

Hands-on Lab: Build an In...

Skills Network Labs

launching (1081x415)

scatter-point.PNG (1911x415)

labs.cognitiveclass.ai/v2/tools/cloud-ide

Correo: Gabriel Sa...

Páginas de inicio ...

Classroom

Google Académico

Tus proyectos - O...

La Tríada en Cour...

Hello World - OIH...

Gmail

ChatGPT

libros - Google Dri...

Facebook

Dashboard 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 application via the following tasks:

- TASK 1: Add a Launch Site Drop-down Input Component
- TASK 2: Add a callback function to render `success_pie_chart` based on selected site dropdown
- TASK 3: Add a Range Slider to Select Payload
- TASK 4: Add a callback function to render the `success_payload_scatter_chart` scatter plot

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

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

Your completed dashboard application should look like the following screenshots:



After visual analysis using the dashboard, you should be able to obtain some insights to answer the following five questions:

space\_dash\_app.py

Your Application

space\_launch\_dash.csv

settings.json 1

http://gabrielmuri-8050.theia.net-O-lab-prod-misc-tools-us-east-2.proxy.cognitiveclass.ai/

## SpaceX Launch Records Dashboard

All Sites

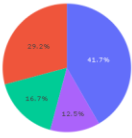
All Sites

CCAFS LC-40

CCAFS SLC-40

KSC LC-39A

VAFB SLC-4E



Payload range (Kg):

0

2500

5000

7500

10000

Correlation between Payload and Launch Success

Problems

theia@theia-gabrielmuri:/home/project

The `dash_core_components` package is deprecated. Please replace `import dash_core_components as dcc` with `from dash import dcc`

import dash\_core\_components as dcc

File Edit Selection View Go Run Terminal Help

space\_dash\_app.py

Your Application

space\_launch\_dash.csv

settings.json 1

http://gabrielmuri-8050.theia.net-O-lab-prod-misc-tools-us-east-2.proxy.cognitiveclass.ai/

## SpaceX Launch Records Dashboard

All Sites

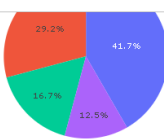
All Sites

CCAFS LC-40

CCAFS SLC-40

KSC LC-39A

VAFB SLC-4E



Payload range (Kg):

0

2500

5000

7500

10000

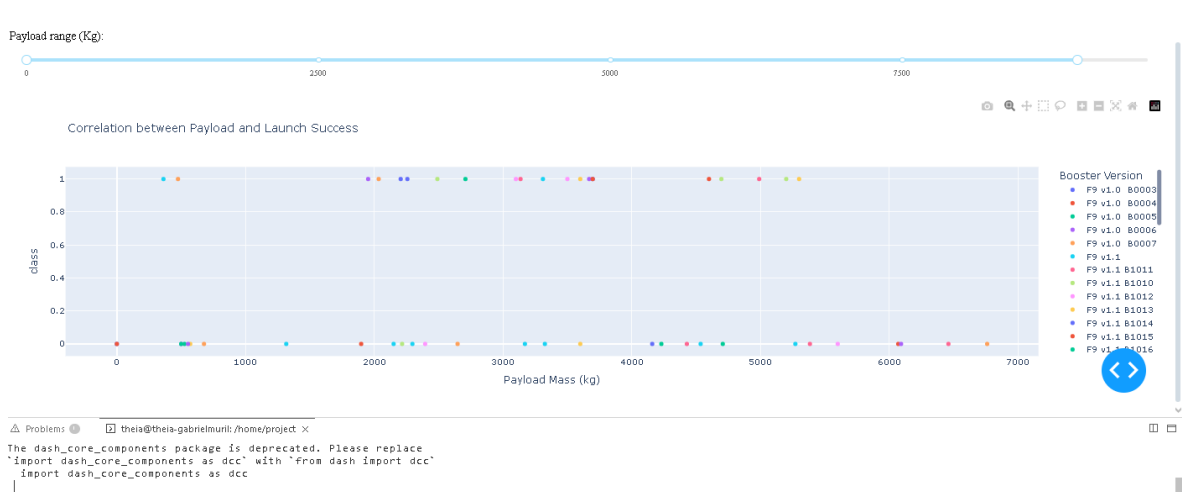
Correlation between Payload and Launch Success

Problems

theia@theia-gabrielmuri:/home/project

The `dash_core_components` package is deprecated. Please replace `import dash_core_components as dcc` with `from dash import dcc`

import dash\_core\_components as dcc



# Import required libraries

import pandas as pd

import dash

import dash\_html\_components as html

import dash\_core\_components as dcc

from dash.dependencies import Input, Output

import plotly.express as px

# Read the SpaceX launch data into a pandas DataFrame

spacex\_df = pd.read\_csv("spacex\_launch\_dash.csv")

# Calculate the minimum and maximum payload mass for the slider

min\_payload = spacex\_df['Payload Mass (kg)'].min()

max\_payload = spacex\_df['Payload Mass (kg)'].max()

# Create a Dash application

app = dash.Dash(\_\_name\_\_)

# Define the layout of the app

```
app.layout = html.Div(children=[
```

```
    # App title
```

```
    html.H1('SpaceX Launch Records Dashboard',
```

```
        style={'textAlign': 'center', 'color': '#503D36', 'font-size': 40}),
```

# TASK 1: Add a dropdown list to enable Launch Site selection.

# Default option is "All Sites"

```
    dcc.Dropdown(
```

```
        id='site-dropdown',
```

```
        options=[
```

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

```
            {'label': 'CCAFS LC-40', 'value': 'CCAFS LC-40'},
```

```
            {'label': 'CCAFS SLC-40', 'value': 'CCAFS SLC-40'},
```

```
            {'label': 'KSC LC-39A', 'value': 'KSC LC-39A'},
```

```
            {'label': 'VAFB SLC-4E', 'value': 'VAFB SLC-4E'}]
```

```
    ),
```

```
    value='ALL',
```

```
    placeholder="Select a Launch Site",
```

```
    searchable=True
```

```
),
```

```
    html.Br(),
```

# TASK 2: Add a pie chart to show the total successful launches count for all sites.

# If a specific launch site is selected, show the success vs. failure counts for that site.

```
html.Div(dcc.Graph(id='success-pie-chart')),
```

```
html.Br(),
```

# Display label for the payload slider

```
html.P("Payload range (Kg):"),
```

# TASK 3: Add a range slider to select payload mass range

```
dcc.RangeSlider(
```

```
    id='payload-slider',
```

```
    min=min_payload,
```

```
    max=max_payload,
```

```
    step=1000,
```

```
    marks={i: f'{i}' for i in range(int(min_payload), int(max_payload)+1, 2500)},
```

```
    value=[min_payload, max_payload]
```

```
),
```

```
html.Br(),
```

# TASK 4: Add a scatter chart to show the correlation between payload and launch success

```
html.Div(dcc.Graph(id='success-payload-scatter-chart'))
```

```
])
```

# TASK 2: Callback function for updating the pie chart based on the selected launch site

```
@app.callback(
```

```

Output(component_id='success-pie-chart', component_property='figure'),
Input(component_id='site-dropdown', component_property='value')
)

def update_pie_chart(selected_site):
    """
    Update the pie chart based on the selected launch site.

    If 'ALL' is selected, display the total success launches by site.

    Otherwise, filter the DataFrame for the selected site and display success vs. failure
    counts.
    """

    if selected_site == 'ALL':
        # For all sites, group by 'Launch Site' using the 'class' column as measure
        fig = px.pie(spacex_df,
                     names='Launch Site',
                     values='class',
                     title='Total Success Launches by Site')
    else:
        # Filter the DataFrame for the selected site
        filtered_df = spacex_df[spacex_df['Launch Site'] == selected_site]

        # Count outcomes (success = 1, failure = 0)
        outcome_counts = filtered_df['class'].value_counts().reset_index()

        outcome_counts.columns = ['Outcome', 'Count']

        fig = px.pie(outcome_counts,
                     names='Outcome',
                     values='Count',
                     title=f"Launch Outcomes for {selected_site}")

```

```

return fig

# TASK 4: Callback function for updating the scatter plot based on selected launch
site and payload range

@app.callback(
    Output(component_id='success-payload-scatter-chart',
component_property='figure'),
    [Input(component_id='site-dropdown', component_property='value'),
    Input(component_id='payload-slider', component_property='value')]
)
def update_scatter_plot(selected_site, payload_range):
    """
    Update the scatter plot based on the selected launch site and payload range.
    Filter the DataFrame based on payload mass and, if a specific site is selected,
    filter by that site as well. The scatter plot shows the correlation between payload
    mass
    and launch success, with the color indicating the Booster Version.
    """
    low, high = payload_range
    filtered_df = spacex_df[(spacex_df['Payload Mass (kg)'] >= low) &
                            (spacex_df['Payload Mass (kg)'] <= high)]

    if selected_site != 'ALL':
        filtered_df = filtered_df[filtered_df['Launch Site'] == selected_site]

    fig = px.scatter(filtered_df,
                     x='Payload Mass (kg)',

```

```
    y='class',  
    color='Booster Version',  
    title='Correlation between Payload and Launch Success')  
return fig
```

```
# Run the Dash app
```

```
if __name__ == '__main__':  
    app.run_server(debug=True)
```