Deploying ML to Flask

Learning Objectives

By the end of today's lesson, you will: - Explain what web development is, including many of the associated technologies and frameworks used for deployment - Describe what Flask and Heroku are, and why they're advantageous - Deploy a pickled machine learning (KNN) application to production

Deployment Guide

We will be deploying your very own KNN predictor for the iris dataset. The final product will look like this.

Our Flask app

You'll need to clone this repository but **not in your existing repository.** Do not ever place a repository within a repository. You will have a bad time.

Running it locally

Before running it locally, we need to validate that you have all the requisite packages. To do so, run:

```
pip install –r requirements_clean.txt
```

If that throws an error, you can individually pip install each of the requirements in the requirements_clean.txt file. They are:

```
Flask==0.10
Jinja2==2.6
gunicorn==19.1.1
flask_bootstrap==3.3.5.6
wtforms==2.0.2
Flask-WTF==0.11
simplejson==3.5.3
```

Once you've done this, you can visit your home on port 5000 and see our Flask app!

http://127.0.0.1:5000

Deploying

We'll be deploying to Heroku. That requires a Heroku account. Create one

Next, you need to go to your Heroku dashboard and create a new app. Give it a great name! Like psychic-iris. (But your name must be unique).

We now need to install the command line interface (CLI) tool for Heroku. Because we installed homebrew during installfest, we'll use brew to grab this tool:

```
brew install heroku
```

Great! Now, to connect our local code to Heroku, we're going to create a branch using git. We will push our changes from our local machine to that Heroku branch. (Think: just like when we push our code to a remote repository on git.) We're going to add our app name (that you created above) as a remote branch, replacing with the name of your app.

```
heroku git:remote —a <app>
```

The next step is super intuitive, but it's easy to implement. When we push to Heroku, bear in mind that the servers we're renting from Heroku will not have the required pre-requisites that our local environment has. As a result, we need to add what's called a buildpack: a set of packages that will install dependencies on our rented server at the time of deployment. Fortunately, there's already a buildpack created for what pre-requisite we need. It's the included URL below:

```
Heroku config:set BUILDPACK_URL=https://github.com/thenovices/heroku-buildpack-sci
py app=<app>
```

Now we can deploy! We'll push to the Heroku master branch (not origin master), and the build will begin! It will take a little while on the first time. Go to your Heroku app dashboard, and watch the build complete. In the upper righthand corner, you will have an option to "Open app"

''' Git push heroku master '''

Footnote: killing that port

We opened a port (5000) on your local machine that we need to close!

To do that, complete the follow steps:

This returns all active python processes: `ps aux | grep python`

You then want to find the PID for the process you want to eliminate. For our example, this will be the

process running `controller.py`. To kill that process: `sudo kill -9 <PID>` (where PID is the process you want to kill -- the five digit number)

Resources

- Here's a nice Medium post on using scikit learn in a Flask app as an API