



Laboratorio práctico: Crea un panel interactivo con Plotly Dash

En este laboratorio, construirás una aplicación de Plotly Dash para que los usuarios realicen análisis visuales interactivos sobre los datos de lanzamiento de SpaceX en tiempo real.

Esta aplicación de panel contiene componentes de entrada como una lista desplegable y un control deslizante de rango para interactuar con un gráfico de pastel y un gráfico de dispersión. Se te guiará para construir esta aplicación de panel a través de las siguientes tareas:

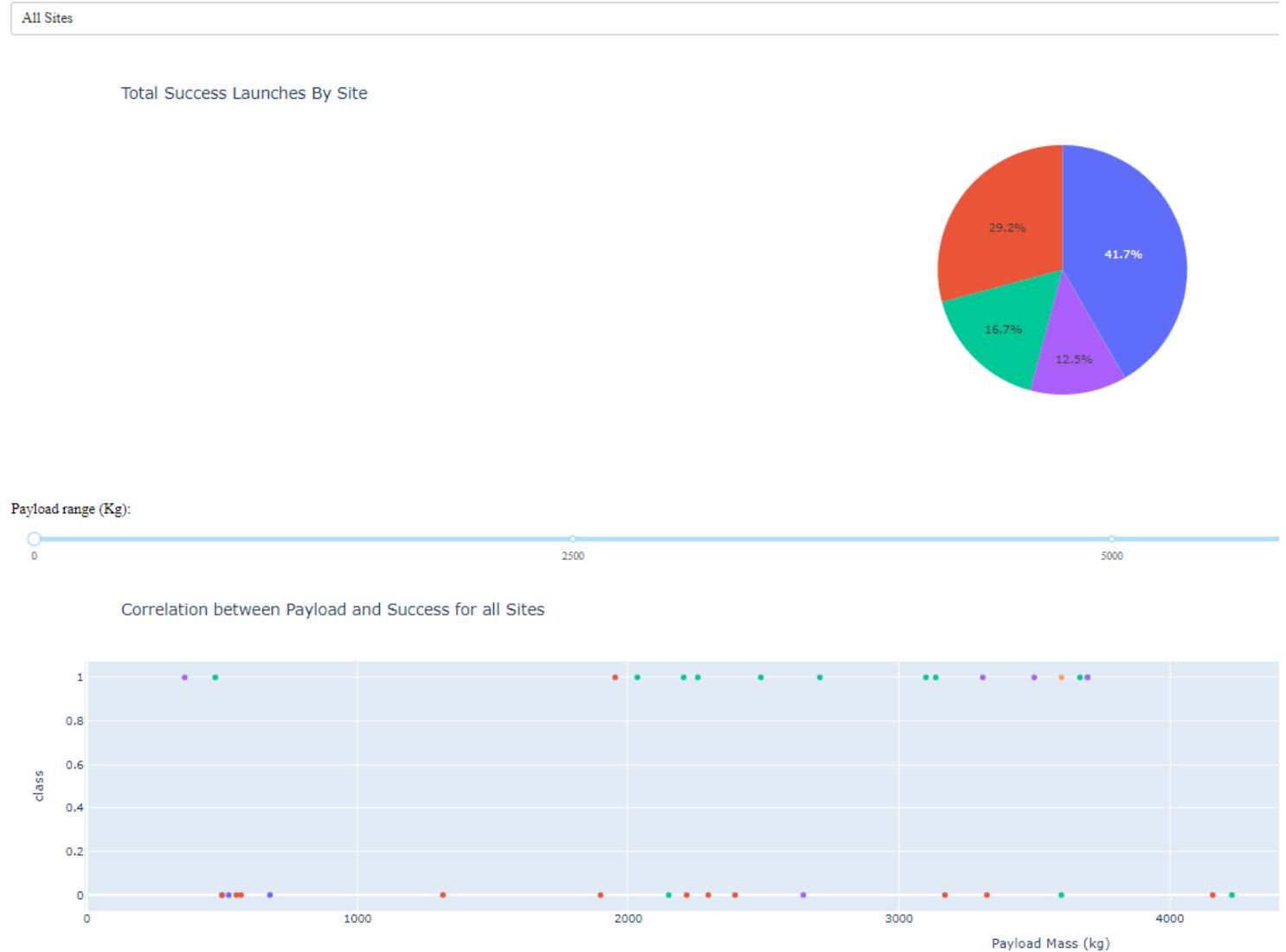
- TAREA 1: Agregar un componente de entrada desplegable para el sitio de lanzamiento
- TAREA 2: Agregar una función de callback para renderizar success-pie-chart basado en el sitio seleccionado en la lista desplegable
- TAREA 3: Agregar un control deslizante de rango para seleccionar la carga útil
- TAREA 4: Agregar una función de callback para renderizar el gráfico de dispersión success-payload-scatter-chart

Nota: Por favor, toma capturas de pantalla del panel y guárdalas. Luego, sube tu cuaderno a github.

La URL de github y las capturas de pantalla se requerirán más adelante en las diapositivas de presentación.

Tu aplicación de panel completada debería verse como la siguiente captura de pantalla:

SpaceX Launch Records Dashboard



Después del análisis visual utilizando el panel, deberías ser capaz de obtener algunos insights para responder las siguientes cinco preguntas:

1. ¿Qué sitio tiene el mayor número de lanzamientos exitosos?
2. ¿Qué sitio tiene la tasa de éxito de lanzamiento más alta?

3. ¿Qué rango(s) de carga útil tiene la tasa de éxito de lanzamiento más alta?
4. ¿Qué rango(s) de carga útil tiene la tasa de éxito de lanzamiento más baja?
5. ¿Qué versión de F9 Booster (v1.0, v1.1, FT, B4, B5, etc.) tiene la tasa de éxito de lanzamiento más alta?

Tiempo estimado necesario: 90 minutos

Aviso importante sobre este entorno de laboratorio

Ten en cuenta que las sesiones para este entorno de laboratorio no se persisten. Cuando inicias el IDE en la nube, se te presenta una ‘computadora dedicada en la nube’ exclusivamente para ti. Esto está disponible mientras trabajas activamente en los laboratorios. Una vez que cierras tu sesión o se agota el tiempo por inactividad, se cierra tu sesión, y esta computadora dedicada en la nube se elimina junto con cualquier archivo que hayas creado, descargado o instalado.

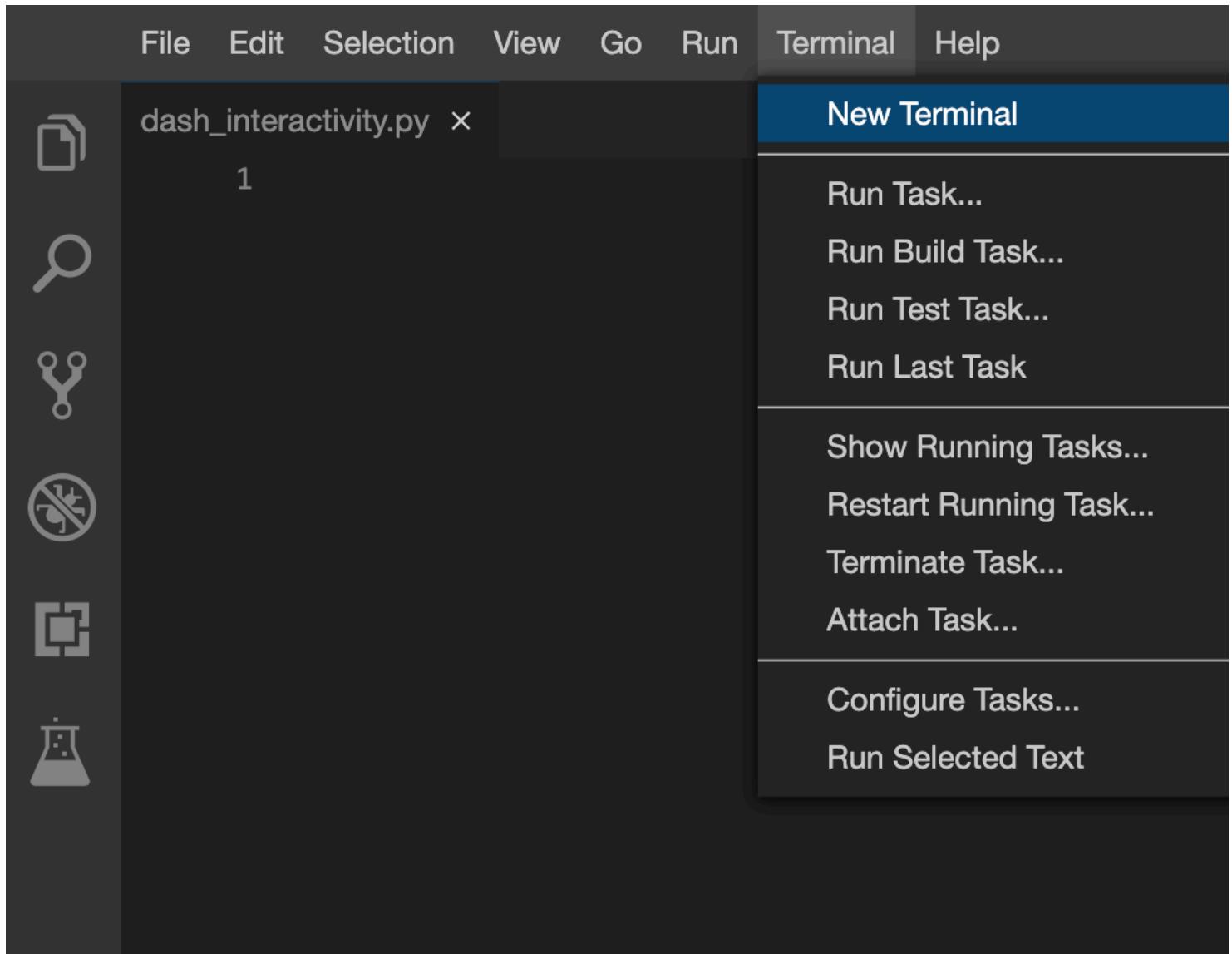
La próxima vez que inicies este laboratorio, se crea un nuevo entorno para ti.

Si solo terminas parte del laboratorio y regresas más tarde, es posible que debas comenzar desde el principio. Por lo tanto, es una buena idea planificar tu tiempo de acuerdo y terminar tus laboratorios en una sola sesión.

Configuración del entorno de desarrollo

Instalar los paquetes de Python requeridos

- Abre una nueva terminal, haciendo clic en la barra de menú y seleccionando Terminal->Nueva Terminal, como en la imagen a continuación.



- Ahora, tienes el script y la terminal listos para comenzar el laboratorio.

File Edit Selection View Go Run Terminal Help

dash_interactivity.py ×

1



theia@theiadocker-saishruthitn: /home/project ×

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

- Instala los paquetes de Python necesarios para ejecutar la aplicación.

Copia y pega el siguiente comando en la terminal.

```
python3.11 -m pip install pandas dash
```

```
theia@theiadocker-anitaj: /home/project ×
```

```
theia@theiadocker-anitaj: /home/project$  
Defaulting to user installation because  
Collecting pandas  
  Downloading pandas-1.5.3-cp38-cp38-ma
```

```
Collecting dash
```

```
  Downloading dash-2.8.1-py3-none-any.w
```

```
Requirement already satisfied: pytz>=20  
Requirement already satisfied: python-d  
Collecting numpy>=1.20.3  
  Downloading numpy-1.24.2-cp38-cp38-ma
```

```
Collecting dash-html-components==2.0.0
```

```
  Downloading dash_html_components-2.0.
```

```
Collecting plotly>=5.0.0
```

```
  Downloading plotly-5.13.1-py2.py3-non
```

```
Collecting dash-core-components==2.0.0
```

```
  Downloading dash_core_components-2.0.
```

```
Collecting dash-table==5.0.0
```

```
  Downloading dash_table-5.0.0-py3-none
```

```
Requirement already satisfied: Flask>=1
```

```
Requirement already satisfied: importlib  
(4.12.0)  
Requirement already satisfied: Werkzeug
```

Descarga una aplicación de panel esqueleto y conjunto de datos

Primero, obtengamos el conjunto de datos de lanzamientos de SpaceX para este laboratorio:

- Ejecuta el siguiente comando wget en la terminal para descargar el conjunto de datos como `spacex_launch_dash.csv`

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

- Descarga una aplicación Dash esqueleto que se completará en este laboratorio:

```
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6E3Px_ww/spacex-dash-app.py"
```

- Prueba la aplicación esqueleto ejecutando el siguiente comando en la terminal:

```
python3.11 spacex-dash-app.py
```

- Observe el número de puerto (8050) que se muestra en la terminal.

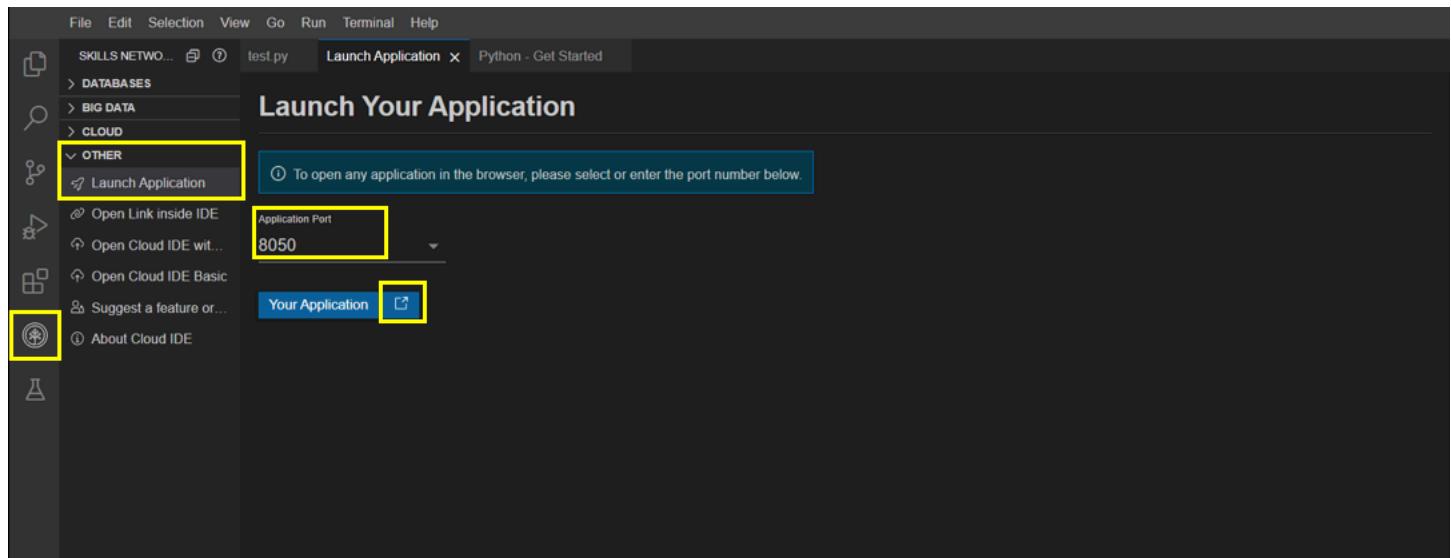
theia@theiadocker-anitaj: /home/project ×

```
theia@theiadocker-anitaj:/home/project$ spacex_dash_app.py:4: UserWarning: The dash_html_components package is depr `import dash_html_components as html` with import dash_html_components as html spacex_dash_app.py:5: UserWarning: The dash_core_components package is depr `import dash_core_components as dcc` with import dash_core_components as dcc Dash is running on http://127.0.0.1:8050
```

```
* Serving Flask app 'spacex_dash_app'  
* Debug mode: off  
WARNING: This is a development server. It is highly  
* Running on http://127.0.0.1:8050  
Press CTRL+C to quit
```



- En el Panel de Navegación izquierdo, haga clic en Others y seleccione la opción Launch Application debajo de él. Ingrese el número de puerto de la aplicación como 8050.
Haga clic en Your Application.



- Debería ver una página web casi en blanco que indica que una aplicación dash se está ejecutando correctamente.
- A continuación, vamos a llenar la aplicación esqueleto con los componentes de entrada/salida requeridos y las funciones de callback.

Si necesita refrescar su memoria sobre los componentes de Plotly Dash y las funciones de callback, puede referirse al laboratorio que aprendió anteriormente:

[Plotly Dash Lab](#)

TAREA 1: Agregar un Componente de Entrada de Desplegable para el Sitio de Lanzamiento

Tenemos cuatro sitios de lanzamiento diferentes y nos gustaría primero ver cuál tiene la mayor cantidad de éxitos. Luego, nos gustaría seleccionar un sitio específico y verificar su tasa de éxito detallada (clase=0 vs. clase=1).

Por lo tanto, necesitaremos un menú desplegable que nos permita seleccionar diferentes sitios de lanzamiento.

- Encuentre y complete un `dcc.Dropdown(id='site-dropdown', ...)` comentado con los siguientes atributos:
 - o atributo `id` con el valor `site-dropdown`
 - o atributo `options` es una lista de objetos de opción similares a diccionarios (con atributos `label` y `value`). Puede establecer el `label` y `value` todos como los nombres de los sitios de lanzamiento en el `spacex_df` y debe incluir la opción predeterminada `All`. por ejemplo,

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

- o atributo `value` con el valor predeterminado del desplegable como `ALL`, lo que significa que se seleccionan todos los sitios
- o atributo `placeholder` para mostrar una descripción de texto sobre esta área de entrada, como `Select a Launch Site here`
- o atributo `searchable` para ser `True` para que podamos ingresar palabras clave para buscar sitios de lanzamiento

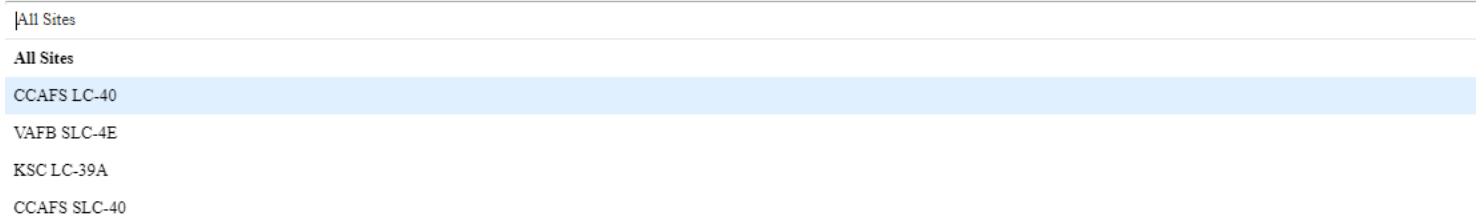
Aquí hay un ejemplo de `dcc.Dropdown`:

```
dcc.Dropdown(id='id',
              options=[
                  {'label': 'All Sites', 'value': 'ALL'},
                  {'label': 'site1', 'value': 'site1'},
              ],
              value='ALL',
              placeholder="place holder here",
              searchable=True
),
```

Si necesitas más ayuda sobre `Dropdown()`, consulta la sección de Referencia de Plotly Dash hacia el final de este laboratorio.

Tu menú desplegable completado debería verse como la siguiente captura de pantalla:

SpaceX Launch Records Dash



TAREA 2: Agregar una función de callback para renderizar success-pie-chart basado en el sitio seleccionado en el menú desplegable

La idea general de esta función de callback es obtener el sitio de lanzamiento seleccionado del site-dropdown y renderizar un gráfico de pastel que visualice los conteos de éxito de lanzamientos.

La función de callback de Dash es un tipo de función de Python que será llamada automáticamente por Dash cada vez que se actualice un componente de entrada, como un clic o un evento de selección en el menú desplegable.

Si necesitas refrescar tu memoria sobre las funciones de callback de Plotly Dash, puedes consultar el laboratorio que aprendiste anteriormente:

[Laboratorio de Plotly Dash](#)

Vamos a agregar una función de callback en `spacex_dash_app.py` que incluya la siguiente lógica de aplicación:

- La entrada se establece en el menú desplegable site-dropdown, es decir, `Input(component_id='site-dropdown', component_property='value')`
- La salida será el gráfico con id success-pie-chart, es decir, `Output(component_id='success-pie-chart', component_property='figure')`
- Una declaración If-Else para verificar si se seleccionaron TODOS los sitios o solo un sitio de lanzamiento específico
 - Si se seleccionan TODOS los sitios, utilizaremos todas las filas en el datafram `spacex_df` para renderizar y devolver un gráfico de pastel que muestre el total de lanzamientos exitosos (es decir, el conteo total de la columna `class`)
 - Si se selecciona un sitio de lanzamiento específico, primero necesitas filtrar el datafram `spacex_df` para incluir solo los datos del sitio seleccionado. Luego, renderiza y devuelve un gráfico de pastel que muestre el conteo de éxitos (`class=1`) y fracasos (`class=0`) para el sitio seleccionado.

Aquí hay un ejemplo de una función de callback:

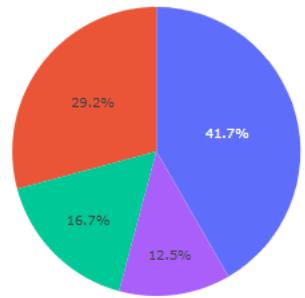
```
# Function decorator to specify function input and output
@app.callback(Output(component_id='success-pie-chart', component_property='figure'),
              Input(component_id='site-dropdown', component_property='value'))
def get_pie_chart(entered_site):
    filtered_df = spacex_df
    if entered_site == 'ALL':
        fig = px.pie(data, values='class',
                      names='pie chart names',
                      title='title')
        return fig
    else:
        # return the outcomes piechart for a selected site
```

El gráfico circular renderizado debería verse como las siguientes capturas de pantalla:

- Gráfico circular para todos los sitios seleccionados

All Sites

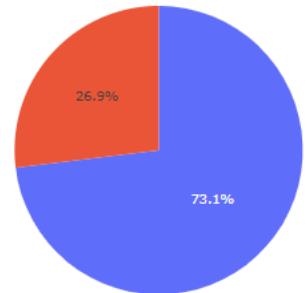
Total Success Launches By Site



- Gráfico circular para un sitio seleccionado

CCAFS LC-40

Total Success Launches for site CCAFS LC-40



Si necesitas más referencias sobre callbacks de dash y gráficos circulares de plotly, consulta la sección [Plotly Dash Reference](#) hacia el final de este laboratorio.

TAREA 3: Agregar un Control deslizante de rango para seleccionar la carga útil

A continuación, queremos averiguar si la variable carga útil está correlacionada con el resultado de la misión. Desde el punto de vista del panel de control, queremos poder seleccionar fácilmente diferentes rangos de carga útil y ver si podemos identificar algunos patrones visuales.

Encuentra y completa un `dcc.RangeSlider(id='payload-slider',...)` comentado con los siguientes atributos:

- `id` que sea `payload-slider`
- `min` indicando el punto de inicio del control deslizante, establecemos su valor en 0 (Kg)
- `max` indicando el punto final del control deslizante, establecemos su valor en 10000 (Kg)
- `step` indicando el intervalo del control deslizante, establecemos su valor en 1000 (Kg)
- `value` indicando el rango seleccionado actualmente, podríamos establecerlo en `min_payload` y `max_payload`

Aquí hay un ejemplo de `RangeSlider`:

```
dcc.RangeSlider(id='id',
    min=0, max=10000, step=1000,
    marks={0: '0',
           100: '100'},
    value=[min_value, max_value])
```

Has completado el control deslizante de rango de carga útil que debería ser similar a la siguiente captura de pantalla:

`Payload range (Kg):`



Si necesitas más referencias sobre el control deslizante de rango, consulta la [Referencia de Plotly Dash](#) hacia el final de este laboratorio.

TAREA 4: Agregar una función de callback para renderizar el gráfico de dispersión success-payload-scatter-chart

A continuación, queremos trazar un gráfico de dispersión con el eje x como la carga útil y el eje y como el resultado del lanzamiento (es decir, la columna `class`). De esta manera, podemos observar visualmente cómo la carga útil puede estar correlacionada con los resultados de la misión para el(s) sitio(s) seleccionado(s).

Además, queremos etiquetar con colores la versión del cohete en cada punto de dispersión para que podamos observar los resultados de las misiones con diferentes cohetes.

Ahora, agreguemos una función de llamada que incluya la siguiente lógica de aplicación:

- Entrada a ser `[Input(component_id='site-dropdown', component_property='value'), Input(component_id="payload-slider", component_property="value")]`
Ten en cuenta que tenemos dos componentes de entrada, uno para recibir el sitio de lanzamiento seleccionado y otro para recibir el rango de carga útil seleccionado.
- Salida a ser `Output(component_id='success-payload-scatter-chart', component_property='figure')`
- Una declaración `If-Else` para verificar si SELECCIONARON TODOS los sitios o solo un sitio de lanzamiento específico.
 - Si SELECCIONAN TODOS los sitios, renderiza un gráfico de dispersión para mostrar todos los valores de la variable `Payload Mass (kg)` y la variable `class`. Además, el color del punto debe establecerse en la versión del cohete, es decir, `color="Booster Version Category"`.
 - Si se selecciona un sitio de lanzamiento específico, primero debes filtrar el `spaceX_df`, y renderizar un gráfico de dispersión para mostrar los valores `Payload Mass (kg)` y `class` para el sitio seleccionado, y etiquetar el punto con color usando `Booster Version Category` de igual manera.

El punto de dispersión que renderizaste debería verse como la siguiente captura de pantalla:



Si necesitas más referencias sobre callbacks de dash y gráficos de dispersión de plotly, consulta la [Referencia de Plotly Dash](#) hacia el final de este laboratorio.

Encontrando Perspectivas Visualmente

Ahora que el panel de control está completo, deberías poder usarlo para analizar los datos de lanzamientos de SpaceX y responder las siguientes preguntas:

- ¿Qué sitio tiene el mayor número de lanzamientos exitosos?
- ¿Qué sitio tiene la tasa de éxito de lanzamiento más alta?
- ¿Qué rango(s) de carga útil tiene la tasa de éxito de lanzamiento más alta?
- ¿Qué rango(s) de carga útil tiene la tasa de éxito de lanzamiento más baja?
- ¿Qué versión del cohete F9 (v1.0, v1.1, FT, B4, B5, etc.) tiene la tasa de éxito de lanzamiento más alta?

Referencia de Plotly Dash

Componente de dropdown (entrada)

Consulta [aquí](#) para más detalles sobre `dcc.Dropdown()`

Componente de control deslizante de rango (entrada)

Consulta [aquí](#) para más detalles sobre `dcc.RangeSlider()`

Componente de gráfico de pastel (salida)

Consulta [aquí](#) para más detalles sobre gráficos de pastel de plotly

Componente de gráfico de dispersión (salida)

Consulta [aquí](#) para más detalles sobre gráficos de dispersión de plotly

Autor

[Yan Luo](#)

Otros contribuyentes

Joseph Santarcangelo