Project Objectives

This is a solo project where you will create a vignette about selecting a predictive model. Then you'll make that model available as an API. You'll save that in a docker image that I can run and access! You'll produce a short video of you demonstrating your API through a docker container.

Setting Things Up

Creating the Repo

The first step is to create a github repo. All project work should be done within this repo so we can track your activity. You should have at least six substantial commits or else you will lose credit.

One output of this project is a docker image. You won't be able to upload your docker image to the repo as it will be too large. Remember that you can use the `.gitignore` file to tell git to ignore certain files.

Repo Settings

On your github project repo you should go into the settings and enable github pages. You'll output two files (one for the EDA of the data set and one for the model fitting) to a docs folder that will each be converted into a web page, similar to how we've done our homework assignments.

This means you'll have two quarto docs that you are creating. Unless one document is called `index.qmd` the 'landing' site will be the first page alphabetically (almost always). As such, call one file `EDA.qmd` and one `Modeling.qmd`. Place these files in the root (main) folder. At the bottom of the `EDA.qmd` file, give a (relative) link to the modeling page. Something like this will do:

```
[Click here for the Modeling Page](Modeling.html)
```

This should make your 'landing page' the rendered EDA page which then has a link to your Modeling page.

Project Topic

You'll read in and analyze a Diabetes Health Indicators Dataset. Specifically, the `diabetes_binary_health_indicators_BRFSS2015.csv` data. You should read about the data and especially about each variable. **Many are coded numerically but most are not numeric variables.** We'll use the `Diabetes_binary` as our response variable. Download the data so you have it locally in your repo.

Outputs

We'll have six different deliverables in total (more information on each part is available below):

- A quarto file where you do an EDA on the data (this is to be rendered to a nice page via github pages)
- A quarto file where you go through the process of fitting and selecting a 'best' model based on a training/test split using the `tidymodels` framework (to be rendered as a nice page via github pages)
- A `.R` file where you fit your 'best' model as chosen in your modeling file. This file should also define an API for obtaining predictions from that model (among other things, see below).

- A `Dockerfile` that you'll use to build your Docker image
- A saved `.tar` file that corresponds to your Docker image
- A short video of you demonstrating your Docker container

EDA file

You'll want to create a quarto document where you go through an exploratory data analysis of our data set.

Introduction section

You should have an introduction section that

- briefly describes the data and the variables you have to work with (just discuss the ones you'll investigate/use in your analysis).
- describes the purpose of your EDA and ultimate goal of modeling.

Data

Use a relative path to import the data. You likely want to convert a lot of the variables to factors with meaningful level names, check on missingness, etc.

Summarizations

You should then produce meaningful summary statistics and plots about the data you are working with (especially as it relates to your response). Although a best practice would be to split the data at hand into a training and testing set first, go ahead and do your EDA on the full data.

Be sure to have a narrative about what you are exploring and what the summaries and graphs you created say about the relationships in your data.

Modeling File

Start with a basic introduction (feel free to repeat some things from the other file).

Then, split the data into a training (70% of the data) and test set (30% of the data). Set a seed to make things reproducible.

The goal is to create models for predicting the `Diabetes_binary` variable (using `tidymodels`). We'll use log-loss as our `metric` to evaluate the models. For both model types, use log-loss with 5 fold cross-validation to select the best model from that family of models. You should set up your own grid of tuning parameters for each model (even if it is just the number of levels to look at).

Classification Tree

You should provide a thorough explanation of what a classification tree model is. Then you should fit a classification tree with varying values for the complexity parameter and choose the best model (based on 5 fold CV on the training set). Include at least 5 predictors in this model.

Random Forest

You should provide a thorough explanation of what a random forest is and why we might use it (be sure to relate this to a basic classification tree). You should then fit a random forest model with varying values for the `mtry` parameter and choose the best model (based on 5 fold CV on the training set). Include at least 5 predictors in this model.

Final Model Selection

You should now have two *best* models (one for each model type above). Compare both models on the test set and declare an overall winner!

API .R File

You'll create a file that defines an API (via the `plumber` package). At the top of the file, read in your data and fit your 'best' model to the entire data set.

Then you should create three API endpoints:

- A `pred` endpoint. This endpoint should take in any predictors used in your 'best' model. You should have default values for each that is the mean of that variable's values (if numeric) or the most prevalent class (if categorical). **Below this API put three example function calls to the API in comments so that I can easily copy and paste to check that it works!**
- An `info` endpoint. This endpoint shouldn't have any inputs. The output should be a message with:
  - Your name
  - A URL for your rendered github pages site
- A `confusion` endpoint. This endpoint should produce a **plot** of the confusion matrix for your model fit. That is, comparing the predictions from the model to the actual values from the data set (again you fit the model on the entire data set for this part).

Dockerfile

Create a `Dockerfile` similar to that used in the notes. You'll likely need to update packages used and the files copied (the dataset will need to be copied as well since you need to fit your model).

`.tar` of Docker Image

Use this `Dockerfile` to create your Docker image (using `docker build`). Check to make sure everything works by running the container.

Then output your image (not your container) to a `.tar` or `.tar.gz` file using `docker save`. Upload this file to your NC State google drive (you each have 15GB of storage so that should work).

Completely optional: for those that want to try and get fancy with it, you can try to follow this blog to create the `Dockerfile` and image. I think it is likely a bit harder than just following our steps (and I'm not going to help troubleshoot this method as I haven't done it myself).

Video Demonstration

Lastly, I'd like you to create a short (approximately 1 minute) video demonstrating your running the container and using the API. I'd like you to record the video via the following steps:

- Open zoom with your NC State account.
- Create a "New Meeting".
- Turn on your camera so I can see you.
- Share your screen where you'll be using your command prompt (or terminal window) and browser
- Select 'Record' -> 'Record to the Cloud'
- Record yourself **narrating** the running of your image (creating a container) and the use of your API to obtain a prediction from your model.
- Stop the recording and close the zoom meeting

You should then get a link to your recording via email (this may take a little while to process but shouldn't be long as the video is short!). Notice that the link to others (me) includes a password.

Submission

On Moodle you should submit four items in total:

- The link to your github site (non-rendered)
- The link to your rendered github site
- The link to your `.tar` file on your google drive. **Be sure to go to sharing for this file and share it with me**
- The link to your zoom recording **with passcode**.

Rubric for Grading (total = 100 points)

Item	Points	Notes
EDA File/Page	30	Worth either 0, 2.5, 5, ..., 30
Model Introductions	10	Worth either 0, 2.5, 5, 7.5, 10
Classification Tree Fit	10	Worth either 0, 2.5, 5, 7.5, 10
Random Forest Fit	10	Worth either 0, 2.5, 5, 7.5, 10
API file/demo	20	Worth either 0, 2.5, 5, ..., 20
Docker Image	20	Worth either 0, 2.5, 5, ..., 20

Notes on grading:

- For each item in the rubric, your grade will be lowered one level for each each error (syntax, logical, or other) in the code and for each required item that is missing or lacking a description.
- **If your work was not completed and documented using your github repo and/or you do not use GPP you can lose up to 50 points on the project.**
- If your repo is not set up correctly, you can lose up to 30 points.