

Winning Space Race with Data Science

Huynh Ba Hai
05 November 2024



Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
 - SpaceX Data Collection using SpaceX API
 - SpaceX Data Collection with Web Scraping
 - SpaceX Data Wrangling
 - SpaceX Exploratory Data Analysis using SQL
 - Space-X EDA DataViz Using Python Pandas and Matplotlib
 - Space-X Launch Sites Analysis with Folium-Interactive Visual Analytics and Plotly Dash
 - SpaceX Machine Learning Landing Prediction
- Summary of all results
 - EDA results
 - Interactive Visual Analytics and Dashboards
 - Predictive Analysis(Classification)

Introduction

- Project background and context
 - SpaceX advertises Falcon 9 rocket launches on its cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
- Problems you want to find answers
 - In this capstone, we will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website.

Section 1

Methodology

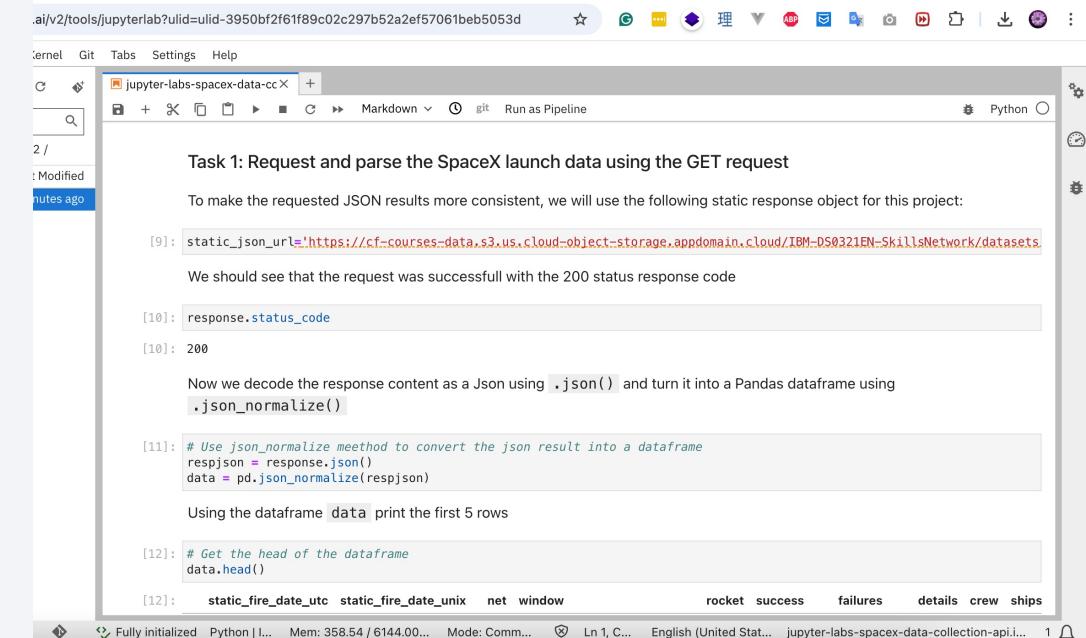
Methodology

Executive Summary

- Data collection methodology:
 - Describe how data was collected
- Perform data wrangling
 - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
 - How to build, tune, evaluate classification models

Data Collection

- Data collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API then requested and parsed the SpaceX launch data using the GET request and decoded the response content as a Json result which was then converted into a Pandas data frame.



The screenshot shows a Jupyter Notebook interface with the following code and comments:

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

Kernel Git Tabs Settings Help

jupyter-labs-spacex-data-cc

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/
```

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

```
[10]: response.status_code
```

```
[10]: 200
```

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

```
[11]: # Use json_normalize method to convert the json result into a dataframe
respjson = response.json()
data = pd.json_normalize(respjson)
```

Using the dataframe `data` print the first 5 rows

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

```
[12]: static_fire_date_utc static_fire_date_unix net window rocket success failures details crew ships
```

Fully initialized Python | ... Mem: 358.54 / 6144.00... Mode: Comm... Ln 1, C... English (United Stat... jupyter-labs-spacex-data-collection-api.i... 1 □

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts
- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose

Place your flowchart of SpaceX API calls here

Data Collection - Scraping

- Performed web scraping to collect Falcon 9 historical launch records from a Wikipedia using BeautifulSoup and request, to extract the Falcon 9 launch records from HTML table of the Wikipedia page, then created a data frame by parsing the launch HTML.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** Shows the URL `labs.cognitiveclass.ai/v2/tools/jupyterlab?ulid=d8f550da53a199b98550928754ae733955db18ec` and various browser icons.
- File Menu:** File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help.
- File Explorer:** Shows a directory structure under `/.../labs/module_1_L2/` with two files: `jupyter-lab...` (modified 7 minutes ago) and `jupyter-lab...` (modified seconds ago).
- Code Cells:**
 - [4]:

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

 Next, request the HTML page from the above URL and get a response object
 - [5]:

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

 TASK 1: Request the Falcon9 Launch Wiki page from its URL
 - [6]:

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, 'html.parser')
```

 First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.
 - [7]:

```
# Use soup.title attribute
soup.title
```

 Create a BeautifulSoup object from the HTML response
 - [7]:

```
<title>List of Falcon 9 and Falcon Heavy launches – Wikipedia</title>
```

 Print the page title to verify if the BeautifulSoup object was created properly
- Output Cells:**
 - [4]:

```
Next, request the HTML page from the above URL and get a response object
```
 - [5]:

```
TASK 1: Request the Falcon9 Launch Wiki page from its URL
```
 - [6]:

```
First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.
```
 - [7]:

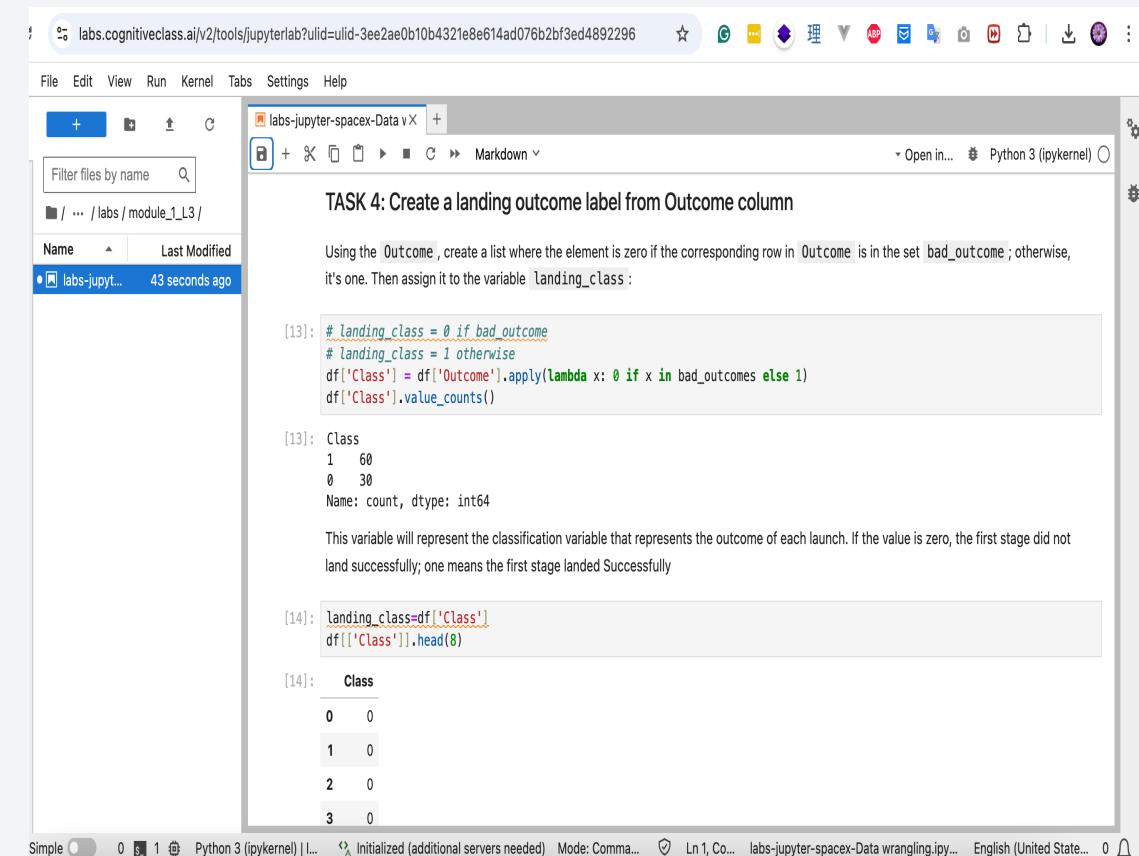
```
Create a BeautifulSoup object from the HTML response
```
 - [7]:

```
Print the page title to verify if the BeautifulSoup object was created properly
```
- Bottom Status Bar:** Shows "Fully initialized", "Python | Idle", "Mem: 488.91 / 6144.00 MB", "Mode: Command", "Ln 1, Col 1", "English (United States)", "jupyter-labs-webscraping.ipynb", and a file number "0".

- Add the GitHub URL of the completed web scraping notebook, as an external

Data Wrangling

- After obtaining and creating a Pandas DF from the collected data, data was filtered using the BoosterVersion column to only keep the Falcon 9 launches, then dealt with the missing data values in the LandingPad and PayloadMass columns. For the PayloadMass, missing data values were replaced using mean value of column.
- Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose



The screenshot shows a Jupyter Notebook environment with a file browser on the left and a code editor on the right.

File Browser: Shows a list of files in the directory `/.../labs/module_1/L3/`. One file, `labs-jupyter-spacex-Data wrangling.ipynb`, is selected and was last modified 43 seconds ago.

Code Editor:

```
[13]: # landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df['Class'].value_counts()
```

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`:

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

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

Class	Count
0	60
1	30

Name: count, dtype: int64

Class	Count
0	0
1	0
2	0
3	0

0 1 Python 3 (ipykernel) | I... Initialized (additional servers needed) Mode: Comma... Ln 1, Co... labs-jupyter-spacex-Data wrangling.ipynb English (United States...) 0

EDA with Data Visualization

- SPerformed data Analysis and Feature Engineering using Pandas and Matplotlib.i.e.
 - Exploratory Data Analysis
 - Preparing Data Feature Engineering
- Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, FlightNumber and Orbit type, Payload and Orbit type.
- Used Bar chart to Visualize the relationship between success rate of each orbit type
- Line plot to Visualize the launch success yearly trend.
- Add the GitHub URL of your completed EDA with data visualization notebook, as an external reference and peer-review purpose

EDA with SQL

- The following SQL queries were performed for EDA
 - Display the names of the unique launch sites in the space mission
 - %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
 - Display 5 records where launch sites begin with the string 'CCA'
 - %sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
 - Display average payload mass carried by booster version F9 v1.1
 - %sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
 - List the date when the first successful landing outcome in ground pad was achieved
 - %sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
- Add the GitHub URL of your completed EDA with SQL notebook, as an external reference and peer-review purpose

Build an Interactive Map with Folium

- Created folium map to marked all the launch sites, and created map objects such as markers, circles, lines to mark the success or failure of launches for each launch site.
- Created a launch set outcomes (failure=0 or success=1).
- Add the GitHub URL of your completed interactive map with Folium map, as an external reference and peer-review purpose

Build a Dashboard with Plotly Dash

- Built an interactive dashboard application with Plotly dash by:
 - Adding a Launch Site Drop-down Input Component
 - Adding a callback function to render success-pie-chart based on selected site dropdown
 - Adding a Range Slider to Select Payload
 - Adding a callback function to render the success-payload-scatter-chart scatter plot
- Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

Predictive Analysis (Classification)

- Summary of how I built, evaluated, improved, and found the best performing classification model
 - After loading the data as a Pandas Dataframe, I set out to perform exploratory Data Analysis and determine Training Labels by;
 - creating a NumPy array from the column Class in data, by applying the method `to_numpy()` then assigned it to the variable Y as the outcome variable.
 - Then standardized the feature dataset (x) by transforming it using `preprocessing.StandardScaler()` function from Sklearn.
 - After which the data was split into training and testing sets using the function `train_test_split` from `sklearn.model_selection` with the `test_size` parameter set to 0.2 and `random_state` to 2.
 - Add the GitHub URL of your completed predictive analysis lab, as an external reference and peer-review purpose

Results

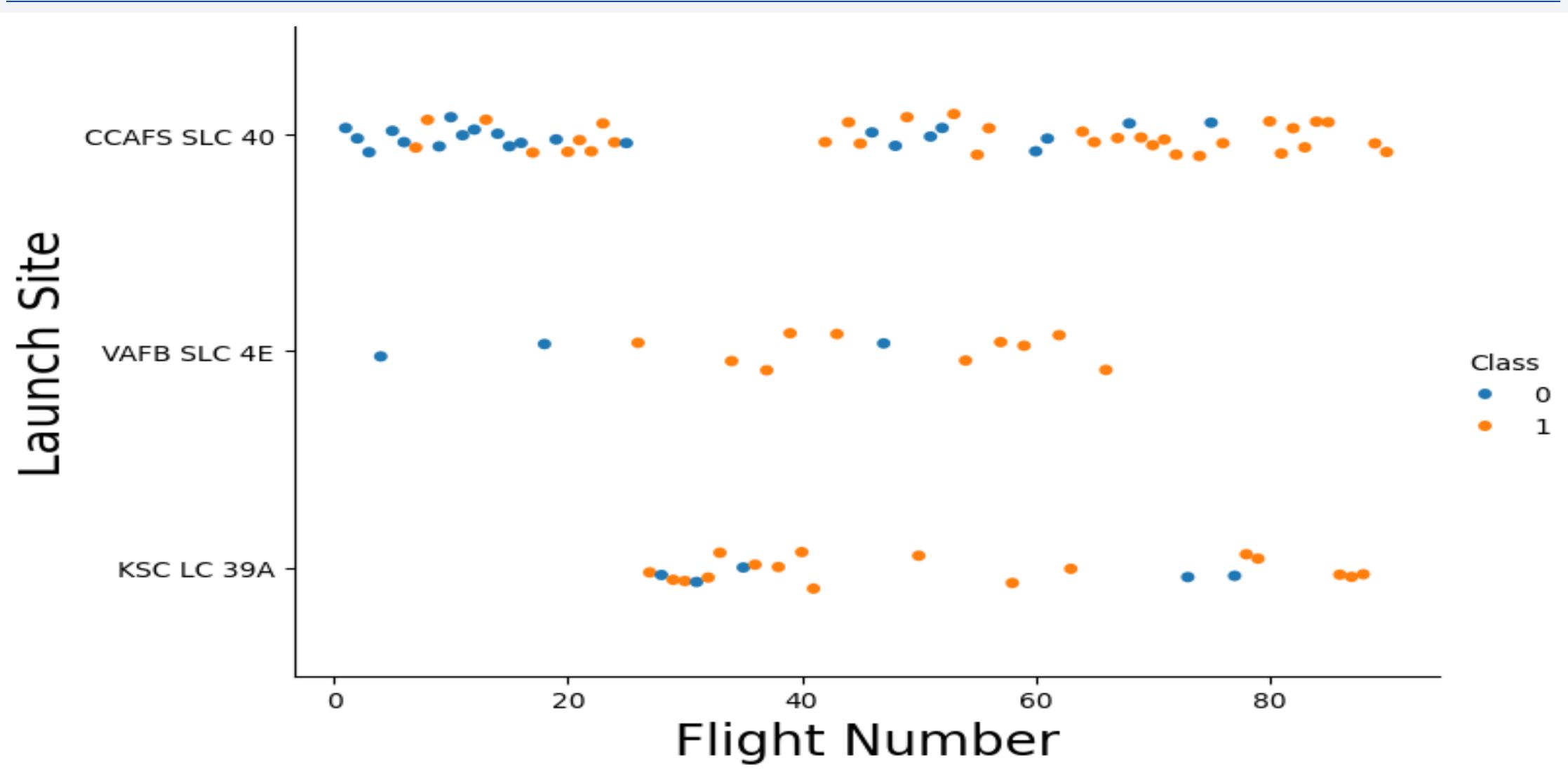
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

The background of the slide features a complex, abstract digital visualization. It consists of numerous thin, glowing lines that create a sense of depth and motion. The lines are primarily blue and red, with some green and purple highlights. They form a grid-like structure that curves and twists across the frame, resembling a three-dimensional space or a network of data points. The overall effect is futuristic and dynamic.

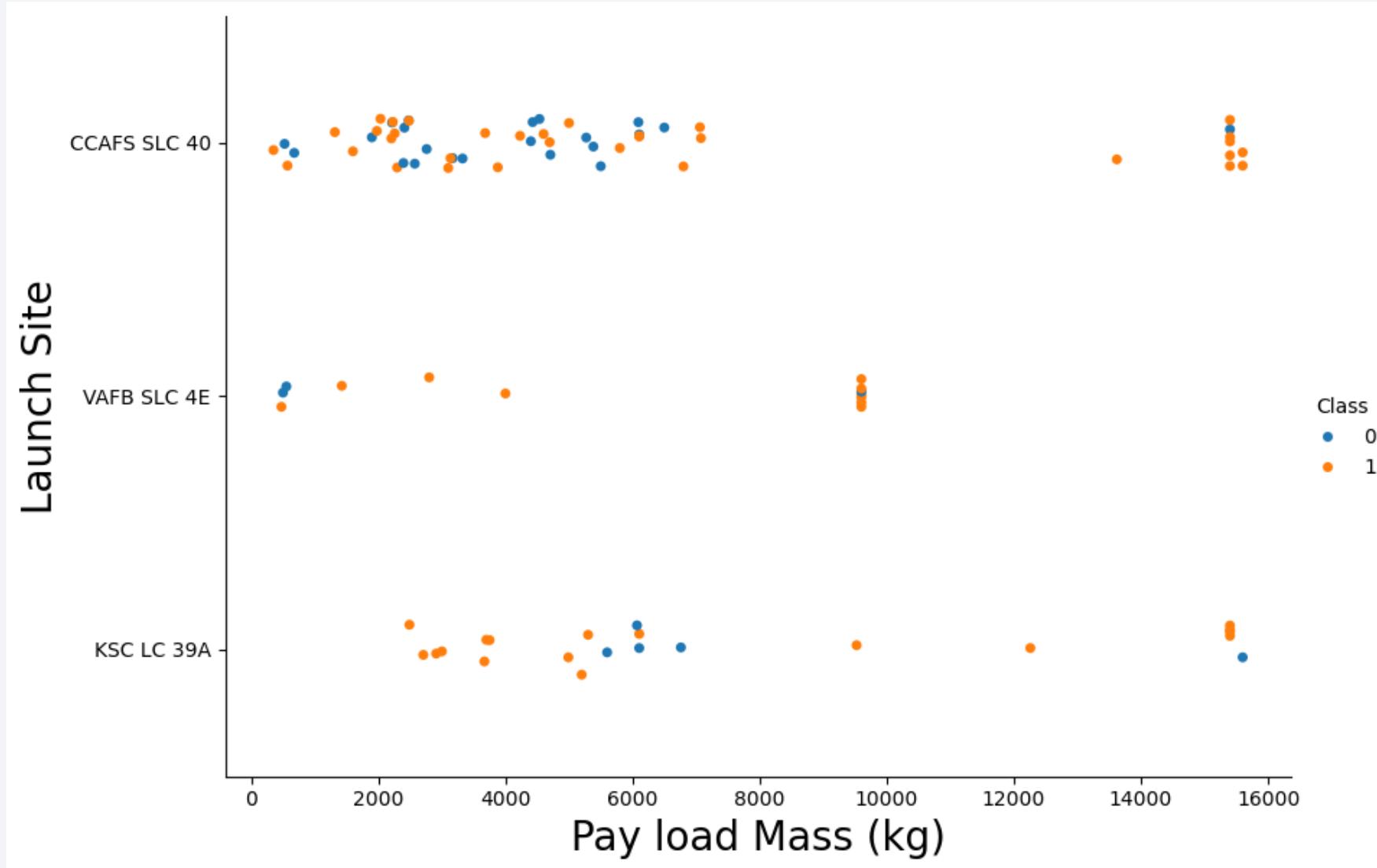
Section 2

Insights drawn from EDA

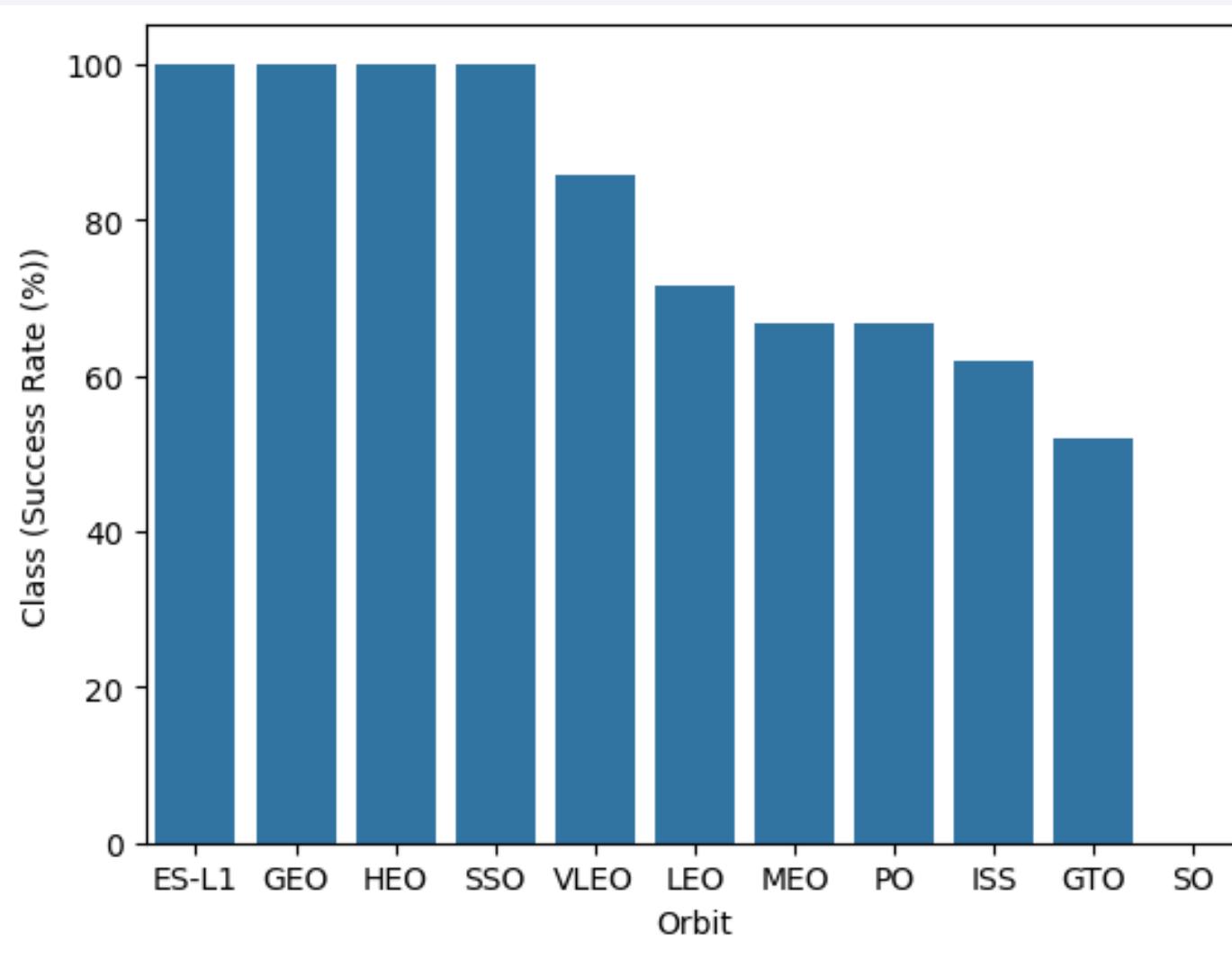
Flight Number vs. Launch Site



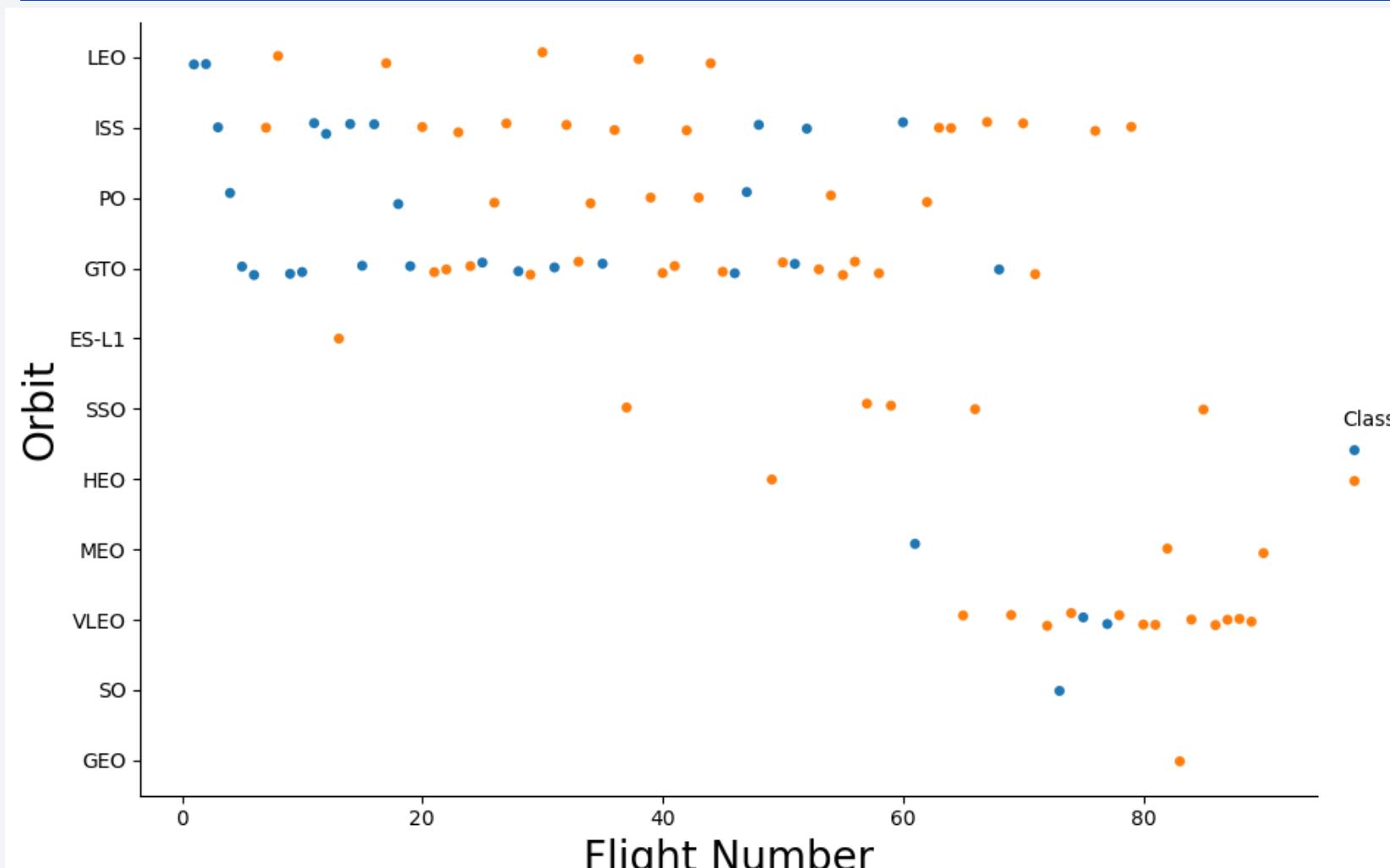
Payload vs. Launch Site



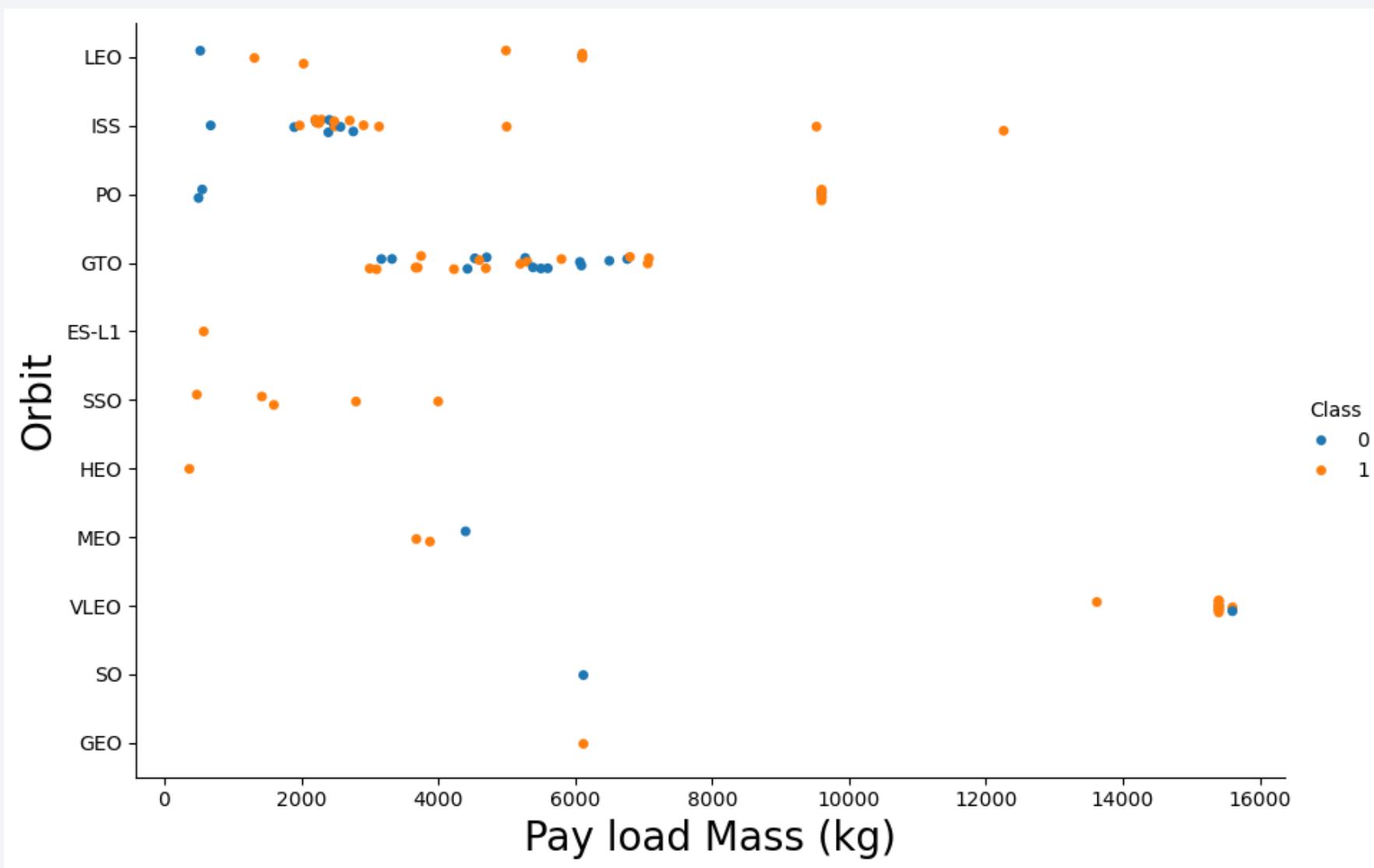
Success Rate vs. Orbit Type



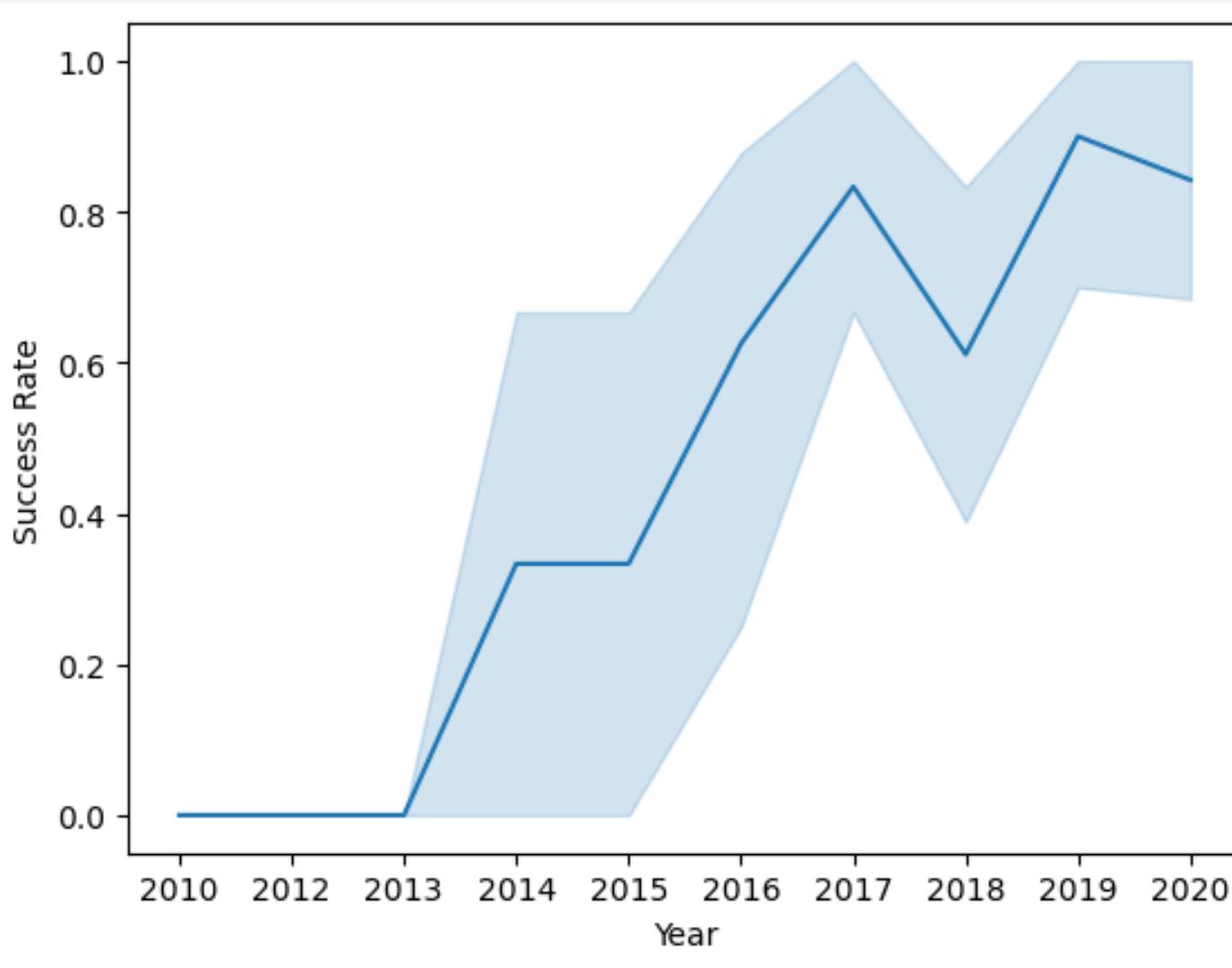
Flight Number vs. Orbit Type



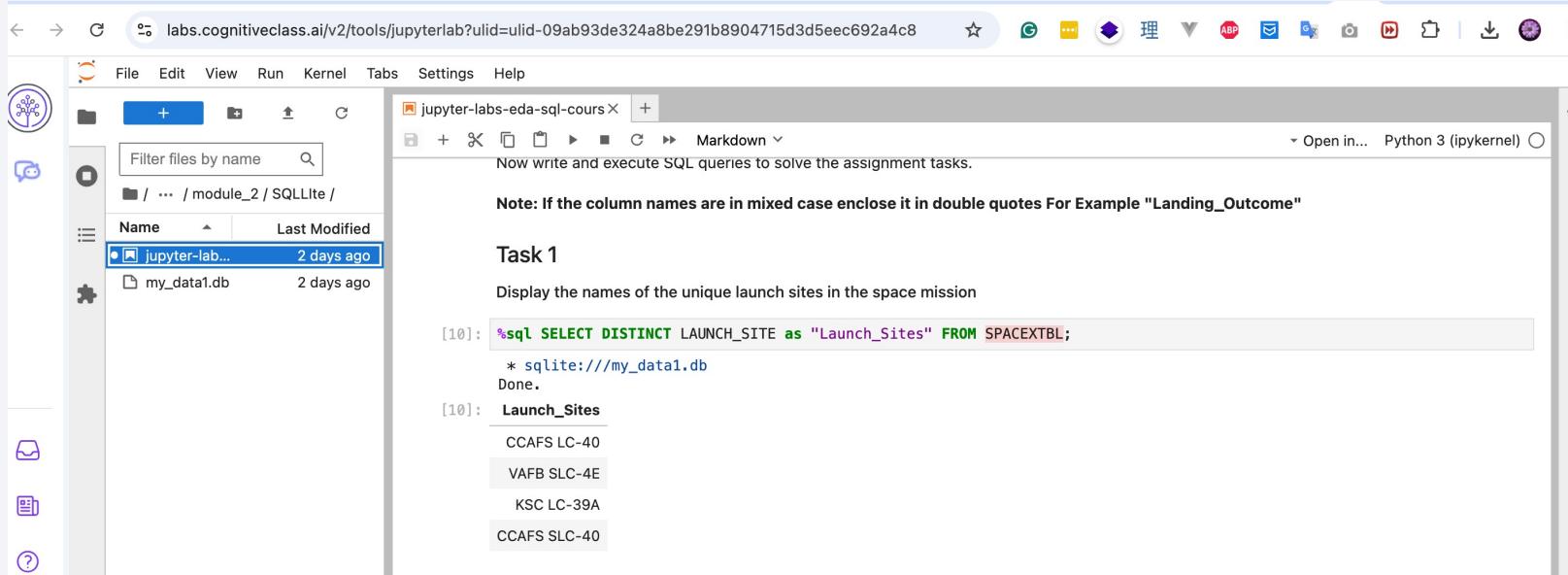
Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names



The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** labs.cognitiveclass.ai/v2/tools/jupyterlab?ulid=ulid-09ab93de324a8be291b8904715d3d5eec692a4c8
- File Menu:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Toolbar:** Includes icons for back, forward, search, and various file operations.
- Left Sidebar:** Shows a file tree with a folder named "module_2 / SQLite /". Inside, there are two files: "jupyter-lab..." (modified 2 days ago) and "my_data1.db" (modified 2 days ago).
- Central Area:**
 - A tab titled "jupyter-labs-eda-sql-cours" is active.
 - The content area displays a Markdown cell with the instruction: "Now write and execute SQL queries to solve the assignment tasks."
 - A note below it says: "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"
 - An input cell [10]:

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```
 - The output cell [10] shows the results:

```
* sqlite:///my_data1.db
Done.
[10]: Launch_Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
```

Launch Site Names Begin with 'CCA'

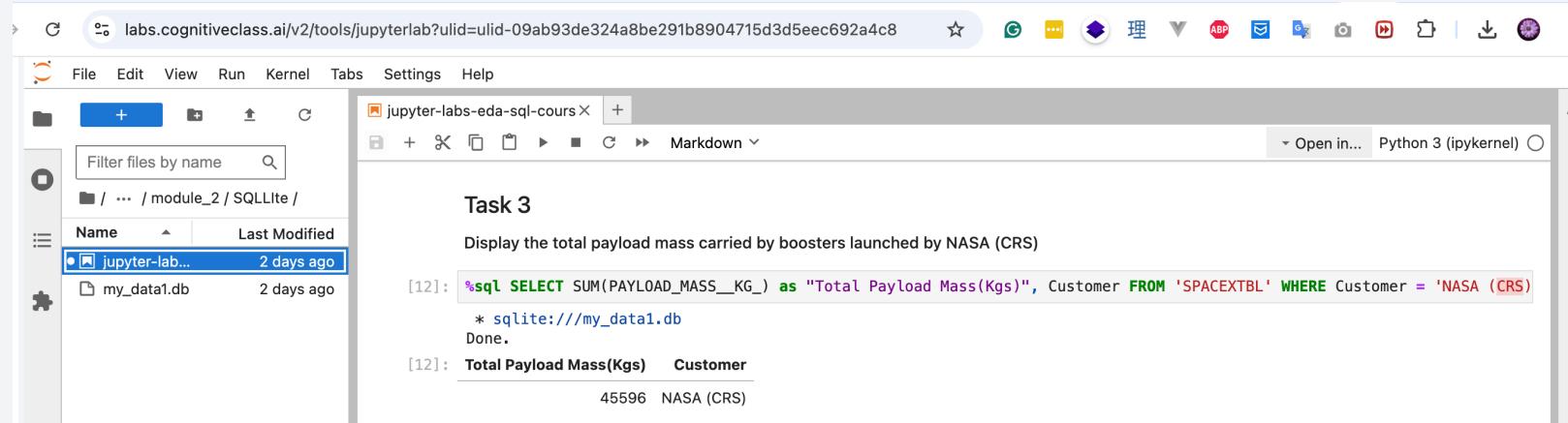
The screenshot shows a Jupyter Notebook interface with a sidebar containing file management icons and a list of files. The main area displays a task titled "Task 2" which asks to "Display 5 records where launch sites begin with the string 'CCA'". A SQL query is run in cell [11]:

```
%sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

The output shows five rows of data from the SPACEXTBL table:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outco
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachu
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 (parachu
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No atten
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No atten
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No atten

Total Payload Mass



The screenshot shows a Jupyter Notebook interface with the following details:

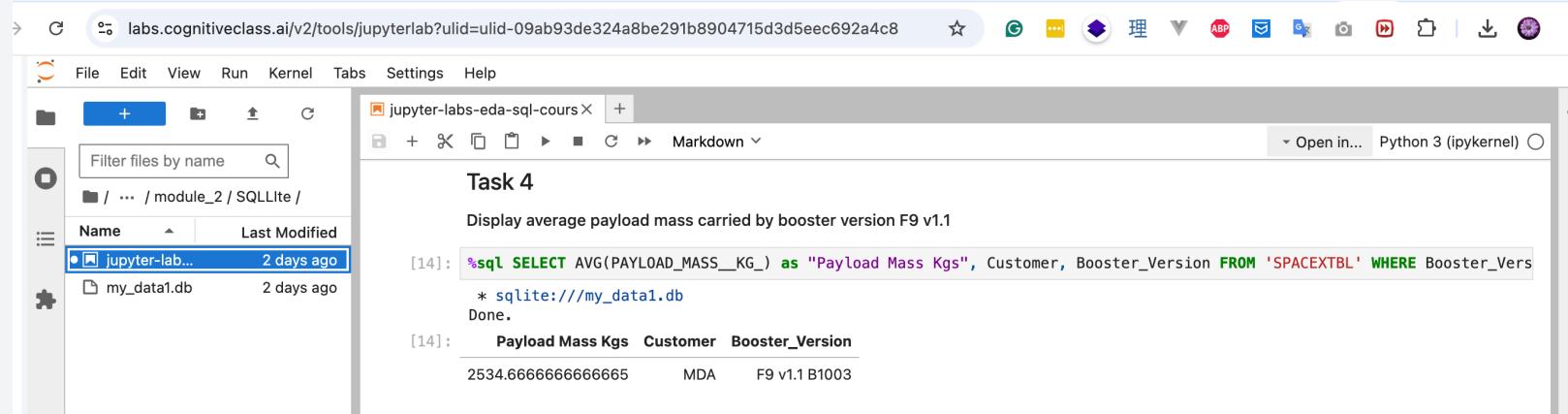
- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Toolbar:** Includes icons for file operations like New, Open, Save, and a search bar labeled "Filter files by name".
- File Explorer:** Shows a directory structure under "/ ... / module_2 / SQLlite /". Files listed include "jupyter-lab..." (modified 2 days ago) and "my_data1.db" (modified 2 days ago).
- Code Cell:** Task 3: Display the total payload mass carried by boosters launched by NASA (CRS). The code cell contains:

```
[12]: %sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)'  
* sqlite:///my_data1.db  
Done.
```

[12]: Total Payload Mass(Kgs) Customer

Total Payload Mass(Kgs)	Customer
45596	NASA (CRS)
- Output:** The output of the code cell is a table showing one row: 45596 NASA (CRS).

Average Payload Mass by F9 v1.1



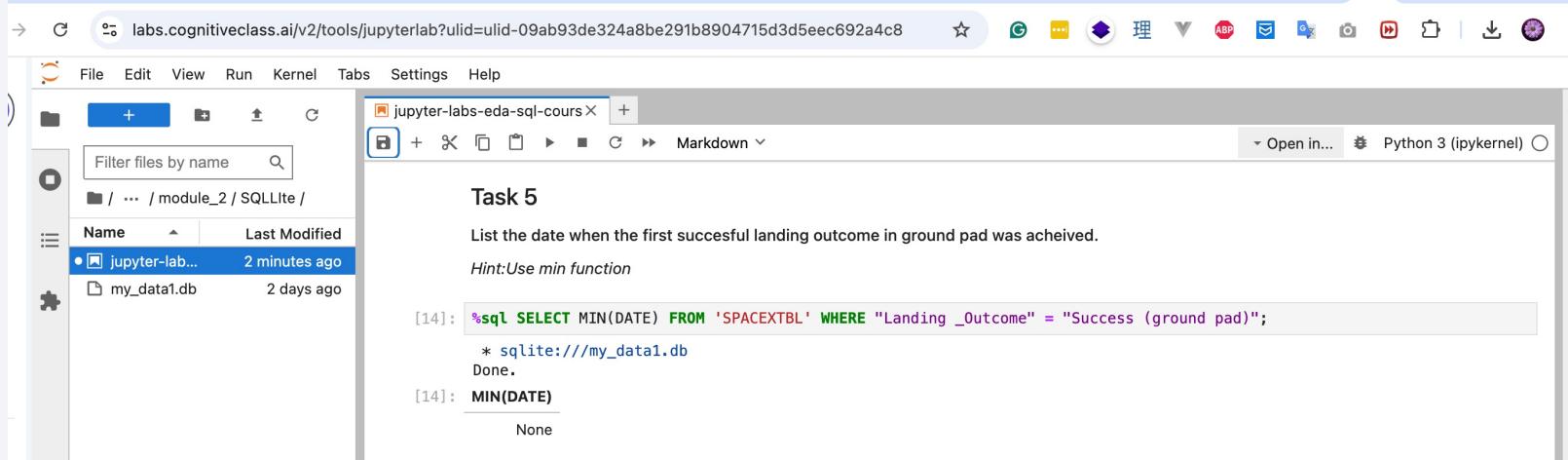
The screenshot shows a Jupyter Notebook interface with a sidebar containing file navigation and a list of files. The main area displays a code cell and its output.

Task 4
Display average payload mass carried by booster version F9 v1.1

```
[14]: %sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Vers  
* sqlite:///my_data1.db  
Done.
```

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

First Successful Ground Landing Date



The screenshot shows a Jupyter Notebook interface with a sidebar containing a file browser and a main notebook area.

File Browser: Shows a directory structure under "module_2 / SQLite /". The "jupyter-lab..." file is selected, and "my_data1.db" is also listed.

Notebook Area:

- Task 5:** A task description: "List the date when the first successful landing outcome in ground pad was achieved." followed by a hint: "Hint: Use min function".
- Code Cell 1:** SQL query:

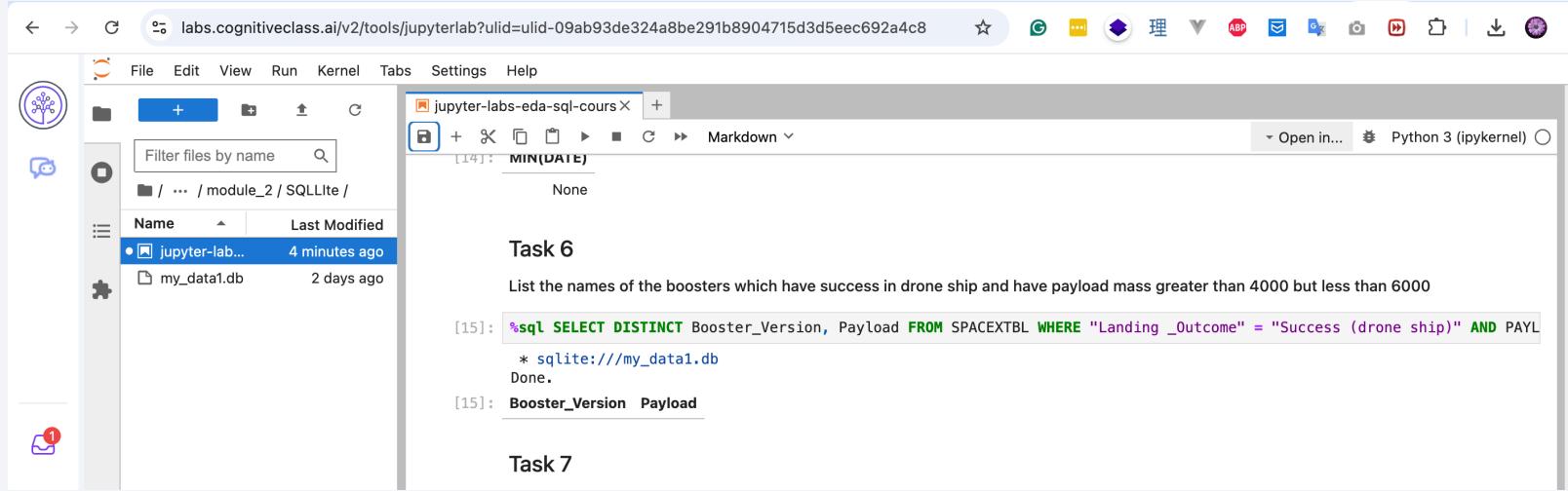
```
%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
```
- Output 1:** Result of the query:

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

 and "Done."
- Code Cell 2:** SQL query:

```
MIN(DATE)
```
- Output 2:** Result of the query: "None"

Successful Drone Ship Landing with Payload between 4000 and 6000



The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Toolbar:** Includes icons for file operations like New, Open, Save, and a search bar.
- Left Sidebar:** Shows a file tree with a folder named "module_2 / SQLLite/" containing "jupyter-lab..." (4 minutes ago) and "my_data1.db" (2 days ago). A red notification icon with the number "1" is visible.
- Current Tab:** "jupyter-labs-eda-sql-cours x" (Python 3 (ipykernel))
- Content Area:**
 - Task 6:** A question asking for the names of boosters with success in drone ship landing and payload mass greater than 4000 but less than 6000. Below it is a code cell starting with "%sql SELECT DISTINCT".
 - Code Cell Output:** Shows the command "%sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)" AND PAYL" followed by the response "* sqlite:///my_data1.db" and "Done."
 - Task 7:** A question asking for the Booster_Version and Payload.

Total Number of Successful and Failure Mission Outcomes

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Toolbar:** Includes icons for file operations like New, Open, Save, and a search bar.
- File Explorer:** Shows a directory structure under /.../module_2/SQlite/. It contains two items: "jupyter-lab..." (modified 5 minutes ago) and "my_data1.db" (modified 2 days ago).
- Task 7:** A section titled "Task 7" with the instruction "List the total number of successful and failure mission outcomes".
- Code Cell [16]:** Contains the following SQL query:

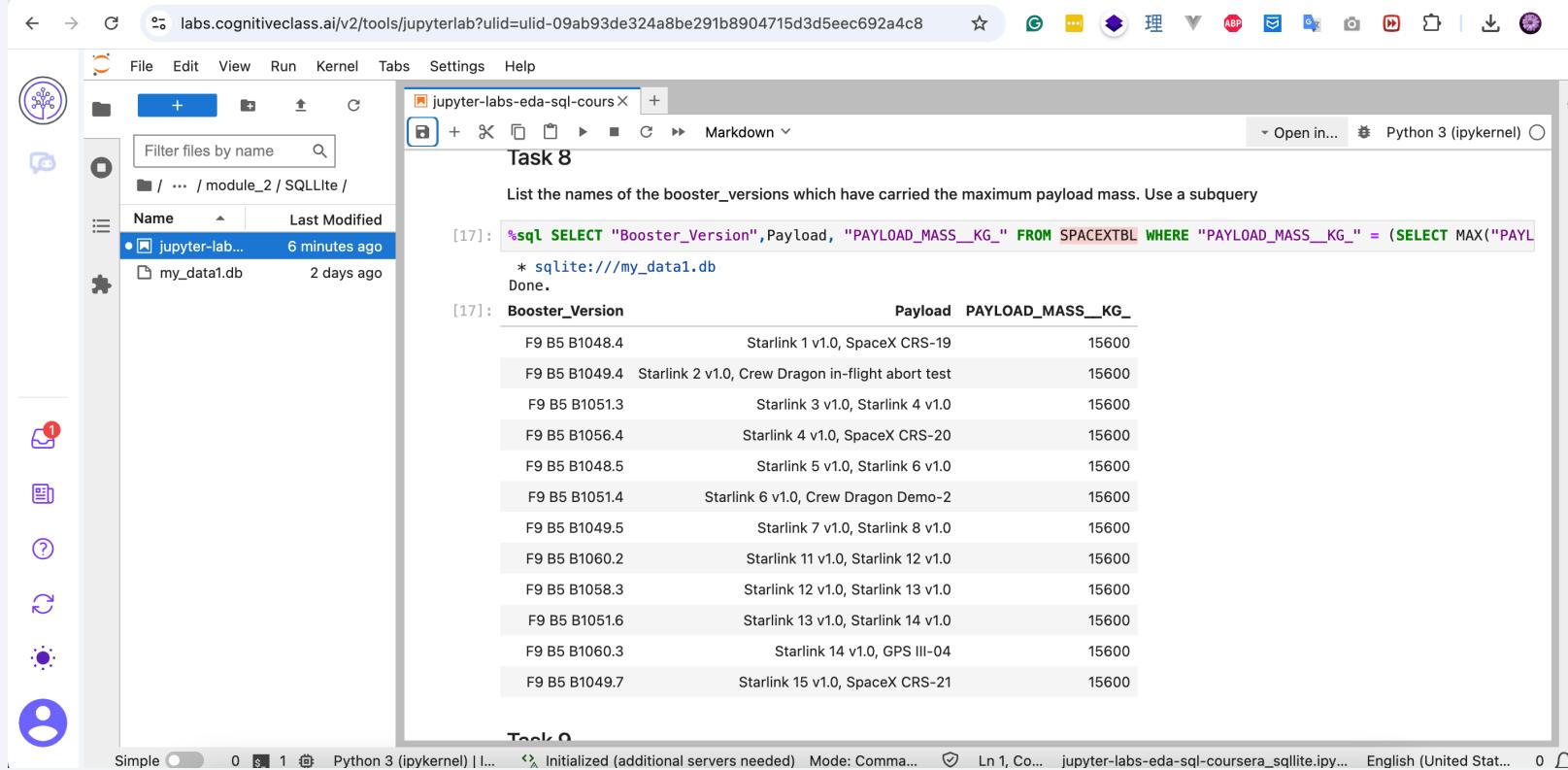
```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";  
* sqlite:///my_data1.db
```

Output: Done.
- Table Output:** A table showing the count of missions by outcome. The data is as follows:

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

- Task 8:** A section titled "Task 8".

Boosters Carried Maximum Payload



The screenshot shows a Jupyter Notebook interface with the following details:

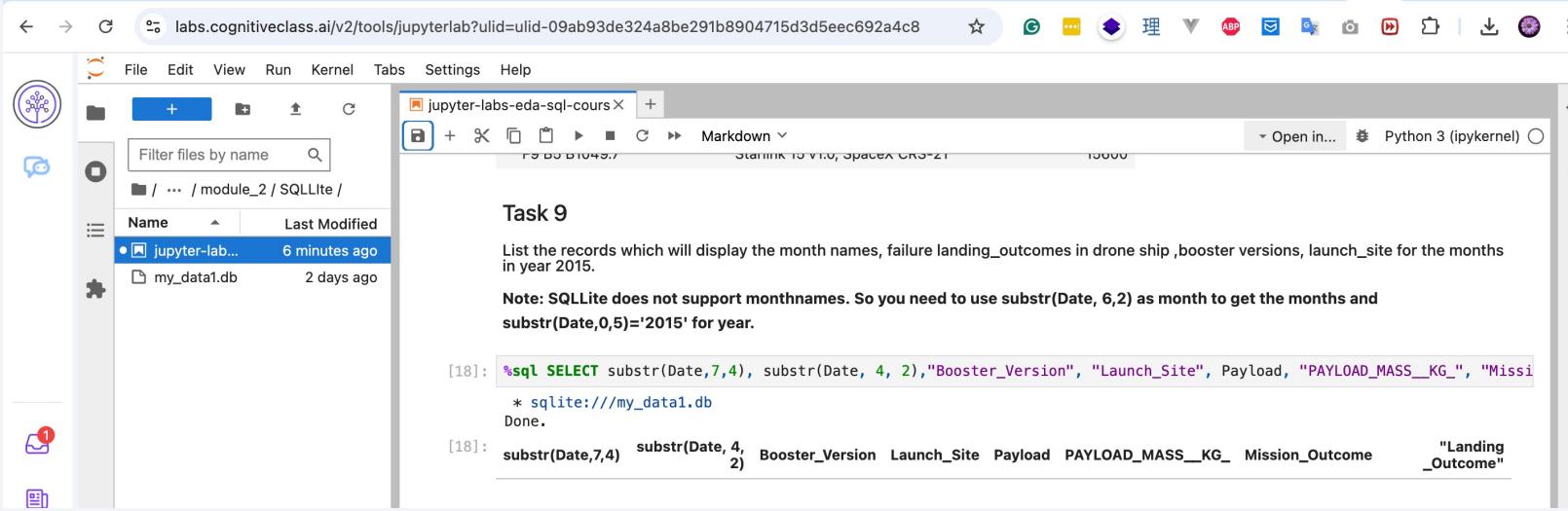
- Header:** labs.cognitiveclass.ai/v2/tools/jupyterlab?ulid=ulid-09ab93de324a8be291b8904715d3d5eec692a4c8
- Toolbar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- File Explorer:** Shows a folder named "jupyter-labs-eda-sql-cours" containing "my_data1.db" (modified 2 days ago).
- Code Cell:** Task 8: List the names of the booster_versions which have carried the maximum payload mass. Use a subquery.

```
[17]: %sql SELECT "Booster_Version", Payload, "PAYLOAD_MASS_KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS_KG_" = (SELECT MAX("PAYL"))
```
- Output:** A table showing the results of the query. The columns are Booster_Version, Payload, and PAYLOAD_MASS_KG_. All entries show a value of 15600.

Booster_Version	Payload	PAYLOAD_MASS_KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

- Bottom Status Bar:** Simple, Python 3 (ipykernel) | I..., Initialized (additional servers needed), Mode: Comma..., Ln 1, Co..., jupyter-labs-eda-sql-coursera_sqlite.ip..., English (United Stat..., 0

2015 Launch Records



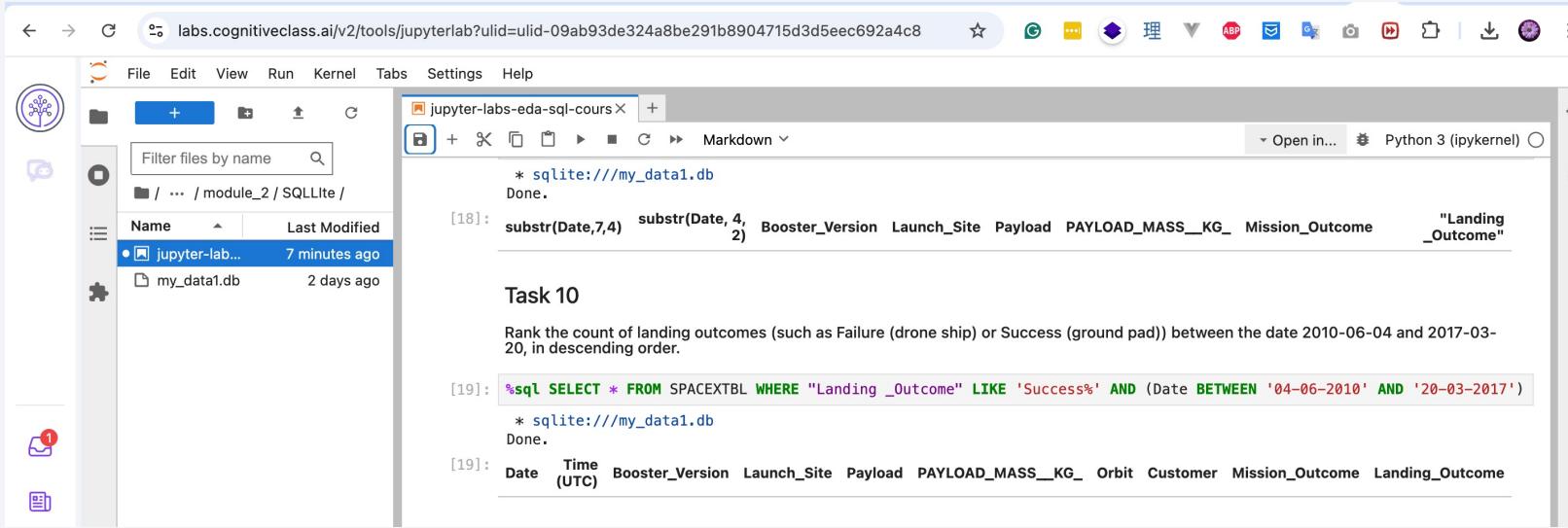
The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** labs.cognitiveclass.ai/v2/tools/jupyterlab?ulid=ulid-09ab93de324a8be291b8904715d3d5eec692a4c8
- File Menu:** File, Edit, View, Run, Kernel, Tabs, Settings, Help
- Toolbar:** Includes icons for back, forward, search, and various notebook operations.
- Left Sidebar:** Shows a file tree with a folder named "module_2 / SQLite /". Inside, there are two files: "jupyter-lab..." (modified 6 minutes ago) and "my_data1.db" (modified 2 days ago). A red notification badge with the number "1" is visible on the sidebar.
- Central Notebook Area:**
 - Title:** jupyter-labs-eda-sql-cours X
 - Kernel:** Python 3 (ipykernel)
 - Section:** Task 9
 - Description:** 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:** SQLite 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.
 - Code:**

```
[18]: %sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome", "Landing_Outcome"
* sqlite:///my_data1.db
Done.
```

```
[18]: substr(Date,7,4)  substr(Date, 4, 2)  Booster_Version  Launch_Site  Payload  PAYLOAD_MASS__KG_  Mission_Outcome  Landing_Outcome
```

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20



The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, View, Run, Kernel, Tabs, Settings, Help.
- Toolbar:** Includes icons for file operations like New, Open, Save, and a search bar labeled "Filter files by name".
- File Explorer:** Shows a folder structure with "jupyter-labs-eda-sql-cours" containing "my_data1.db" (modified 2 days ago) and "jupyter-lab..." (modified 7 minutes ago).
- Code Cell [18]:** Contains the following SQL query:

```
* sqlite:///my_data1.db
Done.
[18]: substr(Date,7,4)  substr(Date, 4, 2)  Booster_Version  Launch_Site  Payload  PAYLOAD_MASS__KG_  Mission_Outcome  "Landing
 _Outcome"
```
- Text Cell [19]:** Labeled "Task 10" and contains the instruction: "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."
- Code Cell [19]:** Contains the following SQL query:

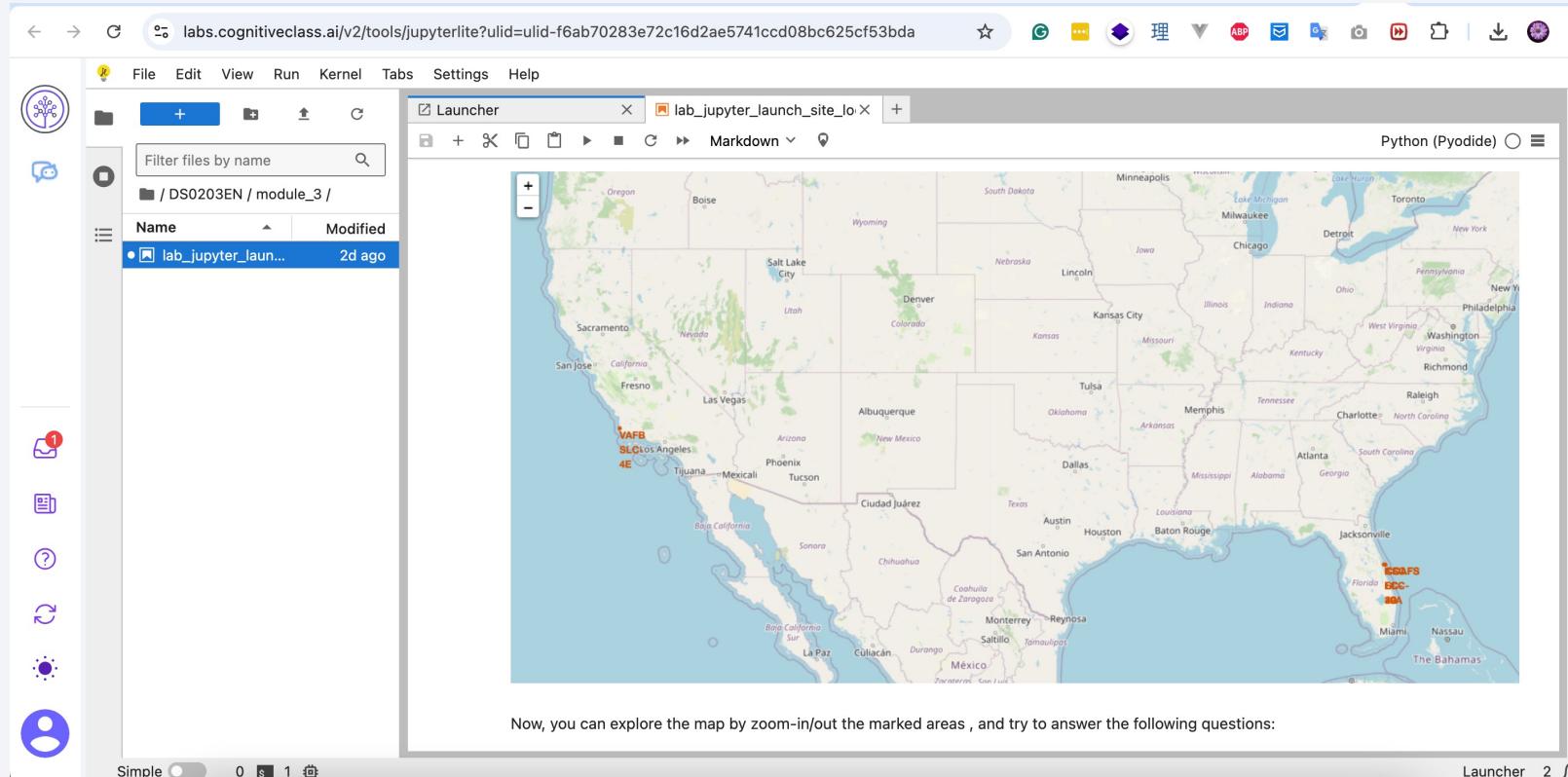
```
%sql SELECT * FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017')
* sqlite:///my_data1.db
Done.
[19]: Date  Time  Booster_Version  Launch_Site  Payload  PAYLOAD_MASS__KG_  Orbit  Customer  Mission_Outcome  Landing_Outcome
```

The background of the slide is a photograph taken from space at night. It shows the curvature of the Earth against a dark blue-black void of space. City lights are visible as numerous small white and yellow dots, primarily concentrated in the lower right quadrant where the United States appears. In the upper right, the green and yellow glow of the aurora borealis is visible. The atmosphere of the Earth is thin and hazy, appearing as a light blue band near the horizon.

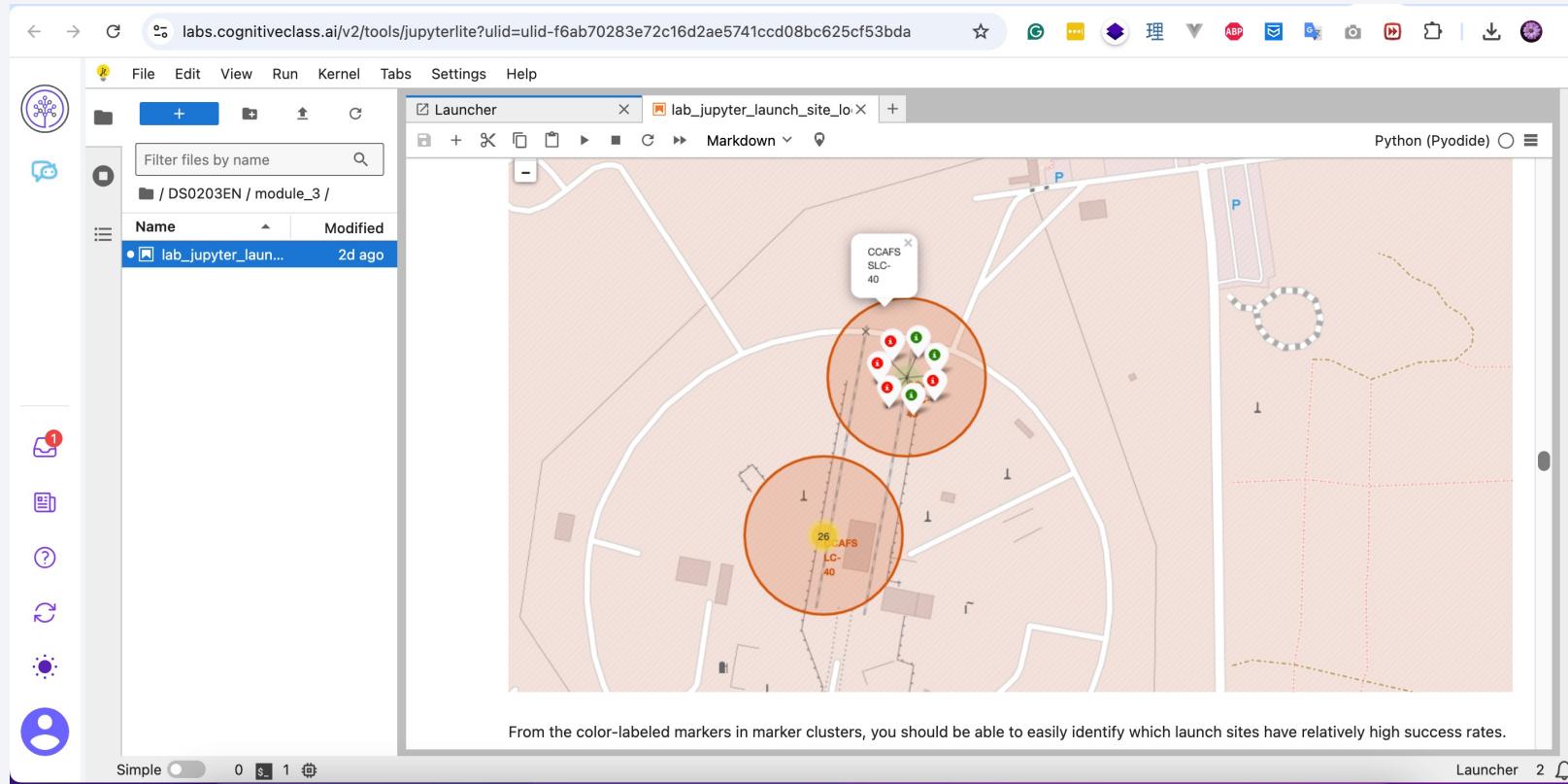
Section 3

Launch Sites Proximities Analysis

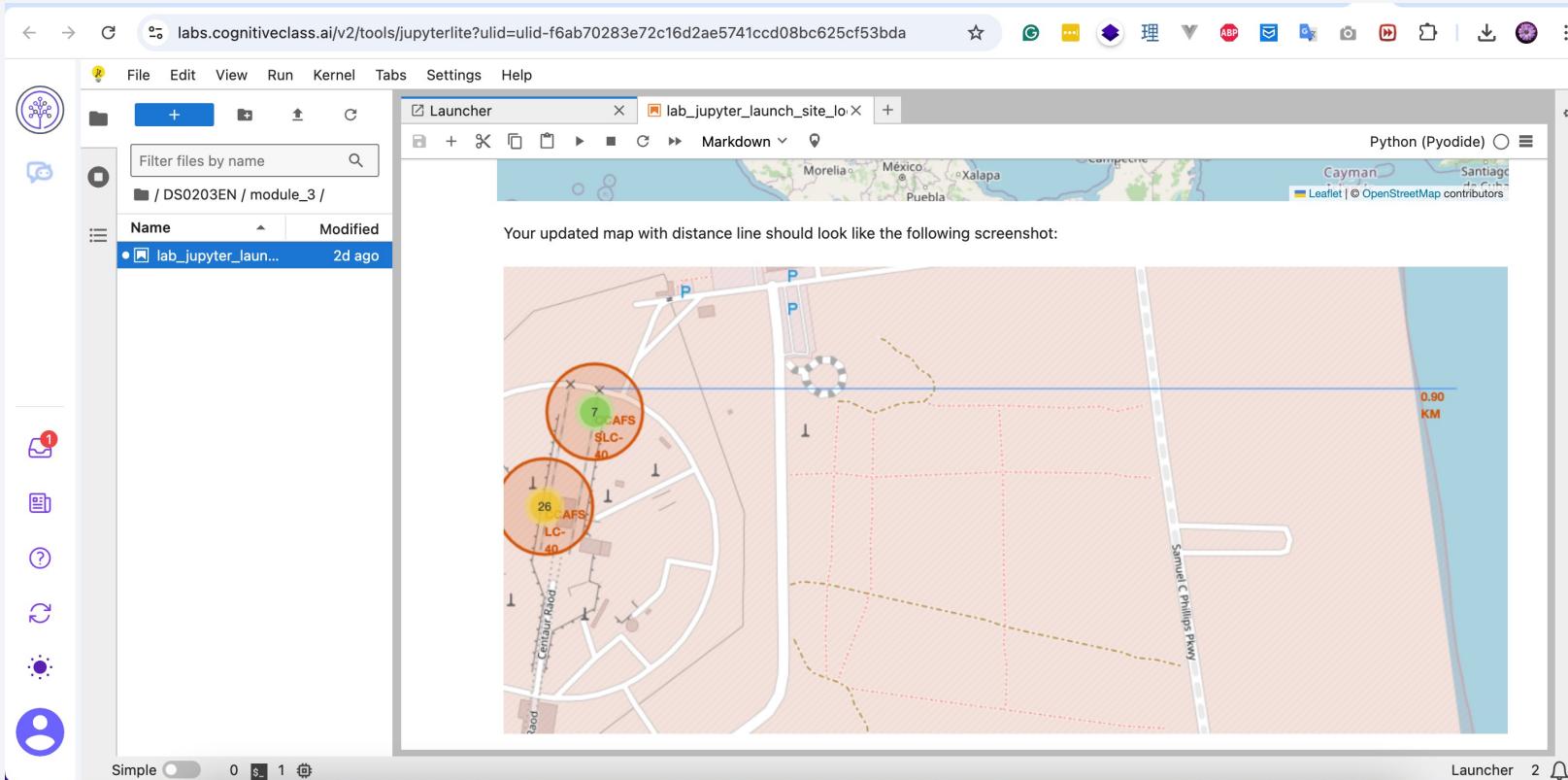
<Folium Map Screenshot 1>



<Folium Map Screenshot 2>

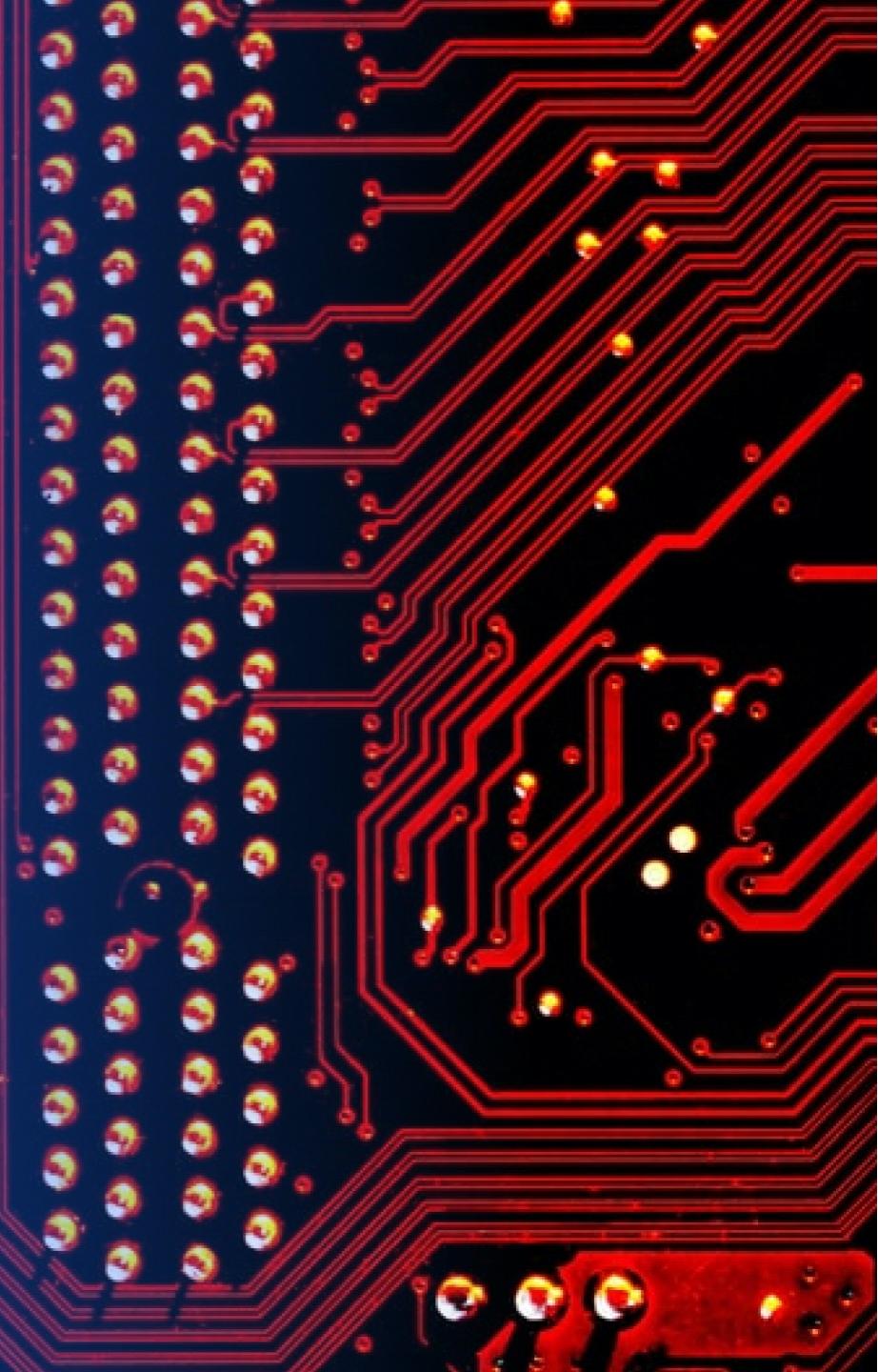


<Folium Map Screenshot 3>



Section 4

Build a Dashboard with Plotly Dash



<Dashboard Screenshot 1>

- Replace <Dashboard screenshot 1> title with an appropriate title
- Show the screenshot of launch success count for all sites, in a piechart
- Explain the important elements and findings on the screenshot

<Dashboard Screenshot 2>

- Replace <Dashboard screenshot 2> title with an appropriate title
- Show the screenshot of the piechart for the launch site with highest launch success ratio
- Explain the important elements and findings on the screenshot

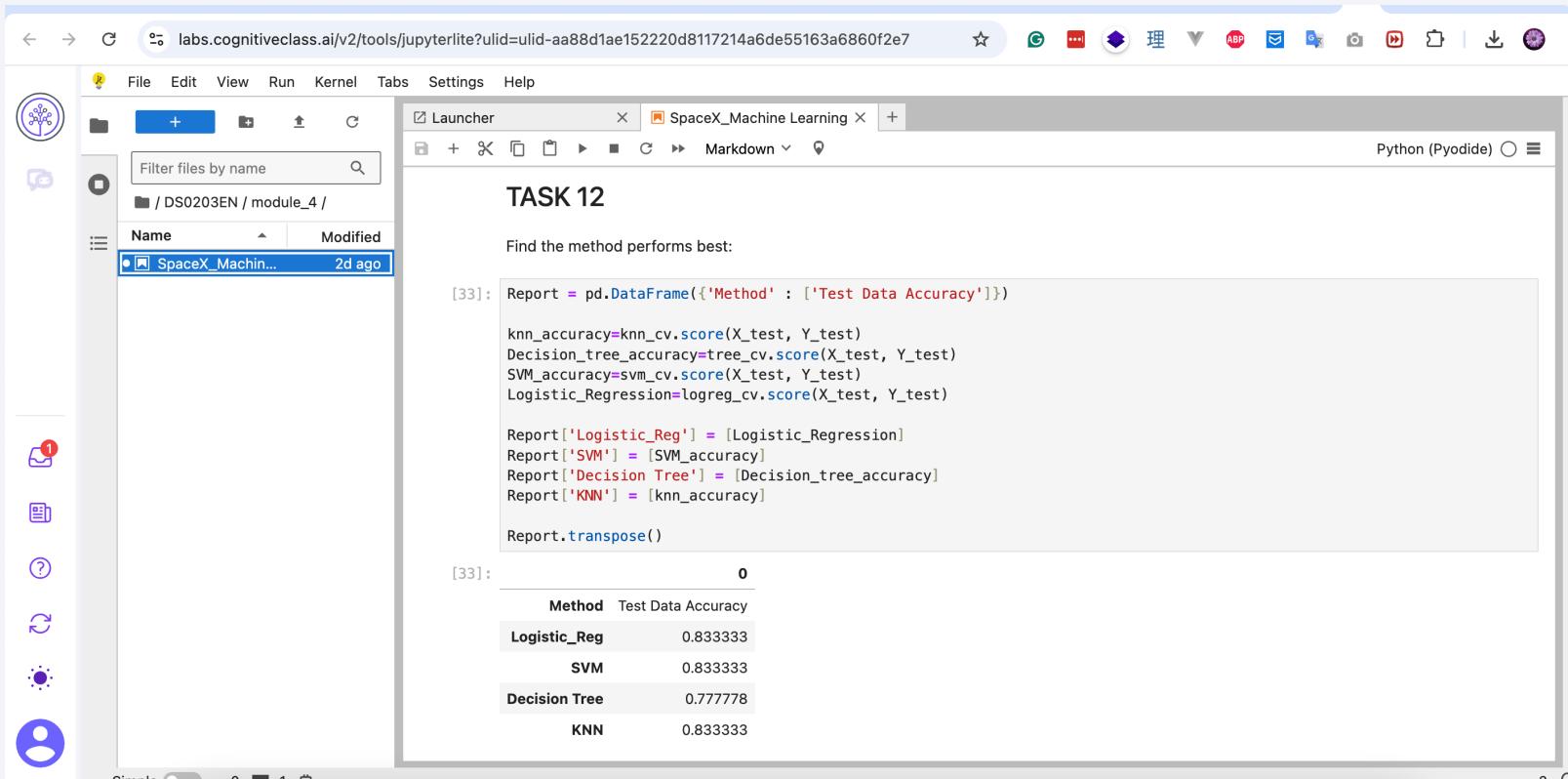
<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title
- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider
- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



The screenshot shows a Jupyter Notebook interface with a sidebar containing icons for file operations, a launcher, and user information. The main area has tabs for 'Launcher' and 'SpaceX_Machine Learning'. The 'SpaceX_Machine Learning' tab is active and displays Python code in a code cell and its output in an HTML table.

TASK 12

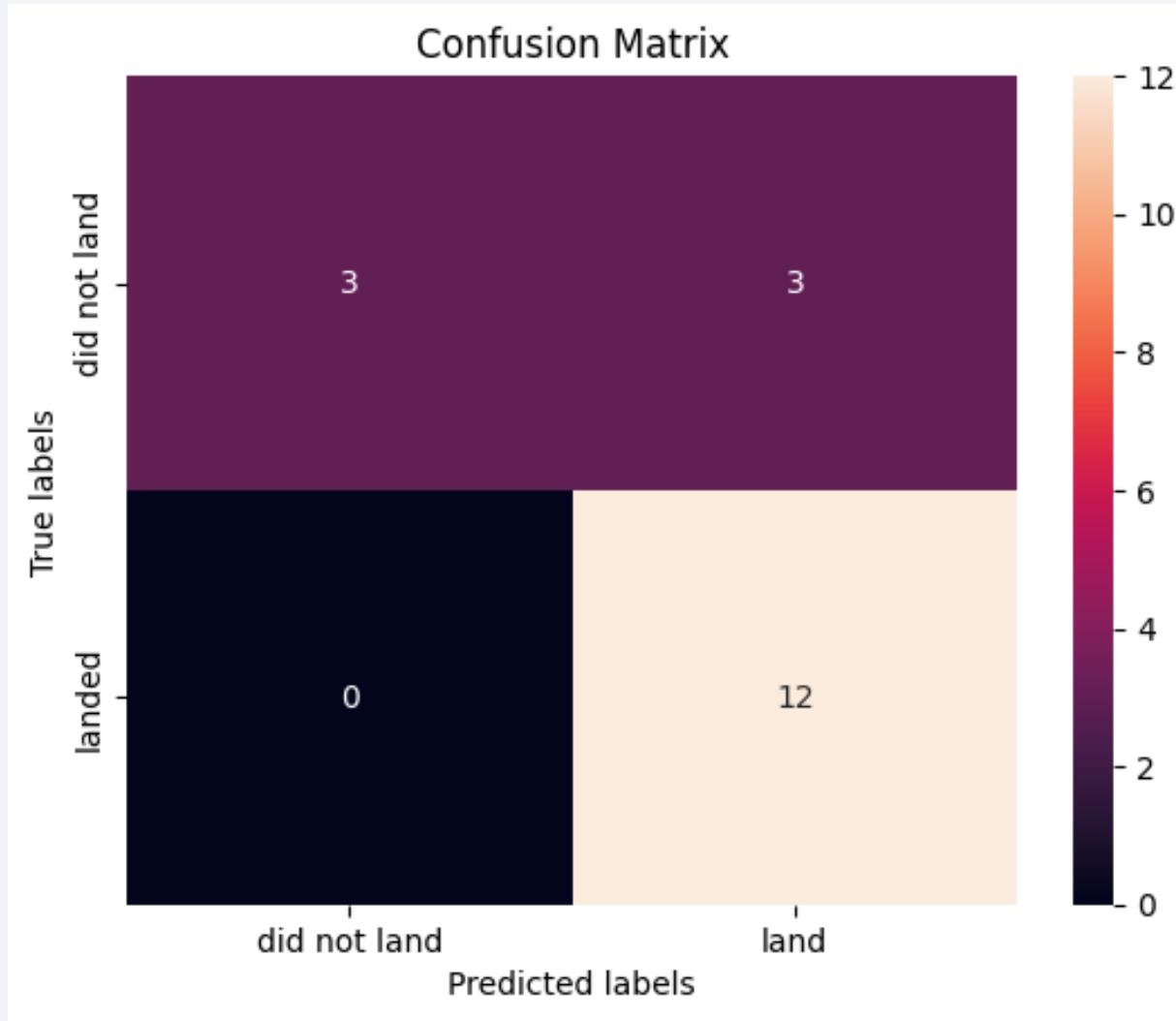
Find the method performs best:

```
[33]: Report = pd.DataFrame({'Method' : ['Test Data Accuracy']})  
  
knn_accuracy=knn_cv.score(X_test, Y_test)  
Decision_tree_accuracy=tree_cv.score(X_test, Y_test)  
SVM_accuracy=svm_cv.score(X_test, Y_test)  
Logistic_Regression=logreg_cv.score(X_test, Y_test)  
  
Report['Logistic_Reg'] = [Logistic_Regression]  
Report['SVM'] = [SVM_accuracy]  
Report['Decision Tree'] = [Decision_tree_accuracy]  
Report['KNN'] = [knn_accuracy]  
  
Report.transpose()
```

[33]:

Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.777778
KNN	0.833333

Confusion Matrix



Conclusions

- Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
- We can deduce that, as the flight number increases in each of the 3 launch sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80th flight
- If you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavy payload mass(greater than 10000).
- Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.
- LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS. However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here
- And finally the success rate since 2013 kept increasing till 2020

Thank you!

