2
  t <- (1:n) / n * 2 * pi
  x1 <- c(
      t * (sin(t) + rnorm(n, 0, noise)),
      t * (sin(t + 2 * pi/3) + rnorm(n, 0, noise)),
      t * (sin(t + 4 * pi/3) + rnorm(n, 0, noise))
    )
  x2 <- c(
      t * (cos(t) + 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 <- ... # Insert your code here
df_test <- ... # Insert your code here
```

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 <- ... # Insert your code here
rpart_classes <- ... # Insert your code here
```

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 <- ... # Insert your code here
svm_classes <- ... # Insert your code here
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 100
- 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$hidden1 <- ... # Insert your code here
    self$output <- ... # Insert your code here
    self$activation <- ... # Insert your code here
  },
  forward = function(x){
    x %>%
      self$hidden1() %>%
      self$activation() %>%
      self$output()
  }
)

fit_1 <- NN1 %>%
  setup(
    ... # Insert your code here
  ) %>%
  set_hparams(
    ... # Insert your code here
  ) %>%
  set_opt_params(
    ... # Insert your code here
```

```r
  ) %>%
  fit(
    data = list(
      df %>% select(x1, x2) %>% as.matrix,
      df$y %>% as.integer
    ),
    ... # Insert your code here
    dataloader_options = ... # Insert your code here
    verbose = FALSE
  )
```

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

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

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

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

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){
    ... # Insert your code here
  },
  forward = function(x){
    x %>%
    ... # Insert your code here
  }
)

fit_0 <- NN0 %>%
  setup(...) %>%
  set_hparams(...) %>%
  set_opt_params(...) %>%
  fit(...)
```

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

3.6 (10 points)

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

```
NN2 <- nn_module(
  initialize = function(p, q1, q2, o){
    ... # Insert your code here
  },
  forward = function(x){
    x %>%
    ... # Insert your code here
  }
)

fit_2 <- NN3 %>%
  setup(...) %>%
  set_hparams(...) %>%
  set_opt_params(...) %>%
  fit(...)
```

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

3.7 (5 points)

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

Session Information

Print your R session information using the following command

```
sessionInfo()
```

```
R version 4.2.2 (2022-10-31)
Platform: x86_64-conda-linux-gnu (64-bit)
Running under: Storage

Matrix products: default
BLAS/LAPACK: /gpfs/group/hvt5139/default/tools/mambaforge/lib/libopenblasp-r0.3.21.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

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

other attached packages:
 [1] luz_0.3.1         torch_0.9.0       e1071_1.7-13      rpart.plot_3.1.1
 [5] rpart_4.1.19      caret_6.0-94      lattice_0.20-45   ggplot2_3.4.1
 [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_1.1.0

loaded via a namespace (and not attached):
 [1] nlme_3.1-162         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_3.6.0
[19] scales_1.2.1         callr_3.7.3          proxy_0.4-27
[22] stringr_1.5.0        digest_0.6.31        rmarkdown_2.20
[25] coro_1.0.3           pkgconfig_2.0.3      htmltools_0.5.4
```

```
[28] parallelly_1.35.0    fastmap_1.1.0        rlang_1.0.6
[31] generics_0.1.3       jsonlite_1.8.4       ModelMetrics_1.2.2.2
[34] Matrix_1.5-3         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_7.3-58.3
[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_1.1.2            zeallot_0.1.0
[52] knitr_1.42           ps_1.7.2             pillar_1.8.1
[55] future.apply_1.10.0  reshape2_1.4.4       codetools_0.2-19
[58] stats4_4.2.2         glue_1.6.2           evaluate_0.20
[61] data.table_1.14.8    renv_0.16.0-53       vctrs_0.5.2
[64] tzdb_0.3.0           foreach_1.5.2        gtable_0.3.1
[67] future_1.32.0        xfun_0.37            gower_1.0.1
[70] prodlim_2019.11.13   class_7.3-21         survival_3.5-5
[73] timeDate_4022.108    tibble_3.1.8         iterators_1.0.14
[76] hardhat_1.2.0        lava_1.7.2.1         timechange_0.2.0
[79] globals_0.16.2       ellipsis_0.3.2       ipred_0.9-14
```