IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Ethan Flint
13 March 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
  - EDA with SQL
  - Interactive map with Folium
  - Dashboard with Plotly Dash
  - Predictive analysis (classification modeling)
- Summary of all results
  - Optimal model was selected
  - Insights into different variables influence on a successful landing

Introduction

- SpaceX advertises its rocket launches for a cost that is significantly lower that other providers. The reason for this cost difference is the ability for SpaceX to reuse its first stage in some cases.
- In order for an alternative company to bid against SpaceX, it would need to determine whether or not the first stage for SpaceX Falcon 9 will be able to land successfully.
- Our goal will be to evaluate the different variables that influence the likelihood of a successful landing so that the stakeholders can make accurate conclusions about the potential cost and if they should make a bid.

Section 1

Methodology

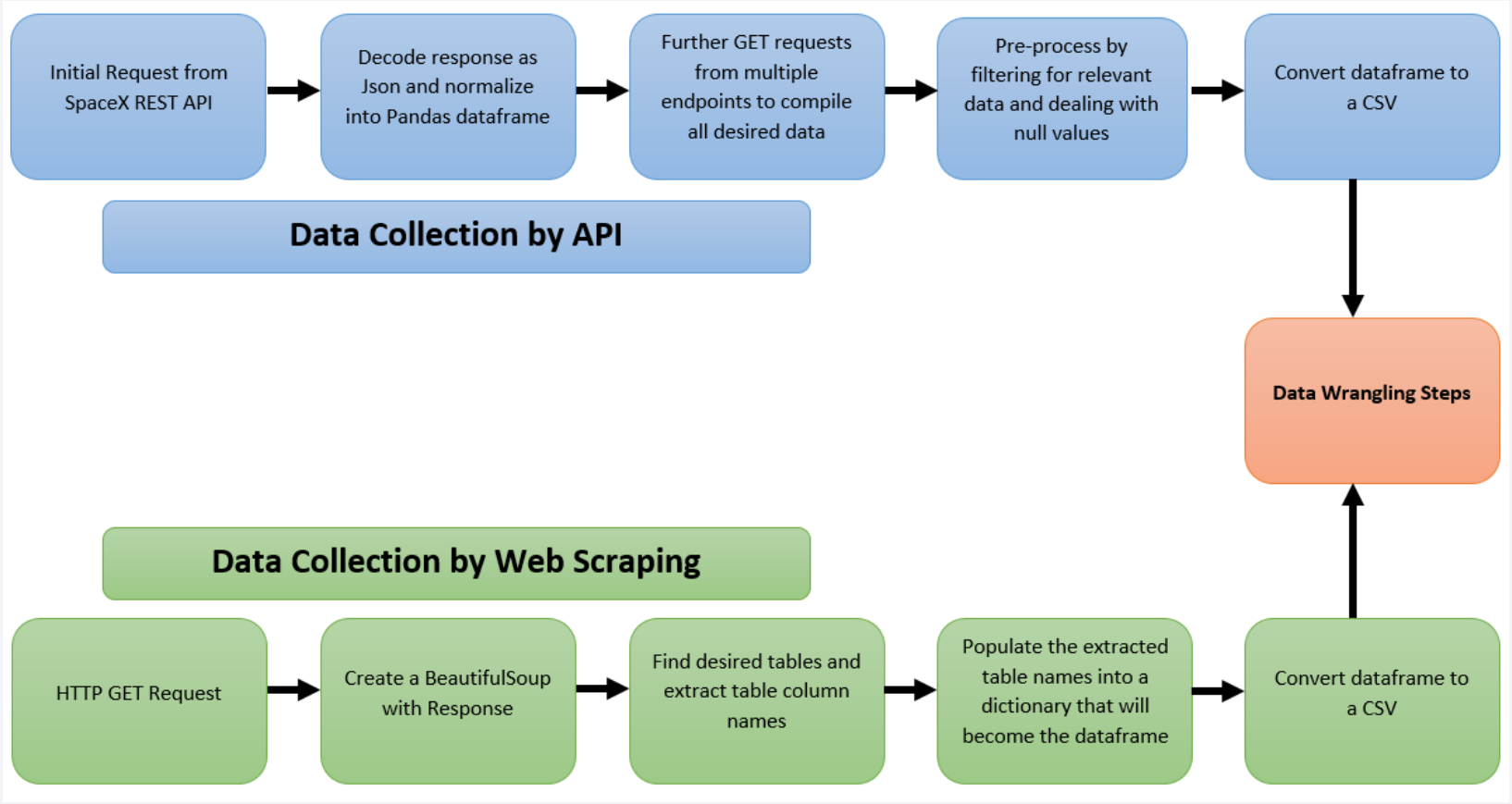
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web scraping HTML tables
- Perform data wrangling
  - Access multiple API endpoints, remove irrelevant launch data, replace null values, convert outcomes to a binary ‘0’ for failure and ‘1’ for success.
- Perform EDA using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression, Support Vector Machine, Decision Tree Classifier, and K-nearest Neighbors

Data Collection

- Data sets were collected through:
  - Multiple endpoints from the SpaceX REST API
  - Web scraping HTML tables from Wikipedia by using the Python BeautifulSoup package



Data Collection - SpaceX API

- Data collection through SpaceX REST API

- GitHub URL:

https://github.com/esflint/IBM-Data-Science-Professional-Certificate---Capstone/blob/master/SpaceX%20Data%20Collection%20API.ipynb

1 – Initiate a Request and confirm Response:

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
In [7]: response = requests.get(spacex_url)
```

→

```
In [10]: response.status_code
Out[10]: 200
```

2 – Decode response as Json and normalize into Pandas dataframe:

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

3 – Collect SpaceX launch data from multiple end points using GET requests on predefined functions and combine into new data frame:

```
In [16]: # Call getBoosterVersion
         getBoosterVersion(data)
In [18]: # Call getLaunchSite
         getLaunchSite(data)
In [19]: # Call getPayloadData
         getPayloadData(data)
In [20]: # Call getCoreData
         getCoreData(data)
```

→

```
In [21]: launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

→

```
In [22]: # Create a data from launch_dict
         data2 = pd.DataFrame(launch_dict)
```

4 – Pre-processing to filter for only relevant (Falcon 9) data and to replace null values:

```
In [27]: # Hint data['BoosterVersion']!='Falcon 1'
         data_falcon9 = data2[data2['BoosterVersion']!='Falcon 1']
```

→

```
In [33]: # Calculate the mean value of PayloadMass column
         payload_mean = data_falcon9['PayloadMass'].mean()
         # Replace the np.nan values with its mean value
         data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, payload_mean)
```

5 – Export final data frame as a CSV:

```
In [34]: data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

- Data collection through web scraping using BeautifulSoup

- GitHub URL:

https://github.com/esflint/IBM-Data-Science-Professional-Certificate---Capstone/blob/master/Data%20Collection%20with%20Web%20Scraping.ipynb

1 – Perform an HTTP GET request and create a BeautifulSoup with the response:

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        response = requests.get(static_url)
```

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(response.text, 'html')
```

2 – Find the tables on the wiki page and extract table column names:

```
In [9]: # Let's print the third table and check its content
        first_launch_table = html_tables[2]
        print(first_launch_table)
```

```
In [10]: column_names = []

         # Apply find_all() function with `th` element on
         # Iterate each th element and apply the provided
         # Append the Non-empty column name (`if name is
         for item in first_launch_table.find_all('th'):
             name = extract_column_from_header(item)
             if name is not None and len(name) > 0:
                 column_names.append(name)
```

```
In [11]: print(column_names)

['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

3 – Create and populate a dictionary using the extracted column names and convert into Pandas dataframe:

```
In [12]: launch_dict= dict.fromkeys(column_names)

         # Remove an irrelvant column
         del launch_dict['Date and time ( )']

         # Let's initial the launch_dict with each value to be an empty list
         launch_dict['Flight No.'] = []
         launch_dict['Launch site'] = []
         launch_dict['Payload'] = []
         launch_dict['Payload mass'] = []
         launch_dict['Orbit'] = []
         launch_dict['Customer'] = []
         launch_dict['Launch outcome'] = []
         # Added some new columns
         launch_dict['Version Booster']=[]
         launch_dict['Booster landing']=[]
         launch_dict['Date']=[]
         launch_dict['Time']=[]
```

Populate the launch_dict

```
In [15]: df=pd.DataFrame(launch_dict)
         df
```

Out[15]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.0B0003.1	Failure	4 June 2010	18:45
1	2	CCAFS	Dragon	0	LEO	NASA	Success	F9 v1.0B0004.1	Failure	8 December 2010	15:43

4 – Export final data frame as a CSV:

```
In [16]: df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- Previously wrangled data to exclude irrelevant data and deal with null values
- Converted the multiple landing outcomes to either ‘0’ for unsuccessful and ‘1’ for successful (as seen in the image to the right)
- GitHub URL: https://github.com/esflint/IBM-Data-Science-Professional-Certificate---Capstone/blob/master/Data%20Wrangling.ipynb

1 – Review the number of launches at each site, the number of each orbit, and the number of each outcome type:

```
In [5]: # Apply value_counts() on column LaunchSite
        df['LaunchSite'].value_counts()
Out[5]: CCAFS SLC 40    55
        KSC LC 39A      22
        VAFB SLC 4E     13
        Name: LaunchSite, dtype: int64
```

```
In [6]: # Apply value_counts on Orbit column
        df['Orbit'].value_counts()
Out[6]: GTO      27
        ISS      21
        VLEO     14
        PO        9
        LEO       7
        SSO       5
        MEO       3
        ES-L1     1
        HEO       1
        SO        1
        GEO       1
        Name: Orbit, dtype: int64
```

```
In [7]: # landing_outcomes = values on Outcome column
        landing_outcomes = df['Outcome'].value_counts()
        landing_outcomes
Out[7]: 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: Outcome, dtype: int64
```

2 – Categorize a set of the outcomes in which the landing was unsuccessful:

```
In [8]: for i,outcome in enumerate(landing_outcomes.keys()):
            print(i,outcome)
        0 True ASDS
        1 None None
        2 True RTLS
        3 False ASDS
        4 True Ocean
        5 False Ocean
        6 None ASDS
        7 False RTLS
```

```
In [9]: bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
        bad_outcomes
Out[9]: {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

3 – Assign the outcomes to ‘0’ if the landing was unsuccessful, and ‘1’ if the landing was successful:

```
In [11]: # landing_class = 0 if bad_outcome
         # landing_class = 1 otherwise
         landing_class = []
         for key, value in df['Outcome'].items():
             if value in bad_outcomes:
                 landing_class.append(0)
             else:
                 landing_class.append(1)
```

```
In [12]: df['Class']=landing_class
         df[['Class']].head(8)
Out[12]:
```

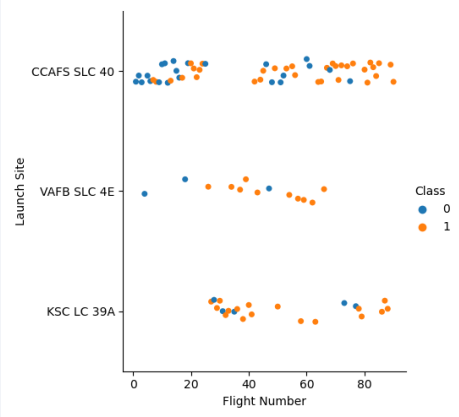
	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

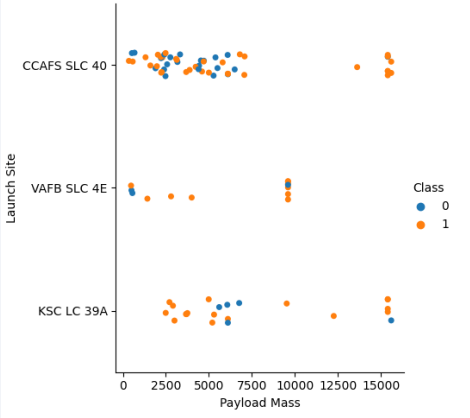
4 – Determine the success rate:

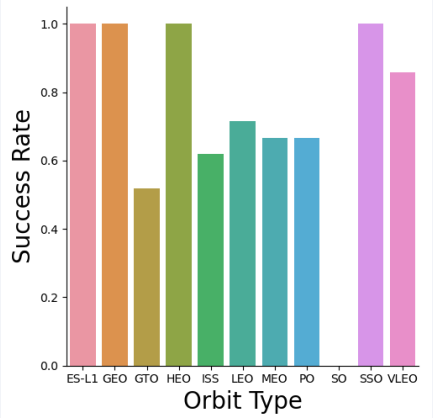
```
In [14]: df["Class"].mean()
Out[14]: 0.6666666666666666
```

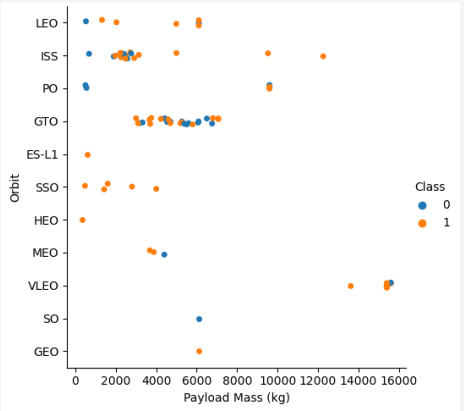
EDA with Data Visualization

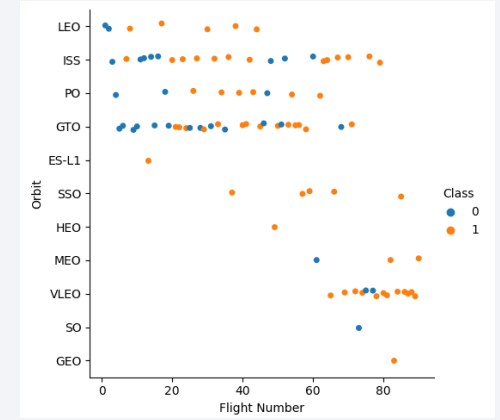
- Github URL: https://github.com/esflint/IBM-Data-Science-Professional-Certificate---Capstone/blob/master/EDA%20with%20Data%20Visualization.ipynb

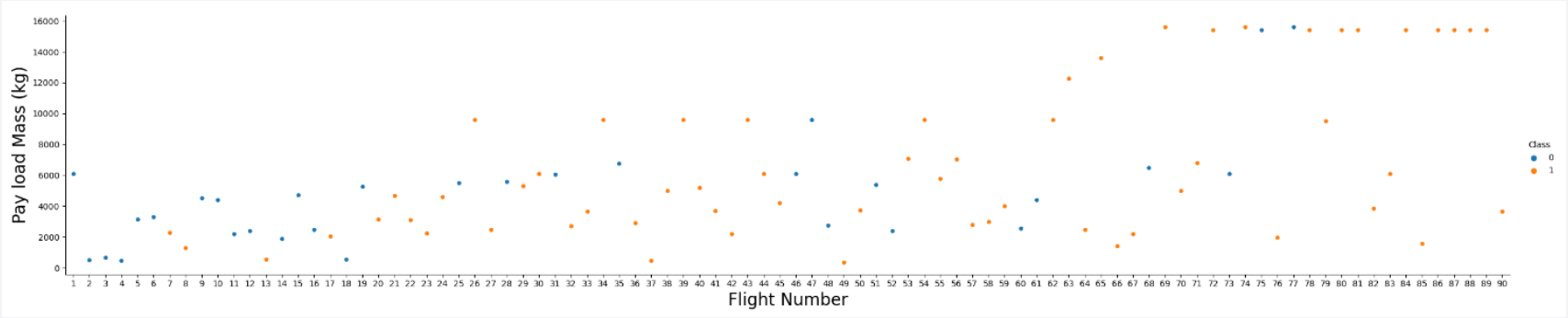


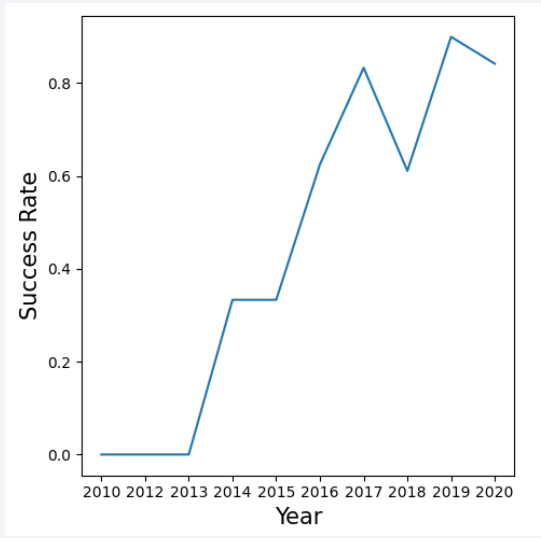












EDA with SQL

Github URL: https://github.com/esflint/IBM-Data-Science-Professional-Certificate---Capstone/blob/master/EDA%20with%20SQL.ipynb

- The following SQL queries were performed:
  - Displayed the names of unique launch sites
  - Displayed 5 records where launch sites begin with the string 'CCA'
  - Displayed the total payload mass carried by boosters launched by NASA (CRS)
  - Displayed average payload mass carried by booster version F9 v1.1
  - Listed the date when the first successful landing outcome in ground pad was achieved
  - Listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
  - Listed the total number of successful and failure mission outcomes
  - Listed the names of the booster versions which have carried the maximum payload mass by using a subquery
  - Listed the failed landing outcomes in drone ship, their booster versions, and the launch site names for in the year 2015
  - Ranked 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.

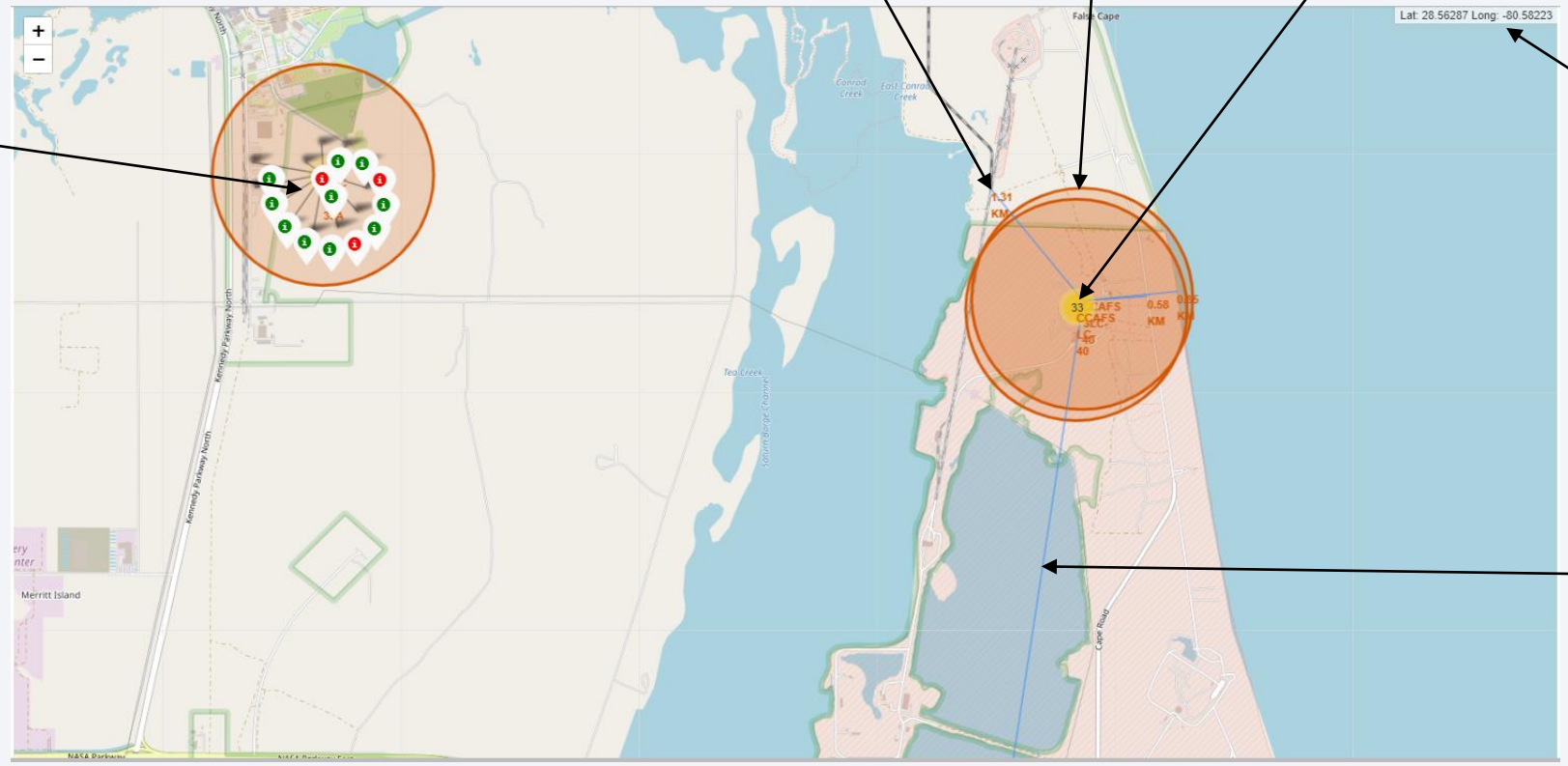
Build an Interactive Map with Folium

Marker – mark a location (leaflet, text, popup, etc)

Circle - circle a location

Marker cluster – combine nearby markers to reduce clutter

Expanded marker cluster with customized marker color based on success or failure



Latitude and longitude of the mouse position on the map

PolyLine connecting two locations on the map to visualize distance

- GitHub URL: https://github.com/esflint/IBM-Data-Science-Professional-Certificate---Capstone/blob/master/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

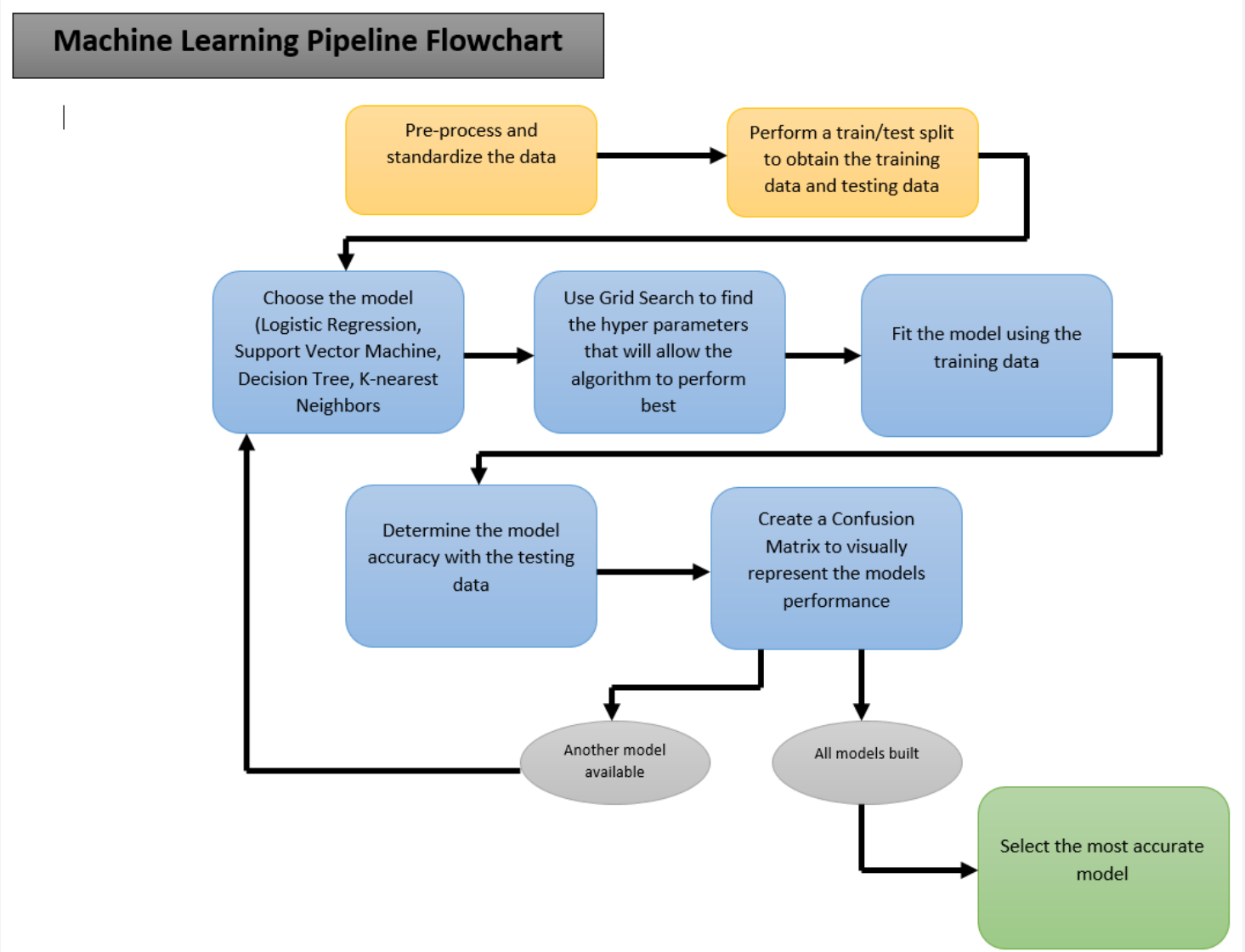
Build a Dashboard with Plotly Dash

- The SpaceX Launch Records Dashboard allows one to interactively investigate the success and failure of launches on criteria of site, booster version, and payload.
- It has two plots/graphs:
  - Pie Chart – shows the successes of all sites or the success and failures of a specific site
  - Scatter Plot – shows the mission outcomes of different booster versions based on payload
- Both plots/graphs can be filtered through a dropdown to view the data from all sites or a specific one
- The scatter plot also has a slider that allows one to filter a payload range.

GitHub URL: https://github.com/esflint/IBM-Data-Science-Professional-Certificate---Capstone/blob/master/spacex_dash_app.py

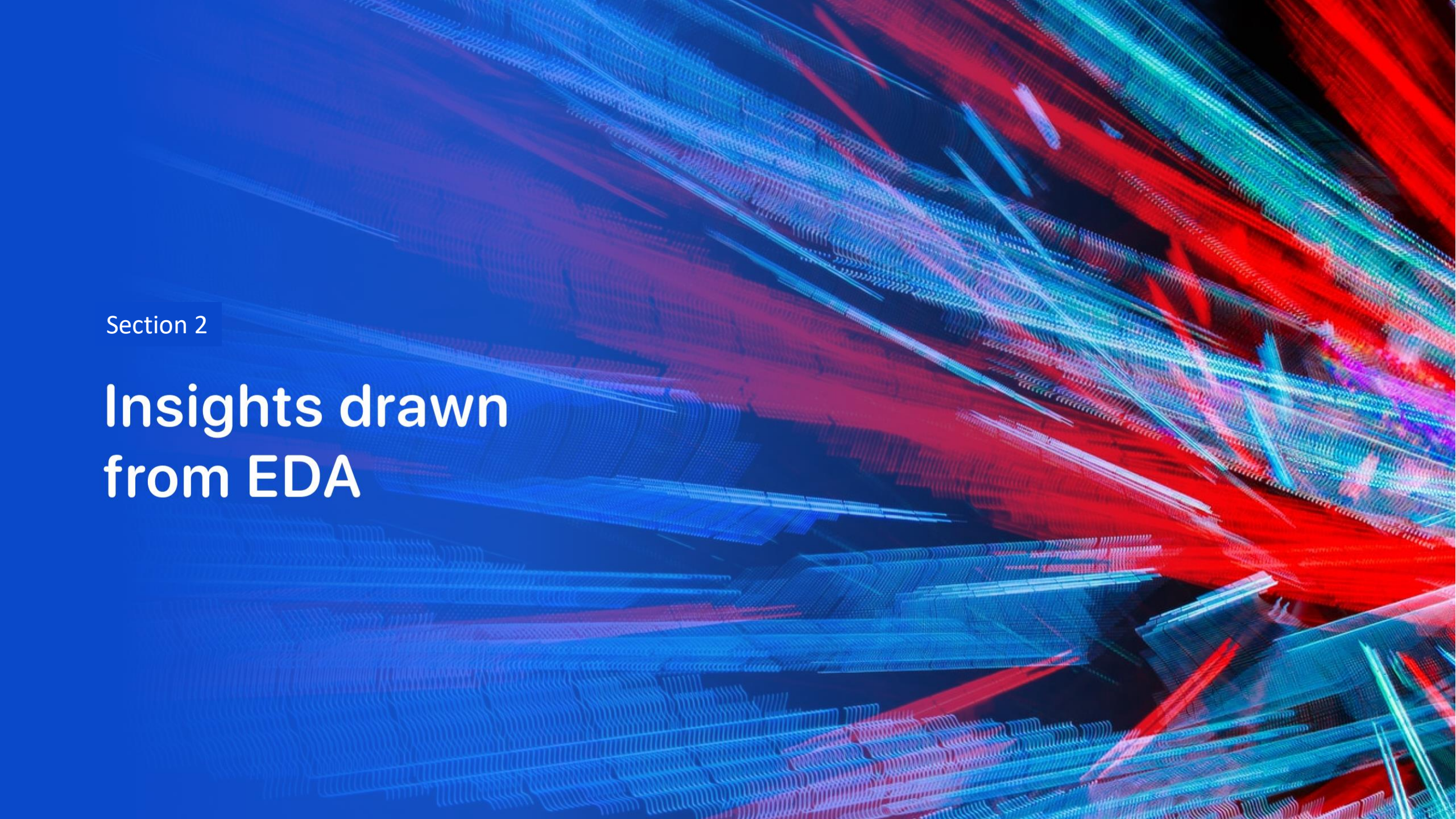
Predictive Analysis (Classification)

GitHub URL:
https://github.com/esflint/IBM-Data-Science-Professional-Certificate---Capstone/blob/master/Machine%20Learning%20Prediction.ipynb



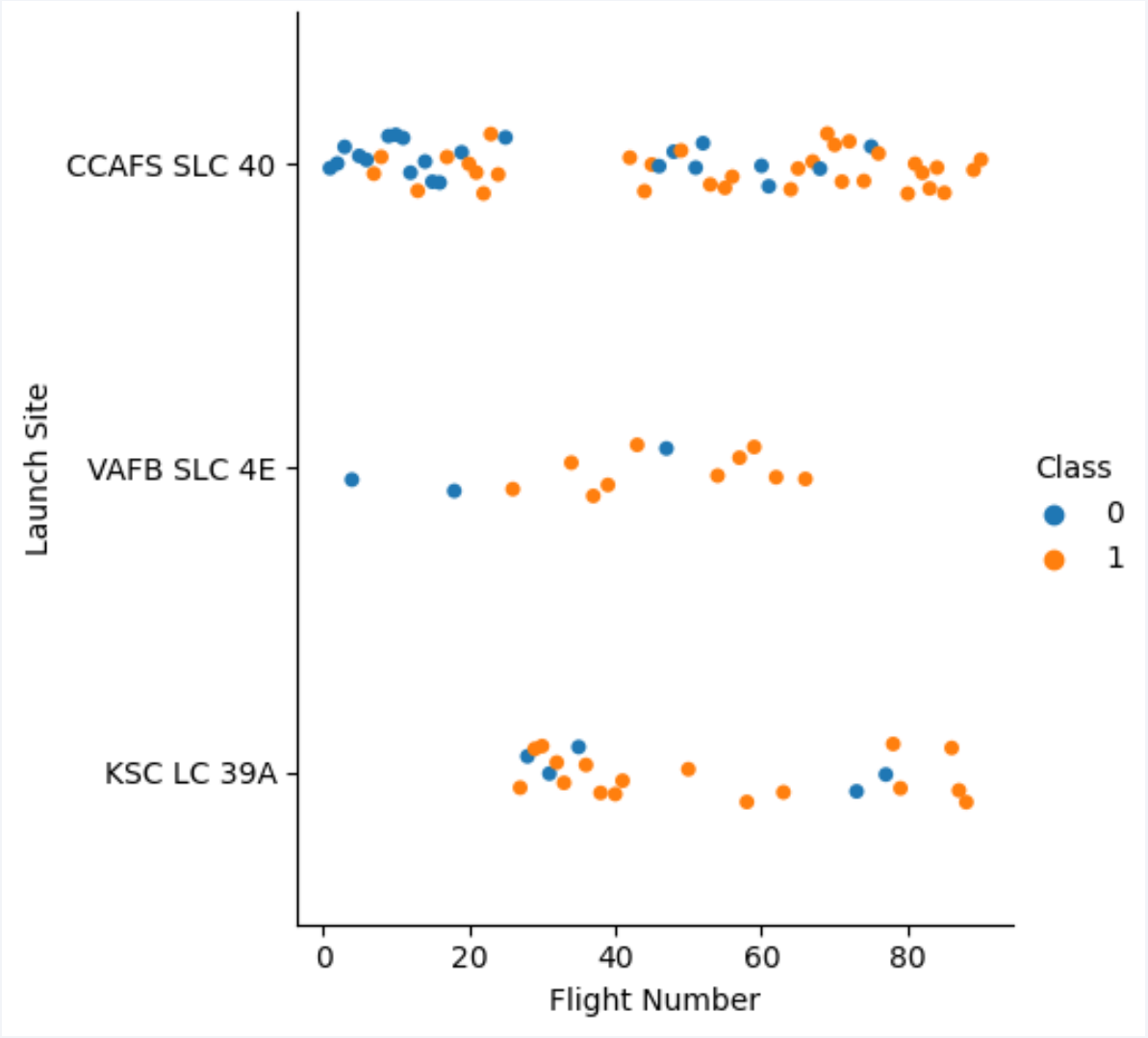
Results

- CCAFS SLC-40 had the most launches
- Orbits ES-L1, GEO, HEO, and SSO had the highest success rates
- The success of the LEO orbit appears to correspond with the number of flights (higher flight number = higher success)
- SpaceX has become more successful in landing Stage 1 over the years
- Lighter payloads tend to have a higher success rate than heavier payloads
- KSC LC-39A site had the highest percentage of successful launches.
- The Decision Tree model ended up with the best testing accuracy. Logistic Regression, Support Vector Machine, and K-nearest Neighbors all performed with the same accuracy.


Section 2
Insights drawn from EDA

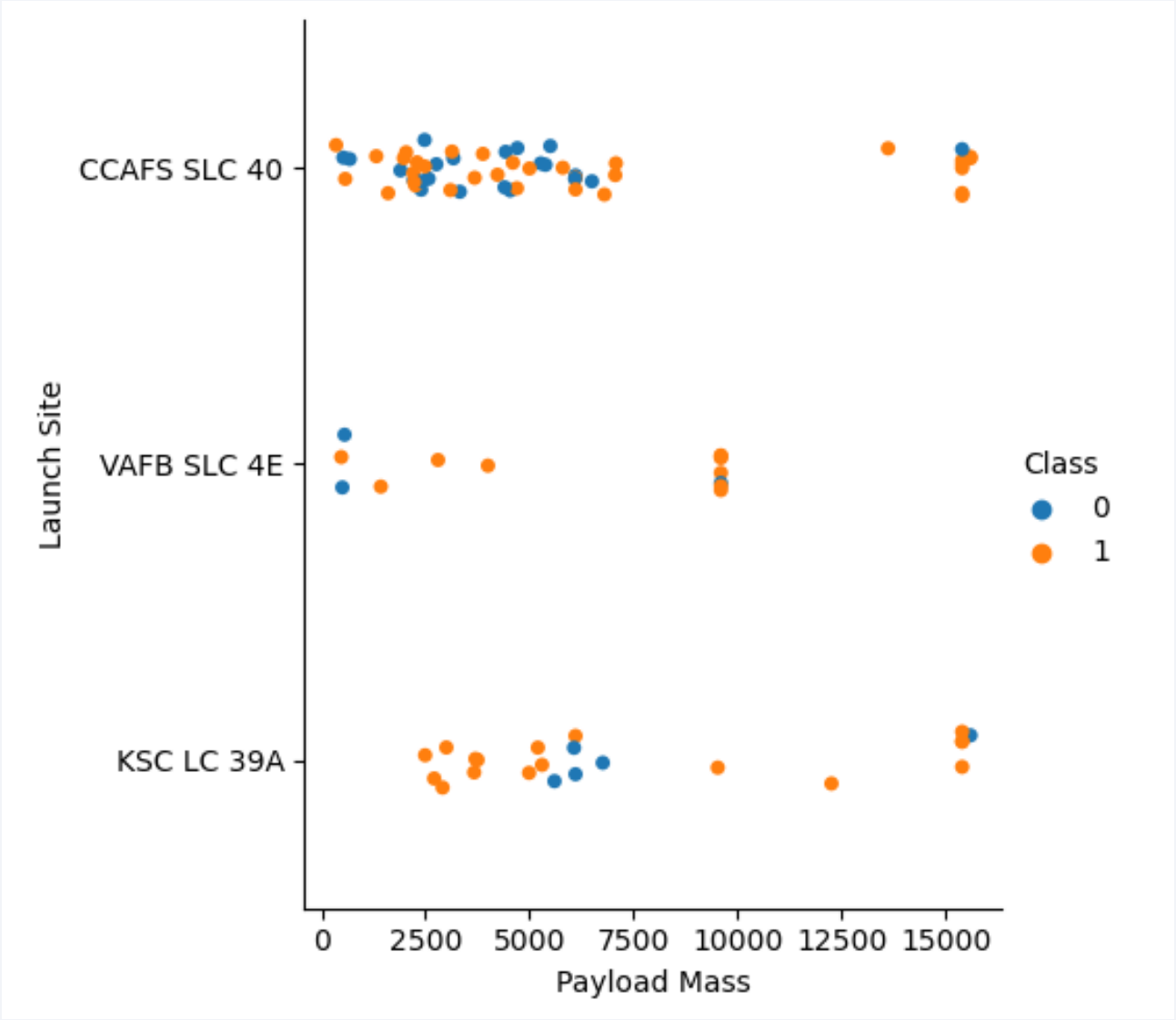
Flight Number vs. Launch Site

- Observations:
  - Launches from site CCAFS SLC 40 outnumber each other the other two sites
  - Launches from VAFB SLC 4E appear to be more successful with a higher flight nubmer



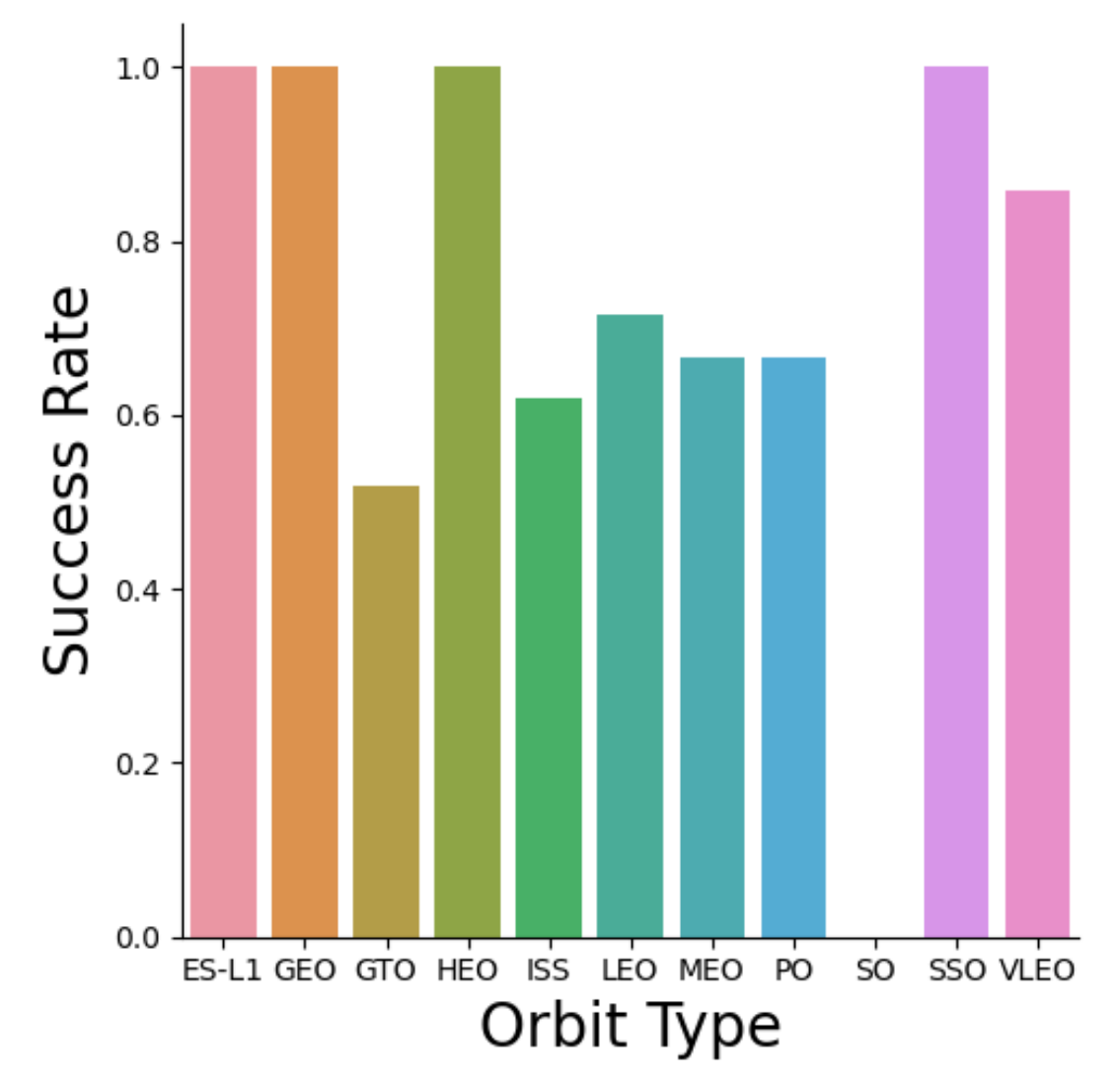
Payload vs. Launch Site

- Observations:
  - No launches of payload > 10000 at VAFB SLC 4E
  - Launches from CCAFS SLC 40 are mainly with payloads of < 7500
  - No launches of payload <2500 at KSC LC 39A



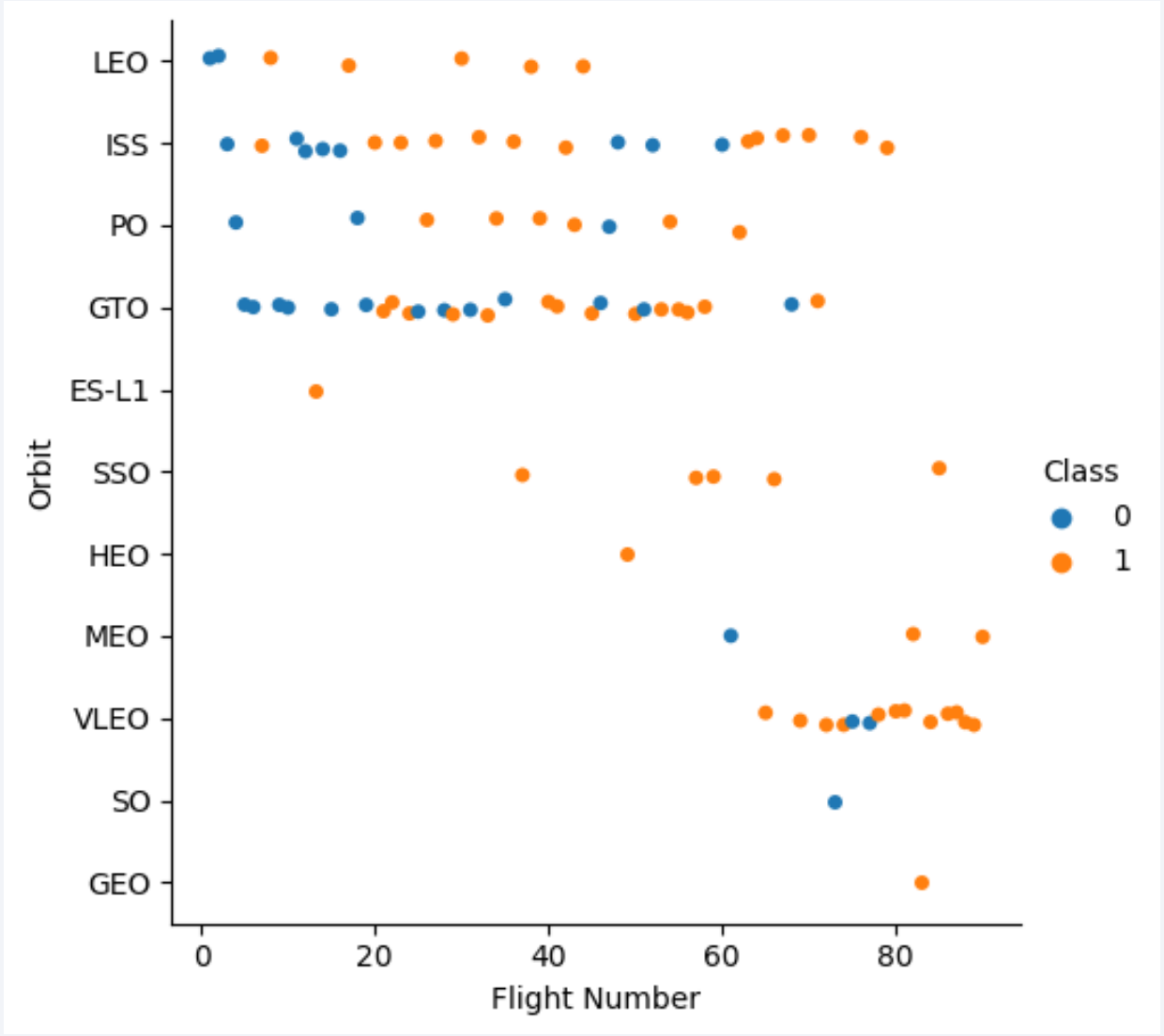
Success Rate vs. Orbit Type

- Observations:
  - Orbit types ES-L1, GEO, HEO, and SSO have the highest success rate (100%)
  - Orbit type GTO has the lowest success rate (approx. 50%)



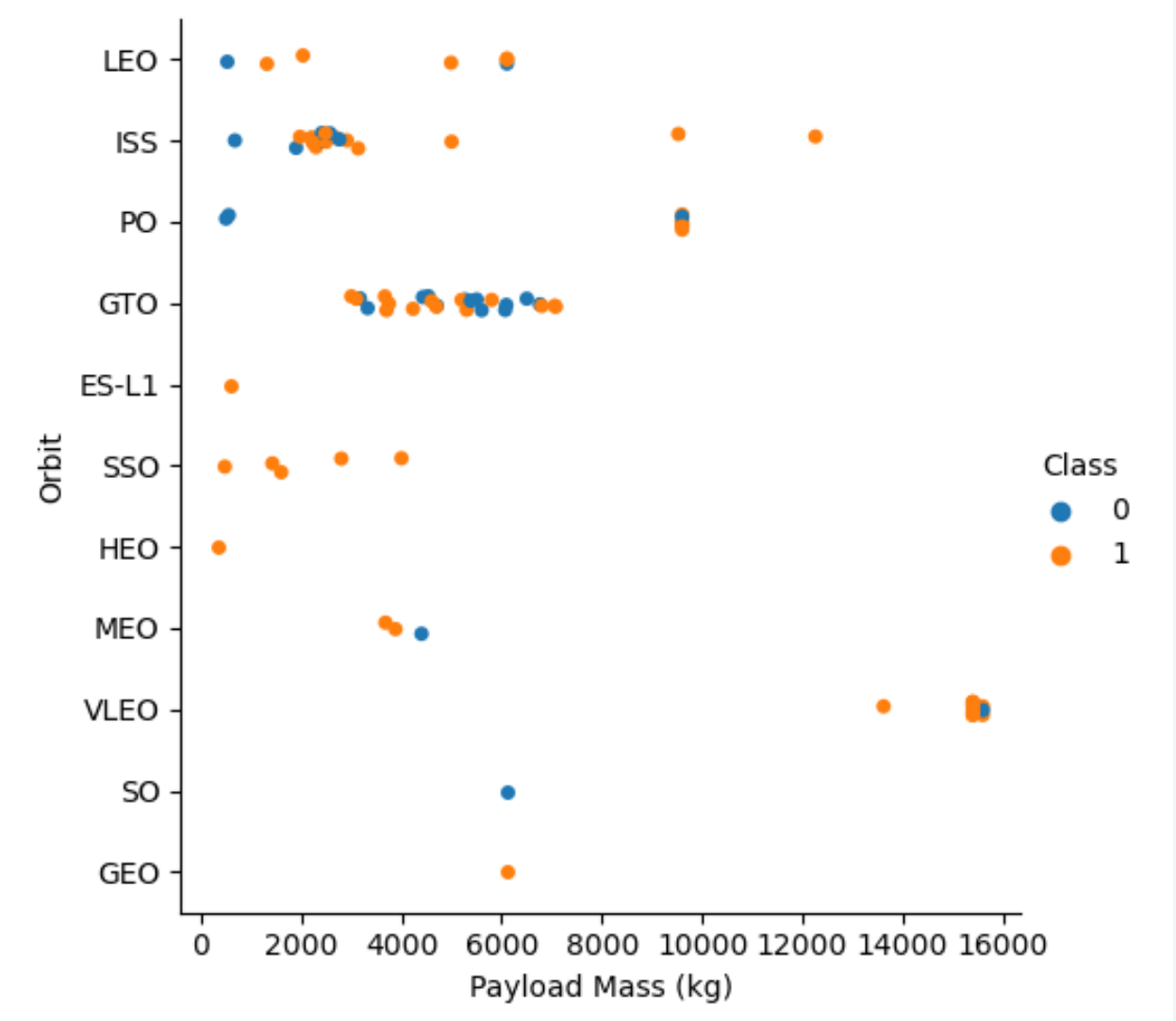
Flight Number vs. Orbit Type

- Observations
  - Success in the LEO orbit appears to be related to # of flights
  - It appears as though there is no relationship with flight number when in GTO orbit



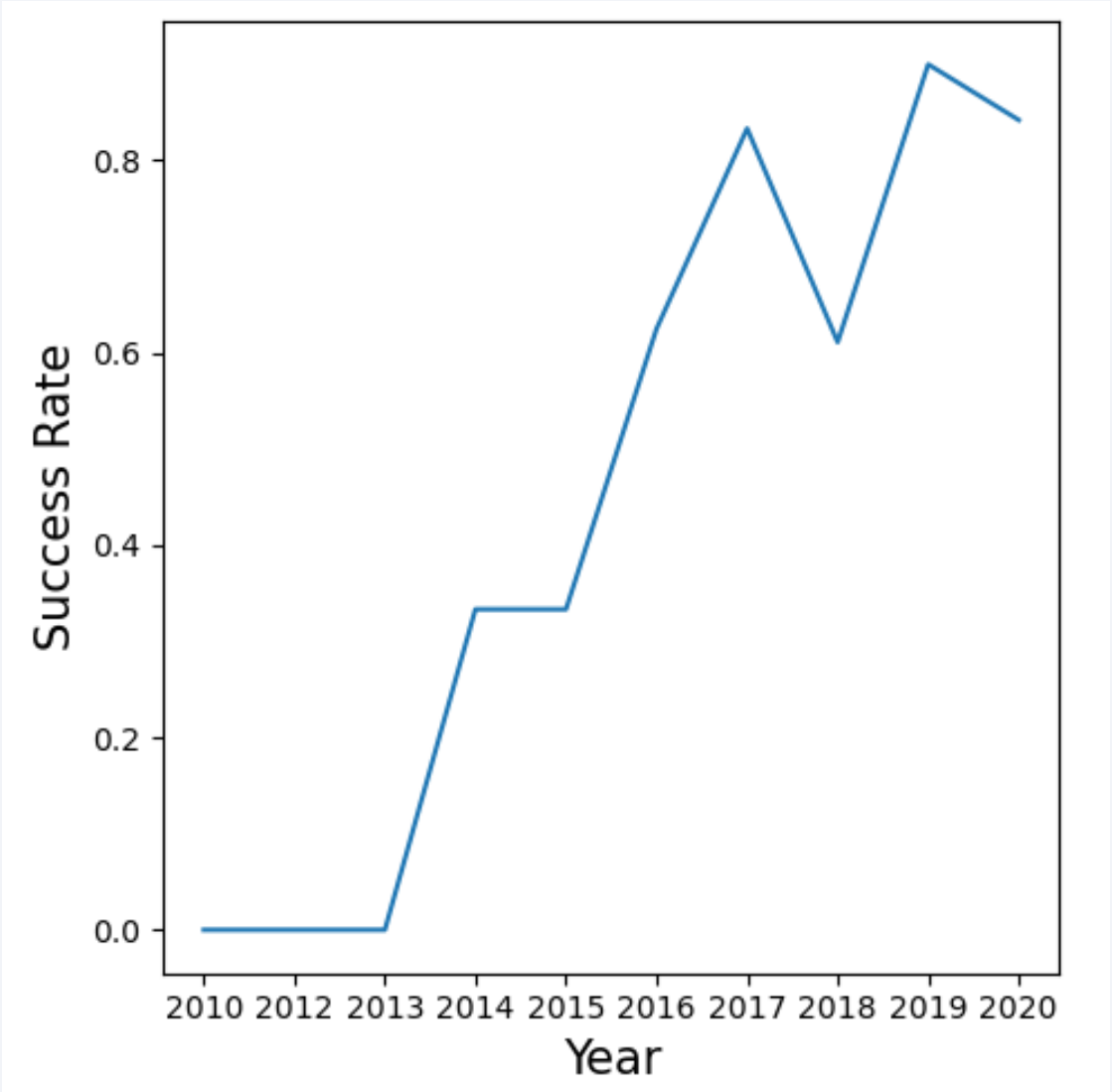
Payload vs. Orbit Type

- Observations:
  - VLEO orbit seems to be best option for success with heavier payloads
  - Payload mass does not seem to affect outcome in the GTO orbit



Launch Success Yearly Trend

- Observations:
  - There were no successful landings prior to 2014
  - Over the years, SpaceX has gotten better at successfully landing stage 1 of launches



All Launch Site Names

- The query returned all distinct launch site names
- The DISTINCT command returns only unique values of the specified column

```
In [7]: %sql select distinct LAUNCH_SITE from SPACEXTBL;
 * sqlite:///my_data1.db
Done.
```

Out[7]:

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

Launch Site Names Begin with 'CCA'

In [8]:
```
%sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5;
```

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

Out[8]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- The query returned the first 5 results that had a launch site that began with ‘CCA’.
- The LIKE clause allows one to return entries that contain a specified value
- ‘%’ wildcard represents zero, one or multiple characters
- The LIMIT clause will return the specified number of entries

Total Payload Mass

```
In [9]: %sql select sum(payload_mass__kg_) as total_payload_mass from SPACEXTBL where customer = 'NASA (CRS)';
 * sqlite:///my_data1.db
Done.
Out[9]: total_payload_mass
--------------------------
                   45596
```

- The query returned the total payload mass of all launches from the customer “NASA (CRS)”
- The SUM function returns the total of the specified numeric column/field
- The AS command is used to provide a temporary name (alias) for the preceding column/function/etc
- The WHERE clause specifies a condition(s) that must be met for a value to be returned (like a filter)

Average Payload Mass by F9 v1.1

```
In [10]: %sql select avg(payload_mass__kg_) as average_payload_mass from SPACEXTBL where booster_version like 'F9 v1.1%';
 * sqlite:///my_data1.db
Done.
Out[10]:
```

average_payload_mass
2534.6666666666665

- The query returned the average payload mass for the specific booster version ‘F9 v1.1’
- The AVG function returns the average value of the specified column/field

First Successful Ground Landing Date

```
In [11]: %sql select min(date) as first_successful_landing from SPACEXTBL where "landing _outcome" = 'Success (ground pad)';
 * sqlite:///my_data1.db
Done.
Out[11]: first_successful_landing
         01-05-2017
```

- The query returned date for the first successful ground pad landing
- The MIN function returns the smallest value of the specified column/field

Successful Drone Ship Landing with Payload between 4000 and 6000

In [12]:
```
%sql select distinct "Booster_Version" from SPACEXTBL where "landing _outcome" = 'Success (drone ship)' and payload_mass__kg_ between 4000 and 6000;
```

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

Out[12]:

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

- The query returned the booster versions for a successful drone ship landing of launches with payloads between 4000 and 6000kg.
- The BETWEEN command will return only values in the specified range

Total Number of Successful and Failure Mission Outcomes

```
In [13]: %sql select mission_outcome, count(*) as total_number from SPACEXTBL group by mission_outcome;
 * sqlite:///my_data1.db
Done.
```

Out[13]:

Mission_Outcome	total_number
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- The query returned the totals for each mission outcome
- The COUNT function returns a count/number of rows that matches the specified criteria
- The GROUP BY clause is used to arrange/group similar data by a specific column(s)

Boosters Carried Maximum Payload

```
In [14]: %sql select booster_version from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL);
 * sqlite:///my_data1.db
Done.
```

Out[14]:

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

- The query returned the booster versions that carried the largest documented payload

- A subquery is a SELECT query that is nested inside a main SELECT statement

2015 Launch Records

In [15]:

```
%sql select substr(Date, 4, 2) as month, date, booster_version, launch_site, "landing _outcome" from SPACEXTBL where "landing _outcome" = 'Failure (drone ship)' and substr(Date,7,4)='2015';
```

Out[15]:

month	Date	Booster_Version	Launch_Site	Landing _Outcome
01	10-01-2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	14-04-2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

- The query returned rows of specific selected columns for which a drone ship landing failed in 2015.

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

In [16]:
```
%%sql select "landing _outcome", count(*) as count_outcomes from SPACEXTBL
    where date between '04-06-2010' and '20-03-2017'
    group by "landing _outcome"
    order by count_outcomes desc;
```

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

Out[16]:

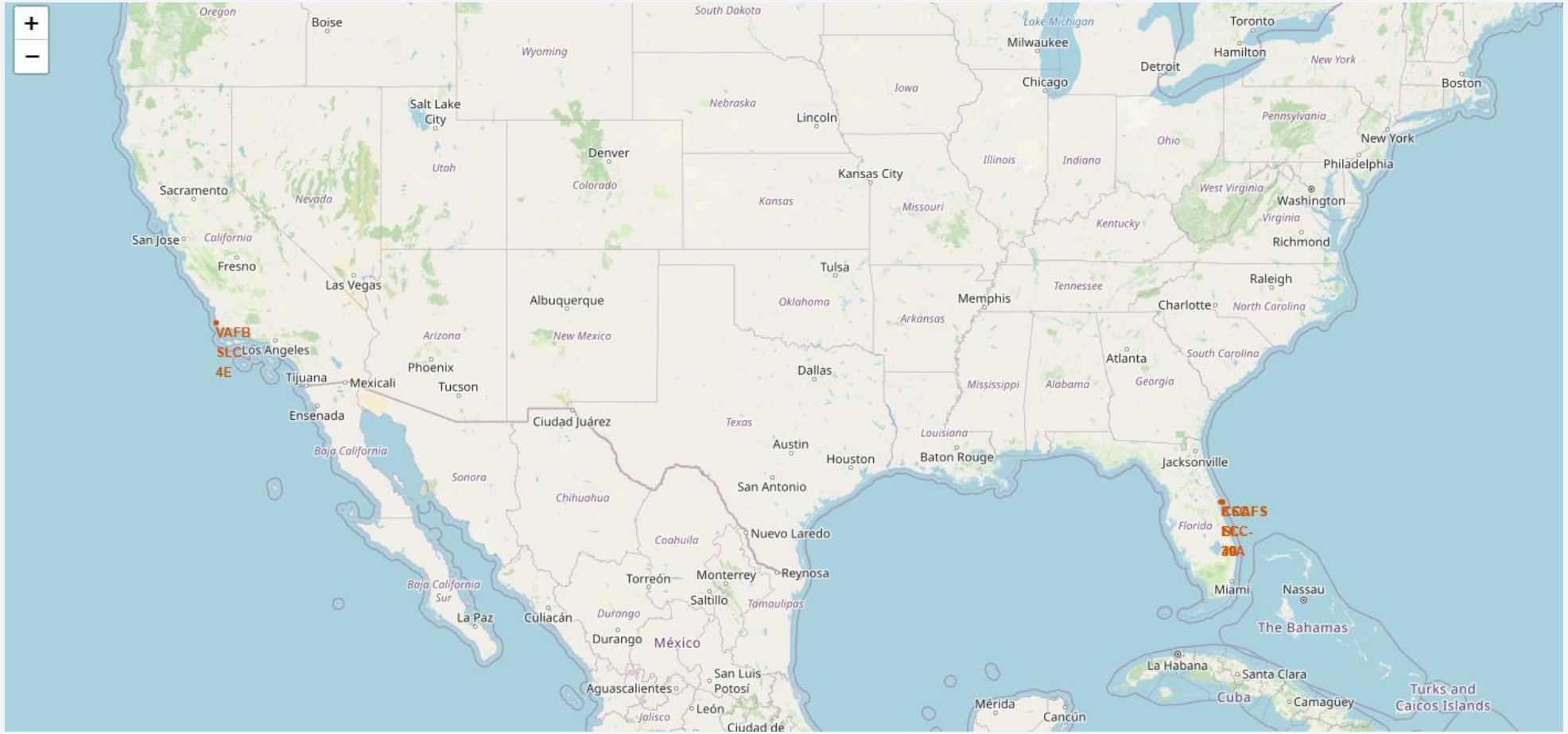
Landing _Outcome	count_outcomes
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

- The query returned the count of each landing outcome between two sets of dates presented in descending order

- The ORDER BY clause sorts the data either ascending or descending based on the specified column(s)

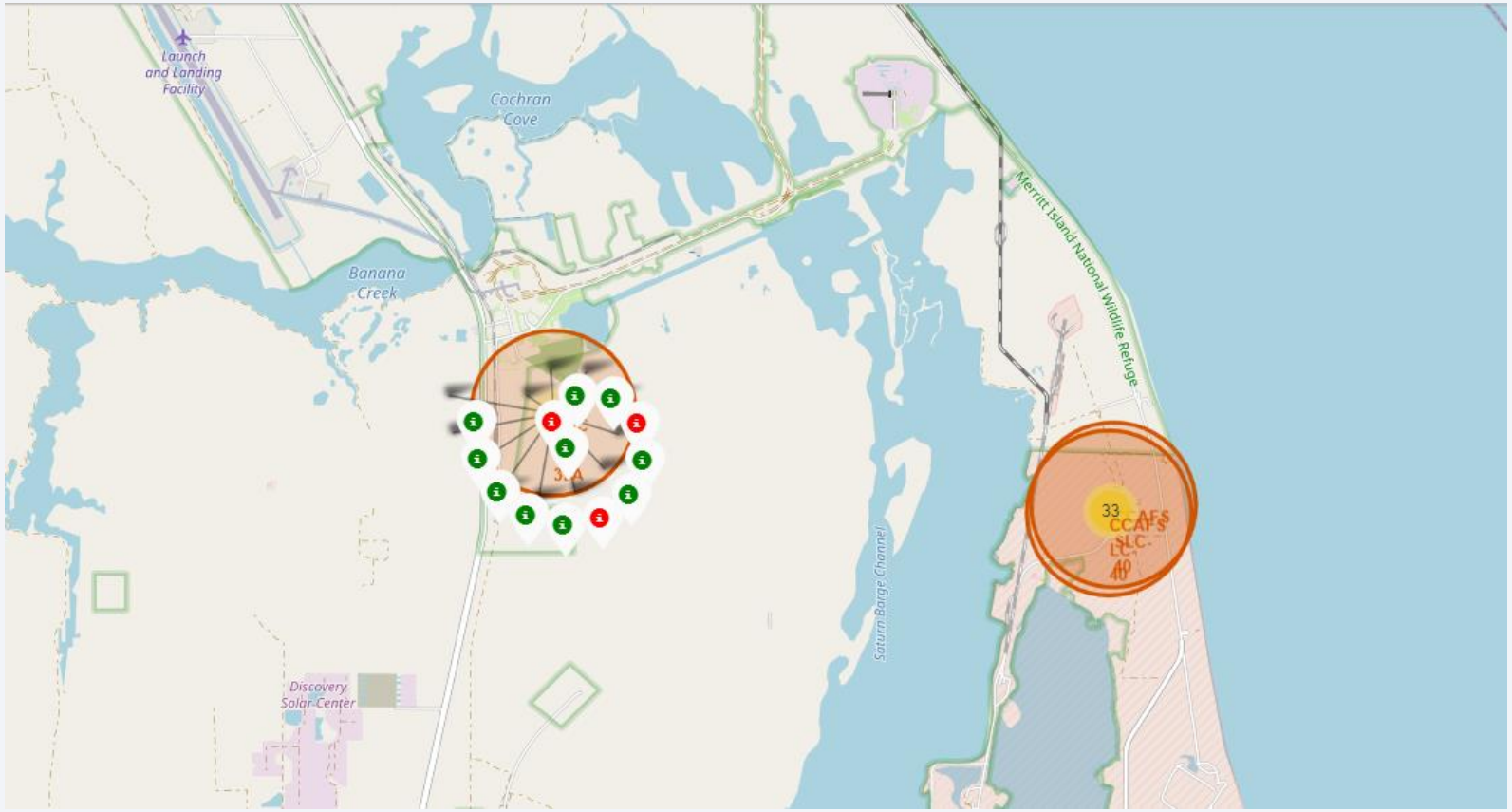

Section 3
Launch Sites
Proximities Analysis

Map of Launch Site Locations



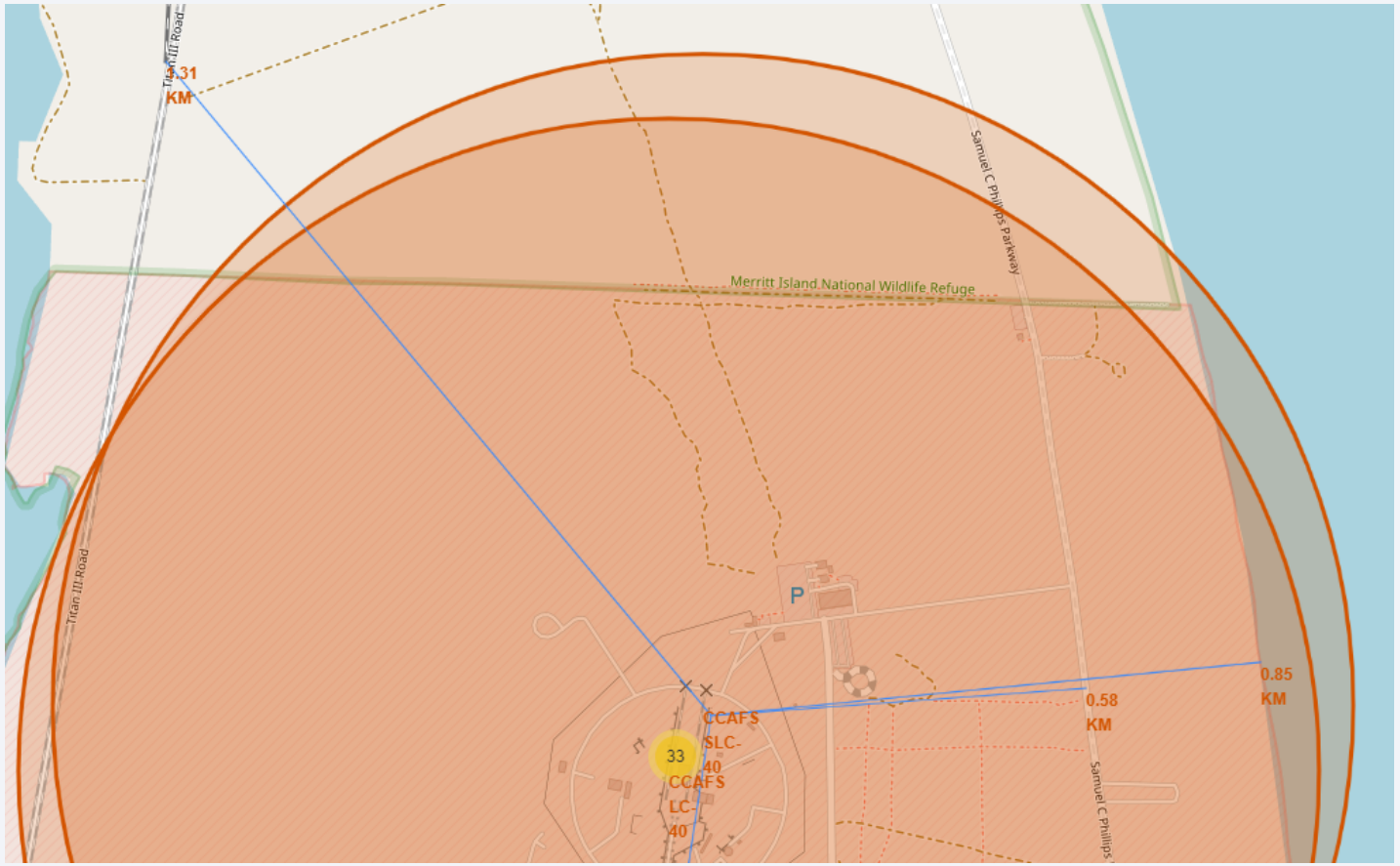
The map highlights the Launch Site locations with both a Circle (with label popup) and a Marker (site name).

Map of Launch Outcomes

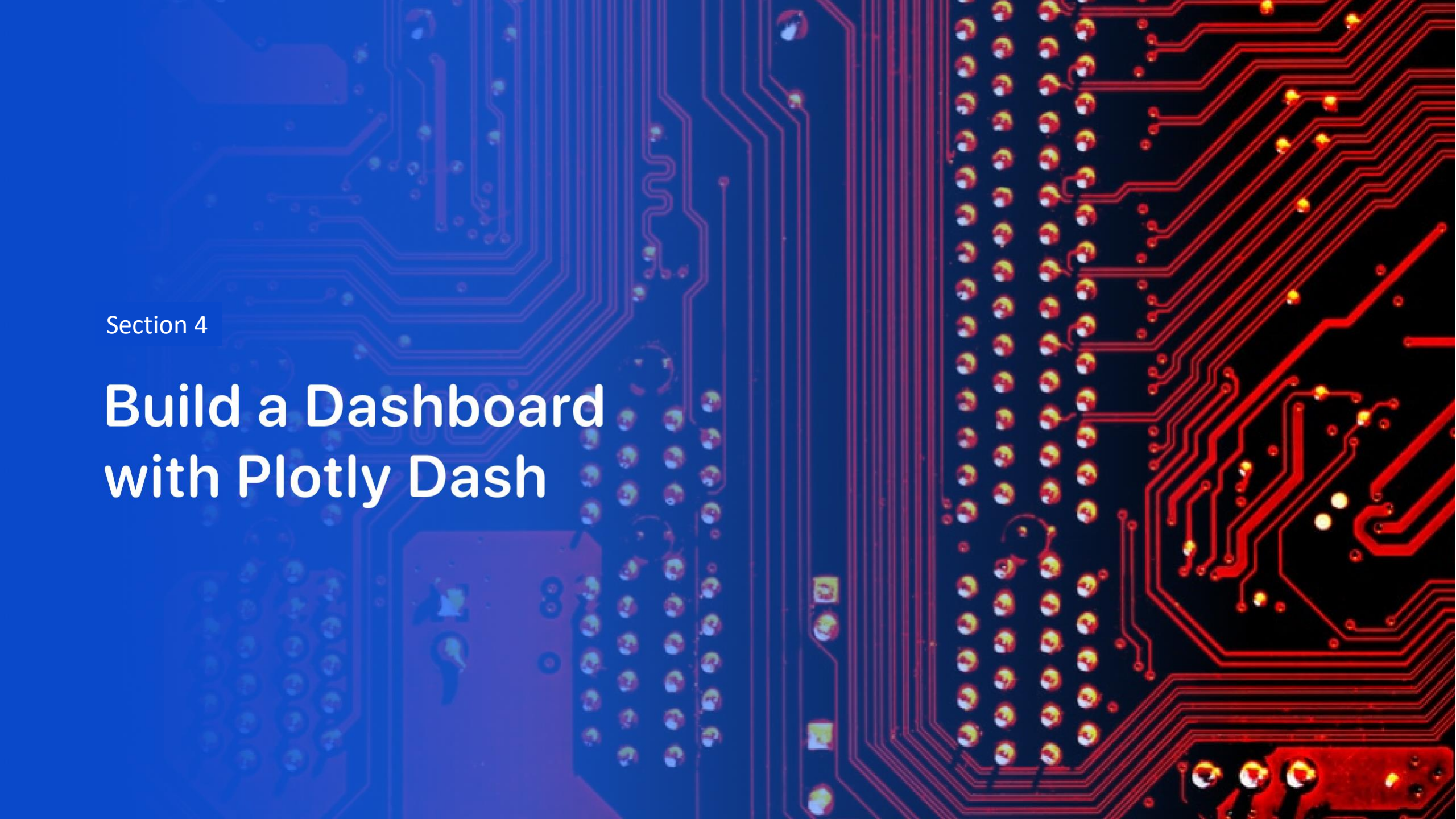


- The map shows both a Cluster Marker not expanded (right) and when it is expanded (left). A Cluster Marker is used to combine nearby markers to reduce clutter.
- The leaflet markers on the left are further customized with a Red Info symbol indicating a failed launch and a Green Info symbol indicating a successful launch. In this case, 10 of 13 cases from KSC LC-39A were successful.

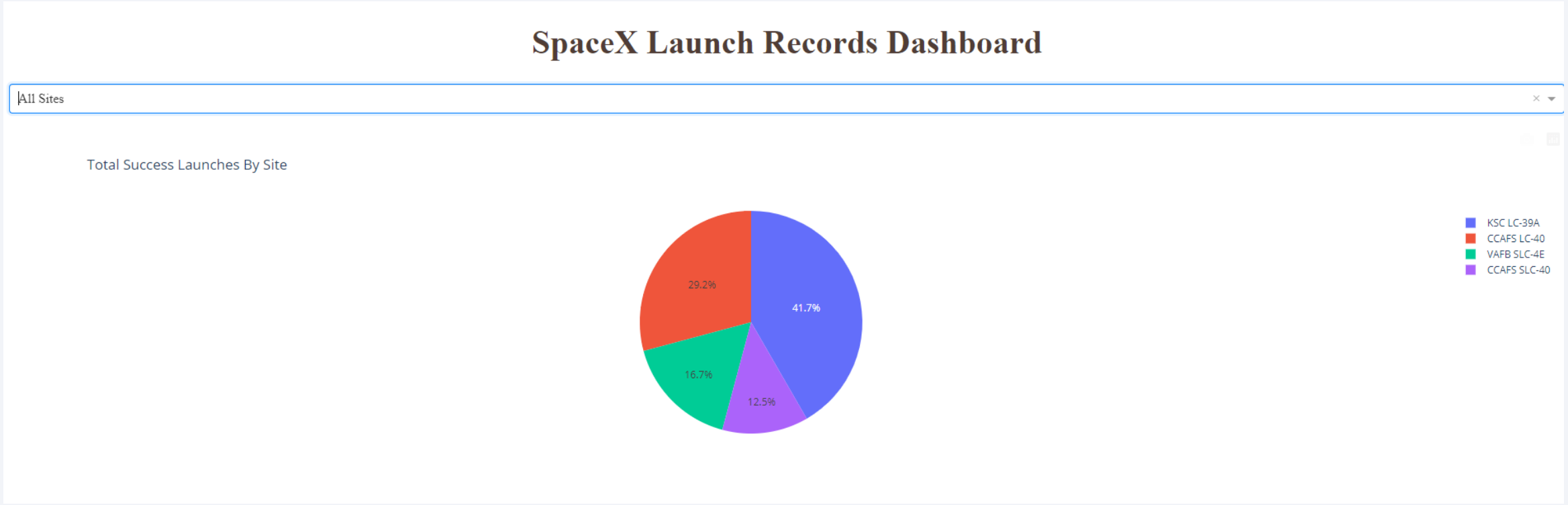
Map showing Proximity of Launch Site to Other Locations



- The blue lines show the path from the launch site (CCAFS SLC-40) to the other location. A Marker with the distance in km from the launch site is visible at the other location.
- In the image you can see that the railway is 1.31 km from CCAFS SLC-40, the highway is 0.58 km from CCAFS SLC-40, and the coastline is 0.85 km from CCAFS SLC-40. The distance to a city is not shown in this screenshot.

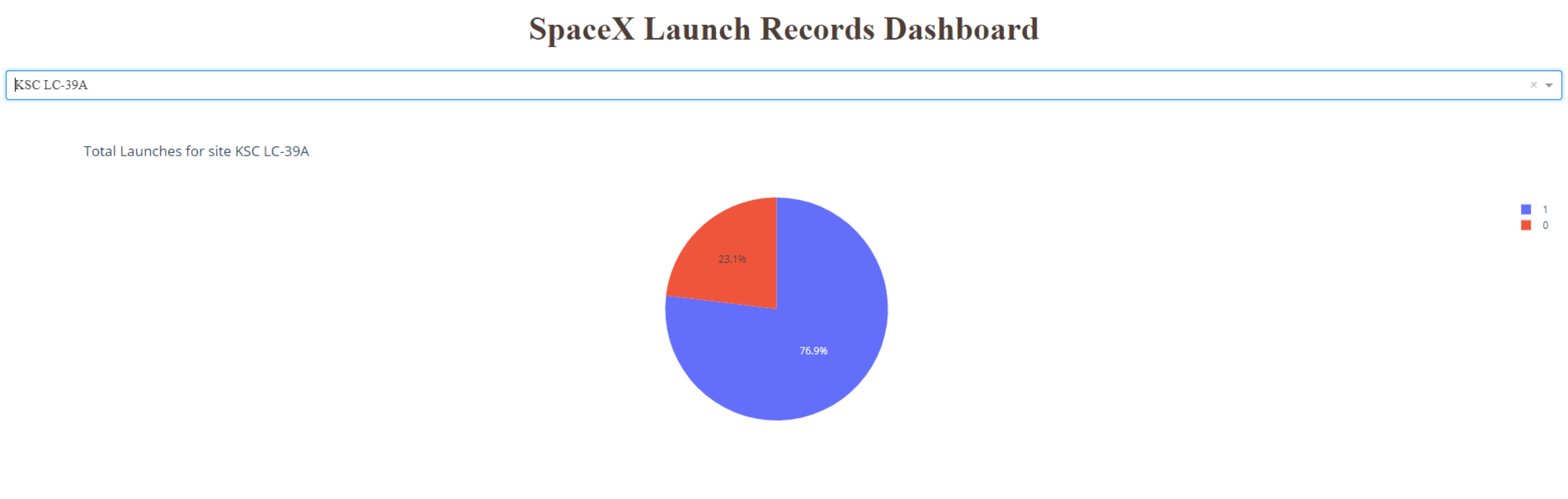

Section 4
Build a Dashboard with Plotly Dash

SpaceX Dashboard Pie Chart - Successful Launches by Site



- The dropdown menu is set on “All Sites” creating a pie chart that shows the percentage of the successful launches from each site.
- KSC LC-39A site had the highest percentage of successful launches.

SpaceX Dashboard Pie Chart – Site with Highest Launch Success Ratio



- The dropdown menu is set to “KSC LC-39A” which is the site that had the highest launch success ratio.
- The majority of the pie chart (76.9%) is Blue, representing a “1” which indicates a successful launch.

SpaceX Dashboard Scatter Plot - Payload vs. Launch Outcome

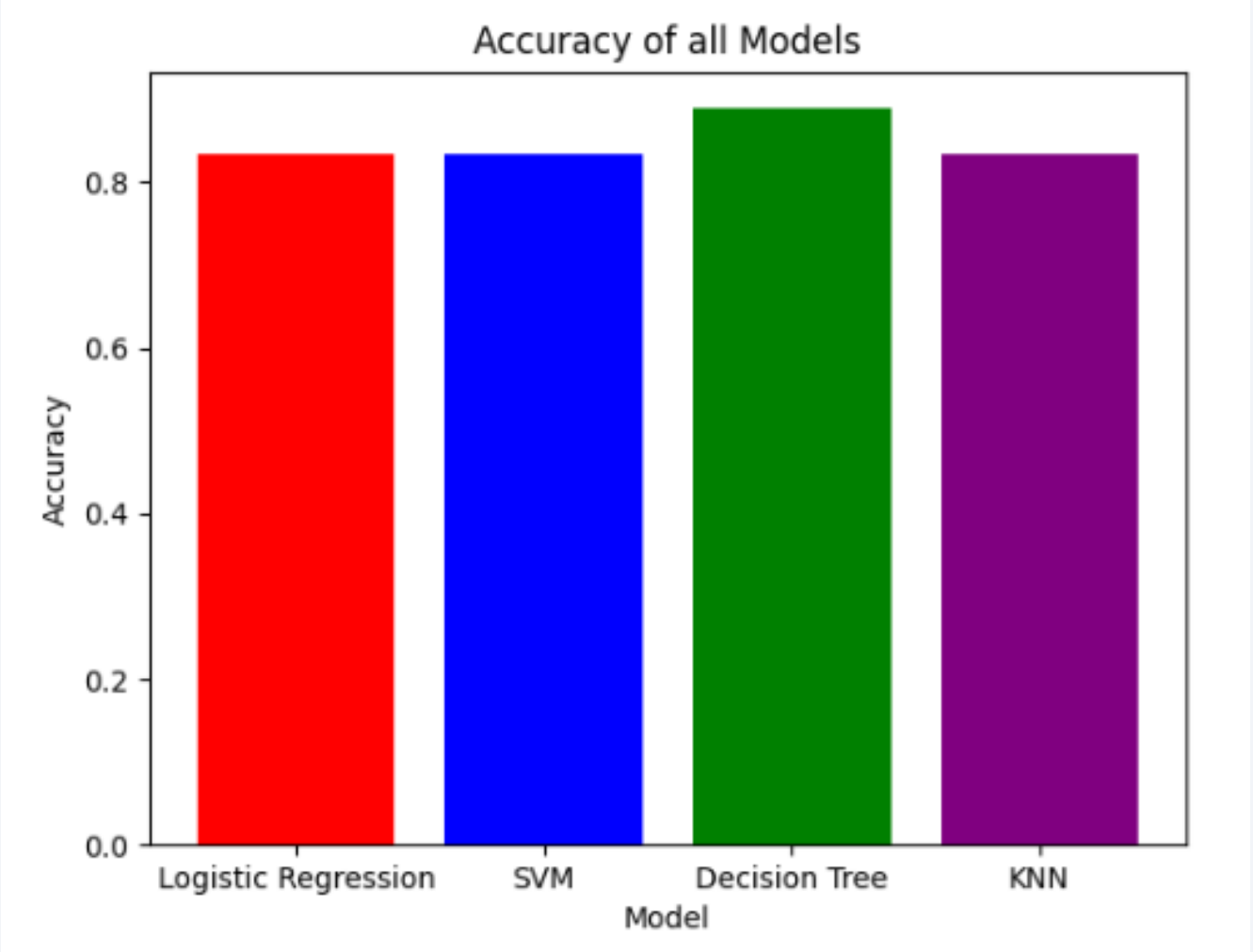
- The first scatter plot shows the results from all sites when the payload mass is between 2,000kg and 8,000kg. One observation is that the FT booster has better success as the payload mass decreases.
- The second scatter plot shows the results from all sites when the payload mass is between 6,000kg and 10,000kg. One observation is that launches at these payloads are more likely to fail and there are only 2 booster versions that have launched with this range of payload.

Section 5

Predictive Analysis (Classification)

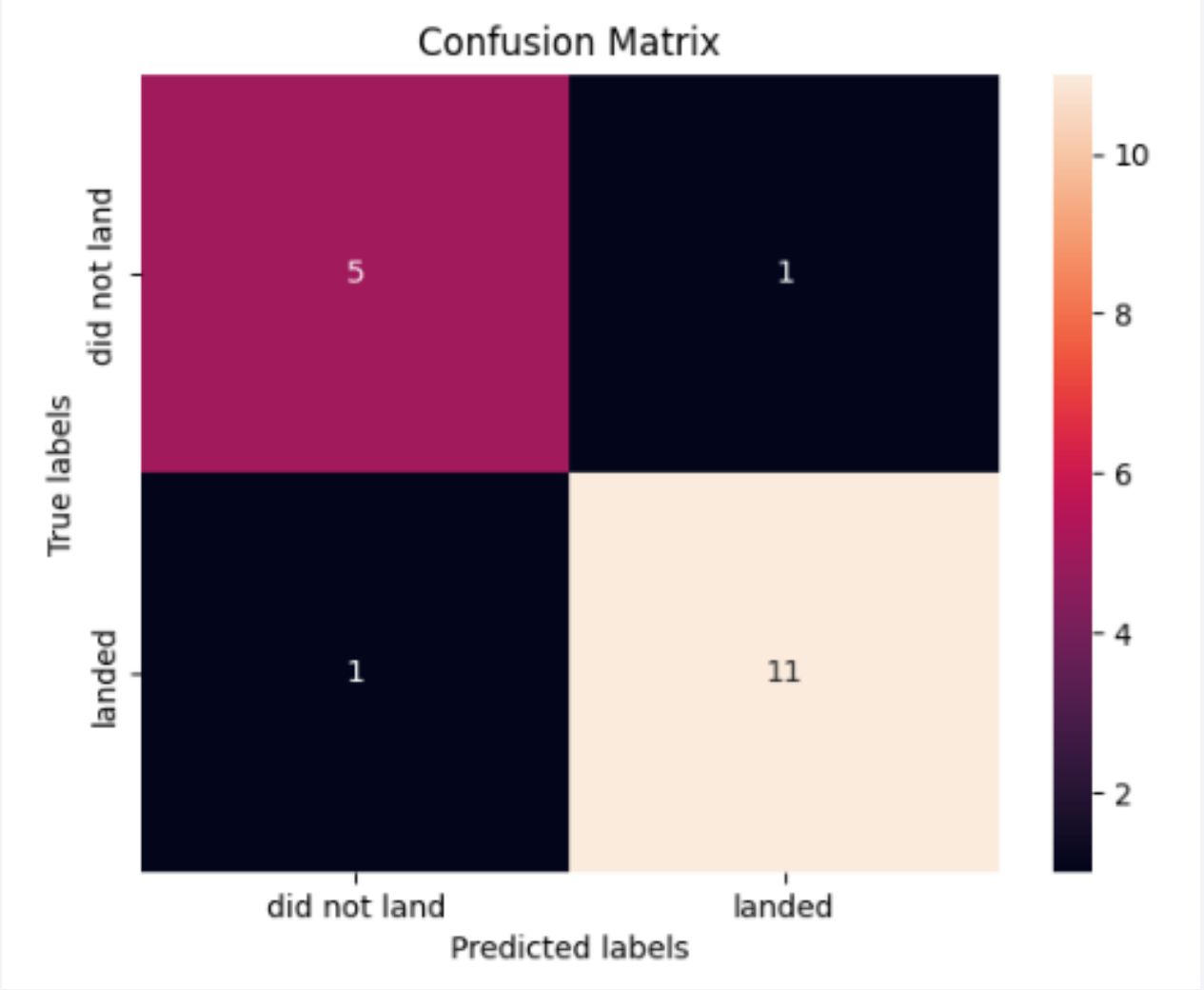
Classification Accuracy

- The highest classification accuracy of approximately 88.88% is found with the Decision Tree model.
- The remaining models (Logistic Regression, SVM, KNN) all had the same classification accuracy of approximately 83.34%.



Confusion Matrix - Decision Tree

- 11 predicted as “landed” were actually “landed” (true positives)
- 5 predicted as “did not land” were actually “did not land” (true negatives)
- 1 predicted as “did not land” was actually “landed” (false negative)
- 1 predicted as “landed” was actually “did not land” (false positive)



Conclusions

- The Decision Tree model ended up producing the highest test accuracy for the current data set.
- A lighter payload is more likely than a heavier payload to have a successful stage 1 landing
- ES-L1, GEO, HEO, and SSO are the most successful orbits at landing stage 1
- KSC LC-39A is the most successful site at landing stage 1
- SpaceX has continuously improved their success rate over the years

Appendix

All Python code snippets, SQL queries, charts, Notebook outputs, or data sets that have been created during this project can be found in the following GitHub repository:

https://github.com/esflint/IBM-Data-Science-Professional-Certificate---Capstone

Thank you!