IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

By: Brandon Eley
July 16, 2025

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis with Data Visualization
  - Exploratory Data Analysis with SQL
  - Building an Interactive Map with Folium
  - Building an Interactive Dashboard with Dash
  - Predictive Analysis
- Summary of all results
  - EDA Results
  - Interactive Analytics
  - Predictive Analytics

Introduction

- Background and Context

SpaceX is the most successful commercial space exploration company to date. Overtime, they've been able to successfully launch more and more rockets at a lower cost point than competing companies (65 mil versus 165 million per rocket). The reason they've achieved this cost point is their ability to reuse the first stage in takeoff. SpaceX has amassed 10+ years of publicly available data on each rocket launch spanning various launch sites. We have set out to determine based on this data if we can predict whether a launch will be successful.

- Questions to be answered
  - What variables affect the success of landing at the first stage?
  - Can we use machine learning to anticipate the success of a launch?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web scrapping from Wikipedia
- Perform data wrangling
  - Missing/Null Values
  - One Hot Encoding for categorical variables
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression, Support Vector Machines, Decision Tree, and K-Nearest Neighbor

Data Collection

SpaceX REST API:

1. Requested and parsed the data using the GET request
2. Normalized the data to be used to construct a library to be stored in a Pandas Data Frame
3. Filtered the data frame to only include Falcon 9 launches for the project analysis
4. Cleaned the data and replaced missing values

Web scrapping Wikipedia:

1. Requested data using an HTTP GET request and BeautifulSoup
2. Extract all column/variable names from the HTML table header
3. Cleaned data, created empty dictionary and appended column names
4. Created a data frame by parsing the launch HTML table

Data Collection – SpaceX API

1

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

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
[66]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-
```

We should see that the request was successfull with the 200 status response code

```
[67]: response=requests.get(static_json_url)

[68]: response.status_code

[68]: 200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
[69]: # Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

2

Finally lets construct our dataset using the data we have obtained. We we combine the columns into a dictionary.

```
[79]: 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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

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

3

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

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
[82]: # Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data['BoosterVersion'] == 'Falcon 9']

data_falcon9.head()
```

[82]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFin
4	6	2010-06	Falcon 9	NaN	LEO	CCSFS SLC	None	1	Fals

Now that we have removed some values we should reset the FlightNumber column

```
[83]: data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

4

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

```
[91]: # Calculate the mean value of PayloadMass column
mean = data['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data['PayloadMass'].replace(np.nan,mean,inplace=True)
data.head()
data_falcon9.isnull().sum()
```

```
[91]: FlightNumber       0
Date               0
BoosterVersion     0
PayloadMass        0
Orbit              0
LaunchSite         0
Outcome            0
Flights            0
GridFins           0
Reused             0
Legs               0
LandingPad        26
Block              0
ReusedCount        0
Serial             0
Longitude          0
Latitude           0
dtype: int64
```

https://github.com/ChildishB/IBM-DS-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection - Scraping

1

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

[6]:
```
# use requests.get() method with the provided static_url

response = requests.get(static_url).text

# assign the response to a object
```

Create a `BeautifulSoup` object from the HTML `response`

[9]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content

soup = BeautifulSoup(response,'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

11]:
```
# Use soup.title attribute

print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches – Wikipedia</title>
```

2

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

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

[13]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all("table")
```

Starting from the third table is our target table contains the actual launch records.

[14]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
```

3

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

[17]:
```
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']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.

4

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

[ ]:
```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

https://github.com/ChildishB/IBM-DS-Capstone/blob/main/jupyter-labs-webscraping.ipynb

Data Wrangling

1.Calculate the number of launches at each site

2. Calculate the number of occurrences of each orbit

3. Calculate the number and occurrence of mission outcome of the orbits

4. Create a landing outcome label from Outcome called “Class”

5. The mean of the created label column values is the Success Rate

Data Wrangling

1

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

```
[6]: # Apply value_counts() on column LaunchSite
     df['LaunchSite'].value_counts()
```

```
[6]: LaunchSite
     CCAFS SLC 40    55
     KSC LC 39A      22
     VAFB SLC 4E     13
     Name: count, dtype: int64
```

2

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

```
[7]: # Apply value_counts on Orbit column

     df['Orbit'].value_counts()
```

```
[7]: Orbit
     GTO      27
     ISS      21
     VLEO     14
     PO        9
     LEO       7
     SSO       5
     MEO       3
     HEO       1
     ES-L1     1
     SO        1
     GEO       1
     Name: count, dtype: int64
```

3

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

```
[10]: # landing_outcomes = values on Outcome column

      landing_outcomes = df['Outcome'].value_counts()

      landing_outcomes
```

```
[10]: Outcome
      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: count, dtype: int64
```

4

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome` , create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome` ; otherwise, it's one. Then assign it to the variable `landing_class` :

```
[13]: # landing_class = 0 if bad_outcome
      # landing_class = 1 otherwise

      landing_class =[]
      for outcome in df['Outcome']:
          if outcome in bad_outcomes:
              landing_class.append(0)
          else:
              landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
[14]: df['Class']=landing_class
      df[['Class']].head(8)
```

```
[14]:   Class
      0     0
```

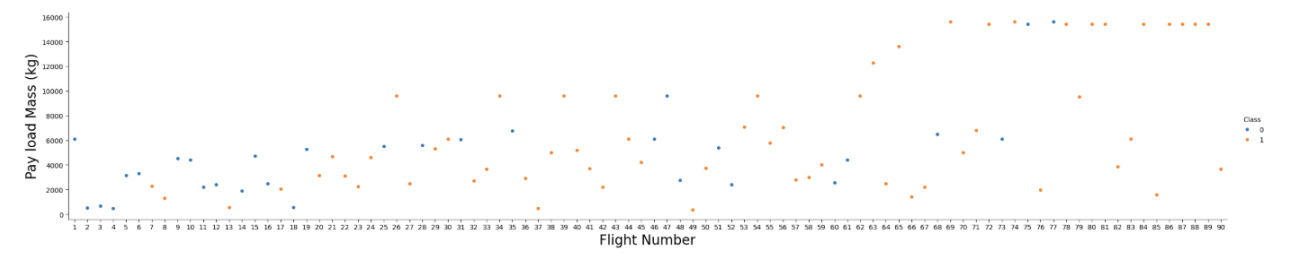
We can use the following line of code to determine the success rate:

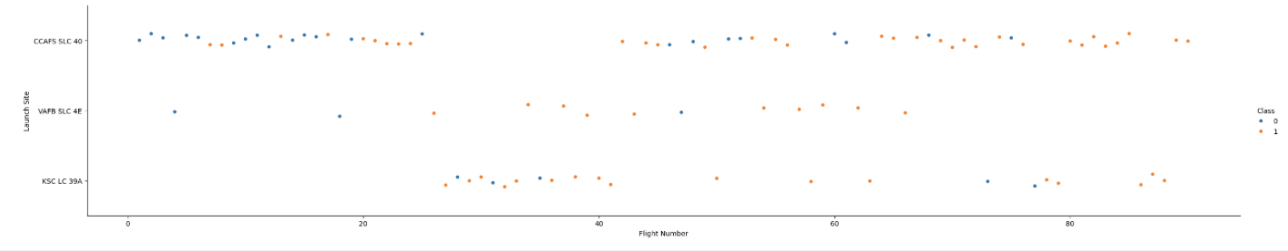
```
:  df["Class"].mean()

   df.to_csv("dataset_part_2.csv", index=False
```

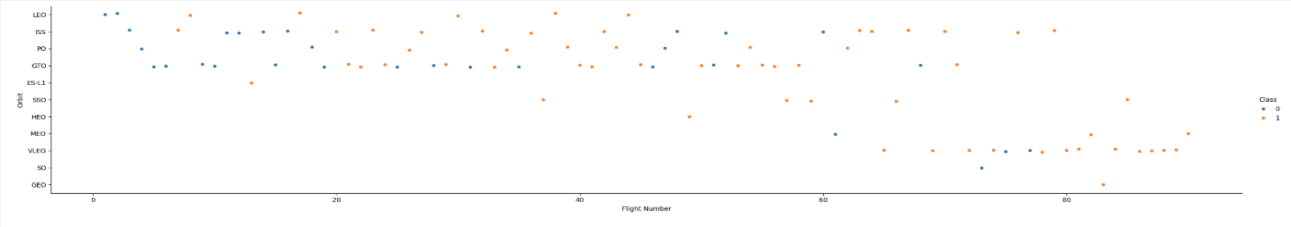
```
:  np.float64(0.6666666666666666)
```

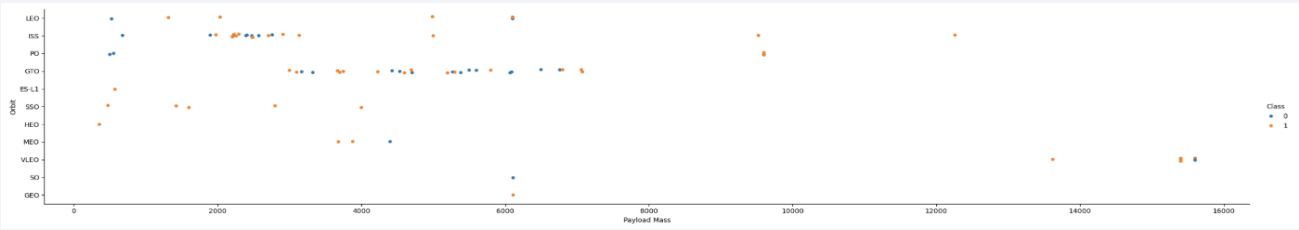

https://github.com/ChildishB/IBM-DS-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

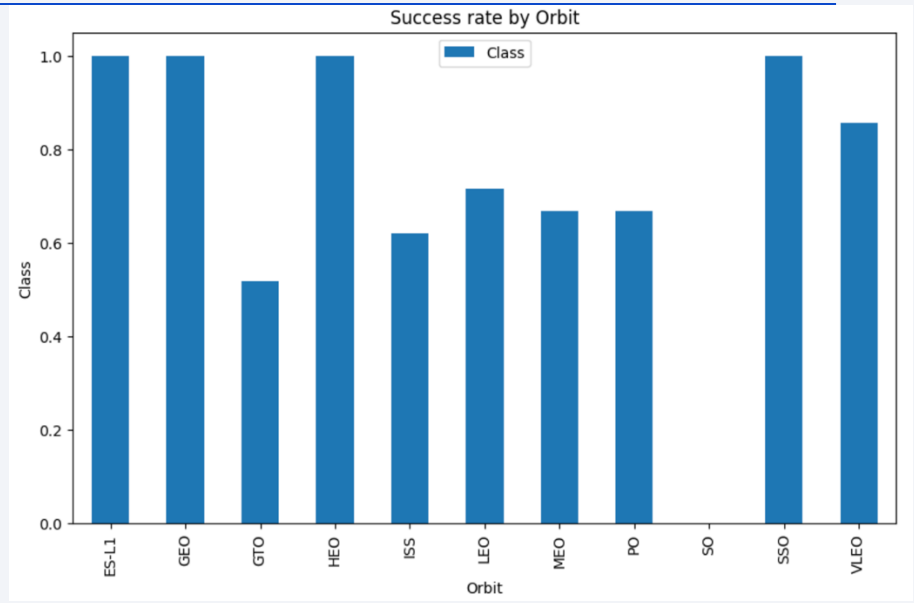
EDA with Data Visualization

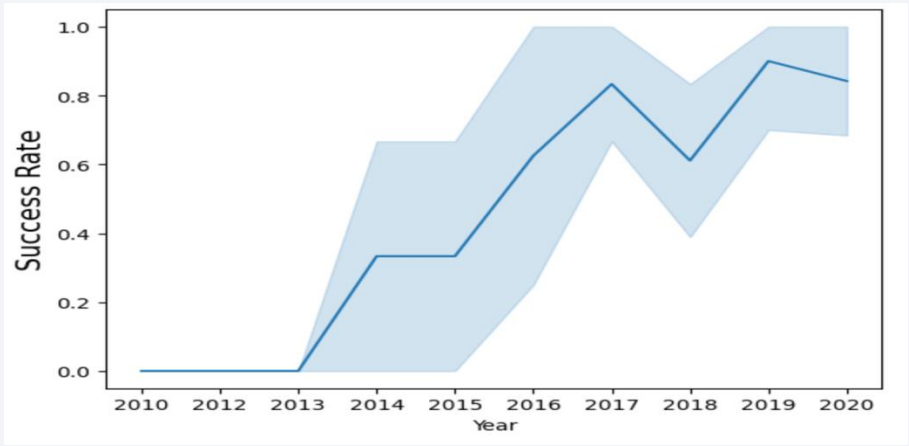












https://github.com/ChildishB/IBM-DS-Capstone/blob/main/edadataviz.ipynb

EDA with SQL

Performed SQL Queries:

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display average payload mass carried by booster version F9 v1.1
5. List the date when the first succesful landing outcome in ground pad was acheived.
6. List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
7. List the total number of successful and failure mission outcomes
8. List all the booster_versions that have carried the maximum payload mass, using a subquery with a suitable aggregate function.
9. 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.
10. 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.

https://github.com/ChildishB/IBM-DS-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Markers for all Launch Sites:

-Added markers to the coordinates of all launch sites including circles, popup labels, and text labels to show their proximity to the Equator and each coast. Additionally, the Johnson Space Center was used as a start location.

Markers for Launch Outcomes for each Launch Site:

-Added markers representing a successful launch (Green) and failed launches (Red) making use of the MarketCluster function to identify launch sites with high success rates.

Launch Site Proximity:

-Distances mapped using colored lines showing proximity to railways, highways, coastlines and closest city for all launch sites.

https://github.com/ChildishB/IBM-DS-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

Dropdown List for Launch Sites

-Added a dropdown list to view data by launch site

Pie Chart to show breakdown of successful Launches:

-Added a pie chart to show proportions of successful launches for all sites and whether a launch would succeed or fail depending on which site was selected

Slider for Payload Mass:

-Added a slider to filter data by payload mass

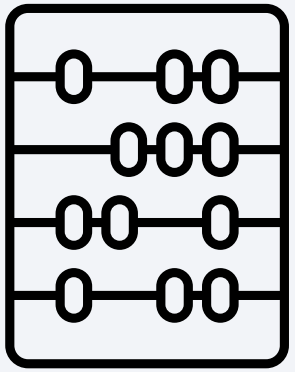
Payload Mass vs. Success Rate for different Booster Variation

-Added a scatter plot to show the correlation between Payload and Launch Success

https://github.com/ChildishB/IBM-DS-Capstone/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

1. Created a Numpy array from the created label “Class” column data.
2. Standardized the data with StandardScaler, then fit and transformed it.
3. Split the data into a training set and testing set with the train_test_split function
4. Performed a Grid Search to find the best hyperparameters for Logistic Regression, SVM, Decision Tree, and K-Nearest models.
5. Find which method best performed using the testing data



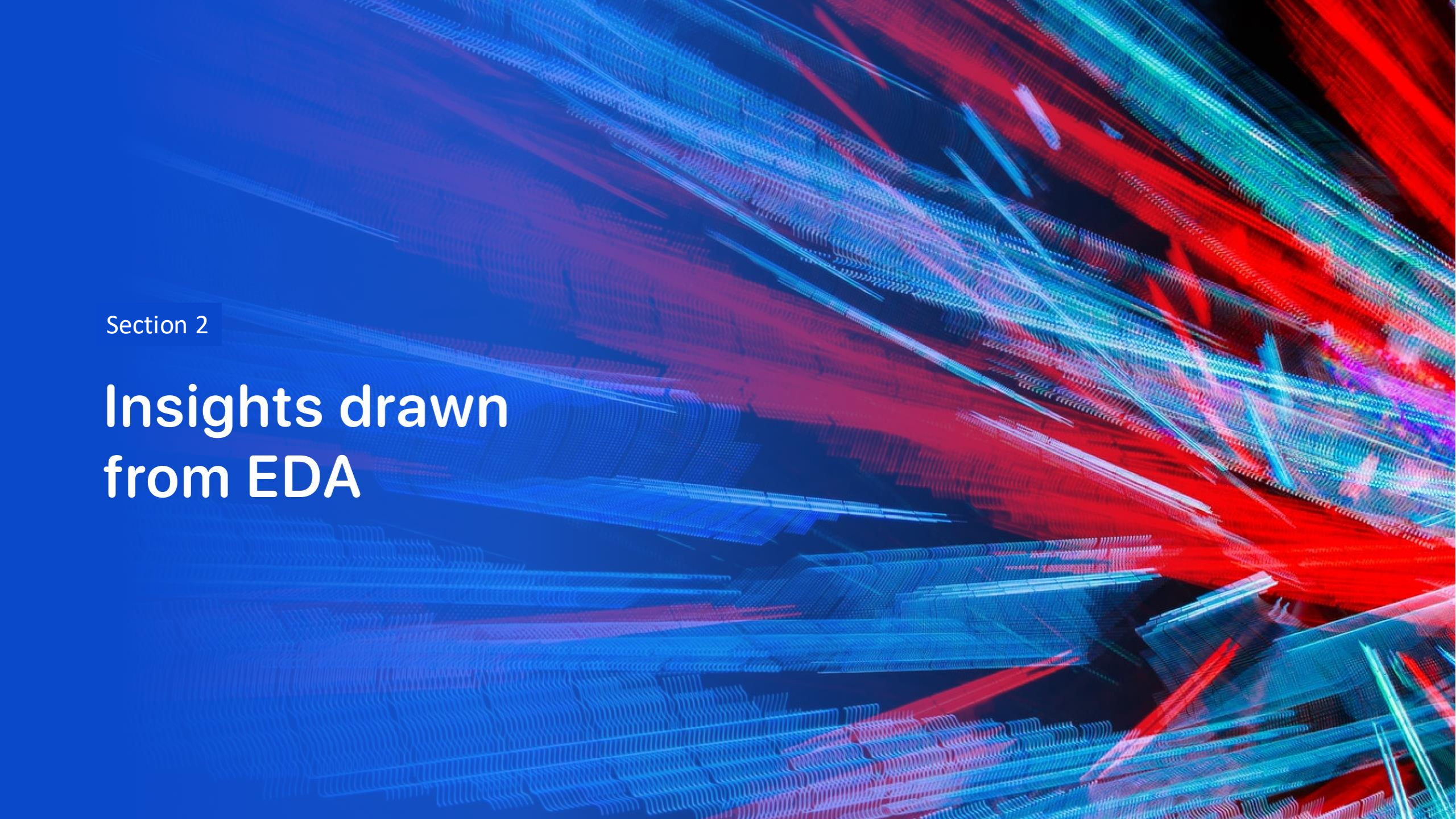
https://github.com/ChildishB/IBM-DS-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

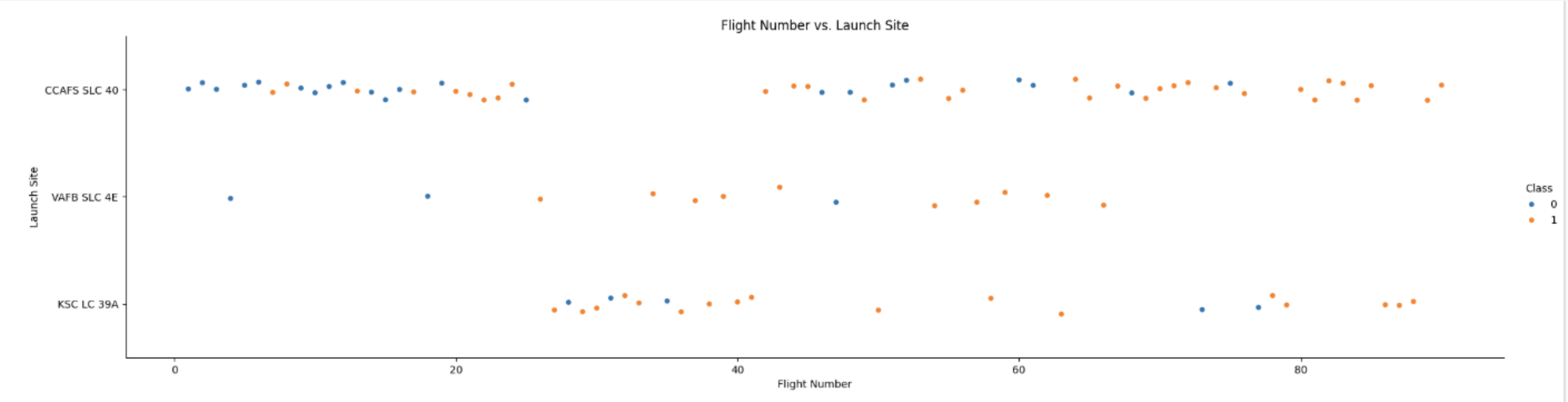
Exploratory data analysis results

Interactive analytics demo in screenshots

Predictive analysis results


Section 2
Insights drawn from EDA

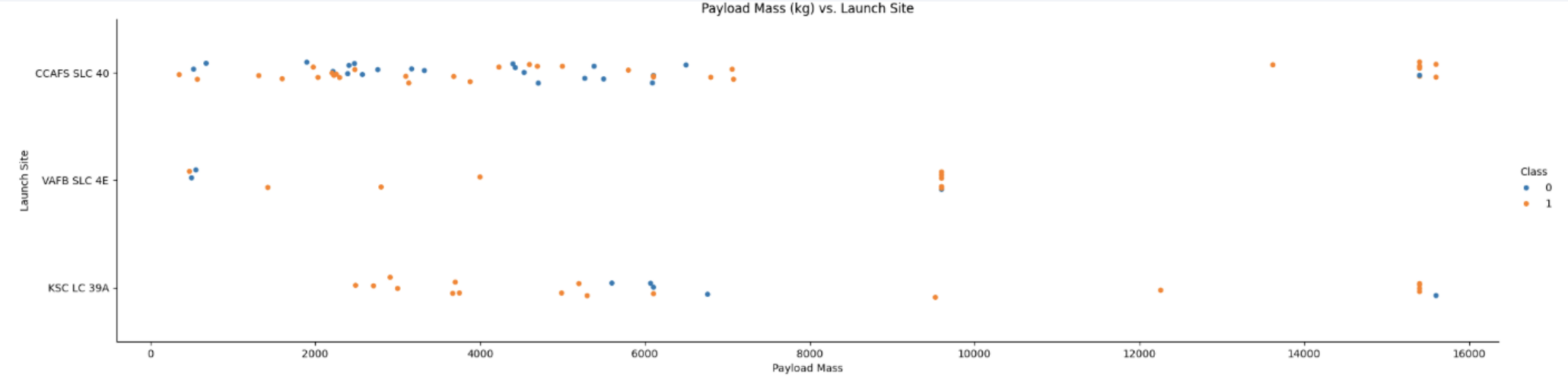
Flight Number vs. Launch Site



Analysis:

- As the number of flight number increases there are more successful flights.
- Launch Site CCAFS SLC 40 has conducted a large proportion of all flights.
- Both VAFB SLC 4E and KSC LC 39A have had more successful flights.

Payload vs. Launch Site



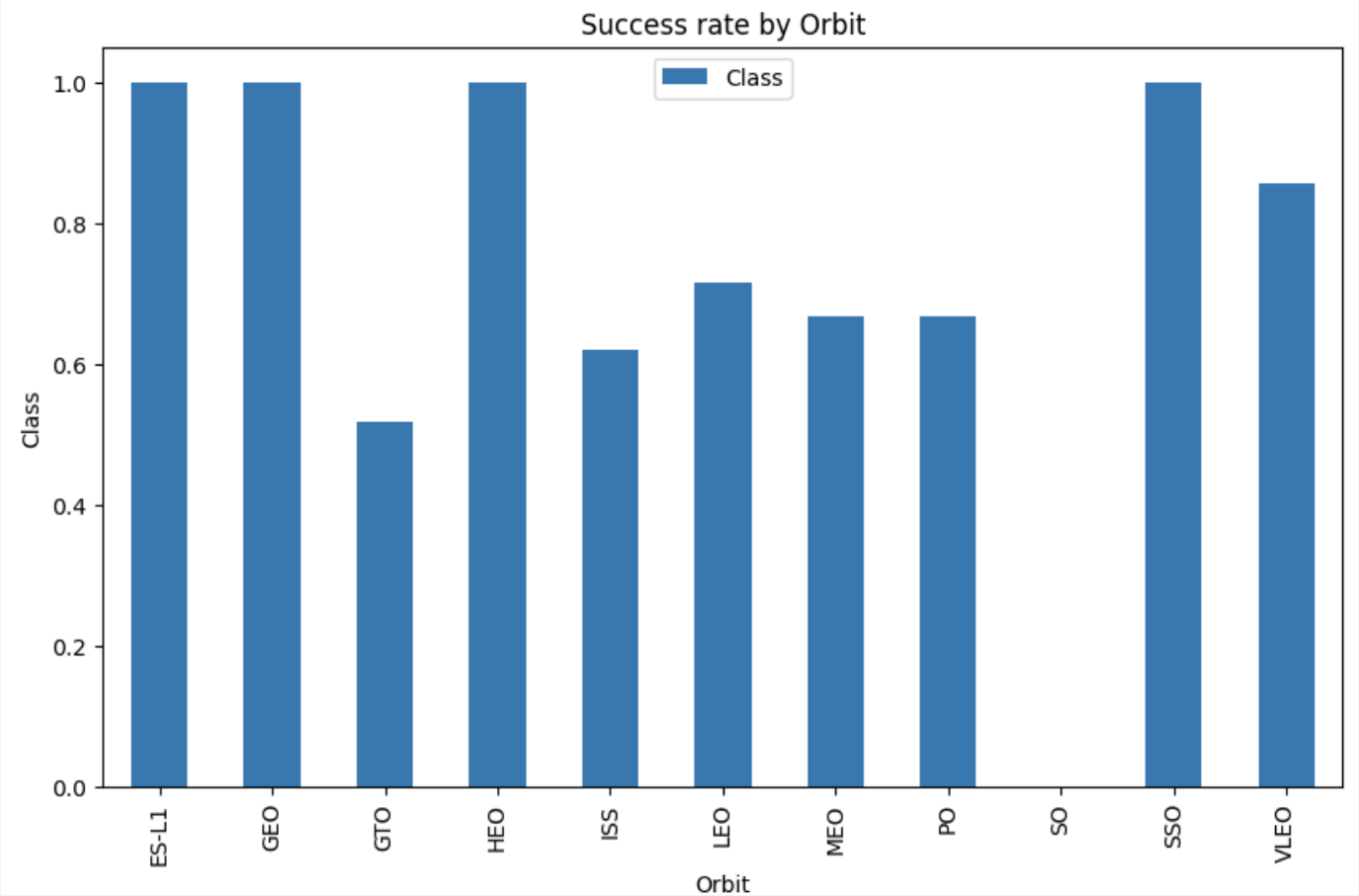
Analysis:

- VAFB SLC 40 had no payloads above 10,000 kg
- CCAFS SLC 40 and KSC LC 39A conducted most flights with payloads less than 10,000 kg
- Payloads over 7,000 kg had a higher success rate across all three sites

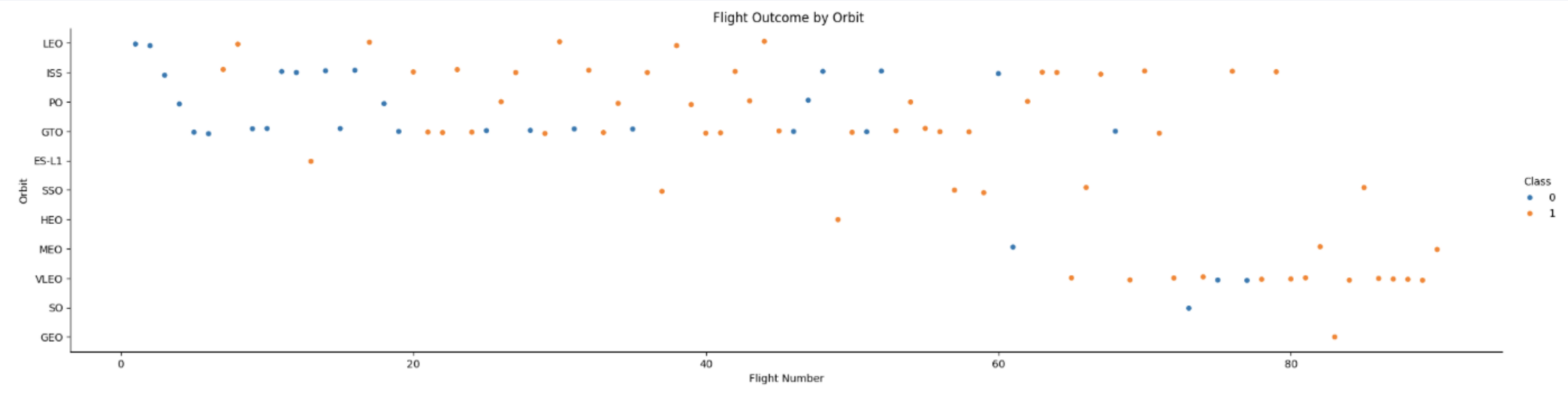
Success Rate vs. Orbit Type

Analysis:

- ES-L1, GEO, HEO, and SSO had a success rate of 100%
- Orbit SO had no successful launches
- GTO, ISS, LEO, MEO, and PO were successful between 50% and 85% of the time.



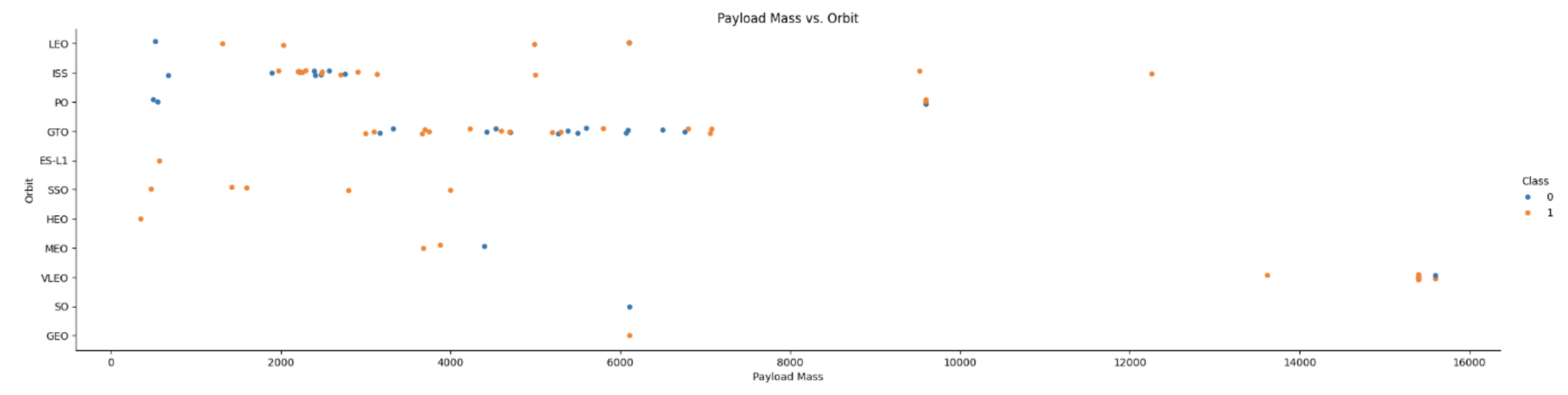
Flight Number vs. Orbit Type



Analysis:

- LEO success seems to be related to the number of the flights, conversely, GTO seems to have no relationship between success and flight number.

Payload vs. Orbit Type



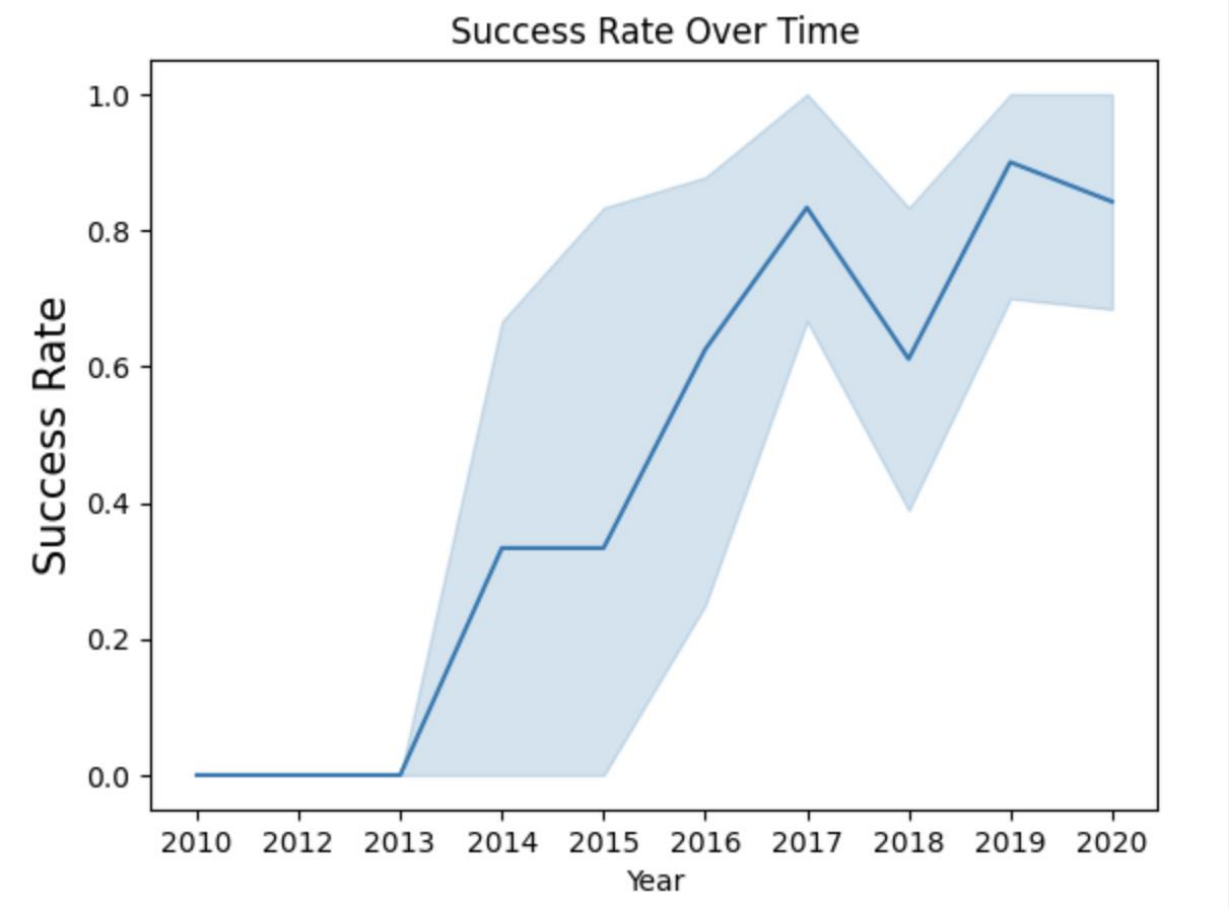
Analysis:

- PO, LEO, and ISS have greater success with a higher payload. Moreover, it is hard to distinguish this for GTO because of the presence of both outcomes

Launch Success Yearly Trend

Analysis:

- Since 2013 to 202 the rate of success has increased.



All Launch Site Names

- Performed a SQL query to obtain the unique launch sites.
- Site names:
  - CCAFS LC-40
  - CCAFS SLC-40
  - KSC LC-39A
  - VAFB SLC-4E

Task 1

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

```
%sql select distinct launch_site from spacextable
```

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

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

Launch Site Names Begin with 'CCA'

- Displaying 5 launch sites beginning with the string 'CCA'

Task 2

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
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX
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
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)

Total Payload Mass

Task 3

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

```
%sql select sum(payload_mass__kg_) from spacextable where customer = 'NASA (CRS)';
```

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

sum(payload_mass__kg_)
45596

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

Average Payload Mass by F9 v1.1

Task 4

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

```
%sql select avg(payload_mass__kg_) from spacextable where booster_version = 'F9 v1.1';
```

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

avg(payload_mass__kg_)
2928.4

- Displaying average payload mass carried by booster version F9 v1.1.

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
%sql select distinct landing_outcome from spacextable ;

%sql select min(date) from spacextable where landing_outcome = 'Success (ground pad)';
```

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

min(date)
2015-12-22

- Displaying the date of the first successful ground pad landing.

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

```
%sql select booster_version from spacextable where landing_outcome = 'Success (drone shi
```

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

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

- Boosters which have success in drone ship and have payload mass between 4,000 and 6,000 kg.

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql select mission_outcome, count(*) from spacextable group by mission_outcome;
```

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

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

- Successful Outcomes: 99
- Failed Outcomes: 1

Boosters Carried Maximum Payload

- Each booster version that carried the maximum payload

Task 8

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

```
%sql select booster_version from spacextable where payload_mass__kg_ = (select max(payld
```

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

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

- Displaying failed drone ship landings, their booster version, and launch site by month in the year 2015
  - January
  - April

Task 9

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 substr(date,6,2), landing_outcome, booster_version, launch_site from spacext
```

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

substr(date,6,2)	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

Task 10

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

- Ranking the landing outcomes in descending order within the date range

Task 10

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(*) from spacextable where date between '2010-06-04'
```

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

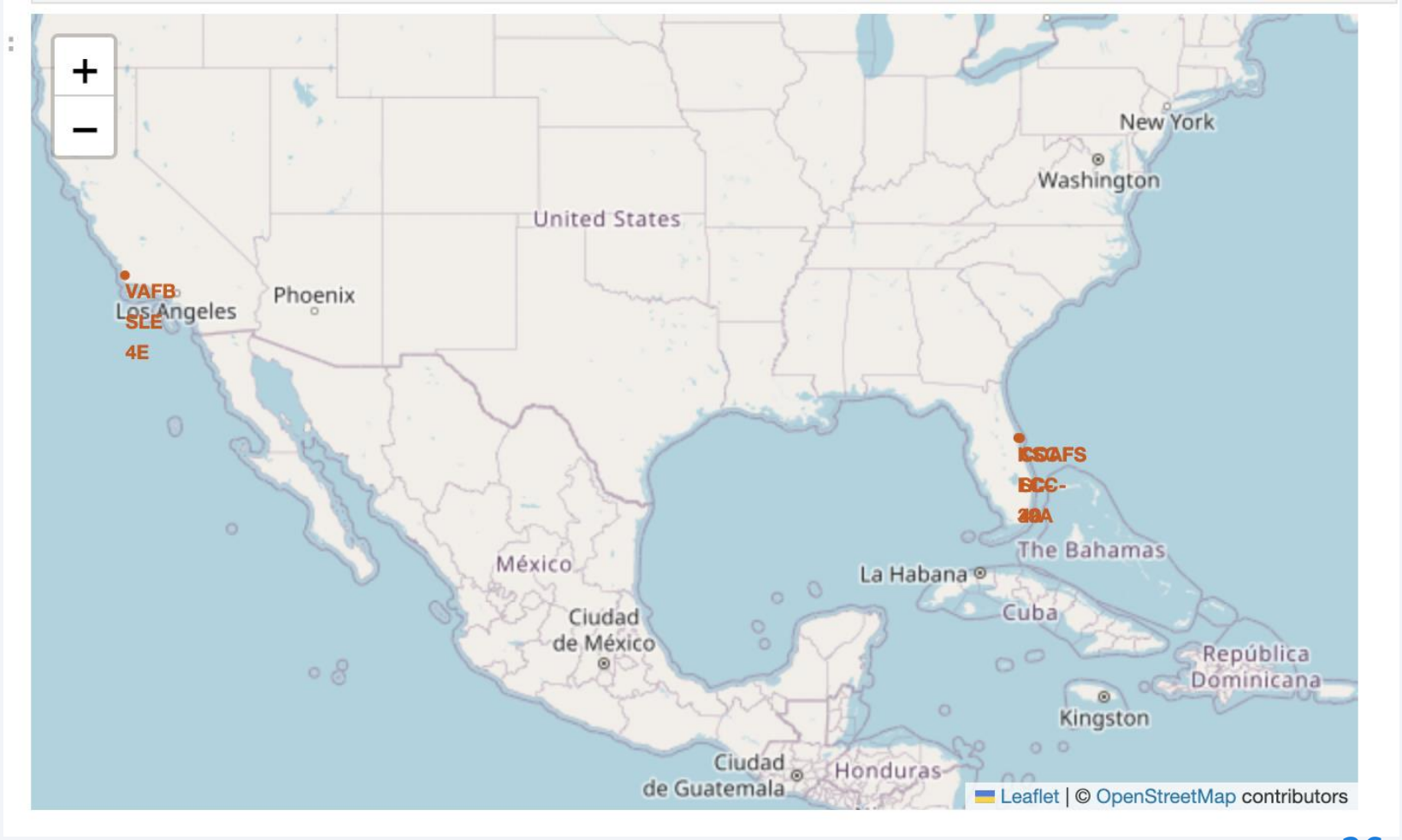
Landing_Outcome	count(*)
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

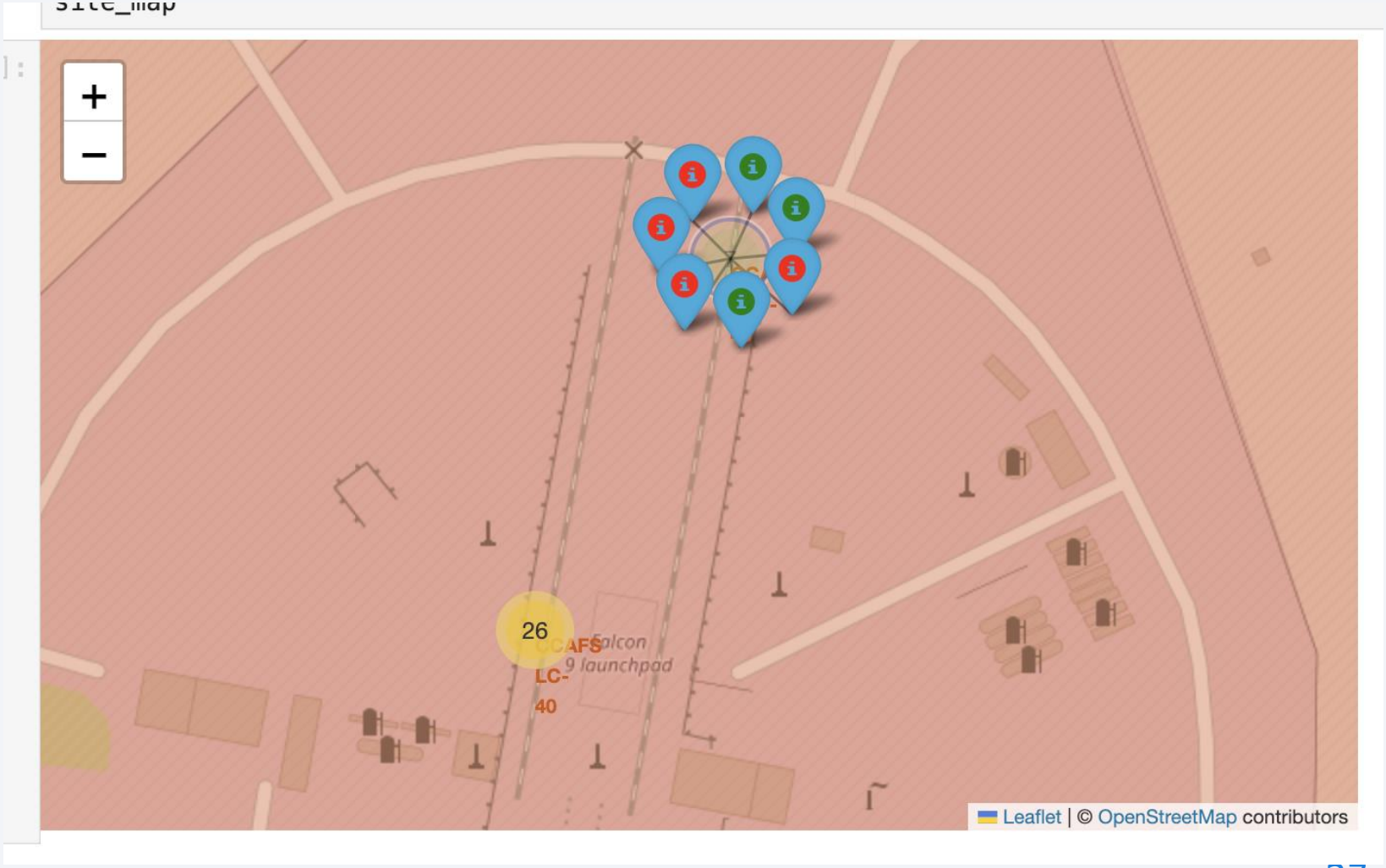
Markers for All Launch Sites

- All launch sites are in close proximity to the Equator line and US coastlines.



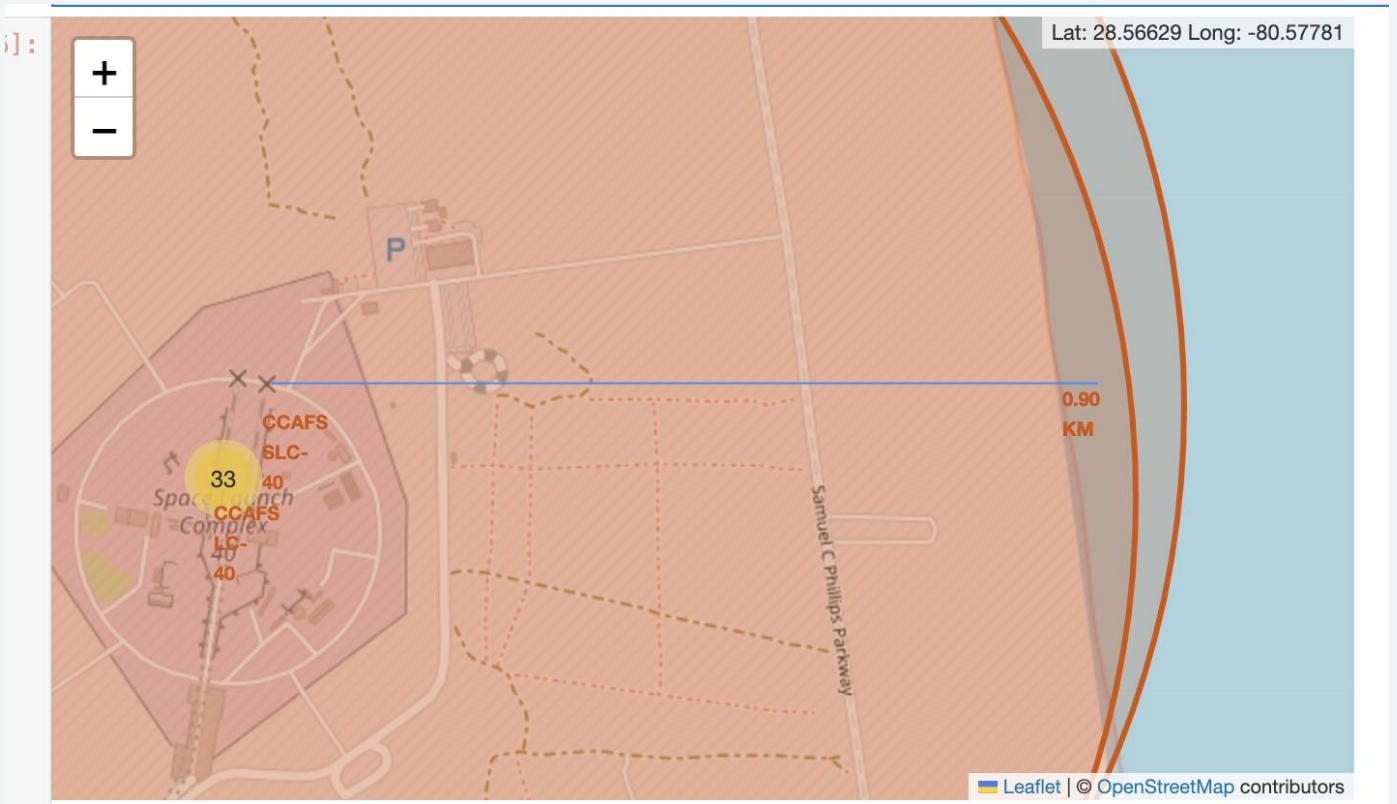
Markers for Launch Outcomes for each Site

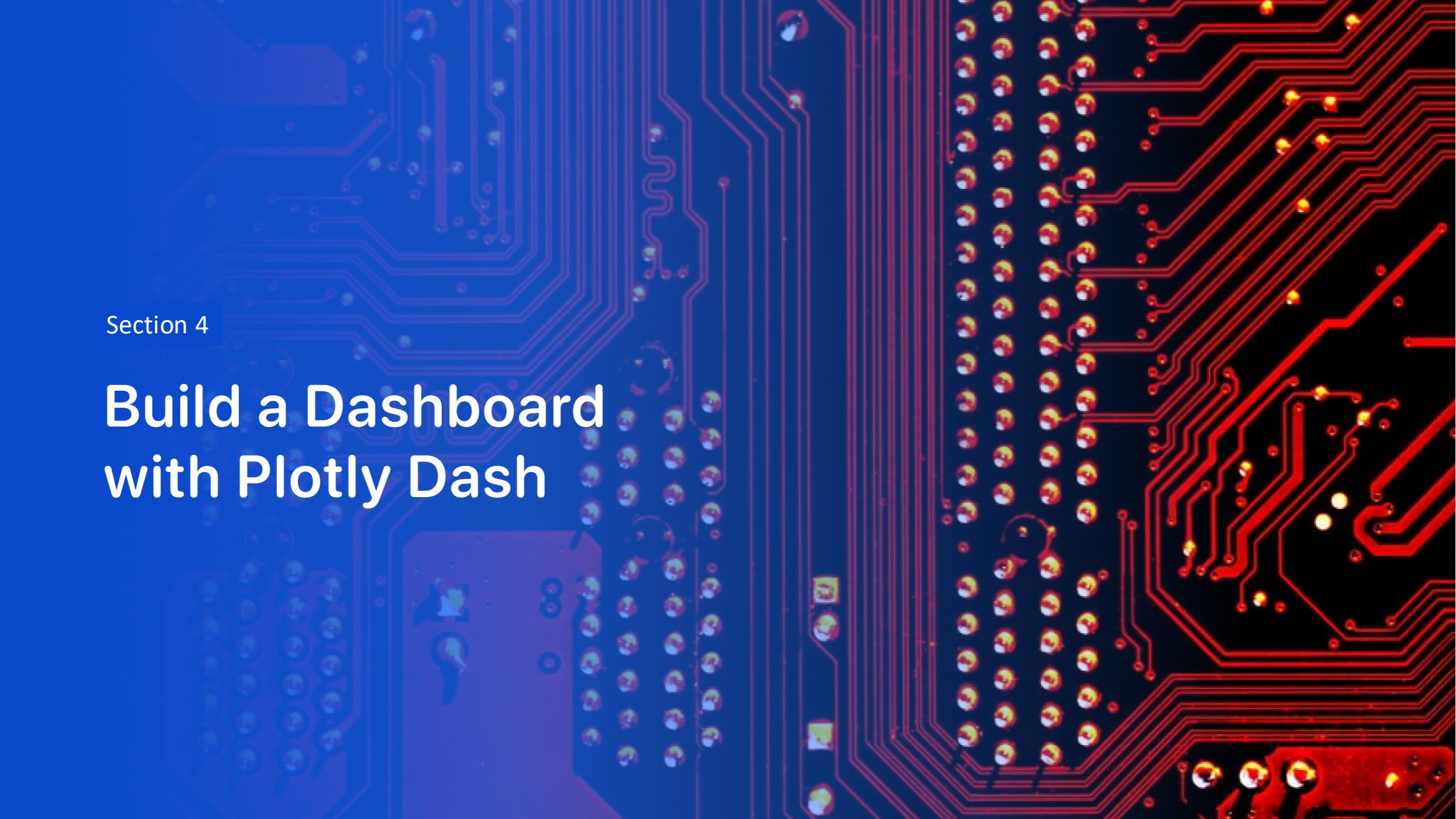
- The green markers represent successful launch outcomes while the red markers represent failed launch outcomes.
- KSC LC-39A has the highest success rate of of all sites



Launch Site Proximity

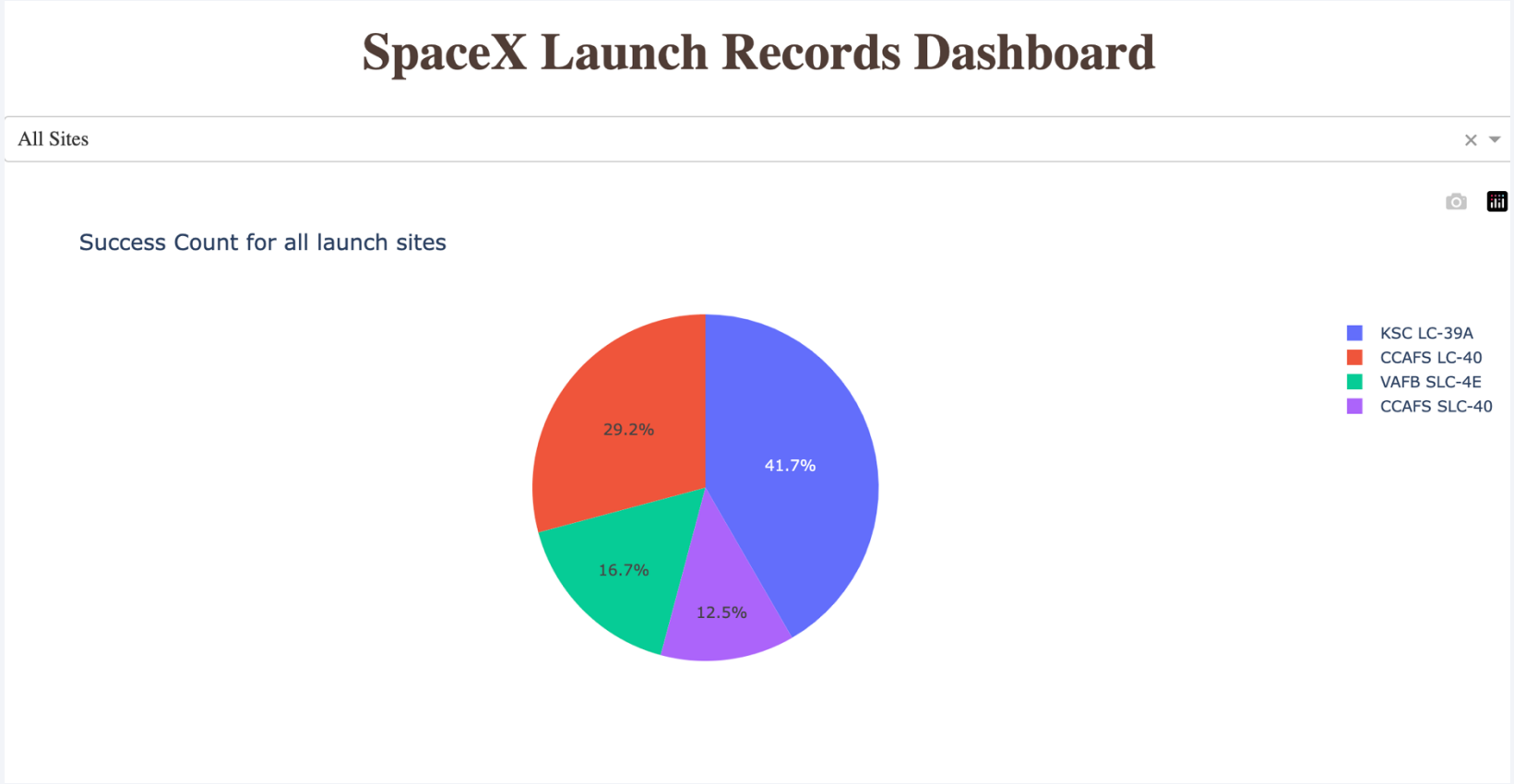
- Using Mouse Position, it was found that the nearest coastline was 0.90 kilometers away from CCAPS SLC.




Section 4
Build a Dashboard with Plotly Dash

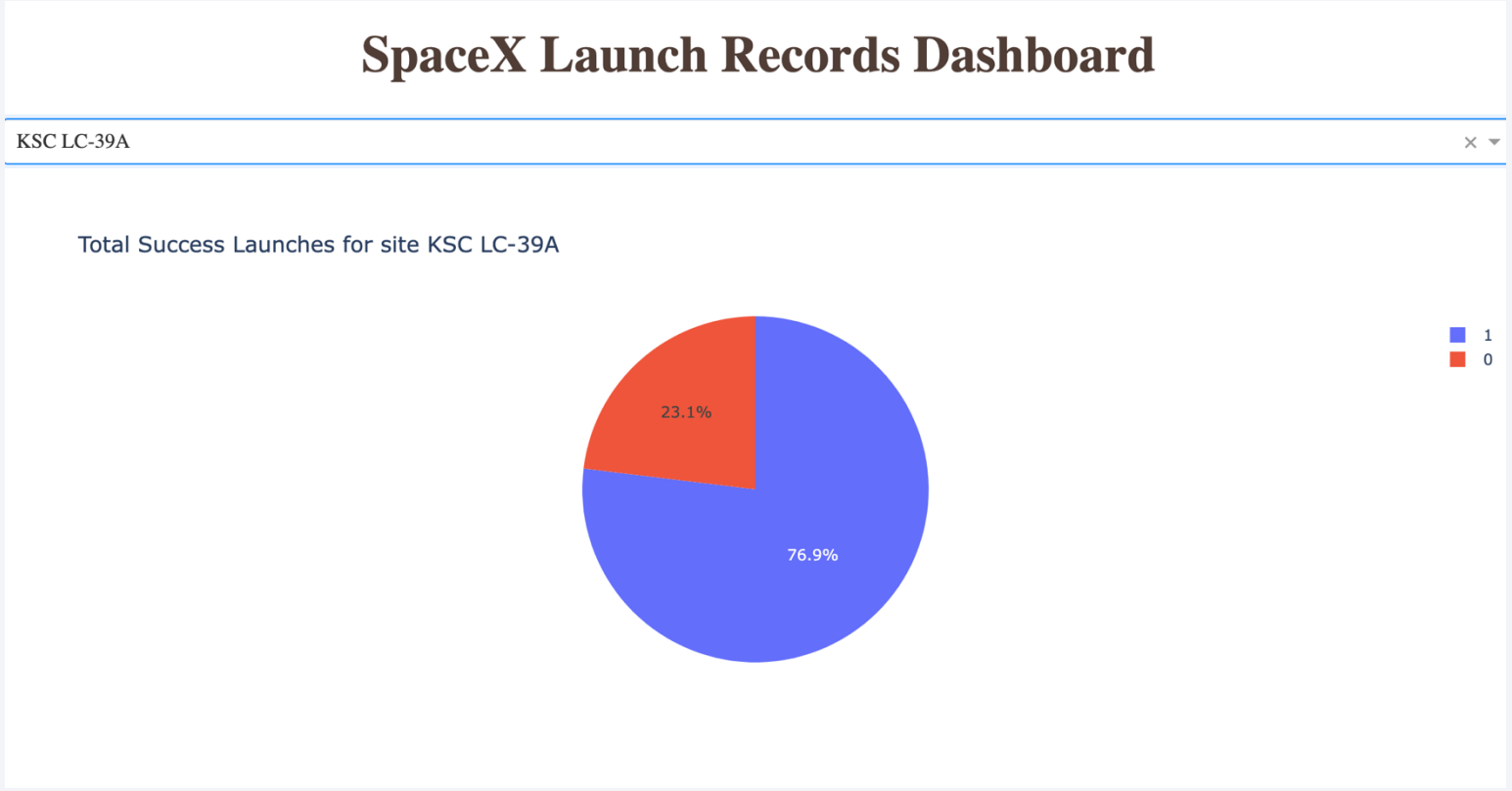
Successful Launches by Launch Site

- Displayed are the portions of successful outcomes of all launch sites.
- KSC LC-39A has the highest amount of successful launches



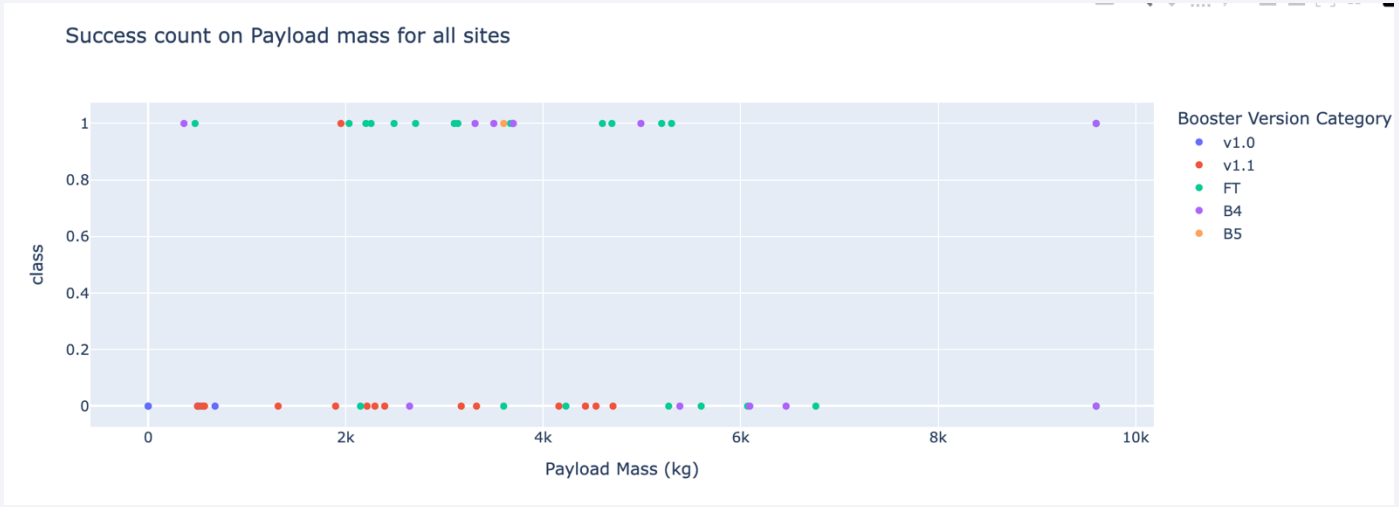
Highest Rate of Success by Launch Site

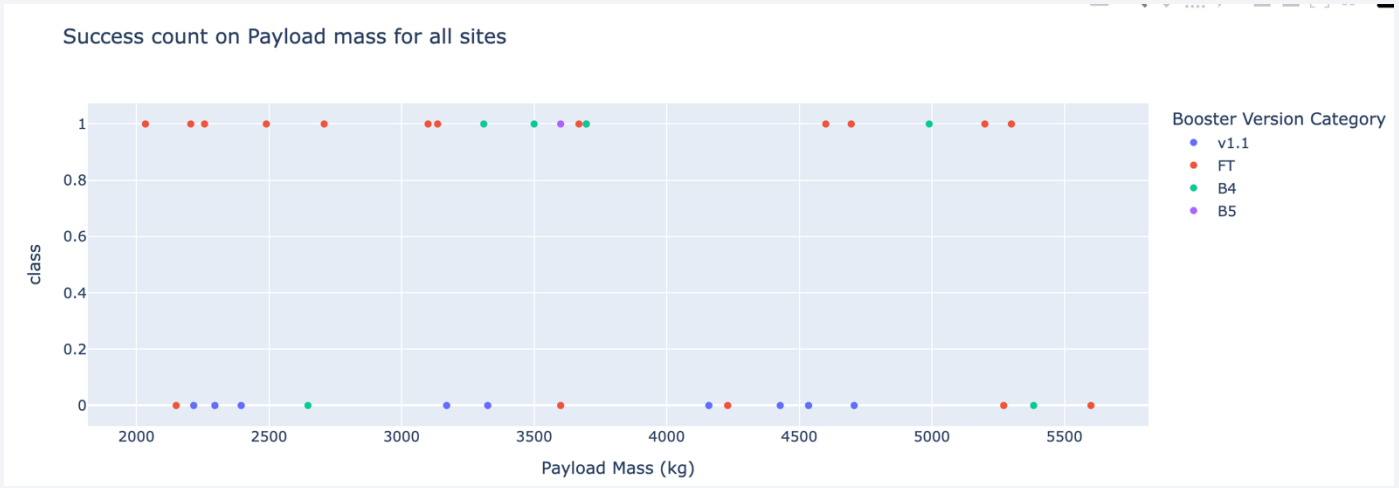
- KSC LC-39A has a 76.9% success rate with 10 successful landing and 3 failed landings.



Payload Mass vs. Launch Outcomes for all Sites

- From the first graphic we observe that FT has the highest success rate of all booster types.
- From the second graphic we observe that payloads between 2000 and 5500 have the highest rate of success.





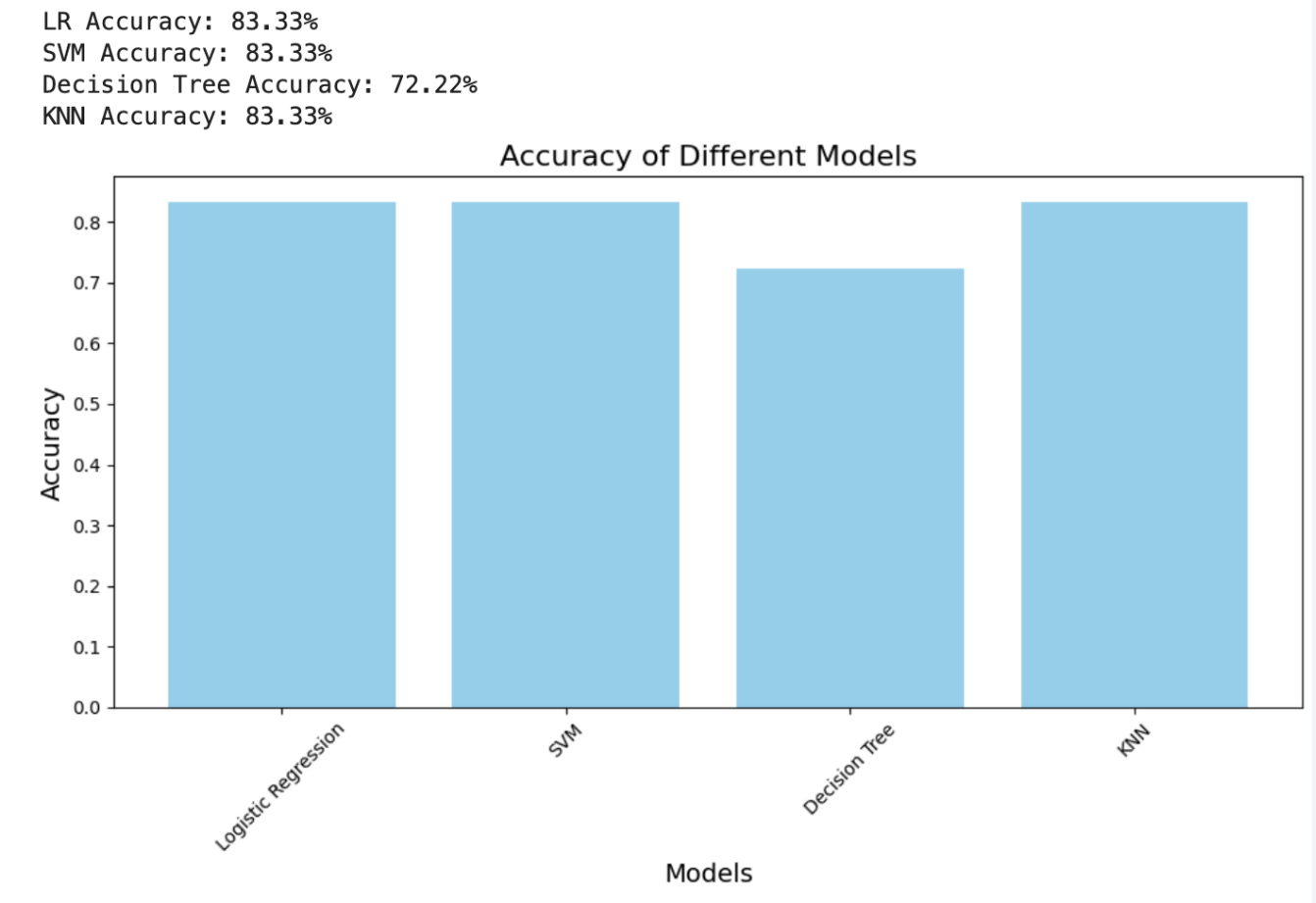
Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Although Decision Tree had the highest training accuracy the testing accuracy is lower. It's important to mention that half the instances resulted in NaN due to an error.

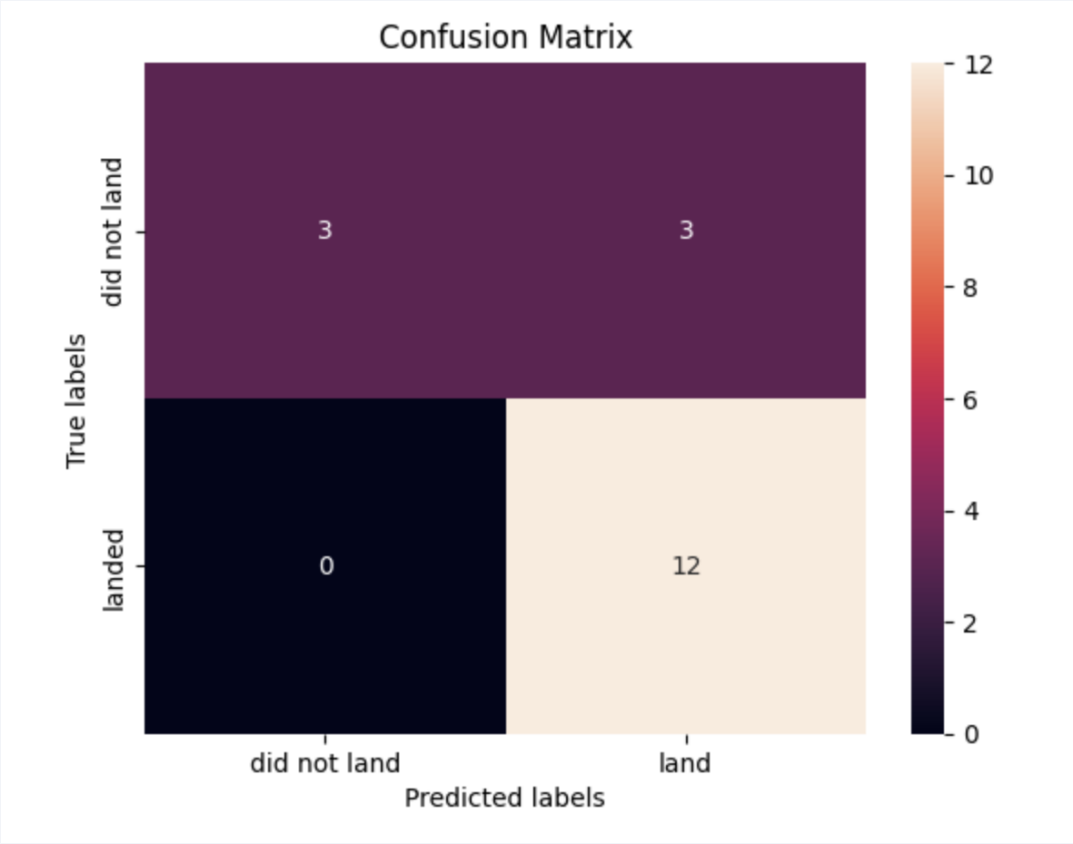
```
LR Accuracy: 83.33%
SVM Accuracy: 83.33%
Decision Tree Accuracy: 72.22%
KNN Accuracy: 83.33%
```



Confusion Matrix

Analysis:

- Displayed is the confusion matrix for having 3 false positives and 12 true positives.



Conclusions

- KNN is the best model for the data set although Decision Tree may end up being better if all instances are accounted for
- The likelihood of successful launches has trended upward over time
- Launches with lower payloads have a better success rate than launches with higher payloads.
- KSC LC-39A has the highest success rate out of all the other launch sites.
- GEO, ES-L1, HEO, and SSO orbits all have the highest success ratio for launches

Thank you!