



Winning Space Race with Data Science

Joseoh Okelly
July 22, 2024



Outline

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

Executive Summary

- Summary of methodologies
- Jupyter labs spacex data collection api
- Jupyter labs webscraping Falcon 9
- Labs Jupyter spacex data wrangling
- Jupyter labs eda sql coursera sqlite
- Eda dataviz using pandas and matplotlib
- SpaceX launch site locations Analysis
- Build Interactive Dashboard – spacex dashboard
- SpaceX Machine Learning Predictions

Summary of results

- EDA
- Visual Analysis
- Predictive Analysis

Introduction



- Project background and context

SpaceX advertises Falcon 9 rocket launches on its website with a 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

- Describe how data sets were collected.
- Data was collected using SpaceX Restful webservices, using Get request SpaceX api.

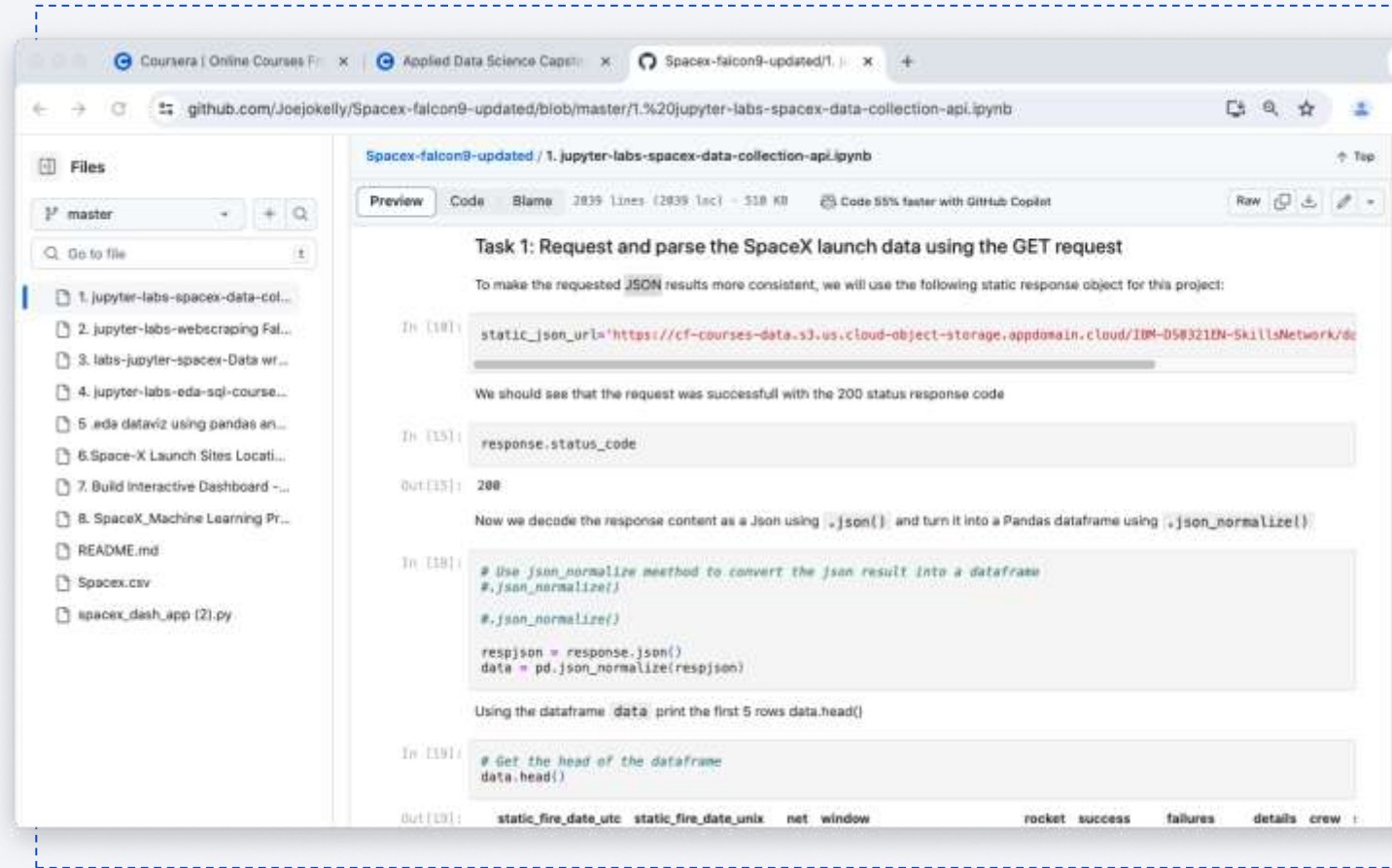
The request was later stored for processing using Pandas Dataframe

Data was collected using SpaceX Restful webservices, Get request SpaceX api.

- Web scraping performed to Falcon9 historical records.
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

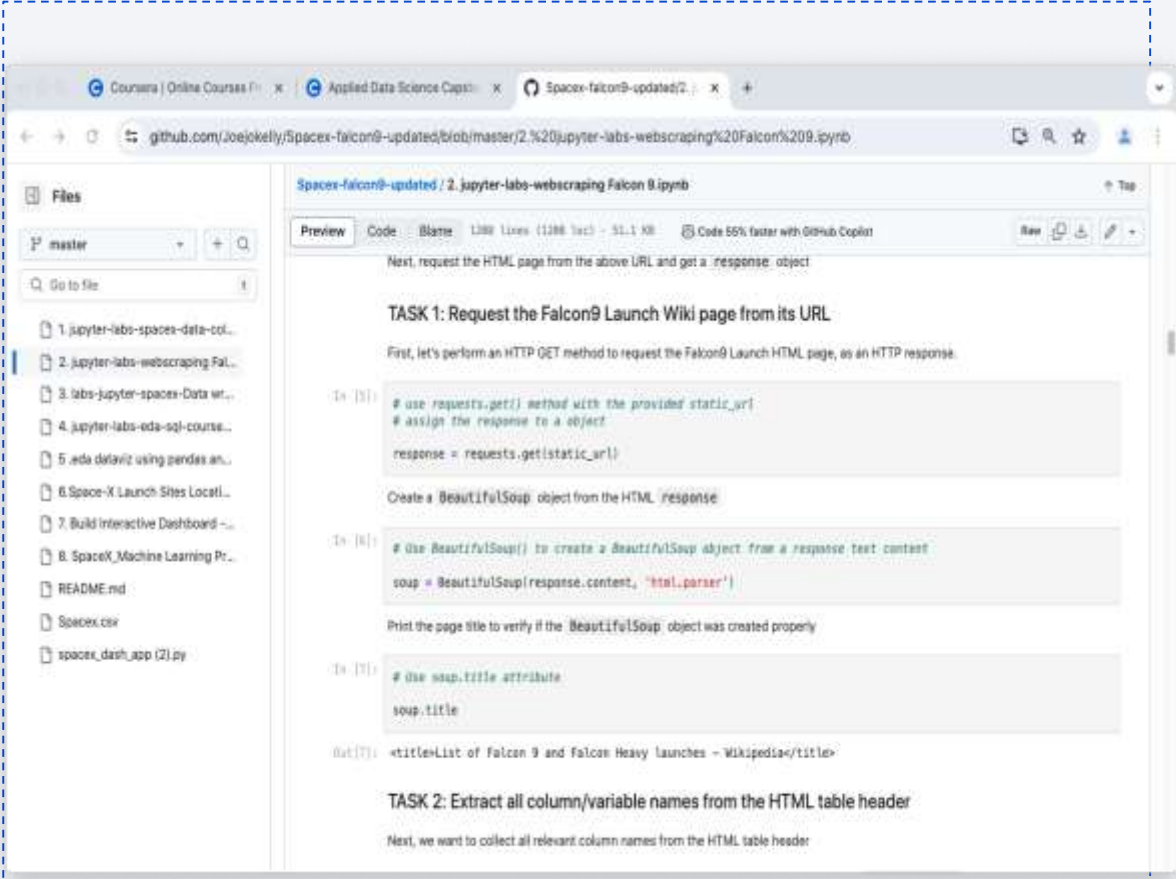
- Data collected using SpaceX Api (Restful call using get parameter)
- Get request is decoded by pandas data frame and it is stored for processing.
- <https://github.com/Joejokelly/Spacex-falcon9-updated/blob/master/1.%20jupyter-labs-spacex-data-collection-api.ipynb>



The screenshot shows a web browser displaying a Jupyter Notebook on GitHub. The browser tabs include 'Coursera | Online Courses', 'Applied Data Science Capstone', and 'Spacex-falcon9-updated/1'. The address bar shows the GitHub repository URL: `github.com/Joejokelly/Spacex-falcon9-updated/blob/master/1.%20jupyter-labs-spacex-data-collection-api.ipynb`. The notebook interface has a 'Files' sidebar on the left with a file list including `1. jupyter-labs-spacex-data-col...`, `2. jupyter-labs-webscraping Fal...`, `3. labs-jupyter-spacex-Data wr...`, `4. jupyter-labs-eda-sql-course...`, `5. eda dataviz using pandas an...`, `6.Space-X Launch Sites Locati...`, `7. Build Interactive Dashboard -...`, `8. SpaceX_Machine Learning Pr...`, `README.md`, `Spacex.csv`, and `spacex_dash_app (2).py`. The main area shows the notebook content for '1. jupyter-labs-spacex-data-collection-api.ipynb'. It includes a 'Task 1: Request and parse the SpaceX launch data using the GET request' section. The text explains that a static JSON URL is used for consistency: `static_json_url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/example_json_1.json'`. It shows a successful GET request with a status code of 200. The response is then decoded using `response.json()` and converted into a Pandas DataFrame using `pd.json_normalize(response.json())`. The final output shows the first 5 rows of the DataFrame, with columns: `static_fire_date_utc`, `static_fire_date_unix`, `net`, `window`, `rocket`, `success`, `failures`, `details`, and `crew`.

Data Collection - Scraping

- Webscraping is performed using historical launch records using Beautiful utility, next the Data frame is created by passing the Launch HTML file.
- Github url :
- <https://github.com/Joejokelly/Spacex-falcon9-updated/blob/master/2.%20jupyter-labs-webscraping%20Falcon%209.ipynb> purpose



The screenshot shows a web browser displaying a GitHub repository page. The URL in the address bar is `github.com/Joejokelly/Spacex-falcon9-updated/blob/master/2.%20jupyter-labs-webscraping%20Falcon%209.ipynb`. The page shows a file explorer on the left with a list of files, including `1. jupyter-labs-spaces-data-col...`, `2. jupyter-labs-webscraping Fal...`, `3. labs-jupyter-spaces-Data wr...`, `4. jupyter-labs-eda-sql-course...`, `5. eda dataviz using pandas an...`, `6. Space-X Launch Sites Locati...`, `7. Build Interactive Dashboard ...`, `8. SpaceX_Machine Learning Pr...`, `README.md`, `Spacex.csv`, and `spacex_dash_app (2).py`. The main content area shows the Jupyter notebook file `2. jupyter-labs-webscraping Falcon 9.ipynb` with a preview of the code. The code includes comments and Python snippets for requesting the Falcon9 Launch Wiki page and parsing the HTML response using BeautifulSoup.

```
Next, request the HTML page from the above URL and get a .response object
```

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

```
In [3]: # use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

Create a BeautifulSoup object from the HTML .response

```
In [4]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, 'html.parser')
```

Print the page title to verify if the BeautifulSoup object was created properly

```
In [7]: # Use soup.title attribute
soup.title
```

```
Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

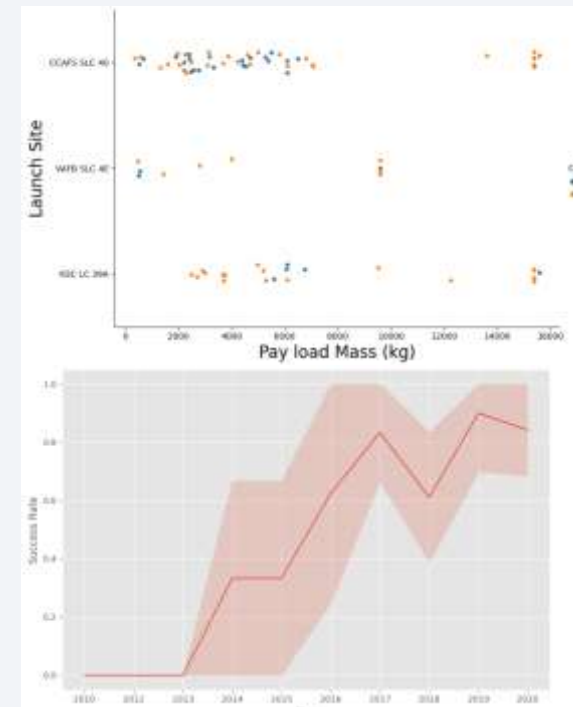
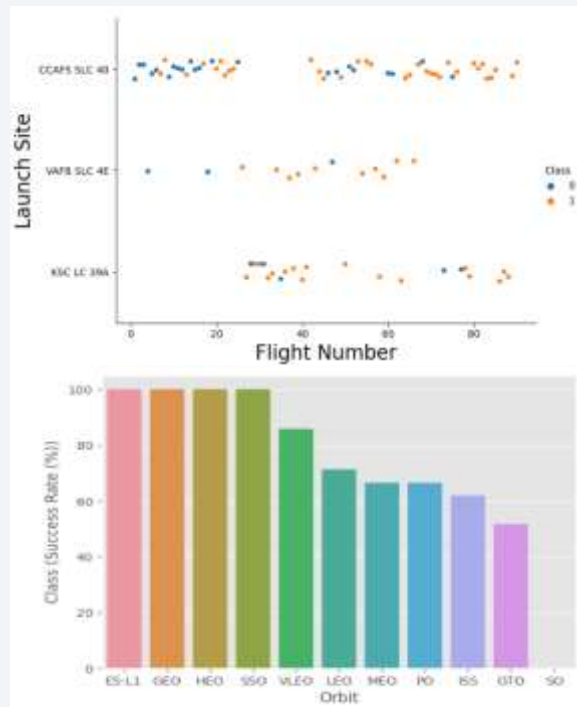
Next, we want to collect all relevant column names from the HTML table header

EDA with Data Visualization

- Data Analysis and Feature Engineering using Pandas and Matplotlib
 - Exploratory Data Analysis
 - Preparing Data Feature Engineering
- <https://github.com/Joejokelly/SpaceX-falcon9-updated/blob/master/5%20.eda%20dataviz%20using%20pandas%20and%20matplotlib%20.ipynb>

EDA with sql (contd)..<

EDA with Data Visualization (Plots Cont....)



EDA with SQL

- **Display the names of the unique launch sites in the space mission**
- **%sql** select distinct Launch_site as Launch_site from SPACEXTABLE;
- **Display 5 records where launch sites begin with the string 'CCA'**
- **%sql** select * from 'spacextbl' where Launch_site like 'CCA%' limit 5
- **Display the total payload mass carried by boosters launched by NASA (CRS)**
- *#%sql select sum(payload_mass_kg) as payload_mass from 'SPACEXTBL'* **%sql** SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload KG)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';

Sql contd..

- **Display average payload mass carried by booster version F9 v1.1**
- `%sql select avg(PAYLOAD__MASS__KG_) as "Average mass kg", customer Booster__Version FROM 'SPACEXTBL' WHERE Booster__Version LIKE 'F9 v1.1%';`
- **List the date when the first succesful landing outcome in ground pad was acheived.**
- `%sql SELECT min(date) FROM 'SPACEXTBL' WHERE "Landing__outcome" = "Success (ground pad)"`
- `(%sql SELECT DISTINCT Booster__Version, Payload FROM SPACEXTBL WHERE "Landing __Outcome" = "Success (drone ship)" AND PAYLOAD__MASS__KG_ > 4000 AND PAYLOAD__MASS__KG_ < 6000;`
- Github link :
- https://github.com/Joejokelly/Spacex-falcon9-updated/blob/master/4.%20jupyter-labs-eda-sql-coursera_sqlite.ipynb

Build an Interactive Map with Folium

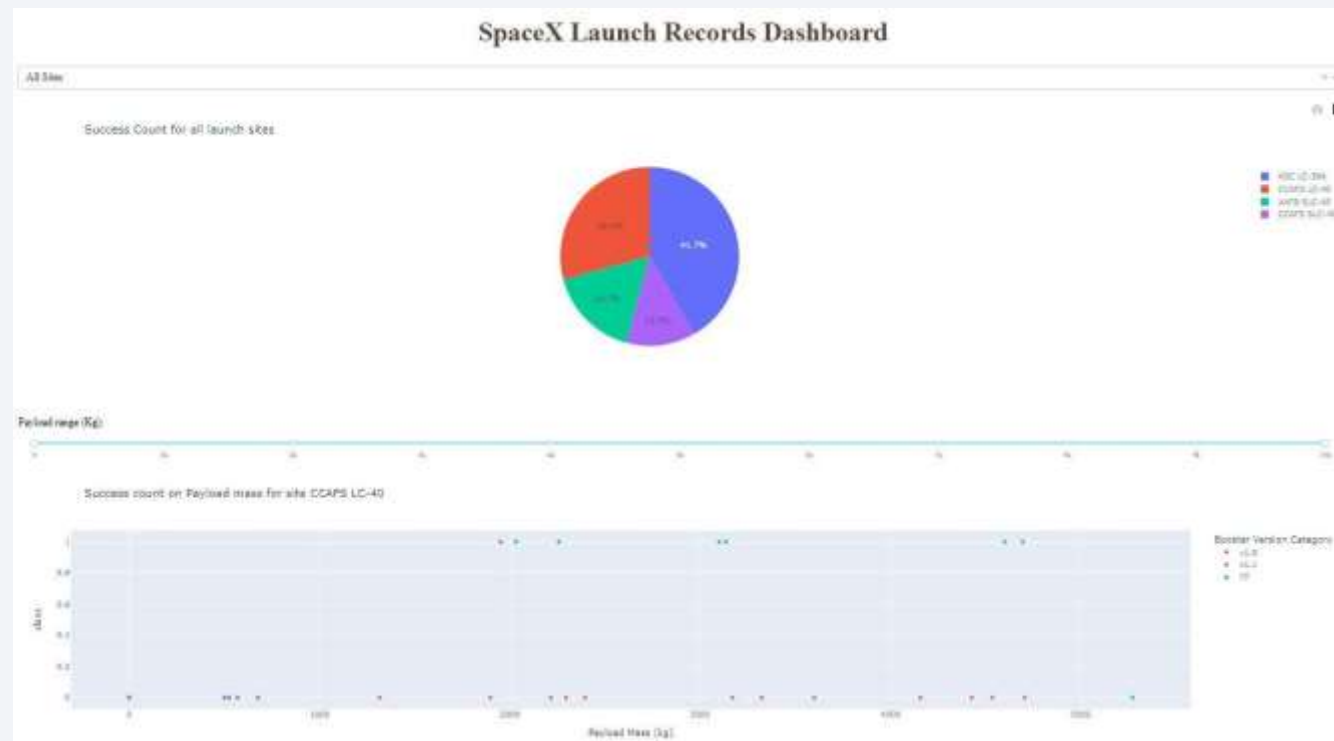
- Created folium map to mark launch sites, and markers, circles, lines to mark failure or success of launches for each site
- Created launch outcomes (failure = 0, success = 1)
- <https://github.com/Joejokelly/Spacex-falcon9-updated/blob/master/6.Space-X%20Launch%20Sites%20Locations%20Analysis%20with%20Folium-Interactive%20Visual%20Analytics.ipynb>

Build a Dashboard with Plotly Dash

- Build an interactive dashboard application with Plotly dash ;
 - Adding Launch Site Drop-down Input component
 - Adding call-back function success-pie-chart
 - Adding range slider to selected payload
 - Adding call back function success-payload-scaler –chart
-
- Github link :
 - https://github.com/Joekelly/Spacex-falcon9-updated/blob/master/7.%20%20Build%20Interactive%20Dashboard%20-%20spacex_dash_app.py

Launch Record Dashboard

SpaceX Dash App



Predictive Analysis (Classification)

- Summary of how model was built, evaluated and improved :
 - After initial processing, data was loaded to the model using Pandas Data frame, created Numpy Array, then used StandardScaler function, to standardize the data.
 - Split the resulting data into training data and test data, train_test_Split from sklearn.
- In order to find the best Machine Language
 - Created GridsearchCV object, after fitting the training data, into GridsearchCV, displayed the best parameter using data attributes.
 - Used method score to calculate accuracy of each model and plotted confusion matrix for teach test and predicted outcomes.

Github link :

- <https://github.com/Joejokelly/Spacex-falcon9-updated/blob/master/6.Space-X%20Launch%20Sites%20Locations%20Analysis%20with%20Folium-Interactive%20Visual%20Analytics.ipynb>

Predictive Analysis (Classification, contd)

- The table below shows the test accuracy score :

```
Out[68]:
```

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

- Github link :

<https://github.com/cgatama/SpaceX-Falcon-9-1st-stage-Success-Landing-Prediction/blob/main/8.%20SpaceX%20Machine%20Learning%20Prediction.ipynb>
<https://github.com/cgatama/SpaceX-Falcon-9-1st-stage-Success-Landing-Prediction/blob/main/8.%20SpaceX%20Machine%20Learning%20Prediction.ipynb>

Results

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

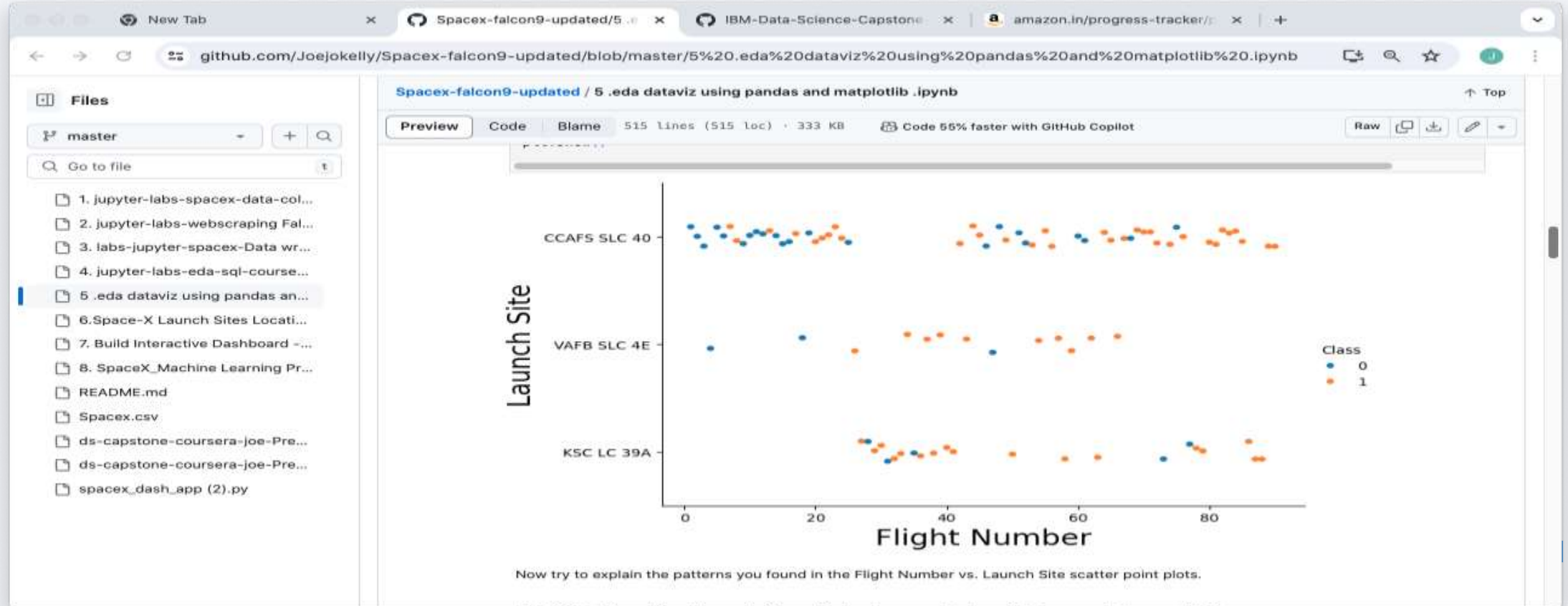
The background of the slide is an abstract composition. It features a solid blue area on the left side, which transitions into a dynamic pattern of diagonal streaks in shades of blue, red, and cyan on the right. A faint, light-blue grid or mesh pattern is overlaid across the entire image, particularly visible in the blue and cyan areas.

Section 2

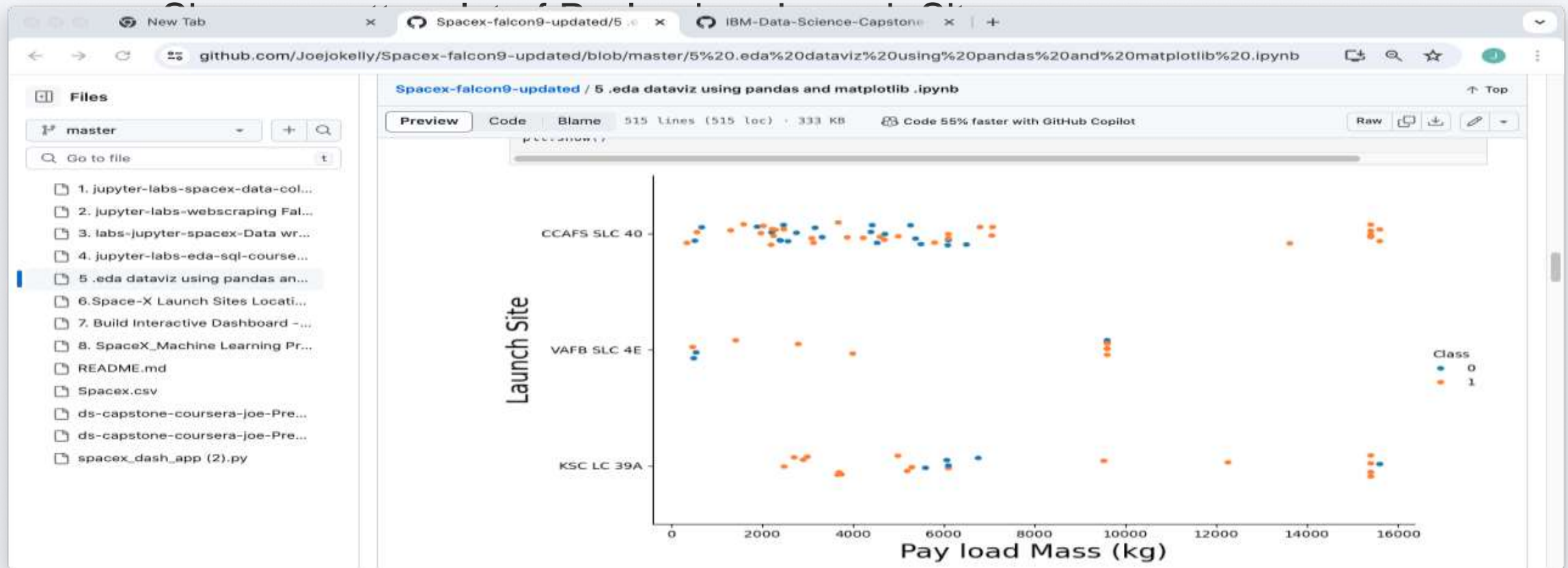
Insights drawn from EDA

Flight Number vs. Launch Site

- scatter plot of Flight Number vs. Launch Site

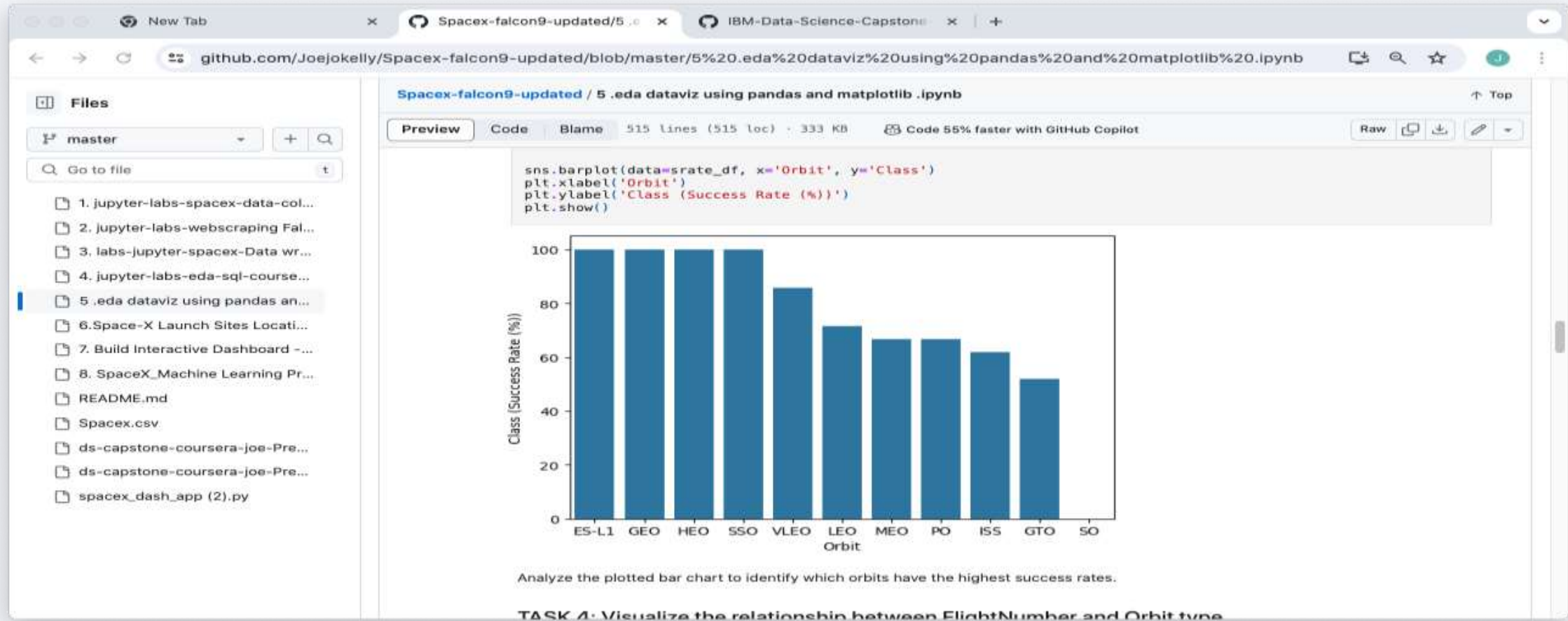


Payload vs. Launch Site



Payload vs Launch site scatter plot, there are no rocket launches for heavy payload > 10000

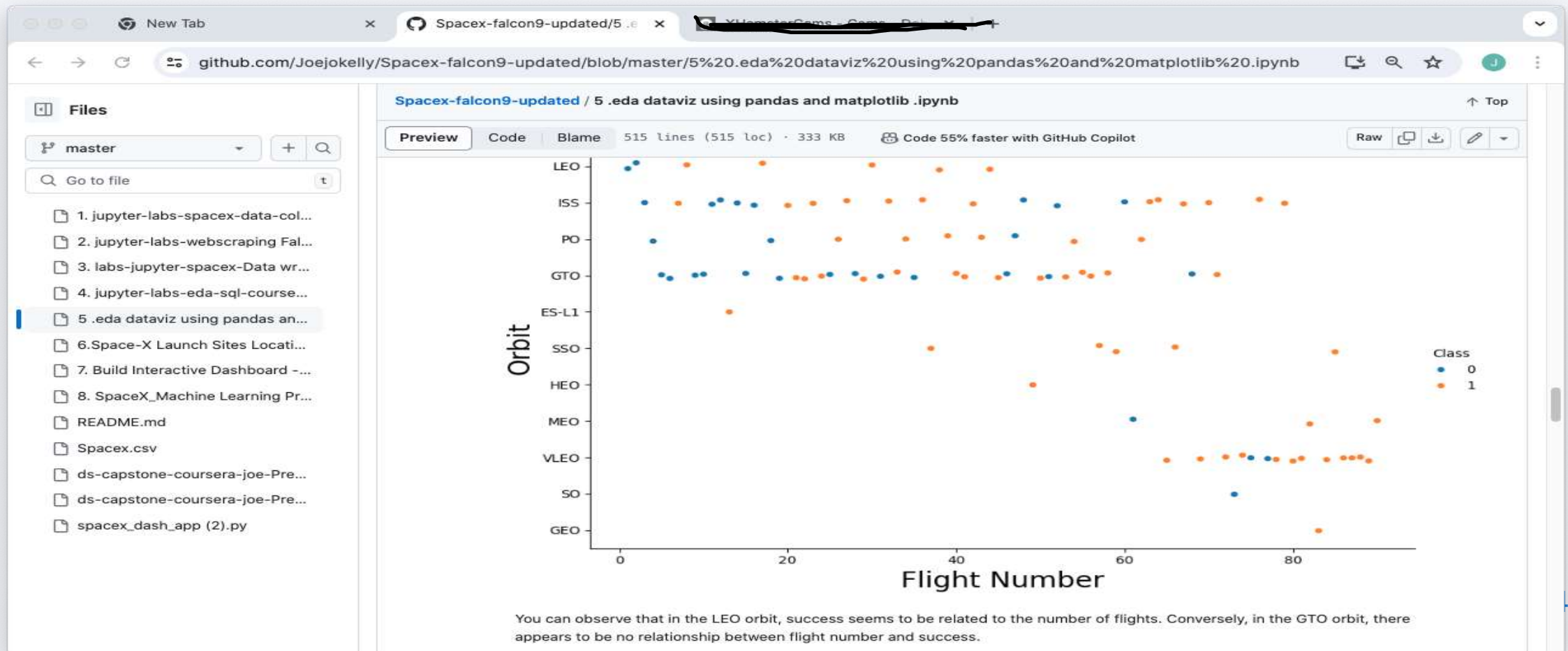
Success Rate vs. Orbit Type



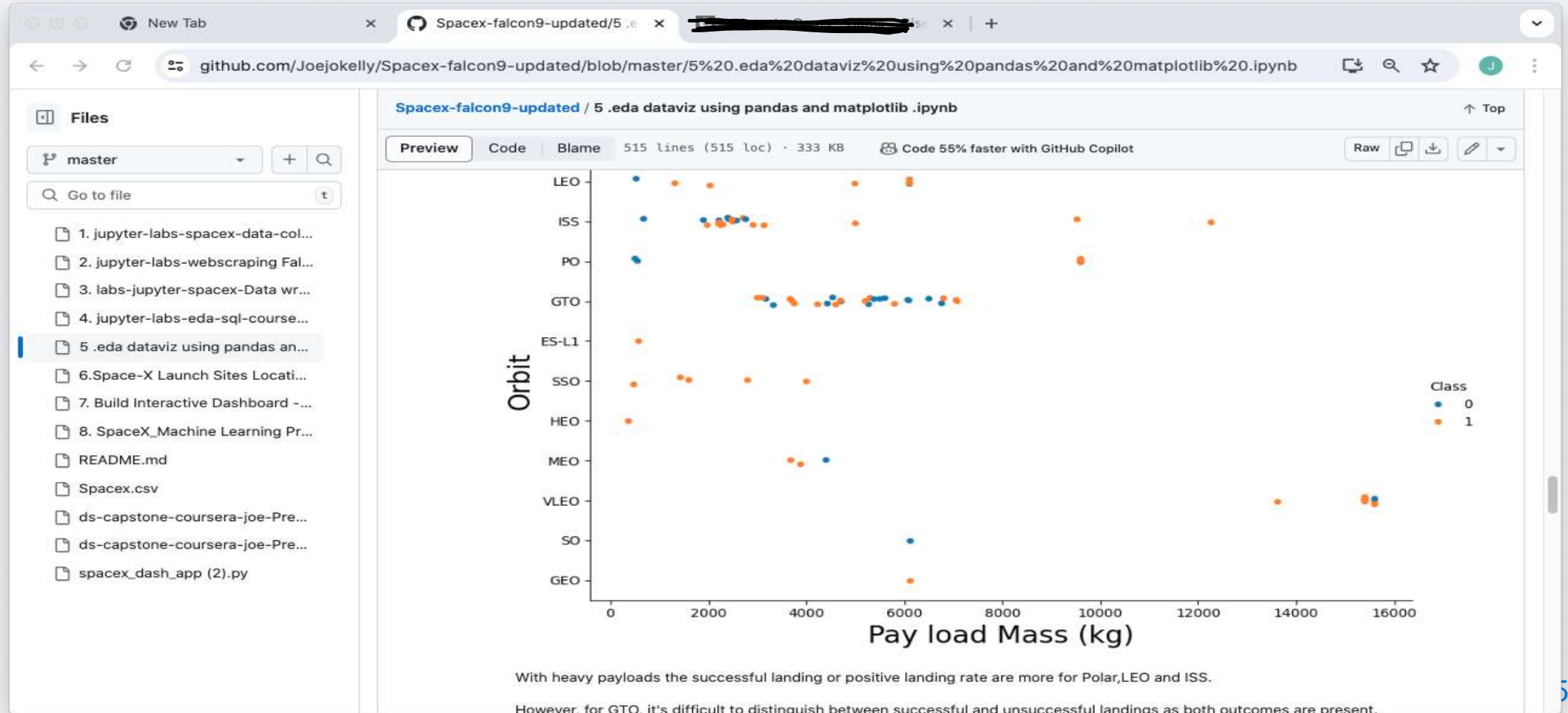
Orbits ES-L1, Geo, Heo, seo, have a higher rate of success, 100% while GTO has about 50%

Flight Number vs. Orbit Type

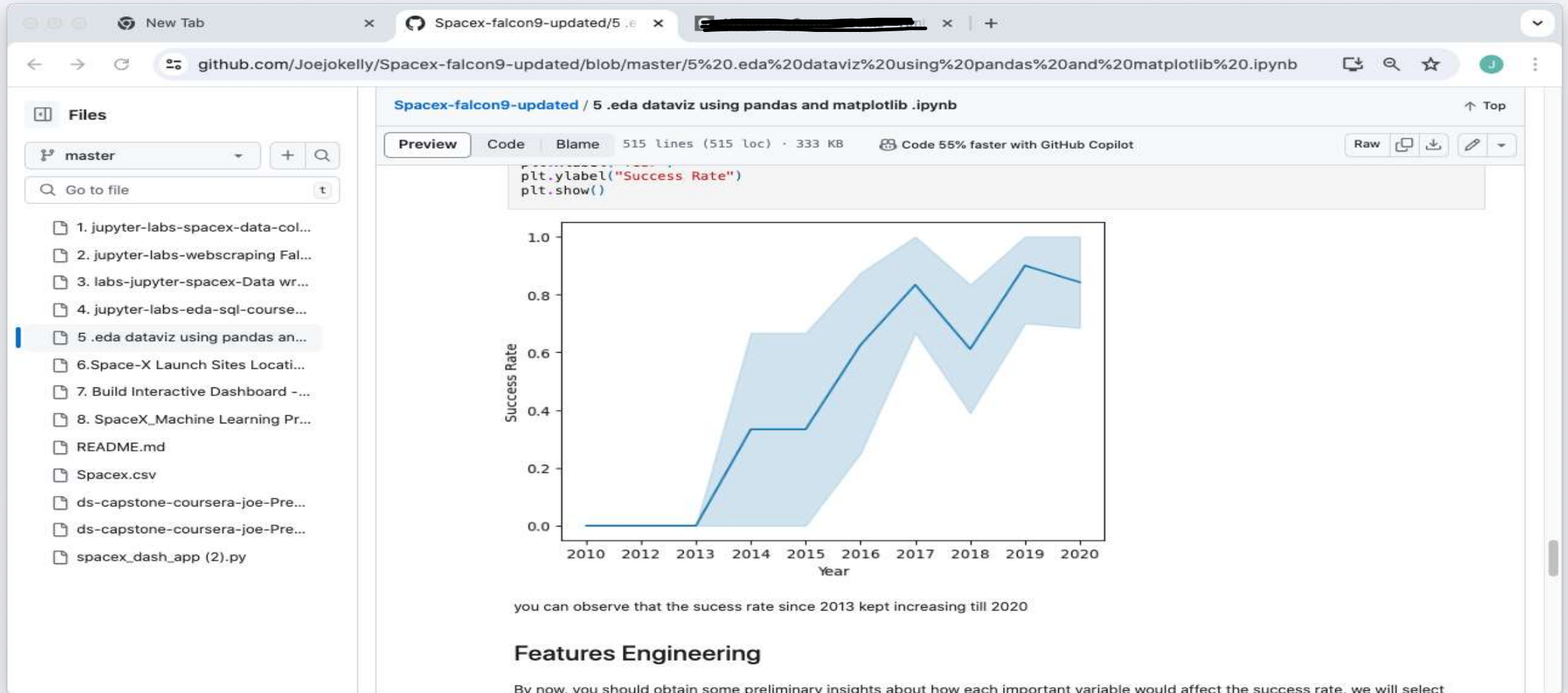
- Show a scatter point o Flight number vs. Orbit type



Payload vs. Orbit Type

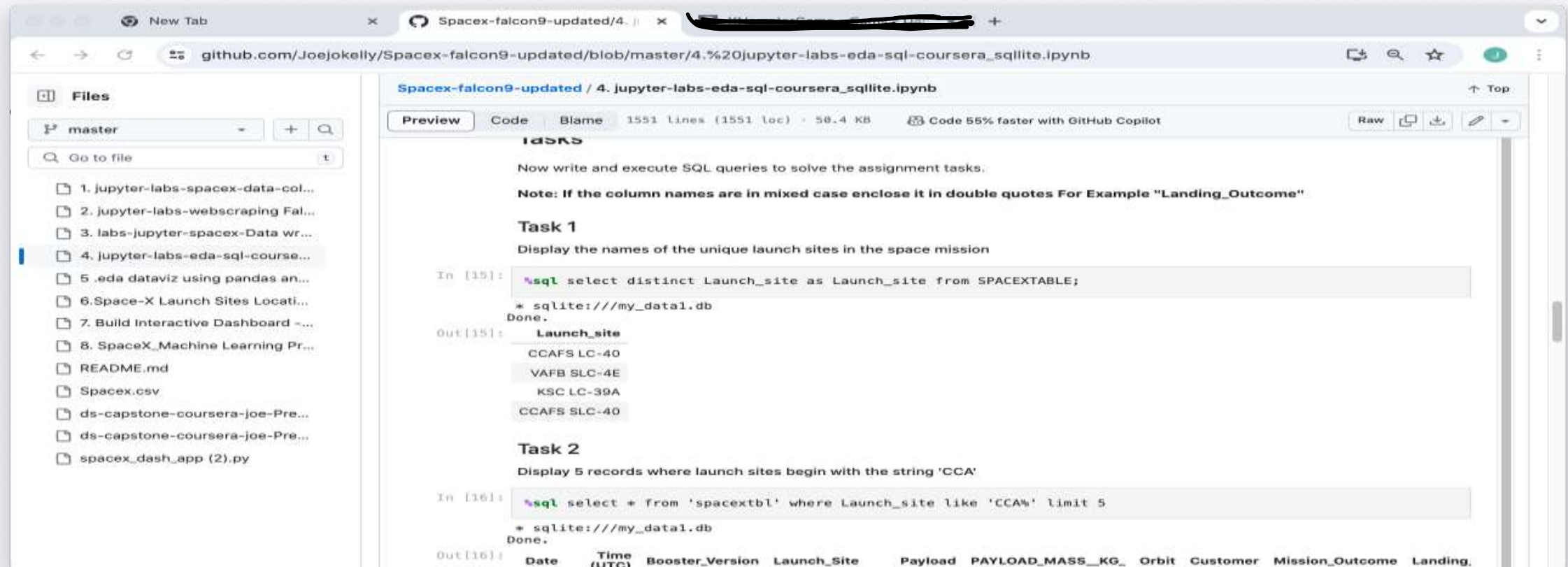


Launch Success Yearly Trend



All Launch Site Names

- Find the names of the unique launch sites



The screenshot shows a Jupyter Notebook titled "4. jupyter-labs-eda-sql-coursera_sqlite.ipynb" on GitHub. The notebook contains two tasks:

Task 1
Display the names of the unique launch sites in the space mission

```
In [15]: %sql select distinct Launch_site as Launch_site from SPACEXTABLE;
```

Out[15]:

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'

```
In [16]: %sql select * from 'spacextbl' where Launch_site like 'CCA%' limit 5
```

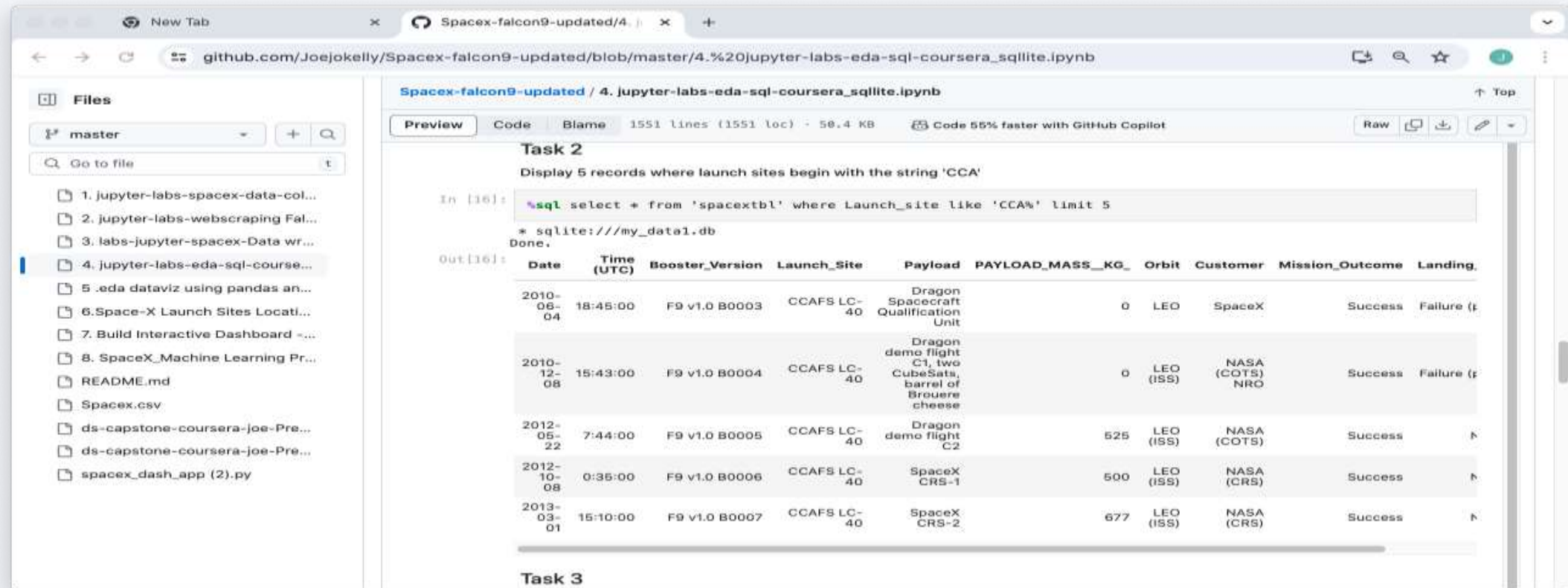
Out[16]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG	Orbit	Customer	Mission_Outcome	Landing
------	------------	-----------------	-------------	---------	-----------------	-------	----------	-----------------	---------

`%sql select distinct Launch_site as Launch_site from SPACEXTABLE;`

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with 'CCA'



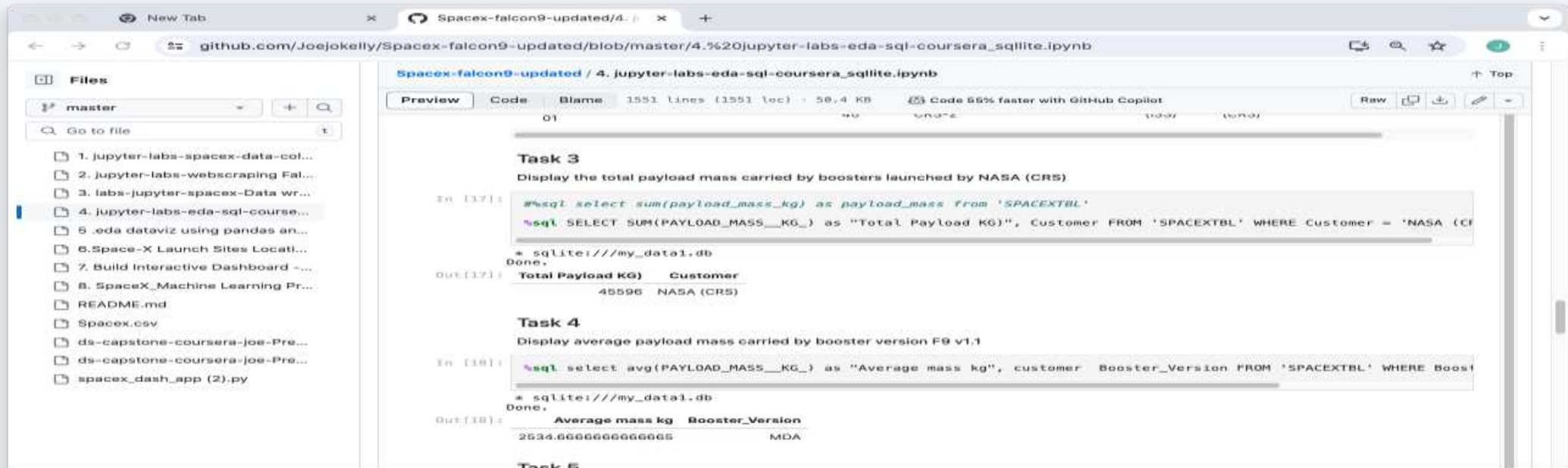
The screenshot shows a Jupyter Notebook titled "Task 2" with the instruction "Display 5 records where launch sites begin with the string 'CCA'". The notebook is open to a file named "4. jupyter-labs-eda-sql-coursera_sqlite.ipynb". The code cell shows a SQL query using the SQLite database "my_data1.db". The output displays a table with 11 columns: Date, Time (UTC), Booster_Version, Launch_Site, Payload, PAYLOAD_MASS_KG_, Orbit, Customer, Mission_Outcome, and Landing. The table contains 5 records where the launch site begins with "CCA".

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Orbit	Customer	Mission_Outcome	Landing
2010-08-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (p
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 (p
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

Used wild cars in the above query to display records with site begin with 'CCA'

Total Payload Mass

- Calculate the total payload carried by boosters from NASA



The screenshot shows a Jupyter Notebook titled "SpaceX-falcon9-updated / 4. jupyter-labs-eda-sql-coursera_sqlite.ipynb". The notebook is open to a cell labeled "Task 3" with the instruction "Display the total payload mass carried by boosters launched by NASA (CR5)". The code in the cell is:

```
In [17]: %sql select sum(payload_mass_kg) as payload_mass from 'SPACEXTBL'
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload KG)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CR5)
```

The output of the second query is displayed as a table:

Total Payload KG)	Customer
45596	NASA (CR5)

Below this, "Task 4" is visible with the instruction "Display average payload mass carried by booster version F9 v1.1". The code for Task 4 is:

```
In [18]: %sql select avg(PAYLOAD_MASS__KG_) as "Average mass kg", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Boost
```

The output for Task 4 is a table:

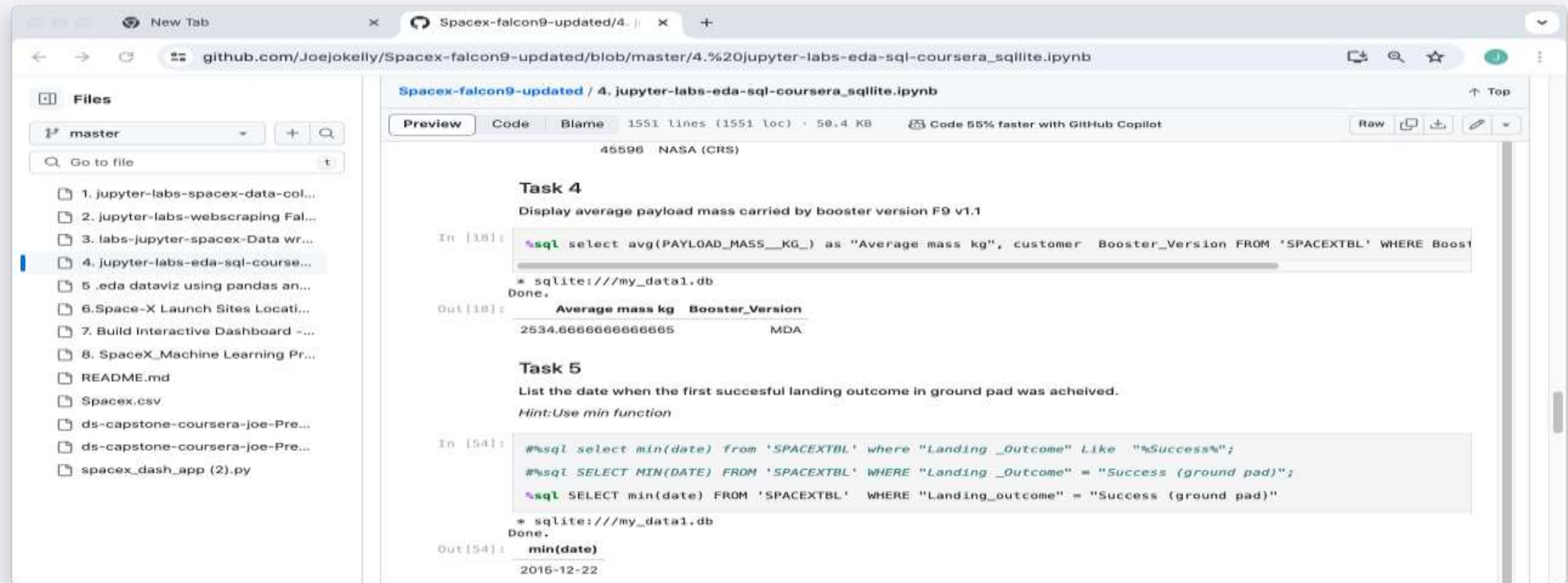
Average mass kg	Customer	Booster_Version
2534.6666666666665	MDA	

Task 5 is partially visible at the bottom of the notebook.

Used the sum function to display the total payload (kg)

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1



The screenshot shows a Jupyter Notebook titled "Spacex-falcon9-updated / 4. jupyter-labs-eda-sql-coursera_sqlite.ipynb" on the GitHub interface. The notebook is open to a cell labeled "Task 4" with the instruction "Display average payload mass carried by booster version F9 v1.1". The code in the cell is a SQL query: `%sql select avg(PAYLOAD_MASS__KG_) as "Average mass kg", customer, Booster_Version FROM 'SPACEXTBL' WHERE Boost`. The output shows a table with two columns: "Average mass kg" and "Booster_Version". The first row of data shows "2534.6666666666665" and "MDA". Below this, "Task 5" is shown with the instruction "List the date when the first succesful landing outcome in ground pad was acheived." and a hint "Hint: Use min function". The code for Task 5 is: `%sql select min(date) from 'SPACEXTBL' where "Landing _Outcome" Like "%Success%";` and `%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";`. The output for Task 5 shows a table with one column: "min(date)" and one row of data: "2016-12-22".

```
In [18]: %sql select avg(PAYLOAD_MASS__KG_) as "Average mass kg", customer, Booster_Version FROM 'SPACEXTBL' WHERE Boost

* sqlite:///my_data1.db
Done.
Out[18]:
```

Average mass kg	Booster_Version
2534.6666666666665	MDA

```
In [54]: %sql select min(date) from 'SPACEXTBL' where "Landing _Outcome" Like "%Success%";
          %sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
          %sql SELECT min(date) FROM 'SPACEXTBL' WHERE "Landing_outcome" = "Success (ground pad)"

* sqlite:///my_data1.db
Done.
Out[54]:
```

min(date)
2016-12-22

Used the avg() function to display average payload booster version

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

The screenshot shows a Jupyter Notebook titled "4. jupyter-labs-eda-sql-coursera_sqlite.ipynb" on GitHub. The notebook is open to a cell containing SQL queries. The left sidebar shows a file explorer with a list of files, including "1. jupyter-labs-spacex-data-col...", "2. jupyter-labs-webscraping Fal...", "3. labs-jupyter-spacex-Data wr...", "4. jupyter-labs-eda-sql-course...", "5 .eda dataviz using pandas an...", "6.Space-X Launch Sites Locati...", "7. Build Interactive Dashboard -...", "8. SpaceX_Machine Learning Pr...", "README.md", "Spacex.csv", "ds-capstone-coursera-joe-Pre...", "ds-capstone-coursera-joe-Pre...", and "spacex_dash_app (2).py".

The main content area shows the following code and output:

```
* sqlite:///my_data1.db
Done.
Out[18]:
```

Average mass kg	Booster_Version
2534.6666666666665	MDA

Task 5
List the date when the first succesful landing outcome in ground pad was acheived.
Hint: Use min function

```
In [54]:
#%sql select min(date) from 'SPACEXTBL' where "Landing_Outcome" Like "%Success%";
#%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing_Outcome" = "Success (ground pad)";
%sql SELECT min(date) FROM 'SPACEXTBL' WHERE "Landing_outcome" = "Success (ground pad)"

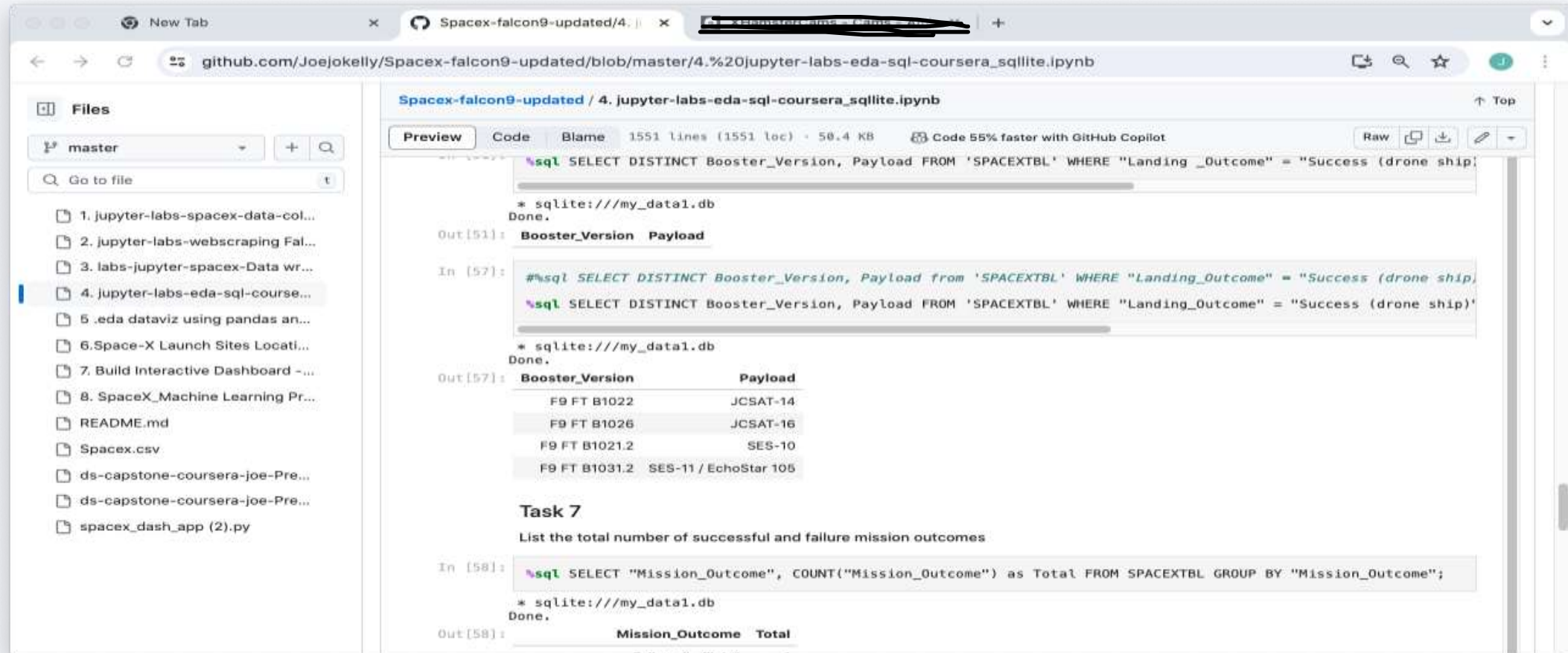
* sqlite:///my_data1.db
Done.
Out[54]:
```

min(date)
2016-12-22

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

```
In [55]:
#%sql SELECT DISTINCT Booster_Version, Payload FROM 'SPACEXTBL'
%sql SELECT DISTINCT Booster_Version, Payload, Landing_Outcome FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)"
#%sql SELECT DISTINCT Booster_Version, Payload, Landing_Outcome FROM SPACEXTBL WHERE "Landing_Outcome" = "Success (drone ship)"
```

Successful Drone Ship Landing with Payload between 4000 and 6000



The screenshot shows a Jupyter Notebook titled "4. jupyter-labs-eda-sql-coursera_sqlite.ipynb" on the GitHub interface. The notebook contains two SQL queries and their corresponding outputs.

Query 1:

```
%sql SELECT DISTINCT Booster_Version, Payload FROM 'SPACEXTBL' WHERE "Landing_Outcome" = "Success (drone ship)"
```

Output 1:

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

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

Task 7

List the total number of successful and failure mission outcomes

Query 2:

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

Output 2:

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

Mission_Outcome	Total
Failure (no splash)	4

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

The screenshot shows a Jupyter Notebook titled "4. jupyter-labs-eda-sql-coursera_sqlite.ipynb" on the GitHub interface. The notebook is open to a cell containing a SQL query and its output.

Task 7
List the total number of successful and failure mission outcomes

In [58]:

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

* sqlite:///my_data1.db
Done.

Out [58]:

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

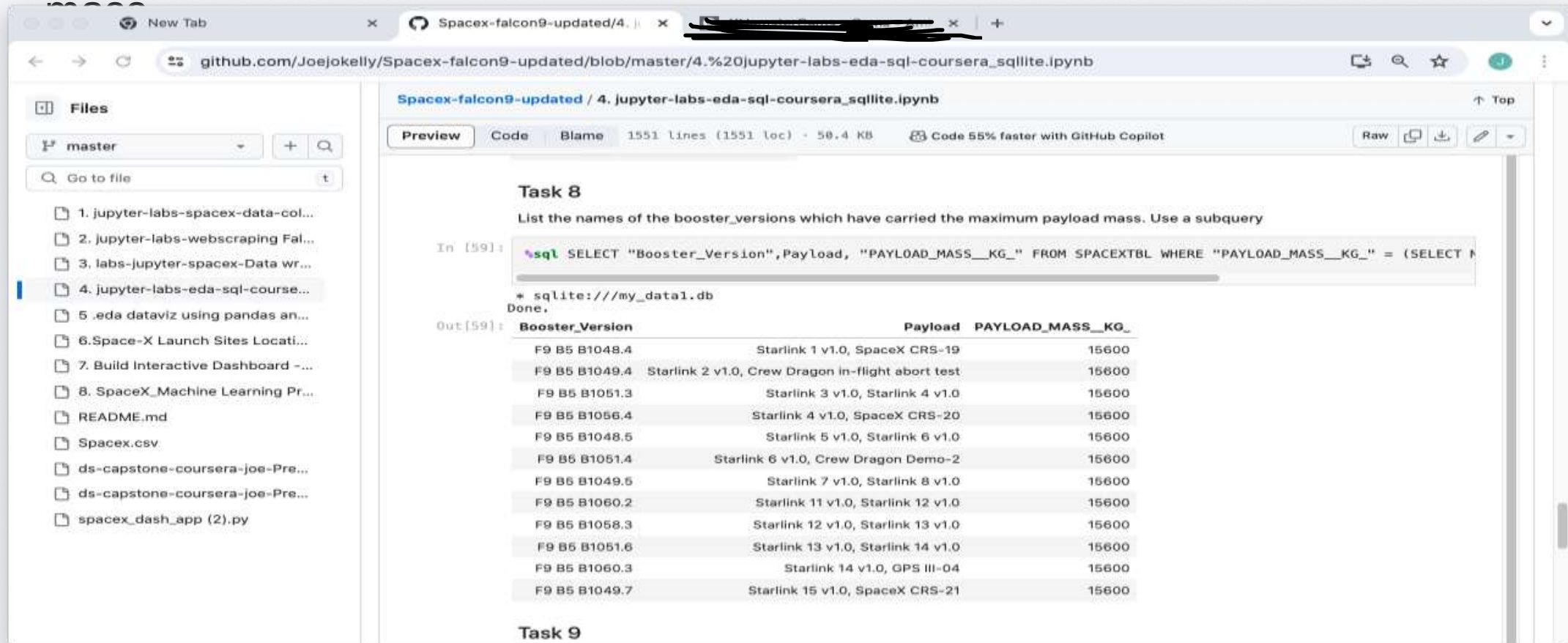
Task 8
List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [59]:

```
%sql SELECT "Booster_Version", "Payload_Mass_KG" FROM SPACEXTBL WHERE "Payload_Mass_KG" = (SELECT
```

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload



The screenshot shows a Jupyter Notebook titled "4. jupyter-labs-eda-sql-coursera_sqlite.ipynb" on the GitHub interface. The notebook is in the "Code" view, showing a SQL query for Task 8. The query is: `SELECT "Booster_Version", Payload, "PAYLOAD_MASS_KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS_KG_" = (SELECT MAX("PAYLOAD_MASS_KG_") FROM SPACEXTBL)`. The output of the query is displayed as a table with three columns: **Booster_Version**, **Payload**, and **PAYLOAD_MASS_KG_**. The table lists 15 different booster versions, all of which have carried a maximum payload of 15600 kg. The tasks are as follows:

Task 8
List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
In [59]: %sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS_KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS_KG_" = (SELECT MAX("PAYLOAD_MASS_KG_") FROM SPACEXTBL)
```

* sqlite:///my_data1.db
Done.

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, OPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

Task 9

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

The screenshot shows a GitHub repository page for 'Spacex-falcon9-updated'. The file '4. jupyter-labs-eda-sql-coursera_sqlite.ipynb' is open, displaying a Jupyter notebook. The notebook contains a task description and a SQL query. The output of the query is a table showing launch records for 2015.

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: 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)` as '2015' for year.

In [68]:

```
#%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS_KG_"
```

```
%sql SELECT substr(Date,0,5), substr(Date, 6, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS_KG_"
```

* sqlite:///my_data1.db
Done.

Out [68]:

substr(Date,0,5)	substr(Date, 6, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG_	Mission_Outcome	"Landing_Outcome"
2015	01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Landing_Outcome
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Landing_Outcome

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.

In [72]:

```
#%sql SELECT * FROM SPACEXTBL WHERE "Landing_Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017')
```

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

- 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

The screenshot shows a Jupyter Notebook titled "4. jupyter-labs-eda-sql-coursera_sqlite.ipynb" on GitHub. The notebook is open to the "Code" tab, displaying a SQL query that filters for successful landings between 2010-06-04 and 2017-03-20. The query is executed, and the output is displayed as a table with 11 columns: Date, Time (UTC), Booster_Version, Launch_Site, Payload, PAYLOAD_MASS_KG, Orbit, Customer, Mission_Outcome, and Landing_Outcome. The table shows 10 rows of data, including launches by SpaceX, Iridium, and SKY Perfect JSAT Group.

```
#%sql SELECT * FROM SPACEXTBL WHERE "Landing_Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017')
%sql SELECT * FROM SPACEXTBL WHERE "Landing_Outcome" LIKE 'Success%' AND (Date BETWEEN '2010-06-04' AND '2017-03-20')
```

* sqlite:///my_data1.db
Done.

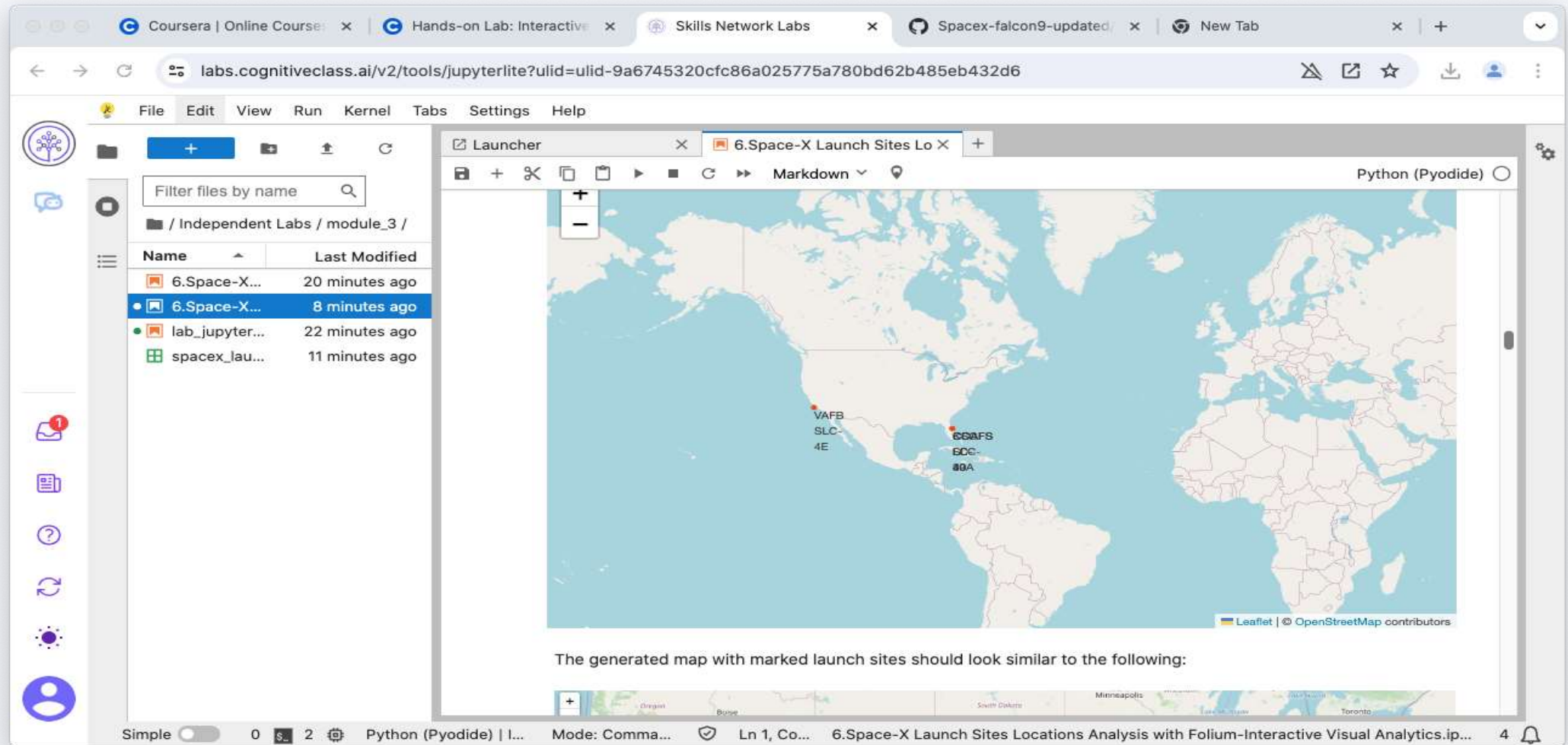
Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG	Orbit	Customer	Mission_Outcome	Landing_Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600	Polar LEO	Iridium Communications	Success	Success
2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600	GTO	SKY Perfect JSAT Group	Success	Success
2016-07-18	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	Thalcom 8	3100	GTO	Thalcom	Success	Success
2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696	GTO	SKY Perfect JSAT Group	Success	Success
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136	LEO (ISS)	NASA (CRS)	Success	Success
2015-12-22	1:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a solid blue background on the left and a satellite photograph of Earth on the right. The Earth's surface is dark, with numerous bright yellow and orange lights representing cities and urban areas. The horizon line of the Earth is visible, separating the dark surface from the deep blue of the upper atmosphere and space.

Section 3

Launch Sites Proximities Analysis

launch site on the global map



The screenshot displays a JupyterLab environment with a web browser at the top showing the URL `labs.cognitiveclass.ai/v2/tools/jupyterlite?ulid=ulid-9a6745320cfc86a025775a780bd62b485eb432d6`. The interface includes a file explorer on the left, a central workspace, and a bottom status bar.

File Explorer:

Name	Last Modified
6.Space-X...	20 minutes ago
6.Space-X...	8 minutes ago
lab_jupyter...	22 minutes ago
spacex_lau...	11 minutes ago

Workspace:

The workspace contains a map titled "6.Space-X Launch Sites Lo". The map shows the Americas with several launch sites marked: VAFB, SLC-4E, EGAFS, ECC-80A, and 80A. The map is powered by Leaflet and OpenStreetMap contributors.

Status Bar:

Simple 0 \$ 2 Python (Pyodide) | ... Mode: Comma... Ln 1, Co... 6.Space-X Launch Sites Locations Analysis with Folium-Interactive Visual Analytics.ip... 4

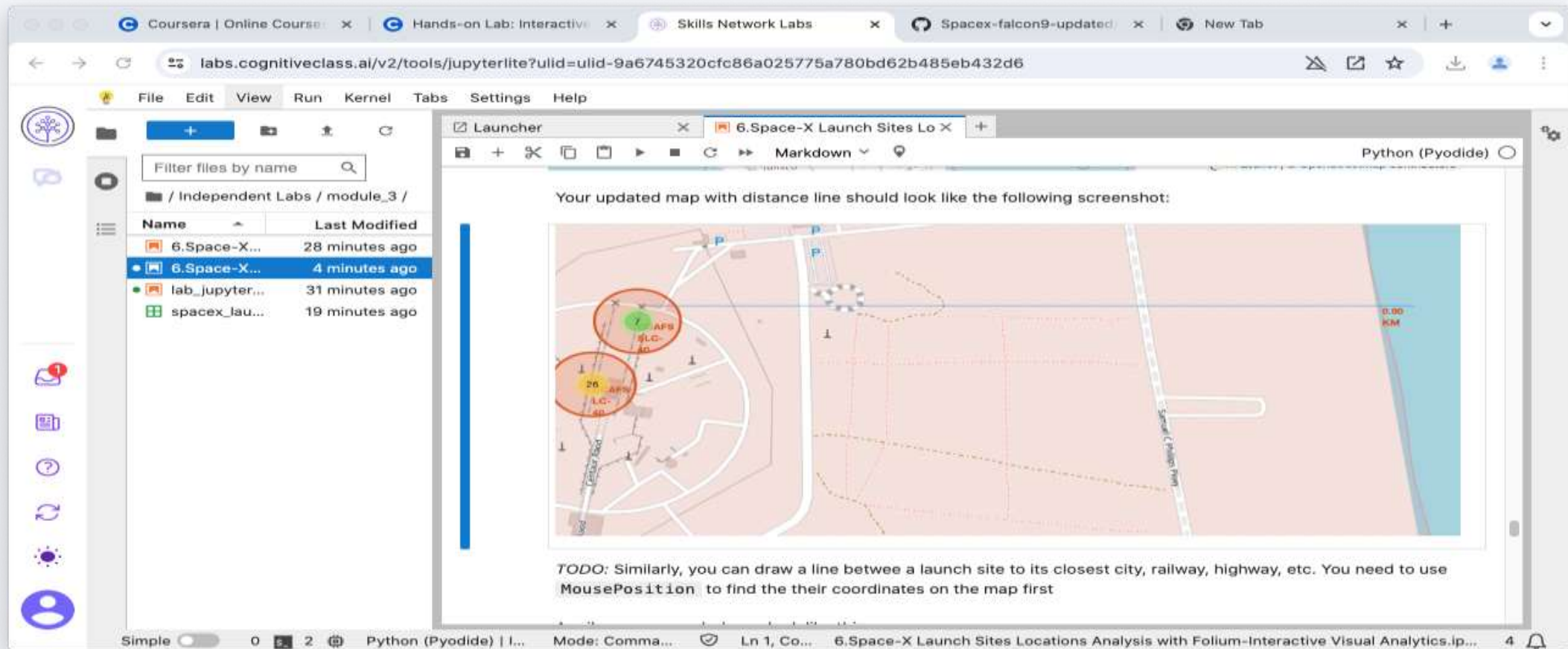
color-labeled launch outcome

- Explore the folium map and make a proper screenshot to show the color-labeled launch outcomes on the map

From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates.

Proximity of Launch site to railway, highway

- Replace <Folium map screenshot 3> title with an appropriate title



The screenshot displays a JupyterLab environment with a file browser on the left and a code editor on the right. The file browser shows a directory structure with files like '6.Space-X...', 'lab_jupyter...', and 'spacex_lau...'. The code editor shows a Python script with a map visualization. The map shows a launch site (LC-40) and its proximity to a railway and highway. The map is titled '6.Space-X Launch Sites Locations Analysis with Folium-Interactive Visual Analytics.ip...'. The code editor shows a TODO comment: 'TODO: Similarly, you can draw a line between a launch site to its closest city, railway, highway, etc. You need to use MousePosition to find the their coordinates on the map first'.

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/ Independent Labs / module_3 /

Name	Last Modified
6.Space-X...	28 minutes ago
6.Space-X...	4 minutes ago
lab_jupyter...	31 minutes ago
spacex_lau...	19 minutes ago

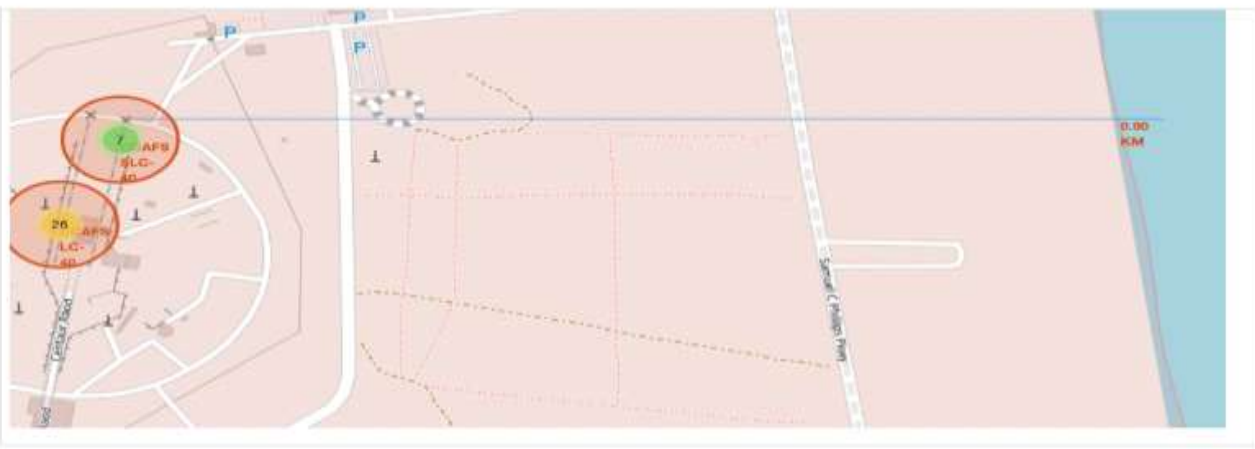
Launcher

6.Space-X Launch Sites Lo X

Markdown

Python (Pyodide)

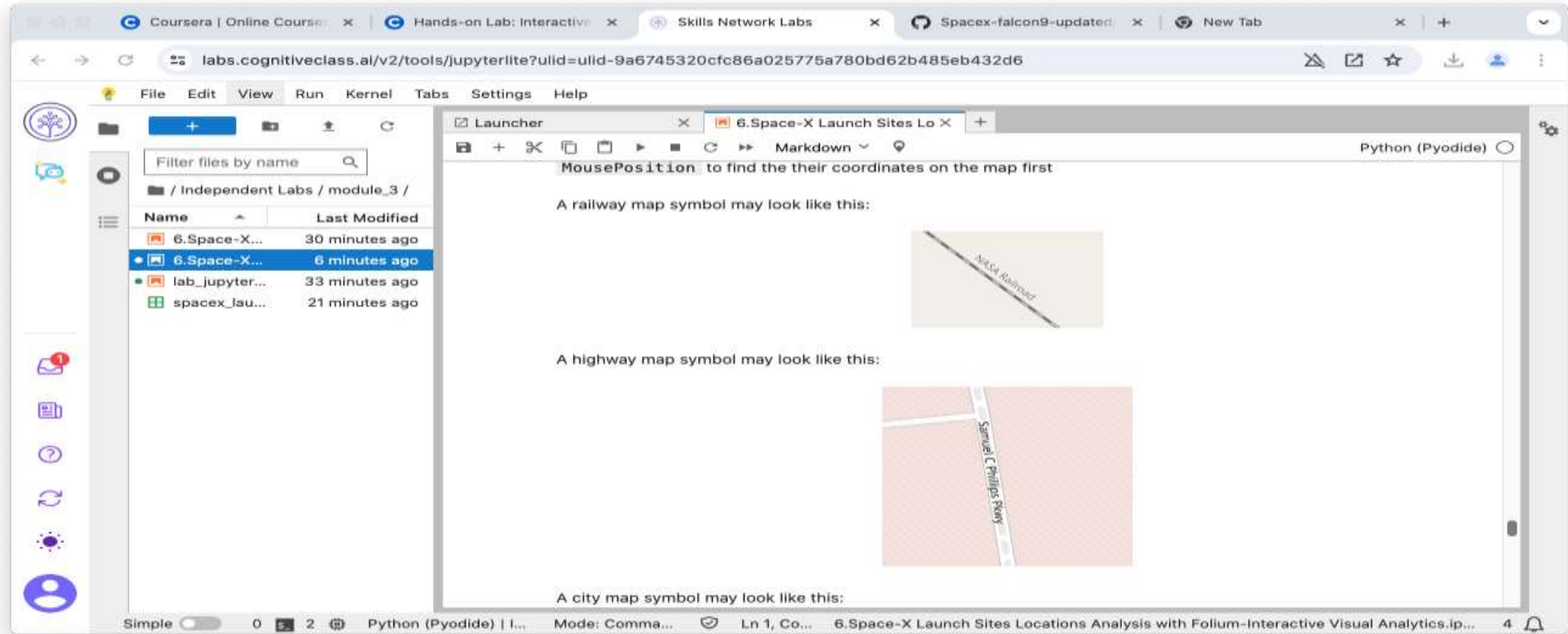
Your updated map with distance line should look like the following screenshot:



TODO: Similarly, you can draw a line between a launch site to its closest city, railway, highway, etc. You need to use MousePosition to find the their coordinates on the map first

Simple 0 2 Python (Pyodide) | I... Mode: Comma... Ln 1, Co... 6.Space-X Launch Sites Locations Analysis with Folium-Interactive Visual Analytics.ip... 4

Proximity contd...




The screenshot shows a JupyterLab environment with a file explorer on the left and a code editor on the right. The file explorer displays a list of files in the directory `/Independent Labs / module_3 /`.

Name	Last Modified
6.Space-X...	30 minutes ago
6.Space-X...	6 minutes ago
lab_jupyter...	33 minutes ago
spacex_lau...	21 minutes ago


The code editor shows a file named `6.Space-X Launch Sites Lo` with the following content:

```
MousePosition to find the their coordinates on the map first
```

A railway map symbol may look like this:



A highway map symbol may look like this:



A city map symbol may look like this:

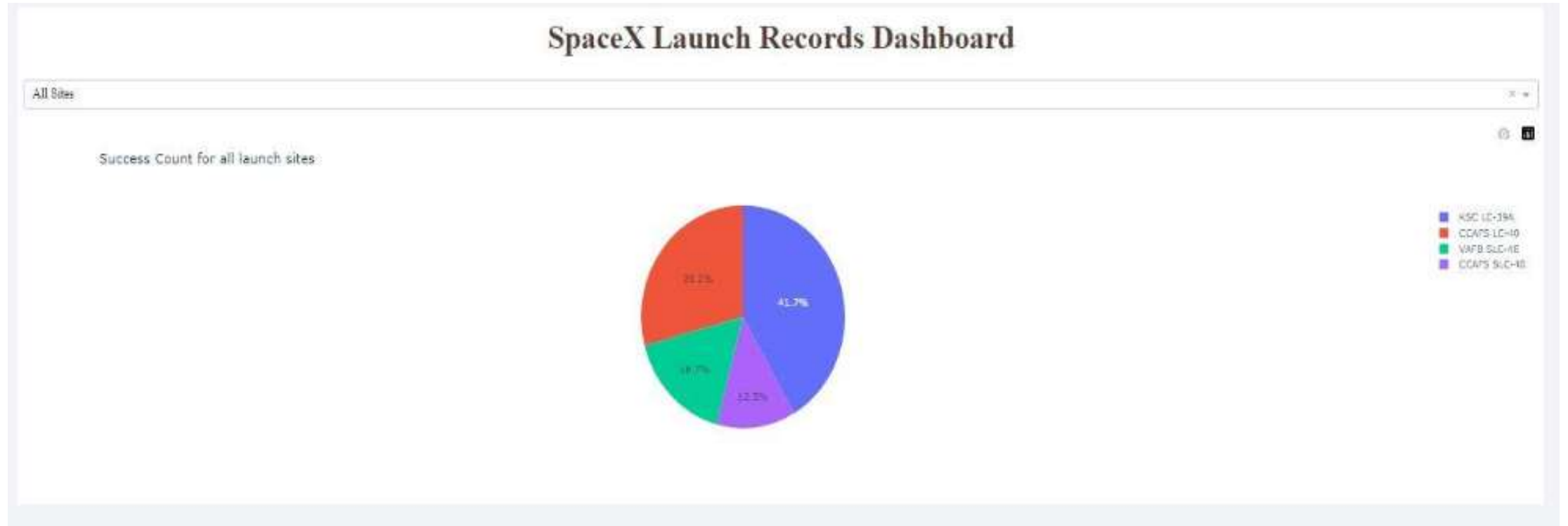
The bottom status bar indicates the current mode is 'Simple', the kernel is 'Python (Pyodide)', and the file path is `6.Space-X Launch Sites Locations Analysis with Folium-Interactive Visual Analytics.ip...`.



Section 4

Build a Dashboard with Plotly Dash

Pie chart for launch Success



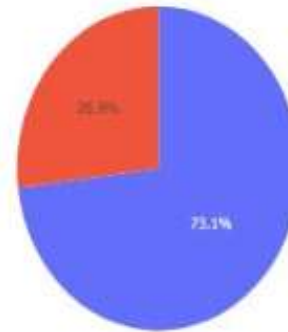
Launch site KSC LC-39A highest success rate 42%
CCAFS LC-40 at 29%, CCAFS LC-40 at 29%

Launch site success rate

SpaceX Launch Records Dashboard

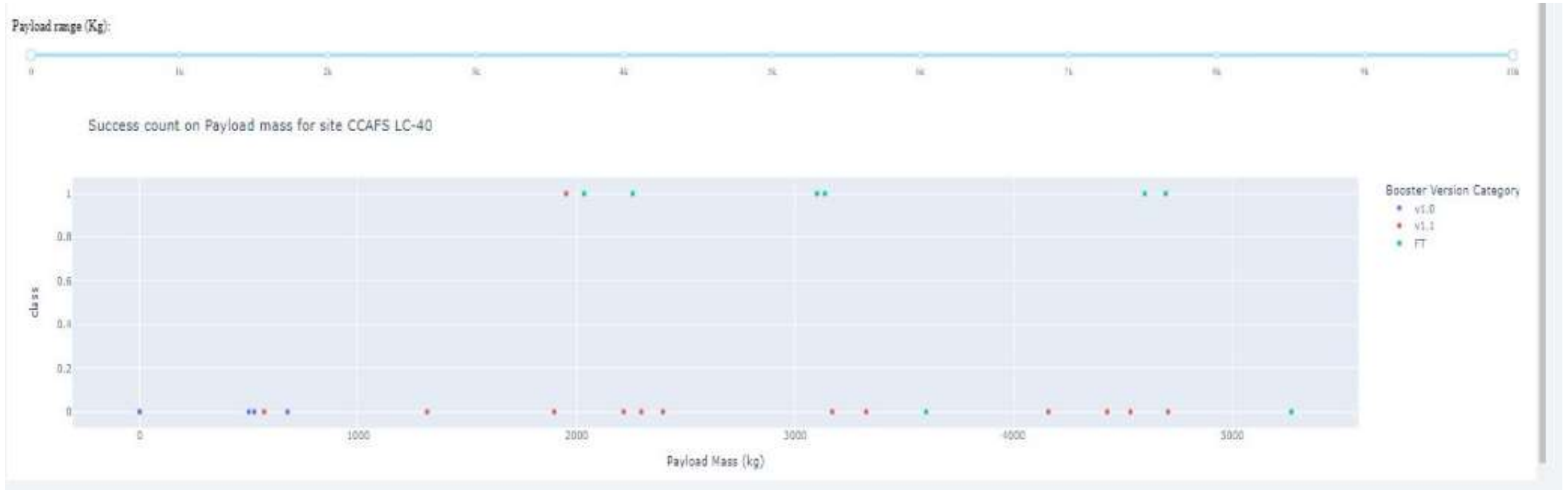
CCAFS LC-40

Total Success Launches for site CCAFS LC-40



- Launch site CCAFS LC-40 had the 2nd highest success ratio of 73% success against 27% failed launches

Payload vs launch site



For launch ccfa-lc 40, the booster version has the largest success range payload mass > 2000



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- All models perform the same, accuracy of 0.83333 on test Data

Out[68]:

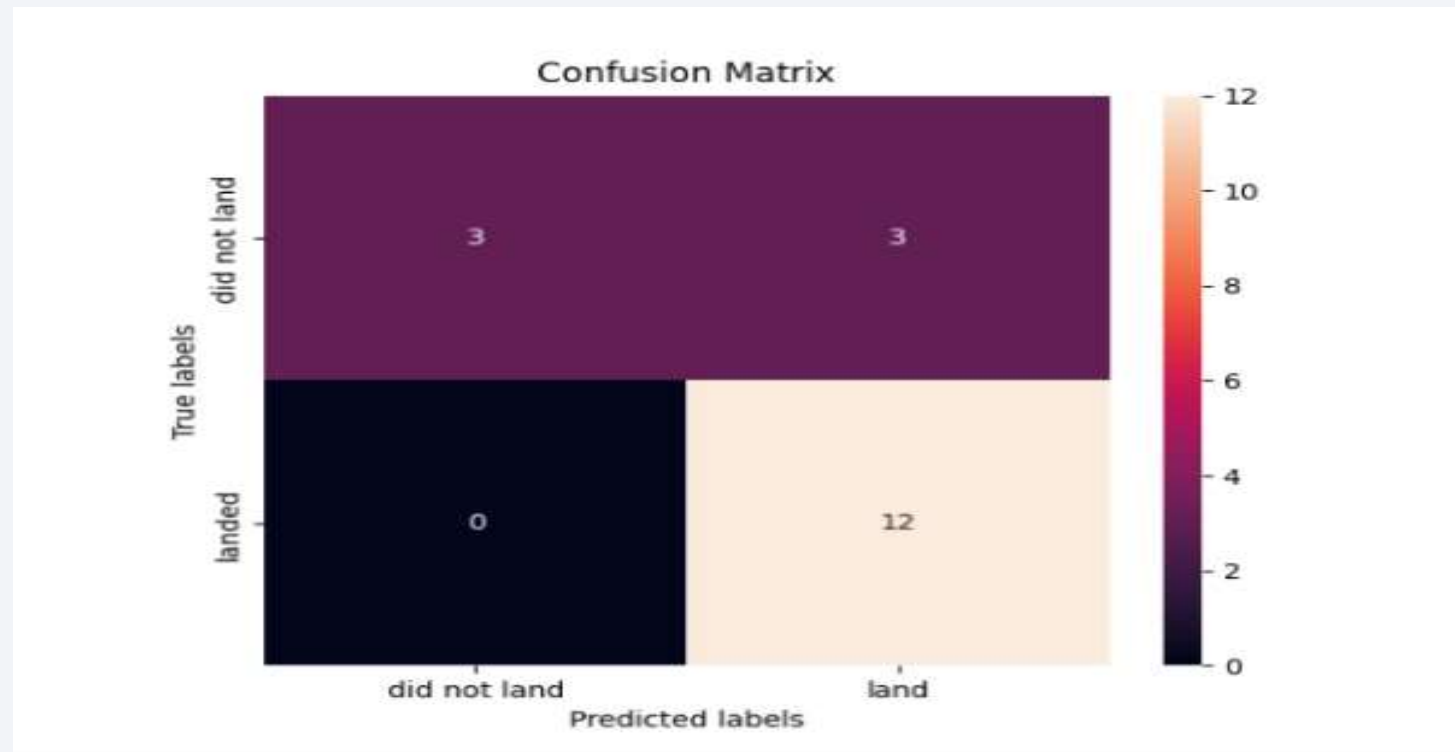
0

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

All the methods perform equally on the test data: i.e. They all have the same accuracy of 0.833333 on the test Data

Confusion Matrix

- All models have the same confusion matrix



Conclusions

- Success rates are different for different launch sites.

CCAFS LC-40 has success rate of 60%

KSC LC-39A and VAFB SLC 4E has a success rate of 77%

- The flight number increases so does the success rate,
- Payload vs Launch site Scatter plot, we find there are no rockets launched s
- 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

Conclusion contd..

S

- Heavy payloads the successful landing is higher for Polar, LEO and ISS. GTO has both positive and negative landings
- Finally the success rate since 2013 kept increasing till 2020

Thank you!

