SPACE-Y FIRST STAGE LANDING SUCCESS STUDY

Summary:

1. A link to your GitHub repository containing all completed notebooks and Python files
2. The completed presentation submitted in PDF format
3. Executive Summary slide
4. Introduction slide
5. Data collection and data wrangling methodology slides
6. EDA and interactive visual analytics methodology slides
7. Predictive analysis methodology slides
8. EDA with visualization results slides
9. EDA with SQL results slides
10. Interactive map results using Folium
11. Plotly Dash dashboard results slides
12. Predictive analysis (classification) results slides
13. Conclusion

Introduction:

This presentation will outline my data science study to find the best success landing rate of the first stage rocket. Public data will be used and analyzed by SQL, visual analytic visualization methodology, Plotly dashboard, Folium, pie-chart … In addition, a machine learning of predictive analysis (classification) will be trained to find the best parameters to get the highest success rate.

DATA COLLECTION AND WRANGLING

Most unsuccessful landings are planned. Space X performs a controlled landing in the oceans.

Objectives

In this lab, you will make a get request to the SpaceX API. You will also do some basic data wrangling and formating.

- Request to the SpaceX API
- Clean the requested data

Import Libraries and Define Auxiliary Functions

We will import the following libraries into the lab

[1]:
```
# Requests allows us to make HTTP requests which we will use to get data from an API
import requests
# Pandas is a software library written for the Python programming language for data manipulation and analysis.
import pandas as pd
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical funct
import numpy as np
# Datetime is a library that allows us to represent dates
import datetime

# Setting this option will print all collumns of a dataframe
pd.set_option('display.max_columns', None)
# Setting this option will print all of the data in a feature
pd.set_option('display.max_colwidth', None)
```

Below we will define a series of helper functions that will help us use the API to extract information using identification numbers in the launch data.

From the `rocket` column we would like to learn the booster name.

[2]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        if x:
```

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

[9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

[10]:
```
response=requests.get(static_json_url)
```

[11]:
```
response.status_code
```

[11]:
```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

[12]:
```
# Use json_normalize meethod to convert the json result into a dataframe
import pandas as pd

# 1. Convert the response to JSON
data_json = response.json()

# 2. Flatten the JSON into a dataframe
data = pd.json_normalize(data_json)
```

Using the dataframe `data` print the first 5 rows

[13]:
```
# Get the head of the dataframe
data.head()
```

[13]:

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules	payloads	launchpad	auto_upd
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e		

labs.cognitiveclass.ai/v2/tools/jupyterlab?ulid=ulid-3950bf2f61f89c02c297b52a2ef57061beb5053d

File Edit View Run Kernel Git Tabs Settings Help

jupyter-labs-spacex-data-cc

Markdown | git | Run as Pipeline | Python

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

[4]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

[5]:
```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Now let's start requesting rocket launch data from SpaceX API with the following URL:

[6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

[7]:
```
response = requests.get(spacex_url)
```

Check the content of the response

Would you like to receive official Jupyter news?
Please read the privacy policy.
Open privacy policy | Yes | No

Simple | Fully initialized | Python | Idle | Mem: 301.00 / 6144.00 MB | Mode: Command | Ln 1, Col 1 | English (United States) | jupyter-labs-spacex-data-collection-api.ipynb

-3°C Cloudy | Search | ENG US | 5:24 PM 2026-02-06

'landpad': '5e9e3032383ecb6bb234e7ca'} 05T23:24:00.000Z 11-05

90 rows × 9 columns

Data Wrangling

We can see below that some of the rows are missing values in our dataset.

[32]:
```
data_falcon9.isnull().sum()
```

[32]:
```
rocket           0
payloads         0
launchpad        0
cores            0
flight_number    0
date_utc         0
date             0
BoosterVersion   0
FlightNumber     0
dtype: int64
```

Before we can continue we must deal with these missing values. The `LandingPad` column will retain None values to represent when landing pads were not used.

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

[33]:
```
# Check if Falcon 1 still exists in the new dataframe
print(data_falcon9['BoosterVersion'].value_counts())
```

```
Falcon 9    90
Name: BoosterVersion, dtype: int64
```

[35]:
```
print(data_falcon9.columns)
data['PayloadMass'] = data['payloads'].map(lambda x: x.get('mass_kg') if isinstance(x, dict) else None)
```

```
Index(['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc',
       'date', 'BoosterVersion', 'FlightNumber'],
      dtype='object')
```

EDA AND INTERACTIVE VISUAL ANALYTICS METHODOLOGY

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

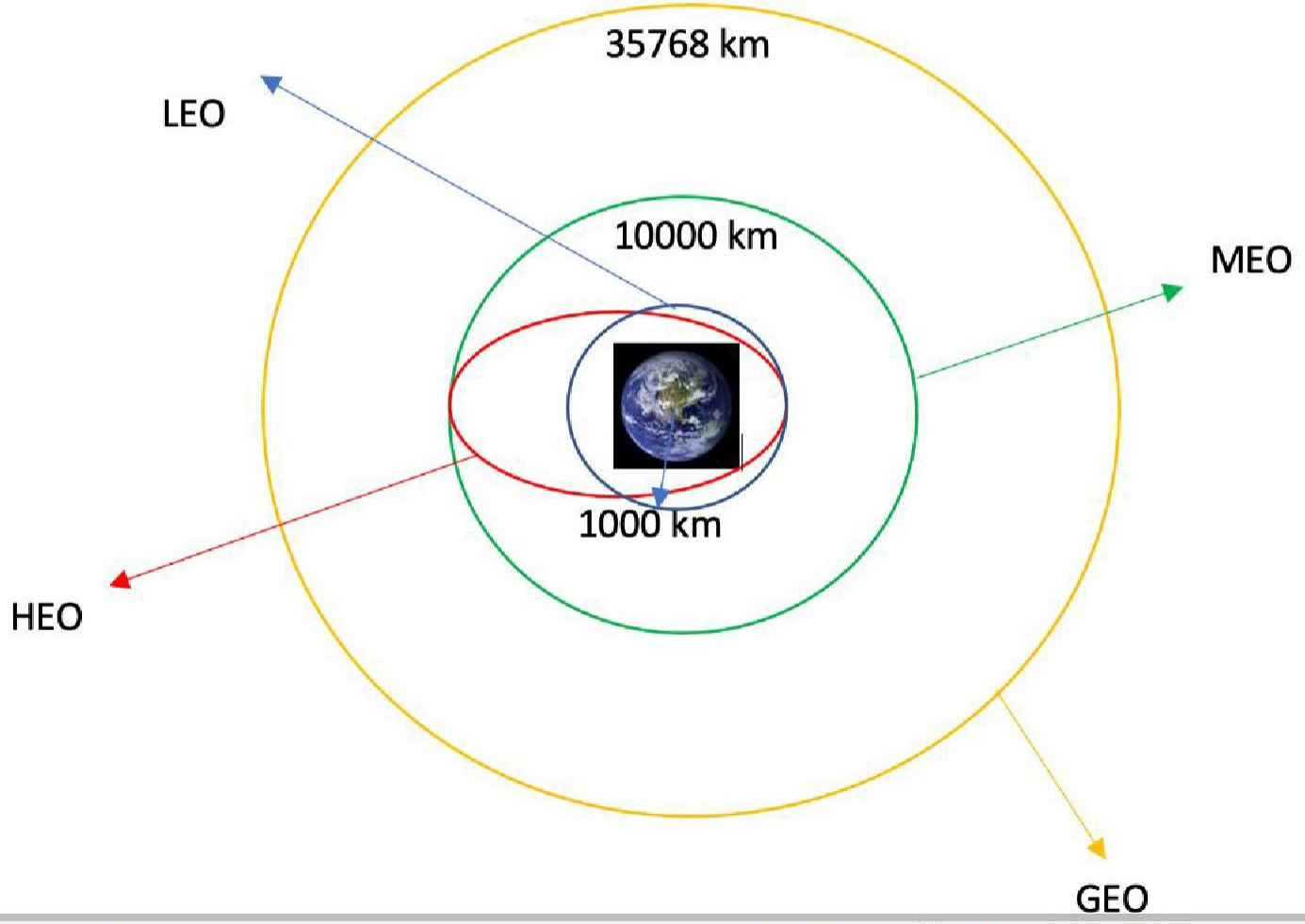
Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

[6]:
```
# Calculate the number of launches for each site
launch_site_counts = df['LaunchSite'].value_counts()

# Display the results
print(launch_site_counts)
```

```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

Each launch aims to an dedicated orbit, and here are some common orbit types:

- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO [1].
- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation[2].
- **GTO**(Geostationary Transfer Orbit): A geostationary transfer orbit is an elliptical Earth orbit used to transfer satellites from low Earth orbit (LEO) to geostationary orbit (GEO). In a GTO, the perigee (closest point to Earth) is much lower than GEO altitude, while the apogee (farthest point) reaches approximately 22,236 miles (35,786 kilometers) above Earth's equator — the altitude of a geostationary orbit. Satellites in GTO use onboard propulsion to circularize their orbit at GEO altitude, where they can provide services such as weather monitoring, communications, and surveillance. [3] .
- **SSO (or SO)**: It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .
- **ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth [5] .


35768 km
LEO
10000 km
MEO
1000 km
HEO
GEO

File Edit View Run Kernel Tabs Settings Help

labs-jupyter-spacex-Data wra

Markdown

Open in... Python 3 (ipykernel)

/ ... / labs / module_1_L3 /

Name	Modified
labs-jupyter-spac...	yesterday

GEO

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

Note: Do not count GTO, as it is a transfer orbit and not itself geostationary.

[7]:
```
# Apply value_counts on Orbit column
# Filter out GTO and then count the occurrences of each remaining orbit
orbit_counts = df[df['Orbit'] != 'GTO']['Orbit'].value_counts()

# Display the results
print(orbit_counts)
```

```
Orbit
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: count, dtype: int64
```

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

[8]:
```
# Calculate the occurrences of each landing outcome
landing_outcomes = df['Outcome'].value_counts()

# Display the results to see the different categories
print(landing_outcomes)
```

Simple 0 1 Python 3 (ipykernel) | Idle Initialized (additional servers needed) Mode: Command Ln 1, Col 1 labs-jupyter-spacex-Data wrangling.ipynb English (United States) 0

Weather alert In effect

Search

ENG US 3:03 PM 2026-02-06

labs.cognitiveclass.ai/v2/tools/jupyterlab?ulid=ulid-3ee2ae0b10b4321e8e614ad076b2bf3ed4892296

File Edit View Run Kernel Tabs Settings Help

labs-jupyter-spacex-Data wra

Markdown

Open in... Python 3 (ipykernel)

/ ... / labs / module_1_L3 /

Name	Modified
labs-jupyter-spac...	yesterday

```
SO        1
GEO       1
Name: count, dtype: int64
```

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

[8]:
```
# Calculate the occurrences of each landing outcome
landing_outcomes = df['Outcome'].value_counts()

# Display the results to see the different categories
print(landing_outcomes)
```

```
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: count, dtype: int64
```

[ ]:
```
# landing_outcomes = values on Outcome column
```

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

[9]:
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

Simple 0 1 Python 3 (ipykernel) | Idle Initialized (additional servers needed) Mode: Command Ln 1, Col 1 labs-jupyter-spacex-Data wrangling.ipynb English (United States) 0

Weather alert In effect

Search

ENG US 3:03 PM 2026-02-06

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

[11]:
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
# Create the landing_class list
# 0 if the outcome is in bad_outcomes, otherwise 1
landing_class = [0 if outcome in bad_outcomes else 1 for outcome in df['Outcome']]

# Assign the list to a new column in the dataframe
df['Class'] = landing_class
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

[12]:
```
df['Class']=landing_class
df[['Class']].head(8)
```

[12]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

[13]:
```
df.head(5)
```

[13]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0

File Edit View Run Kernel Tabs Settings Help

/ ... / labs / module_1_L3 /

Name | Modified

labs-jupyter-spac... | yesterday

labs-jupyter-spacex-Data wra

Markdown | Open in... | Python 3 (ipykernel)

[12]:

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

[13]:
```
df.head(5)
```

[13]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

We can use the following line of code to determine the success rate:

[14]:
```
df["Class"].mean()
```

[14]:
```
np.float64(0.6666666666666666)
```

We can now export it to a CSV for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

```
df.to_csv("dataset_part_2.csv", index=False)
```

Authors

Simple | 0 | 1 | Python 3 (ipykernel) | Idle | Initialized (additional servers needed) | Mode: Command | Ln 1, Col 1 | labs-jupyter-spacex-Data wrangling.ipynb | English (United States) | 0

Weather alert In effect | Search | ENG US | 3:03 PM 2026-02-06

EDA WITH VISUALIZATIONS RESULTS

Launcher | edadataviz.ipynb

Markdown | Python (Pyodide)

Flight Number

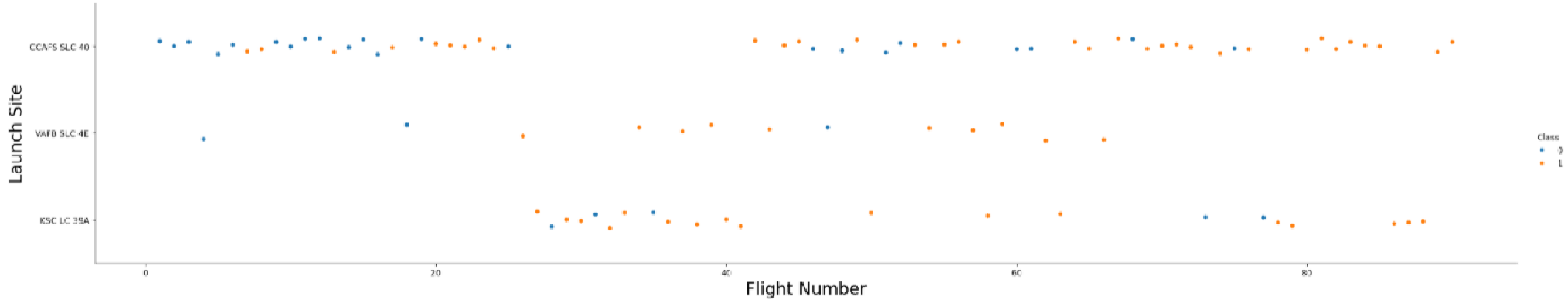
Next, let's drill down to each site visualize its detailed launch records.

TASK 1: Visualize the relationship between Flight Number and Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

[5]:

```
# Create the catplot
sns.catplot(data=df, x="FlightNumber", y="LaunchSite", hue="Class", aspect=5)

# Add descriptive labels
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.show()
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the Launch site, and hue to be the class value
```



Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

TASK 2: Visualize the relationship between Payload Mass and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

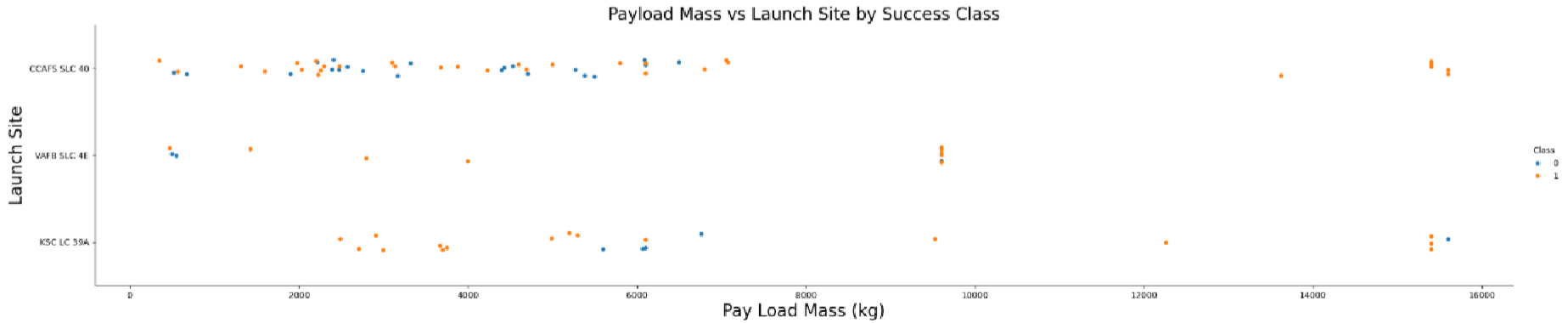
Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

TASK 2: Visualize the relationship between Payload Mass and Launch Site

We also want to observe if there is any relationship between launch sites and their payload mass.

[6]:

```
# Plotting Payload Mass vs Launch Site with Class as hue
# We use aspect=5 to stretch the plot horizontally for better readability
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect=5)

# Setting labels for clarity
plt.xlabel("Pay Load Mass (kg)", fontsize=20)
plt.ylabel("Launch Site", fontsize=20)
plt.title("Payload Mass vs Launch Site by Success Class", fontsize=20)

plt.show()
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the Launch site, and hue to be the class value
```



Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

TASK 3: Visualize the relationship between success rate of each orbit type

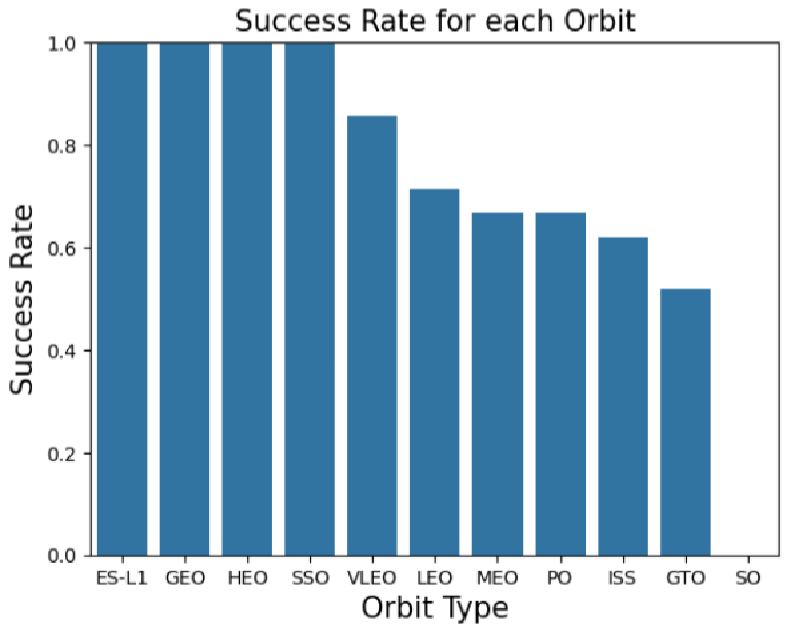
File Edit View Run Kernel Tabs Settings Help

EDADATAVIZ.IPYNB

Launcher | edadataviz.ipynb

Markdown | Python (Pyodide)

```
sns.barplot(x="Orbit", y="Class", data=df_orbit)

# 4. Add labels and title
plt.xlabel("Orbit Type", fontsize=15)
plt.ylabel("Success Rate", fontsize=15)
plt.title("Success Rate for each Orbit", fontsize=15)

# Optional: Set the y-axis to a percentage scale (0 to 1)
plt.ylim(0, 1)

plt.show()
# HINT use groupby method on Orbit column and get the mean of Class column
```



Analyze the plotted bar chart to identify which orbits have the highest success rates.

TASK 4: Visualize the relationship between FlightNumber and Orbit type

Simple | 0 | 1 | Python (Pyodide) | Idle | Mode: Command | Ln 1, Col 1 | edadataviz.ipynb | 2

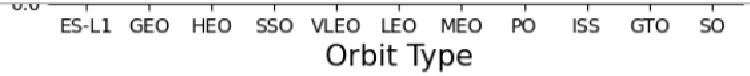
Cold warning In effect | Search | ENG US | 3:07 PM 2026-02-06

File Edit View Run Kernel Tabs Settings Help

EDADATAVIZ.IPYNB

Launcher | edadataviz.ipynb

Markdown | Python (Pyodide)

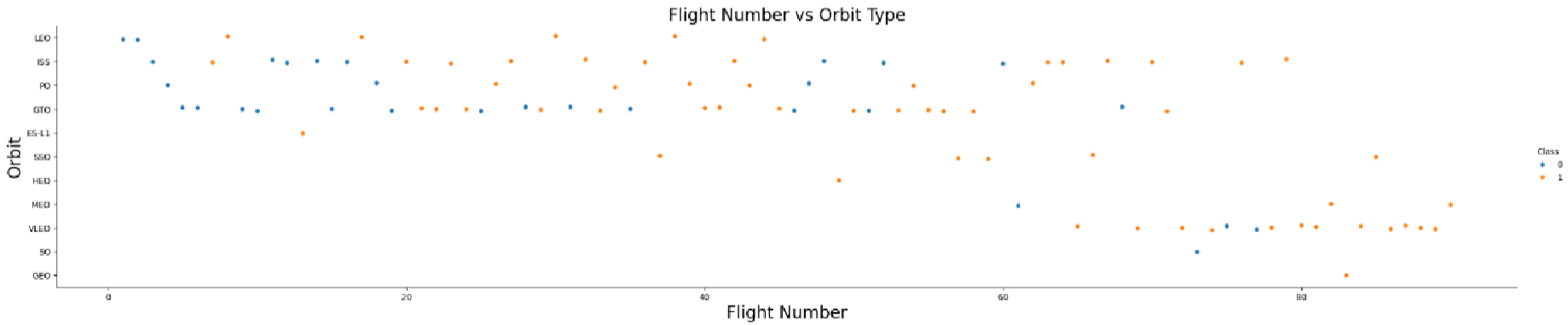


Analyze the plotted bar chart to identify which orbits have the highest success rates.

TASK 4: Visualize the relationship between FlightNumber and Orbit type

For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

[8]:
```
# Plotting Flight Number vs Orbit with Class as hue
sns.catplot(x="FlightNumber", y="Orbit", hue="Class", data=df, aspect=5)

# Setting labels
plt.xlabel("Flight Number", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.title("Flight Number vs Orbit Type", fontsize=20)

plt.show()
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
```



You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

TASK 5: Visualize the relationship between Payload Mass and Orbit type

Simple | 0 | 1 | Python (Pyodide) | Idle | Mode: Command | Ln 1, Col 1 | edadataviz.ipynb | 2

Cold warning In effect | Search | ENG US | 3:07 PM 2026-02-06

File Edit View Run Kernel Tabs Settings Help

EDADATAVIZ.IPYNB

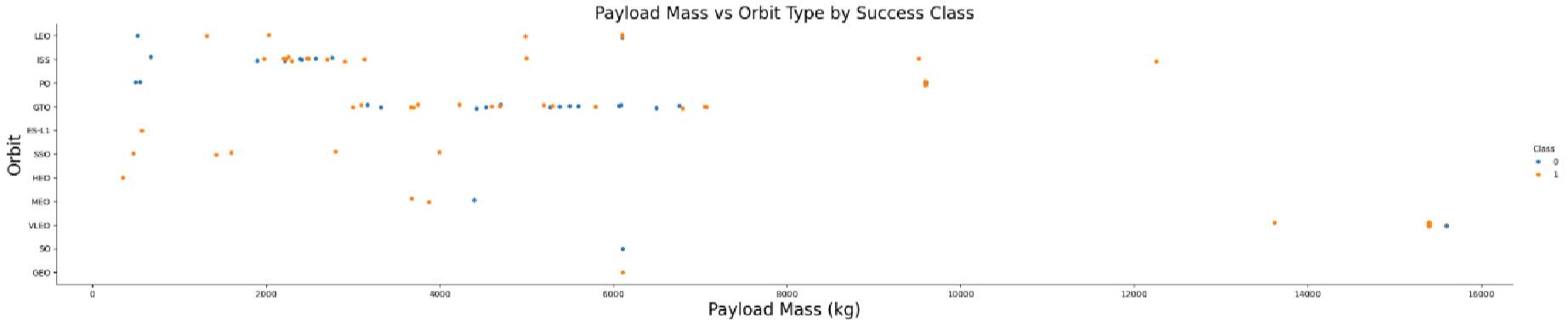
Launcher | edadataviz.ipynb

Markdown | Python (Pyodide)

TASK 5: Visualize the relationship between Payload Mass and Orbit type

Similarly, we can plot the Payload Mass vs. Orbit scatter point charts to reveal the relationship between Payload Mass and Orbit type

[9]:
```
# Plotting Payload Mass vs Orbit with Class as hue
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df, aspect=5)

# Setting labels for clarity
plt.xlabel("Payload Mass (kg)", fontsize=20)
plt.ylabel("Orbit", fontsize=20)
plt.title("Payload Mass vs Orbit Type by Success Class", fontsize=20)

plt.show()
# Plot a scatter point chart with x axis to be Payload Mass and y axis to be the Orbit, and hue to be the class value
```



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

TASK 6: Visualize the launch success yearly trend

You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

Simple | 0 | 1 | Python (Pyodide) | Idle | Mode: Command | Ln 1, Col 1 | edadataviz.ipynb | 2

Cold warning In effect | Search | ENG US | 3:07 PM 2026-02-06

labs.cognitiveclass.ai/v2/tools/jupyterlite?ulid=ulid-bcb171190569d4ac32e7a0fba87ff5868cbfdd09

File Edit View Run Kernel Tabs Settings Help

EDADATAVIZ.IPYNB

Launcher | edadataviz.ipynb

Markdown | Python (Pyodide)

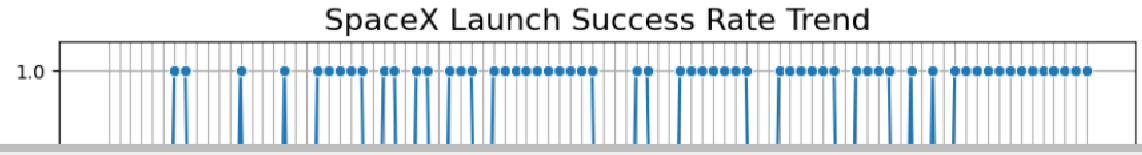
TASK 6: Visualize the launch success yearly trend

You can plot a line chart with x axis to be `Year` and y axis to be average success rate, to get the average launch success trend.

The function will help you get the year from the date:

[ ]:
```
# A function to Extract years from the date
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```

[10]:
```
# 1. Group by the updated 'Date' column (years) and calculate the success rate
# The mean of the 'Class' column (1s and 0s) gives the percentage of success
df_yearly_success = df.groupby('Date')['Class'].mean().reset_index()

# 2. Create the line chart
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_yearly_success, x="Date", y="Class", marker='o')

# 3. Add titles and labels for clarity
plt.title('SpaceX Launch Success Rate Trend', fontsize=16)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Success Rate', fontsize=14)
plt.grid(True)

plt.show()
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
```



Simple | 0 | 1 | Python (Pyodide) | Idle | Mode: Command | Ln 1, Col 1 | edadataviz.ipynb | 2

Trending videos The Muppet Sh...

Search

ENG US

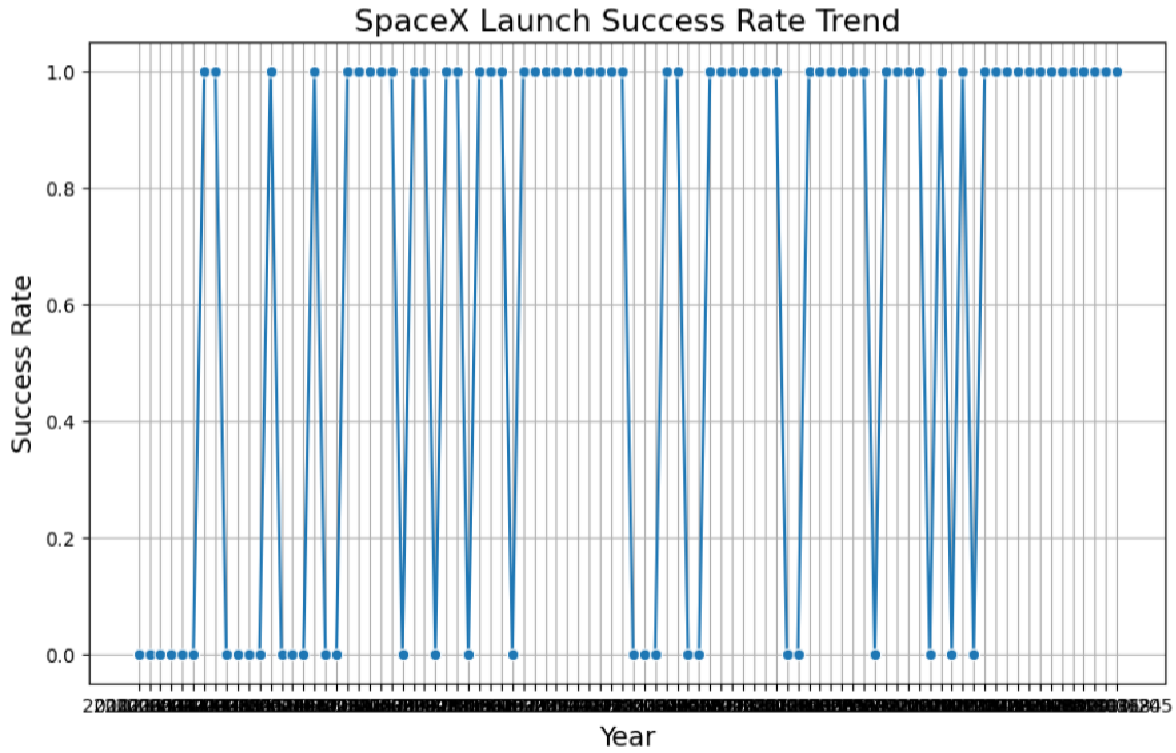
3:08 PM 2026-02-06

PREDICTIVE ANALYSIS METHODOLOGY

Launcher | edadataviz.ipynb

Markdown | Python (Pyodide)

```
# 3. Add titles and labels for clarity
plt.title('SpaceX Launch Success Rate Trend', fontsize=16)
plt.xlabel('Year', fontsize=14)
plt.ylabel('Success Rate', fontsize=14)
plt.grid(True)

plt.show()
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
```



you can observe that the sucess rate since 2013 kept increasing till 2020

File Edit View Run Kernel Tabs Settings Help

EDADATAVIZ.IPYNB

Launcher | edadataviz.ipynb

Markdown | Python (Pyodide)

TASK 7: Create dummy variables to categorical columns

Use the function `get_dummies` and `features` dataframe to apply OneHotEncoder to the column `Orbits`, `LaunchSite`, `LandingPad`, and `Serial`. Assign the value to the variable `features_one_hot`, display the results using the method head. Your result dataframe must include all features including the encoded ones.

[12]:

```
# List of categorical columns to encode
# Note: In the standard SpaceX dataset, the column is 'Orbit' (singular).
# If your dataframe uses 'Orbits', ensure the name matches exactly.
categorical_cols = ['Orbit', 'LaunchSite', 'LandingPad', 'Serial']

# Apply get_dummies to the specified columns
# This will create new columns for each unique value and drop the original categorical columns
features_one_hot = pd.get_dummies(features, columns=categorical_cols)

# Display the first 5 rows of the new dataframe
features_one_hot.head()
# HINT: Use get_dummies() function on the categorical columns
```

[12]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_
0	1	6104.959412	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	
1	2	525.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	
2	3	677.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	
3	4	500.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	
4	5	3170.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False	

5 rows × 80 columns

TASK 8: Cast all numeric columns to `float64`

Now that our `features_one_hot` dataframe only contains numbers, cast the entire dataframe to variable type `float64`

Simple | 0 | 1 | Python (Pyodide) | Idle | Mode: Command | Ln 1, Col 1 | edadataviz.ipynb | 2

Trending videos The Muppet Sh... | Search | ENG US | 3:08 PM 2026-02-06

											...						
2	3	677.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False
3	4	500.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False
4	5	3170.000000	1	False	False	False	1.0	0	False	False	...	False	False	False	False	False	False

5 rows × 80 columns

TASK 8: Cast all numeric columns to `float64`

Now that our `features_one_hot` dataframe only contains numbers, cast the entire dataframe to variable type `float64`

[13]:
```
# Cast the entire dataframe to float64
features_one_hot = features_one_hot.astype('float64')

# Display the first few rows to verify
features_one_hot.head()
# HINT: use astype function
```

[13]:

	FlightNumber	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCount	Orbit_ES-L1	Orbit_GEO	...	Serial_B1048	Serial_B1049	Serial_B1050	Serial_B1051	Serial_B1054	Serial_B1056	Serial_I
0	1.0	6104.959412	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
1	2.0	525.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
2	3.0	677.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	4.0	500.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
4	5.0	3170.000000	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	

5 rows × 80 columns

We can now export it to a **CSV** for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

```
features_one_hot.to_csv('dataset_part_3.csv', index=False)
```

```
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).agg({
    'Lat': 'first',
    'Long': 'first',
    'class': 'mean'
})

# 2. Rename the column for clarity (optional)
launch_sites_df.rename(columns={'class': 'Success Rate'}, inplace=True)

launch_sites_df
```

[37]:

	Launch Site	Lat	Long	Success Rate
0	CCAFS LC-40	28.562302	-80.577356	0.269231
1	CCAFS SLC-40	28.563197	-80.576820	0.428571
2	KSC LC-39A	28.573255	-80.646895	0.769231
3	VAFB SLC-4E	34.632834	-120.610745	0.400000

Above coordinates are just plain numbers that can not give you any intuitive insights about where are those launch sites. If you are very good at geography, you can interpret those numbers directly in your mind. If not, that's fine too. Let's visualize those locations by pinning them on a map.

We first need to create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

[13]:
```
# Start location is NASA Johnson Space Center
nasa_coordinate = [29.559684888503615, -95.0830971930759]
site_map = folium.Map(location=nasa_coordinate, zoom_start=10)
```

We could use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate. For example,

[30]:
```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Space Center'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
```

EDA WITH SQL RESULTS

labs.cognitiveclass.ai/v2/tools/jupyterlab?ulid=ulid-09ab93de324a8be291b8904715d3d5eec692a4c8

File Edit View Run Kernel Tabs Settings Help

labs-jupyter-spacex-Data wra | jupyter-labs-eda-sql-courser

Markdown | Open in... | Python 3 (ipykernel)

[9]:
```
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null
```

```
 * sqlite:///my_data1.db
Done.
```

[9]: []

Tasks

Now write and execute SQL queries to solve the assignment tasks.

Note: If the column names are in mixed case enclose it in double quotes For Example "Landing_Outcome"

Task 1

Display the names of the unique launch sites in the space mission

[23]:
```
%sql  SELECT DISTINCT "Launch_Site" FROM SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

[23]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

Task 2

Display 5 records where launch sites begin with the string 'CCA'

[33]:
```
%%sql
SELECT * FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'  LIMIT 5;
```

Simple | 0 | 2 | Python 3 (ipykernel) | Idle | Initialized (additional servers needed) | Mode: Command | Ln 1, Col 1 | jupyter-labs-eda-sql-coursera_sqllite.ipynb | English (United States) | 0

Weather alert In effect | Search | ENG US | 3:04 PM 2026-02-06

Task 2

Display 5 records where launch sites begin with the string 'CCA'

[33]:
```
%%sql
SELECT * FROM SPACEXTBL
WHERE "Launch_Site" LIKE 'CCA%'  LIMIT 5;
```

```
 * sqlite:///my_data1.db
Done.
```

[33]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

[34]:
```
%%sql
SELECT SUM("PAYLOAD_MASS__KG_") AS "Total_Payload_Mass_NASA_CRS"
FROM SPACEXTBL
WHERE "Customer" = 'NASA (CRS)';
```

```
 * sqlite:///my_data1.db
Done.
```

[34]:

Total_Payload_Mass_NASA_CRS
45596

Task 4

Display average payload mass carried by booster version F9 v1.1

```
 * sqlite:///my_data1.db
Done.
```

[34]:

Total_Payload_Mass_NASA_CRS
45596

Task 4

Display average payload mass carried by booster version F9 v1.1

[35]:

```
%%sql
SELECT AVG("PAYLOAD_MASS__KG_") AS "Average_Payload_Mass"
FROM SPACEXTBL
WHERE "Booster_Version" = 'F9 v1.1';
```

```
 * sqlite:///my_data1.db
Done.
```

[35]:

Average_Payload_Mass
2928.4

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

[36]:

```
%%sql
SELECT MIN("Date") AS "First_Successful_Ground_Landing"
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (ground pad)';
```

```
 * sqlite:///my_data1.db
Done.
```

[36]:

First_Successful_Ground_Landing
2015-12-22

Task 6

File Edit View Run Kernel Tabs Settings Help

labs-jupyter-spacex-Data wra | jupyter-labs-eda-sql-courser

Code | Open in... | Python 3 (ipykernel)

[36]:
```
%%sql
SELECT MIN("Date") AS "First_Successful_Ground_Landing"
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (ground pad)';
```

```
 * sqlite:///my_data1.db
Done.
```

[36]:

First_Successful_Ground_Landing
2015-12-22

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

[37]:
```
%%sql
SELECT DISTINCT "Booster_Version"
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (drone ship)'
  AND "PAYLOAD_MASS__KG_" > 4000
  AND "PAYLOAD_MASS__KG_" < 6000;
```

```
 * sqlite:///my_data1.db
Done.
```

[37]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Task 7

List the total number of successful and failure mission outcomes

[38]:
```
%%sql
SELECT "Mission_Outcome", COUNT(*) AS "Total_Count"
FROM SPACEXTBL
```

File Edit View Run Kernel Tabs Settings Help

labs-jupyter-spacex-Data wra | jupyter-labs-eda-sql-courser

Code | Open in... | Python 3 (ipykernel)

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

Task 7

List the total number of successful and failure mission outcomes

[38]:
```
%%sql
SELECT "Mission_Outcome", COUNT(*) AS "Total_Count"
FROM SPACEXTBL
GROUP BY "Mission_Outcome";
```

```
 * sqlite:///my_data1.db
Done.
```

[38]:

Mission_Outcome	Total_Count
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

[39]:
```
%%sql
SELECT "Booster_Version", "PAYLOAD_MASS__KG_"
FROM SPACEXTBL
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_")
    FROM SPACEXTBL
);
```

```
 * sqlite:///my_data1.db
Done.
```

Task 8

List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

[39]:
```
%%sql
SELECT "Booster_Version", "PAYLOAD_MASS__KG_"
FROM SPACEXTBL
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_")
    FROM SPACEXTBL
);
```

```
 * sqlite:///my_data1.db
Done.
```

[39]:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

F9 B5 B1049.7	15600

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

[40]:
```
%%sql
SELECT
    substr("Date", 6, 2) AS "Month",
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Failure (drone ship)'
  AND substr("Date", 0, 5) = '2015';
```

```
 * sqlite:///my_data1.db
Done.
```

[40]:

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

[41]:
```
%%sql
SELECT "Landing_Outcome", COUNT(*) AS "Outcome_Count"
FROM SPACEXTBL
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY "Outcome_Count" DESC;
```

```
 * sqlite:///my_data1.db
Done.
```

[41]:

Landing_Outcome	Outcome_Count

```
 * sqlite:///my_data1.db
Done.
```

[40]:

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

Task 10

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

[41]:

```
%%sql
SELECT "Landing_Outcome", COUNT(*) AS "Outcome_Count"
FROM SPACEXTBL
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY "Outcome_Count" DESC;
```

```
 * sqlite:///my_data1.db
Done.
```

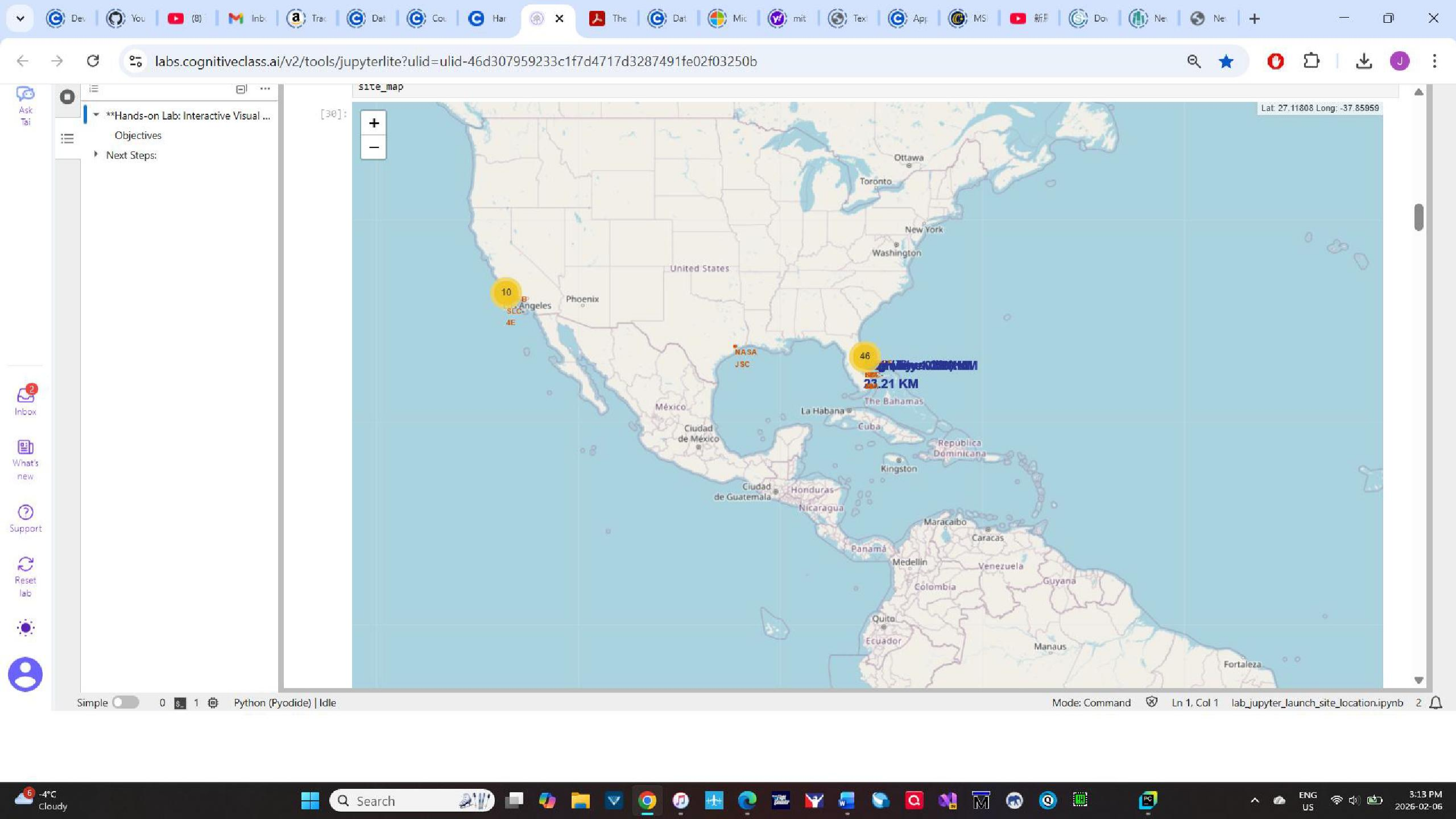
[41]:

Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Reference Links

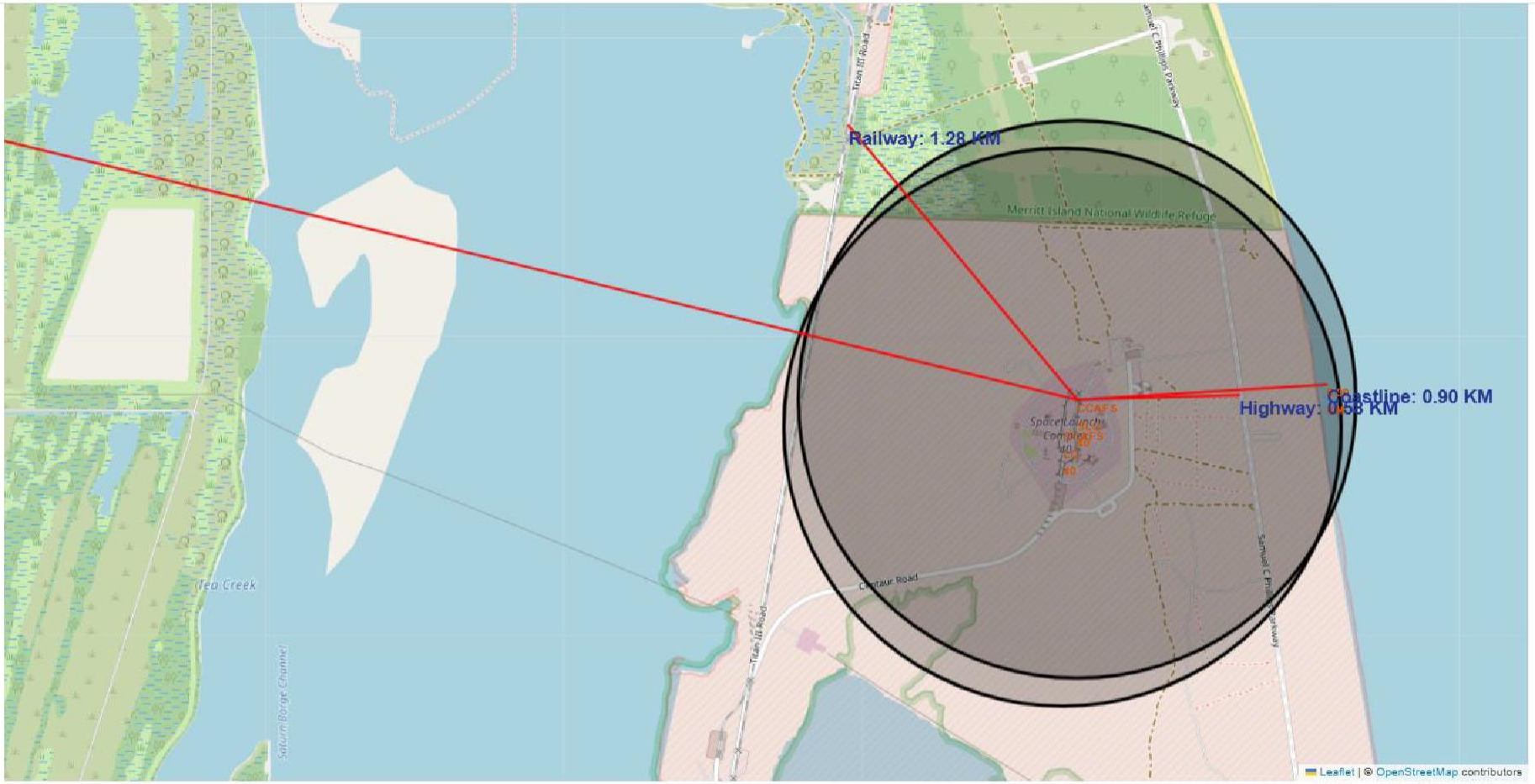
- Hands-on Lab : String Patterns, Sorting and Grouping

FOLIUM RESULTS


labs.cognitiveclass.ai/v2/tools/jupyterlite?ulid=ulid-46d307959233c1f7d4717d3287491fe02f03250b
Ask Tai
**Hands-on Lab: Interactive Visual ...
Objectives
Next Steps:
site_map
[30]:
Lat: 27.11808 Long: -37.85959
Ottawa
Toronto
New York
Washington
United States
Phoenix
10
NASA JSC
46
22.21 KM
The Bahamas
La Habana
Cuba
México
Ciudad de México
República Dominicana
Kingston
Ciudad de Guatemala
Honduras
Nicaragua
Maracaibo
Caracas
Panamá
Medellín
Venezuela
Guyana
Colombia
Quito
Ecuador
Manaus
Fortaleza
Inbox
What's new
Support
Reset lab
Simple
Python (Pyodide) | Idle
Mode: Command
Ln 1, Col 1
lab_jupyter_launch_site_location.ipynb
-4°C Cloudy
Search
ENG US
3:13 PM 2026-02-06

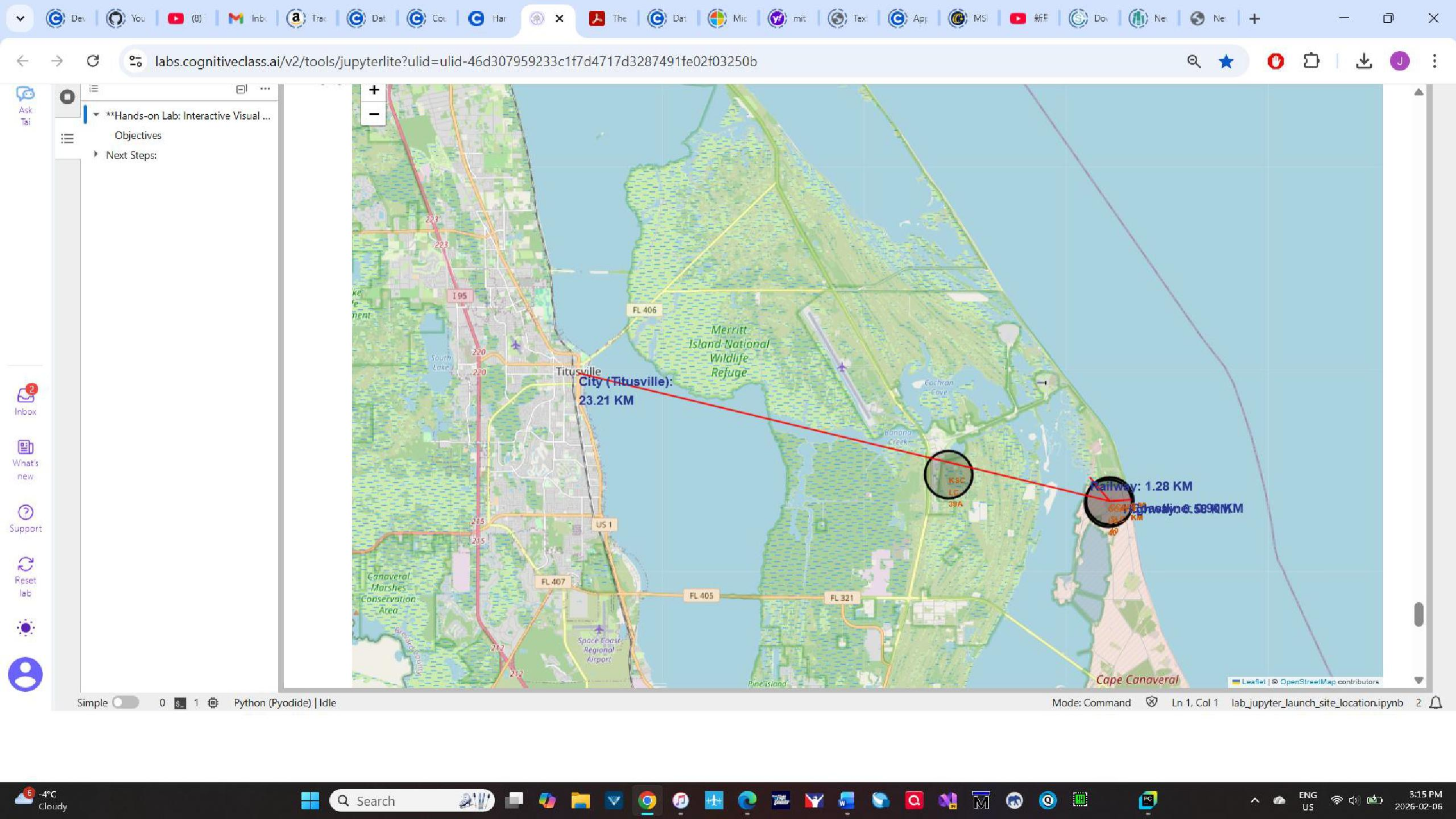
After you plot distance lines to the proximities, you can answer the following questions easily:

- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?



After you plot distance lines to the proximities, you can answer the following questions easily:

- Are launch sites in close proximity to railways?
- Are launch sites in close proximity to highways?


labs.cognitiveclass.ai/v2/tools/jupyterlite?ulid=ulid-46d307959233c1f7d4717d3287491fe02f03250b
**Hands-on Lab: Interactive Visual ...
Objectives
Next Steps:
Titusville
City (Titusville):
23.21 KM
Railway: 1.28 KM
Merritt Island National Wildlife Refuge
Canaveral Marshes Conservation Area
Space Coast Regional Airport
Cape Canaveral
I 95
FL 406
US 1
FL 407
FL 405
FL 321
Leaflet | © OpenStreetMap contributors
Simple
Python (Pyodide) | Idle
Mode: Command
Ln 1, Col 1
lab_jupyter_launch_site_location.ipynb

PLOTLY DASH DASHBOARD RESULTS

labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

File Edit Selection View Go Run Terminal Help

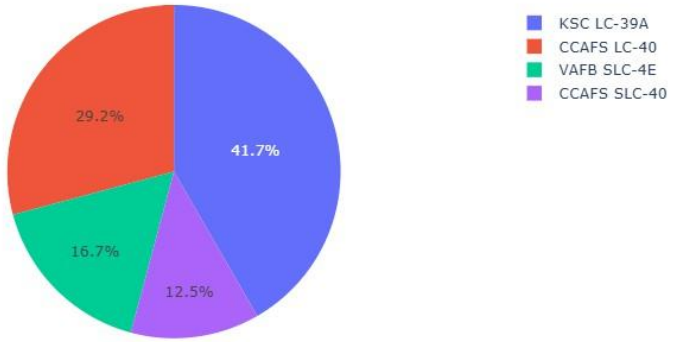
Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

SpaceX Launch Records Dashboard

All Sites

Total Success Launches By Site




```
theia@theiadocker-jkmchow2016: /home/project
127.0.0.1 - - [06/Feb/2026 15:23:28] "GET /_dash-component-suites/dash/dcc/async-slider.js HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

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

theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:23 PM 2026-02-06

labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

File Edit Selection View Go Run Terminal Help

Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

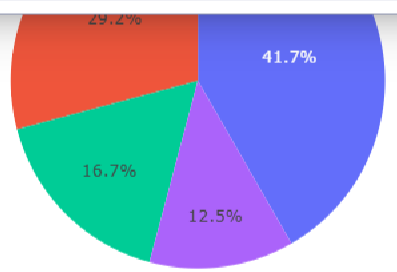
https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

SpaceX Launch Records Dashboard

All Sites

Search

All Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A




```
theia@theiadocker-jkmchow2016: /home/project
127.0.0.1 - - [06/Feb/2026 15:23:28] "GET /_dash-component-suites/dash/dcc/async-slider.js
 HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

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

theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:23 PM 2026-02-06

labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

File Edit Selection View Go Run Terminal Help

Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

SpaceX Launch Records Dashboard

All Sites

Search

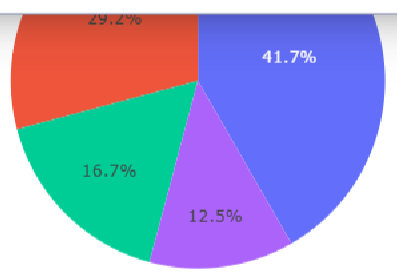
All Sites

CCAFS LC-40

VAFB SLC-4E

KSC LC-39A

CCAFS SLC-40



theia@theiadocker-jkmchow2016: /home/project

```
127.0.0.1 - - [06/Feb/2026 15:23:28] "GET /_dash-component-suites/dash/dcc/async-slider.js
 HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

theia@theiadocker-anitaj: /home/project

```
theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:24 PM 2026-02-06

labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

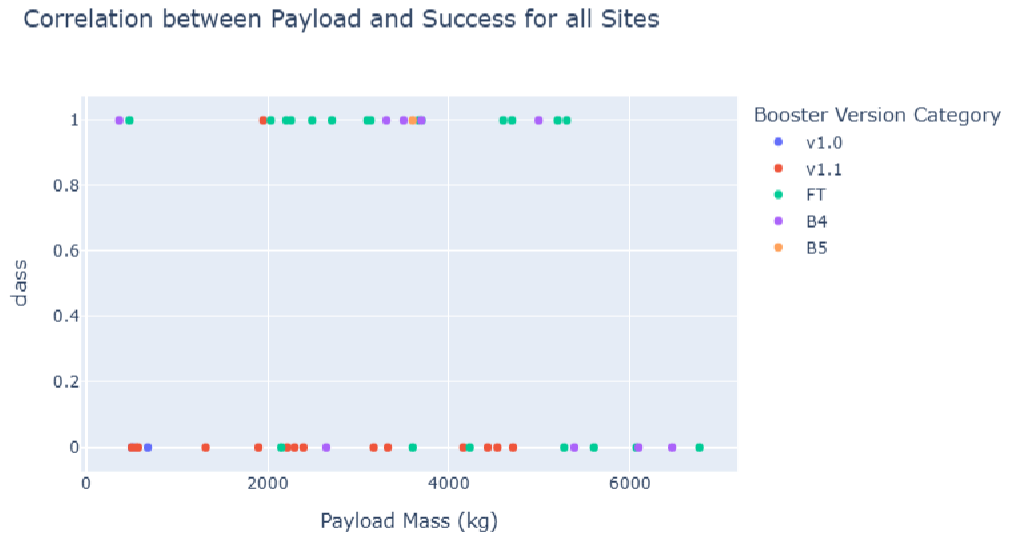
File Edit Selection View Go Run Terminal Help

Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Payload range (Kg):

0 — 0 2500 5000 7500 10000 — 9600



theia@theiadocker-jkmchow2016: /home/project

```
127.0.0.1 - - [06/Feb/2026 15:23:28] "GET /_dash-component-suites/dash/dcc/async-slider.js
 HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

theia@theiadocker-anitaj: /home/project

```
theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:24 PM 2026-02-06

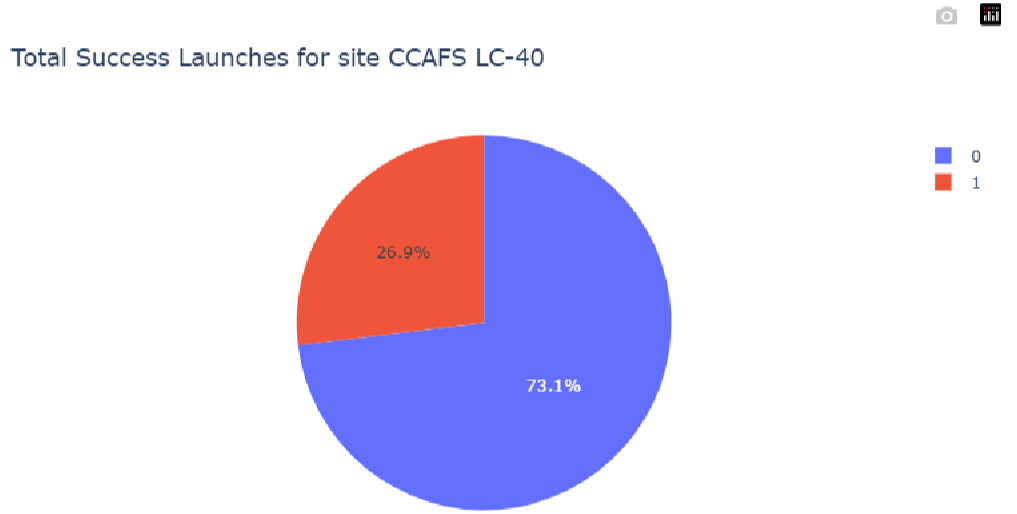
labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

File Edit Selection View Go Run Terminal Help

Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

CCAFS LC-40



Payload range (Kg):

```
theia@theiadocker-jkmchow2016: /home/project
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:24:36] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:24:36] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

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

theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

Lab guide | Ask Tai | Inbox | What's new | Support | Reset lab

-4°C Cloudy | Search | ENG US | 3:24 PM 2026-02-06

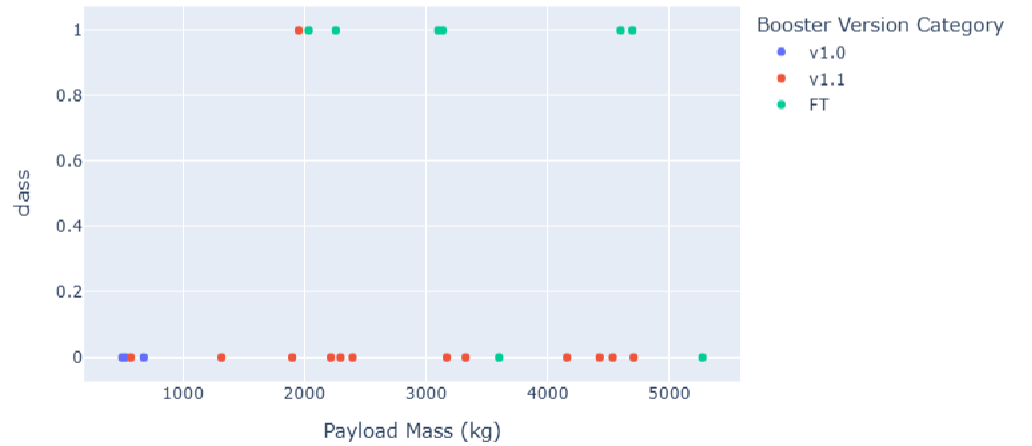
File Edit Selection View Go Run Terminal Help

Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Payload range (Kg):

0 — 0 2500 5000 7500 10000 — 9600




```
theia@theiadocker-jkmchow2016: /home/project
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:23:28] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:24:36] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:24:36] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

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

theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:24 PM 2026-02-06

labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

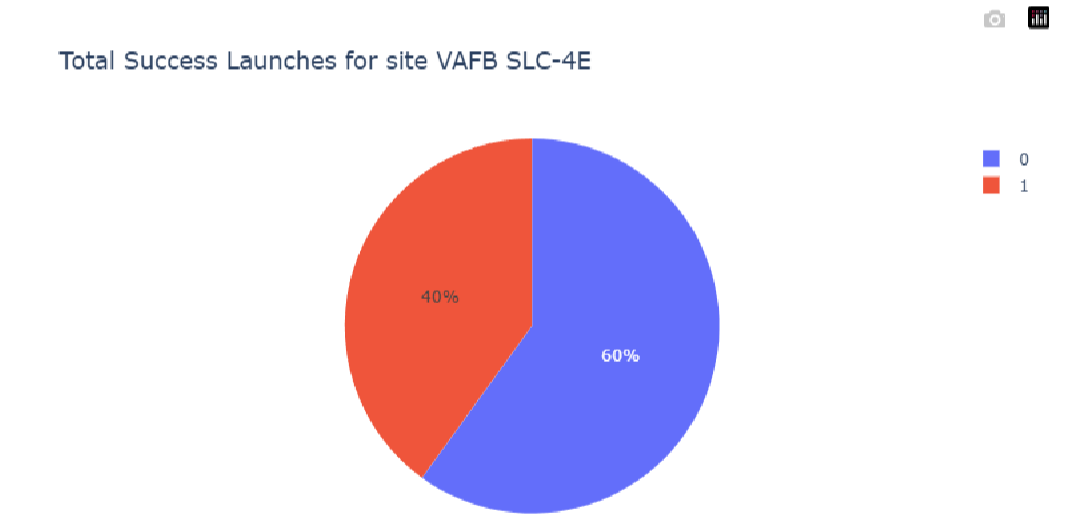
File Edit Selection View Go Run Terminal Help

Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

SpaceX Launch Records Dashboard

VAFB SLC-4E




```
theia@theiadocker-jkmchow2016: /home/project
127.0.0.1 - - [06/Feb/2026 15:24:36] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:24:36] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:05] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:05] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

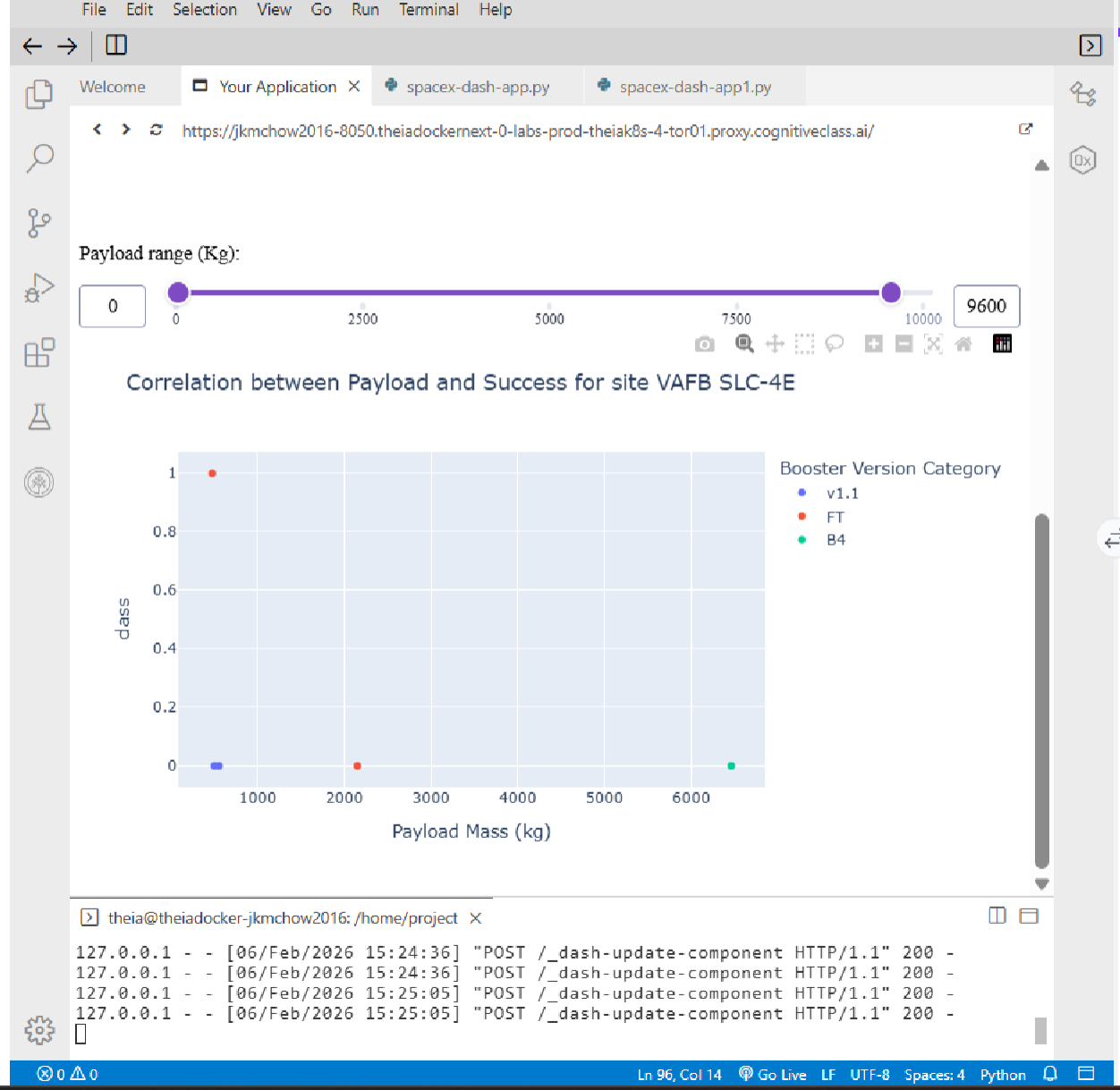
- Test the skeleton app by running the following command in the terminal:

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

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

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

theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:25 PM 2026-02-06




```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

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

theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

File Edit Selection View Go Run Terminal Help

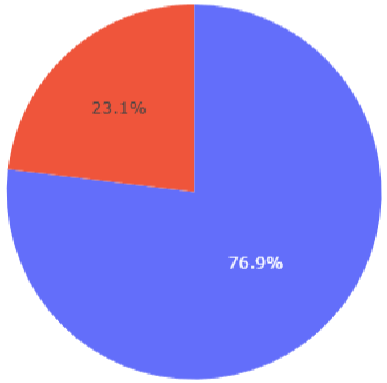
Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site KSC LC-39A




```
theia@theiadocker-jkmchow2016: /home/project
127.0.0.1 - - [06/Feb/2026 15:25:05] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:05] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:21] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:21] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

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

theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:25 PM 2026-02-06

labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

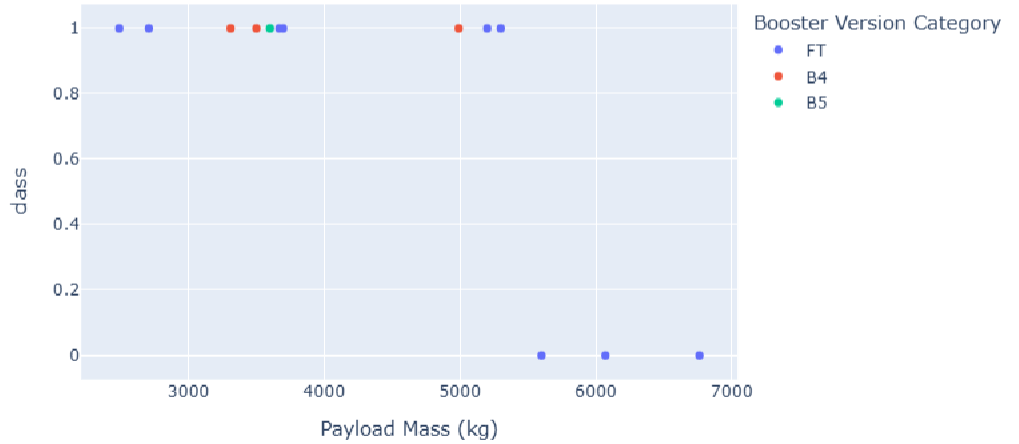
File Edit Selection View Go Run Terminal Help

Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Payload range (Kg):

0 — 0 2500 5000 7500 10000 — 9600



theia@theiadocker-jkmchow2016: /home/project

```
127.0.0.1 - - [06/Feb/2026 15:25:05] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:05] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:21] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:21] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

theia@theiadocker-anitaj: /home/project

```
theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:25 PM 2026-02-06

labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

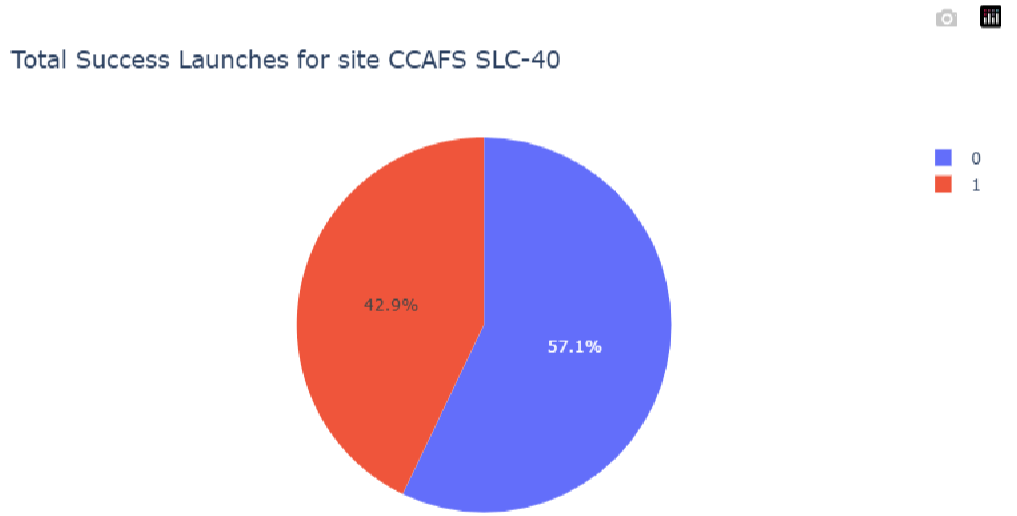
File Edit Selection View Go Run Terminal Help

Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

SpaceX Launch Records Dashboard

CCAFS SLC-40




```
theia@theiadocker-jkmchow2016: /home/project
127.0.0.1 - - [06/Feb/2026 15:25:21] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:21] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:35] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:35] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

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

theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:25 PM 2026-02-06

labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-995b4342107e4c20f51620fda3814d437767d6f4

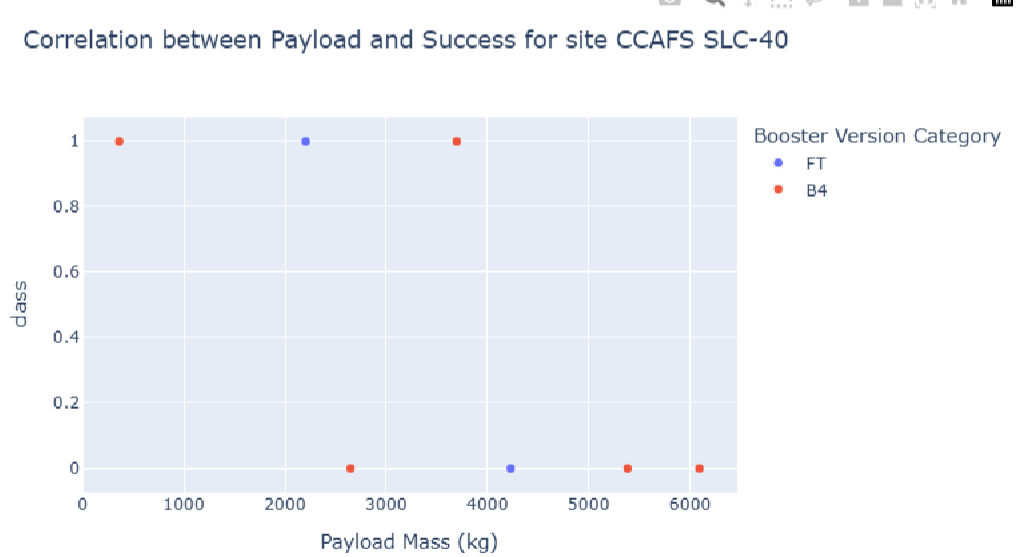
File Edit Selection View Go Run Terminal Help

Welcome | Your Application | spacex-dash-app.py | spacex-dash-app1.py

https://jkmchow2016-8050.theiadockernext-0-labs-prod-theiak8s-4-tor01.proxy.cognitiveclass.ai/

Payload range (Kg):

0 — 0 2500 5000 7500 10000 — 9600



theia@theiadocker-jkmchow2016: /home/project

```
127.0.0.1 - - [06/Feb/2026 15:25:21] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:21] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:35] "POST /_dash-update-component HTTP/1.1" 200 -
127.0.0.1 - - [06/Feb/2026 15:25:35] "POST /_dash-update-component HTTP/1.1" 200 -
```

Ln 96, Col 14 Go Live LF UTF-8 Spaces: 4 Python

```
Requirement already satisfied: pytz>=2020.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /home/theia/.local/lib/python3.8/site-packages (from pandas) (2.8.2)
Collecting numpy>=1.20.3
  Downloading numpy-1.24.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.3/17.3 MB 36.6 MB/s eta 0:00:00
Collecting dash-html-components==2.0.0
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting plotly>=5.0.0
  Downloading plotly-5.13.1-py2.py3-none-any.whl (15.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 15.2/15.2 MB 45.6 MB/s eta 0:00:00
Collecting dash-core-components==2.0.0
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: Flask>=1.0.4 in /home/theia/.local/lib/python3.8/site-packages (from dash) (2.2.2)
Requirement already satisfied: importlib-metadata>=3.6.0 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash)
(4.12.0)
Requirement already satisfied: Werkzeug>=2.2.2 in /home/theia/.local/lib/python3.8/site-packages (from Flask>=1.0.4->dash) (2.2.2)
```

Download a skeleton dashboard application and dataset

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

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

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

```
1  wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/t4-Vy4iOU19i8y6
```

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

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

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

theia@theiadocker-anitaj: /home/project

```
theia@theiadocker-anitaj:/home/project$ python3.8 spacex_dash_app.py
spacex_dash_app.py:4: UserWarning:
The dash_html_components package is deprecated. Please replace
`import dash_html_components as html` with `from dash import html`
  import dash_html_components as html
spacex_dash_app.py:5: UserWarning:
The dash_core_components package is deprecated. Please replace
`import dash_core_components as dcc` with `from dash import dcc`
```

-4°C Cloudy | Search | ENG US | 3:25 PM 2026-02-06

PREDICTIVE ANALYSIS(CLASSIFICATION) RESULTS

File Edit View Run Kernel Tabs Settings Help

Launcher | SpaceX_Machine Learning Pr

Markdown | Python (Pyodide)

89	90.0	3681.000000	1.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0	

90 rows × 83 columns

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` then assign it to the variable `Y`,make sure the output is a Pandas series (only one bracket df['name of column']).

[8]:
```
# Extract the 'Class' column as a Series and convert to a NumPy array
Y = data['Class'].to_numpy()

# To verify the type and content
print(type(Y))
print(Y)
```

```
<class 'numpy.ndarray'>
[0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 1 0 0 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1
 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 0 1
 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

TASK 2

Standardize the data in `X` then reassign it to the variable `X` using the transform provided below.

[9]:
```
# students get this
# 1. Create the StandardScaler object
transform = preprocessing.StandardScaler()

# 2. Fit the scaler to the data and transform it
# 3. Reassign the result back to X
X = transform.fit_transform(X)
```

We split the data into training and testing data using the function `train_test_split`. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function `GridSearchCV`.

TASK 3

```
# 3. Reassign the result back to X
X = transform.fit_transform(X)
```

We split the data into training and testing data using the function `train_test_split`. The training data is divided into validation data, a second set used for training data; then the models are trained and hyperparameters are selected using the function `GridSearchCV`.

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

[10]:
```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

[11]:
```
# Y_test.shape
print(f"Number of test samples: {Y_test.shape[0]}")
```

```
Number of test samples: 18
```

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[15]:
```
# 1. Define the parameter grid
parameters = {'C': [0.01, 0.1, 1], 'penalty': ['l2'], 'solver': ['lbfgs']}

# 2. Create a Logistic Regression object
lr = LogisticRegression()

# 3. Create a GridSearchCV object with 10-fold cross-validation
logreg_cv = GridSearchCV(lr, parameters, cv=10)

# 4. Fit the object to the training data
logreg_cv.fit(X_train, Y_train)
```

[11]:
```
# Y_test.shape
print(f"Number of test samples: {Y_test.shape[0]}")
```

```
Number of test samples: 18
```

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[15]:
```
# 1. Define the parameter grid
parameters = {'C': [0.01, 0.1, 1], 'penalty': ['l2'], 'solver': ['lbfgs']}

# 2. Create a Logistic Regression object
lr = LogisticRegression()

# 3. Create a GridSearchCV object with 10-fold cross-validation
logreg_cv = GridSearchCV(lr, parameters, cv=10)

# 4. Fit the object to the training data
logreg_cv.fit(X_train, Y_train)
```

[15]:

GridSearchCV

estimator: LogisticRegression

LogisticRegression

We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the accuracy on the validation data using the data attribute `best_score_`.

[16]:
```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8464285714285713
```

TASK 5

Calculate the accuracy on the test data using the method `score`:

TASK 5

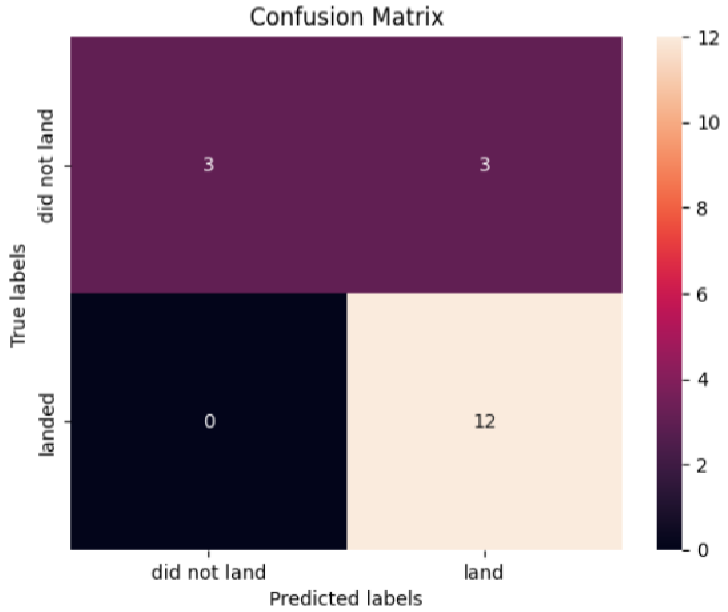
Calculate the accuracy on the test data using the method `score`:

[17]:
```
test_accuracy = logreg_cv.score(X_test, Y_test)

print(f"Accuracy on test data: {test_accuracy}")
```

```
Accuracy on test data: 0.8333333333333334
```

Lets look at the confusion matrix:

[18]:
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the problem is false positives.

Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the problem is false positives.

Overview:

True Postive - 12 (True label is landed, Predicted label is also landed)

False Postive - 3 (True label is not landed, Predicted label is landed)

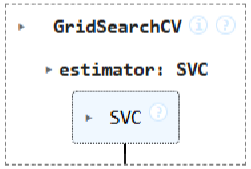
TASK 6

Create a support vector machine object then create a `GridSearchCV` object `svm_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[19]:
```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

[20]:
```
# 3. Create the GridSearchCV object with 10-fold cross-validation
svm_cv = GridSearchCV(svm, parameters, cv=10)

# 4. Fit the model to find the best parameters
svm_cv.fit(X_train, Y_train)
```

[20]:



[21]:
```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

TASK 7

Calculate the accuracy on the test data using the method `score`:

File Edit View Run Kernel Tabs Settings Help

Launcher | SpaceX_Machine Learning Pr

Markdown | Python (Pyodide)

TASK 7

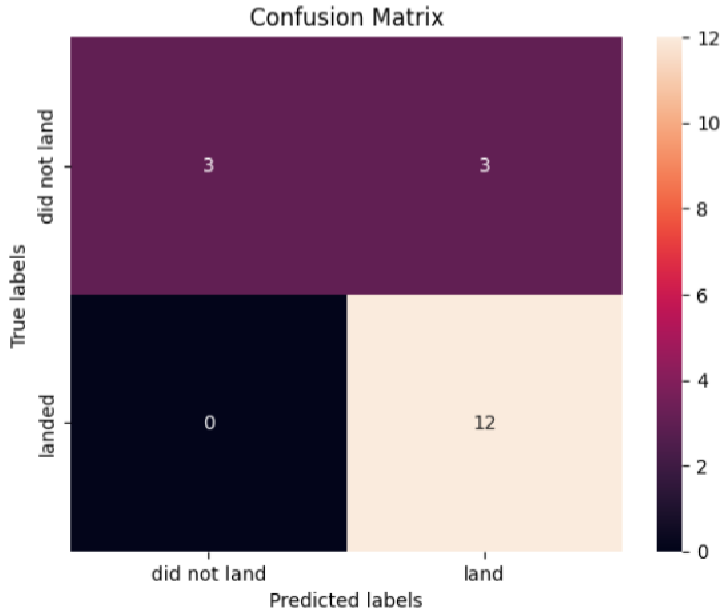
Calculate the accuracy on the test data using the method `score`:

[22]:
```
svm_test_accuracy = svm_cv.score(X_test, Y_test)

print(f"SVM Accuracy on test data: {svm_test_accuracy}")
```

```
SVM Accuracy on test data: 0.8333333333333334
```

We can plot the confusion matrix

[23]:
```
yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 8

Create a decision tree classifier object then create a `GridSearchCV` object `tree_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[24]:
```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

# 2. Create a Decision Tree classifier object
tree = DecisionTreeClassifier()
```

[25]:
```
# 3. Create a GridSearchCV object with 10-fold cross-validation
tree_cv = GridSearchCV(tree, parameters, cv=10)

# 4. Fit the object to the training data
tree_cv.fit(X_train, Y_train)
```

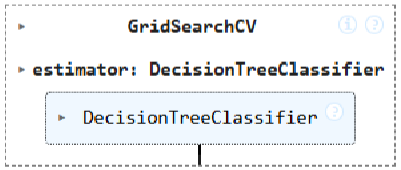
```
/lib/python3.12/site-packages/sklearn/model_selection/_validation.py:547: FitFailedWarning:
3240 fits failed out of a total of 6480.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
3240 fits failed with the following error:
Traceback (most recent call last):
  File "/lib/python3.12/site-packages/sklearn/model_selection/_validation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/lib/python3.12/site-packages/sklearn/base.py", line 1467, in wrapper
    estimator._validate_params()
```

```
    'splitter': ['best', 'random'],
    'max_depth': [2*n for n in range(1,10)],
    'max_features': ['auto', 'sqrt'],
    'min_samples_leaf': [1, 2, 4],
    'min_samples_split': [2, 5, 10]}

# 2. Create a Decision Tree classifier object
tree = DecisionTreeClassifier()
```

[25]:
```
# 3. Create a GridSearchCV object with 10-fold cross-validation
tree_cv = GridSearchCV(tree, parameters, cv=10)

# 4. Fit the object to the training data
tree_cv.fit(X_train, Y_train)
```

```
0.81785714 0.80535714 0.76071429 0.80714286 0.78928571 0.83214286
0.775      0.79821429 0.73214286 0.69107143 0.6875     0.83214286
0.81785714 0.77678571 0.71964286 0.86071429 0.81785714 0.77857143
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
       nan        nan        nan        nan        nan        nan
0.71964286 0.79107143 0.70714286 0.76428571 0.78928571 0.77857143
0.75       0.75       0.76071429 0.72142857 0.76071429 0.80535714
0.73214286 0.79107143 0.71964286 0.77678571 0.77857143 0.79107143]
 warnings.warn(
```

[25]:



[26]:
```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
accuracy : 0.8892857142857142
```

TASK 9

Calculate the accuracy of tree_cv on the test data using the method `score`:

File Edit View Run Kernel Tabs Settings Help

Launcher | SpaceX_Machine Learning Pr

Code | Python (Pyodide)

TASK 9

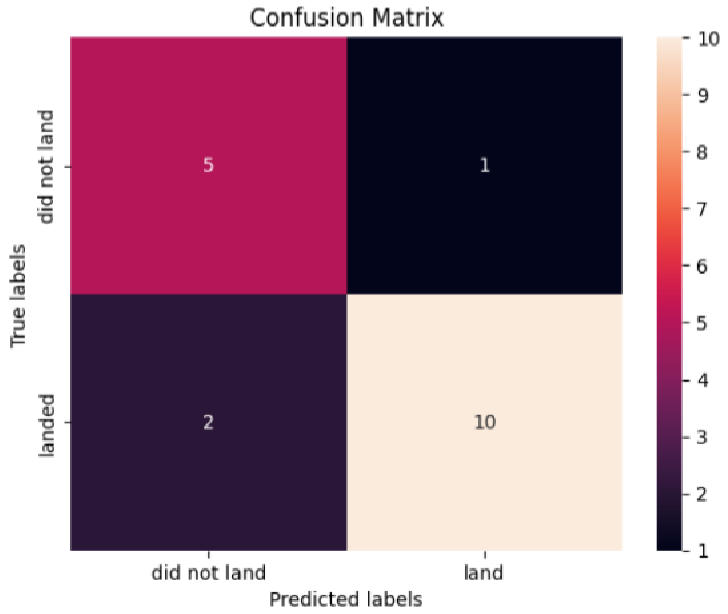
Calculate the accuracy of tree_cv on the test data using the method `score`:

[27]:
```
# Calculate the accuracy on the test data for the Decision Tree model
tree_test_accuracy = tree_cv.score(X_test, Y_test)

print(f"Decision Tree Accuracy on test data: {tree_test_accuracy}")
```

```
Decision Tree Accuracy on test data: 0.8333333333333334
```

We can plot the confusion matrix

[28]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Simple | 0 | 1 | Python (Pyodide) | Idle | Mode: Command | Ln 1, Col 1 | SpaceX_Machine Learning Prediction_Part_5.ipynb | 2

TASK 10

Create a k nearest neighbors object then create a `GridSearchCV` object `knn_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

[29]:
```
parameters = {'n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'p': [1,2]}

# 2. Create a KNN classifier object
KNN = KNeighborsClassifier()
```

[30]:
```
# 3. Create a GridSearchCV object with 10-fold cross-validation
knn_cv = GridSearchCV(KNN, parameters, cv=10)

# 4. Fit the object to the training data
knn_cv.fit(X_train, Y_train)
```

[30]:
GridSearchCV
estimator: KNeighborsClassifier
KNeighborsClassifier

[31]:
```
print("tuned hpyerparameters :(best parameters) ",knn_cv.best_params_)
print("accuracy :",knn_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'algorithm': 'auto', 'n_neighbors': 10, 'p': 1}
accuracy : 0.8482142857142858
```

TASK 11

TASK 11

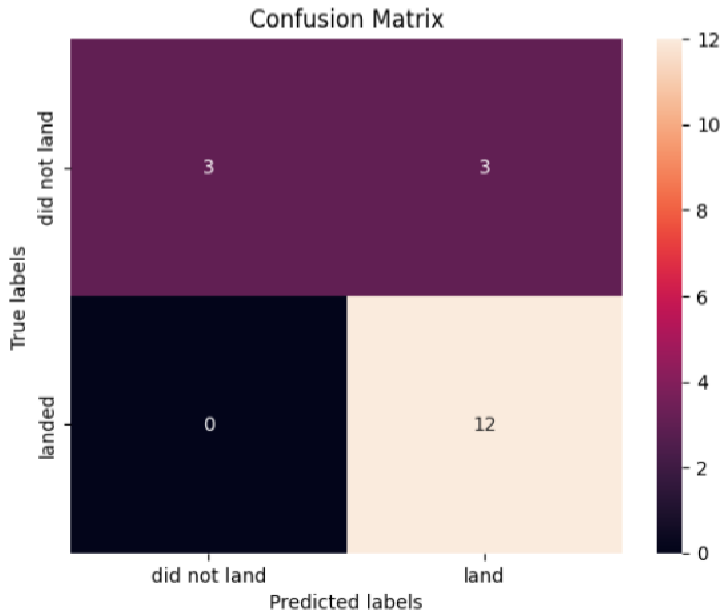
Calculate the accuracy of knn_cv on the test data using the method `score`:

[32]:
```
# Calculate the accuracy on the test data for the KNN model
knn_test_accuracy = knn_cv.score(X_test, Y_test)

print(f"KNN Accuracy on test data: {knn_test_accuracy}")
```

```
KNN Accuracy on test data: 0.8333333333333334
```

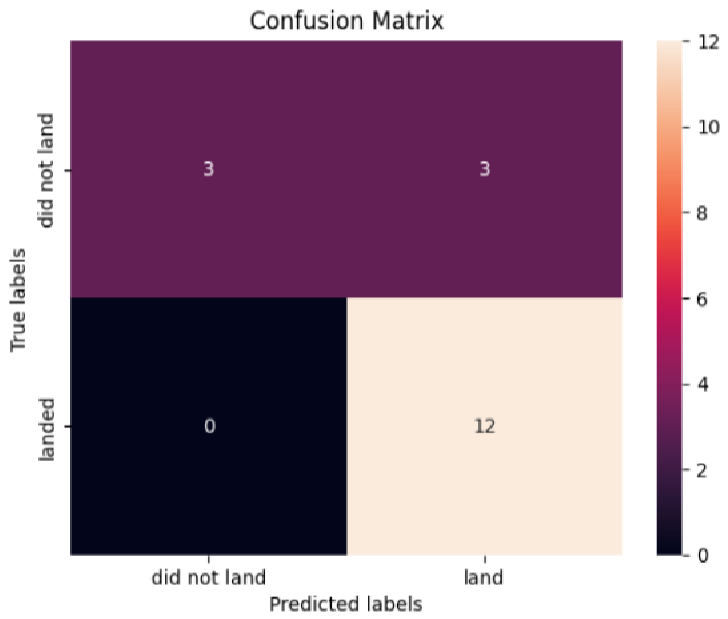
We can plot the confusion matrix

[33]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



We can plot the confusion matrix

[33]:
```
yhat = knn_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



TASK 12

Find the method performs best:

[1]:
```
Algorithm,Best GridSearchCV Score,Test Accuracy
Decision Tree,~0.889,0.833
KNN,~0.848,0.833
SVM,~0.848,0.833
Logistic Regression,~0.846,0.833
```

Conclusion:

1. The following conditions are required for the highest rate:

-

Algorithm,Best GridSearchCV Score,Test Accuracy

Decision Tr , ~0.889, 0.833

KNN, ~0.848, 0.833

SVM, ~0.848, 0.833

Logistic Regression, ~0.846, 0.833

The decision tree with highest GridSearchCV Score of machine learning will be used for predictive analysis(classification)

2. Conditions to achieve the best landing rate:

- Launch site: KSC LC-39A;
- Landing Pad: Drone-ship;
- Orbit type: ES-L1, CEO. HCO;
- Booster version: F9FT810512.