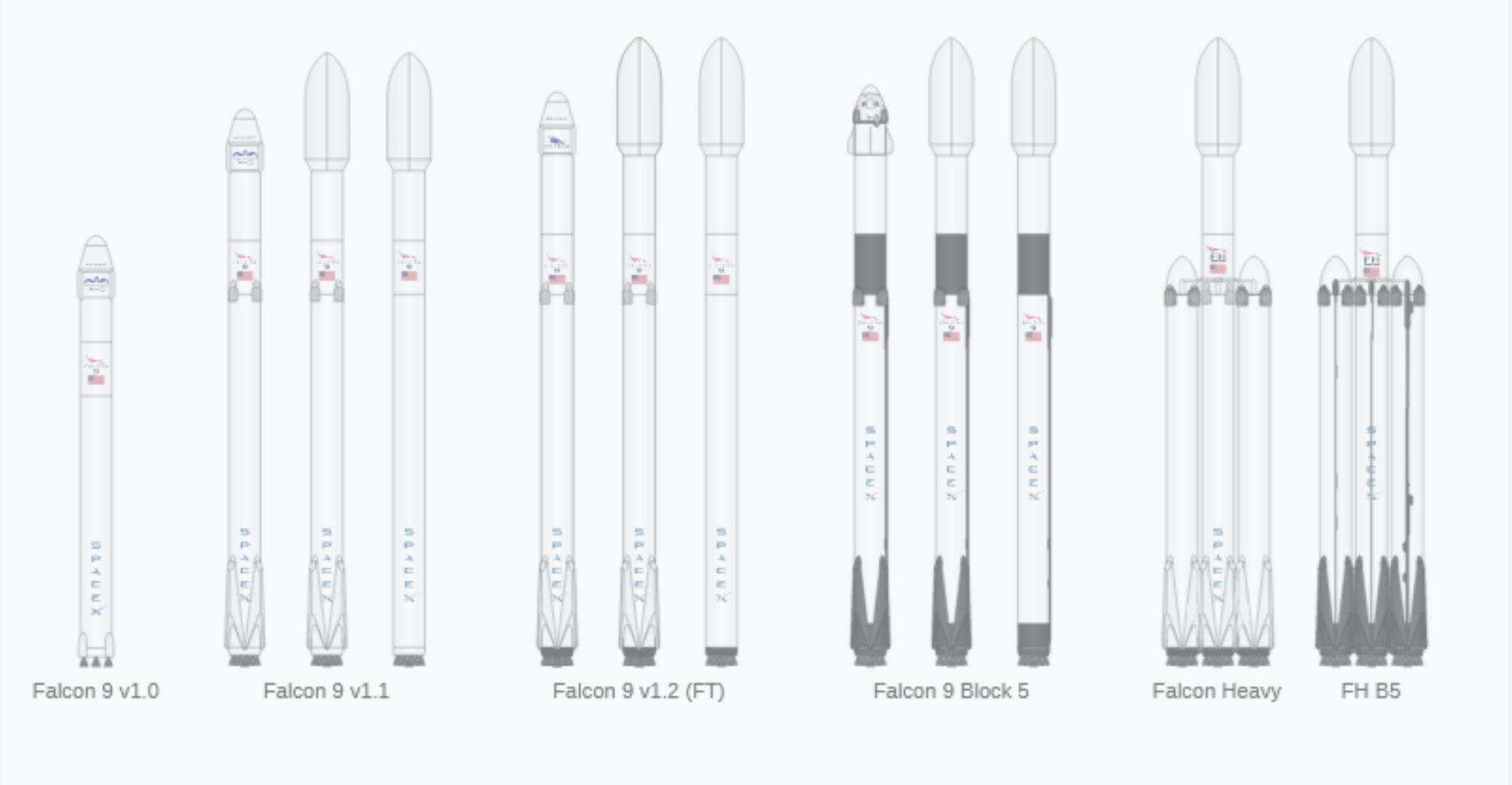
Winning the Space Race with Data Analysis



Jason Cheers
11/18/2022

IBM Developer
SKILLS NETWORK

Outline

- Executive Summary
- Introduction
- Methodology
- Insights Drawn From EDA
- Launch Sites Proximities Analysis
- Build a Dashboard with Plotly Dash
- Predictive Analysis (Classification)

Executive Summary

The purpose of this project tis to create a model that can provide accurate predictions on successful stage 1 landings.

The model was built using data collected through API and webscraping. This data was standardized and reviewed to gain insights to help construct the model. Class plotting and dashboarding was utilized in the analysis of data.

Multiple predictive models were built and compared to gain the best training accuracy to prepare for testing. The final product is one that will provide a prediction of successful landings within a reasonable range of accuracy.

Introduction

The data was collected using the SpaceX API and by using web scraping techniques on the Wikipedia site. Both sources have benefits and are analyzed for best use.

The data is then reviewed using exploratory methods to determine if any insights can be made on the most impactful categories to be used in the prediction of landing outcomes.

Multiple modeling techniques are then used to determine which would provide the best results to be used in future analysis and prediction.

GitHub Project URL:

Data Science Capstone

Section 1

Methodology

Methodology

Data collection methodology:

- Data was collected using the SpaceX API and by performing webscraping launch records

Perform data wrangling

- Orbit and landing statistics were reviewed to generate a success class column

Perform exploratory data analysis (EDA) using visualization and SQL

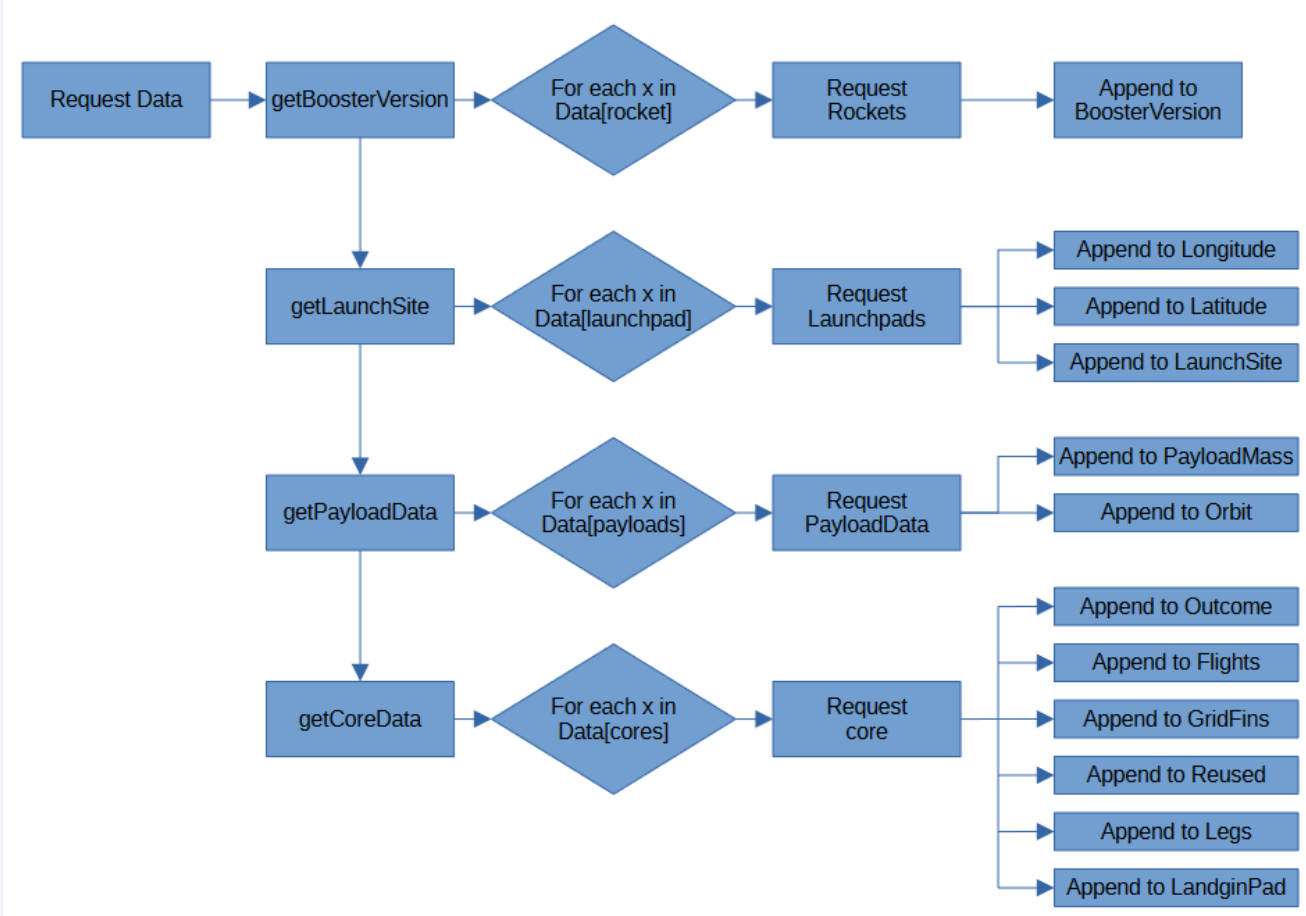
Perform interactive visual analytics using Folium and Plotly Dash

Perform predictive analysis using classification models

- How to build, tune, evaluate classification models

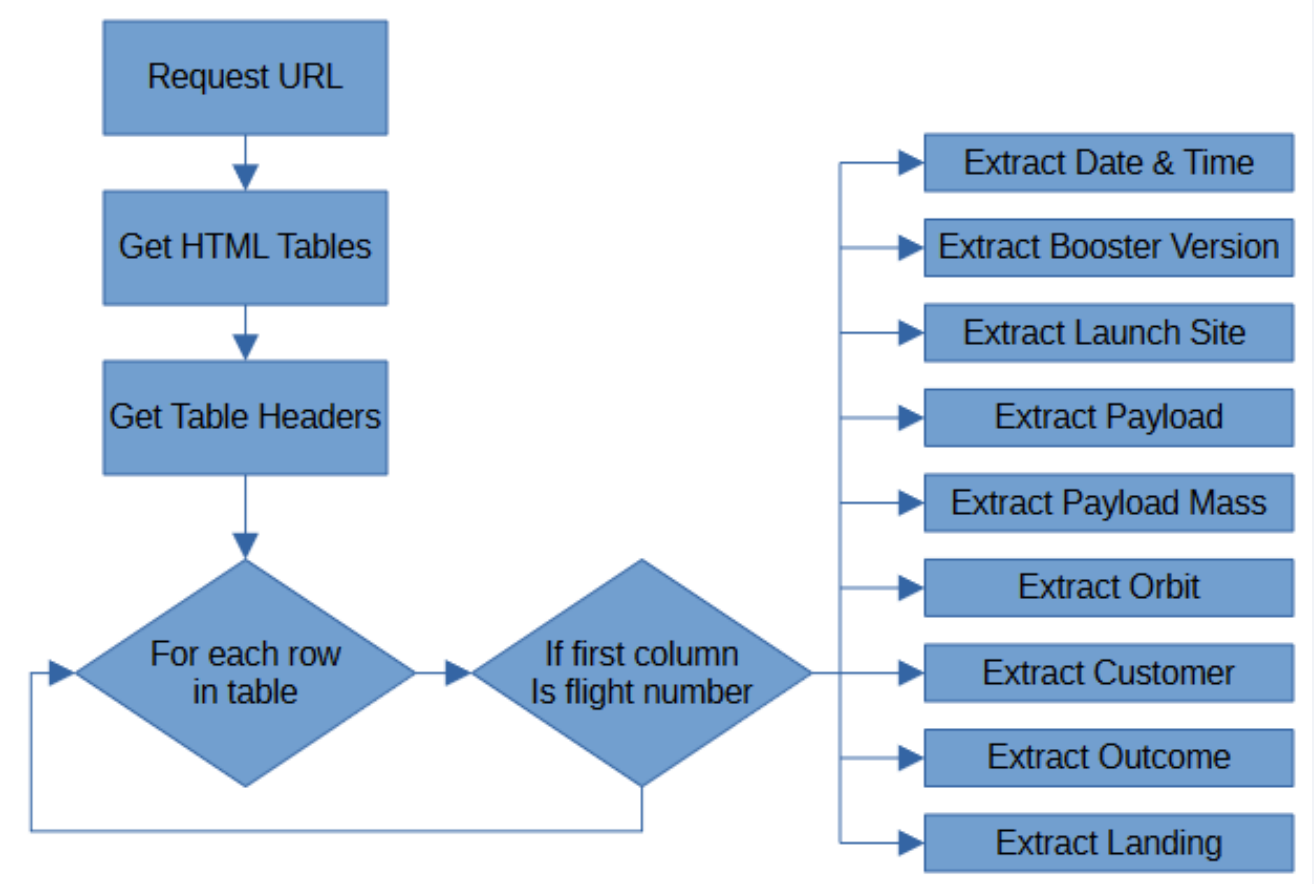
Data Collection – SpaceX API

- The summary data collection was requested from the API
- The data was then analyzed by row
  - If flight number was present, the required data was then requested through the API
    - Booster Version, Launch Details, Payload, and Core Flight Information
- GitHub URL:
  - [Data Collection API]



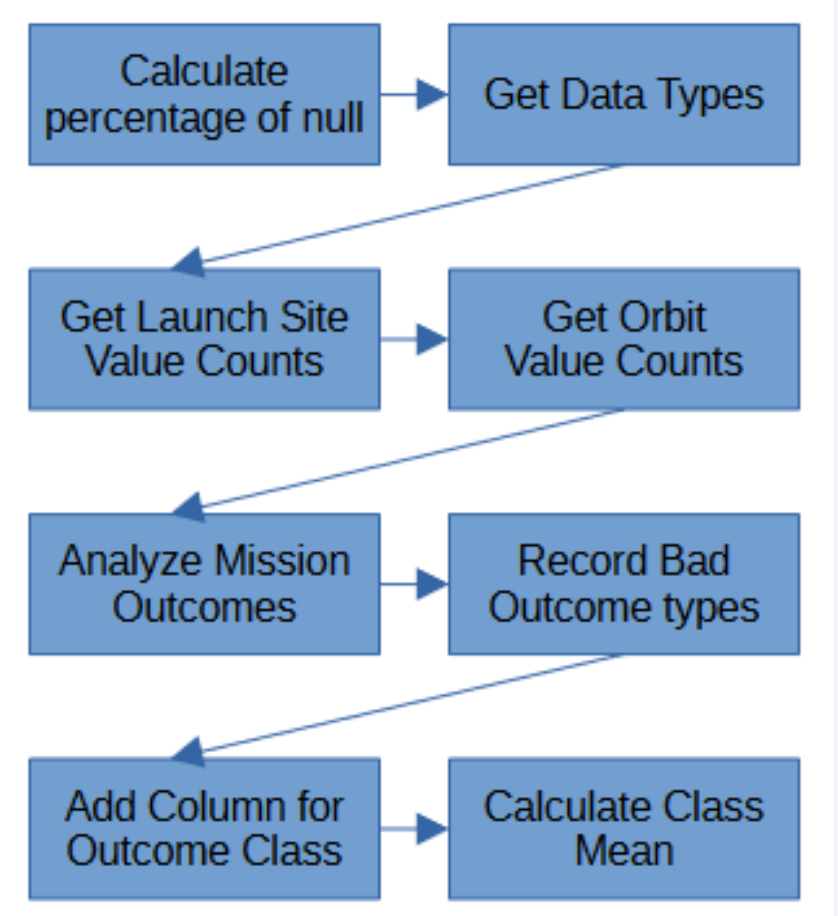
Data Collection - Scraping

- Using the requests and BeautifulSoup libraries the web site was parsed.
  - The table of data was extracted, and the headers were stored. Each row of the table was then analyzed
  - If flight number was valid, the rest of the columns were processed and stored
- GitHub URL
  - Web Scraping



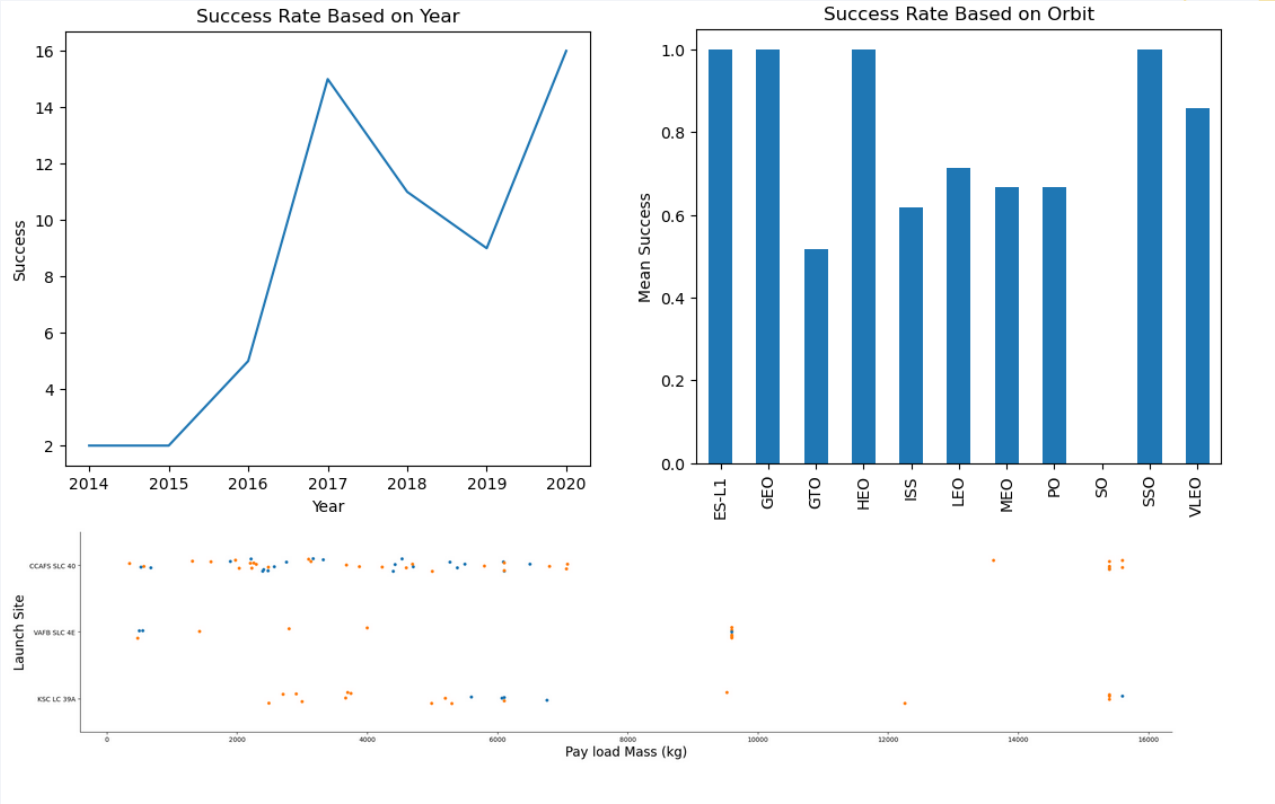
Data Wrangling

- The data verification process first involved checking for null values and verify data types.
- The counts of each launch site and orbit types were recorded.
- Mission outcomes were next analyzed.
  - The quantity of bad or failed outcomes was recorded.
  - A column was added to contain the outcome class for success/fail.
  - The mean of the outcome class was calculated.
- GitHub URL
  - Data Wrangling



EDA with Data Visualization

- The charts show correlation between different categories
  - Success rate of flight numbers by payload size
  - Success rate of flight numbers by launch site
  - Success rate of payload size by launch site
  - Mean success rate by orbit type
  - Success rate of flight number by orbit type
  - Success rate of payload mass by orbit type
  - Launch success trend by year
- GitHub URL
  - EDA with Visualization

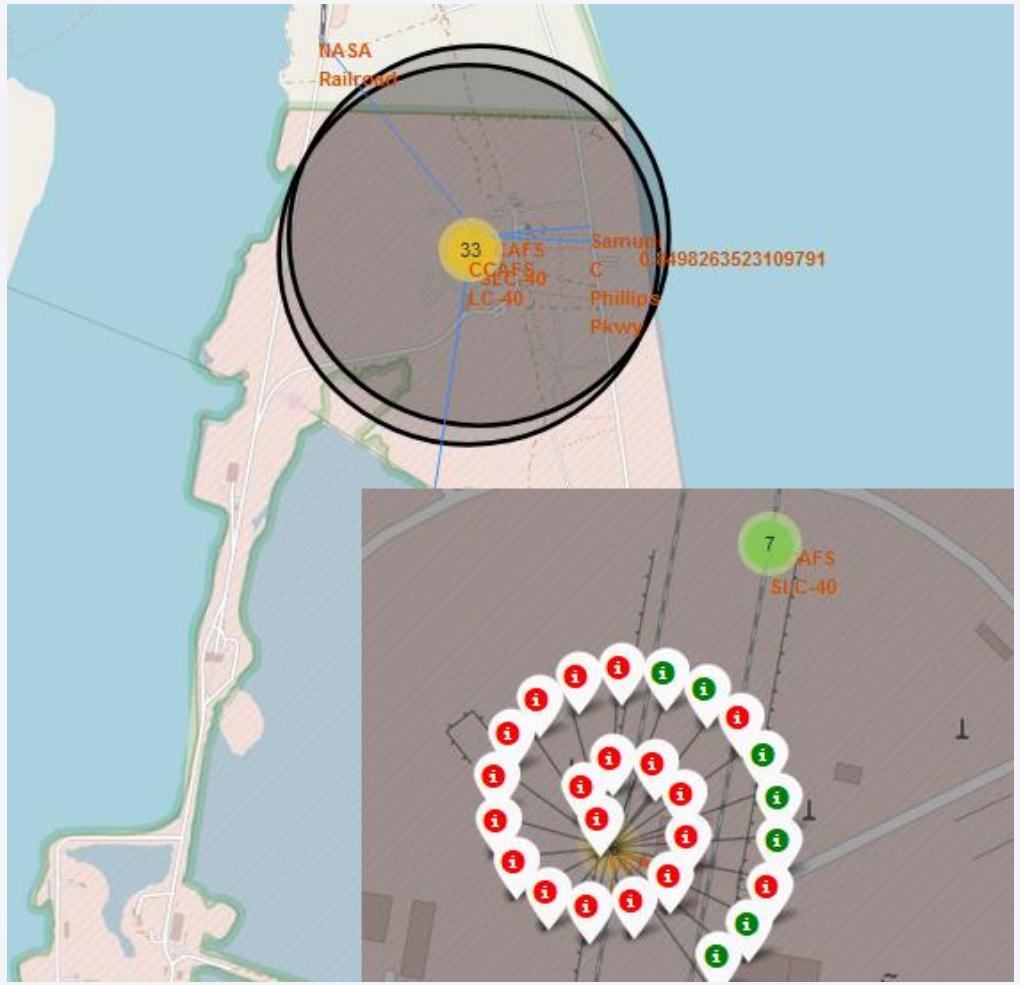


EDA with SQL

- Unique launch sites and Launch sites associated with KSC
- Total payload mass of launches performed by NASA (CRS)
- Average payload mass of booster versions F9 v1.1
- First successful drone ship landing
- Booster versions with successful ground pad landings between 4000 and 6000 kg payload mass
- Total number of landing success and failures
- Successful ground pad landings for a year and overall success for date range
- GitHub URL
    - Exploratory Data Analysis with SQL

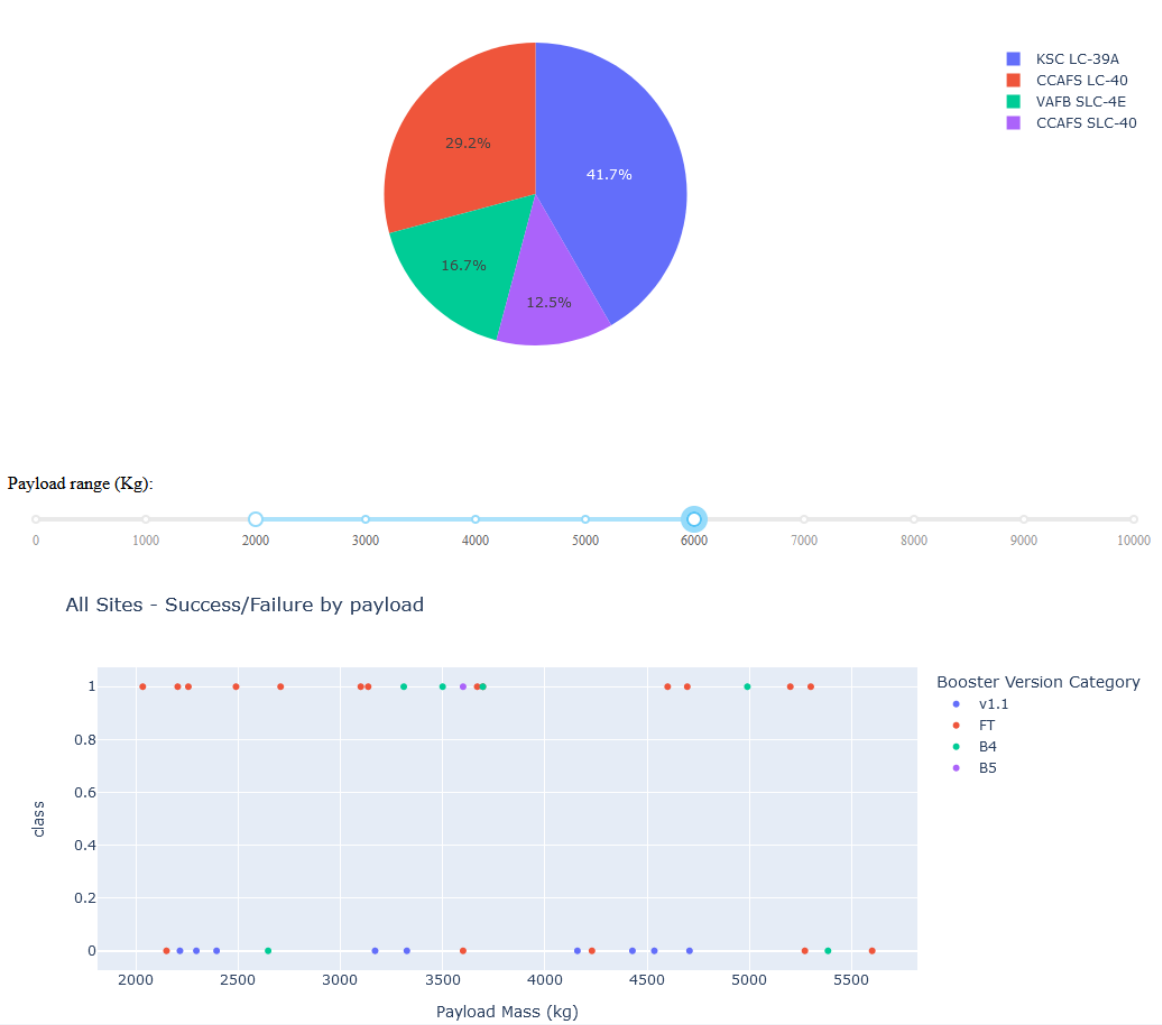
Build an Interactive Map with Folium

- Added NASA location and launch site markers
- Added mission launch success and failure markers
- Added polyline from coast to closest launch site
- Added polylines from closest launch site to nearest railroad, city, and highway.
- GitHub URL
  - Folium Analysis



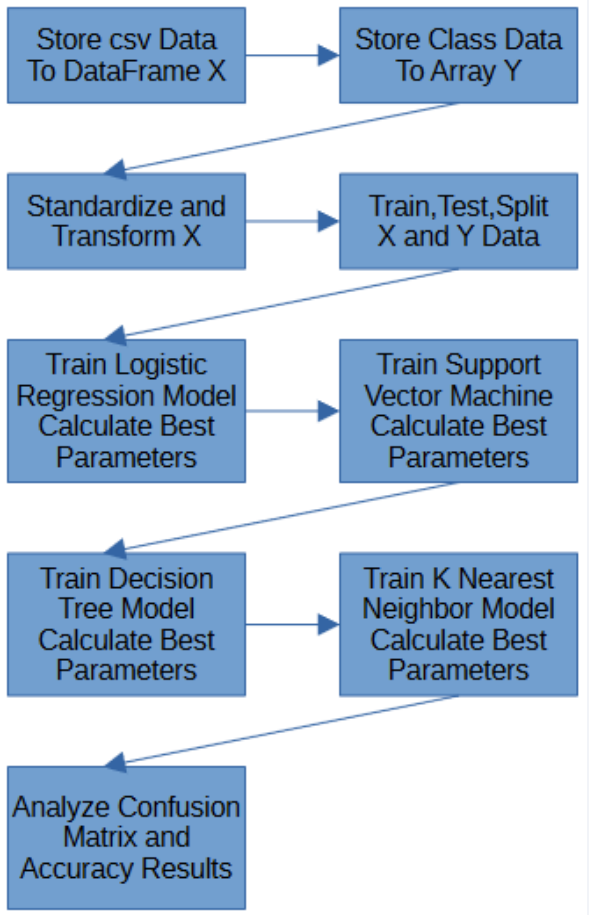
Build a Dashboard with Plotly Dash

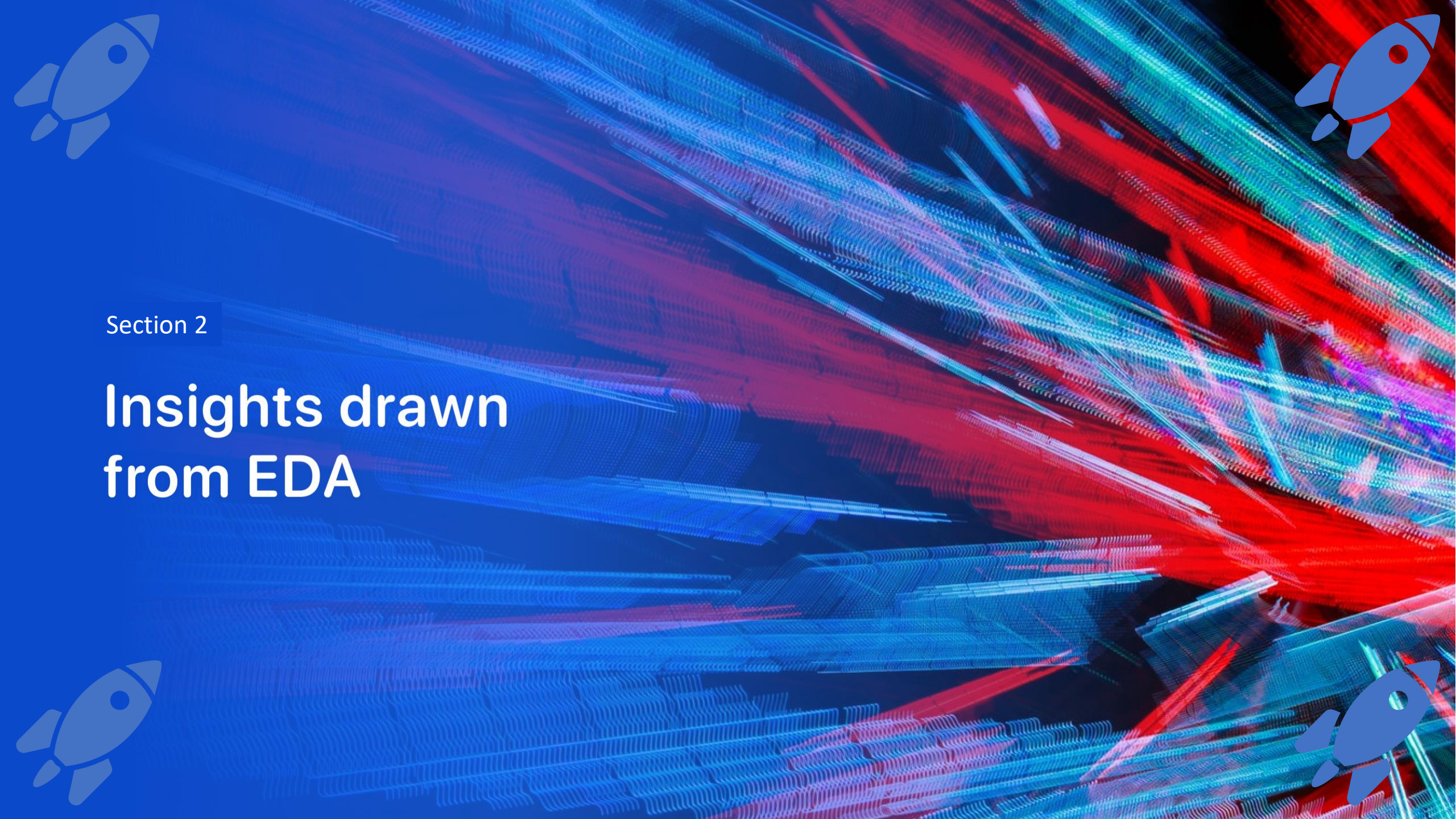
- Pie graphs were added to show overall success rates of each landing site, or success/fail rate of each individual launch site.
- A scatter plot was also added to show success rates for payload mass, the range of payload mass is also adjustable.
  - The scatter plot can either show all sites or a particular landing site.
- GitHub URL
  - Python Dashboard



Predictive Analysis (Classification)

- The overall data structure was stored in a DataFrame for analysis
  - This data was then standardized, transformed and train/test/split
- GridsearchCV was used to determine the optimal parameters for the models.
  - Models used were Logistic Regression, Support Vector Machine, Decision Tree, and K-Nearest Neighbor.
- All techniques were then compared to determine the best fit model.
- GitHub URL
  - Predictive Analysis

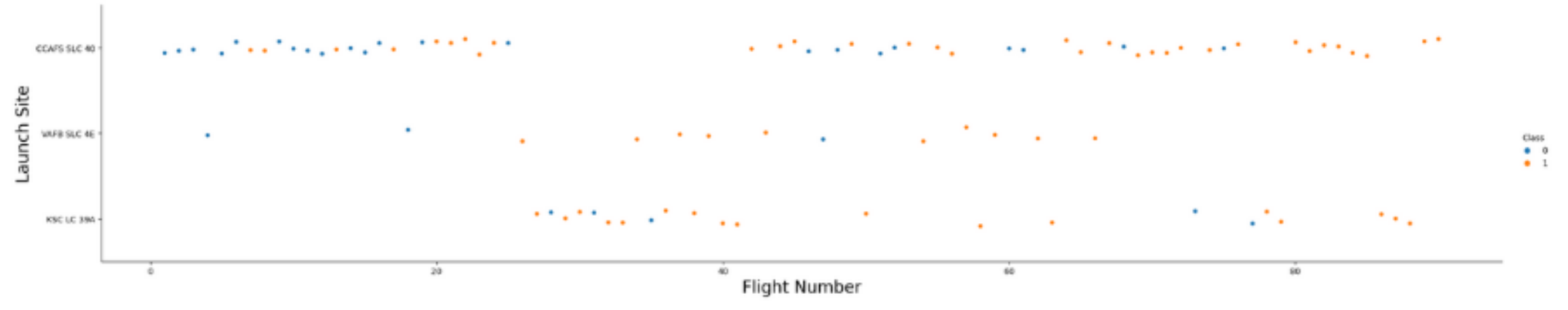




Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

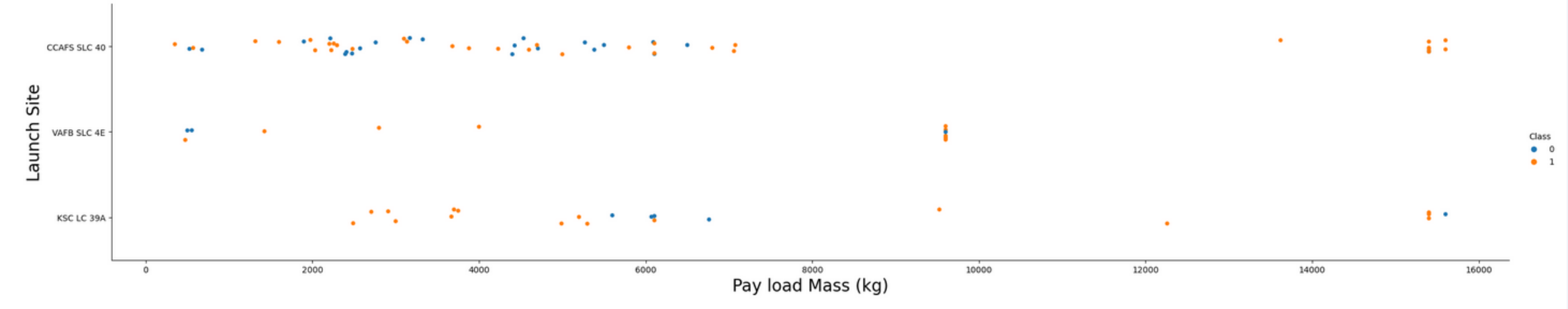


CCAFS LC-40 had a high level of failures in the early launches, but the success rates increase near the end.

VAFB SLC 4E had the lowest launches overall but had a high success rate.

KSC LC 39A had had similar success rates to VAFB SLC 4E.

Payload vs. Launch Site



CCAFS LC-40 Mostly involved low payload mass launches but did have high success with large payloads.

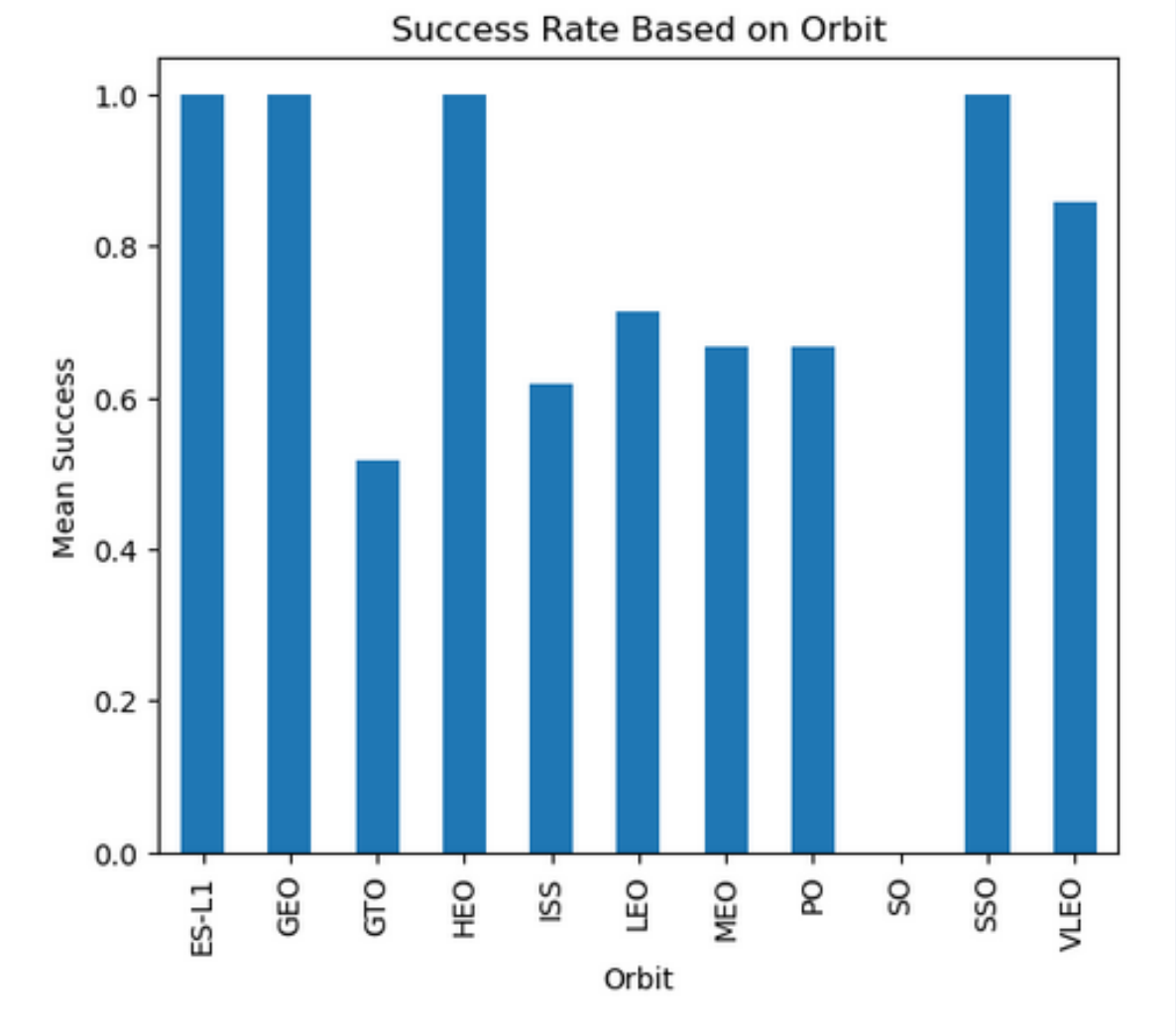
VAFB SLC 4E Generally had good success.

KSC LC 39A had higher success with small and large payloads.

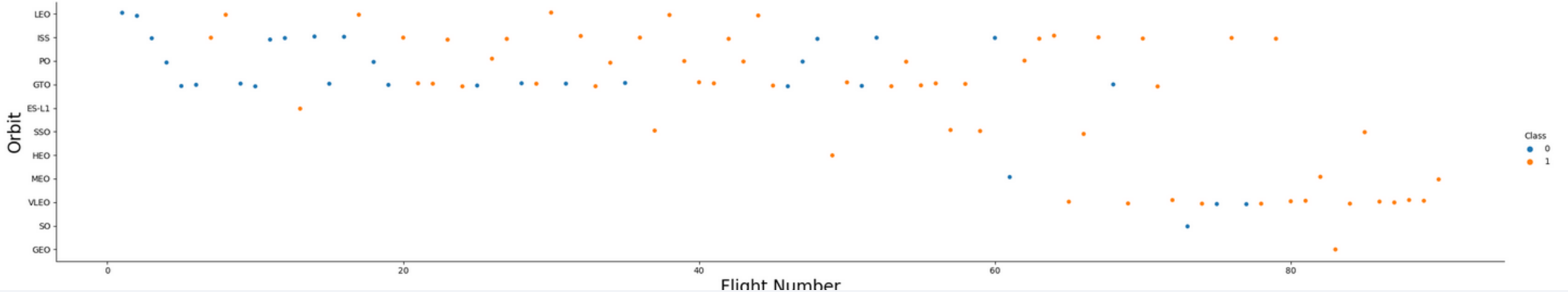
Success Rate vs. Orbit Type

Orbit types with the highest mean success rate were ES-L1, GEO, HEO, SSO, and VLEO.

The remaining orbits were closer to 50% success rate, with the exception being the SO orbit.



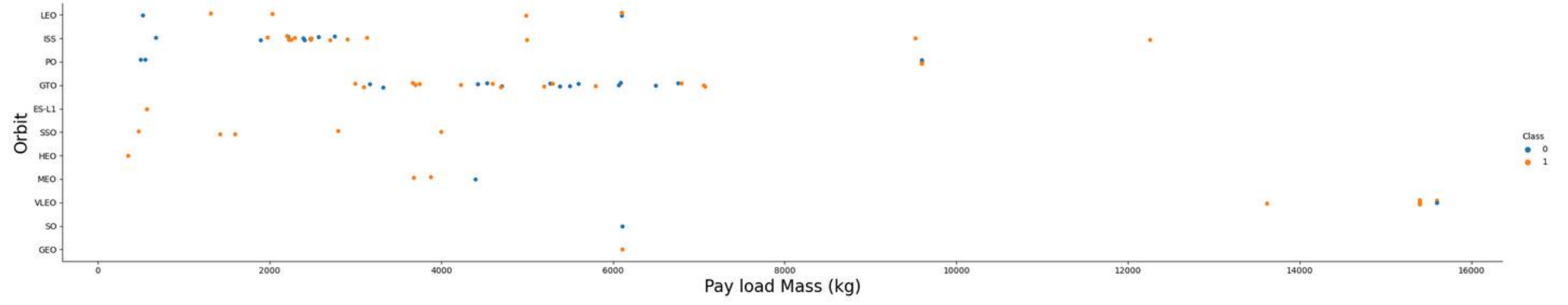
Flight Number vs. Orbit Type



Unlike the mean success rate, this chart builds upon that information by displaying success and failure data.

The mean success rate is best used with this information since having only one launch can skew the interpretation, Orbits like GEO, MEO, HEO, and SO are an example.

Payload vs. Orbit Type



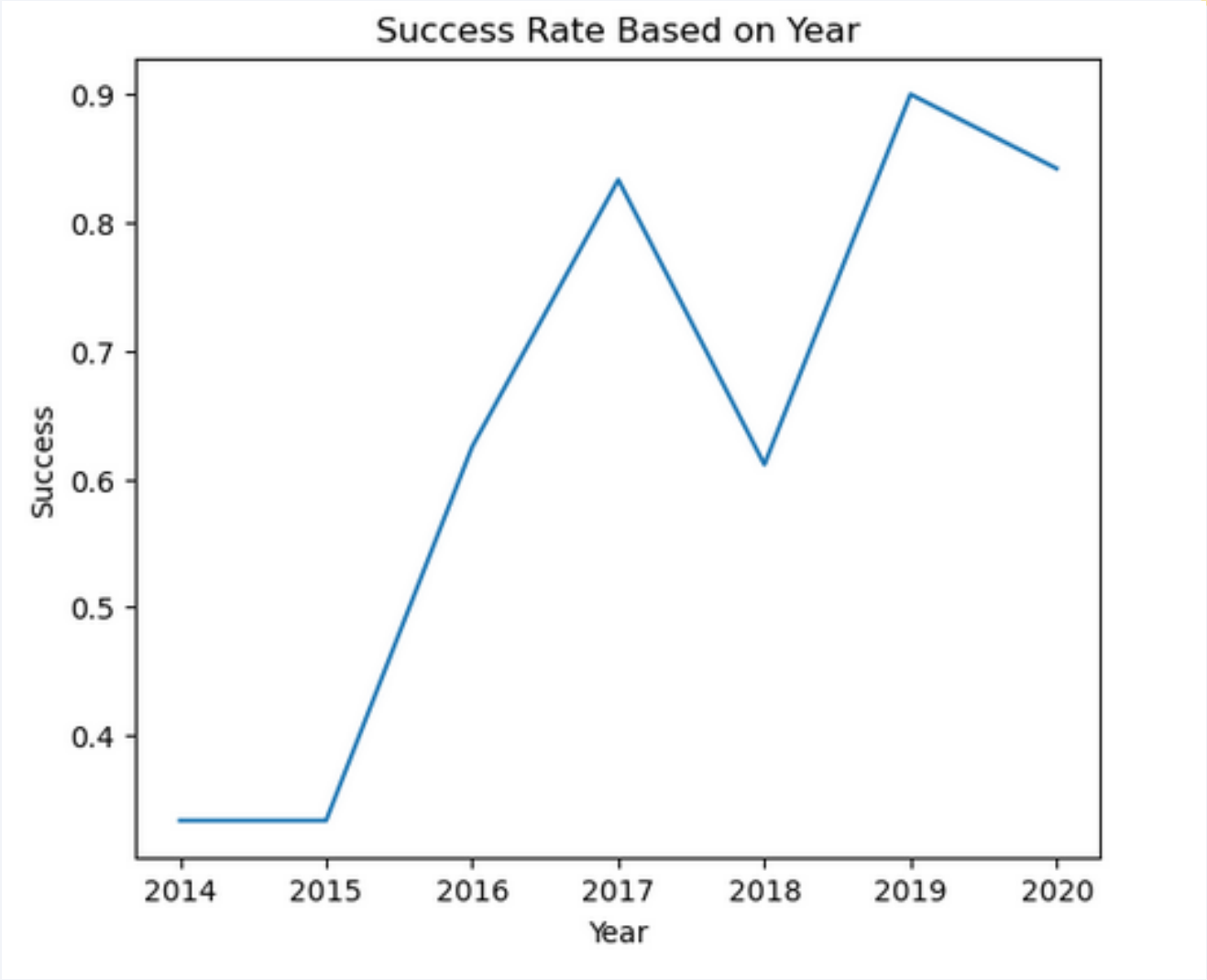
Combining the results of the orbit success rates and the payload more information on the success factors can be analyzed.

Again Orbits like GEO, MEO, HEO, and SO lack enough data to gather much statistically meaningful information.

Launch Success Yearly Trend

The success rate of the launches had a positive trend from 2015 to 2020.

2018 had a significant number of failed launches which did cause the average to drop.



All Launch Site Names

```
cur.execute("Select distinct Launch_Site from SPACEXTBL").fetchall()
```

```
[('CCAFS LC-40',), ('VAFB SLC-4E',), ('KSC LC-39A',), ('CCAFS SLC-40',)]
```

By utilizing the distinct command in the selection, the values returned would not contain any duplicates.

Launch Site Names Begin with 'KSC'

```
cur.execute("Select Launch_Site from SPACEXTBL WHERE Launch_Site like '%KSC%' LIMIT 5").fetchall()
```

```
[('KSC LC-39A',),
 ('KSC LC-39A',),
 ('KSC LC-39A',),
 ('KSC LC-39A',),
 ('KSC LC-39A',)]
```

By utilizing the like command in the selection, only values that contained the string after like will be returned.

Total Payload Mass

```
cur.execute("Select SUM(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE Customer = 'NASA (CRS)'").fetchall()
```

```
[(45596,)]
```

By utilizing the sum command in the selection, the total of the column will be returned, which in this case must match the customer ‘NASA (CRS)’

Average Payload Mass by F9 v1.1

```
cur.execute("Select AVG(PAYLOAD_MASS__KG_) from SPACEXTBL WHERE Booster_Version = 'F9 v1.1'").fetchall()
```

```
[(2928.4,)]
```

By utilizing the avg command in the selection, the average of the column will be returned, which in this case must match the booster version ‘F9 v1.1’

First Successful Ground Landing Date

```
cur.execute("Select min(Date) from SPACEXTBL WHERE \"Landing _Outcome\" like '%Success%(Drone Ship)%'").fetchall()
```

```
[('06-05-2016',)]
```

By utilizing the min command in the selection, the minimum of the column will be returned, which in this case must contain ‘Success’ and ‘(Drone Ship)’

Successful Drone Ship Landing with Payload between 4000 and 6000

```
cur.execute("Select Booster_Version from SPACEXTBL WHERE \"Landing _Outcome\"
    like '%Success%(Ground Pad)%' AND PAYLOAD_MASS__KG_>4000 and PAYLOAD_MASS__KG_<6000").fetchall()
```

```
[('F9 FT B1032.1',), ('F9 B4 B1040.1',), ('F9 B4 B1043.1',)]
```

The booster versions will be returned that meet the criteria by using greater then/less then operators, which in this case must contain ‘Success’ and ‘(Ground Pad)’, note the AND keyword must be present between conditions.

Total Number of Successful and Failure Mission Outcomes

```
print(cur.execute("Select count(Date) from SPACEXTBL WHERE \"Landing _Outcome\" like '%Success%'").fetchall())
print(cur.execute("Select count(Date) from SPACEXTBL WHERE \"Landing _Outcome\" like '%Failure%'").fetchall())
```

```
[(61,)]
[(10,)]
```

By utilizing the count command in the selection, the count of the column will be returned, which in this case must contain ‘Success’ or ‘Failure’

Boosters Carried Maximum Payload

```
maxmass=cur.execute("Select MAX(PAYLOAD_MASS__KG_) from SPACEXTBL").fetchall()
print(maxmass[0][0])
cur.execute("Select Booster_Version from SPACEXTBL WHERE PAYLOAD_MASS__KG_={}".format(maxmass[0][0])).fetchall()
```

```
15600
[('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 ',)]
```

By utilizing the max command in the selection, the maximum of the column will be returned, as a check the list of those which contained the maximum mass can be listed.

2015 Launch Records

```
monthnames=['January','February','March','April','May','June','July','August','September','October','November','December']
sql_results=cur.execute("Select substr(Date, 4, 2),Booster_Version,\"Landing _Outcome\",Launch_Site from SPACEXTBL WHERE
                         substr(Date,7,4)='2017' AND \"Landing _Outcome\" like '%Success%(Ground Pad)%'").fetchall()

result_list=[]
for rows in sql_results:
    result_list.append([monthnames[int(rows[0])-1],rows[1],rows[2],rows[3]])
result_list
```

```
[['February', 'F9 FT B1031.1', 'Success (ground pad)', 'KSC LC-39A'],
 ['May', 'F9 FT B1032.1', 'Success (ground pad)', 'KSC LC-39A'],
 ['June', 'F9 FT B1035.1', 'Success (ground pad)', 'KSC LC-39A'],
 ['August', 'F9 B4 B1039.1', 'Success (ground pad)', 'KSC LC-39A'],
 ['September', 'F9 B4 B1040.1', 'Success (ground pad)', 'KSC LC-39A'],
 ['December', 'F9 FT  B1035.2', 'Success (ground pad)', 'CCAFS SLC-40']]
```

The sqlite code will return the data with the month numbers, the additional python code was added to convert the numbers to months due to sqlite limitations.

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

```
cur.execute("Select Date,Booster_Version,\"Landing _Outcome\",Launch_Site from SPACEXTBL WHERE \"Landing _Outcome\"
                like '%Success%' AND DATE(substr(Date,7,4)||'-'||substr(Date,4,2)||'-'||substr(Date,1,2)) BETWEEN DATE('2010-06-04') 
                AND DATE('2017-03-20') ORDER BY DATE(substr(Date,7,4)||'-'||substr(Date,4,2)||'-'||substr(Date,1,2)) DESC").fetchall()
```

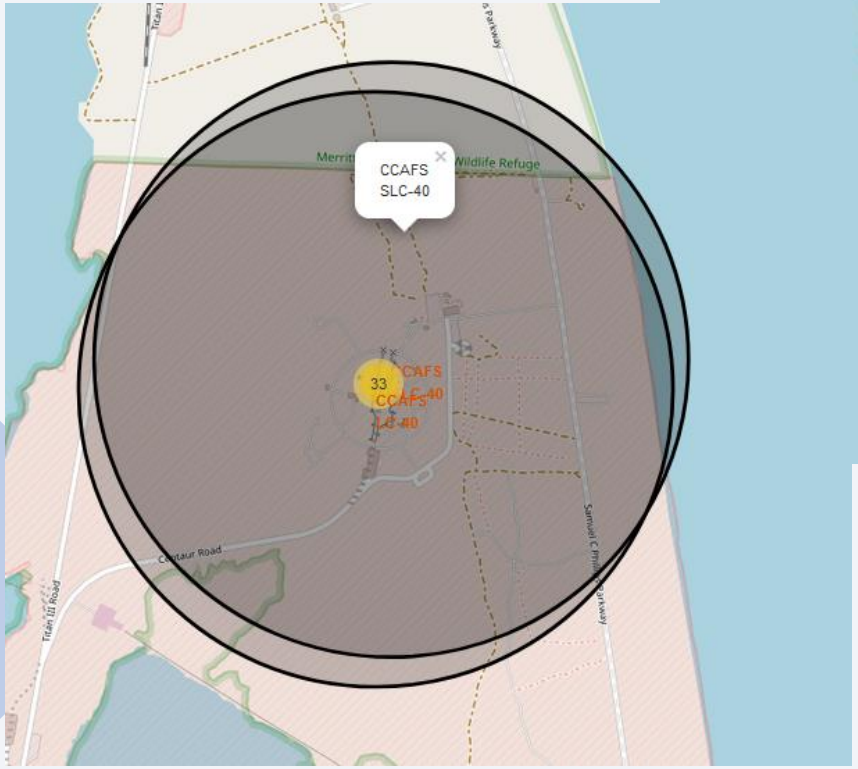
```
[('19-02-2017', 'F9 FT B1031.1', 'Success (ground pad)', 'KSC LC-39A'),
 ('14-01-2017', 'F9 FT B1029.1', 'Success (drone ship)', 'VAFB SLC-4E'),
 ('14-08-2016', 'F9 FT B1026', 'Success (drone ship)', 'CCAFS LC-40'),
 ('18-07-2016', 'F9 FT B1025.1', 'Success (ground pad)', 'CCAFS LC-40'),
 ('27-05-2016', 'F9 FT B1023.1', 'Success (drone ship)', 'CCAFS LC-40'),
 ('06-05-2016', 'F9 FT B1022', 'Success (drone ship)', 'CCAFS LC-40'),
 ('08-04-2016', 'F9 FT B1021.1', 'Success (drone ship)', 'CCAFS LC-40'),
 ('22-12-2015', 'F9 FT B1019', 'Success (ground pad)', 'CCAFS LC-40')]
```

The sqlite code does not contain as many date/time commands as others, so dates had to be reconstructed to allow for proper sorting.

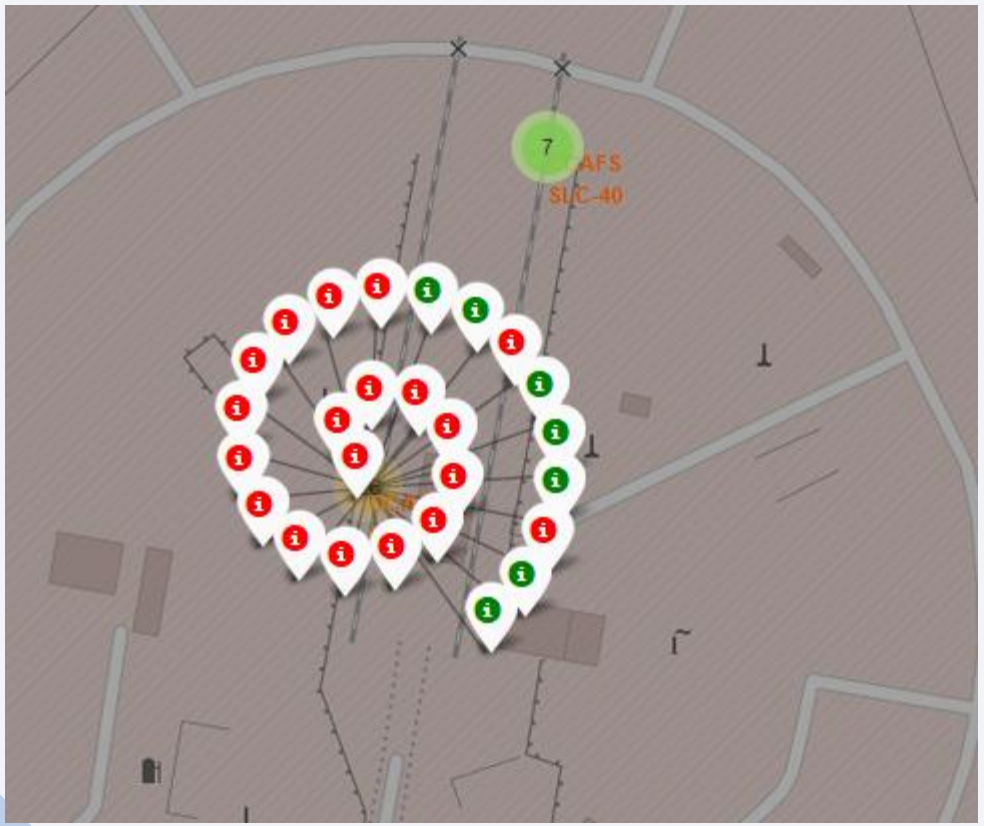
Section 3

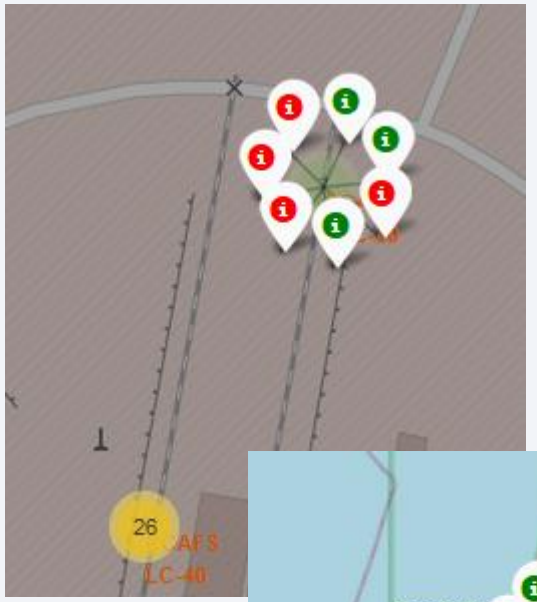
Launch Sites Proximities Analysis

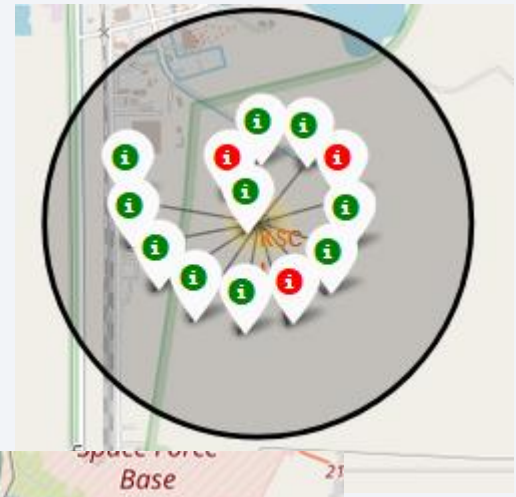
Folium Map Showing Launch Sites and Clusters

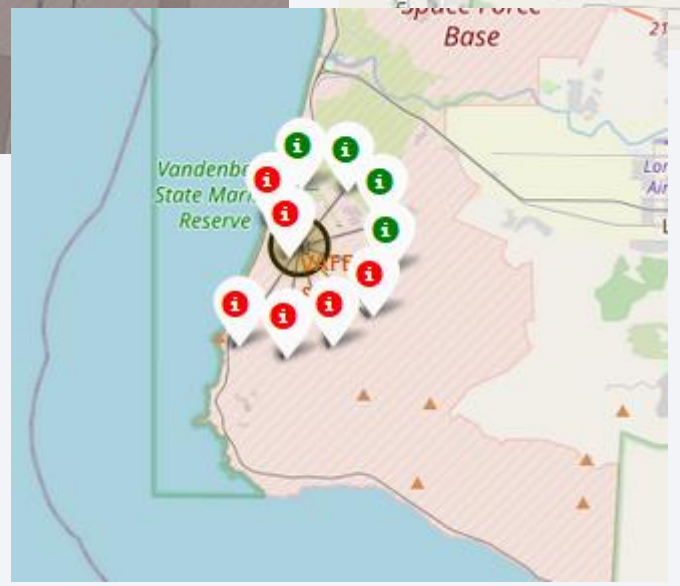


Folium Map Showing Launch Outcomes

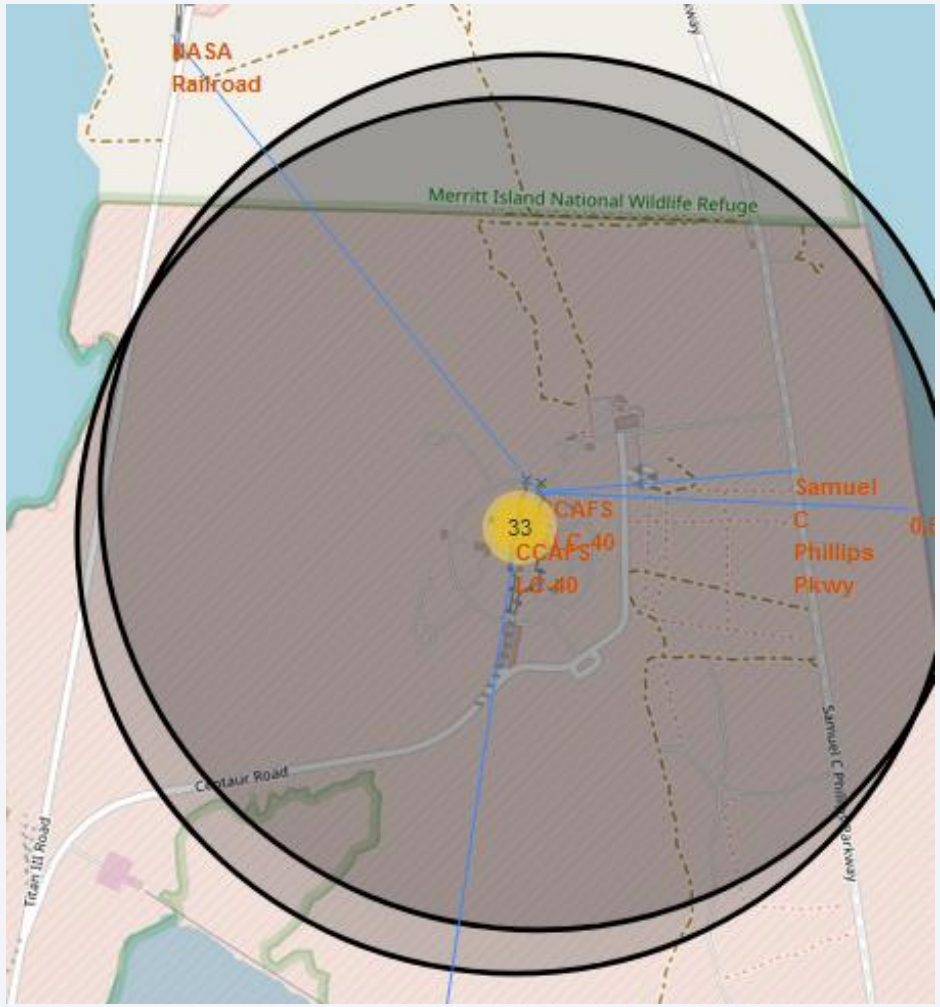


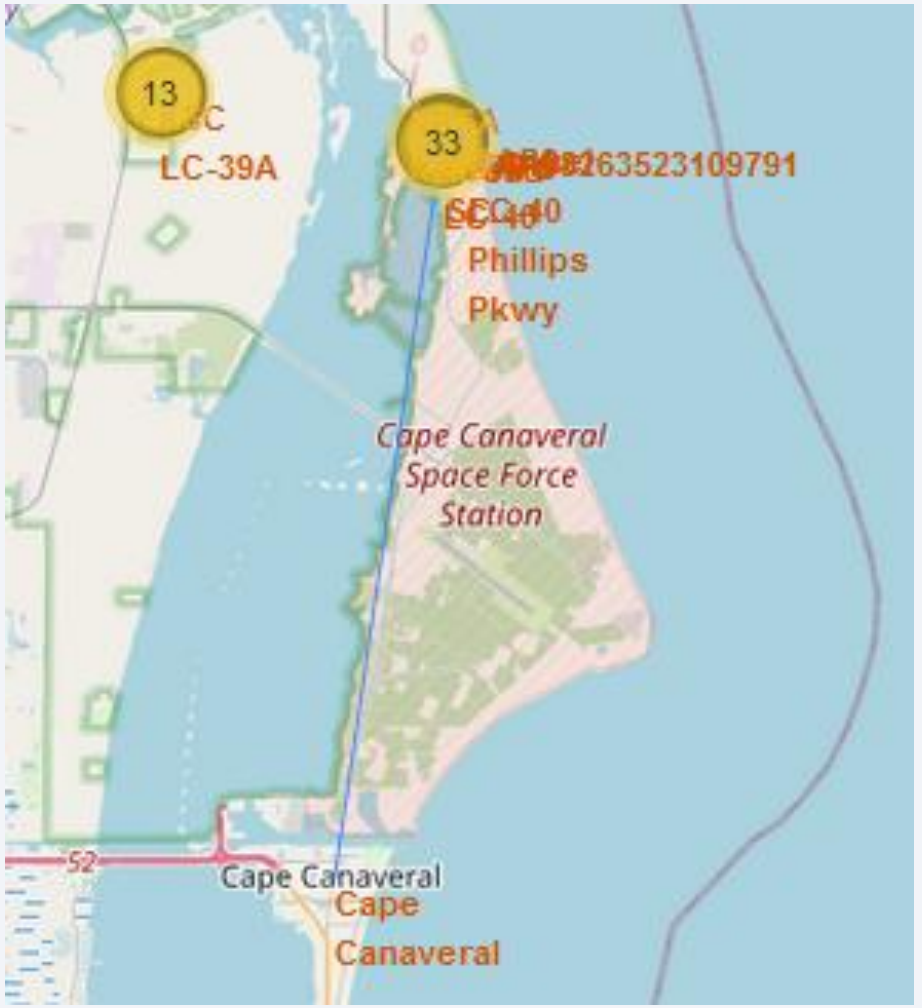


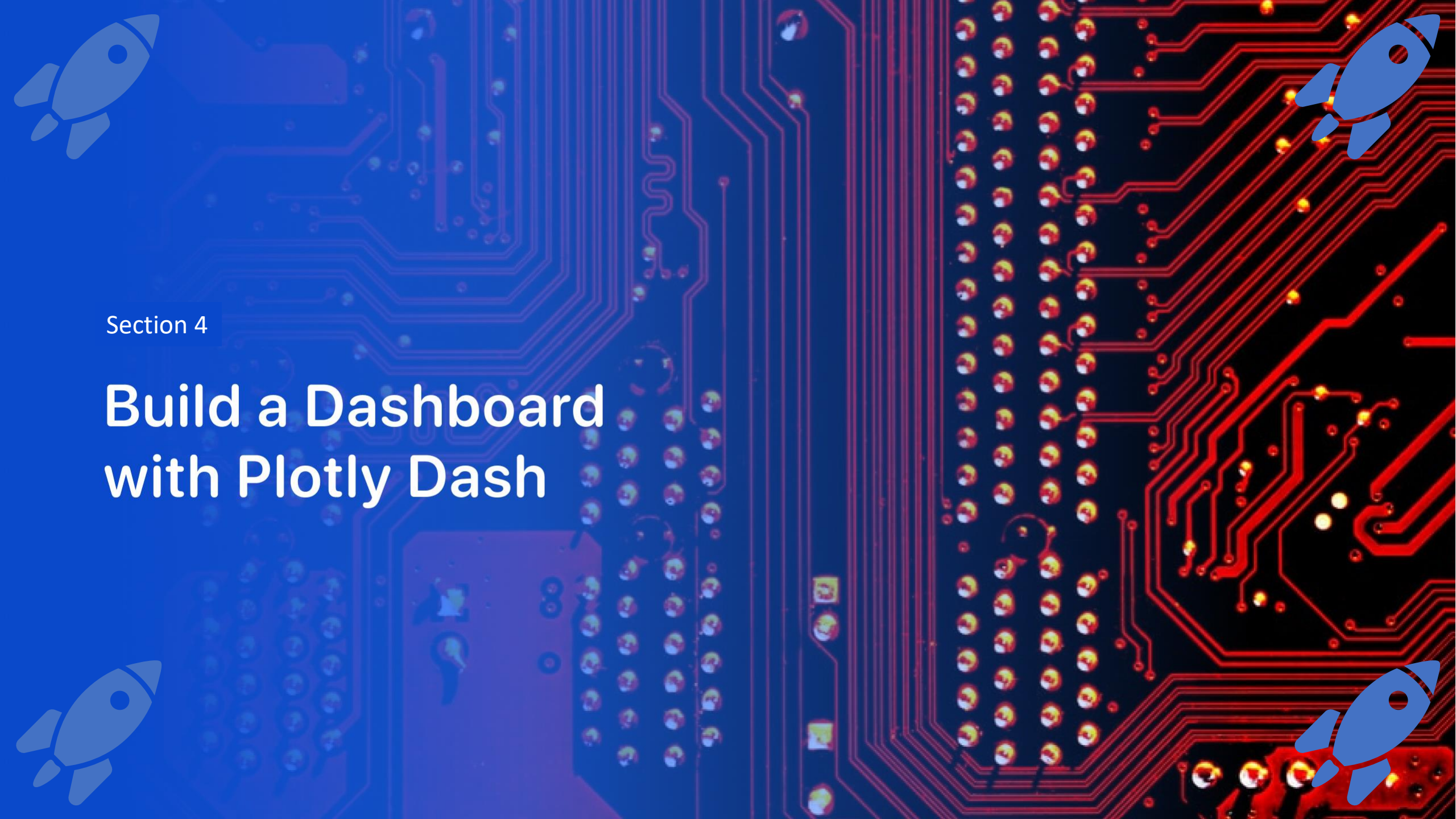




Folium Map Showing Proximity Lines



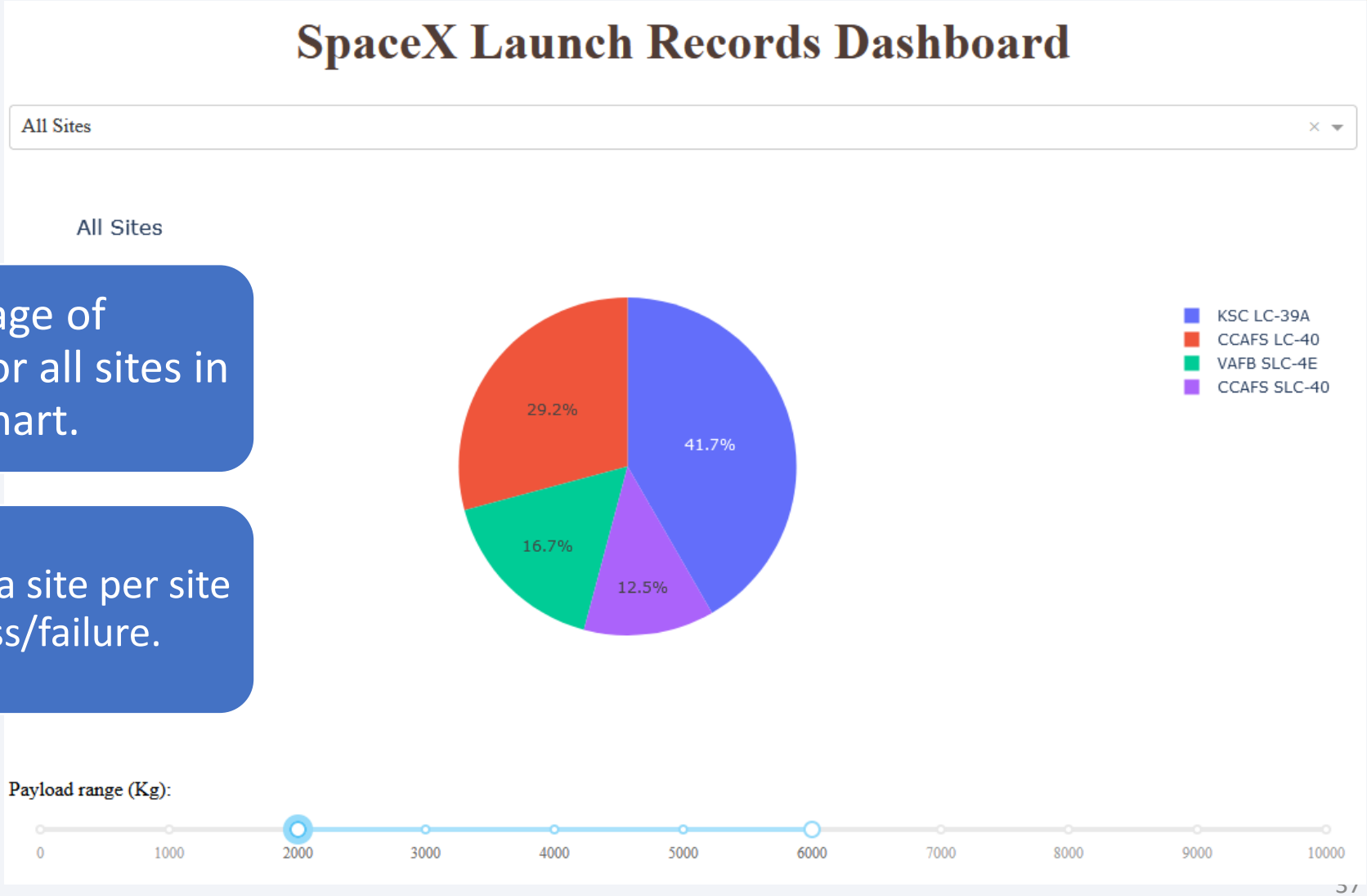



Section 4
Build a Dashboard with Plotly Dash

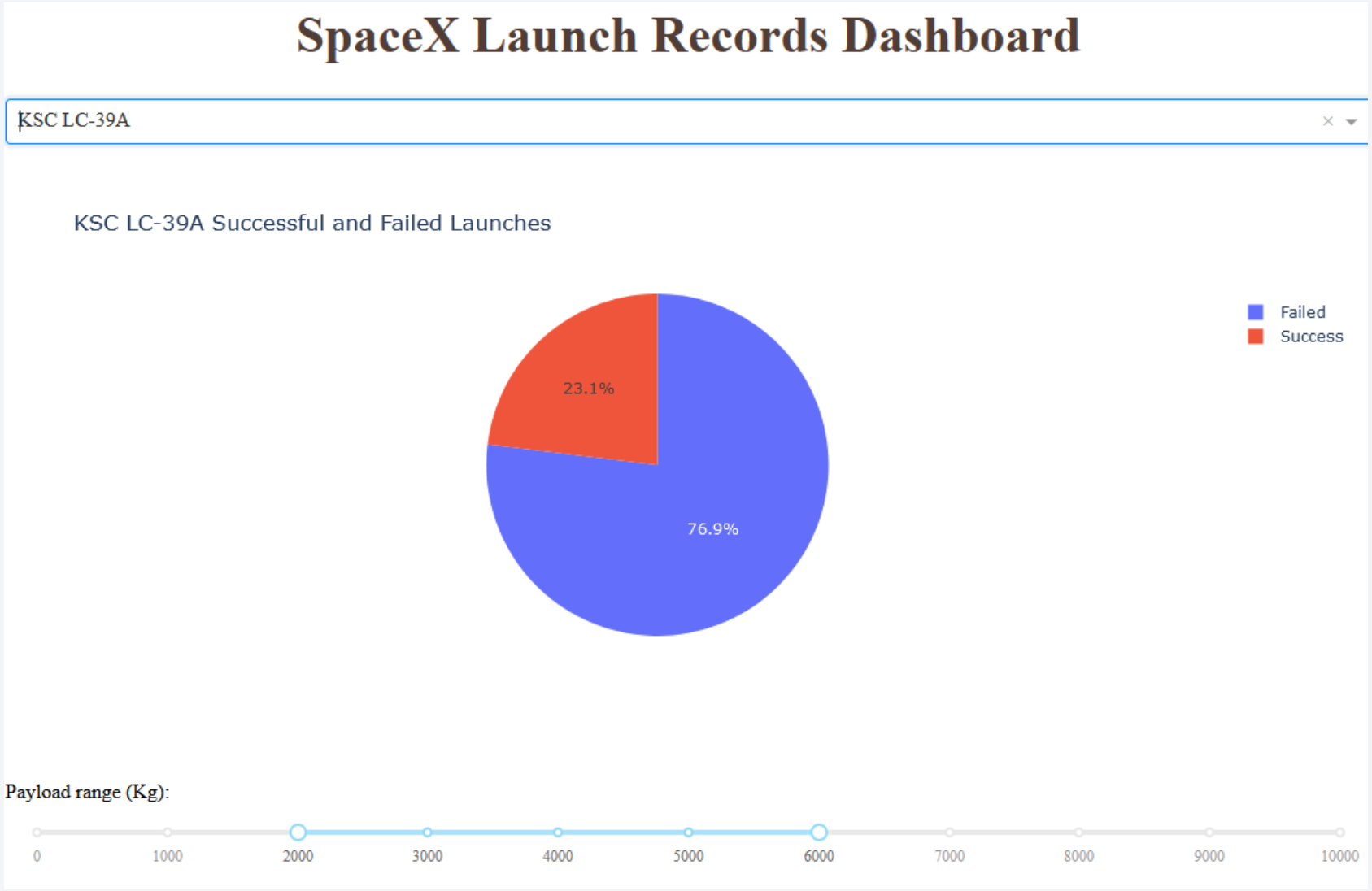
Dashboard Results of All Sites

Displays the percentage of successful landings for all sites in the format of a pie chart.

This can be applied on a site per site to show percent success/failure.



Dashboard Site with Highest Success Rate



Dashboard Payload Mass vs Success Rate

The lower chart displays the payload information as it relates to the success and failure rate.

The payload mass slider allows the data to be analyzed using different limits in payload to gain deeper insight into the data.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

The results for the best accuracy in training were all close, but in using test data all but the decision tree provide most accurate results.

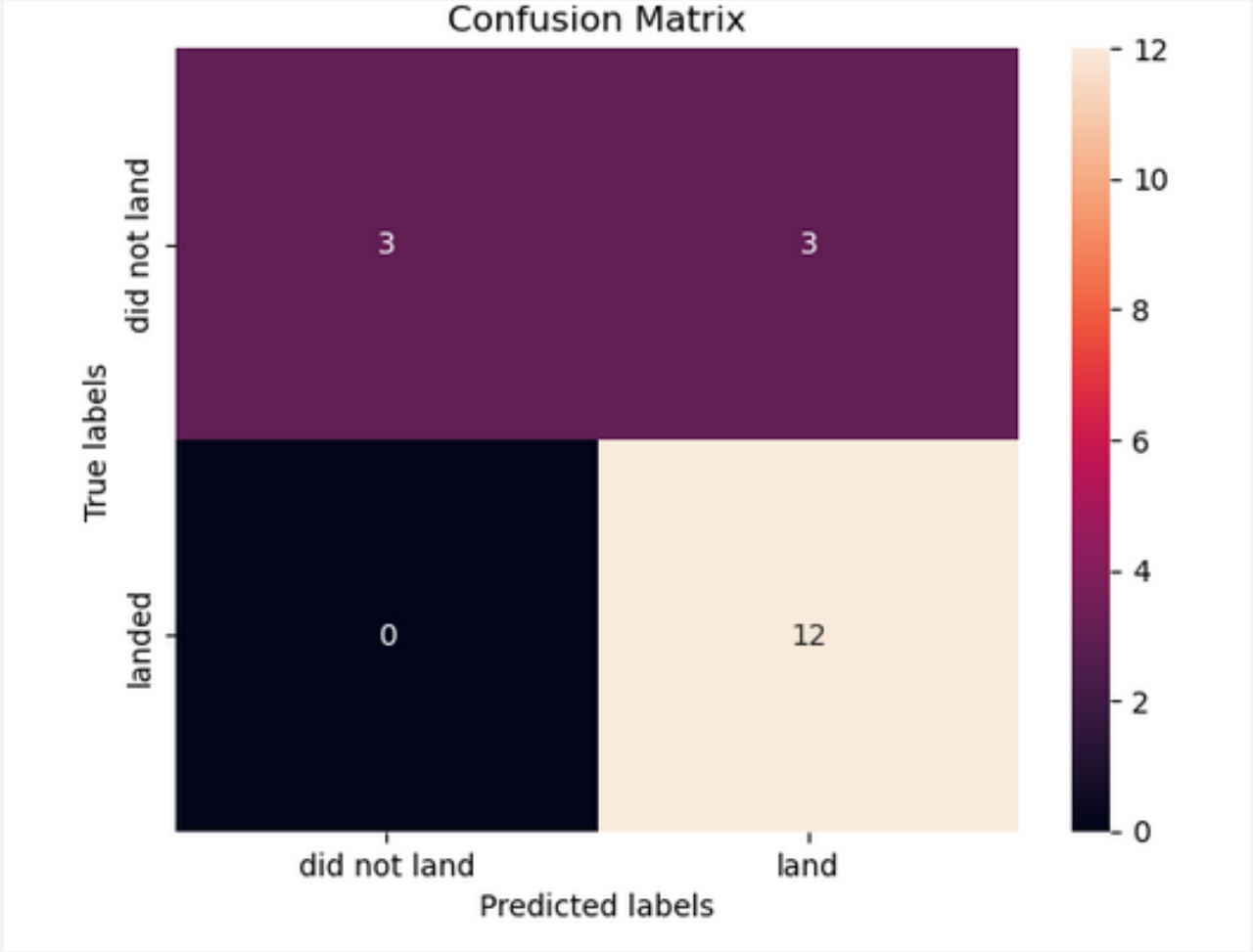
	Best Train Accuracy	Accuracy	Jaccard	F1
Logistic	0.846429	0.833333	0.80	0.888889
SVM	0.848214	0.833333	0.80	0.888889
Decision Tree	0.875000	0.777778	0.75	0.857143
KNN	0.848214	0.833333	0.80	0.888889

In the end the Logistic Regression, Support Vector Machine, and the K-Nearest Neighbor all resulted in the same prediction accuracy, so those methods would produce the most reliable results in practice.

Confusion Matrix

As stated in the previous slide, the Logistic Regression, Support Vector Machine, and the K-Nearest Neighbor provided similar results.

All models only had false positives in the confusion matrix, leading to the assumptions that all models will more commonly predict a successful landing.



Thank you!