

Data Science Project Set-Up Guide

Tanu Nanda Prabhu

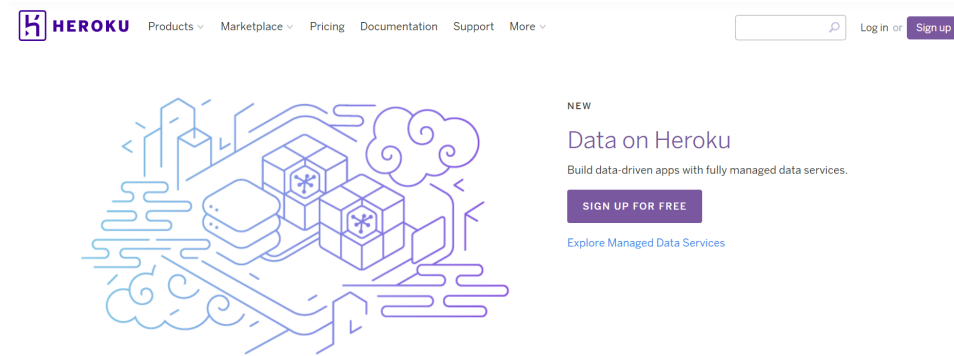
University of Regina, Canada



Contents

1	Heroku - Cloud Application Platform Setup	3
2	Installations	6
2.1	Heroku CLI	6
2.2	Install dependencies	7
2.3	Create a Procfile	8
3	Steps to run the code	9
3.1	Steps to run the notebooks	9
3.2	Steps to deploy the code	10
4	Working of Django framework	11
5	Running the code on your (local) system	13

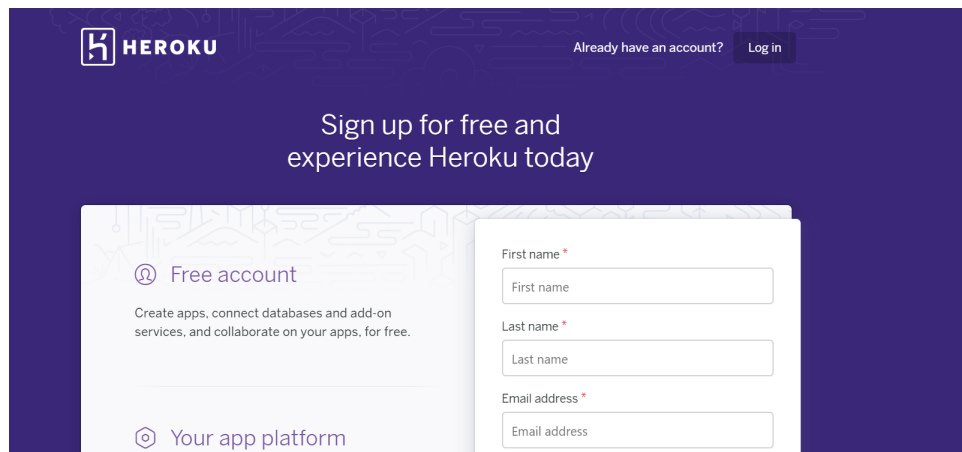
1 Heroku - Cloud Application Platform Setup



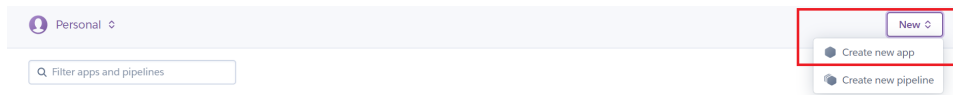
Heroku makes it easy to deploy and scale Python apps. Whether you prefer frameworks like Django or Flask, or getting your hands dirty with Twisted or raw sockets, Heroku helps you build things your way with the tools you love. Source: [Heroku](#)

The steps for setting up a Heroku profile and deployment guide is as shown below:

1. Create a [free new account](#) on heroku or [login](#) if you have an existing one. Don't worry heroku will not ask your credit card details



2. Create a [new app](#). Added all the necessary details like app name and location, etc. This will just create a web application to you.



As of now there are only two locations USA and Europe (Select anyone)

A screenshot of the 'Create New App' form in Heroku. The form has a light purple header with the text 'Create New App'. Below the header, there's a section for 'App name' with a text input field containing 'app-name'. Underneath is a 'Choose a region' section with a dropdown menu showing 'United States' and a small flag icon. Below the region selection is a section titled 'Add to pipeline...' with a purple icon and text. At the bottom of the form is a large purple button labeled 'Create app'.

3. Add one of the [deployment methods](#) (GitHub). There are **Heroku Git**, **GitHub**, **Container Registry**.

Please select the **GitHub** as the deployment method. Because it is one of the easiest methods out there.

A screenshot of the Heroku deployment method selection screen. The screen is divided into two main sections. The top section is titled 'Add this app to a pipeline' and contains two columns of information. The left column explains how to create a new pipeline or choose an existing one. The right column explains how to connect the app to a pipeline to enable additional features. Below this is a 'Choose a pipeline' dropdown menu. The bottom section is titled 'Deployment method' and contains three options: 'Heroku Git' (Use Heroku CLI), 'GitHub' (Connect to GitHub), and 'Container Registry' (Use Heroku CLI). The 'GitHub' option is highlighted with a red rectangular box. Below the deployment method selection is a section titled 'Connect to GitHub' which contains a search bar for a repository to connect to, a dropdown for selecting a GitHub account, and a 'Search' button.

Make sure that you connect your GitHub account and select your repository

Don't worry if your don't have a repositor use [my repository](#) as a reference. For that you need to fork or clone my repository.

This is because to deploy in a Heroku cloud we must write the code using a Django or Flask python web framework. Only then you can successfully deploy the code to the cloud. To read more details this process visit [Django](#) and [Flask](#)

4. Deploy the branch. Here you can select two options **Automatic deploy** and **Manual Deploy**. Just select **Manual Deploy** as of now. The whole point is to deploy the options doesnot matter.

Automatic deploys
Enables a chosen branch to be automatically deployed to this app.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically**; be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more](#).

Choose a branch to deploy

master

☐ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys

Manual deploy
Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

master

Deploy Branch

It might take some time to deploy you can obviously see all the commands, logs that would be running background. Once everything is deployed you will get a link that you can copy and paste it in your favourite browser. This is how you can successfully deploy any app on Heroku cloud. The above is just a example which shows you how to create a account and keep going with an example.

Every time you change code, don't forget to commit the changes on GitHub and simultaneously deploy the changes on Heroku. Here you can use the automatic deploy method.

2 Installations

2.1 Heroku CLI

The [Heroku Command Line Interface \(CLI\)](#) makes it easy to create and manage your Heroku apps directly from the terminal. It's an essential part of using Heroku. Below is the download options for different operating systems

1. Mac OS
 - (a) Download [here](#) or
 - (b) `brew tap heroku/brew && brew install heroku`
2. Windows OS
 - (a) 64-bit Installer download [here](#) or
 - (b) 32-bit Installer download [here](#)
3. Ubuntu 16+
 - (a) Run the following from your terminal: `sudo snap install --classic heroku`

Then you need to login to heroku to use the CLI interface. Go to your command prompt and type `heroku login`

```
$ heroku login
heroku: Press any key to open up the browser to login or q to exit
> Warning: If browser does not open, visit
> https://cli-auth.heroku.com/auth/browser/**
heroku: Waiting for login...
Logging in... done
Logged in as me@example.com
```

This is really helpful because whenever your app crashes you can access the crash report via the heroku logs as `heroku logs -app appname`.

```
heroku logs

display recent log output

USAGE
$ heroku logs

OPTIONS
-a, --app=app      (required) app to run command against
-d, --dyno=dyno    only show output from this dyno type (such as "web" or "worker")
-n, --num=num      number of lines to display
-r, --remote=remote git remote of app to use
-s, --source=source only show output from this source (such as "app" or "heroku")
-t, --tail         continually stream logs
--force-colors     force use of colors (even on non-tty output)

DESCRIPTION
disable colors with --no-color, HEROKU_LOGS_COLOR=0, or HEROKU_COLOR=0

EXAMPLES
$ heroku logs
2012-01-01T12:00:00+00:00 heroku[api]: Config add EXAMPLE by email@example.com
```

To read the documentation of Heroku CLI commands click [here](#)

2.2 Install dependencies

Below shown are some important requirements that you need to install and include it as a `.txt` file in your GitHub file. This is how Heroku will recognise all the dependencies and install it for your web application. Install the necessary libraries in your terminal (Command Prompt), because you need to manually install them. There are two ways you can do this.

1. Download the [requirements.txt](#) file

Most importantly make sure you install Python first. Download [here](#)

1. `Django==3.0.3`
2. `googletrans==2.4.0`
3. `gunicorn==20.0.4`
4. `matplotlib==3.1.3`
5. `numpy==1.18.1`
6. `pandas==1.0.1`
7. `scikit-learn==0.22.2.post1`

To install any of the above library navigate to your command prompt and then tell `pip install library-name`. If you don't have `pip` installed already then you might see a error. Download it [here](#)

2.3 Create a Procfile

Heroku apps include a Procfile that specifies the commands that are executed by the app on startup. The Procfile is always a simple text file that is named **Procfile** without a file extension. For example, **Procfile.txt** is not valid. To read more about Procfile click [here](#).

```
<process type>: <command>
```

- `<process type>` is an alphanumeric name for your command, such as `web`, `worker`, `urgentworker`, `clock`, and so on.
- `<command>` indicates the command that every dyno of the process type should execute on startup, such as `rake jobs:work`.

Click [here](#) to access my [procfile](#). Add your application name in the place of my application name.

All the above changes and installations files must be updated on GitHub. Because Heroku only depends and believes GitHub

3 Steps to run the code

3.1 Steps to run the notebooks

Below are the steps to execute the notebooks

1. Download the [raw](#) data set or [cleaned data](#) set to your local system.
2. Install [Jupyter Notebook](#) or use [Google Colab](#).

***Note:** Use anaconda jupyter notebook while building the model. Because this is much faster than google colab. Also make sure you install all the necessary libraries shown above. Also make sure your system RAM and disk space is free.*

3. Download the notebooks to execute. You can individually download the below notebooks and then upload to your jupyter environment.
 - (a) Download the [Discovery notebook](#)
 - (b) Download the [Data preparation notebook](#) to perform data exploration
 - (c) Download the [Model Planning](#).
 - (d) Download the [Model Building](#)
 - (e) Download additional notebooks which has the [manual](#) and [grid search](#) parameter tuning of the model
4. Run all the cells of the desired notebooks. (NOTE: You have to run each and every phase of my project notebooks for successful execution). **To execute in google colab, copy the google drive dataset and export it to your own drive.** More details of the necessary files can be found on my [GitHub repository](#)

3.2 Steps to deploy the code

As explained above I have used Heroku Cloud service to deploy and host my code. To install and configure please visit to section

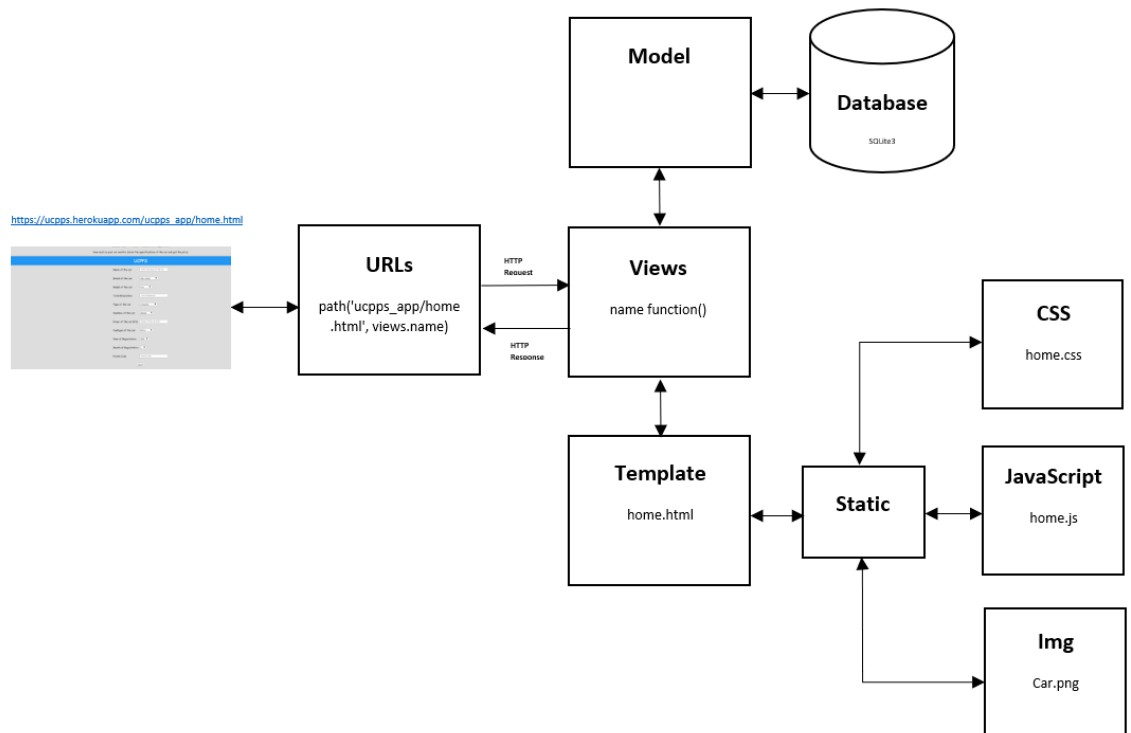
1. Clone or download my complete code deployment folder from my [GitHub repository](#).
Or run the command
`https://github.com/Tanu-N-Prabhu/Used_Car_Price_Prediction_System1.git`
in your command prompt.
2. Connect your GitHub account and your repository (the downloaded repository) to heroku as shown above in section 1.
3. Make sure all both the `requirements.txt` and `Procfile` are present in your folder or repository.
4. Deploy the application. Usually this might take some time depending on the internet connection or local system speed. After a few minutes you will get a successful log message along with the application link as shown below:

```
1 -----> Python app detected
2
3 -----> Installing SQLite3
4         SQLite3 successfully installed.
5
6 -----> Installing requirements with pip
7 -----> $ python manage.py collectstatic --noinput
8
9         146 static files copied to
10
11         '/tmp/build_bf9922d1c1d6dba410b1ffe54daf6c4d/assets'.
12
13 -----> Discovering process types
14         Procfile declares types -> web
15 -----> Compressing...
16         Done: 335.4M
17
18 -----> Launching...
19         !Warning: Your slug size (335 MB) exceeds our soft
20         limit (300 MB) which may affect boot time.
21
22         Released v9
23         https://ucpps.herokuapp.com/ deployed to Heroku
```

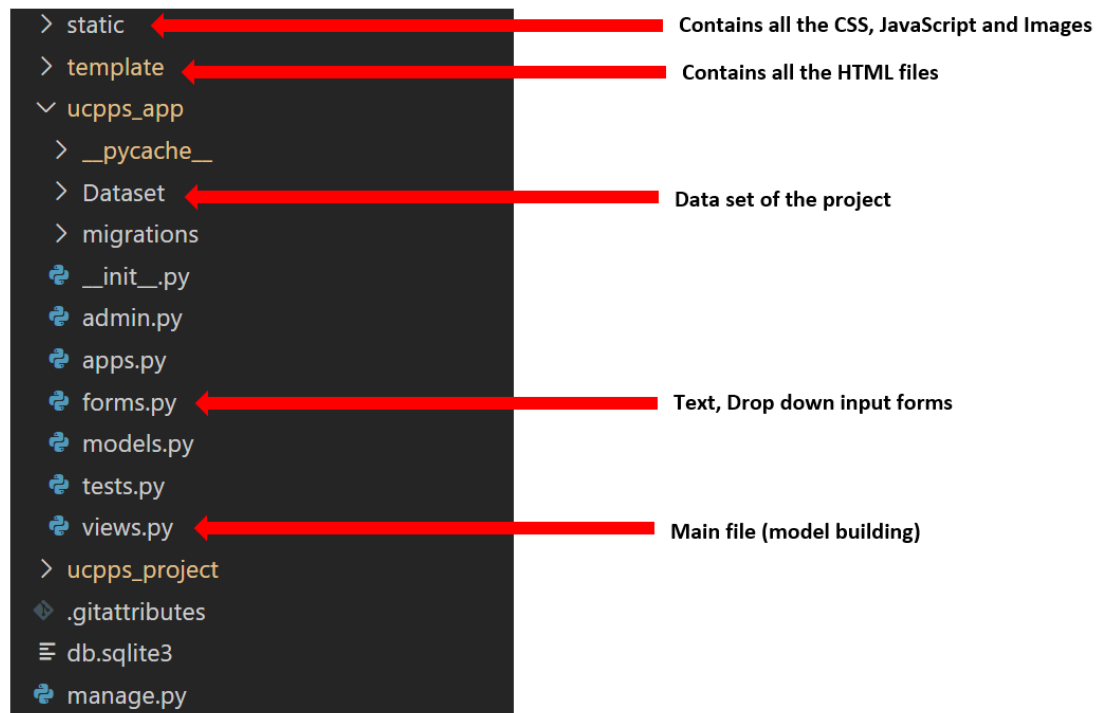
My web application link is https://ucpps.herokuapp.com/ucpps_app/home.html

4 Working of Django framework

Django has a Model View Template software architecture. To briefly explain the Model is more likely a database, here is where you write all the programs that has to do with the database. The view is the place where we write the main code. Here all the model code (main file) is written. The template comprises all the front-end codes (HTML). Also, there is a static folder which has all the necessary CSS, JavaScript and Images folder. Django is loosely coupled meaning all the components are independent. If we need to change the front-end, then we need not touch the main code rather we need to navigate the template folder. Similarly, if we need to update the database, then we need to navigate to the model folder and then change the code without touch any other component. Below is the brief MVT diagram which summarizes the explanation.



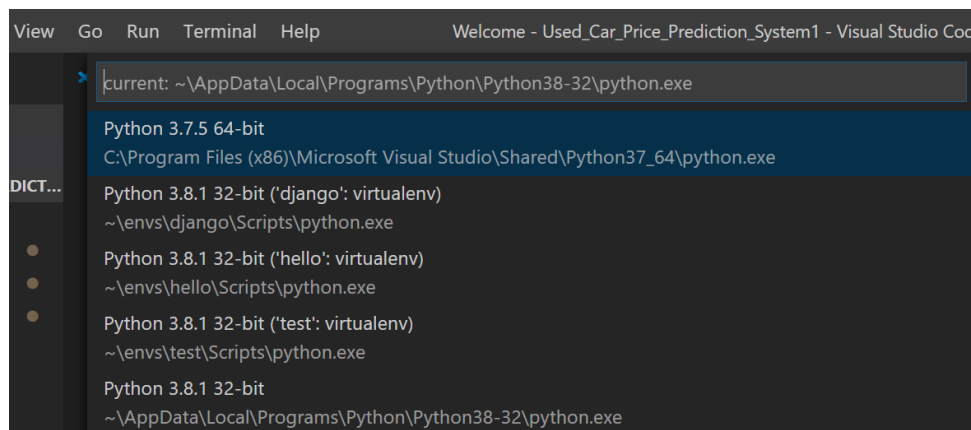
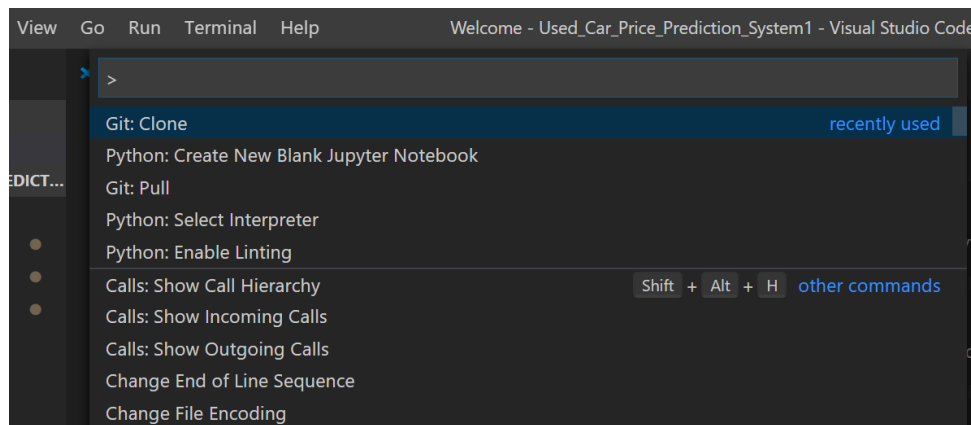
Also below is the just a handout guide which can be used to see where all the files and the code could be placed and written.



5 Running the code on your (local) system

Below are the steps that you need to follow to run the code in your local system. Here the main catch phrase is you need to install an IDE. I prefer [Visual Studio Code](#). This is because for the hosting purpose I have extensively used Django Web framework. Because with the help of this I could build and develop a web application. With the help of Google Colab or Jupyter Notebook, we cannot create a web application. These environments were primarily used to data pre-processing, model planning and building. You can run all the notebooks (.ipynb) file in your notebook environment one by one. But to build and develop the front-end Django was used, and I used Visual Studio IDE along with it.

1. Install Django (`pip install Django`) and [Microsoft Visual Studio Code](#)
2. Clone your GitHub repository this is because only then you can work with the base code. Also don't forget to select the python interpreter. For both of this operations you can use the **View** to get the command as shown below:



Alternatively if you want to start a new app use `django-admin startapp my_new_app`

3. After successfully cloning the repository you can execute the following Django application by saying `python manage.py runserver`. By executing the command, the application will start running on your local system and server as shown below. You can navigate and see the beautiful website by clicking on the below link from your respected terminal.

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\tanup\OneDrive\Desktop\Used_Car_Price_Prediction_System1> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
April 16, 2020 - 19:25:27
Django version 3.0.3, using settings 'ucpps_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```