Build a Dashboard Application with Plotly Dash In this lab, you will be building a Plotly Dash application for users to perform interactive visual analytics on SpaceX launch data in real-time. This dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart. You will be guided to build this dashboard

SpaceX Launch Records Dashboard

Payload Mass (kg)

Run

Terminal

Help

^ ①`

ひ Ж K

New Terminal

Run Build Task...

Run Test Task...

Run Last Task

Terminate Task...

Configure Tasks...

Run Selected Text

Help

Attach Task...

Show Running Tasks...

Restart Running Task...

Run Task...

Go

• TASK 2: Add a callback function to render success-pie-chart based on selected site dropdown

Payload range (Kg): Correlation between Payload and Success for all Sites After visual analysis using the dashboard, you should be able to obtain some insights to answer the following five questions:

1. Which site has the largest successful launches? 2. Which site has the highest launch success rate? 3. Which payload range(s) has the highest launch success rate? 4. Which payload range(s) has the lowest launch success rate? 5. Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest launch success rate? Estimated time needed: 90 minutes Important Notice about this lab environment Please be aware that sessions for this lab environment are not persisted. When you launch the Cloud IDE, you are presented with a 'dedicated computer on the cloud' exclusively for you. This is available

application via the following tasks:

All Sites

• TASK 1: Add a Launch Site Drop-down Input Component

• TASK 4: Add a callback function to render the success-payload-scatter-chart scatter plot

The github url and the screenshots are later required in the presentation slides.

Your completed dashboard application should look like the following screenshot:

Note:Please take screenshots of the Dashboard and save them. Further upload your notebook to github.

• TASK 3: Add a Range Slider to Select Payload

Total Success Launches By Site

to you as long as you are actively working on the labs. Once you close your session or it is timed out due to inactivity, you are logged off, and this dedicated computer on the cloud is deleted along with any files you may have created, downloaded or installed. The next time you launch this lab, a new environment is created for you. If you finish only part of the lab and return later, you may have to start from the beginning. So, it is a good idea to plan your time accordingly and finish your labs in a single session.

Setup development environment Install required Python packages • Open a new terminal, by clicking on the menu bar and selecting **Terminal**->**New Terminal**, as in the image below. Selection View dash_interactivity.py ×

Now, you have script and terminal ready to start the lab.

dash_interactivity.py ×

Selection

View

theia@theiadocker-saishruthitn: /home/project ×

theia@theiadocker-saishruthitn:/home/project\$ □

Downloading https://files.pythonhosted.org/packages/c3/e2/00cacecafbab071c787019f00ad84ca3185952f6bb9bca9550ed83870d4d/pandas-1.1.5-cp36-cp36m-manylinux1_x8

Downloading https://files.pythonhosted.org/packages/cc/42/e1692b2d34e4135569db680efe3438e809a6b3f0ae607ad41aeff7741672/dash-2.6.1-py3-none-any.whl (9.9MB)

Downloading https://files.pythonhosted.org/packages/d5/50/54451e88e3da4616286029a3a17fc377de817f66a0f50e1faaee90161724/pytz-2022.2.1-py2.py3-none-any.whl (5

Downloading https://files.pythonhosted.org/packages/36/7a/87837f39d0296e723bb9b62bbb257d0355c7f6128853c78955f57342a56d/python_dateutil-2.8.2-py2.py3-none-an

Downloading https://files.pythonhosted.org/packages/45/b2/6c7545bb7a38754d63048c7696804a0d947328125d81bf12beaa692c3ae3/numpy-1.19.5-cp36-cp36m-manylinux1_x8

Downloading https://files.pythonhosted.org/packages/da/ce/43f77dc8e7bbad02a9f88d07bf794eaf68359df756a28bb9f2f78e255bb1/dash_table-5.0.0-py3-none-any.whl

wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex launch dash.csv"

wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_3/spacex_dash_app.py"

• In the left Navigation Pane click on Others and click Launch Application option under it. Enter the application port number as 8050. Click Your Application.

 \Box

Downloading https://files.pythonhosted.org/packages/83/96/55b82d9f13763be9d672622e1b8106c85acb83edd7cc2fa5bc67cd9877e9/contextvars-2.4.tar.gz

Go

Run

Terminal

Edit

• Install python packages required to run the application.

theia@theiadocker-malikas:/home/projects python3 -m pip install pandas dash

9.5MB 163kB/s

9.9MB 159kB/s

| 501kB 3.2MB/s

256kB 5.8MB/s

| 13.4MB 111kB/s

Download a skeleton dashboard application and dataset

Run the following wget command line in the terminal to download dataset as spacex_launch_dash.csv

Copy and paste the below command to the terminal.

Cache entry deserialization failed, entry ignored

Cache entry deserialization failed, entry ignored Cache entry deserialization failed, entry ignored

Collecting contextvars==2.4; python_version < "3.7" (from dash)

First, let's get the SpaceX Launch dataset for this lab:

Download a skeleton Dash app to be completed in this lab:

Observe the port number (8050) shown in the terminal.

* Running or http://127.0.0.1:8050/ (Press CTRL+C to quit)

SKILLS NETWO...

Test.py

Launch Application

Python - Get Started

Launch Your Application

You should see a nearly blank web page indicating a successfully running dash app.

Next, let's fill the skeleton app with required input/output components and callback functions.

As such, we will need a dropdown menu to let us select different launch sites.

id attribute with value site-dropdown

the default All option. e.g.,

options=[

value='ALL',

searchable=True

],

),

Here is an example of dcc. Dropdown:

dcc.Dropdown(id='id',

TASK 1: Add a Launch Site Drop-down Input Component

• Find and complete a commented dcc.Dropdown(id='site-dropdown',...) input with following attributes:

value attribute with default dropdown value to be ALL meaning all sites are selected

searchable attribute to be True so we can enter keywords to search launch sites

{'label': 'All Sites', 'value': 'ALL'},

If you need more help about Dropdown(), refer to the Plotly Dash Reference section towards the end of this lab.

{'label': 'site1', 'value': 'site1'},

placeholder="place holder here",

Your completed dropdown menu should look like the following screenshot:

options=[{'label': 'All Sites', 'value': 'ALL'},{'label': 'site1', 'value': 'site1'}, ...]

o placeholder attribute to show a text description about this input area, such as Select a Launch Site here

If you need to refresh your memory about Plotly Dash components and callback functions, you may refer to the lab you have learned before:

We have four different launch sites and we would like to first see which one has the largest success count. Then, we would like to select one specific site and check its detailed success rate (class=0 vs.

TASK 2: Add a callback function to render success-pie-chart based on selected

• If ALL sites are selected, we will use all rows in the dataframe spacex_df to render and return a pie chart graph to show the total success launches (i.e., the total count of class column)

41.7%

73.1%

Next, we want to find if variable payload is correlated to mission outcome. From a dashboard point of view, we want to be able to easily select different payload range and see if we can identify some visual

TASK 4: Add a callback function to render the success-payload-scatter-chart

Next, we want to plot a scatter plot with the x axis to be the payload and the y axis to be the launch outcome (i.e., class column). As such, we can visually observe how payload may be correlated with

• Input to be [Input(component_id='site-dropdown', component_property='value'), Input(component_id="payload-slider", component_property="value")] Note that we have two input

Payload Mass (kg)

© IBM Corporation 2021. All rights reserved.

Dash callback function is a type of Python function which will be automatically called by Dash whenever receiving an input component updates, such as a click or dropdown selecting event.

to include the only data for the selected site. Then, render and return a pie chart graph to show the success (class=1) count and failed (class=0) count for the selected site.

The general idea of this callback function is to get the selected launch site from site-dropdown and render a pie chart visualizing launch success counts.

If you need to refresh your memory about Plotly Dash callback functions, you may refer to the lab you have learned before:

• Input is set to be the site-dropdown dropdown, i.e., Input(component_id='site-dropdown', component_property='value')

• Output to be the graph with id success-pie-chart, i.e., Output(component_id='success-pie-chart', component_property='figure')

If you need more reference about dash callbacks and plotly pie charts, refer to the Plotly Dash Reference section towards the end of this lab.

Let's add a callback function in spacex_dash_app.py including the following application logic:

• A If-Else statement to check if ALL sites were selected or just a specific launch site was selected

@app.callback(Output(component_id='success-pie-chart', component_property='figure'),

Input(component_id='site-dropdown', component_property='value'))

• If a specific launch site is selected, you need to filter the dataframe spacex_df first in order

options attribute is a list of dict-like option objects (with label and value attributes). You can set the label and value all to be the launch site names in the spacex_df and you need to include

SpaceX Launch Records Dashboard

(i) - (ii)

(i) To open any application in the browser, please select or enter the port number below.

• Test the skeleton app by running the following command in the terminal:

theia@theiadocker-saishruthitn:/home/project\$ python dash_basics.py

Collecting python-dateutil>=2.7.3 (from pandas)

python3 -m pip **install** pandas dash

theia@theiadocker-malikas: /home/project ×

Collecting pytz>=2017.2 (from pandas)

Collecting numpy>=1.15.4 (from pandas)

Collecting dash-table==5.0.0 (from dash)

python3 spacex_dash_app.py

* Environment: production

* Debug mode: off

> CLOUD √ OTHER

Launch Application

Open Link inside IDE

Open Cloud IDE wit.

Suggest a feature or.

About Cloud IDE

Plotly Dash Lab

class=1).

All Sites

All Sites

VAFB SLC-4E

KSC LC-39A

CCAFS SLC-40

Plotly Dash Lab

site dropdown

Here is an example of a callback function:

def get_pie_chart(entered_site):

filtered_df = spacex_df

title='title')

return fig

All Sites

CCAFS LC-40

patterns.

• id to be payload-slider

Here is an example of RangeSlider:

dcc.RangeSlider(id='id',

scatter plot

mission outcomes for selected site(s).

Payload range (Kg):

Payload range (Kg):

0.8

launch success rate?

Author

Joseph Santarcangelo

Changelog

Yan Luo

Date

20-09-2022

29-08-2022

03-09-2021

06-01-2021

if entered_site == 'ALL':

Pie chart for all sites are selected

Total Success Launches By Site

Pie chart for is selected

Total Success Launches for site CCAFS LC-40

TASK 3: Add a Range Slider to Select Payload

• min indicating the slider starting point, we set its value to be 0 (Kg)

min=0, max=10000, step=1000,

value=[min_value, max_value])

You completed payload range slider should be similar the following screenshot:

100: '100'},

Now, let's add a call function including the following application logic:

You rendered scatter point should look like the following screenshot:

Correlation between Payload and Success for all Sites

Finding Insights Visually

Plotly Dash Reference

Dropdown (input) component

Refer here for more details about dcc.Dropdown()

Range slider (input) component

Refer here for more details about dcc.RangeSlider()

Pie chart (output) component

Refer here for more details about plotly pie charts

Scatter chart (output) component

Changed by

Lakshmi Holla

Lakshmi Holla

Lakshmi Holla

Yan

Change Description

Updated screenshot.

Updated screenshot.

Initial version created

Added a note.

Refer here for more details about plotly scatter charts

Other contributor(s)

Version

1.3

1.2

1.1

1.0

1. Which site has the largest successful launches?

2. Which site has the highest launch success rate?

3. Which payload range(s) has the highest launch success rate?

4. Which payload range(s) has the lowest launch success rate?

5. Which F9 Booster version (v1.0, v1.1, FT, B4, B5, etc.) has the highest

marks={0: '0',

• max indicating the slider ending point to, we set its value to be 10000 (Kg)

• step indicating the slider interval on the slider, we set its value to be 1000 (Kg)

Find and complete a commented dcc.RangeSlider(id='payload-slider',...) input with the following attribute:

• value indicating the current selected range, we could set it to be min_payload and max_payload

If you need more reference about range slider, refer to the Plotly Dash Reference towards the end of this lab.

components, one to receive selected launch site and another to receive selected payload range

• A If-Else statement to check if ALL sites were selected or just a specific launch site was selected

• Output to be Output(component_id='success-payload-scatter-chart', component_property='figure')

In addition, the point color needs to be set to the booster version i.e., color="Booster Version Category"

• If a specific launch site is selected, you need to filter the spacex_df first, and render a scatter chart to show

In addition, we want to color-label the Booster version on each scatter point so that we may observe mission outcomes with different boosters.

• If ALL sites are selected, render a scatter plot to display all values for variable Payload Mass (kg) and variable class.

values Payload Mass (kg) and class for the selected site, and color-label the point using Boosster Version Category likewise.

If you need more reference about dash callbacks and plotly scatter plots, refer to the Plotly Dash Reference towards the end of this lab.

Now with the dashboard completed, you should be able to use it to analyze SpaceX launch data, and answer the following questions:

names='pie chart names',

Function decorator to specify function input and output

return the outcomes piechart for a selected site

fig = px.pie(data, values='class',

The rendered pie chart should look like the following screenshots:

Python ×

Dash is running on http://127.0.0.1:8050/

Use a production WSGI server stead.

File Edit Selection View Go Run Terminal Help

8050

Your Application

* Serving Flask app "dash_basics" (lazy loading)

Problems 5

hl (9.5MB) 100%

Collecting dash

100%

hl (13.4MB) 100%

File