GitHub Training for Data Scientists Workbook

Author: Mark Nguyen (http://www.marknguyen.me)

This workbook aims to teach a Data Scientist how code submission to GitHub works with a team of three people.

You will learn how to:

1. Clone a GitHub repository.
2. Create your own branch to track your own work.
3. Commit code on your local machine.
4. Push your committed code to GitHub.
5. Submit a Pull Request into the main development branch.
6. Collaborate with other team members online and discuss code changes relating to your pull requests.
7. Fix and add additional changes to your code based on a code review.
8. Track how code commits in GitHub kick starts automatic processes to test the code basis and deploy the application.

Project Details

This project takes data from the UCI Machine Learning Repository for Boston housing prices accessed through the scikit-learn library.

You will be training and deploying a machine learning model to predict Boston housing prices. The model is deployed with a web app that will facilitate the data transfer from the Internet user to the machine learning model.

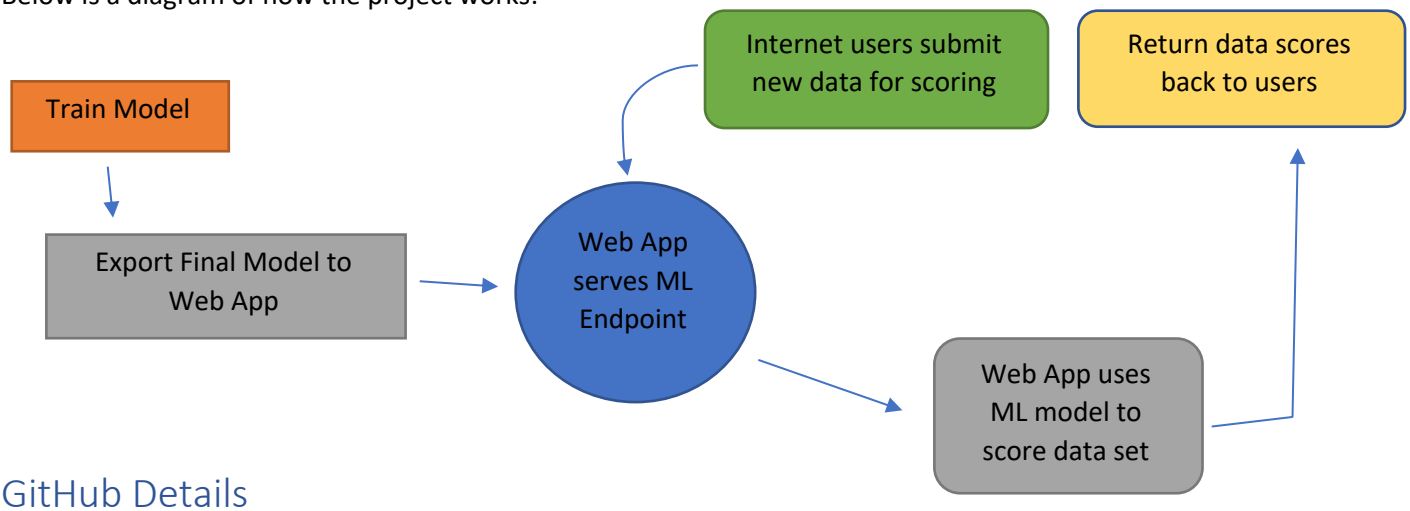
Most of the project has been built out. You are responsible for filling out the gaps in various files to enable the project to test and deploy successfully.

You can find the beginning code here at: https://github.com/Mrk-Nguyen/boston-predict/tree/main.

The completed code can be found here at: https://github.com/Mrk-Nguyen/boston-predict/tree/final.

Your project lead will give you the actual GitHub repository that you will submit code to.

Below is a diagram of how the project works:



GitHub Details

The code repository (repo) currently has two branches: `main` and `dev`

`Main` is the branch that will be used to build the codebase and deploy the application. `Dev` will be the branch to merge code from other feature branches via Pull Requests (PR). Your workflow involves:

1. Working on your own feature branch
2. Submitting PRs to merge your code into the `dev` branch

Once all features are built out and merged into the dev branch, your project leader will merge the `dev` branch into the `main` branch. Once the `main` branch is updated, an automatic process will get triggered to build and deploy the application.

Steps Before You Begin

1. Make sure git is installed on your computer. You may also want to install a code editor that has built-in git and GitHub integration such as Visual Studio Code.
2. Create a GitHub account with your personal email.
3. Give your project leader your GitHub username so they can add you as a collaborator on the project:
    a. The repo has been setup using CircleCI to validate all code committed on GitHub.
4. Copy the **final files** for the project.
    a. You will use these files to copy and paste the necessary code to have the project pass all validation checks and deploy the ML endpoint.
5. Name the folder containing the final files: `final.`
6. Ask your project leader to assign you a role for this project. Once you have your role, move on to the instructions below for your own role after you setup your computer.
7. Make sure your python environment has the following libraries installed:
    a. pandas
    b. matplotlib
    c. seaborn
    d. scikit-learn
    e. lightgbm

Setup Your Computer

1. In your command prompt, navigate to the directory where you want to work on this project.
    f. Clone the GitHub repo:
        i. https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/cloning-a-repository
2. Navigate to the project folder that was created.
3. Confirm that the only branch you have on your machine is `main` using the command: `git branch`
4. Move to the next section as Developer 1 or Developer 2.

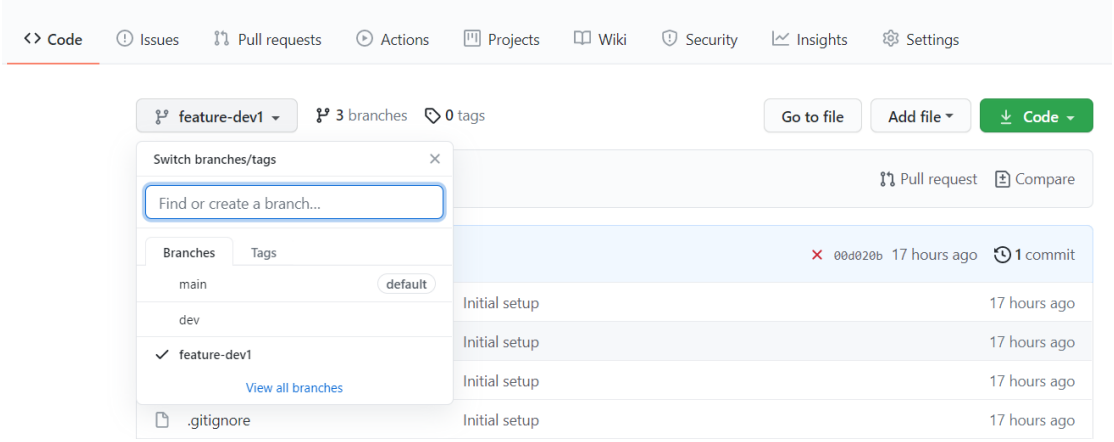
Developer 1

Set Up New Feature Branch

1. Create a new branch named: `feature-dev1`
    a. https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
2. Push your code to a new branch in the GitHub repo:

```
git push --set-upstream origin feature-dev1
```

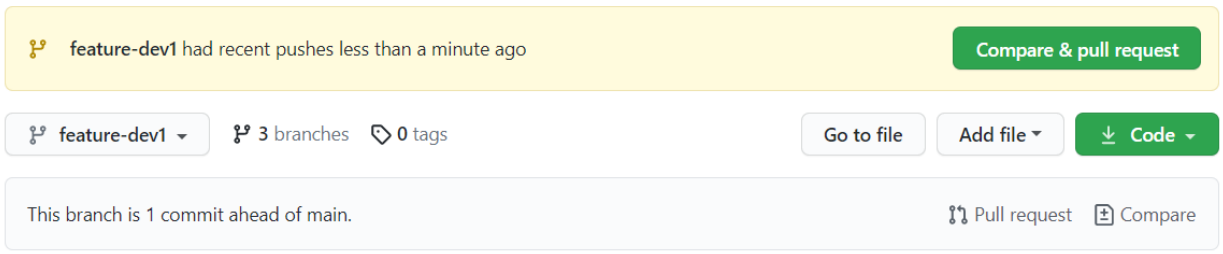
3. Enter your Github account and password if your computer prompts you.
4. Navigate with your web browser to the Github repo and confirm that your feature branch has been uploaded to the repo:



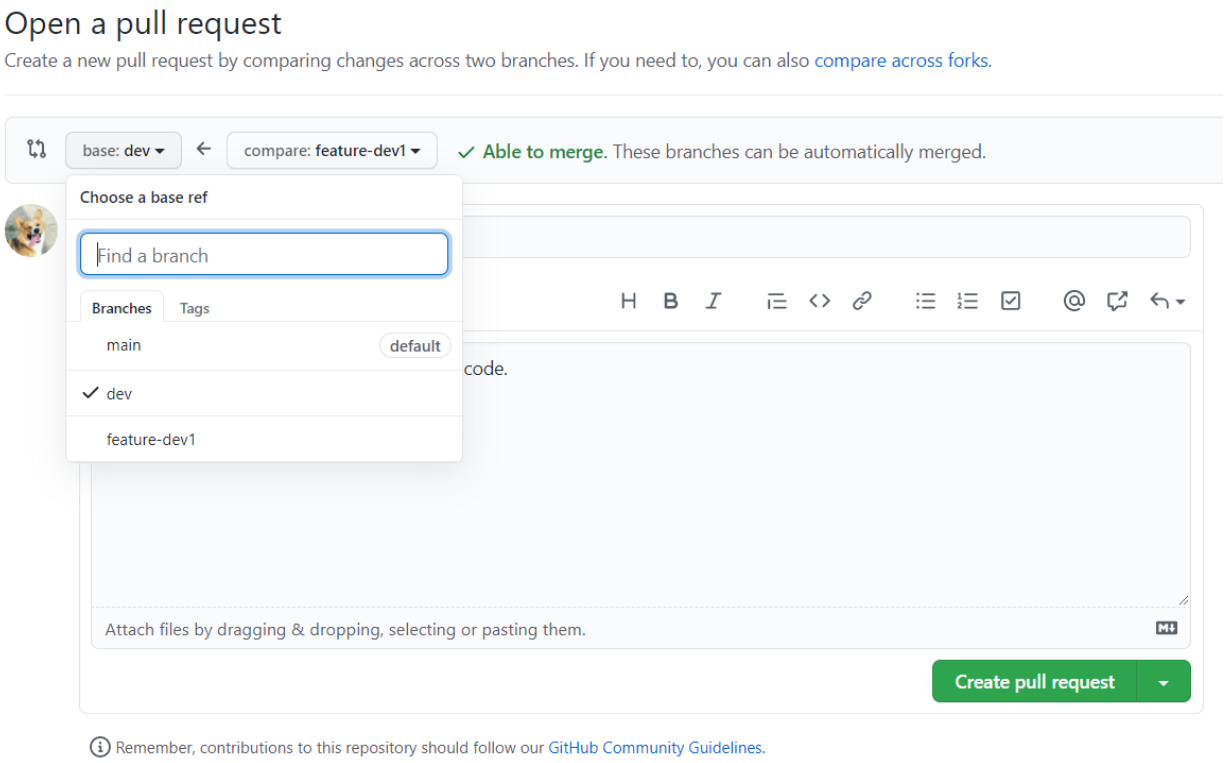
Build Basic Naïve Models

You will be responsible to build the base class models that trainmodel/boston.py depends on.

1. Within the project folder, open `trainmodel/basic_models.py` in your project folder.
2. Replace the current code with the code in `final/trainmodel/basics_models.py`
3. Save and commit the code to your feature-dev1 branch; name your commit appropriately.
4. Push the code to your remote GitHub branch: `git push`
5. Navigate to https://<<yourgithubrepo>>/tree/feature-dev1 and verify that the code has been pushed to GitHub.



6. In your browser, click the link **Pull request** to create a request to merge your code into the **dev** branch.



7. To the upper left, choose the drop-down menu marked with **base: main.**
8. Select **dev** as the branch you want to merge your code into. Add some comments regarding your code change.
9. To the right of the page, click the gear icon next to Reviewers and assign the code review to your project lead.

10. Click the button labeled: **Create pull request** to start the code review process.
11. Notify your project lead about the PR and they will respond to your PR with comments and additional requests through the GitHub interface.

Export the LightGBM Model

Your next step is to finish up the `trainmodel/boston.py` file within the project folder, and export the model as a Python pickle object.

1. Consult with your team on whether Developer 2 has finished up writing code for the results function in the boston.py file. You **cannot continue** until your teammate has finished committing their code and has successfully merged their feature branch into the **dev** branch.
2. Within the project folder, open `trainmodel/boston.py` and confirm that the results function is empty.
3. Close the file `trainmodel/boston.py`.

Create a dev branch

4. Next, create a local **dev** branch and download the code from the remote dev branch into your local dev branch:

```
git checkout -b dev
```

```
git pull origin dev
```

5. Within the project folder, open `trainmodel/boston.py` and confirm that the results function is now complete. The results function was completed by Developer 2.

Delete feature-dev1 branch and create a new feature branch

It is good practice to delete your small feature branches once your feature has been merged into the dev branch.

6. Delete the local feature-dev1 branch: `git branch -D feature-dev1`
7. Next, delete the remote feature-dev1 branch on GitHub: `git push origin -d feature-dev1`
8. Create a new branch with the name: `feature-dev1-exportmodel`
9. Make sure you are now on the feature-dev1-exportmodel branch.

Export the LightGBMModel

10. Within the project folder, open `trainmodel/boston.py.`
11. Scroll towards the end of the file.
1. Replace the Chosen model section with the Chosen model section from `final/trainmodel/boston.py`
12. Save your code.
13. Create two new folders within the project folder: **model** and **test_resources**
14. Save and run the boston.py code: `python boston.py`
15. Verify that four new files have been exported:

```
model/feature_sequence.txt
model/model.pkl
test_resources/data.json
test_resources/score.json
```

16. Add the modified boston.py file and the new files to the Git staging area.
17. Commit your code.
18. Push your code to a new branch (**feature-dev1-exportmodel**) in GitHub.
19. Submit a Pull Request (PR) and assign the code review to your team lead.
20. Notify your team lead for final code review.

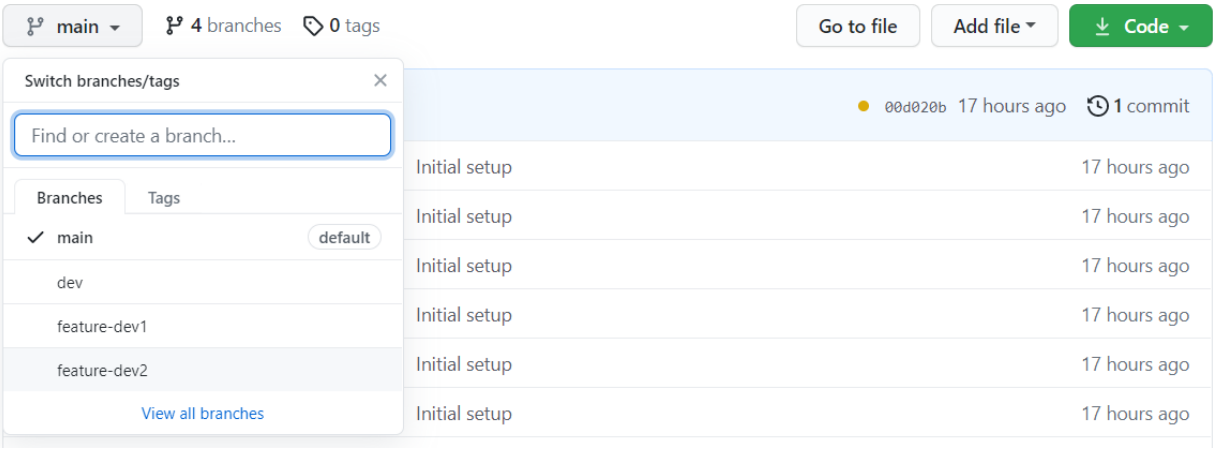
Developer 2

Set Up New Feature Branch

1. Create a new branch named: `feature-dev2`
    a. https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
2. Push your code to a new branch in the GitHub repo:

```
git push --set-upstream origin feature-dev2
```

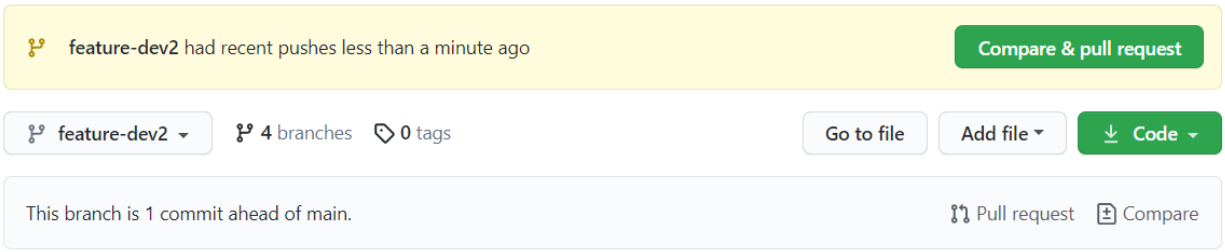
3. Use your web browser to navigate to the Github repo and confirm that your feature branch has been uploaded to the repo:



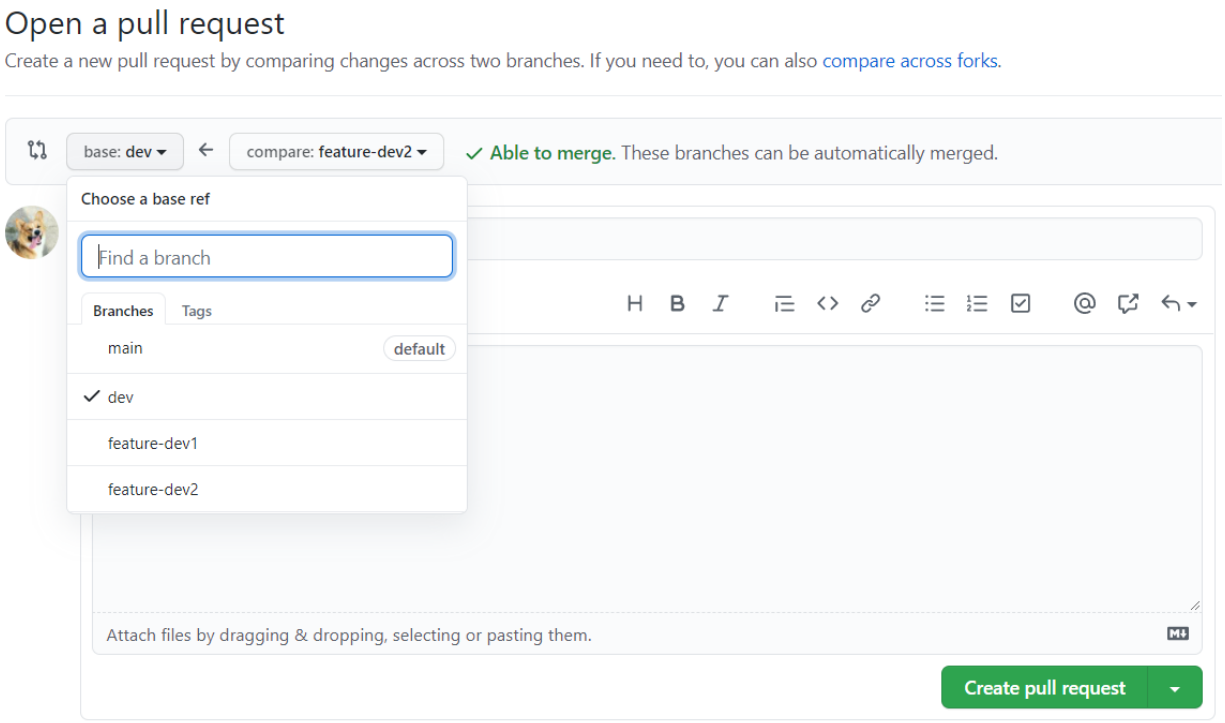
Build results Function

You will be responsible to build out the results function to compare various ML models.

2. Within the project folder, open `trainmodel/boston.py`.
3. Find the results function and replace the function with the code in `final/trainmodel/boston.py`
4. Save and commit the code to your feature-dev2 branch; name your commit appropriately.
5. Push the code to your remote GitHub branch: `git push`
6. Use your web browser to navigate to https://<<yourgithubrepo>>/tree/feature and verify that the code has been pushed to GitHub.



12. In your browser, click the link **Pull request** to create a request to merge your code into the **dev** branch.



13. To the upper left, choose the drop-down marked with **base: main.**
14. Select **dev** as the branch you want to merge your code into. Add some comments regarding your code change.
15. To the right of the page, click the gear icon next to Reviewers and assign the code review to your project lead.

16. Click the button labeled: **Create pull request** to start the code review process.
17. Notify your project lead about the PR and they will respond to your PR with comments and additional requests through the GitHub interface.

Complete the Web Application

You will complete the entire project by loading the exported model into the web application and complete the /score endpoint.

1. Consult with your team on whether Developer 1 has finished up training the ML model and exported the model. You **cannot continue** until your teammate has finished committing their code and has successfully merged their feature branch into the **dev** branch.
2. Confirm that the files **do not** exist in your current project folder:

```
model/feature_sequence.txt
model/model.pkl
test_resources/data.json
test_resources/score.json
```

Create a dev branch

3. Next, create a local **dev** branch and download the code from the remote dev branch into your local dev branch:

```
git checkout -b dev
```

```
git pull origin dev
```

4. Verify that the four previous files mentioned now exist in your project folder; the model has been trained and exported as a Python pickle object under model/model.pkl.

Delete feature-dev2 branch and create a new feature branch

It is good practice to delete your small feature branches once your feature has been merged into the dev branch.

5. Delete the local feature-dev1 branch: `git branch -D feature-dev2`
6. Next, delete the remote feature-dev1 branch on GitHub: `git push origin -d feature-dev2`
7. Create a new branch named: `feature-dev2-scoreendpoint`
8. Make sure you are now on the feature-dev2-scoreendpoint branch.

Import the ML Model and Complete the /score Endpoint

9. Open `app.py`.
10. Replace the import model code with the import model code found in `final/app.py`.
11. Replace the score function with the score function found in `final/app.py`.
12. Save and commit your code.
13. Push your code to a new branch (**feature-dev2-scoreendpoint**) in GitHub.
14. Submit a Pull Request (PR) and assign the code review to your team lead.
15. Notify your team lead for final code review.