IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Yuan Chung Ho

2nd March 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
  - I predicted if the Falcon 9 first stage will land successfully. SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.
- Problems you want to find answers
  - What factors does it impact the rocket land successfully ?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions does SpaceX have to achieve to get the best results and make the best rocket success landing rate ?

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 on data field for ML models.
- 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

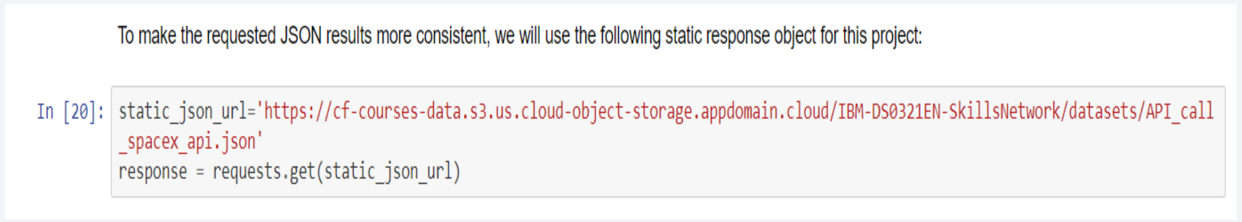
- The data was collected and cleaned using various methods.
    - Using get request to the SpaceX API.
    - Decoded the response content as a Json string using .json() function and turn it into a pandas data frame with .json_normalize().
    - Cleaned the data, checked for missing values and filled in missing value with .mean() function.
    - 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 data frame for further analysis.

# Data Collection – SpaceX API

- Used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.

- Link to the Jupyter notebook. https://github.com/yuanchung1987/Data-Science-Final-Project/blob/master/Data%20Collection%20API.ipynb

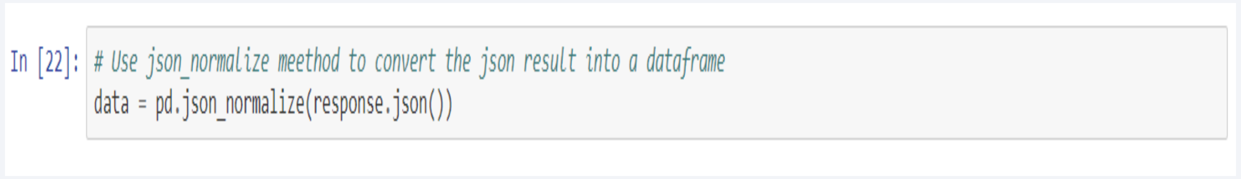
1. Get request with launch data using SpaceX API.



To make the requested JSON results more consistent, we will use the following static response object for this project:

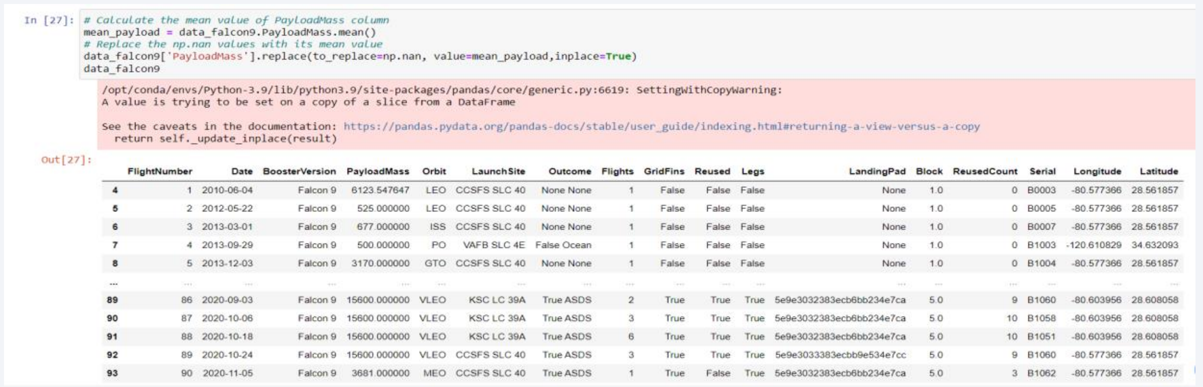
```
In [20]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call
         _spacex_api.json'
         response = requests.get(static_json_url)
```

2. Use json_normalize method to convert json result to dataframe.



```
In [22]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

2. Performed data cleaning and filing the missing values.



```
In [27]: # Calculate the mean value of PayloadMass column
         mean_payload = data_falcon9.PayloadMass.mean()
         # Replace the np.nan values with its mean value
         data_falcon9['PayloadMass'].replace(to_replace=np.nan, value=mean_payload,inplace=True)
         data_falcon9

/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/pandas/core/generic.py:6619: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return self._update_inplace(result)
```

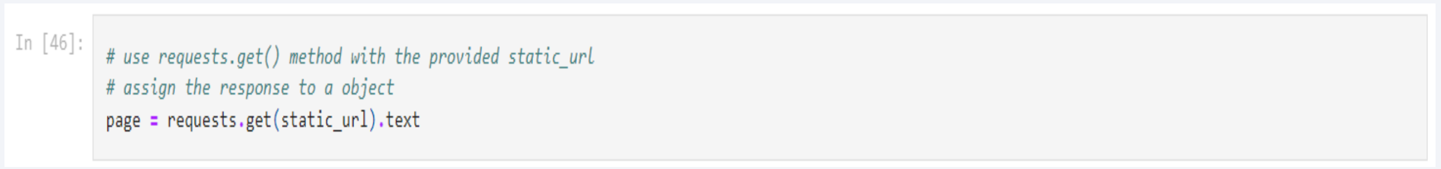
Out[27]:

| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad | Block | ReusedCount | Serial | Longitude | Latitude |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 1 | 2010-06-04 | Falcon 9 | 6123.547647 | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B0003 | -80.577366 | 28.561857 |
| 5 | 2 | 2012-05-22 | Falcon 9 | 525.000000 | LEO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B0005 | -80.577366 | 28.561857 |
| 6 | 3 | 2013-03-01 | Falcon 9 | 677.000000 | ISS | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B0007 | -80.577366 | 28.561857 |
| 7 | 4 | 2013-09-29 | Falcon 9 | 500.000000 | PO | VAFB SLC 4E | False Ocean | 1 | False | False | False | None | 1.0 | 0 | B1003 | -120.610829 | 34.632093 |
| 8 | 5 | 2013-12-03 | Falcon 9 | 3170.000000 | GTO | CCSFS SLC 40 | None None | 1 | False | False | False | None | 1.0 | 0 | B1004 | -80.577366 | 28.561857 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 89 | 86 | 2020-09-03 | Falcon 9 | 15600.000000 | VLEO | KSC LC 39A | True ASDS | 2 | True | True | True | 5e9e3032383ecb6bb234e7ca | 5.0 | 9 | B1060 | -80.603956 | 28.608058 |
| 90 | 87 | 2020-10-06 | Falcon 9 | 15600.000000 | VLEO | KSC LC 39A | True ASDS | 3 | True | True | True | 5e9e3032383ecb6bb234e7ca | 5.0 | 10 | B1058 | -80.603956 | 28.608058 |
| 91 | 88 | 2020-10-18 | Falcon 9 | 15600.000000 | VLEO | KSC LC 39A | True ASDS | 6 | True | True | True | 5e9e3032383ecb6bb234e7ca | 5.0 | 10 | B1051 | -80.603956 | 28.608058 |
| 92 | 89 | 2020-10-24 | Falcon 9 | 15600.000000 | VLEO | CCSFS SLC 40 | True ASDS | 3 | True | True | True | 5e9e3033383ecbb9e534e7cc | 5.0 | 9 | B1060 | -80.577366 | 28.561857 |
| 93 | 90 | 2020-11-05 | Falcon 9 | 3681.000000 | MEO | CCSFS SLC 40 | True ASDS | 1 | True | False | True | 5e9e3032383ecb6bb234e7ca | 5.0 | 3 | B1062 | -80.577366 | 28.561857 |

## Data Collection - Scraping

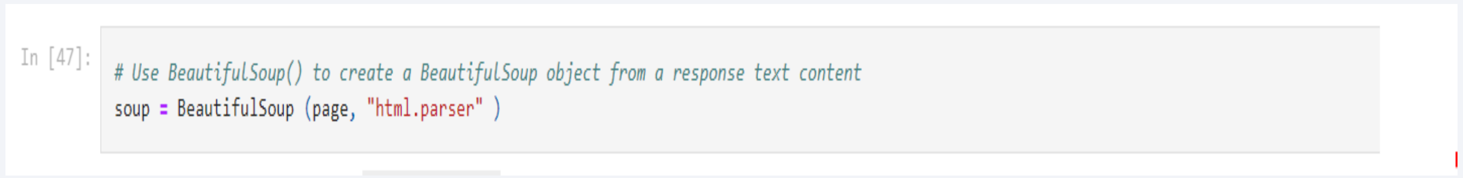
- Applied web scrapping to wikipedia Falcon 9 launch records with BeautifulSoup.
- Parsed the table and converted it into a pandas dataframe.
- Link as below. https://github.com/yuanchung1987/Data-Science-Final-Project/blob/master/Data%20Collection%20With%20Web%20Scraping%20Lab.ipynb

1. Applied HTTP Get method to request the Falcon 9 wiki page.



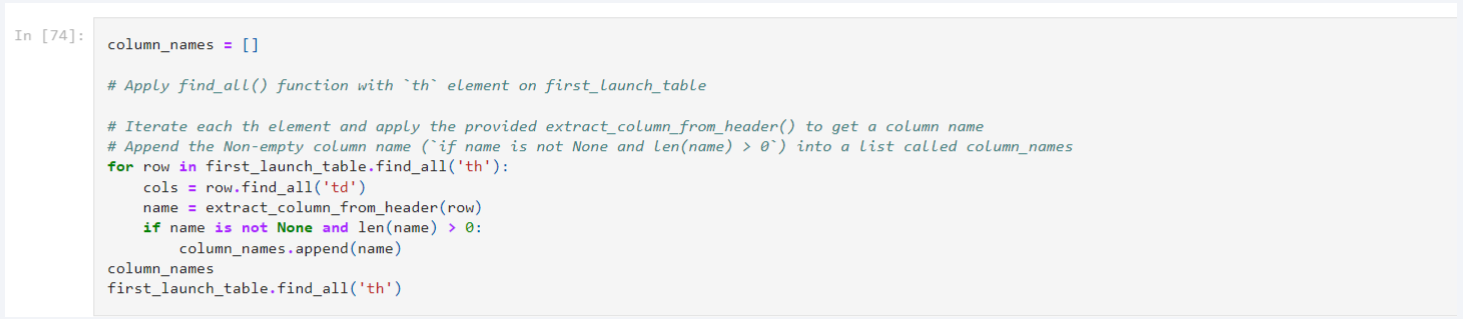
```
In [46]:
# use requests.get() method with the provided static_url
# assign the response to a object
page = requests.get(static_url).text
```

2. Create BeautifulSoup object from the HTML response.



```
In [47]:
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup (page, "html.parser" )
```

3. Extract all column names from the HTML table header.



```
In [74]:
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'):
    cols = row.find_all('td')
    name = extract_column_from_header(row)
    if name is not None and len(name) > 0:
        column_names.append(name)
column_names
first_launch_table.find_all('th')
```

4. Create a dataframe by parsing the launch HTML tables.
5. Export the data to CSV file.

## Data Wrangling

- Objective to perform exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site, and the number and occurrence of each orbits
- Created landing outcome label from outcome column and exported the results to csv.
- The link to Jupyter notebook is https://github.com/yuanchung1987/Data-Science-Final-Project/blob/master/EDA-%20Data%20Wrangling.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 with scatterplot and line chart to find the pattern of each data feature.
- The link to Jupyter notebook is: https://github.com/yuanchung1987/Data-Science-Final-Project/blob/master/EDA-%20Visualization.ipynb

## EDA with SQL

- Loaded the SpaceX dataset into a IBM db2 database for data storage.
- Applied EDA with SQL to get insight from the data. 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.
- Link to Jupyter notebook: https://github.com/yuanchung1987/Data-Science-Final-Project/blob/master/EDA-%20SQL.ipynb

## Build an Interactive Map with Folium

- 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.
- 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.
- Link to Jupyter notebook: https://github.com/yuanchung1987/Data-Science-Final-Project/blob/master/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

## Build a Dashboard with Plotly Dash

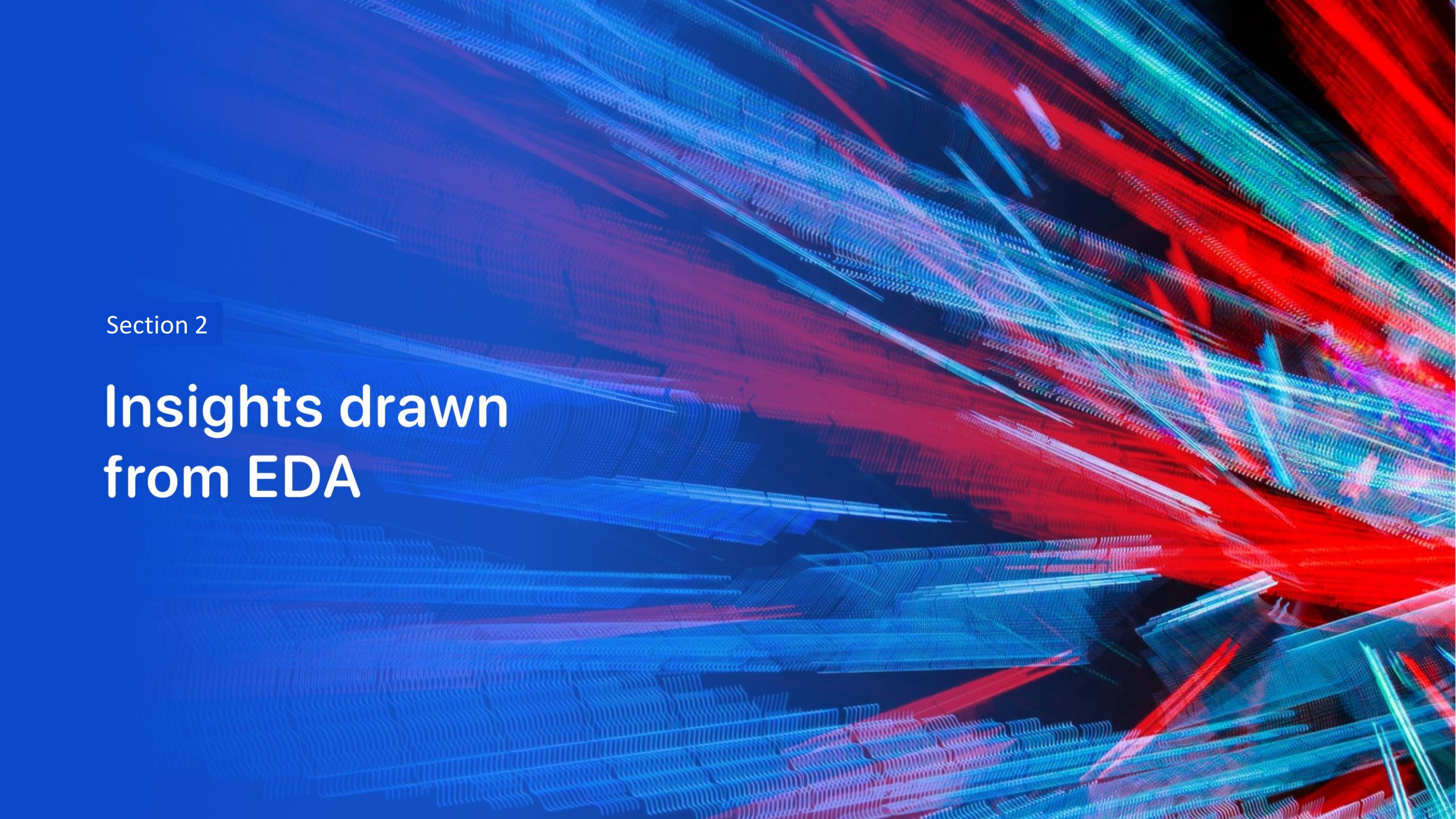
- Built an interactive dashboard with Plotly dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to python code is: https://github.com/yuanchung1987/Data-Science-Final-Project/blob/master/app.py

# Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Found the best performing classification model.
- Link to Jupyter notebook: https://github.com/yuanchung1987/Data-Science-Final-Project/blob/master/Machine%20Learning%20Prediction.ipynb

## Results

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

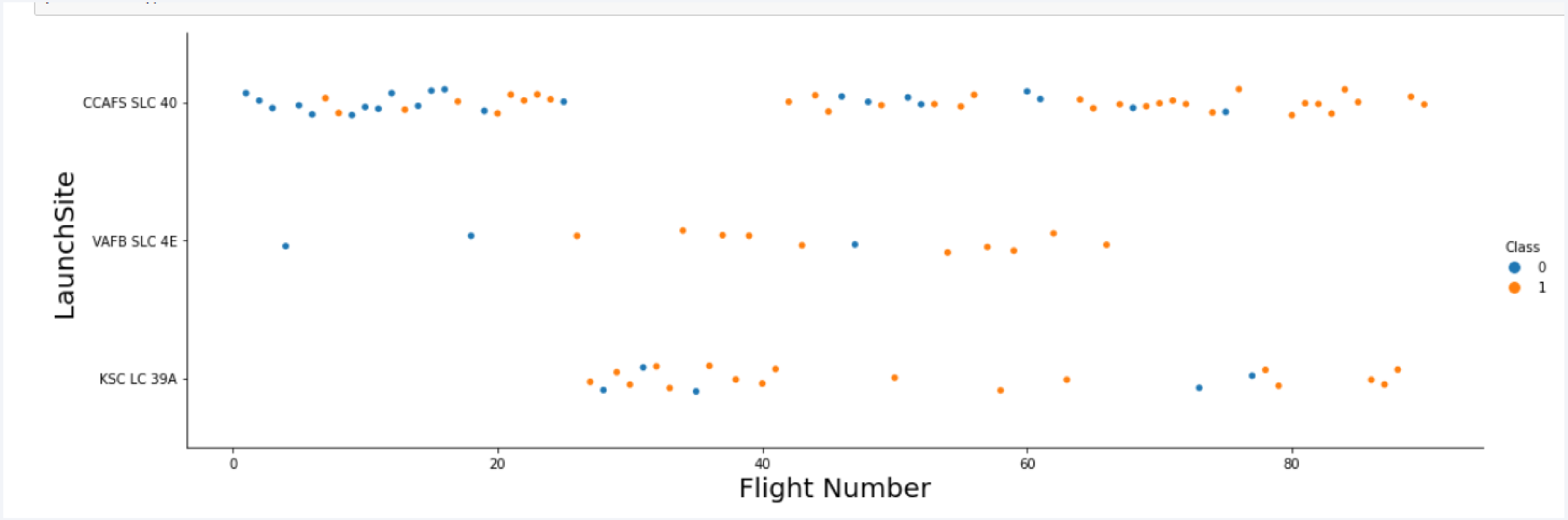


Section 2

## Insights drawn from EDA

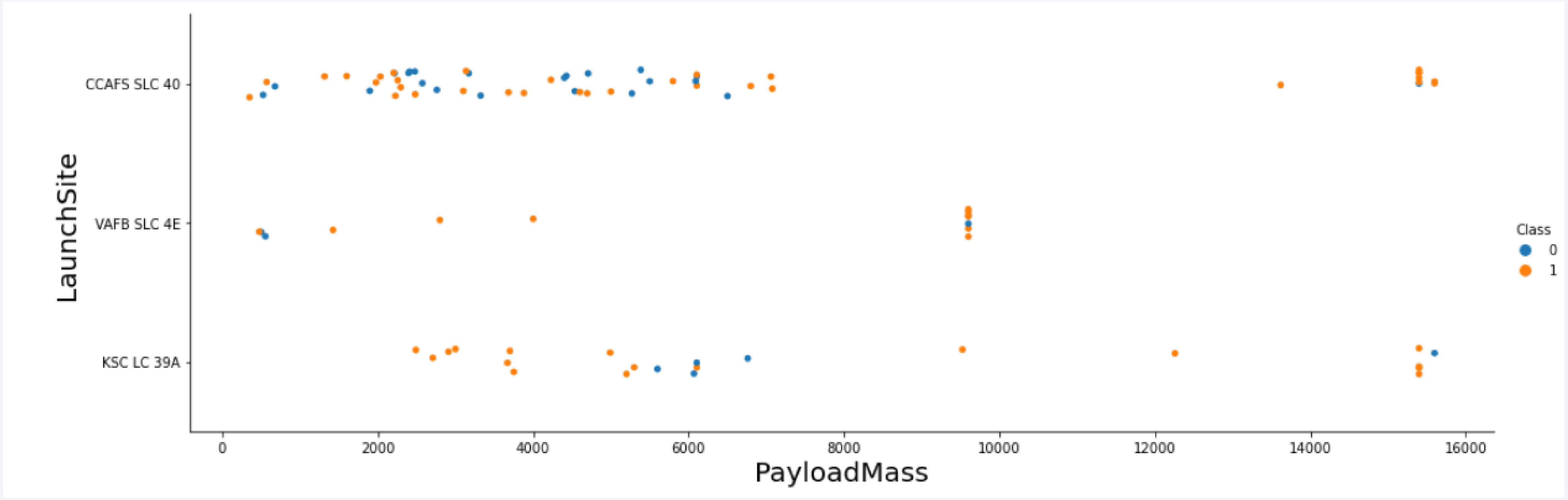
## Flight Number vs. Launch Site

- The more amount number of flight at launch site, the higher the success rate.



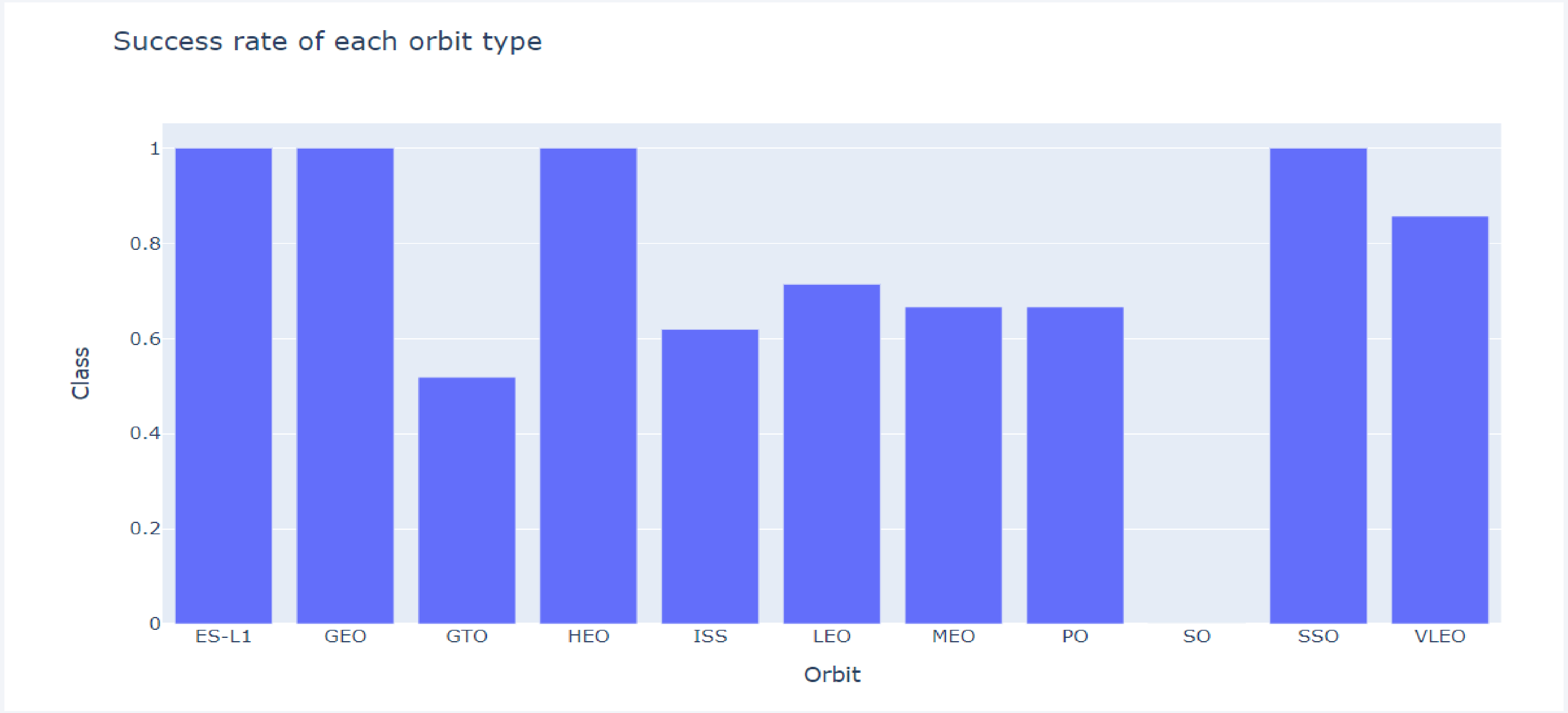
# Payload vs. Launch Site

- The higher payload mass the higher success rate for CCAFS SLC 40 site.



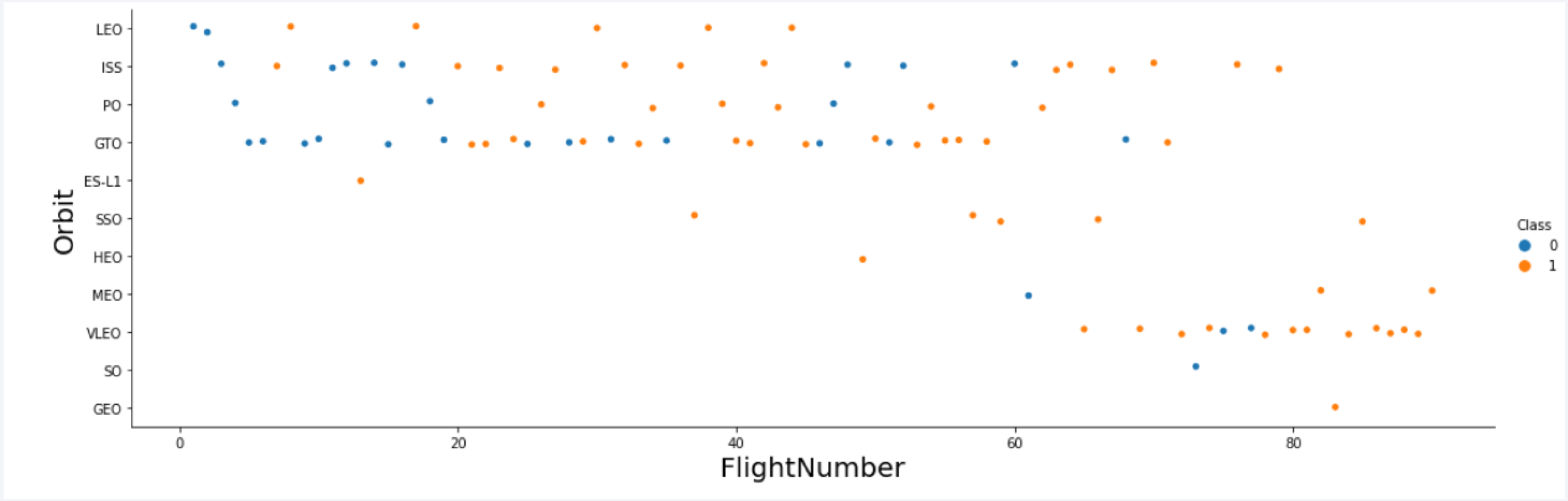
## Success Rate vs. Orbit Type

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



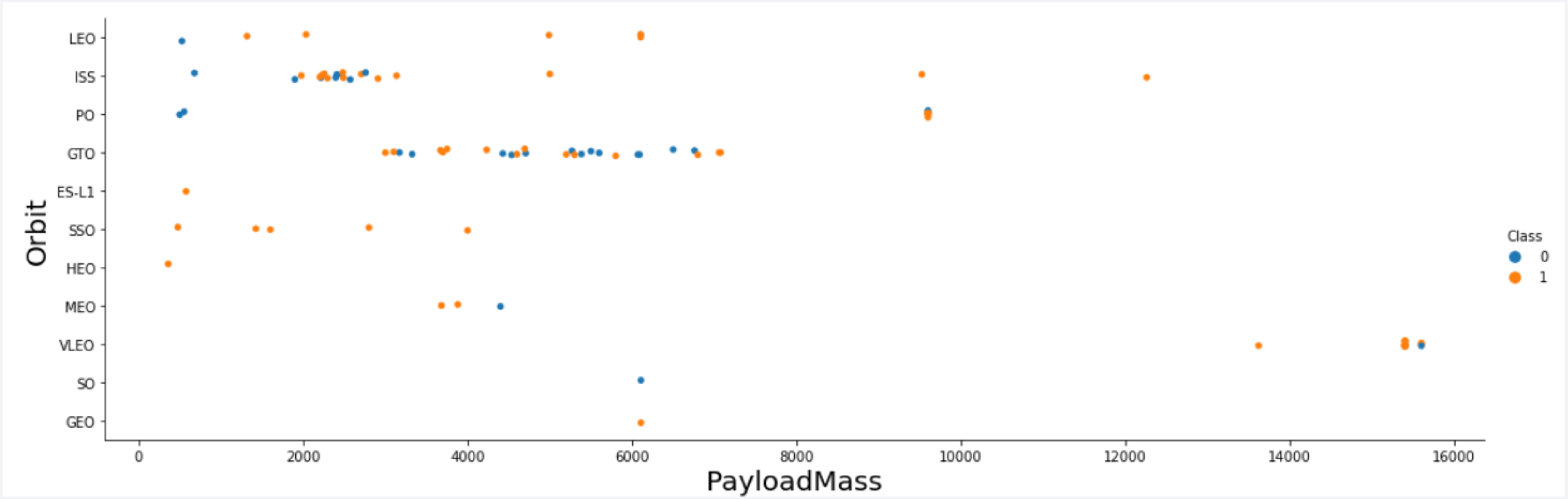
## Flight Number vs. Orbit Type

- Observed that in the LEO orbit, success rate is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



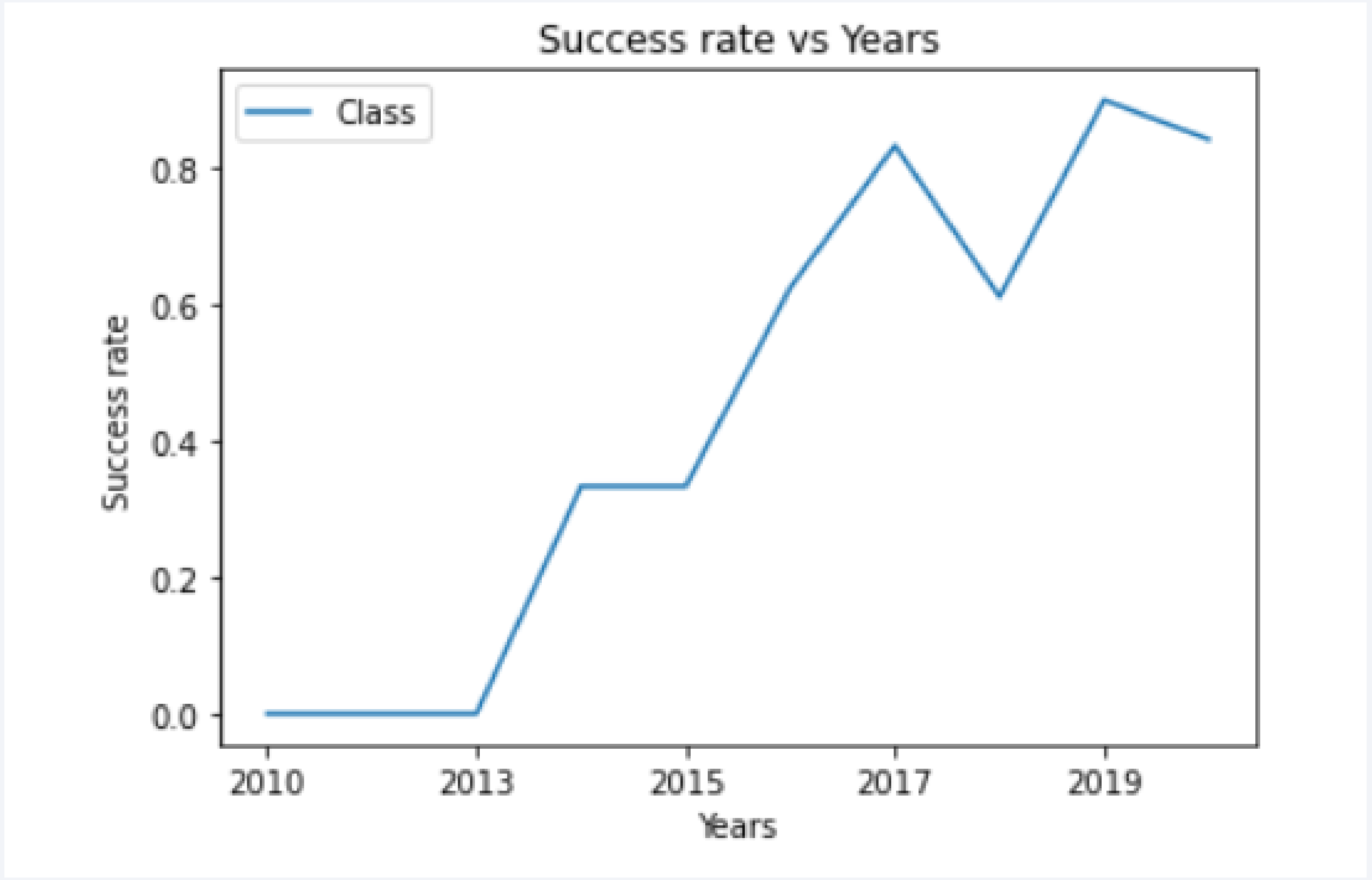
# Payload vs. Orbit Type

- There are heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



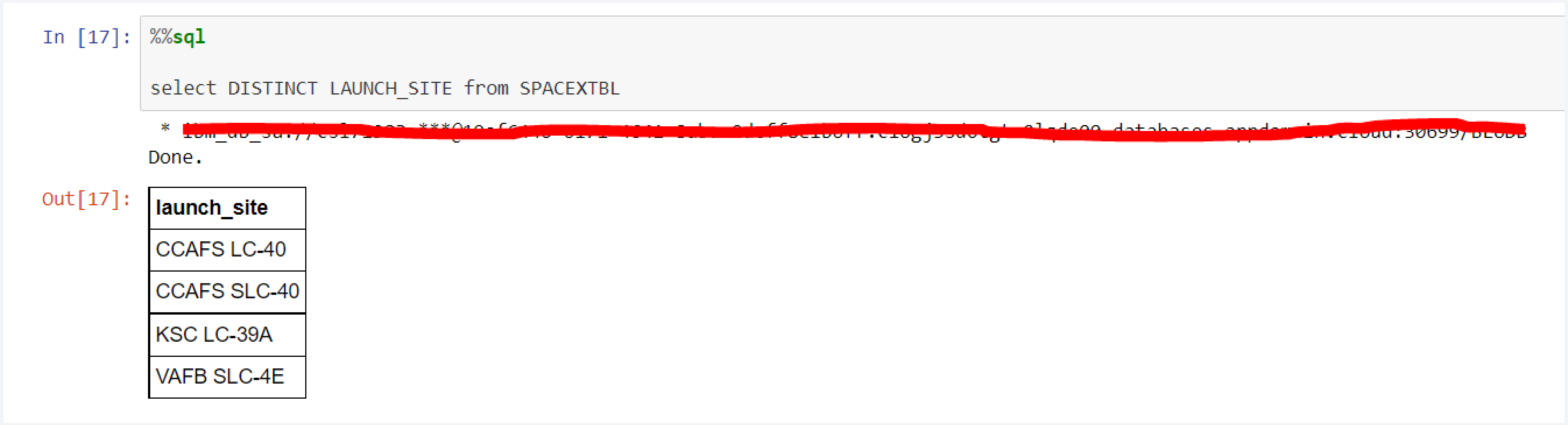
## Launch Success Yearly Trend

- We can observe that the success rate are increasing since 2013 till 2020.



## All Launch Site Names

- Used the key word DISTINCT to show only unique launch sites from the SpaceX data.



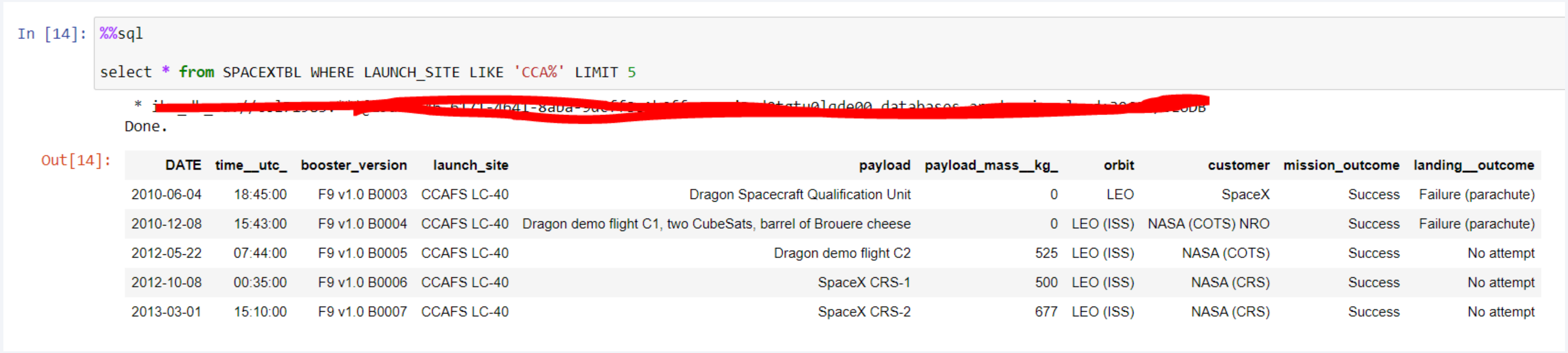
```
In [17]: %%sql

         select DISTINCT LAUNCH_SITE from SPACEXTBL

 * [redacted]
Done.
```

Out[17]:

| launch_site |
|---|
| CCAFS LC-40 |
| CCAFS SLC-40 |
| KSC LC-39A |
| VAFB SLC-4E |

# Launch Site Names Begin with 'CCA'

- Used the query below to display 5 records where launch sites begin with \`CCA\`



```
In [14]: %%sql

         select * from SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

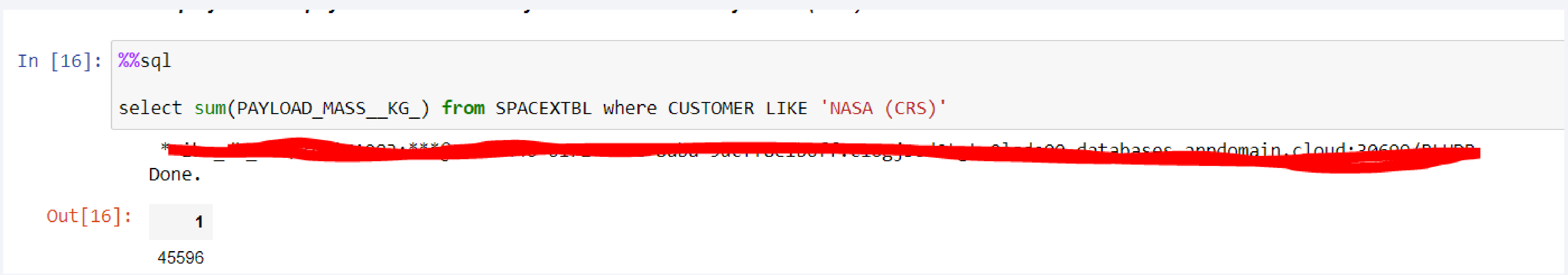
```
 * [redacted]
Done.
```

Out[14]:

| 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 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 00:35:00 | 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

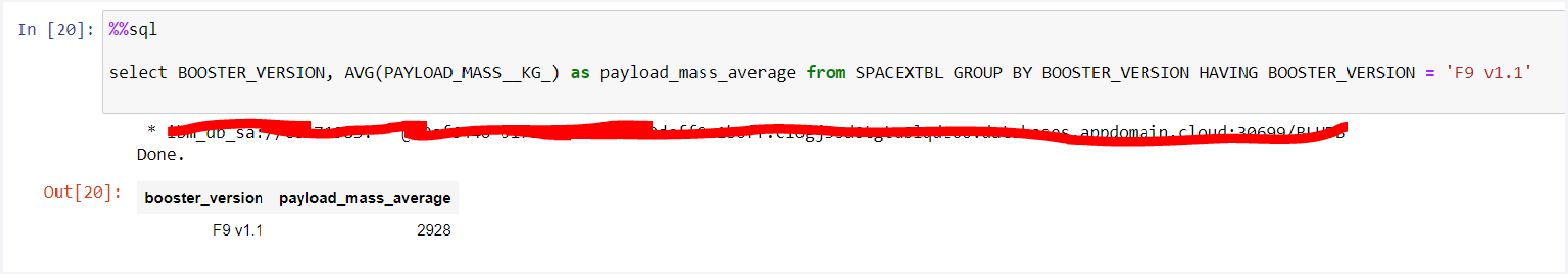
- Calculated the total payload carried by boosters from NASA as 45596 using the query below



```
In [16]: %%sql

         select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where CUSTOMER LIKE 'NASA (CRS)'

          [redacted]
         Done.

Out[16]:     1
         45596
```

## Average Payload Mass by F9 v1.1

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



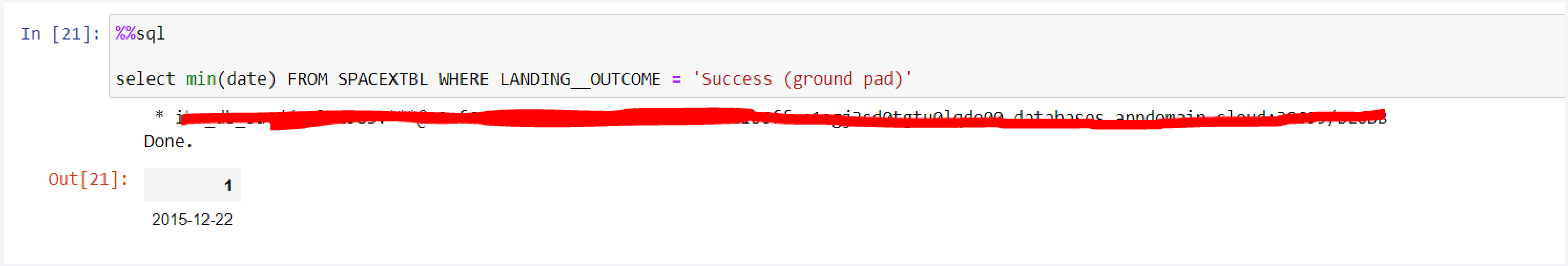
```
In [20]: %%sql

         select BOOSTER_VERSION, AVG(PAYLOAD_MASS__KG_) as payload_mass_average from SPACEXTBL GROUP BY BOOSTER_VERSION HAVING BOOSTER_VERSION = 'F9 v1.1'

          * [redacted]
         Done.
```

Out[20]:

| booster_version | payload_mass_average |
|---|---|
| F9 v1.1 | 2928 |

## First Successful Ground Landing Date

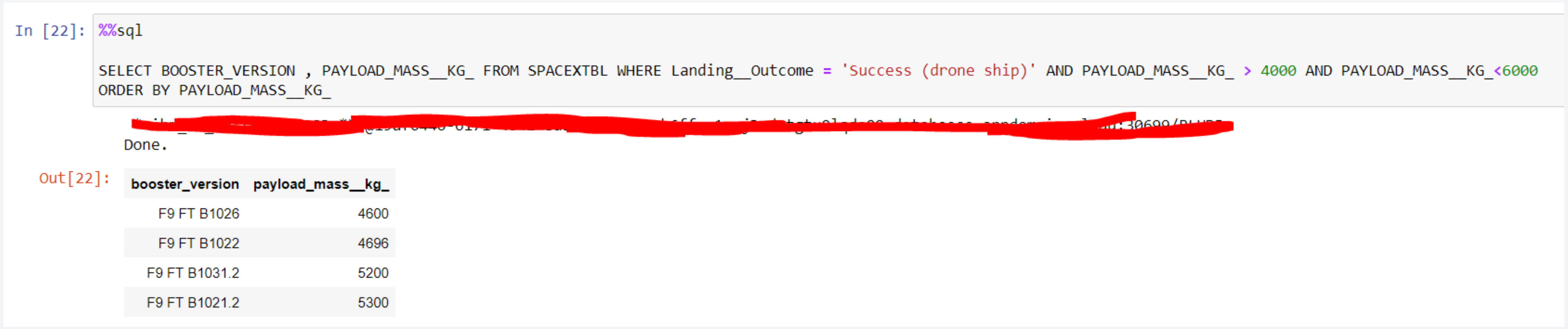
- Observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015



```
In [21]: %%sql

         select min(date) FROM SPACEXTBL WHERE LANDING__OUTCOME = 'Success (ground pad)'

          * [redacted]
         Done.

Out[21]:          1
           2015-12-22
```

## Successful Drone Ship Landing with Payload between 4000 and 6000

- 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

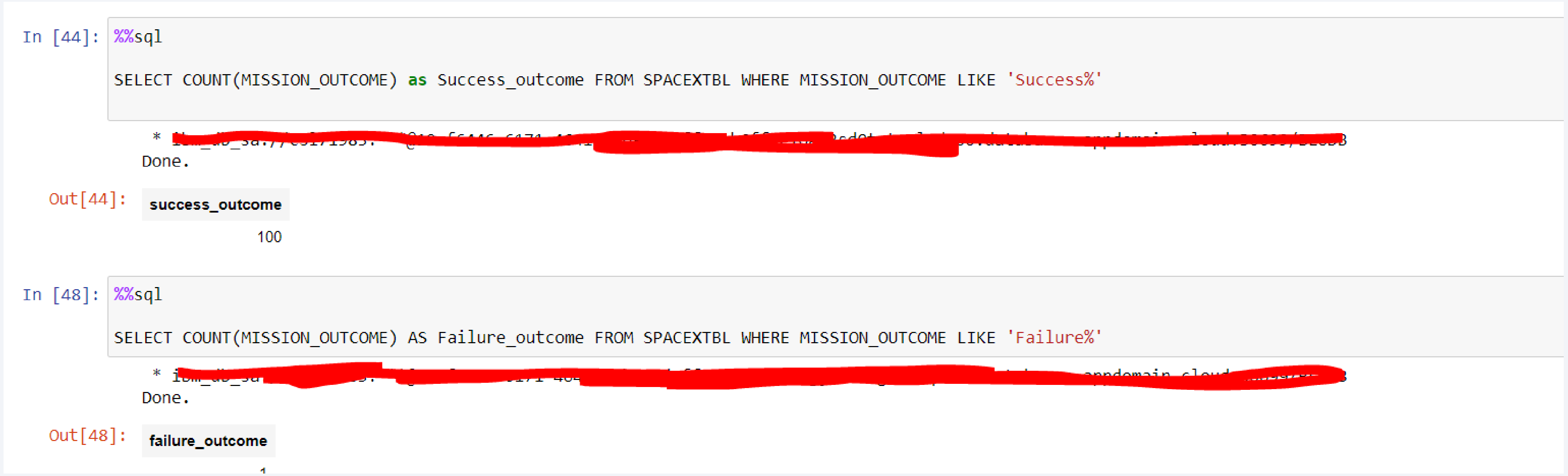


```
In [22]: %%sql

         SELECT BOOSTER_VERSION , PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE Landing__Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_<6000
         ORDER BY PAYLOAD_MASS__KG_

          [redacted]
          Done.

Out[22]:
```

| booster_version | payload_mass__kg_ |
|---|---|
| F9 FT B1026 | 4600 |
| F9 FT B1022 | 4696 |
| F9 FT B1031.2 | 5200 |
| F9 FT B1021.2 | 5300 |

## Total Number of Successful and Failure Mission Outcomes

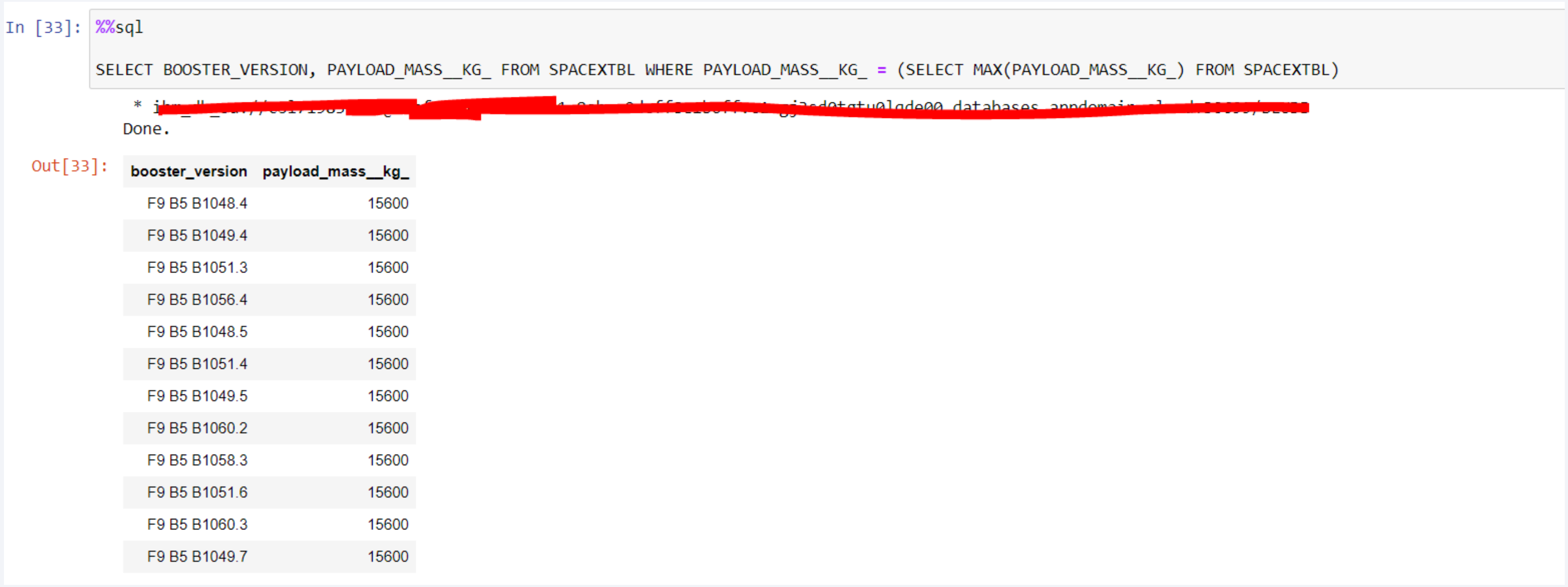
- Used code like '%' to filter for **WHERE** Mission_Outcome was a success or a failure.



```
In [44]: %%sql

         SELECT COUNT(MISSION_OUTCOME) as Success_outcome FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%'

          * [redacted]
         Done.

Out[44]:  success_outcome
                      100

In [48]: %%sql

         SELECT COUNT(MISSION_OUTCOME) AS Failure_outcome FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Failure%'

          * [redacted]
         Done.

Out[48]:  failure_outcome
                        1
```

## Boosters Carried Maximum Payload

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

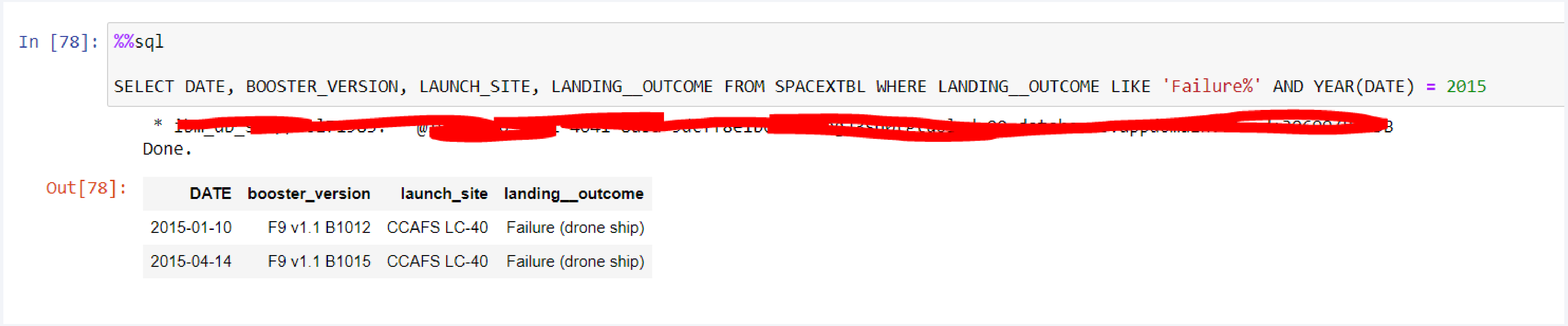


```
In [33]: %%sql

         SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)

          * [redacted]
         Done.

Out[33]:
```

| 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, YEAR** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015



```
In [78]: %%sql

         SELECT DATE, BOOSTER_VERSION, LAUNCH_SITE, LANDING__OUTCOME FROM SPACEXTBL WHERE LANDING__OUTCOME LIKE 'Failure%' AND YEAR(DATE) = 2015
```

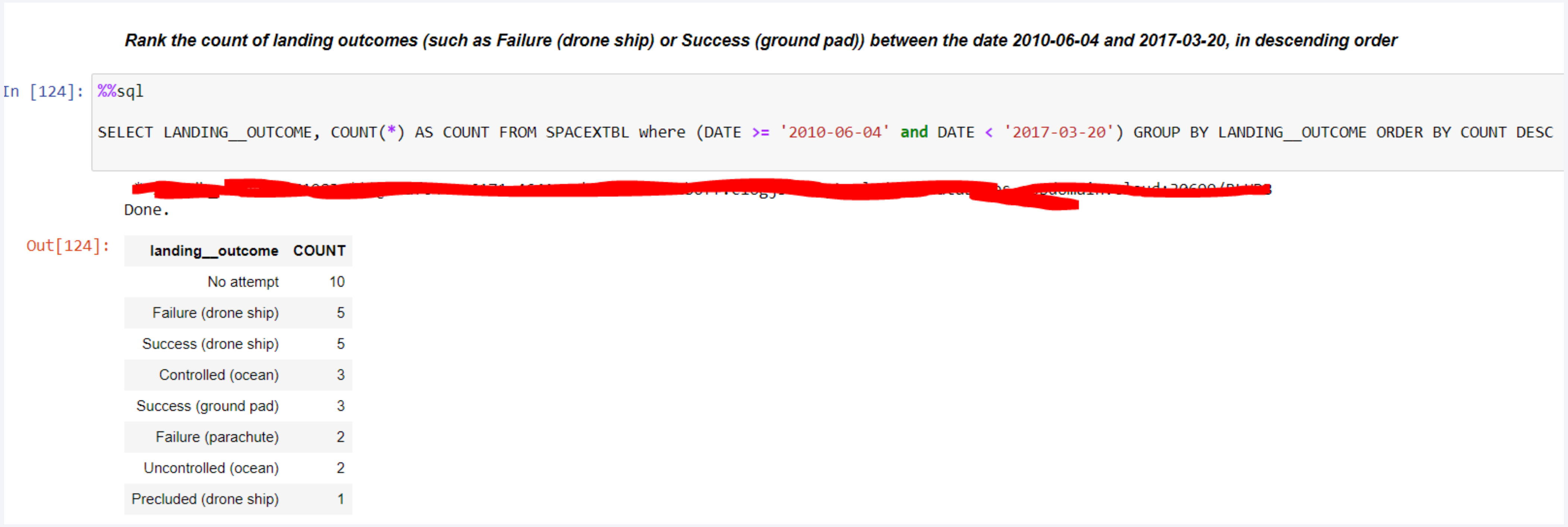
```
 * [redacted]
Done.
```

Out[78]:

| DATE | booster_version | launch_site | landing__outcome |
|---|---|---|---|
| 2015-01-10 | F9 v1.1 B1012 | CCAFS LC-40 | Failure (drone ship) |
| 2015-04-14 | F9 v1.1 B1015 | CCAFS LC-40 | Failure (drone ship) |

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

- 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.



***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 [124]: %%sql

SELECT LANDING__OUTCOME, COUNT(*) AS COUNT FROM SPACEXTBL where (DATE >= '2010-06-04' and DATE < '2017-03-20') GROUP BY LANDING__OUTCOME ORDER BY COUNT DESC
```

```
[redacted]
Done.
```

Out[124]:

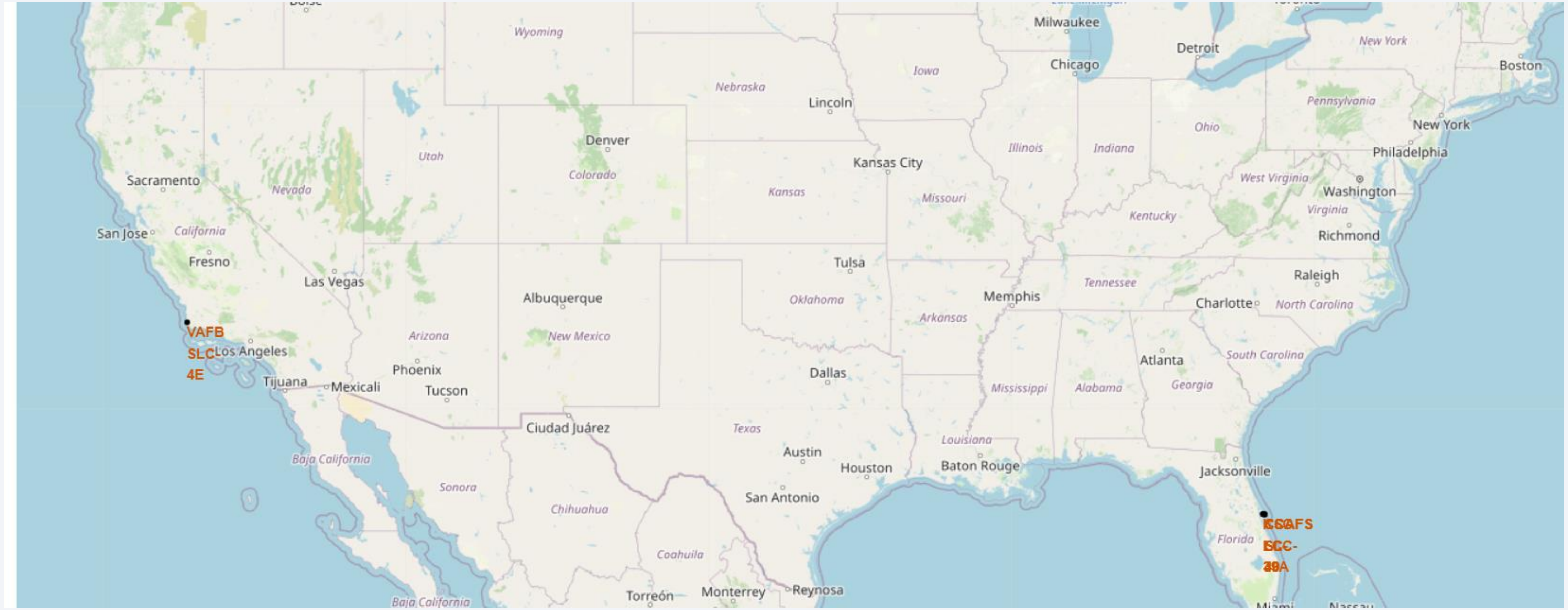
| landing__outcome | COUNT |
|---|---|
| No attempt | 10 |
| Failure (drone ship) | 5 |
| Success (drone ship) | 5 |
| Controlled (ocean) | 3 |
| Success (ground pad) | 3 |
| Failure (parachute) | 2 |
| Uncontrolled (ocean) | 2 |
| Precluded (drone ship) | 1 |

Section 3

## Launch Sites Proximities Analysis

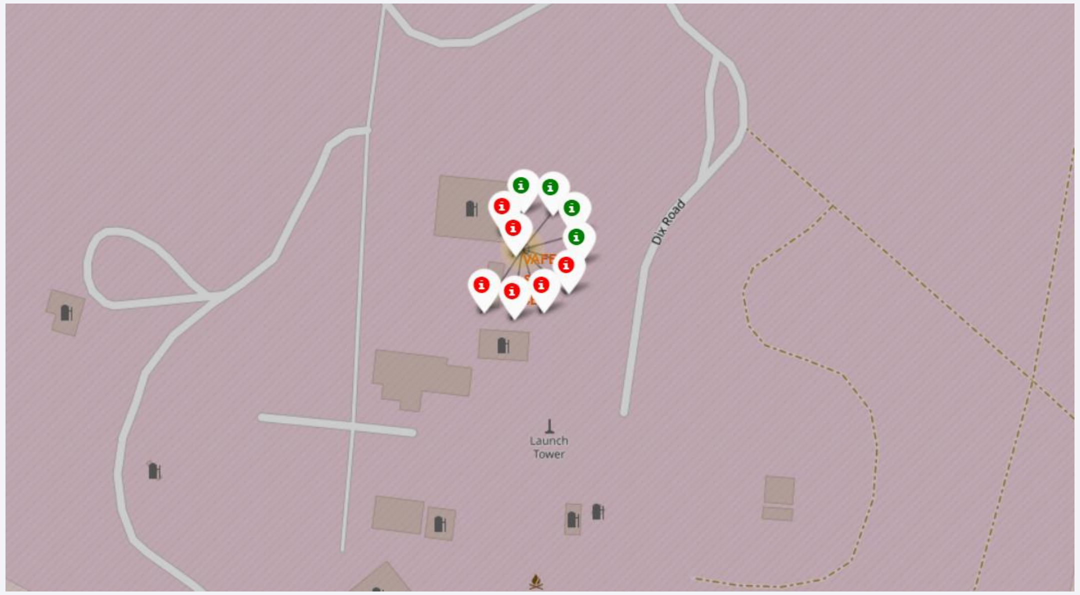
## All launch sites global map markers

- We can see the Falcon 9 launch sites are in United Stated coastline. Florida and California

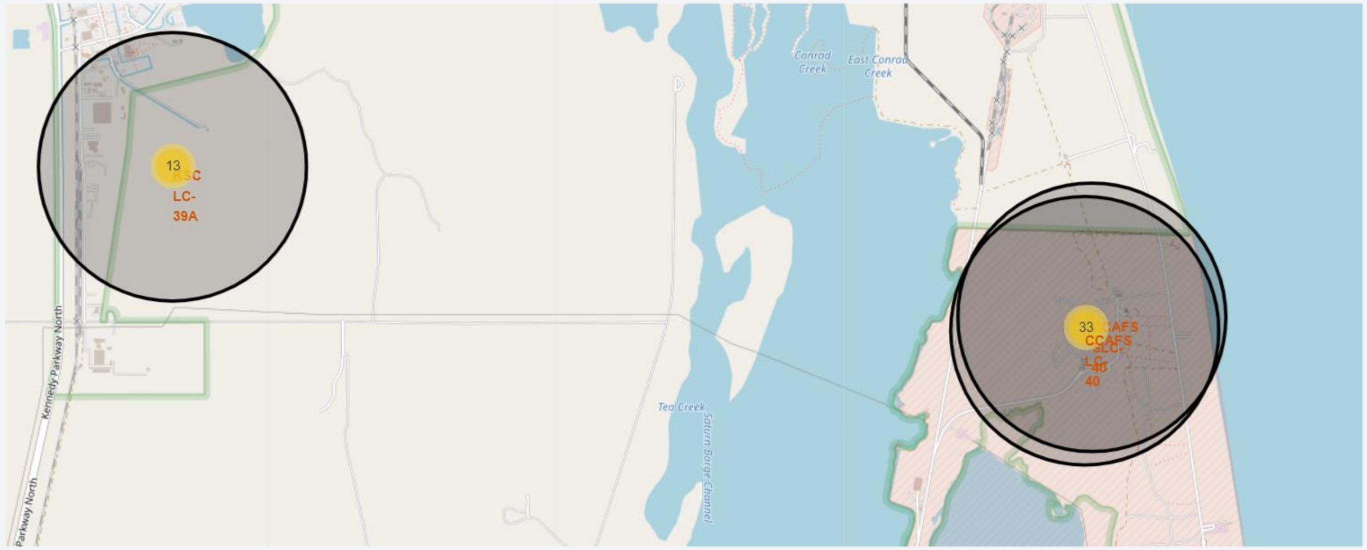


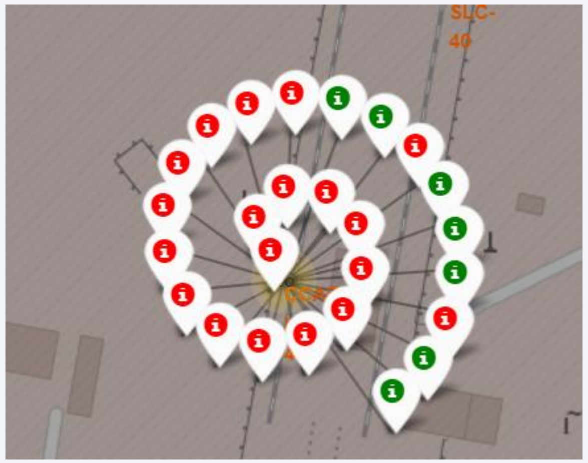
## Markers showing launch sites with color labels

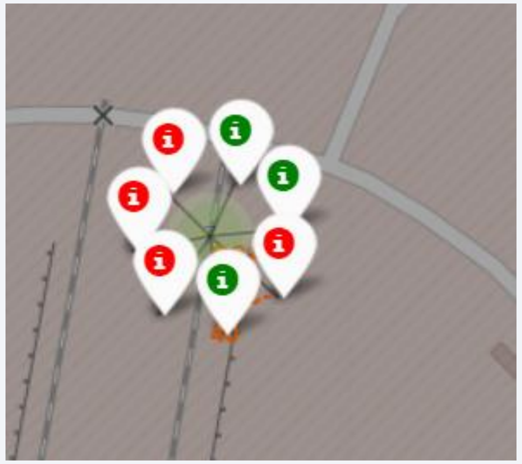
- California site.



- Florida site.





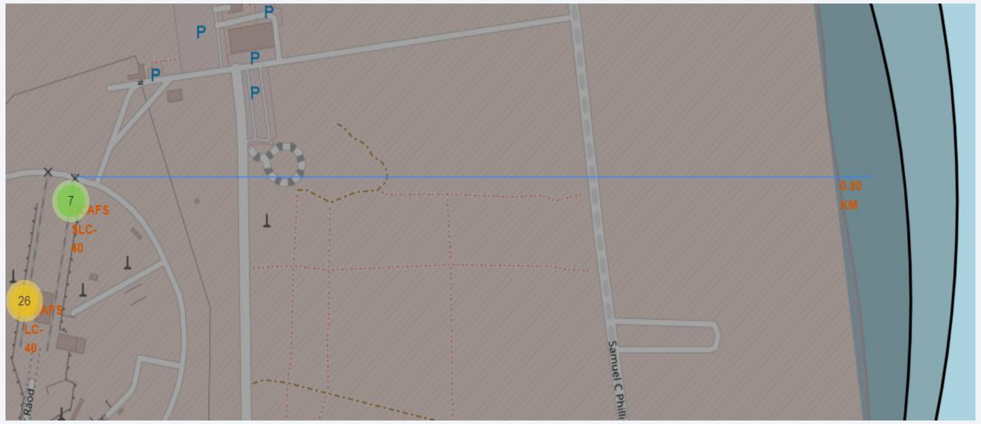


Green marker shows successful launch.

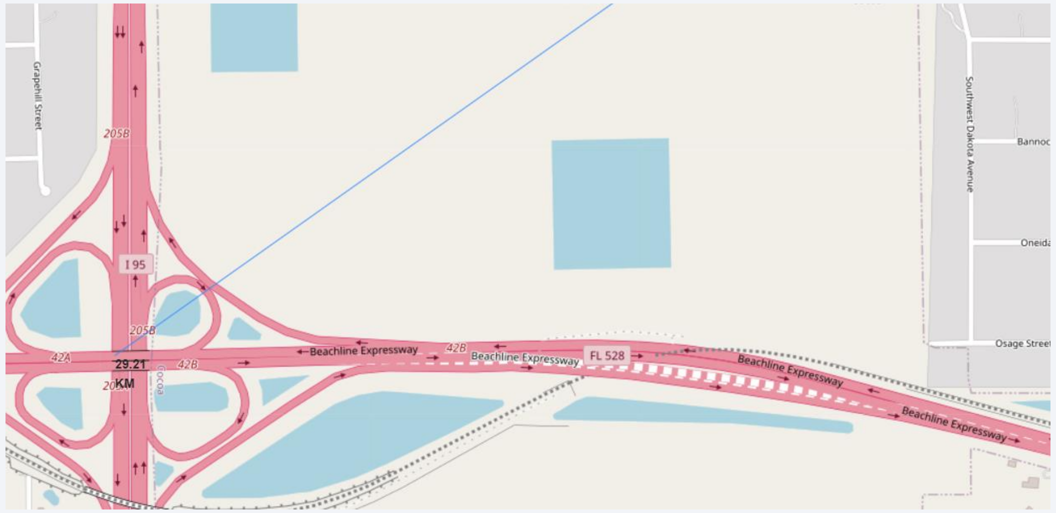
Red marker shows failure launch.

## Launch Site distance to landmarks

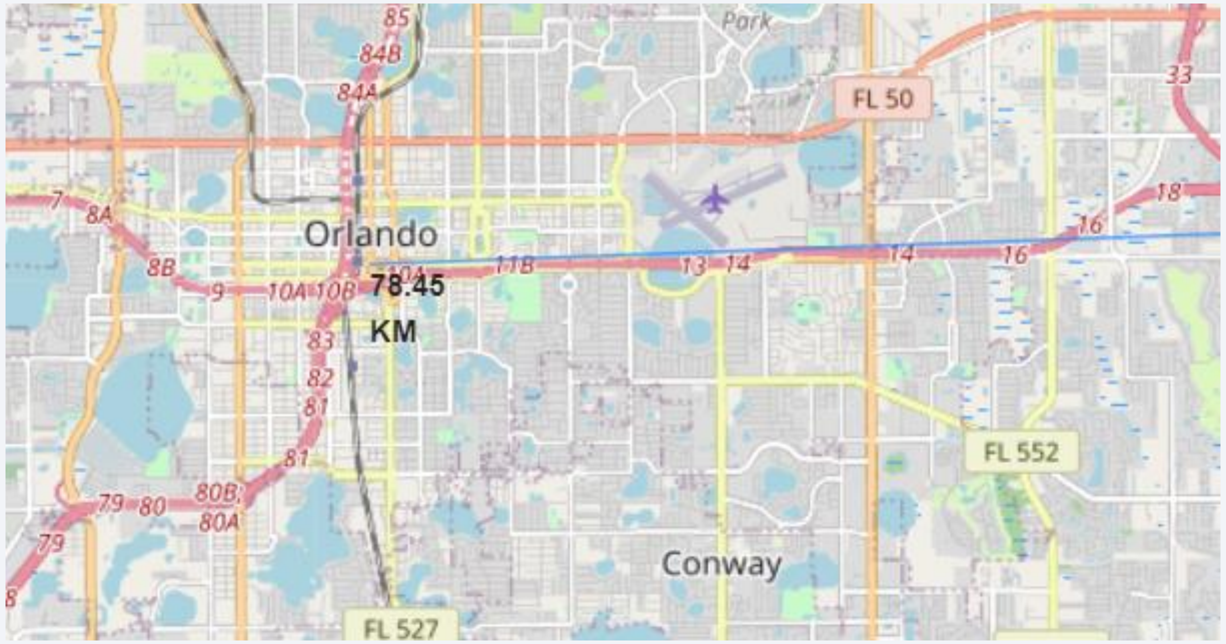
- Distance to coastline.

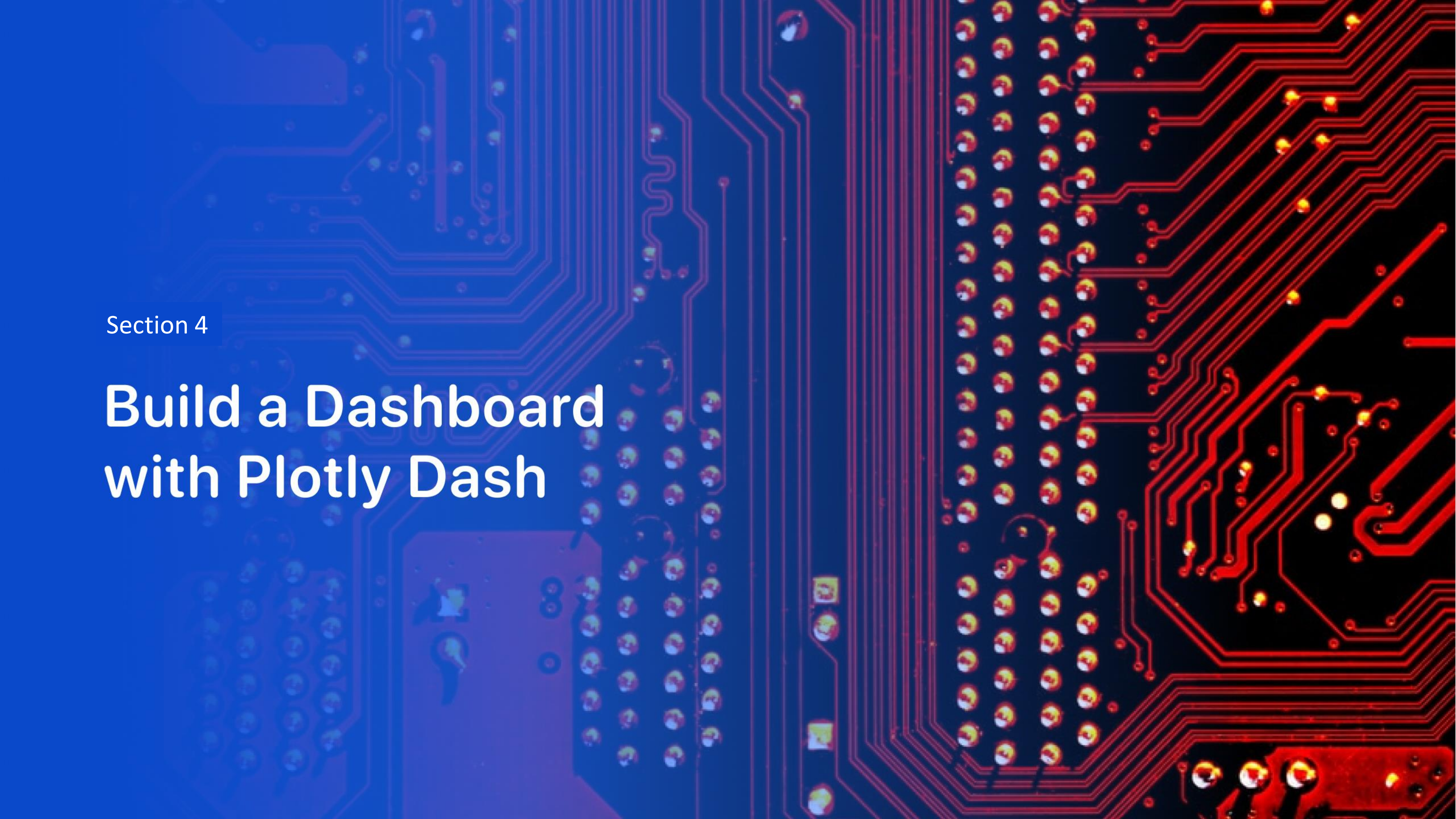


- Distance to highway.



- Distance to Orlando.

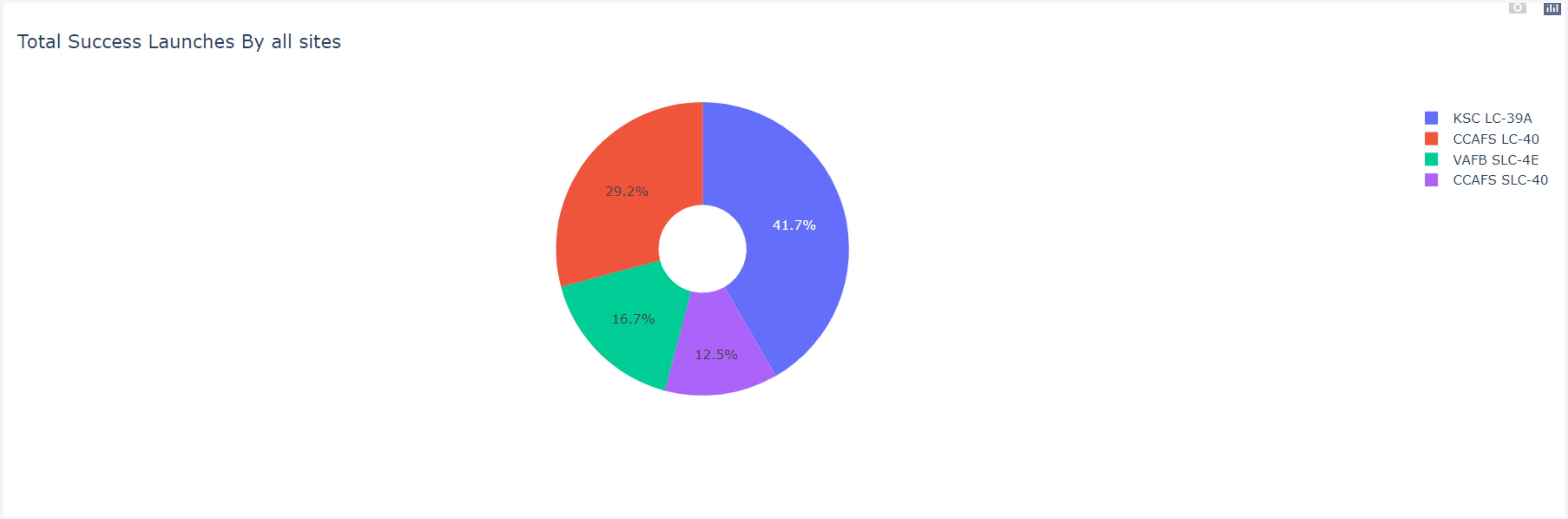




Section 4

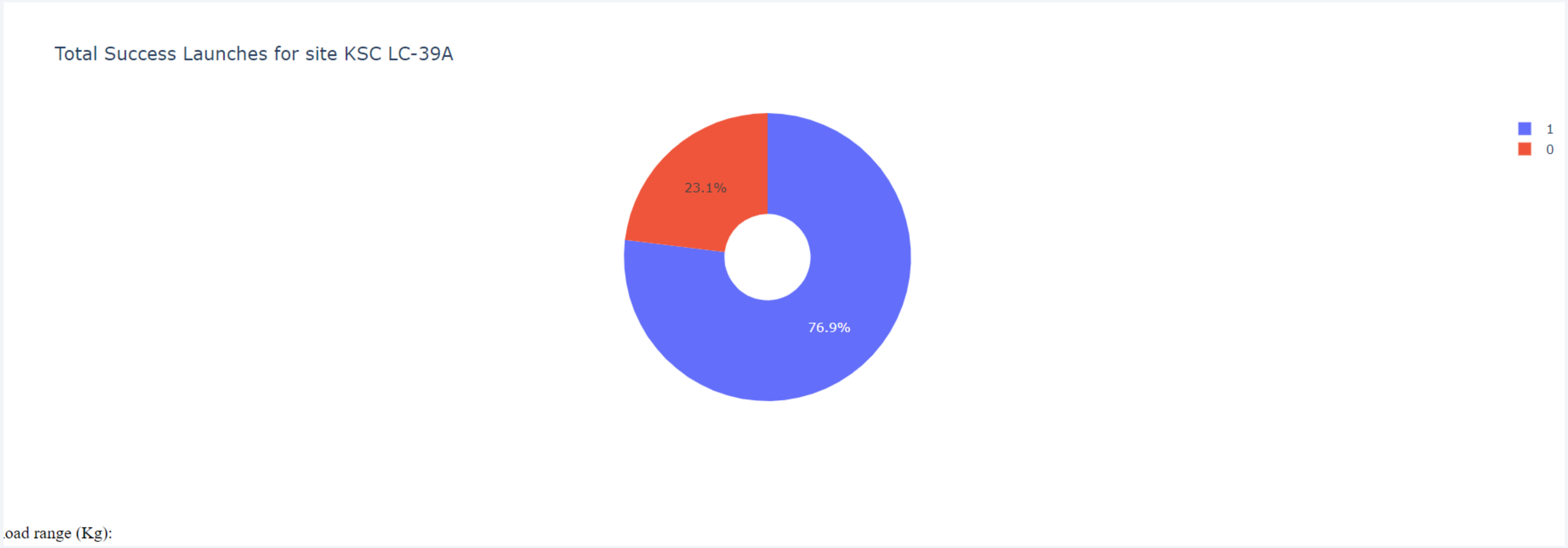
## Build a Dashboard with Plotly Dash

## Pie chart showing the success percentage achieved by each launch site



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

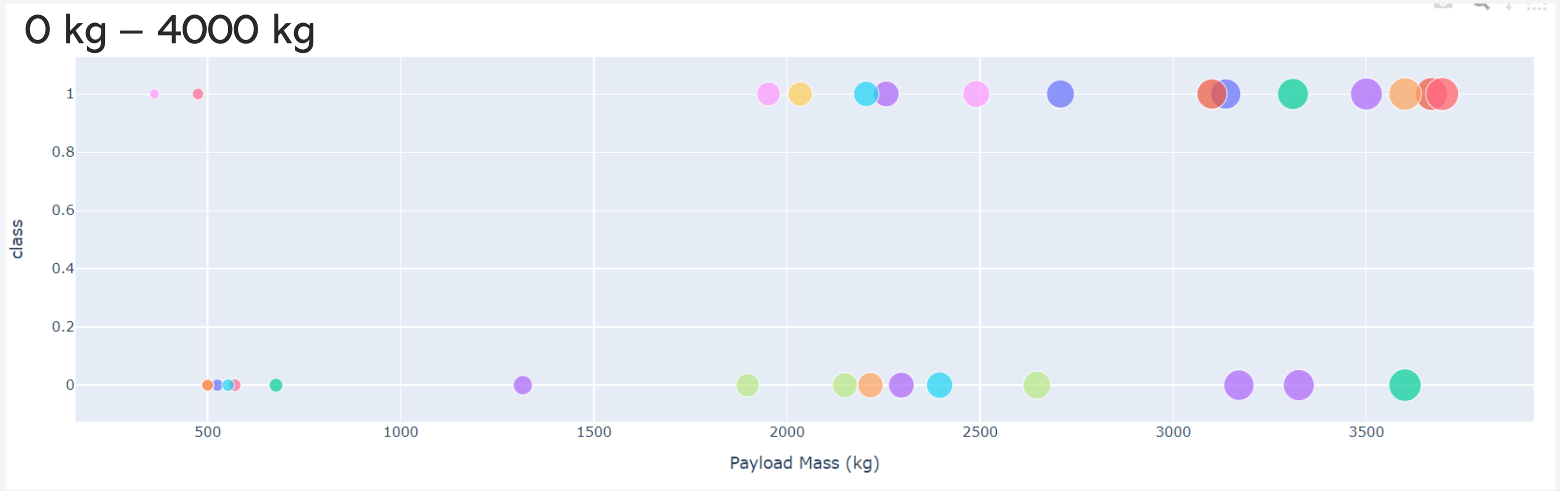
- The most success rate is on KSC LC-39A site.



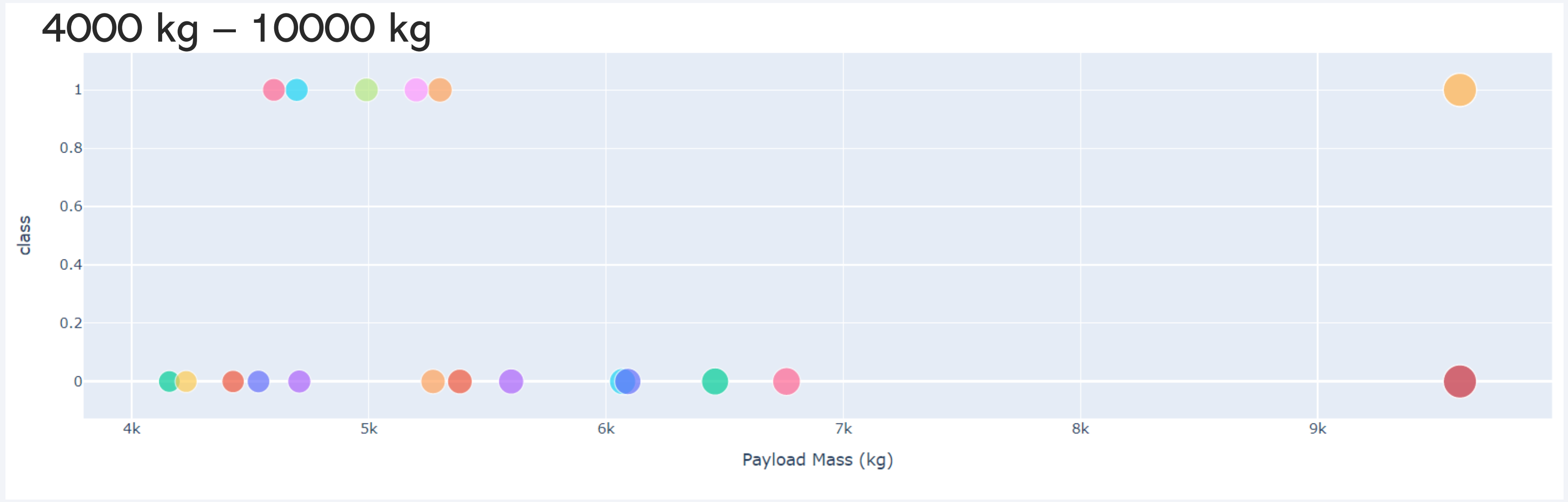
## Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

- There seems to be lower weight has higher success rate.

0 kg – 4000 kg



4000 kg – 10000 kg

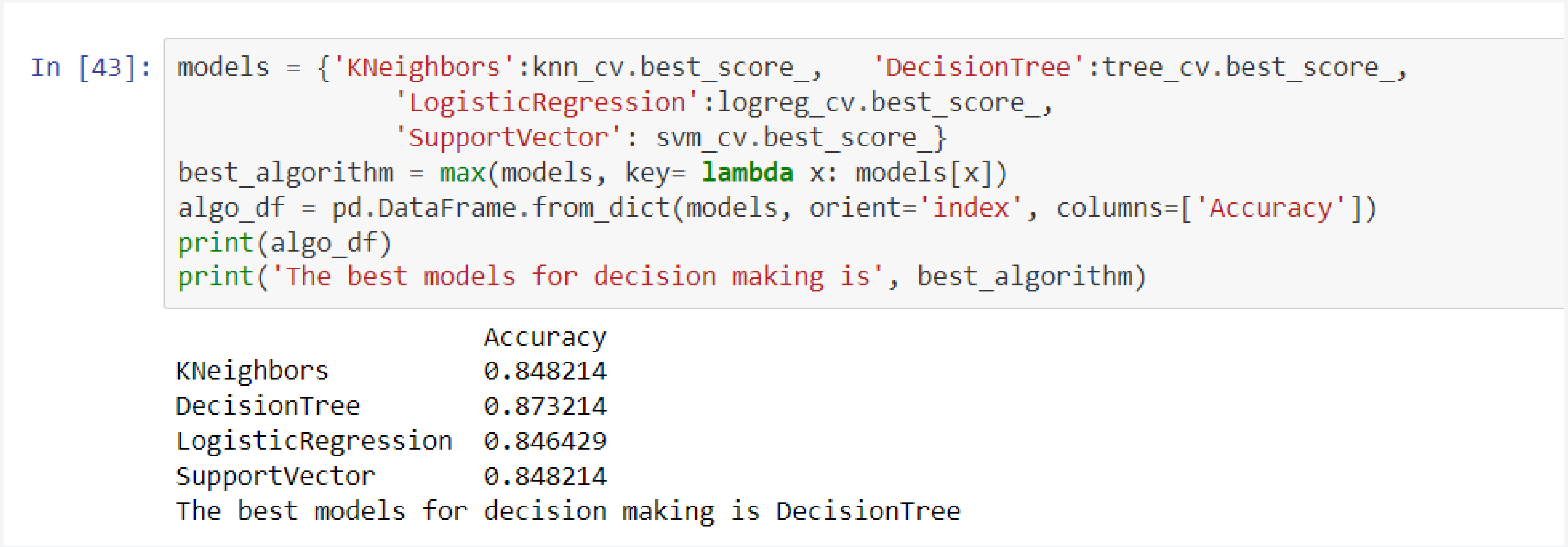


Section 5

## Predictive Analysis (Classification)

## Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy

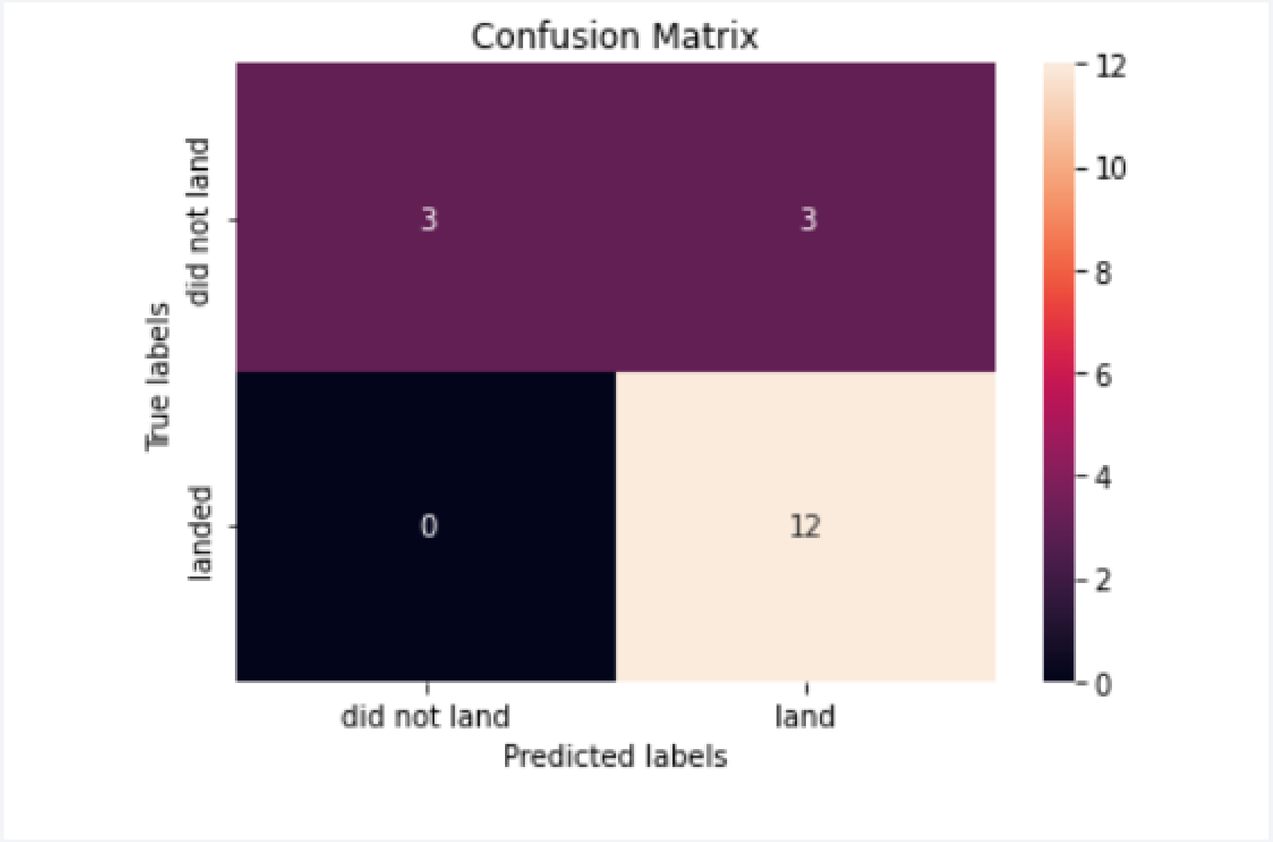


```
In [43]: models = {'KNeighbors':knn_cv.best_score_,   'DecisionTree':tree_cv.best_score_,
                       'LogisticRegression':logreg_cv.best_score_,
                       'SupportVector': svm_cv.best_score_}
         best_algorithm = max(models, key= lambda x: models[x])
         algo_df = pd.DataFrame.from_dict(models, orient='index', columns=['Accuracy'])
         print(algo_df)
         print('The best models for decision making is', best_algorithm)
```

```
                    Accuracy
KNeighbors          0.848214
DecisionTree        0.873214
LogisticRegression  0.846429
SupportVector       0.848214
The best models for decision making is DecisionTree
```

## Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positive, i.e., the model predicts rocket land and it actually not landed.



## Conclusions

- From the analysis 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!