IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Braulio H Gutierrez Roldan
21 April 2024

Outline

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

Executive Summary

- ✓ Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- ✓ Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

Introduction

Space X offers its Falcon 9 rocket launches at a notable price point of 62 million dollars on its website. This is substantially lower compared to other providers whose charges are over 165 million dollars per launch. A significant portion of these cost savings stems from Space X's ability to recycle the rocket's initial stage. Understanding whether this first stage can be landed is crucial, as it plays a pivotal role in determining the overall expenses of the launch. Such insights are invaluable, especially for any rival firms that may want to submit a competitive bid for launching services. The main aim of this initiative is to develop a computational model capable of accurately forecasting the chances of the first stage landing successfully.

The project seeks to unravel several queries:

- What are the key factors that influence the successful landing of the rocket?
- How do different elements interact and contribute to the likelihood of a successful landing?
- What specific conditions must be met to facilitate a consistent and reliable landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Create 3 funcionts to obtain From the link https://api.spacexdata.com/v4/rockets the data of Rocket, Launchpad, and payloads data, then from the same link and save in a data frame
- Perform data wrangling
  - Identify the null data in the data frame and work with them
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - We build different strategies to identify the correct model to predict the launch success

Data Collection

We gathered the necessary information through a combination of techniques. Initially, we made GET requests to the SpaceX API to retrieve the data. Following this, we used the .json() method to decode the received JSON content and then utilized the .json_normalize() function to structure this data into a Pandas DataFrame. Subsequent steps involved cleaning the data, filling any gaps where data might be missing, and ensuring its completeness. Moreover, we harnessed the power of BeautifulSoup to scrape Falcon 9 launch data from Wikipedia. The aim here was to meticulously extract the launch data embedded within the HTML tables, process this data to parse it accurately, and ultimately convert it into a DataFrame format conducive to thorough analysis.

Data Collection – SpaceX API

- Using the get request to collect data

- https://github.com/pigeon0016/FinalDCMod10.git

In [2]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [3]:
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [4]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Data Collection - Scraping

- Applied webscrapping to obtain data from falcon 9 with beautiful soap

- https://github.com/pigeon0016/FinalDCMod10.git

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.

In [6]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

In [7]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

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

In [8]:
```
# Use soup.title attribute
print(soup.title)
```

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

In [9]:
```
# 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.

In [10]:
```
# 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.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC
</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">V
ersion,<br/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></su
p>
```

Next, we just need to iterate through the `<th>` elements and apply the provided `extract_column_from_header()` to extract column name one by one

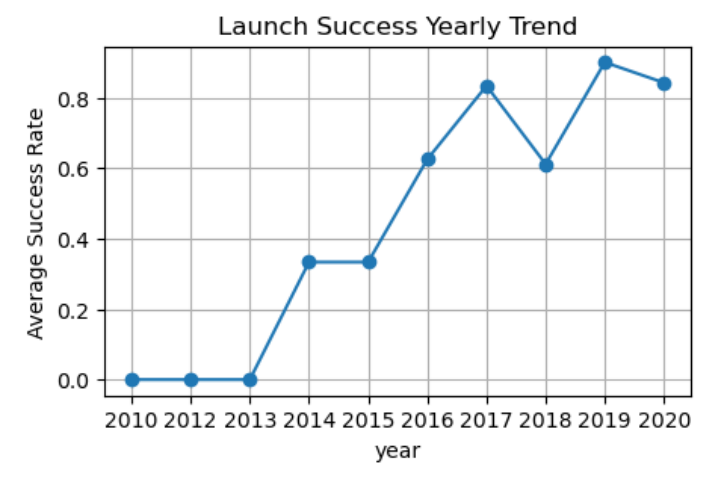
In [11]:
```
column_names = []
for th in first_launch_table.find_all('th'):
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names
```

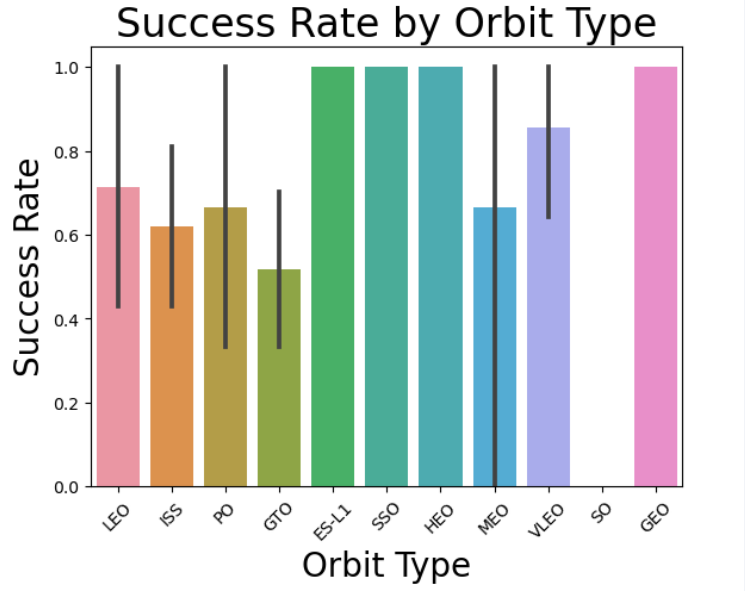
Data Wrangling

- During the preliminary phase, we conducted an exploratory analysis of the data to establish the labels needed for training our model. This included tallying the frequency of launches from each site as well as documenting the various orbits' occurrences. Furthermore, we synthesized a label for the landing outcome based on the data in the outcome column. The final step involved collating these insights and exporting them into a CSV file for future reference.
- https://github.com/pigeon0016/FinalDCMod10.git

EDA with Data Visualization

- We delved into the dataset by crafting visual representations to discern patterns between various elements: we plotted the correlation between the sequence of flights and their respective launch sites, examined the association between payload mass and launch sites, and analyzed the success rates correlated to each type of orbit. Additionally, we investigated the relationship between the flight sequence and orbit types, and charted the annual trends in launch successes to gain deeper insights.
- https://github.com/pigeon0016/FinalDCMod10.git





EDA with SQL

We seamlessly integrated the SpaceX dataset into a PostgreSQL database directly from our Jupyter notebook environment. By employing SQL-based exploratory data analysis (EDA) within the notebook, we were able to derive valuable insights from the dataset. Specifically, we executed SQL queries to uncover unique details such as:

- Identifying each distinct launch site involved in the space missions.
- Calculating the cumulative payload mass ferried by the boosters on NASA (CRS) missions.
- Determining the mean payload mass transported by the Falcon 9 version 1.1 boosters.
- Counting the overall successful and unsuccessful mission outcomes.
- Listing all unsuccessful landing attempts on drone ships, along with the associated booster versions and the names of the launch sites from which they were deployed.

- https://github.com/pigeon0016/FinalDCMod10.git

Build an Interactive Map with Folium

We annotated all the launch sites on the folium map, incorporating various map elements like markers, circles, and lines to denote the outcomes of the launches from each site. We categorized launch outcomes into two classes: class 0 for failures and class 1 for successes.

Utilizing color-coded marker clusters, we were able to discern the launch sites with higher success rates. We meticulously measured the distances from each launch site to nearby points of interest. This allowed us to explore questions such as:

- The proximity of launch sites to railways, highways, and coastlines.
- Whether launch sites maintain a certain distance from populated areas for safety and security reasons.
- https://github.com/pigeon0016/FinalDCMod10.git

Build a Dashboard with Plotly Dash

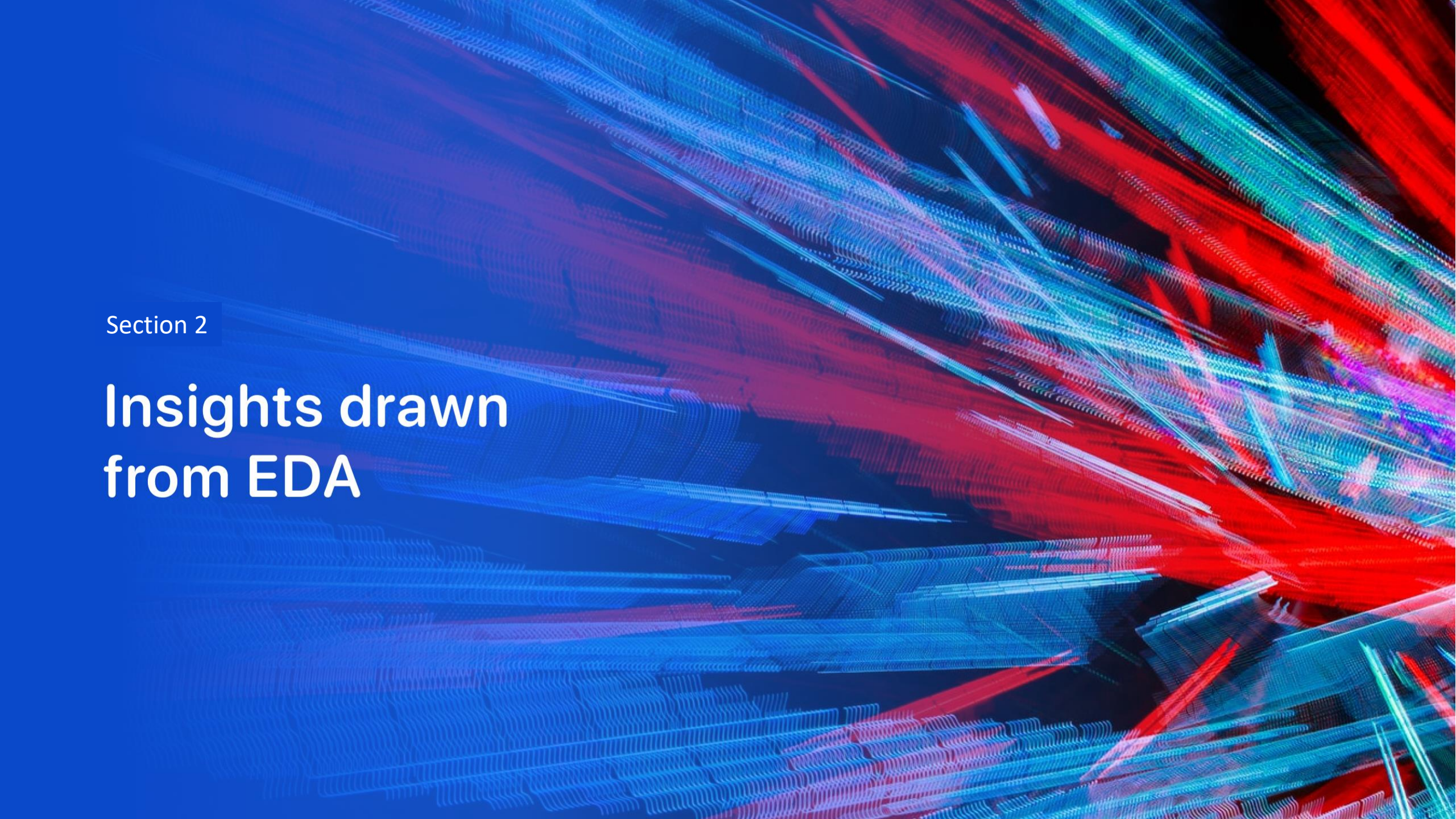
- We constructed an interactive dashboard using Plotly Dash to visualize various aspects of the space missions. This dashboard includes pie charts that display the distribution of total launches from different sites. Additionally, we created scatter plots to illustrate the correlation between launch outcomes and payload mass in kilograms, taking into account the variations across different booster versions.
- https://github.com/pigeon0016/FinalDCMod10.git

Predictive Analysis (Classification)

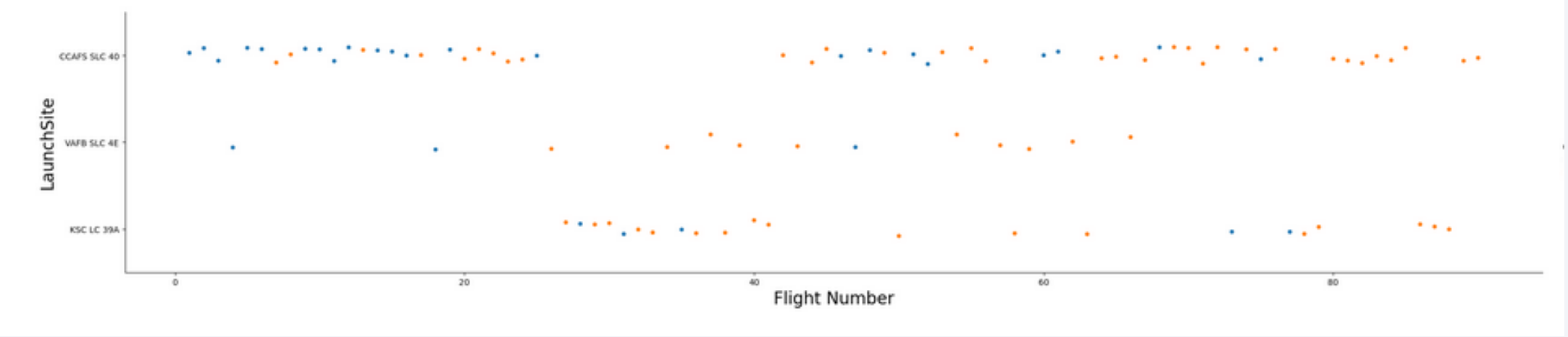
- We initiated our analysis by importing the data utilizing libraries such as NumPy and Pandas. After transforming the dataset, we partitioned it into training and testing subsets for model evaluation. A range of machine learning models were then constructed, and their hyperparameters meticulously optimized with the aid of GridSearchCV. Our primary metric to gauge model performance was accuracy. To further refine our models, we applied techniques of feature engineering and algorithm tuning. Through this process, we were able to identify the classification model that delivered the most robust performance.
- https://github.com/pigeon0016/FinalDCMod10.git

Results

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

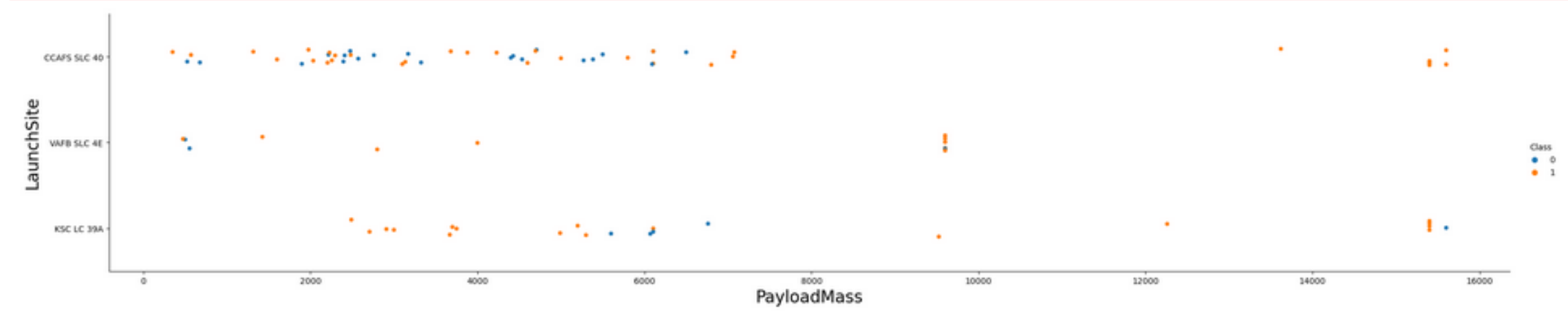

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



- The large flught at the launch site has the success rate

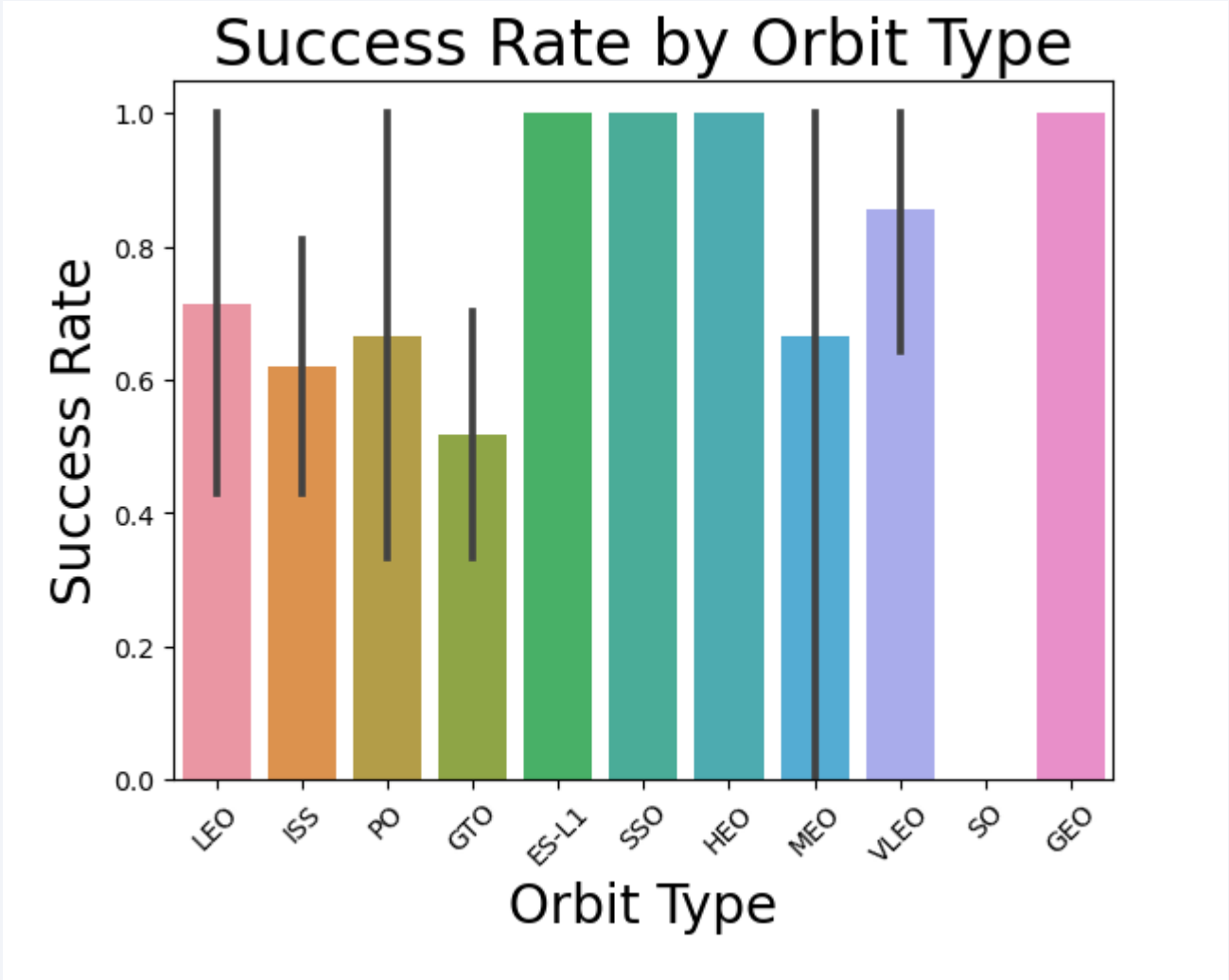
Payload vs. Launch Site



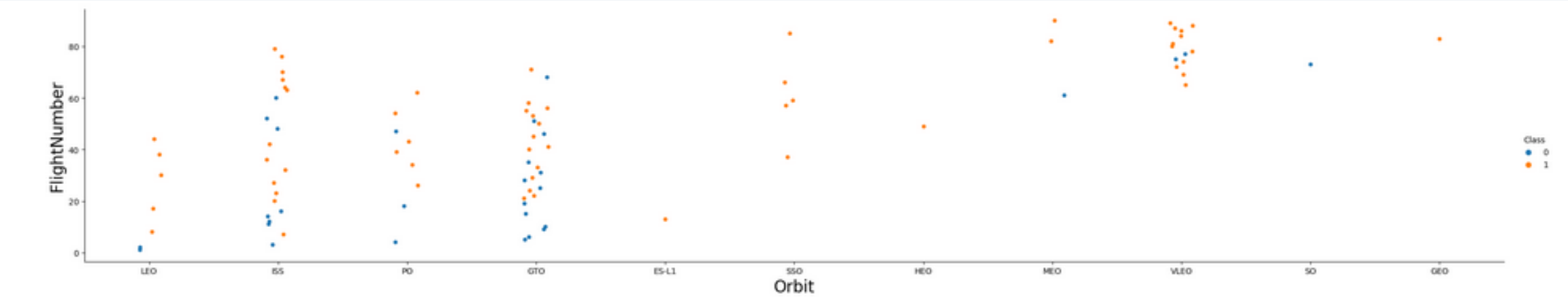
- The VAFB SLC launchstite there are no rockets for heavypayload

Success Rate vs. Orbit Type

- The ESL1 sso and heo has the best success rate

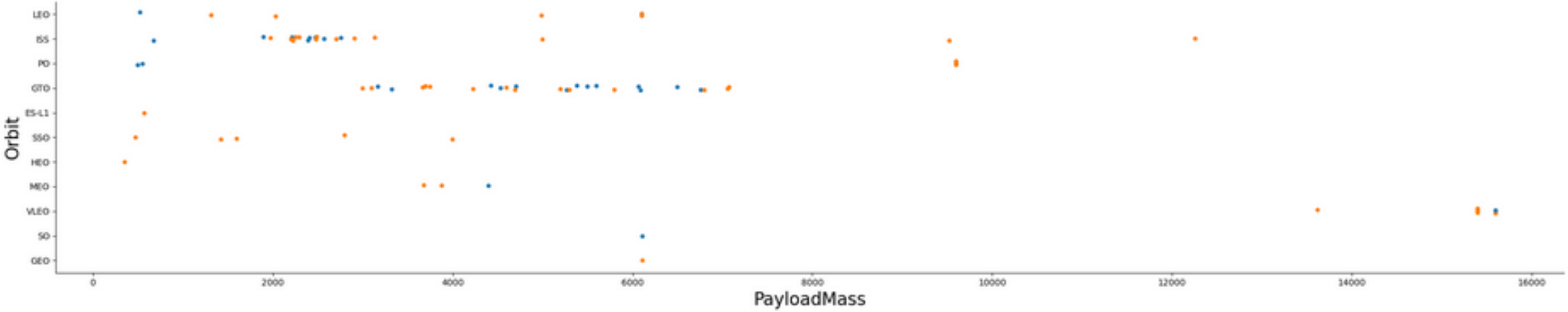


Flight Number vs. Orbit Type



- the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit

Payload vs. Orbit Type



- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS

Launch Success Yearly Trend

- the success rate since 2013 kept increasing till 2017 (stable in 2014) and after 2015 it started increasing

All Launch Site Names

```
In [16]:  %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

Out[16]:

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

All the sites names using SQL

Launch Site Names Begin with 'CCA'

Task 2

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

In [17]:
```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

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

Out[17]:

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

Task 3

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

In [28]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer LIKE 'NASA (CRS)'
```

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

Out[28]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

Task 4

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

In [30]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL WHERE Booster_version = 'F9 v1.1'
```

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

Out[30]:

AVERAGE_PAYLOAD_MASS
2928.4

First Successful Ground Landing Date

In [14]:

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [41]:
```
%sql SELECT Mission_Outcome, COUNT(*) as Total FROM SPACEXTBL GROUP BY Mission_Outcome
```

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

Out[41]:

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

Boosters Carried Maximum Payload

Task 8

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

In [43]:
```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Out[43]:

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

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.

In [46]:
```
%sql SELECT substr(Date, 6, 2) AS Month, Booster_Version,Launch_Site,Landing_Outcome FROM SPACEXTBL WHERE substr(Date, 0
```

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

Out[46]:

Month	Booster_Version	Launch_Site	Landing_Outcome
01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

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.

In [45]:
```
%sql SELECT Landing_Outcome, COUNT(*) as Count FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY
```

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

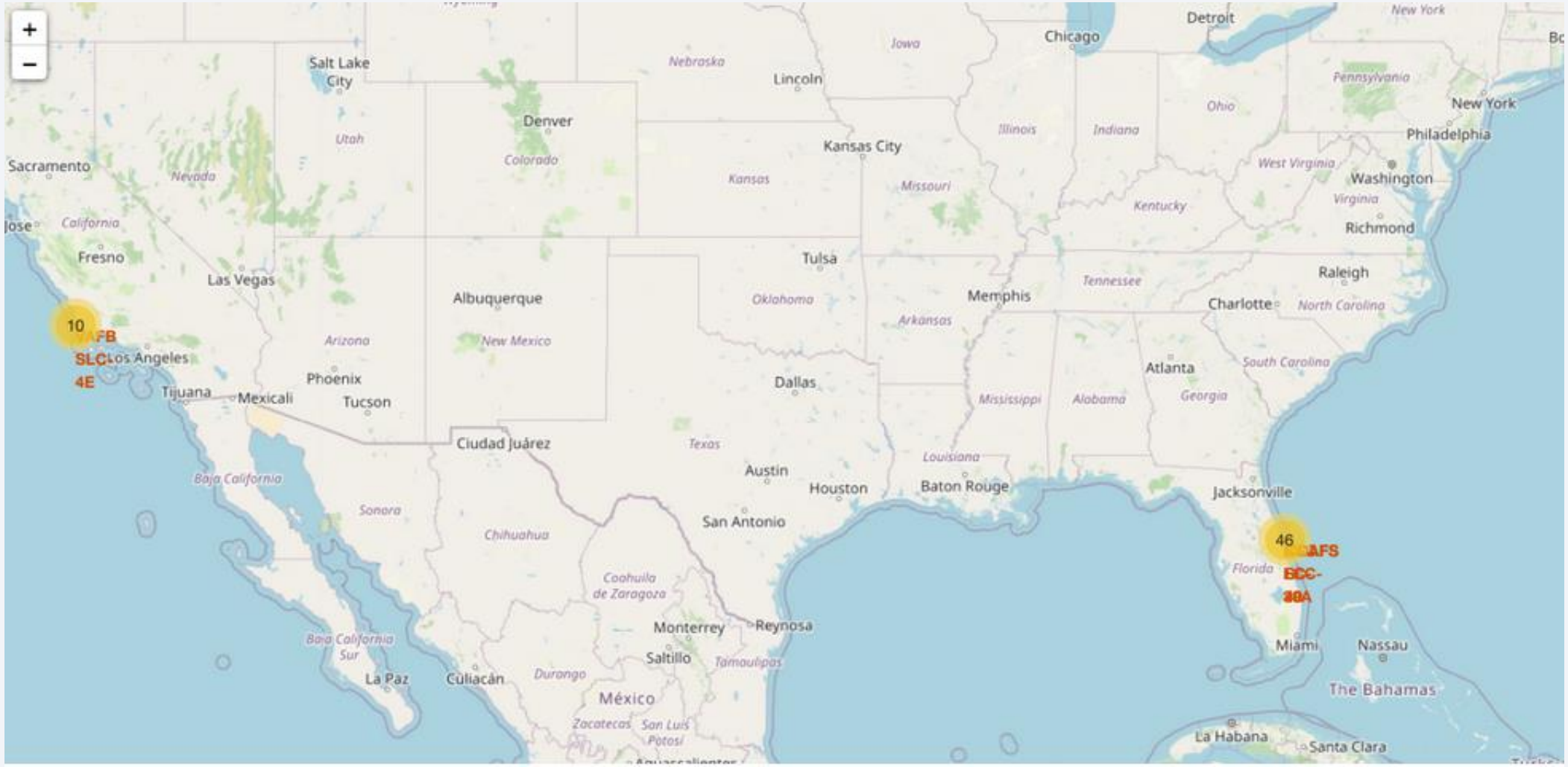
Out[45]:

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

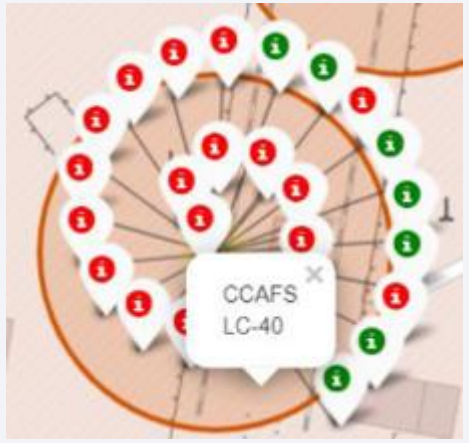
Section 3

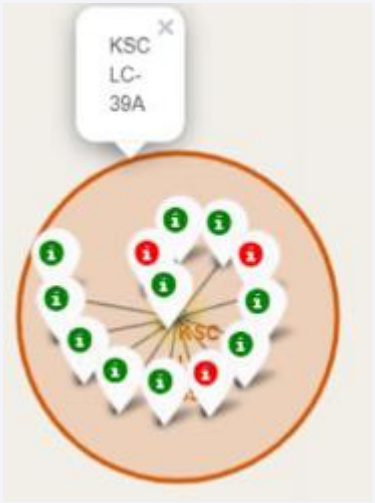
Launch Sites Proximities Analysis

All launch sites

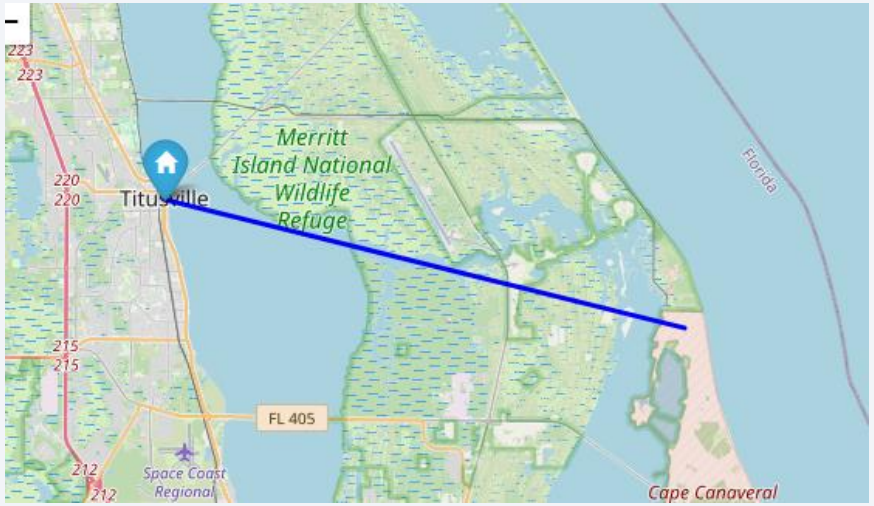


Launch sites

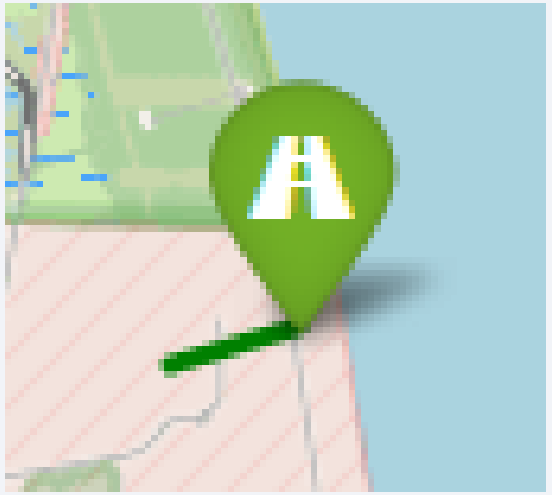




Distances from lounch site

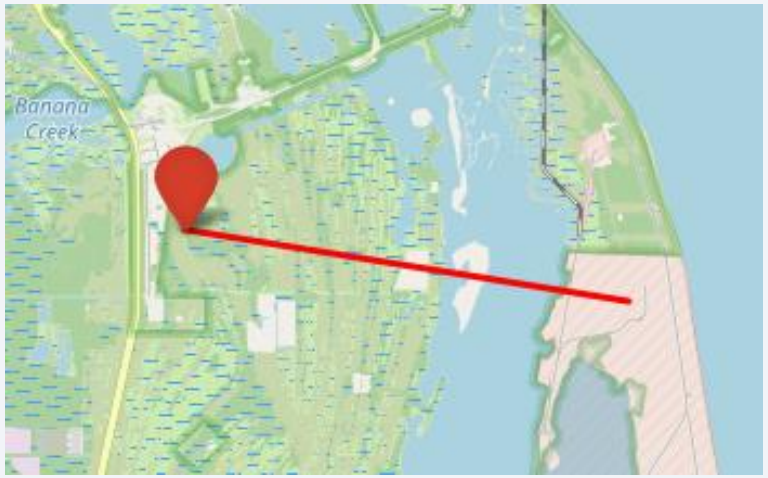


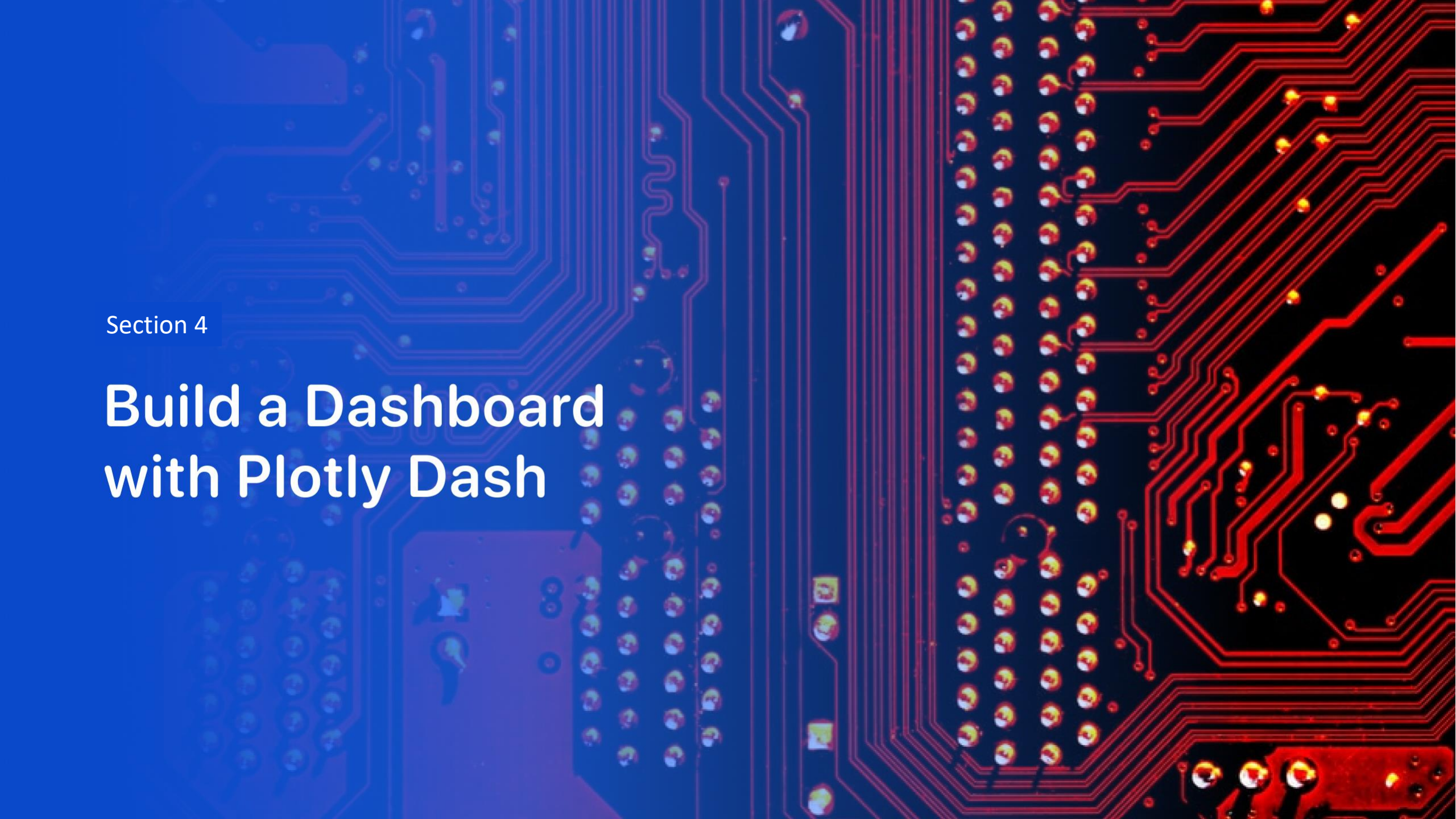
Near City



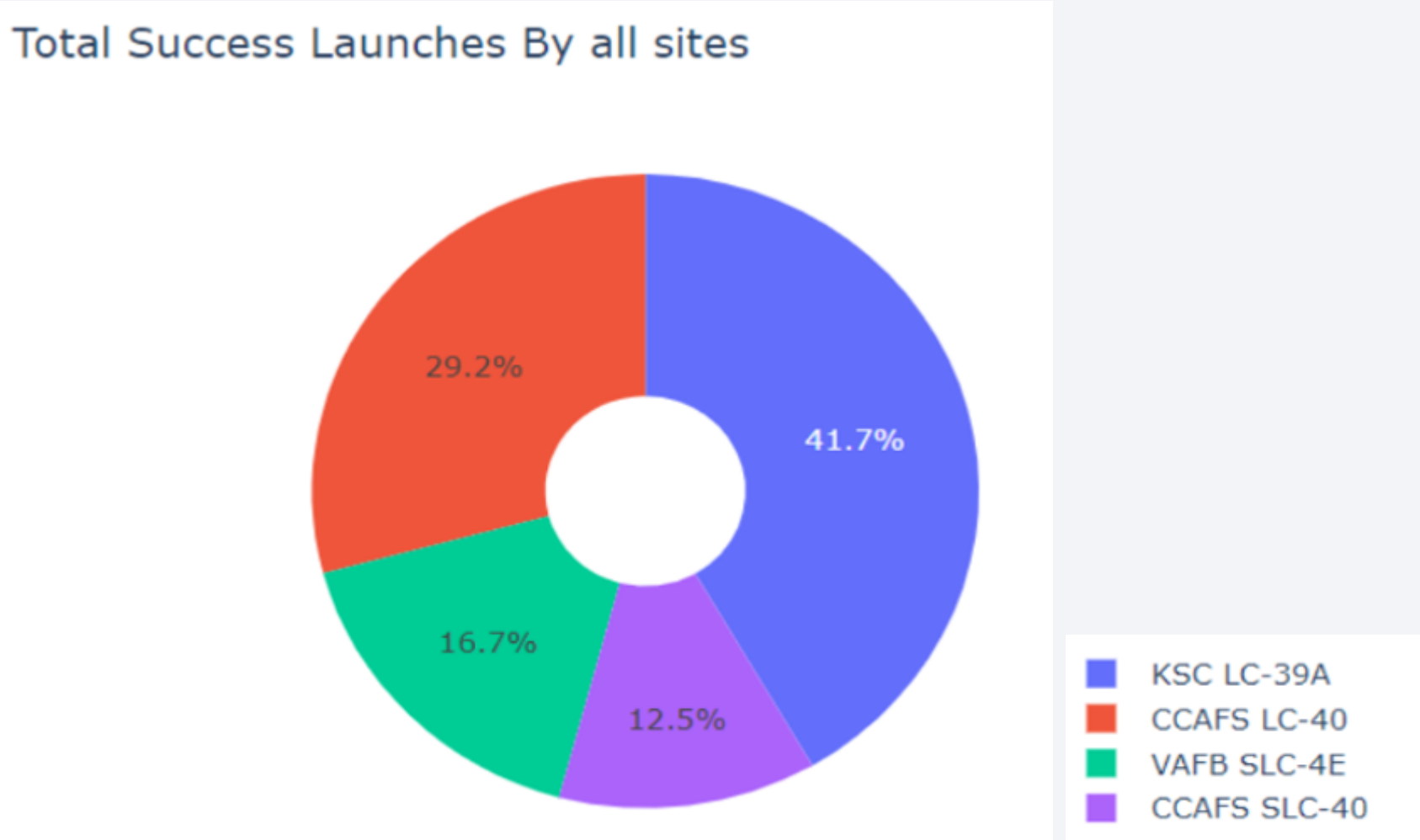
Near highway

Near railway

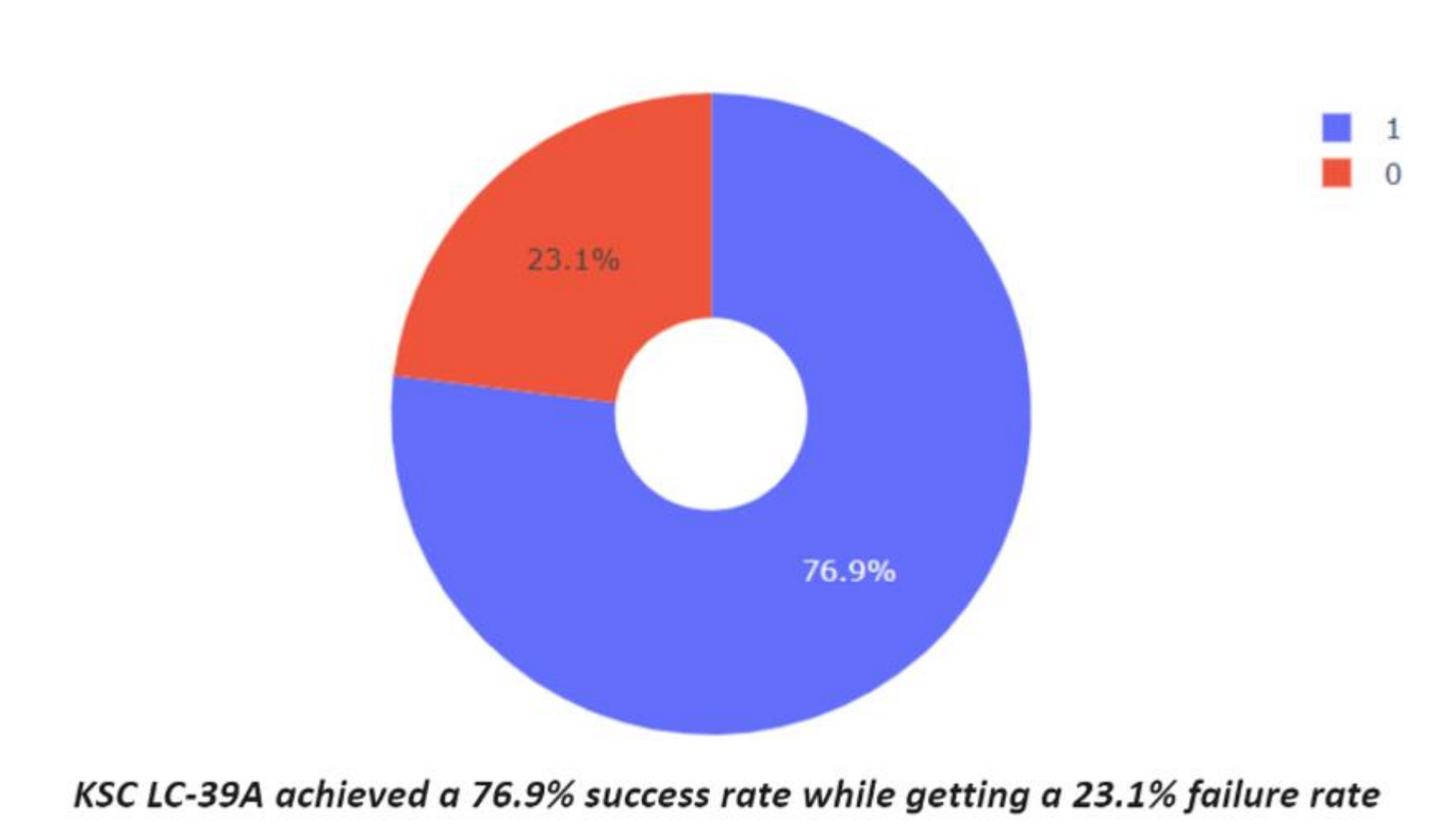



Section 4
Build a Dashboard with Plotly Dash

Pie chart of success

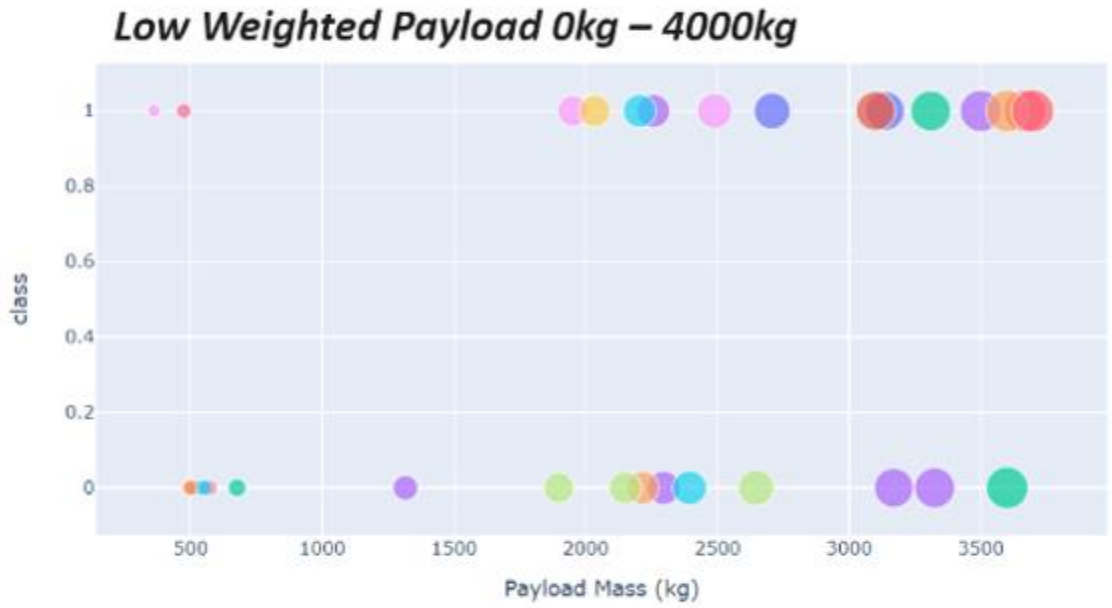


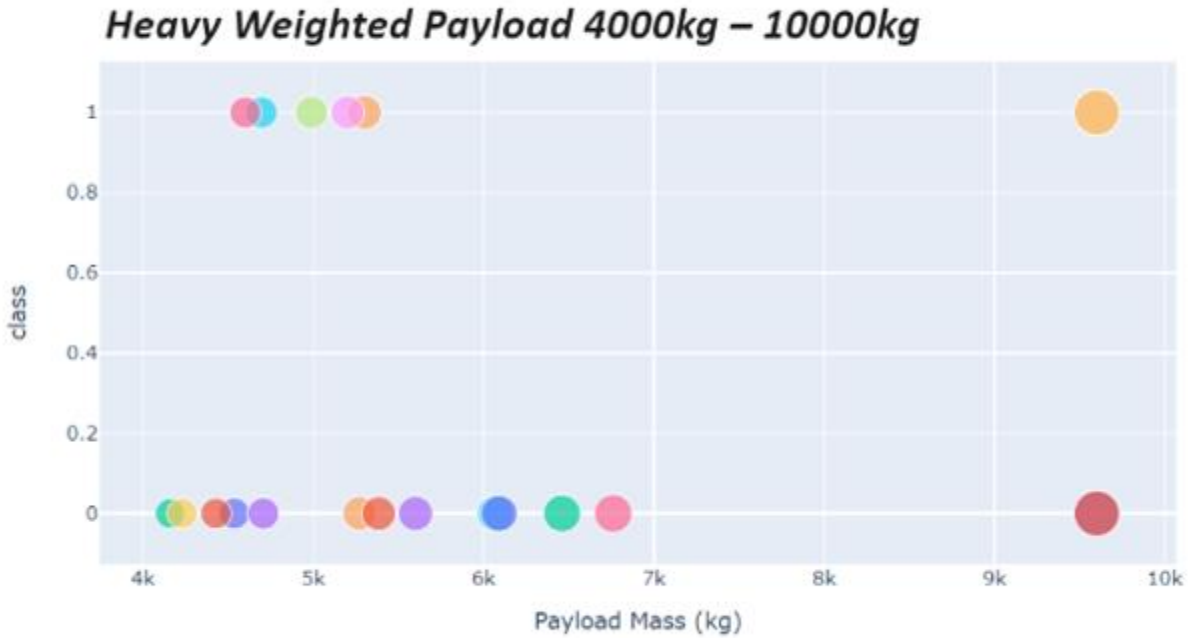
Launch site with highest launch success ratio



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs launch outcome for all sites





Section 5

Predictive Analysis (Classification)

Classification Accuracy

TASK 12 ¶

Find the method performs best:

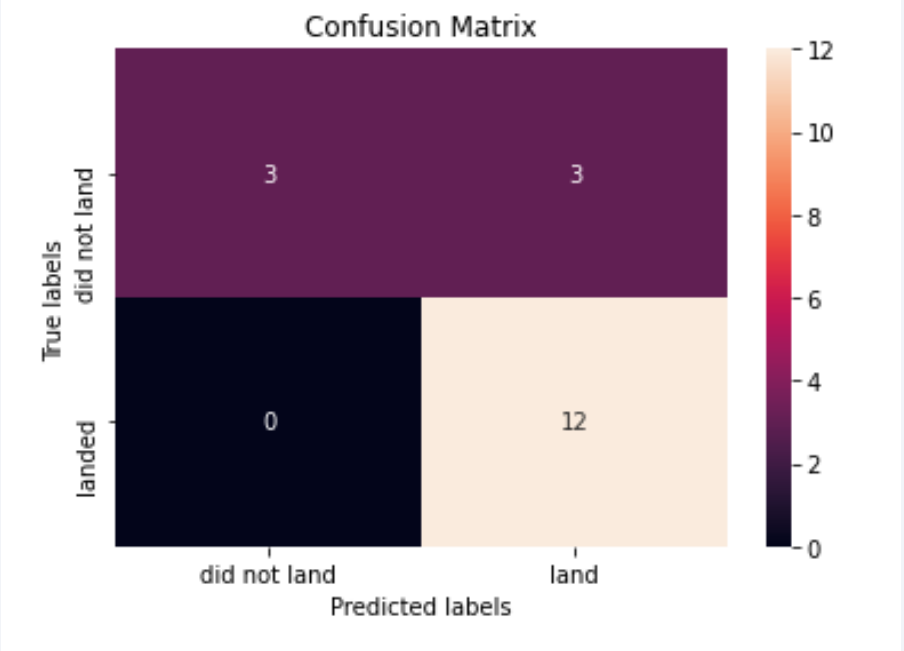
In [31]:

```
# Calcular la precisión en el conjunto de datos de prueba para cada modelo
accuracy_logreg = logreg_cv.score(X_test, Y_test)
accuracy_svm = svm_cv.score(X_test, Y_test)
accuracy_tree = tree_cv.score(X_test, Y_test)
accuracy_knn = knn_cv.score(X_test, Y_test)

# Imprimir las precisiones
print("Accuracy of Logistic Regression: ", accuracy_logreg)
print("Accuracy of Support Vector Machine: ", accuracy_svm)
print("Accuracy of Decision Tree: ", accuracy_tree)
print("Accuracy of K-Nearest Neighbors: ", accuracy_knn)

# Determinar el modelo con la mejor precisión
best_accuracy = max(accuracy_logreg, accuracy_svm, accuracy_tree, accuracy_knn)
print("Best performing model has an accuracy of: ", best_accuracy)

# Identificar cuál modelo es
if best_accuracy == accuracy_logreg:
    print("Best performing model: Logistic Regression")
elif best_accuracy == accuracy_svm:
    print("Best performing model: Support Vector Machine")
elif best_accuracy == accuracy_tree:
    print("Best performing model: Decision Tree")
elif best_accuracy == accuracy_knn:
    print("Best performing model: K-Nearest Neighbors")
```

```
Accuracy of Logistic Regression:  0.8333333333333334
Accuracy of Support Vector Machine:  0.8333333333333334
Accuracy of Decision Tree:  0.8333333333333334
Accuracy of K-Nearest Neighbors:  0.8333333333333334
Best performing model has an accuracy of:  0.8333333333333334
Best performing model: Logistic Regression
```

Confusion Matrix

- The confusion matrix generated for the decision tree classifier illustrates its capability to differentiate between various classes. However, a notable issue is the occurrence of false positives, wherein unsuccessful landings are incorrectly identified as successful by the classifier.



Conclusions

- In summary, it can be inferred that:

- A higher frequency of flights at a given launch site correlates with an elevated success rate for launches from that site.
- There has been a steady rise in launch success rates from 2013 through 2020.
- Orbits designated as ES-L1, GEO, HEO, SSO, and VLEO demonstrate the highest rates of successful launches.
- The launch site KSC LC-39A boasts the highest number of successful launches compared to other sites.
- For the task at hand, the Decision Tree Classifier emerges as the most effective machine learning model.

Thank you!