Problem Set 5

Classification

[YOUR NAME]

Due Date: 2024-07-21 @ 11:59PM

Getting Set Up

Open `RStudio` and create a new RMarkDown file (`.Rmd`) by going to `File -> New File -> R Markdown...`. Accept defaults and save this file as `[YOUR NAME]_ps5.Rmd` to your `code` folder.

Copy and paste the contents of this `.Rmd` file into your `[YOUR NAME]_ps5.Rmd` file. Then change the `author: [Your Name]` on line 2 to your name.

We will be using the fn_cleaned_final.Rds file from the course github page (https://github.com/jbisbee1/ISP_Data_Science_2024/blob/main/data/fn_cleaned_final.Rds). The codebook for this dataset is produced below.

Name	Description
player	A unique ID for each player
gameId	A unique ID for each game
startTime	When the game started
sessionId	A unique ID for each gaming session
gameIdSession	A unique ID for each game within a session
won	A binary variable indicating whether the game was won or not
mental_state	Whether the player was drunk or sober when they played
eliminations	How many times they killed an enemy player
assists	How many times they helped a teammate kill an enemy player
revives	How many times was the player revived (brought back to life)?
accuracy	The proportion of total shots that hit an enemy player
hits	The number of shots that hit an enemy player
head_shots	The number of shots that hit an enemy player in the head
distance_traveled	How far did the player move in the game?
materials_gathered	How many materials did the player gather in the game?
materials_used	How many materials did the player use in the game?
damage_taken	The amount of damage the player received from enemy players

Name	Description
damage_to_players	How much damage did the player commit to other players in the game?
damage_to_structures	How much damage did the player commit to structures in the game?

All of the following questions should be answered in this `.Rmd` file. There are code chunks with incomplete code that need to be filled in.

This problem set is worth 10 total points, plus **two** extra credit questions, each worth **two** points. The point values for each question are indicated in brackets below. To receive full credit, you must have the correct code. In addition, some questions ask you to provide a written response in addition to the code.

You are free to rely on whatever resources you need to complete this problem set, including lecture notes, lecture presentations, Google, your classmates…you name it. However, the final submission must be complete by you. There are no group assignments. To submit, upload the knitted output to Assignment 5 on the eTL website **as a PDF** by 11:59PM on Sunday, July 21st. If you need help converting to a PDF, see this tutorial (https://github.com/jbisbee1/ISP_Data_Science_2024/blob/main/Psets/ISP_pset_0_HELPER.pdf).

Good luck!

*Copy the link to ChatGPT you used here: _____

Question 0

Require `tidyverse`, `tidymodels` and `ranger`, and then load the `fn_cleaned_final.Rds` (https://github.com/jbisbee1/ISP_Data_Science_2024/blob/main/Lectures/5_Regression/data/fn_cleaned_final.Rds?raw=true') data to an object called `fn`.

```
require(...)
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```
require(...)
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```
require(...)
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```
fn <- read_rds('')
```

```
## Error in read_rds(""): could not find function "read_rds"
```

Question 1 [2 points]

In this problem set, we are interested in developing a classifier that maximizes our accuracy for predicting Fortnite victories. To do so we will use a linear probability model, a logit, and a random forest. We will start by using two X variables to predict the probability of winning: accuracy (`accuracy`), and head shots (`head_shots`). Our outcome variable of interest Y is whether the player won the game (`won`).

Start by **looking** *at these variables. Why types of variables are they? How much missingness do they have? What do their univariate visualizations look like?*

```
# What types?
glimpse(...)
```

```
## Error in glimpse(...): could not find function "glimpse"
```

```
# How much missingness?
summary(...)
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```
# Univariate: accuracy
# INSERT CODE HERE

# Univariate: head_shots
# INSERT CODE HERE

# Univariate: won
# INSERT CODE HERE
```

> Write answer here.

Then create two multivariate visualizations of the relationship between `won` and each of the two X variables one-by-one.

```
# Multivariate: accuracy
# INSERT CODE HERE

# Multivariate: head_shots
# INSERT CODE HERE
```

Finally, use `geom_tile()` to create a heatmap of the three-way relationship, where quintiles of `accuracy` is on the x-axis, quintiles of `head_shots` is on the y-axis, and tiles are filled according to the average winning probability. (NB: look up what "quintile" means if you are not sure.) Is there anything surprising about this result?

```
# Multivariate: 3-dimensions
fn %>%
  mutate(accuracy_quintile = ntile(...),  # Create 5 quantiles for the accuracy variable
         head_quintile = ntile(...)) %>%  # Create 5 quantiles for the head_shots variab
le
  group_by(...) %>% # Calculate the probability of winning and number of games by both q
uantiles
  summarise(prob_win = ...,
            nGames = ...) %>%
  ggplot(aes(x = factor(...), # Put one set of quantiles on the x-axis and convert to a
factor
             y = factor(...), # Put the other set of quantiles on the y-axis and again c
onvert to a factor
             fill = ...)) +   # Fill by the probability of winning
  geom_...() + # Choose the correct geom_...()
  labs(x = '',  # Give it good labels
       y = '',
       fill = '')
```

```
## Error in fn %>% mutate(accuracy_quintile = ntile(...), head_quintile = ntile(...)) %
>% : could not find function "%>%"
```

Write answer here.

Question 2 [2 points]

Now let's run a linear model and evaluate it in terms of overall accuracy, sensitivity and specificity using a threshold of 0.5.

```
# Running logit model
model_glm <- lm(formula = ..., # Define the regression equation
                data = ...) # Specify the data

# Calculating accuracy, sensitivity, and specificity
fn %>%
  mutate(prob_win = predict(..., # Specify the model name here
                            data = ...) %>% # Specify the data here
  mutate(pred_win = ifelse(...,...,...)) %>% # Convert the probability to a 1 if the pro
babability is greater than 0.5, or zero otherwise
  group_by(...) %>% # Calculate the total games by whether they were actually won or los
t
  mutate(total_games = ...) %>%
  group_by(...,...,...) %>% # Calculate the number of games by whether they were actuall
y won or lost, and by whether they were predicted to be won or lost
  summarise(nGames=...) %>%
  mutate(prop = ... / ...) %>% # Calculate the proportion of game by the total games
  ungroup() %>%
  mutate(accuracy = sum((... == ...)*...) / sum(...)) # Calculate the overall accuracy
```

```
## Error: <text>:17:0: unexpected end of input
## 15:   ungroup() %>%
## 16:   mutate(accuracy = sum((... == ...)*...) / sum(...)) # Calculate the overall acc
uracy
##    ^
```

Then, determine the threshold that maximizes both specificity and sensitivity by creating a sentivity-vs-specificity plot and adding a vertical line to pinpoint where the two measures intersect.

```
# Create the sensitivity vs specificity plot
toplot <- NULL # Instantiate an empty object
for(thresh in seq(...,...,by = ...)) { # Iterate over thresholds between 0 and 1 by 0.02
5
  toplot <- fn %>%
    mutate(prob_win = predict(..., # Specify the model name here
                              data = ...) %>% # Specify the data here
    mutate(pred_win = ifelse(...,...,...)) %>% # Convert the probability to a 1 if the p
robability is greater than the current threshold (thresh), or zero otherwise
    group_by(...) %>% # Calculate the total games by whether they were actually won or l
ost
    mutate(total_games = ...) %>%
    group_by(...,...,...) %>% # Calculate the number of games by whether they were actua
lly won or lost, and by whether they were predicted to be won or lost
    summarise(nGames=...) %>%
    mutate(prop = ... / ...) %>% # Calculate the proportion of game by the total games
    ungroup() %>%
    mutate(accuracy = sum((... == ...)*...) / sum(...)) %>% # Calculate the overall accu
racy
    mutate(threshold = ...) %>% # Record the threshold level
    bind_rows(...) # Add it to the toplot object
}

toplot %>%
  mutate(metric = ifelse(won == ... & pred_win == ...,'Sensitivity',
                         ifelse(won == ... & pred_win == ...,'Specificity',
                                NA))) %>% # Using an ifelse() function, label each row a
s either Sensitivity (if the predicted win is 1 and the true win is 1), Specificity (if
the predicted win is 0 and the true win is 0), or NA
  drop_na(...) %>% # Drop rows that are neither sensitivity nor specificity measures
  ggplot(aes(x = ...,
             y = ...,
             color = ...)) + # Visualize the Sensitivity and Specificity curves by putti
ng the threshold on the x-axis, the proportion of all games on the y-axis, and coloring
by Sensitivity or Specificity
  geom_...() + # Use a line graph
  geom_vline(xintercept = ...) # Try different values of the xintercept until you find w
here the two measures intersect.
```

```
## Error: <text>:17:1: unexpected '}'
## 16:     bind_rows(...) # Add it to the toplot object
## 17: }
##     ^
```

Write answer here.

Finally, plot the area under the curve (AUC) and then calculate it. What grade would you give this model?

```
# Plot the AUC
toplot %>%
  mutate(metric = ifelse(won == ... & pred_win == ...,'Sensitivity',
                         ifelse(won == ... & pred_win == ...,'Specificity',
                                NA))) %>% # Using an ifelse() function, label each row a
s either Sensitivity (if the predicted win is 1 and the true win is 1), Specificity (if
the predicted win is 0 and the true win is 0), or NA
  drop_na(...) %>% # Drop rows that are neither sensitivity nor specificity measures
  select(...,
         ...,
         ...) %>% # Select only the prop, metric, and threshold columns
  pivot_wider(names_from = ...,
              values_from = ...) %>% # Pivot the data to wide format, where the new colu
mns should be the metric
  arrange(desc(...),
          ...) %>% # Arrange by descending specificity, and then by sensitivity
  ggplot(aes(x = ..., # Plot 1 minus the Specificity on the x-axis
             y = ...)) +  # Plot the Sensitivity on the y-axis
  geom_...() + # Visualize with a line
  xlim(c(...)) + ylim(c(...)) + # Expand the x and y-axis limits to be between 0 and 1
  geom_abline(slope = ...,
              intercept = ...,
              linetype = ...) + # Add a dotted 45-degree line using geom_abline()
  labs(x = '',# Add clear labels! (Make sure to indicate that this is the result of a li
near regression model)
       y = '',
       title = '',
       subtitle = '')
```

```
## Error in toplot %>% mutate(metric = ifelse(won == ... & pred_win == ..., : could not
find function "%>%"
```

```
# Calculate the AUC
toEval <- fn %>%
    mutate(prob_win = predict(...,   # Put the model object here
                              data = ...), # Predict on the original data
           truth = factor(..., # Transform the outcome variable to a factor
                          levels = c(...,...))) # Set the levels to be 1 first, and 0 se
cond
```

```
## Error in fn %>% mutate(prob_win = predict(..., data = ...), truth = factor(..., : cou
ld not find function "%>%"
```

```
roc_auc(data = ..., # Run the roc_auc() function on the dataset we just created
        ..., # Tell it which column contains the true outcomes
        ...) # Tell it which column contains our model's predicted probabilities
```

```
## Error in roc_auc(data = ..., ..., ...): could not find function "roc_auc"
```

Write answer here.

Question 3 [1 point]

Now let's re-do the exact same work, except use a **logit model** *instead of a linear model. Based on your analysis, which model has a larger AUC?*

```
# Running logit model
model_glm <- glm(formula = ..., # Define the regression equation
                 data = ...,  # Specify the data
                 family = ...)  # Set the family to a binomial model with a logit link f
unction

# Calculating accuracy, sensitivity, and specificity
fn %>%
  mutate(prob_win = predict(..., # Specify the model name here
                            data = ..., # Specify the data here
                            type = ...) %>%  # Remember to set the correct type becaus
e this is a logit!
  mutate(pred_win = ifelse(...,...,...)) %>% # Convert the probability to a 1 if the pro
bability is greater than 0.5, or zero otherwise
  group_by(...) %>% # Calculate the total games by whether they were actually won or los
t
  mutate(total_games = ...) %>%
  group_by(...,...,...) %>% # Calculate the number of games by whether they were actuall
y won or lost, and by whether they were predicted to be won or lost
  summarise(nGames=...) %>%
  mutate(prop = ... / ...) %>% # Calculate the proportion of game by the total games
  ungroup() %>%
  mutate(accuracy = sum((... == ...)*...) / sum(...)) # Calculate the overall accuracy

# Create the sensitivity vs specificity plot
toplot <- NULL # Instantiate an empty object
for(thresh in seq(...,...,by = ...)) { # Iterate over thresholds between 0 and 1 by 0.02
5
  toplot <- fn %>%
    mutate(prob_win = predict(..., # Specify the model name here
                              data = ..., # Specify the data here
                              type = ...) %>%  # Remember to set the correct type becaus
e this is a logit!
    mutate(pred_win = ifelse(...,...,...)) %>% # Convert the probability to a 1 if the p
robability is greater than the current threshold (thresh), or zero otherwise
    group_by(...) %>% # Calculate the total games by whether they were actually won or l
ost
    mutate(total_games = ...) %>%
    group_by(...,...,...) %>% # Calculate the number of games by whether they were actua
lly won or lost, and by whether they were predicted to be won or lost
    summarise(nGames=...) %>%
    mutate(prop = ... / ...) %>% # Calculate the proportion of game by the total games
    ungroup() %>%
    mutate(accuracy = sum((... == ...)*...) / sum(...)) %>% # Calculate the overall accu
racy
    mutate(threshold = ...) %>% # Record the threshold level
    bind_rows(...) # Add it to the toplot object
}

toplot %>%
  mutate(metric = ifelse(won == ... & pred_win == ...,'Sensitivity',
                         ifelse(won == ... & pred_win == ...,'Specificity',
```

```
                              NA))) %>% # Using an ifelse() function, label each row a
s either Sensitivity (if the predicted win is 1 and the true win is 1), Specificity (if
the predicted win is 0 and the true win is 0), or NA
  drop_na(...) %>% # Drop rows that are neither sensitivity nor specificity measures
  ggplot(aes(x = ...,
             y = ...,
             color = ...)) + # Visualize the Sensitivity and Specificity curves by putti
ng the threshold on the x-axis, the proportion of all games on the y-axis, and coloring
by Sensitivity or Specificity
  geom_...() + # Use a line graph
  geom_vline(xintercept = ...) # Try different values of the xintercept until you find w
here the two measures intersect.

# Plot the AUC
toplot %>%
  mutate(metric = ifelse(won == ... & pred_win == ...,'Sensitivity',
                         ifelse(won == ... & pred_win == ...,'Specificity',
                                NA))) %>% # Using an ifelse() function, label each row a
s either Sensitivity (if the predicted win is 1 and the true win is 1), Specificity (if
the predicted win is 0 and the true win is 0), or NA
  drop_na(...) %>% # Drop rows that are neither sensitivity nor specificity measures
  select(...,
         ...,
         ...) %>% # Select only the prop, metric, and threshold columns
  pivot_wider(names_from = ...,
              values_from = ...) %>% # Pivot the data to wide format, where the new colu
mns should be the metric
  arrange(desc(...),
          ...) %>% # Arrange by descending specificity, and then by sensitivity
  ggplot(aes(x = ..., # Plot 1 minus the Specificity on the x-axis
             y = ...)) +  # Plot the Sensitivity on the y-axis
  geom_...() + # Visualize with a line
  xlim(c(...)) + ylim(c(...)) + # Expand the x and y-axis limits to be between 0 and 1
  geom_abline(slope = ...,
              intercept = ...,
              linetype = ...) + # Add a dotted 45-degree line using geom_abline()
  labs(x = '',# Add clear labels! (Make sure to indicate that this is the result of a lo
git regression model)
       y = '',
       title = '',
       subtitle = '')

# Calculate the AUC
toEval <- fn %>%
    mutate(prob_win = predict(...,   # Put the model object here
                              data = ..., # Predict on the original data
                              type = ....), # Remember to set the correct type because t
his is a logit!
           truth = factor(..., # Transform the outcome variable to a factor
                          levels = c(...,...))) # Set the levels to be 1 first, and 0 se
cond
```

```
roc_auc(data = ..., # Run the roc_auc() function on the dataset we just created
        ..., # Tell it which column contains the true outcomes
        ...) # Tell it which column contains our model's predicted probabilities
```

```
## Error: <text>:21:1: unexpected symbol
## 20: # Create the sensitivity vs specificity plot
## 21: toplot
##     ^
```

Write answer here.

Question 4 [1 points]

Now let's re-do the exact same work, except use a **random forest** *model instead of a linear model. Based on your analysis, which model has a larger AUC?*

```
# Running linear model
model_rf <- ranger(formula = ..., # Define the regression equation
                   data = ...) # Specify the data
```

```
## Error in ranger(formula = ..., data = ...): could not find function "ranger"
```

```
# Calculating accuracy, sensitivity, and specificity
fn %>%
  mutate(prob_win = predict(..., # Specify the model name here
                            data = ...)$predictions) %>% # Specify the data here and ext
ract the predictions with $predictions
  mutate(pred_win = ifelse(...,...,...)) %>% # Convert the probability to a 1 if the pro
bability is greater than 0.5, or zero otherwise
  group_by(...) %>% # Calculate the total games by whether they were actually won or los
t
  mutate(total_games = ...) %>%
  group_by(...,...,...) %>% # Calculate the number of games by whether they were actuall
y won or lost, and by whether they were predicted to be won or lost
  summarise(nGames=...) %>%
  mutate(prop = ... / ...) %>% # Calculate the proportion of game by the total games
  ungroup() %>%
  mutate(accuracy = sum((... == ...)*...) / sum(...)) # Calculate the overall accuracy
```

```
## Error in fn %>% mutate(prob_win = predict(..., data = ...)$predictions) %>% : could n
ot find function "%>%"
```

```
# Create the sensitivity vs specificity plot
toplot <- NULL # Instantiate an empty object
for(thresh in seq(...,...,by = ...)) { # Iterate over thresholds between 0 and 1 by 0.02
5
  toplot <- fn %>%
    mutate(prob_win = predict(..., # Specify the model name here
                              data = ...)$predictions) %>% # Specify the data here and e
xtract the predictions with $predictions
    mutate(pred_win = ifelse(...,...,...)) %>% # Convert the probability to a 1 if the p
robability is greater than the current threshold (thresh), or zero otherwise
    group_by(...) %>% # Calculate the total games by whether they were actually won or l
ost
    mutate(total_games = ...) %>%
    group_by(...,...,...) %>% # Calculate the number of games by whether they were actua
lly won or lost, and by whether they were predicted to be won or lost
    summarise(nGames=...) %>%
    mutate(prop = ... / ...) %>% # Calculate the proportion of game by the total games
    ungroup() %>%
    mutate(accuracy = sum((... == ...)*...) / sum(...)) %>% # Calculate the overall accu
racy
    mutate(threshold = ...) %>% # Record the threshold level
    bind_rows(...) # Add it to the toplot object
}
```

```
## Error in seq(..., ..., by = ...): '...' used in an incorrect context
```

```
toplot %>%
  mutate(metric = ifelse(won == ... & pred_win == ...,'Sensitivity',
                         ifelse(won == ... & pred_win == ...,'Specificity',
                                NA))) %>% # Using an ifelse() function, label each row a
s either Sensitivity (if the predicted win is 1 and the true win is 1), Specificity (if
the predicted win is 0 and the true win is 0), or NA
  drop_na(...) %>% # Drop rows that are neither sensitivity nor specificity measures
  ggplot(aes(x = ...,
             y = ...,
             color = ...)) + # Visualize the Sensitivity and Specificity curves by putti
ng the threshold on the x-axis, the proportion of all games on the y-axis, and coloring
by Sensitivity or Specificity
  geom_...() + # Use a line graph
  geom_vline(xintercept = ...) # Try different values of the xintercept until you find w
here the two measures intersect.
```

```
## Error in toplot %>% mutate(metric = ifelse(won == ... & pred_win == ..., : could not
find function "%>%"
```

```
# Plot the AUC
toplot %>%
  mutate(metric = ifelse(won == ... & pred_win == ...,'Sensitivity',
                         ifelse(won == ... & pred_win == ...,'Specificity',
                                NA))) %>% # Using an ifelse() function, label each row a
s either Sensitivity (if the predicted win is 1 and the true win is 1), Specificity (if
the predicted win is 0 and the true win is 0), or NA
  drop_na(...) %>% # Drop rows that are neither sensitivity nor specificity measures
  select(...,
         ...,
         ...) %>% # Select only the prop, metric, and threshold columns
  pivot_wider(names_from = ...,
              values_from = ...) %>% # Pivot the data to wide format, where the new colu
mns should be the metric
  arrange(desc(...),
          ...) %>% # Arrange by descending specificity, and then by sensitivity
  ggplot(aes(x = ..., # Plot 1 minus the Specificity on the x-axis
             y = ...)) +  # Plot the Sensitivity on the y-axis
  geom_...() + # Visualize with a line
  xlim(c(...)) + ylim(c(...)) + # Expand the x and y-axis limits to be between 0 and 1
  geom_abline(slope = ...,
              intercept = ...,
              linetype = ...) + # Add a dotted 45-degree line using geom_abline()
  labs(x = '',# Add clear labels! (Make sure to indicate that this is the result of a ra
ndom forest regression model)
       y = '',
       title = '',
       subtitle = '')
```

```
## Error in toplot %>% mutate(metric = ifelse(won == ... & pred_win == ..., : could not
find function "%>%"
```

```
# Calculate the AUC
toEval <- fn %>%
    mutate(prob_win = predict(...,   # Put the model object here
                              data = ...)$predictions, # Predict on the original data an
d extract the predictions using the $ symbol
           truth = factor(..., # Transform the outcome variable to a factor
                          levels = c(...,...))) # Set the levels to be 1 first, and 0 se
cond
```

```
## Error in fn %>% mutate(prob_win = predict(..., data = ...)$predictions, : could not f
ind function "%>%"
```

```
roc_auc(data = ..., # Run the roc_auc() function on the dataset we just created
        ..., # Tell it which column contains the true outcomes
        ...) # Tell it which column contains our model's predicted probabilities
```

```
## Error in roc_auc(data = ..., ..., ...): could not find function "roc_auc"
```

Write answer here.

Question 5 [2 points]

Use 100-fold cross validation with a 60-40 split to calculate the average AUC for all three models. Which is better?

```
set.seed(123) # Set the seed
cvRes <- NULL # Instantiate the empty object
for(i in 1:100) {
  # Split the data into training a test sets
  train <- fn %>%
    sample_n(size = ...,    # Set the size here (60-40 split)
             replace = ...) # Make sure NOT to replace the rows after you randomly sampl
e them

  test <- ... # Create the test object with anti_join()

  # Training models
  mLM <- lm(formula = ..., # Estimate the linear model here
            data = ...) # Give it the correct data

  mGLM <- glm(formula = ...,  # Estimate the logistic model here
            data = ..., # Give it the correct data
            family = ...) # Set the family to the binomial distribution with a logit lin
k

  mRF <- ranger(formula = ..., # Estimate the random forest here
                data = ...) # Give it the correct data

  # Predicting models
  toEval <- test %>%
    mutate(mLMPreds = predict(...,   # Put the model object here
                              newdata = ...), # Predict on the new data
           mGLMPreds = predict(...,   # Put the model object here
                               newdata = ..., # Predict on the new data
                               type = ...), # Make sure to set the type to response!
           mRFPreds = predict(...,   # Put the model object here
                              data = ...)$predictions, # Predict on the new data and ext
ract the predictions using the $ symbol
           truth = factor(..., # Transform the outcome variable to a factor
                          levels = c(...,...))) # Set the levels to be 1 first, and 0 se
cond

  # Evaluating models
  rocLM <- roc_auc(..., # Give the function the data to analyze
                   ..., # Put the true outcome here
                   ...) %>% # Put the linear model predictions here
    mutate(model = ...) %>% # Save the model name to a new column called "model"
    rename(... = ...) # Rename the .estimate column to auc

  rocGLM <- roc_auc(..., # Give the function the data to analyze
                   ..., # Put the true outcome here
                   ...) %>% # Put the logit model predictions here
    mutate(model = ...) %>% # Save the model name to a new column called "model"
    rename(... = ...) # Rename the .estimate column to auc

  rocRF <- roc_auc(..., # Give the function the data to analyze
                   ..., # Put the true outcome here
```

```
                  ...) %>% # Put the random forest predictions here
    mutate(model = ...) %>% # Save the model name to a new column called "model"
    rename(... = ...) # Rename the .estimate column to auc

  cvRes <- rocLM %>%
    bind_rows(..., # Bind both the logit and random forest results to the LM results
              ...) %>%
    mutate(cvInd = ...) %>% # Save the cross valudation number here
    bind_rows(...) # Add the results to the top of the cvRes object
}
```

```
## Error in fn %>% sample_n(size = ..., replace = ...): could not find function "%>%"
```

```
# Calculate overall AUC averages by each model
cvRes %>%
  group_by(...) %>%
  summarise(mean_auc = ...)
```

```
## Error in cvRes %>% group_by(...) %>% summarise(mean_auc = ...): could not find functi
on "%>%"
```

Write answer here.

Extra Credit [2 Points]

Using the AUC result, can you create a multivariate visualization of the AUC by model? What proportion of cross validations does the linear model outperform the logit and random forest models?

```
# INSERT CODE HERE
```

Write answer here.

Question 6 [2 points]

Now let's run a kitchen sink model using many different predictors. We will use the following X variables:

- hits
- assists
- accuracy
- head_shots
- damage_to_players
- eliminations
- revives
- distance_traveled
- materials_gathered

- `mental_state`

Which of these variables do you think will be **most important** *for predicting whether the player wins the game? Why? What about the variables that are* **least important***? Why?*

> Write answer here.

Now run a random forest that uses all of these variables as predictors and use `importance = 'permutation'` *to see which variables the random forest thinks are most important. Visualize these results with a barplot. Where do the variables you thought would be best and worst appear?*

```
# Estimate the random forest
model_rf  <- ranger(formula = ...,    # Write regression equation here
                    data = ...,       # Set the data here
                    importance = ...) # Use permutation importance
```

```
## Error in ranger(formula = ..., data = ..., importance = ...): could not find function
"ranger"
```

```
# Create an object to plot the variable importance results
toplot <- data.frame(vimp = ...,        # Save the variable importance values here
                     vars = names(...)) # Save the variable names here
```

```
## Error in eval(expr, envir, enclos): '...' used in an incorrect context
```

```
# Create a plot of the variable importance results
toplot %>%
  ggplot(aes(x = ..., # Put one variable on the x-axis
             y = reorder(..., ...))) +  # Put the other variable on the y-axis and reord
er
  geom_...(...) +  # Choose the best geom_...()
  labs(x = '', # Provide descriptive labels
       y = '',
       title = '')
```

```
## Error in toplot %>% ggplot(aes(x = ..., y = reorder(..., ...))): could not find funct
ion "%>%"
```

> Write answer here.

Extra Credit [2 Points]

What is the overall AUC from your random forest model using all these variables?

```
# INSERT CODE HERE
```

Write answer here.

Then use 100 cross validation with a 60-40 split to get a better measure of the true model performance. Is the AUC from running the model on the full data different from the cross validation answer? Calculate the proportion of cross validation splits where the AUC is worse than the value calculated on the full data. Do you think there is evidence of overfitting?

```
# INSERT CODE HERE
```

Write answer here.