IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nikolaos Doumouras
19nd December 2022

Outline

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

- Project background and context

  Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

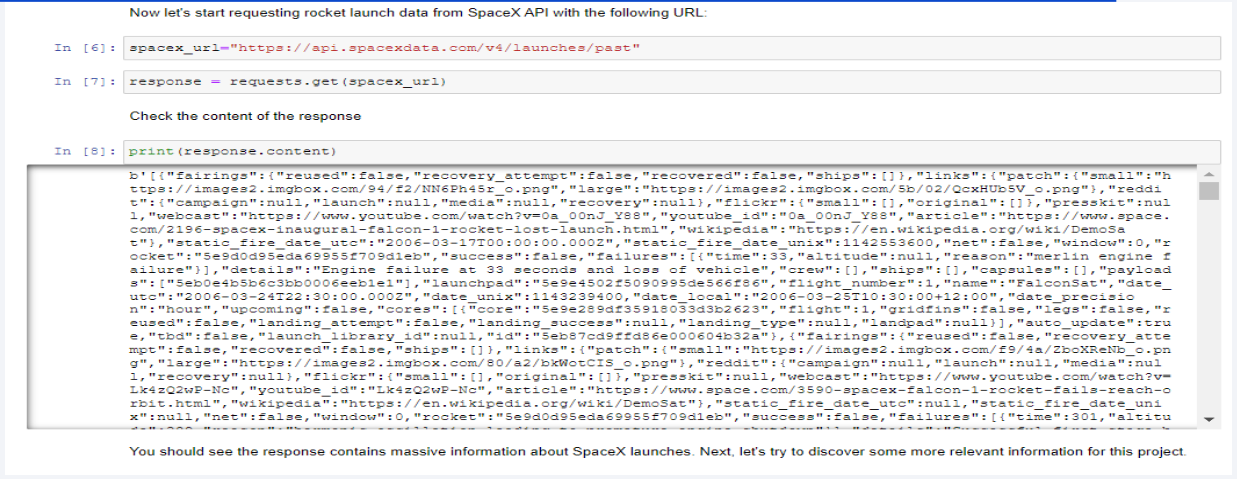
- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is https://github.com/Dmnik14/Applied-Data-Science-Capstone-SpaceX/blob/main/Space%20X%20Assignment_data%20collection.ipynb



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

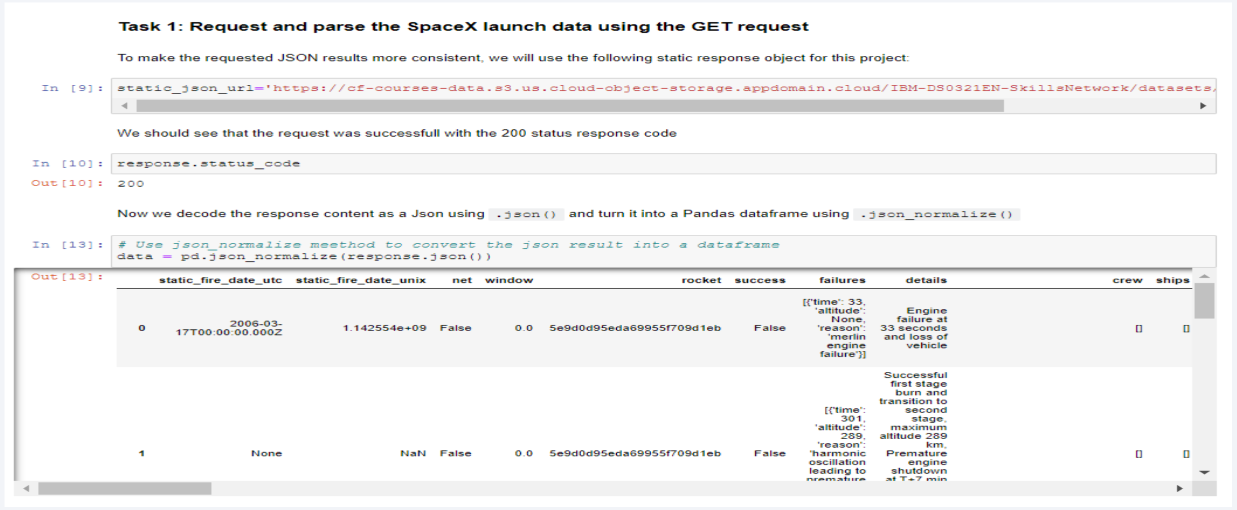
```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

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

Check the content of the response

```
In [8]: print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"h
ttps://images2.imgbox.com/94/f2/NN6Ph45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddi
t":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":nul
l,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.
com/2196-spacex-inaugural-falcon-1-rocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSa
t"},"static_fire_date_utc":"2006-03-17T00:00:00.000Z","static_fire_date_unix":1142553600,"net":false,"window":0,"r
ocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":33,"altitude":null,"reason":"merlin engine f
ailure"}],"details":"Engine failure at 33 seconds and loss of vehicle","crew":[],"ships":[],"capsules":[],"payload
s":["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flight_number":1,"name":"FalconSat","date_
utc":"2006-03-24T22:30:00.000Z","date_unix":1143239400,"date_local":"2006-03-25T10:30:00+12:00","date_precisio
n":"hour","upcoming":false,"cores":[{"core":"5e9e289df35918033d3b2623","flight":1,"gridfins":false,"legs":false,"r
eused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_update":tru
e,"tbd":false,"launch_library_id":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reused":false,"recovery_atte
mpt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/f9/4a/ZboXReNb_o.pn
g","large":"https://images2.imgbox.com/80/a2/bkWotCIS_o.png"},"reddit":{"campaign":null,"launch":null,"media":nul
l,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=
Lk4zQ2wP-Nc","youtube_id":"Lk4zQ2wP-Nc","article":"https://www.space.com/3590-spacex-falcon-1-rocket-fails-reach-o
rbit.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":null,"static_fire_date_uni
x":null,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":301,"altitu
```

You should see the response contains massive information about SpaceX launches. Next, let's try to discover some more relevant information for this project.



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:

```
In [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 successfull with the 200 status response code

```
In [10]: response.status_code
Out[10]: 200
```

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

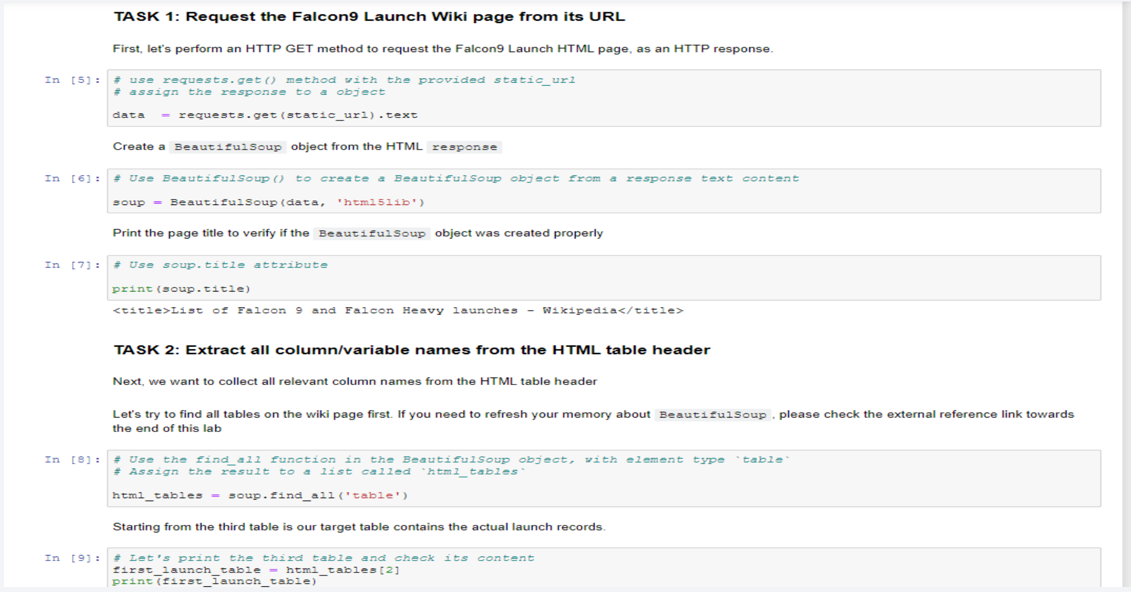
```
In [13]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

Out[13]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min	[]	[]

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/Dmnik14/Applied-Data-Science-Capstone-SpaceX/blob/main/Space%20X%20Assignment_data%20wrangling.ipynb



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

        data  = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content

        soup = BeautifulSoup(data, 'html5lib')
```

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

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

        print(soup.title)

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

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

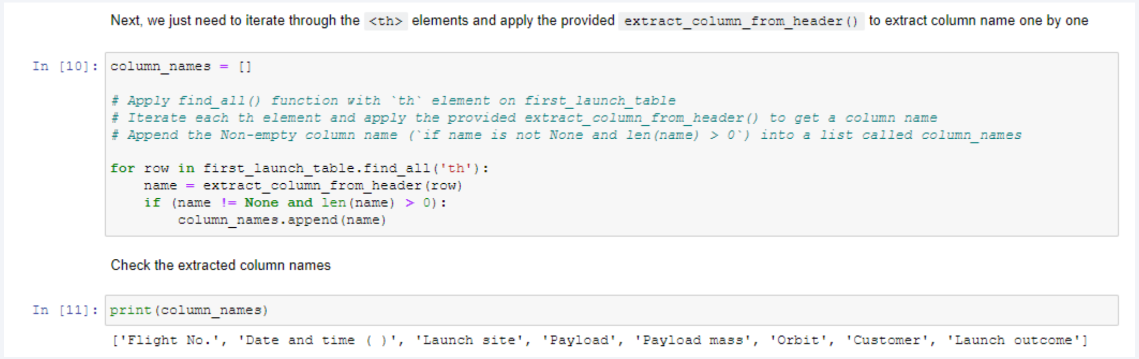
Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup` , please check the external reference link towards the end of this lab

```
In [8]: # Use the find_all function in the BeautifulSoup object, with element type `table`
        # Assign the result to a list called `html_tables`

        html_tables = soup.find_all('table')
```

Starting from the third table is our target table contains the actual launch records.

```
In [9]: # Let's print the third table and check its content
        first_launch_table = html_tables[2]
        print(first_launch_table)
```



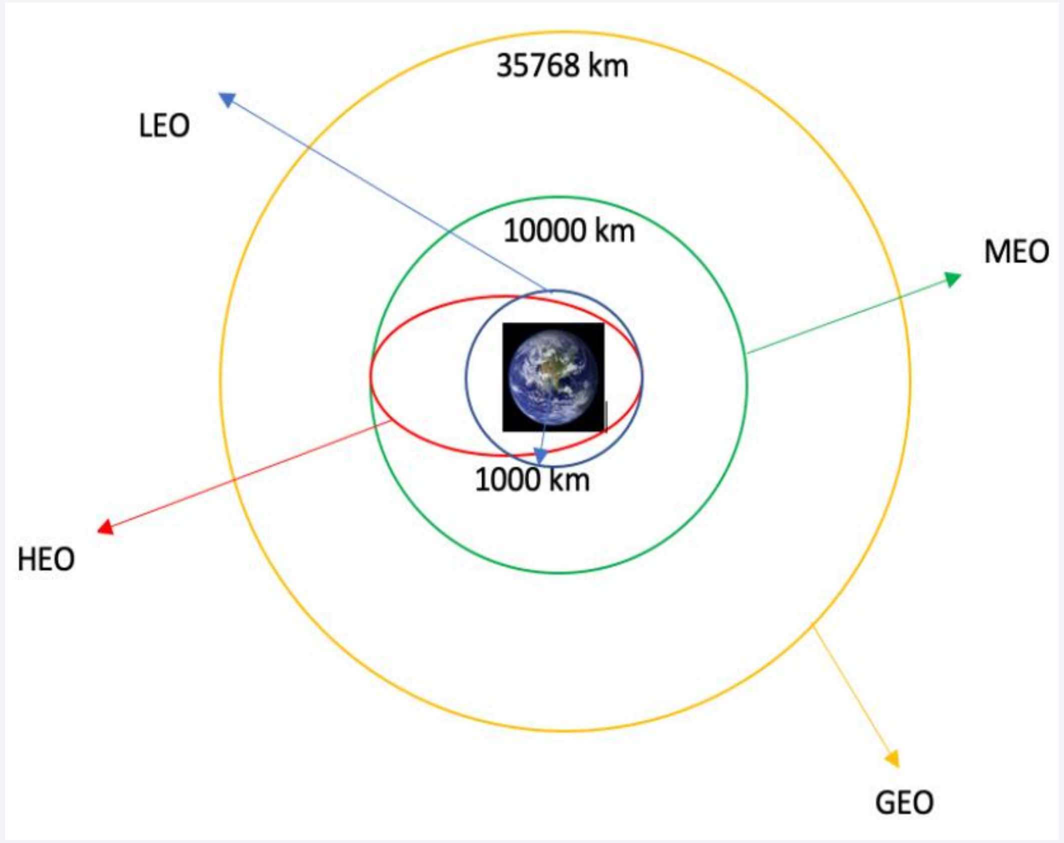
Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

```
In [10]: column_names = []

         # Apply find_all() function with `th` element on first_launch_table
         # Iterate each th element and apply the provided extract_column_from_header() to get a column name
         # Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

         for row in first_launch_table.find_all('th'):
             name = extract_column_from_header(row)
             if (name != None and len(name) > 0):
                 column_names.append(name)
```

Check the extracted column names

```
In [11]: print(column_names)

         ['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

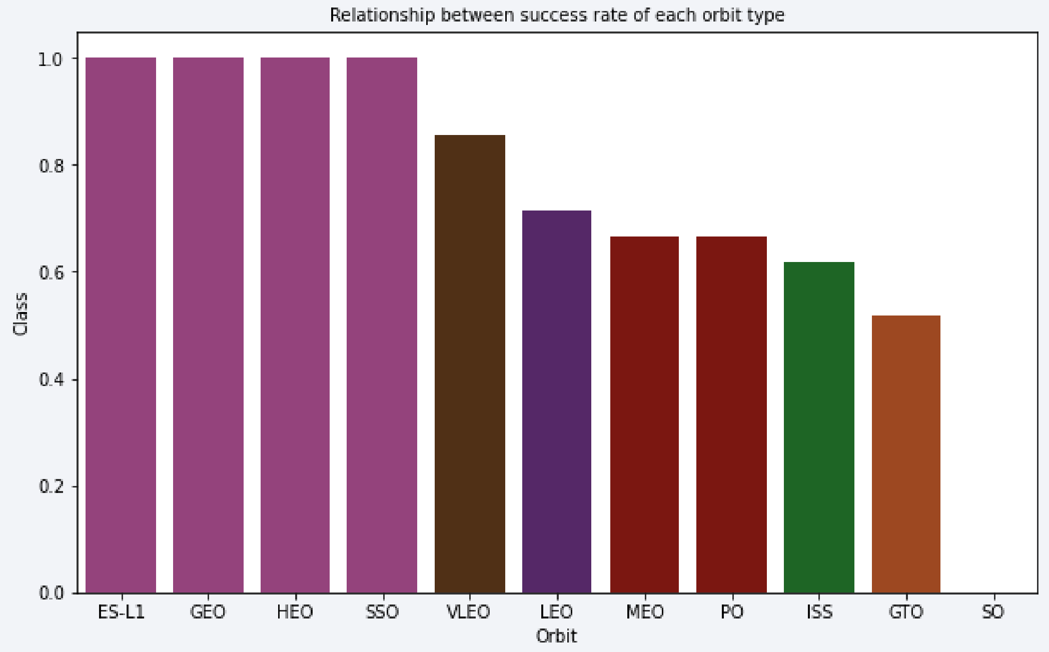
Data Wrangling

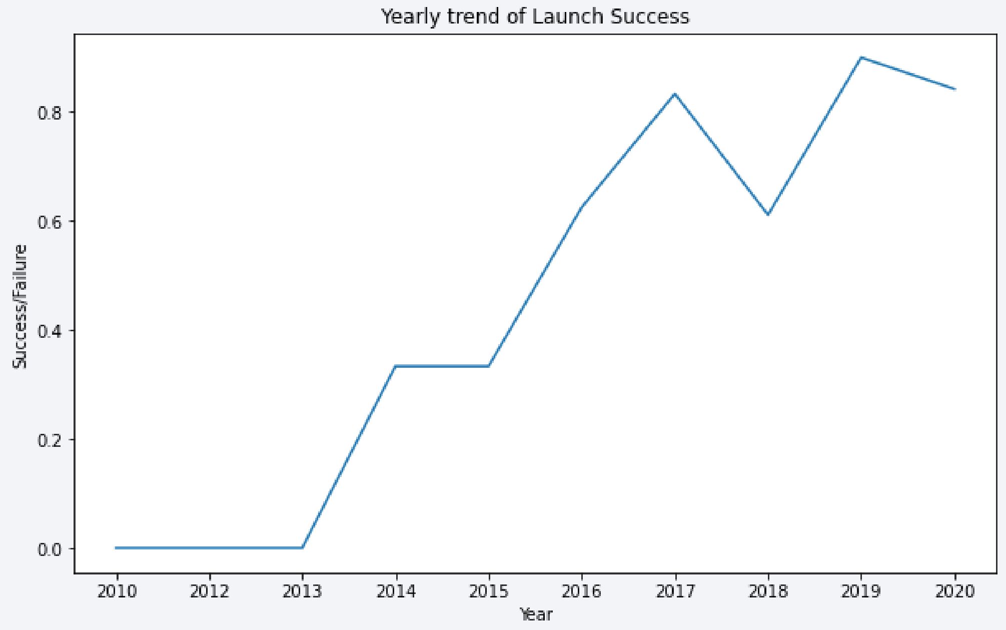


- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/Dmnik14/Applied-Data-Science-Capstone-SpaceX/blob/main/Data%20Wrangling_EDA.ipynb

EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.





- The link to the notebook is https://github.com/Dmnik14/Applied-Data-Science-Capstone-SpaceX/blob/main/EDA%20with%20Data%20Visualization.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is https://github.com/Dmnik14/Applied-Data-Science-Capstone-SpaceX/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is https://github.com/Dmnik14/Applied-Data-Science-Capstone-SpaceX/blob/main/SpaceX_app.ipynb

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is https://github.com/Dmnik14/Applied-Data-Science-Capstone-SpaceX/blob/main/SpaceX_Machine%20Learning%20Prediction.ipynb

Results

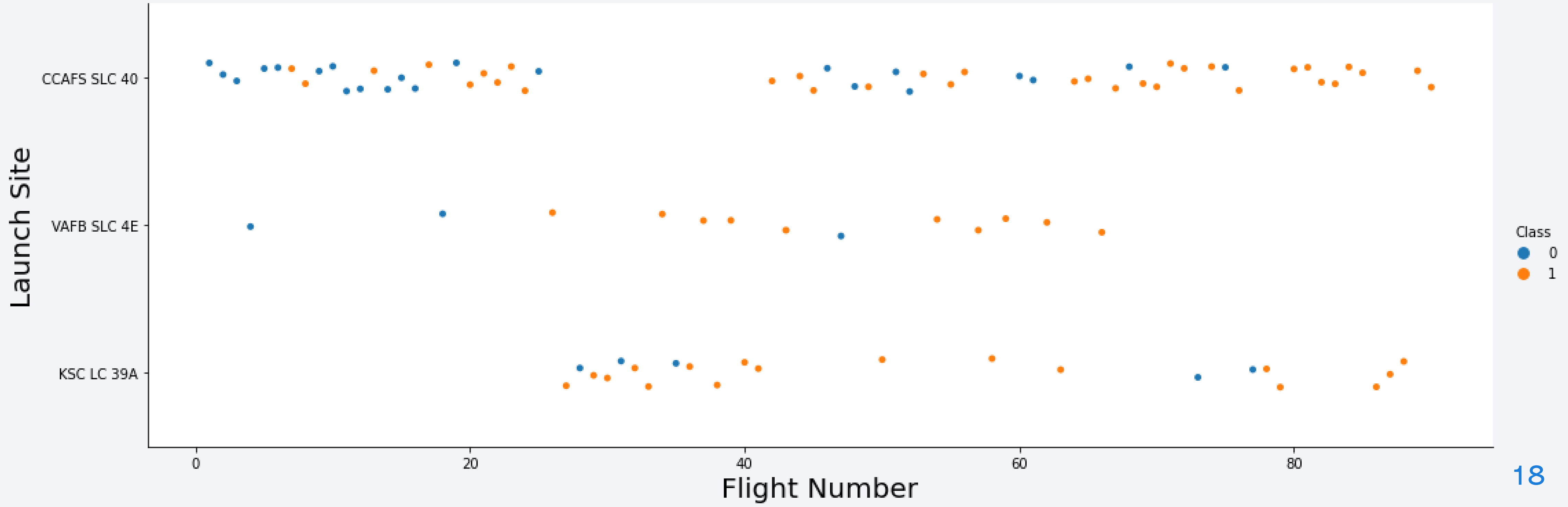
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

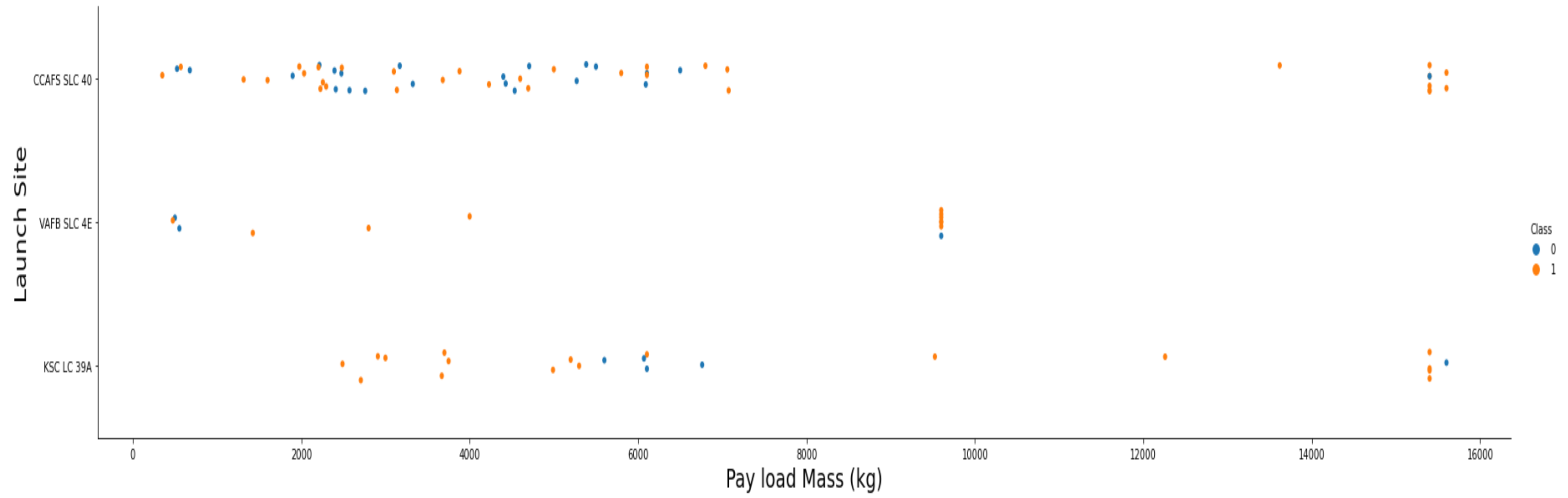
Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



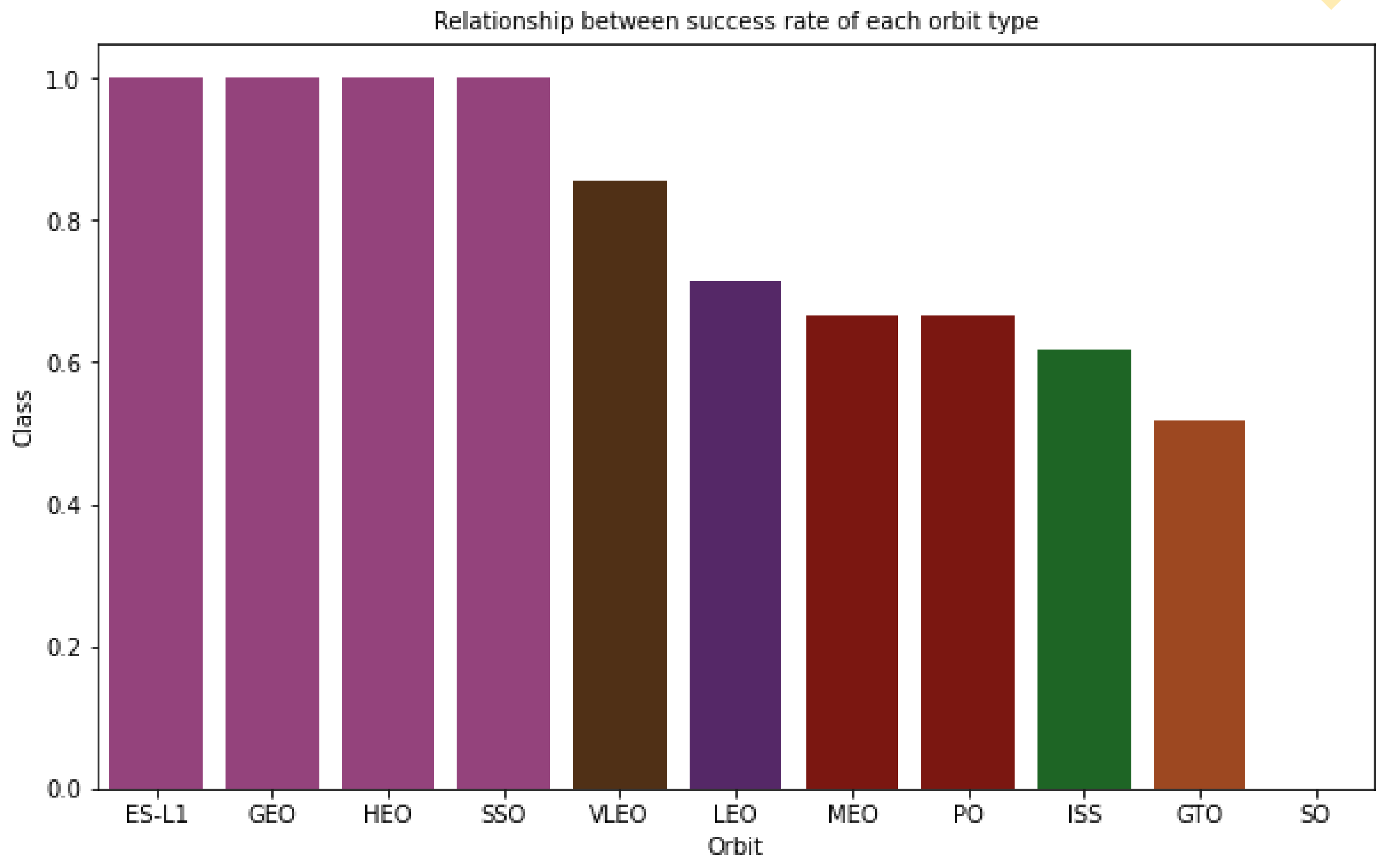
Payload vs. Launch Site

The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



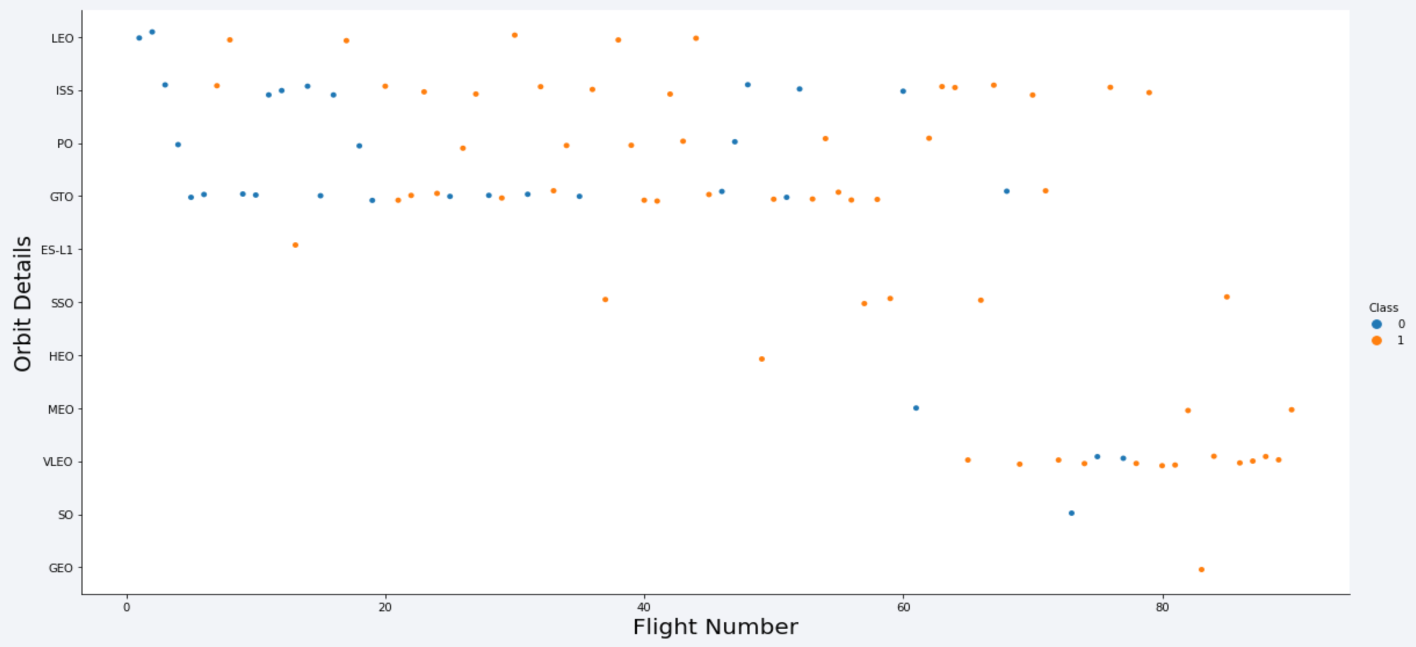
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



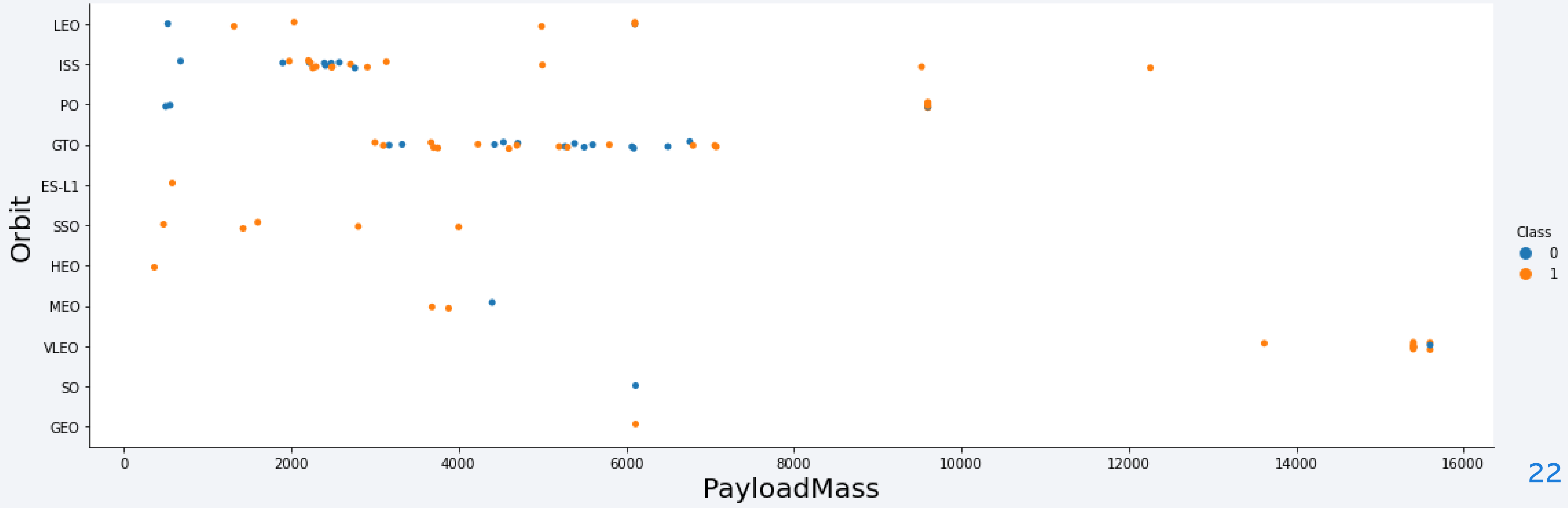
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



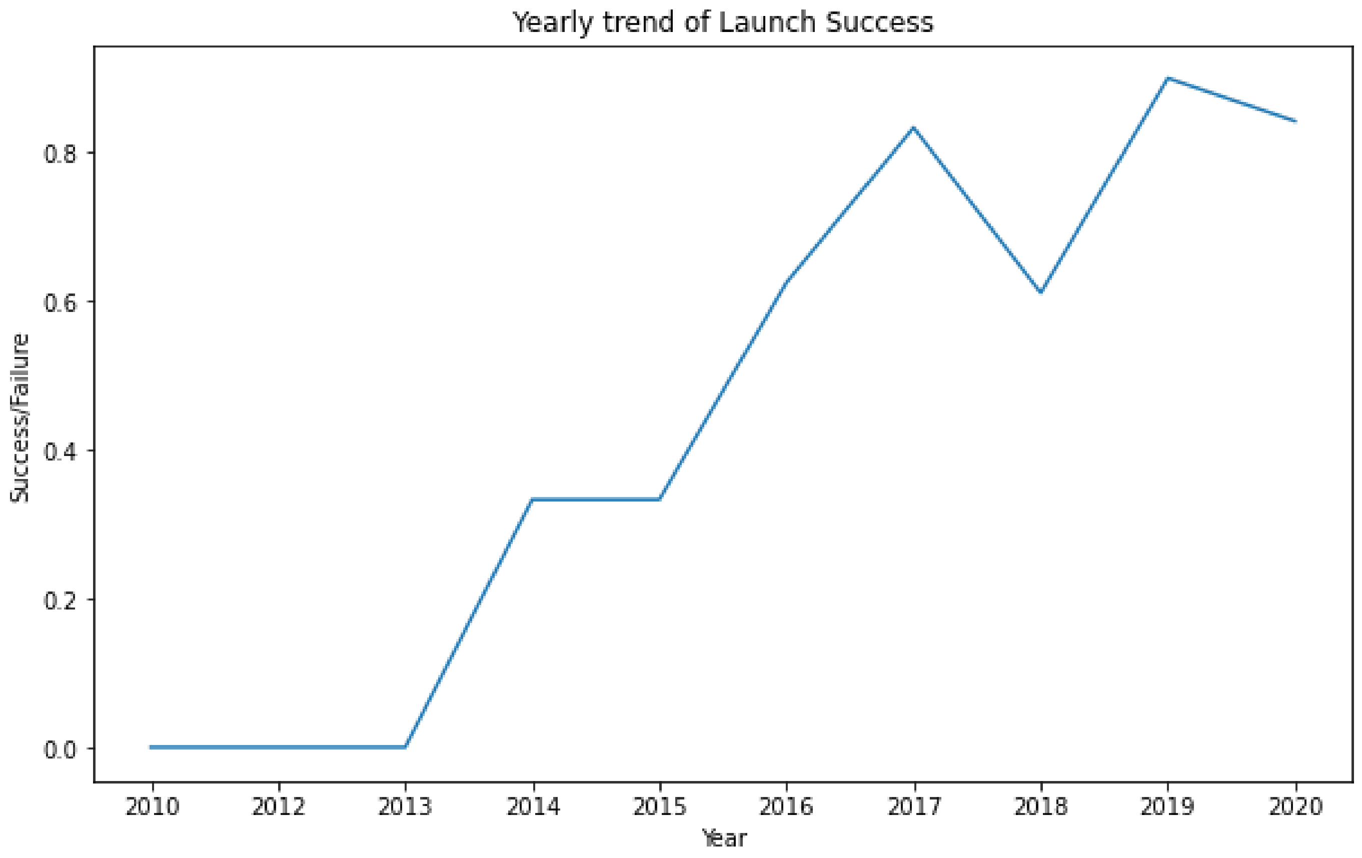
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



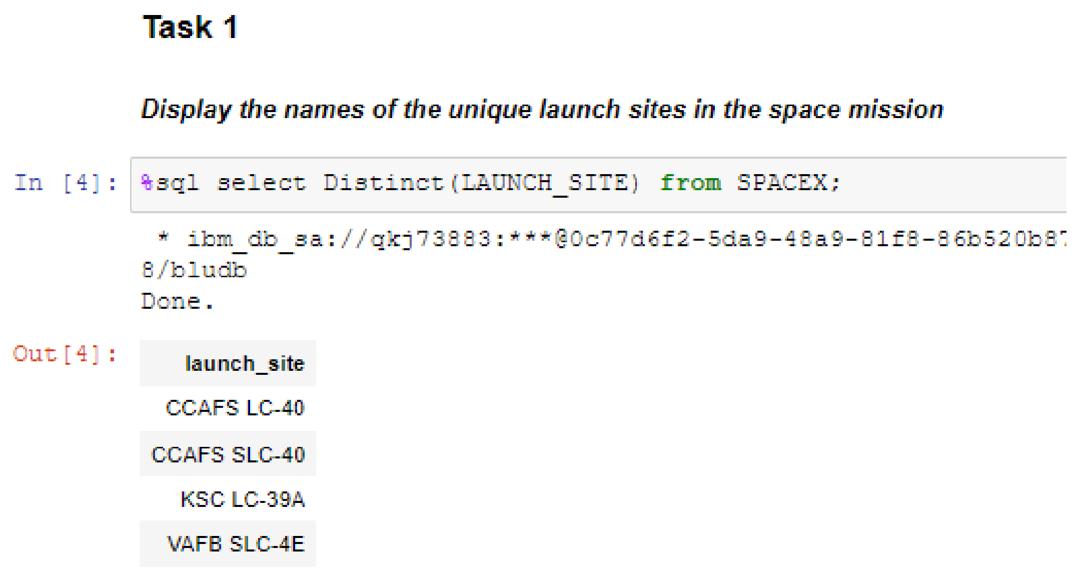
Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

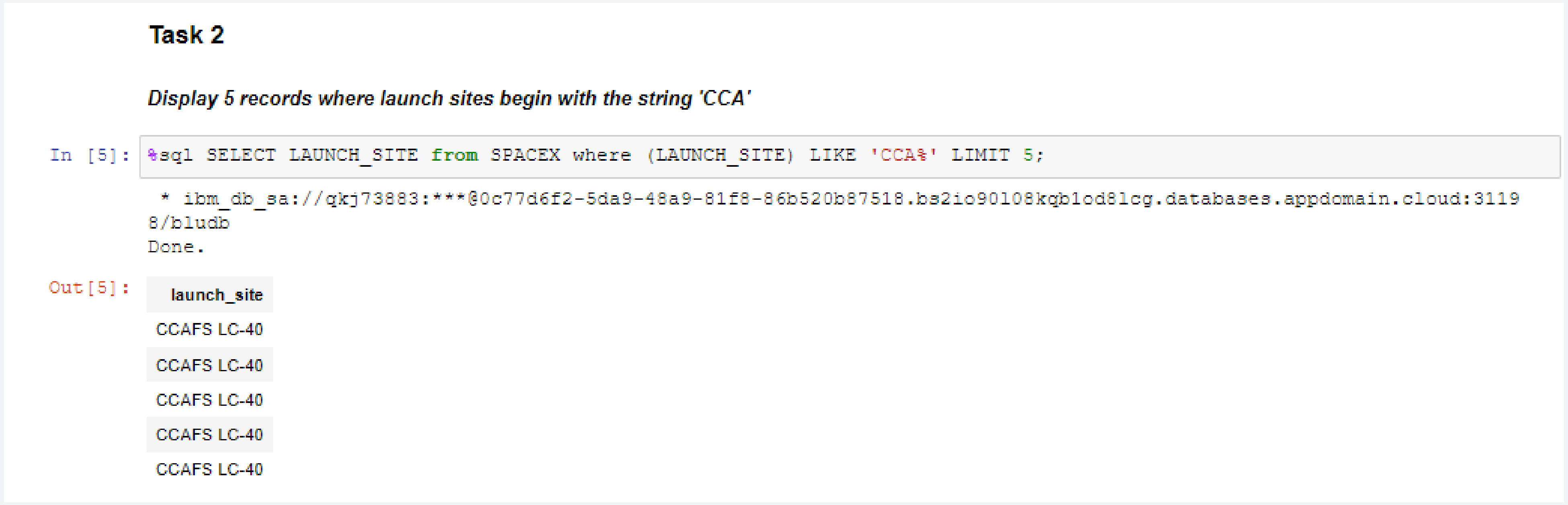


Task 1

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

```
In [4]: %sql select Distinct(LAUNCH_SITE) from SPACEX;

 * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87
8/bludb
Done.

Out[4]:
   launch_site
  CCAFS LC-40
 CCAFS SLC-40
  KSC LC-39A
  VAFB SLC-4E
```

Launch Site Names Begin with 'CCA'



Task 2

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

```
In [5]: %sql SELECT LAUNCH_SITE from SPACEX where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;

 * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:3119
8/bludb
Done.
```

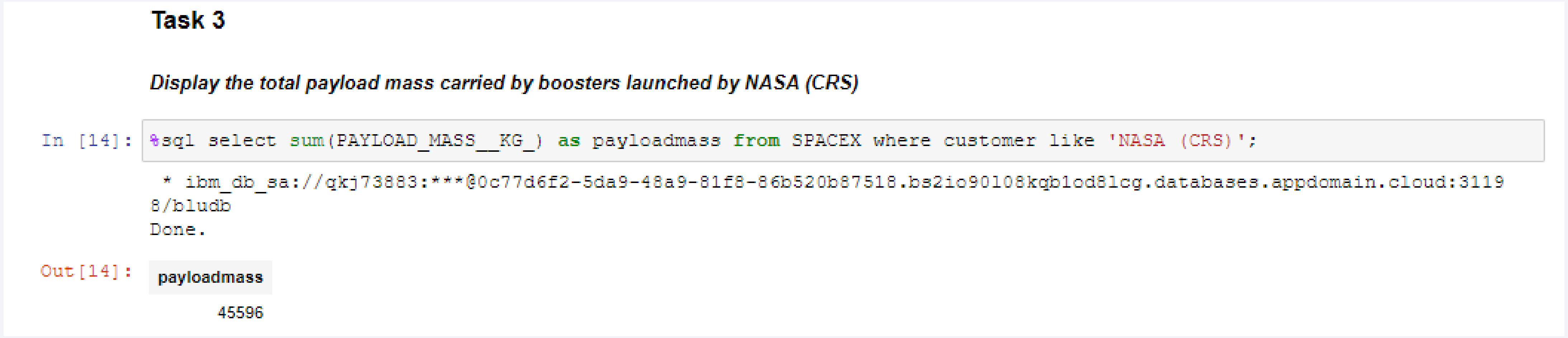
Out[5]:

launch_site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

- We used the query above to display 5 records where launch sites begin with `CCA`

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below



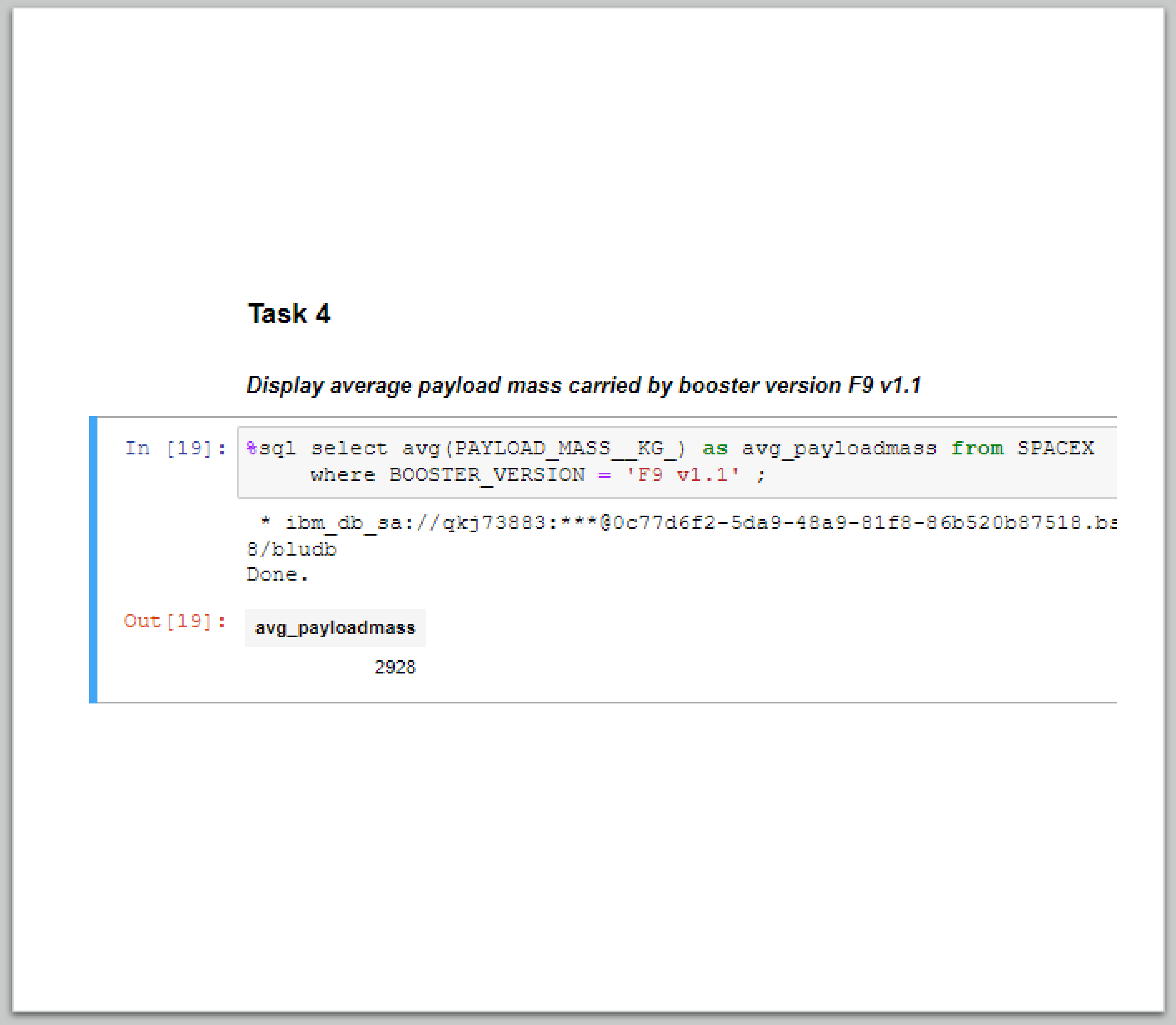
Task 3

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

```
In [14]: %sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEX where customer like 'NASA (CRS)';

 * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqb1od8lcg.databases.appdomain.cloud:3119
8/bludb
Done.

Out[14]: payloadmass
               45596
```

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4



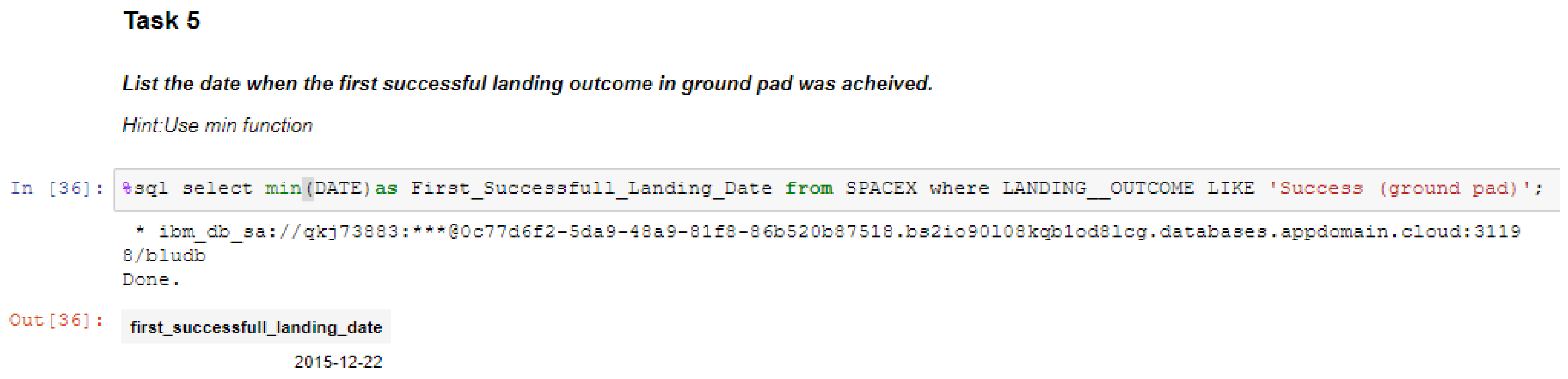
Task 4

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

```
In [19]: %sql select avg(PAYLOAD_MASS__KG_) as avg_payloadmass from SPACEX
             where BOOSTER_VERSION = 'F9 v1.1' ;

          * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs
         8/bludb
         Done.

Out[19]: avg_payloadmass
                    2928
```

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015



Task 5

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

Hint:Use min function

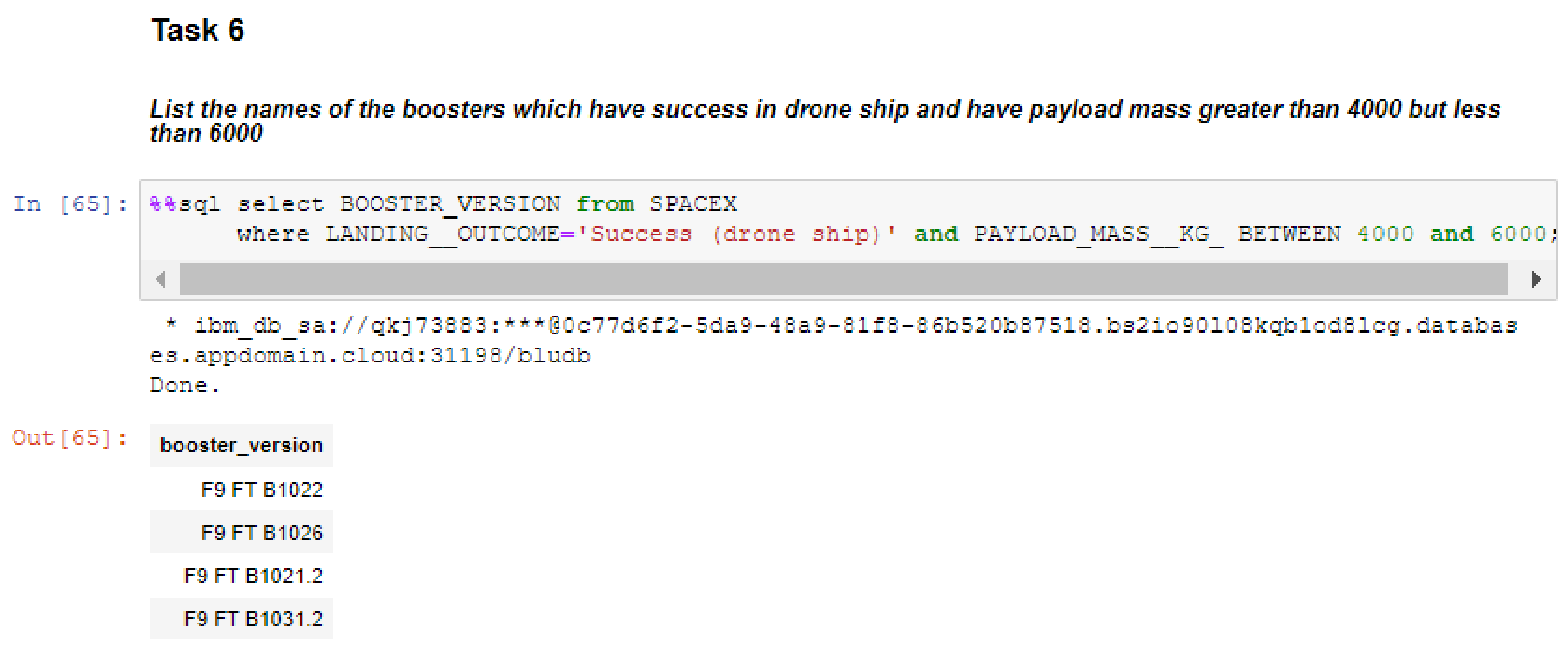
```
In [36]: %sql select min(DATE)as First_Successfull_Landing_Date from SPACEX where LANDING__OUTCOME LIKE 'Success (ground pad)';
```

```
 * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:3119
8/bludb
Done.
```

```
Out[36]: first_successfull_landing_date
                             2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000



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

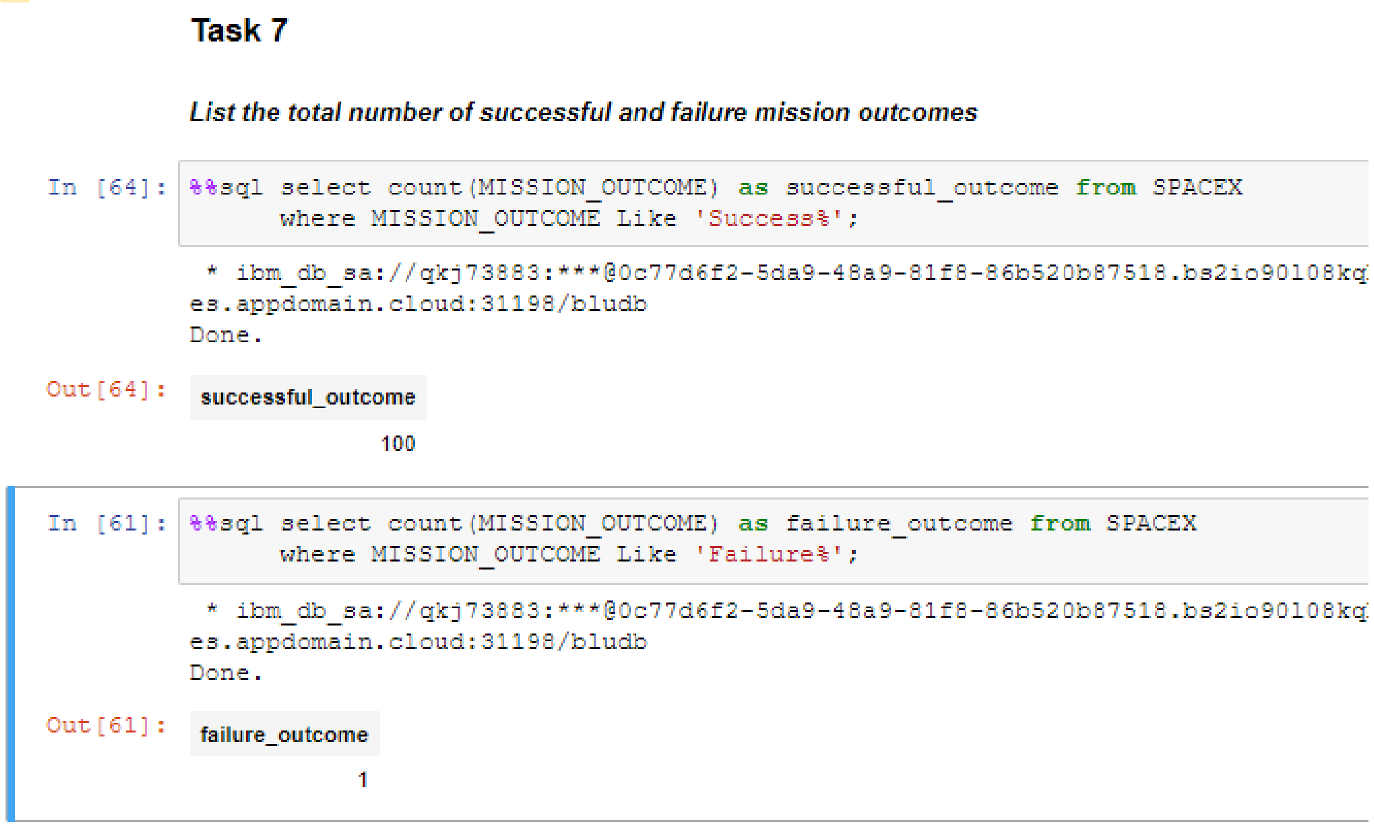
```
In [65]: %%sql select BOOSTER_VERSION from SPACEX
               where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

```
 * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31198/bludb
Done.
```

Out[65]:

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

Total Number of Successful and Failure Mission Outcomes



Task 7

List the total number of successful and failure mission outcomes

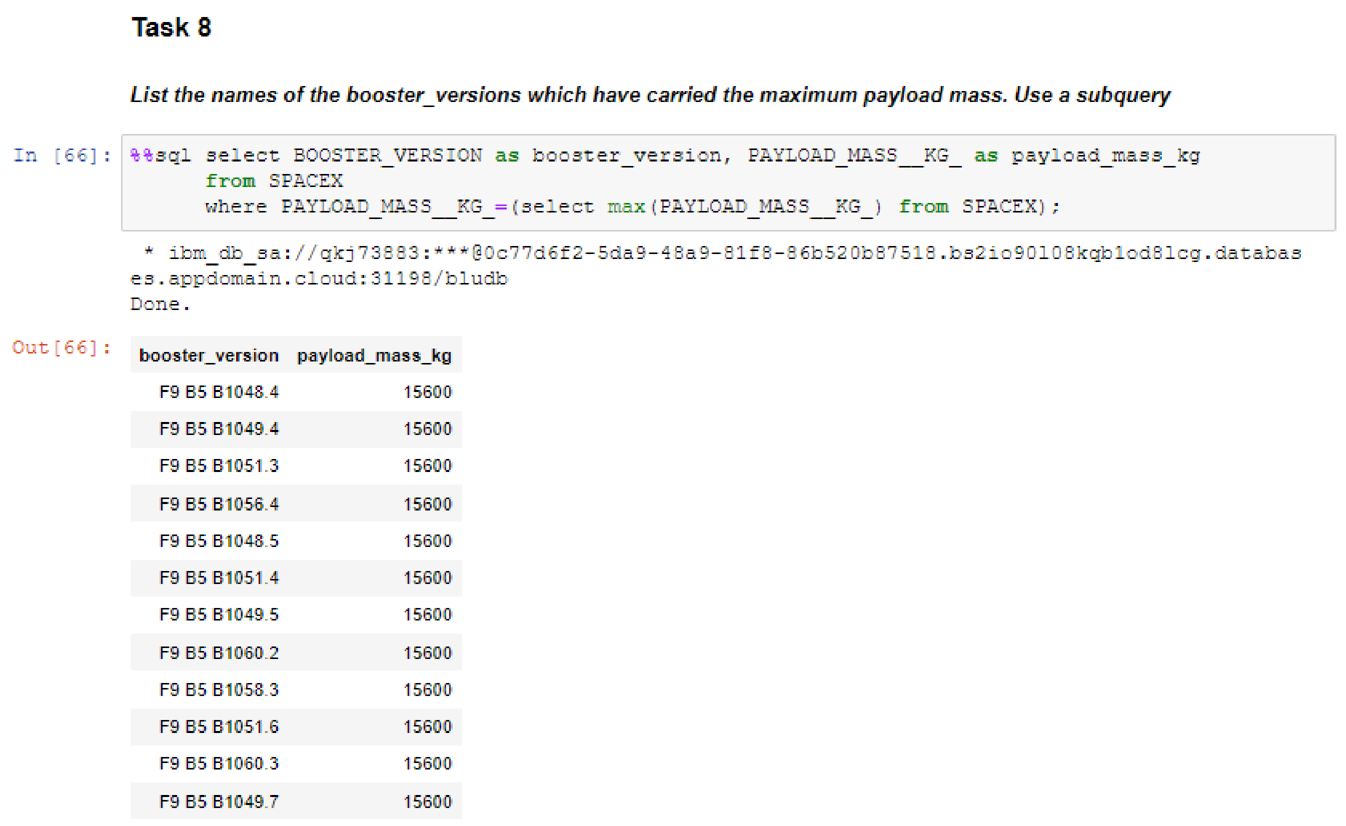
```
In [64]: %%sql select count(MISSION_OUTCOME) as successful_outcome from SPACEX
               where MISSION_OUTCOME Like 'Success%';

          * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kq
         es.appdomain.cloud:31198/bludb
         Done.

Out[64]: successful_outcome
                        100
```

```
In [61]: %%sql select count(MISSION_OUTCOME) as failure_outcome from SPACEX
               where MISSION_OUTCOME Like 'Failure%';

          * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kq
         es.appdomain.cloud:31198/bludb
         Done.

Out[61]: failure_outcome
                      1
```

- We used wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.



Task 8

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

```
In [66]: %%sql select BOOSTER_VERSION as booster_version, PAYLOAD_MASS__KG_ as payload_mass_kg
               from SPACEX
               where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEX);
```

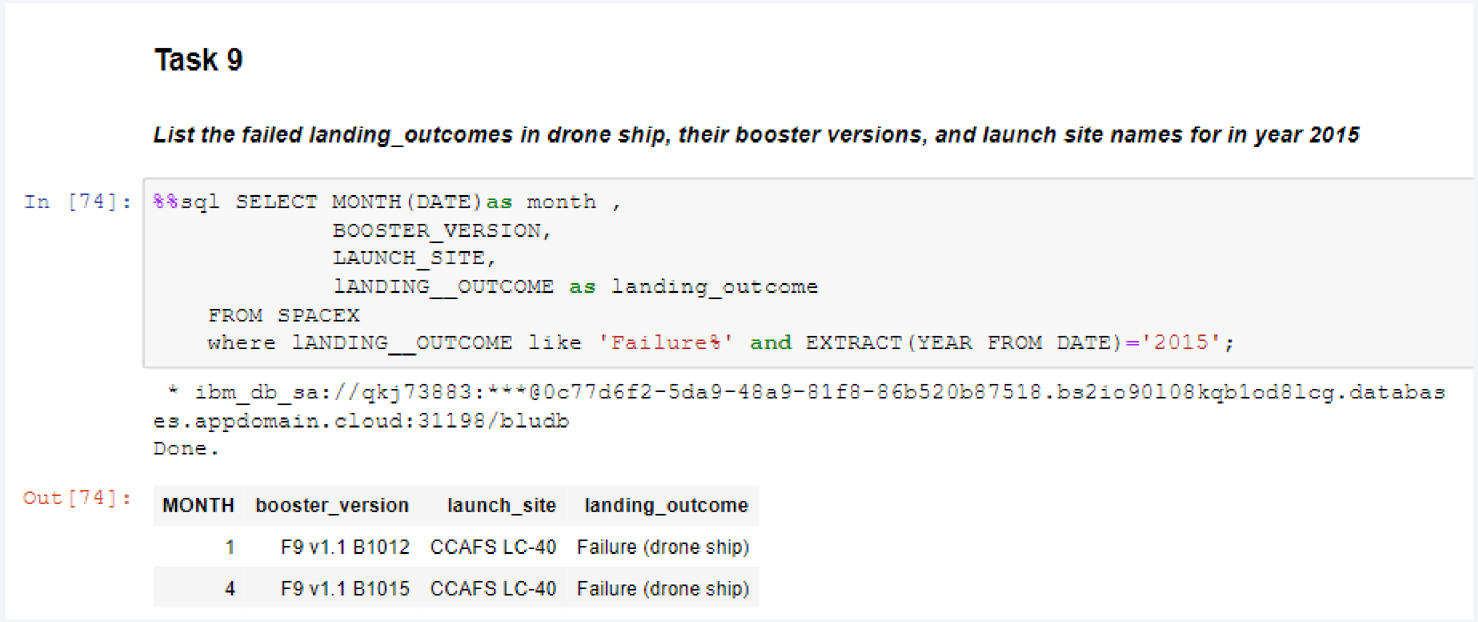
```
 * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqb1od8lcg.databas
es.appdomain.cloud:31198/bludb
Done.
```

Out[66]:

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

2015 Launch Records

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015



Task 9

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

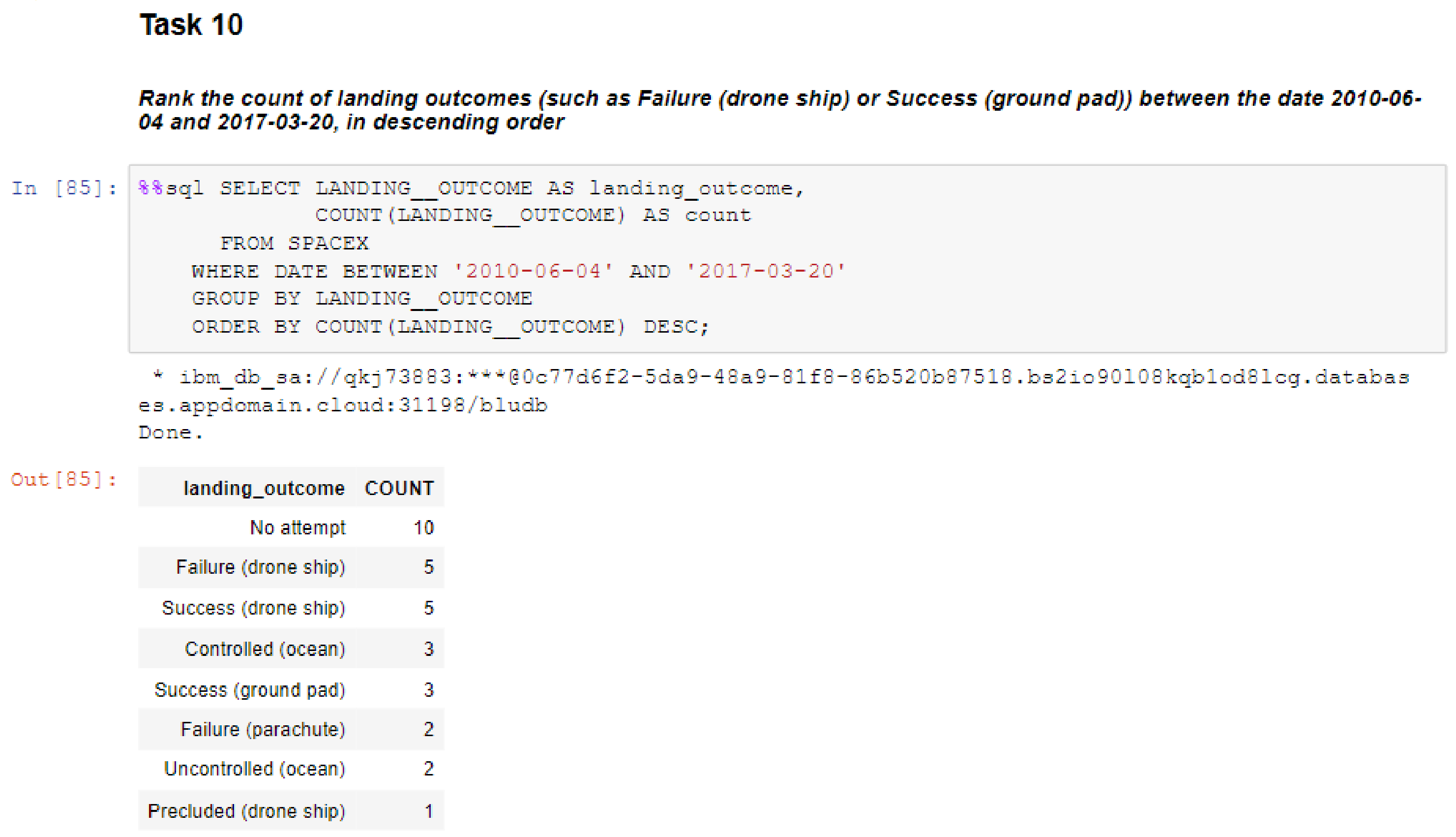
In [74]:
```
%%sql SELECT MONTH(DATE)as month ,
             BOOSTER_VERSION,
             LAUNCH_SITE,
             lANDING__OUTCOME as landing_outcome
    FROM SPACEX
    where lANDING__OUTCOME like 'Failure%' and EXTRACT(YEAR FROM DATE)='2015';
```

```
 * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90l08kqb1od8lcg.databas
es.appdomain.cloud:31198/bludb
Done.
```

Out[74]:

MONTH	booster_version	launch_site	landing_outcome
1	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
4	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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



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 [85]: %%sql SELECT LANDING__OUTCOME AS landing_outcome,
                    COUNT(LANDING__OUTCOME) AS count
             FROM SPACEX
           WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
           GROUP BY LANDING__OUTCOME
           ORDER BY COUNT(LANDING__OUTCOME) DESC;
```

```
 * ibm_db_sa://qkj73883:***@0c77d6f2-5da9-48a9-81f8-86b520b87518.bs2io90108kqblod8lcg.databas
es.appdomain.cloud:31198/bludb
Done.
```

Out [85]:

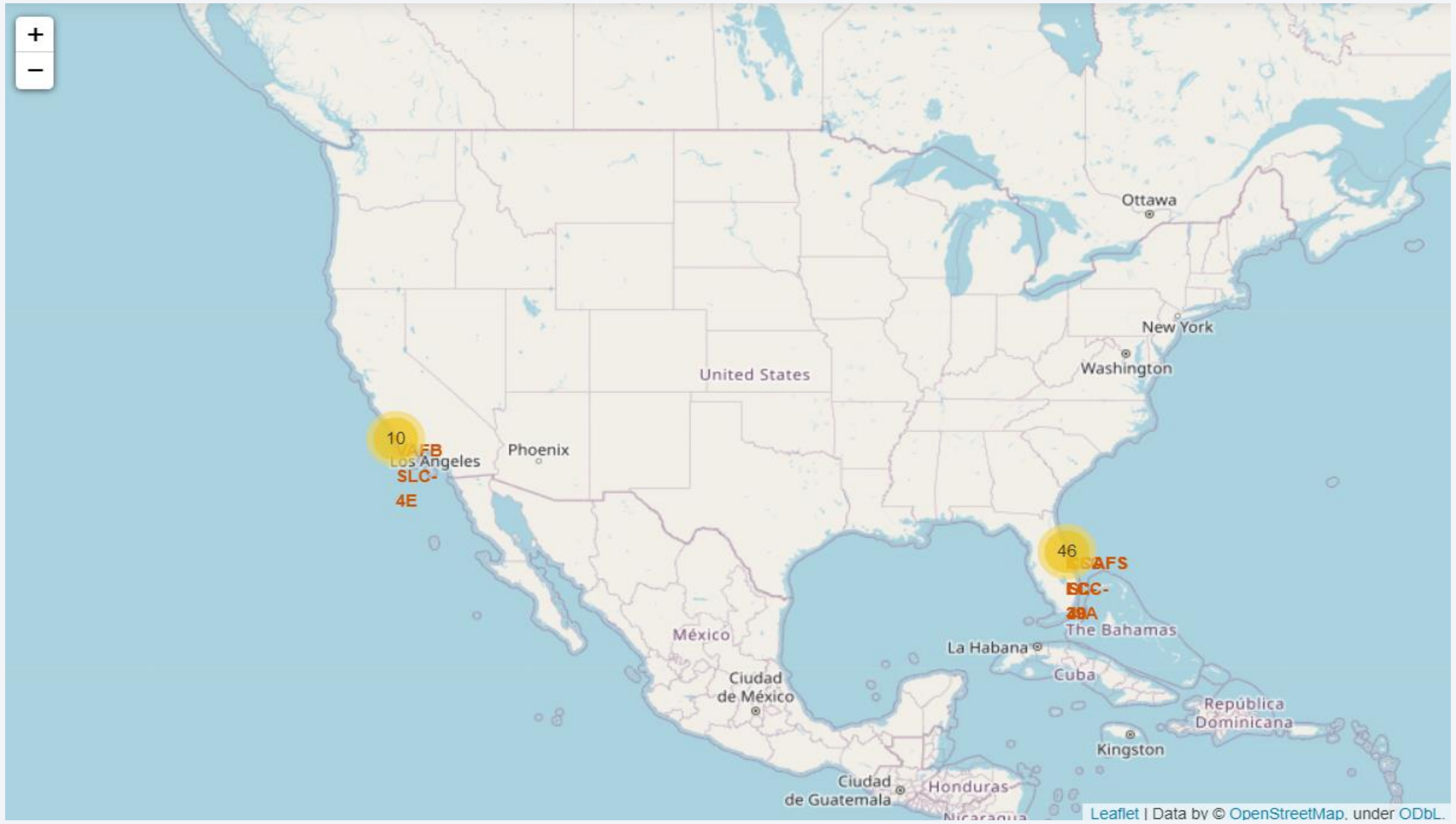
landing_outcome	COUNT
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Section 4

Launch Sites Proximities Analysis

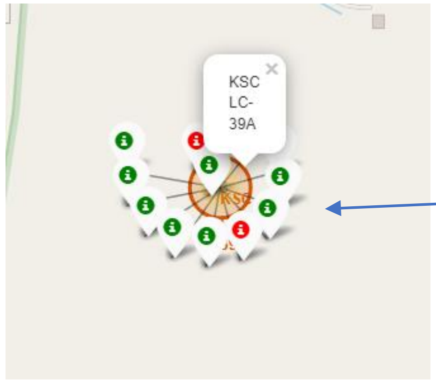
All launch sites global map markers

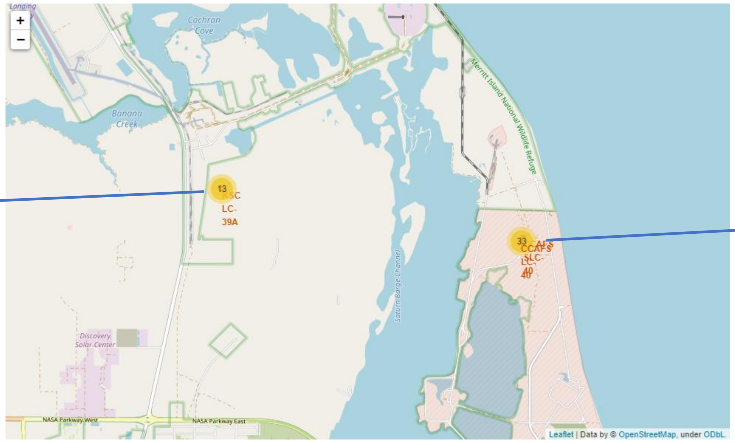


- The Launch sites of Spaxe X are located on the west and east coasts of USA, California and Florida, respectively.

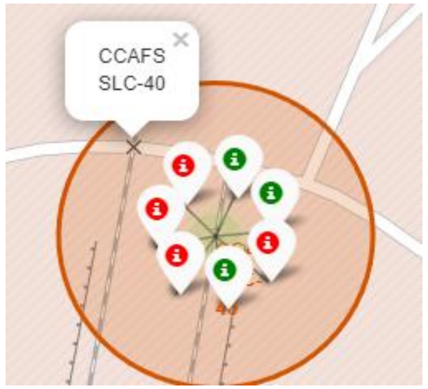
Markers showing launch sites with color labels

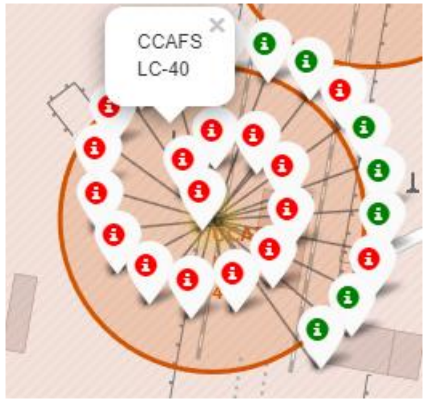
- Florida Launch sites



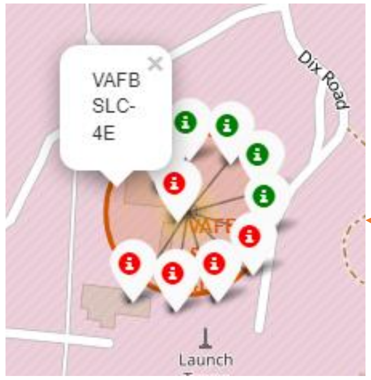


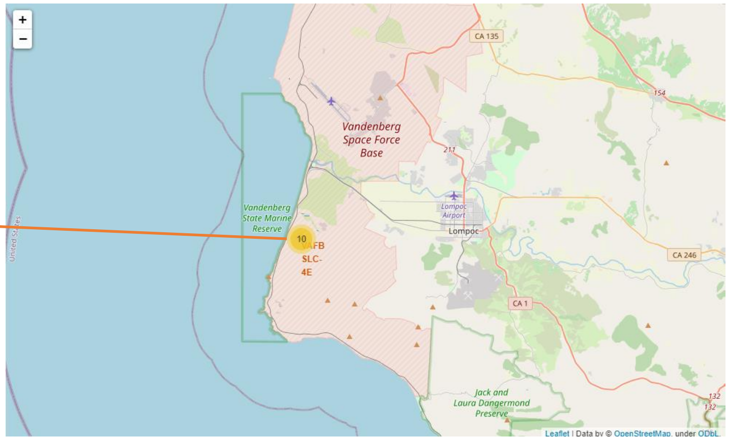
7
26
Leaflet | Data by © OpenStreetMap, under ODbL.





- California Launch sites

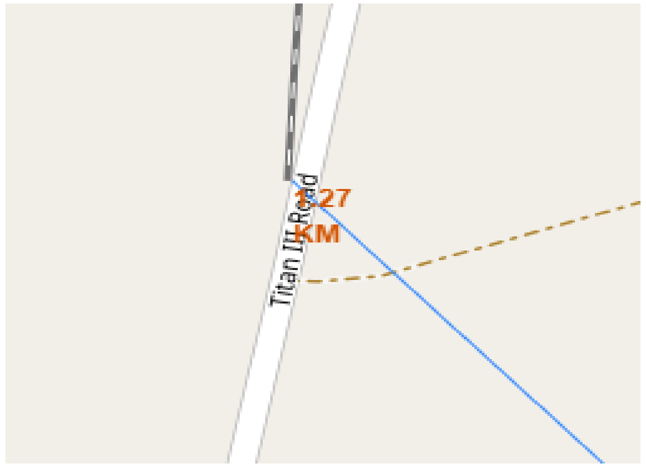




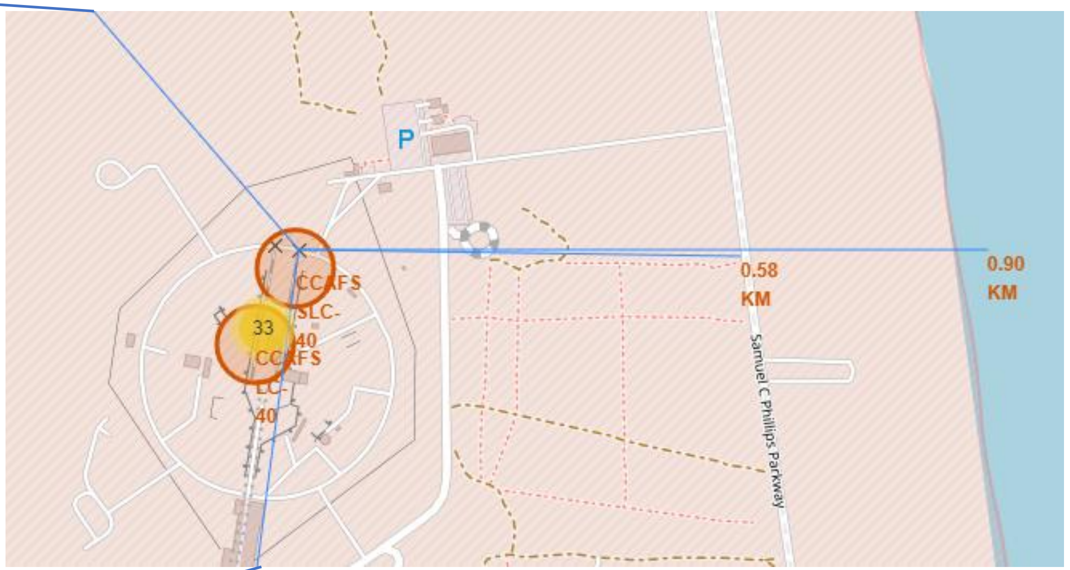
- Red indicator shows a successful launch
- Green indicator shows a failed launch

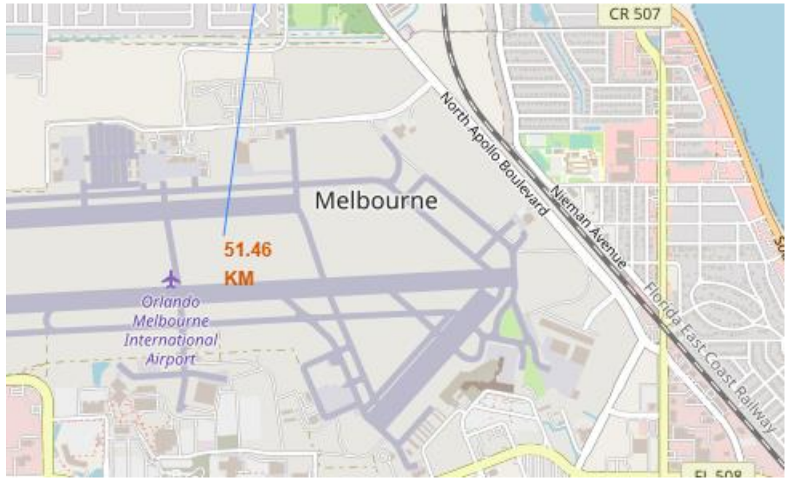
Launch Site distance to landmarks

- Distance to closest railroad

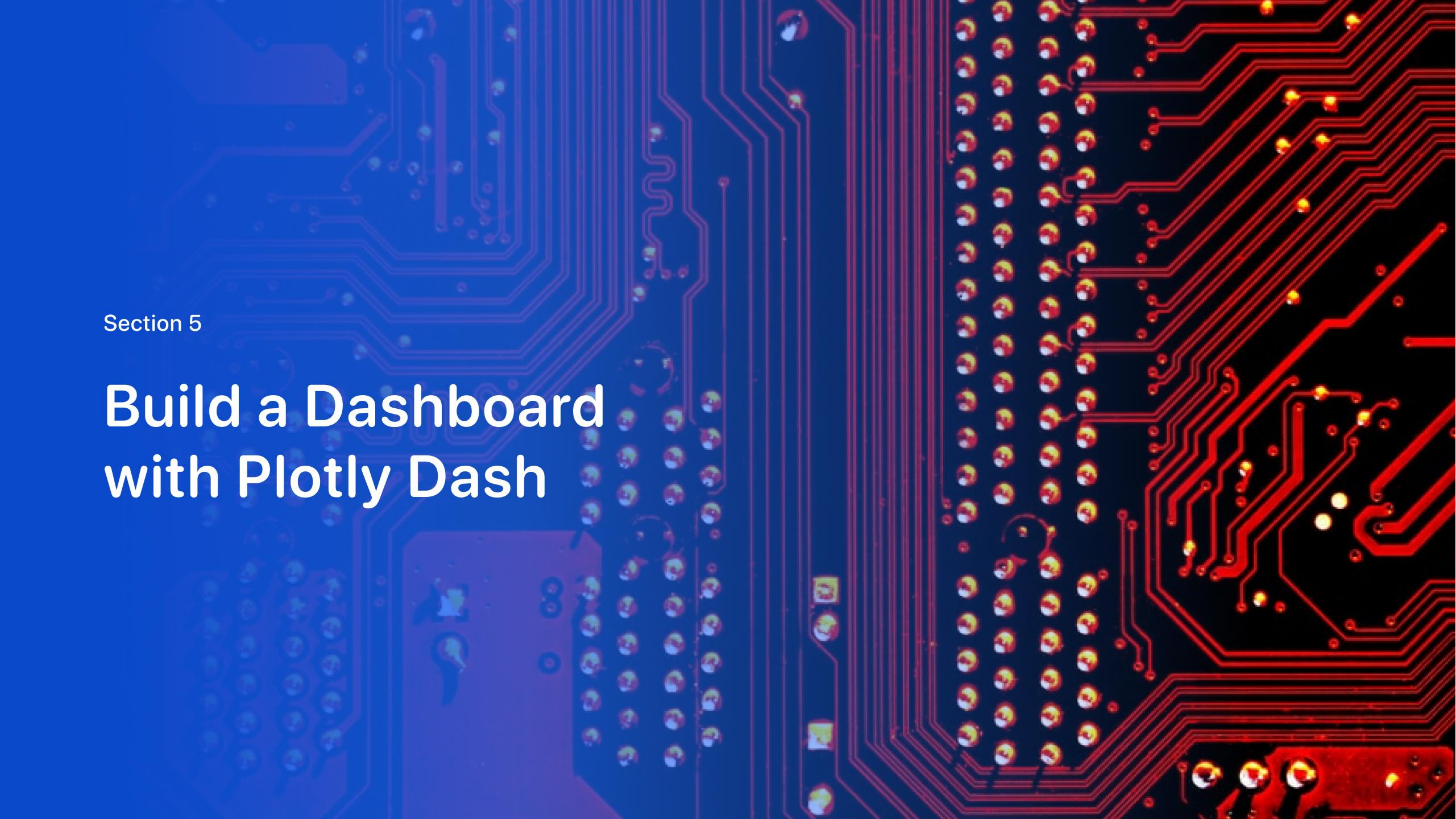


- Distance to closest highway and coast





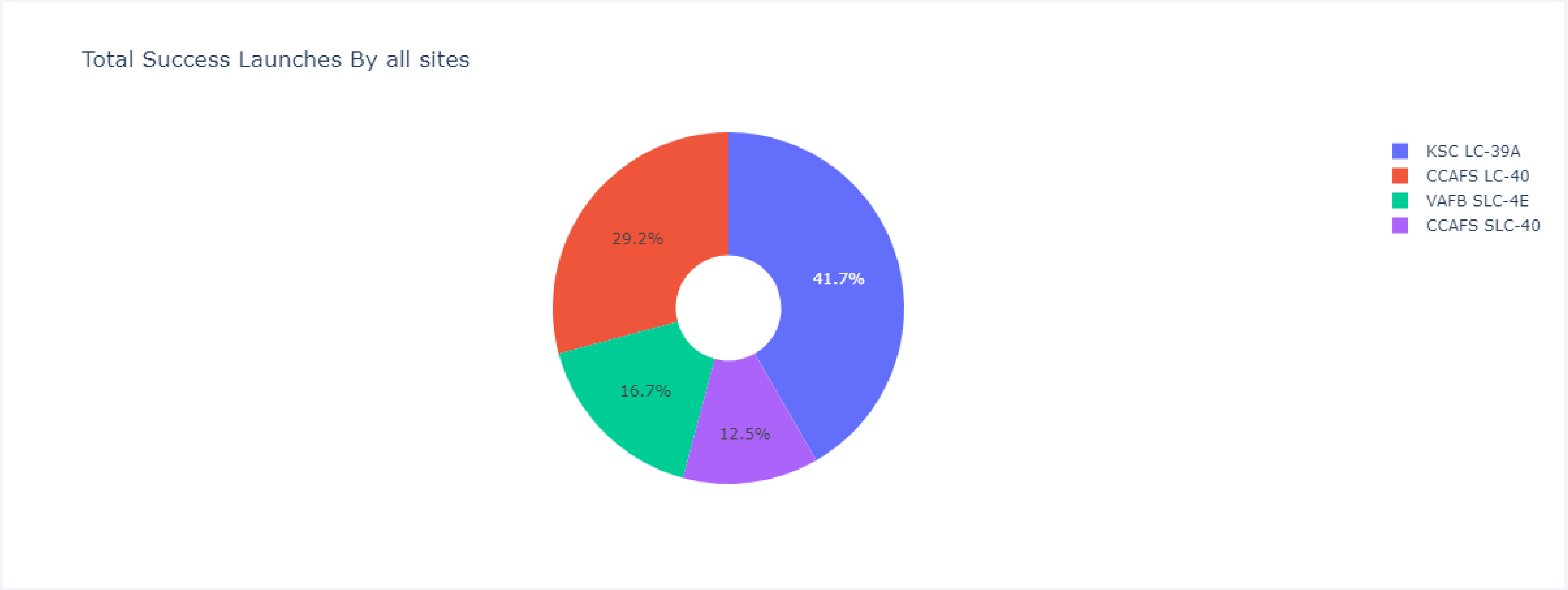
- Distance to closest city



Section 5

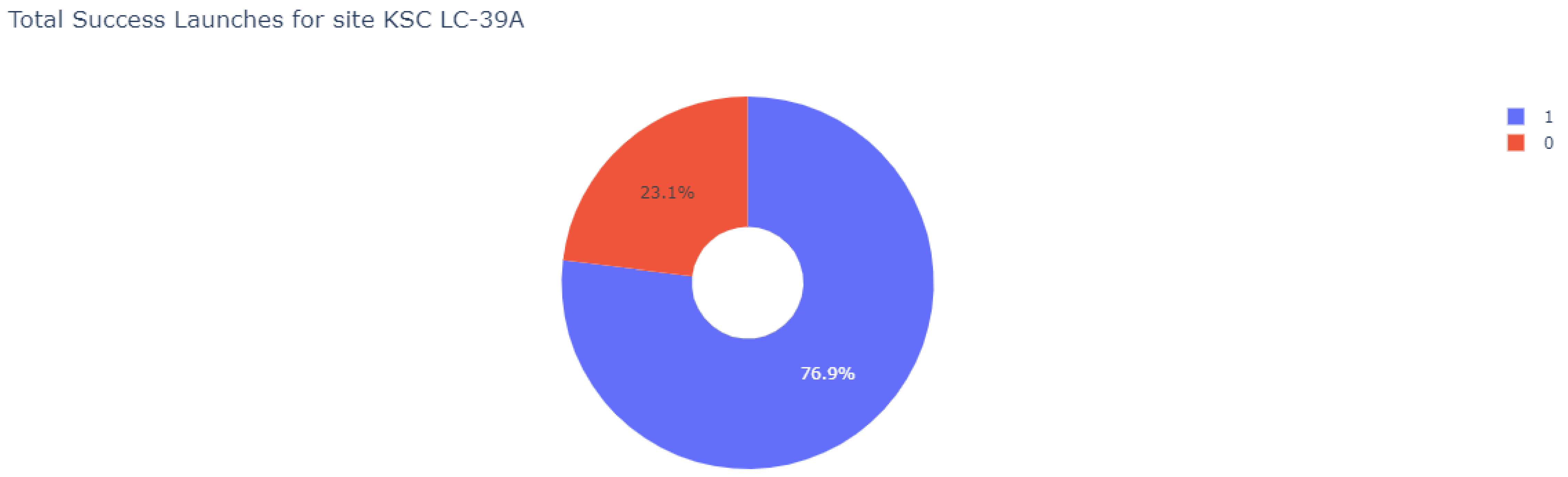
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



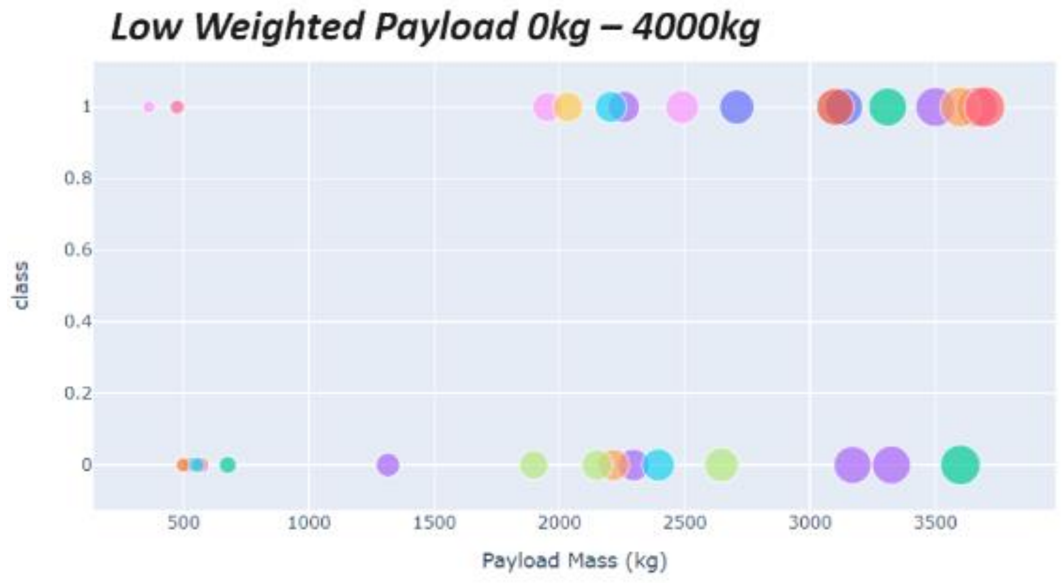
- The KSC LC-39A launch site had the most successful launch rate between all sites

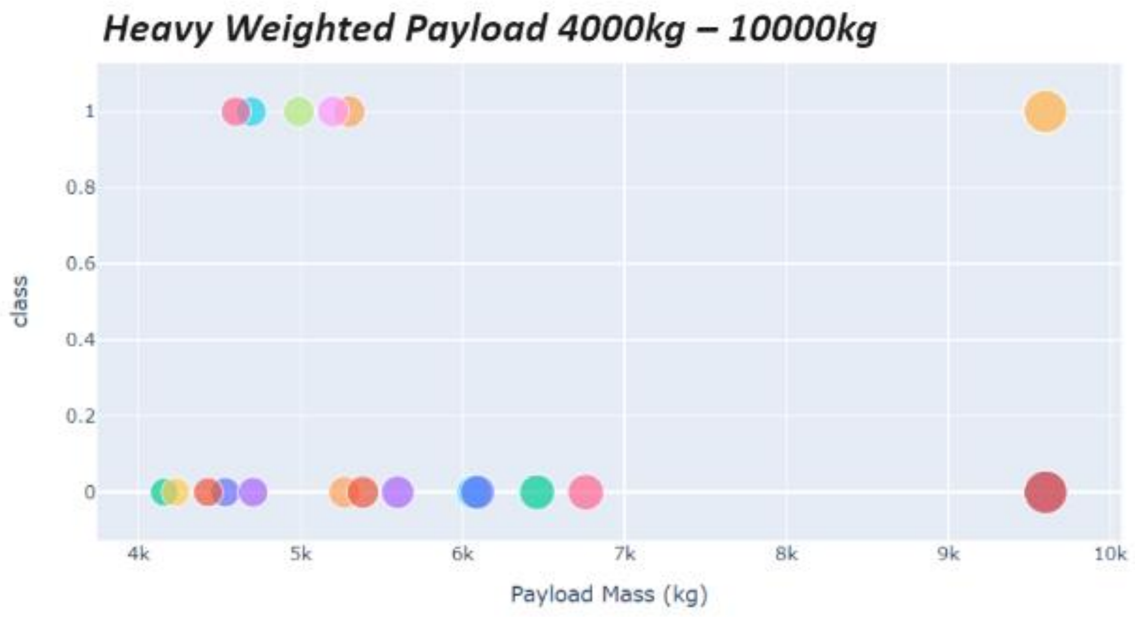
Pie chart showing the Launch site with the highest launch success ratio



- The KSC LC-39A launch site achieved a 76.9% success launch rate. It's the highest among all launch sites.

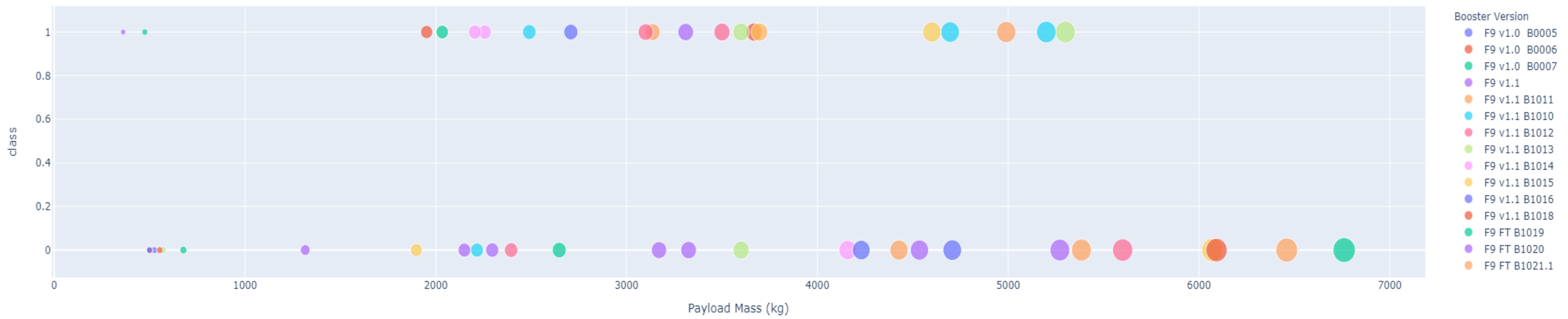
Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider





We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



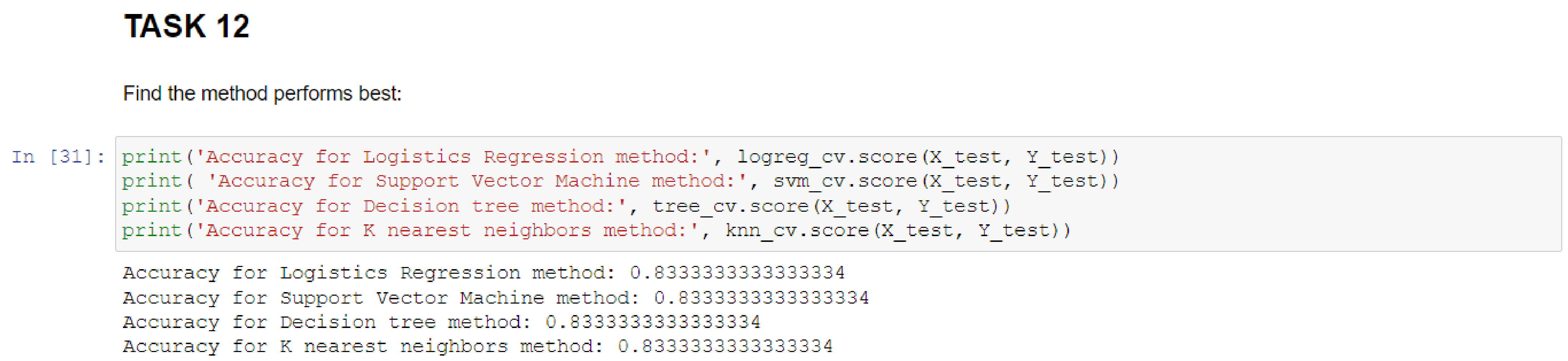
- The conclusion we can draw from the above diagram is that the success rates of low weighted payloads (below 4000 kg) is higher than that of the heavier ones.

Section 6

Predictive Analysis (Classification)

Classification Accuracy

- All the models have equal predicted accuracy



TASK 12

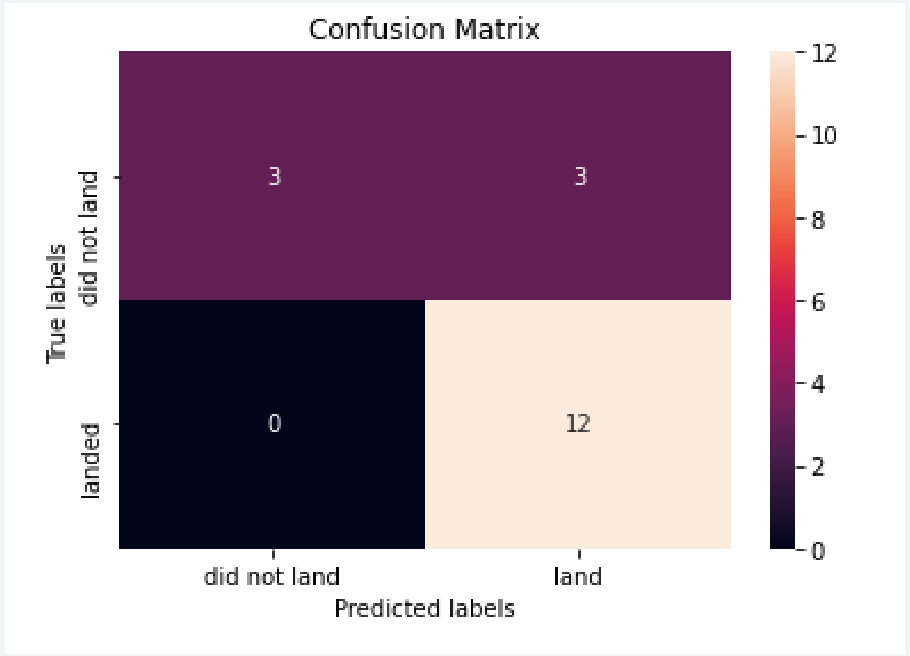
Find the method performs best:

```
In [31]: print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
         print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
         print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
         print('Accuracy for K nearest neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.8333333333333334
Accuracy for K nearest neighbors method: 0.8333333333333334
```

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes.
- The only problem is that although the model seems to predict the successful landings (True Positives) it does not seem to do the same for the failed ones.



Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!