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!

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 five substantial commits on the repo or else you will lose credit.

Repo Settings

On your github project repo you should go into the settings and enable github pages. You'll output your vignettes (for EDA and model fitting) to a docs folder that will each be converted into a web page, similar to how we did on homework-1. Please see homework-1 instructions if you have any questions.

You'll have two quarto docs that you are creating. This means you'll have two pages in the end. Unless one document is called `index.qmd` the 'landing' site will be the first page alphabetically. As such, call one file `EDA.qmd` and one `Modeling.qmd`. At the bottom of the `EDA.qmd` file, give a link to the modeling page. Something like this will do:

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

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.

Outputs

We'll have five different files in the end.

- A quarto file where you do an EDA on the data.
- A quarto file where you go through the process of fitting and selecting a 'best' model based on a training/test split.
- A .R file where you will fit your 'best' model and define your API components. (You can try out the API with code similar to class but you don't need a separate file for that code.)
- A Dockerfile that you'll use to build your Docker image.
- A saved `.tar` file that corresponds to your docker image

EDA file

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

The goal of the EDA is to not explore every combination of all variables in the data set. For the EDA section, you are required to pick 3 variables (not including our response variable) to explore. There is no wrong answer, but please provide some justification as to why you picked the variables that you did. For example, maybe you point to research that claims a relationship between physical activity and the presence of diabetes, so you believe it's an important predictor and will be used in your analysis.

Be sure to comment on the plots / summary statistics you make.

You are not bound to just using these three variables in your modeling analysis.... I just am limiting the scope of your EDA by only requiring 3 variables.

Hint: I would suggest at least doing a univariate exploration on your 4 total variables, as well as looking at the bivariate relationship between each of your explanatory variables + the response.

Introduction section

You should have an introduction section that

- briefly describes the data and the variables you have to work with.
- 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). Use `set.seed()` to make things reproducible.

The goal is to create models for predicting the `Diabetes_binary` variable (using `caret`). We'll use `logLoss` as our `metric` to evaluate models. For all model types use `logLoss` with 5 fold cross-validation to select the best model. You should set up your own grid of tuning parameters in any model where that is possible.

Logistic Regression Models

You should provide a reasonably thorough explanation of what a logistic regression model is and why we apply it to this kind of data. Then you should fit three candidate logistic regression models and choose the best model using CV with log-loss as your metric.

Classification Tree

You should provide a reasonably thorough explanation of what a classification tree model is and why we might try to use it. Then you should fit a classification tree with varying values for the complexity parameter and choose the best model (best complexity parameter).

Random Forest

You should provide a reasonably thorough explanation of what a random forest is and why we might use it (especially in relation to a basic classification tree). You should then fit a random forest model and choose the best model.

Final Model Selection

You should now have three *best* models (one for each model type above). Now compare all three models on the test set and declare an overall winner!

API .R File

Next, 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.

Then you should create two 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 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

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).

`.tar` of Docker Image

Use this Dockerfile to create your Docker image (`docker build`). Check to make sure everything works by running the container (you'll need to expose some ports similar to what we did with our API example).

Then output your image (not your container - see the week-10 overview on the website) to a `.tar` file.

Submission

All of your files other than your `.tar` should be available on your github site. The `.tar` file should be uploaded to your google drive (you each have 15GB so I don't think this should be a problem!).

You should submit a link to your rendered github pages site, a link to your basic github.com site, and a google drive link to your `.tar` file (**be sure to share this file with me so I can download it**).

Rubric for Grading (total = 100 points)

Item	Points	Notes
EDA Introduction & Reading Data	5	Worth either 0, 2, or 5
Summarizations & discussions	20	Worth either 0, 5,. . . , or 20
Model Introduction & Data Split	5	Worth either 0, 2, 5
Logistic Regression	10	Worth either 0, 3, 6, . . . , 10
Classification Tree	10	Worth either 0, 3, 6, or 10
Random Forest	10	Worth either 0, 3, 6, or 10
Final Model Selection	5	Worth either 0, 2, or 5
API file	15	Worth either 0, 5, 10, or 15
Docker Image	20	Worth either 0, 5, . . . or 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 you will lose 50 points on the project.**
- You should use Good Programming Practices when coding (see wolfware). If you do not follow GPP you can lose up to 40 points on the project.
- You should use appropriate markdown options/formatting (you can lose up to 20 points) for not doing so.
- If your repo is not set up correctly, you can lose up to 30 points.