IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Chang Hee Han>
<15 January 2022>

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Goal: by examining the launch data of SpaceX, determine whether it will be feasible to build a new company ‘Space Y’ to compete against SpaceX
- Objectives…
    1) Determine the price of each launch
    2) Decide whether to reuse the first stage rocket
    3) Find the best algorithm for predicting launch success

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Requesting data from SpaceX API using the requests package
  - Web scraping from Wikipedia using beautifulsoup and requests packages
- Performed data wrangling
  - Using Pandas and NumPy, cleaned and organized the data in various ways
- Performed exploratory data analysis (EDA) using visualization and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models
  - Using standardized/original data values, create various algorithms tuned for their hyperparmeters, and select the one that performs the best out of them all

Data Collection

- SpaceX provides an API that allows users to query and receive data related to their launches (api.spacexdata.com/v4). Using the API for data on past launches (api.spacexdata.com/v4/launches/past), the necessary data was obtained
- In addition to the API data, another source is helpful for cross-examining and validating the data. Wikipedia is a relatively reliable source of information, so information on the Falcon 9 page was scraped using beautifulsoup and requests packages

Data Collection – SpaceX API

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- GitHub URL: https://github.com/garliccat024/ads_capstone/blob/master/SpaceX%20data%20collection%20API.ipynb

1. Define necessary functions (e.g. getting the booster version and launch site, then appending the data)
2. Using the requests package, give the SpaceX API url to create a response object
3. Use the json_normalize on the json version of the response, then create a pandas dataframe from this result
4. From the original dataframe, create a new one with only the columns we need, and use the API again to append relevant information from ID-based data in the original dataframe
5. Limit the new dataframe to only Falcon 9 launches, eliminate null values, then export the data as a csv file for later use

Data Collection - Scraping

- Similar to the SpaceX API procedures, but with different procedures for processing html data using tags for navigation

- GitHub URL: https://github.com/garliccat024/ads_capstone/blob/master/Web%20scraping%20for%20data%20collection.ipynb

1. Define functions necessary for processing the beautifulsoup html data (e.g. searching for necessary information using tags and isolating the results)
2. Using provided static_url, scrape data using requests.get(), then create a beautifulsoup object
3. Using defined functions, create list of columns from table headers, then append dictionaries created with column names with relevant information
4. Using results from above, create a pandas dataframe, and export it to a csv file for future use

Data Wrangling

- Data were processed using Pandas and NumPy packages
  1. Import the data, and run some basic inspections (e.g. value_counts() of columns) to better understand what the components mean
  2. Create a new label that will help with labeling of each item (e.g. launch outcome)
  3. Save the new dataframe with new information in csv format, for future use

- GitHub URL: https://github.com/garliccat024/ads_capstone/blob/master/Data%20wrangling.ipynb

EDA with Data Visualization

- Visualization is often essential for the intuitive understanding and further manipulation of data, done using the seaborn package:
  - PayloadMass vs. FlightNumber scatterplot: color-coded for seeing success rate
  - Success rate vs. Orbit type bar graph: inspection of success rate for each launch site
  - More examples will be presented in later slides with visualizations
- GitHub URL: https://github.com/garliccat024/ads_capstone/blob/master/EDA%20with%20Visualization%20Lab.ipynb

EDA with SQL

- Using the sqlalchemy and ipython packages, use SQL to perform various manipulations on the data:
  - Select unique launch site names from SPACEXTBL
  - Select entries with conditionals (e.g. launch site begins with "CCA")
  - Perform operations on conditionals (e.g. sum, maximum, minimum, unique values)
  - Select multiple columns from conditionally selections, and present them neatly with new column names using SELECT-AS queries
- GitHub URL: https://github.com/garliccat024/ads_capstone/blob/master/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Using the Folium package, various objects were created and added to the map to represent various qualities of the dataset
  - Circles for launch sites: created using latitude/longitude coordinates
  - Marker clusters of launch sites: contains color-coded records of every launch and their results
  - Lines between launch sites and other locations: for measuring distances between locations
- GitHub URL: https://github.com/garliccat024/ads_capstone/blob/master/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- Using the dash and plotly packages, create interactive graphs so that the user can select for information they would like to see
  - Dropdown menu: for selecting input for plots
  - Range slider for selecting payloads of launches
  - Pie-chart for representing ratio of launch sites
  - Scatterplot of launch sites and payloads
- GitHub URL: https://github.com/garliccat024/ads_capstone/blob/master/spacex_dash_app.py

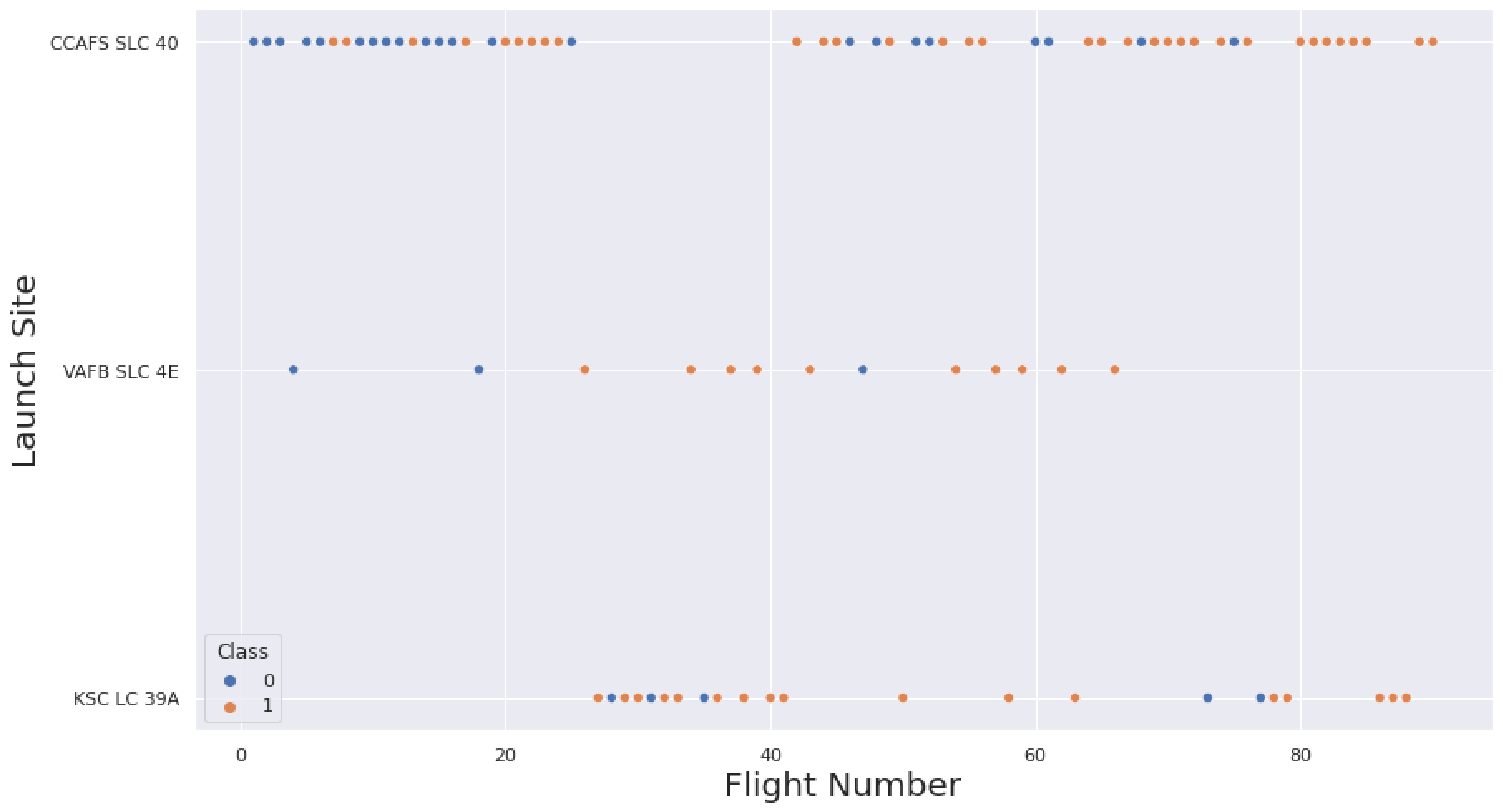
Predictive Analysis (Classification)

- In order to have multiple candidates for the final selection process, create various models for training and testing
  - GridSearchCV, Logistic Regression, SVC, Decision Tree, and K-Nearest Neighbors
- After training and testing the models with various parameters, compare their performance (accuracy) with each other, before selecting the best model for the dataset
- GitHub URL: https://github.com/garliccat024/ads_capstone/blob/master/Machine%20Learning%20Prediction.ipynb

Section 2

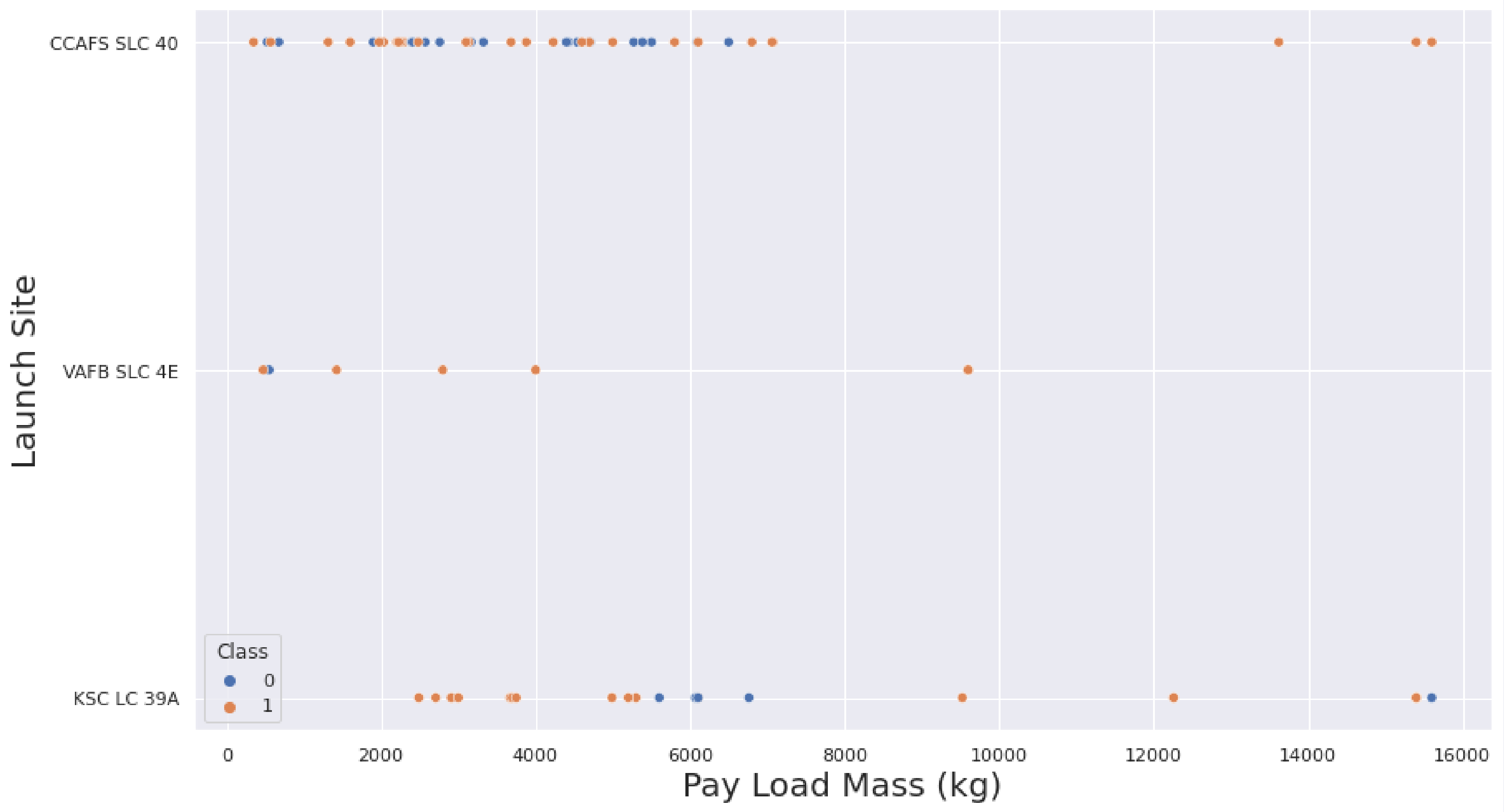
Insights drawn from EDA

Flight Number vs. Launch Site



- Scatterplot showing the distribution of flights among different launch sites, and whether each launch was successful or not

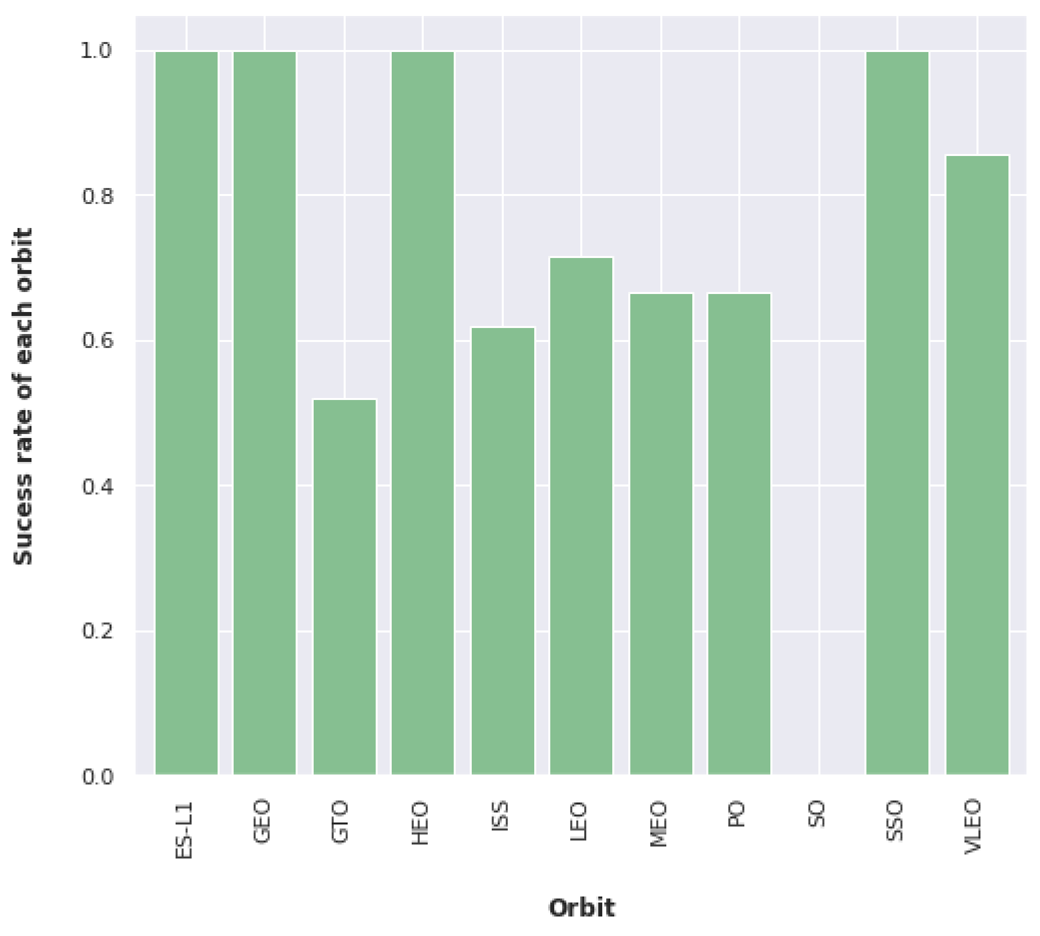
Payload vs. Launch Site



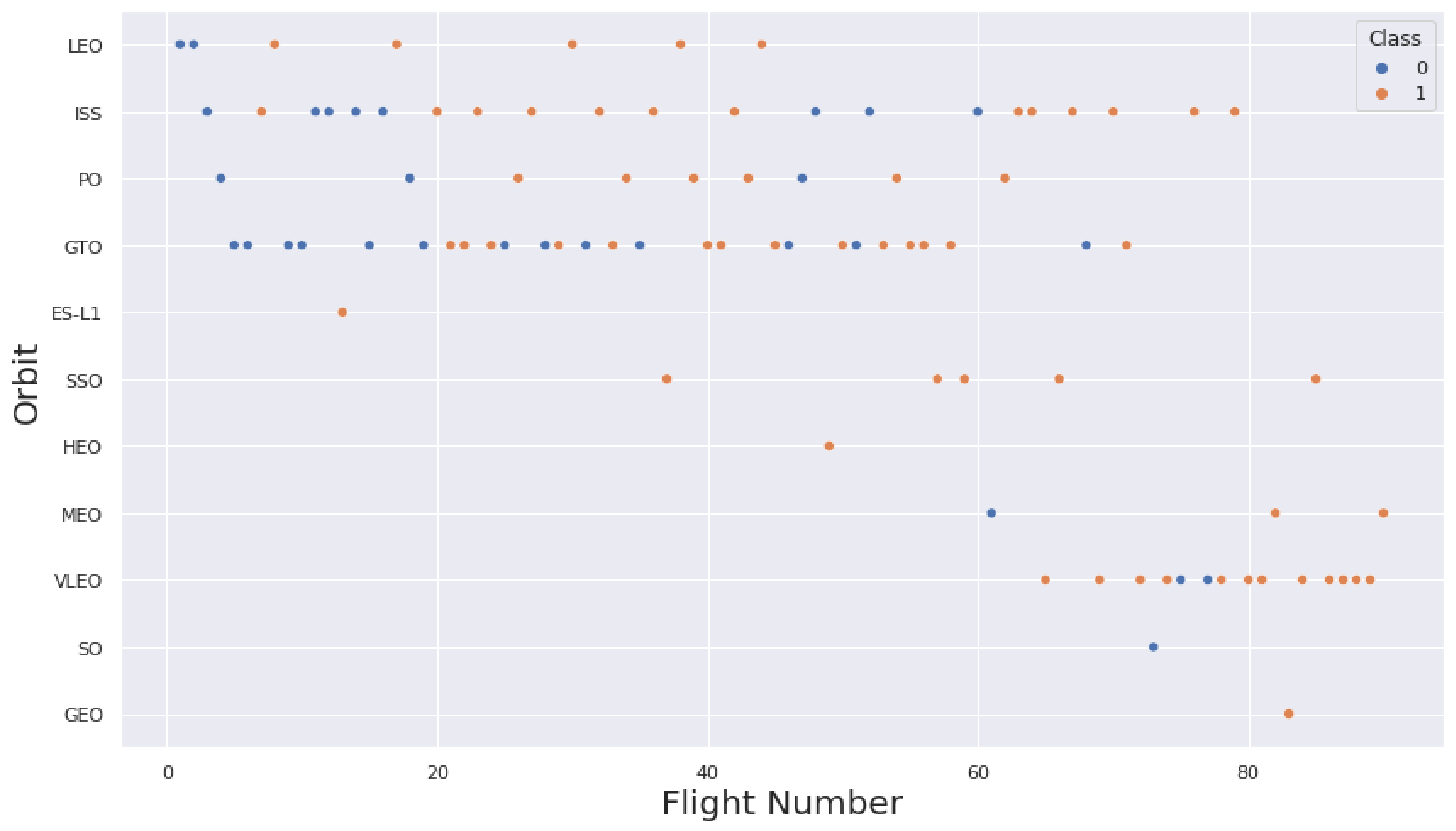
- Scatterplot showing the distribution of payload mass for each launch site, with color coding that indicates whether each launch was successful or not

Success Rate vs. Orbit Type

- Bar graph with success rates of each orbit type

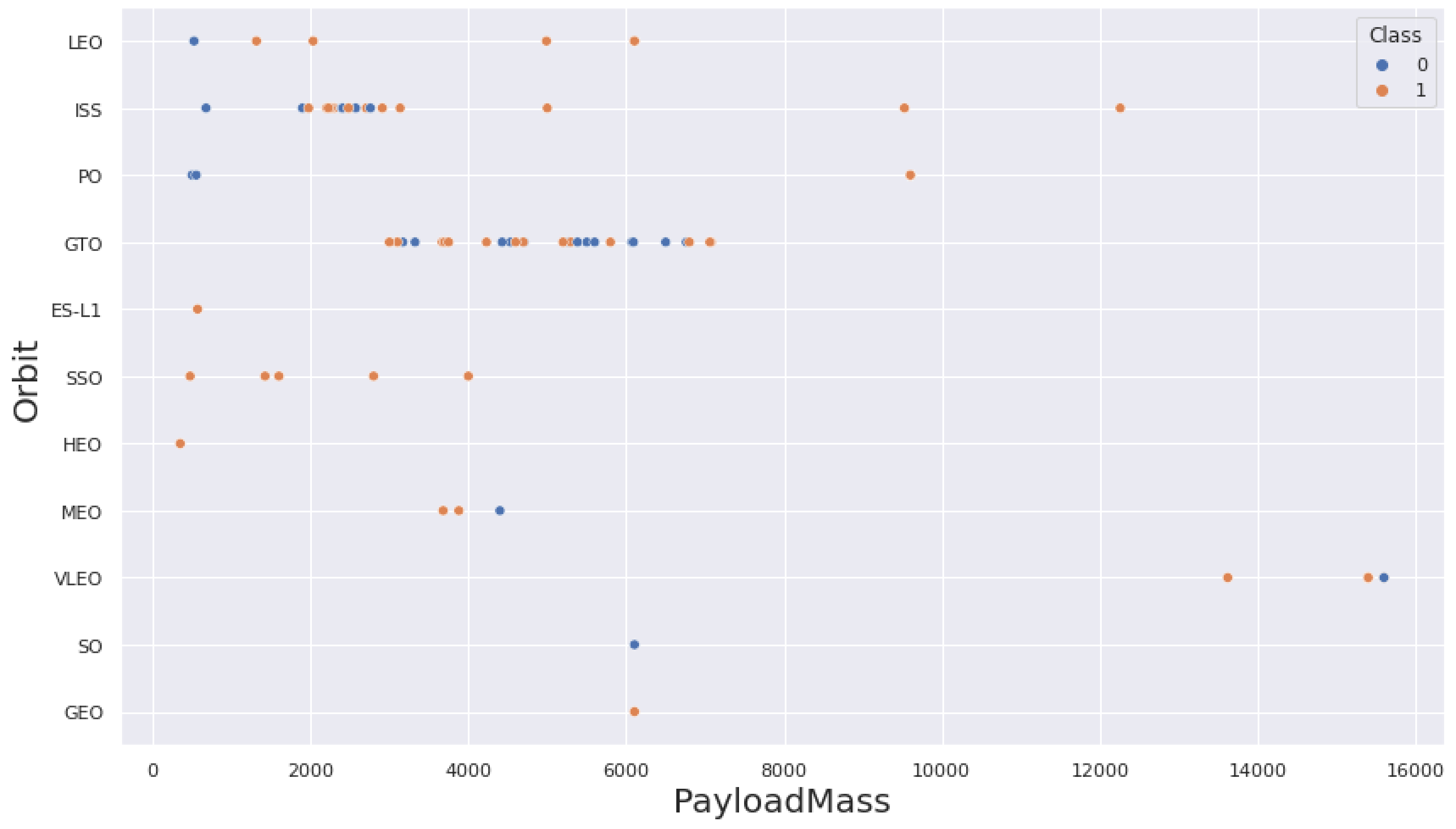


Flight Number vs. Orbit Type



- Scatterplot of launch distributions for different types of orbit, with color-coding for success of each launch

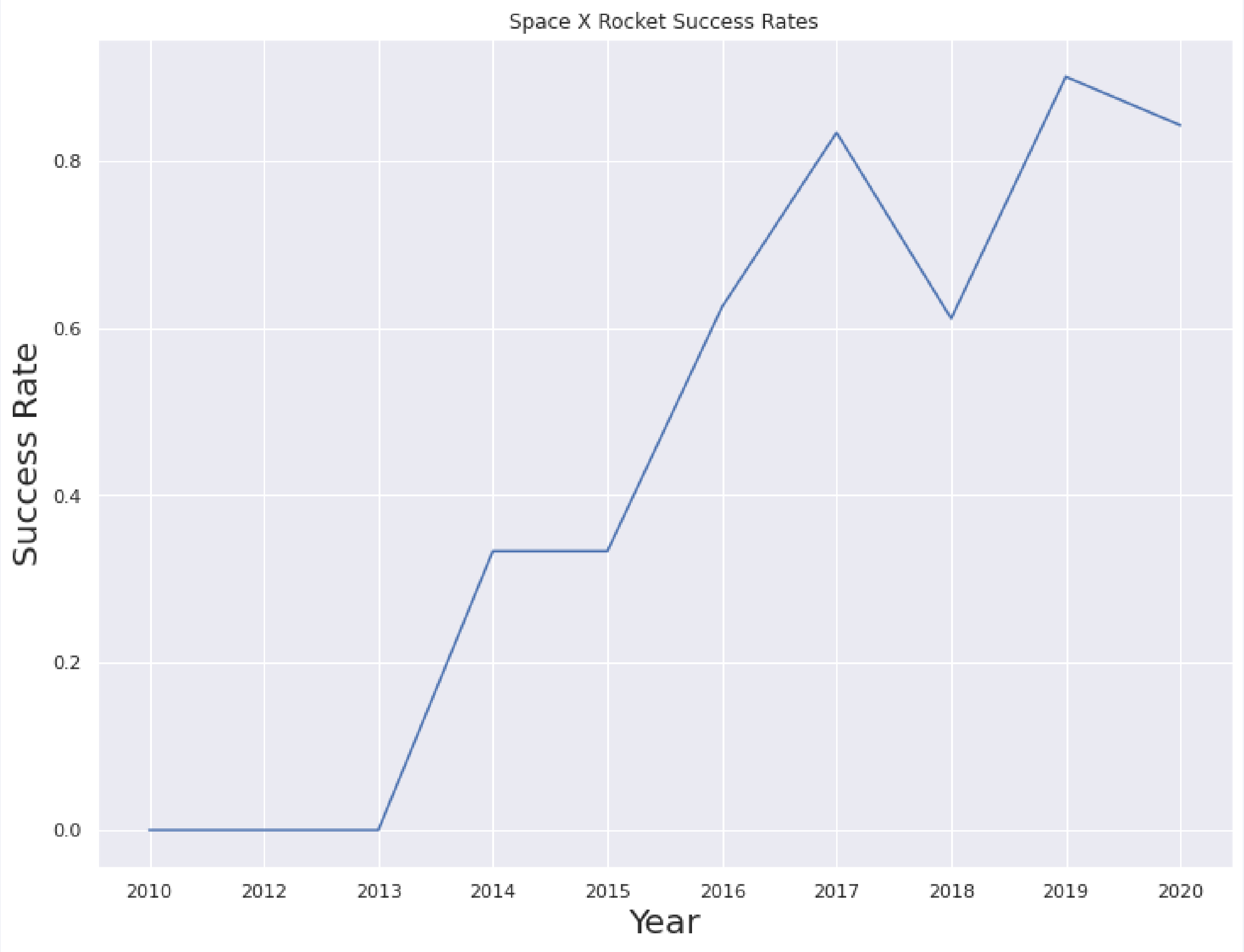
Payload vs. Orbit Type



- Scatterplot that shows distribution of launches for each orbit type based on the payload, with color coding for success

Launch Success Yearly Trend

- Line chart of yearly average success rate, which shows a generally increasing trend



All Launch Site Names

- Selecting unique launch sites from table using DISTINCT

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

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

Launch Site Names Begin with 'CCA'

- Selecting launch site names with LIKE and wildcard(%)

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

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
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	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

- Total payload carried by boosters from NASA using WHERE to select customer name

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

Total Payload Mass by NASA (CRS)
45596

Average Payload Mass by F9 v1.1

- Average payload mass carried by booster version F9 v1.1, using AVG

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM SPACEXTBL \
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

Average Payload Mass by Booster Version F9 v1.1
2928

First Successful Ground Landing Date

- Date of the first successful landing outcome on ground pad, using WHERE to select the condition

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

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

First Succesful Landing Outcome in Ground Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000, using conditionals

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

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/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

- Total number of successful and failure mission outcomes, using SELECT-AS to organize the outcome

```
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", ₩
    sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" ₩
FROM SPACEXTBL;
```

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

Successful Mission	Failure Mission
100	1

Boosters Carried Maximum Payload

- Names of the boosters which have carried the maximum payload mass

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEXTBL ₩
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

- Failed landing_outcomes in drone ship, their booster versions, and launch site names for the year 2015

```
%sql SELECT {fn MONTHNAME(DATE)} as "Month", BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE year(DATE) = '2015' AND ₩
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/bludb
Done.
```

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

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

- Count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, ranked in descending order

```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEXTBL ₩
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' ₩
GROUP BY  LANDING__OUTCOME ₩
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://bjh14620:***@125f9f61-9715-46f9-9399-c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30426/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

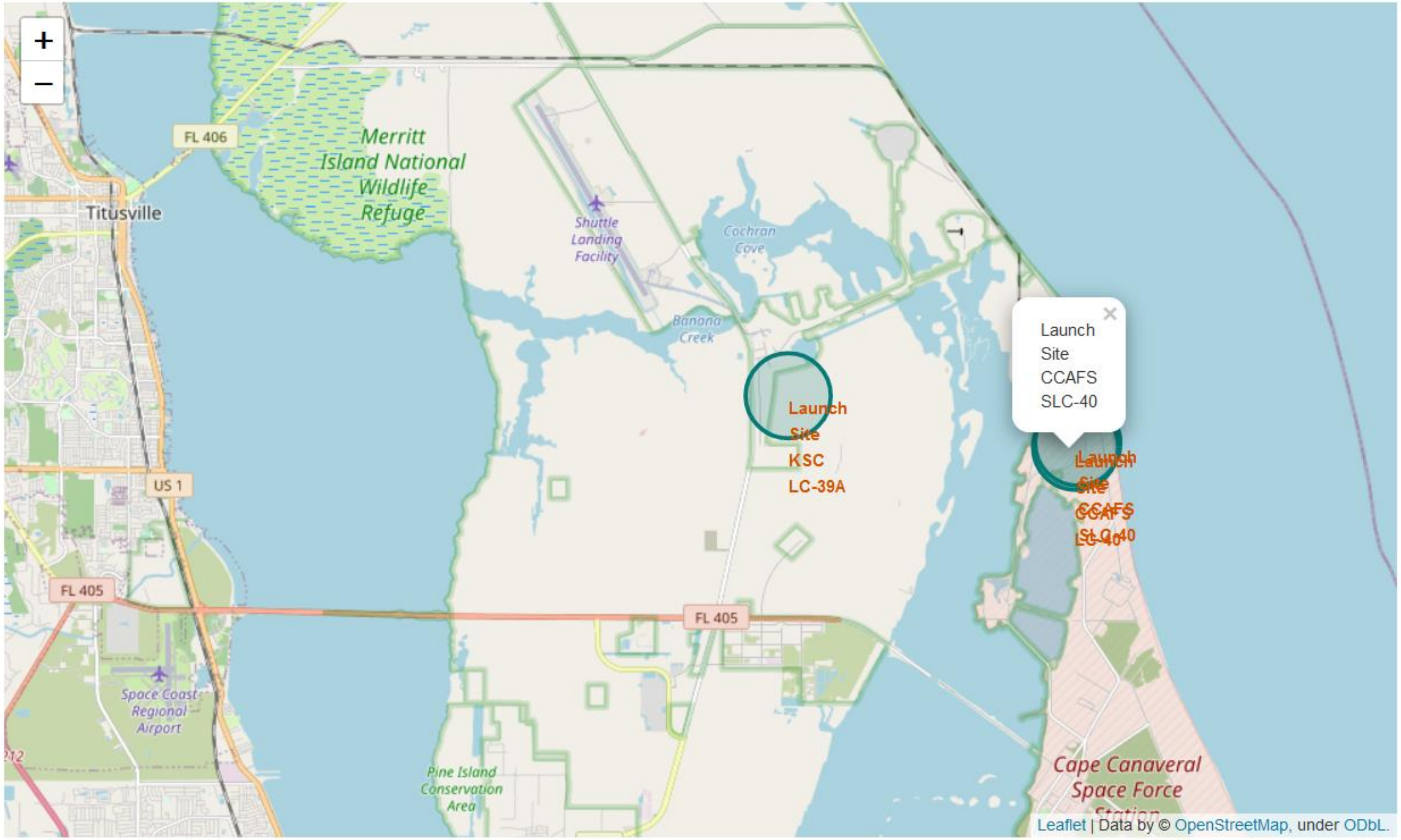
Section 4

Launch Sites Proximities Analysis

Folium – Launch Sites' Location Markers

- Folium objects with popups marking where the launch sites are

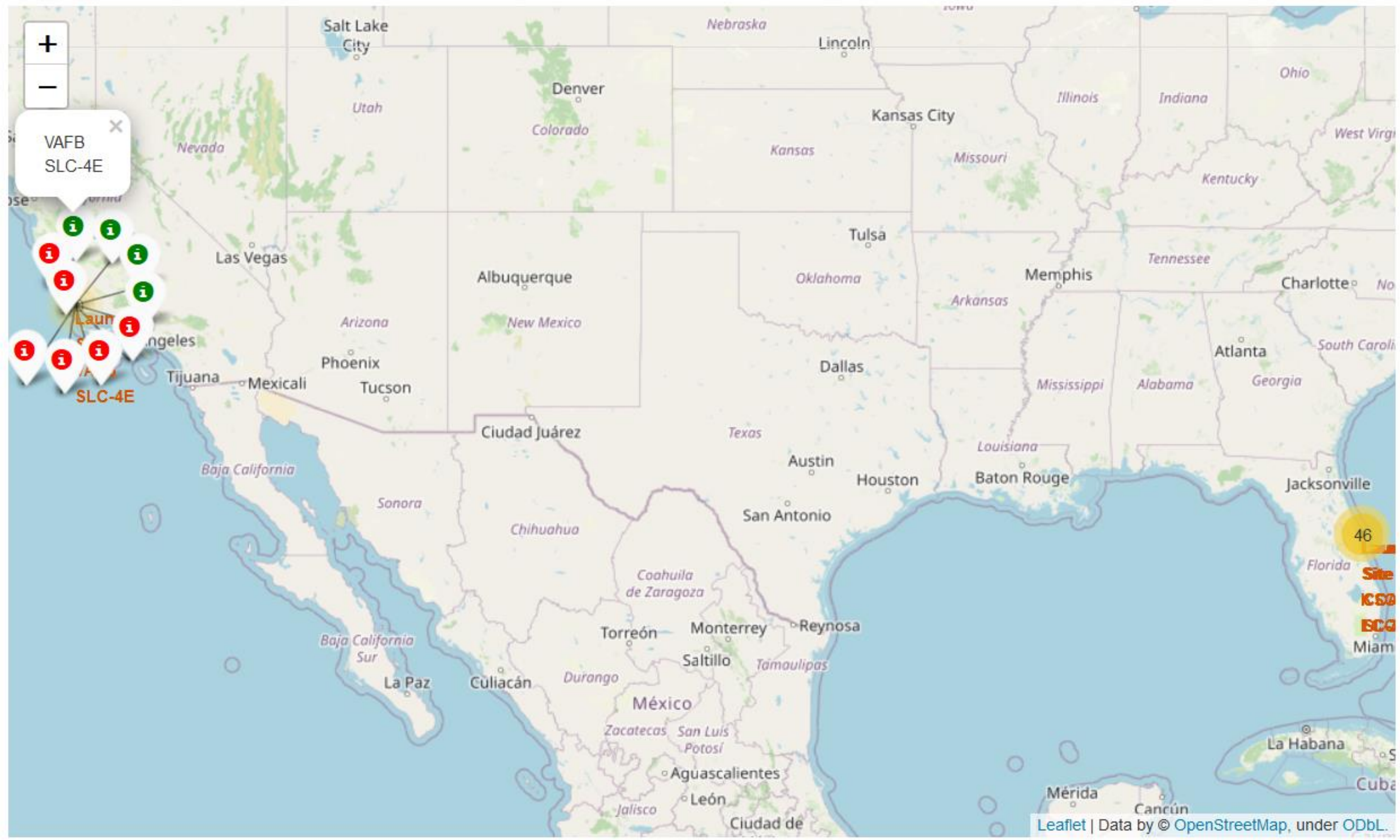
Out[8]:



Folium – Color-labeled Launch Outcomes

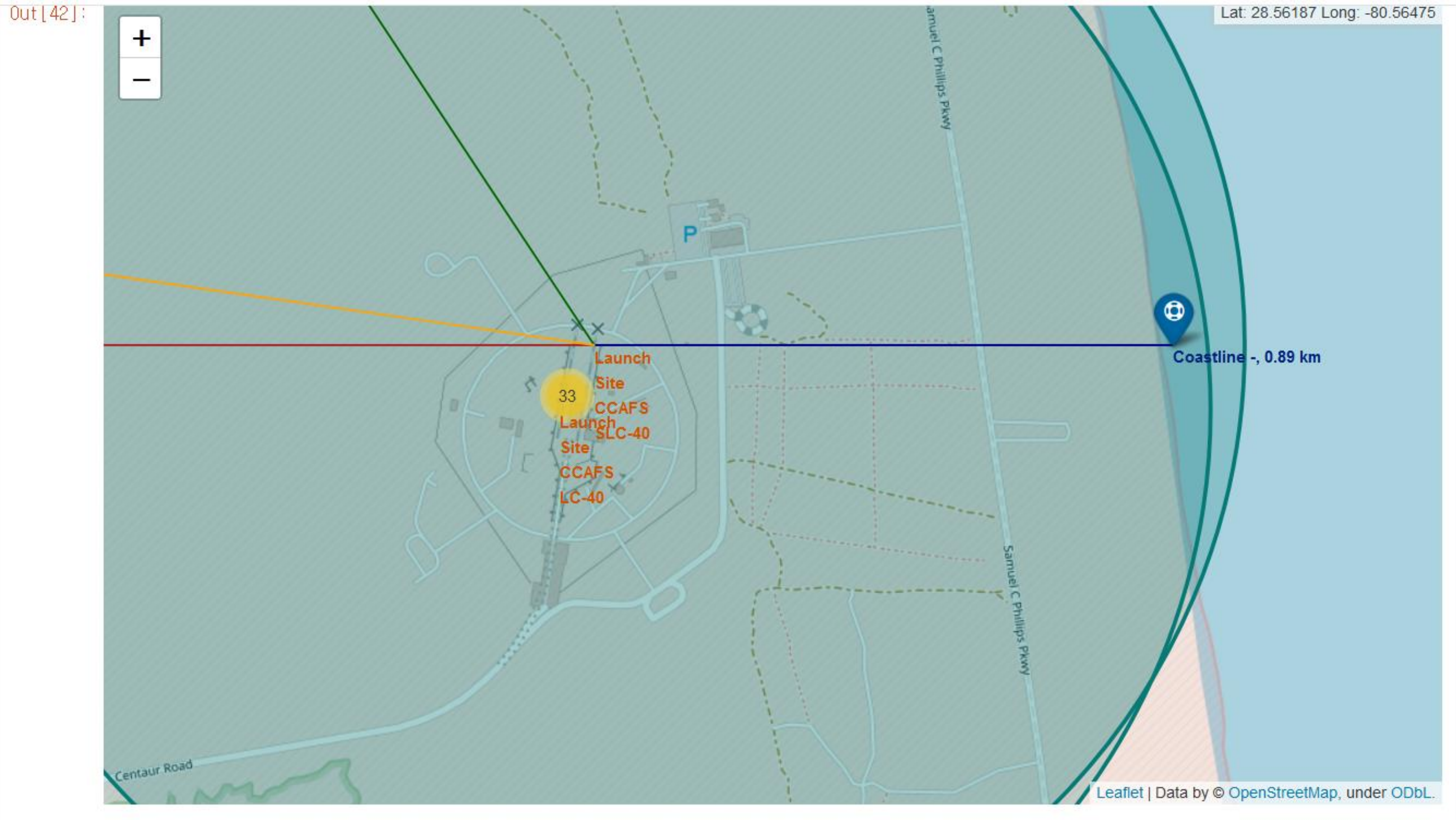
- Launch sites with marker clusters to show success of each launch

Out[14]:



Folium – Lines with Distance Labels

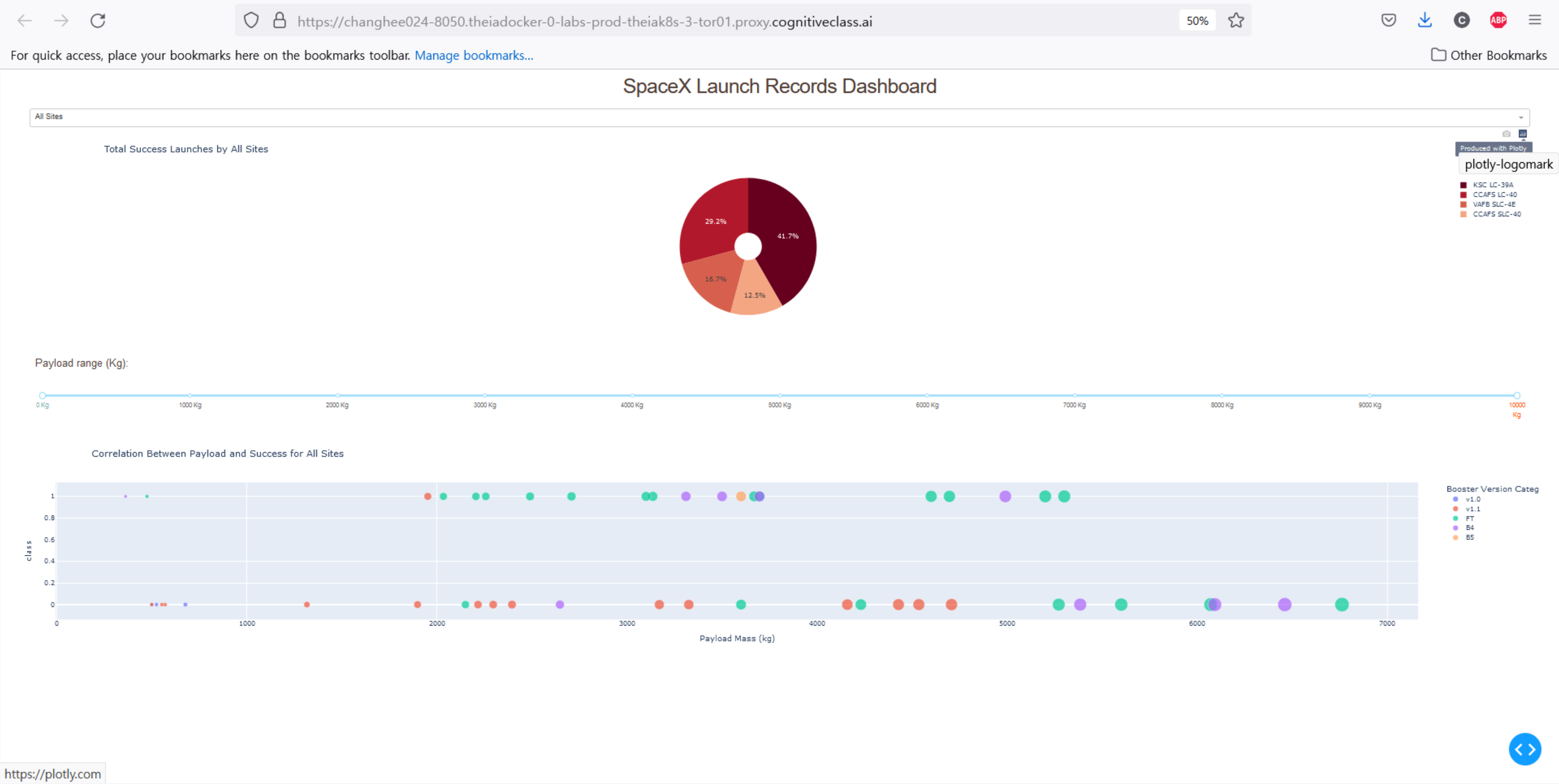
- Line objects to designate distance between target and launch site



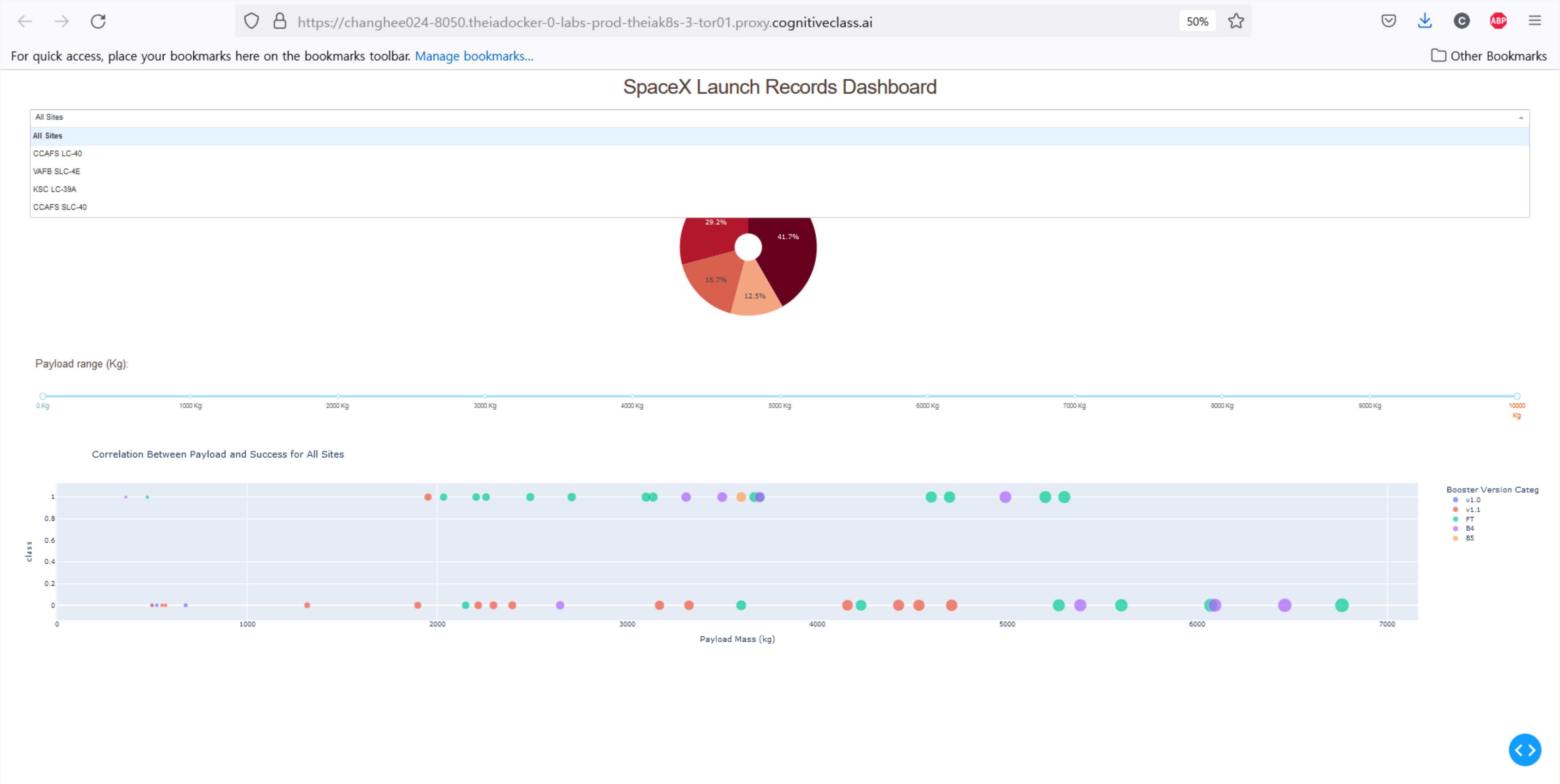
Section 5

Build a Dashboard with Plotly Dash

Dashboard Screenshot – Main Screen with Plots



Dashboard Screenshot – Dropdown for Input Options



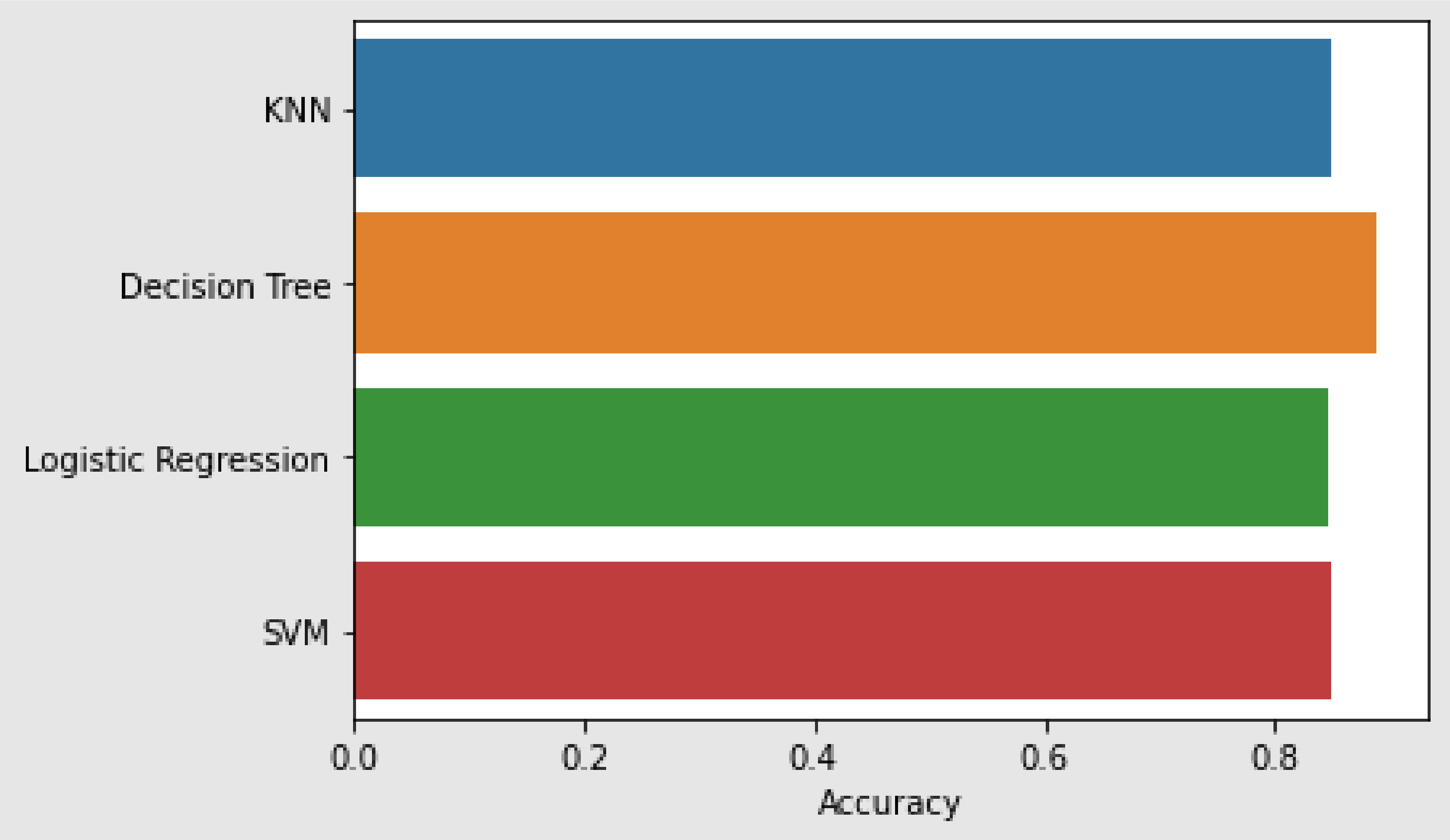
Section 6

Predictive Analysis (Classification)

Classification Accuracy

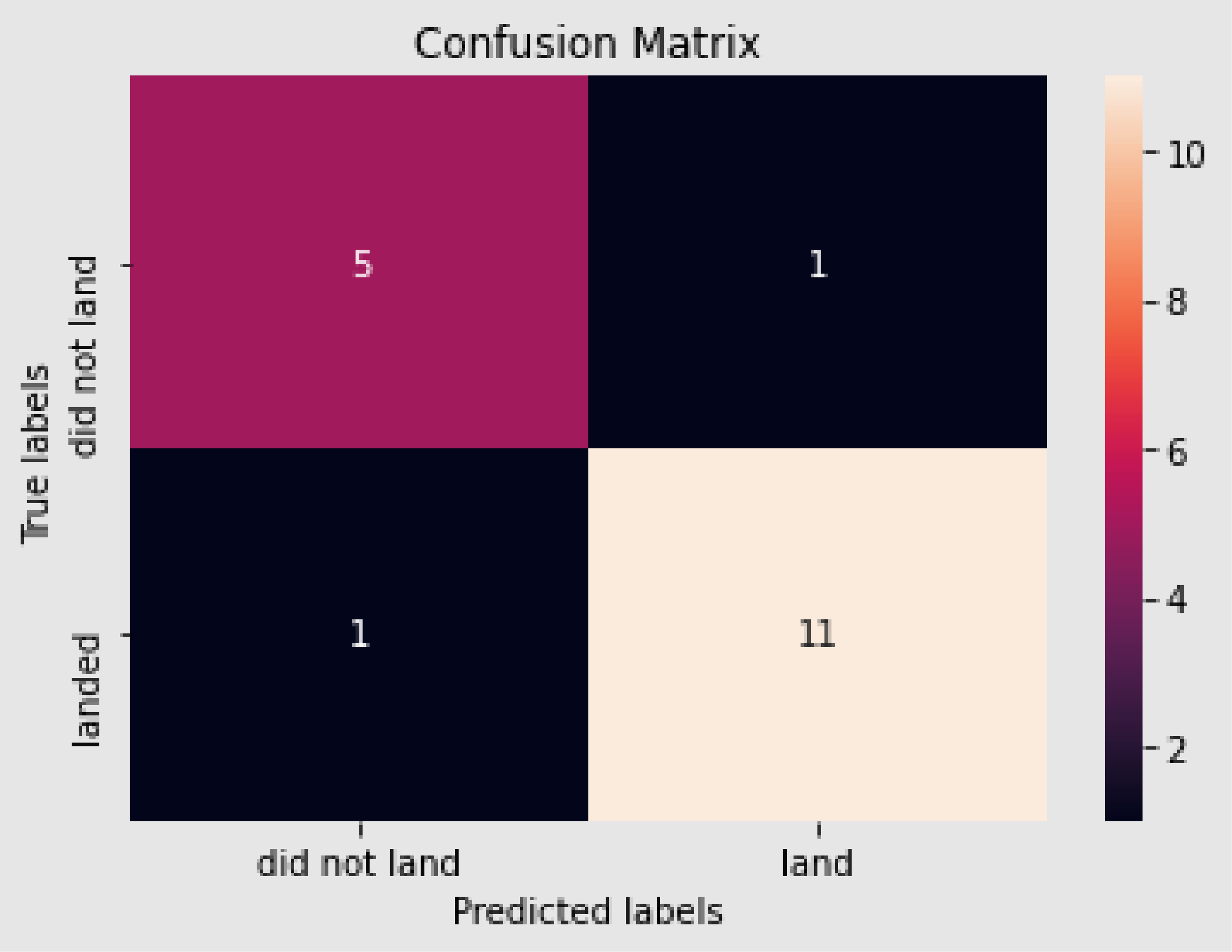
- Decision Tree has highest accuracy of all models used for this project at 88.75%

	Accuracy
KNN	0.848214
Decision Tree	0.887500
Logistic Regression	0.846429
SVM	0.848214



Confusion Matrix

- Number of samples is too small
- The diagonal with 5 and 11 in them shows accurate predictions (those that match the actual outcomes)



Conclusions

- For future analysis of SpaceX launch data, the Decision Tree can be the best ML algorithm to apply for achieving best results
- Running the ML procedures with larger datasets can improve accuracy
- With better inspection of launch data, it will be easier to establish the cost-benefit and business model for the hypothetical competitor to SpaceX: “Space Y”

Thank you!