IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Thabang Teddy
03/05/2023

Outline

Executive Summary

- Summary of methodologies
- Data Collection with an API
- Data Collection with Web Scrapping
- Data Wrangling
- Exploratory Data Analysis (EDA) with SQL
- Exploratory Data Analysis (EDA) with Data Dashboards
- Interactive Visual Analytics with Folium
- Machine Learning Prediction for SpaceX
- Summary of all results
- Exploratory Data Analysis results
- Interactive analytics with screenshots
- Predictive results for the best Model of SpaceX

Introduction

SpaceX, one of the most revolutionary and most accomplished commercial in the space age. Some accomplishments of which may include sending space crafts to space, manned missions to space and star-links for the internet. One of the reasons for its success my include lower costs for rocket launches where the Falcon 9 sits at a $62 million cost, relatively lower than its competitors. The reason for this cost is by the reusable rockets that can be sent for more than just one mission into space. The aim of this project is to create the machine learning pipeline to predict the landing outcome of the first stage in the future. This project is crucial in identifying the right price to bid against SpaceX for a rocket launch.

The problems included:

- Identifying all factors that influence the landing outcome.
- The relationship between each variables and how it is affecting the outcome.
- The best condition needed to increase the probability of successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collection was done using a SpaceX REST API and web scrapping from Wikipedia
- Perform data wrangling
  - To preprocess the data, I used one-hot encoding
- 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

Data Collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes.

As mentioned already, the dataset was collected by REST API and Web Scrapping from Wikipedia. Using the REST API, its started by using the GET request. Then, we decoded the response content as JSON and turn it into a pandas dataframe using json_normalize().

We then cleaned the data, checked for missing values and fill with the mean of the column data.

Then for web scrapping, we will used the BeautifulSoup to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for further analysis

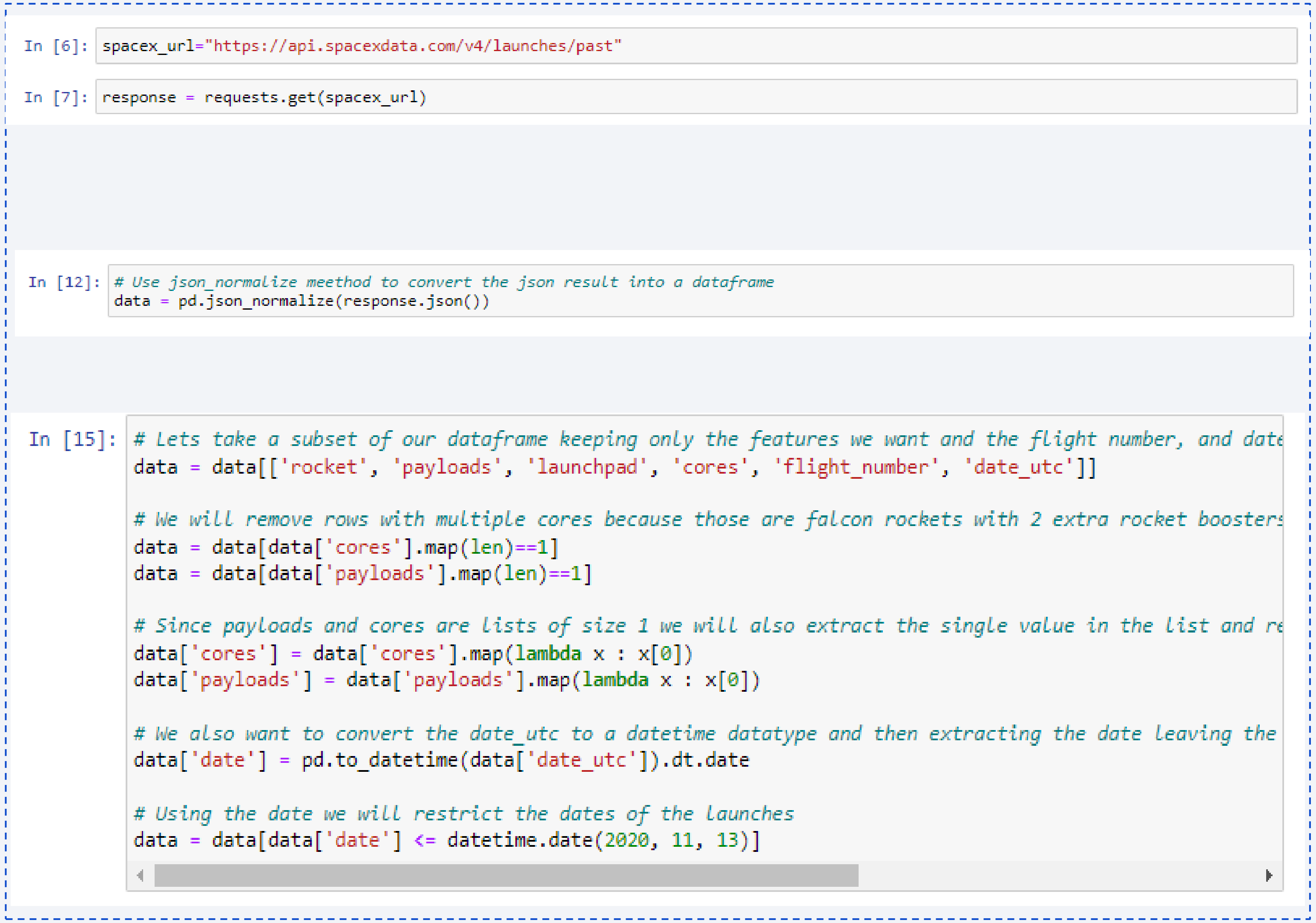
Data Collection – SpaceX API

Using the ‘get’ request for the rocket launch data using REST API

Used the json.normalize() method to convert the json result to a dataframe

Data Cleaning and filling in the missing values using the mean

[GitHub Link: Data Collection API]



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

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

```
In [12]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

```
In [15]: # Lets take a subset of our dataframe keeping only the features we want and the flight number, and date
         data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

         # We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters
         data = data[data['cores'].map(len)==1]
         data = data[data['payloads'].map(len)==1]

         # Since payloads and cores are lists of size 1 we will also extract the single value in the list and re
         data['cores'] = data['cores'].map(lambda x : x[0])
         data['payloads'] = data['payloads'].map(lambda x : x[0])

         # We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the
         data['date'] = pd.to_datetime(data['date_utc']).dt.date

         # Using the date we will restrict the dates of the launches
         data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

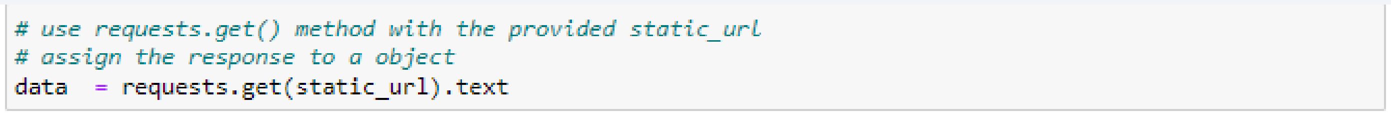
Data Collection - Scraping

Request the Falcon 9 Launch Wiki Page from url

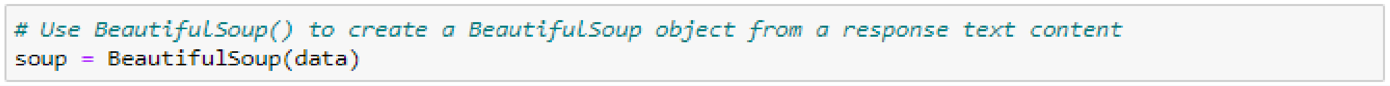
Create BeautifulSoup from the HTML response

Extracting all the columns names from the HTML header tag

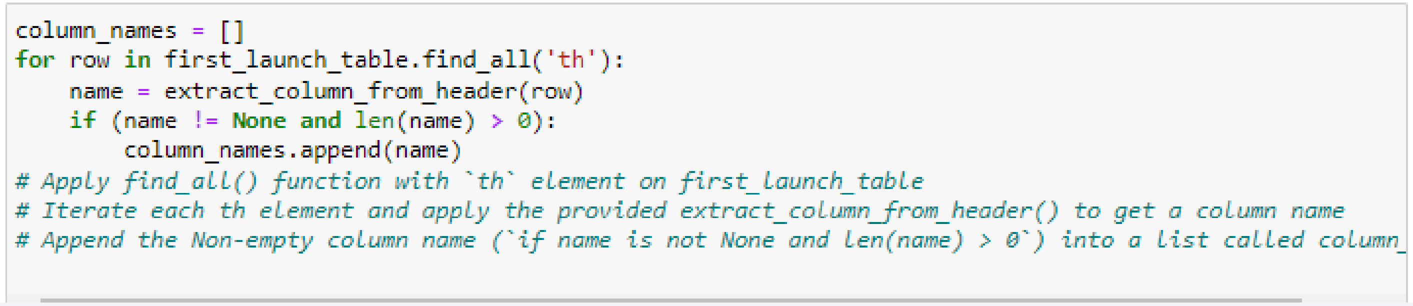
[Github link-Webscrapping]



```
# use requests.get() method with the provided static_url
# assign the response to a object
data  = requests.get(static_url).text
```



```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data)
```



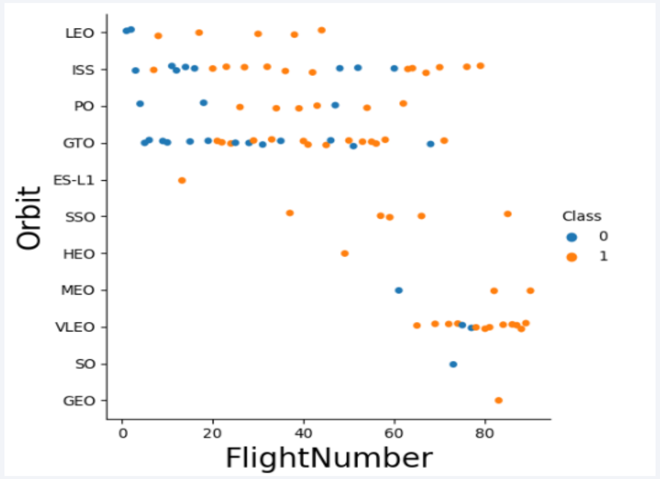
```
column_names = []
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided 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_
```

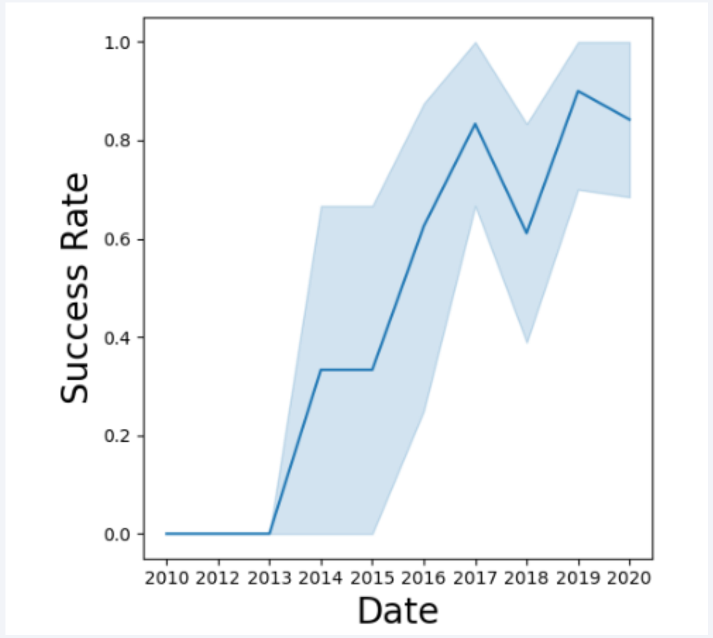
Data Wrangling

35768 km
LEO
10000 km
MEO
1000 km
HEO
GEO

- Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).
- We first calculated the number of launches on each site, then calculate the number and occurrence of mission outcome per orbit type. We then create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Lastly, we will export the result to a CSV fie.
- Github link- DataWrangling

EDA with Data Visualization





We first started by using scatter graph to find the relationship between the attributes such as Flight Number and Orbit Type.

Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs. It's very easy to see which factors affecting the most to the success of the landing outcomes

Then we secondly used a line chart to find the success rate each year for each launch. As we can see the chart represents a steep upward slope that indicates that with the increasing years the success rate for each launch got better aswell.

[Githublink Data Visualization]

EDA with SQL

Using SQL, we had performed many queries to get better understanding of the dataset such as:

- Displaying the names of the launch sites.
- Displaying 5 records where launch sites begin with the string ‘CCA’.
- Displaying the total payload mass carried by booster launched by NASA (CRS).
- Displaying the average payload mass carried by booster version F9 v1.1.
- Listing the date when the first successful landing outcome in ground pad was achieved.
- Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
- Listing the total number of successful and failure mission outcomes.
- Listing the names of the booster versions which have carried the maximum payload mass.

GitHub link-EDA with SQL

Build an Interactive Map with Folium

To visualize the launch data into an interactive map. We took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.

We then assigned the dataframe launch_outcomes(failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().

We then used the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:

- How close the launch sites with railways, highways and coastlines?
- How close the launch sites with nearby cities?
- Github link for DataVisualization with folium

Build a Dashboard with Plotly Dash

We built an interactive dashboard with Plotly Dash which would allow the user to play around with the data as they need.

We plotted pie charts showing the total launches by a certain sites.

We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

Generally speaking we plotted these graphs to showcase the data availability of using dash

- GitHub URL

Predictive Analysis (Classification)

Building the Model

- Load in the dataset using Pandas and Numpy
- Transform the data and split it into train and test datasets
- Then Set the parameters and algorithms to GridSearchCV and fit the dataset

Evaluating the Model

- Check the accuracy of each model
- Fine tune the hyperparameters of each algorithm
- Finally plot the confusion matrix

Improving the Model

- Use Feature Engineering and Algorithm Tuning

Finding the best Model

- The model with the best and highest accuracy score will become the best performing model

GitHub URL

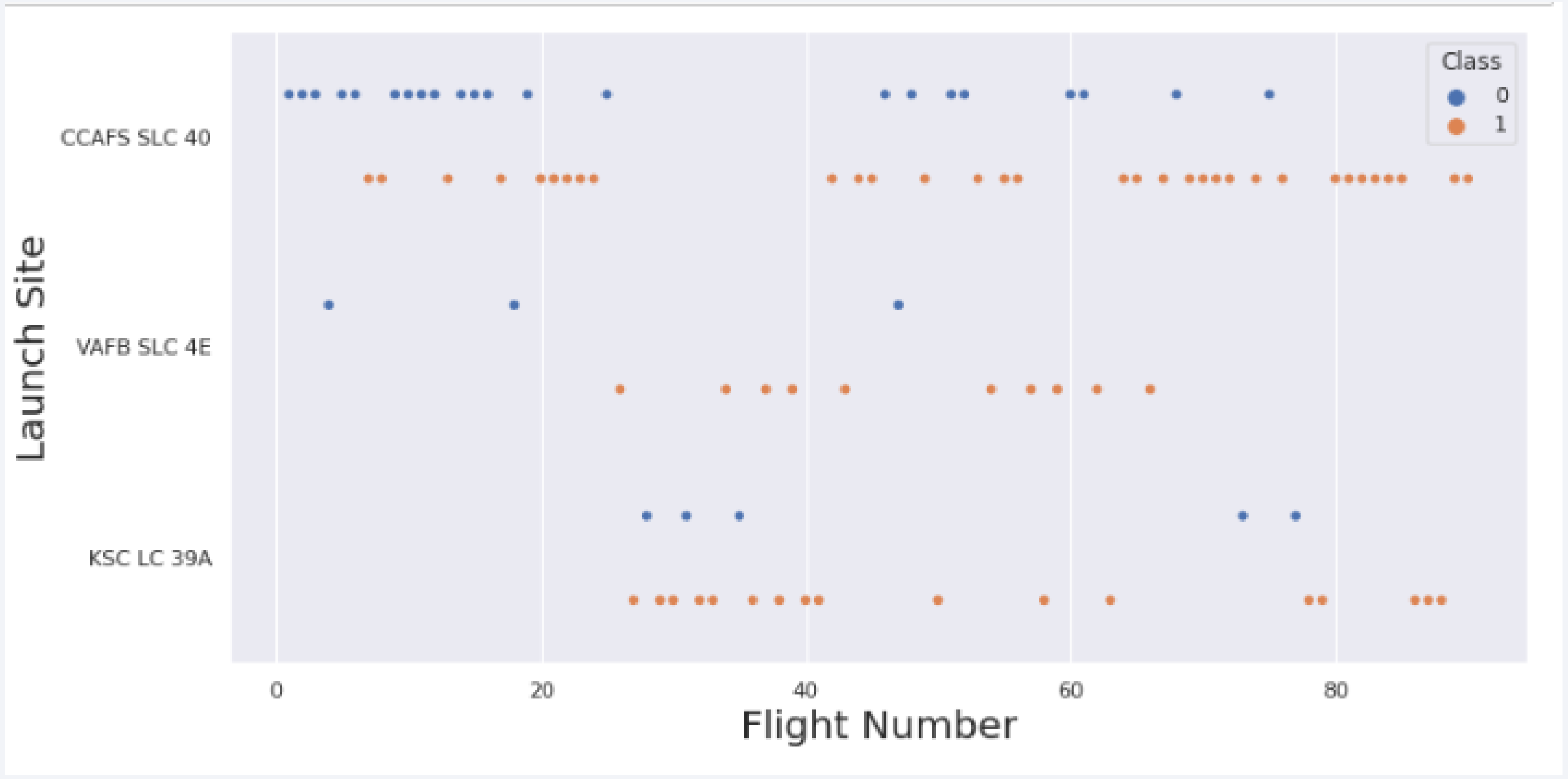
Results

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

Section 2

Insights drawn from EDA

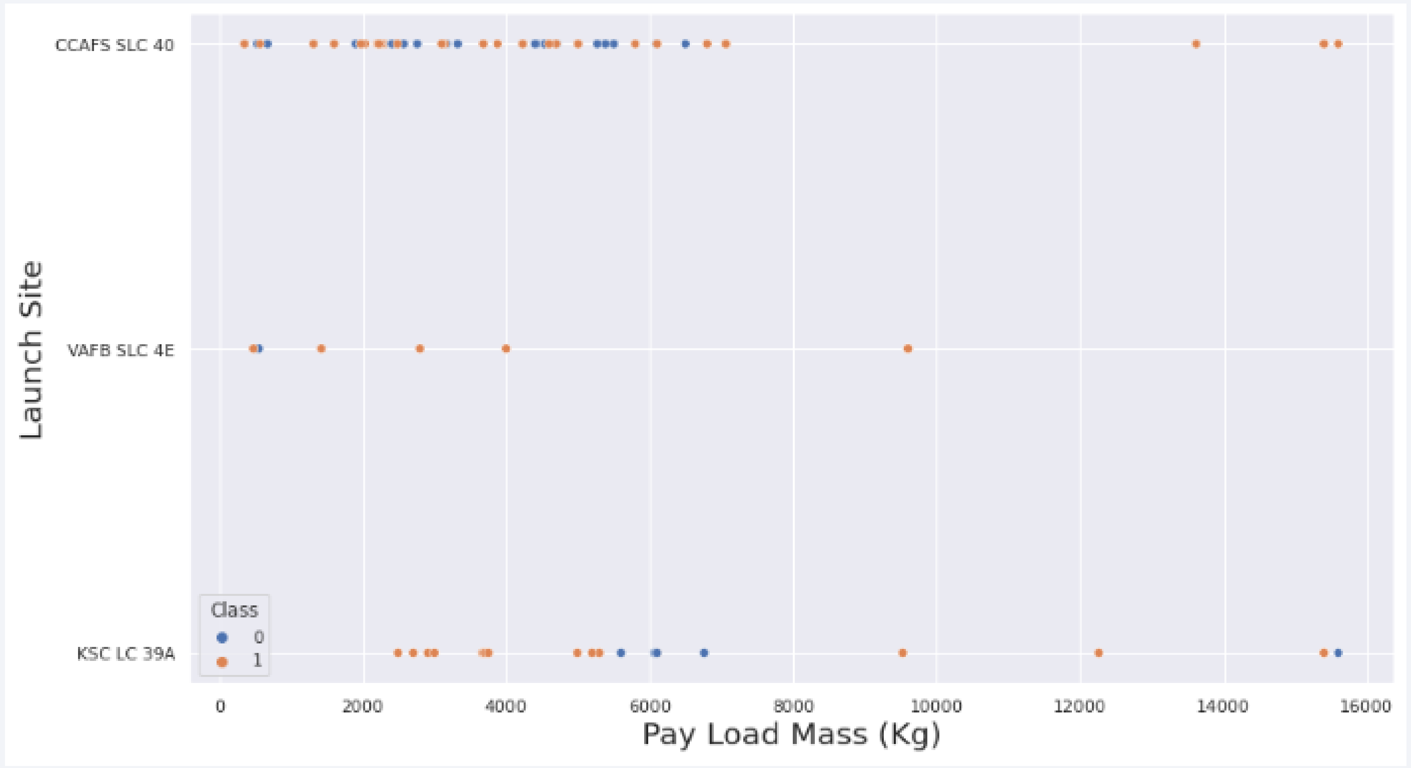
Flight Number vs. Launch Site



- This scatter plot shows that the larger the flights amount of the launch site, the greater the success rate will be

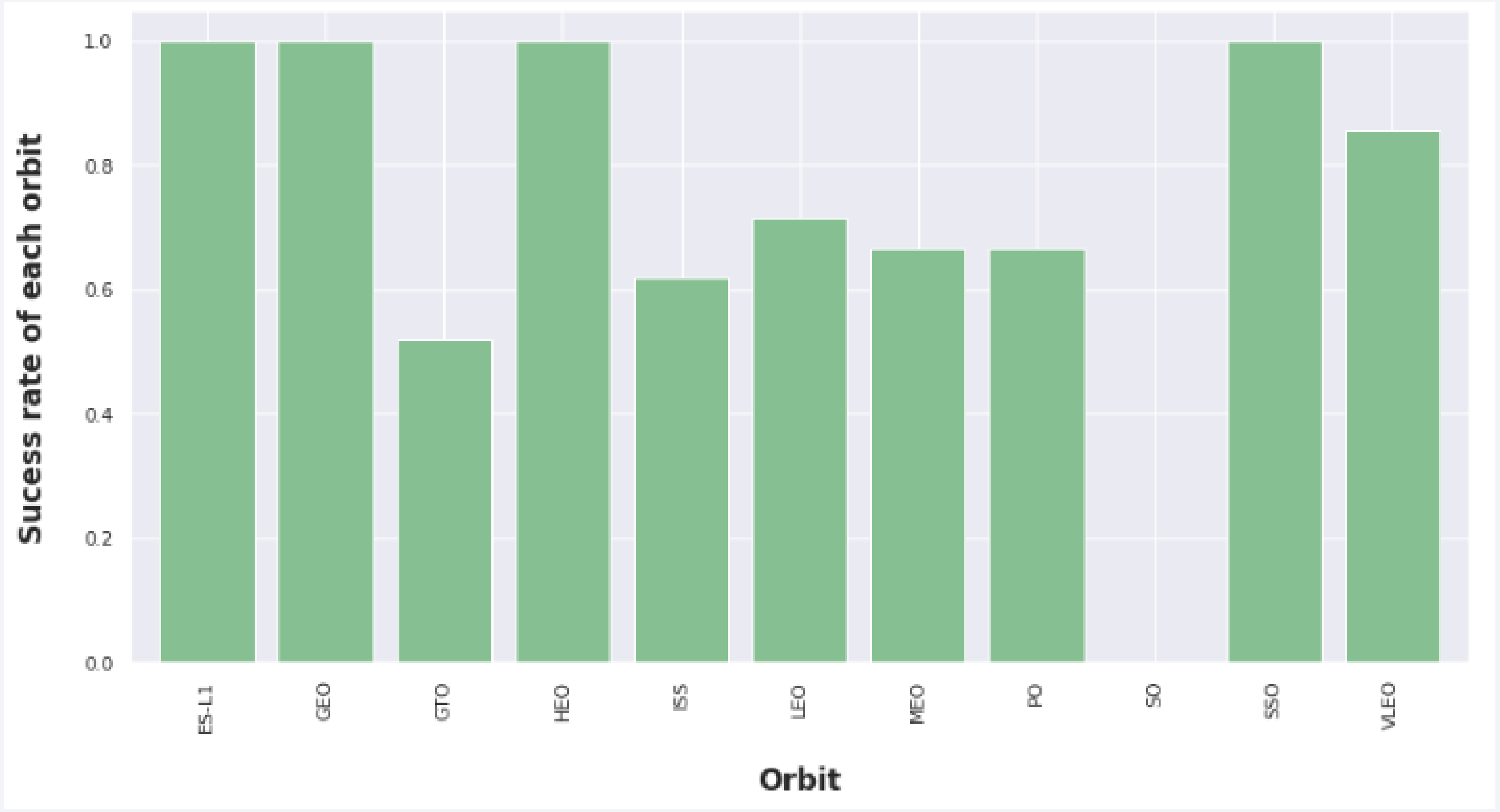
The launch site CCAFS SLC40 indicates the least launch attempts looking at the appearance of it on the chart

Payload vs. Launch Site



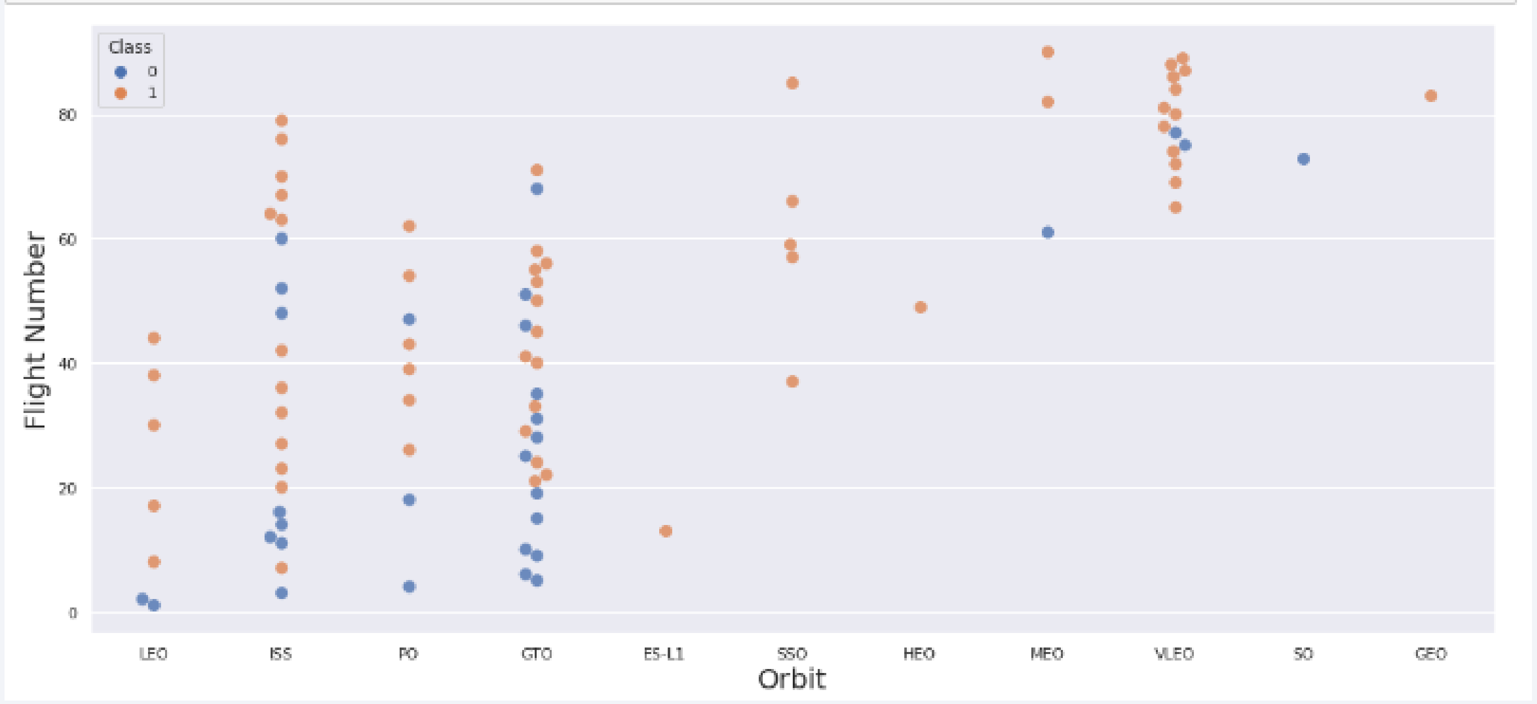
- This scatter plot shows once the pay load mass is greater than 7000kg, the probability of the success rate will be highly increased.
- However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate

Success Rate vs. Orbit Type



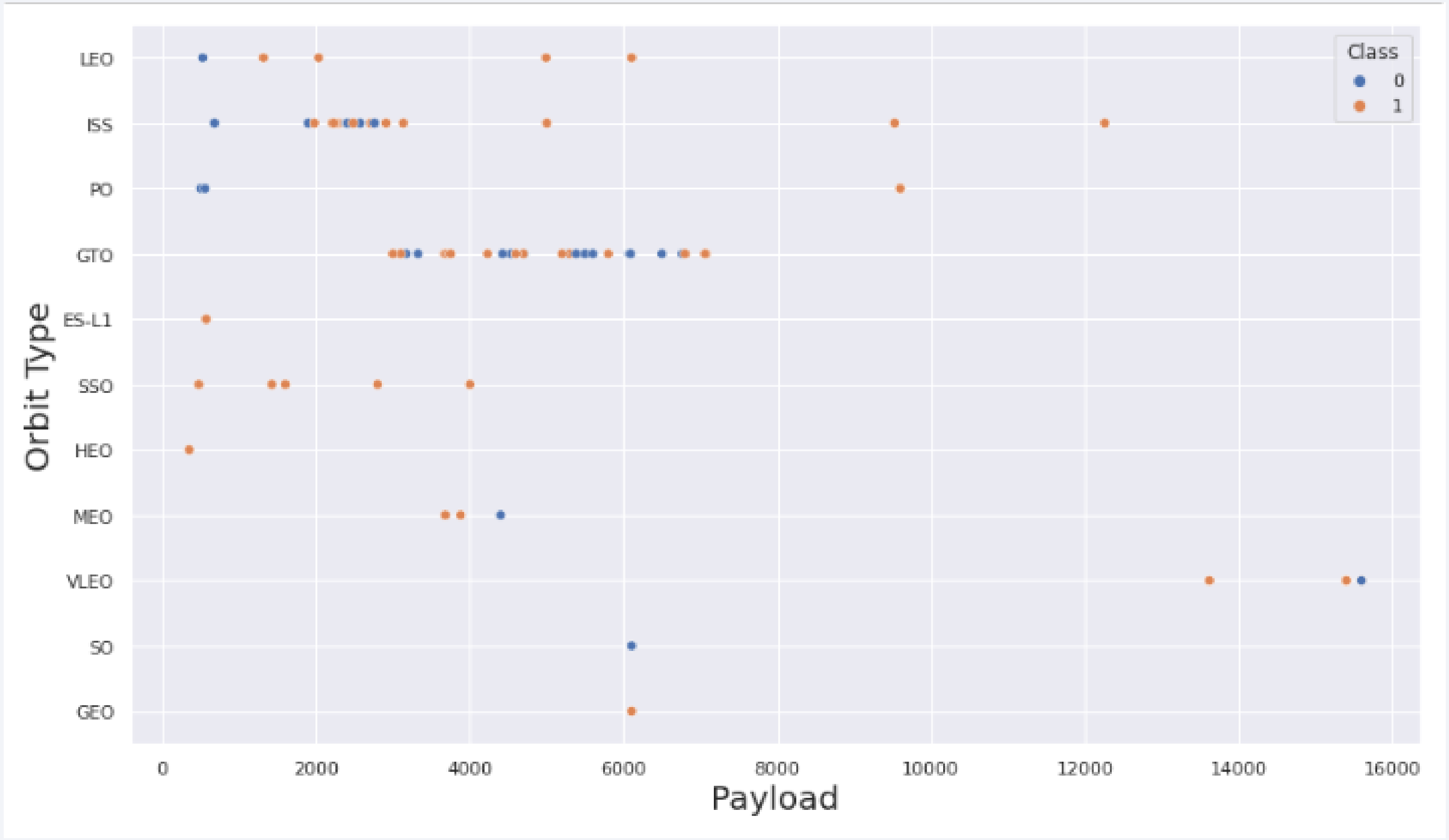
- This figure depicted the possibility of the orbits to influences the landing outcomes as some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.
- However, deeper analysis show that some of this orbits has only 1 occurrence such as GEO, SO, HEO and ES-L1 which mean this data need more dataset to see pattern or trend before we draw any conclusion.

Flight Number vs. Orbit Type



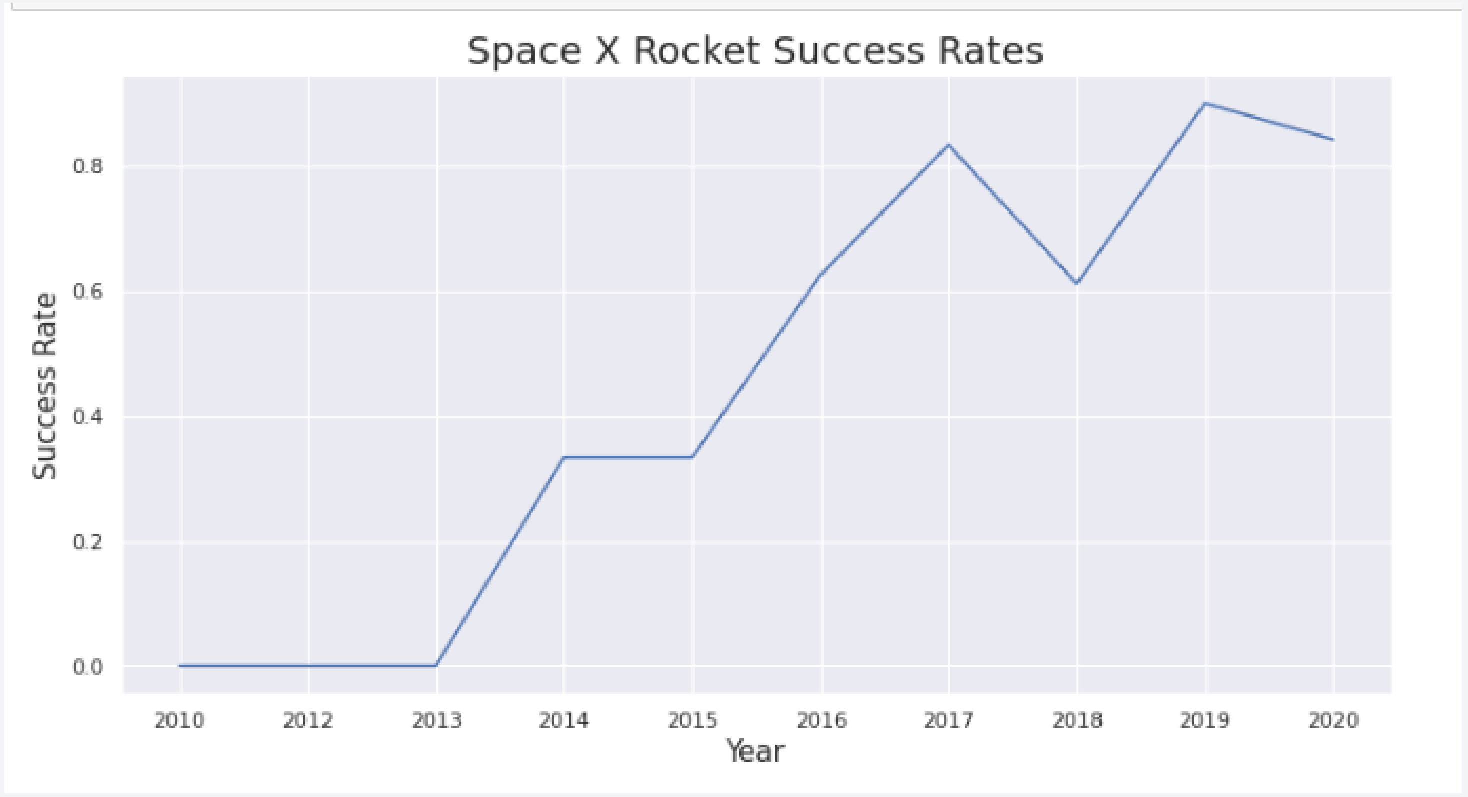
- This scatter plot shows that generally, the larger the flight number on each orbits, the greater the success rate (especially LEO orbit) except for GTO orbit which depicts no relationship between both attributes.
- Orbit that only has 1 occurrence should also be excluded from above statement as it's needed more dataset.

Payload vs. Orbit Type



- Heavier payload has positive impact on LEO, ISS and PO orbit. However, it has negative impact on MEO and VLEO orbit.
- GTO orbit seem to depict no relation between the attributes. Meanwhile, again, SO, GEO and HEO orbit need more dataset to see any pattern or trend

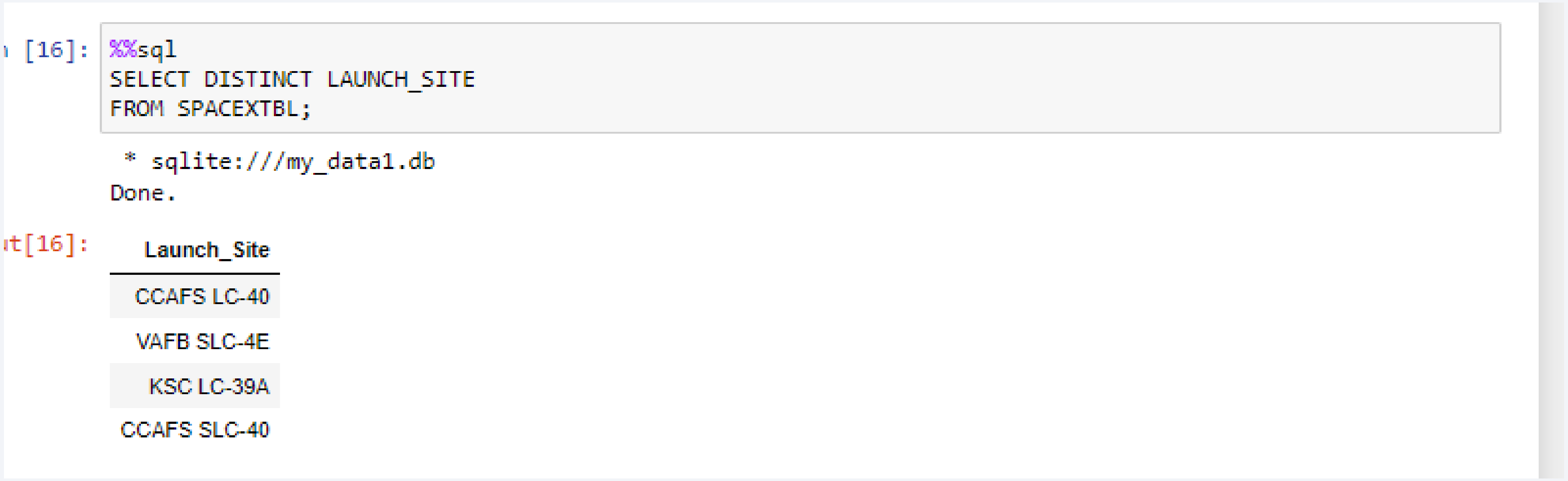
Launch Success Yearly Trend



- This figures clearly depicted and increasing trend from the year 2013 until 2020.
- If this trend continue for the next year onward. The success rate will steadily increase until reaching 1/100% success rate.

All Launch Site Names

- To find the unique launch sites we used Distinct



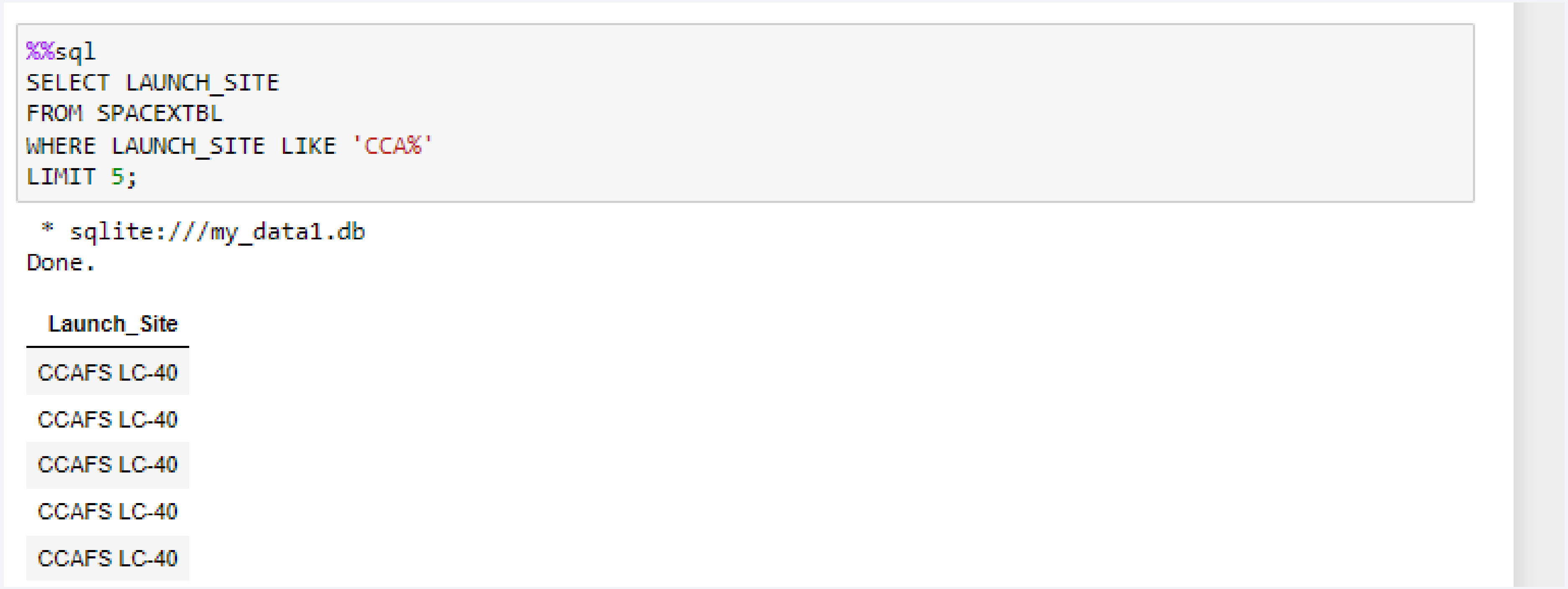
```
In [16]: %%sql
         SELECT DISTINCT LAUNCH_SITE
         FROM SPACEXTBL;

          * sqlite:///my_data1.db
         Done.

Out[16]:
          Launch_Site
         CCAFS LC-40
         VAFB SLC-4E
          KSC LC-39A
         CCAFS SLC-40
```

Launch Site Names Begin with 'CCA'

- To find all the records with the “CCA” we used “LIKE and limited the display to 5 rows”



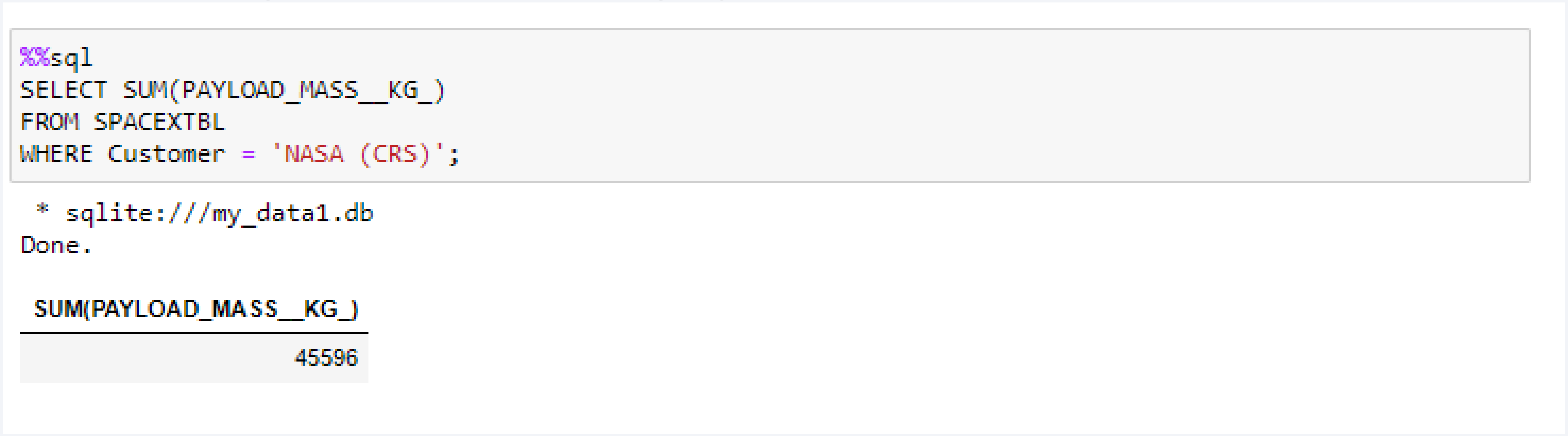
```
%%sql
SELECT LAUNCH_SITE
FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

Total Payload Mass

- To calculate the total payload carried by boosters from NASA we first use the sum to first get the total flirted to get just those labeled “NASA”



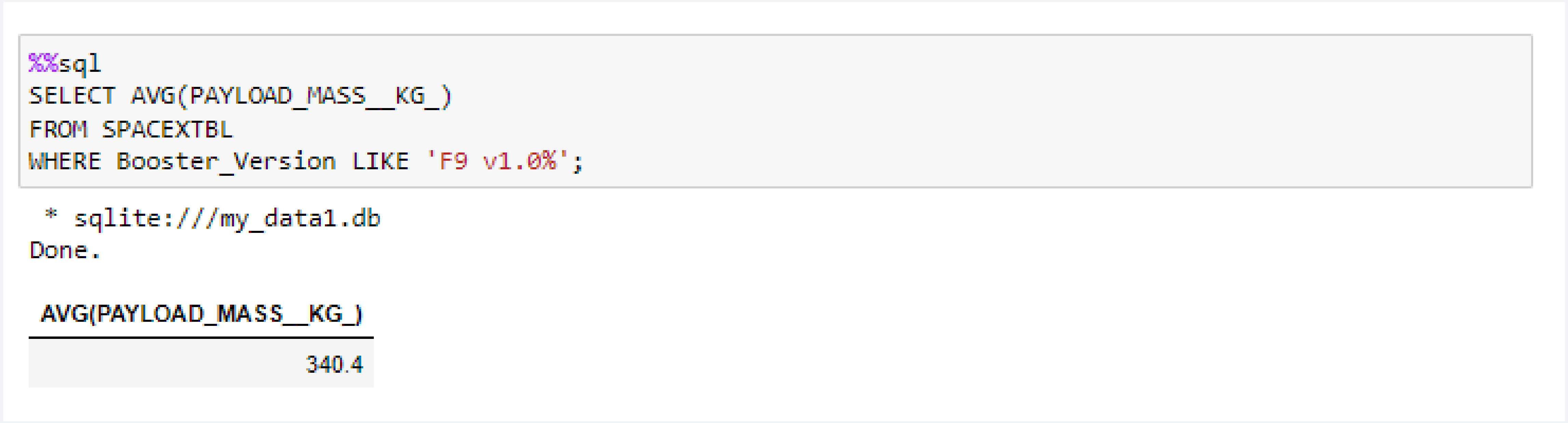
```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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



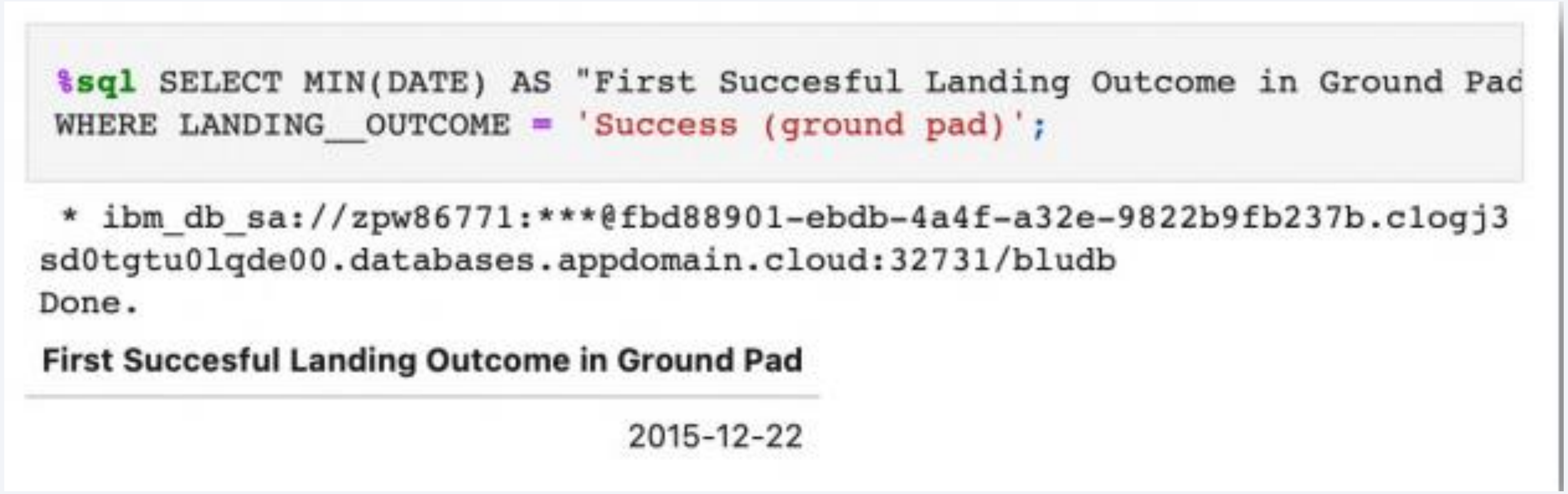
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Booster_Version LIKE 'F9 v1.0%';
```

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

AVG(PAYLOAD_MASS__KG_)
340.4

First Successful Ground Landing Date

- We use the min() function to find the result We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015



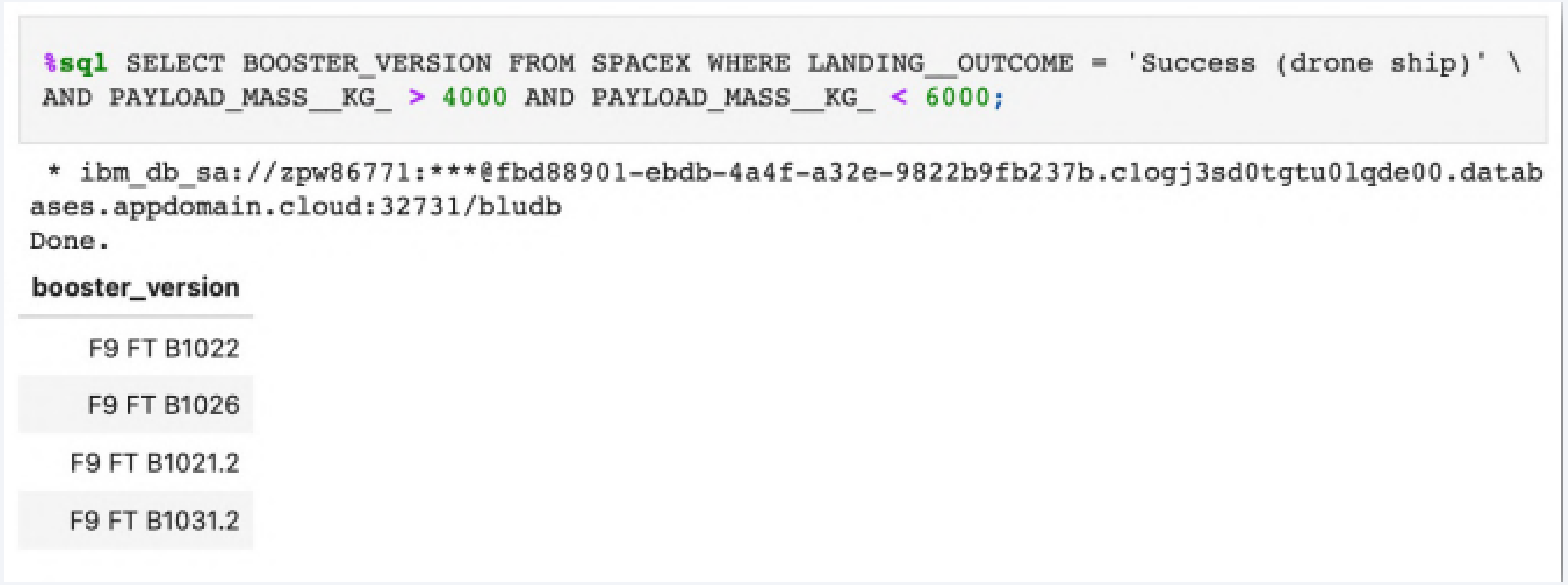
```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

First Succesful Landing Outcome in Ground Pad
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



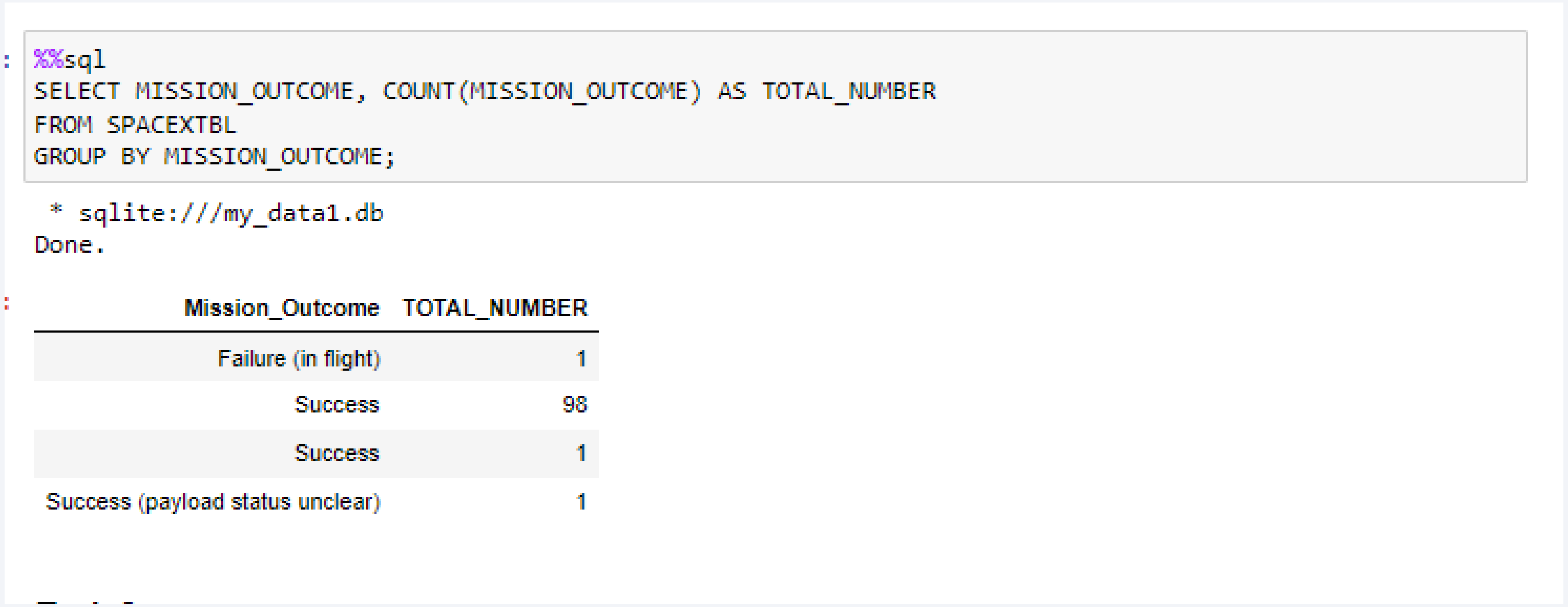
```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.datab
ases.appdomain.cloud:32731/bludb
Done.
```

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 used count method to filter for WHERE Mission_Outcome was a success or a failure

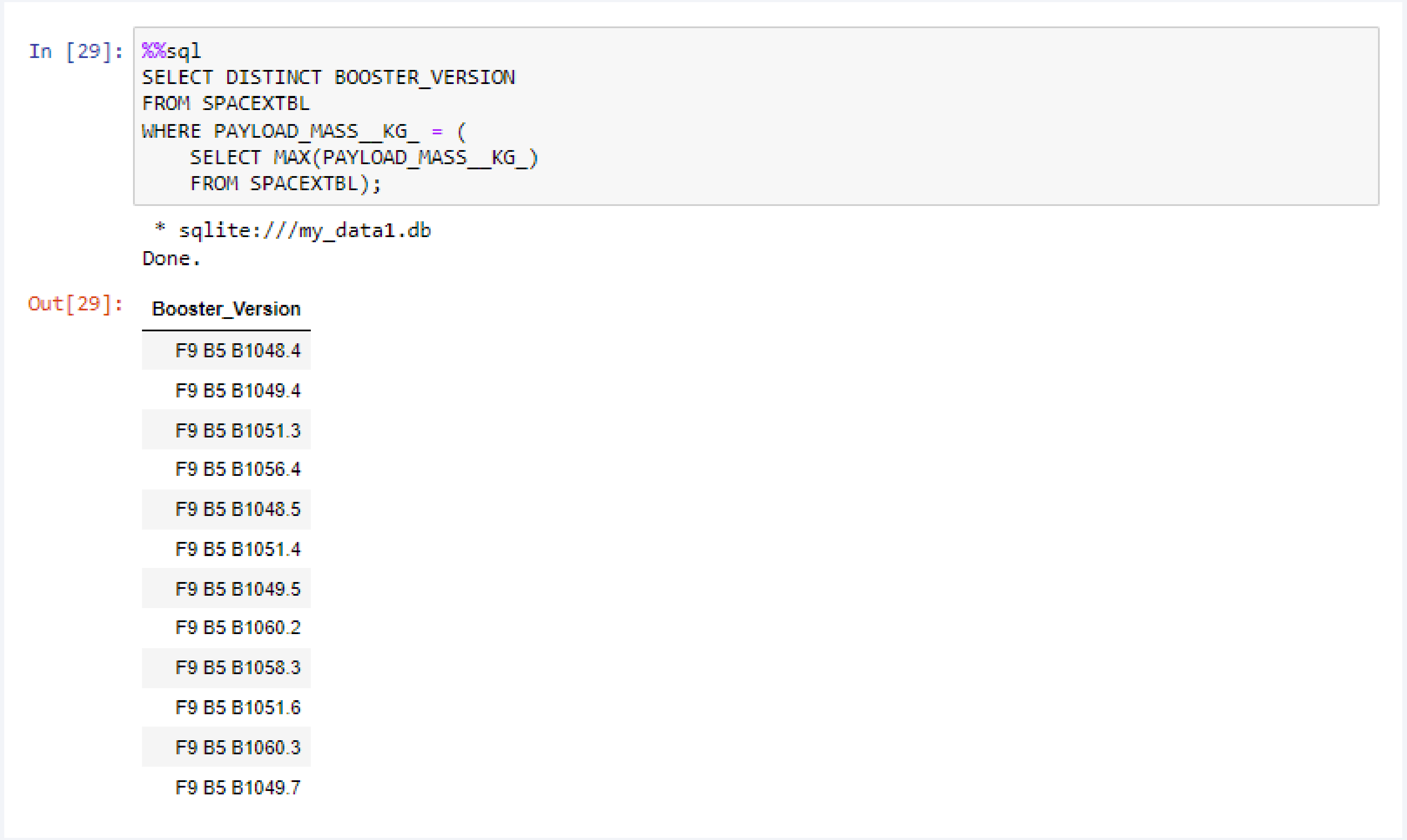


```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

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

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

Boosters Carried Maximum Payload



```
In [29]: %%sql
         SELECT DISTINCT BOOSTER_VERSION
         FROM SPACEXTBL
         WHERE PAYLOAD_MASS__KG_ = (
             SELECT MAX(PAYLOAD_MASS__KG_)
             FROM SPACEXTBL);

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

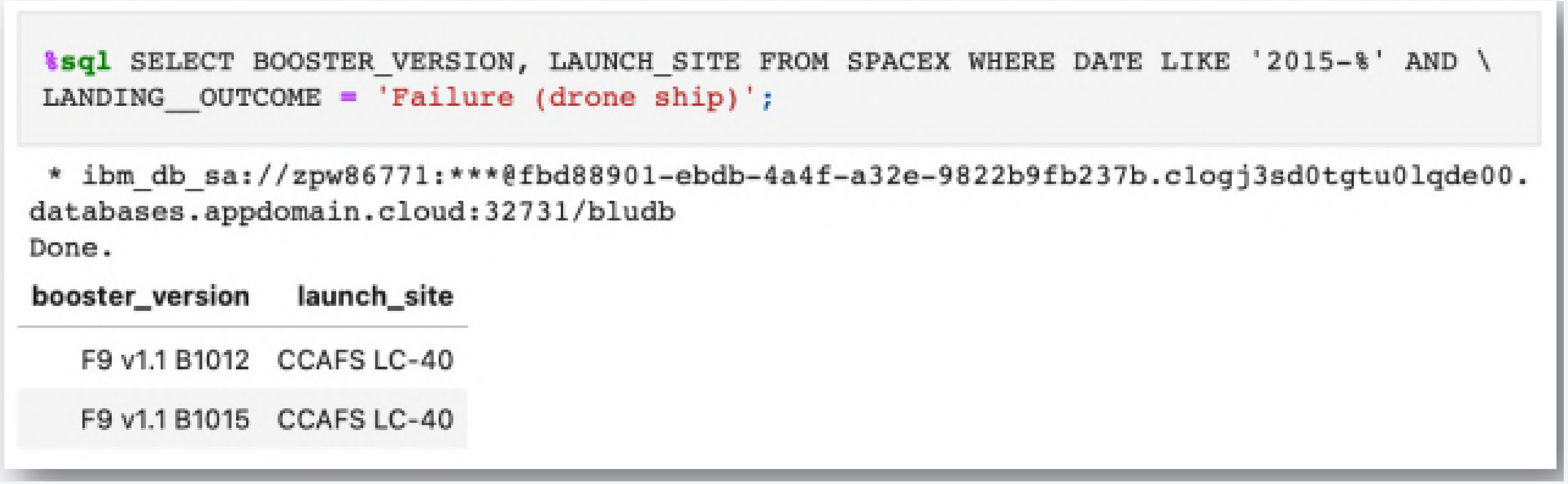
Out[29]:

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

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

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

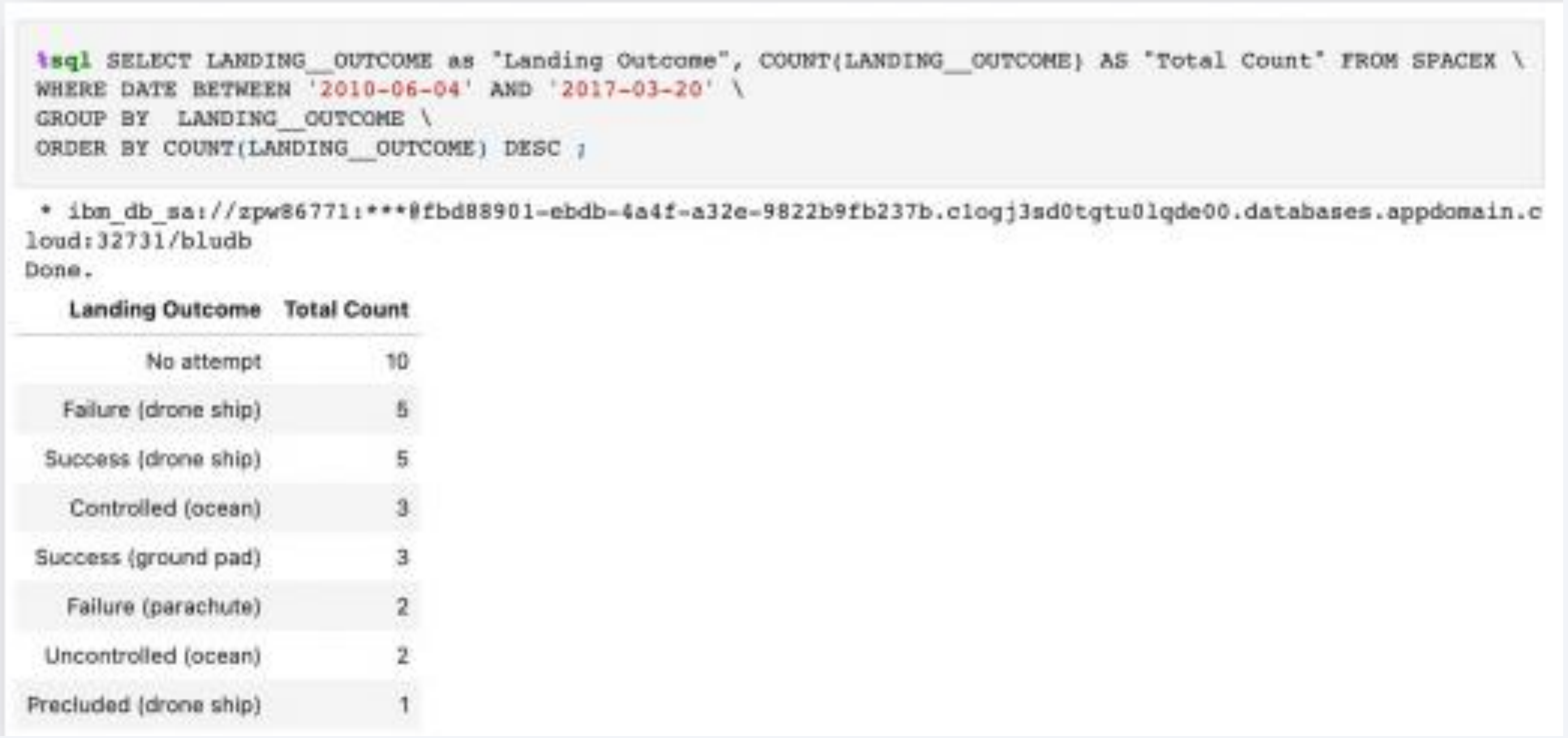


```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.
databases.appdomain.cloud:32731/bludb
Done.
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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



```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total 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://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.c
loud:32731/bludb
Done.
```

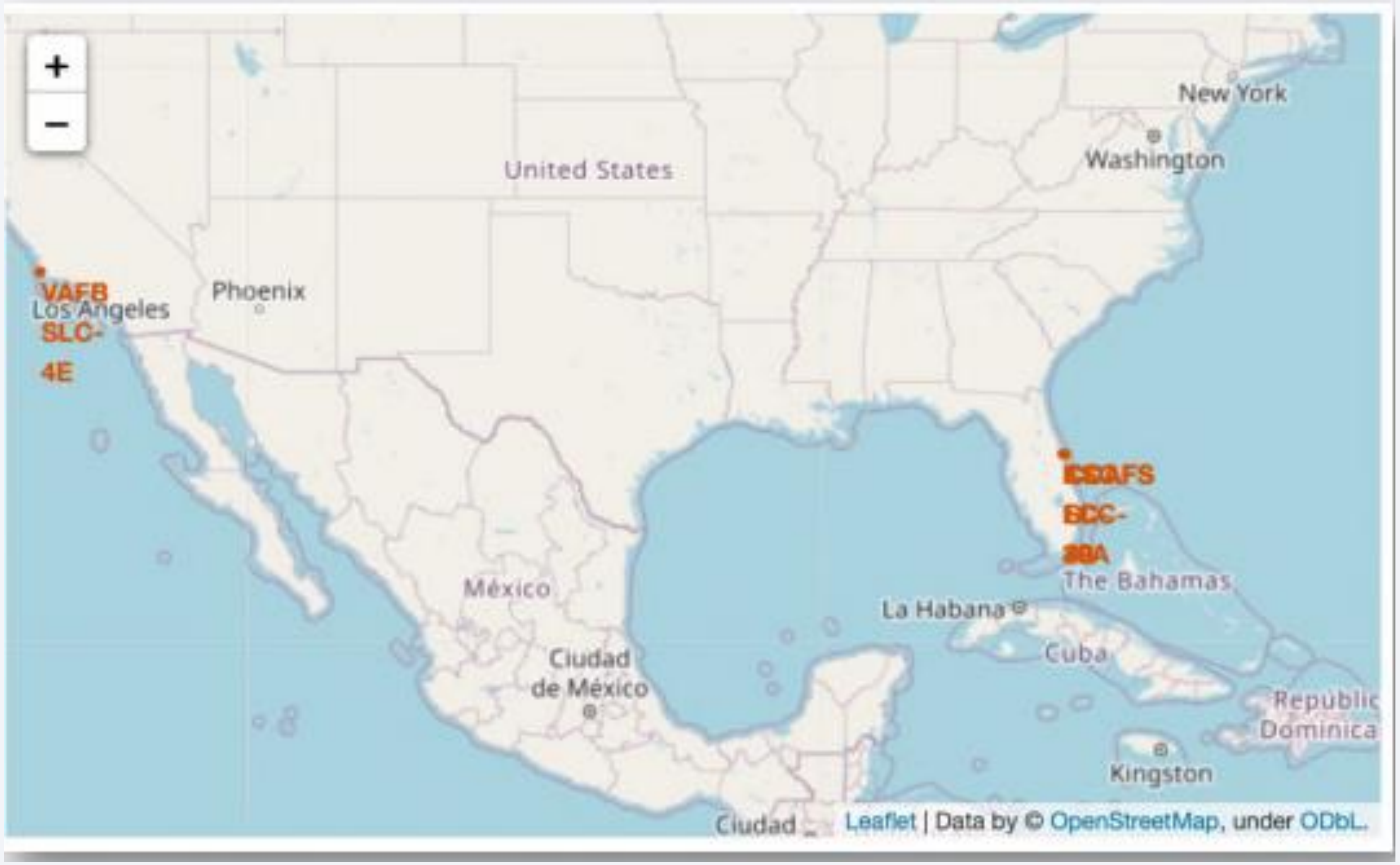
Landing Outcome	Total Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

- 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 3

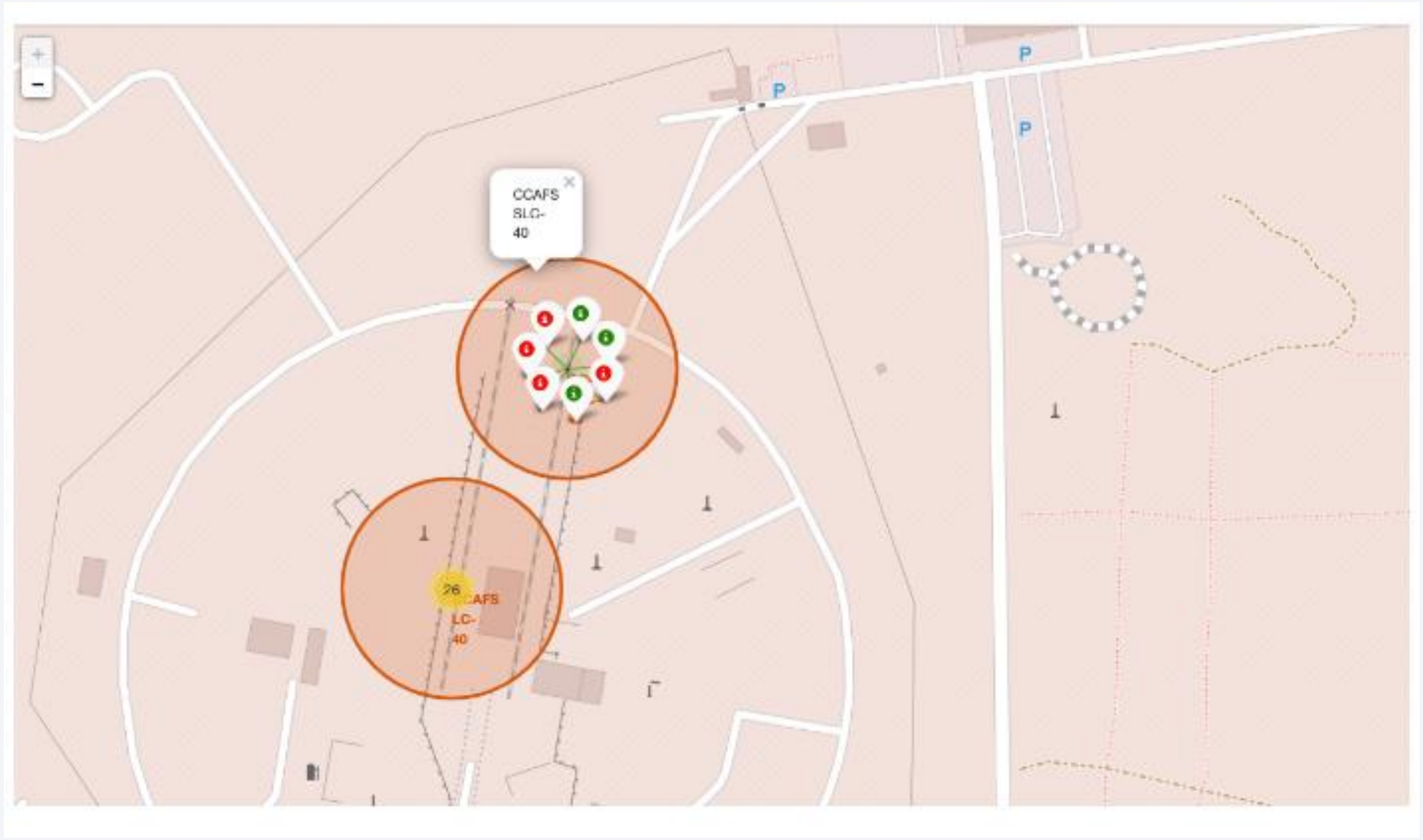
Launch Sites Proximities Analysis

Location of all launch sites



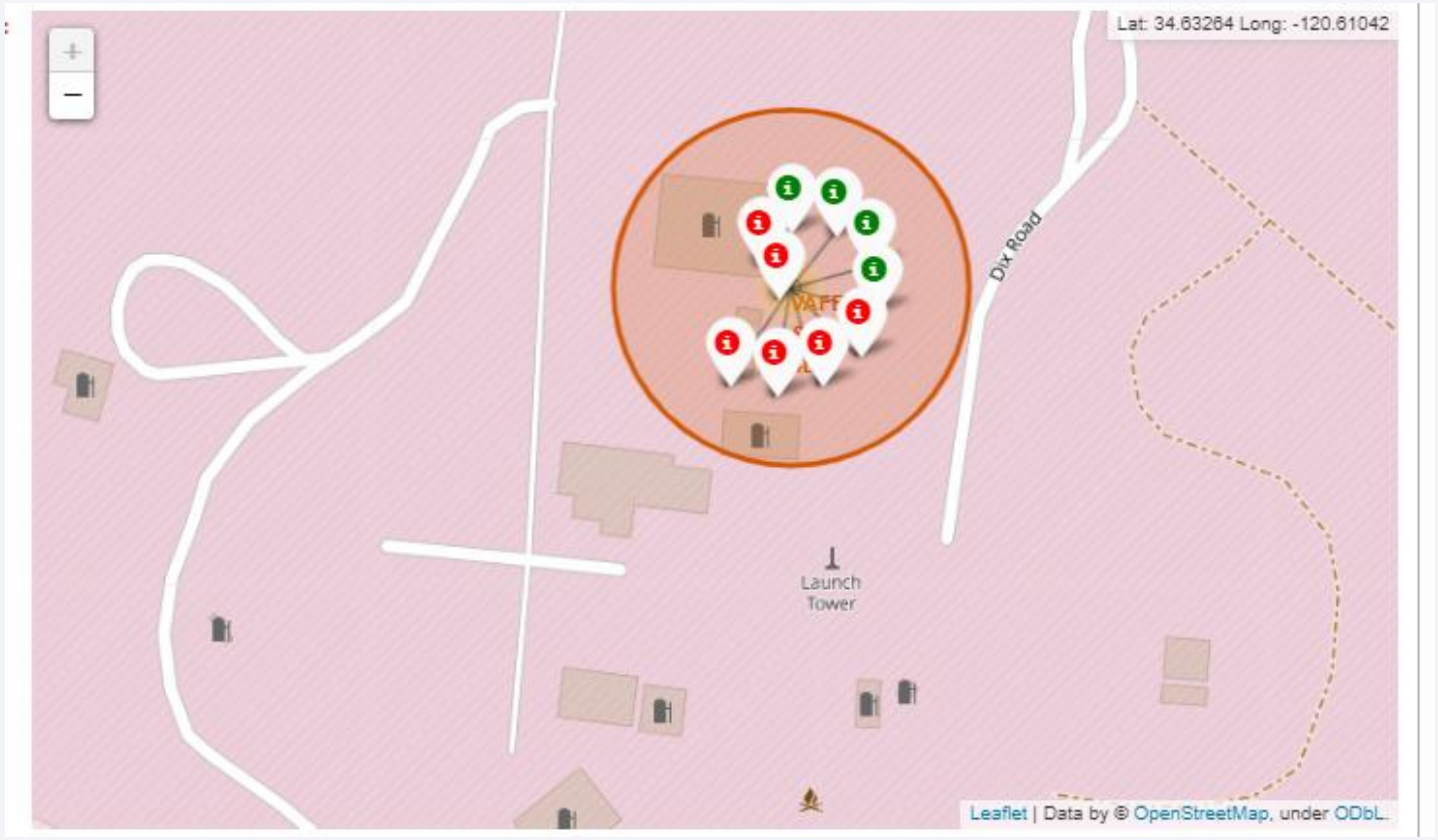
- We can see that all the SpaceX launch sites are located inside the United States

Markers showing Launch Sites with “Color Labels”



- The marker clusters identify with launch sites have the highest success rate with green and the lowest with red

Launch Sites Distance to Landmarks

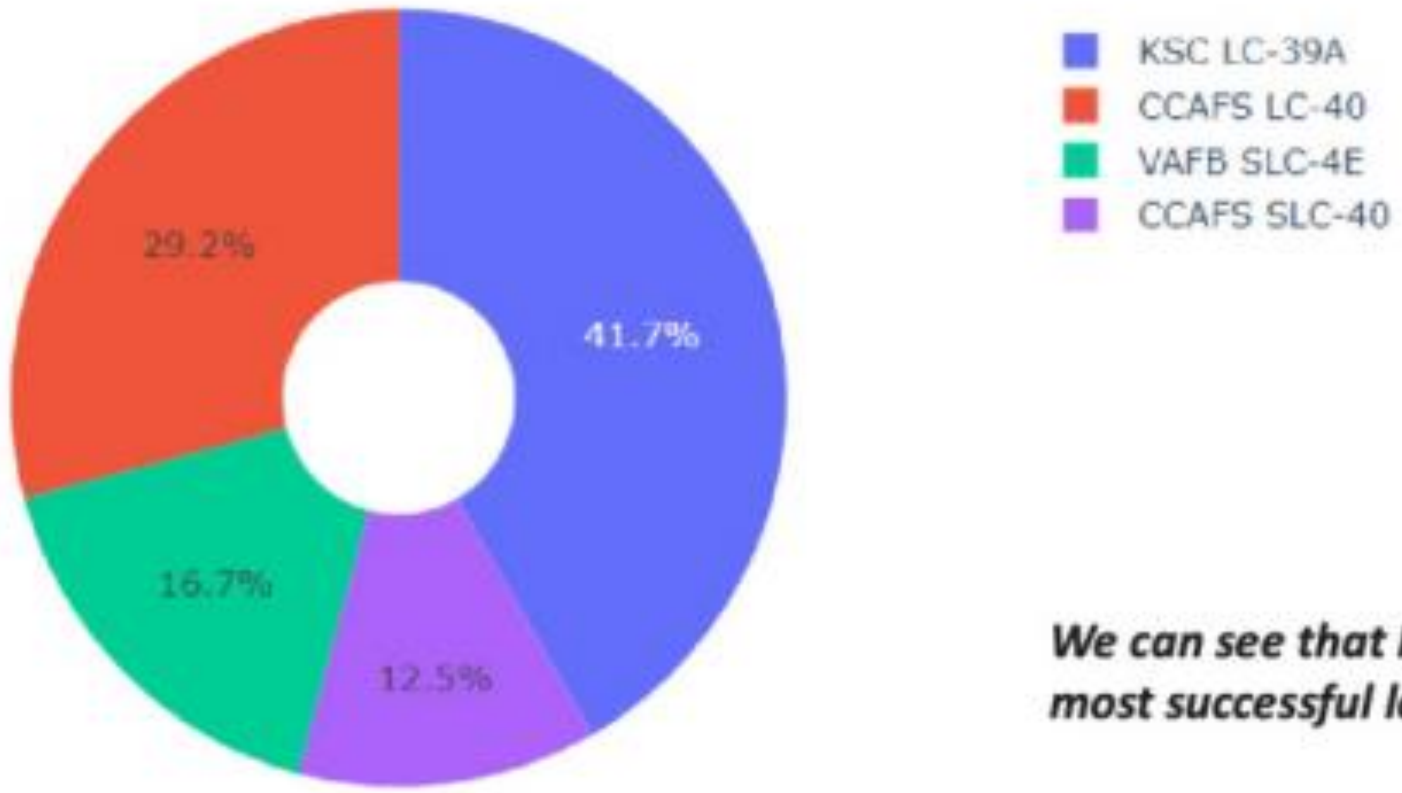

Lat: 34.63264 Long: -120.61042
Dix Road
Launch Tower
Leaflet | Data by © OpenStreetMap, under ODbL.

- In this screenshot we can identify important elements such as roads, railways and launch towers

Section 4

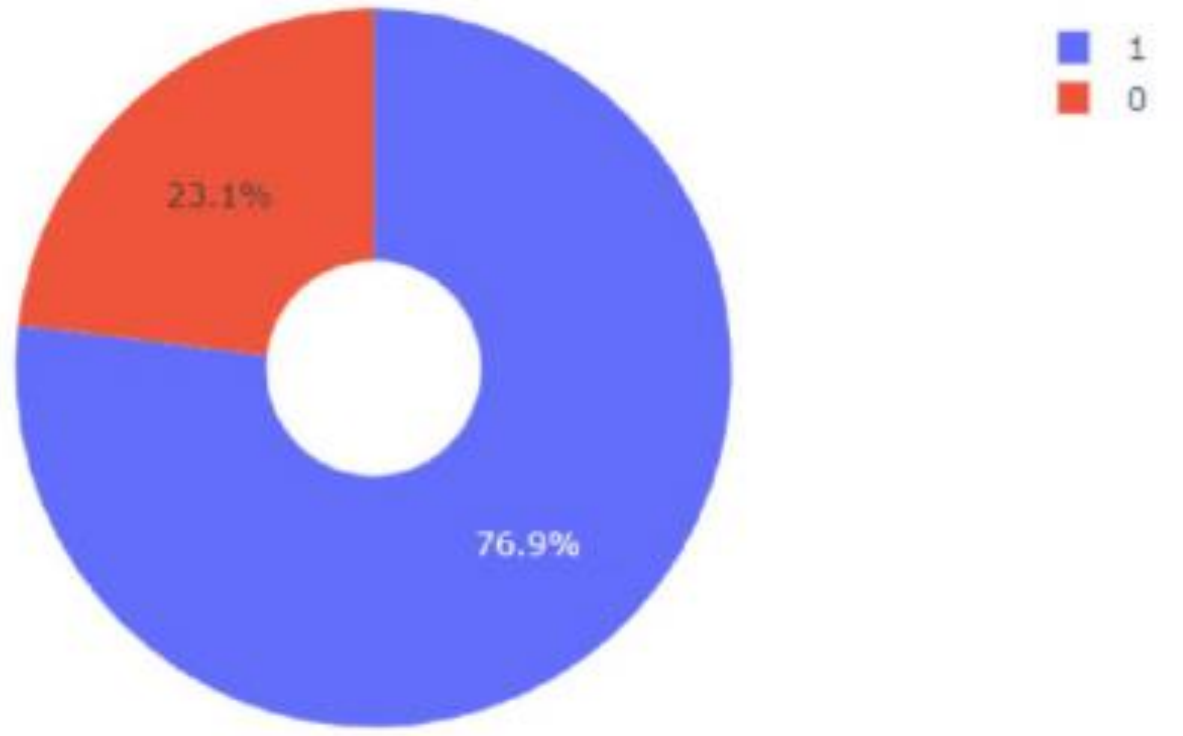
Build a Dashboard with Plotly Dash

Success Percentage by each launch sites



We can see that KSC LC-39A had the most successful launches from all the sites

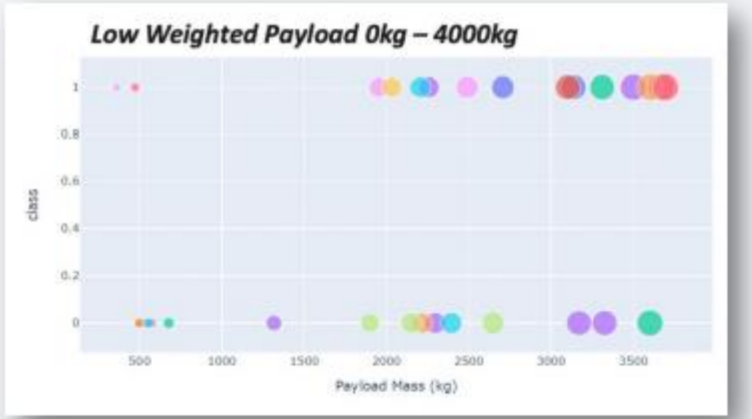
Highest Launch Success Ratio; KSC LC-39A

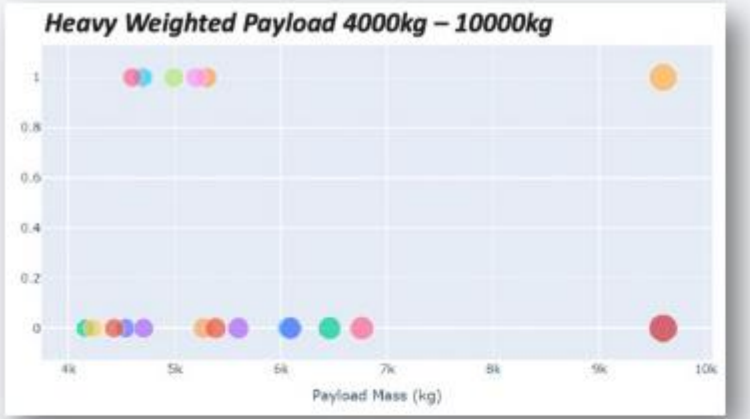


KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs Launch Outcome Scatter Plots

We can see that all the success rate for low weighted payload is higher than heavy weighted payload



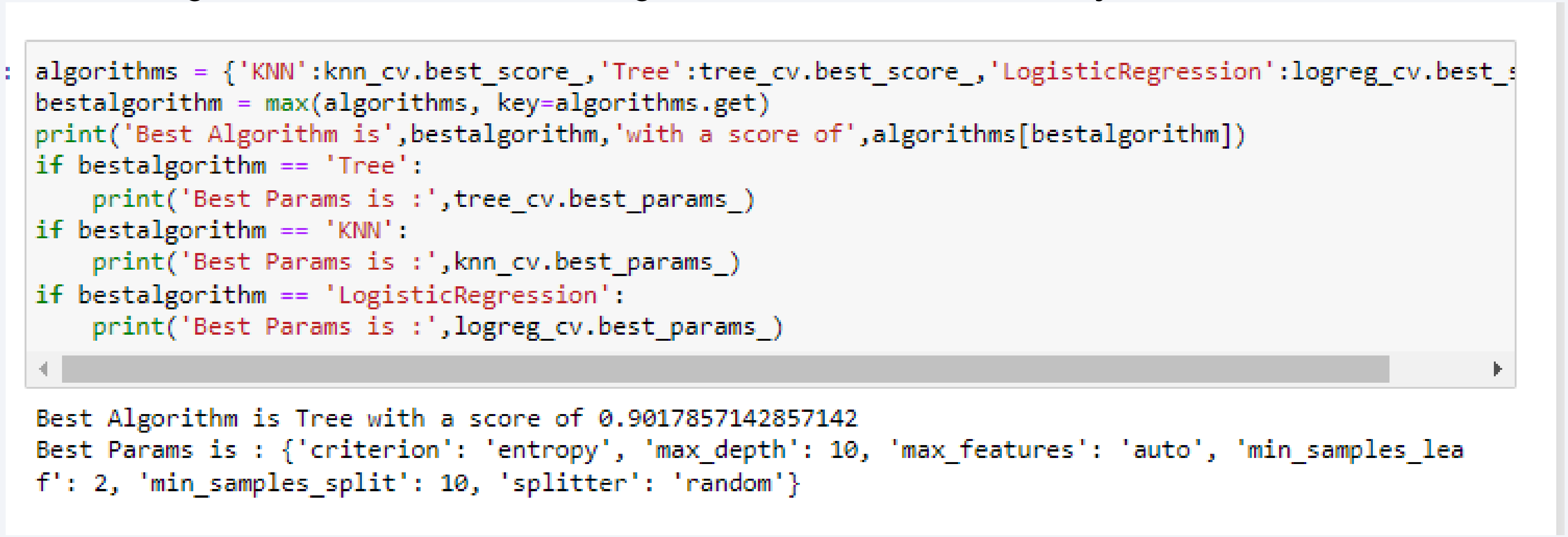


Section 5

Predictive Analysis (Classification)

Classification Accuracy

- As we can see, by using the code as below: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

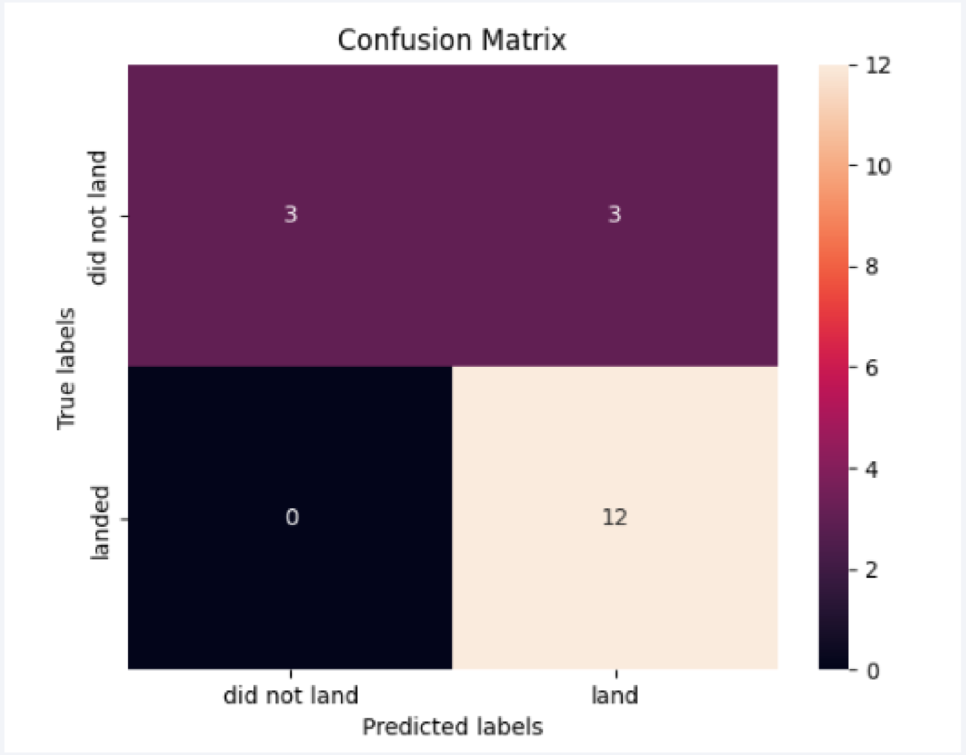


```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_s
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.9017857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'auto', 'min_samples_lea
f': 2, 'min_samples_split': 10, 'splitter': 'random'}
```

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 positives .i.e., unsuccessful landing marked as successful landing by the classifier



Conclusions

The Tree Classifier Algorithm is the best Machine Learning approach for this dataset.

The low weighted payloads (which define as 4000kg and below) performed better than the heavy weighted payloads.

Starting from the year 2013, the success rate for SpaceX launches is increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.

KSC LC-39A have the most successful launches of any sites; 76.9%

SSO orbit have the most success rate; 100% and more than 1 occurrence

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!