Local Model Deployment Walkthrough
Notebook & Resource Guide

This guide is intended to be supplemental to the "Model Building & Deployment with H2O Flow" Jupyter Notebook.

Step 1:
Download the zipped file which contains the (6) files needed for following along with the guide and ensure that they are saved in the same directory on your local machine:

1. "Model Building & Deployment with H2O Flow.ipynb" – the Jupyter Notebook
2. training_data.csv – 35k records for training your model
3. hold_out_data.csv – 15k records to score from using your trained model
4. model_call.py – a Python-script for scoring your hold_out_data.csv locally
5. flask_app_demo.py – a Python-script to locally serve the API-endpoint and render the HTML for scoring new instances
6. api_calls.py – a Python-script to locally score data against your model via the API-endpoint exposed through the script above
7. h2o-genmodel.jar – a generic jar-file needed to decouple the H2O model from the package used to create it
8. grid_ba7754d4_6682_4d3b_9283_846b4ea90764_model_17.zip – the MOJO-file for the model we trained during class

Step 2:
Open the "Model Building & Deployment with H2O Flow.ipynb" and **review the dependencies** listed in the first cell. You'll need to ensure that you **have the full list of packages installed** on your machine before proceeding. All packages should be candidates for a simple – "pip install" – meaning that there are no major hurdles to cross (other than ensuring each package is installed; and being mindful of versions)

Step 3:
Enter your local working directory (i.e., the folder location of the files you downloaded and saved in Step 1 above) in **1.1 - Set Your Working Directory** (image below; simply type in your folder location using the "text box" preceded by the "Directory:" label)

Step 4:
If you've correctly entered in your directory in Step 3 and have saved the (3) input files to the directory entered, you should be able to run each cell with the default inputs until

you reach the header labeled **1.3 - Select a Data Frame.** The first input box will be preselected for you, but you'll need to select which variables to include from the selected dataframe in the next cell (image below). Highlight all variables **excluding the "name" variable** at this time.

```
In [24]: review_variables = widgets.SelectMultiple(
             options=master_data[dict_keys.value].columns.tolist(),
             description='Variables:',
             disabled=False,
             layout=Layout(width='50%')
         )

         display(review_variables)
```

Variables: id, name, category, main_category, currency

The model included with this walkthrough uses the following variables:

```
In [29]: review_variables.value

Out[29]: ('id',
          'main_category',
          'goal',
          'state',
          'country',
          'launch_month',
          'launch_dow',
          'duration')
```

Step 5:

Continue running each subsequent cell using the default inputs until you reach the cell with the header **2.1 - Select Your Target Variable** where you'll need to make (2) selections:

1. Choose a "Target" variable from the list of columns available – **choose the "state" variable**
2. Choose a "Target-type" from the list (either Continuous or Categorical) – **choose the "Categorical" label**

```
In [31]: target = widgets.Select(
             options=master_data['custom_table'].columns.tolist(),
             description='Target',
             disabled=False
         )

         target_type = widgets.Select(
             options=['Continuous','Categorical'],
             description='Type',
             disabled=False,
         )

         display(target)
         display(target_type)
```

Target: id, main_category, goal, state, country

Type: Continuous, Categorical

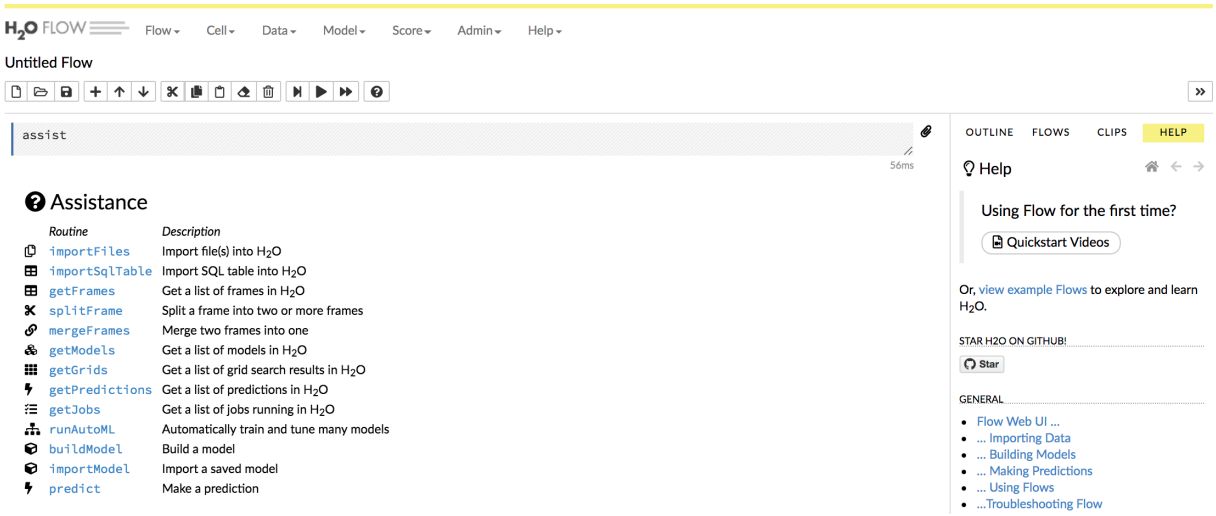
Step 6:
Initiate H2O by running the cell with the header **2.2 – Initiate H2O** just below those shown in Step 5. If you've successfully launched h2o, you should see the following output in your notebook:

```
Connecting to H2O server at http://127.0.0.1:54321 ... successful.
```

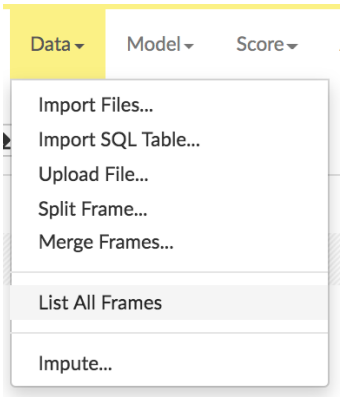
Step 7:
Run the subsequent cells using the default settings until you reach the header **2.3 - Configure Models**. At this point in the guide, you're ready to switch from the Jupyter Notebook to h2o.ai's Flow browser-based tool. You should be able to click:

http://127.0.0.1:54321

Which will bring you to a browser that should look like this:

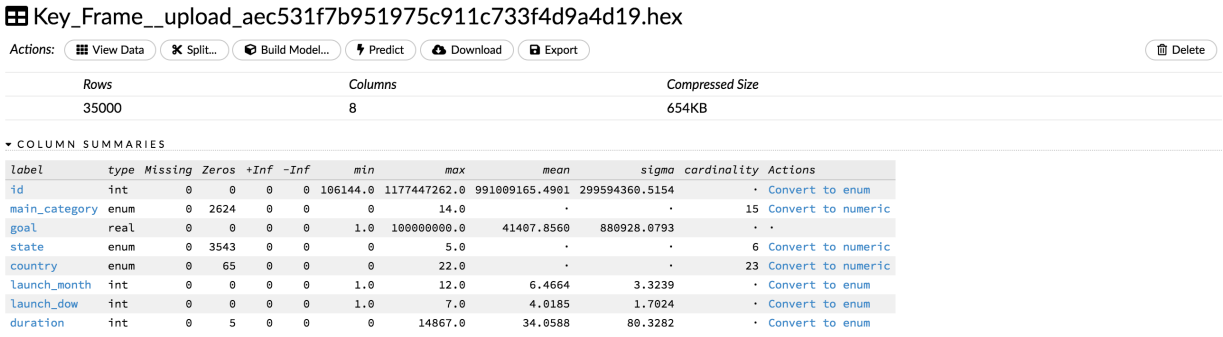


Step 8:
From the toolbar at the top of the page, click the down caret next to "Data" and select "List All Frames"

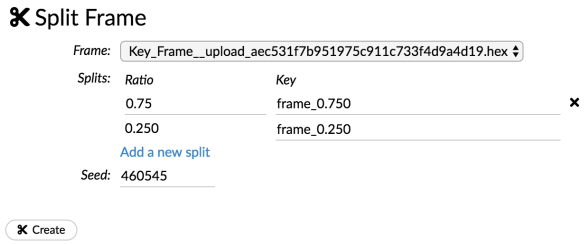


Step 9:
Look for a “Key_Frame” in the menu that appears that resembles the one shown below. This is the “training_data” and it contains the 35k records we loaded into our Juypter Notebook. Click on the ID (blue-text) to view the frame summary:

Step 10:
In the next menu that appears, click the “Split” button in the frame summary toolbar:



Step 11:
Enter a split ratio (H2O Flow defaults to a 75/25-split) and click the “Create” button:



Step 12:
Two new frames will appear (if you ran the defaults). Click the “frame_0.750” (blue-text) below (Note: your name may be different if you changed the defaults above):

Which will bring up the frame summary for the “training data” (which is represented by the frame_0.750 frame above. Click the “Build Model” button which is visible in the frame summary toolbar.

frame_0.750

Actions: View Data | Split... | Build Model... | Predict | Download | Export | Delete

Rows	Columns	Compressed Size
26286	8	493KB

COLUMN SUMMARIES

label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality	Actions
id	int	0	0	0	0	106144.0	1177447262.0	990787021.6167	299621154.2972	·	Convert to enum
main_category	enum	0	1966	0	0	0	14.0	·	·	15	Convert to numeric
goal	real	0	0	0	0	1.0	100000000.0	39668.2047	933208.9103	·	·
state	enum	0	2640	0	0	0	5.0	·	·	6	Convert to numeric
country	enum	0	50	0	0	0	22.0	·	·	23	Convert to numeric
launch_month	int	0	0	0	0	1.0	12.0	6.4594	3.3223	·	Convert to enum
launch_dow	int	0	0	0	0	1.0	7.0	4.0223	1.7038	·	Convert to enum
duration	int	0	4	0	0	0	14867.0	34.2049	92.3938	·	Convert to enum

Step 13:

Use the “Select an algorithm” menu to begin the model building:

Build a Model

Select an algorithm: (Algorithm)

Selecting “Gradient Boosting Machine” will return the following parameters:

Build a Model

Select an algorithm: Gradient Boosting Machine

PARAMETERS — GRID?

model_id	gbm-ec4fe3bd-2320-4b7c-9928-effb617b297d	Destination id for this model; auto-generated if not specified.
training_frame	frame_0.750	Id of the training data frame.
validation_frame	frame_0.250	Id of the validation data frame.
nfolds	0	Number of folds for K-fold cross-validation (0 to disable or >= 2).
response_column	state	Response variable column.
ignored_columns	Search...	

Showing page 1 of 1. 1 ignored.

id	INT
main_category	ENUM(15)
goal	REAL
state	ENUM(6)
country	ENUM(23)
launch_month	INT
launch_dow	INT
duration	INT

All | None | ← Previous 100 | Next 100

Click the “check boxes” on the right-hand side of the window to perform a “grid search” on specific parameters (example shown below):

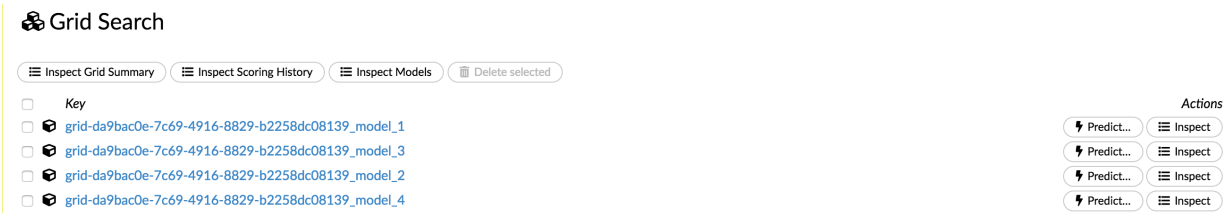
ntrees	50;100	Number of trees.
max_depth	5;6	Maximum tree depth.

Once you have made your inputs, scroll to the bottom and click “Build Model”

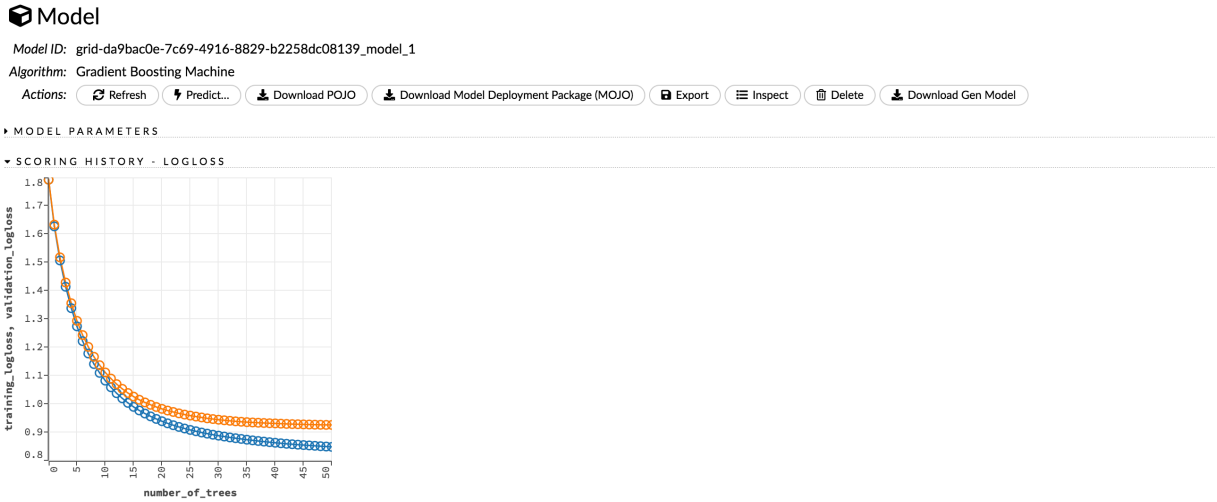
Build Model

Step 14:
Once the model has been built, click the “View” button:

If you trained multiple models (by using grid search), the “View” button will expose a list of models (if one model was trained; the view will jump directly to the model summary). Assuming that you used grid search, you should see a table similar to the one below (which contains a ranked-list of model ids):



Step 15:
After you have inspected the models listed above (using the other buttons in the view’s toolbar), click on your desired key (blue-text) to expose the model summary (you should see a view similar to the one below):

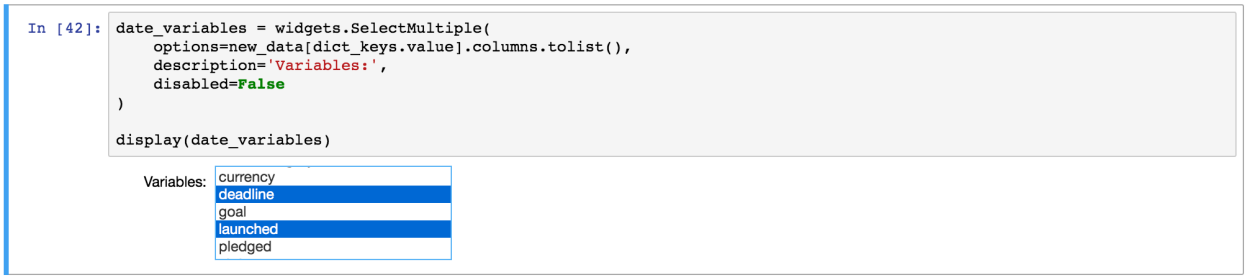


Scroll down in the view to see other important model attributes such as the confusion matrix for both the training and testing set, plotting of the important variables, and drilldowns for exposing the parameters that were used to train the current model. Note: the options I listed may not be visible for other algorithms you may select.

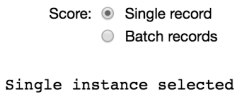
Step 16:
If after evaluating your model you decide to deploy it, click on the "Download Model Deployment Package (MOJO)" and "Download Gen Model" buttons in the model summary toolbar. Doing so will download the (2) files needed for using your model to score new data in subsequent steps. Move the downloaded files to the same folder location as the one holding your other scripts.

Step 17:
At this point, return back to the Jupyter Notebook and jump to **3.0 – Score a New Dataset** (Note: if you closed your kernel between the steps listed above, you'll need to rerun the Jupyter Notebook to this point). If you're following along with this workflow, you should be able to run the cells to load the "hold_out_data.csv" without making any changes.

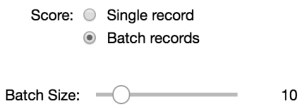
Step 18:
In **3.2 – Select a Data Frame to be Scored**, you have the option to change the data type of any "date fields" in your dataset to be consistent with the date format of H2O. You can skip this step if no date fields were used in your model training exercise.



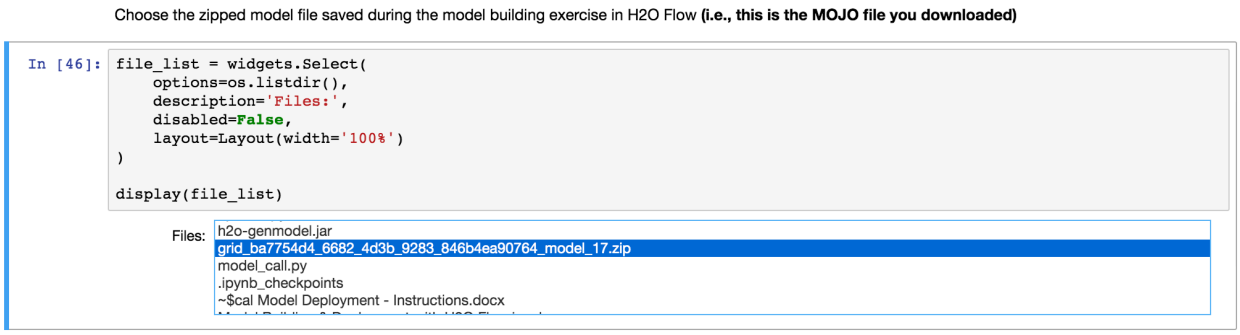
Step 19:
In **3.3 – Single of Batch Scoring**, you have the option to pass one record through the scoring process or multiple records:



If Batch records are selected, you have the option of determining how many records:



Step 20:
Proceed until you see the menu to select the zipped file containing your model. This is the MOJO file you downloaded from H2O flow (select it from the menu box)



Step 21:
You should be able to run the remaining cells completely through to see your predicted results (using your saved model on the new data):

In [50]:
```
pf
```

Out[50]:

	labelIndex	label	classProbabilities
0	3	successful	[0.061620989023153, 0.326686199528232, 0.00072...
1	1	failed	[0.088079156575999, 0.6546569156541711, 0.0007...
2	1	failed	[0.09491996702034801, 0.462752264713559, 0.000...
3	1	failed	[0.087470593604413, 0.624300565412503, 0.00097...
4	1	failed	[0.198709362110981, 0.658420763849431, 0.00087...
5	1	failed	[0.169101948636207, 0.739902424551629, 0.00168...
6	1	failed	[0.09227236518715701, 0.6221993440918401, 0.00...
7	1	failed	[0.11118028030966001, 0.6583856934378041, 0.00...
8	1	failed	[0.053858366785661006, 0.673290680746043, 0.00...
9	3	successful	[0.032201695411663005, 0.19140933937970603, 0....

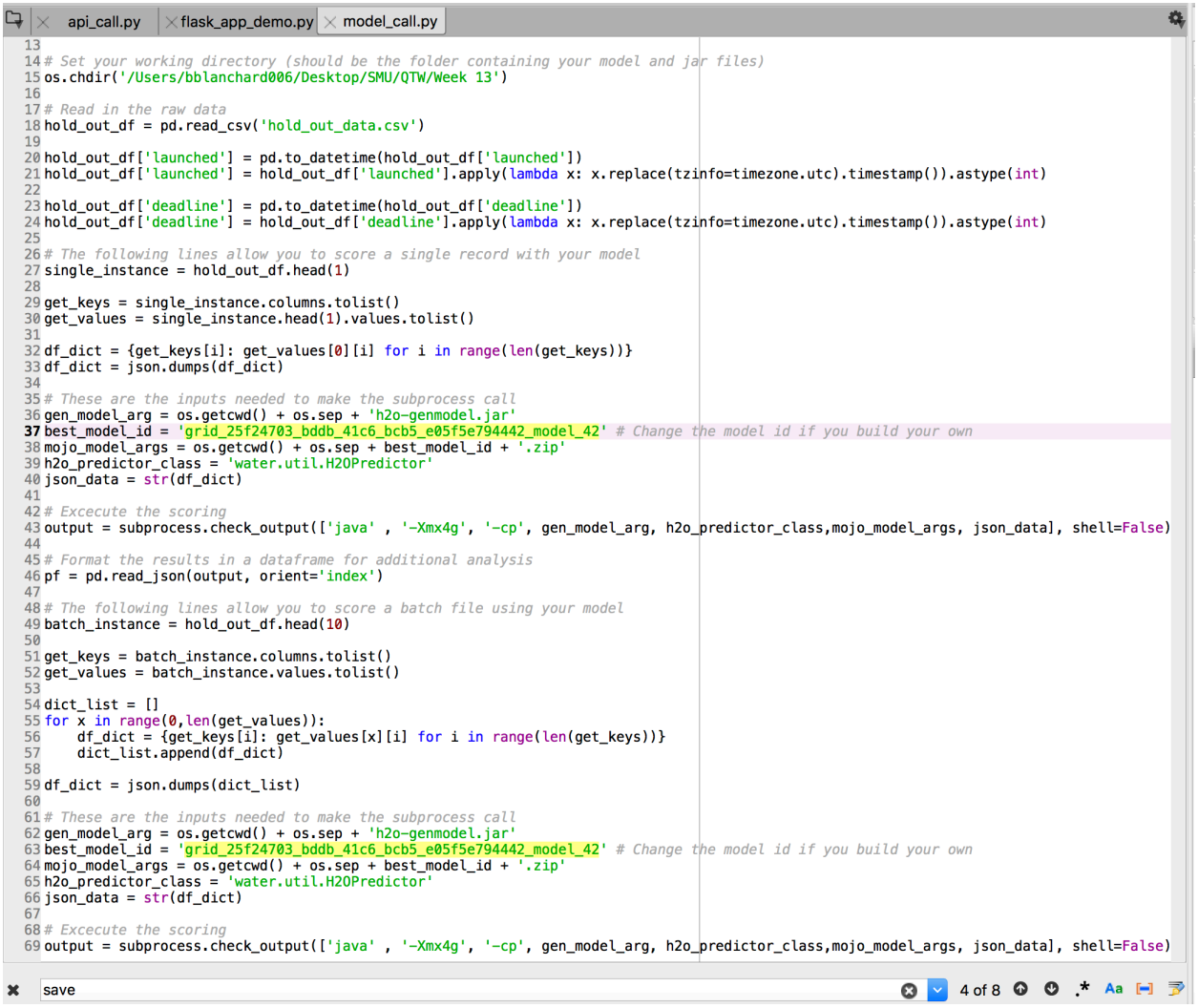
(Instructions continued on next page)

Model Call Python-script

Supplemental to the guide documented above

Open the model_call.py script in your IDE (Spyder shown below). You will need to make the following changes to the script once downloaded.

1) Change the working directory (line 15 below)
2) Overwrite the “best_model_id” value (in yellow below) to the one you created



You should be able to run the script line-by-line to perform both the single-instance and batch-instance scoring found in the Jupyter Notebook.

Flask App Demo Python-script

Supplemental to the guide documented above

Open the flask_app_demo.py script in your IDE (Spyder shown below). You will need to make the following changes to the script once downloaded.

1) Change the working directory (line 27 below)
2) Overwrite the “best_model_id” value (in yellow below) to the one you created

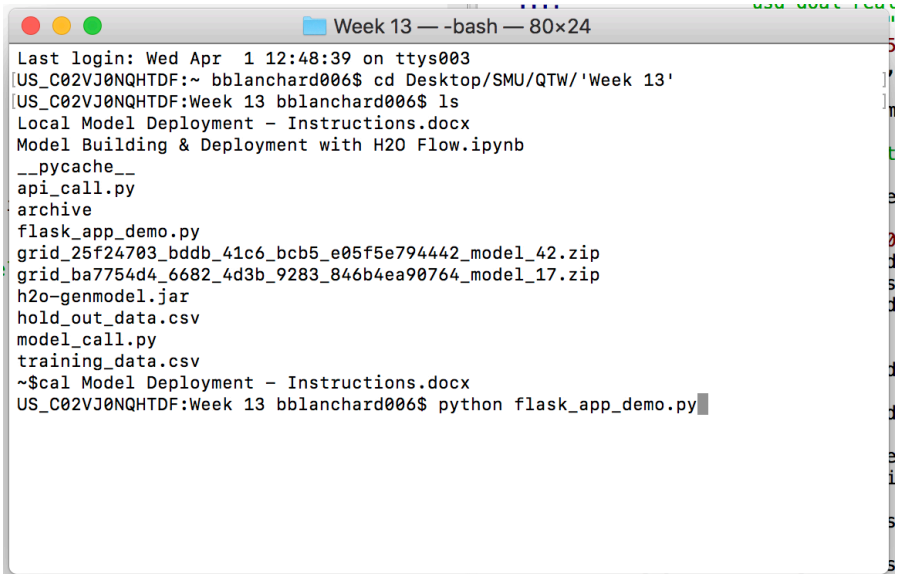
```
api_call.py | flask_app_demo.py | model_call.py

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  """
4  Created on Mon Mar 30 22:49:12 2020
5
6  @author: bablanchard
7  """
8
9  from flask import Flask, jsonify, request
10 import pandas as pd
11 import os
12 import subprocess
13 import json
14
15 app = Flask(__name__)
16
17 @app.route('/predict', methods=['POST'])
18 def predict():
19
20     test_json = request.json
21     test = pd.read_json(test_json, orient='records')
22     record_ids = test['ID']
23
24     try:
25         os.chdir(os.path.dirname(os.path.realpath(__file__)))
26     except:
27         os.chdir('/Users/bblanchard006/Desktop/SMU/QTW/Week 13')
28
29     gen_model_arg = os.getcwd() + os.sep + 'h2o-genmodel.jar'
30     best_model_id = 'grid_25f24703_bddb_41c6_bcb5_e05f5e794442_model_42' # UPDATE MODEL ID HERE
31     mojo_model_args = os.getcwd() + os.sep + best_model_id + '.zip'
32     h2o_predictor_class = 'water.util.H2OPredictor'
33     json_data = str(test_json)
34
35     output = subprocess.check_output(['java' , '-Xmx4g', '-cp', gen_model_arg, h2o_predictor_class,mojo_model_args, json_data], shell=F
36     pf = pd.read_json(output, orient='records')
37     full_frame = pd.concat([record_ids.reset_index(drop=True), pf], axis=1)
38
39     responses = jsonify(predictions=full_frame.to_json(orient="records"))
40     responses.status_code = 200
41
42     return (responses)
```

Scroll down in the script to reveal the second model id reference:

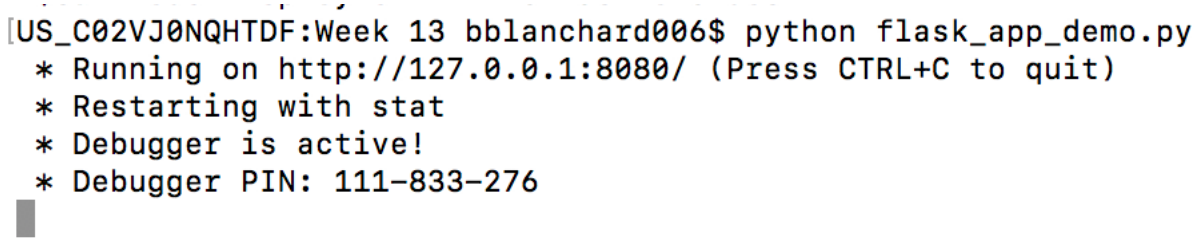
```
96 @app.route('/result', methods=['POST','GET'])
97 def result():
98
99     if request.method == 'POST':
100        name = request.form['name']
101        main_category = request.form['main_category']
102        goal = request.form['goal']
103        country = request.form['country']
104        launch_month = request.form['launch_month']
105        launch_dow = request.form['launch_dow']
106        duration = request.form['duration']
107
108        dframe = [{
109                   "name":name,
110 #                  "category":category,
111                   "main_category":main_category,
112                   "goal":goal,
113                   "country":country,
114                   "launch_month":launch_month,
115                   "launch_dow":launch_dow,
116                   "duration":duration,
117        }]
118
119        dframe = json.dumps(dframe)
120
121    try:
122        test_json = dframe
123        test = pd.read_json(test_json, orient='records')
124
125    except Exception as e:
126        raise e
127
128    gen_model_arg = os.getcwd() + os.sep + 'h2o-genmodel.jar'
129    best_model_id = 'grid_25f24703_bddb_41c6_bcb5_e05f5e794442_model_42' # UPDATE MODEL ID HERE
130    mojo_model_args = os.getcwd() + os.sep + best_model_id + '.zip'
131    h2o_predictor_class = 'water.util.H2OPredictor'
132    json_data = str(test_json)
```

(Instructions continued on next page)

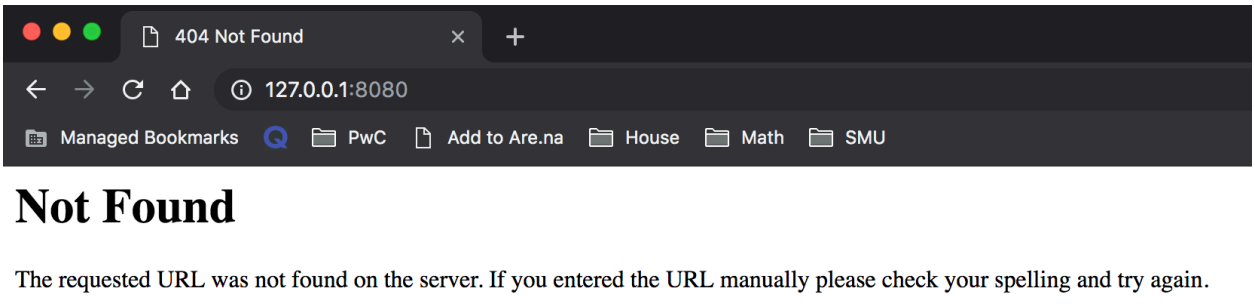
Navigate to your terminal and change your current directory to the one storing the files you have downloaded in this guide. Once there, type in **python flask_app_demo.py**



If execute successfully, you should see the following in your terminal:

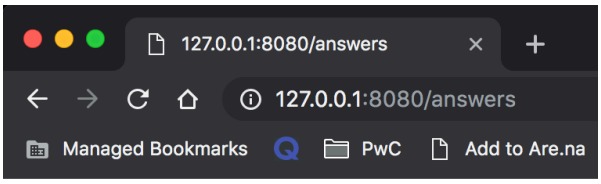


Depending on your configuration, you should be able to navigate in your browser to the IP-address found in the text above (for example: http://127.0.0.1:8080/) which will show the following on the page:



This result is to be expected, since we have not appended an “app-route” (or page) to that domain locally. From your browser, add “/answers” to the end of the URL (ex.; 127.0.0.1:8080/answers)

Doing so should return the following view in your browser:



Campaign Inputs

Campaign Name: The Everyday Project

Goal Amount: 1 50000

Month of Launch: 8

Weekday of Launch: 3

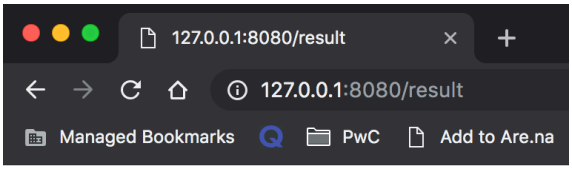
Duration of Campaign: 21

Select Main Category: Film & Video

Select Country: US

Submit

Hitting the submit button (before or after changing the input values) will return:



	Values
Campaign Name	The Everyday Project
main_category	Film & Video
goal	25000
country	US
launch_month	8
launch_dow	3
duration	21
Predicted Label ID	1
Predicted Label	failed
Class 1 Probability	0.087753
Class 2 Probability	0.502152
Class 3 Probability	0.000754
Class 4 Probability	0.406379
Class 5 Probability	0.001997
Class 6 Probability	0.000964

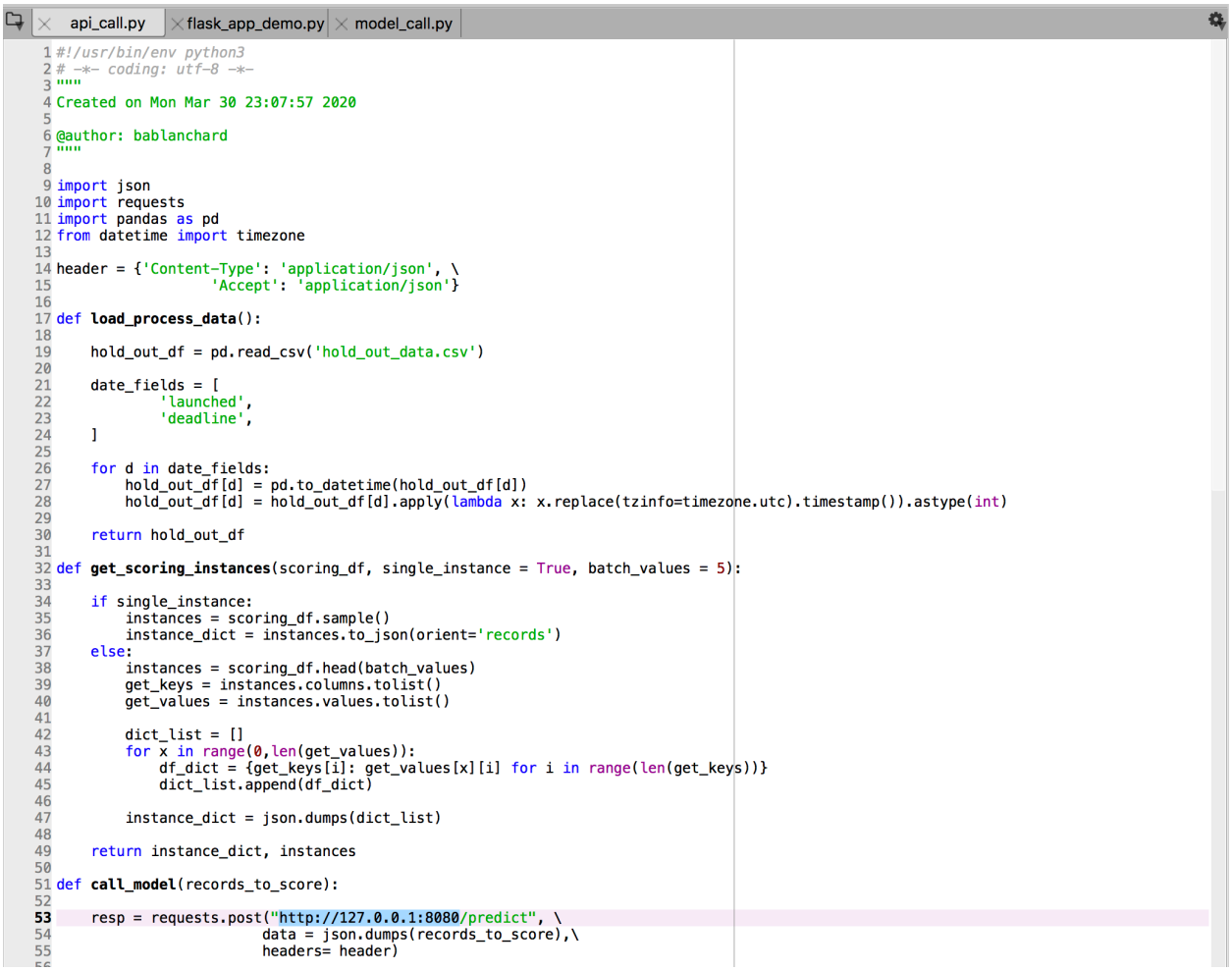
You have successfully deployed your application locally.

API Call Python-script

Supplemental to the guide documented above

Open the api_call.py script in your IDE (Spyder shown below). You will need to make the following changes to the script once downloaded.

1) Possibly change the IP-address shown to the one displayed in your terminal from the instructions in the prior guide (line 53 below)
2) The other reference to this address can be found on Line 91



Run the “main()” function at the bottom of the script to score data from this script using the model hosted by the flask_app_demo.py script. For this to work, you’ll need to ensure that you flask app is still being hosted in your terminal.