# Homework 5

Johnny Lydon

2024-04-10

## HW5

packages <- c( “tibble”, “dplyr”, “readr”, “tidyr”, “purrr”, “broom”, “magrittr”, “corrplot”, “caret”, “rpart”, “rpart.plot”, “e1071”, “luz” )

#renv::install(packages) install.packages(packages)

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

### Question 1

```
#1.1

path <- "data/housing.csv"

df <- read_csv(path)

df <- df %>%
  mutate(across(where(is.character), as.factor)) %>%
  rename_all(tolower) %>%
  drop_na()
```

```
#1.2

df %>% select_if(is.numeric) %>%
  cor() %>%
  corrplot()
```

```
#1.3

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

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

```r
#1.4

lm_fit <- lm(median_house_value ~ latitude +
               longitude +
               housing_median_age +
               total_rooms +
               total_bedrooms +
               population +
               median_income +
               ocean_proximity, data = df_train)

summary(lm_fit)
```

```r
#1.5

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

lm_predictions <- predict(lm_fit, newdata = df_test)

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

```r
#1.6

rpart_fit <- rpart(median_house_value ~ latitude +
                     longitude +
                     housing_median_age +
                     total_rooms +
                     total_bedrooms +
                     population +
                     median_income +
                     ocean_proximity, data = df_train)

rpart_predictions <- predict(rpart_fit, newdata = df_test)

rpart.plot(rpart_fit)

rpart_predictions <- predict(rpart_fit, newdata = df_test)
rmse_rpart_predictions <- rmse(df_test$median_house_value, rpart_predictions)
print(rmse_rpart_predictions)
```

```r
#1.7

svm_fit <- svm(median_house_value ~ latitude +
                 longitude +
                 housing_median_age +
                 total_rooms +
                 total_bedrooms +
                 population +
                 median_income +
                 ocean_proximity, data = df_train)
```

```r
svm_predictions <- predict(svm_fit, newdata = df_test)

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

```r
#1.8

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()
    }
)

nnet_fit <- NNet %>%
  setup(loss = nn_bce_loss(),
        opt = optim_adam,
        metrics = list(luz_metric_accuracy())) %>%
  set_hparams(p= 12,
              q1 = 32,
              q2 = 16,
              q3 = 8) %>%
  set_opt_params(lr = 0.02) %>%
  fit(data = list(covariate_matrix, df_train %>%
                    select(median_house_value) %>%
                    as.matrix),
      valid_data = list(model.matrix(median_house_value ~ 0 + ., data =  df_test),
                        df_test %>%
                          select(median_house_value) %>%
                          as.matrix),
      epochs  = 10,
      verbose = TRUE)

nnet_predictions <- predict(nnet_fit, data = df_test)

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

```r
#1.9

model_rmse_table <- data.frame(Models = c("Linear Regression", "Decision Tree", "SVM", "Neural Network"
                               RMSE = c(rmse_lm_predictions, rmse_rpart_predictions, rmse_svm_predictio

print(model_rmse_table)
```

## Question 2

```
#2.1

path2 <- "data/spambase.csv"

df2 <- read.csv(path2)

df2 <- df2 %>%
  mutate_if(is.character, as.factor) %>%
  mutate_if(is.numeric, as.numeric) %>%
  rename_all(tolower) %>%
  drop_na()
```

```
#2.2

set.seed(42)
test_ind2 <- sample(
  1:nrow(df2),
  floor( nrow(df2)/10 ),
  replace=FALSE
)

df2_train <- df2[-test_ind2, ]
df2_test <- df2[test_ind2, ]

overview <- function(pred_class, true_class) {
  accuracy <- sum(pred_class == true_class) / length(true_class)
  error <- 1 - 'accuracy'
  true_positives <- sum(pred_class == 1 & true_class == 1)
  true_negatives <- sum(pred_class == 0 & true_class == 0)
  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 / (false_positives + true_negatives)
  return(
    data.frame(
      accuracy = accuracy,
      error = error,
      true_positive_rate = true_positive_rate,
      false_positive_rate = false_positive_rate
    )
  )
}
```

```
#2.3

glm_fit <- glm(spam ~ ., data = df2_train, family = binomial)
glm_classes <- glm_classes <- predict(glm_fit, newdata = df2_test, type = "response")

glm_prediction <- ifelse(glm_classes > 0.5, 1, 0)

glm_prediction_accuracy <- overview(glm_prediction, df2_test$spam)
```

```r
#2.4

rpart2 <- rpart(spam ~ ., data = df2_train, method = 'class')

rpart2_prediction <- predict(rpart2, newdata = df2_test, type = 'class')

rpart.plot(rpart2)

rpart2_classes <- overview(rpart2_prediction, df2_test$spam)
```

```r
#2.5

svm2_fit <- svm(spam ~. , data = df_train2, type = "C-classification")

svm2_prediction <- predict(svm2_fit, newdata = df2_test, type  = "class")

svm2_classes <- overview(svm2_prediction, df2_test$spam)
```

```r
#2.6

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()
    }
)

nnet2_fit <- NNet %>%
  setup(loss = nn_bce_loss(),
        optimizer = optim_adam,
        metrics = list(luz_metric_accuracy()) %>%
          set_hparams(p = ncol(M),
                      q1 = 32,
                      q2 = 16,
                      q3 = 8) %>%
          set_opt_params(lr = 0.02) %>%
          fit(data = list(model.matrix(spam~ 0 +., data = df2_train),
                df2_train %>%
                  select(spam) %>%
                  as.matrix),
    valid_data = list(model.matrix(spam ~ 0+., data = df2_test),
                      df2_test %>%
                        select(spam) %>%
```

```
                          as.matrix),
    epochs = 10,
    verbose = TRUE))

nnet2_predictions <- predict(nnet2_fit, data = df2_test)

nnet2_classes <- overview(df2_test$spam, nnet2_predictions)
```

```
#2.7

model_accuracy_table <- data.frame(Models = c("Linear Regression", "Decision Tree", "SVM", "Neural Netw
                      Prediction Accuracy = c(glm_prediction_accuracy, rpart2_classes, svm2_classes, nne

print(model_accuracy_table)
```

## Question 3

```
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::tibble(x1=x1, x2=x2, y=y))
}
```

```
#3.1

df3 <- generate_three_spirals()

plot(
  df3$x1, df3$x2,
  col = df3$y,
  pch = 20
)
```

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

grid <- expand.grid(x1 = x1, x2 = x2)

df3_test <- as_tibble(grid)
```

```r
#3.2

rpart3_fit <- rpart(y ~ ., data = df3, method = "class")
rpart3_classes <- predict(rpart3_fit, newdata = df3_test, type = "class")

plot_decision_boundary <- function(predictions){
  plot(
    df3_test$x1, df3_test$x2,
    col = predictions,
    pch = 0
  )
  points(
    df3$x1, df3$x2,
    col = df3$y,
    pch = 20
  )
}

plot_decision_boundary(rpart3_classes)
```

```r
#3.3

svm3_fit <- svm(y~., data = df3, kernel = 'radial', type = 'C-classification')
svm3_classes <- predict(svm3_fit, newdata = df3_test)

plot_decision_boundary(svm3_classes)
```

```r
#3.4

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,
        metrics = list(luz_metric_accuracy())) %>%
  set_hparams(p = 2,
              q1 = 10,
              o = 3) %>%
  set_opt_params(lr = 0.02) %>%
  fit(
    data = list(
      df3 %>% select(x1, x2) %>% as.matrix,
      df3$y %>% as.integer
    ),
    epochs = 50,
    verbose = TRUE)

test_matrix <- df3_test %>% select(x1, x2) %>% as.matrix

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

```
#3.5

NN0 <- nn_module(
  initialize = function(p, o){
    self$output = nn_linear(p, o)
    self$activation <- nn_relu()
  },
  forward = function(x){
    x %>%
      self$activation() %>%
      self$output()
  }
)

fit_0 <- NN0 %>%
  setup(loss = nn_cross_entropy_loss(),
        opt = optim_adam,
        metric = luz_metric_accuracy()) %>%
  set_hparams(p=2, o = 3) %>%
  set_opt_params(lr = 0.02) %>%
  fit(data = list(
     df3 %>%
       select(x1,x2) %>%
       as.matrix, df3 %>%
       select(y)
    ),
    valid = list(
            df3_test %>% select(x1,x2) %>% as.matrix,
            df3_test %>% select(y)),
    epochs = 50,
    verbose = TRUE)
```

```
#3.6

NN2 <- nn_module(
  initialize = function(p, q1, q2, o){
    self$hidden1 <- nn_linear(p,q1)
    self$hidden2 <- nn_linear(q1, q2)
    self$output <- nn_linear(q2,o)
    self$activation <- nn_relu()
  },
  forward = function(x){
    x %>%
      self$hidden1() %>%
      self$activation() %>%
      self$hidden2() %>%
      self$activation() %>%
      self$output()
  }
)

fit_2 <- NN3 %>%
  setup(loss = nn_cross_entropy_loss(),
        opt = optim_adam,
        metric = luz_metric_accuracy()) %>%
  set_hparams(
    p = 2,
    q1 = 10,
    q2 = 10,
    o = 3
  ) %>%
  set_opt_params(lr = 0.02) %>%
  fit(data = list(
      model.matrix(y ~ 0 + ., data = df3), df3 %>%
        select(y) %>% as.matrix),
      epochs = 50,
      verbose = TRUE)
```

```
#3.7

#When there is more hidden layers, the models become more accurate.
```

Just going to put a little note at the end here. I was doing fine with the homework, until I had a problem with a function where it just refused to recognize that the function even existed. I assumed that there was some sort of problem with the packages from the beginning, so I tried to reload the packages just in case. I ended up having to restart the whole program, which somehow screwed up everything. So now nothing really works and I'm quite frustrated because I was just going over what I hadn't finished. So now I'm forced to put eval = FALSE for every question or else I can't knit the homework. I'm not sure what happened and it's honestly pretty frustrating. Sorry :(