IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

SARTHAK SARRAF
01 June 2024

Outline

Executive Summary

- Summary of methodologies

  - Data Collection through API
  - Data Collection through Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis through Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction

- Summary of all results

  - Exploratory Data Analysis Results
  - Interactive Analysis in Screenshots
  - Predictive Analysis

Introduction

- Project background and context

  SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; others providers cost upward of 165 million dollars each, much of 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 any alternative company wants to bid against SpaceX for a rocket launch. So, the 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?

  - What factors will be best for rocket to land successfully?

  - Which features are best for rocket to land successfully?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data is collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling:
  - Removing of null values and doing one hot encoding on dataset.
- Perform exploratory data analysis (EDA) using visualization and SQL:
  - Extracting information's and relations between different features of data set and how they are affecting failure and success of rocket landing.
- Perform interactive visual analytics using Folium and Plotly Dash:
  - Marking on Map to visualize in which areas how many successful or failed rocket landings and creating of Dashboard using plotly dash.
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models.

Data Collection

- Data is collected through various methods:

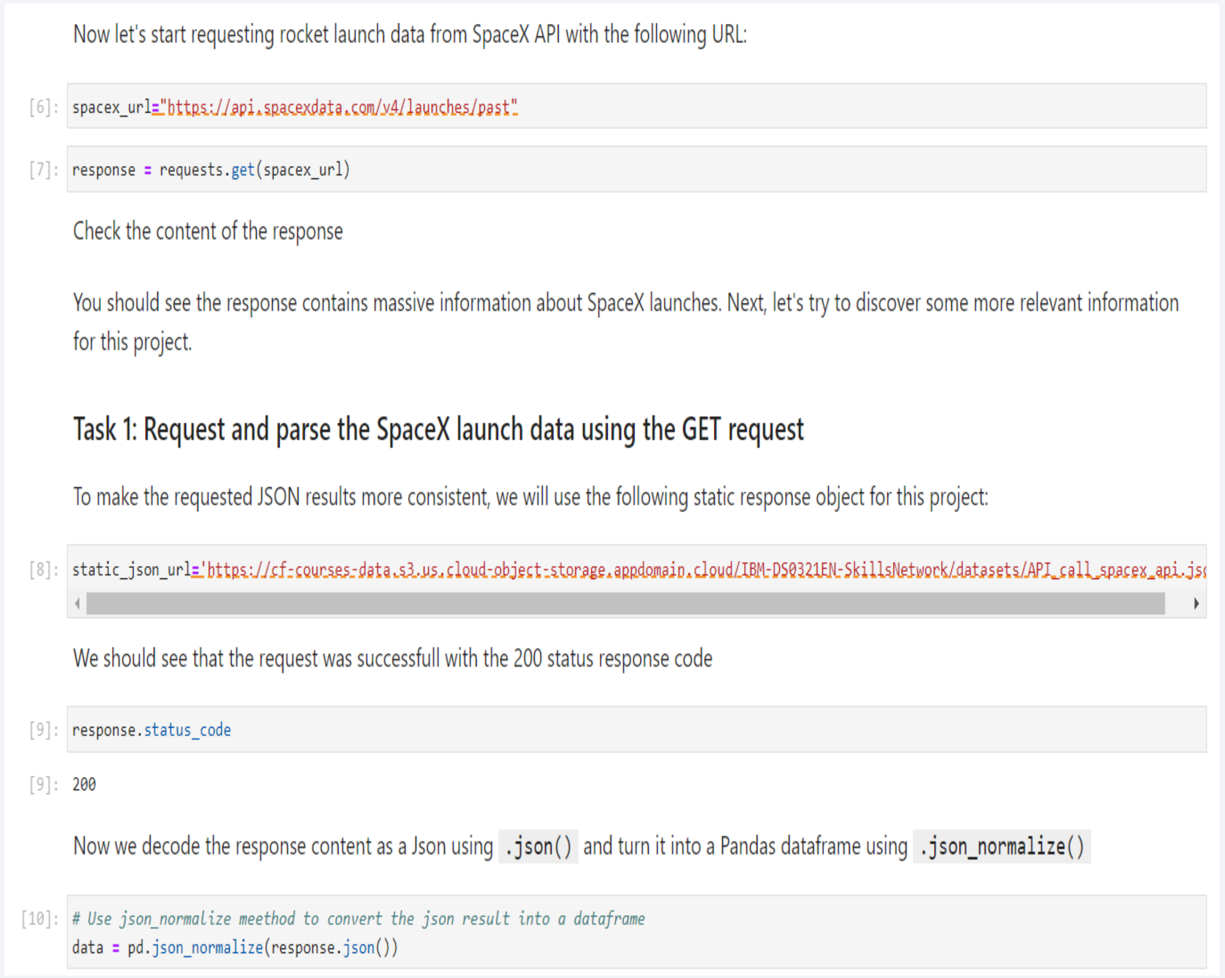
  - First data is collected through URL by using get request to the SpaceX API.
  - Then Normalize the .json() file using .json_normalize().
  - Extracting the columns through API and cleaning the data and fill the missing values.
  - We perform Web Scraping from Wikipedia for Falcon 9 launch record using BeautifulSoup.
  - Extracting the data through HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We have used get request method for data collection and perform some data wrangling and formatting.

- GitHub URL for completed SpaceX API calls notebook:

https://github.com/sarrafsarthak/spacex_project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



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

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

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

Check the content of the response

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:

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

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

```
[9]: response.status_code
```

```
[9]: 200
```

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

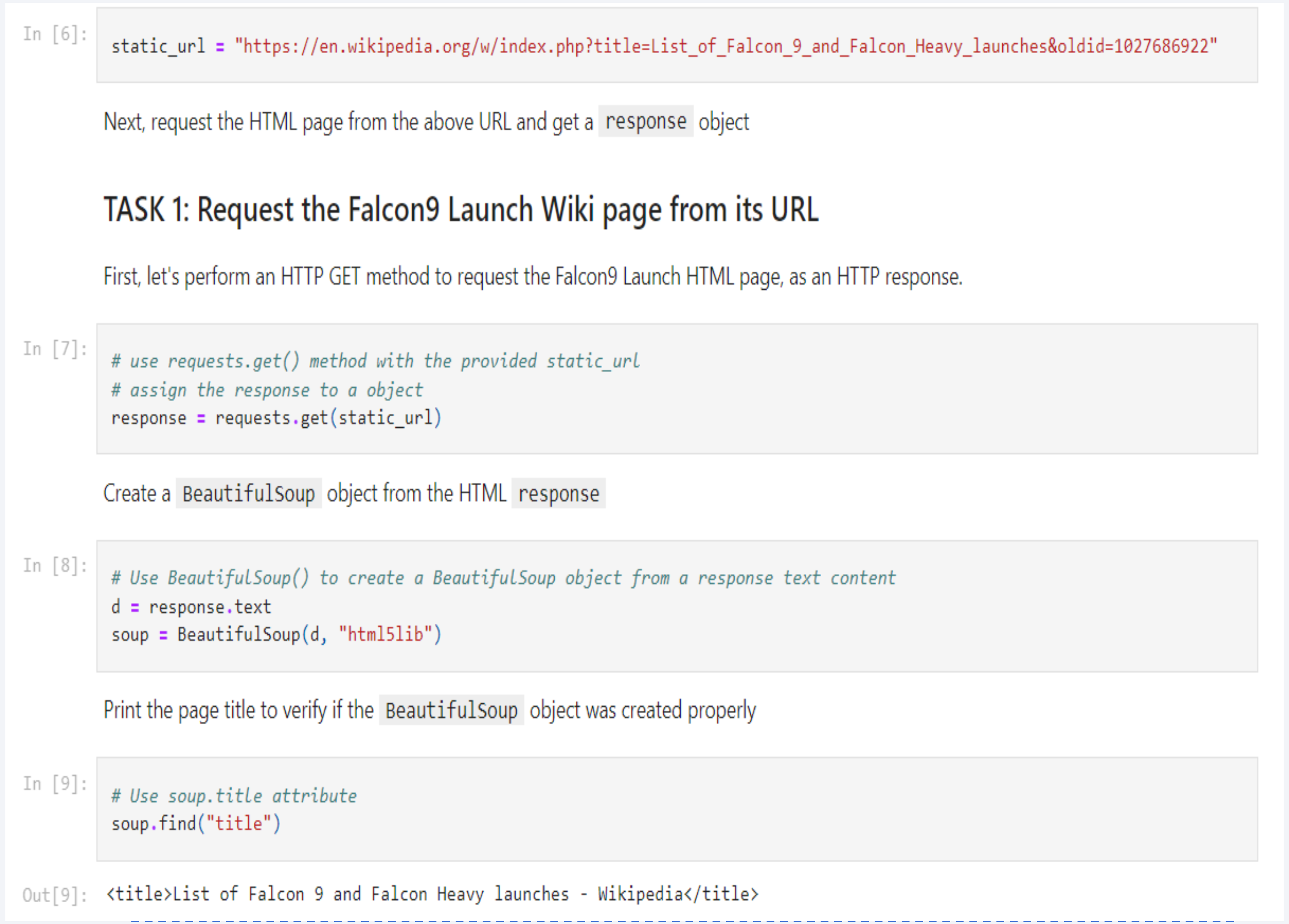
```
[10]: # Use json_normalize meethod to convert the json result into a dataframe
      data = pd.json_normalize(response.json())
```

Data Collection - Scraping

- We have used a request get method to get data from web scraping and Beautiful Soup for parsing in HTML files.

- GitHub URL for completed web scraping notebook:

https://github.com/sarrafsarthak/spacex_project/blob/main/jupyter-labs-webscraping.ipynb



In [6]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

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 [7]:
```
# 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`

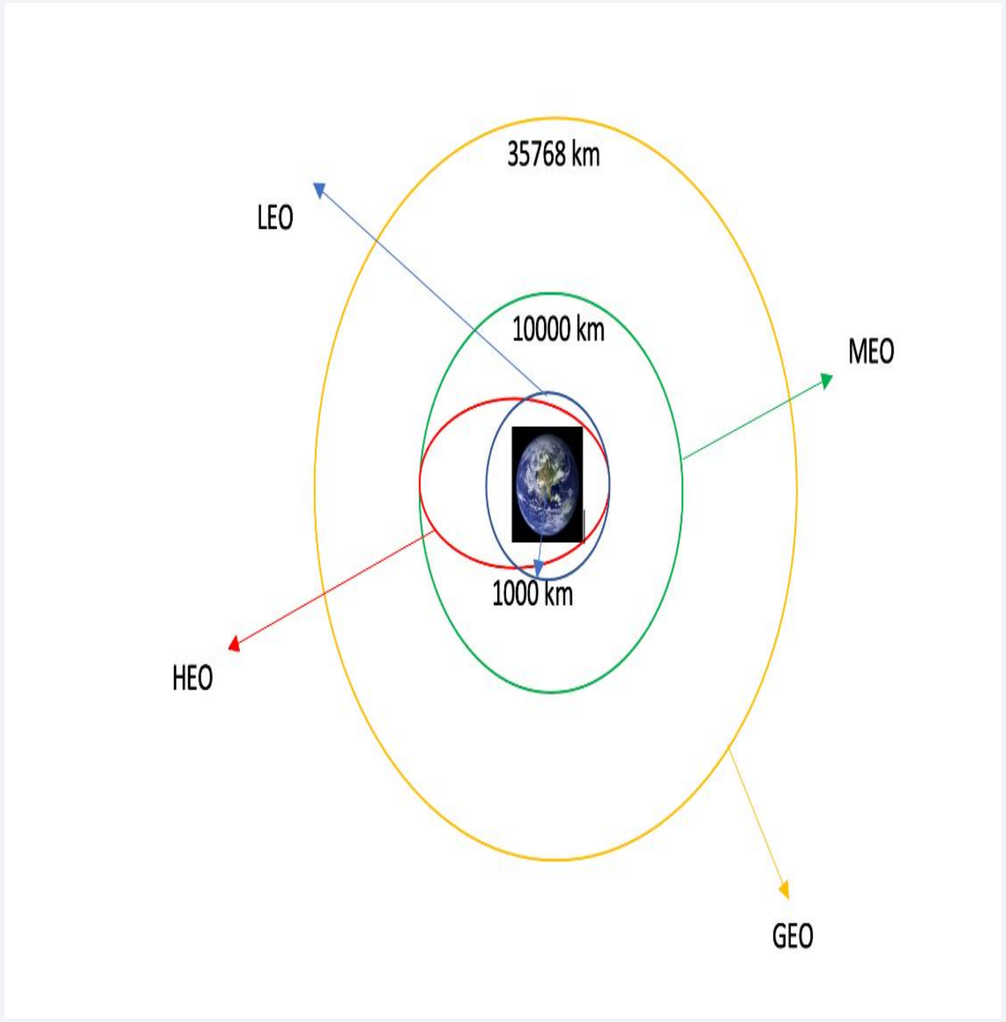
In [8]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
d = response.text
soup = BeautifulSoup(d, "html5lib")
```

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

In [9]:
```
# Use soup.title attribute
soup.find("title")
```

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

Data Wrangling

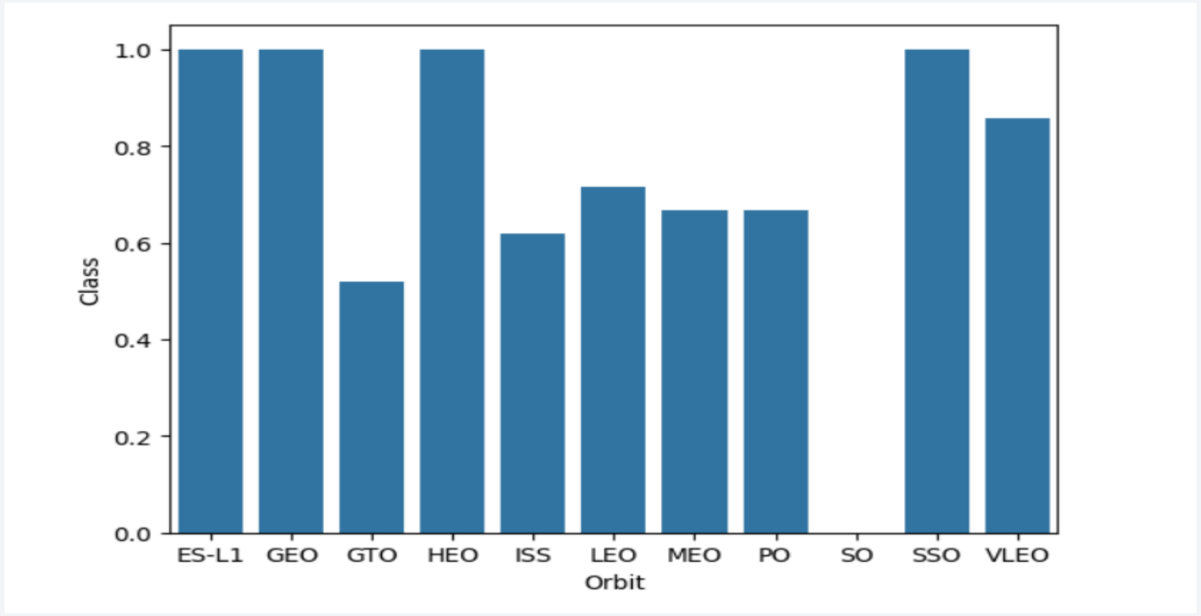


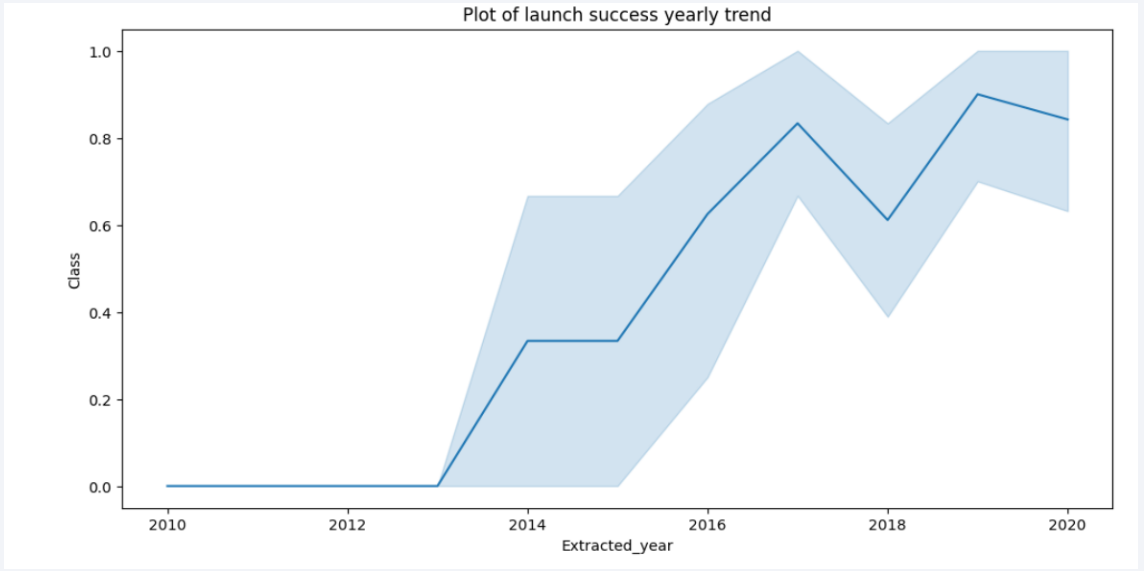
- In Data Wrangling process we have first calculate the number of launches at each site.
- Then we calculate the number and occurrence of each orbit.
- Calculated the number of occurrence of mission outcomes of the orbits.
- Creating a labels for landing outcome using outcome column and insert it into new column Class.
- GitHub URL for completed data wrangling related notebook:

https://github.com/sarrafsarthak/spacex_project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We have used charts such as scatter plot, bar plot to find the relationship between Flight Number & Payload Mass, Flight Number & Launch Site, Payload & Launch Site, Success Rate of each Orbit, Flight Number & Orbit, Payload & Orbit, Yearly success trends.





Git Hub URL for complete Data visualization Notebook:

https://github.com/sarrafsarthak/spacex_project/blob/main/edadataviz.ipynb

EDA with SQL

- We have first loaded the SpaceX dataset into Sqlite3 database.
- We perform EDA using SQL on dataset for finding many insights from the dataset:
    - The name of unique Launch sites in the dataset.
    - The total payload mass carried by boosters launched by NASA(CRS).
    - Average Payload mass carried by booster version F9 V1.1
    - Date when the first successful landing outcome in ground pad was achieved.
    - Total number of successful and failure mission outcomes.
    - Name of booster version having maximum payload mass.
    - Records which display failed landing outcomes, drone ship, booster version, launch site for months in year 2015.
- GitHub URL for completed EDA with SQL notebook:

https://github.com/sarrafsarthak/spacex_project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We first add Circle and Marker on the NASA Johnson space center location.
- Then we add different Launch Site locations in map using Circle and Marker.
- Then we add successful/un-successful launch at each launch site using Marker.
- Used mouse pointer to find location coordinates of each point.
- Used polyline for making line from launch site coordinate to another coordinate and marking its distance.
- GitHub URL for interactive map with Folium map:

https://github.com/sarrafsarthak/spacex_project/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We have added pie chart and scatter plot for showing different distribution on a SpaceX launch Record dashboards.
- Pie chart is used for giving the success percentages of all sites and success/failure percentage of each launch site.
- Scatter plot show relationship between class of success/failure and payload mass for each launch sites for different booster version.
- GitHub URL for Plotly Dash lab:

https://github.com/sarrafsarthak/spacex_project/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- We have loaded the dataset using NumPy, pandas, transform the data and split it into train and test dataset.
- We use GridSearchCV for finding the best hyper parameter for each machine learning models such as LogisticRegression, SVM, DecisionTree, KNN.
- We then find accuracy for each model, improve each model using feature engineering and algorithmic tuning.
- Find the best performing classification model.
- GitHub URL for predictive analysis lab:

https://github.com/sarrafsarthak/spacex_project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

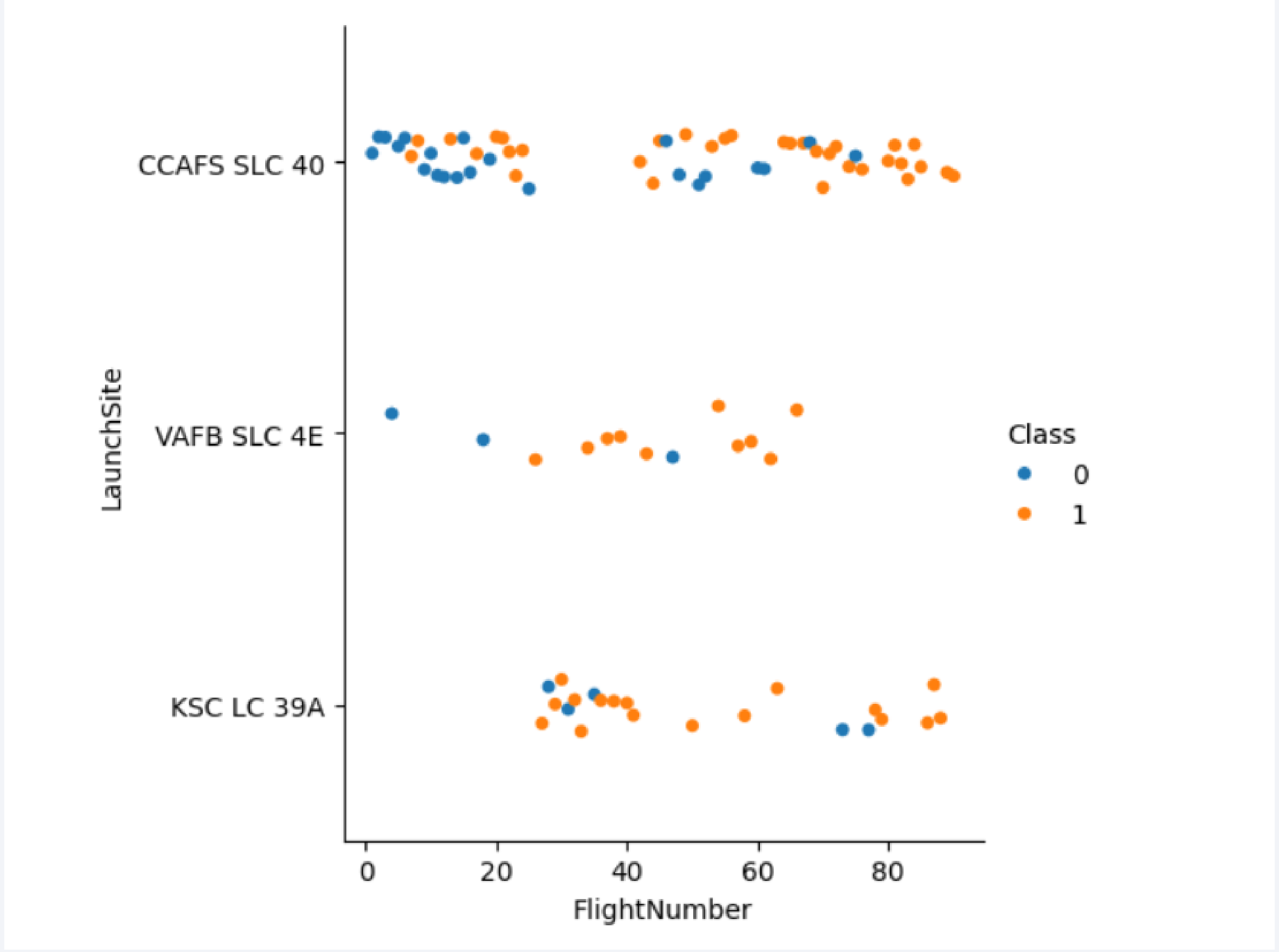
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

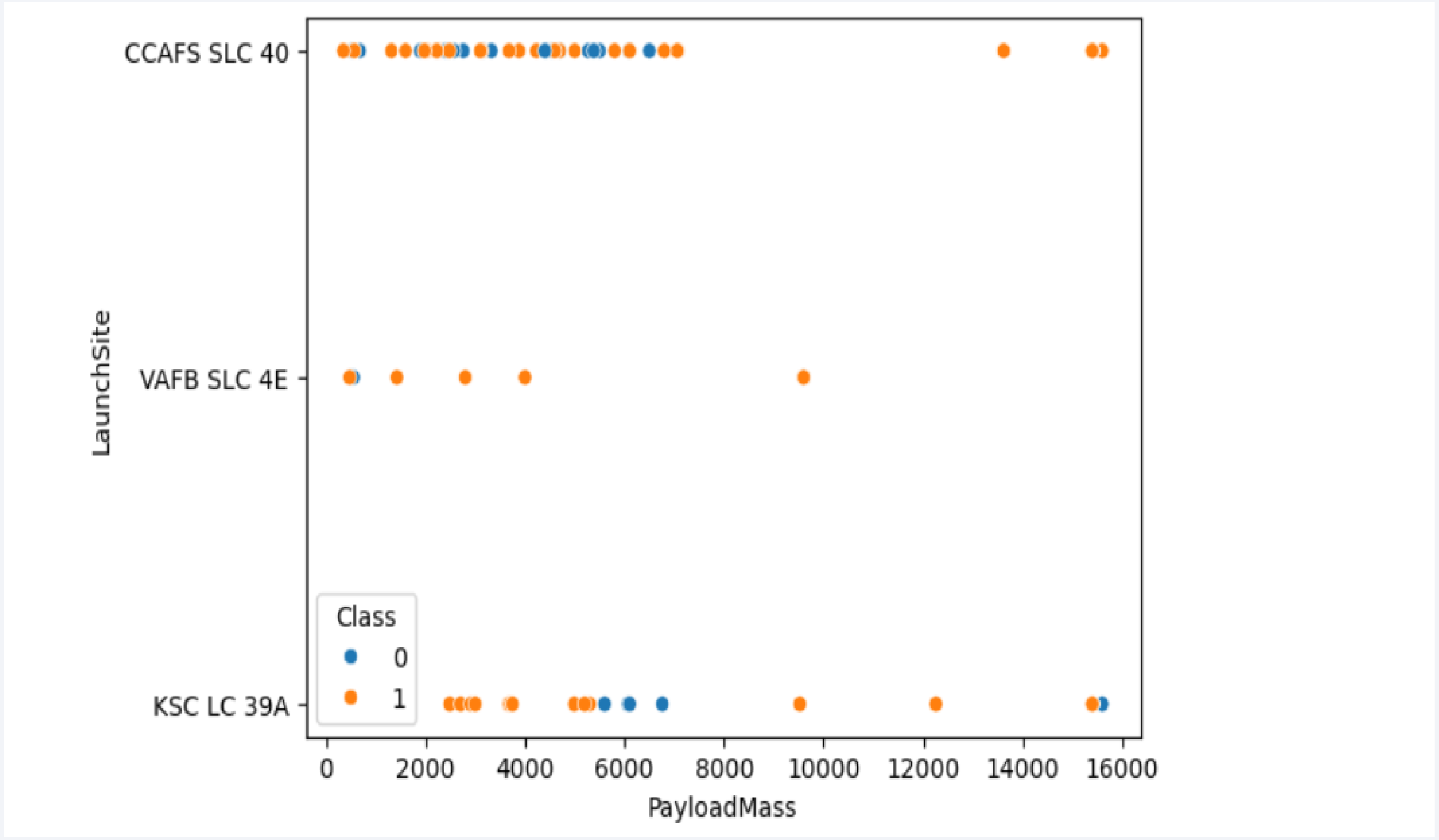
Flight Number vs. Launch Site

- From the plot we found that as the flight number increases the success rate of each launch site increases.



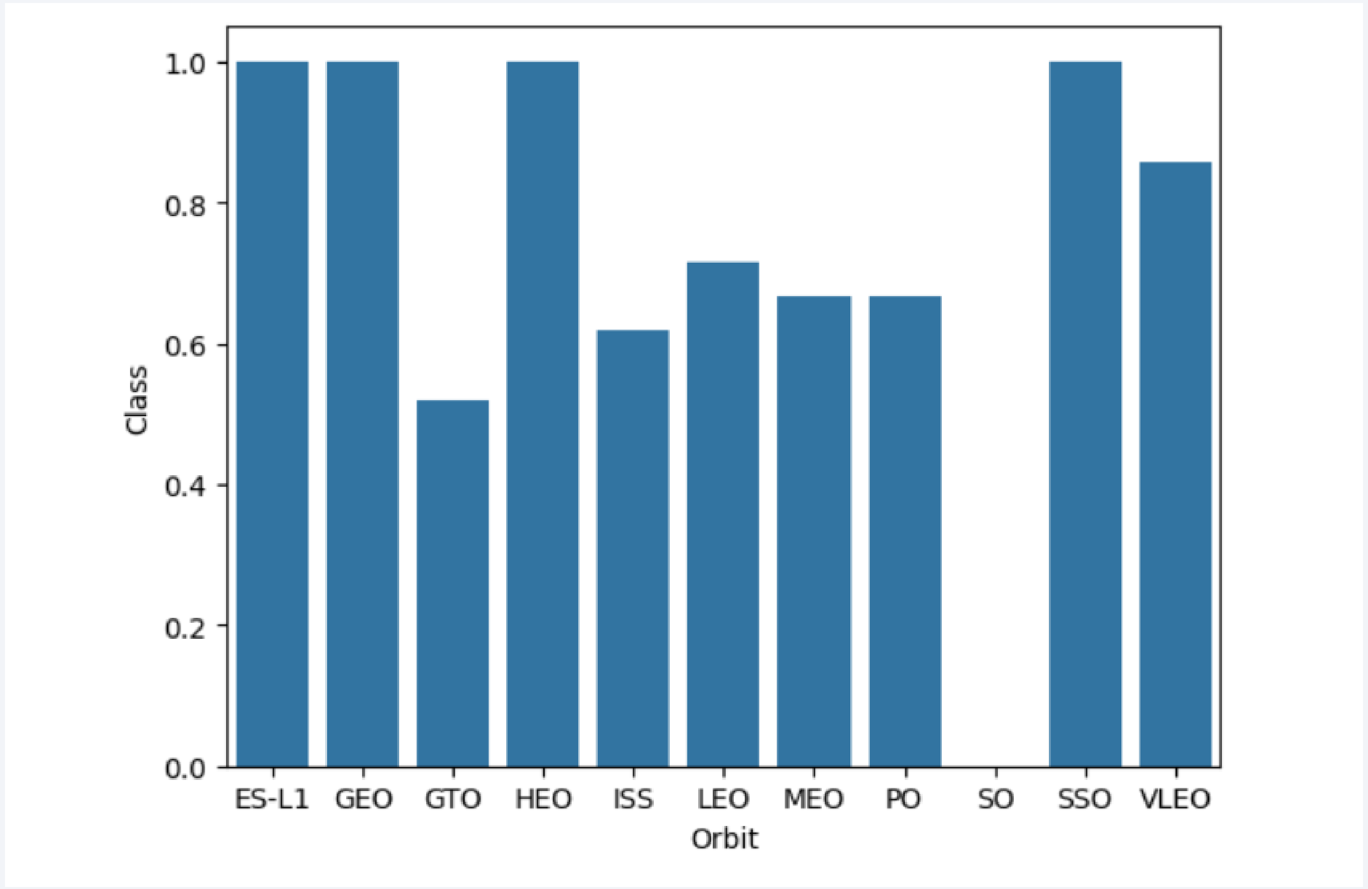
Payload vs. Launch Site

- For CCAFS SLC 40 rocket the success rate increases with increasing payload mass.



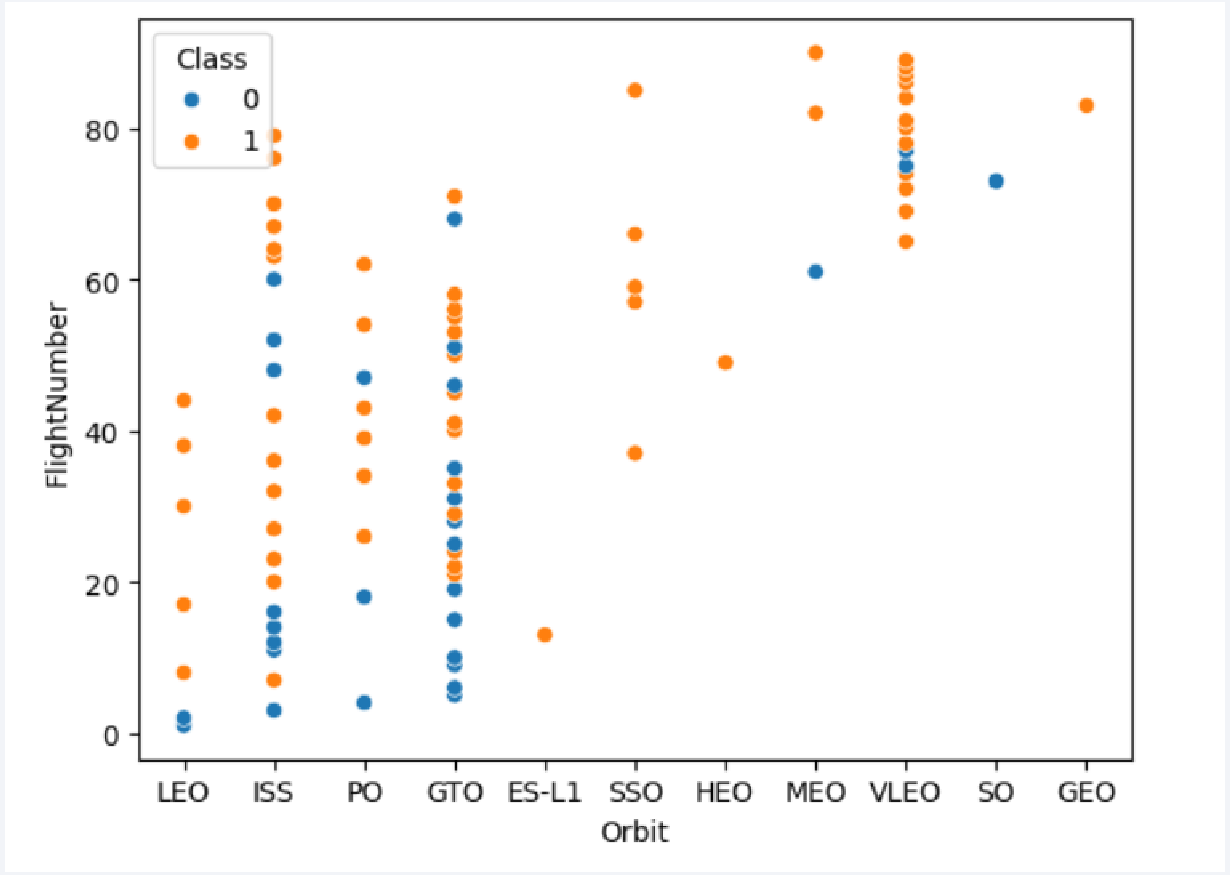
Success Rate vs. Orbit Type

- In this plot Orbit such as ES-L1, GEO, HEO, SSO and VLEO having the highest Success rate compared to other Orbits.



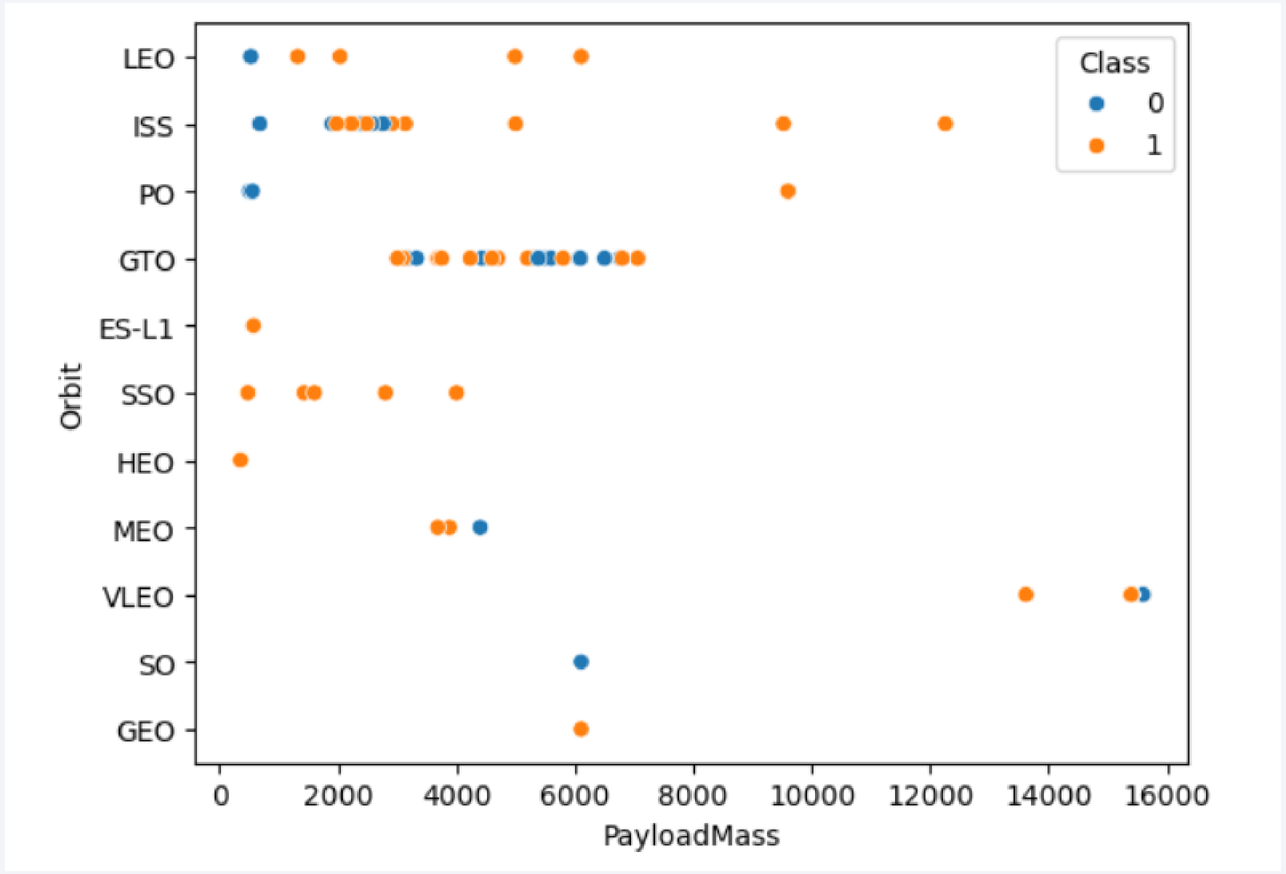
Flight Number vs. Orbit Type

- For LEO orbit success rate increases as flight number increases but on other hand in GTO orbit there is no relationship between flight number and orbit.



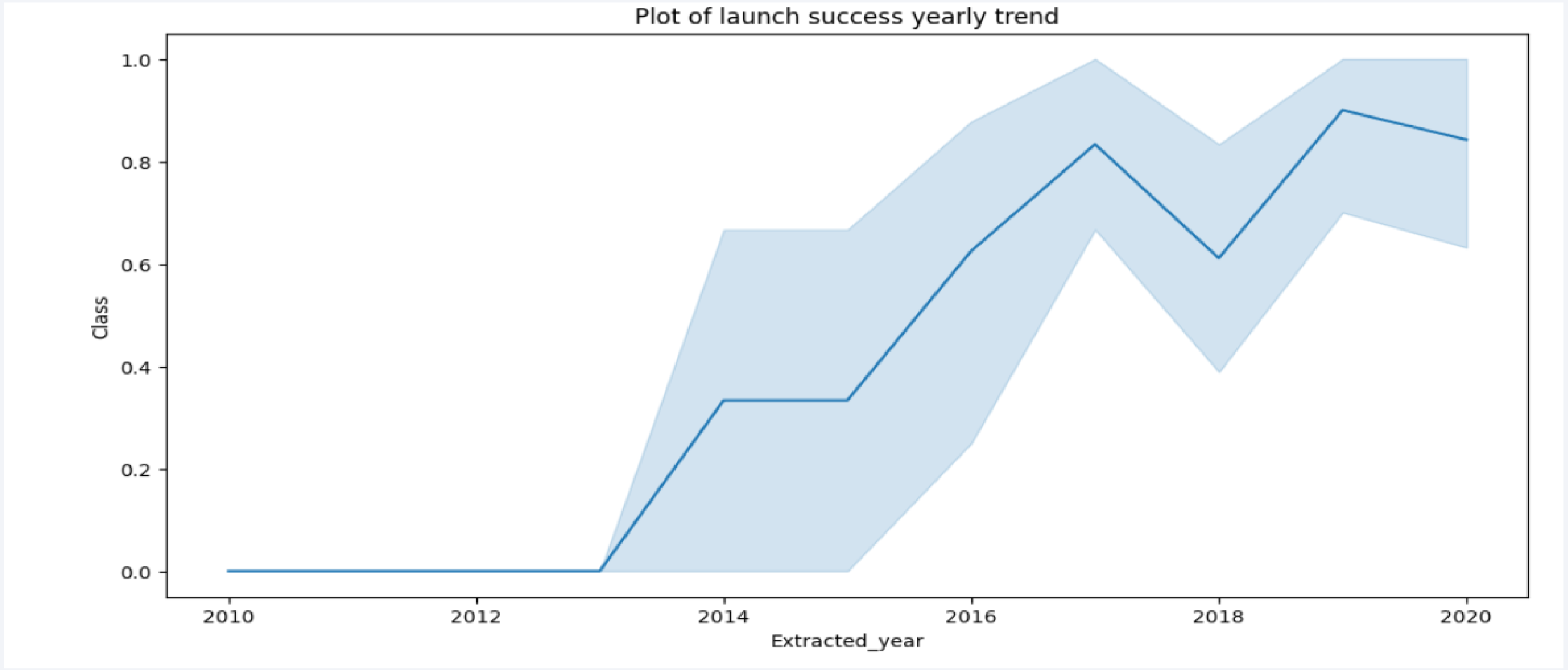
Payload vs. Orbit Type

- Success rate for LEO, ISS, PO increases for the higher payload.



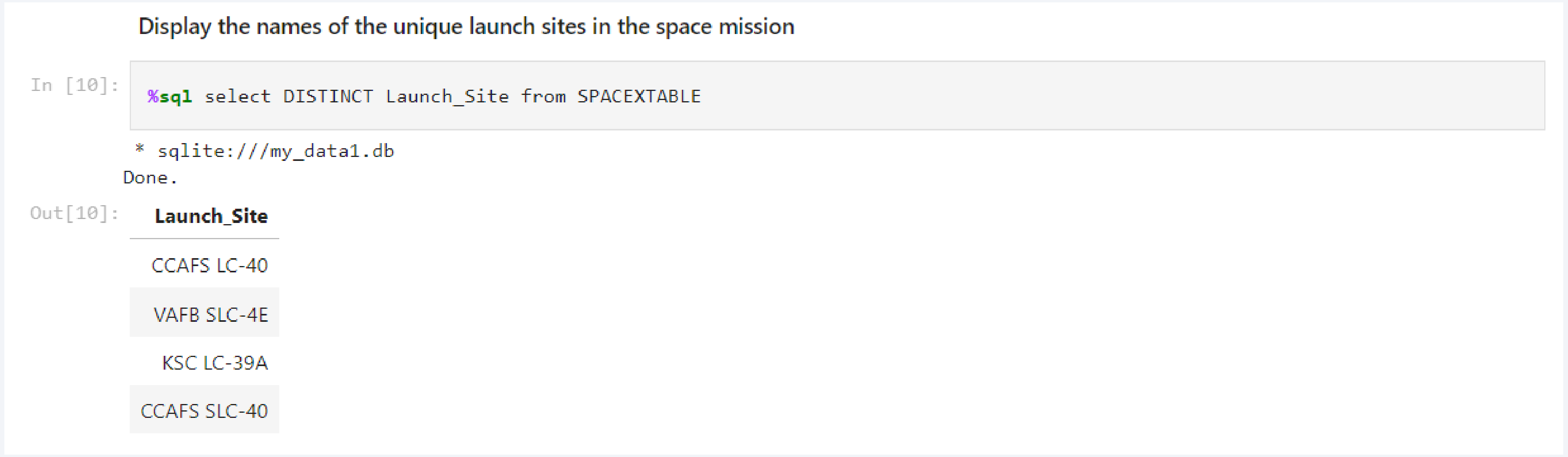
Launch Success Yearly Trend

- In the plot below we can see as the year increases from 2013 to 2020 the success rate also increases.



All Launch Site Names

- SpaceX mission have a launch sites CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, CCAFS SLC-40.
- We use keyword DISTINCT to show only unique launch sites from the SpaceX data.



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

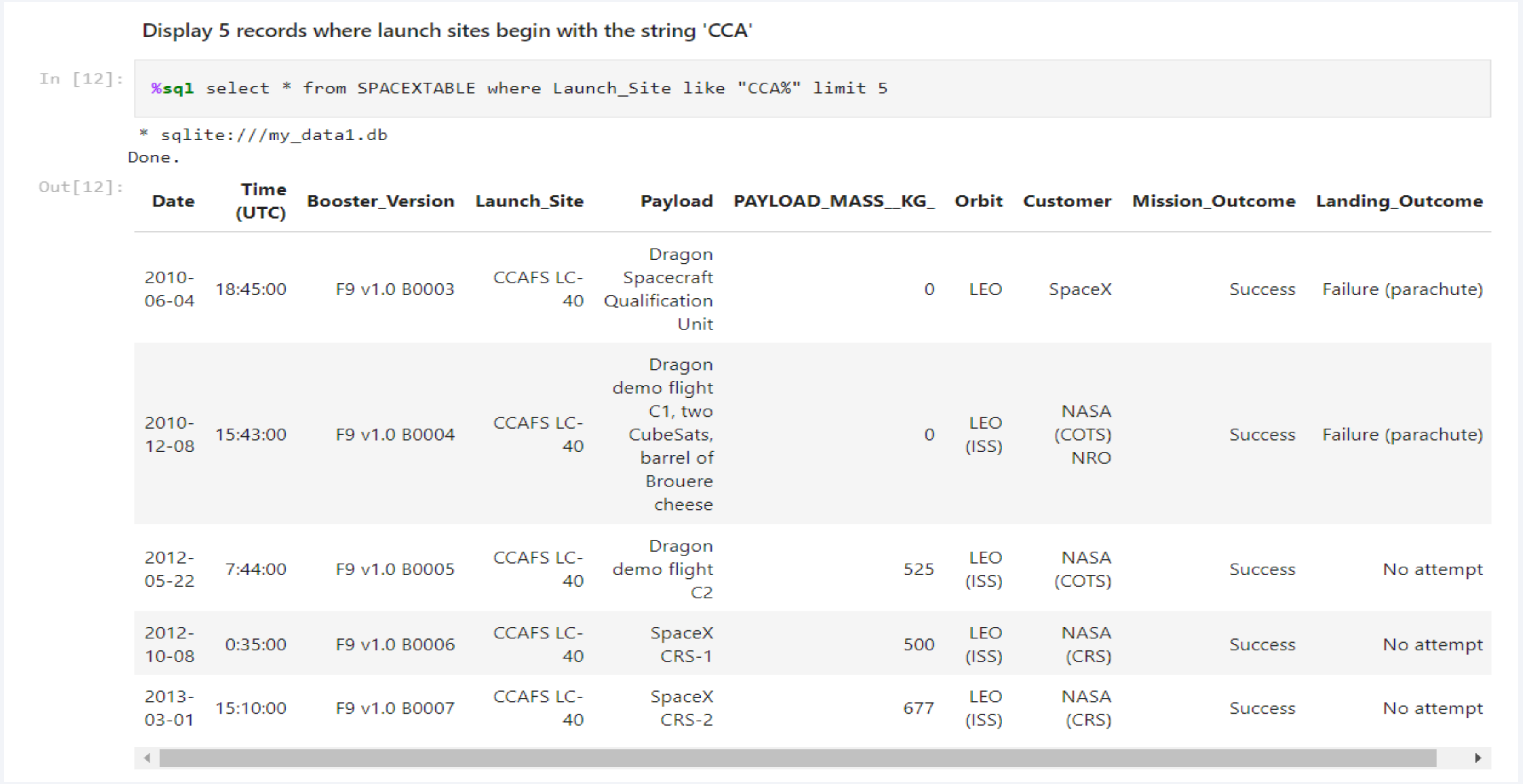
```
In [10]: %sql select DISTINCT Launch_Site from SPACEXTABLE

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

Out[10]:

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

Launch Site Names Begin with 'CCA'

- We use LIKE “CCA%” for finding all the launch site starting with CCA and LIMIT 5 for displaying only 5 rows of selected dataset.



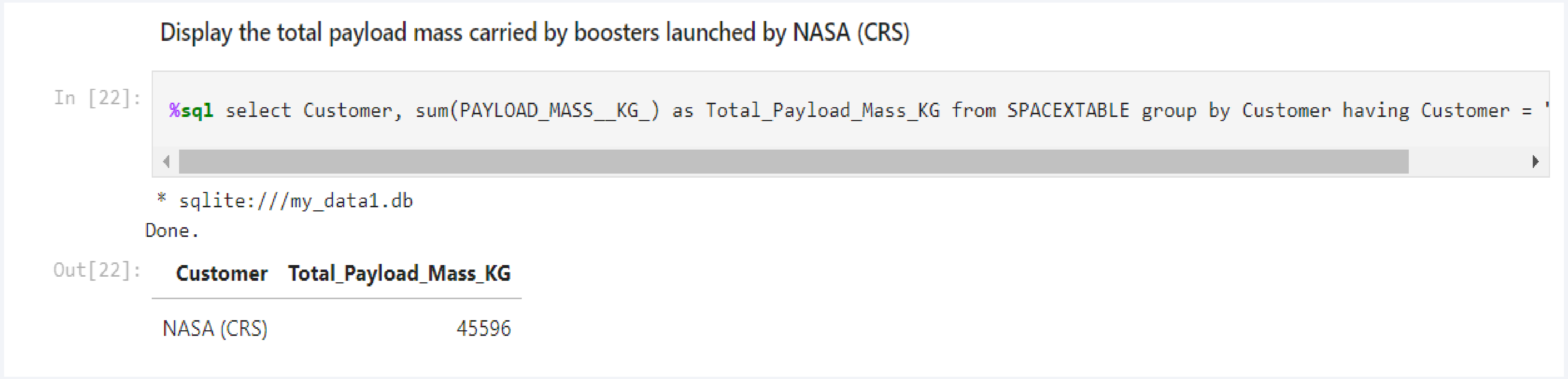
Display 5 records where launch sites begin with the string 'CCA'

```
In [12]: %sql select * from SPACEXTABLE where Launch_Site like "CCA%" limit 5

 * sqlite:///my_data1.db
Done.
Out[12]:
```

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

Total Payload Mass

- We use aggregate function SUM() with GROUP BY to find the total payload mass in kg for booster launched by NASA (CRS).



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

In [22]:

```
%sql select Customer, sum(PAYLOAD_MASS__KG_) as Total_Payload_Mass_KG from SPACEXTABLE group by Customer having Customer = '
```

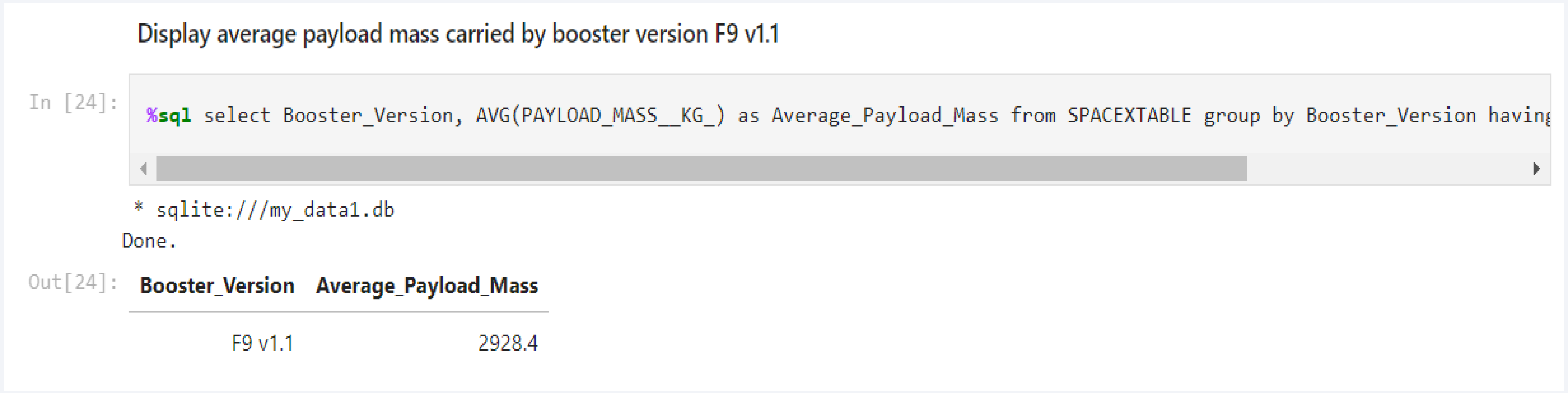
```
 * sqlite:///my_data1.db
Done.
```

Out[22]:

Customer	Total_Payload_Mass_KG
NASA (CRS)	45596

Average Payload Mass by F9 v1.1

- We calculate average payload mass using aggregate function AVG() on payload mass and GROUP BY on booster version F9 v1.1.



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

```
In [24]:  %sql select Booster_Version, AVG(PAYLOAD_MASS__KG_) as Average_Payload_Mass from SPACEXTABLE group by Booster_Version having
```

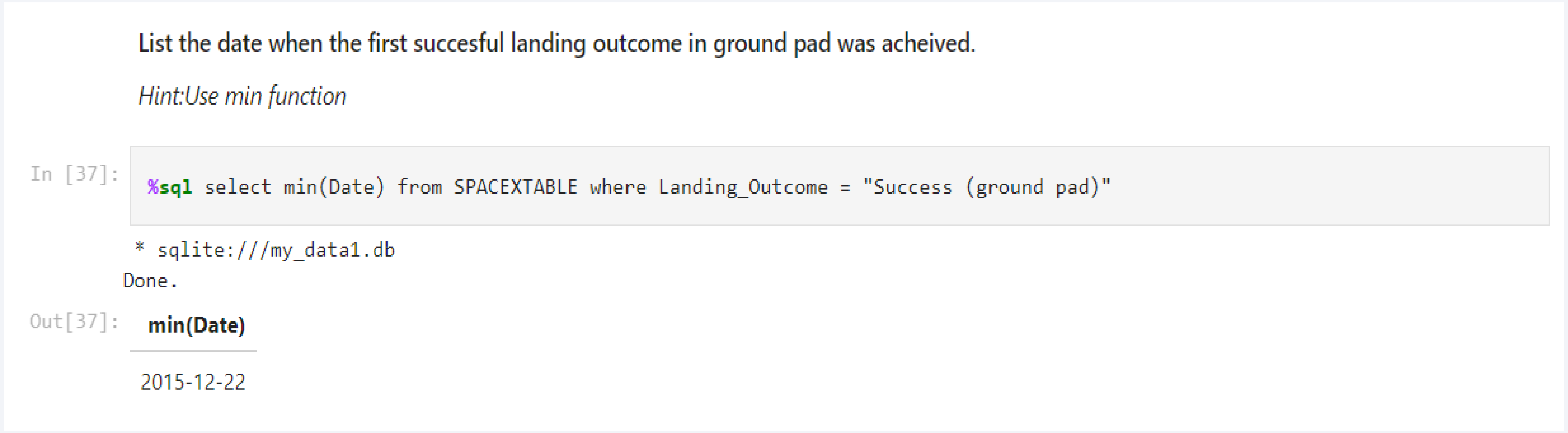
```
 * sqlite:///my_data1.db
Done.
```

Out[24]:

Booster_Version	Average_Payload_Mass
F9 v1.1	2928.4

First Successful Ground Landing Date

- We use min(Date) with where clause to find the first successful ground landing date.



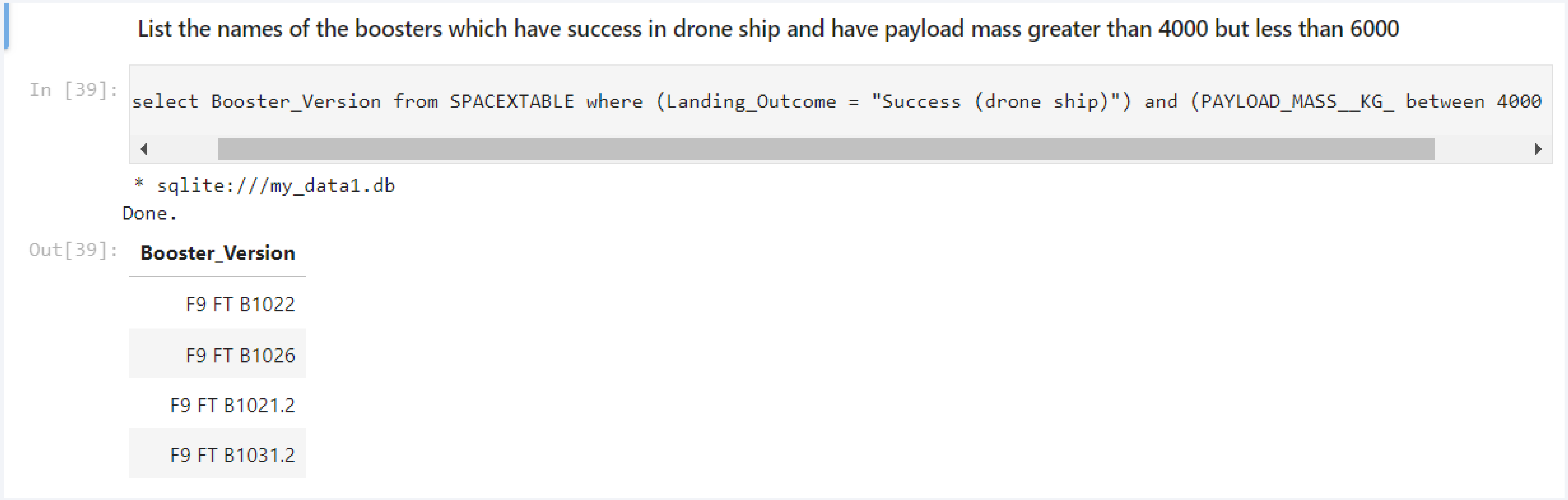
List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

```
In [37]: %sql select min(Date) from SPACEXTABLE where Landing_Outcome = "Success (ground pad)"

 * sqlite:///my_data1.db
Done.

Out[37]: min(Date)
         2015-12-22
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- We use where clause with AND operator and BETWEEN operator on landing outcomes and payload mass in kg for finding the successful drone ship landing with payload between 4000 and 6000.



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

```
In [39]: select Booster_Version from SPACEXTABLE where (Landing_Outcome = "Success (drone ship)") and (PAYLOAD_MASS__KG_ between 4000
```

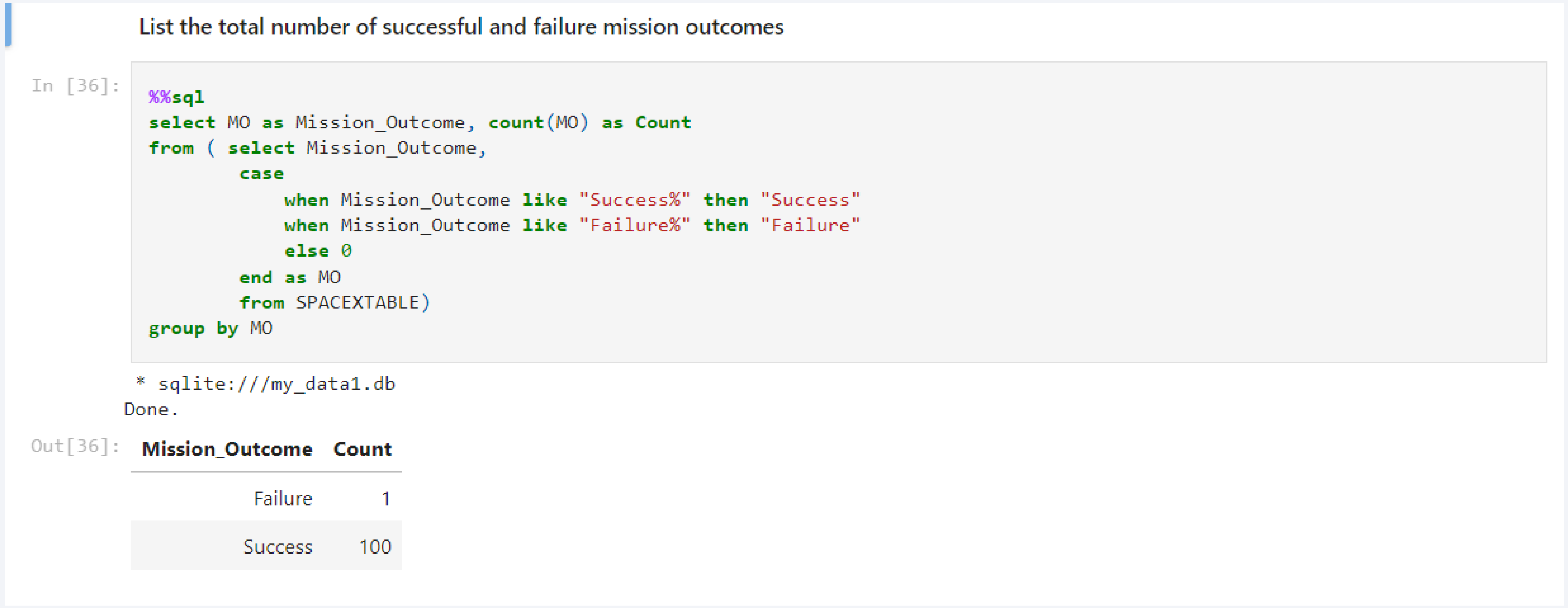
```
 * sqlite:///my_data1.db
Done.
```

Out[39]:

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

Total Number of Successful and Failure Mission Outcomes

- We use COUNT() with GROUPBY clause to find the total number of success and failure using the subquery to convert each “Success%” like statement and “Failure%” like statement into “Success” and “Failure”.



List the total number of successful and failure mission outcomes

```
In [36]:
%%sql
select MO as Mission_Outcome, count(MO) as Count
from ( select Mission_Outcome,
        case
            when Mission_Outcome like "Success%" then "Success"
            when Mission_Outcome like "Failure%" then "Failure"
            else 0
        end as MO
        from SPACEXTABLE)
group by MO
```

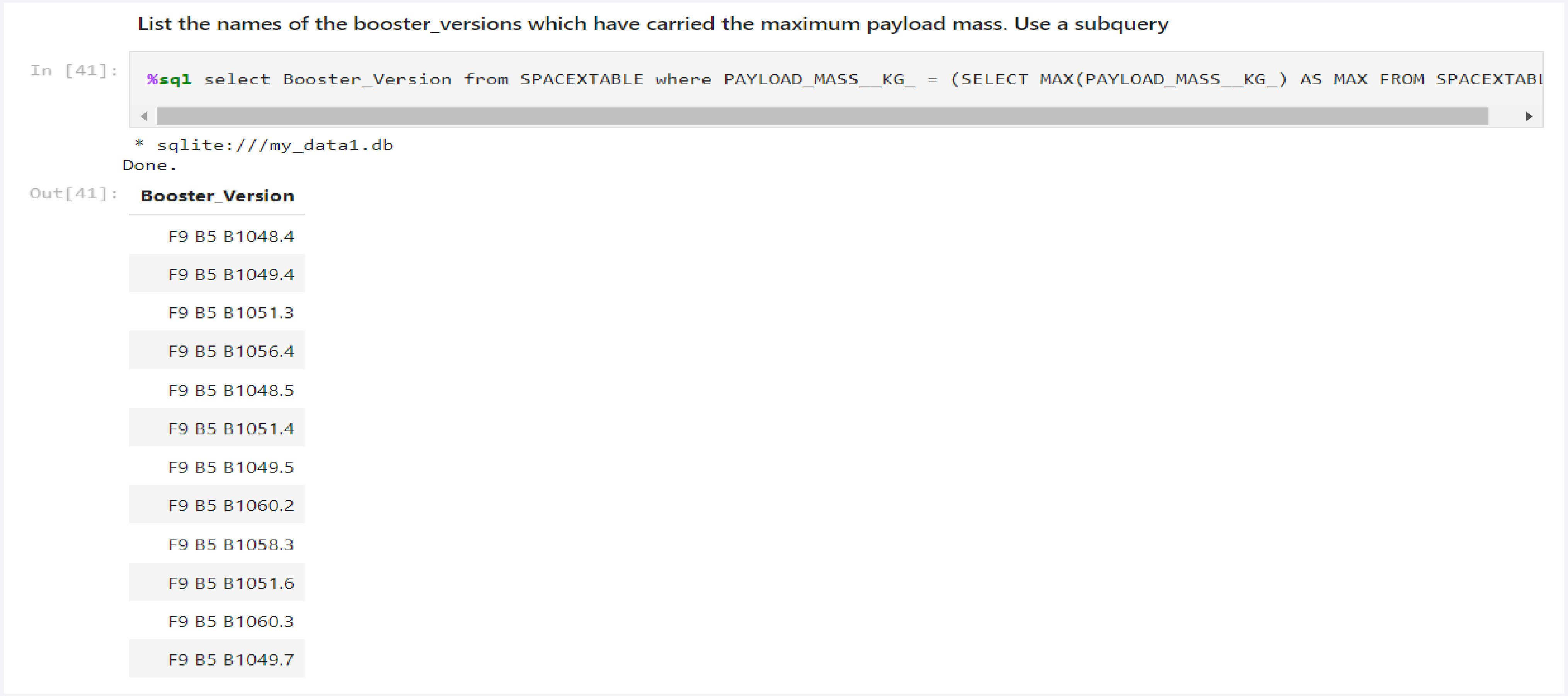
```
 * sqlite:///my_data1.db
Done.
```

Out[36]:

Mission_Outcome	Count
Failure	1
Success	100

Boosters Carried Maximum Payload

- We use subquery with MAX() function to find the maximum payload in SpaceX table and find the booster version with max payload using where clause in main query.



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

```
In [41]:  %sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) AS MAX FROM SPACEXTABL
```

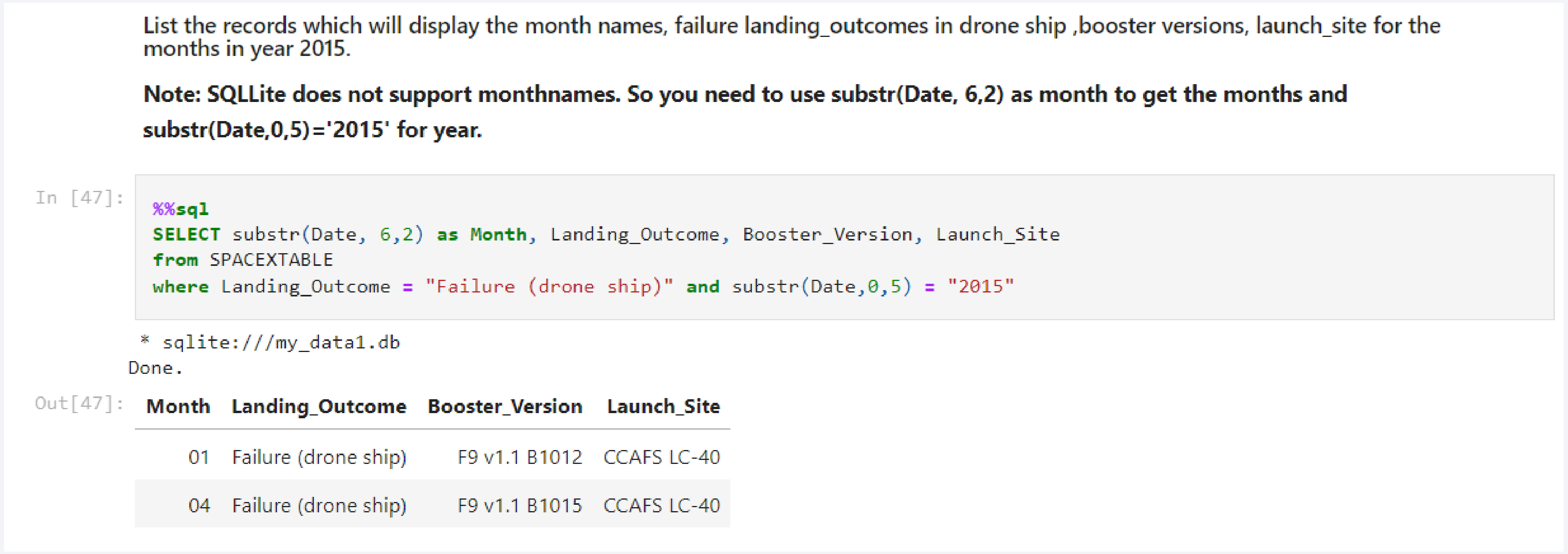
```
 * sqlite:///my_data1.db
Done.
```

Out[41]:

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

2015 Launch Records

- We use SUBSTR() to find the month and year with where clause to find the launch records of 2015 failure landing outcomes.



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

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

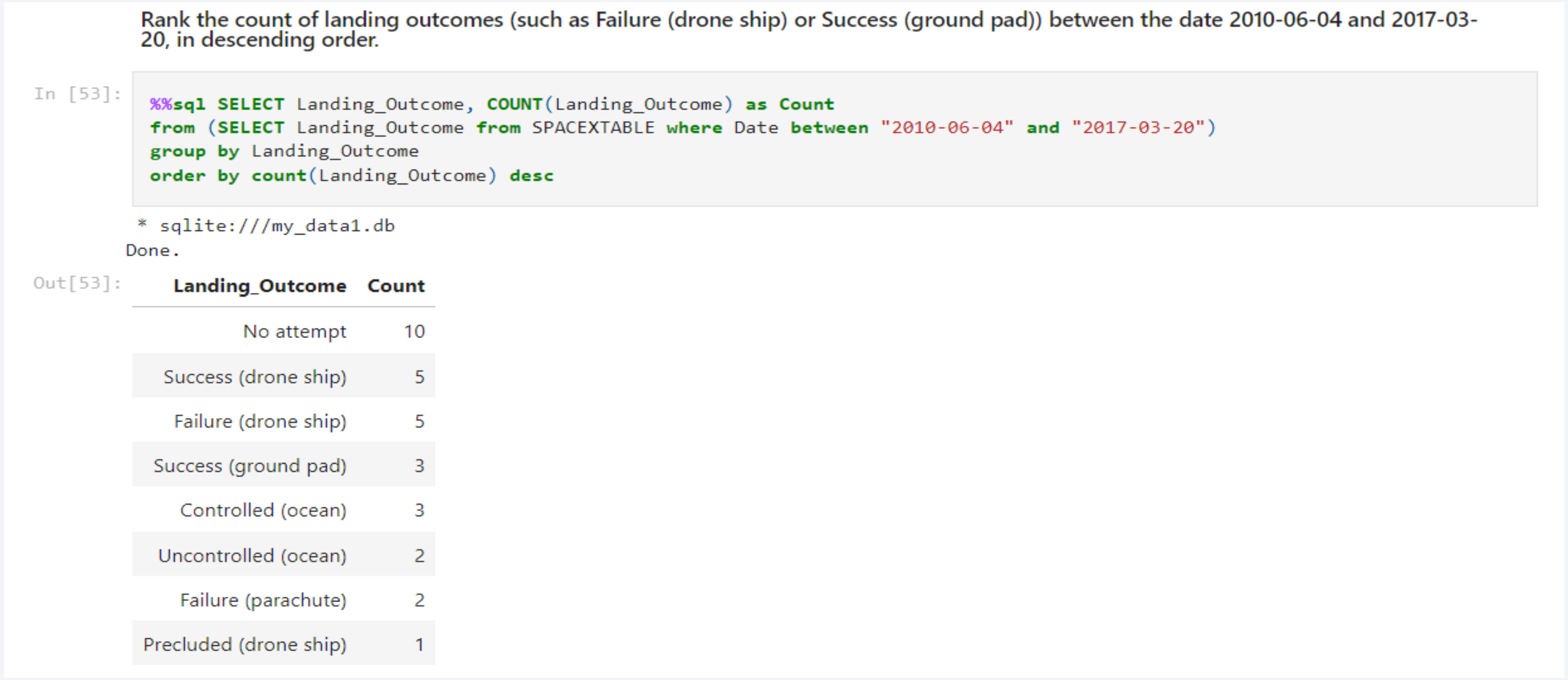
```
In [47]: %%sql
         SELECT substr(Date, 6,2) as Month, Landing_Outcome, Booster_Version, Launch_Site
         from SPACEXTABLE
         where Landing_Outcome = "Failure (drone ship)" and substr(Date,0,5) = "2015"

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

Out[47]:

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

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

- We select landing outcomes and the count of landing outcome using subquery with between clause for date 2010-06-01 to 2017-02-20.
- Then we use GROUP BY and ORDER BY clause for getting landing outcomes in descending order.


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 [53]:
```
%%sql SELECT Landing_Outcome, COUNT(Landing_Outcome) as Count
from (SELECT Landing_Outcome from SPACEXTABLE where Date between "2010-06-04" and "2017-03-20")
group by Landing_Outcome
order by count(Landing_Outcome) desc
```

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

Out[53]:

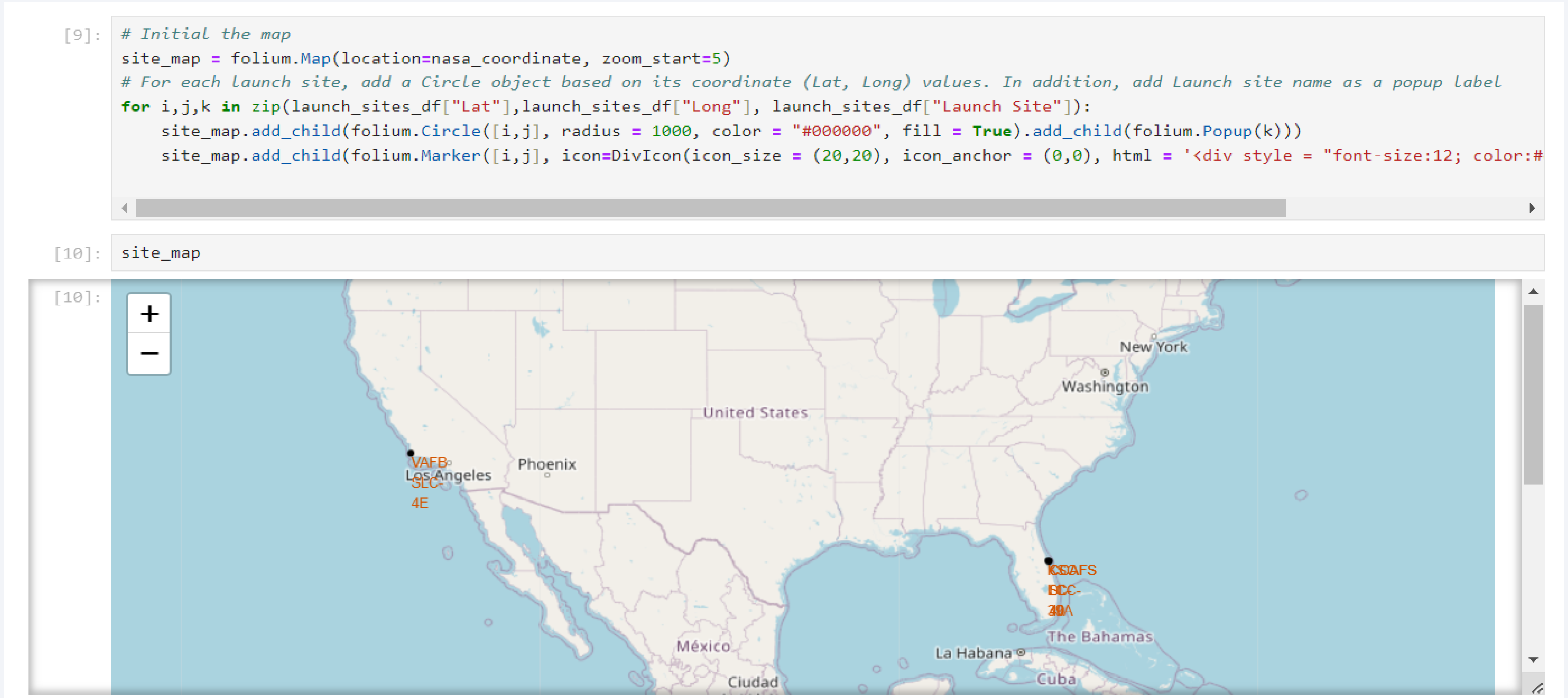
Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

All Launch Sites Mark on Map

- We have used the marker for marking all the locations where the different unique Launch Sites are on the Map.



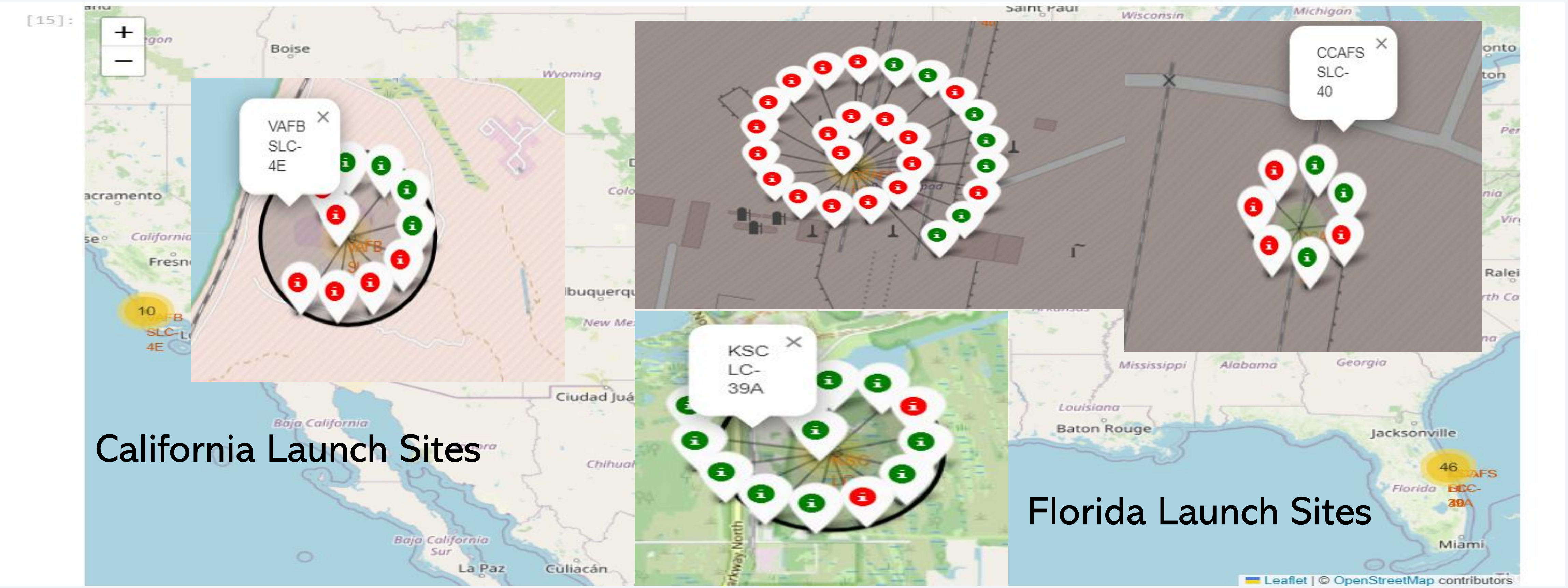
```
[9]: # Initial the map
     site_map = folium.Map(location=nasa_coordinate, zoom_start=5)
     # For each launch site, add a Circle object based on its coordinate (Lat, Long) values. In addition, add Launch site name as a popup label
     for i,j,k in zip(launch_sites_df["Lat"],launch_sites_df["Long"], launch_sites_df["Launch Site"]):
         site_map.add_child(folium.Circle([i,j], radius = 1000, color = "#000000", fill = True).add_child(folium.Popup(k)))
         site_map.add_child(folium.Marker([i,j], icon=DivIcon(icon_size = (20,20), icon_anchor = (0,0), html = '<div style = "font-size:12; color:#
```

```
[10]: site_map
```

[10]:

Marking All Success/Failed Launches For Each Site On The Map

- We will mark all the success and failure for each launch sites on map using Marker Cluster and Marker.
- Green color is given for success and Red for failure.

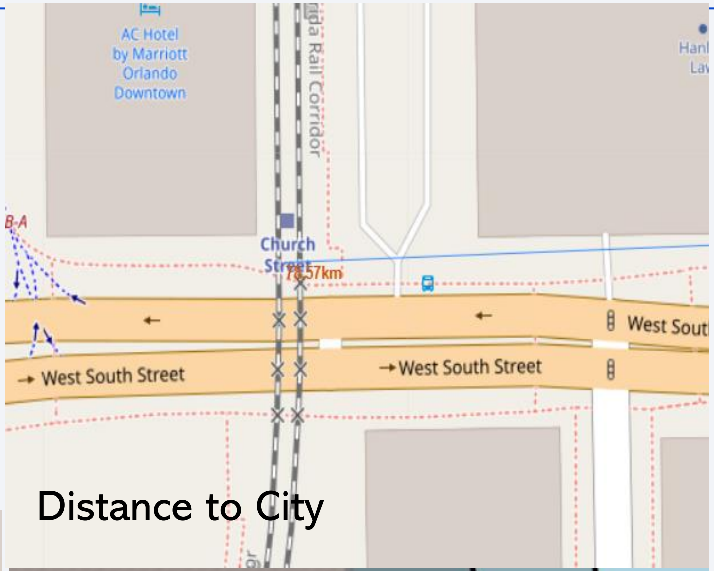


California Launch Sites

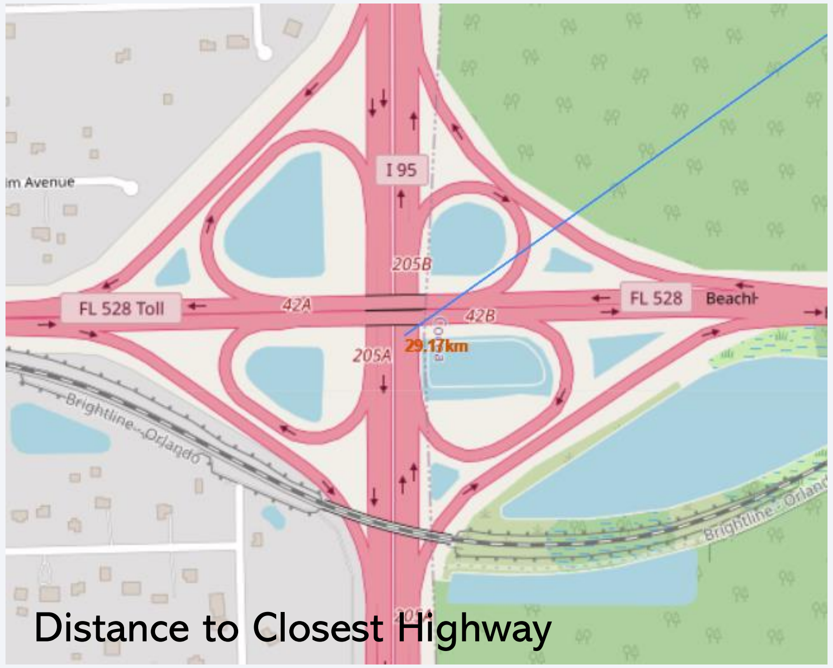
Florida Launch Sites

Distance Between a Launch Site to its Proximities

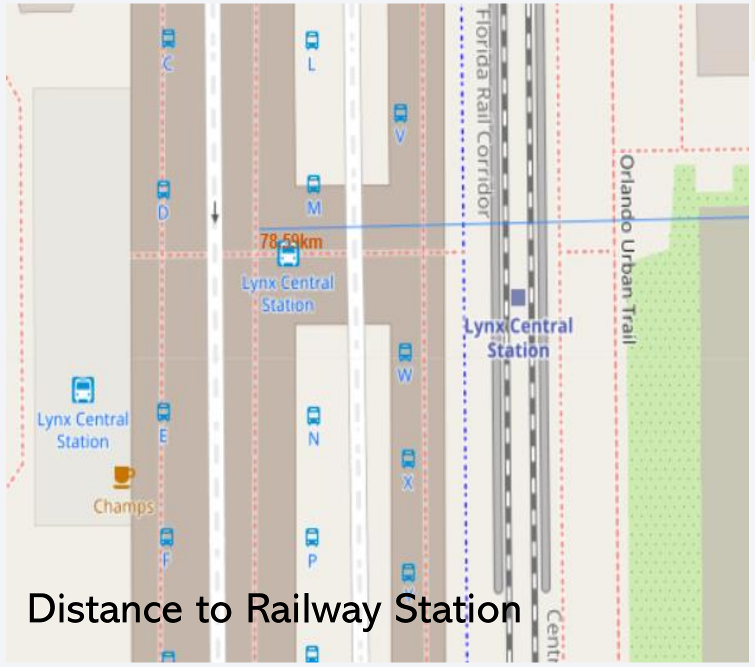
- Are launch sites in close proximity to railways? NO
- Are launch sites in close proximity to highways? NO
- Are launch sites in close proximity to coastline? YES
- Do launch sites keep certain distance away from cities? YES



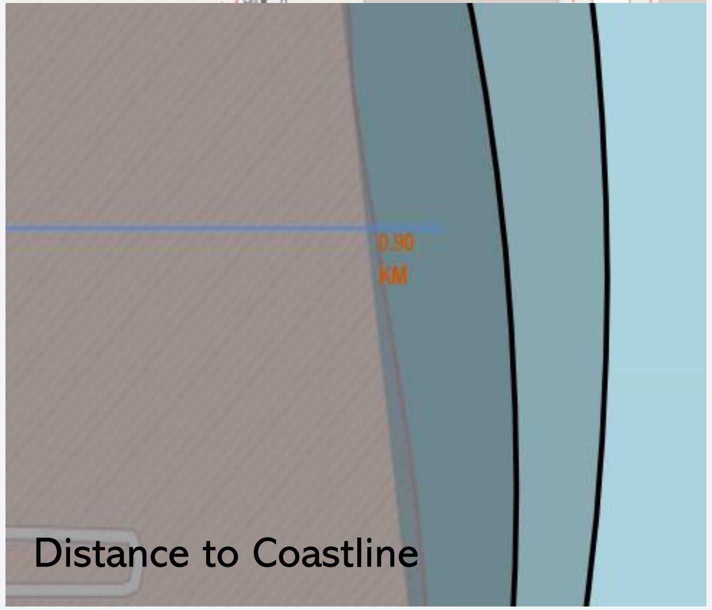
Distance to City



Distance to Closest Highway



Distance to Railway Station



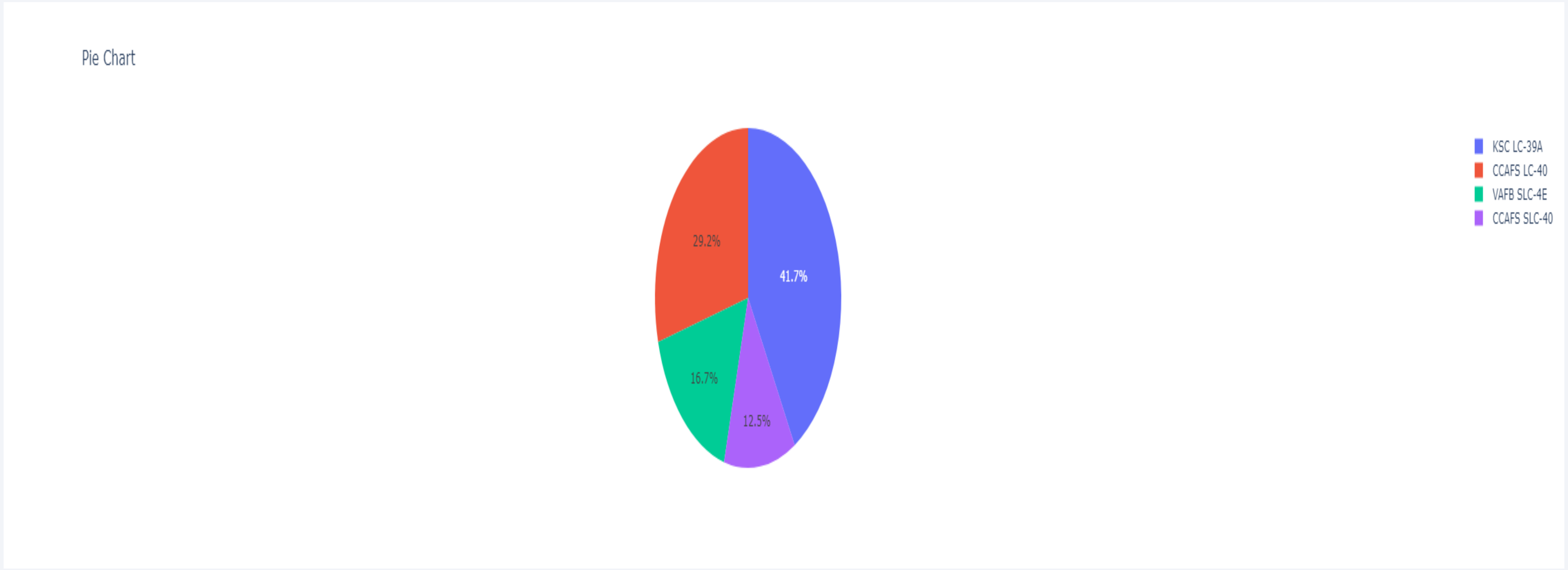
Distance to Coastline

Section 4

Build a Dashboard with Plotly Dash

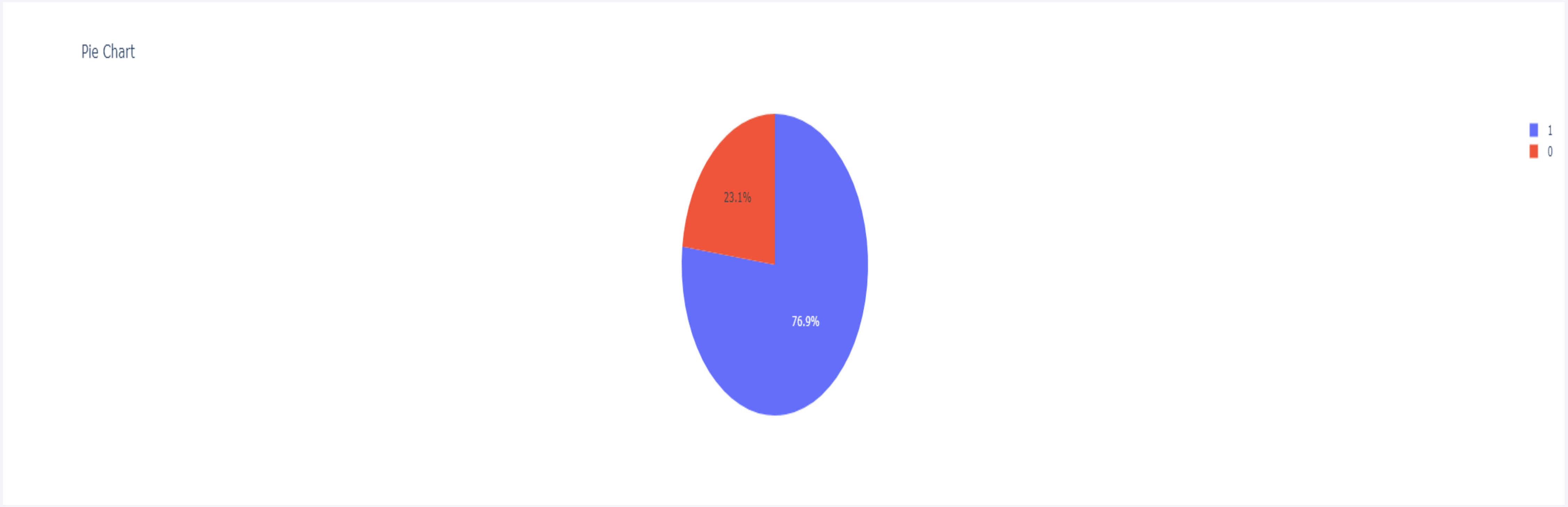
Pie Chart showing success percentage achieved by each launch site

- KSC LC-39A having a largest percentage of success between all launch sites.



Launch Site having highest launch success ratio

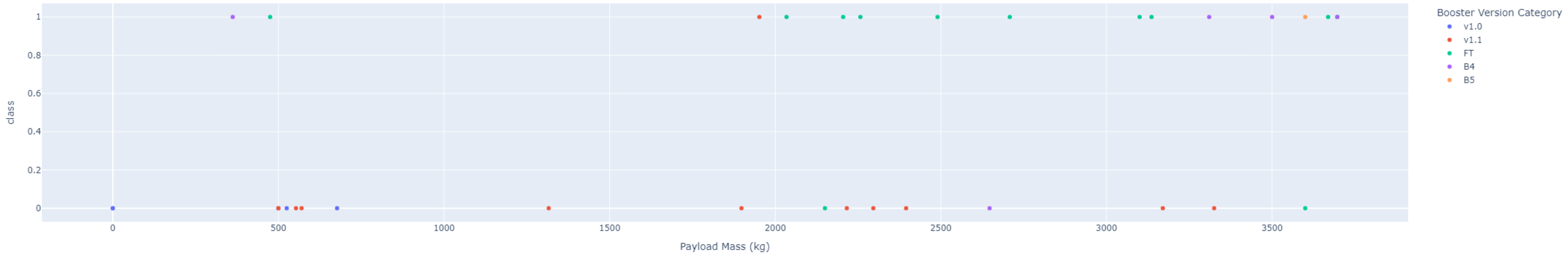
- KSC LC-39A is having a largest success ratio of 76.9%.



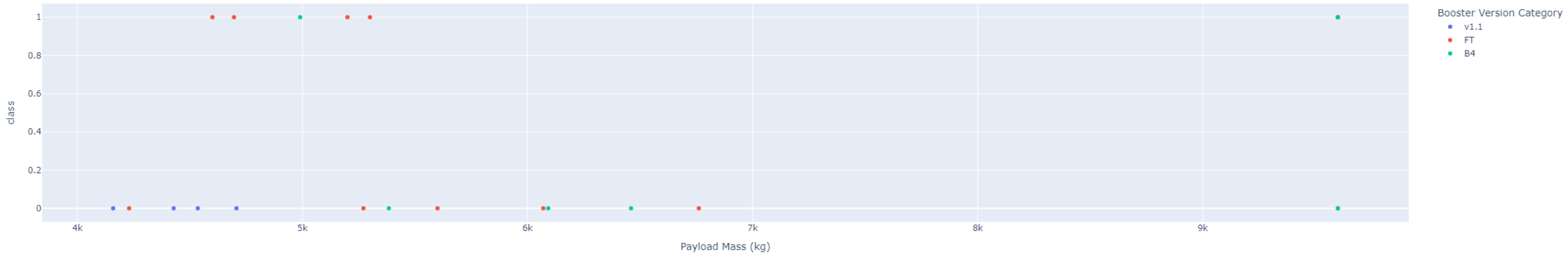
Scatter Plot for all sites with different payload selected in the range slider

- Success rate of less weighted payload is higher than heavy loaded payload.

Payload from range 0 kg to 4000 kg



Payload from range 4000 kg to 10000 kg

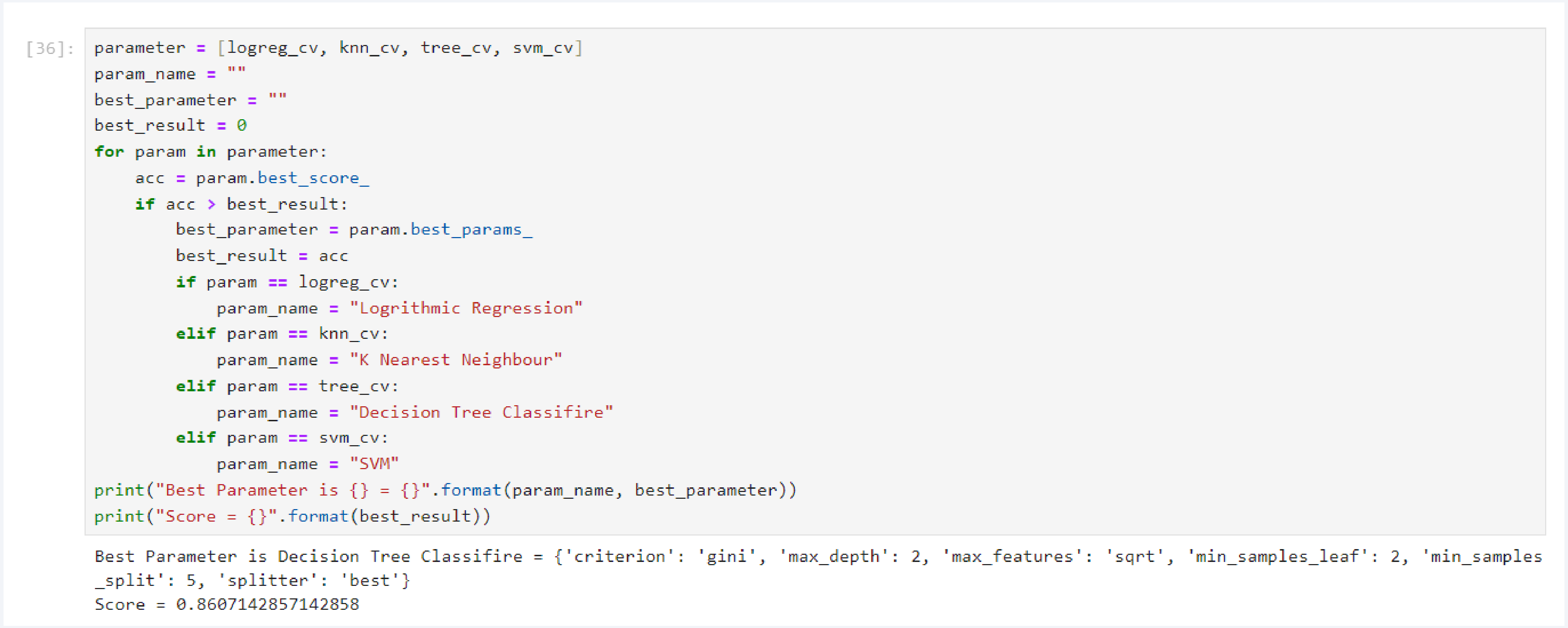


Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The Decision Tree Classifier is the best model with the highest classification accuracy.

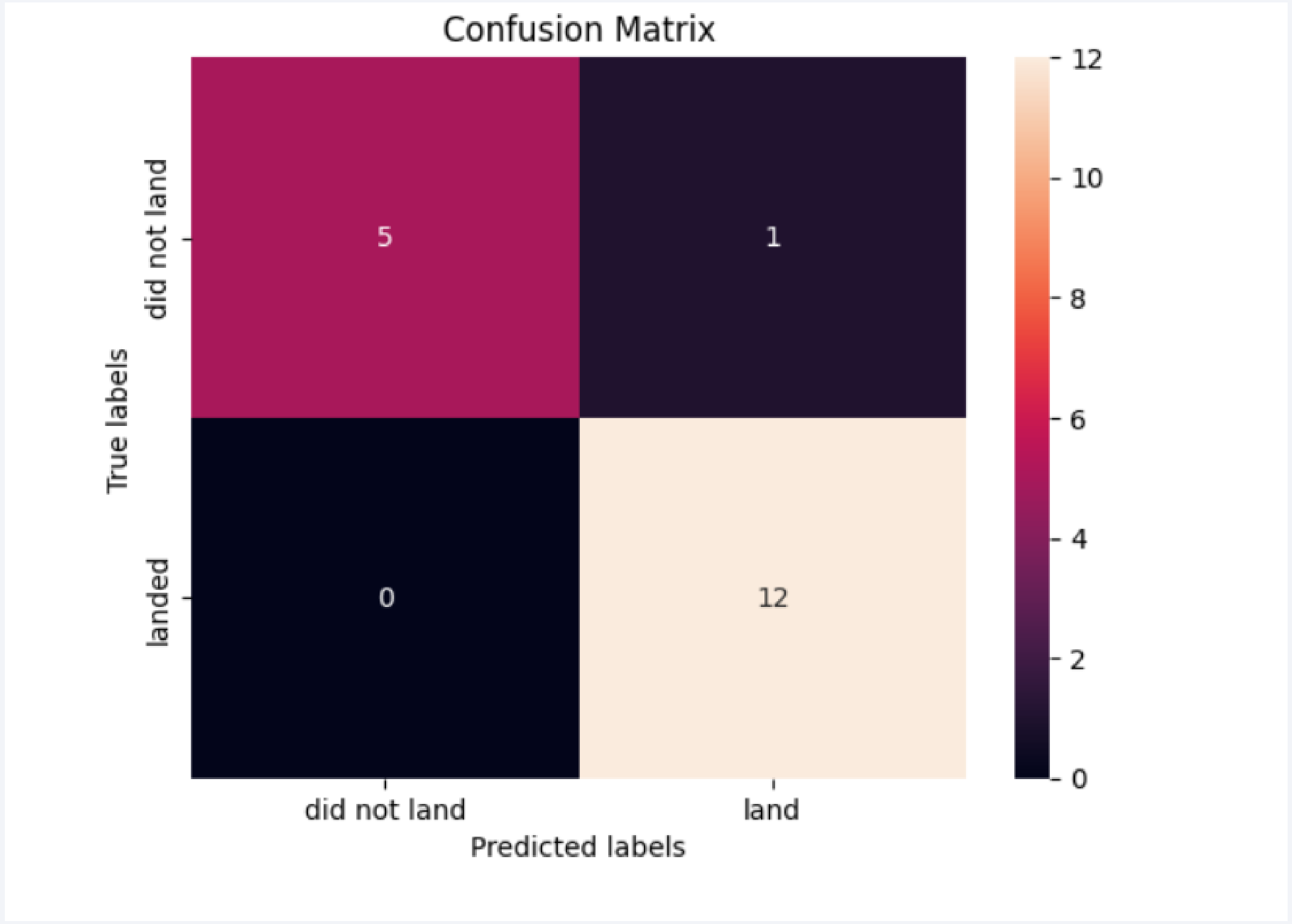


```
[36]: parameter = [logreg_cv, knn_cv, tree_cv, svm_cv]
      param_name = ""
      best_parameter = ""
      best_result = 0
      for param in parameter:
          acc = param.best_score_
          if acc > best_result:
              best_parameter = param.best_params_
              best_result = acc
              if param == logreg_cv:
                  param_name = "Logrithmic Regression"
              elif param == knn_cv:
                  param_name = "K Nearest Neighbour"
              elif param == tree_cv:
                  param_name = "Decision Tree Classifire"
              elif param == svm_cv:
                  param_name = "SVM"
      print("Best Parameter is {} = {}".format(param_name, best_parameter))
      print("Score = {}".format(best_result))
```

```
Best Parameter is Decision Tree Classifire = {'criterion': 'gini', 'max_depth': 2, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples
_split': 5, 'splitter': 'best'}
Score = 0.8607142857142858
```

Confusion Matrix

- The confusion matrix of decision tree classifier showing one false positive in which the did not land flight has been marked as landed by classifier.



Conclusions

We can conclude that:

- The larger the flight number, the greater the success rate at a launch site.
- Launch success rate kept on increasing from 2013 to 2020.
- ES-L1, GEO, HEO, SSO and VLEO orbit having the most success rate than other orbits.
- KSC LC-39A is having the highest success ratio that all other launch site.
- Success rate of less weighted payload is higher than heavy weighted payload.
- Decision Tree Classifier is having a best accuracy for training this model.

Thank you!