Winning Space Race with Data Science

Aleksandr Migunov
Nov 30, 2021

Outline

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

Executive Summary

- Summary of methodologies
- Summary of all results

Introduction

- Project background and context
- Problems you want to find answers

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:

Data was collected through request to the SpaceX API. After that, data was cleansed.

- Perform data wrangling

- Data was wrangled using BeautifulSoup for extracting HTML table and then converted into data frame.

Methodology

Executive Summary

Perform exploratory data analysis (EDA) using visualization and SQL

- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Data was standardized, split into training set and test set, fitted, and tested using four models. Then, the best model with the best parameters was chosen.

Data Collection

Describe how data sets were collected.

You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

Request and parse the SpaceX launch data using the GET request

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

response = requests.get(spacex_url)

GitHub link:
https://github.com/AleksandrMigunov/ds-capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

1. Request the Falcon9 Launch Wiki page from its URL

response = requests.get(static_url)

soup = BeautifulSoup(response.content)

2. Extract all column/variable names from the HTML table header

3. Create a data frame by parsing the launch HTML tables

GitHub link: https://github.com/AleksandrMigunov/ds-capstone/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

1. Calculate the number of launches on each site
2. Calculate the number and occurrence of each orbit
3. Calculate the number and occurrence of mission outcome per orbit type
4. Create a landing outcome label from Outcome column

GitHub link: https://github.com/AleksandrMigunov/ds-capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

1. Visualize the relationship between Flight Number and Launch Site
2. Visualize the relationship between Payload and Launch Site
3. Visualize the relationship between success rate of each orbit type
4. Visualize the relationship between FlightNumber and Orbit type
5. Visualize the relationship between Payload and Orbit type
6. Visualize the launch success yearly trend

GitHub link: https://github.com/AleksandrMigunov/ds-capstone/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Connect to the database
- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved.

EDA with SQL

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

- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass.
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- 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

EDA with SQL

- GitHub link: https://github.com/AleksandrMigunov/ds-capstone/blob/main/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

1. Mark all launch sites on a map
2. Mark the success/failed launches for each site on the map
3. Calculate the distances between a launch site to its proximities

GitHub link:
https://github.com/AleksandrMigunov/ds-capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

1. Add a dropdown list to enable Launch Site selection
2. Add a pie chart to show the total successful launches count for all sites
3. Add a slider to select payload range
4. Add a scatter chart to show the correlation between payload and launch success
5. Add a callback function for `site-dropdown` as input, `success-pie-chart` as output
6. Add a callback function for `site-dropdown` and `payload-slider` as inputs, `success-payload-scatter-chart` as output

Build a Dashboard with Plotly Dash

GitHub link:
https://github.com/AleksandrMigunov/ds-capstone/blob/main/spacex_dash_app.py

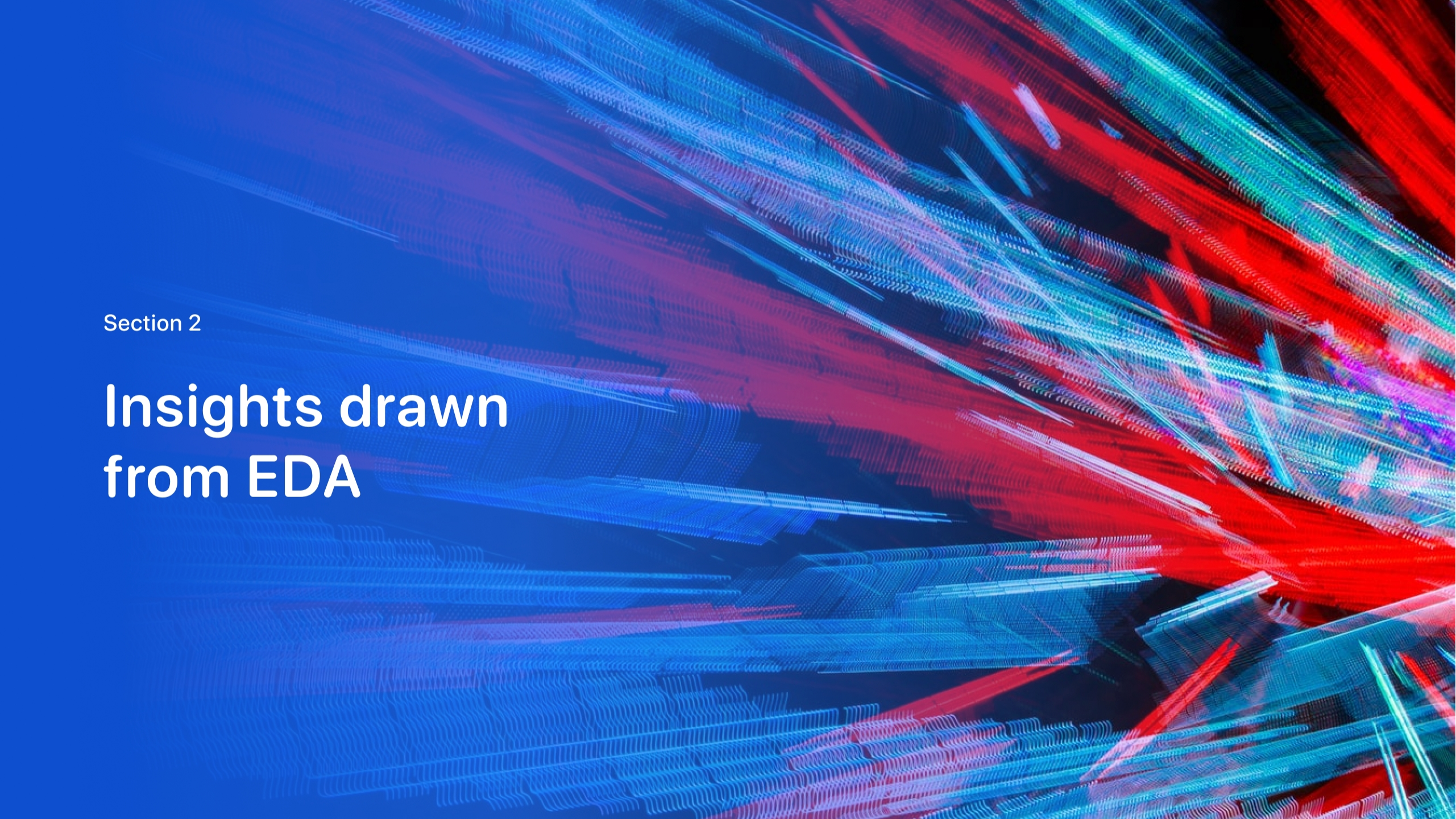
Predictive Analysis (Classification)

1. Create a NumPy array from the column Class in data
2. Standardize the data in X then reassign it to the variable X
3. Use the function train_test_split to split the data X and Y into training and test data
4. Create a logistic regression object then create a GridSearchCV object
5. Calculate the accuracy on the test data using the method score
6. Create a support vector machine object then create a GridSearchCV object
7. Calculate the accuracy on the test data using the method score
8. Create a decision tree classifier object then create a GridSearchCV object
9. Calculate the accuracy of tree_cv on the test data using the method score:
10. Create a k nearest neighbors object then create a GridSearchCV
11. Calculate the accuracy of tree_cv on the test data using the method score
12. Find the method performs best

GitHub link: https://github.com/AleksandrMigunov/ds-capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

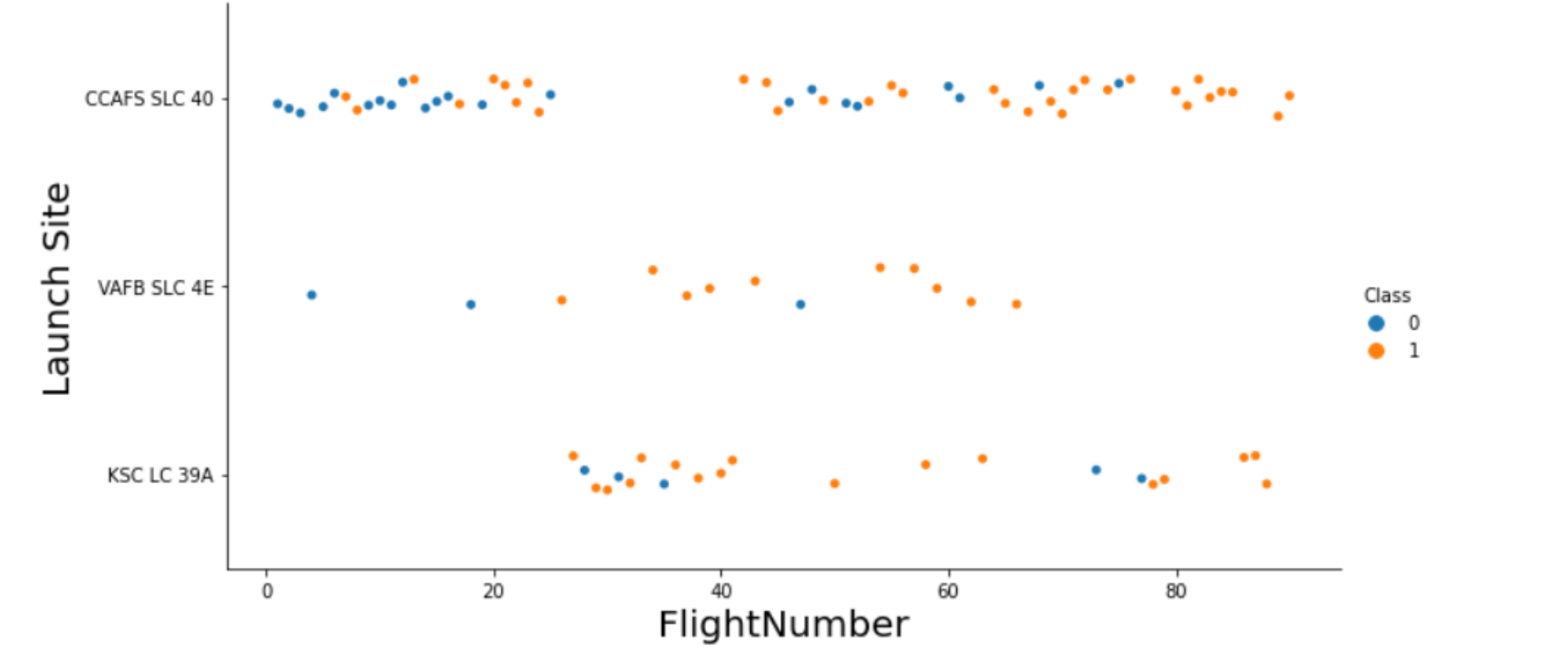
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



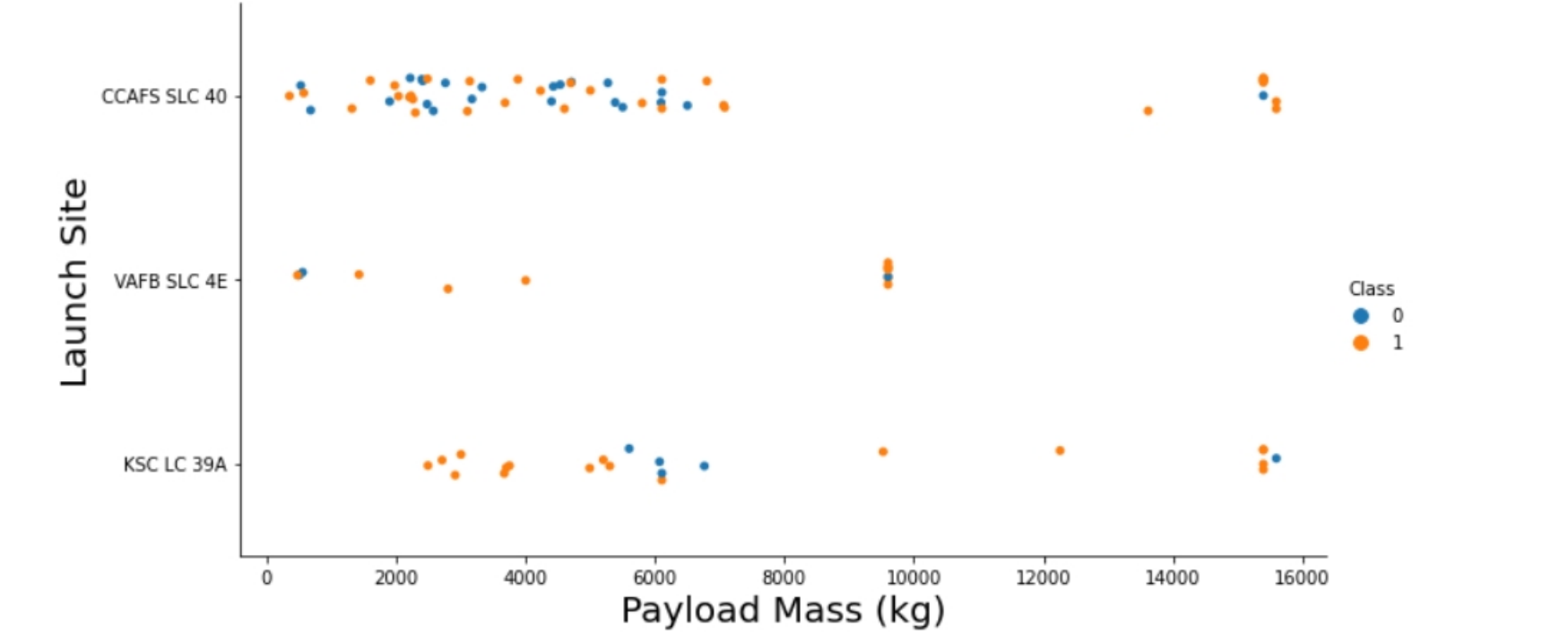
Section 2

Insights drawn from EDA

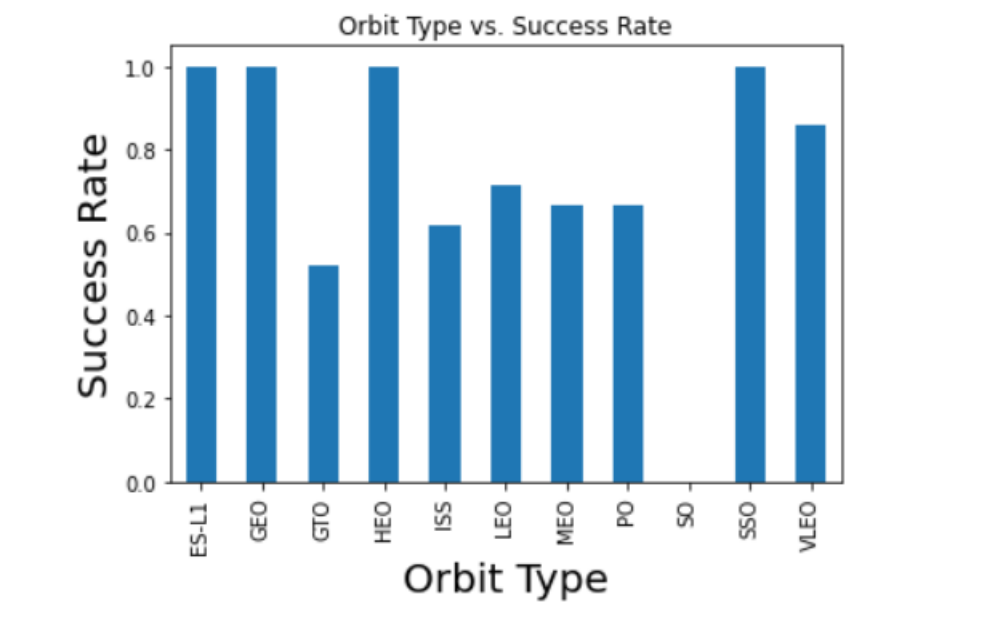
Flight Number vs. Launch Site



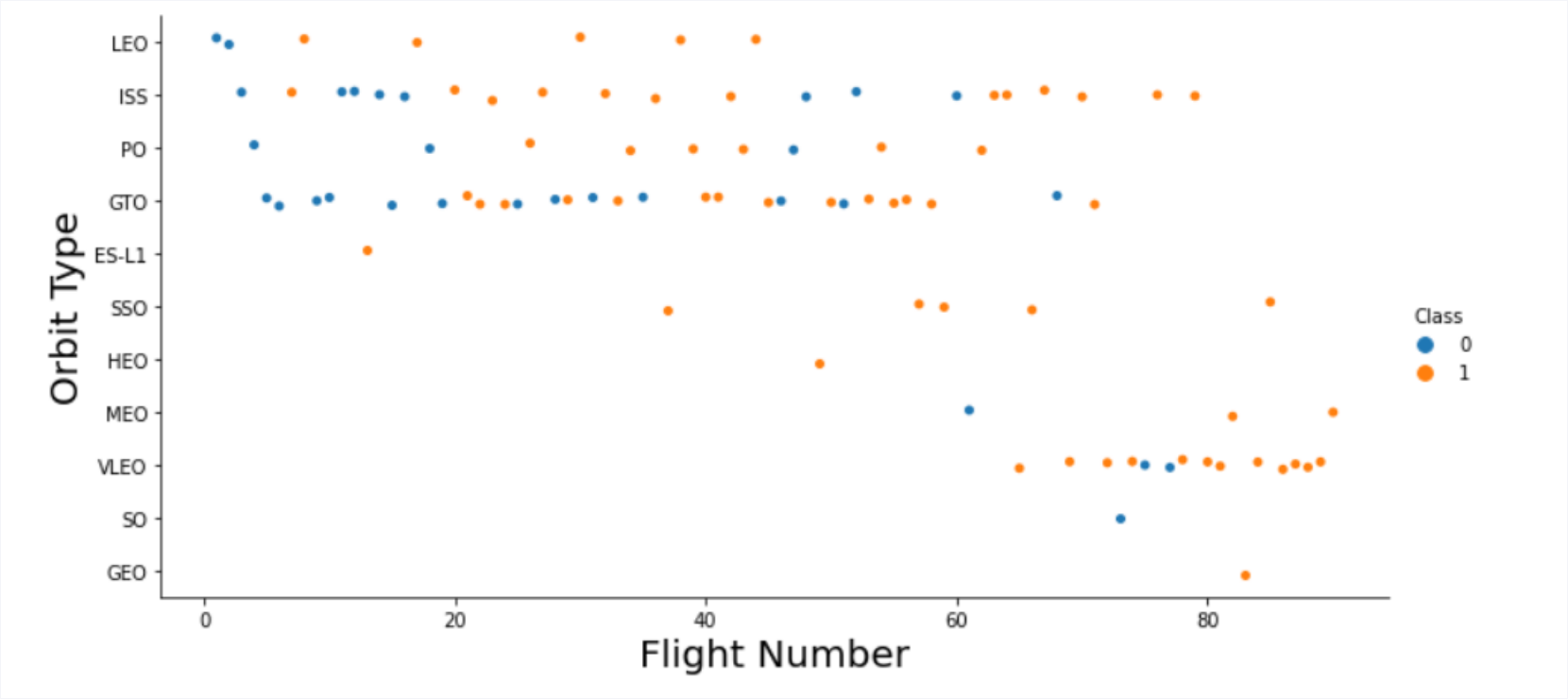
Payload vs. Launch Site



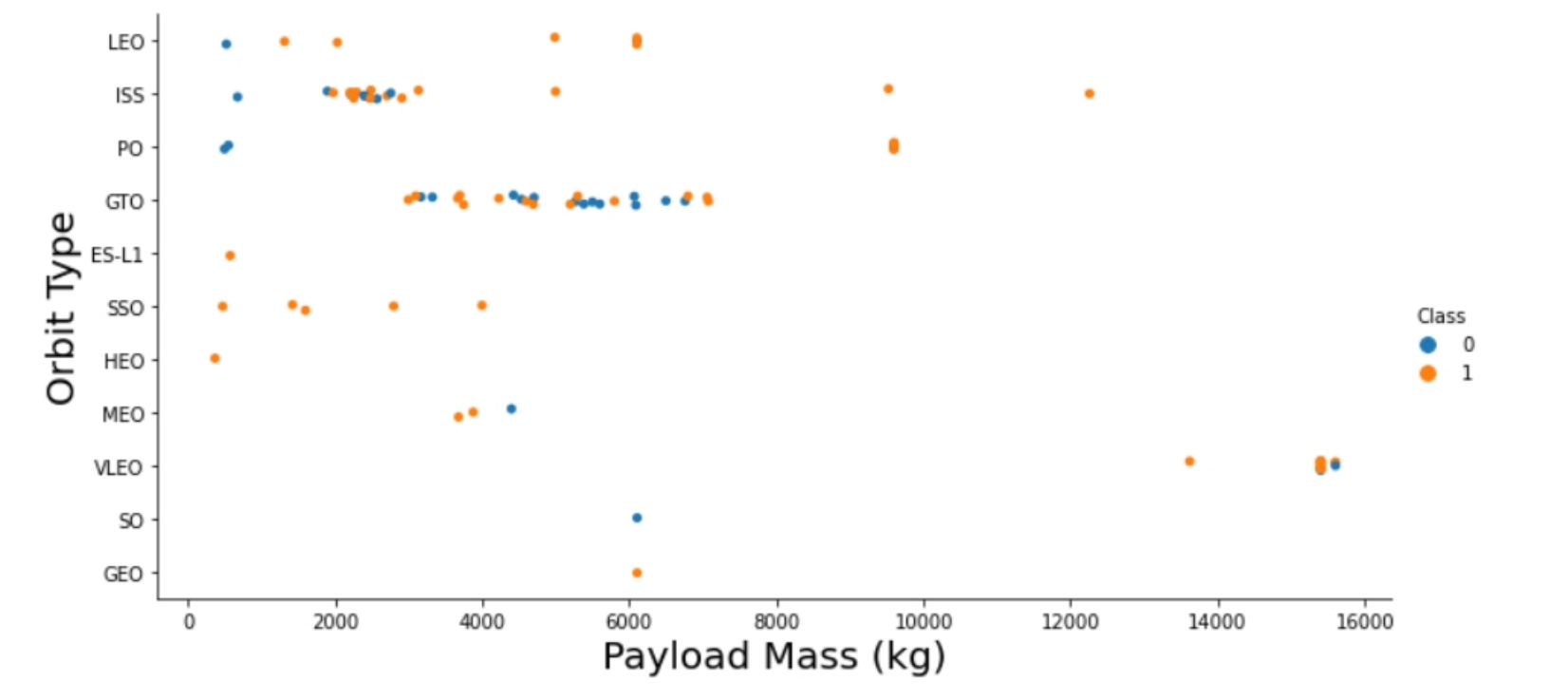
Success Rate vs. Orbit Type



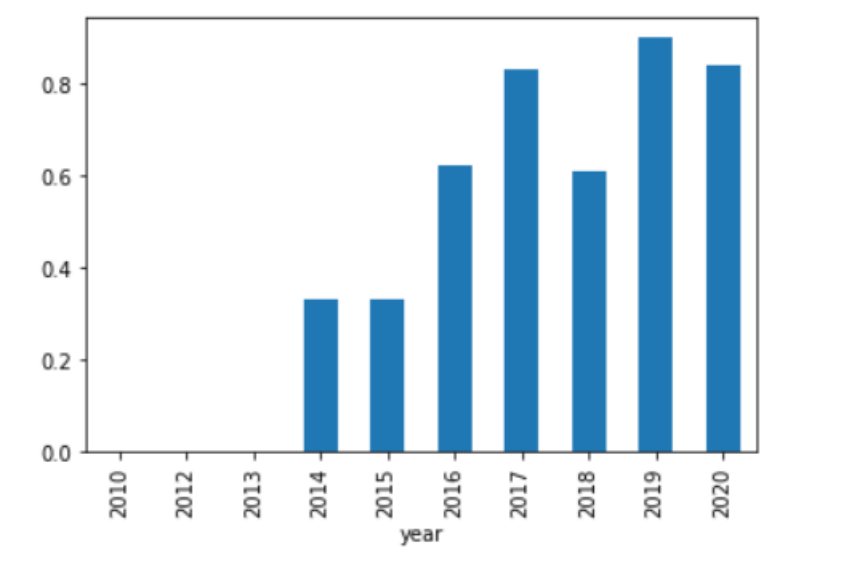
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

Display the names of the unique launch sites in the space mission

In [4]:
```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTBL
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

Out[4]:

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

Launch Site Names Begin with 'CCA'

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

In [5]:
```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

Out[5]:

column_0	DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	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)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

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

In [6]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer='NASA (CRS)';
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

Out[6]:

1
45596

Average Payload Mass by F9 v1.1

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

In [7]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version='F9 v1.1'
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

Out[7]:

1
2928

First Successful Ground Landing Date

List the date when the first successful landing outcome in ground pad was achelved.

Hint:Use min function

In [8]:
```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE Landing__Outcome='Success (ground pad)'
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

Out[8]:

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

In [9]:
```
%sql SELECT Booster_Version FROM SPACEXTBL \
WHERE Landing__Outcome='Success (drone ship)' AND (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000)
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

Out[9]:

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

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [10]:
```
%sql SELECT COUNT(*) FROM SPACEXTBL WHERE (Mission_Outcome LIKE 'Success%' OR Mission_Outcome LIKE 'Failure%')
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

Out[10]:

1
101

Boosters Carried Maximum Payload

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [11]:
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ IN (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

Out[11]:

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

2015 Launch Records

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

In [12]:
```
%sql SELECT Landing__Outcome, Booster_Version, Launch_Site FROM SPACEXTBL \
WHERE (Landing__Outcome='Failure (drone ship)' AND Date LIKE '2015%')
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

Out[12]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

In [13]:
```
%sql SELECT Landing__Outcome, COUNT(Landing__Outcome) \
FROM SPACEXTBL \
GROUP BY Landing__Outcome \
ORDER BY COUNT(Landing__Outcome) DESC;
```

```
 * ibm_db_sa://mnx90803:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/BLUDB
Done.
```

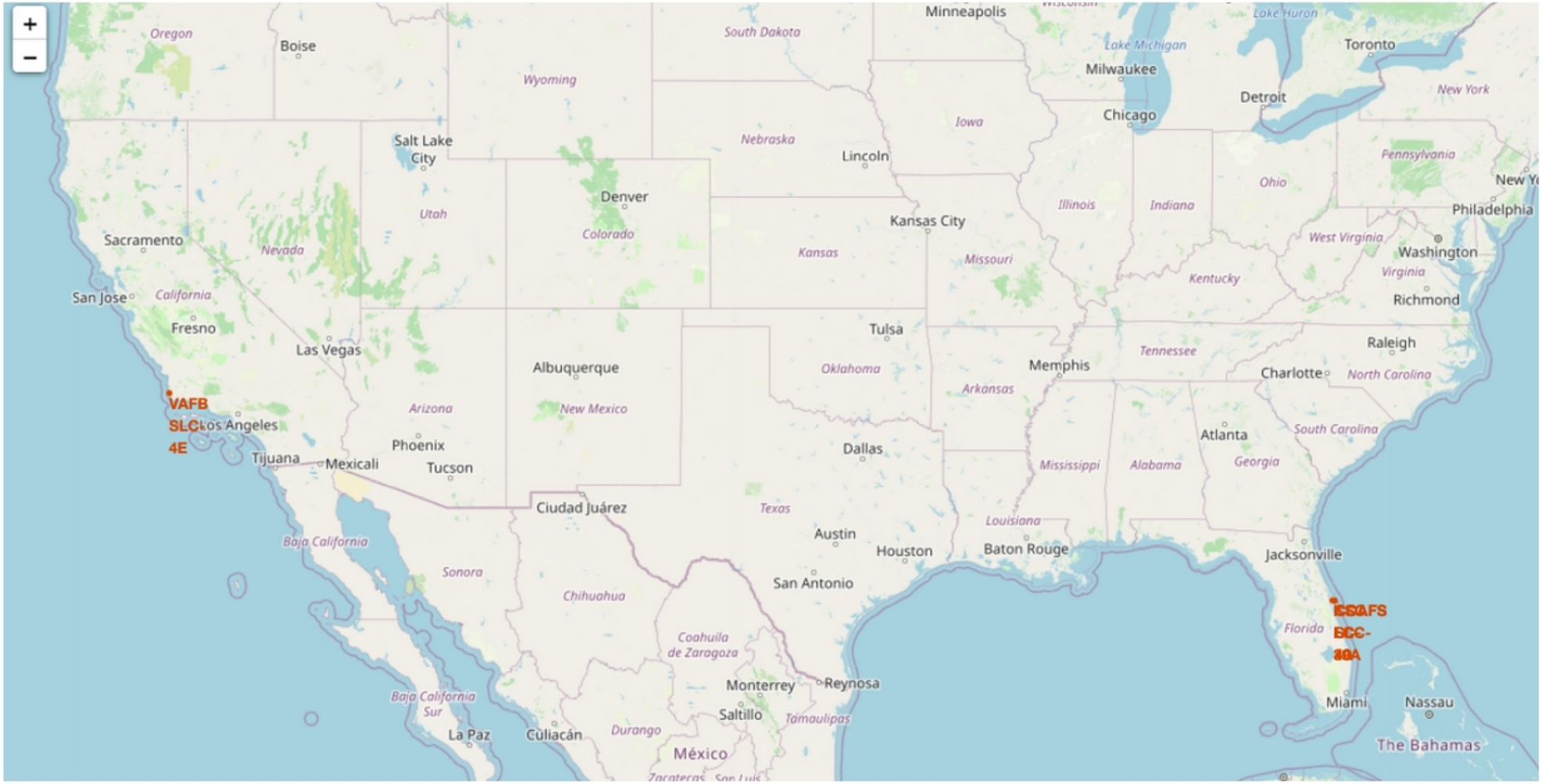
Out[13]:

landing__outcome	2
Success	38
No attempt	22
Success (drone ship)	14
Success (ground pad)	9
Controlled (ocean)	5
Failure (drone ship)	5
Failure	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

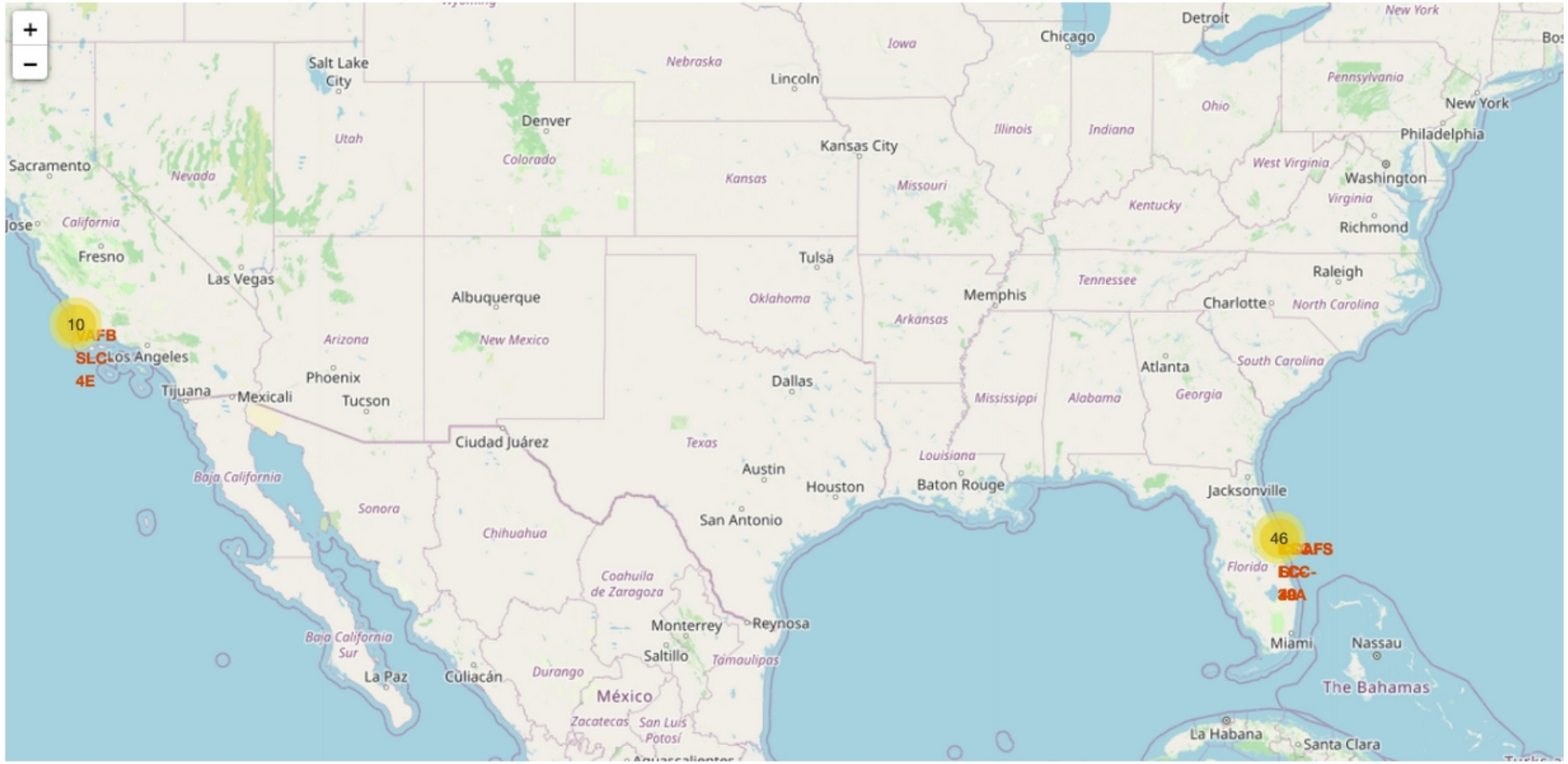
Section 4

Launch Sites Proximities Analysis

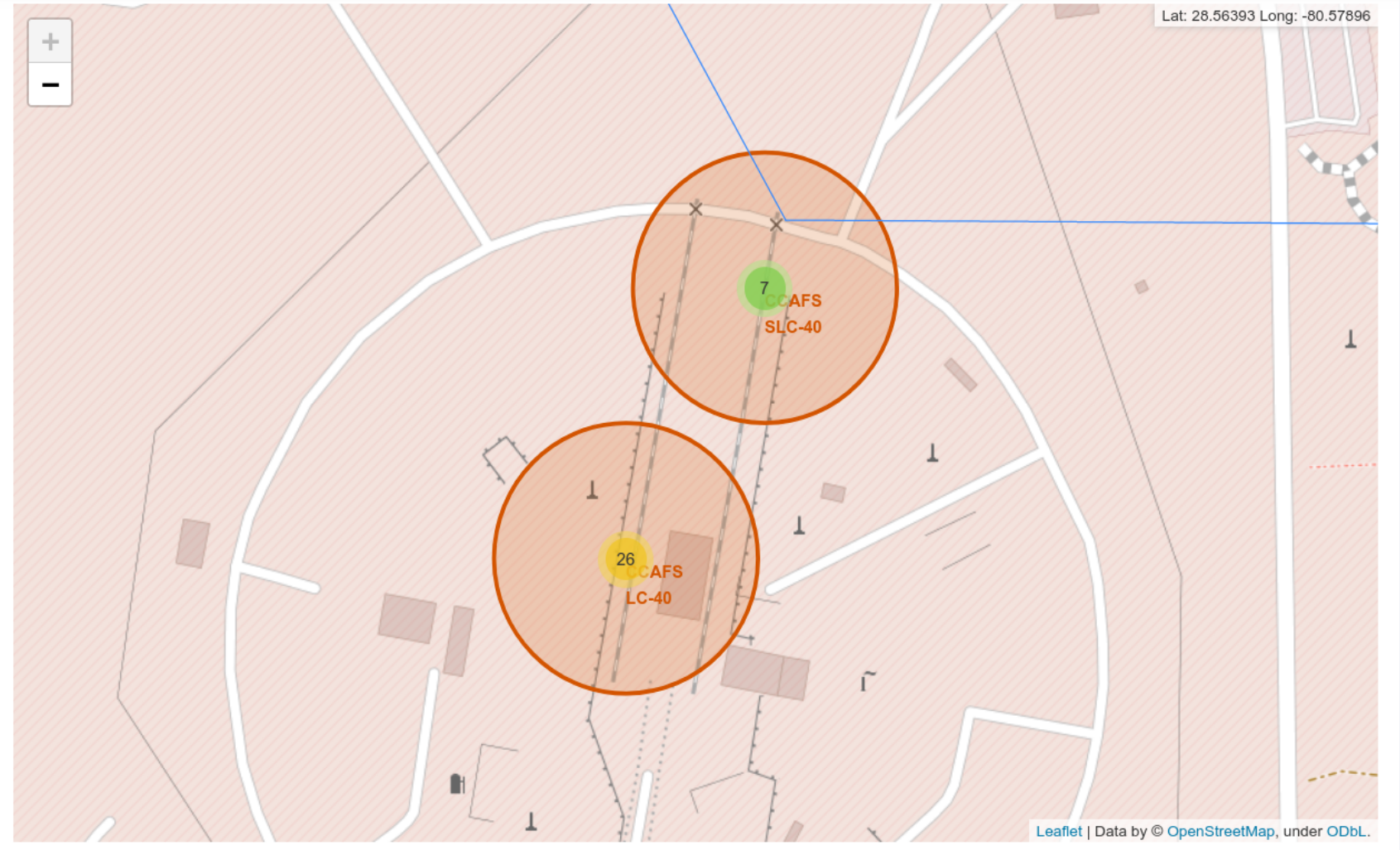
All launch sites

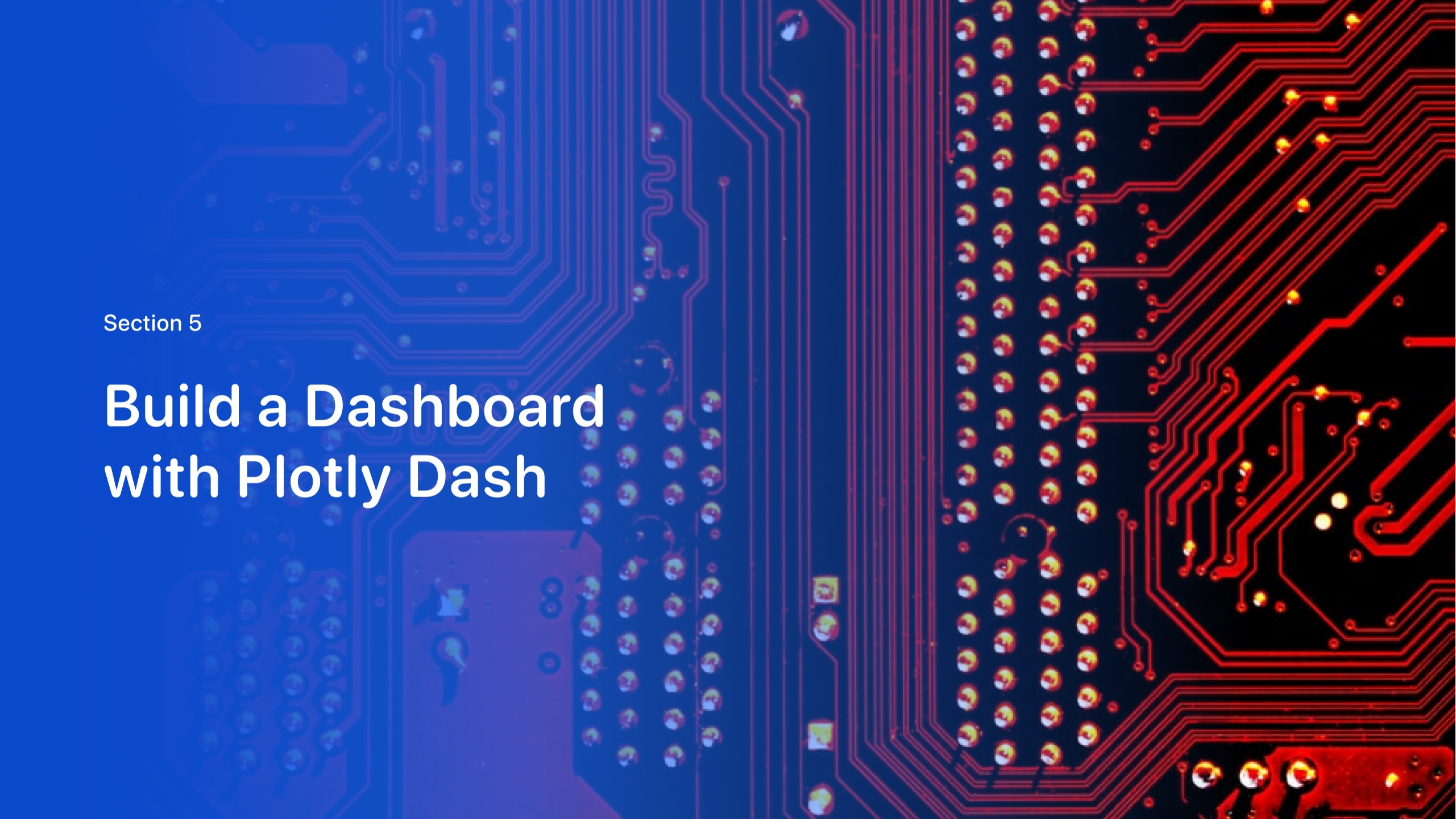


The success/failed launches for each site

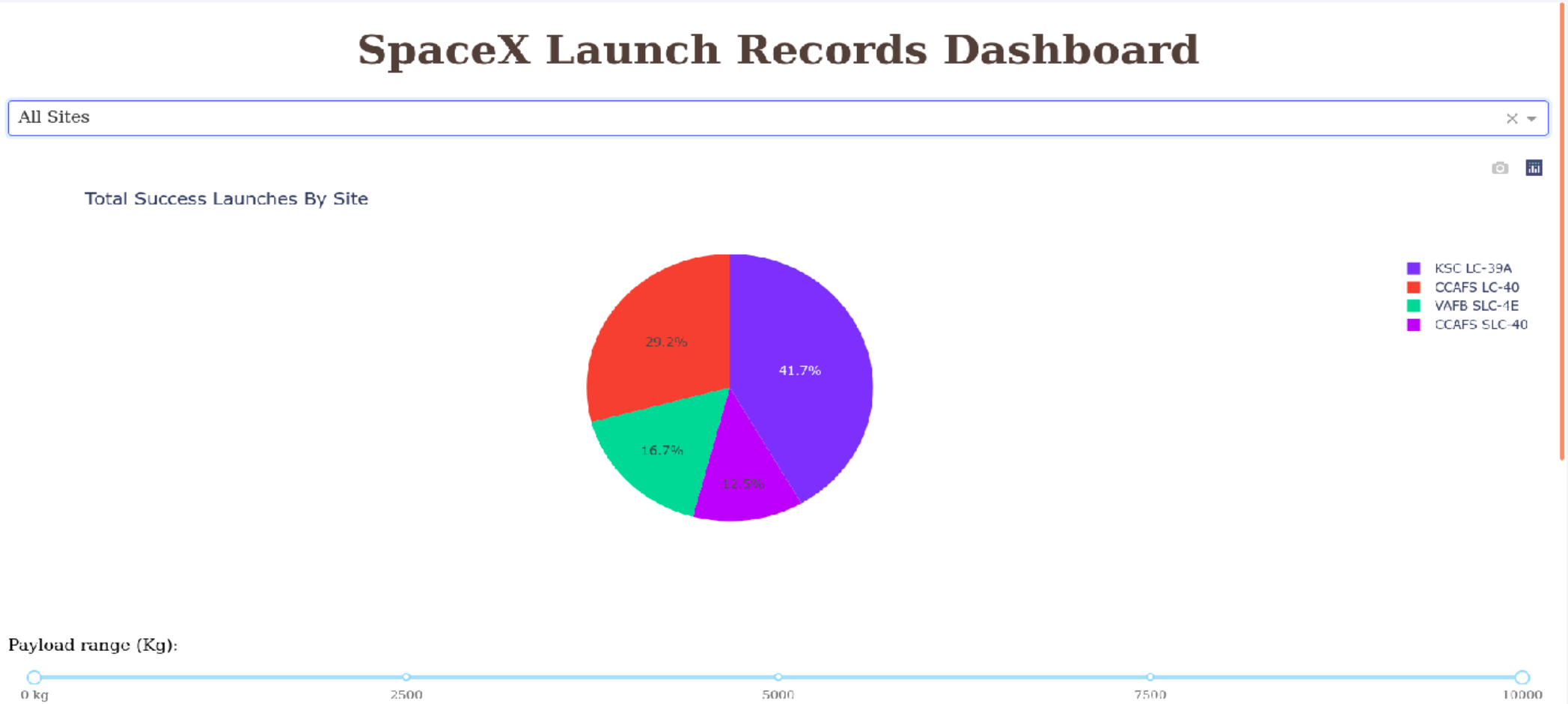


The distances between a launch site to its proximities

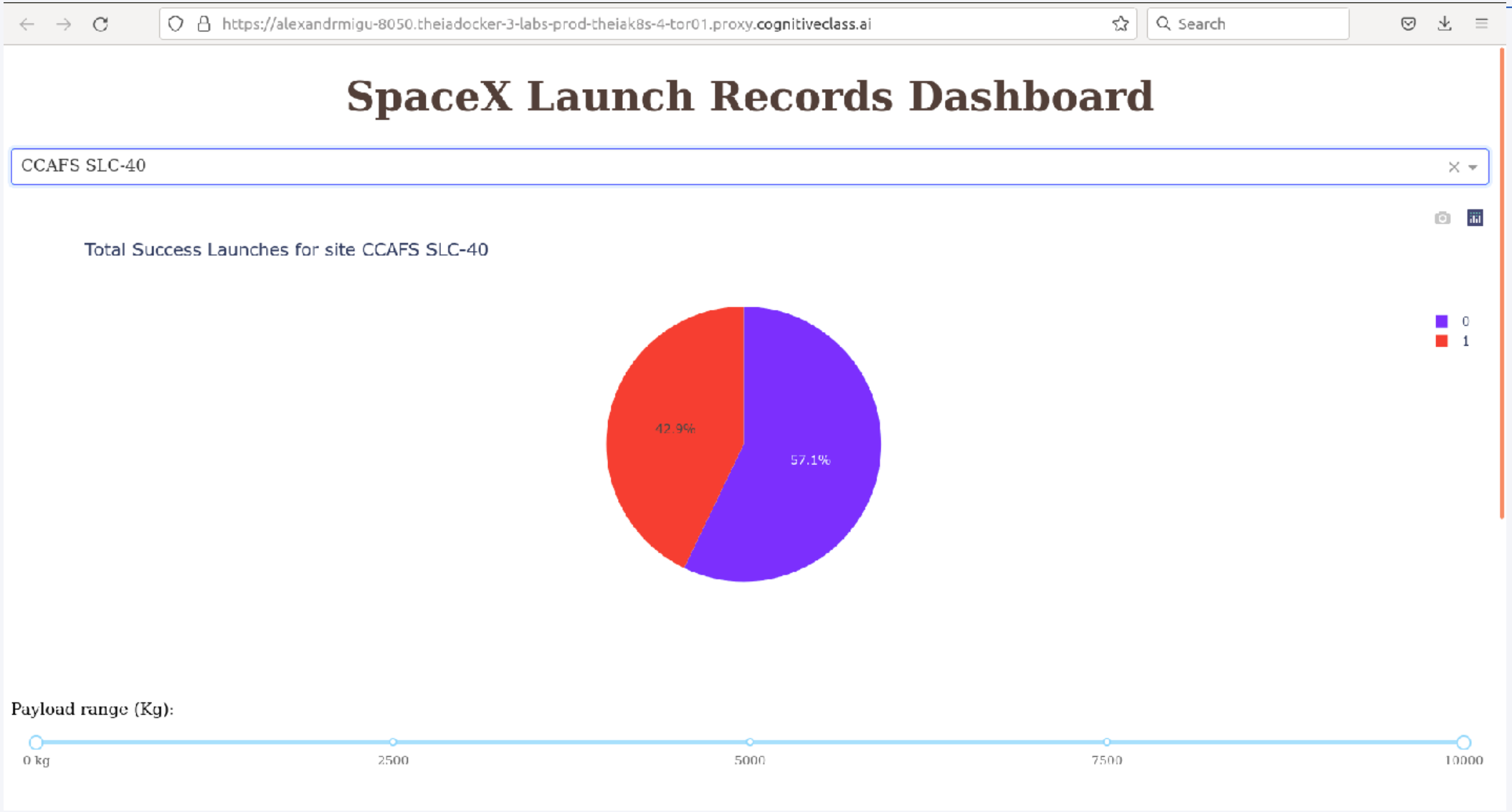



Section 5
Build a Dashboard with Plotly Dash

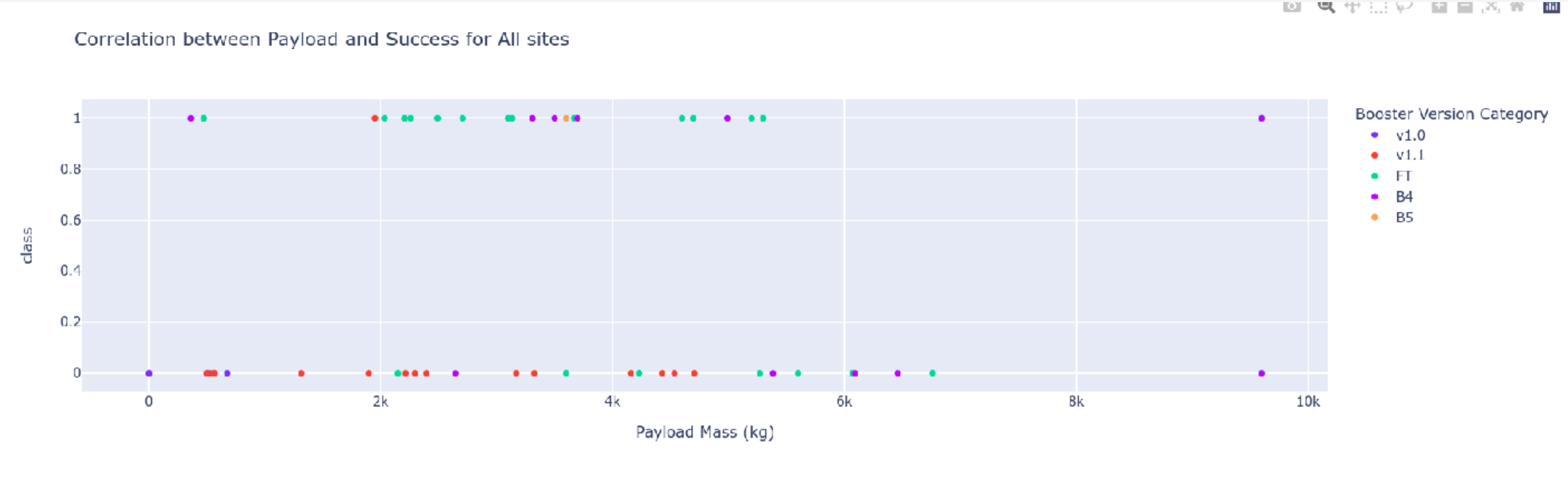
Total Success Launches By Site



Total Success Launches for the site with highest ratio



Correlation between Payload and Success for All Sites



Section 6

Predictive Analysis (Classification)

Classification Accuracy

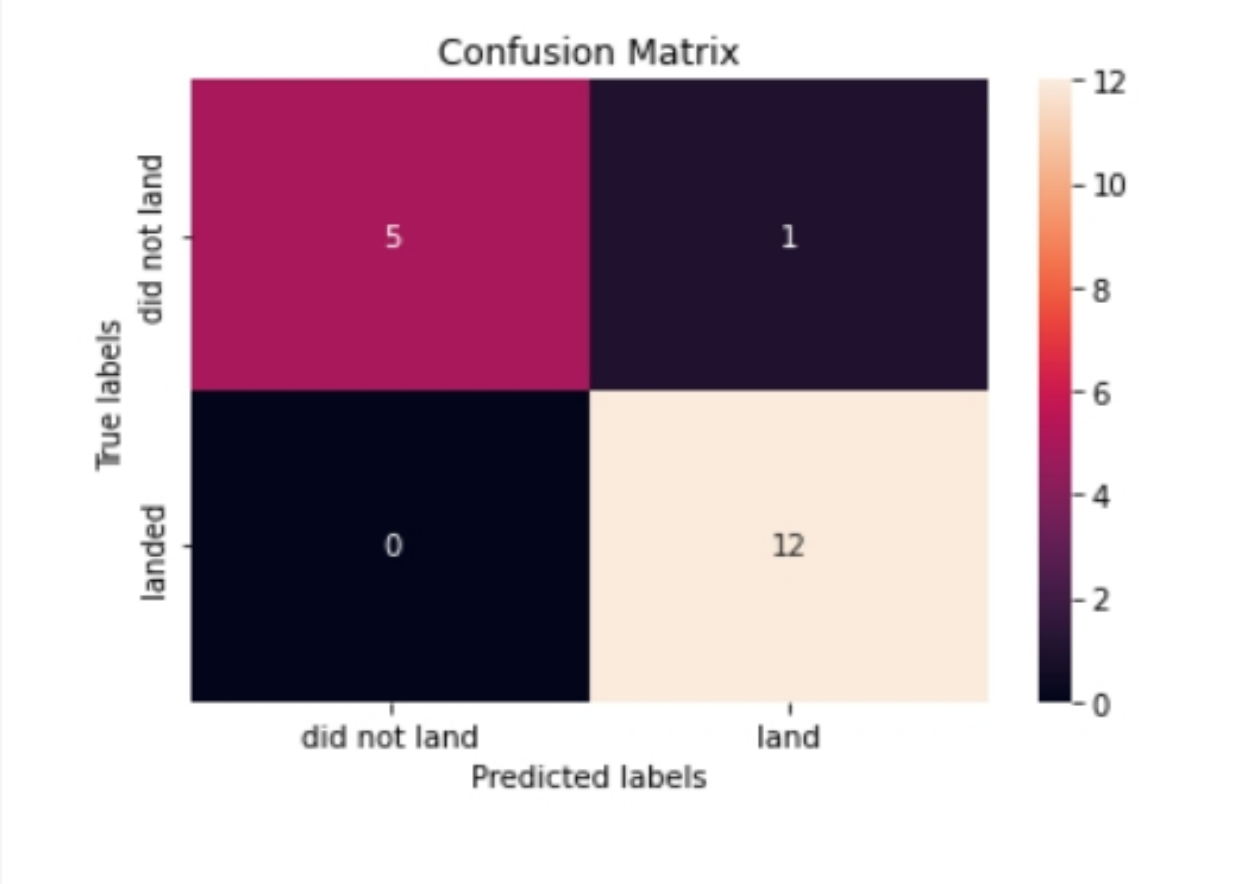
Find the method performs best:

In [31]:
```
print("Accuracy of logistic regression: ", logreg_cv.score(X_test, Y_test))
print("Accuracy of SVM: ", svm_cv.score(X_test, Y_test))
print("Accuracy of decision tree: ", tree_cv.score(X_test, Y_test))
print("Accuracy of KNN: ", knn_cv.score(X_test, Y_test))
```

```
Accuracy of logistic regression:  0.8333333333333334
Accuracy of SVM:  0.8333333333333334
Accuracy of decision tree:  0.9444444444444444
Accuracy of KNN:  0.8333333333333334
```

Decision tree has the highest accuracy = 0.9444. Other methods have accuracy = 0.8333. Therefore, decision tree performs best.

Confusion Matrix



Conclusions

Point 1

Point 2

Point 3

Point 4

...

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!