



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Onayemi Jesutimilehin Temitope
19th November, 2023



Outline

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

Executive Summary

- Summary of methodologies:

Data Collection

Web Scraping

Data Wrangling

Exploratory Data Analysis using SQL

Exploratory Data Analysis and Feature Engineering using Pandas and Matplotlib

Launch Sites Locations Analysis with Folium

Building an Interactive Dashboard with Plotly Dash

Machine Learning Prediction

Executive Summary

- Summary of all results
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Introduction

- Project background and context

The commercial space age is here, companies are making space travel affordable for everyone, examples of companies involved with space travel in one way or the other are: Virgin Galactic, Rocket Lab, Blue Origin, and Space X. Perhaps the most successful is SpaceX. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards 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. There are two stages involved in a rocket launch, the first stage and the second stage. The first stage is quite large and expensive. Unlike other rocket providers, SpaceX's Falcon 9 can recover the first stage.

Introduction

- Problems I want to find answers

In this project, I am taking on the role of a data scientist working for a new rocket company. Space Y that would like to compete with SpaceX founded by Billionaire industrialist Allon Mask. Your job is to determine the price of each launch. I will do this by gathering information about Space X and creating dashboards for your team. I will also determine if SpaceX will reuse the first stage. Instead of using rocket science to determine if the first stage will land successfully, I will train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

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

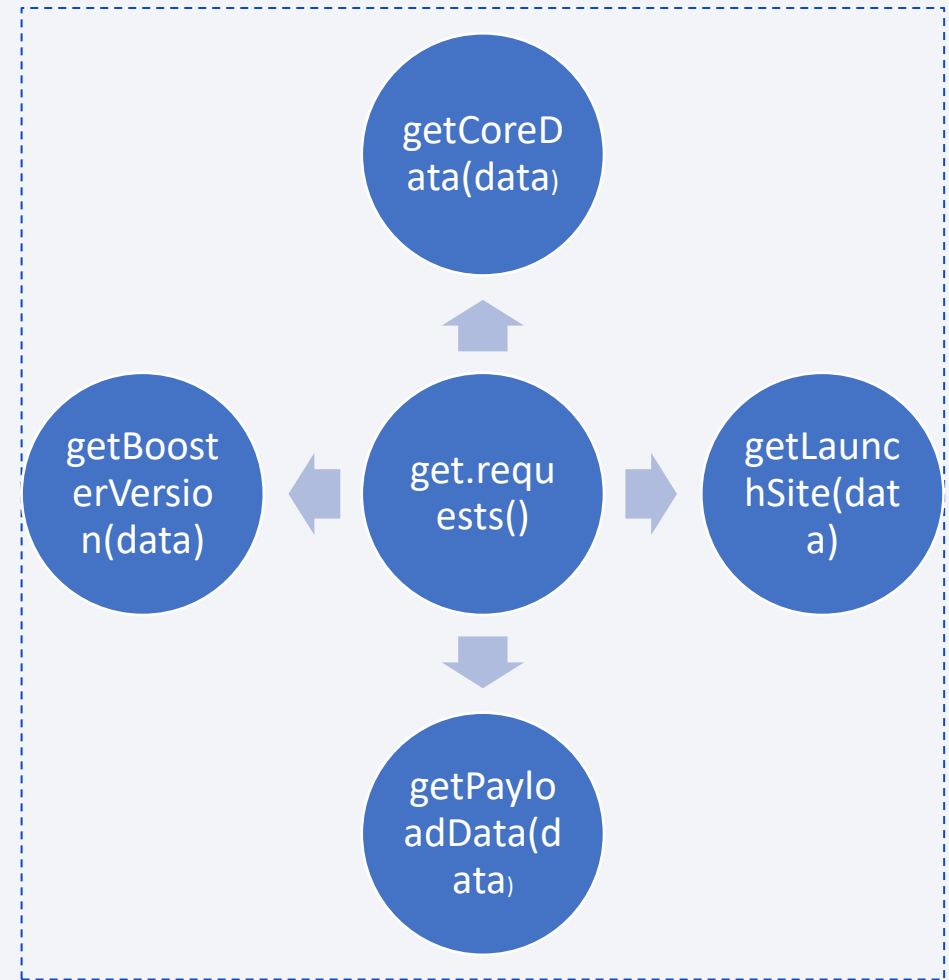
The data was collected by using the `get.requests()` method on the API URL of the data, and also using webscraping using `BeautifulSoup()`

Data Collection – SpaceX API

- As we can see from the flowchart to the right, the `get.requests()` method is the building block in this process of obtaining data. It is used in the four other functions, which are later used to get the data from the API

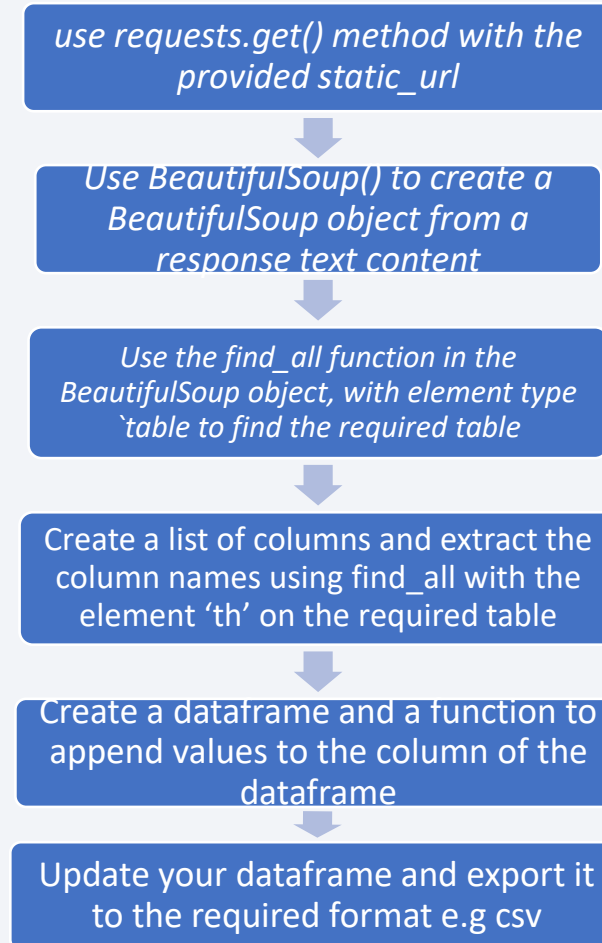
- This is the link of the SpaceX API calls notebook:

[https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/jupyter-labs-spacex-data-collection-api%20\(Solved\).ipynb](https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/jupyter-labs-spacex-data-collection-api%20(Solved).ipynb)



Data Collection - Scraping

- The BeautifulSoup object is used here in scraping data from the web as well the find_all function with the respective elements
- This is the link of the Webscraping notebook:
[https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/jupyter-labs-webscraping%20\(Solved\).ipynb](https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/jupyter-labs-webscraping%20(Solved).ipynb)



Data Wrangling

- Description of how data were processed
- The data is loaded into a pandas dataframe, and checked for null values
- The datatypes function is used to check for all the data types.
- I determined the number of launches on each launchsite
- I determined the number and occurrence of each orbit in the column Orbit.
- I determined the number of landing outcomes, then assigned it to a variable.
- Created a landing outcome label from the Outcome column

This is the link of the Data wrangling notebook:

[https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/labs-jupyter-spacex-Data%20wrangling%20\(Solved\).ipynb](https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/labs-jupyter-spacex-Data%20wrangling%20(Solved).ipynb)

EDA with Data Visualization

1. The first chart here was a categorical plot. It was plotted to see how the FlightNumber (indicating the continuous launch attempts.) and Payload variables would affect the launch outcome.
2. The second chart here was a categorical plot (catplot). To visualize the relationship between Flight Number and Launch Site, with the hue set to 'class'. In order to know the successful and unsuccessful launch attempts associated with each Launch Site.
3. The third chart was a categorical plot. To visualize the relationship between PayloadMass(in kg) and Launch Site, with the hue set to 'class'. The Launch Site is a categorical variable, hence the use of a categorical plot.

EDA with Data Visualization

4. The fourth chart was a barplot, which was constructed in order to visualize the relationship between success rate of each orbit type.
5. In the fifth, I had a categorical plot to visualize the relationship between FlightNumber and Orbit type, Orbit type here, is a categorical data.
6. For the sixth chart, it was also a categorical plot, but in this case, it was constructed to visualize the relationship between Payload and Orbit type.
7. The last chart under the Exploratory Data analysis with Data Visualization section, was a line chart which was plotted to visualize how the success rate of the launches changed from year to year.

EDA with Data Visualization

Below is the GitHub link of the completed Exploratory Data analysis with data visualization notebook:

[https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/jupyter_labs_eda_dataviz\(Solved\).ipynb](https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/jupyter_labs_eda_dataviz(Solved).ipynb)

EDA with SQL

In this section we use SQL magic in python to query from the SPACEXTABLE after establishing a connection with a database.

1. I displayed the names of the unique launch sites in the space mission using sql query.
2. I displayed five records where launch sites begin with the string 'CCA'.
3. I displayed the total payload mass carried by boosters launched by NASA (CRS).
4. I displayed the average payload mass carried by booster version F9 v1.1
5. I listed the date when the first successful landing outcome in ground pad was achieved.

EDA with SQL

6. I listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
7. I listed the total number of successful and failure mission outcomes
8. I listed the names of the booster versions which have carried the maximum payload mass using a subquery.
9. I listed the records which will display the month names, failure landing outcomes in drone ship ,booster versions, launch sites for the months in year 2015.
10. I ranked 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.

EDA with SQL

Below is the GitHub link of the completed Exploratory Data analysis with data visualization notebook:

[https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/jupyter_labs_eda_sql_coursera_sqlite\(Solved\).ipynb](https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/jupyter_labs_eda_sql_coursera_sqlite(Solved).ipynb)

Build an Interactive Map with Folium

I created and added map objects such as markers, circles, markercluster, and lines on the folium map.

- `folium.Circle` was used to add a highlighted circle area with a text label on a specific coordinate
- `folium.Marker()` object was used to plot markers on the map
- Marker clusters were used to simplify a map containing many markers having the same coordinate.

Lines were drawn on the map to indicate a straight line distance from a launch site to its closest city, railway, and highway.

Build an Interactive Map with Folium

Below is the GitHub link of the completed Interactive map with Folium map notebook:

https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/lab_jupyter_launch_site_location.ipynb

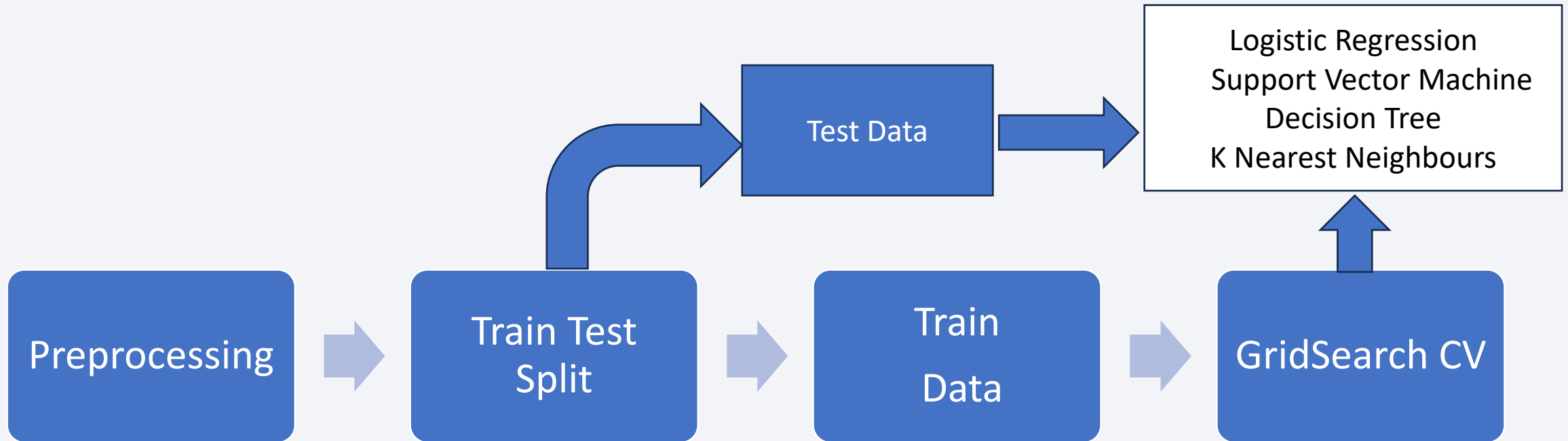
Build a Dashboard with Plotly Dash

- I built a dashboard containing a pie chart and scatterplot, that could change depending on the values or options selected, in terms of Launch Site, or the payload mass.
- I added the pie chart in order to be able to view the percentage of successful launches per location. The scatter plot however, was added to view the successful and failed launches per range of payload mass.

To view the raw python file, with the complete code for creating the dashboard, go to the URL address below:

<https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/Dash.py>

Predictive Analysis (Classification)



Predictive Analysis (Classification)

As represented in the flowchart in the previous page, this is the process by which we found the best performing model:

1. The data was first preprocessed using `StandardScaler()`
2. Then the data was split into train and test data.
3. Created `GridSearch` objects using the four different classification models as indicated in the flowchart.
4. Fit the `GridSearch` object on the train data.
5. Find the best performing parameters and score on the train data for all the models
6. Find the score on the test data for all the models and hence, find the best performing model.

Predictive Analysis (Classification)

To view the predictive analysis notebook, go to the URL address below:

- [https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5_\(solved\).ipynb](https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5_(solved).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 dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower-left quadrant. The overall effect is dynamic and technological.

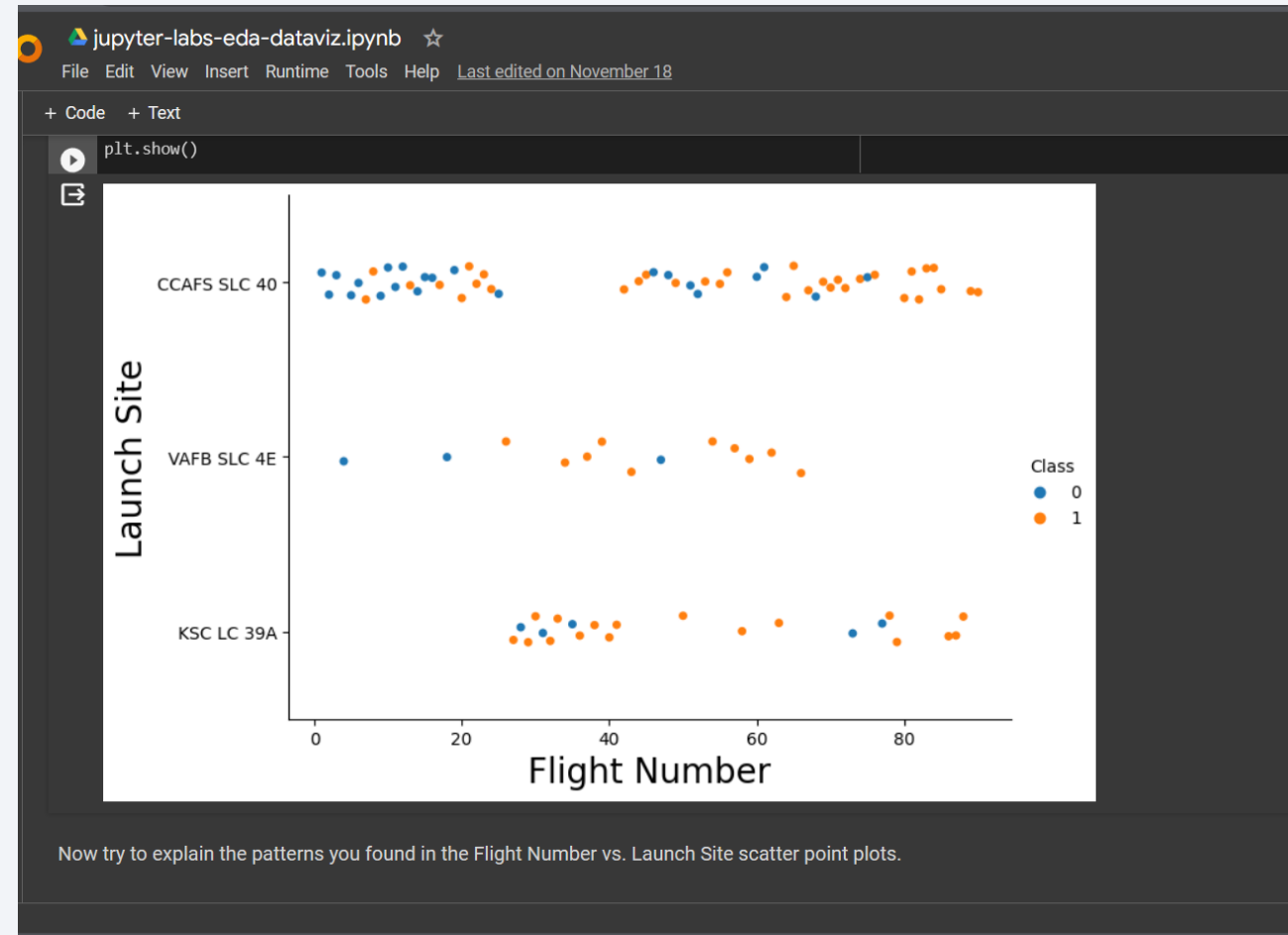
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

The screenshot to the right is a plot of Launch Site against Flight Number. It shows the different Launch Sites with the respective Flight numbers. The Class 0, which are the blue points on the graph are the failures, while Class 1, which are the orange-colored points on the plot, are the successes.

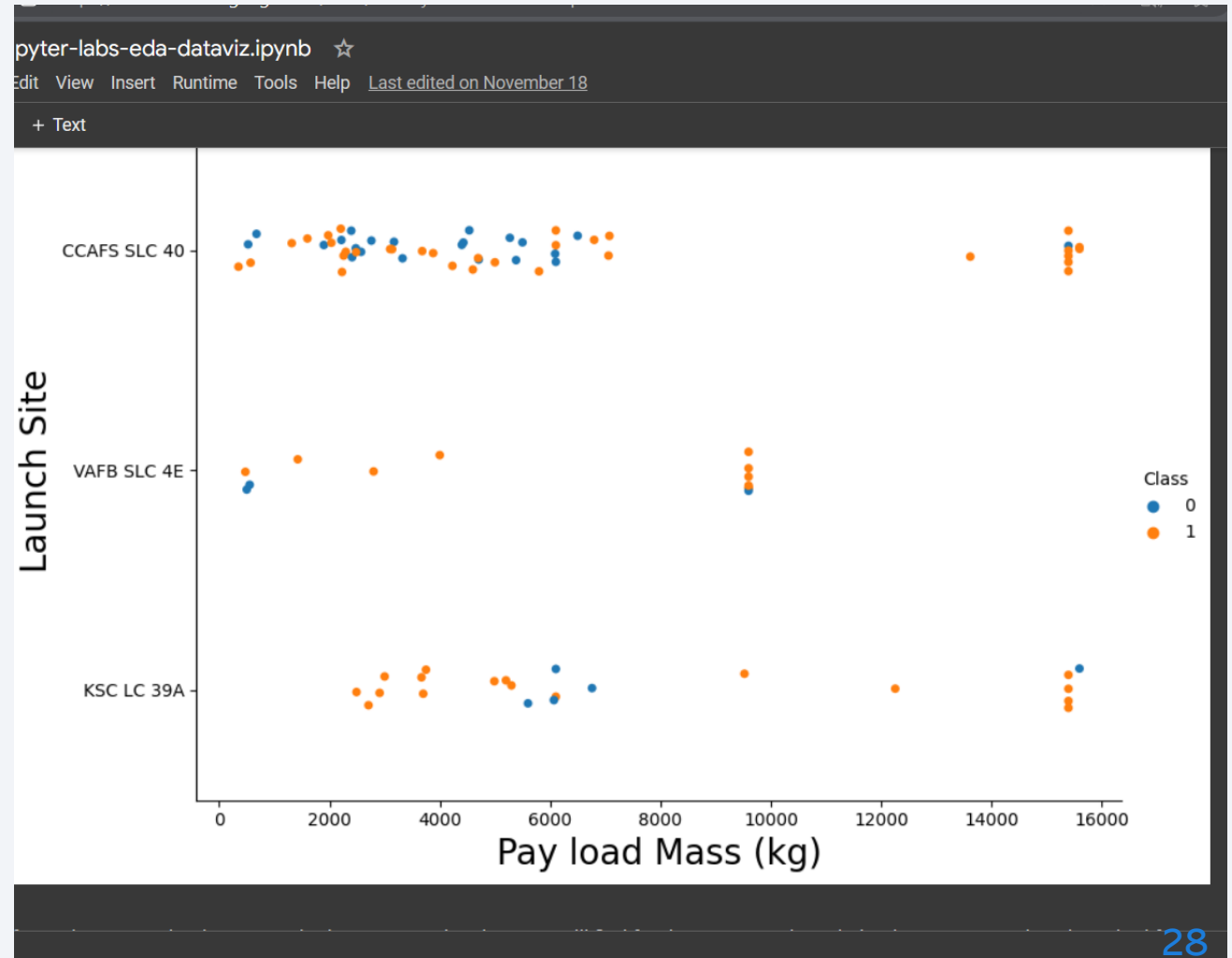
From the screenshot therefore, we can infer that with the increase in Flight Number, the percentage of successful outcomes increases with each Launch site.



Payload vs. Launch Site

The screenshot to the right is a plot of Launch Site against Payload Mass (kg). It shows the different Launches belonging to different Launch sites with the respective Payload Mass associated with them. The Class 0, which are the blue points on the graph are the failures, while Class 1, which are the orange-colored points on the plot, are the successes.

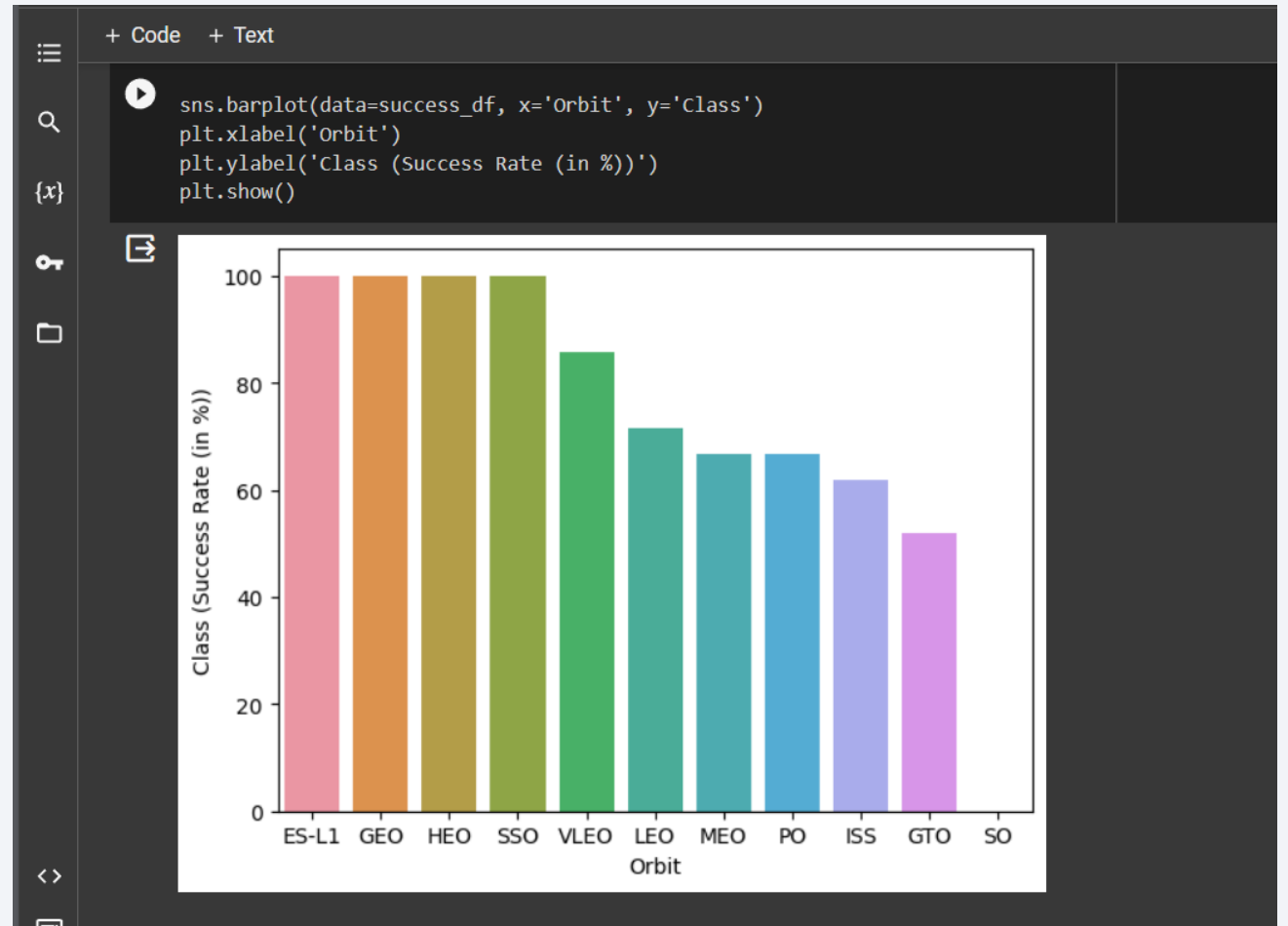
Also, in this case, the closer the payload Mass to 10000kg, the more the tendency for a successful outcome.



Success Rate vs. Orbit Type

The screenshot to the right is a barplot of Success Rate against Orbit type. It shows the different Orbit types with their respective Success Rates. In the graph, we can see that certain orbit types have a higher success rate than others, some even have success rates as high as 100%.

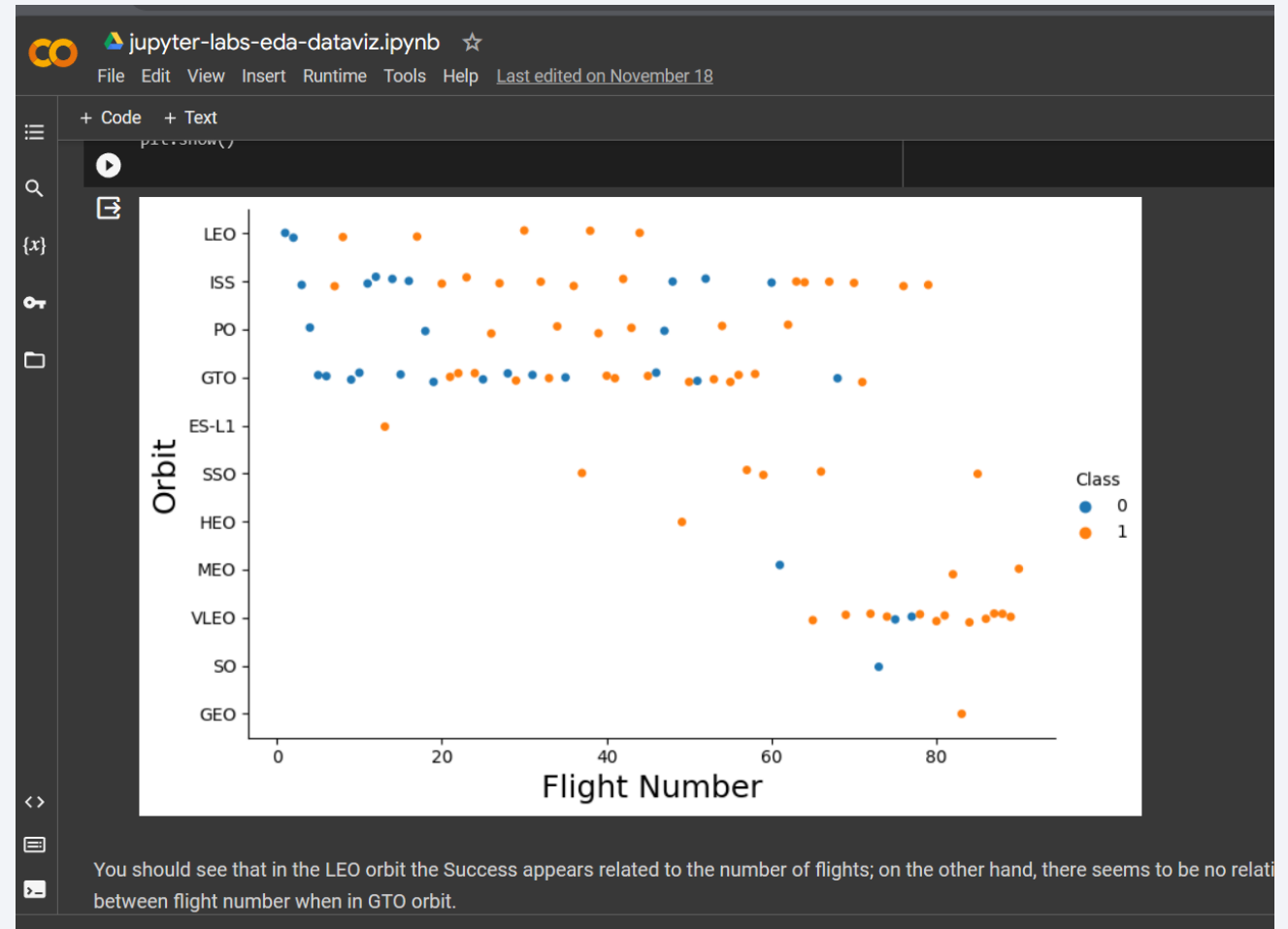
But this measurement can be misleading, and the reason why is that some orbit types may not be used as frequently as others thereby leading to the possibility of higher rates.



Flight Number vs. Orbit Type

The screenshot to the right is a categorical plot of Orbit type against Flight Number. It shows the different Launches, with their orbit types and their respective Flight numbers. The Class 0, which are the blue points on the graph are the failures, while Class 1, which are the orange-colored points on the plot, are the successes.

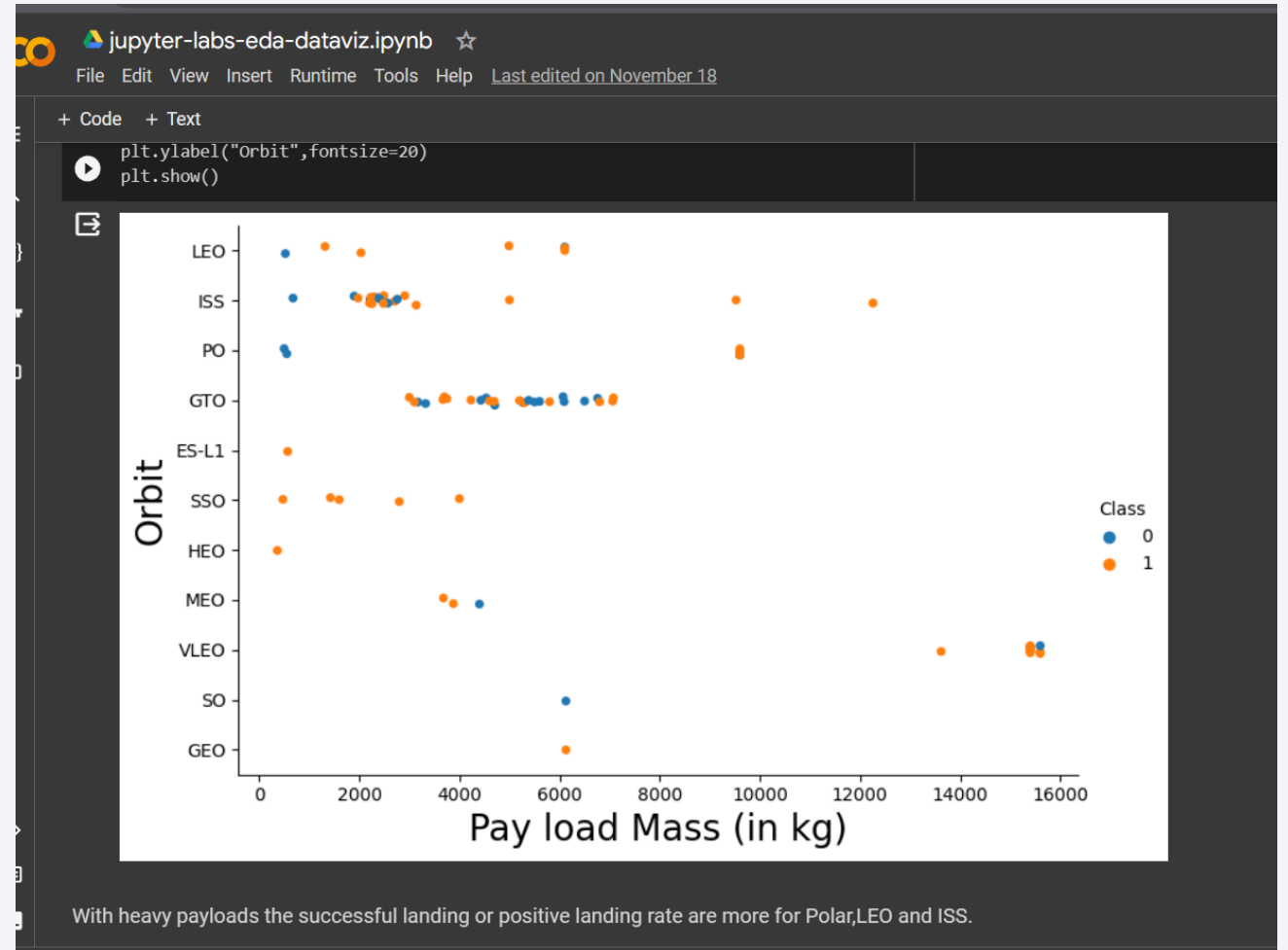
The graph in this picture consolidates the point I made in the previous slide. In the graph, we can see that Orbit type 'ES-L1' was only used once in the space mission and its outcome was positive, leading to a success rate of 100%



Payload vs. Orbit Type

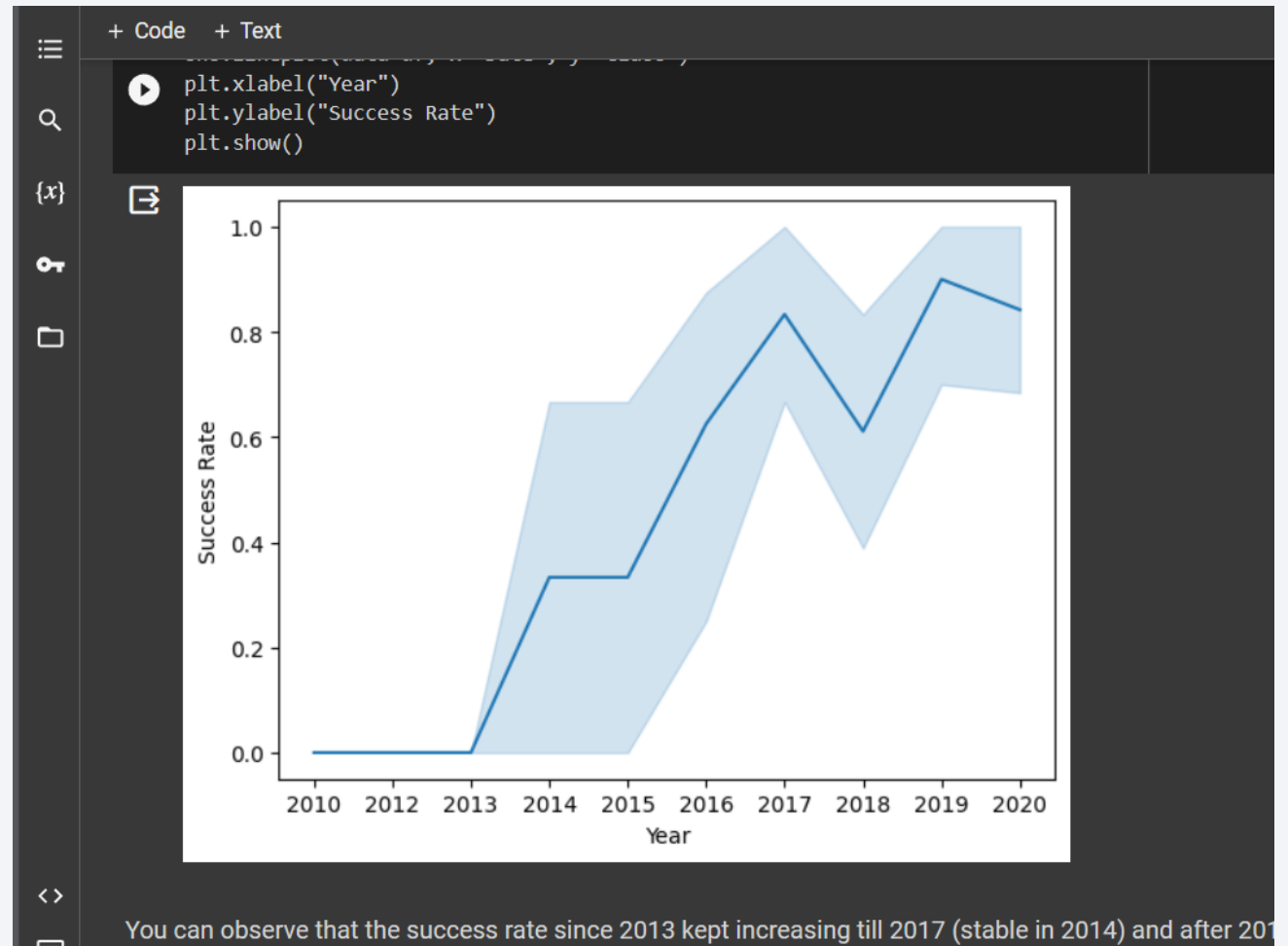
The screenshot to the right is a categorical plot of Orbit type against Payload Mass (kg). It shows the different Launches belonging to different Orbit types with the respective Payload Mass.

The Class 0, which are the blue points on the graph are the launches that failed, while Class 1, which are the orange-colored points on the plot, are the successful launches.



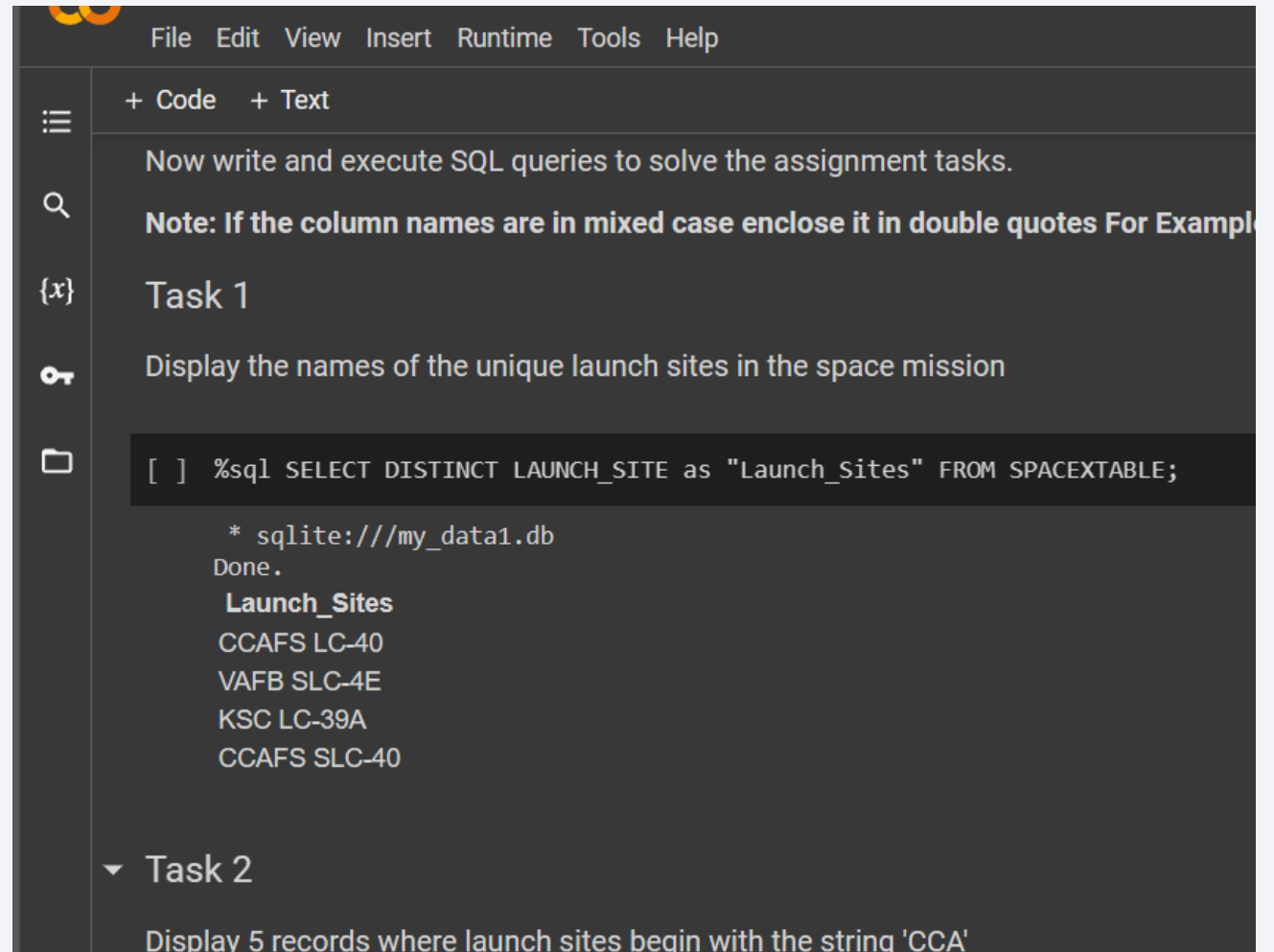
Launch Success Yearly Trend

The screenshot to the right is a line plot of the Launch Success Rate per year. It shows the trend of the average success rate of the Launches for the respective years. This line plot in general suggests an improvement in the success rate with time, and this good sense, because with breakthroughs in research and technology year on year, there is supposed to be a proportionate improvement in success rate.



All Launch Site Names

The picture located to the right of this text shows the unique Launch sites in the mission, and this is achieved using sql magic, writing an sql query in python.



The screenshot shows a Jupyter Notebook with a dark theme. The top menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu, there are tabs for '+ Code' and '+ Text'. The main area contains the following text:

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

Task 1

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

```
[ ] %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTABLE;
```

* sqlite:///my_data1.db
Done.

Launch_Sites
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'

Launch Site Names Begin with 'CCA'

The picture located to the right of this text shows five records with Launch Site names beginning with 'CCA', and this is achieved using sql magic, writing an sql query in python.

```
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

[ ] VAFB SLC-4E
    KSC LC-39A
    CCAFS SLC-40

Task 2

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

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

* sqlite:///my_data1.db
Done.

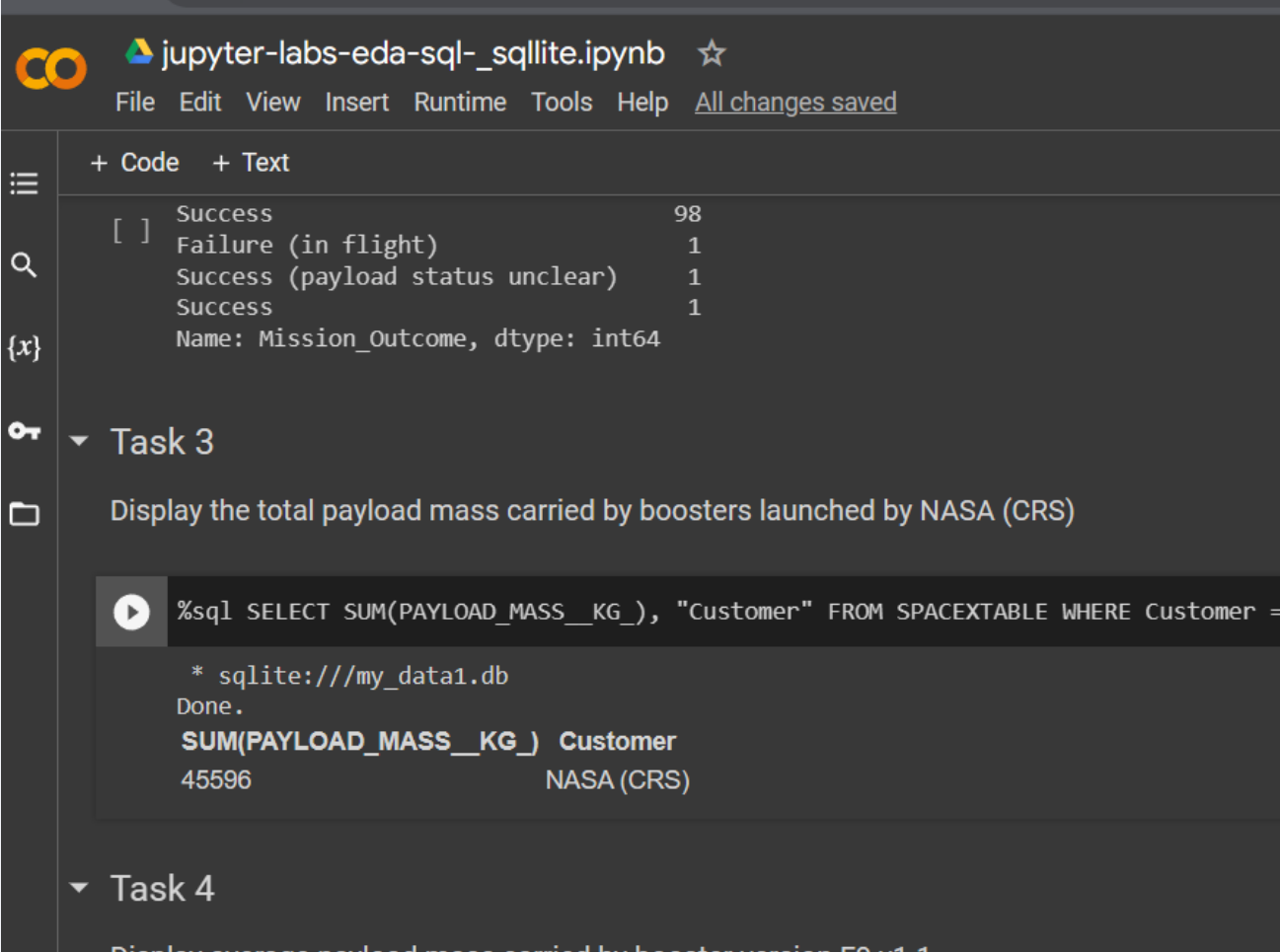
   Date      Time (UTC) Booster_Version Launch_Site      Payload
2010-06-04 18:45:00 F9 v1.0 B0003  CCAFS LC-40 Dragon Spacecraft Qualification Unit
2010-12-08 15:43:00 F9 v1.0 B0004  CCAFS LC-40 Dragon demo flight C1, two CubeSats, barrel of Brou
2012-05-22 7:44:00 F9 v1.0 B0005  CCAFS LC-40 Dragon demo flight C2
2012-10-08 0:35:00 F9 v1.0 B0006  CCAFS LC-40 SpaceX CRS-1
2013-03-01 15:10:00 F9 v1.0 B0007  CCAFS LC-40 SpaceX CRS-2

[ ] df['Mission_Outcome'].value_counts()

Success      98
Failure (in flight)  1
Success (payload status unclear)  1
Success      1
```

Total Payload Mass

The picture located to the right of this text shows the total payload mass carried by boosters launched by NASA (CRS). This is achieved using sql magic, writing an sql query in python.



The screenshot shows a JupyterLab notebook titled 'jupyter-labs-eda-sql-_sqlite.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar ('All changes saved'). The notebook content is organized into tasks. Task 3 is expanded, showing a description: 'Display the total payload mass carried by boosters launched by NASA (CRS)'. Below the description is a code cell with a play button icon. The code cell contains a SQL query using the %sql magic: '%sql SELECT SUM(PAYLOAD_MASS_KG_), "Customer" FROM SPACEXTABLE WHERE Customer = * sqlite:///my_data1.db'. The output of the query is displayed below the code, showing the sum of payload mass for NASA (CRS) as 45596. Task 4 is partially visible below Task 3.

```
+ Code + Text
[ ] Success 98
Failure (in flight) 1
Success (payload status unclear) 1
Success 1
Name: Mission_Outcome, dtype: int64

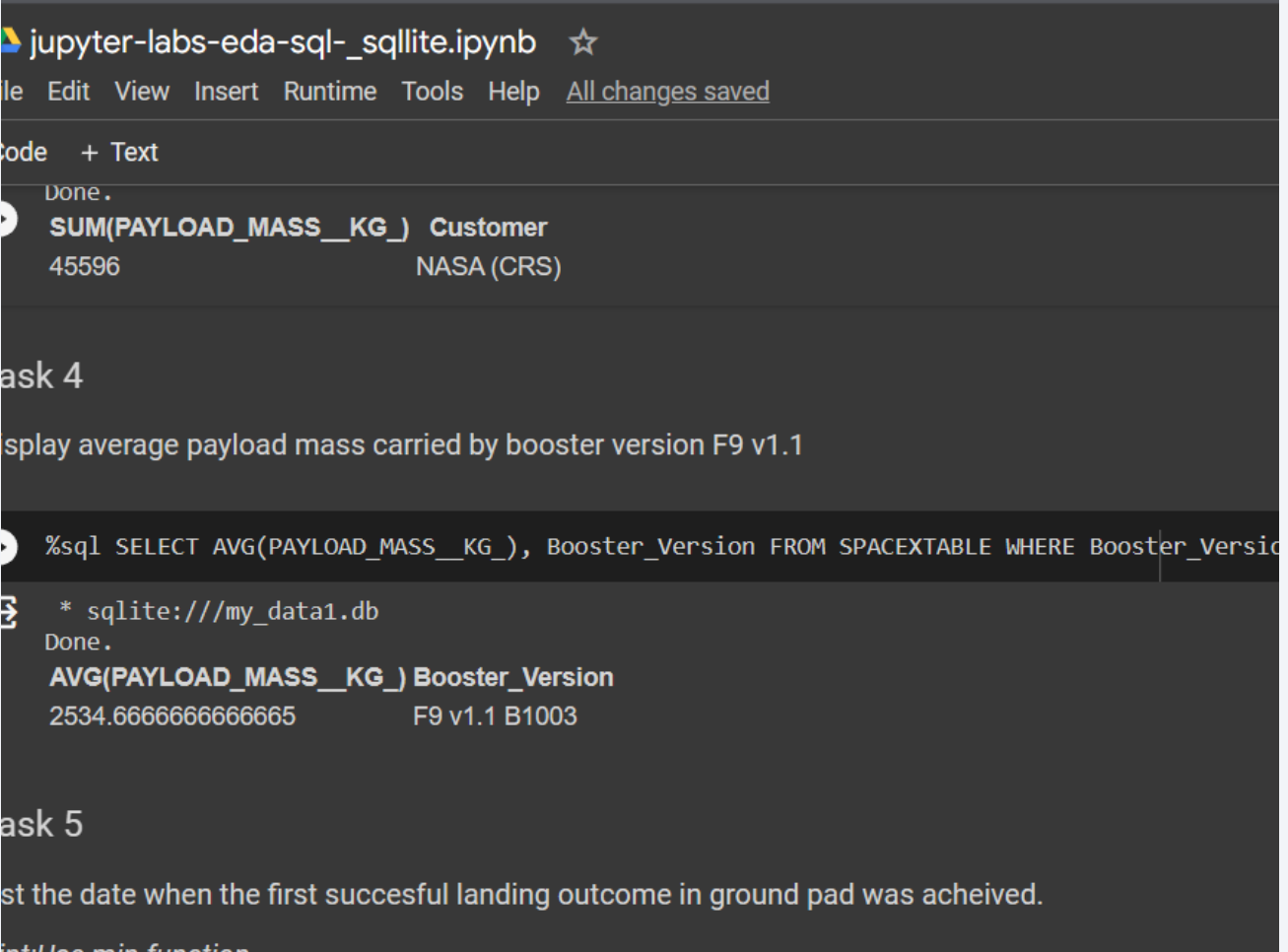
Task 3
Display the total payload mass carried by boosters launched by NASA (CRS)

%sql SELECT SUM(PAYLOAD_MASS_KG_), "Customer" FROM SPACEXTABLE WHERE Customer =
* sqlite:///my_data1.db
Done.
SUM(PAYLOAD_MASS_KG_) Customer
45596 NASA (CRS)

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

Average Payload Mass by F9 v1.1

The picture located to the right of this text shows the average payload mass carried by booster version F9 v1.1. This is achieved using sql magic, writing an sql query in python.



The screenshot shows a Jupyter Notebook interface with the title 'jupyter-labs-eda-sql-_sqlite.ipynb'. The notebook contains two code cells. The first cell shows the output of a SQL query: 'SUM(PAYLOAD_MASS_KG_) Customer' with the result '45596 NASA (CRS)'. The second cell shows the output of another SQL query: 'AVG(PAYLOAD_MASS_KG_) Booster_Version' with the result '2534.6666666666665 F9 v1.1 B1003'. The notebook also includes text prompts for tasks 4 and 5.

```
jupyter-labs-eda-sql-_sqlite.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

code + Text
Done.
SUM(PAYLOAD_MASS_KG_) Customer
45596 NASA (CRS)

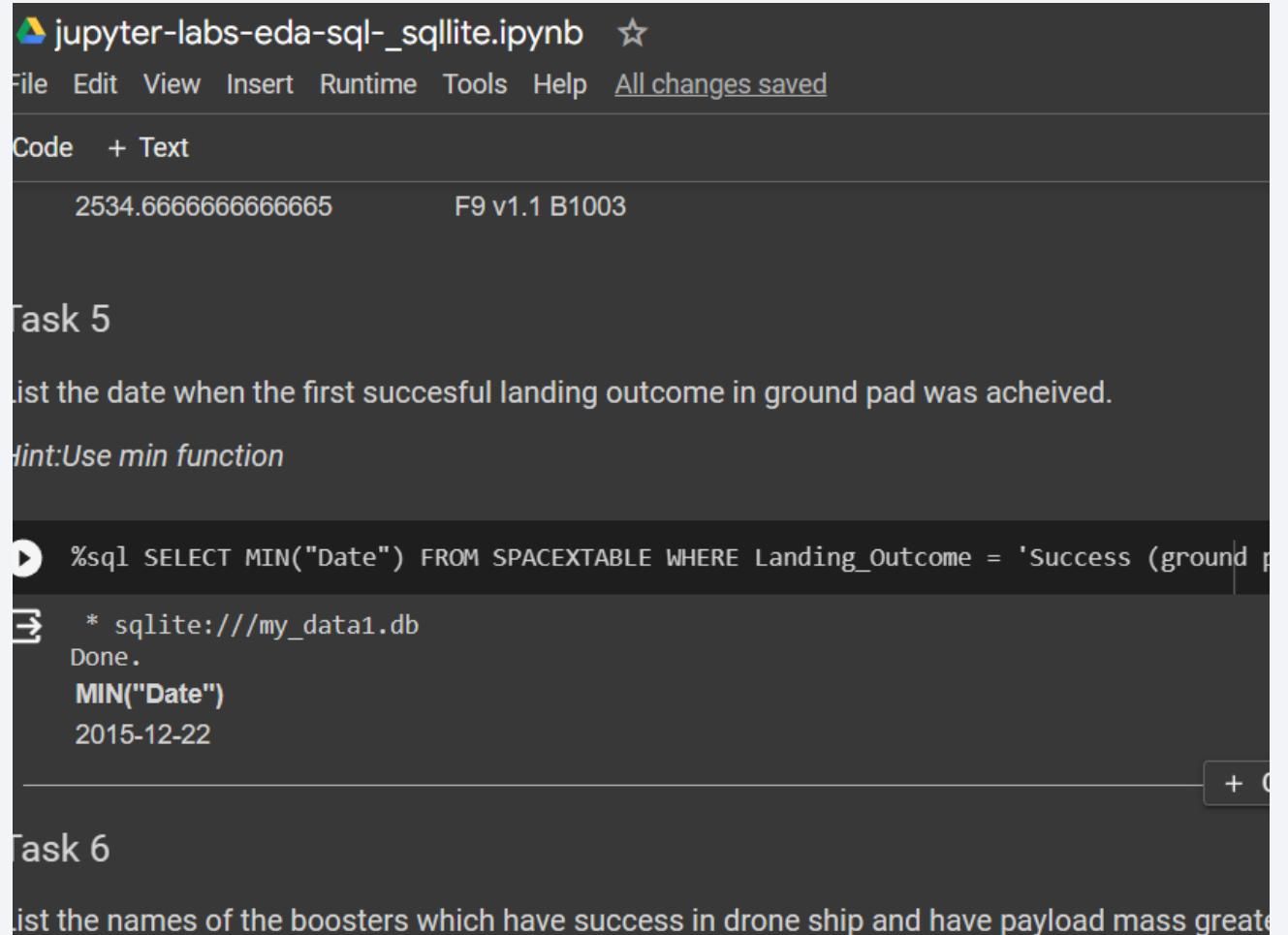
Task 4
display average payload mass carried by booster version F9 v1.1

%sql SELECT AVG(PAYLOAD_MASS_KG_), Booster_Version FROM SPACEXTABLE WHERE Booster_Version
* sqlite:///my_data1.db
Done.
AVG(PAYLOAD_MASS_KG_) Booster_Version
2534.6666666666665 F9 v1.1 B1003

Task 5
st the date when the first succesful landing outcome in ground pad was acheived.
int: Use min function
```


First Successful Ground Landing Date

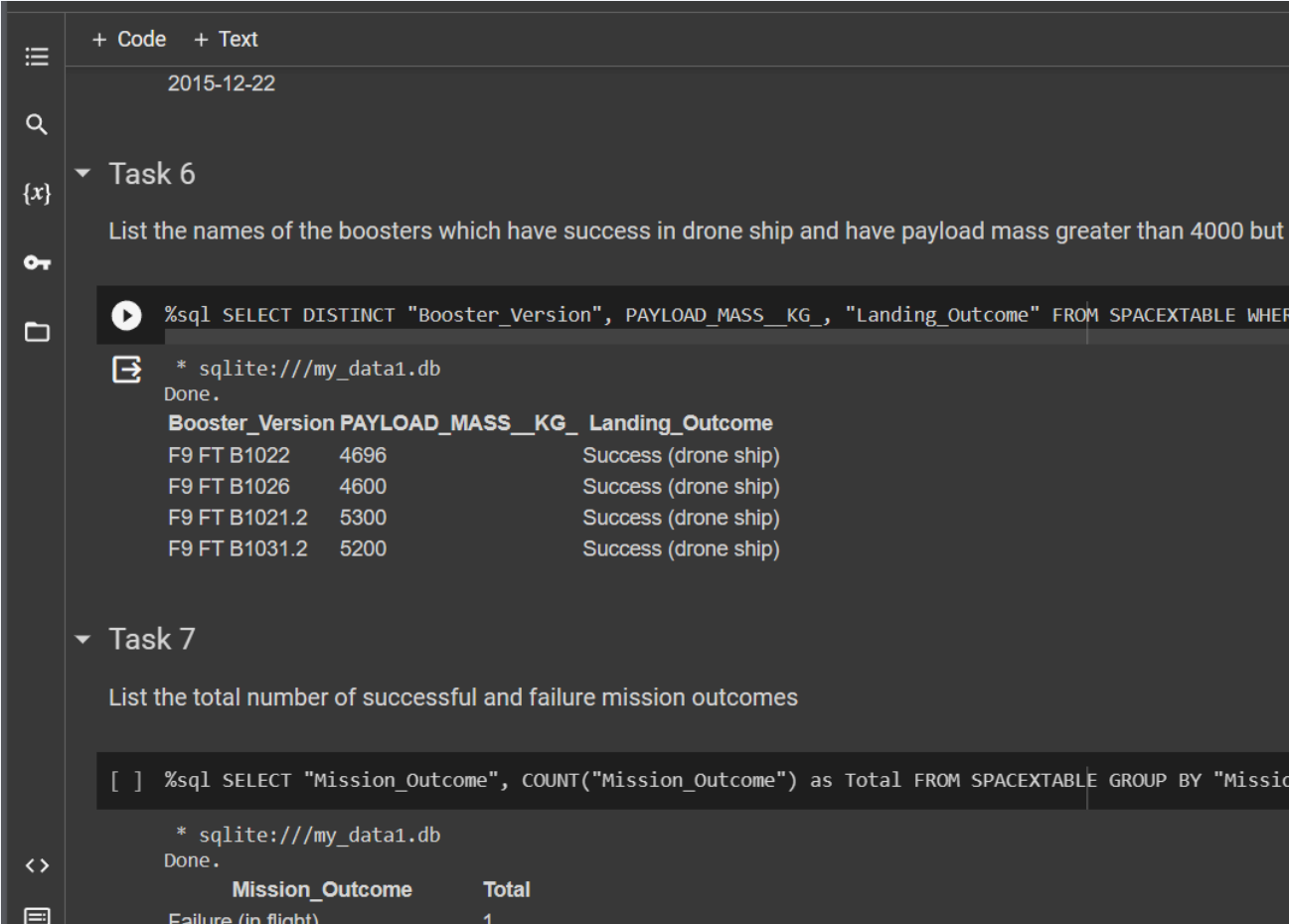
The picture located to the right of this text shows the date of the first successful ground pad landing. This is achieved using sql magic, writing an sql query in python.

A screenshot of a Jupyter Notebook interface. The top bar shows the file name 'jupyter-labs-eda-sql-_sqlite.ipynb' and a star icon. Below the top bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and 'All changes saved'. The main area is divided into two sections. The first section, titled 'Task 5', contains a text prompt: 'list the date when the first succesful landing outcome in ground pad was acheived.' followed by a hint: 'Hint: Use min function'. Below this is a code cell with the following content: a play button icon, the SQL query '%sql SELECT MIN("Date") FROM SPACEXTABLE WHERE Landing_Outcome = \'Success (ground p', a database connection string '* sqlite:///my_data1.db', and the output 'Done.', 'MIN("Date")', and '2015-12-22'. The second section, titled 'Task 6', contains a text prompt: 'list the names of the boosters which have success in drone ship and have payload mass greater'.

```
jupyter-labs-eda-sql-_sqlite.ipynb ☆  
File Edit View Insert Runtime Tools Help All changes saved  
Code + Text  
2534.6666666666665 F9 v1.1 B1003  
  
Task 5  
list the date when the first succesful landing outcome in ground pad was acheived.  
Hint: Use min function  
  
▶ %sql SELECT MIN("Date") FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground p  
* sqlite:///my_data1.db  
Done.  
MIN("Date")  
2015-12-22  
  
Task 6  
list the names of the boosters which have success in drone ship and have payload mass greater
```

Successful Drone Ship Landing with Payload between 4000 and 6000

The picture located to the right of this text shows the names of boosters which have successfully landed on drone ship, and also had a paload mass greater than 4000kg but less than 6000kg. This is achieved using sql magic, writing an sql query in python.



```
+ Code + Text
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

%sql SELECT DISTINCT "Booster_Version", PAYLOAD_MASS_KG_, "Landing_Outcome" FROM SPACEXTABLE WHERE

* sqlite:///my_data1.db
Done.
Booster_Version PAYLOAD_MASS_KG_ Landing_Outcome
F9 FT B1022      4696      Success (drone ship)
F9 FT B1026      4600      Success (drone ship)
F9 FT B1021.2    5300      Success (drone ship)
F9 FT B1031.2    5200      Success (drone ship)

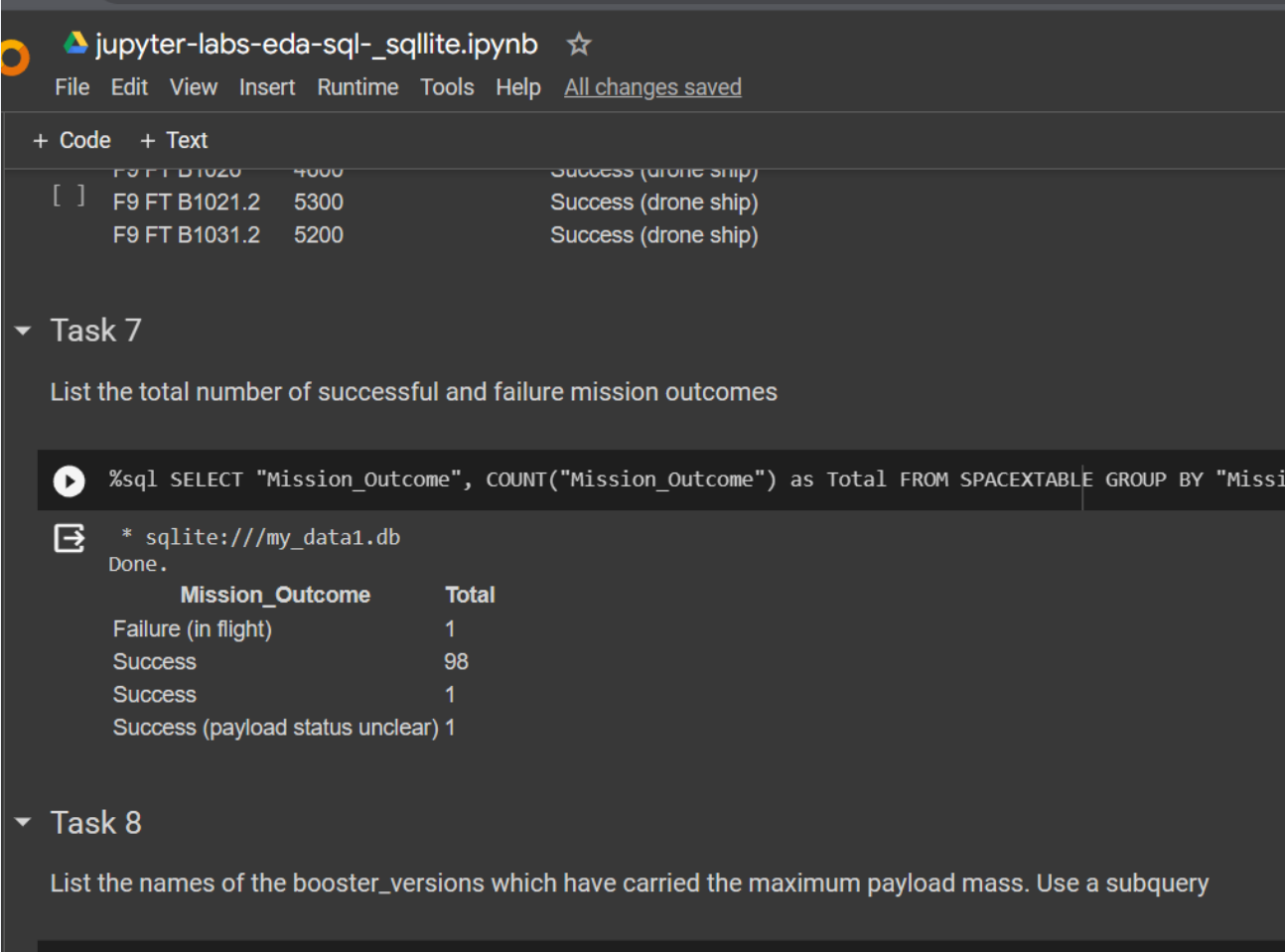
Task 7
List the total number of successful and failure mission outcomes

[ ] %sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTABLE GROUP BY "Mission_Outcome"

* sqlite:///my_data1.db
Done.
Mission_Outcome Total
Failure (in flight) 1
```

Total Number of Successful and Failure Mission Outcomes

The picture to the right displays the total number of successful and failed mission outcomes.



The screenshot shows a Jupyter Notebook titled 'jupyter-labs-eda-sql-sqlite.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar ('All changes saved'). The notebook content is organized into sections: a code cell with a table of mission data, a task section for 'Task 7', a code cell with a SQL query, a console output showing the database connection and query execution, and a table of mission outcomes, followed by another task section for 'Task 8'.

```
+ Code + Text
```

	F9 FT B1020	4000	Success (drone ship)
[]	F9 FT B1021.2	5300	Success (drone ship)
	F9 FT B1031.2	5200	Success (drone ship)

▼ Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTABLE GROUP BY "Missi
```

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

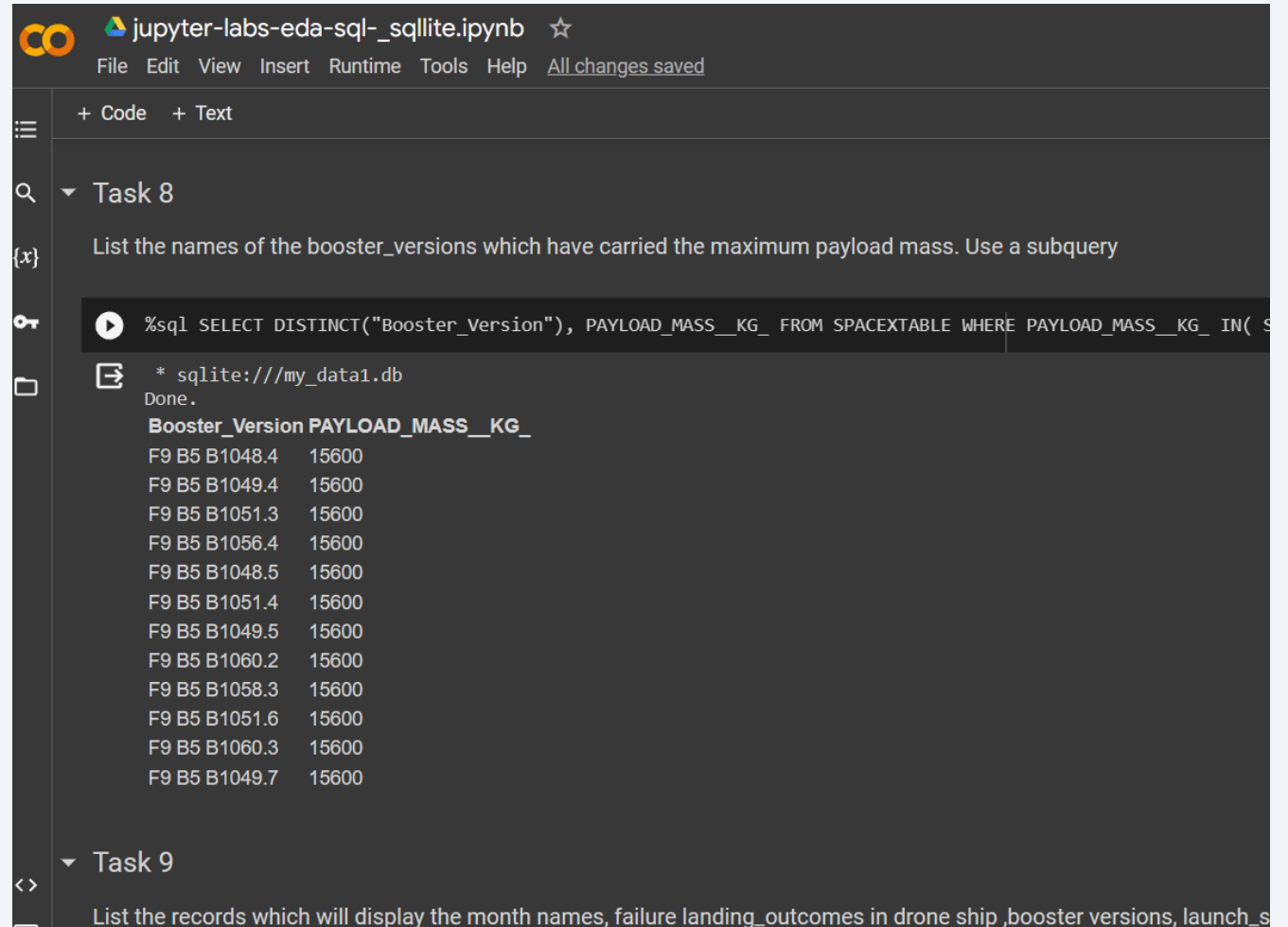
Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

▼ Task 8

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

Boosters Carried Maximum Payload

The picture to the right displays the names of booster versions that have carried the maximum payload mass. To see the full code used in achieving this, follow the GitHub URL on page 18 to go the code page.



The screenshot shows a JupyterLab interface with a file named `jupyter-labs-eda-sql-sqlite.ipynb`. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with icons for file operations. The main area displays a code cell for "Task 8" with the following content:

```
%sql SELECT DISTINCT("Booster_Version"), PAYLOAD_MASS_KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS_KG_ IN( S
```

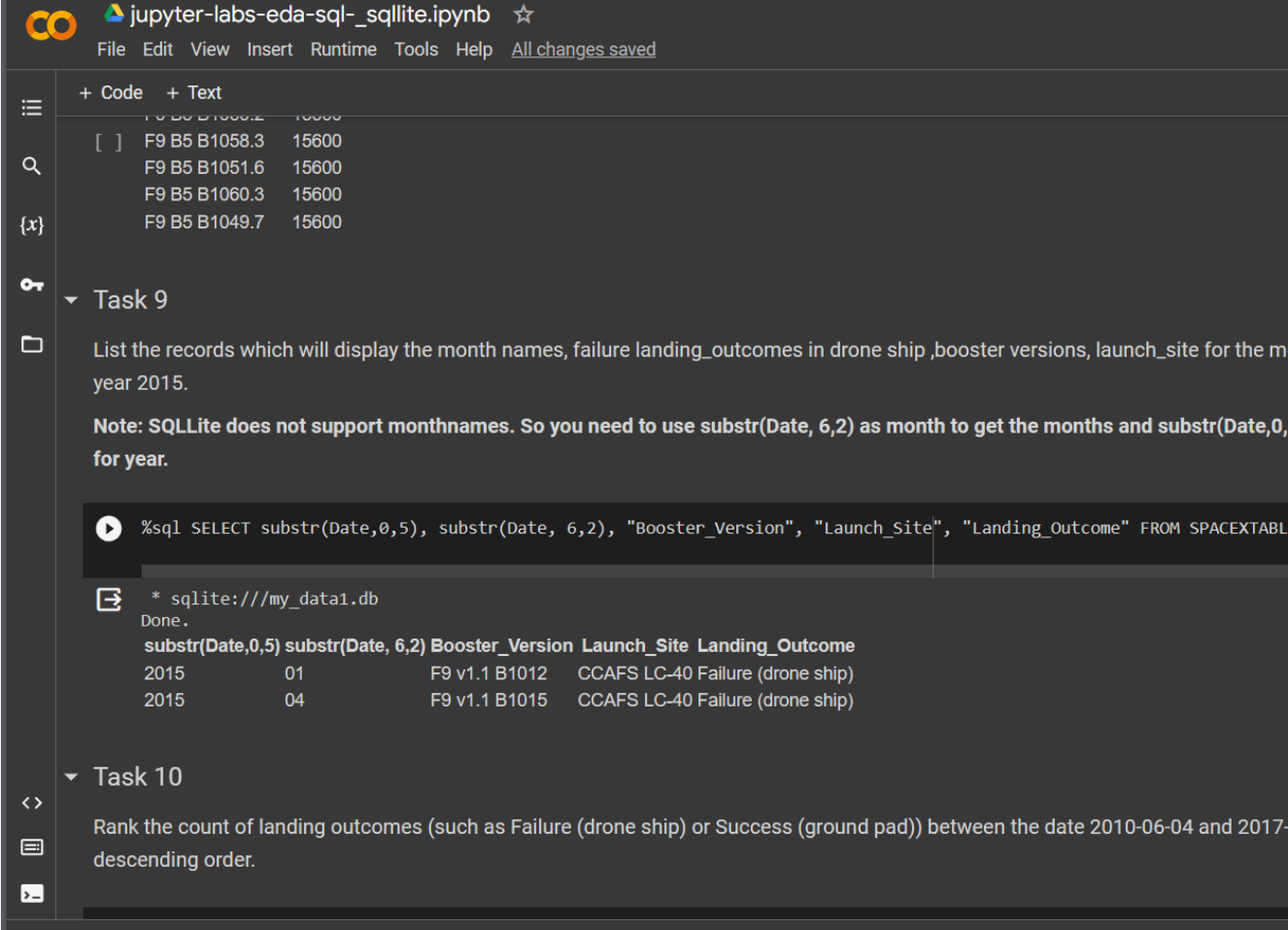
Below the code cell, the output is shown as a table with two columns: `Booster_Version` and `PAYLOAD_MASS_KG_`. The table contains 12 rows of data, all with a payload mass of 15600.

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

Below the table, the interface shows the start of "Task 9" with the instruction: "List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_s".

2015 Launch Records

The picture to the right of this text, displays the failed landing_outcomes in drone ship, their booster versions, and launch site names for the year 2015.



The screenshot shows a JupyterLab interface with a file named `jupyter-labs-eda-sql-sqlite.ipynb`. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a sidebar with icons for file explorer, search, and other tools. The main area displays a code cell with a SQLite query and its output.

Code Cell:

```
%sql SELECT substr(Date,0,5), substr(Date, 6,2), "Booster_Version", "Launch_Site", "Landing_Outcome" FROM SPACESTABLERECORDS
```

Output:

```
* sqlite:///my_data1.db
Done.
substr(Date,0,5) substr(Date, 6,2) Booster_Version Launch_Site Landing_Outcome
2015            01            F9 v1.1 B1012    CCAFS LC-40 Failure (drone ship)
2015            04            F9 v1.1 B1015    CCAFS LC-40 Failure (drone ship)
```

Task 9:

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

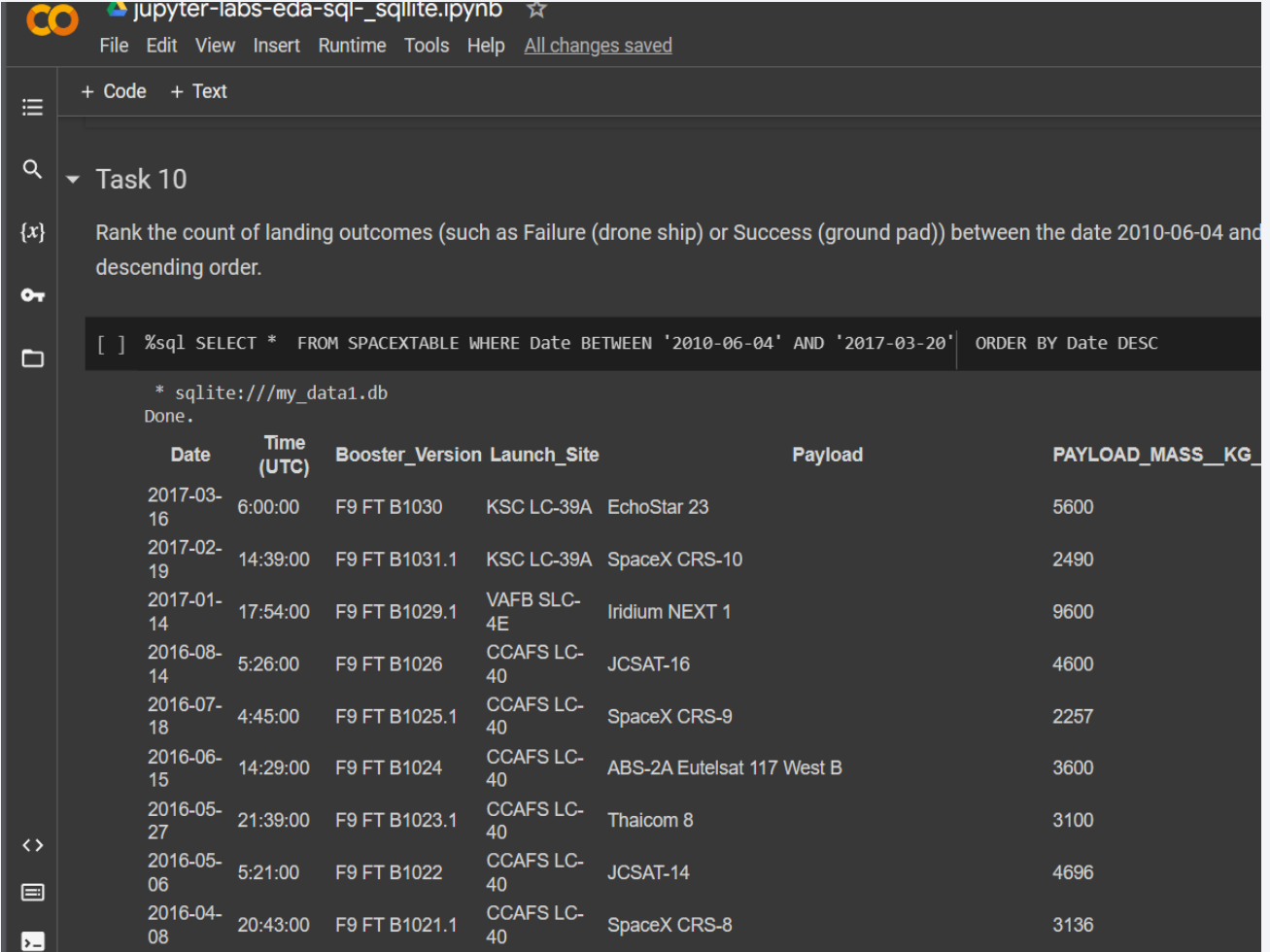
Note: SQLite does not support monthnames. So you need to use `substr(Date, 6,2)` as month to get the months and `substr(Date,0,5)` for year.

Task 10:

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

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

The picture on this page displays the Landing Outcomes between 2010-06-04 and 2017-03-20, to see the full list of output, go to the GitHub URL on page 18.



The screenshot shows a JupyterLab interface with a file named 'jupyter-labs-eda-sql-sqlite.ipynb'. The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with '+ Code' and '+ Text' buttons. A sidebar on the left contains icons for file explorer, search, and other functions. The main area displays 'Task 10' with the instruction: 'Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and descending order.' Below this, a SQL query is shown in a code cell: `%sql SELECT * FROM SPACEXTABLE WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' ORDER BY Date DESC`. The output of the query is a table with the following data:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS_KG
2017-03-16	6:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490
2017-01-14	17:54:00	F9 FT B1029.1	VAFB SLC-4E	Iridium NEXT 1	9600
2016-08-14	5:26:00	F9 FT B1026	CCAFS LC-40	JCSAT-16	4600
2016-07-18	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257
2016-06-15	14:29:00	F9 FT B1024	CCAFS LC-40	ABS-2A Eutelsat 117 West B	3600
2016-05-27	21:39:00	F9 FT B1023.1	CCAFS LC-40	Thaicom 8	3100
2016-05-06	5:21:00	F9 FT B1022	CCAFS LC-40	JCSAT-14	4696
2016-04-08	20:43:00	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136

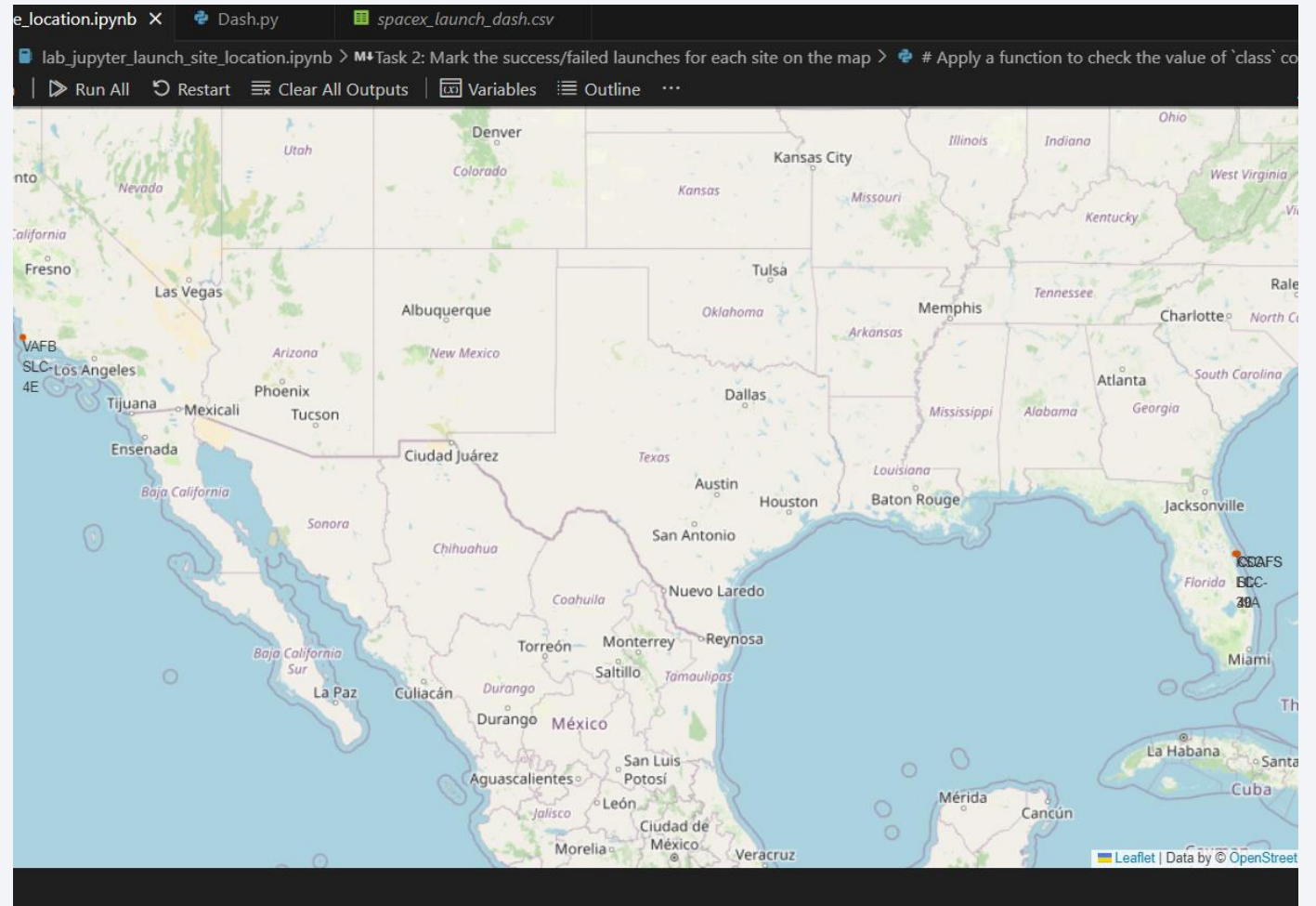
A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

Launch Sites Proximities Analysis

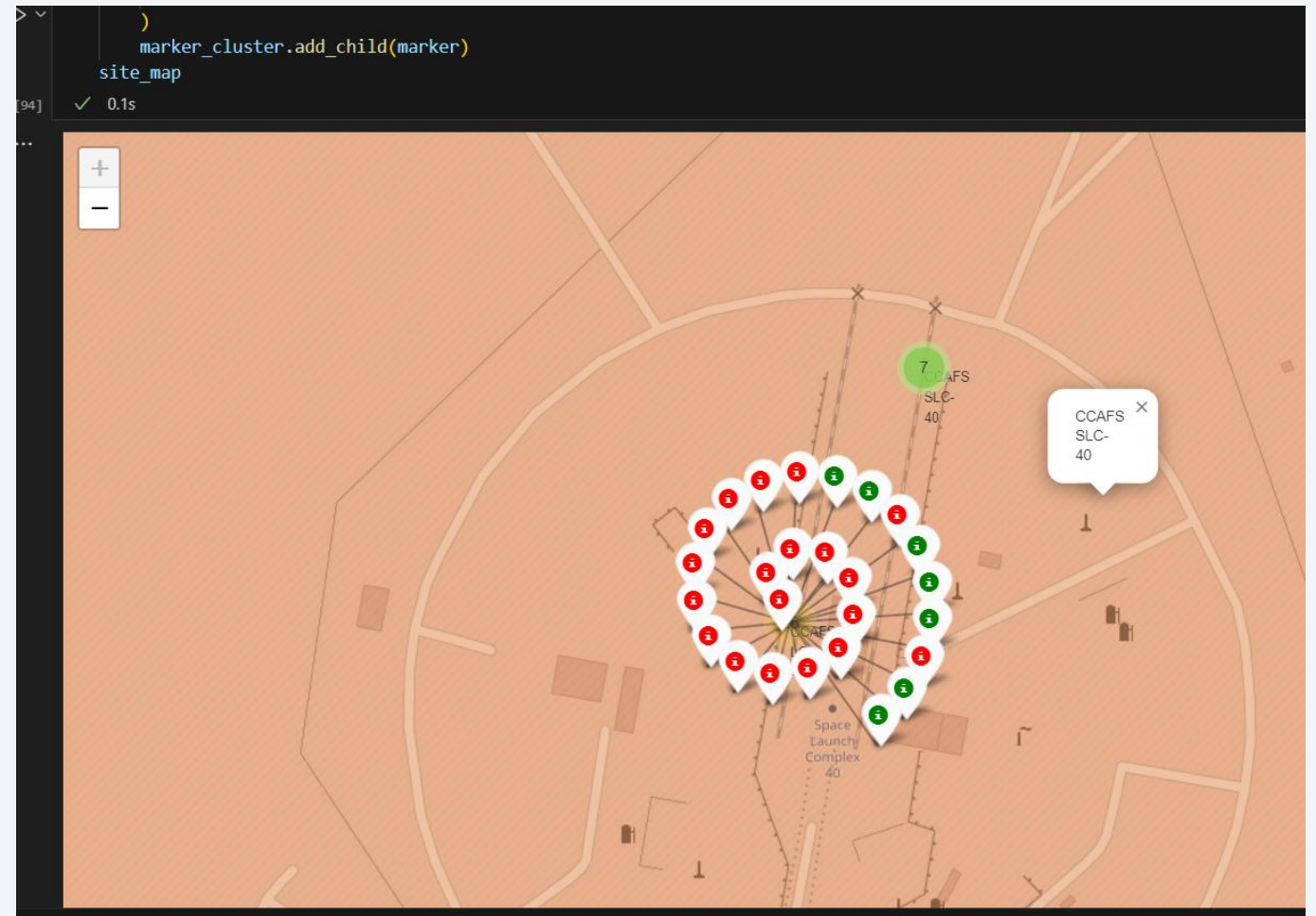
Marked all the Launch Sites on the Map with Folium

The generated map to the right shows the location markers of all the launch sites with a red dot indication. From the map we can see that all the launch sites are situated in USA, and are all close to the coast.



Marked the successful/failed launches for each site

The picture in this slide shows the color-labeled launch outcomes on the map. The red labeled markers indicate the failed launches while the green labeled markers indicate the successful ones.

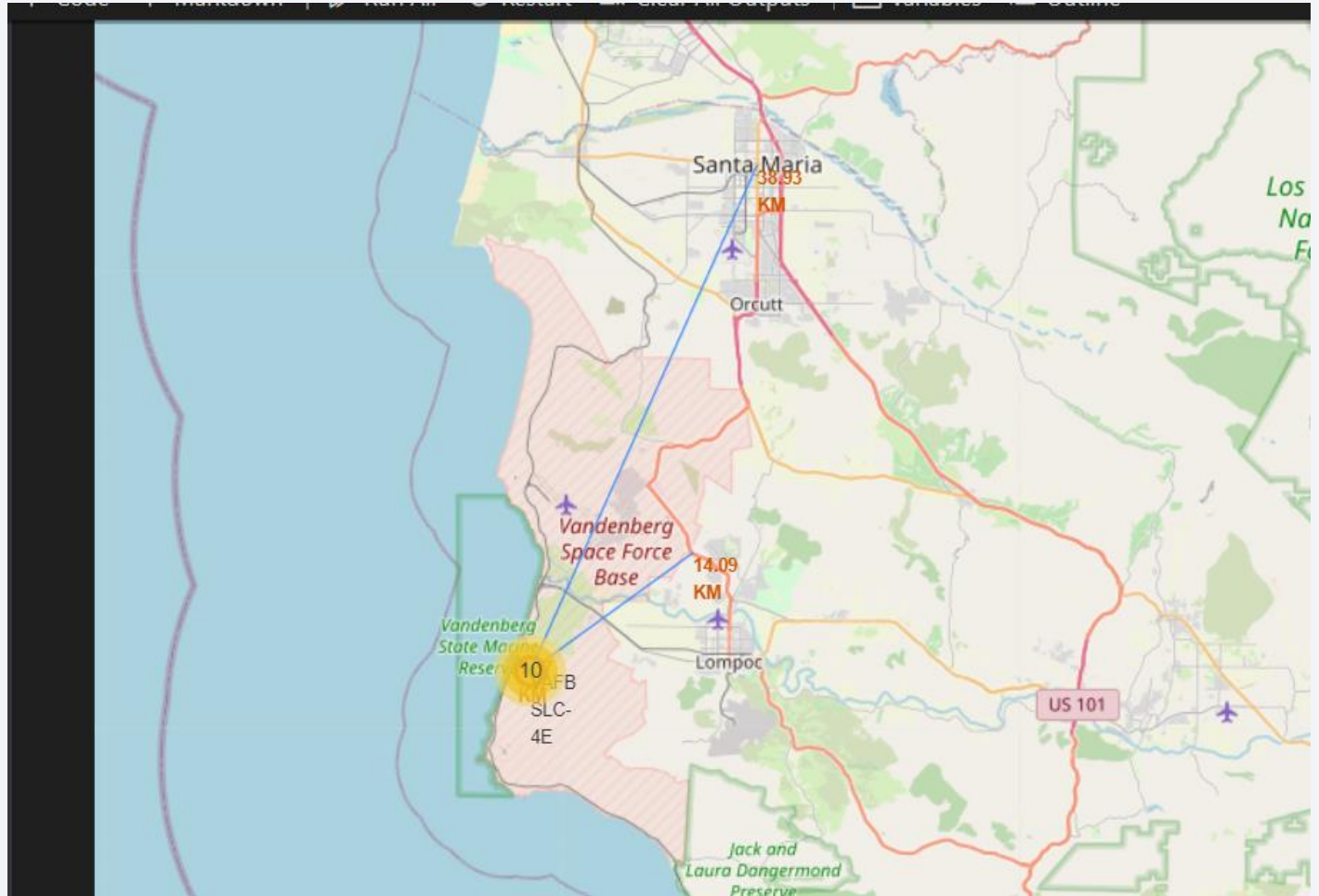


Calculated the distance between a Launchsite and specific structures

There are three distances that were calculated, the shortest distance is to the closest railway line. This is visible in the screenshot to the right of this text. This is because the distance is small, it can only be seen when the map has been zoomed in.

From this map, we can see that the launch site is situated far away from the highway, and even farther away from the city.

Follow the URL on page 20 to view the raw python file that was used to create these folium maps.

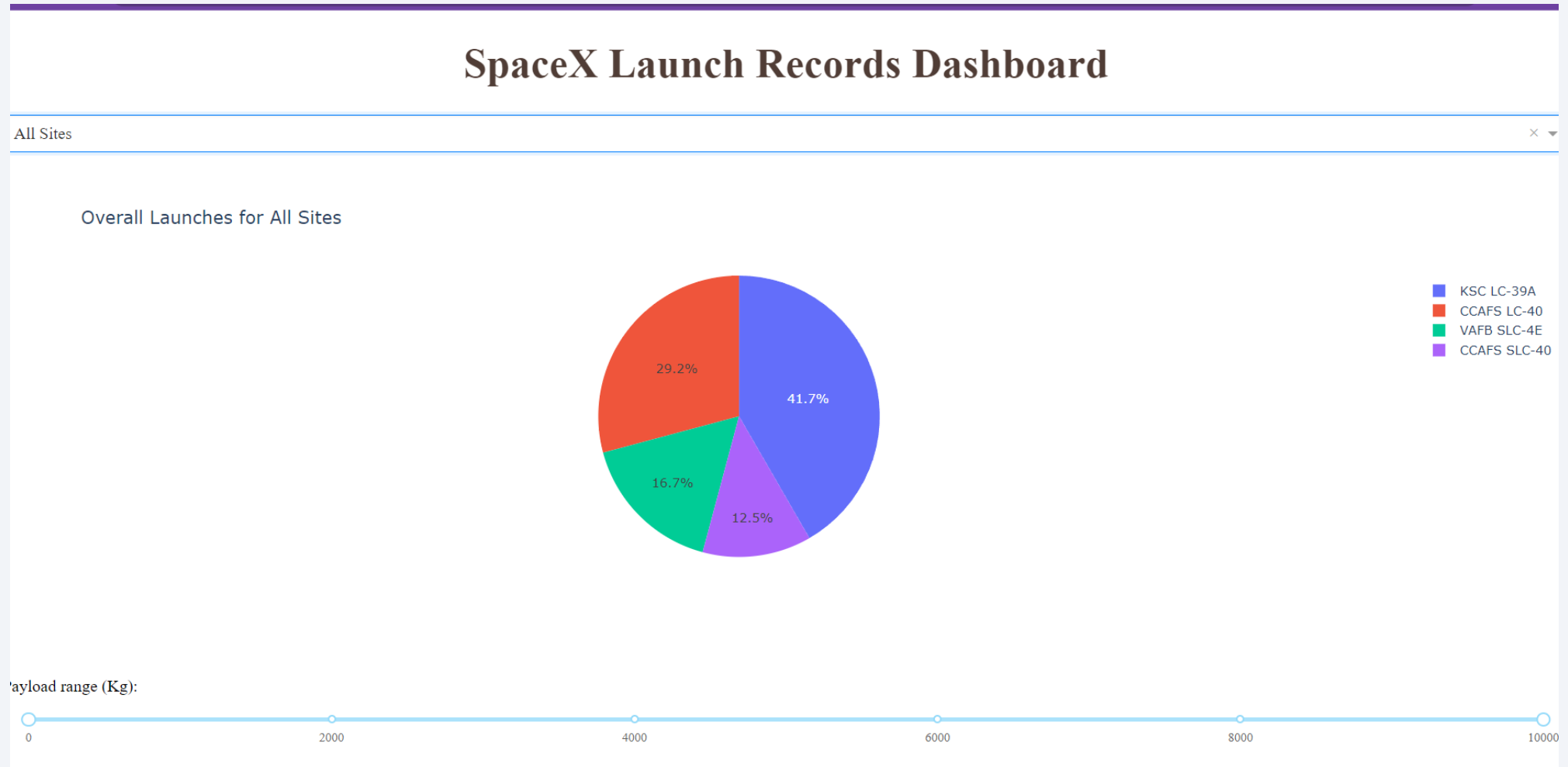




Section 4

Build a Dashboard with Plotly Dash

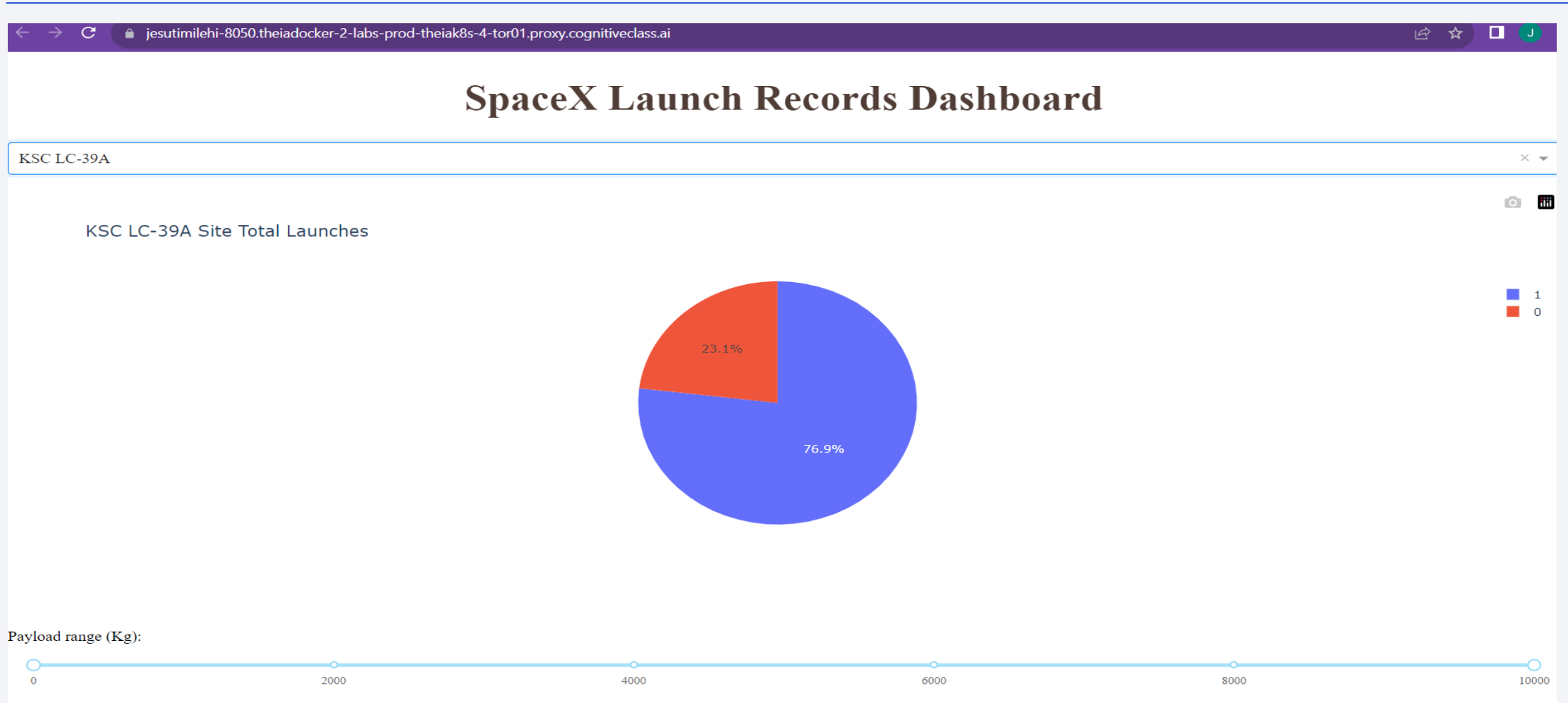
Pie chart for the launch success count for all sites.



Pie chart for the launch success count for all sites.

The picture in the previous slide shows the Launch success count for the various Launch sites in the Space X space mission. From the picture we see that launch site 'KSC LC-39A' has the highest proportion of successful launches out of all the launch sites.

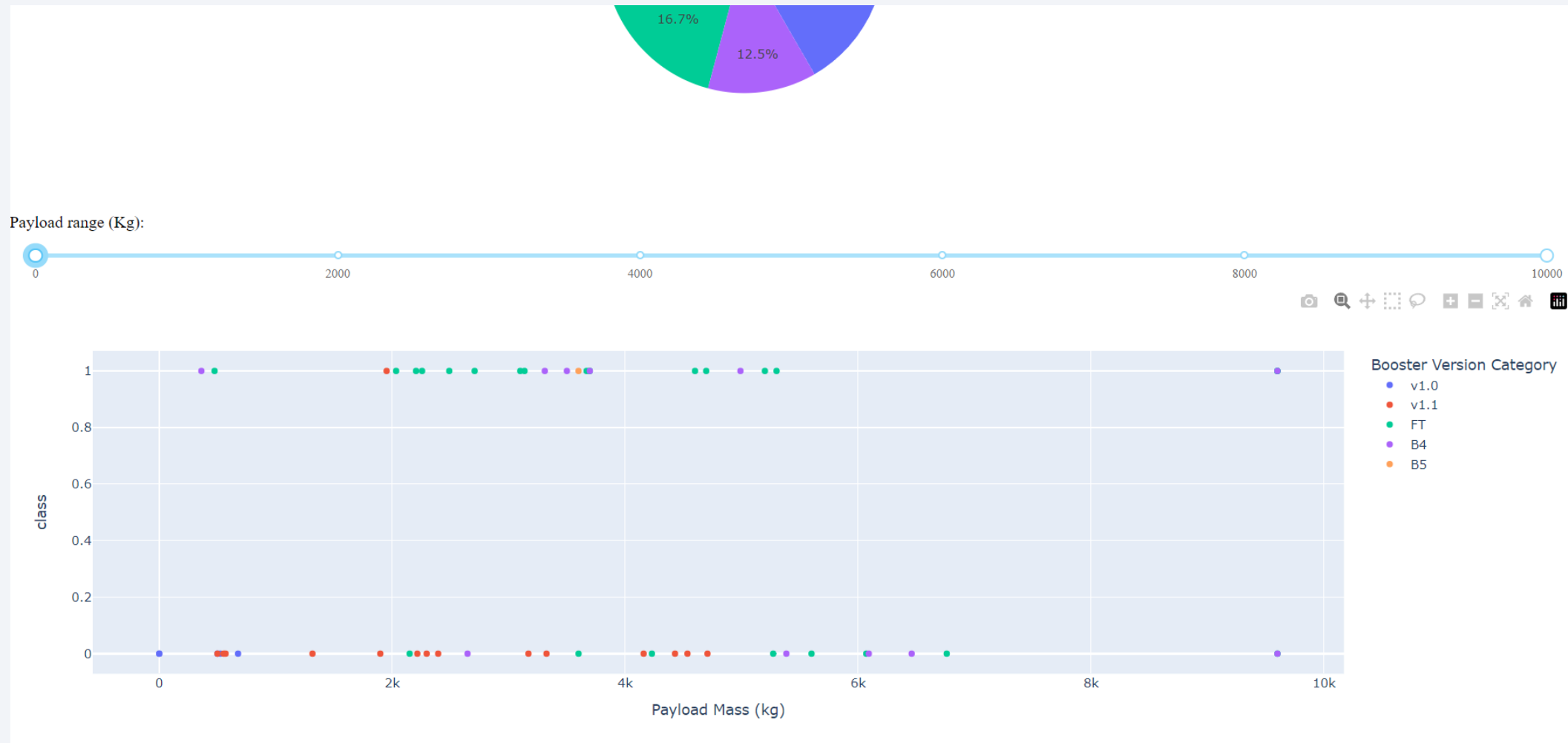
Piechart for the launchsite with the highest launch success ratio



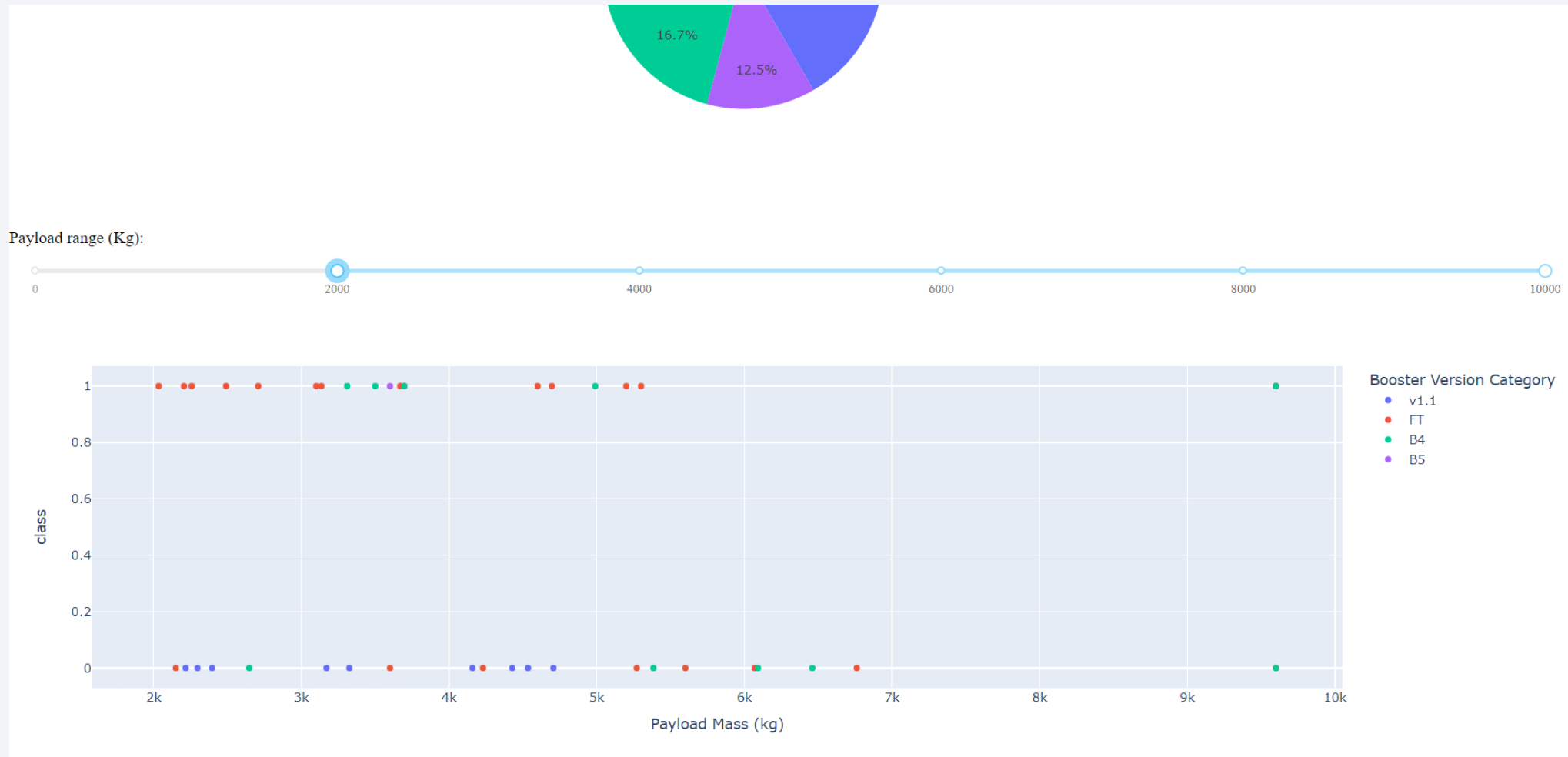
Piechart for the Launchsite with the highest launch success ratio

The picture in the previous slide shows the pie chart indicating the success/failure percentage of the launch site with the highest launch success ratio. This is 'KSC LC-39A'.

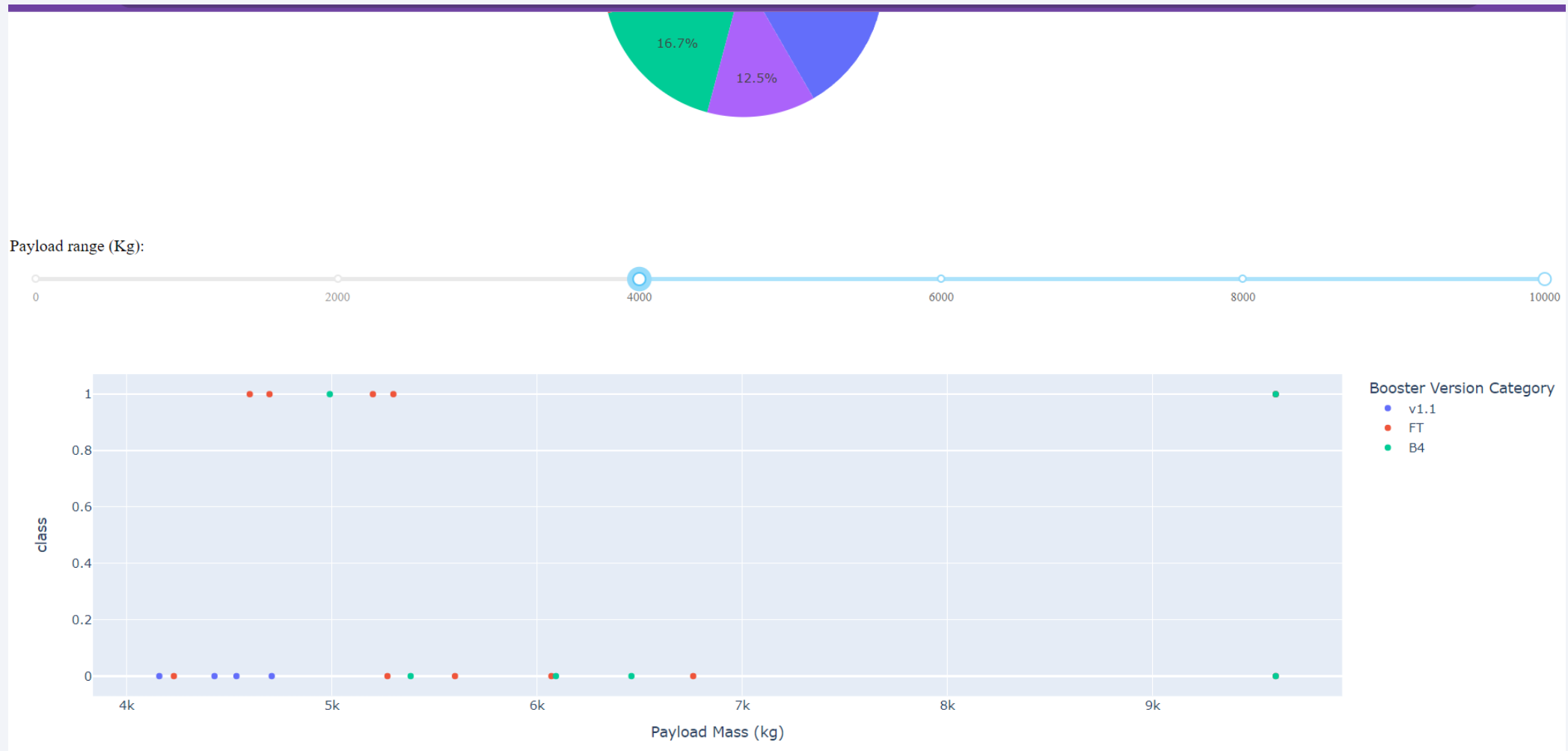
Payload vs Launch Outcome/class



Payload vs Launch Outcome/class



Payload vs Launch Outcome/class



Payload vs Launch Outcome/class

The three preceding pictures are screenshots of the dashboard showing a scatterplot of Payload vs Launch Outcome for the space mission.

The Payload Range that has the highest success rate is 8000-10000kg with 50% success rate.

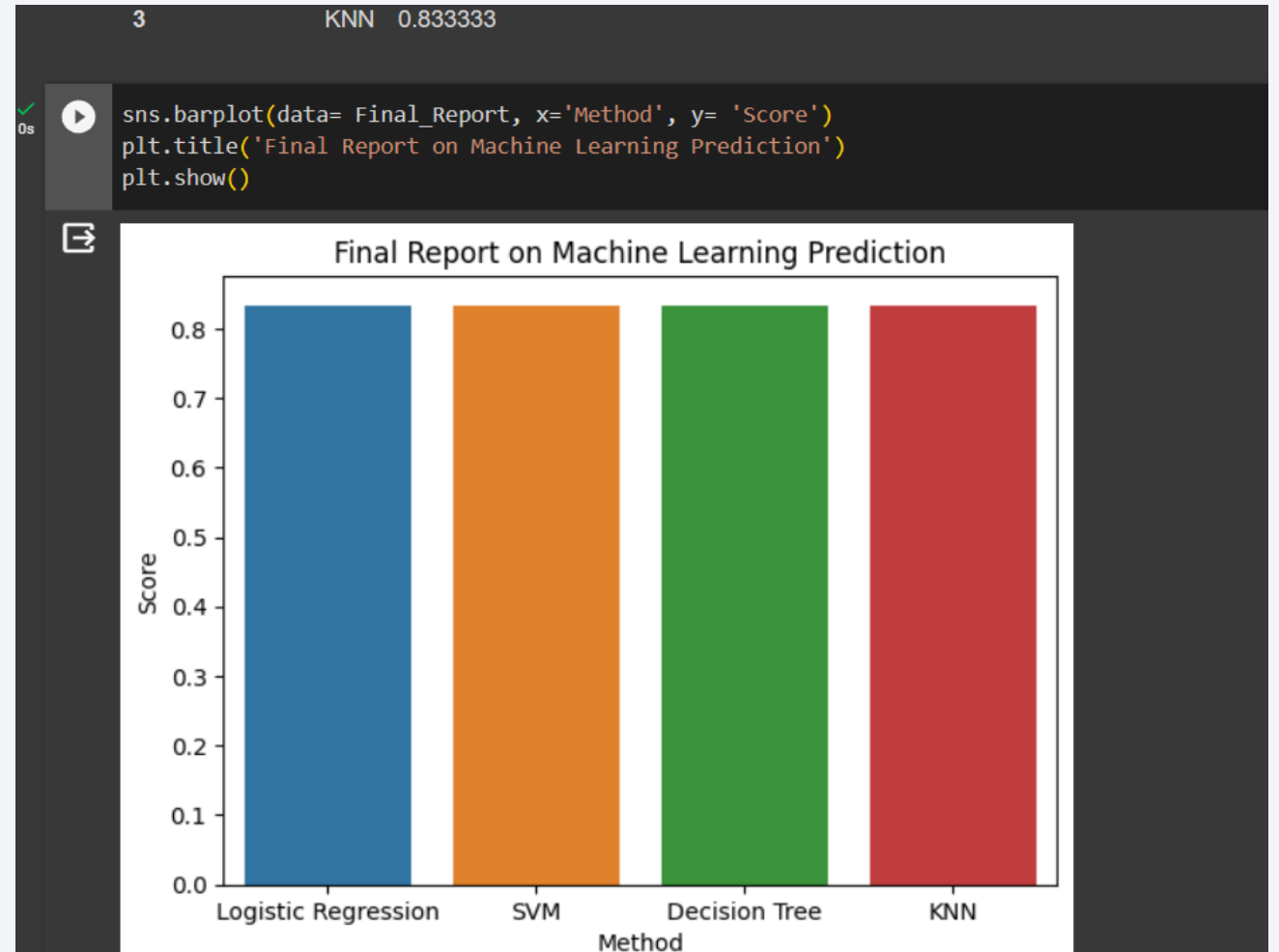
The booster version with the largest success rate is FT.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

All the four models used had the same accuracy score on the test data, which is 0.833 or 83.33%. So technically, there is no 'best performing model'.

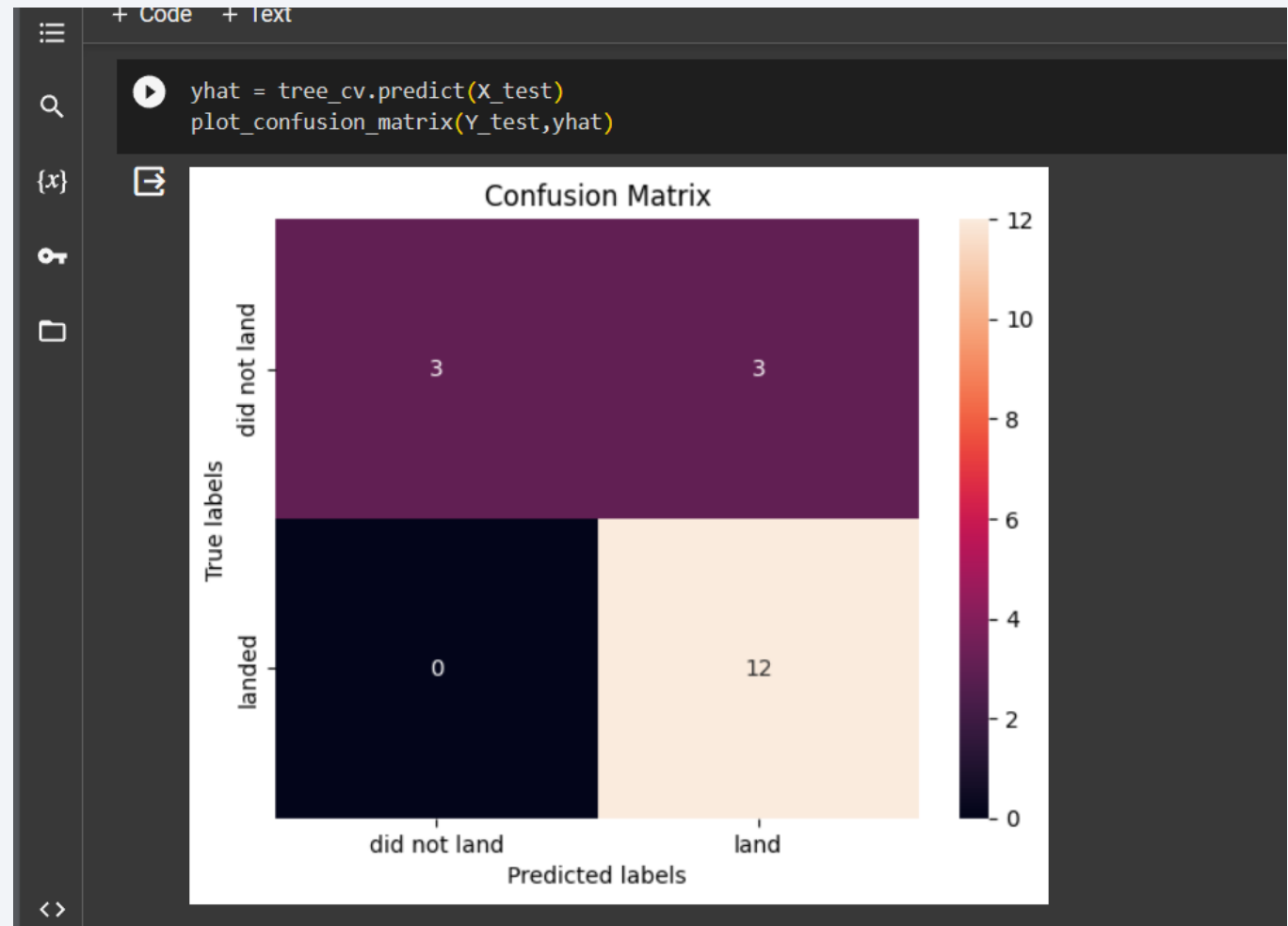


Confusion Matrix

All the four models used had the same accuracy score on the test data, so there is technically no 'best performing model'. But I have chosen the confusion matrix of the Decision tree model here because, out of all the models, it had the best training score.

The URL address to the complete code is this:

[https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5_\(so_lved\).ipynb](https://github.com/Jesutimilehin-Onayemi/Capstone-Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5_(so_lved).ipynb)



Conclusions

1. From the insights drawn from the analysis on page 33, we can see that Rocket Launching should get cheaper in terms of cost, because with time, the success rate of first stage landing has gone up. T Hence, the possibility of re-using the first stage is higher.
2. SSO is the most successful orbit type, so the more this is used, the more likely, that the company will re-use the first stage.
3. KSC LC-39A is the most successful Launches, a further in depth research can be done to find out the reason for this, maybe it's the nature of the wind speed in that location, a research will help unfold this.
4. The most successful booster version, is 'FT'.

Conclusions

In this project, the aim was to determine if SpaceX will reuse the first stage, I have however used Machine Learning prediction to predict this, and the link to the code for creating the model is available on page 58.

So, from the findings in this project, we see that with time, Space X is expected to reuse more of its launches' first stage, they are also more likely to reuse the first stage if; Sun Synchronous Orbit or SSO is used as the orbit type, the launches are carried out are at KSC LC-39A launch site, and if FT is used as the Booster Version.

Appendix

To see the complete code to each of the sections in this presentation, follow the respective URLs to the code page.

Thank you!

