IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jagannath GM
09-12-2023

Outline

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

Executive Summary

- Summary of Methodologies
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis (EDA) with Data Visualization
  - Exploratory Data Analysis (EDA) with SQL
  - Building an Interactive Map with Folium
  - Building a Dashboard with Plotly Dash
- Summary of all results
  - EDA Results
  - Interactive Visuals/Analytics
  - Predictive Analysis

Introduction

- Predicting if SpaceX Falcon 9 first stage launches will land successfully to determine the cost of a launch

- 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 SpaceX can reuse the first stage

- This information can be used if an alternate company wants to bid against SpaceX for a rocket launch

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using a RESTful API and web scraping
- Perform data wrangling
  - Converted the data collected into Pandas dataframes
- Perform Exploratory Data Analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Using Machine Learning, split data into training and test sets to find the best hyperparameters for models.

Data Collection

- SpaceX API
- Web Scraping

Data Collection – SpaceX API

- Requested to the SpaceX API
- Cleaned the requested data

- https://github.com/Gmjagannath172/IBM-DataScience-Capstone-Project/blob/main/1-jupyter-labs-spacex-data-collection-api.ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

Task 2: Filter the dataframe to only include Falcon 9 launches

Task 3: Dealing with Missing Values: Calculate the mean for the PayloadMass feature and replace NaN values

Data Collection - Scraping

- Web scraping Falcon 9 and Falcon Heavy Launches records from Wikipedia
- Web scraped Falcon 9 launch records with Python package for parsing HTML documents: BeautifulSoup
- https://github.com/Gmjagannath172/IBM-DataScience-Capstone-Project/blob/main/2-jupyter-labs-webscraping.ipynb

Task 1: Request the Falcon 9 Launch Wiki page from its URL using HTTP GET method and instantiate BeautifulSoup() object

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

Task 3: Create a dataframe by parsing the launch HTML tables

Data Wrangling

- Data Wrangling process involved determining what would be the labels for training supervised models
- Performed Exploratory Data Analysis (EDA) to find some patterns in the data
- https://github.com/Gmjagannath172/IBM-DataScience-Capstone-Project/blob/main/3-labs-jupyter-spacex-Data%20wrangling.ipynb

Task 1: Calculate the number of launches on each site

Task 2: Calculate the number and occurrence of each orbit

Task 3: Calculate the number and occurrence of mission outcome of the orbits

Task 4: Create a landing outcome label from Outcome column

EDA with Data Visualization

- Created scatter plots and bar charts by writing Python code to analyze data in a Pandas data frame
- Utilized data viz skills to visualize the data and extract meaningful patterns for feature engineering to guide the modeling process
- https://github.com/Gmjagannath172/IBM-DataScience-Capstone-Project/blob/main/4-jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Created and executed SQL queries to answer questions by selecting and sorting through data.
- Loaded dataset into the corresponding table in a Db2 database in Python.
- https://github.com/Gmjagannath172/IBM-DataScience-Capstone-Project/blob/main/5-jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Calculated distances on an interactive map by writing Python code using the Folium library
- Generated interactive maps, plot coordinates, marked clusters, and analyzed the launch site proximities
- https://github.com/Gmjagannath172/IBM-DataScience-Capstone-Project/blob/main/6-lab_jupyter_launch_site_location.ipynb

Task 1: Mark all launch sites on a map

Task 2: Mark the success/failed launches for each site on the map

Task 3: Calculate the distances between a launch site to its proximities

Build a Dashboard with Plotly Dash

- Calculated distances on an interactive map by writing Python code using the Folium library
- Generated interactive maps, plot coordinates, marked clusters, and analyzed the launch site proximities
- https://github.com/Gmjagannath172/IBM-DataScience-Capstone-Project/blob/main/7-dashboard.py

Task 1: Add a dropdown list to enable Launch Site selection

Task 2: Add a pie chart to show the total successful launches for all sites

Task 3: Add a slider to select payload range

Task 4: Add a scatter chart to show the correlation between payload and launch success

Task 5: Add a callback function for ‘site-dropdown’ as input, ‘success-pie-chart’ as output

Task 6: Add a callback function for ‘site-dropdown’ and ‘payload-slider’ as inputs, ‘success-payload-scatter-chart’ as output

Predictive Analysis (Classification)

- Used Machine Learning to determine if the first stage of Falcon 9 will land successfully
- Split the data into training and testing data
- Found best hyperparameters using GridSearchCV for classification models: Logistic Regression, SVM, Decision Trees, KNN
- Selected the method that performed best using testing data
- https://github.com/Gmjagannath172/IBM-DataScience-Capstone-Project/blob/main/8-SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

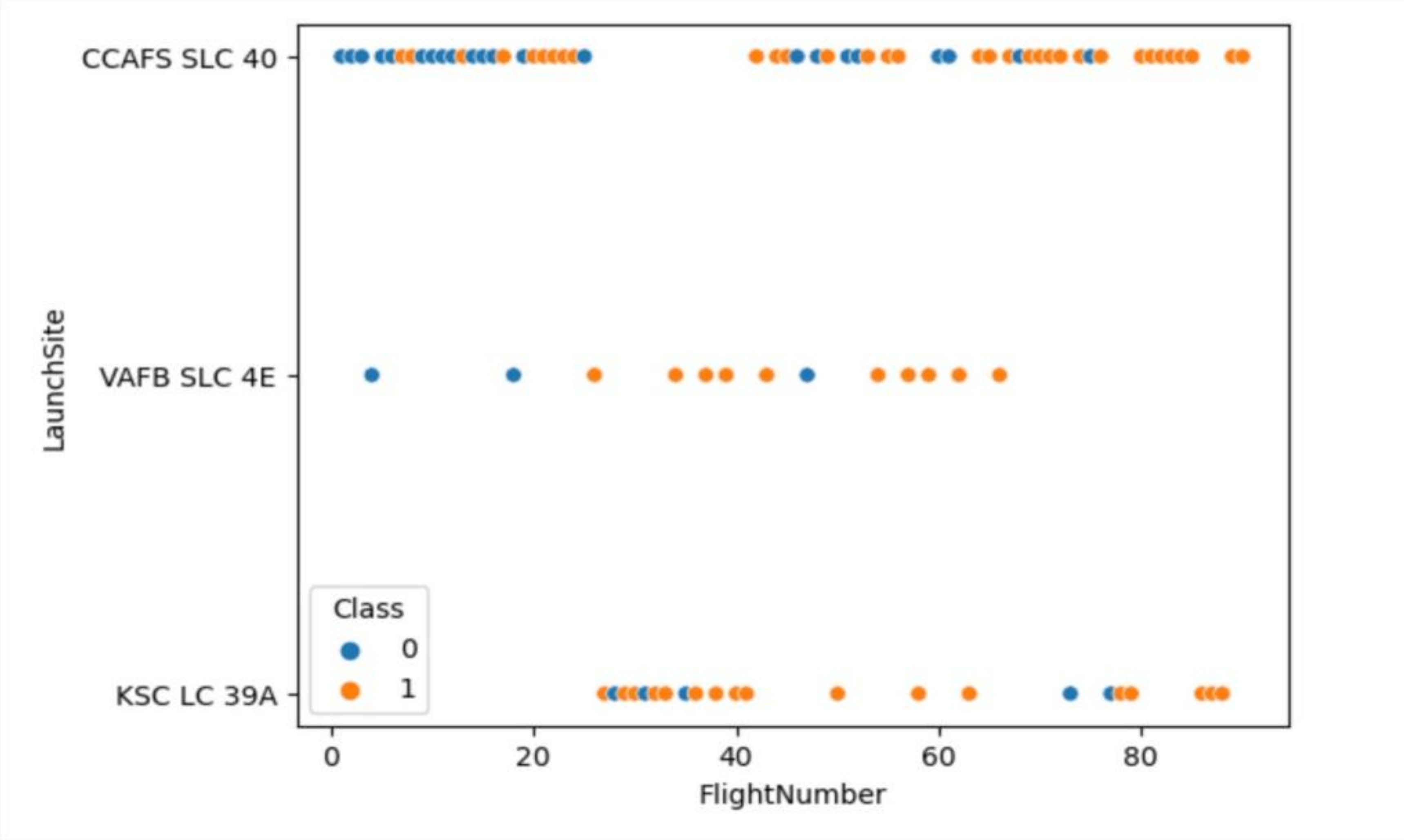
Results

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

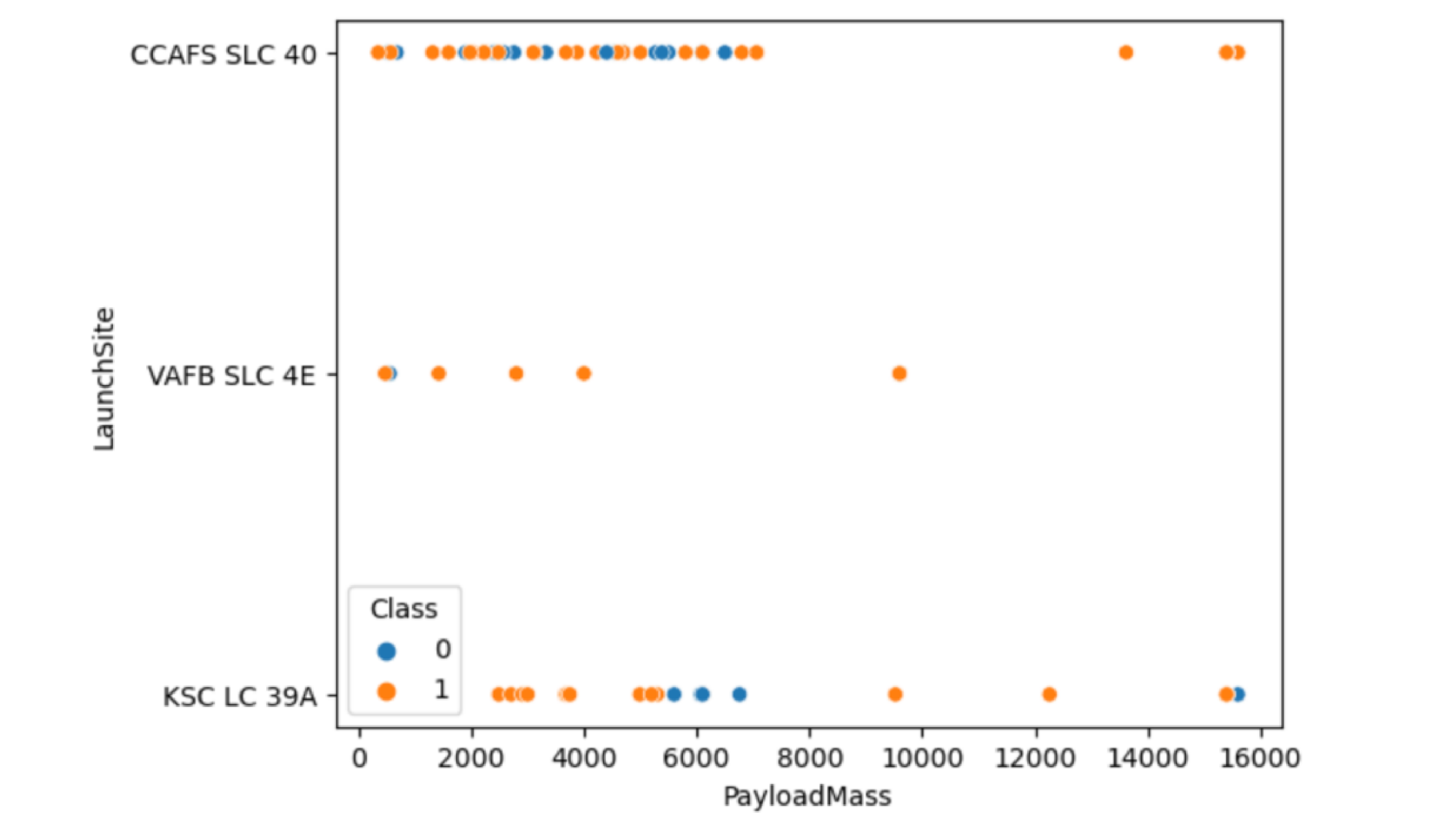
Section 2

Insights drawn from EDA

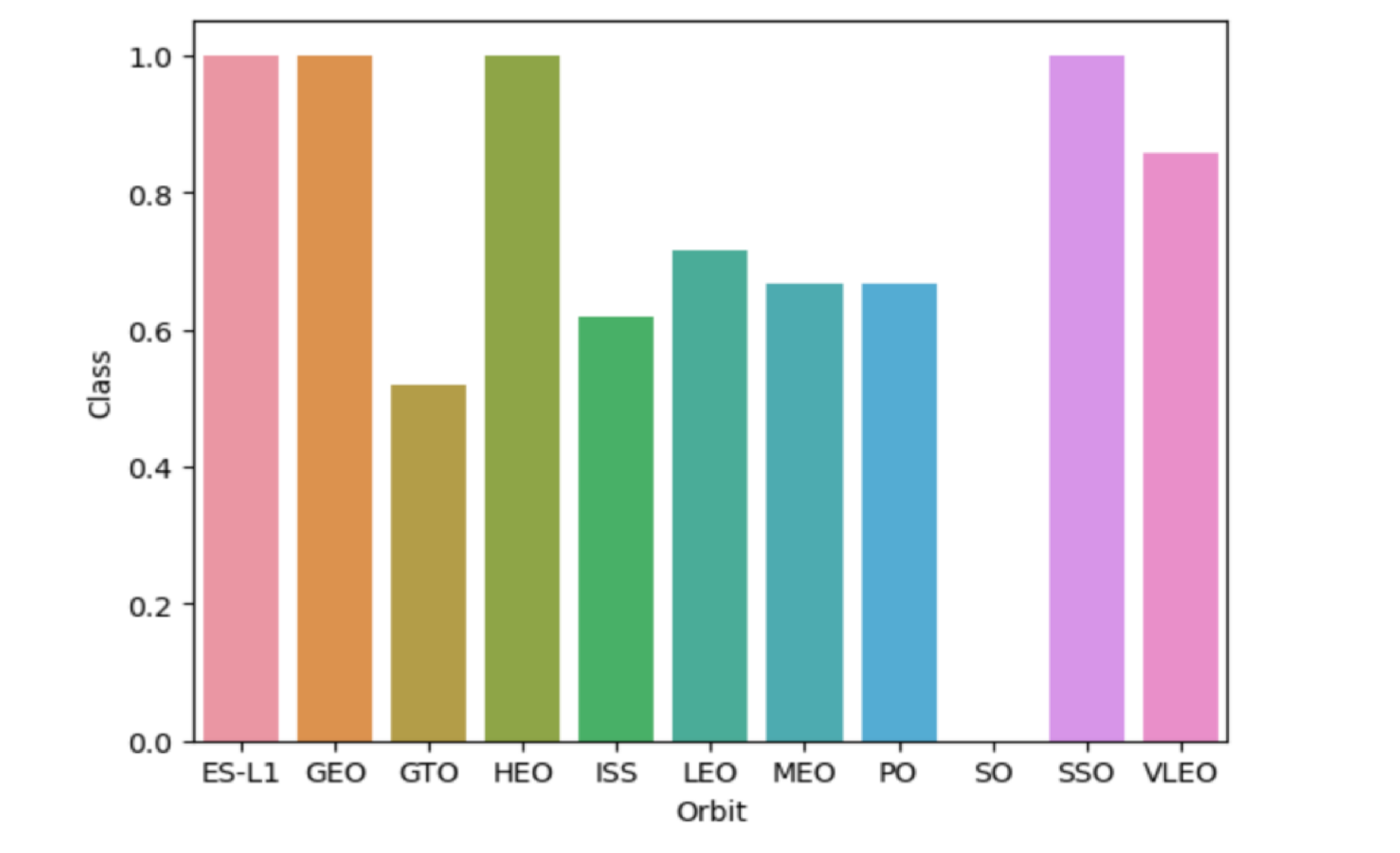
Flight Number vs. Launch Site



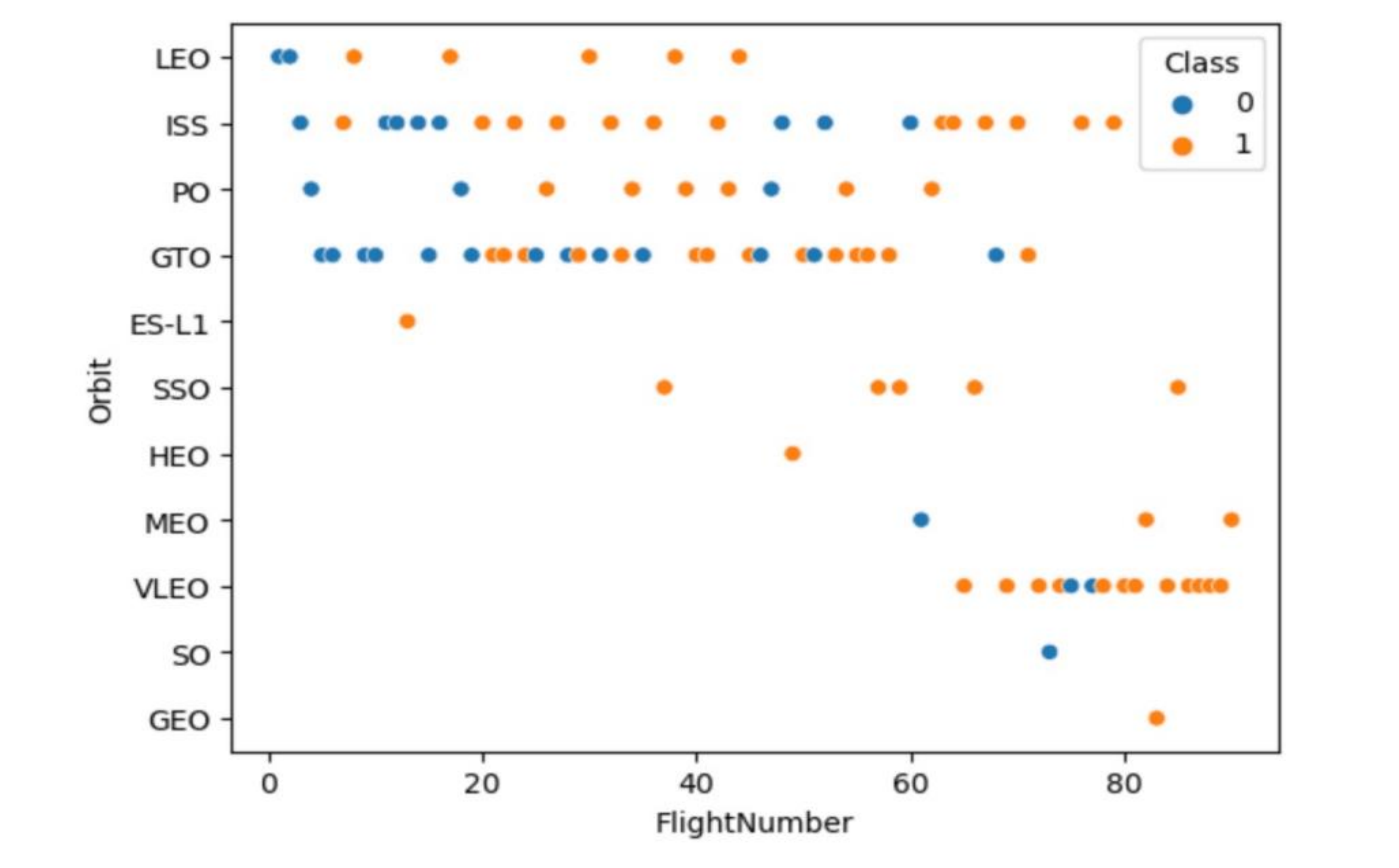
Payload vs. Launch Site



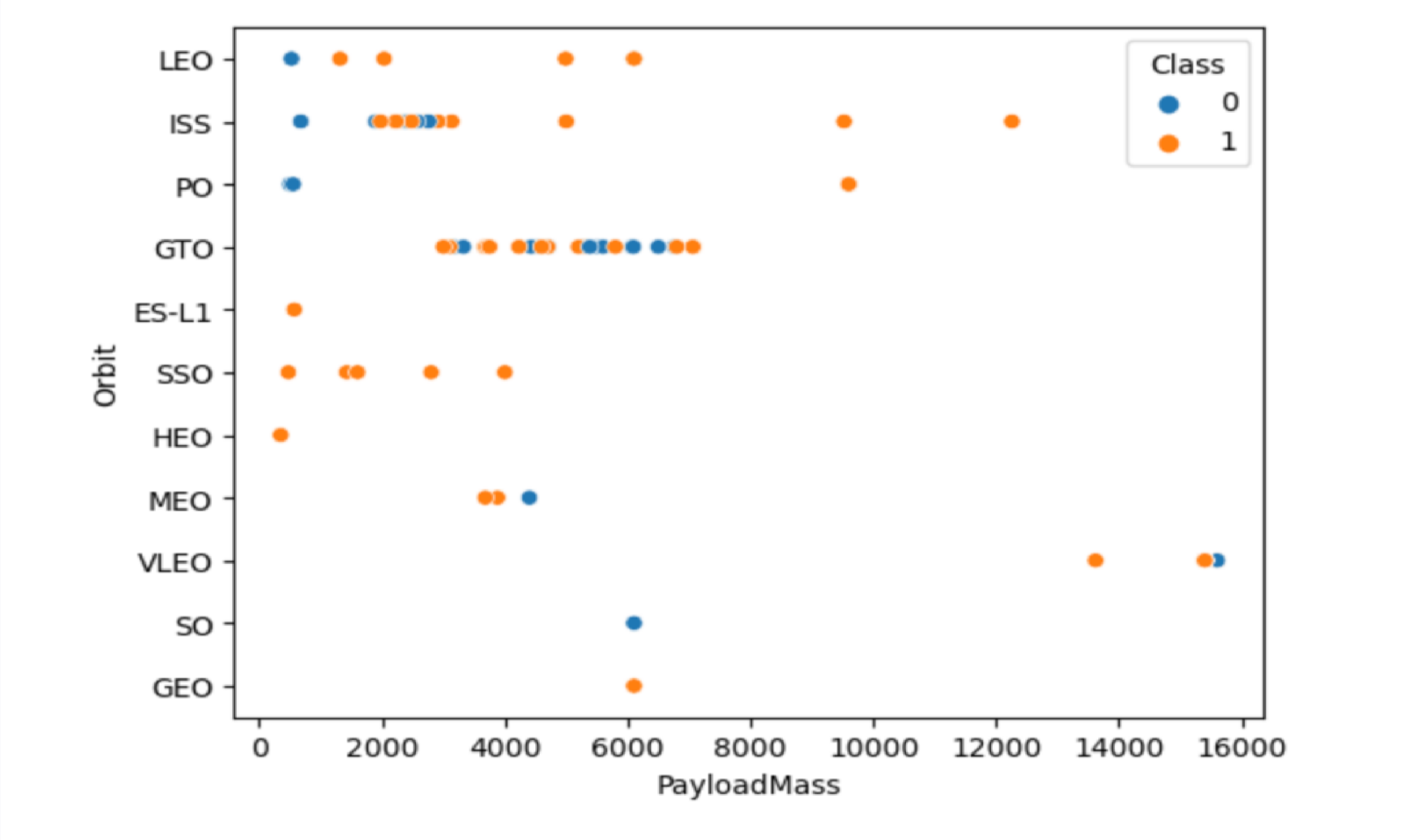
Success Rate vs. Orbit Type



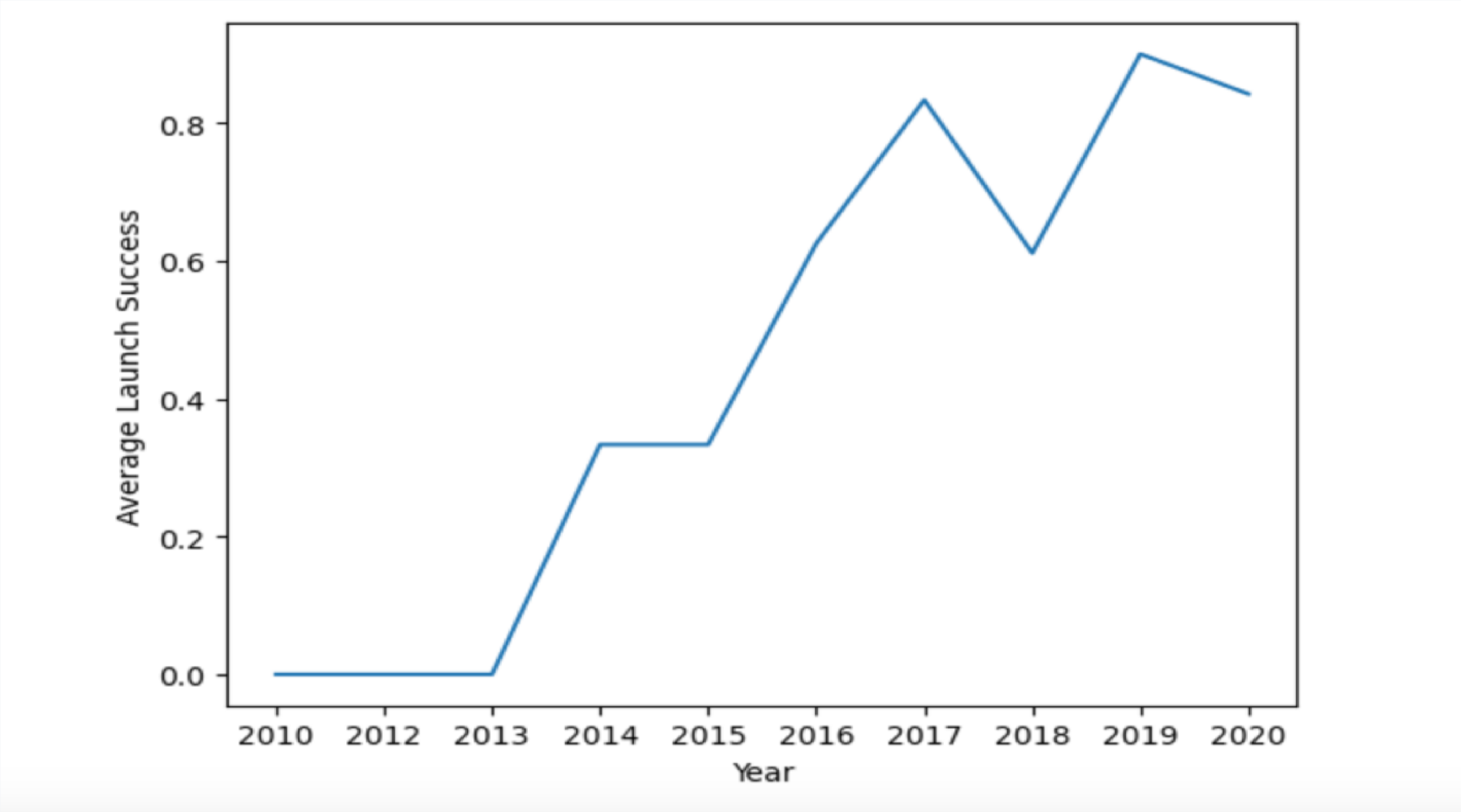
Flight Number vs. Orbit Type



Payload vs. Orbit Type



Launch Success Yearly Trend



All Launch Site Names

```
In [8]: %sql select distinct Launch_Site from SPACEXTABLE;

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

Out[8]:

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

Launch Site Names Begin with 'CCA'

```
In [9]: %sql select * from SPACEXTABLE where Launch_Site like 'CCA%' limit 5;

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

Out[9]:

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

Total Payload Mass

```
In [10]: %sql select distinct(Customer) from SPACEXTABLE where Customer like 'NASA (CRS)%';

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

Out[10]:

Customer
NASA (CRS)
NASA (CRS), Kacific 1

```
In [11]: %sql select sum(PAYLOAD_MASS__KG_) as Total_Payload_Mass_KG from SPACEXTABLE where Customer = 'NASA (CRS)';

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

Out[11]:

Total_Payload_Mass_KG
45596

Average Payload Mass by F9 v1.1

In [12]:
```
%sql select distinct(Booster_Version) from SPACEXTABLE  where Booster_Version like 'F9 v1.1%';
```

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

Out[12]:

Booster_Version
F9 v1.1 B1003
F9 v1.1
F9 v1.1 B1011
F9 v1.1 B1010
F9 v1.1 B1012
F9 v1.1 B1013
F9 v1.1 B1014
F9 v1.1 B1015
F9 v1.1 B1016
F9 v1.1 B1018
F9 v1.1 B1017

In [13]:
```
%sql select avg(PAYLOAD_MASS__KG_) as Avg_Payload_Mass_KG from SPACEXTABLE where Booster_Version = 'F9 v1.1';
```

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

Out[13]:

Avg_Payload_Mass_KG
2928.4

First Successful Ground Landing Date

```
In [14]: %sql select distinct(Landing_Outcome) from SPACEXTABLE;
 * sqlite:///my_data1.db
Done.
```

Out[14]:

Landing_Outcome
Failure (parachute)
No attempt
Uncontrolled (ocean)
Controlled (ocean)
Failure (drone ship)
Precluded (drone ship)
Success (ground pad)
Success (drone ship)
Success
Failure
No attempt

```
In [15]: %sql select min(Date) as first_successful_landing_date from SPACEXTABLE where Landing_Outcome = 'Success (ground pad
 * sqlite:///my_data1.db
Done.
```

Out[15]:

first_successful_landing_date
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

```
In [18]: %%sql select Payload from SPACEXTABLE where Landing_Outcome = 'Success (drone ship)' and
         PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000;

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

Out[18]:

Payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

```
In [19]: %sql select count(*) as Total_Number_Success from SPACEXTABLE where Mission_Outcome like 'Success%';

 * sqlite:///my_data1.db
Done.

Out[19]:
```

Total_Number_Success
100

```
In [20]: %sql select count(*) as Total_Number_Fail from SPACEXTABLE where Mission_Outcome not like 'Success%';

 * sqlite:///my_data1.db
Done.

Out[20]:
```

Total_Number_Fail
1

Boosters Carried Maximum Payload

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

In [31]:
```
%%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_)
                                                                          from SPACEXTABLE);
```

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

Out[31]:

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

In [33]:
```
%%sql select substr(Date, 6, 2) as Month, Landing_Outcome, Booster_Version, Launch_Site
from SPACEXTABLE where substr(Date, 0, 5) = '2015' and Landing_Outcome = 'Failure (drone ship)';
```

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

Out[33]:

Month	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

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

In [56]:
```
%%sql select Landing_Outcome, count(Landing_Outcome) as Count from SPACEXTABLE where
Date between '2010-06-04' and '2017-03-20' group by Landing_Outcome order by Count desc
```

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

Out[56]:

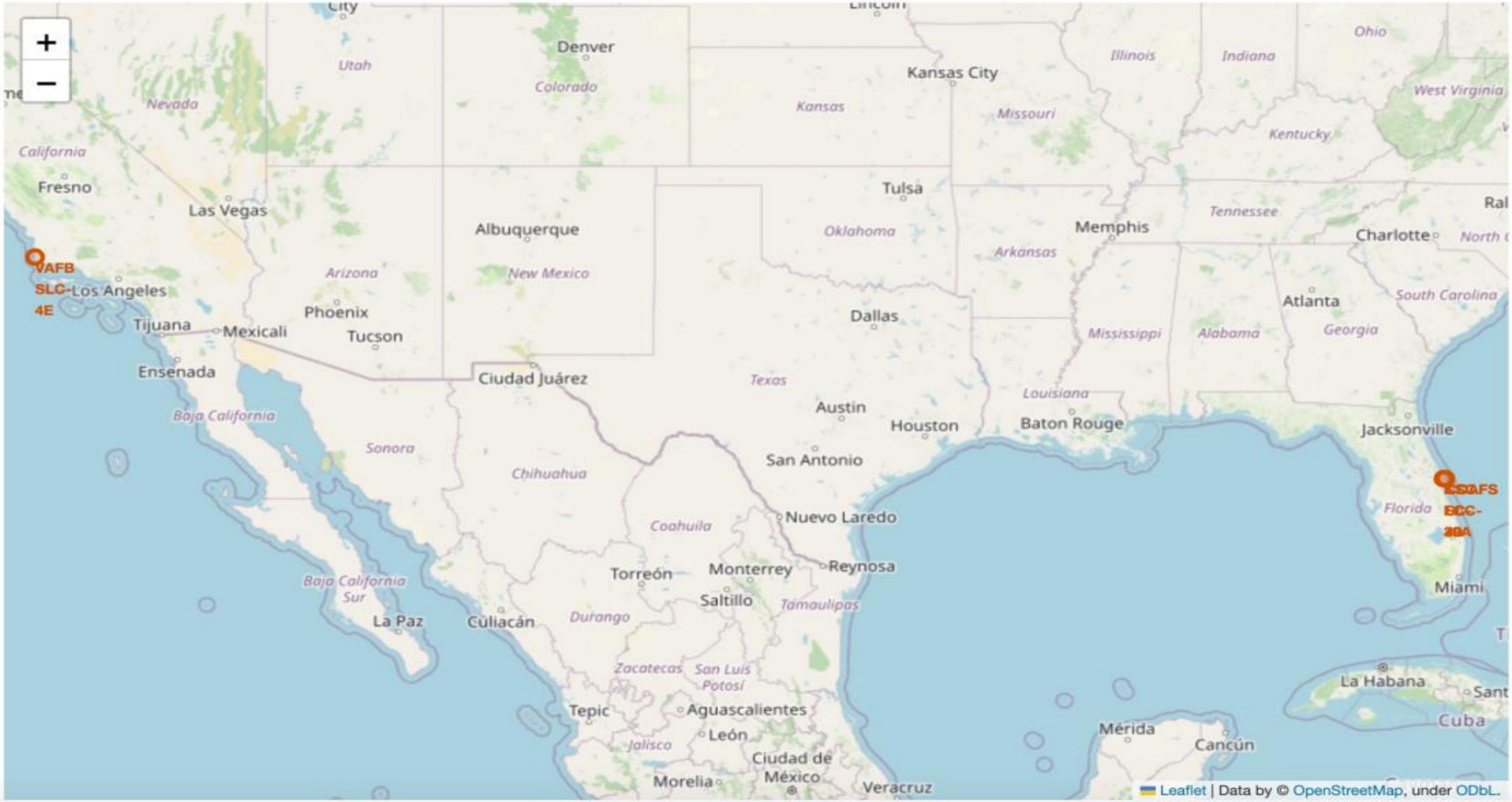
Landing_Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

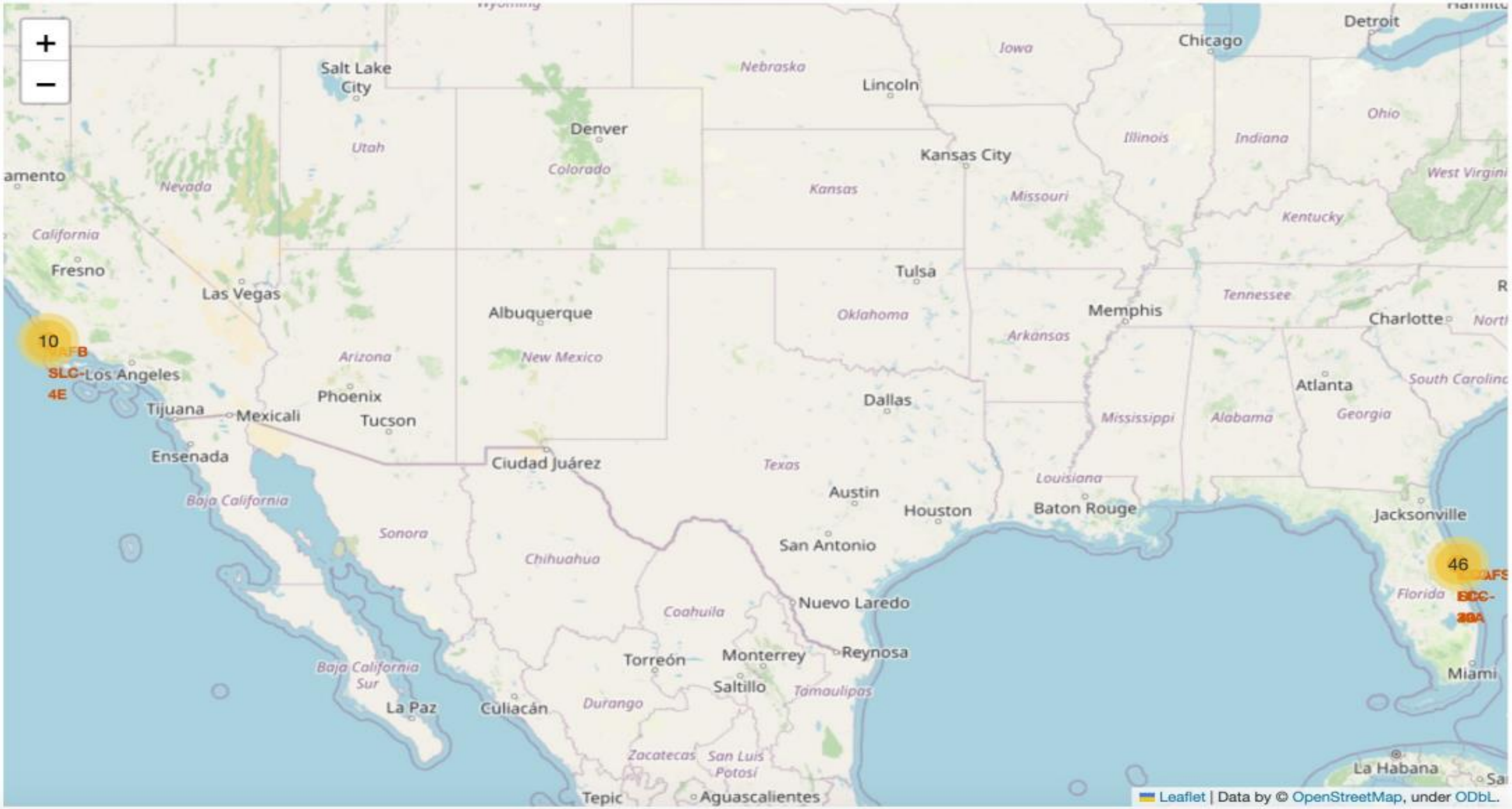
Folium Interactive Map with Launch Sites' locations

Out[48]:



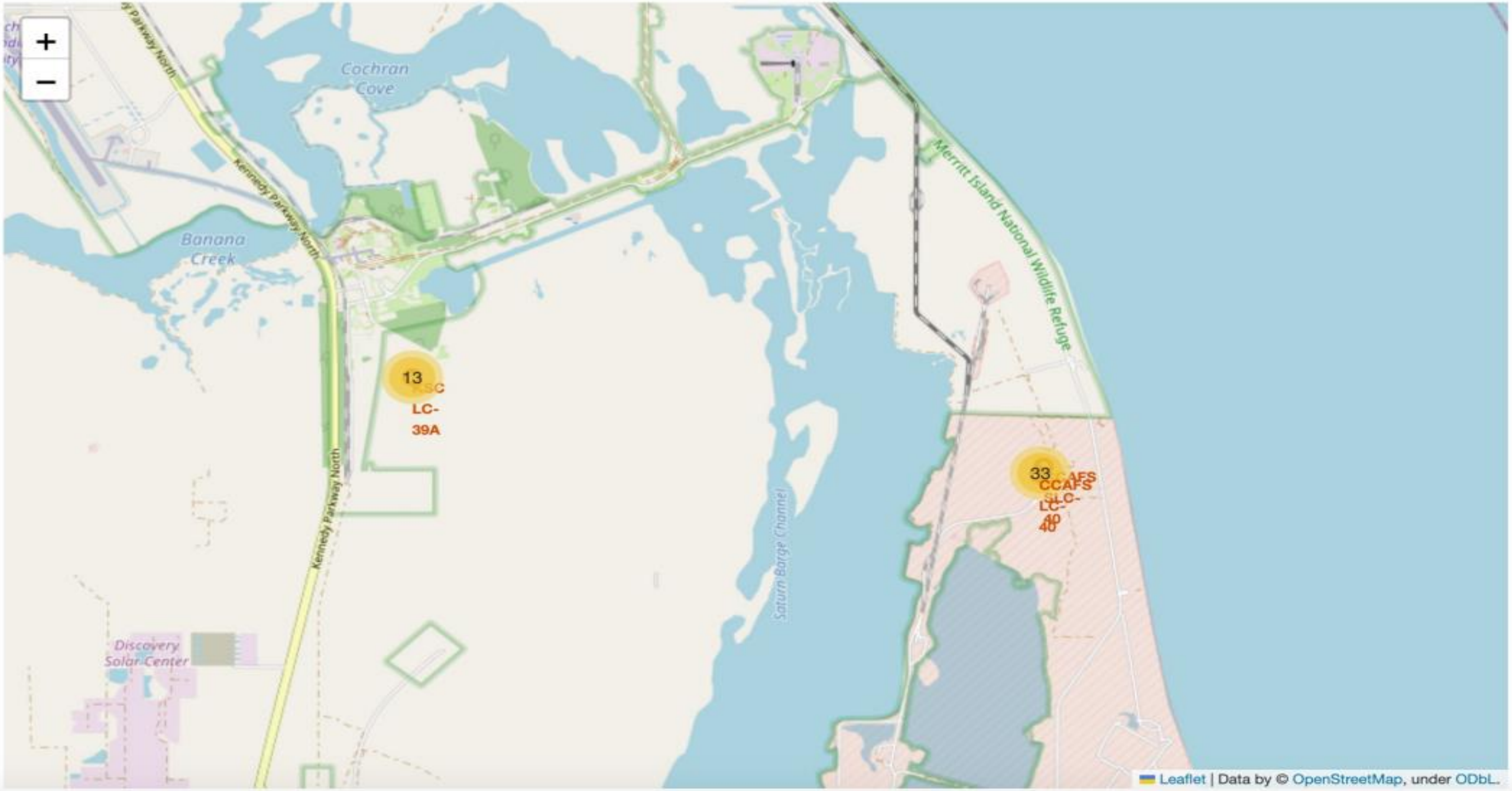
Folium: Color-labeled Launch Outcomes

Out[54]:



Folium: Color-labeled Launch Outcomes Zoomed In

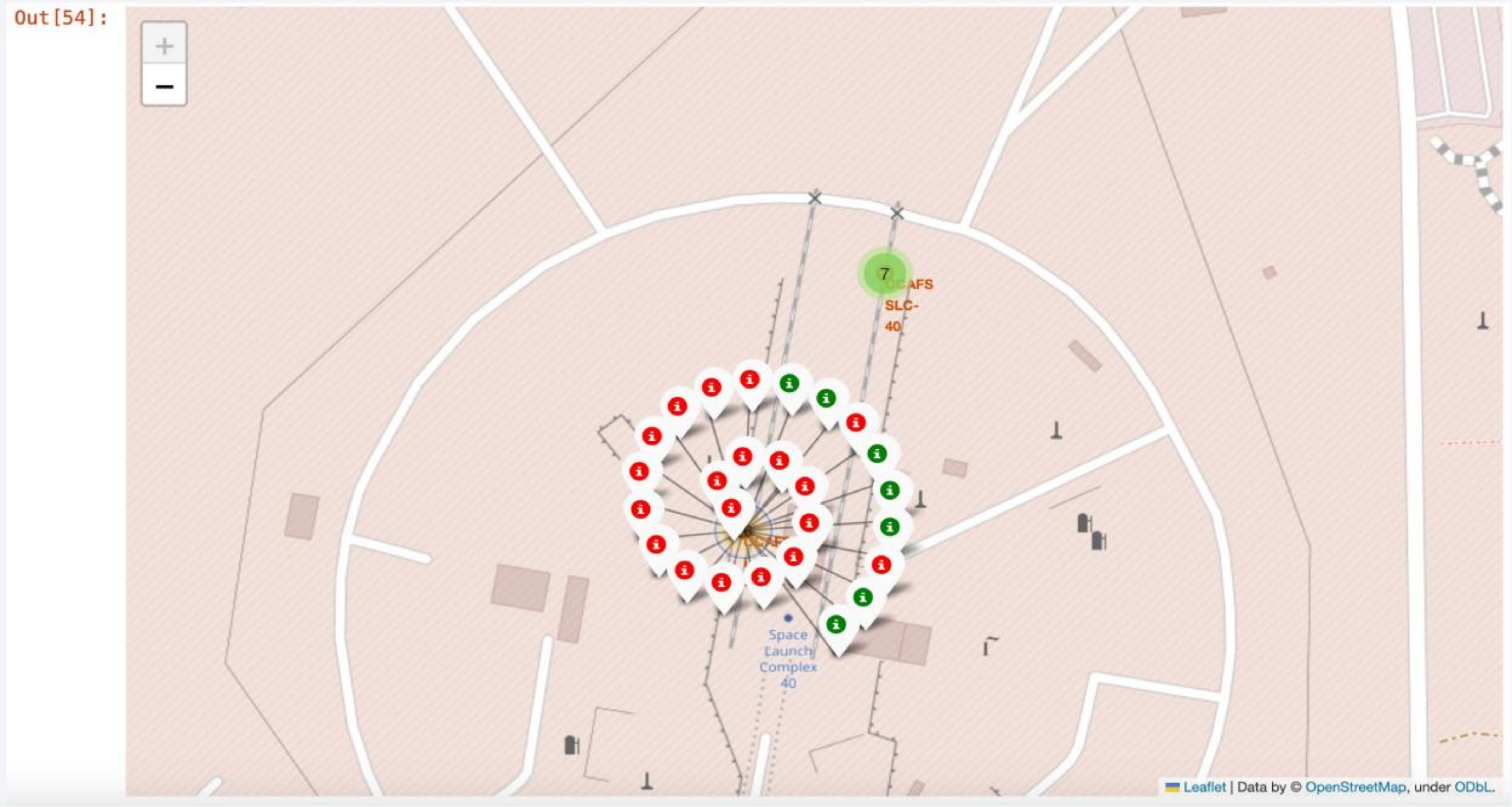
Out[54]:



Folium: Color-labeled Launch Outcomes Zoomed In

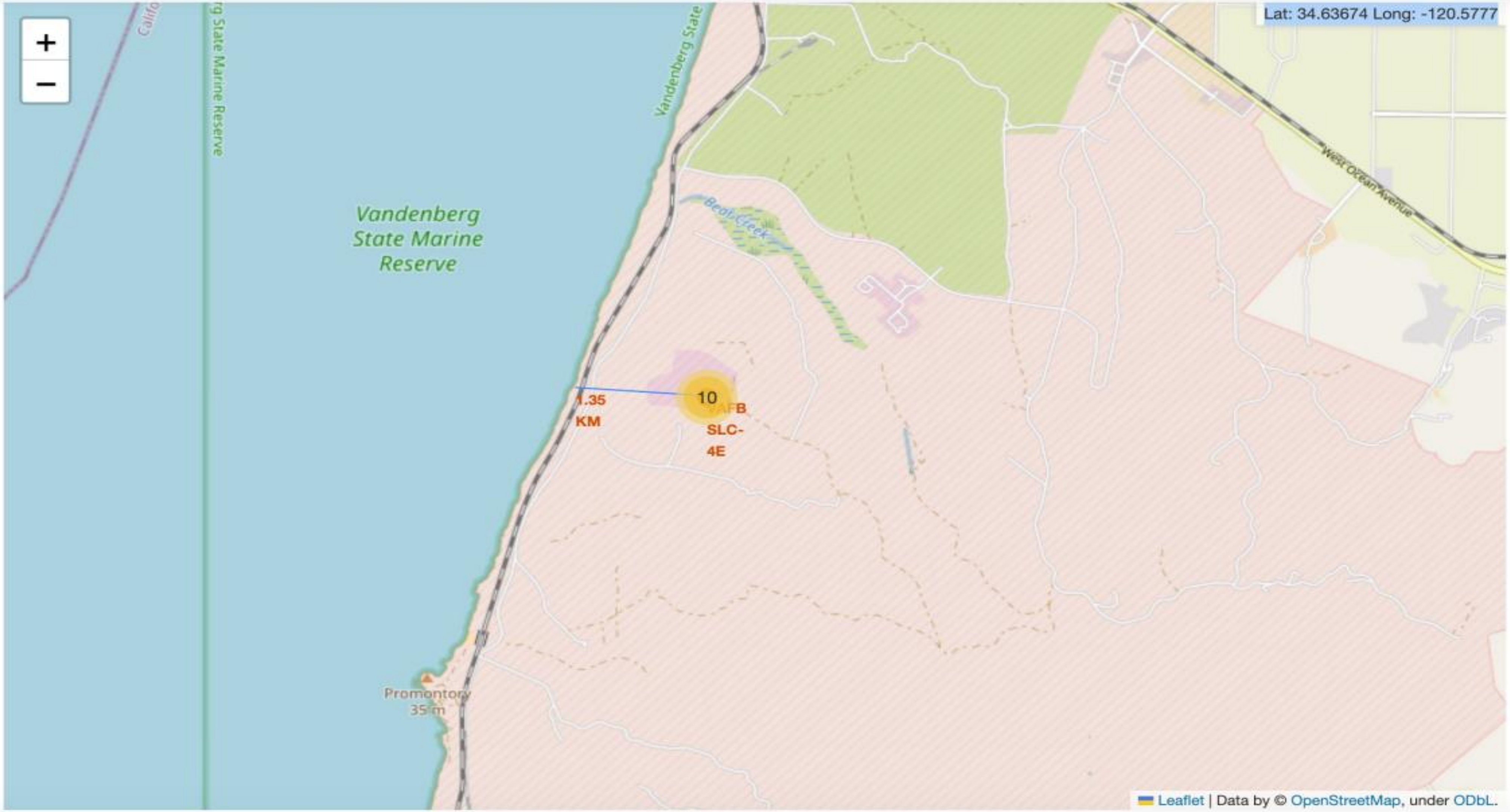
Out[54]:

Folium: Color-labeled Launch Outcomes Zoomed In



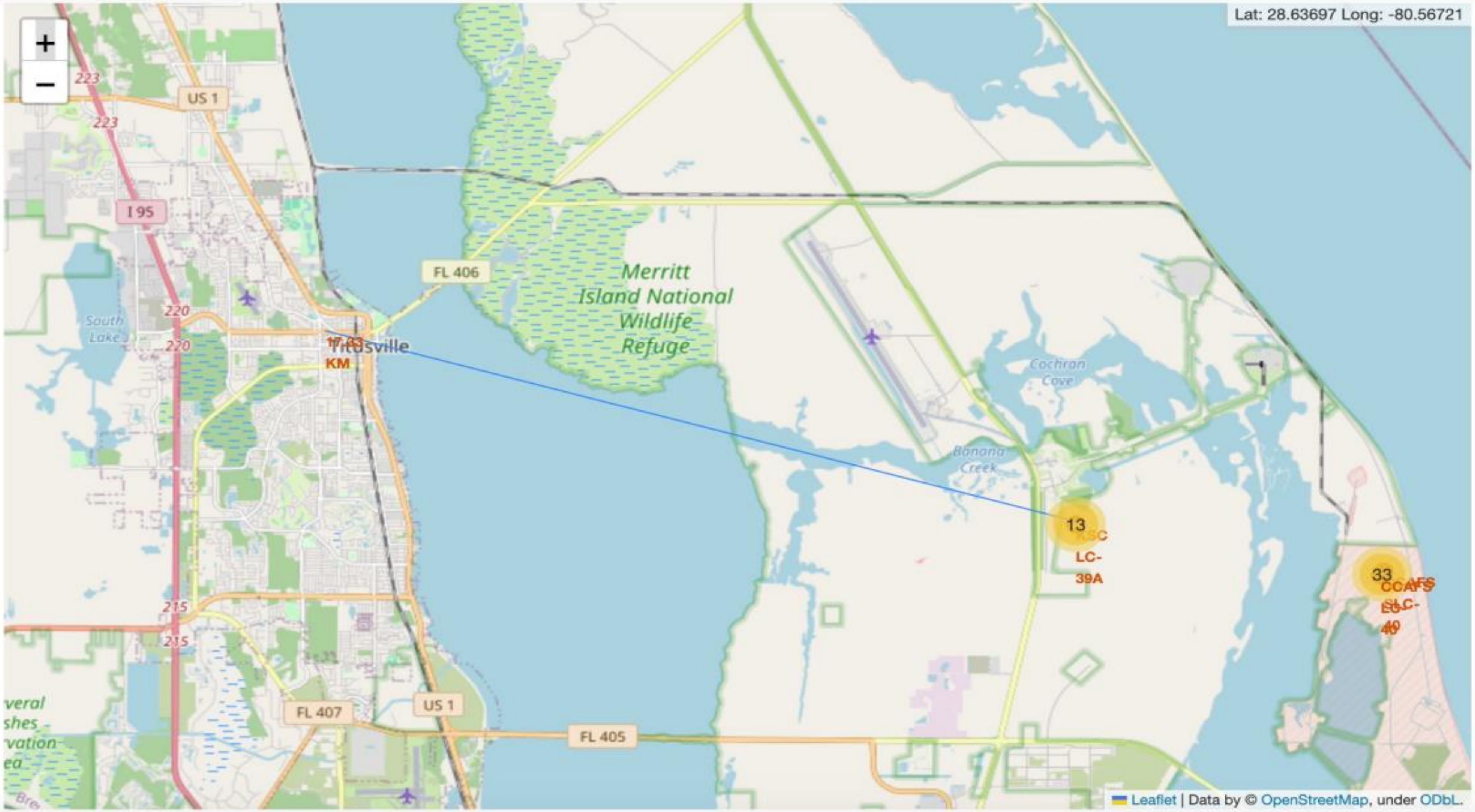
Folium: Selected Launch Cite Distance to its Proximities (Coastline)

Out[60]:



Folium: Selected Launch Cite Distance to its Proximities (City)

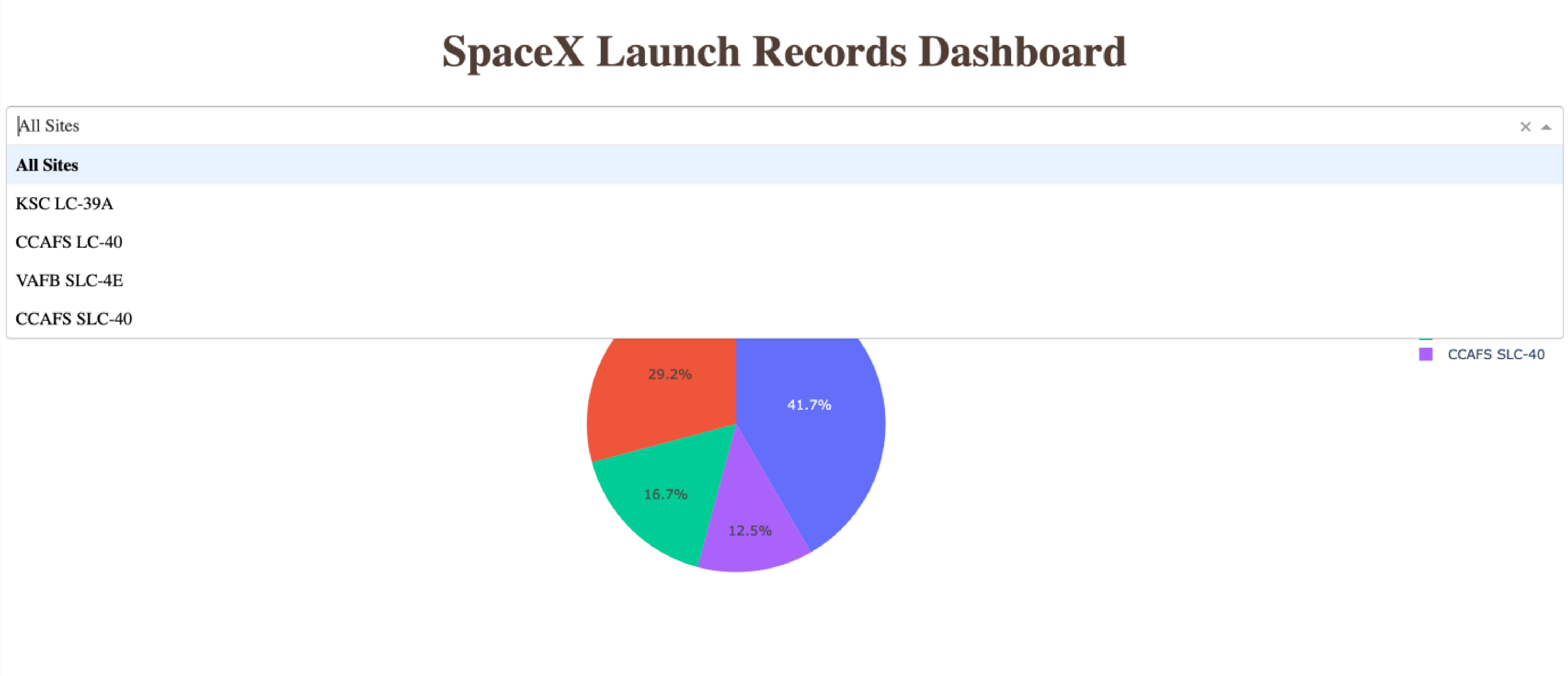
Out[63]:



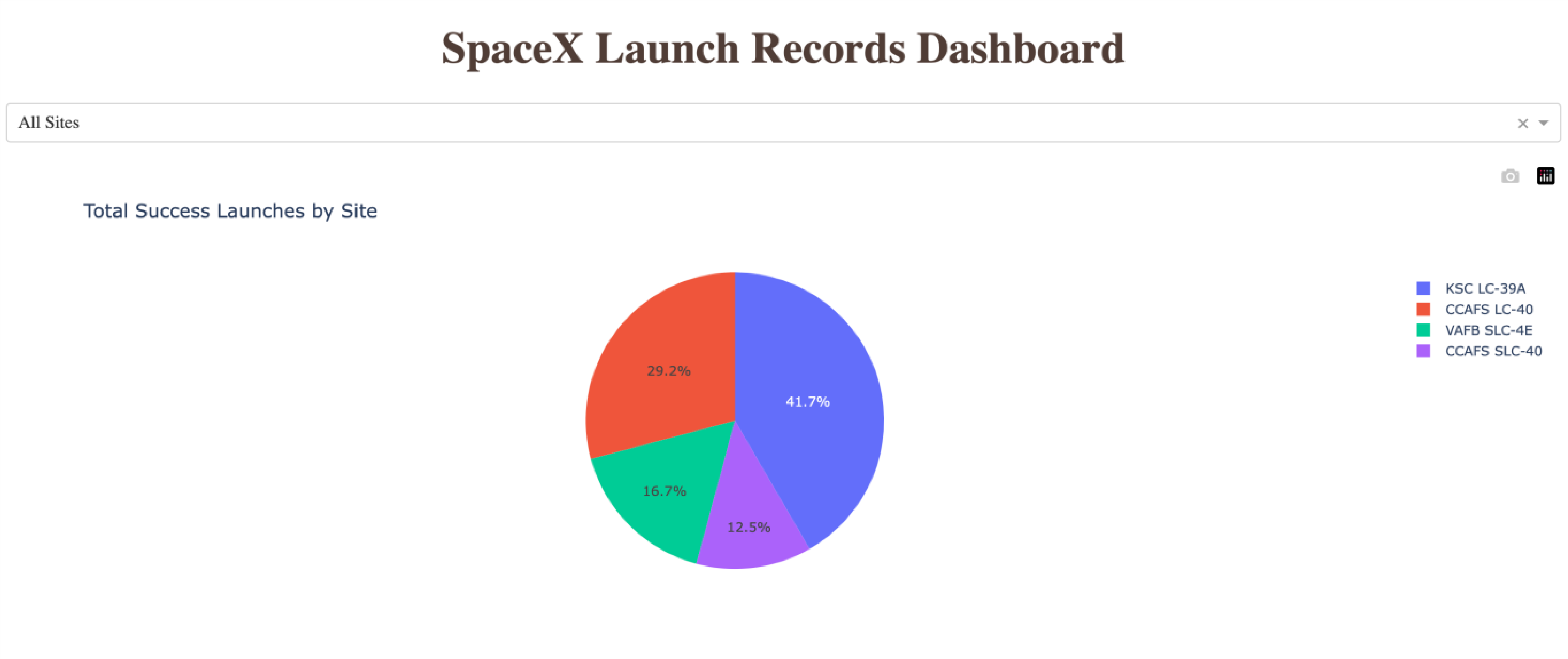
Section 4

Build a Dashboard with Plotly Dash

Plotly Dashboard: Launch Success Count in Pie Chart



Plotly Dashboard: Launch Success Count in Pie Chart



Plotly Dashboard: Payload vs. Launch Success Scatter Plot with Payload Range

Section 5

Predictive Analysis (Classification)

Classification Accuracy

Find the method performs best:

In [256]:
```
for n in logreg_cv, svm_cv, tree_cv, knn_cv:
    print('Training data accuracy score: ', n.best_score_)
    print('Test data accuracy score: ', n.score(X_test, Y_test), '\n')
```

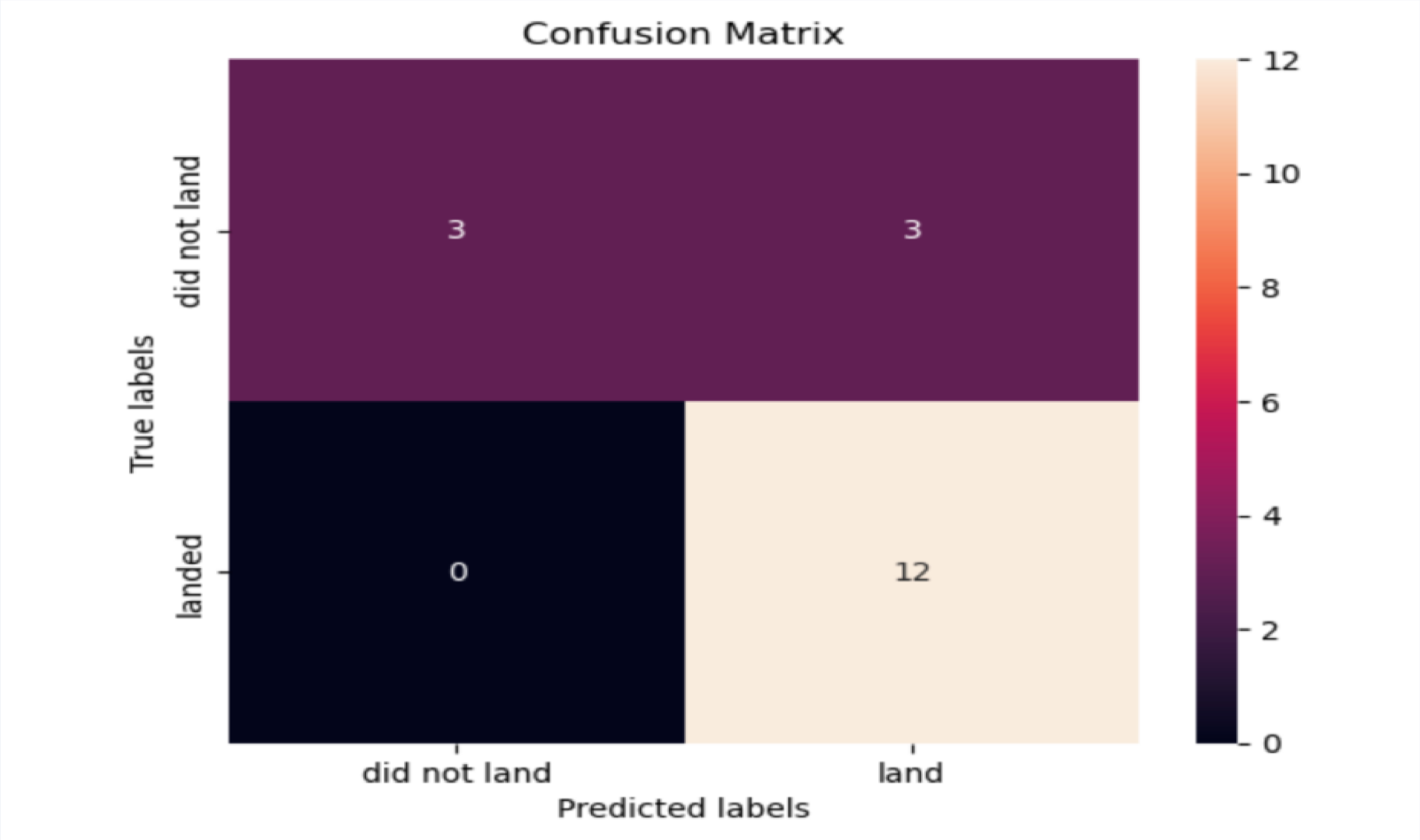
```
Training data accuracy score:  0.8464285714285713
Test data accuracy score:  0.8333333333333334

Training data accuracy score:  0.8482142857142856
Test data accuracy score:  0.8333333333333334

Training data accuracy score:  0.8857142857142858
Test data accuracy score:  0.8333333333333334

Training data accuracy score:  0.8482142857142858
Test data accuracy score:  0.8333333333333334
```

Confusion Matrix



Conclusions

- Launches with lower payload mass (kg) performed best
- The KSC LC 39A launch site has the most successful launches
- Orbits GEO, HEO, SSO, and ES-L1 has the highest success rates out of all orbit types
- All classification models used: Logistic Regression, SVM, Decision Tree, & KNN performed identically on test data (83.33% accuracy achieved)
- SpaceX launches' success rates are proportionally related to time; over time, the overall success rate has increased steadily

Appendix

- Confusion Matrix function :

```
In [216]: def plot_confusion_matrix(y,y_predict):
              "this function plots the confusion matrix"
              from sklearn.metrics import confusion_matrix

              cm = confusion_matrix(y, y_predict)
              ax= plt.subplot()
              sns.heatmap(cm, annot=True, ax = ax); #annot=True to annotate cells
              ax.set_xlabel('Predicted labels')
              ax.set_ylabel('True labels')
              ax.set_title('Confusion Matrix');
              ax.xaxis.set_ticklabels(['did not land', 'land']); ax.yaxis.set_ticklabels(['did not land', 'landed'])
              plt.show()
```

- Predictive Analysis (Classification) Python Notebook :

https://github.com/Gmjagannath172/IBM-DataScience-Capstone-Project/blob/main/8-SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Thank you!