Main Research Question

CT imaging of head-injured children has risks of radiation-induced malignancy. Your goal in this lab is build a model to identify children at very low risk of clinically-important traumatic brain injuries (ciTBI) for whom CT might be unnecessary. *Build an effective model that classifies whether or not a patient has ciTBI.*

Materials Provided

See the Ed post for links to the assignment repo with the data and starter documents.

Context

- Previous work done on this front (https://pubmed.ncbi.nlm.nih.gov/19758692/).
  - pdf included in `data/`.

Data

1. `tbi_data_dictionary.xlsx`: A data dictionary (same document from project 3).
2. `citbi.csv`: A dataset with clinically-important traumatic brain injury (ciTBI) outcomes for 30,379 pediatric patients with head trauma (same dataset as Project 3).

Starter Documents

1. `constant_modeling.qmd`: Modeling and evaluation for parts 2-3.
2. `modeling.qmd`: Modeling and evaluation for parts 4-6.
3. `competition.R`: Self contained R code to calculate outcomes (ciTBI/no ciTBI) of test set for Kaggle competition. Part 7.

Part 1: Data Cleaning R Package

As discussed in lecture, R packages are a common structured approach to organizing code. For this part, you will create an R package for data cleaning.

- Create an R package that contains function(s) to prepare the data for modeling. You may decide to use many functions, or even just one function that including the data cleaning tasks. *Hint: You can base this on your work from Project 3. Feel free to use the project 3 solutions.*
- Document these function(s) using `roxygen2` style documentation.
- Store the original citbi dataset in your package such that it can be accessed by calling `citbi_raw` (lazy-loading) after loading your package. More info about how to do this be found here.
- Document the dataset in your package using `roxygen2` style documentation.
- **Load** your package in `constant_modeling.qmd`, `modeling.qmd`, and `competition.R` files. **Do not** use `read_csv` to load the data in these files. Instead, use *lazy-loading* to access the raw data and then apply the package's cleaning function(s).

Part 2: Constant Prediction Models

Constant models will serve as baselines to compare with more sophisticated models later. They also helpful to understand the tradeoffs between different types of errors.

- Create a function `no_citbi` that predicts no ciTBI for every patient. Describe the real-world drawbacks and upsides of this model.
- Create a function `all_citbi` that predicts every patient has ciTBI. Describe the real-world drawbacks and upsides of this model.

Part 3: Accuracy, Precision, and Recall I

- Describe what each of the following mean in the context of this problem:
  - True Positive (TP)
  - True Negative (TN)
  - False Positive (FP)
  - False Negative (FN)
- What do accuracy, precision, and recall represent in the context of this problem?

Hint:

$$\begin{aligned}
\text{Accuracy} &= \frac{TP+TN}{TP+TN+FP+FN} && = \mathbb{P}[\text{correct prediction}], \\
\text{Precision} &= \frac{TP}{TP+FP} && = \mathbb{P}[(+ \text{ in reality}) \text{ given } (\text{test } +)] \\
\text{Recall} &= \frac{TP}{TP+FN} && = \mathbb{P}[(\text{test } +) \text{ given } (+ \text{ in reality})]
\end{aligned}$$

- For the models in Part 2, calculate:
  - TP, TN, FP, FN
  - Accuracy, Precision, and Recall
- Which metric (out of precision and recall, and accuracy) is most important for this problem? Rank them from most to least important and justify your ranking.

Part 4: Modeling

In `modeling.qmd`, build (binary) classification models using the following algorithms with the `tidymodels` package framework. You may use as little or as many predictor variables as you like, but your resulting predictions should be stored in a list called `model_predictions` with names corresponding to each model.

- Logistic Regression
- K Nearest Neighbors
- (Optional) Decision Tree
- (Optional) Random Forest
- (Optional) LightGBM

Hints:

- *Ensure there are no NA values in your predictor or target variables.*
- *You will need to install the packages* `tidymodels`, `rpart`, `ranger`, *and* `lightgbm` *if you haven't already.*
- *You may need to convert the target variable to a factor for classification tasks.*

Part 5: Accuracy, Precision, Recall II

- For each model in Part 4, calculate:
  - TP, TN, FP, FN
  - Accuracy, Precision, and Recall
  - The $F\beta$ score for $\beta = 0.5$, 1, and 2. The $F\beta$ score is a metric that combines precision and recall into a single number, with β determining the weight of recall in the combined score. It is defined as:

$$F\beta = \frac{(1 + \beta^2) \cdot \text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

- Explain what the $F\beta$ score represents, and which value of β you think is most appropriate in the context of this problem.
- Compare each models using these metrics and discuss which model you prefer and why.

Part 6: Hyperparameter Tuning

- Choose (at least) one model from Part 4 to tune.
- Describe which hyperparameters you tuned and why. You don't have to understand what each hyperparameter represents, but what metric improved?
- Some common hyperparameters to tune for each model:

- Logistic Regression: penalty, mixture (must use “glmnet” engine)
- KNN: neighbors (k), weight_func, dist_power
- Decision Tree: tree_depth, min_n, cost_complexity
- Random Forest: mtry, trees, min_n, importance
- LightGBM: learn_rate, tree_depth, trees, min_n, mtry, sample_size, loss_reduction

Hint: You can adjust hyperparameter in the tidymodels framework using the `set_args()` *function. Example adjusting* `ntree` *in random forest:*

```r
model <- rand_forest() |>
  set_engine("ranger") |>
  set_mode("classification") |>
  # tuning trees
  set_args(tress = 500)
```

Part 7: Kaggle Competition

Notice: For the Kaggle Competition, you are **not permitted** to use any model besides the ones previously listed in this project (i.e. no XGBoost, MLP/Neural Network, etc).

Overview

Kaggle is a data science platform for competitions, datasets, and code sharing. In the final part of this project, you will be fine-tuning your model to maximize the F1-score.

The competition invite link: https://www.kaggle.com/t/9a5e3f8041f8b937da79c490c7bf4896

- You must make your predictions on the test dataset, which you can download in the “Data” tab on the kaggle competition.
  - Notice the new column `id`. This must remain in your final Kaggle submission for the true labels to be measured against your predictions.
- Your `submission.csv` file should look something like this:

```
id,target_feature
1,0
2,0
3,0
4,1
...
13013,0
```

Where id is the original `id` column in `test.csv`, and `target_feature` is your model prediction for that given row. **Note that your result must have these exact column names and exactly 13013 observations**

Recommended Approach

There are many techniques you can employ to maximize the F1-score you achieve on the test set in Kaggle, including but not limited to:

i. Consider splitting your cleaned training data into training and validation sets. This can help you emulate the performance metric on kaggle locally, maximizing *validation* F1-score.
ii. Consider using a model listed as optional in part 4.
iii. Consider using cross validation to tune model hyperparameters. A few guides that explore this using the tidymodels framework:

- https://tidyverse.org/blog/2023/04/tuning-delights/
- https://workshops.tidymodels.org/archive/2022-08-Reykjavik-City/06-tuning-hyperparameters.html#/title-slide

iv. Experiment with feature engineering to create new predictive features from existing ones.

How to Submit

Two requirements:

- GitHub Classroom: Once you're happy with your work, render each of the Quarto documents one more time and commit both the .qmd files and the .pdf files that are produced. Then push these, `competition.R`, and your package to your repo for your GSI to read.
- A Kaggle Submission: Use the `competition.R` script to generate a submission csv file for the Kaggle competition. Follow the instructions in the script to create your predictions on the test set and format them correctly for submission. Once you have your submission file, upload it to the Kaggle competition page.