IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Paul Gayle
June 29th, 2025

Outline

Executive Summary

- Summary of methodologies
    - Data Collection
    - 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
    - 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 Space X can reuse the first stage. **Therefore, if we can determine if the first stage will land, we can determine the cost of a launch**. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.
- Problems you want to find answers
    - What factors determine if the rocket will land successfully?
    - What conditions needs to be in place to ensure a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

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

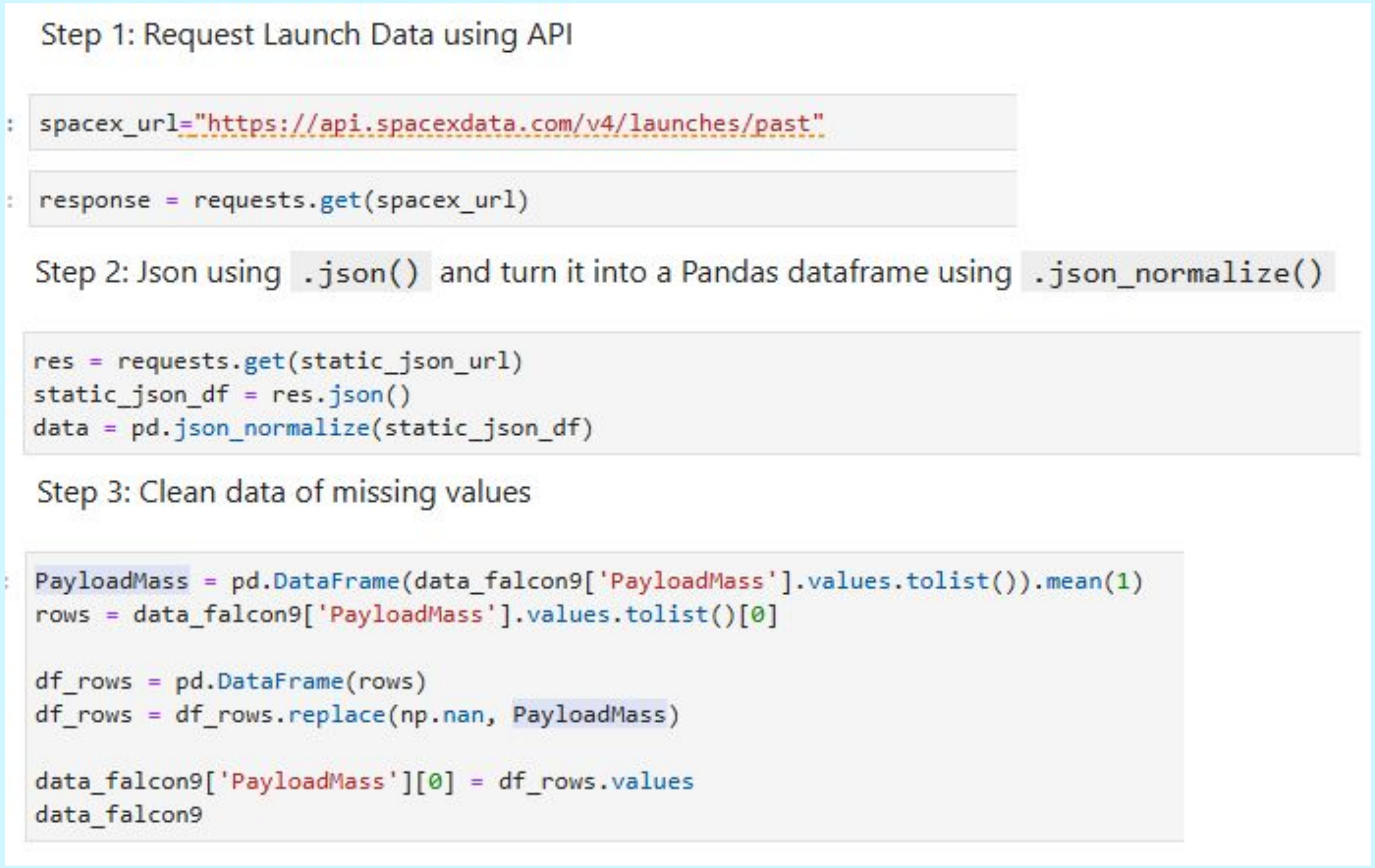
Data Collection

- Objective was to extract the launch records as a HTML table, parse the table and convert it to a Pandas dataframe for future analysis.
- To begin, data collection was done using *get_request* to the SpaceX API.
- Next, we decoded the response content as Json using *.json()* function call and turn it into a Pandas dataframe using *.json_normalize()*
- Lastly, we cleaned the data, checked for missing values and filled in missing values where necessary.

Data Collection – SpaceX API

- Used get request to the SpaceX API to collect data, converted to Pandas dataframe using .json_normalize() and then cleaned the requested data.

- Link to notebook: https://github.com/pauldgayle/ibm-data-science-capstone-spacex/blob/main/data-collection-api.ipynb



Step 1: Request Launch Data using API

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

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

Step 2: Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
res = requests.get(static_json_url)
static_json_df = res.json()
data = pd.json_normalize(static_json_df)
```

Step 3: Clean data of missing values

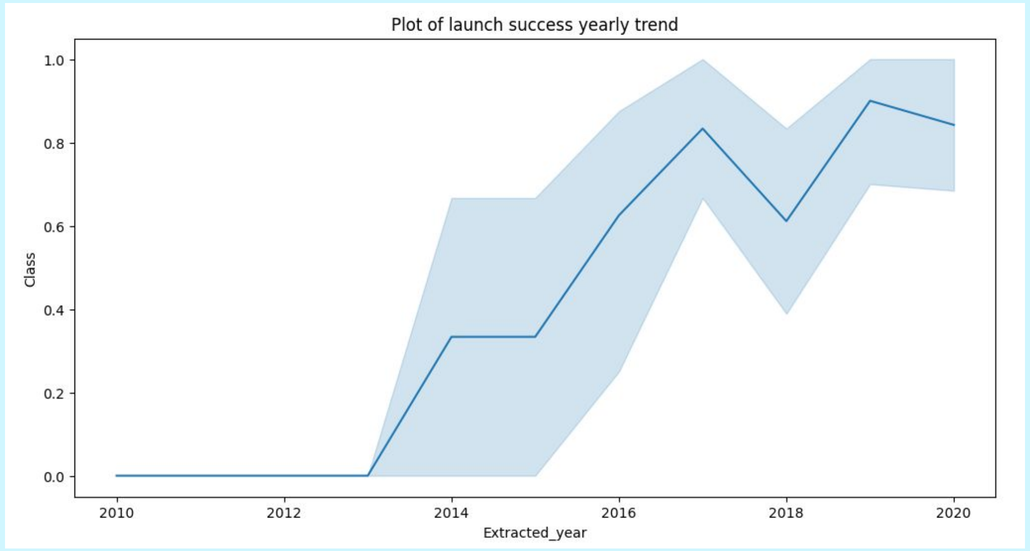
```
PayloadMass = pd.DataFrame(data_falcon9['PayloadMass'].values.tolist()).mean(1)
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

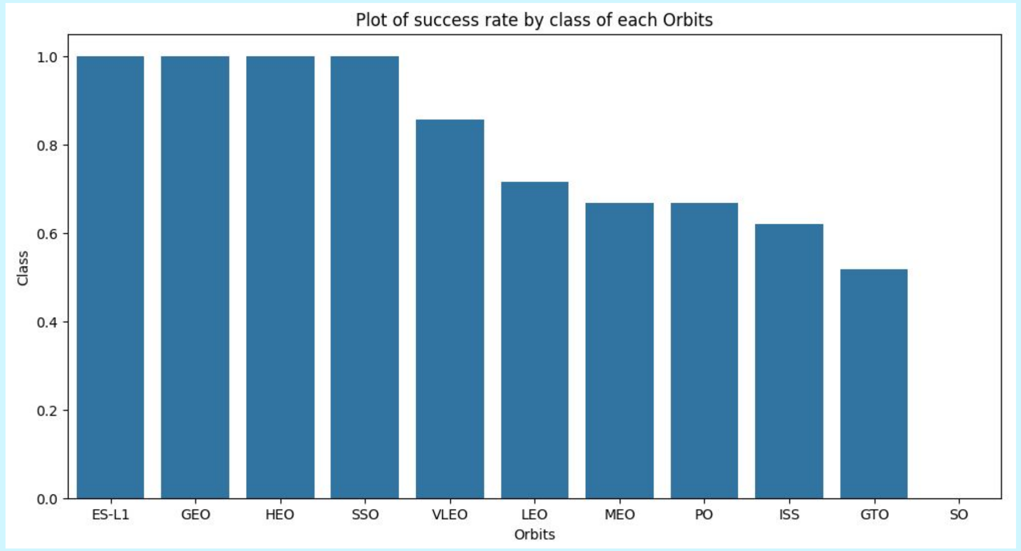
Data Wrangling

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to CSV
- Link to notebook: https://github.com/pauldgayle/ibm-data-science-capstone-spacex/blob/main/data-wrangling.ipynb

EDA with Data Visualization

- Using Seaborn, we were able to visualize the data and find the relationship between flight number and launch site, payload and launch site, success rate of each orbit type, flight number and orbit type and the launch success yearly trend.
- Link to notebook: https://github.com/pauldgayle/ibm-data-science-capstone-spacex/blob/main/eda-data-visualization.ipynb





EDA with SQL

- We applied EDA using SQL to get insights from the data. We wrote the following queries and were able to get answers to each:
    - The names of each 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 notebook: https://github.com/pauldgayle/ibm-data-science-capstone-spacex/blob/main/eda-sql.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to determine the success and/or failure of launches for each site on the folium map.
- We assigned the launch outcomes (failure or success) to class 0 (failure) and 1 (success).
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- Link to notebook: https://github.com/pauldgayle/ibm-data-science-capstone-spacex/blob/main/visual-analytics-folium.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard using Plotly Dash
- We used pie charts to plot the total launches by a certain sites
- We found relationships between Outcome and Payload Mass of different booster versions by plotting them on a scatter graph.

Predictive Analysis (Classification)

- We loaded the data using Numpy and Pandas, transformed the data and split the data into training and test data.
- We built multiple machine learning models using GridSearchCV.
- Then, we used accuracy as the metric for our model
- After multiple iterations, we found the best classification model.
- Link to the notebook:

Results

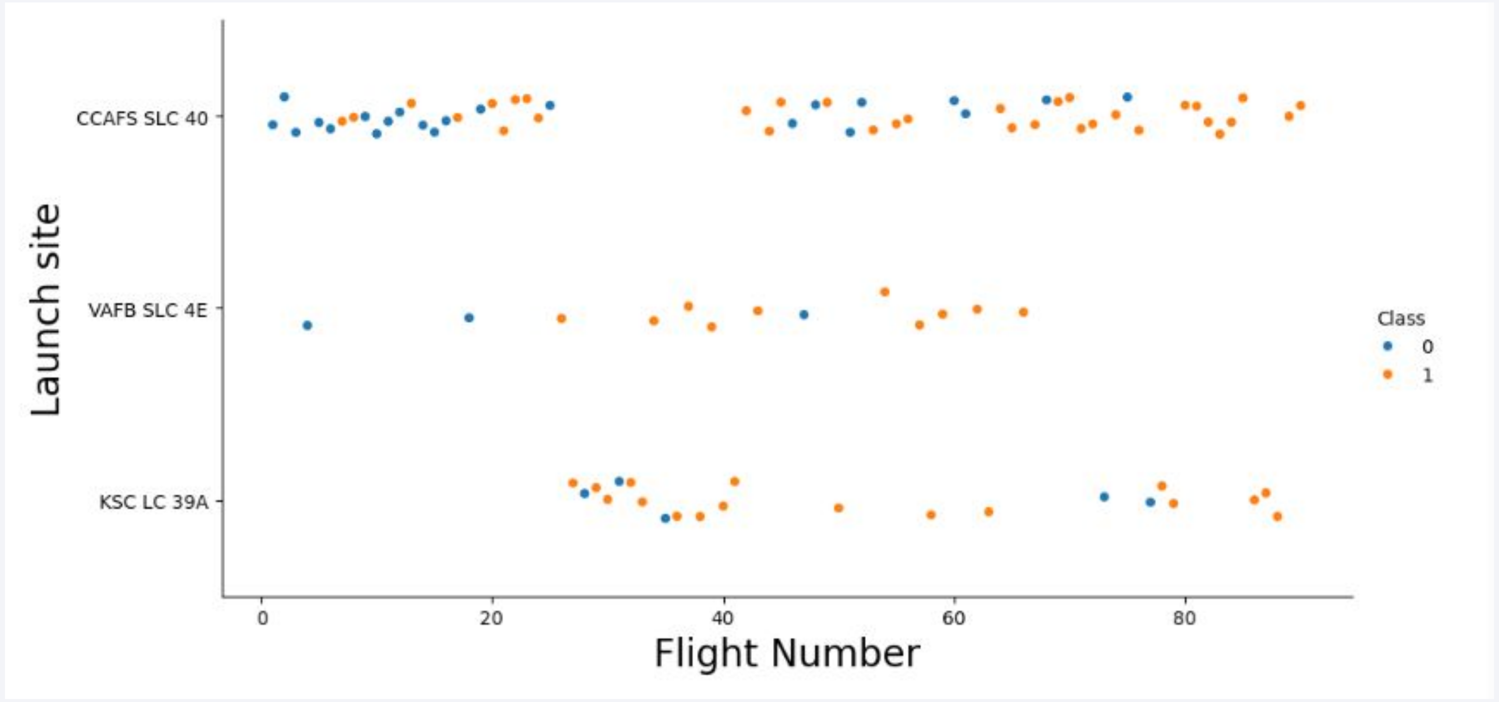
- Exploratory data analysis results
  - Found the relationship between multiple columns of the data
- Interactive analytics demo in screenshots
  - Able to visualize different launch sites and launch outcomes
- Predictive analysis results
  - Used multiple classification models to find the best fit for our

Section 2

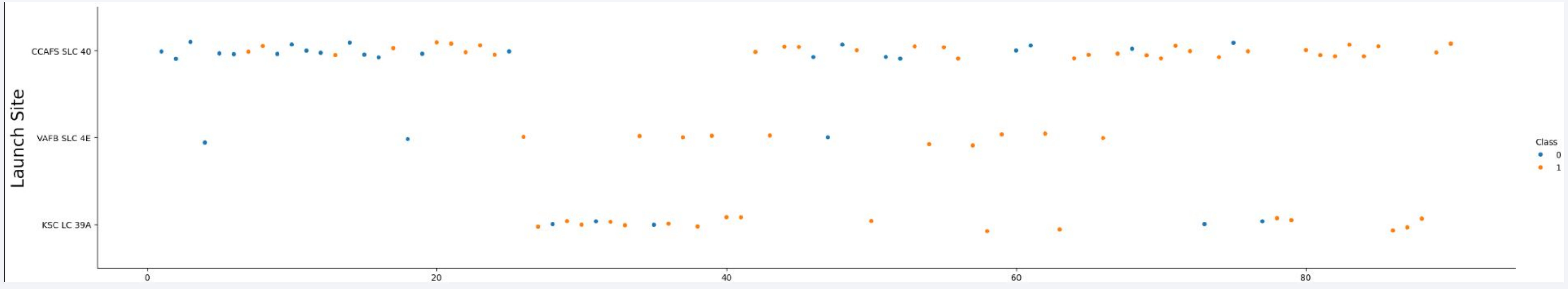
Insights drawn from EDA

Flight Number vs. Launch Site

- This scatter plot shows as the flight number increases at launch sites, you see more orange dots, representing Class 1.
- Meaning, the larger the flight number at a launch site, the higher the success rate.



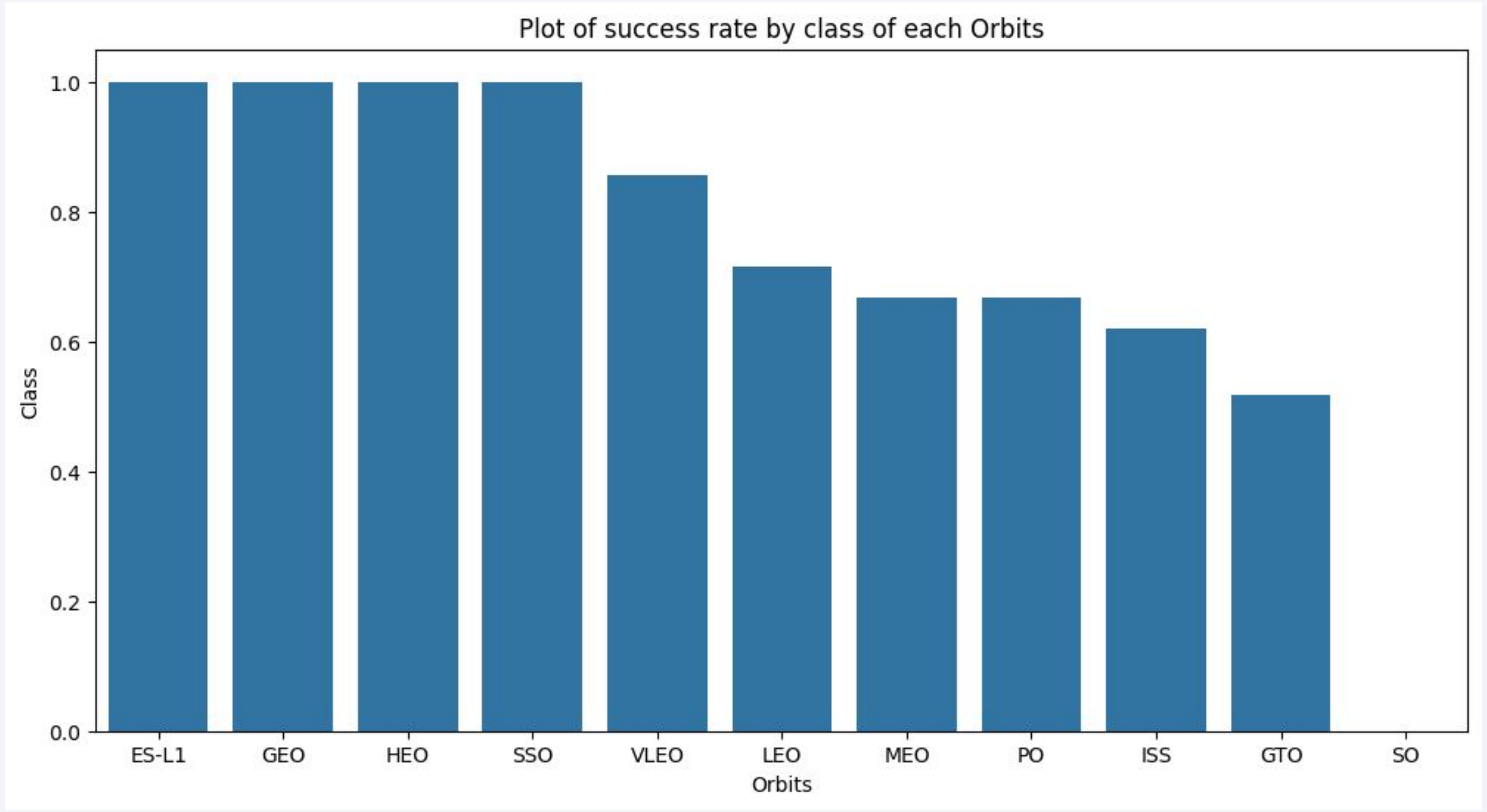
Payload vs. Launch Site



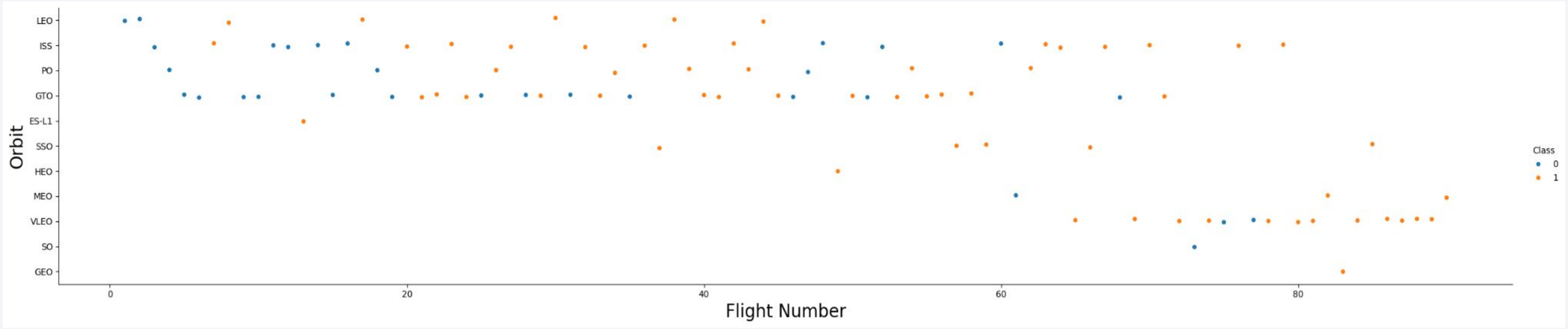
- This graph shows, as the payload mass increases within the launch site, we tend to see more Class 1 markers.
- This yields true more for launch site CCAFS SLC 40.

Success Rate vs. Orbit Type

- Now, we look at the overall success rate for each type of orbit.
- We see a much higher success rate, ~1, for orbits ES-L1, GEO, HEO and SSO.

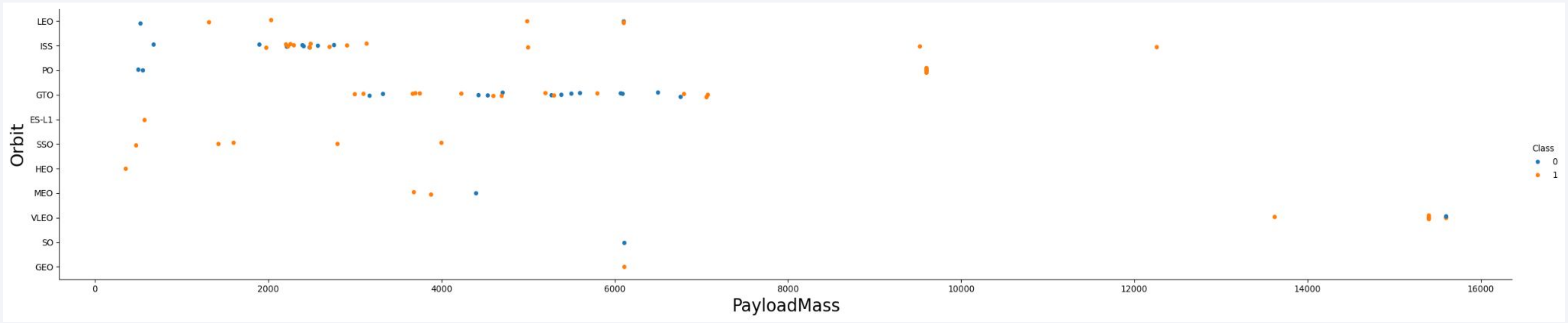


Flight Number vs. Orbit Type



- Above shows a scatter point to determine any relationships between flight number and orbit type
- One relationship we found was with orbit LEO - which shows higher success rate after the number of flights

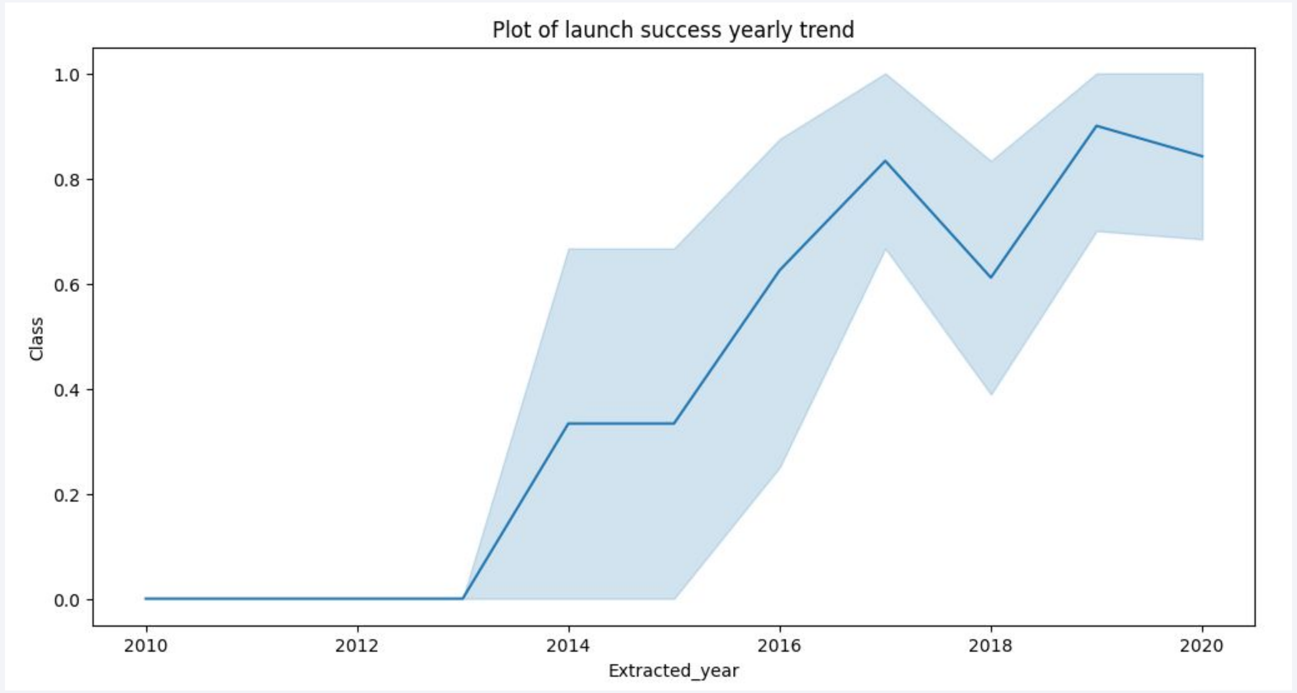
Payload vs. Orbit Type



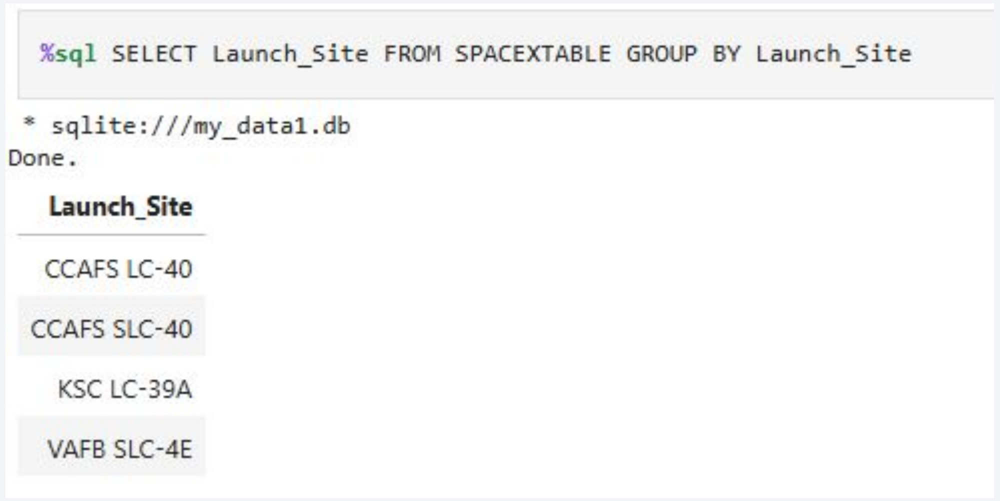
- Above is a scatter point to determine a relationship between payload mass and orbit type
- One relationship found is with orbits LEO, PO and ISS: when the payload mass increases, so does the launch success rate
- However, we do not see the same relationship with orbits GTO, as its success rate decreases with the increase in payload mass.

Launch Success Yearly Trend

- The right shows a line chart of yearly average success rate
- We see that the success rate for flights has a steady increase between the years 2013 and 2020



All Launch Site Names



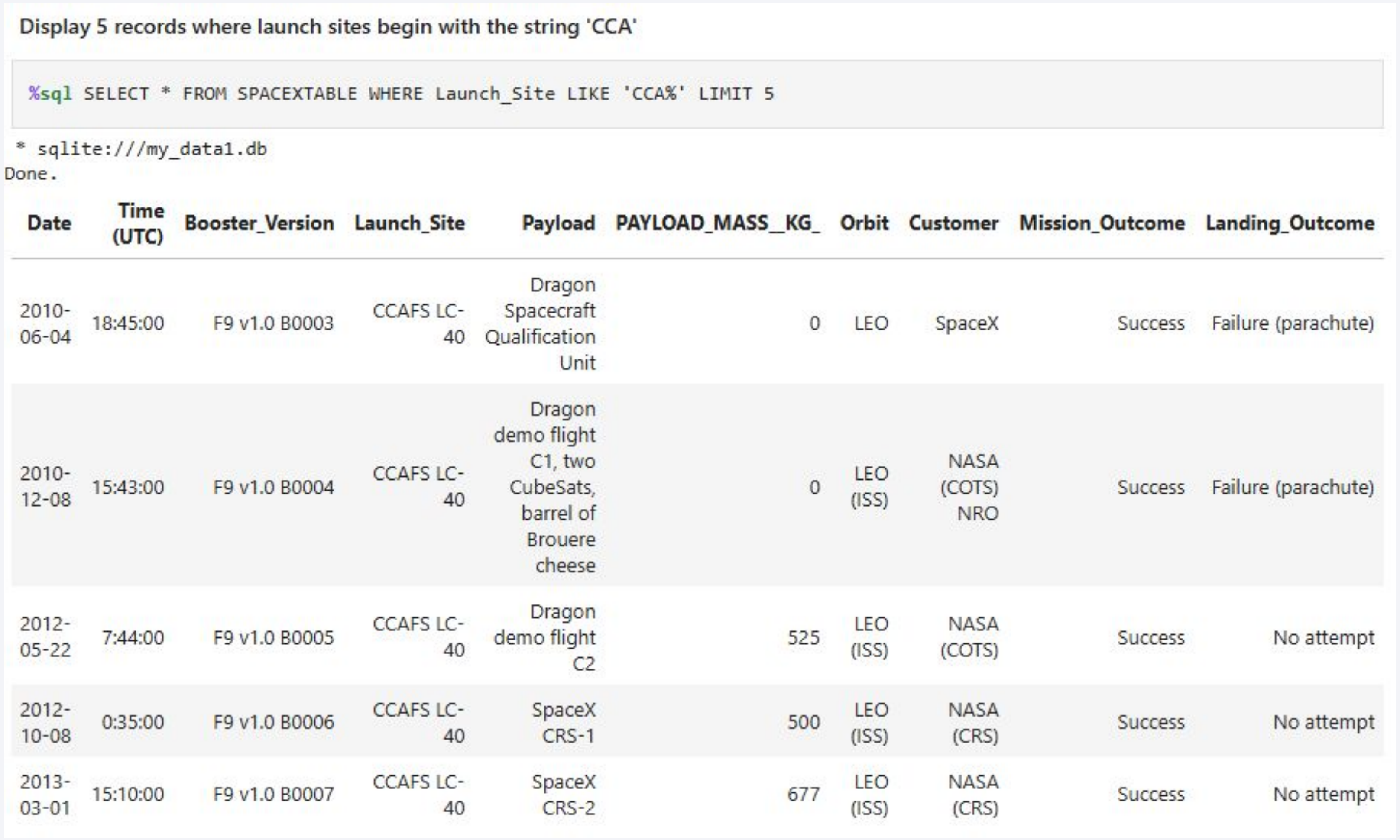
```
%sql SELECT Launch_Site FROM SPACEXTABLE GROUP BY Launch_Site
```

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

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

- Query to pull all launch site names, using a *GROUP BY* in order to not get duplicates

Launch Site Names Begin with 'CCA'



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

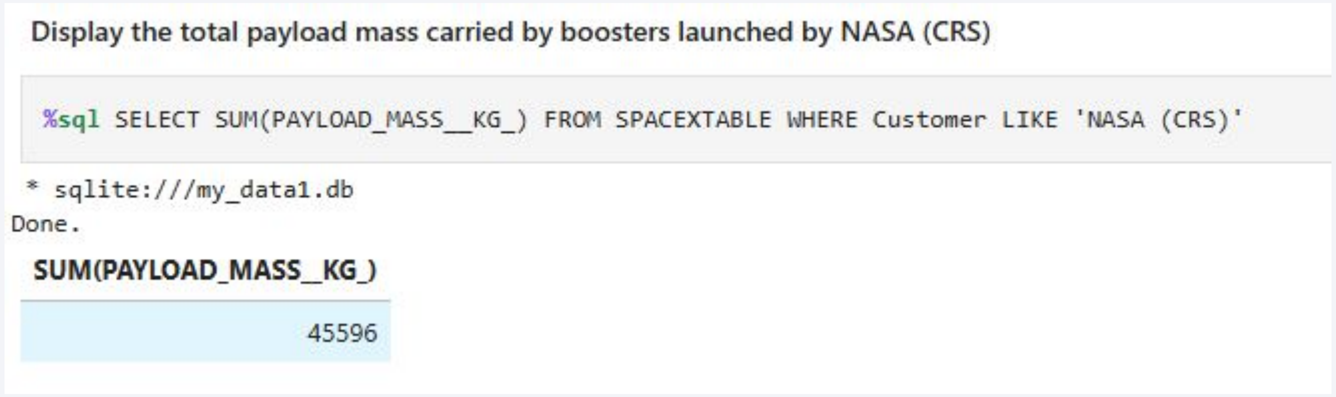
```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

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

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

- Query to pull launch site names beginning with ‘CCA’
- Used *LIKE ‘..%’* in order to pull strings that begin with ‘CCA’, and end with any type of string

Total Payload Mass



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

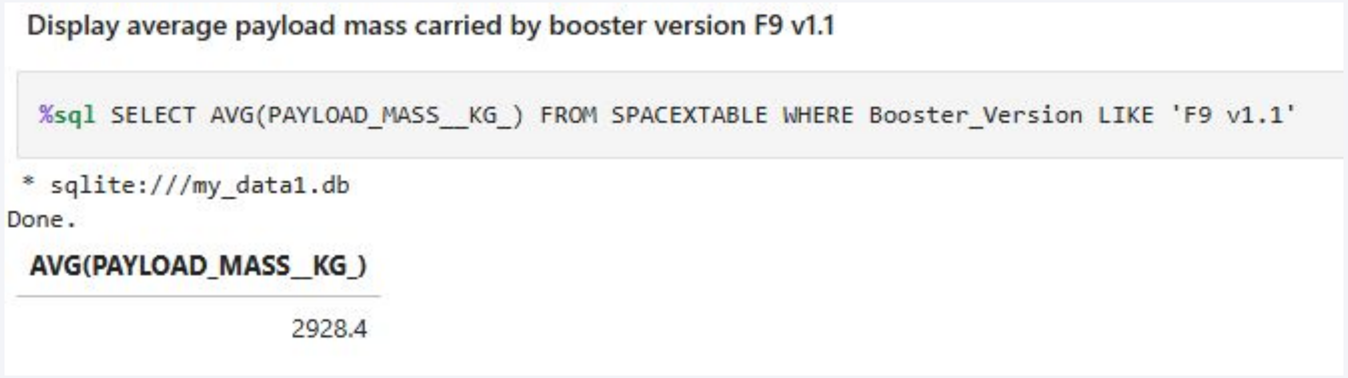
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE 'NASA (CRS)'
```

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

SUM(PAYLOAD_MASS__KG_)
45596

- Query to calculate the total payload carried by boosters from NASA - **45,596 KG**
- Used *SUM* and *LIKE* in order to get the total mass from a specific string

Average Payload Mass by F9 v1.1


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

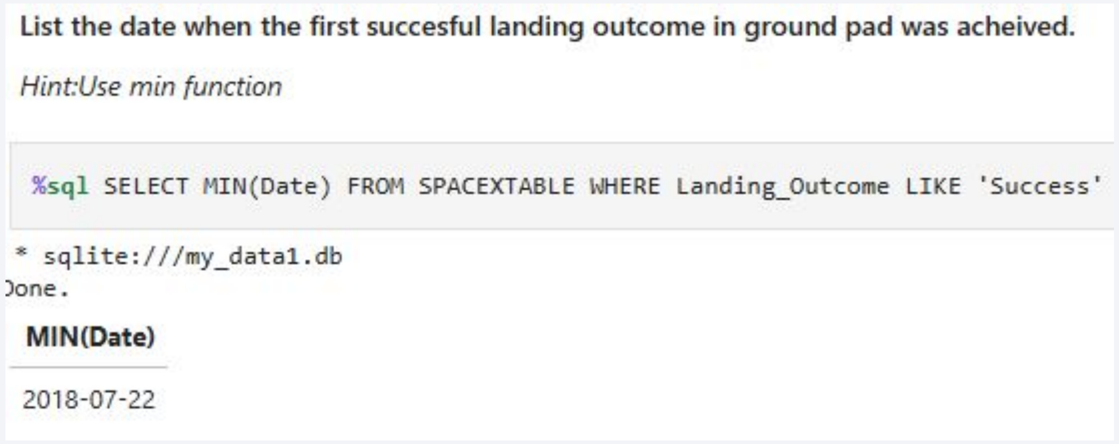
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1'
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

- Query to calculate the average payload mass carried by booster version F9 v1.1 - **2,928.4 KG**
- Used *AVG* and *LIKE* in order to calculate the average of a specific string

First Successful Ground Landing Date


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

Hint:Use min function

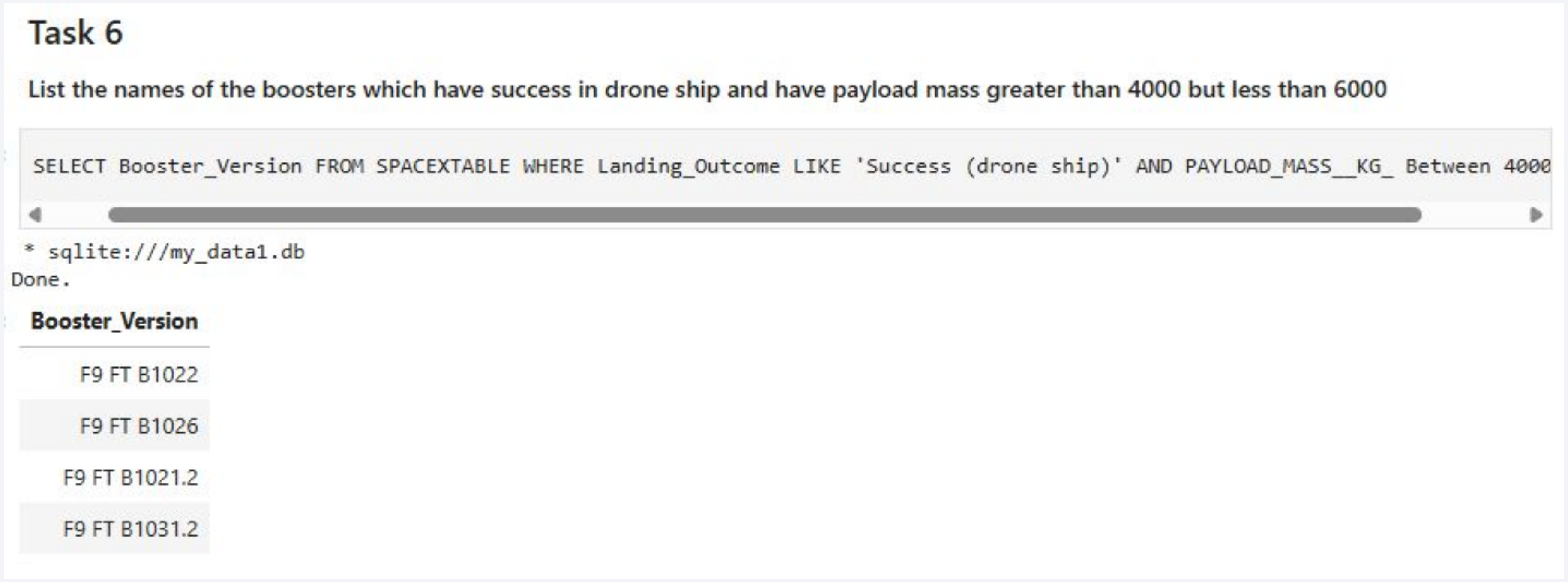
```
%sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success'
```

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

MIN(Date)
2018-07-22

- Query to find the dates of the first successful landing outcome on ground pad - **2018-07-22**
- Used *MIN* in order to get the first date within a string that is *LIKE* success

Successful Drone Ship Landing with Payload between 4000 and 6000



Task 6

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

```
SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (drone ship)' AND PAYLOAD_MASS__KG_ Between 4000
```

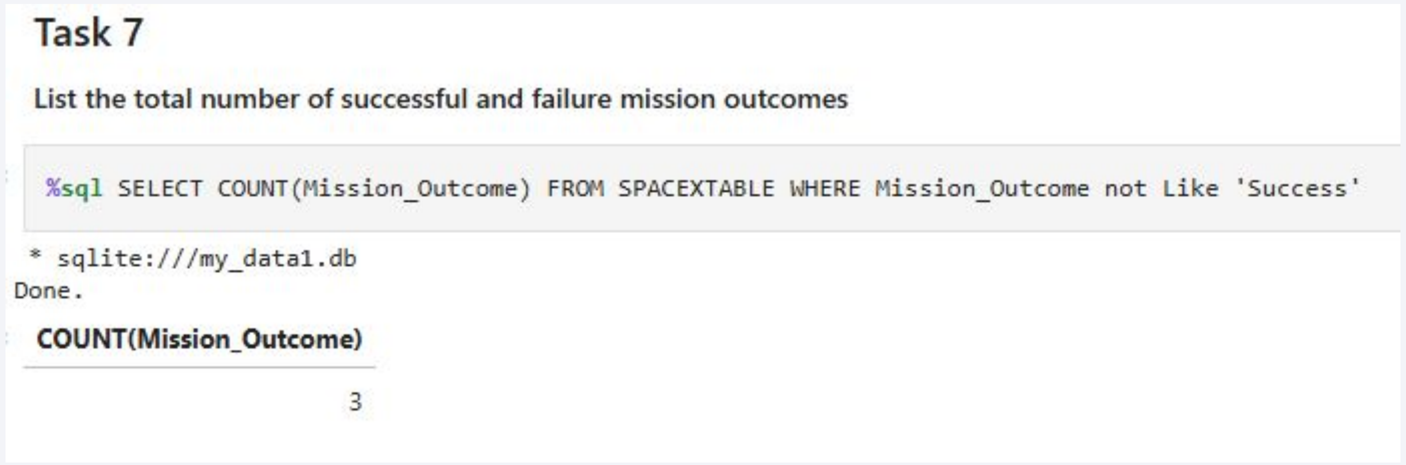
```
 * sqlite:///my_data1.db
Done.
```

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

- Query to list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 - **versions listed above**

- Used *BETWEEN* in order to get a mass between 4000 and 6000

Total Number of Successful and Failure Mission Outcomes



Task 7

List the total number of successful and failure mission outcomes

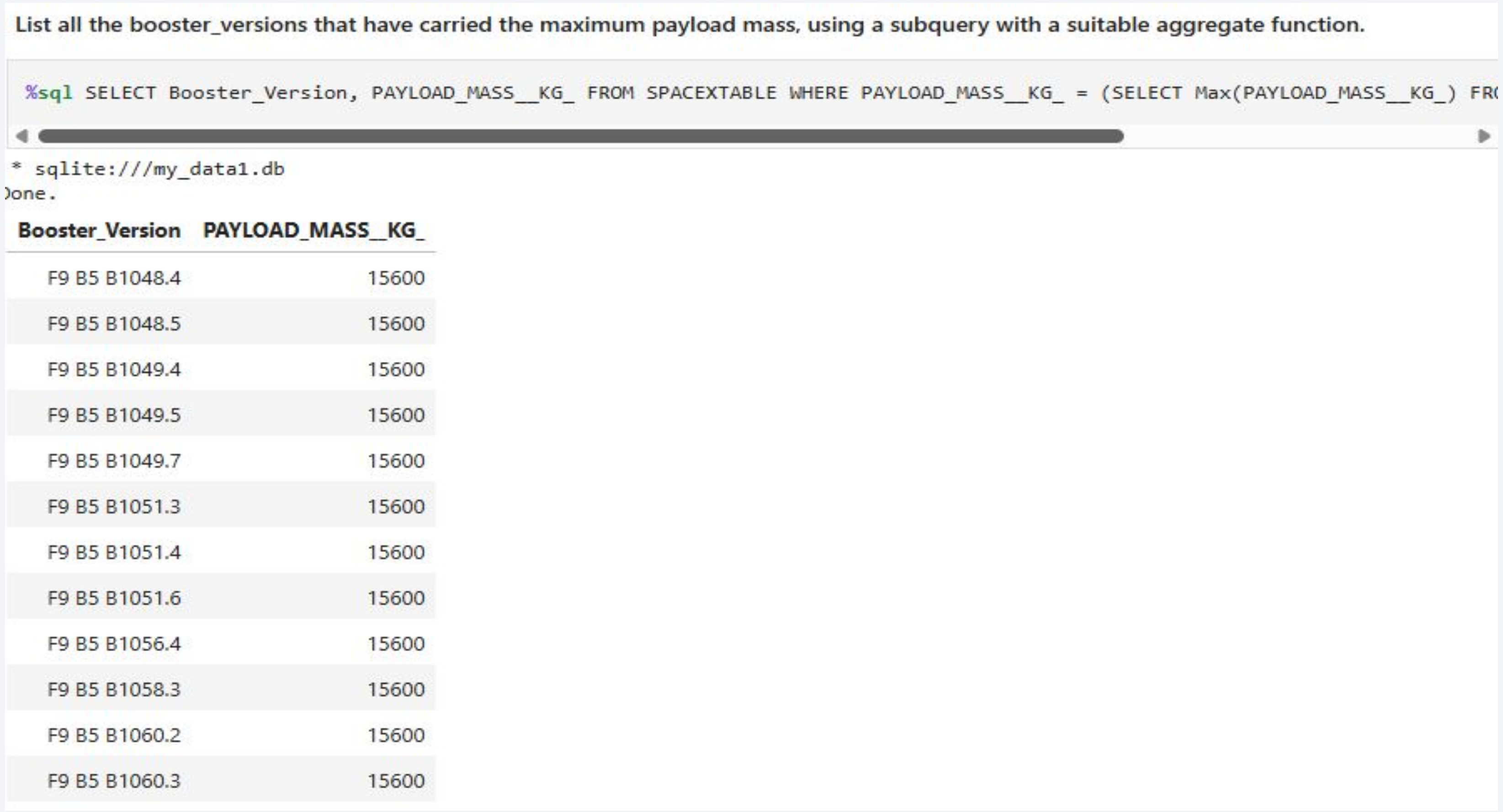
```
%sql SELECT COUNT(Mission_Outcome) FROM SPACEXTABLE WHERE Mission_Outcome not Like 'Success'
```

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

COUNT(Mission_Outcome)
3

- Query to calculate the total number of successful and failure mission outcomes - **Success = 97, Failure = 3**
- Used *COUNT* in order to count the number of Success and Failure

Boosters Carried Maximum Payload



List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.

```
%sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT Max(PAYLOAD_MASS__KG_) FR(
```

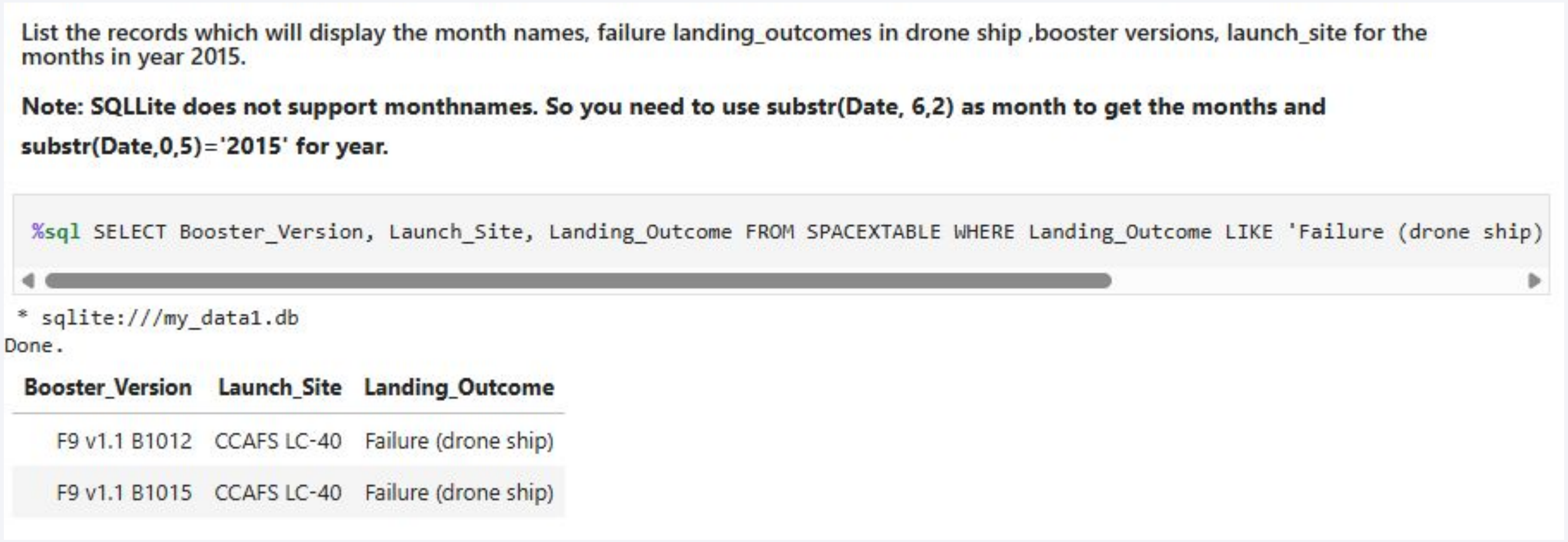
```
 * sqlite:///my_data1.db
)one.
```

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

- Query and subquery to list the names of the booster which have carried the maximum payload mass
- Subquery is cut off, full below

SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT Max(PAYLOAD_MASS__KG_) FROM SPACEXTABLE) ORDER BY Booster_Version

2015 Launch Records



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

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

```
%sql SELECT Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Failure (drone ship)
```

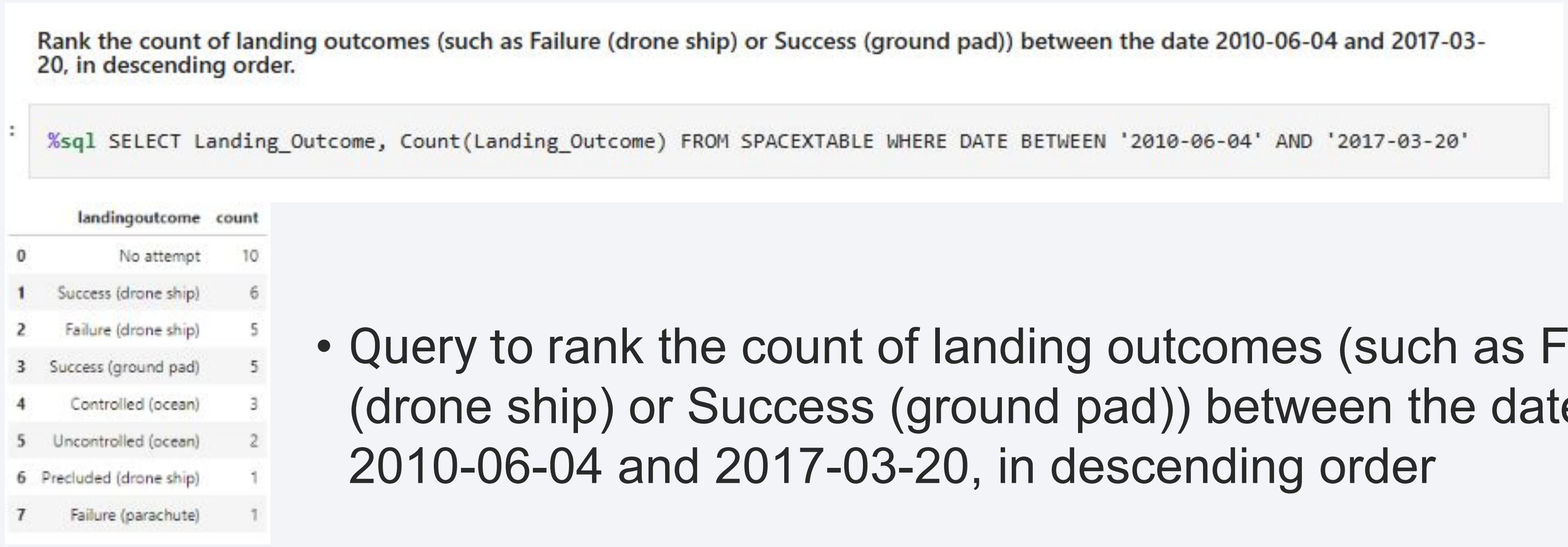
```
 * sqlite:///my_data1.db
Done.
```

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

- Query to list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 - **records listed above**

- Present your query result with a short explanation here

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



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.

```
%sql SELECT Landing_Outcome, Count(Landing_Outcome) FROM SPACEXTABLE WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
```

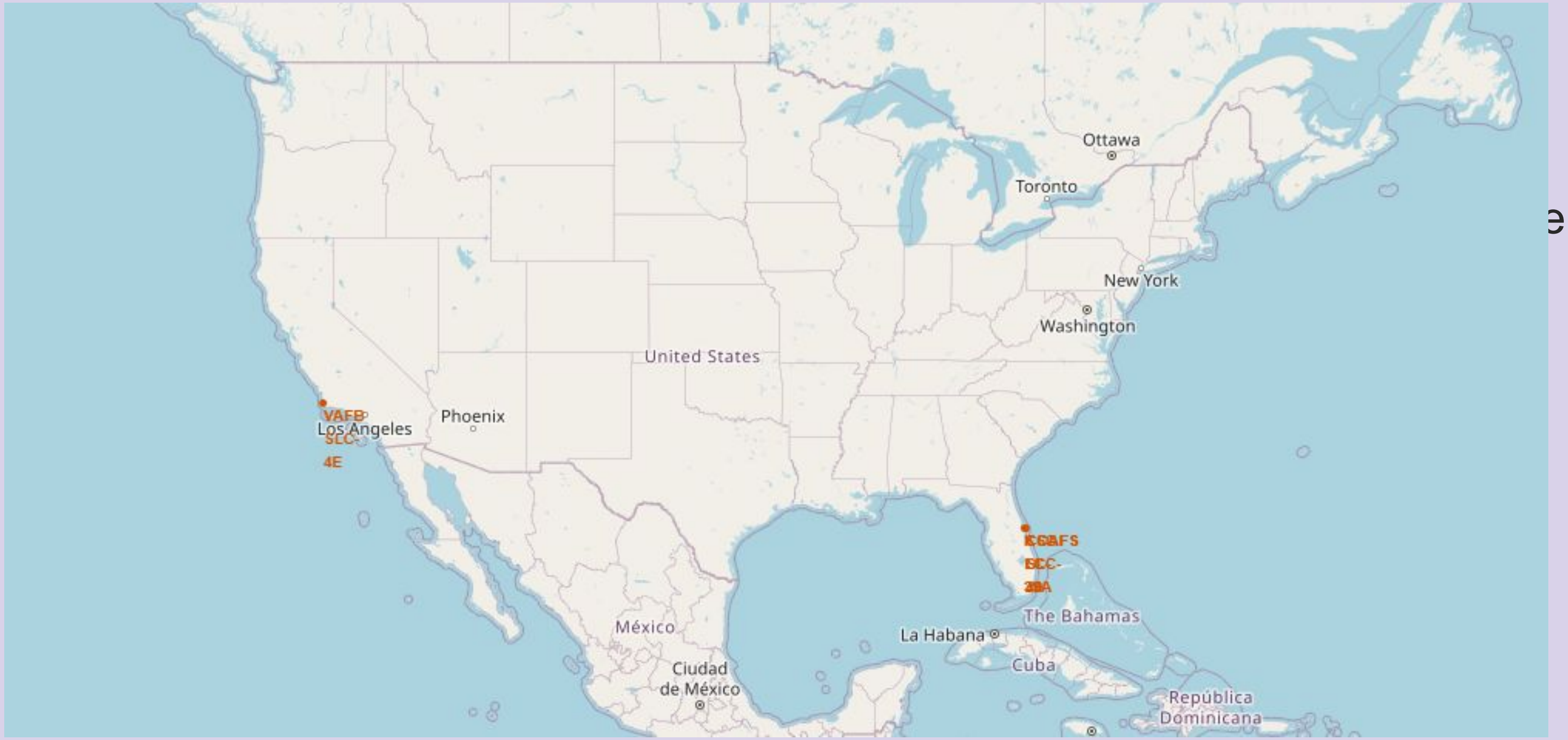
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

- Query to 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

Section 3

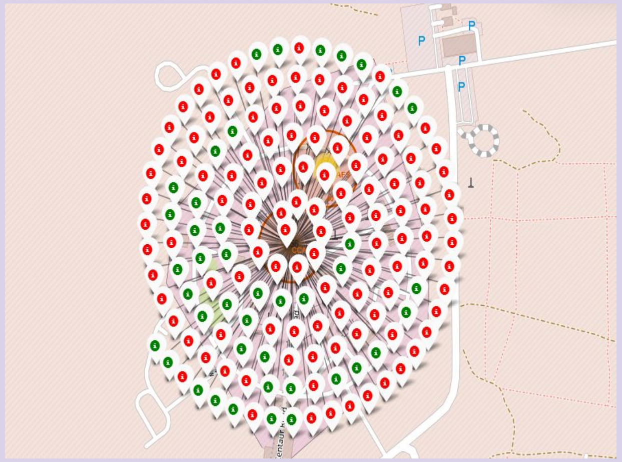
Launch Sites Proximities Analysis

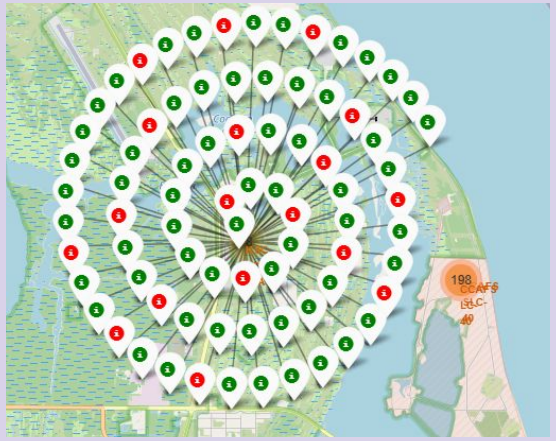
All Launch Sites’ Location Markers on Global Map

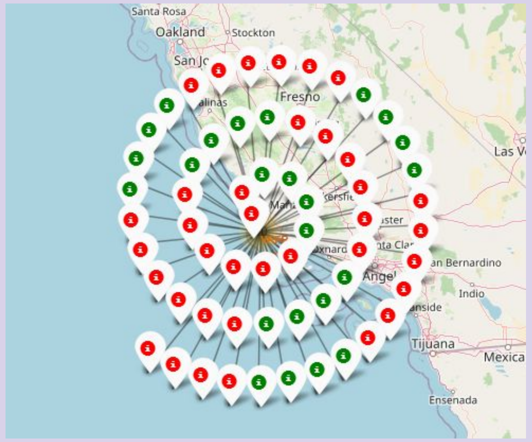


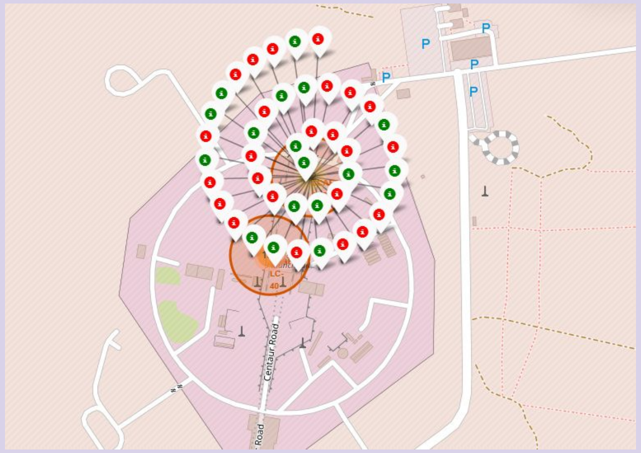
e

Color-Labeled Launch Outcomes



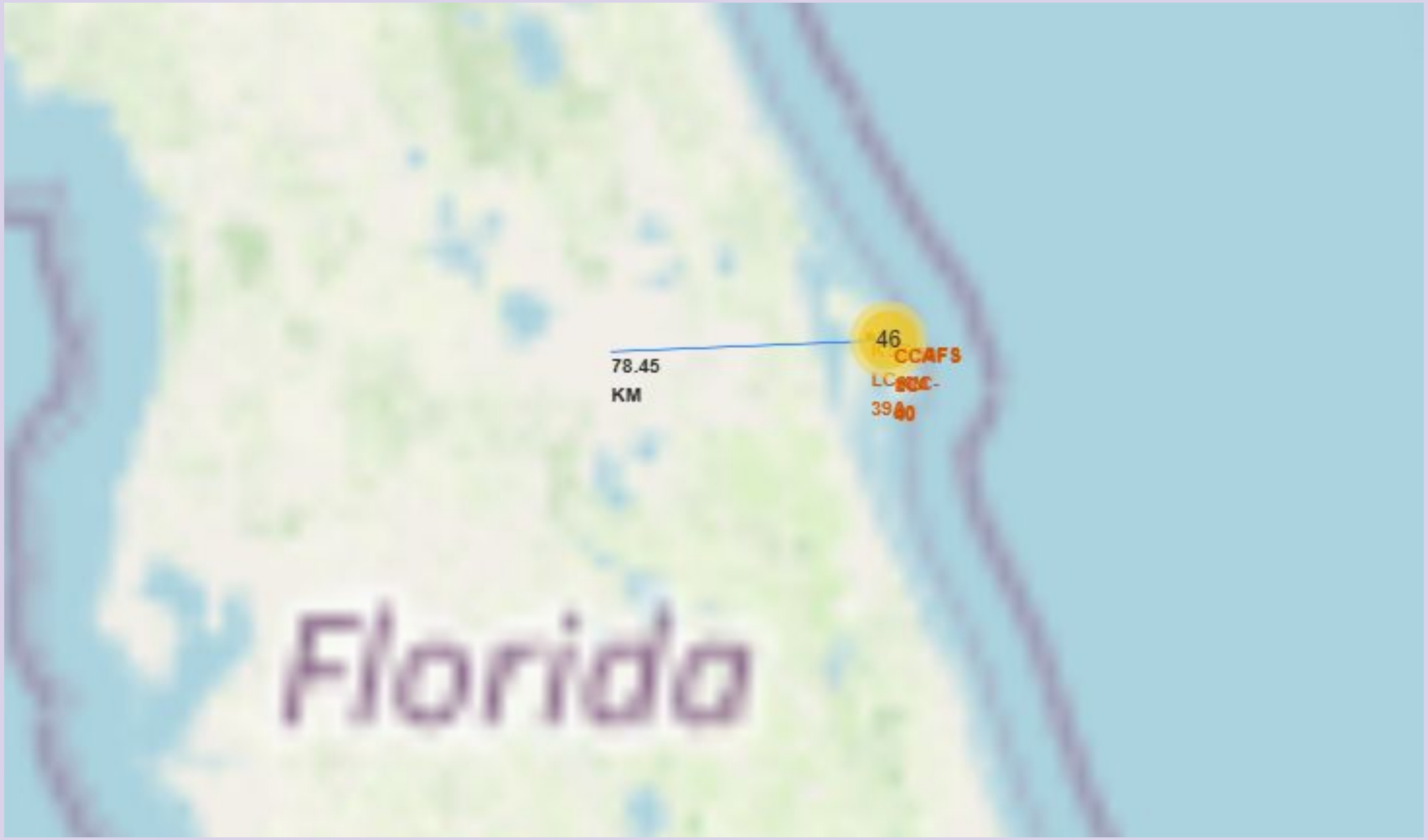






- Green marker = Success
- Red marker = Failure

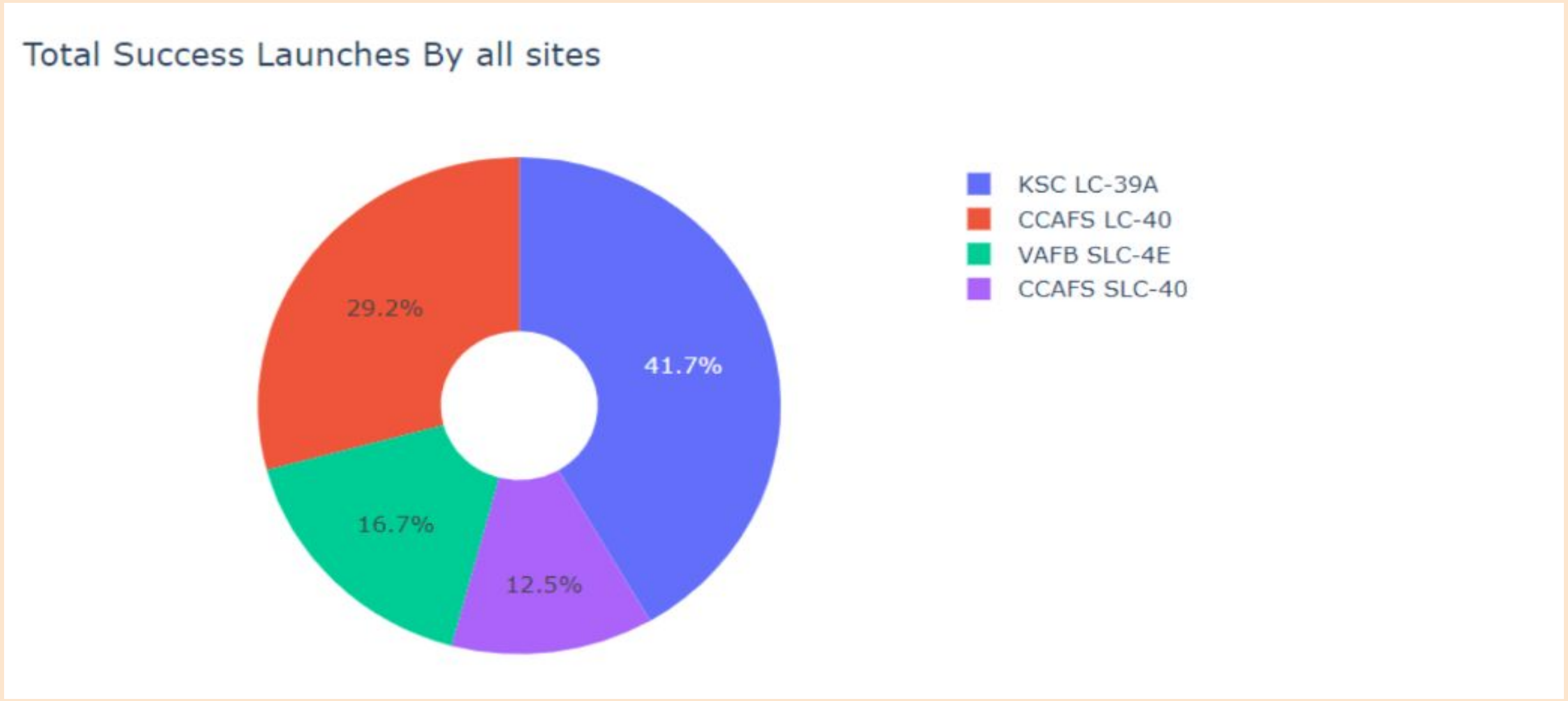
Launch site proximity to coastline



Section 4

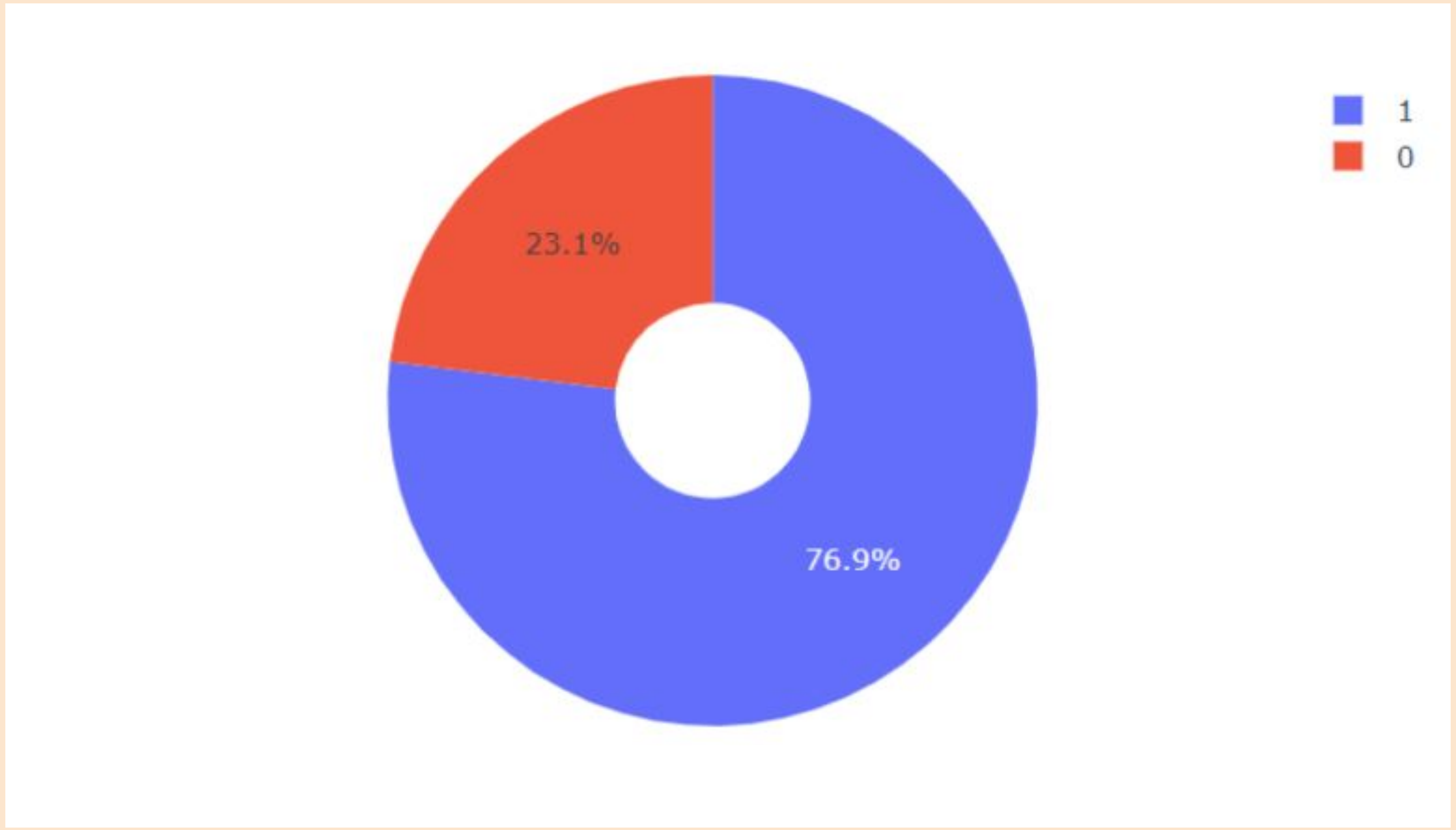
Build a Dashboard with Plotly Dash

Pie Chart of Launch Success Count



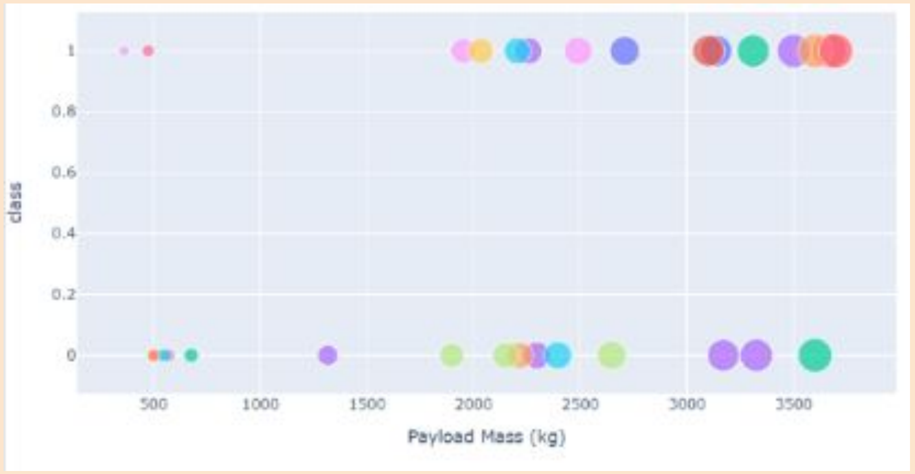
- KSC LC-39A had the most successful launches compared to the other sites

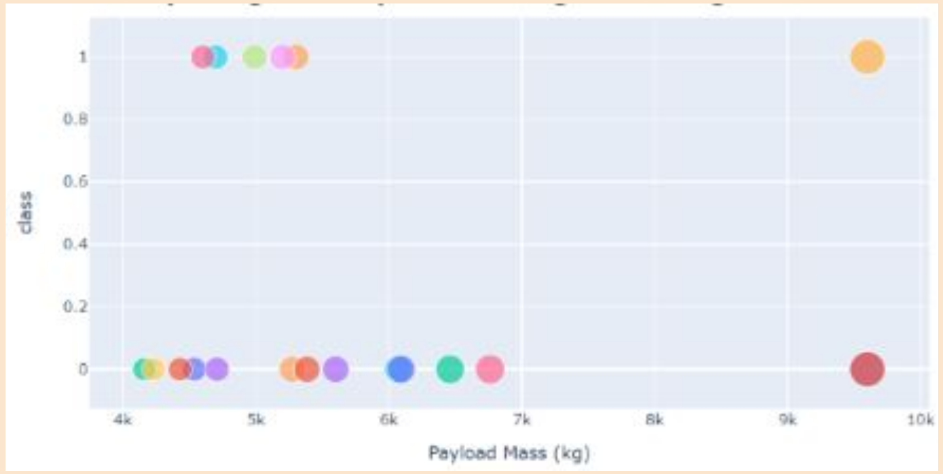
Pie Chart for Launch Sites with the Highest Launch Success Ratio



- KSC LC 39A achieved the highest success ratio - 76.9%

Scatter Plot Payload vs Launch Outcome



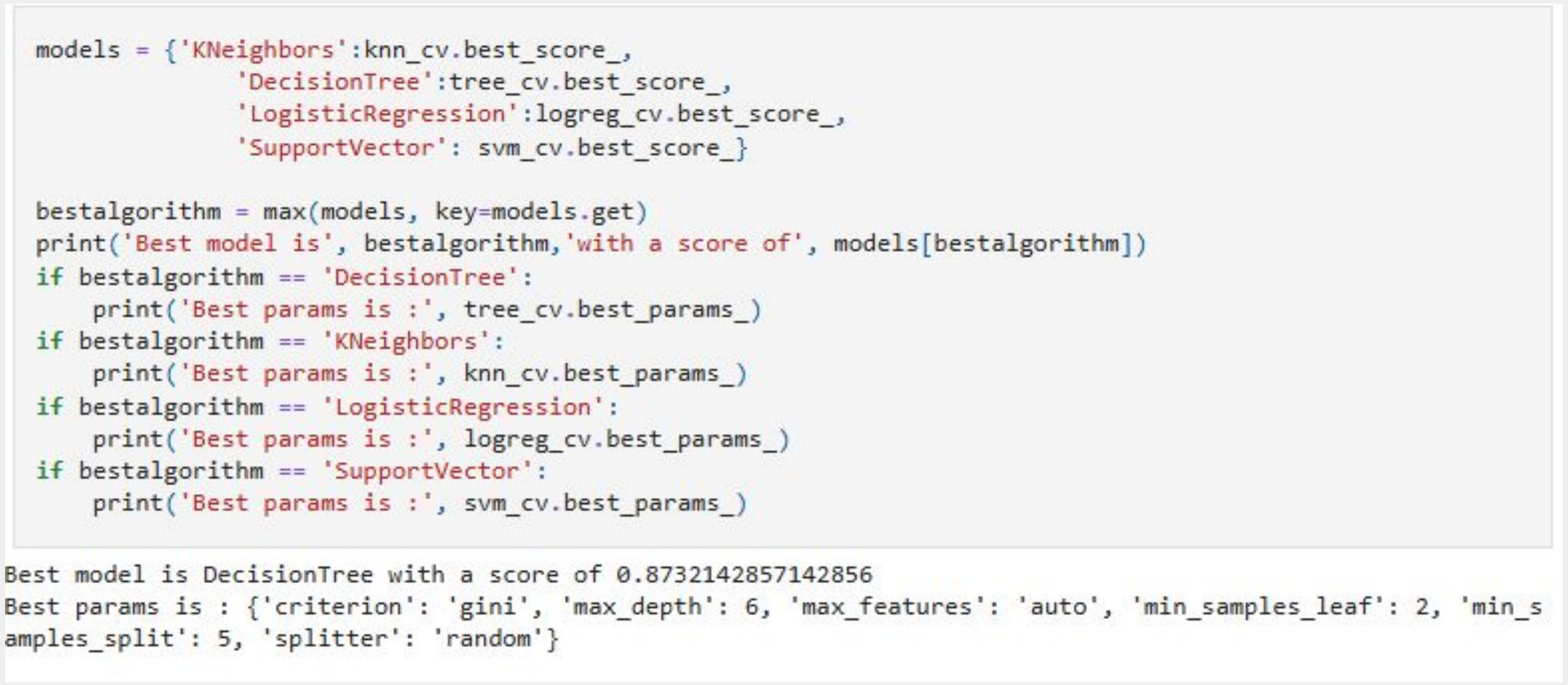


- Success rate is higher with payloads that are less than 4000 KG

Section 5

Predictive Analysis (Classification)

Classification Accuracy

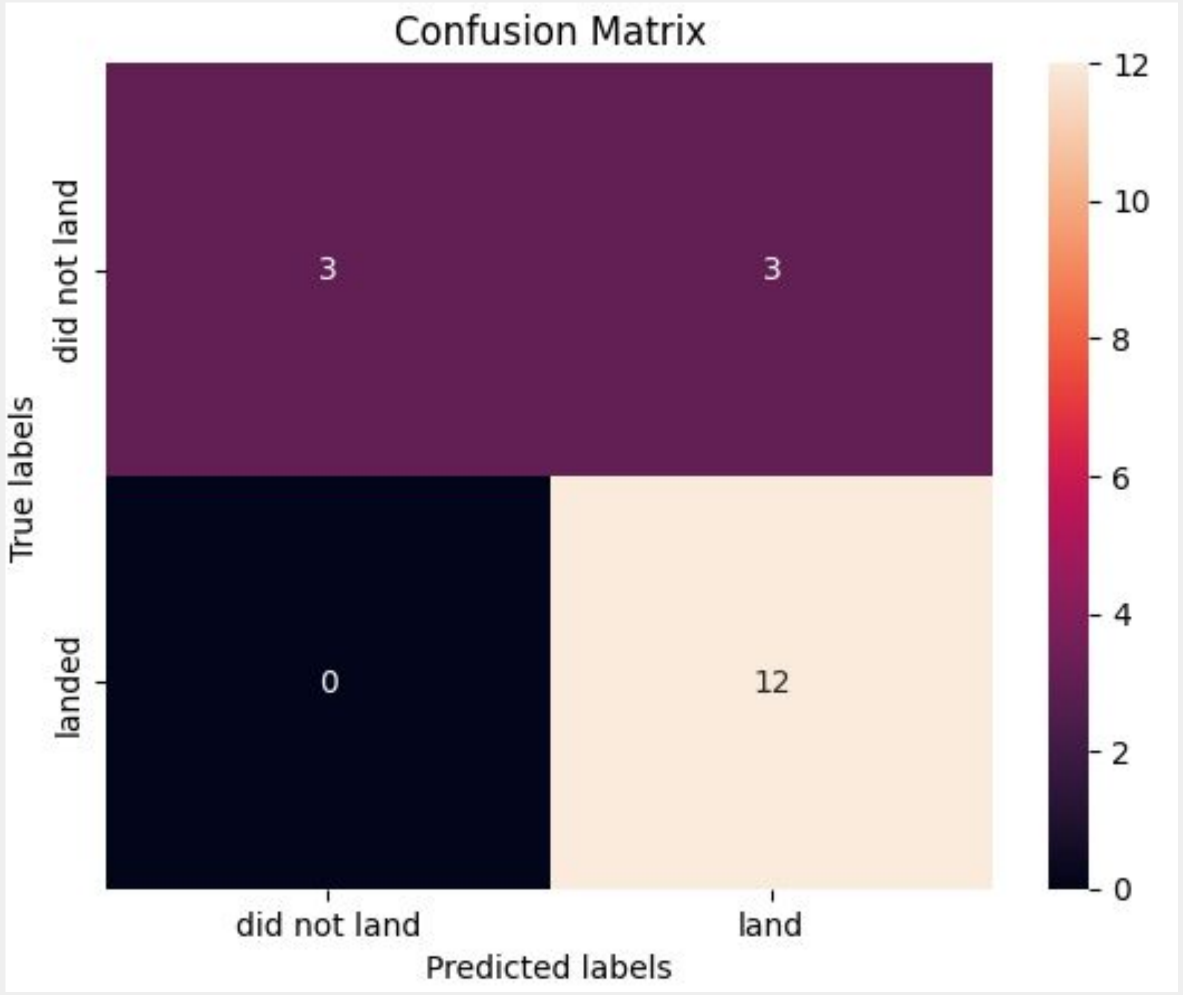


```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_s
amples_split': 5, 'splitter': 'random'}
```

- After multiple classification system tests, we were able to determine that the model with the highest classification accuracy is the….
- **Decision Tree, with a score of 0.8723**

Confusion Matrix



- Confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem with this is the false positives - an unsuccessful landing classifying as a successful landing

Conclusions

With the use of EDA, data visualization, SQL queries and classification models, we drew the following conclusions:

- Launch sites with greater flight amounts have better success rates
- Launch success rates started to increase starting in 2013, going to 2020
- Orbits GEO, ES-L1, SSO, HEO and VLEO had the most success rate.
- KSC LC-39A had the most successful launches compared to other sites
- The Decision Tree classifier is the best machine learning algorithm for this project - as it yields the highest classification accuracy

Appendix

- Github Notebook: https://github.com/pauldgayle/ibm-data-science-capstone-spacex/tree/main

Thank you!