Homework 04

STAT 430, Fall 2017

Due: Friday, October 6, 11:59 PM

Exercise 1 (Comparing Classifiers)

```
library(tibble)
library(readr)

make_hw04_data = function(n_obs = 1000) {
  x1 = runif(n = n_obs, min = -1, max = 1)
  x2 = runif(n = n_obs, min = -1, max = 1)
  prob = ifelse(x2 > (x1 + 1) ^ 2,
                yes = 0.90,
                no = ifelse(x2 < -(x1 - 1) ^ 2,
                            yes = 0.90,
                            no = 0.05))
  y = rbinom(n = n_obs, size = 1, prob = prob)
  y = factor(ifelse(y == 1, "dodgerblue", "darkorange"))
  data.frame(x1, x2, y)
}

# generate datasets
set.seed(42)
hw04_trn_data = make_hw04_data(n_obs = 500)
hw04_tst_data = make_hw04_data(n_obs = 2000)

# write to files
write_csv(hw04_trn_data, "hw04-trn-data.csv")
write_csv(hw04_tst_data, "hw04-tst-data.csv")

# clean up
rm(hw04_trn_data)
rm(hw04_tst_data)
```

[**8 points**] This exercise will use data in `hw04-trn-data.csv` and `hw04-tst-data.csv` which are train and test datasets respectively. Both datasets contain multiple predictors and a categorical response `y`.

The possible values of `y` are `"dodgerblue"` and `"darkorange"` which we will denote mathematically as B (for blue) and O (for orange).

Consider four classifiers.

$$\hat{C}_1(x) = \begin{cases} B & x_1 > 0 \\ O & x_1 \le 0 \end{cases}$$

$$\hat{C}_2(x) = \begin{cases} B & x_2 > x_1 + 1 \\ O & x_2 \le x_1 + 1 \end{cases}$$

$$\hat{C}_3(x) = \begin{cases} B & x_2 > x_1 + 1 \\ B & x_2 < x_1 - 1 \\ O & \text{otherwise} \end{cases}$$

$$\hat{C}_4(x) = \begin{cases} B & x_2 > (x_1 + 1)^2 \\ B & x_2 < -(x_1 - 1)^2 \\ O & \text{otherwise} \end{cases}$$

Obtain train and test error rates for these classifiers. Summarize these results using a single well-formatted table.

- Hint: Write a function for each classifier.
- Hint: The `ifelse()` function may be extremely useful.

Solution:

```r
# read in data
trn_data = read.csv("hw04-trn-data.csv")
tst_data = read.csv("hw04-tst-data.csv")
```

```r
C1 = function(data) {
  with(data, ifelse(x1 > 0, yes = "dodgerblue", no = "darkorange"))
}
```

```r
C2 = function(data) {
  with(data, ifelse(x2 > x1 + 1, yes = "dodgerblue", no = "darkorange"))
}
```

```r
C3 = function(data) {
  with(data, ifelse(x2 > x1 + 1,
                    yes = "dodgerblue",
                    no = ifelse(x2 < x1 - 1,
                                yes = "dodgerblue",
                                no = "darkorange")))
}
```

```r
C4 = function(data) {
  with(data, ifelse(x2 > (x1 + 1) ^ 2,
                    yes = "dodgerblue",
                    no = ifelse(x2 < -(x1 - 1) ^ 2,
                                yes = "dodgerblue",
                                no = "darkorange")))
}
```

```r
# create function to calculate error rates for written classifiers
calc_classifier_error = function(classifier, data) {
  mean(data$y != classifier(data))
}
```

```r
classifiers = list(C1, C2, C3, C4)

results = data.frame(
  c("`C1`", "`C2`", "`C3`", "`C4`"),
  sapply(classifiers, calc_classifier_error, data = trn_data),
  sapply(classifiers, calc_classifier_error, data = tst_data)
)
```

```r
colnames(results) = c("Classifier", "Train Error Rate", "Test Error Rate")
knitr::kable(results)
```

Classifier	Train Error Rate	Test Error Rate
C1	0.468	0.5160
C2	0.216	0.2240
C3	0.096	0.1270
C4	0.050	0.0665

Exercise 2 (Creating Classifiers with Logistic Regression)

[**8 points**] We'll again use data in `hw04-trn-data.csv` and `hw04-tst-data.csv` which are train and test datasets respectively. Both datasets contain multiple predictors and a categorical response `y`.

The possible values of `y` are `"dodgerblue"` and `"darkorange"` which we will denote mathematically as B (for blue) and O (for orange).

Consider classifiers of the form

$$\hat{C}(x) = \begin{cases} B & \hat{p}(x) > 0.5 \\ O & \hat{p}(x) \leq 0.5 \end{cases}$$

Create (four) classifiers based on estimated probabilities from four logistic regressions. Here we'll define $p(x) = P(Y = B \mid X = x)$.

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0$$

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2$$

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2$$

Note that, internally in `glm()`, `R` considers a binary factor variable as `0` and `1` since logistic regression seeks to model $p(x) = P(Y = 1 \mid X = x)$. But here we have `"dodgerblue"` and `"darkorange"`. Which is `0` and which is `1`? Hint: Alphabetically.

Obtain train and test error rates for these classifiers. Summarize these results using a single well-formatted table.

```r
mod_1 = glm(y ~ x1, data = trn_data, family = "binomial")
mod_2 = glm(y ~ x1 + x2, data = trn_data, family = "binomial")
mod_3 = glm(y ~ x1 + x2 + I(x1^2) + I(x2^2), data = trn_data, family = "binomial")
mod_4 = glm(y ~ x1 * x2 + I(x1^2) + I(x2^2), data = trn_data, family = "binomial")
```

```r
# create function to calculate error rates for trained logistic regressions
calc_lr_error = function(model, data) {
  predicted = ifelse(predict(model, data) > 0.5,
                     yes = "dodgerblue",
                     no = "darkorange")
  mean(data$y != predicted)
}
```

```r
models = list(mod_1, mod_2, mod_3, mod_4)
results = data.frame(
  c("`mod1`", "`mod2`", "`mod3`", "`mod4`"),
  sapply(models, calc_lr_error, data = trn_data),
  sapply(models, calc_lr_error, data = tst_data)
)
colnames(results) = c("Model", "Train Error Rate", "Test Error Rate")
knitr::kable(results)
```

Model	Train Error Rate	Test Error Rate
`mod1`	0.334	0.3305
`mod2`	0.334	0.3305
`mod3`	0.330	0.3430
`mod4`	0.098	0.1360

Exercise 3 (Bias-Variance Tradeoff, Logistic Regression)

[**8 points**] Run a simulation study to estimate the bias, variance, and mean squared error of estimating $p(x)$ using logistic regression. Recall that $p(x) = P(Y = 1 \mid X = x)$.

Consider the (true) logistic regression model

$$\log\left(\frac{p(x)}{1 - p(x)}\right) = 1 + 2x_1 - x_2$$

To specify the full data generating process, consider the following `R` function.

```r
make_sim_data = function(n_obs = 25) {
  x1 = runif(n = n_obs, min = 0, max = 2)
  x2 = runif(n = n_obs, min = 0, max = 4)
  prob = exp(1 + 2 * x1 - 1 * x2) / (1 + exp(1 + 2 * x1 - 1 * x2))
  y = rbinom(n = n_obs, size = 1, prob = prob)
  data.frame(y, x1, x2)
}
```

So, the following generates one simulated dataset according to the data generating process defined above.

```r
sim_data = make_sim_data()
```

Evaluate estimates of $p(x_1 = 1, x_2 = 1)$ from fitting three models:

$$\log\left(\frac{p(x)}{1 - p(x)}\right) = \beta_0$$

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2$$

Use 1000 simulations of datasets with a sample size of 25 to estimate squared bias, variance, and the mean squared error of estimating $p(x_1 = 1, x_2 = 1)$ using $\hat{p}(x_1 = 1, x_2 = 1)$ for each model. Report your results using a well formatted table.

At the beginning of your simulation study, run the following code, but with your nine-digit Illinois UIN.

```
set.seed(123456789)
```

```
# setup simulation
n_sims = 1000
n_models = 3
x = data.frame(x1 = 1, x2 = 1) # fixed point at which we make predictions
predictions = matrix(0, nrow = n_sims, ncol = n_models)
```

```
# perform simulations
for(sim in 1:n_sims) {

  # generate datasets according to the data generating process
  sim_data = make_sim_data()

  # fit models
  fit_1 = glm(y ~ 1, data = sim_data, family = "binomial")
  fit_2 = glm(y ~ ., data = sim_data, family = "binomial")
  fit_3 = glm(y ~ x1 * x2 + I(x1^2) + I(x1^2), data = sim_data, family = "binomial")

  # get predictions
  predictions[sim, 1] = predict(fit_1, x, type = "response")
  predictions[sim, 2] = predict(fit_2, x, type = "response")
  predictions[sim, 3] = predict(fit_3, x, type = "response")
}
```

```
# helper functions from R4SL

get_var = function(estimate) {
  mean((estimate - mean(estimate)) ^ 2)
}

get_bias = function(estimate, truth) {
  mean(estimate) - truth
}

get_mse = function(truth, estimate) {
  mean((estimate - truth) ^ 2)
}
```

```
# true funciton p(x) as defined by data generating process
p = function(x) {
  with(x,
      exp(1 + 2 * x1 - 1 * x2) / (1 + exp(1 + 2 * x1 - 1 * x2))
```

```r
  )
}
```

```r
# value we are trying to estimate
p(x = x)
```

```
## [1] 0.8807971
```

```r
# calculate bias, variance, and mse of predictions for each logistic regression
bias = apply(predictions, 2, get_bias, truth = p(x))
variance = apply(predictions, 2, get_var)
mse = apply(predictions, 2, get_mse, truth = p(x))
```

```r
# summarize results
results = data.frame(
  c("Intercept Only", "Additive", "Second Order"),
  round(mse, 5),
  round(bias ^ 2, 5),
  round(variance, 5)
)
colnames(results) = c("Logistic Regression Model",
                      "Mean Squared Error",
                      "Bias Squared",
                      "Variance")
rownames(results) = NULL
knitr::kable(results)
```

Logistic Regression Model	Mean Squared Error	Bias Squared	Variance
Intercept Only	0.05743	0.04902	0.00841
Additive	0.00853	0.00006	0.00847
Second Order	0.01967	0.00020	0.01947

Exercise 4 (Concept Checks)

[1 point each] Answer the following questions based on your results from the three exercises.

(a) Based on your results in Exercise 1, do you believe that the true decision boundaries are linear or non-linear?

Solution: Non-linear since the fourth classifier performs best.

(b) Based on your results in Exercise 2, which of these models performs best?

Solution: `mod4`, which includes an interaction performs best.

(c) Based on your results in Exercise 2, which of these models are underfitting?

Solution: The first three models are underfitting as they are all simpler than the "best" model.

(d) Based on your results in Exercise 2, which of these models are overfitting??

Solution: None of these models are overfitting as the "best" model is also the most complex.

(e) Based on your results in Exercise 3, which models are performing unbiased estimation?

Solution: Both the additive and second order models.

(f) Based on your results in Exercise 3, which of these models performs best?

Solution: The additive model. It was the lowest MSE for estimating the probability.