IBM Developer SKILLS NETWORK

Winning Space Race with Data Science

Jimmy Kayongo
2024 August 23rd

Outline

Executive Summary

- On the journey to study and understand how one company has won the modern space race, the use of data science explores some of the more effective methodologies to obtain, analyze and discuss insights contained within data.
- Although there are many methodologies, this case study utilizes data collection by scraping the internet, data cleaning and preparation in data wrangling, exploratory analysis by querying using SQL, exploratory visualization using Folium, and machine learning techniques in prediction by applying among others logistic regression, support vector machinery, classification trees, as well as k-nearest neighbors.
- The collected data shows that various features of the data correlate and these correlations can even be visualized on a map. The data supports an ability to predict if a launch will be successful with a considerable level of confidence, drawn from observed levels of favorable accuracy from the methods applied in this study.

Introduction

- Information available at SpaceX's website shows that their work horse rocket platform Falcon 9 costs only a fraction of what their competitors charge. The whopping difference is derived from the ability to reuse the booster stage and other parts of the system because the booster stage can land, and other parts of the system can be recovered for reuse with ever increasing efficiency.
- This result has put the company in leading position as far as mass to orbit is concerned all while the runner ups remain at a staggering distance behind. As such, it is critical to understand what facilitates the booster stage's capability to land, and obtain a reliable confidence to also predict if it will, for every launch.
- On the other hand, it is known that there exist launches of this system for which the booster stage did not land. At the same time, most of the unsuccessful landings are known to have been planned to where a controlled 'landing' in the ocean was instead performed. In the study, we try to understand the correlation between the launch locations from which the booster landed as well those locations from which the booster did not land, planned or not.
- How are the various parameters of a launch, for example payload mass, launch site, ancillary supporting facilities by the launch site, customer type, and many others affecting the launch outcome? These questions are addressed in this study.

Section 1

Methodology

Methodology

Executive Summary:

The methodology section describes the process from data collection, cleanup and study, predictive analysis, to visualization and explanation of insights obtained.

- Data collection:

The data for the tasks presented in this project report was obtained by grant of access to preset database sources employing use of an API call, and scraping of publicly available information sites like Wikipedia.

- Data wrangling:

All data used for the tasks in this project report was obtained in tabular form and was immediately organized initially, into a Pandas dataframe. There was a step to either standardize, handle missing values, or clipping columns that were not applicable. In each case, a copy of the cleaned data was then saved in a comma separated values file (csv) for easier subsequent access.

- Performance of exploratory data analysis (EDA) using visualization and SQL:

In the exploratory phase, SQL was applied to filter out key pieces of information like the unique launch site names and codes. This was then followed up with visualization of various data correlations in plots of all types as well as pinning of the launch site locations on a map using Folium.

- Interactive visual analytics using Folium and Plotly Dash:

Further, as part of the Exploratory analysis included, interactive visuals as mentioned before. Using Plotly, a live application based on a browser was constructed with access to manipulating the input parameters like payload mass, and selection of launching site. In return, the display showed corresponding launch success and failure proportions as well as launches that were within the selected payload mass range. The chart types were Pie, Scatter as well as maps.

- Predictive analysis using classification models:

The final part of the project was a task to create the most efficient and accurate predictive model on if a launch will have its booster stage land regardless of the target land site. Among the available predictive model types, four were tested on the same data. Among them was Logistic Regression, Decision Tree Classifier, Support Vector Machinery, and K-nearest neighbors. For the test on each of these model types, the pipeline for the split data was a Grid Search Vector with various appropriate parameters and a ‘cv’ parameter set to 10. Again in each case the model was trained by fitting to the training set and tested by checking the accuracy and best estimator score values when a prediction was made using the test set of the data. To note, the split line was set at 80:20 training to test sample proportions.

Data Collection

Two data collection techniques were used; API calls and web scraping.

1. The URL shown below points to a provided API with spacex launch data and was one of the resources utilized to obtain data that was analyzed for this project.

"https://api.spacexdata.com/v4/launches/past"

2. For comparison, similar launch data is available at Wikipedia but without direct access. As such, scraping methods were applied to obtain this same information to aggregate to file in a similar manner to what was obtained from the API.

"https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

https://github.com/kayscaiwer/capstone/blob/main/capstone_data_collection.ipynb

Data Collection – SpaceX API

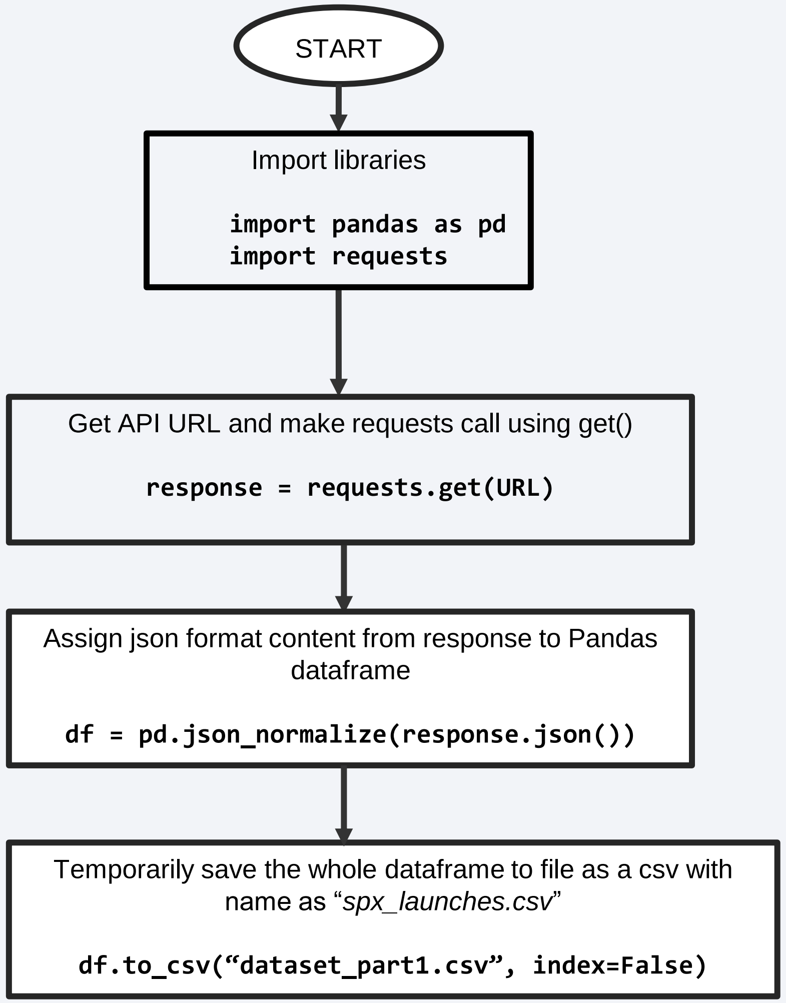
Data was collected by making API calls to provided URL, converting it to manageable format and then saving to user-friendly format for future use in the following steps.

1. API call to get general content if the response corresponded to a successful call.

2. Assuming the call was successful, the content was initially handled in the existing JSON format.

3. The JSON format is converted to a Pandas dataframe. Some of the columns are removed to keep only those that will be applicable but that is discussed under the subsequent step in the process.

4. The Pandas dataframe is written to file in a popular comma separated values format, (CSV).


START
Import libraries
Get API URL and make requests call using get()
Assign json format content from response to Pandas dataframe
Temporarily save the whole dataframe to file as a csv with name as "*spx_launches.csv*"

```
import pandas as pd
import requests
```

```
response = requests.get(URL)
```

```
df = pd.json_normalize(response.json())
```

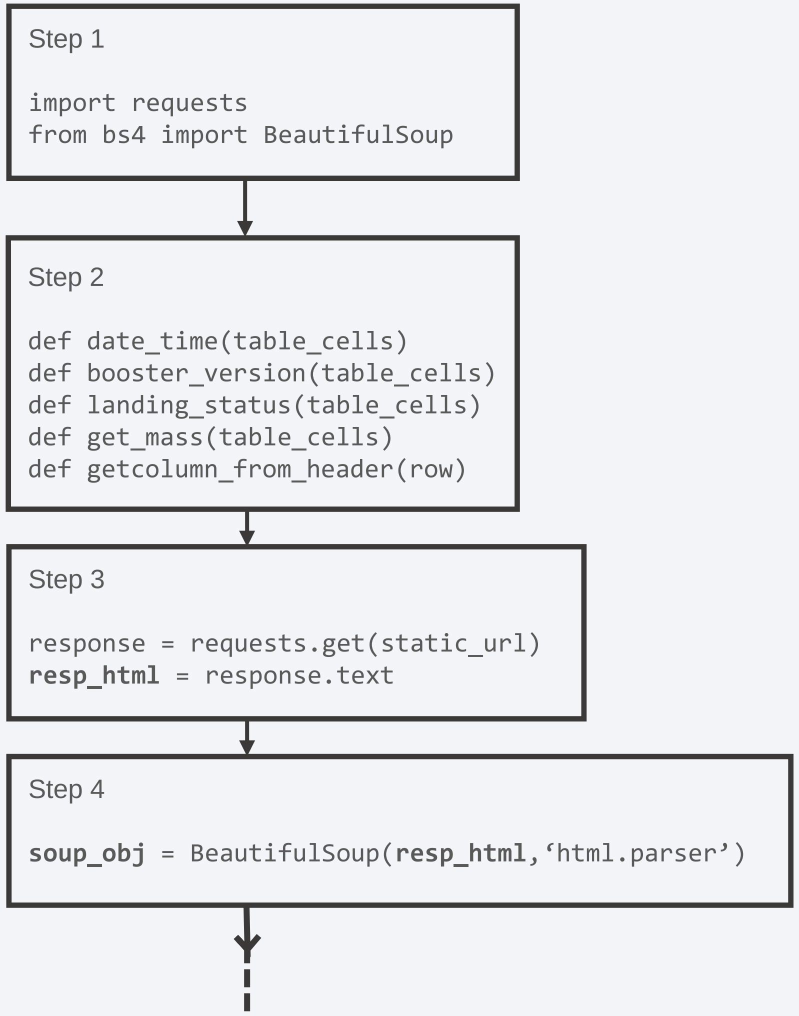
```
df.to_csv("dataset_part1.csv", index=False)
```

https://github.com/kayscaiwer/capstone/blob/main/capstone_data_collection.ipynb

Data Collection - Scraping

The process for the scraping of the Wikipedia source went as follows:

1. Import of the necessary files including `requests`, and `BeautifulSoup`
2. Creation of helper functions to handle the wrangling of the returned text
3. Initiation of requests call to the target URL
4. Creation of a BeautifulSoup object with addition of the parser parameter



Step 1

```
import requests
from bs4 import BeautifulSoup
```

Step 2

```
def date_time(table_cells)
def booster_version(table_cells)
def landing_status(table_cells)
def get_mass(table_cells)
def getcolumn_from_header(row)
```

Step 3

```
response = requests.get(static_url)
resp_html = response.text
```

Step 4

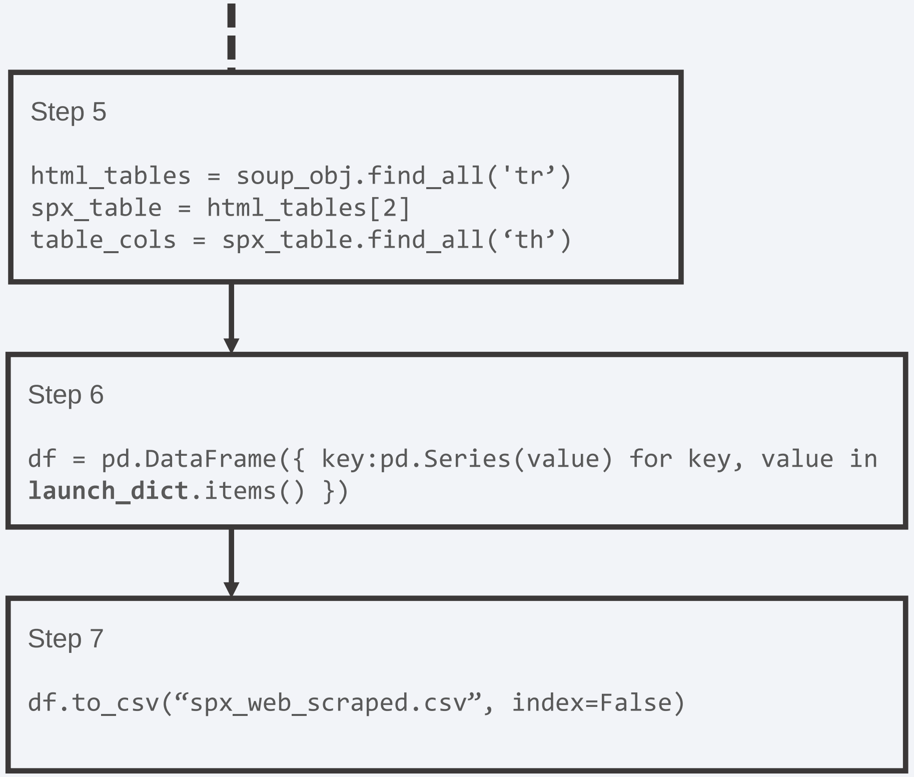
```
soup_obj = BeautifulSoup(resp_html,'html.parser')
```

https://github.com/kayscaiwer/capstone/blob/main/capstone_collect_webscraping.ipynb

Data Collection - Scraping

The process for the scraping of the Wikipedia source went as follows:

1. Import of the necessary files including `requests`, and `BeautifulSoup`
2. Creation of helper functions to handle the wrangling of the returned text
3. Initiation of requests call to the target URL
4. Creation of a BeautifulSoup object with addition of the parser parameter
5. Extraction of target table and all column names from that HTML table. (Note: table 3)
6. Creation of a Pandas dataframe using the aggregated HTML table columns from the scrape in addition to new ones added manually to a dictionary called “launch_dict”.
7. Saving the table to file in CSV format.



Step 5

```
html_tables = soup_obj.find_all('tr’)
spx_table = html_tables[2]
table_cols = spx_table.find_all(‘th’)
```

Step 6

```
df = pd.DataFrame({ key:pd.Series(value) for key, value in
launch_dict.items() })
```

Step 7

```
df.to_csv(“spx_web_scraped.csv”, index=False)
```

https://github.com/kayscaiwer/capstone/blob/main/capstone_collect_webscraping.ipynb

Data Wrangling

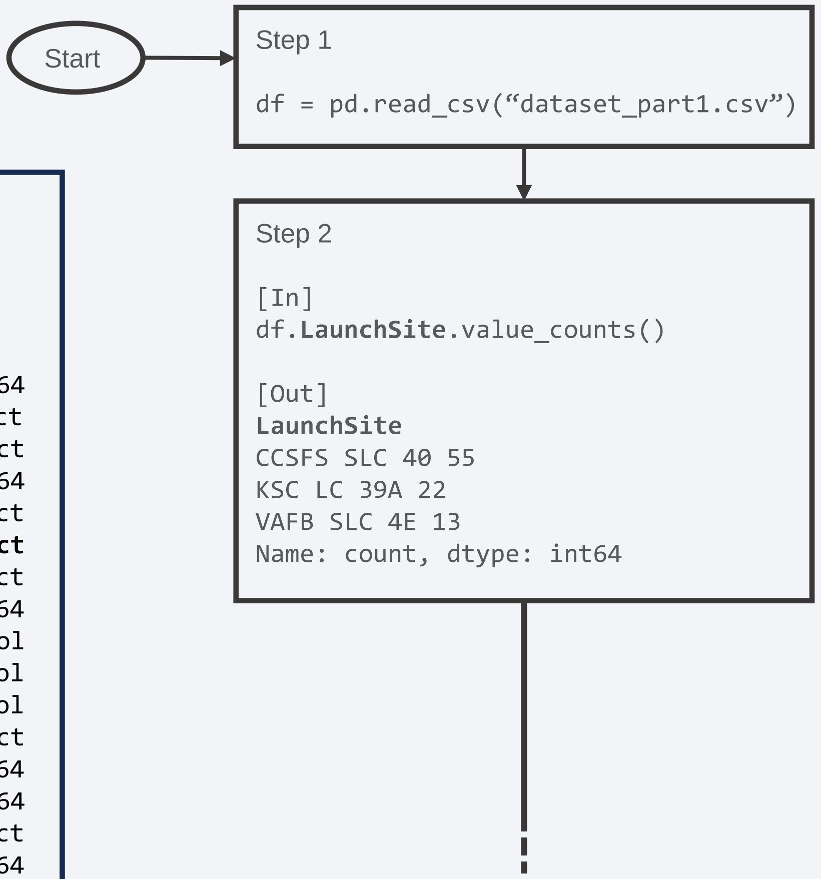
At this point there's a dataframe with information but it still has missing values and other pieces of information to extract, thus the need for data wrangling.

The main objectives for data wrangling were preliminary exploratory analysis as well as determination of training data labels. The following are the steps taken to prepare the data for further utilization:

1. Load the data from where it was last saved
2. Determine the number of launches from each active launch site.
3. Calculate the frequency of each unique orbit as a launch target.
4. Calculation of the number of occurrence of outcome per the target orbit.
5. Creation of landing outcome label using landing outcome column.

These are the columns in the loaded dataset

```
df.dtypes:


FlightNumber           int64
Date                  object
BoosterVersion        object
PayloadMass          float64
Orbit                 object
LaunchSite            object
Outcome               object
Flights                int64
GridFins                bool
Reused                  bool
Legs                    bool
LandingPad            object
Block                float64
ReusedCount            int64
Serial                object
Longitude            float64
Latitude             float64

dtype: object
```



Step 1

```
df = pd.read_csv("dataset_part1.csv")
```

Step 2

```
[In]
df.LaunchSite.value_counts()

[Out]
LaunchSite
CCSFS SLC 40 55
KSC LC 39A 22
VAFB SLC 4E 13
Name: count, dtype: int64
```

https://github.com/kayscaiwer/capstone/blob/main/capstone_prep_wrangling.ipynb

Data Wrangling

1. Load the data from where it was last saved
2. Determine the number of launches from each active launch site.
3. Calculate the frequency of each unique orbit as a launch target.
4. Calculation of the number of occurrence of outcome per the target orbit.
5. Creation of landing Outcome label using landing Outcome column.

Step 3

```
[In]
df.Orbit.value_counts()

[Out]
Orbit
GTO         27
ISS         21
VLEO        14
PO           9
LEO          7
SSO          5
MEO          3
ES-L1        1
HEO          1
SO           1
GEO          1
Name: count, dtype: int64
```

Step 4

```
[In]
landing_outcomes =
df.Outcome.value_counts()

[Out]
Outcome
True ASDS     41
None None     19
True RTLS     14
False ASDS     6
True Ocean     5
False Ocean    2
None ASDS      2
False RTLS     1
Name: count, dtype: int64
```

Step 5

```
[In]
landing_class = []
for val in df.Outcome.values:
    splitstr = val.lower().split()
    if ('false' in splitstr) or ('none' in splitstr):
        landing_class.append((val, 0))
    else:
        landing_class.append((val, 1))
```

Step 5

```
[Out]
('Outcome','Class')
('None None', 0)
('None None', 0)
('None None', 0)
('False Ocean', 0)
('None None', 0)
('None None', 0)
('True Ocean', 1)
('True Ocean', 1)
('None None', 0)
('None None', 0)
```

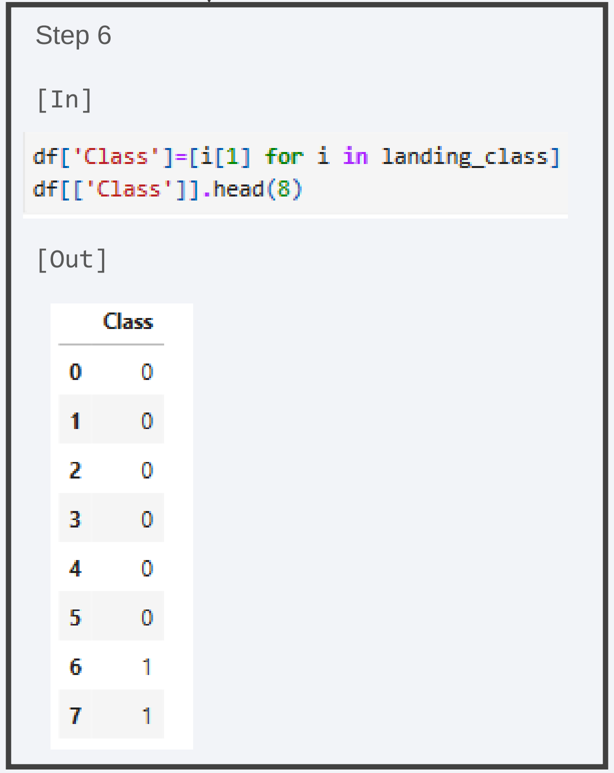
Add the GitHub URL of your completed data wrangling related notebooks, as an external reference and peer-review purpose

https://github.com/kayscaiwer/capstone/blob/main/capstone_prep_wrangling.ipynb

Data Wrangling

1. Load the data from where it was last saved
2. Determine the number of launches from each active launch site.
3. Calculate the frequency of each unique orbit as a launch target.
4. Calculation of the number of occurrence of outcome per the target orbit.
5. Creation of landing Outcome label using landing Outcome column.
6. Creation of the ‘Class’ column

   The data is then saved to file including the new column as “`dataset_part_2.csv`”



Step 6

[In]

```
df['Class']=[i[1] for i in landing_class]
df[['Class']].head(8)
```

[Out]

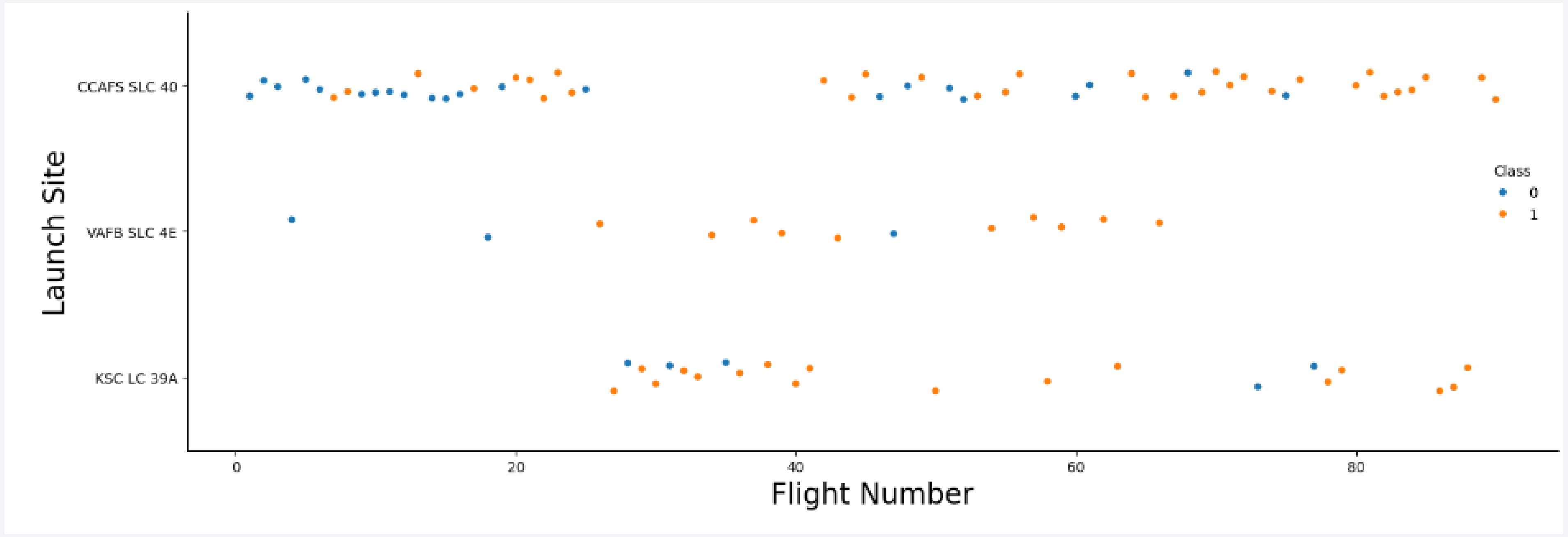
```
   Class
0      0
1      0
2      0
3      0
4      0
5      0
6      1
7      1
```

https://github.com/kayscaiwer/capstone/blob/main/capstone_prep_wrangling.ipynb

EDA with Data Visualization-1

The following charts explored correlations:

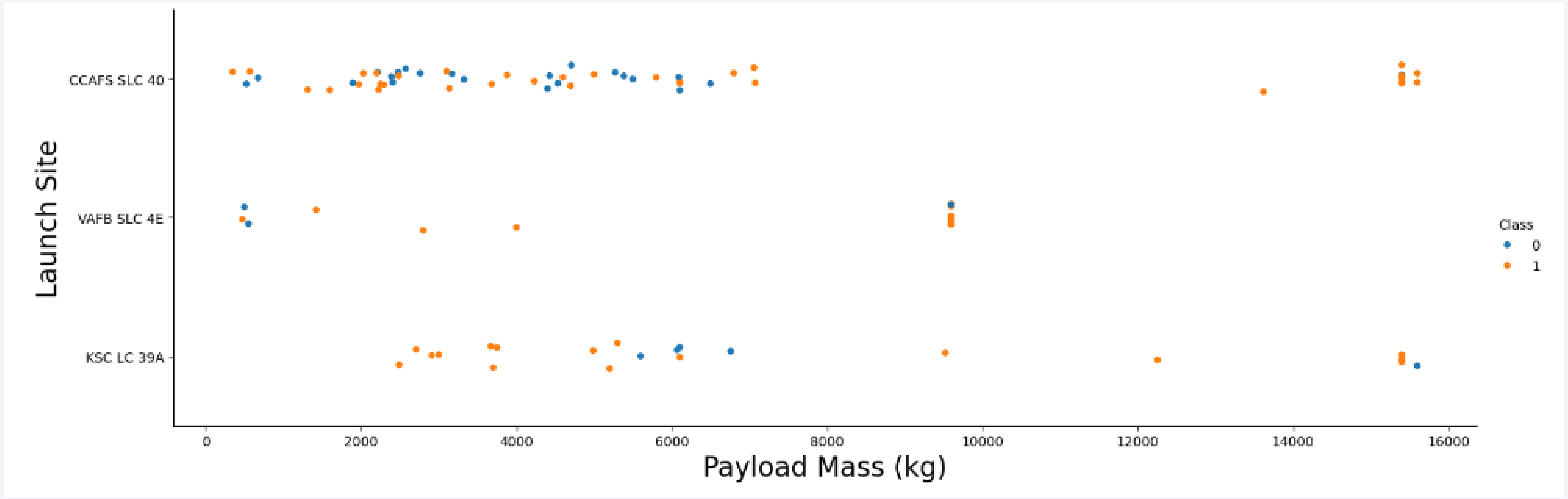
LaunchSite Vs Flight Number



https://github.com/kayscaiwer/capstone/blob/main/capstone_eda-viz.ipynb

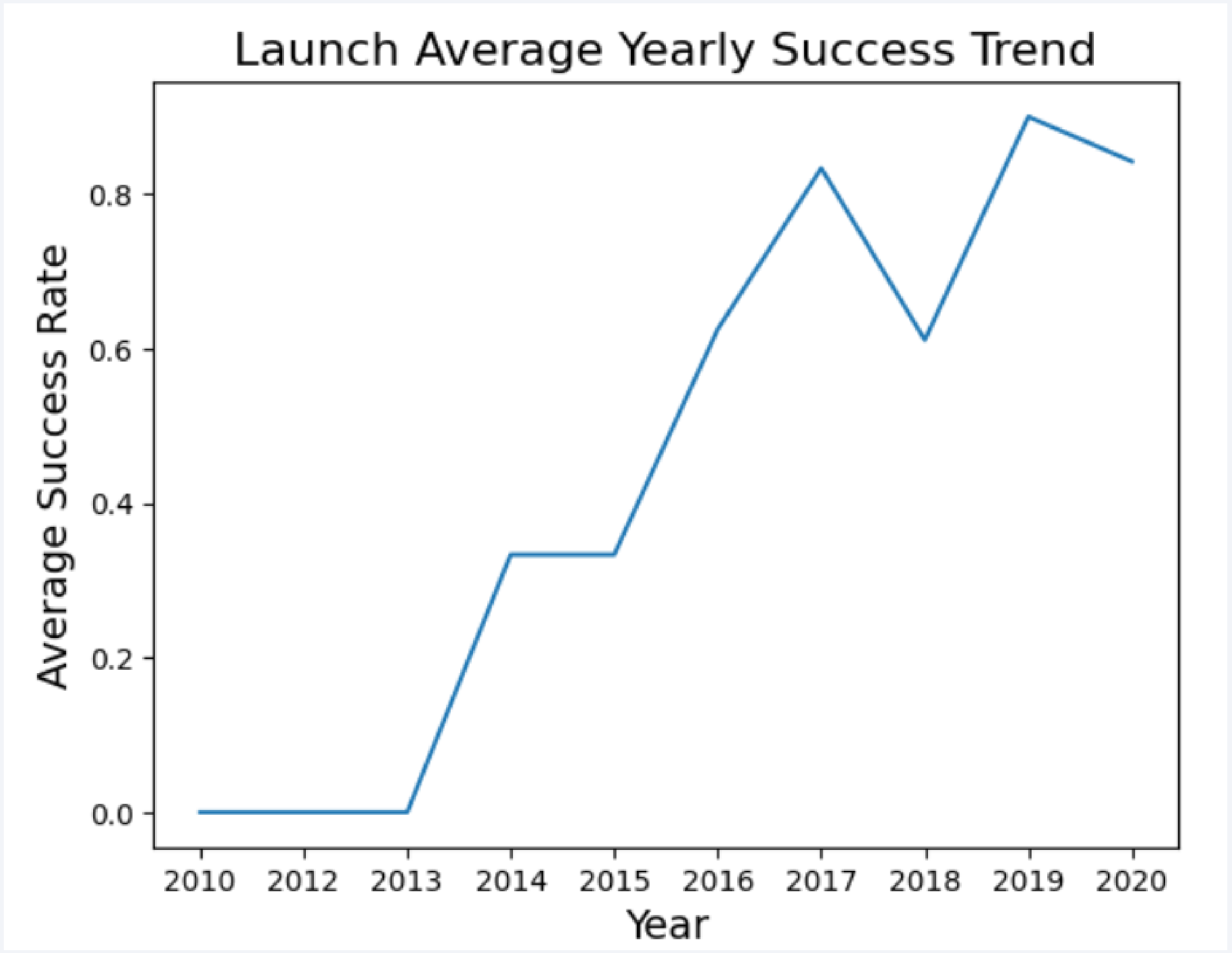
EDA with Data Visualization-2

LaunchSite Vs Payload Mass



https://github.com/kayscaiwer/capstone/blob/main/capstone_eda-viz.ipynb

EDA with Data Visualization-3



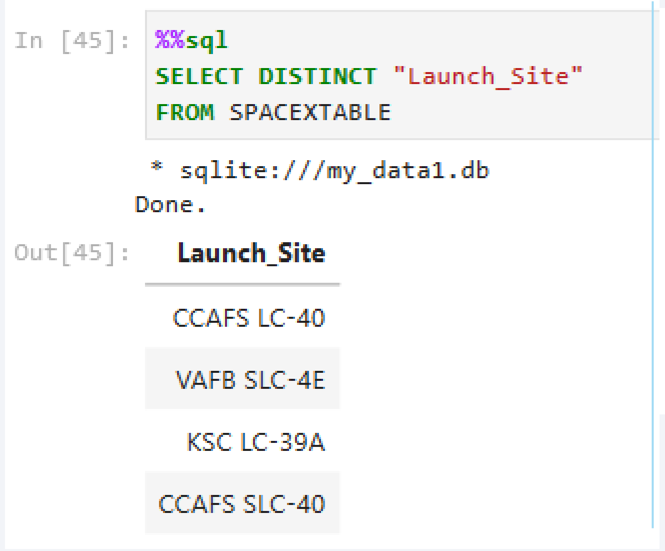
https://github.com/kayscaiwer/capstone/blob/main/capstone_eda-viz.ipynb

EDA with SQL

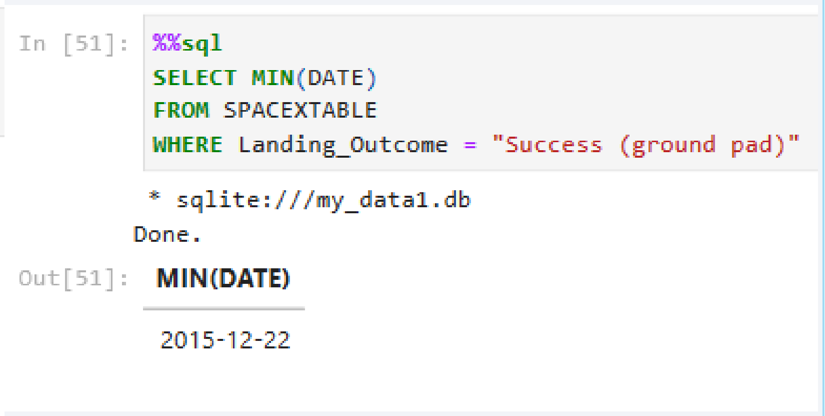
With the possibility of querying numerous features within the dataset, it can be summarized that the following query types were performed for various reasons to build up to the final form of the dataset.

- There were limit queries to extract minimums, maximums, earliest, and latest dates on various pieces of the data
- Queries that can be categorized as filters were also performed to get unique elements and ranges out of numeric data.
- Other queries were of the aggregation type to calculate averages, sums, and populate categories from alphanumeric columns.

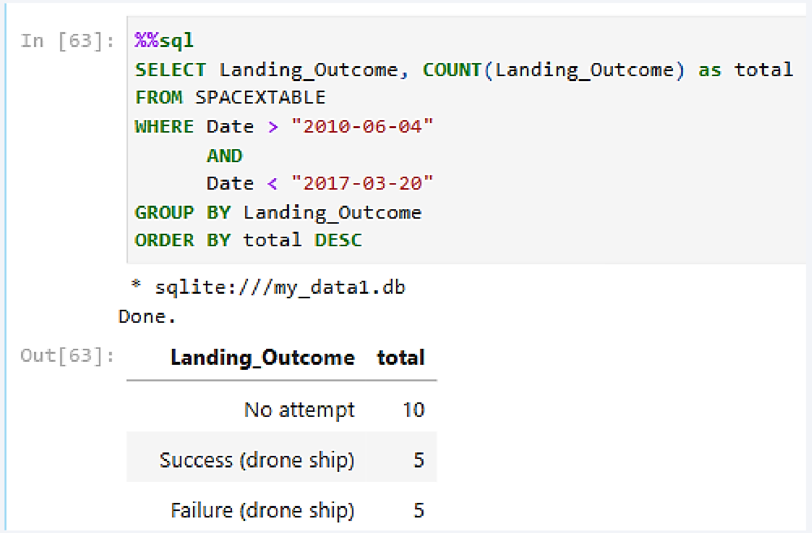
Sample syntax pieces are shown below:



```
In [45]: %%sql
         SELECT DISTINCT "Launch_Site"
         FROM SPACEXTABLE

 * sqlite:///my_data1.db
Done.

Out[45]:   Launch_Site
           CCAFS LC-40
           VAFB SLC-4E
           KSC LC-39A
          CCAFS SLC-40
```



```
In [51]: %%sql
         SELECT MIN(DATE)
         FROM SPACEXTABLE
         WHERE Landing_Outcome = "Success (ground pad)"

 * sqlite:///my_data1.db
Done.

Out[51]:  MIN(DATE)
          2015-12-22
```



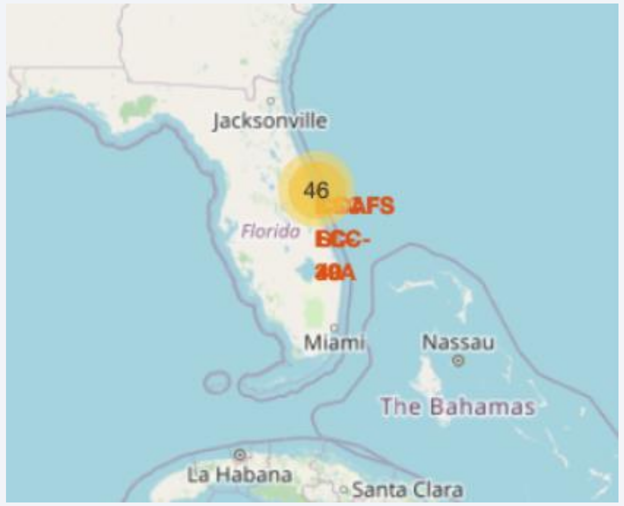
```
In [63]: %%sql
         SELECT Landing_Outcome, COUNT(Landing_Outcome) as total
         FROM SPACEXTABLE
         WHERE Date > "2010-06-04"
               AND
               Date < "2017-03-20"
         GROUP BY Landing_Outcome
         ORDER BY total DESC

 * sqlite:///my_data1.db
Done.

Out[63]:     Landing_Outcome   total
                  No attempt      10
         Success (drone ship)      5
         Failure (drone ship)      5
```

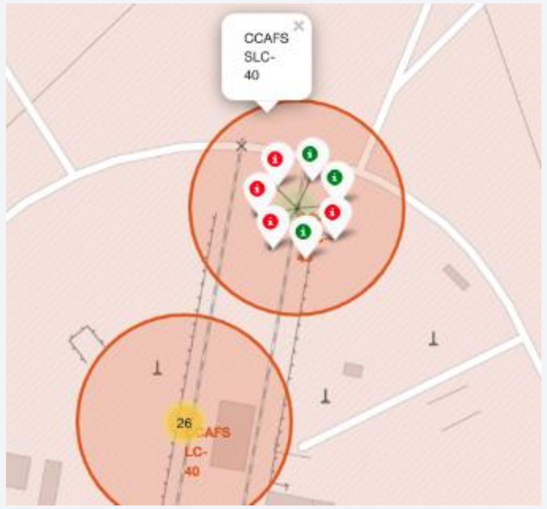
https://github.com/kayscaiwer/capstone/blob/main/capstone_eda_sql.ipynb

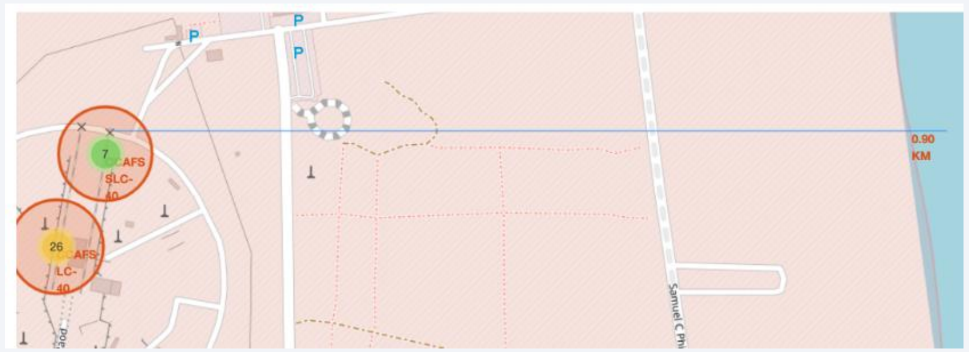
Build an Interactive Map with Folium

The starting point map in Folium is trivial but since we needed visualization of the launch sites, and the successful/failed outcome plots along with them, it was necessary to utilize various markers like Circle, Icon, lines and Clusters.

- **Circle** marker was utilized to indicate the various launch site locations on the map.
- **Line** was used to indicate distance between 2 key points on the folium map.
- **Icons** were utilized to change color according to launch outcome for easy visualization of the rough count on each of the outcomes for every launch site
- Above these basic markers, it was also applied to give the markers a **clustering** behavior that is practical to appear in a single marker or group of markers dynamically according to the concurrent zoom level as changed by the viewer.



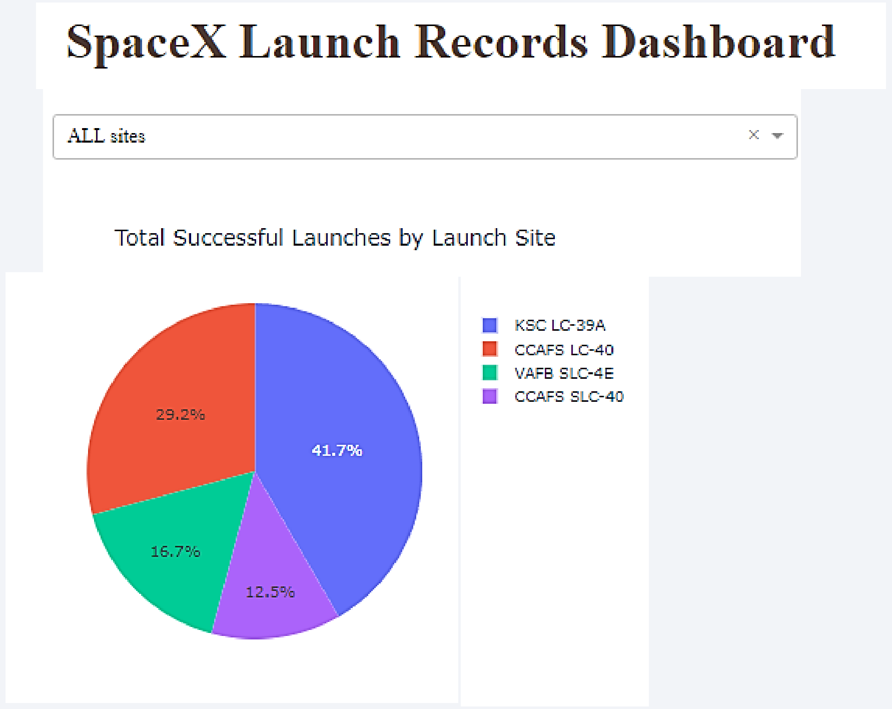


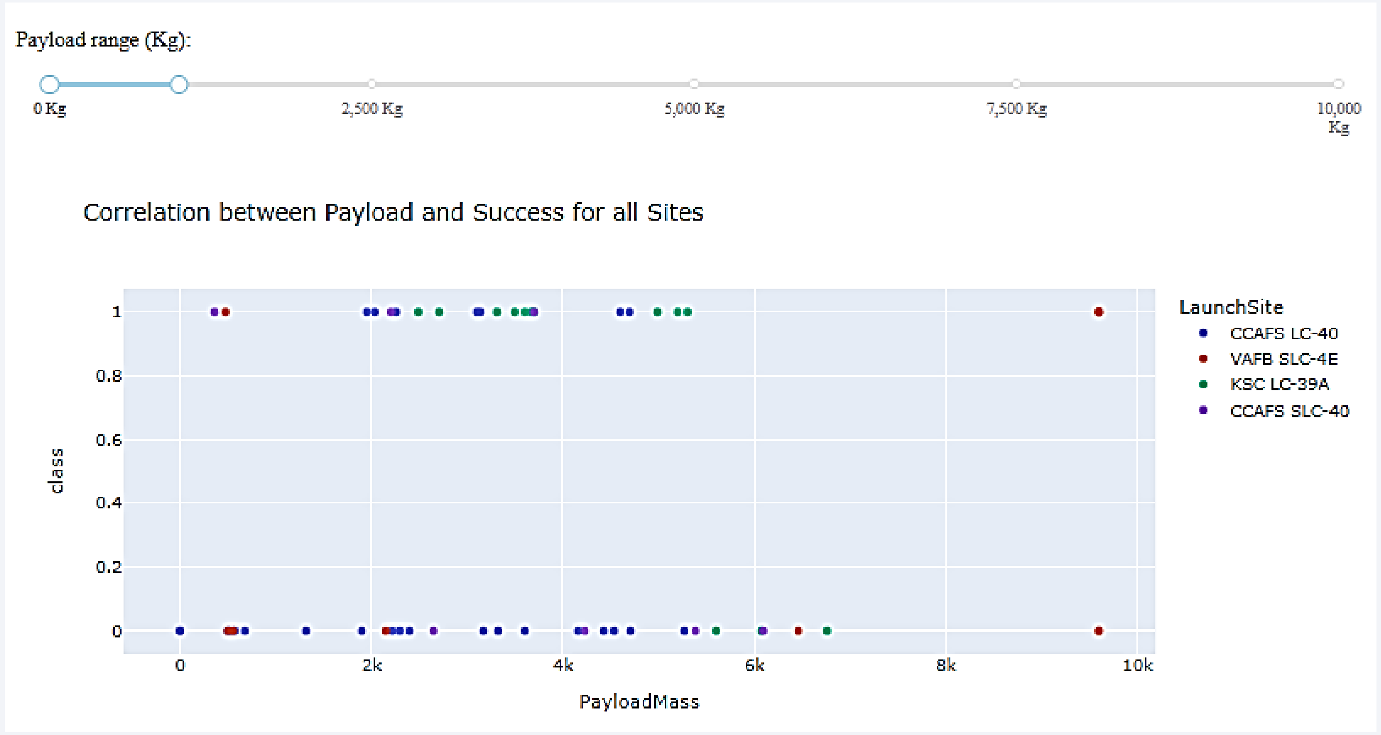


https://github.com/kayscaiwer/capstone/blob/main/capstone_eda_folium%20(3).ipynb

Build a Dashboard with Plotly Dash

In the dashboard were 2 plots; a scatter plot, and a pie chart to display launch success distribution per launch site with the option to display distribution for a single site when the ALL Sites option was not selected in the dropdown widget. The scatter plot displayed the correlation between Payload Mass against Launch success rate. It had the option to vary the range of payload mass included in the scatter plot using a slider widget.



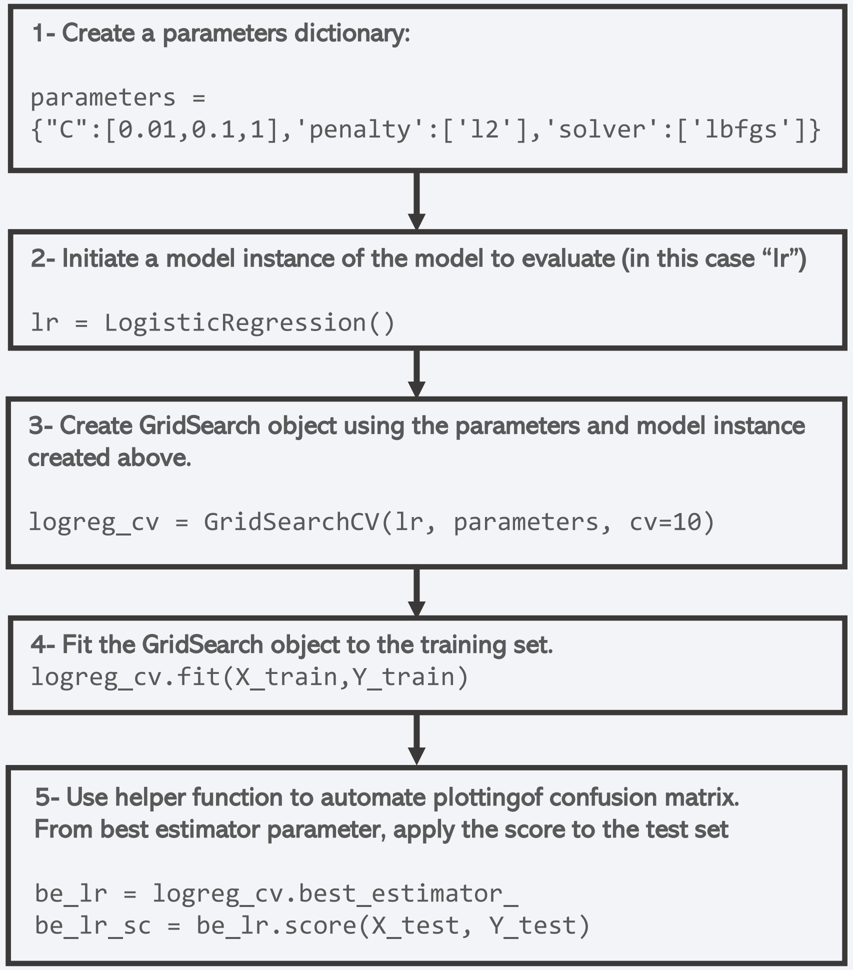


Predictive Analysis (Classification)

- Four types of models were considered for this predictive step, Logistic regression, Decision Tree classification, Support Vector Machinery, and K-nearest neighbors.

- The process for each was similar and used a `GridSearch` path technic. Below are the steps outlined. The data was split into the training and testing sets using the `train_test_split` function from scikitlearn's `model_selection` module namely, `X_train, Y_train, X_test, Y_test`.

- These sets are same that were then used in evaluating each of the models. The chart uses the case of the Logistic regression evaluation to illustrate some detail in the process.



1- Create a parameters dictionary:

```
parameters =
{"C":[0.01,0.1,1],'penalty':['l2'],'solver':['lbfgs']}
```

2- Initiate a model instance of the model to evaluate (in this case "lr")

```
lr = LogisticRegression()
```

3- Create GridSearch object using the parameters and model instance created above.

```
logreg_cv = GridSearchCV(lr, parameters, cv=10)
```

4- Fit the GridSearch object to the training set.

```
logreg_cv.fit(X_train,Y_train)
```

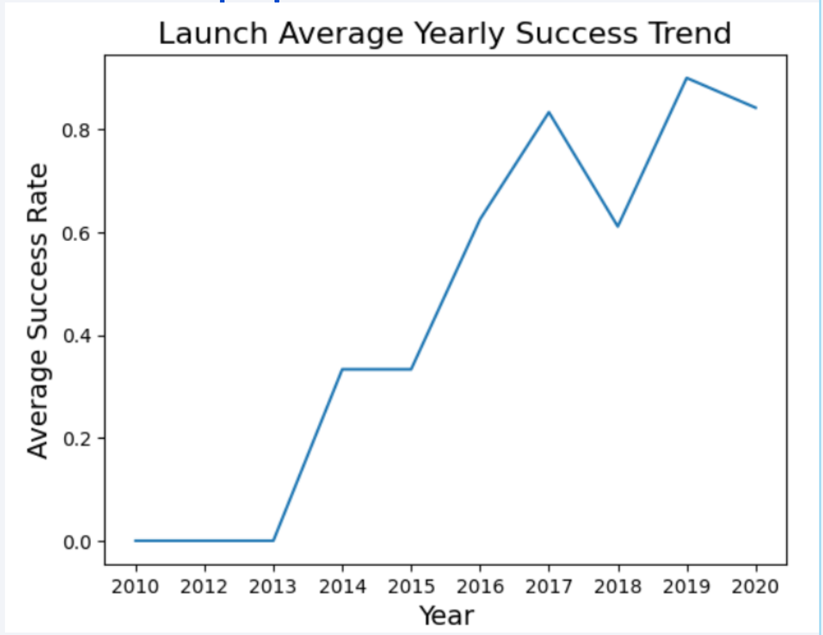
5- Use helper function to automate plottingof confusion matrix. From best estimator parameter, apply the score to the test set

```
be_lr = logreg_cv.best_estimator_
be_lr_sc = be_lr.score(X_test, Y_test)
```

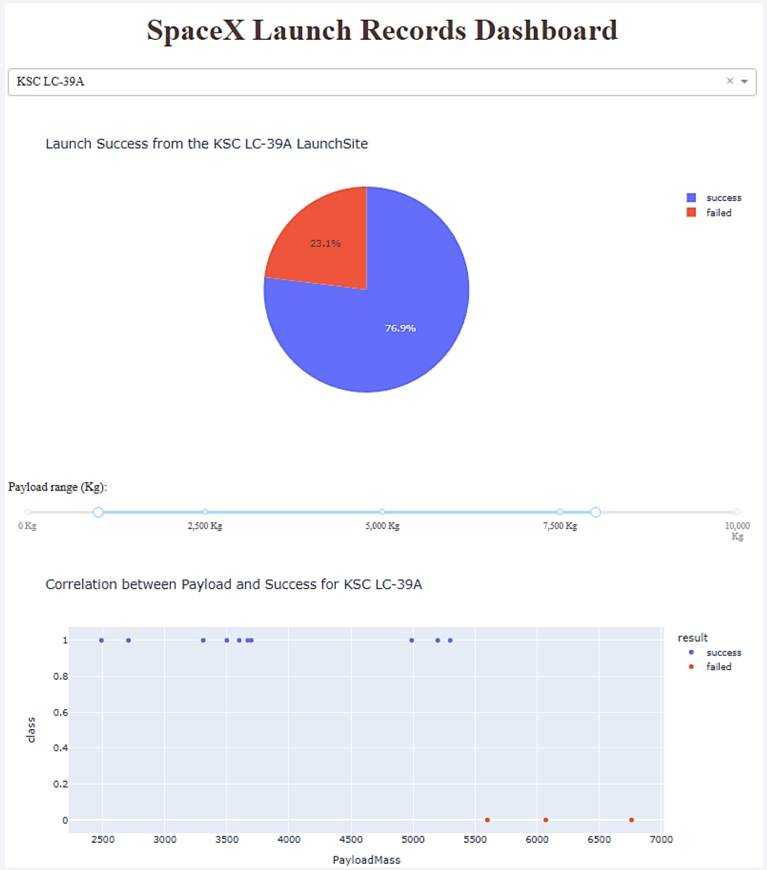
`https://github.com/kayscaiwer/capstone/blob/main/capstone_ml_prediction.ipynb`

Results

- Exploratory data



- Interactive analytics demo


SpaceX Launch Records Dashboard

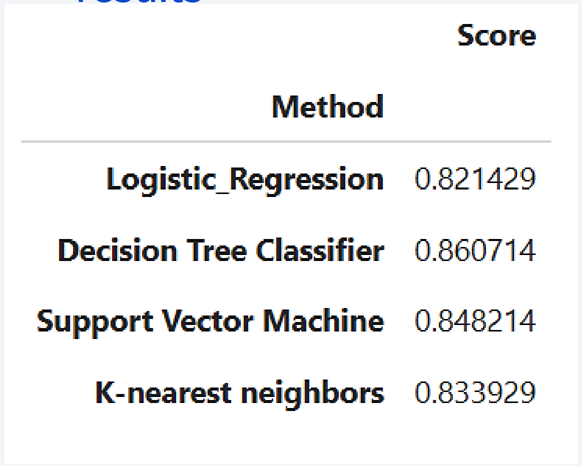
KSC LC-39A

Launch Success from the KSC LC-39A LaunchSite

Payload range (Kg):

Correlation between Payload and Success for KSC LC-39A

- Predictive analysis results

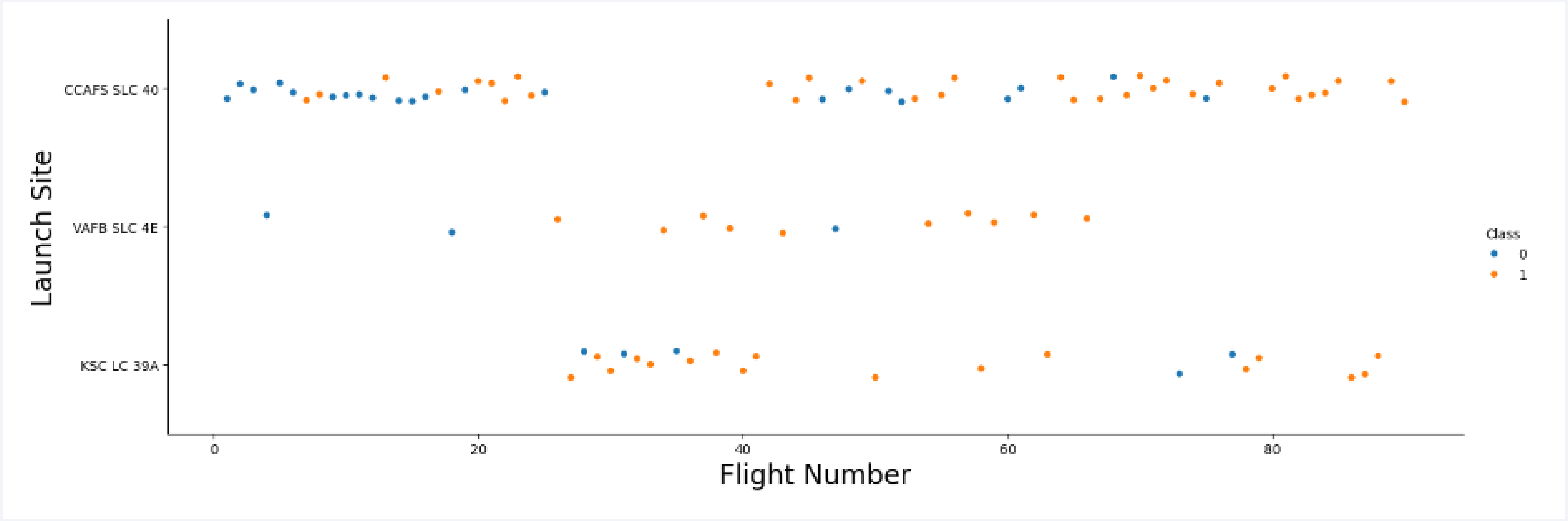


Method	Score
Logistic_Regression	0.821429
Decision Tree Classifier	0.860714
Support Vector Machine	0.848214
K-nearest neighbors	0.833929

Section 2

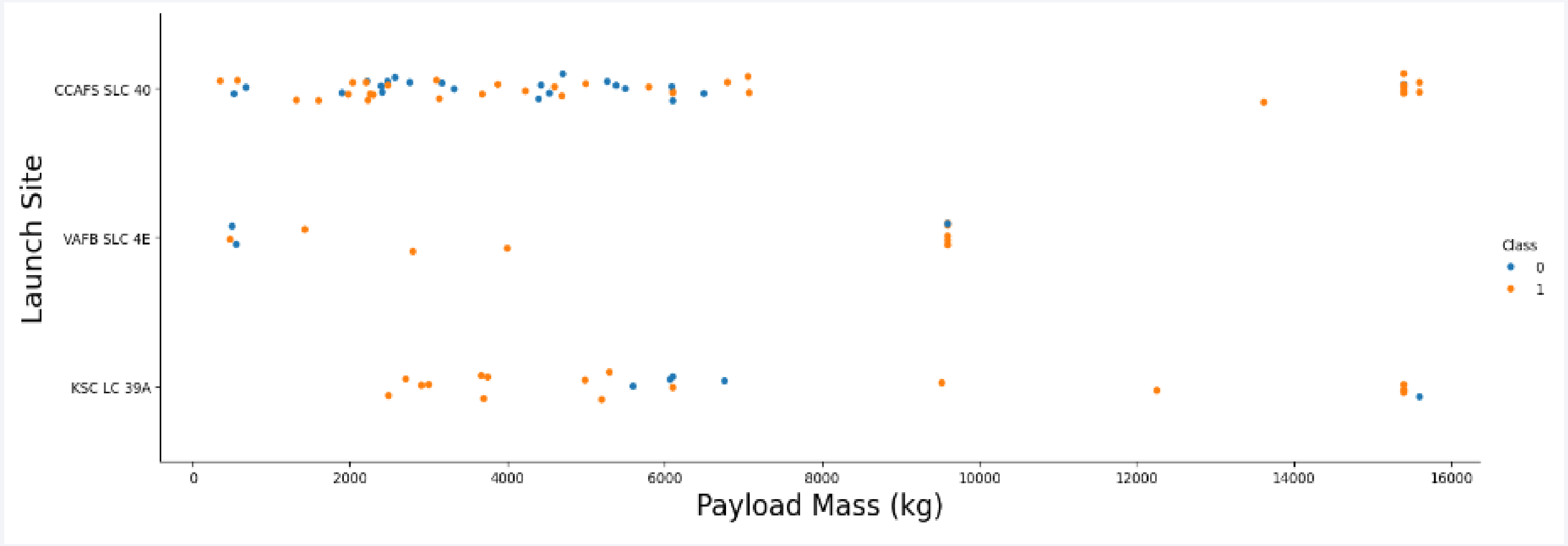
Insights drawn from EDA

Flight Number vs. Launch Site



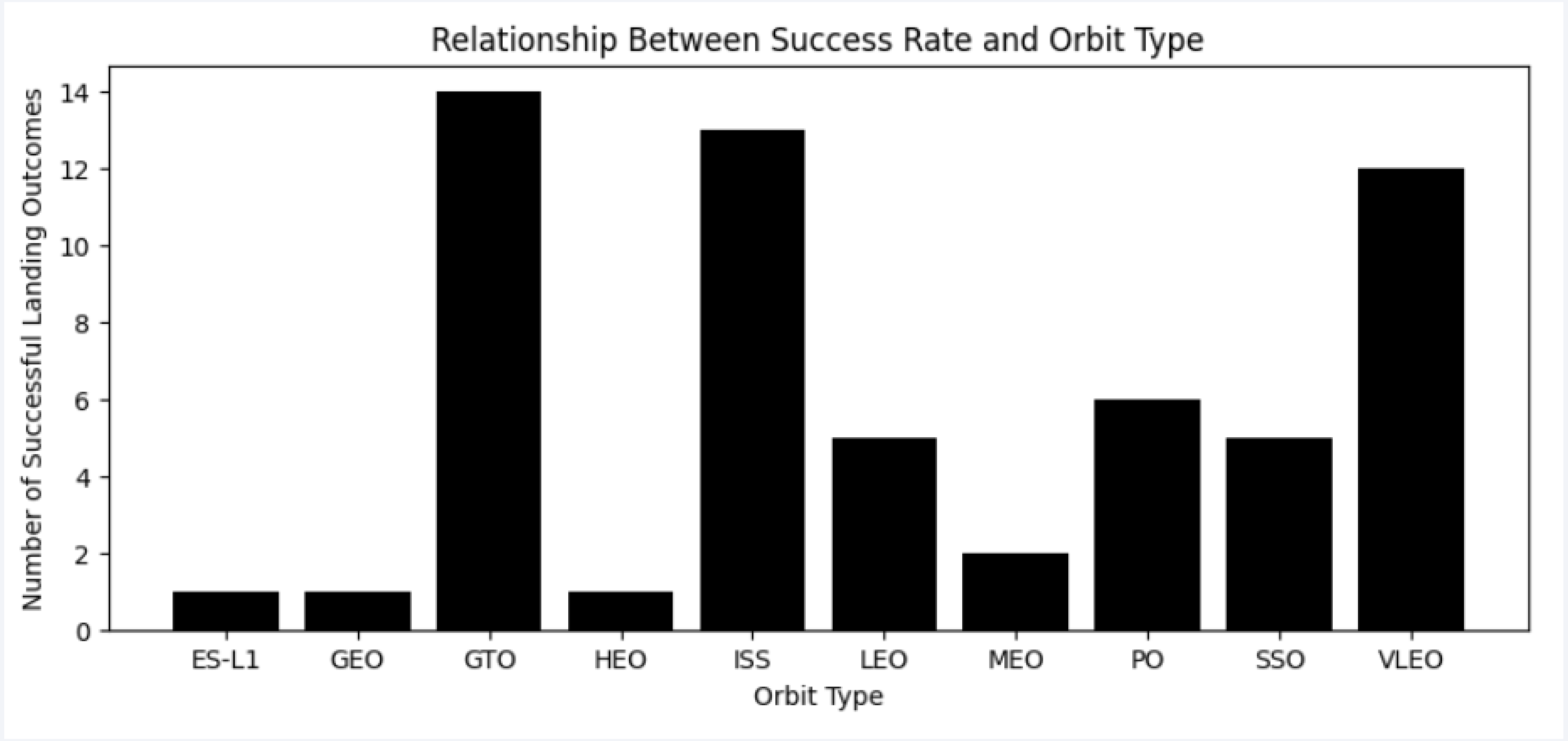
- The Class legend represents landing failure with 0 and Landing success with a 1
- Launch site CCAFS SLC 40 has hosted the most launches than the other 2 sites and thus has more of the failures as well as successes

Payload vs. Launch Site



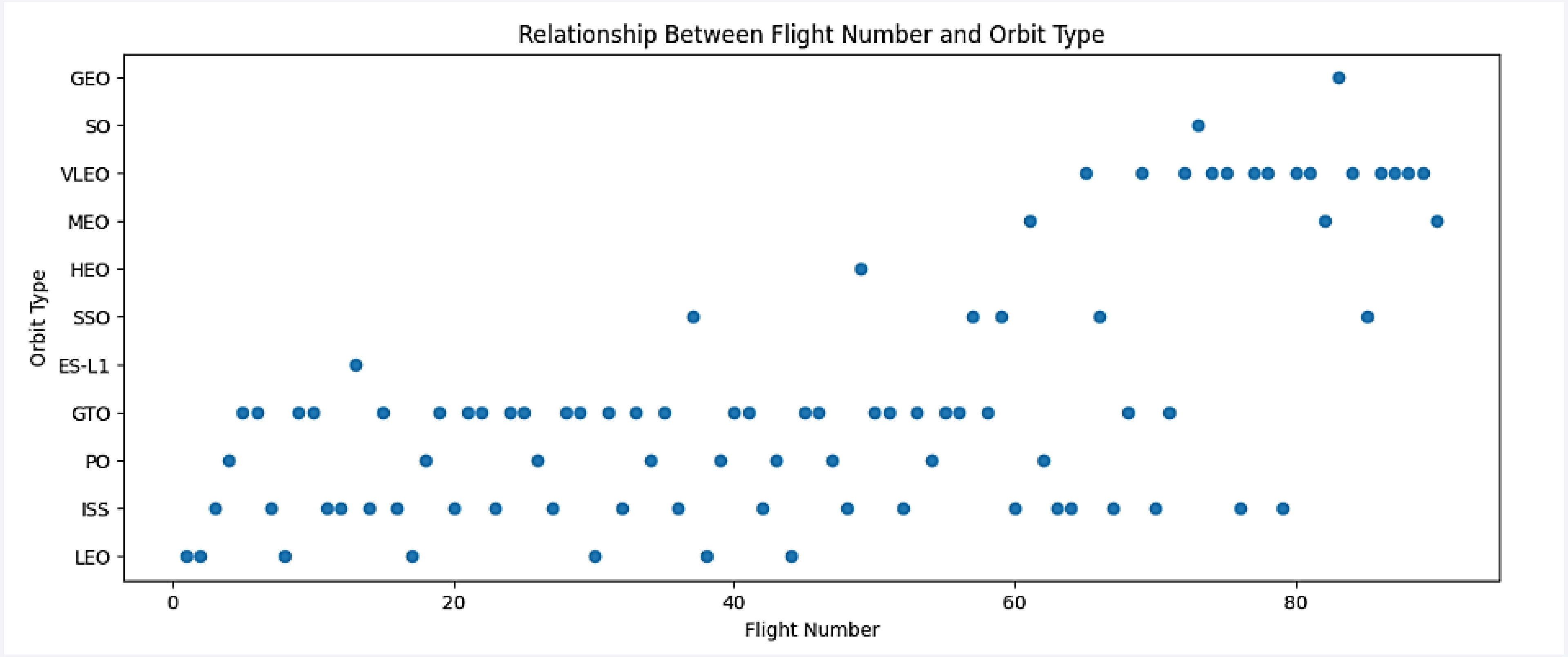
- Notable on this plot is the fact that there are no launches with payload mass above 10,000kg coming out of site VAFB SLC 4E

Success Rate vs. Orbit Type



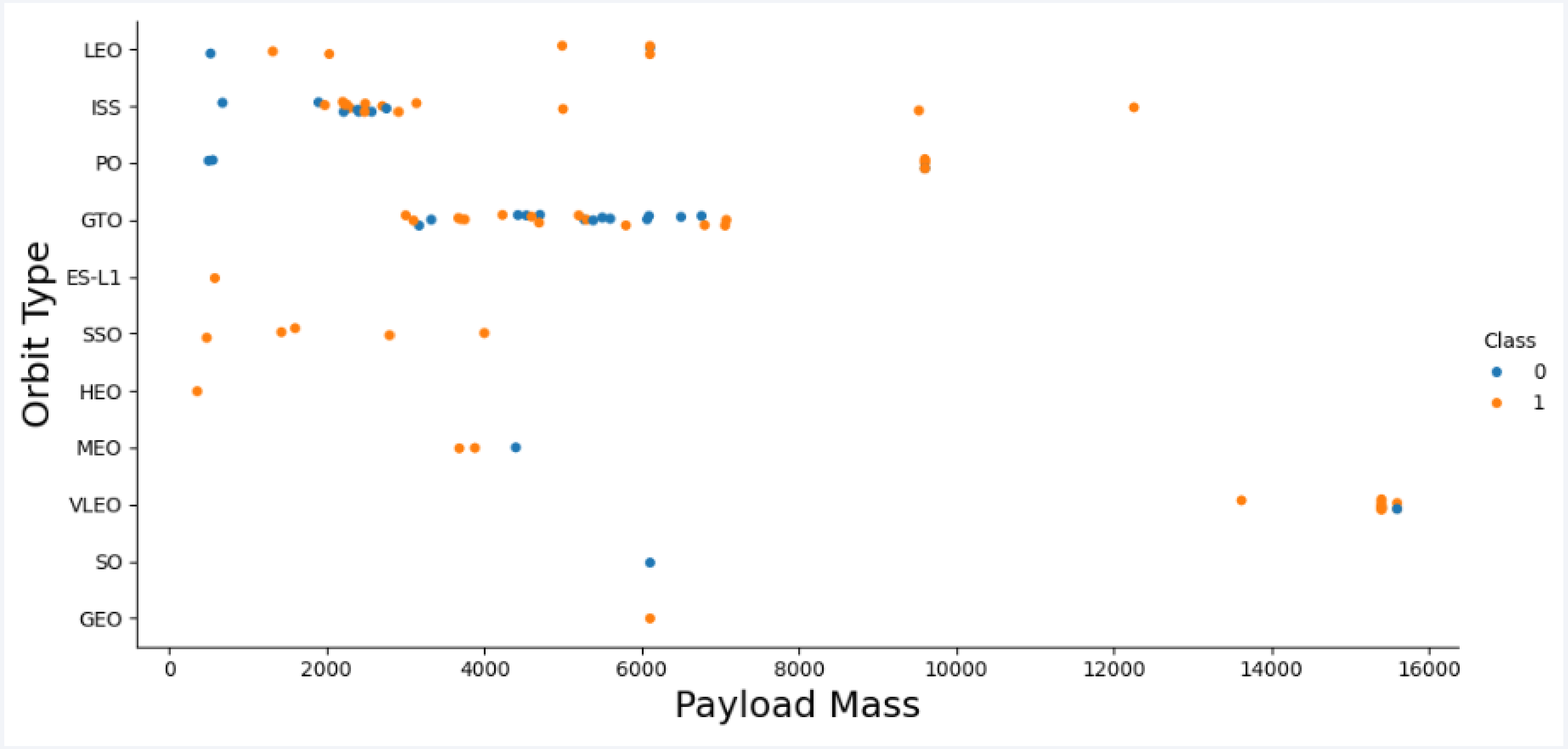
- On this one, GTO, ISS, and VLEO stood out with some of the highest success rates.

Flight Number vs. Orbit Type



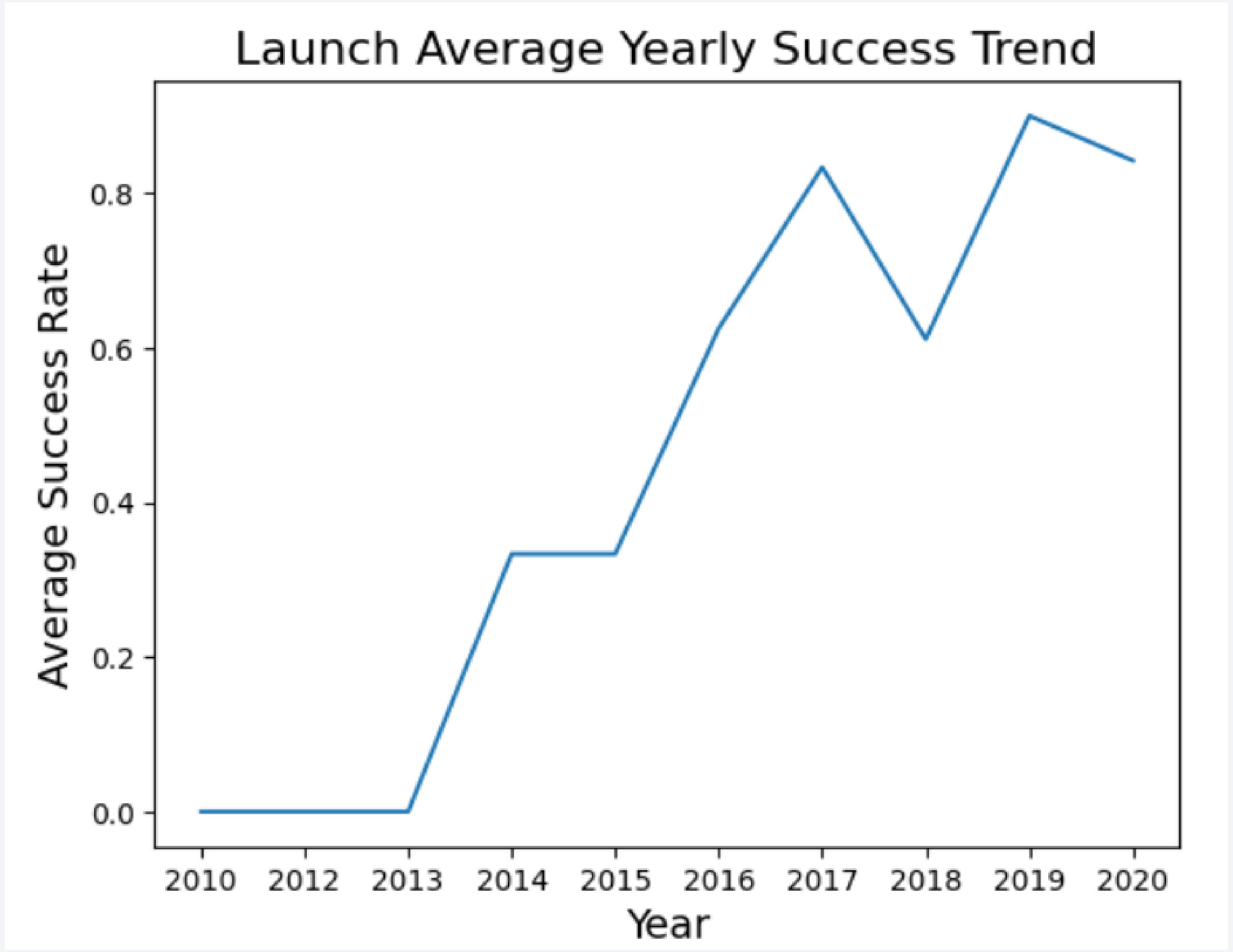
- It appears the GEO orbit became more significant as time went on since it shows as a target orbit more for later flight numbers.

Payload vs. Orbit Type



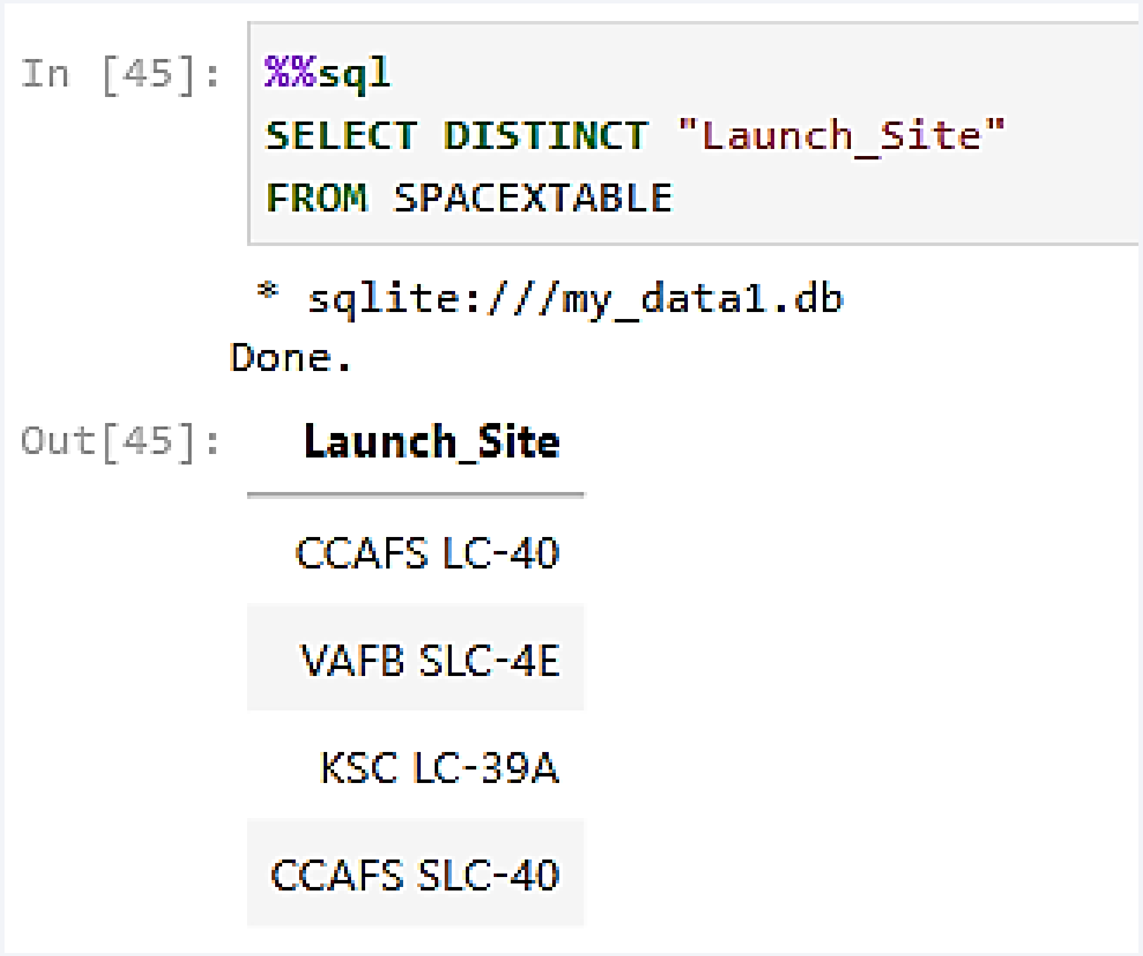
- Launch failures phase out as payload mass increases

Launch Success Yearly Trend



- Average success rate increases with time

All Launch Site Names - SQL



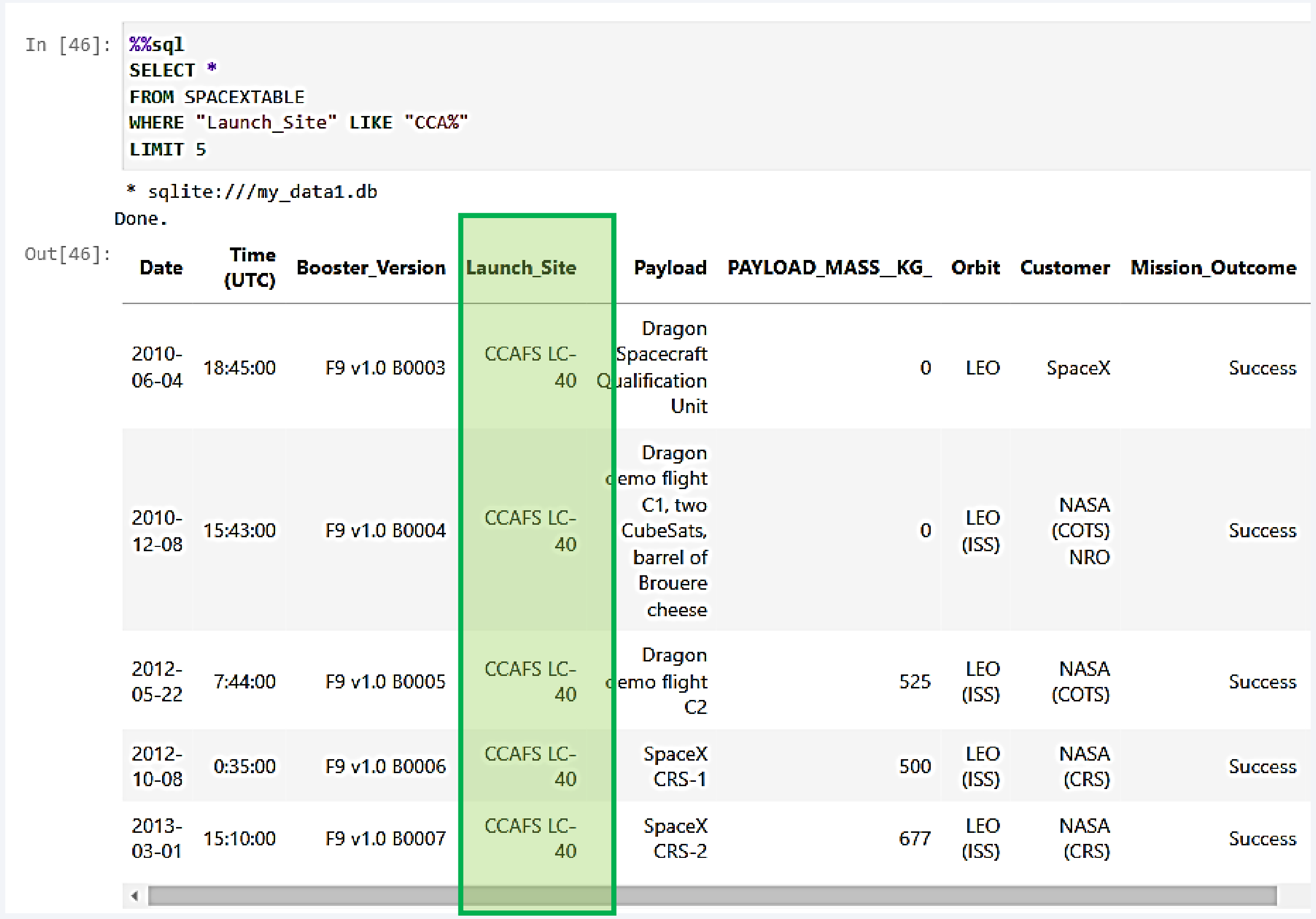
```
In [45]: %%sql
         SELECT DISTINCT "Launch_Site"
         FROM SPACEXTABLE

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

Out[45]:

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

- To find the unique names of the launch sites, SQL was applied and the keyword in the query is DISTINCT.
- The syntax calls for the return of unique items from the specified table in the database indicated below as “my_data1.db”.

Launch Site Names Begin with 'CCA'



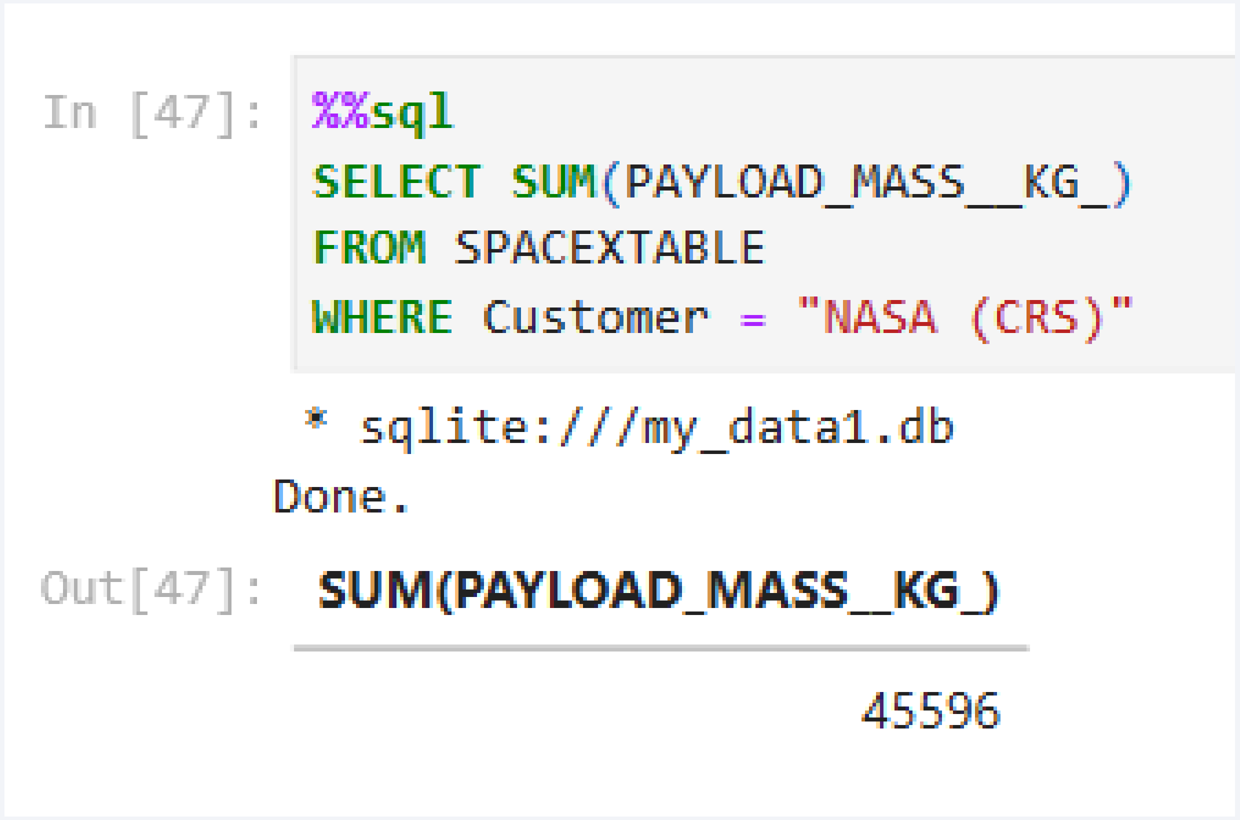
```
In [46]: %%sql
         SELECT *
         FROM SPACEXTABLE
         WHERE "Launch_Site" LIKE "CCA%"
         LIMIT 5

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

Out[46]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success
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
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success

- In this query, we use 2 keywords, WHERE and LIKE to qualify by a condition and also filter on a text field.
- The condition specifies the field we are searching on using WHERE and the action is a text match to the 'CCA' phrase preceeded with no other characters but can be followed by any other characters hence the wild card '%' key character.
- To get exactly 5 returned records and not more, the keyword LIMIT is applied along with an integer, 5.

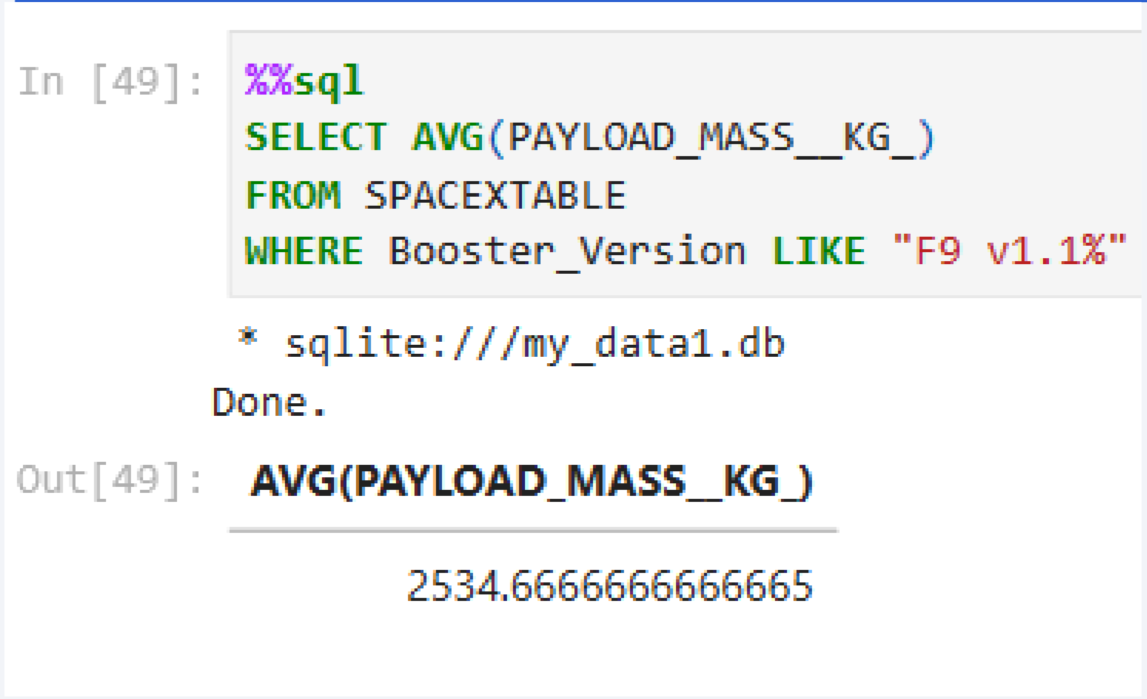
Total Payload Mass



```
In [47]: %%sql
         SELECT SUM(PAYLOAD_MASS__KG_)
         FROM SPACEXTABLE
         WHERE Customer = "NASA (CRS)"

          * sqlite:///my_data1.db
         Done.

Out[47]:  SUM(PAYLOAD_MASS__KG_)
          ----------------------
                           45596
```

- Keywords in this case are WHERE and SUM as well as the name of the customer. This has to be known but can alternatively be filtered out if it was necessary.
- The result is a single item since aggregation by summing was applied.

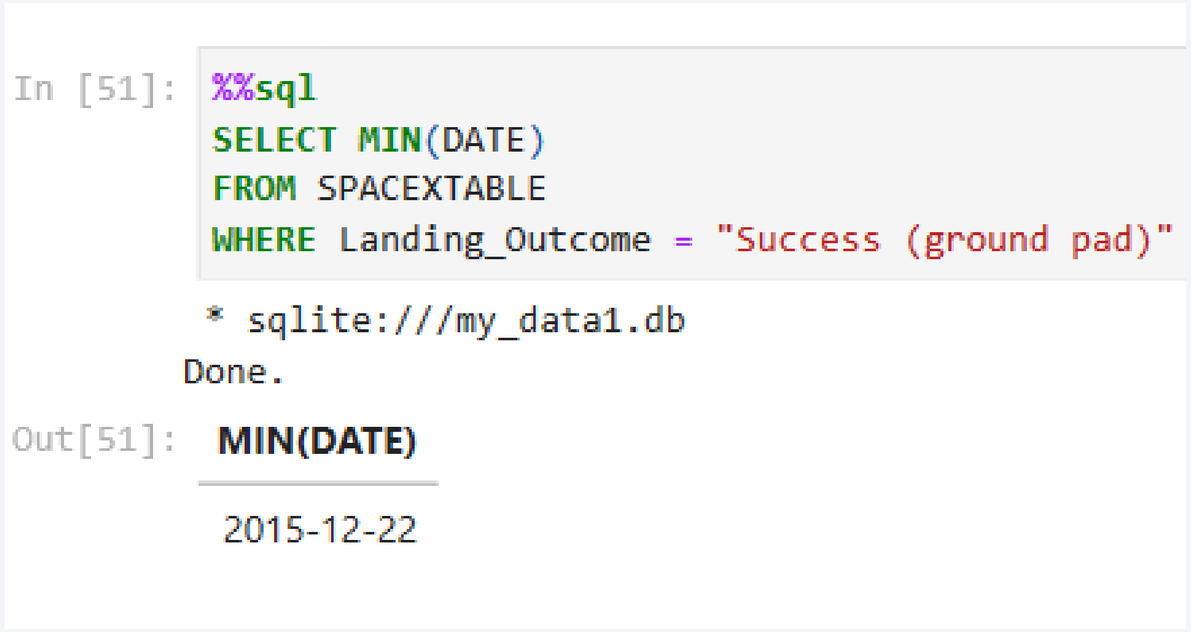
Average Payload Mass by F9 v1.1



```
In [49]: %%sql
         SELECT AVG(PAYLOAD_MASS__KG_)
         FROM SPACEXTABLE
         WHERE Booster_Version LIKE "F9 v1.1%"

 * sqlite:///my_data1.db
Done.

Out[49]: AVG(PAYLOAD_MASS__KG_)
         2534.6666666666665
```

- Keywords in this case are WHERE, and LIKE as well as the name of the target element in the booster version field. This has to be known but can alternatively be filtered out if it was necessary.
- The WHERE clause points to a field and the LIKE keyword is used to match word phrases within the field's records to return the target items.
- The returned records are then aggregated by the average function using the keyword AVG

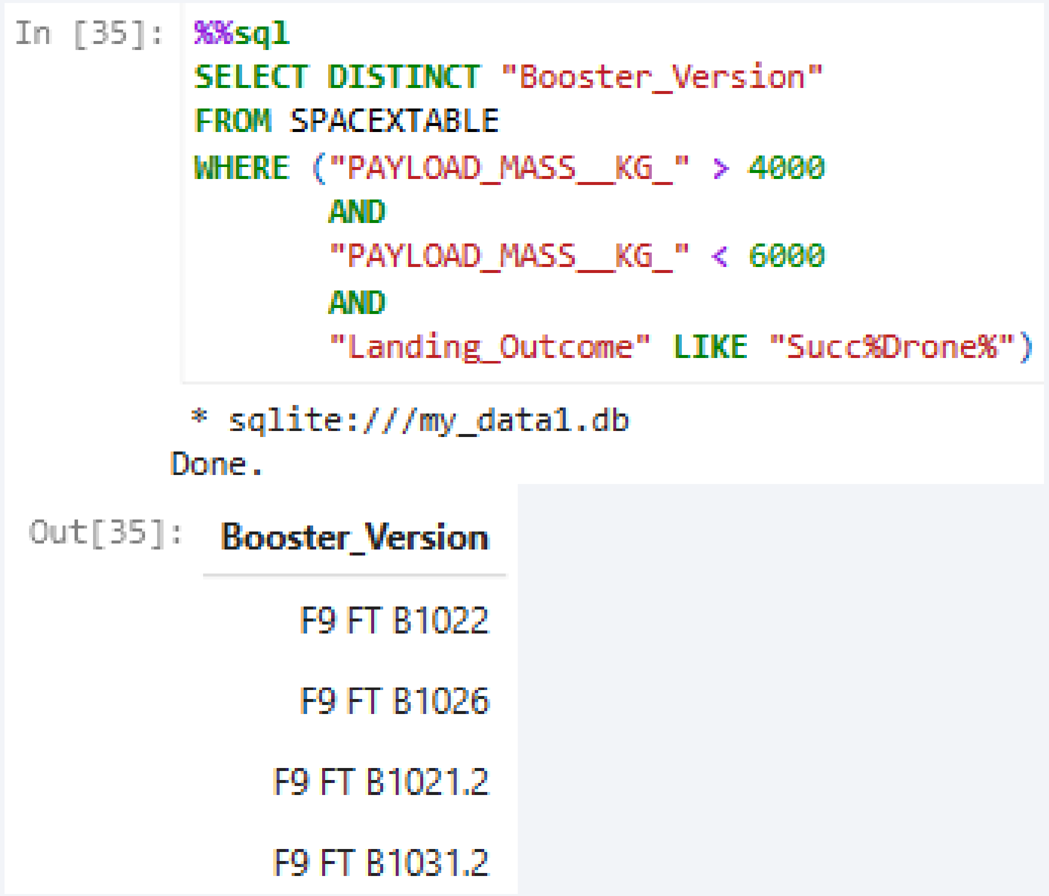
First Successful Ground Landing Date



```
In [51]: %%sql
         SELECT MIN(DATE)
         FROM SPACEXTABLE
         WHERE Landing_Outcome = "Success (ground pad)"

          * sqlite:///my_data1.db
         Done.

Out[51]: MIN(DATE)
         2015-12-22
```

- The key technique in this query is the application of MIN to a date therefore locating the earliest date that meets the condition from the WHERE clause.
- The landing outcome was specified by name to be distinct otherwise it would have had to be inserted by nesting a filter query to create a query-in-a-query.

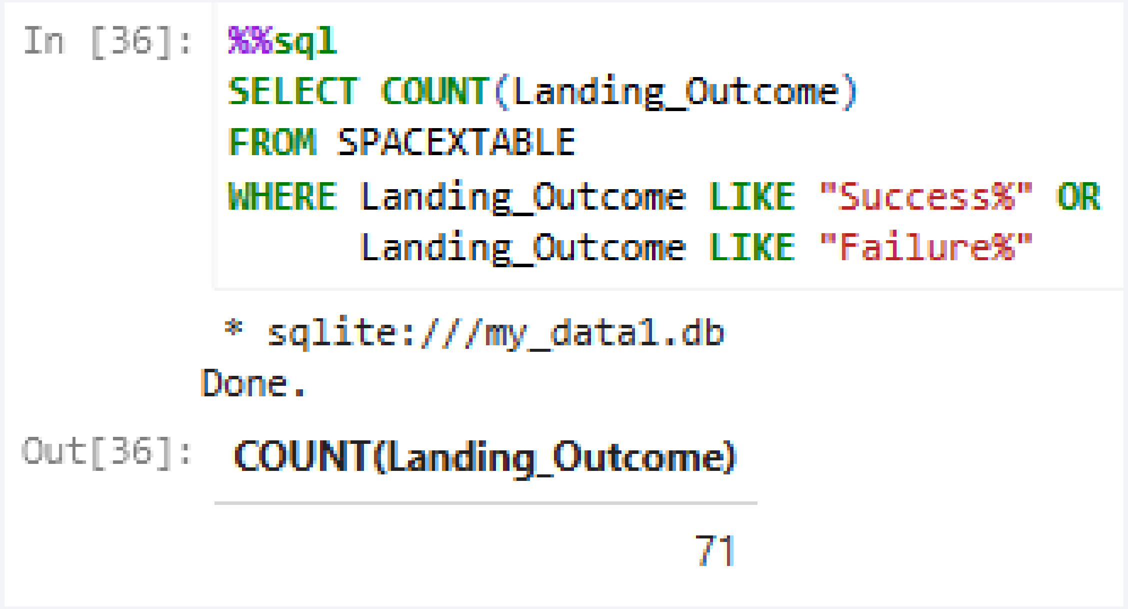
Successful Drone Ship Landing with Payload between 4000 and 6000



```
In [35]: %%sql
         SELECT DISTINCT "Booster_Version"
         FROM SPACEXTABLE
         WHERE ("PAYLOAD_MASS__KG_" > 4000
                AND
                "PAYLOAD_MASS__KG_" < 6000
                AND
                "Landing_Outcome" LIKE "Succ%Drone%")

          * sqlite:///my_data1.db
         Done.

Out[35]: Booster_Version
              F9 FT B1022
              F9 FT B1026
            F9 FT B1021.2
            F9 FT B1031.2
```

- Unlike the queries above, this introduces a condition on a range providing the lower and upper limits.
- The rest is similar to previous queries applying the DISTINCT, WHERE, and LIKE clauses and it has been indicated above how they function.

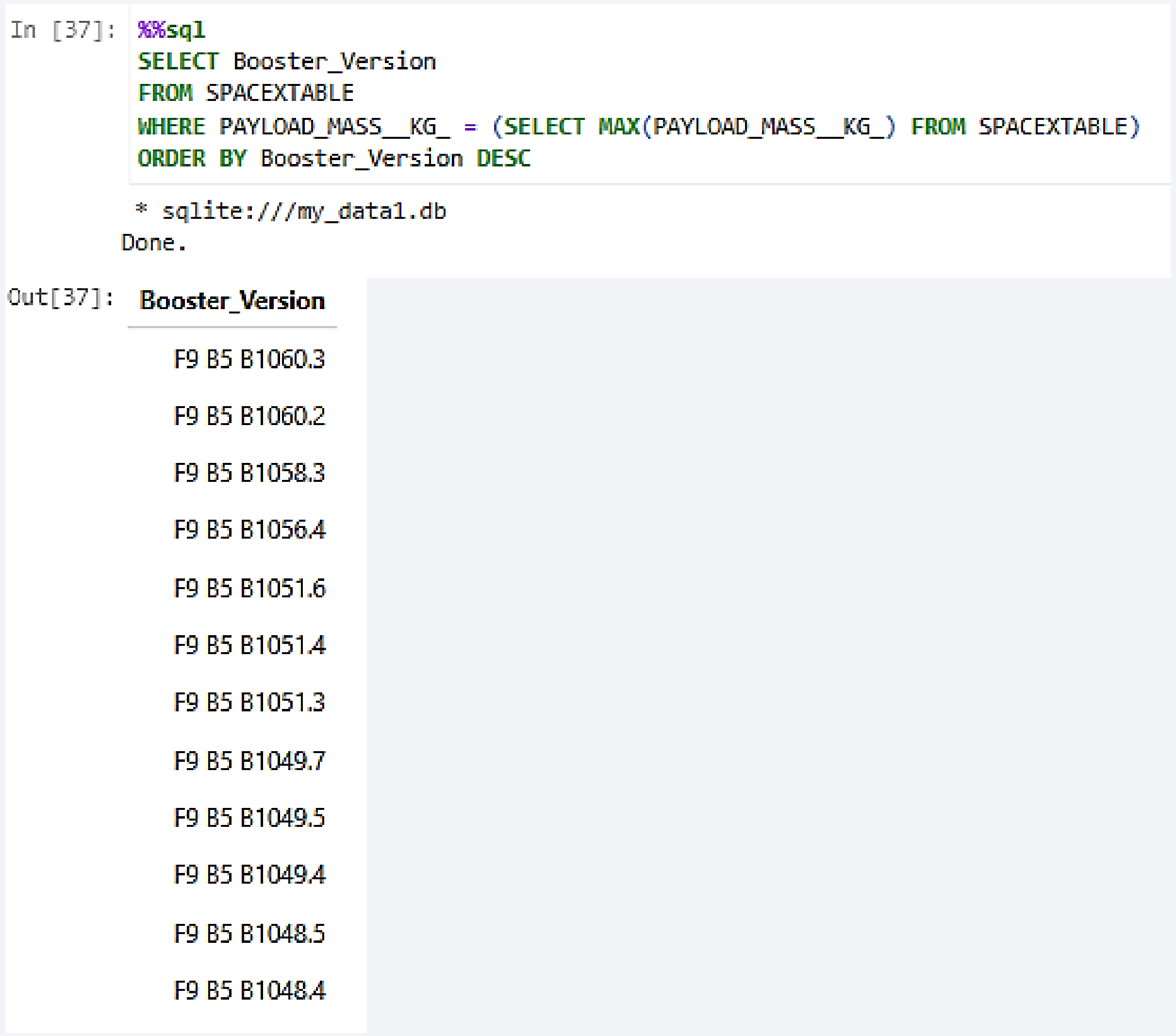
Total Number of Successful and Failure Mission Outcomes



```
In [36]: %%sql
         SELECT COUNT(Landing_Outcome)
         FROM SPACEXTABLE
         WHERE Landing_Outcome LIKE "Success%" OR
               Landing_Outcome LIKE "Failure%"

          * sqlite:///my_data1.db
         Done.

Out[36]: COUNT(Landing_Outcome)
                             71
```

- Similar to the queries already discussed, this one also applies the keywords WHERE, and LIKE to filter and the results returned are aggregated by the COUNT function tum all to a total of 71
- Again the LIKE keyword is used to select phrase matches within an element that qualifies from the target field, in this case "Landing_Outcome".
- The use of OR to return an element matching either phrase is just Boolean logic and works as might be intuitively expected.

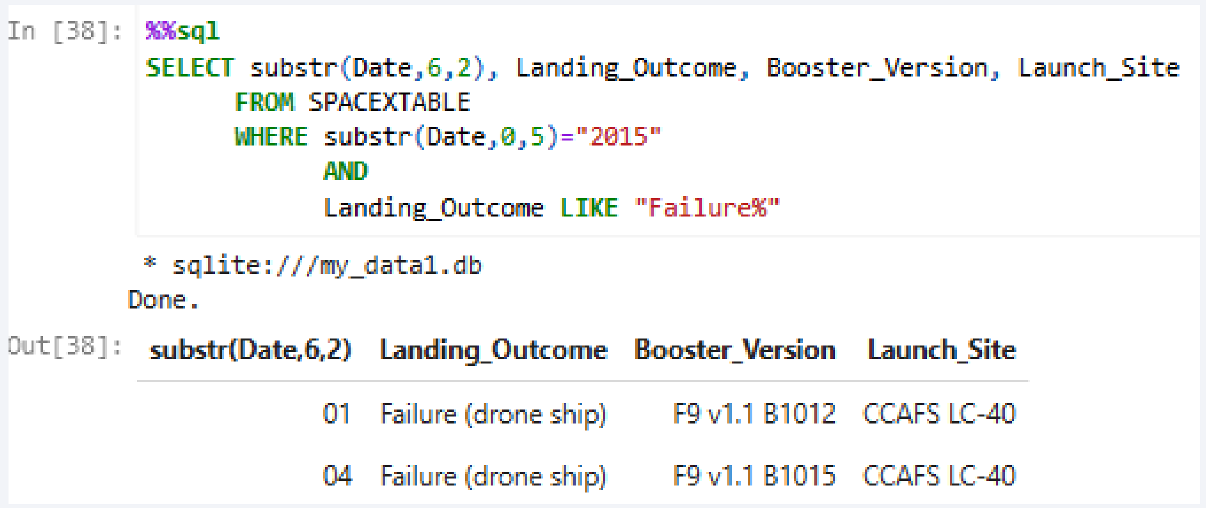
Boosters Carried Maximum Payload



```
In [37]: %%sql
         SELECT Booster_Version
         FROM SPACEXTABLE
         WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
         ORDER BY Booster_Version DESC

          * sqlite:///my_data1.db
         Done.

Out[37]: Booster_Version
          F9 B5 B1060.3
          F9 B5 B1060.2
          F9 B5 B1058.3
          F9 B5 B1056.4
          F9 B5 B1051.6
          F9 B5 B1051.4
          F9 B5 B1051.3
          F9 B5 B1049.7
          F9 B5 B1049.5
          F9 B5 B1049.4
          F9 B5 B1048.5
          F9 B5 B1048.4
```

- For this query, it was necessary to filter on a result from another query in a nest format. This is a great example of a query-in-query situation.
- Additionally, a sort was also applied to the final result indicating direction as descending using the DESC keyword.

2015 Launch Records



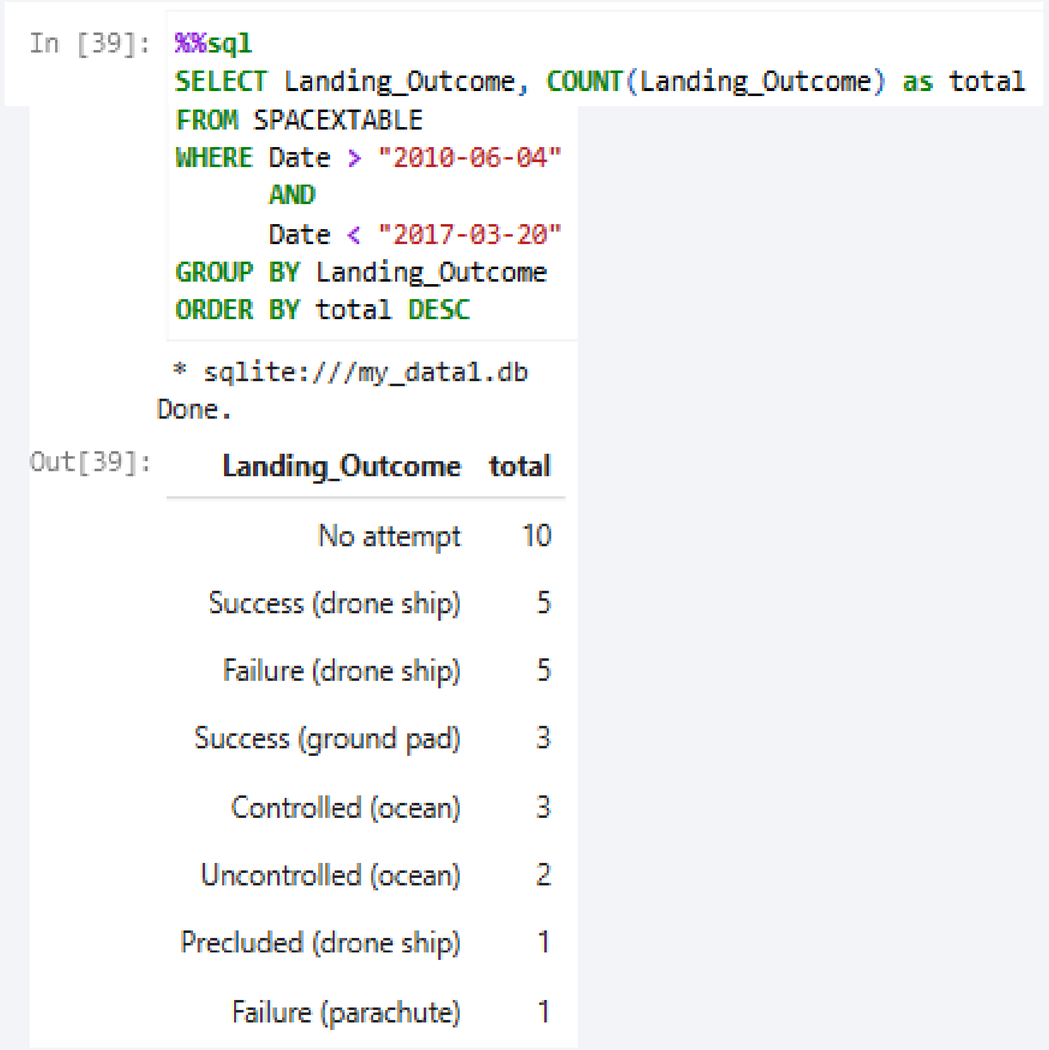
```
In [38]: %%sql
         SELECT substr(Date,6,2), Landing_Outcome, Booster_Version, Launch_Site
               FROM SPACEXTABLE
               WHERE substr(Date,0,5)="2015"
                     AND
                     Landing_Outcome LIKE "Failure%"

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

Out[38]:

substr(Date,6,2)	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

- This query result lists the failed landing outcomes on drone ship, their booster versions, and launch site names from the year 2015.
- For the first instance in this record, the query calls for more than one field. All other keywords have been discussed in the previous slides but note should be taken that the subtr() function handles extraction of the month from a date.

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



```
In [39]: %%sql
         SELECT Landing_Outcome, COUNT(Landing_Outcome) as total
         FROM SPACEXTABLE
         WHERE Date > "2010-06-04"
               AND
               Date < "2017-03-20"
         GROUP BY Landing_Outcome
         ORDER BY total DESC

          * sqlite:///my_data1.db
         Done.
Out[39]:
```

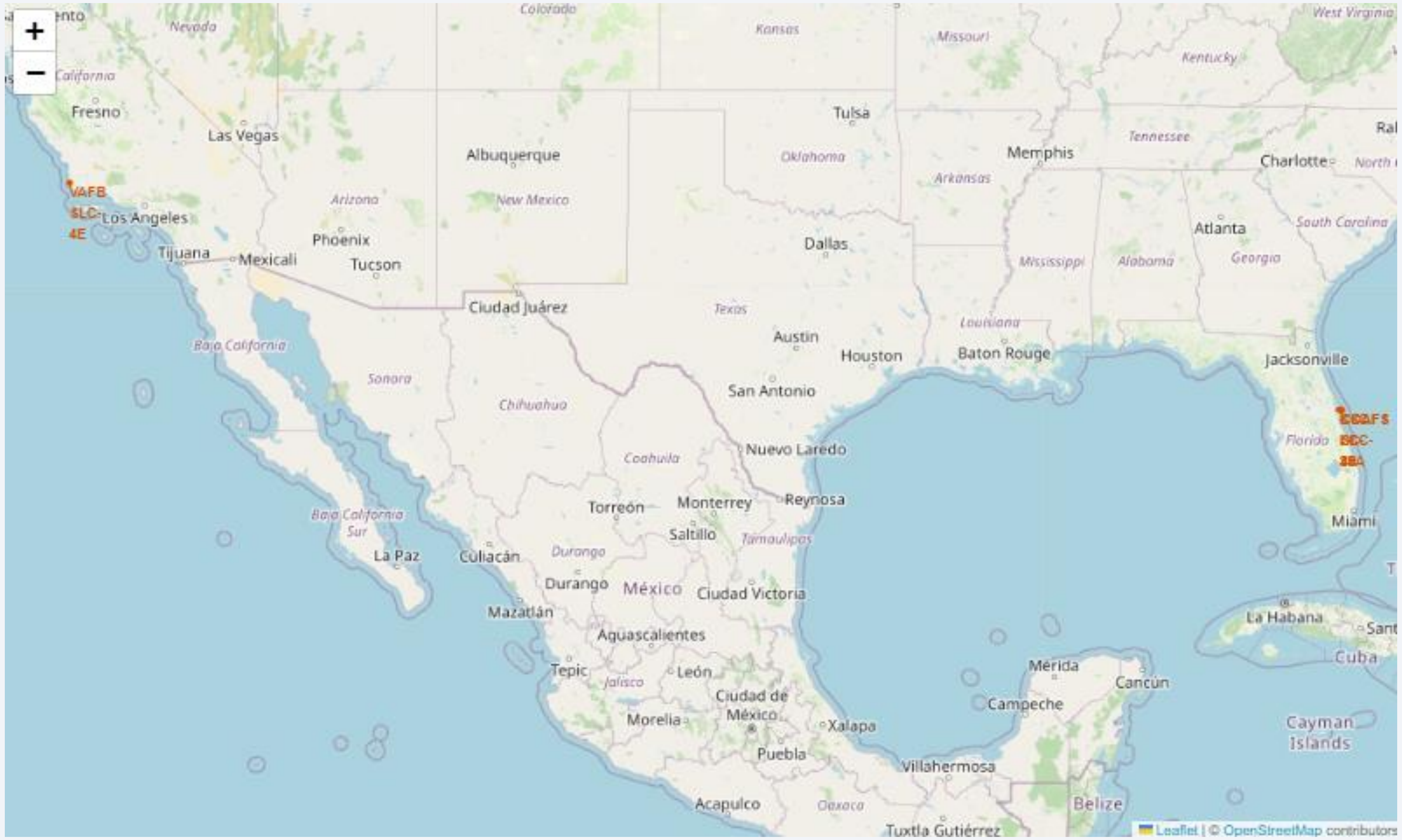
Landing_Outcome	total
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

- Booster landings started late 2015 which is more than halfway between 2010 and 2017. As such, it is clear there were more none-attempts at landing than attempted landings.
- This query combines a good number of techniques to pull off this ranking. There is query nesting, range conditioning, aggregation, and sorting. All these functions and their keywords have been discussed in previous slides.

Section 3

Launch Sites Proximities Analysis

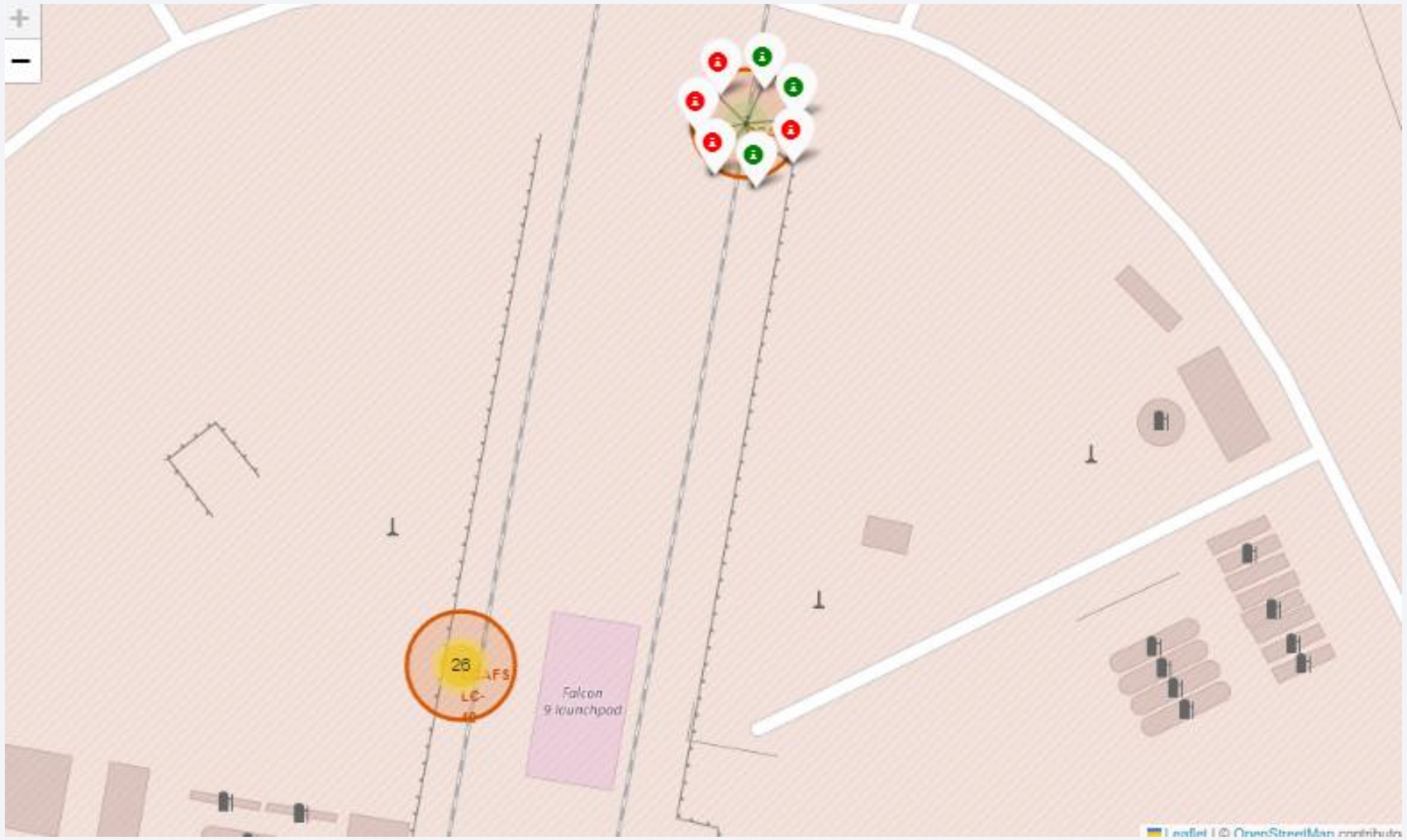
SpaceX Launch Site Locations



Observing the launch site locations, one thing is clear, all sites are in close proximity to a coastal line and latitudinally close to the equator.

The use of markers enables this type of information display.

Application of Clustering to display information

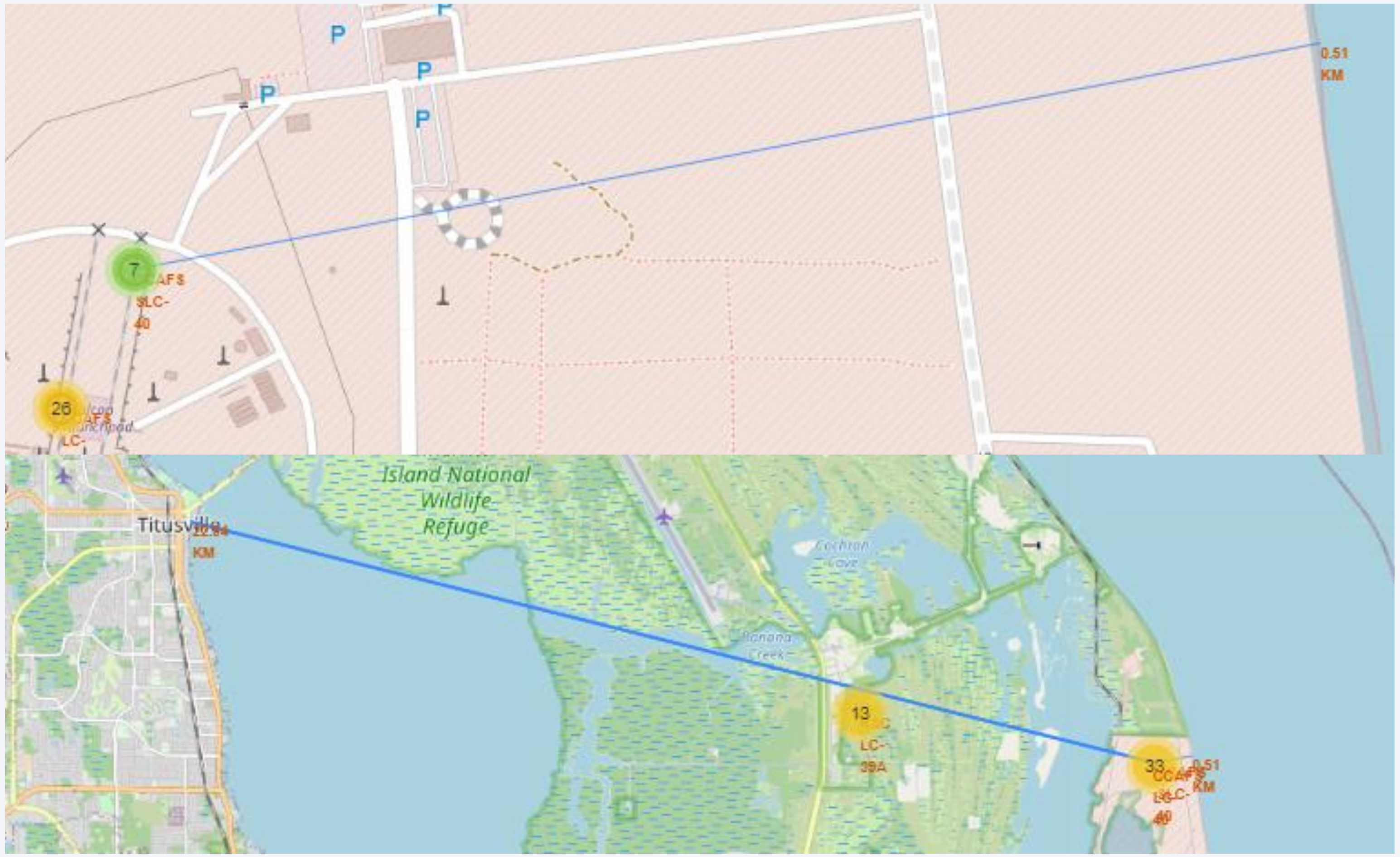


Clustering of markers enables revelation of information on each cluster when it is clicked upon.

The launch site at the top of the image here shows a cluster of icons with color coding to indicate the landing outcome of each flight from the site.

Note how the other cluster indicates the number of launches without revealing their outcomes yet.

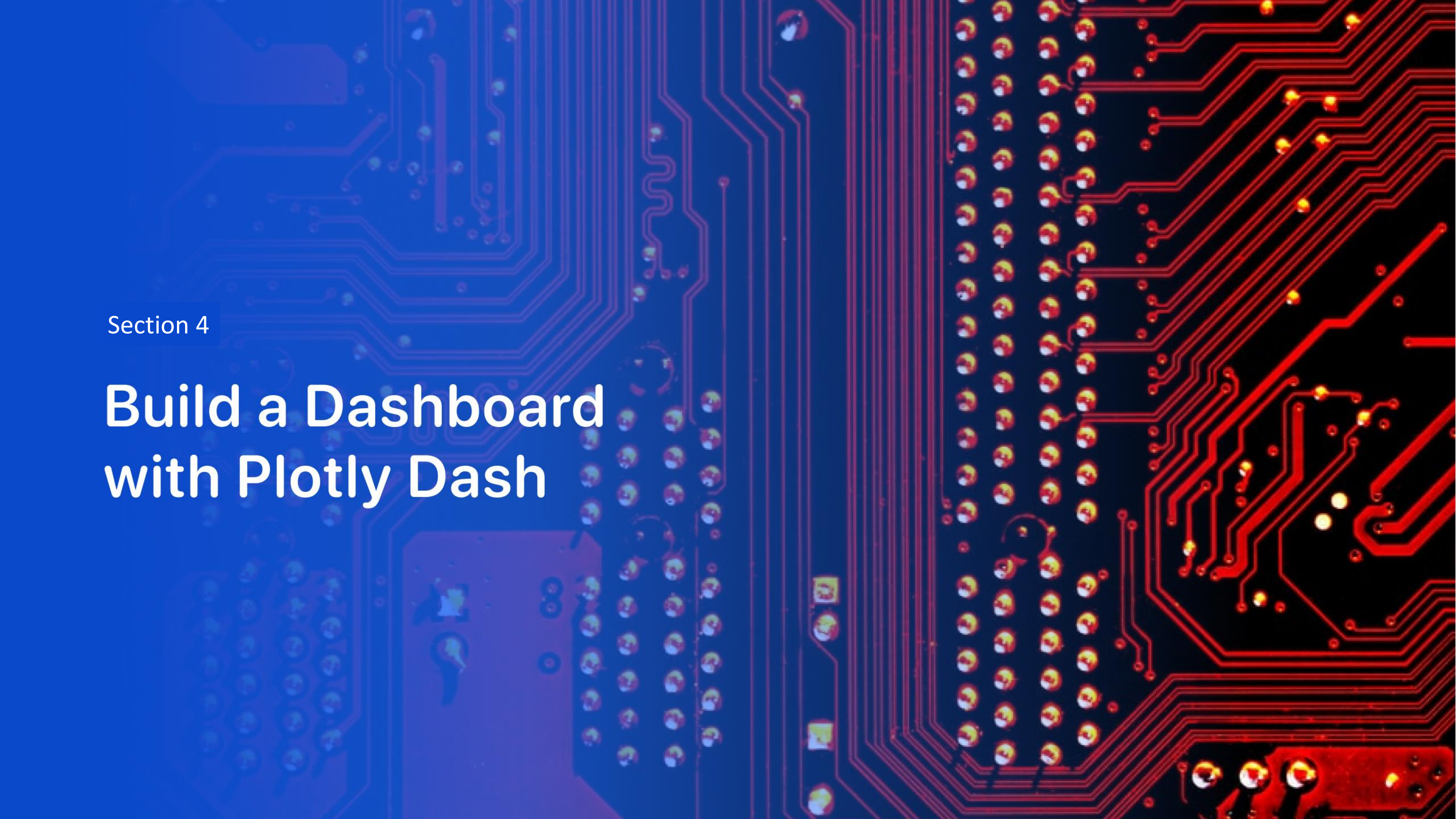
Launch Site Location Constraints



The use of lines and text markers can be very useful.

Comparing the launch site distances to the nearest coastline and nearest city, it is easily demonstrated how safety is factored into the choice of the launch sites location.

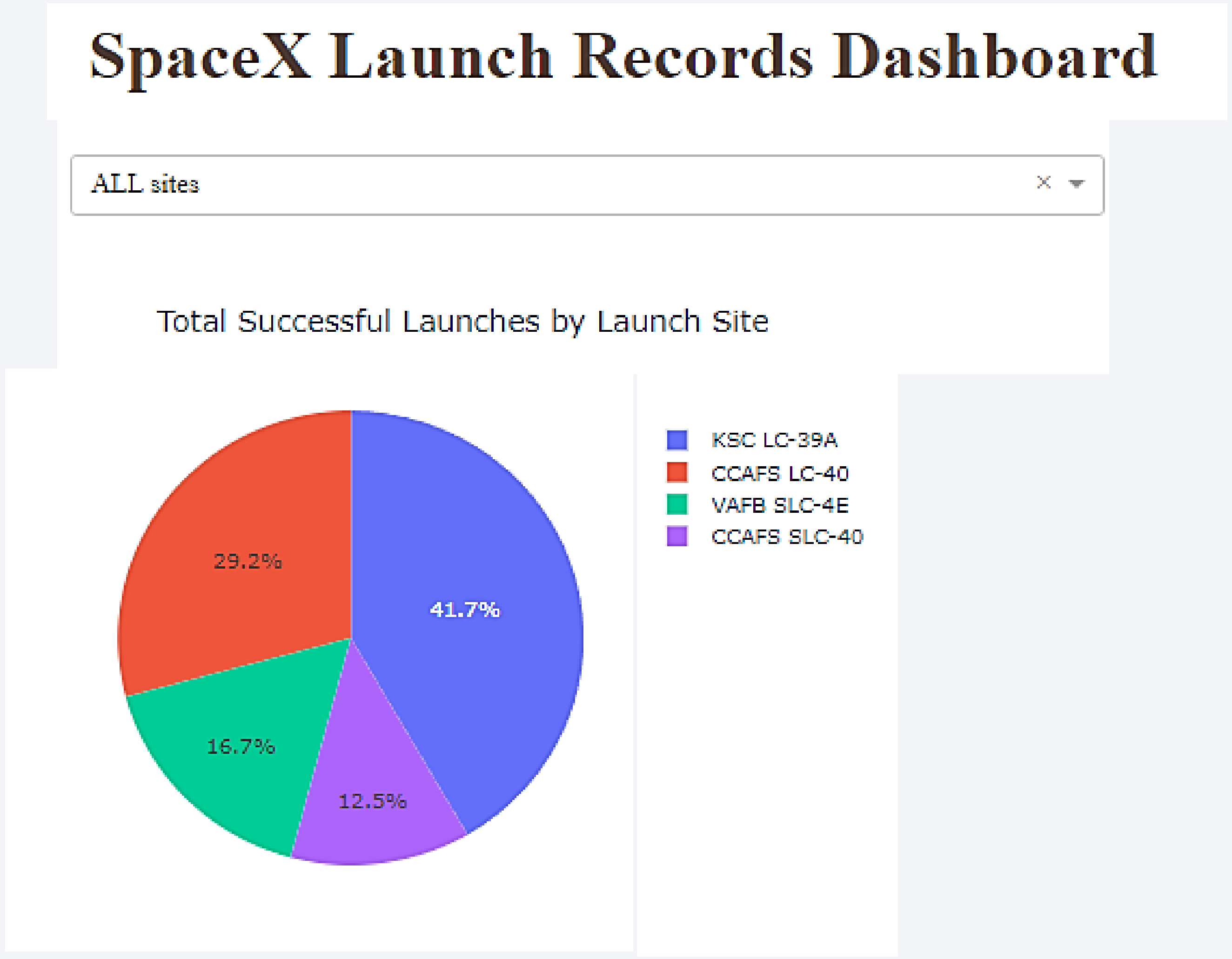
Although the site is only 0.5 km from the nearest coastline, the nearest large population location is a whole 23.8 km away.



Section 4

Build a Dashboard with Plotly Dash

Launch Success For All Sites

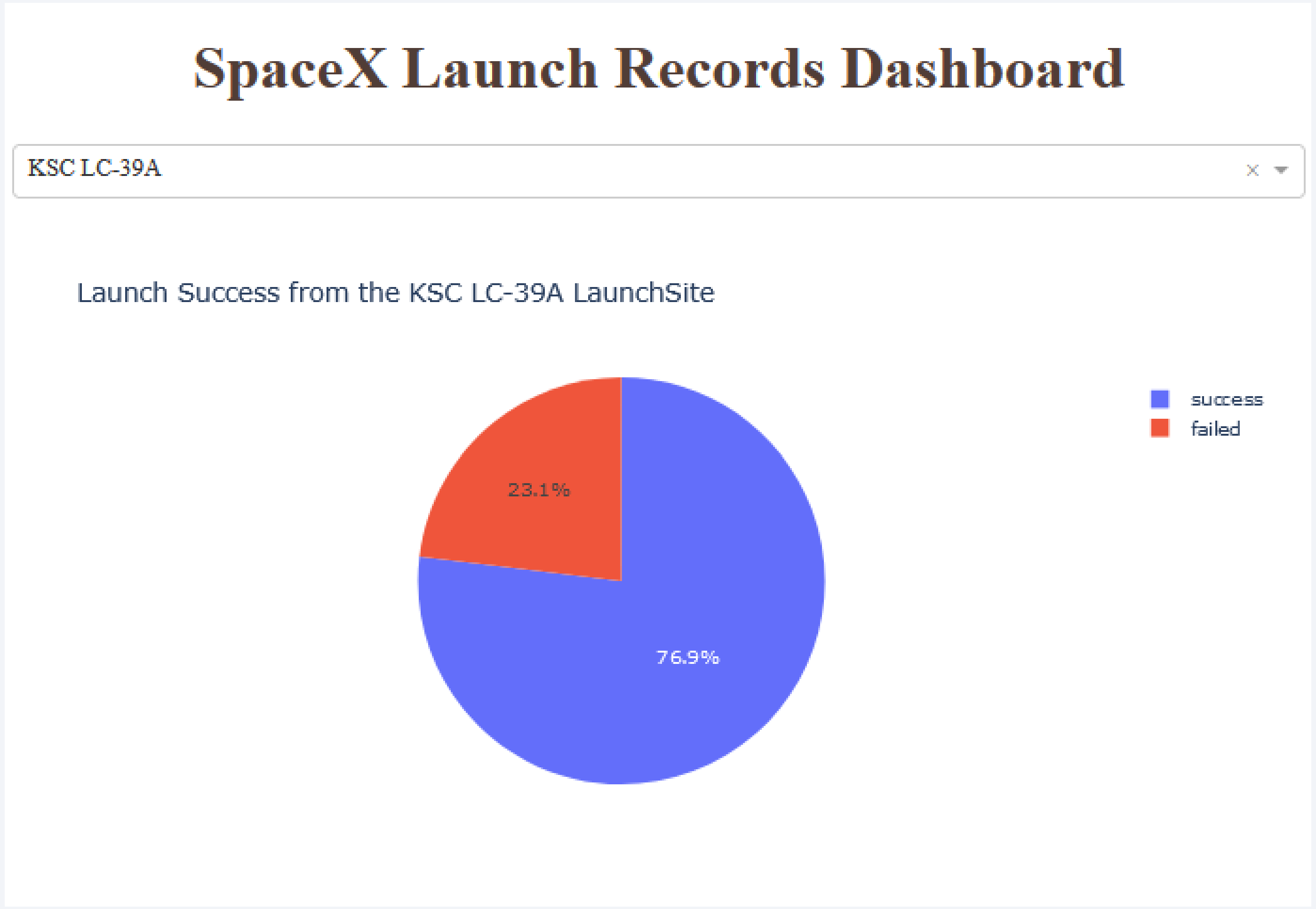

SpaceX Launch Records Dashboard

ALL sites

Total Successful Launches by Launch Site

- The pie chart is from the application that was built using Plotly Dash.
- The logic behind the values is driven by an aggregation by summing.
- Since the outcomes were converted to integer Boolean, the sum represents only the successful outcomes

Highest Success to Failure Ratio Launch Site

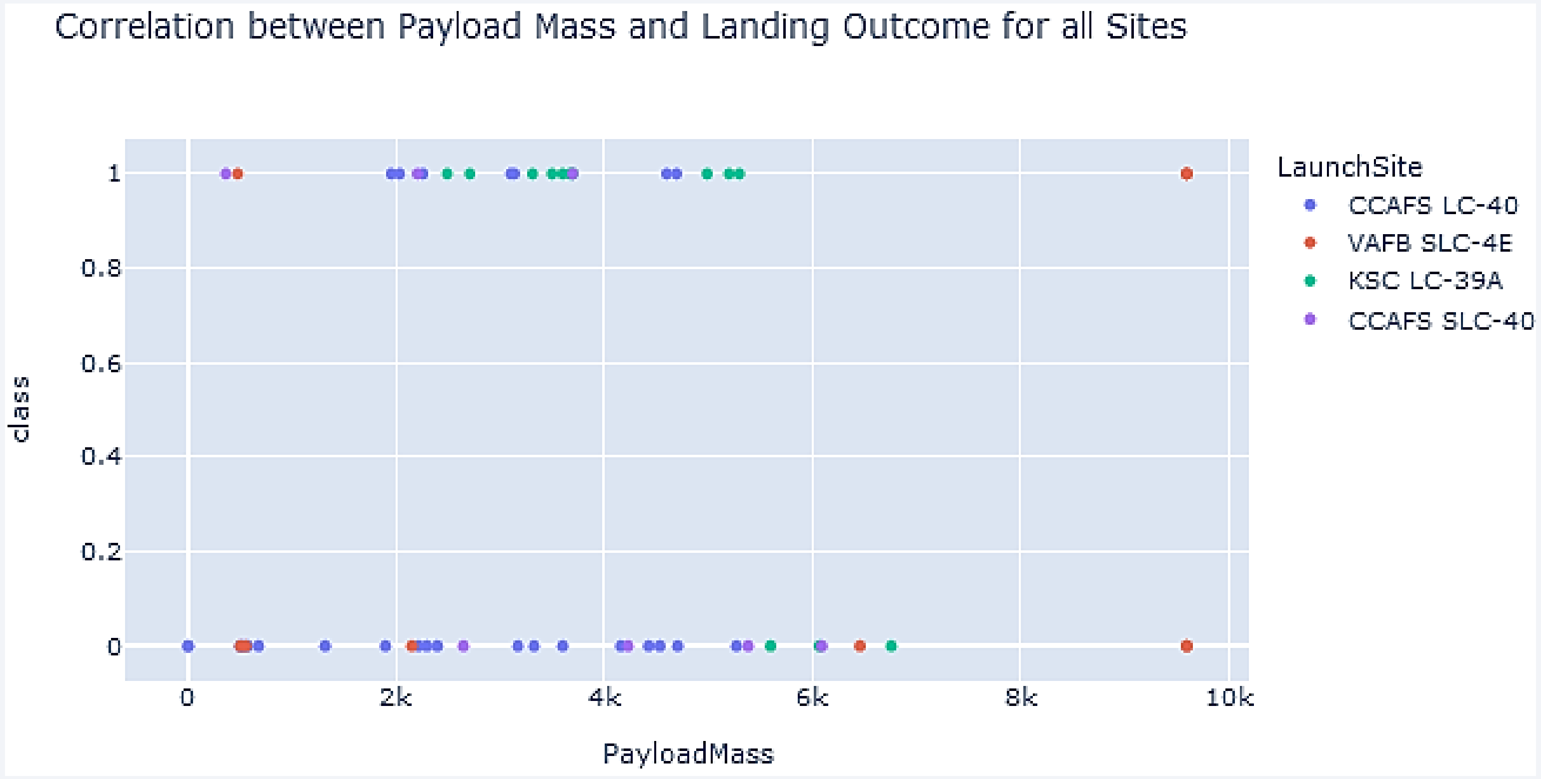

SpaceX Launch Records Dashboard

KSC LC-39A

Launch Success from the KSC LC-39A LaunchSite

- The pie chart is also from the application that was built using Plotly Dash.
- It should be noted that this site also posted the largest number of launches overall. It follows that since the landing success increases with time, the cumulative distribution of the landing outcome should support a success over failure overall ratio.

Payload Mass Vs Landing Outcome

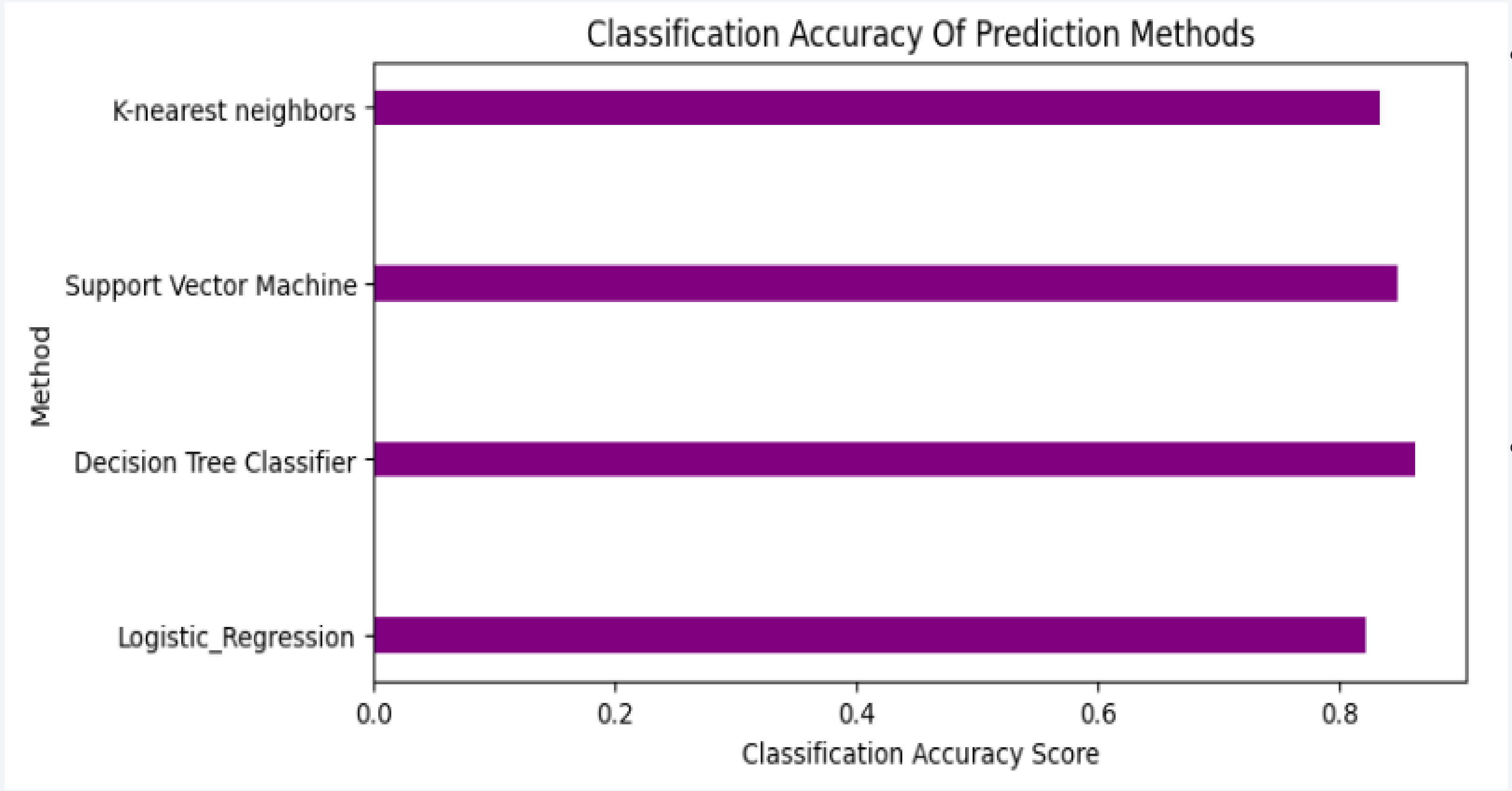


- This plot represents all sites hosting launches that had both successful and unsuccessful landing outcomes.
- There were more launches with payload mass between 2000kg and 5800kg for all sites.

Section 5

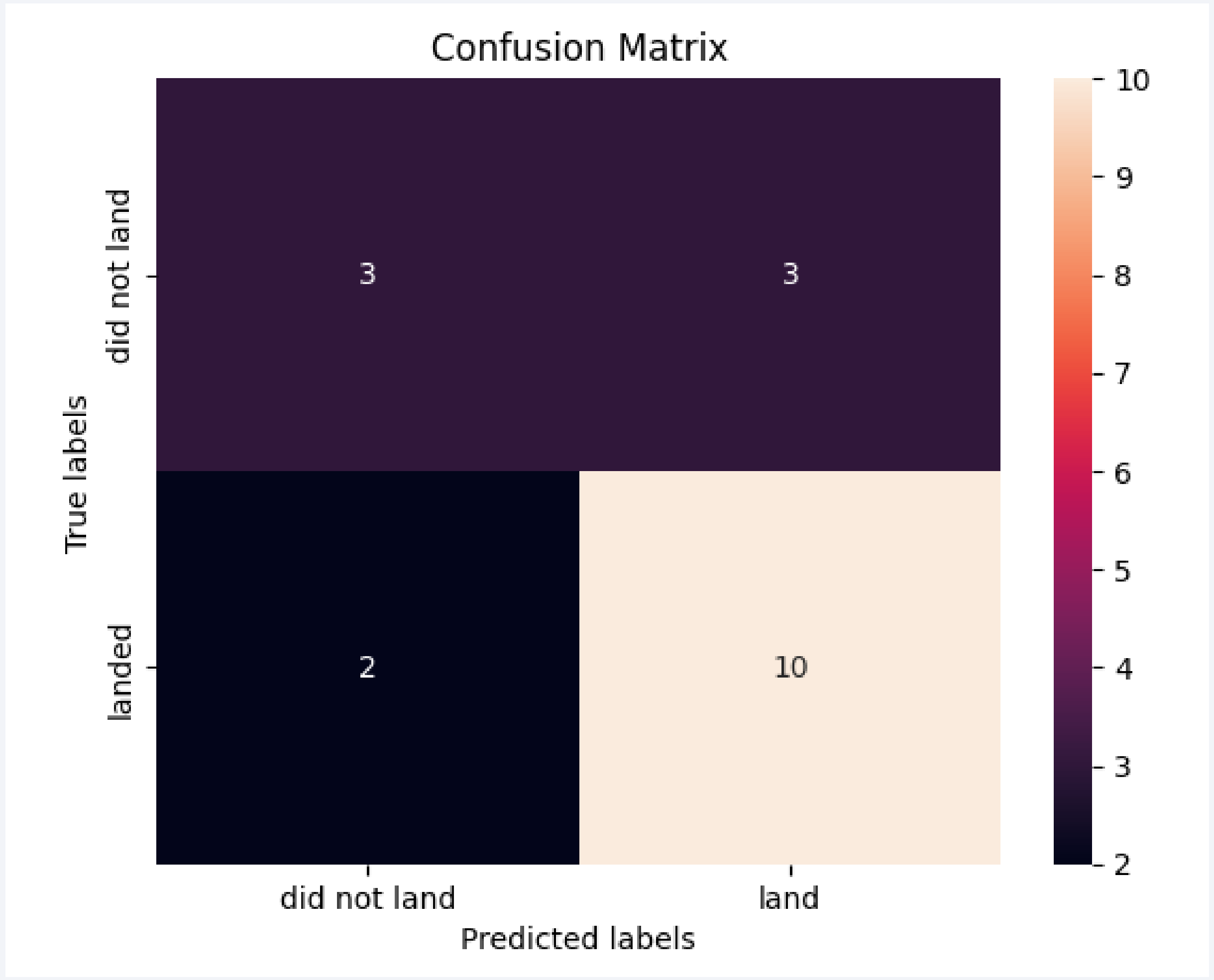
Predictive Analysis (Classification)

Classification Accuracy



- According to the bar chart, the Decision Tree classification model performed best on the prediction out of the 4 models.
- Overall accuracy of all models was close if the confidence level on the prediction were not a factor.

Decision Tree Classifier Confusion Matrix



- The model can distinguish between the different classes but false positives are more of a problem as compared to false negatives.

Conclusions

- The exercise in exploring every step of the data science methodology was very enlightening.
- It is apparent why the steps in the methodology are sequential and how the sequence cannot be altered if the result will be precise.
- Of all features , launch site location is the biggest factor in predicting a landing outcome for any launch.
- The nature of the largest predictor guided the choice of predictive model to apply, hence the classifier models.

Appendix – Launch Sites

Launch Site Names:

KSC LC :	Kennedy Space Center Launch Complex
CCSFS SLC :	Cape Canaveral Space Force Station Space Launch Complex
VAFB SLC :	Vandenberg Air Force Base Space Launch Complex

Target Orbits:

- **HEO**: Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236mi)
- **VLEO**: It is a circular geosynchronous orbit 35,786 kilometers (22,236 miles) above Earth's equator and following the direction of Earth's rotation.
- **PO**: It is one type of satellites in which a satellite passes above or nearly above both poles of the body being orbited (usually a planet such as the Earth

Appendix - Orbits

- **LEO**: Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the man-made objects in outer space are in LEO [1].
- **VLEO**: Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation[2].
- **GTO**: A geosynchronous Transfer Orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website [3] .
- **SSO** (or SO): It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .
- **ES-L1**: At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth[5].
- **HEO**: A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth [6].
- **ISS**: A modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies: NASA (United States), Roscosmos (Russia), JAXA (Japan), ESA (Europe), and CSA (Canada) [7]
- **MEO**: Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit. These are "most commonly at 20,200 kilometers (12,600 mi), or 20,650 kilometers (12,830 mi), with an orbital period of 12 hours [8]

Thank you!