Homework 8

Brady Lamson

2022-04-16

Chapter 11

Problem 6 (Neural Network)

I'll be utilizing tidymodels for this exercise.

```
# Set consistent seed
set.seed(500)

# Read in data, select variables, scale variables
sleep <-
    NHANES::NHANES %>%
    select(SleepTrouble, Age, Weight, HHIncomeMid, BMI, Pulse) %>%
    na.omit()

# Split data into training/testing with 75/25 split (requirement for part C).
# Strata = SleepTrouble to aid in unbalanced data
sleep_split <-
    sleep %>%
    rsample::initial_split(.75, strata = SleepTrouble)

sleep_train <-
    sleep_split %>%
    rsample::training()

sleep_test <-
    sleep_split %>%
    rsample::testing()
```

```
# Create recipe
nn_rec <-
    recipe(SleepTrouble ~ ., data = sleep_train) %>%    # Set all variables as predictors
    step_normalize(all_predictors())                    # Set data on a common scale

# Create a neural net model
nn_model <-
    mlp(hidden_units = 7) %>%
    set_mode("classification") %>%
    set_engine("nnet")
```

```r
# Create workflow
nn_workflow <-
    workflow() %>%
    add_model(nn_model) %>%
    add_recipe(nn_rec)
```

```r
# Fit the workflow to training data
nn_fit <- fit(nn_workflow, sleep_train)

# Bind predictions to actual training values for comparison
results <-
    sleep_train %>%
    select(SleepTrouble) %>%
    dplyr::bind_cols(
        predict(nn_fit, sleep_train),
        predict(nn_fit, sleep_train, type = "prob")

    )
```

```r
# See first few predictions for True Yes
results %>%
    filter(SleepTrouble == "Yes") %>%
    slice(1:5)
```

```
## # A tibble: 5 x 4
##   SleepTrouble .pred_class .pred_No .pred_Yes
##   <fct>        <fct>          <dbl>     <dbl>
## 1 Yes          No             0.633     0.367
## 2 Yes          No             0.633     0.367
## 3 Yes          No             0.537     0.463
## 4 Yes          No             0.561     0.439
## 5 Yes          No             0.516     0.484
```

```r
# Create confusion matrix
results %>%
    conf_mat(truth = SleepTrouble, .pred_class)
```

```
##           Truth
## Prediction   No  Yes
##        No  3728 1193
##        Yes   60  121
```

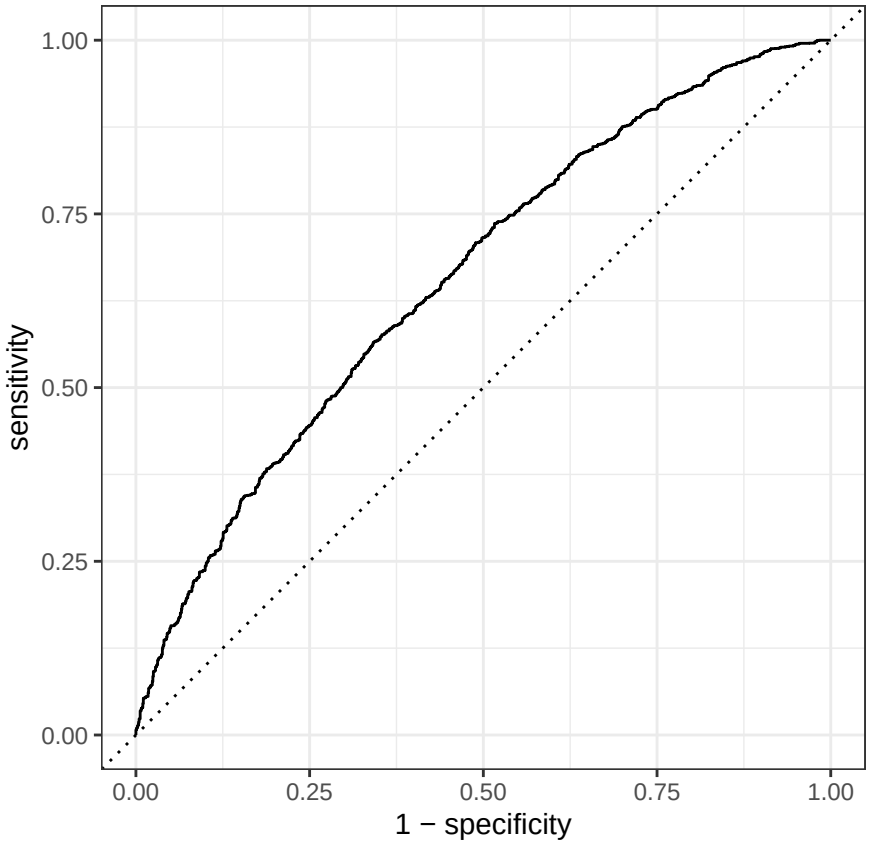
```r
# Evaluate accuracy on training set
results %>%
    accuracy(truth = SleepTrouble, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.754
```

```
# Evaluate area under roc curve
results %>%
    roc_auc(SleepTrouble, .pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.658
```

```
# Visualize the roc curve
results %>%
    roc_curve(SleepTrouble, .pred_No) %>%
    autoplot()
```



The results from the training data are not promising, and the roc_curve is pretty poor as well. Typically, you want the curve to hug the top left corner as much as possible. We can see from the results .pred_Yes that our model is a bit apprehensive about guessing Yes. From the confusion matrix we see it misses the bulk of the people who genuinely have sleep trouble. We'll have to see how it performs on the test data.

```
# Fit the workflow to training data
nn_fit <- fit(nn_workflow, sleep_test)

# Bind predictions to actual training values for comparison
results <-
    sleep_test %>%
```

```
    select(SleepTrouble) %>%
    dplyr::bind_cols(
        predict(nn_fit, sleep_test),
        predict(nn_fit, sleep_test, type = "prob")

    )
```

```
# See first few predictions for True Yes
results %>%
    filter(SleepTrouble == "Yes") %>%
    slice(1:5)
```

```
## # A tibble: 5 x 4
##   SleepTrouble .pred_class .pred_No .pred_Yes
##   <fct>        <fct>          <dbl>     <dbl>
## 1 Yes          No             0.647     0.353
## 2 Yes          Yes            0.441     0.559
## 3 Yes          No             0.659     0.341
## 4 Yes          No             0.659     0.341
## 5 Yes          No             0.628     0.372
```

```
# Create confusion matrix
results %>%
    conf_mat(truth = SleepTrouble, .pred_class)
```

```
##           Truth
## Prediction   No  Yes
##        No  1224  381
##        Yes   39   58
```

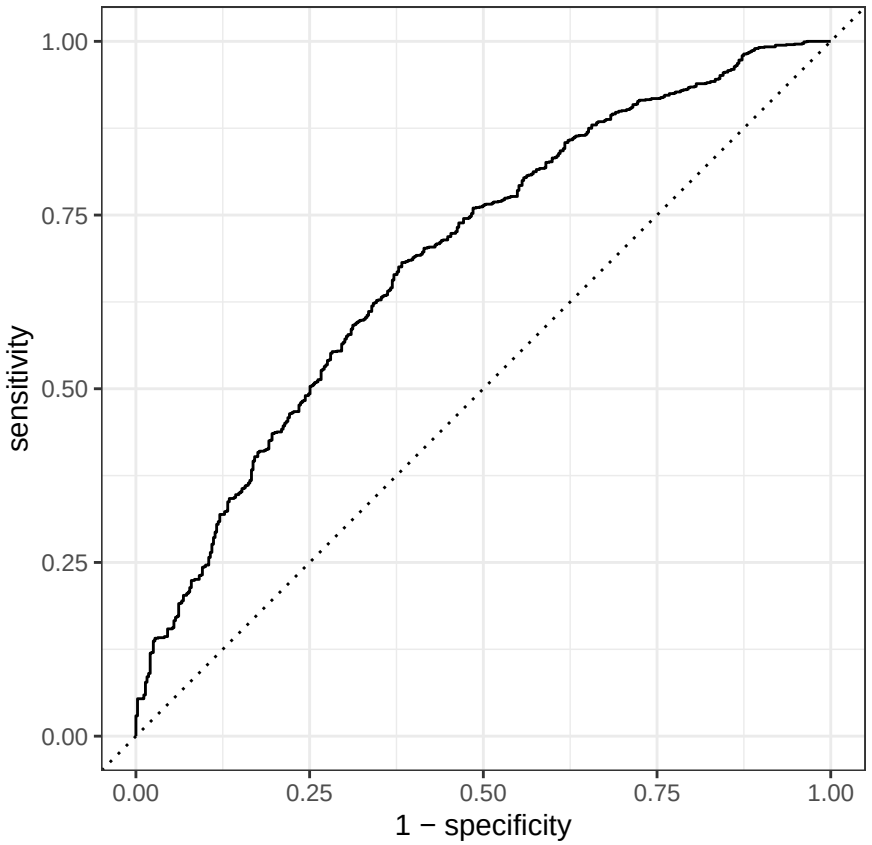
```
# Evaluate accuracy on training set
results %>%
    accuracy(truth = SleepTrouble, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.753
```

```
# Evaluate area under roc curve
results %>%
    roc_auc(SleepTrouble, .pred_No)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.690
```

```
# Visualize the roc curve
results %>%
    roc_curve(SleepTrouble, .pred_No) %>%
    autoplot()
```



The model still under-performs, but there is good news. The metrics are largely the same between the training and testing data sets. We at least have consistency with our model, it just isn't very good. I likely picked mediocre predictors. This model isn't awful, it just isn't great. It's possible that Age, Weight, Income, BMI and pulse are simply just *okay* predictors of sleep trouble and likely should be included in a better model. I likely missed the really good predictors is all, especially as I was trying to dodge as many NAs as I could with my variable selection. There are also hyper-parameters I never bothered to tune. I tinkered with hidden_units and saw a good improvement in performance going to 7, but it's possible better values exist. A grid search would be great for this and the other hyper-parameters. There's a lot of room for improvement here.

Chapter 12

Problem 6

```
hof <-
  Lahman::Batting %>%
  group_by(playerID) %>%
  inner_join(Lahman::HallOfFame, by = c("playerID" = "playerID")) %>%
  filter(inducted == "Y" & votedBy == "BBWAA") %>%
  summarize(tH = sum(H), tHR = sum(HR), tRBI = sum(RBI), tSB = sum(SB)) %>%
  filter(tH > 1000)
```

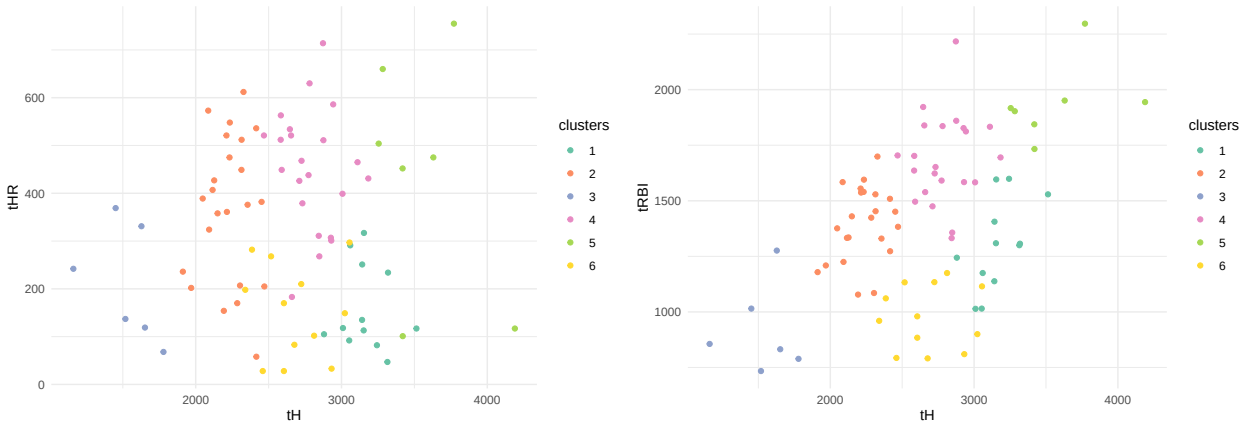
```
clusters <-
  hof %>%
  select(-playerID) %>%
  kmeans(centers = 6) %>%
  fitted("classes") %>%
  as.character()
```

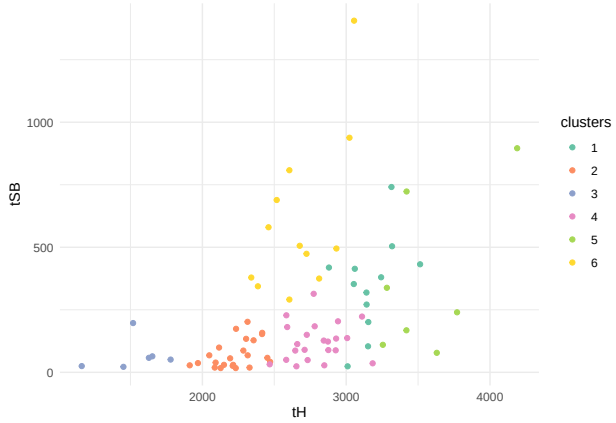
```
hof_clusters <-
  hof %>%
  mutate(clusters = clusters)
```

```
cluster_1 <-
  hof_clusters %>%
  ggplot(aes(x = tH, y = tHR)) +
  geom_point(aes(color = clusters))  +
  scale_color_brewer(palette = "Set2") +
  theme_minimal()
```

Note, I did 3 of these plots but the code was largely identical so I will spare you that.

```
cluster_1
cluster_2
cluster_3
```





What we see here is pretty consistent clustering. `tH vs tHR` is the messiest of the groupings, but the others are quite distinct. I definitely think total hits plays a big part into where everything else falls into place.

Problem 6 (with hclust())

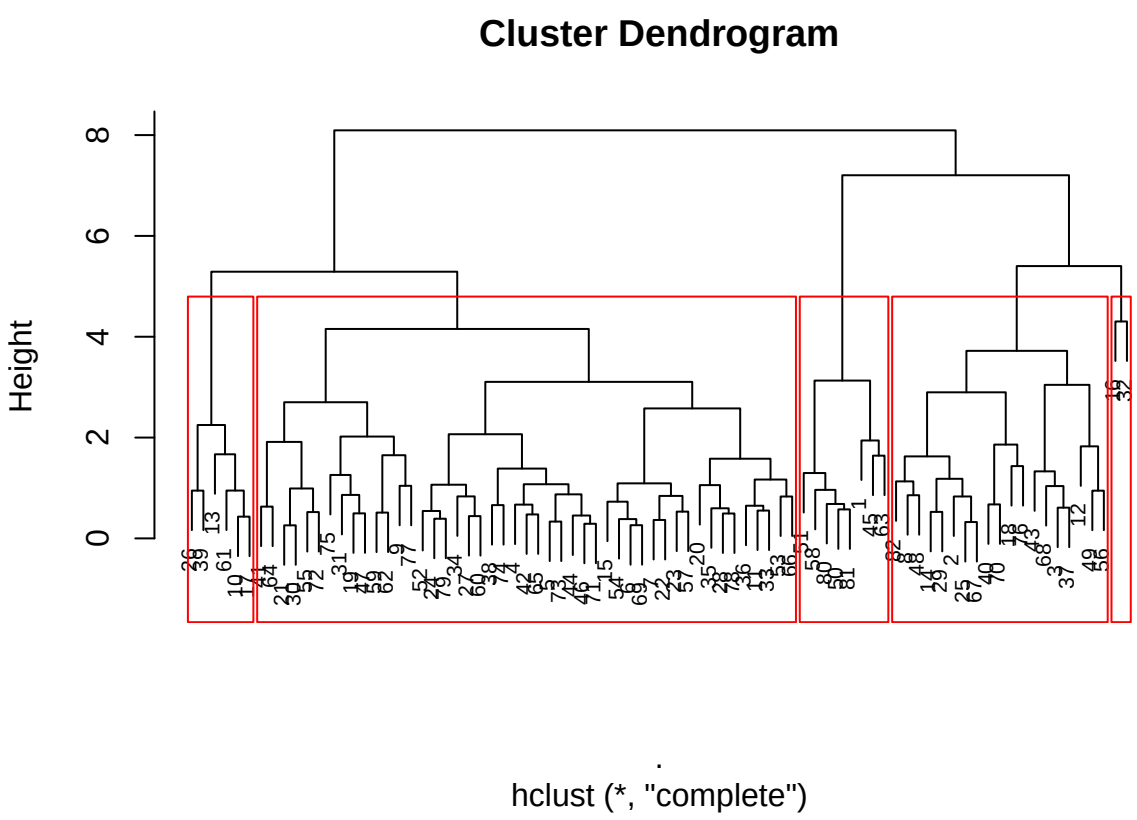
```
my_scale <- function(x) {
  (x - mean(x)) / sd(x)
}

# Scaling the data to get it on the same scale
hof_scaled <-
  hof %>%
  mutate(
    tH = my_scale(tH),
    tHR = my_scale(tHR),
    tRBI = my_scale(tRBI),
    tSB = my_scale(tSB)
  )
```

```
hof_hclust <-
  hof_scaled %>%
  dist(method = "euclidian") %>%
  hclust()
```

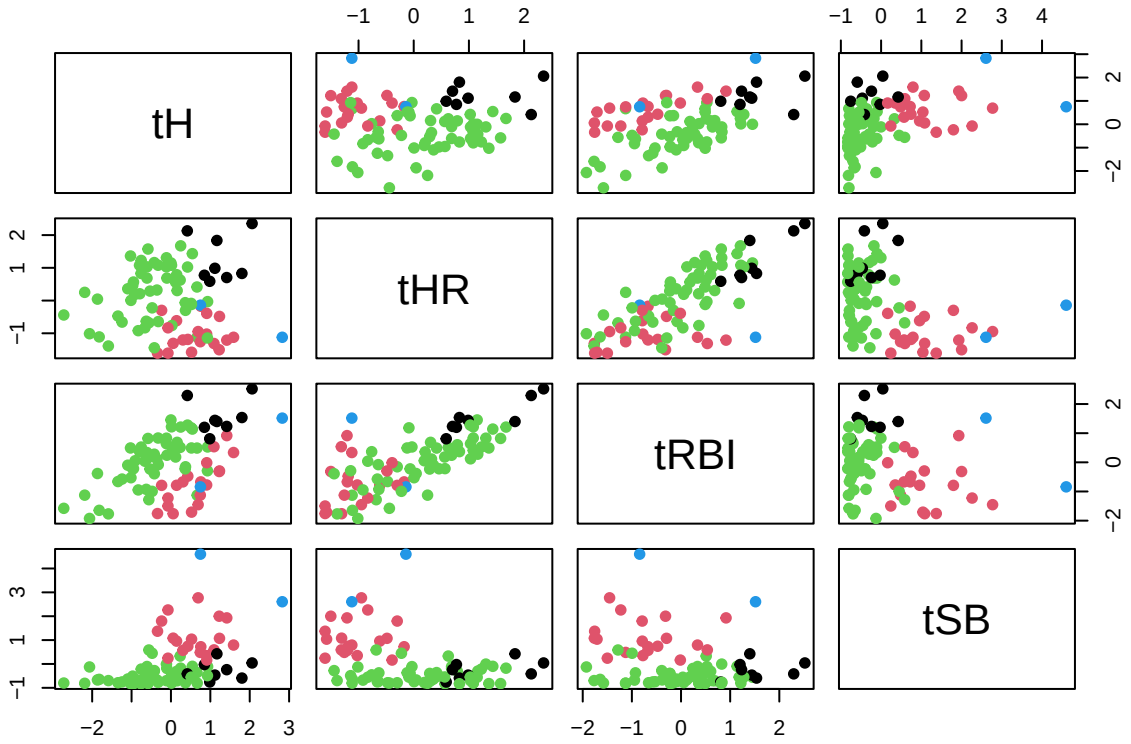
```
## Warning in dist(., method = "euclidian"): NAs introduced by coercion
```

```
plot(hof_hclust, cex = 0.7)
rect.hclust(hof_hclust, k = 5, border = "red")
```




```
hclusters <-
  cutree(hof_hclust, k = 4)
```

```
hof_scaled %>%
  select(-playerID) %>%
  pairs(col = hclusters, pch = 19)
```



I used `k=4` here because, based on the dendogram, it seemed like a reasonable value. It seems to work well, each of those 4 groups is pretty distinct for the most part! The most distinction tends to appear when total hits is involved and total runs batted in appears the messiest of all the variables.