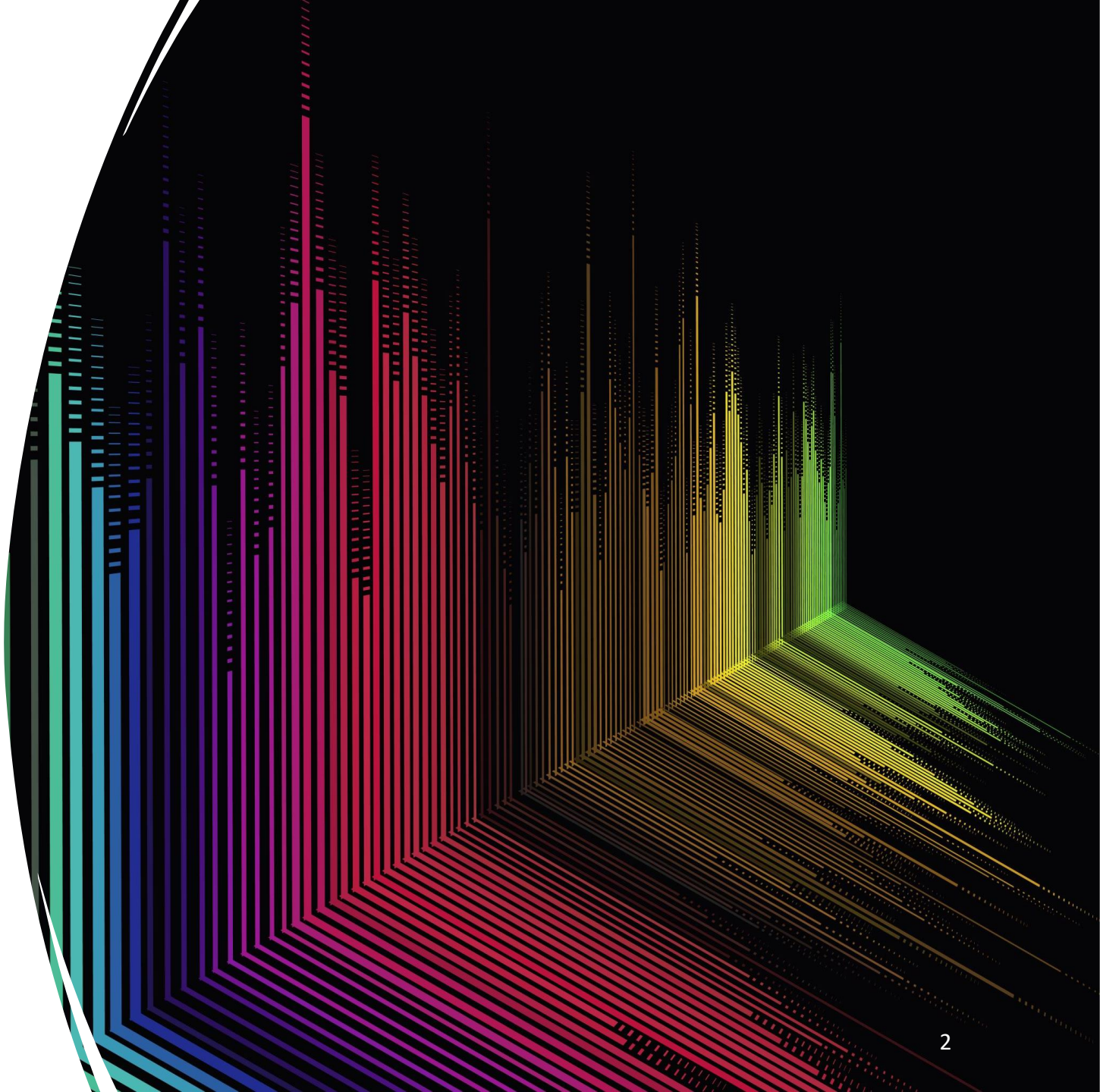
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Daniel Daileg
08 August, 2023

Outline

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



Executive Summary

Summary of methodologies

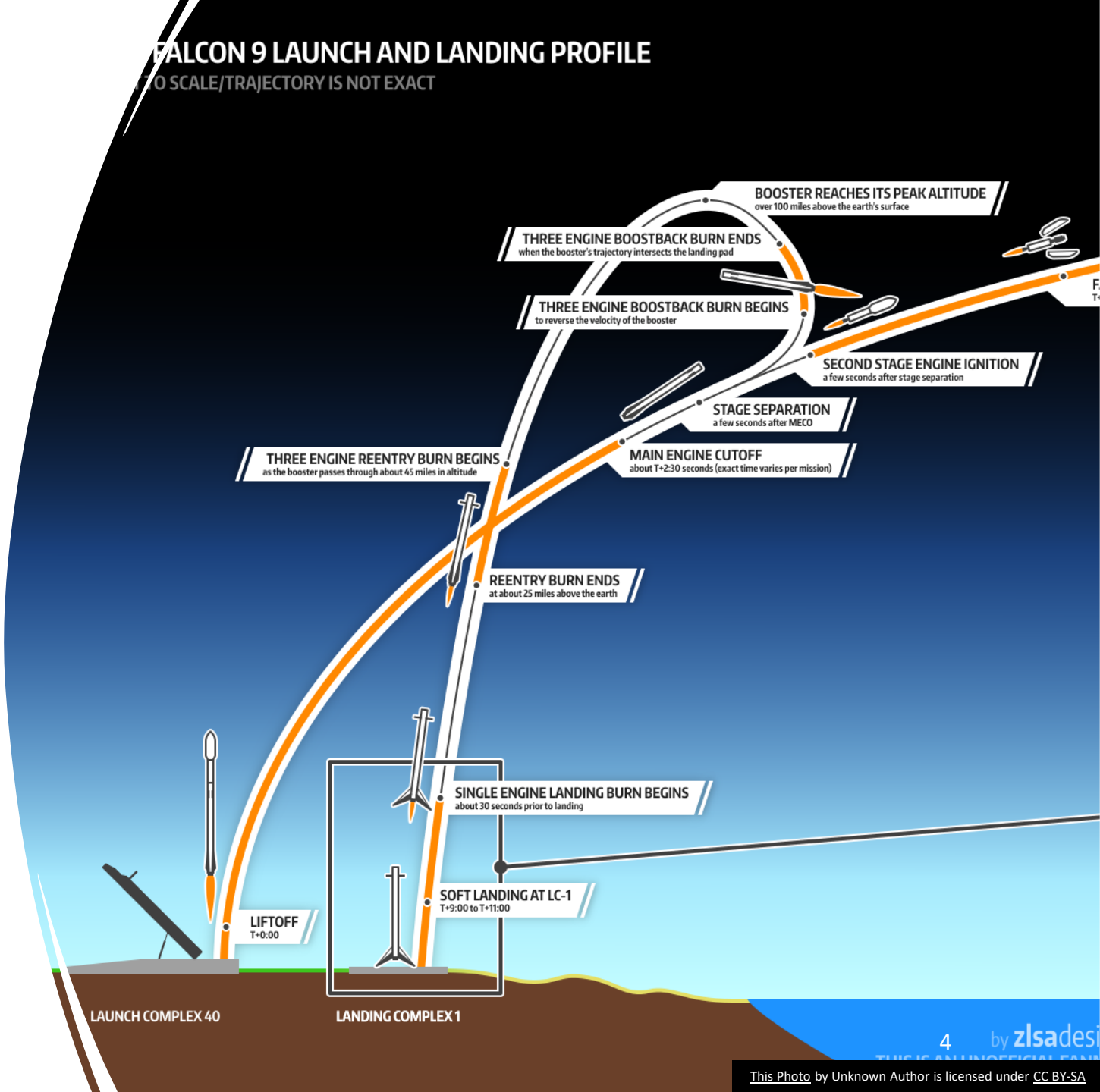
- Data Collection through REST API and Webscraping
- Data Wrangling
- Exploratory Data Analysis through SQL and data visualization
- Interactive Map with Folium
- Interactive Dashboard with DASH
- Predictive Analysis

Summary of all results

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

Introduction

- Rocket Launches are expensive, costing upward of 165 million dollars per launch. However, Space X advertises rocket launches with costs that are more than half of the amount of their competitors with operating costs being advertised at 62 million dollars. This is because their Falcon 9 rockets have a reusable first stage, which effectively cuts costs.
- Problems you want to find answers:
  - What factors influence successful landings?
  - Are there any relationships between the variables that contribute to successes?
  - How does SpaceX ensure the best rocket landing success rate?



Section 1

Methodology

Methodology

- **Data collection methodology:**
  - Data was collected using the SpaceX Rest API in addition to Web Scrapping
- **Perform data wrangling**
  - Data was cleaned by replacing missing values, dropping irrelevant columns and creating and outcome label
- **Perform exploratory data analysis (EDA) using visualization and SQL**
  - Data was plotted using scatter plots, bar graphs, and line graphs
- **Perform interactive visual analytics using Folium and Plotly Dash**
  - Location Data was used to locate the launch sites and launch successes on a world map
  - Launch Data was used to create an interactive dashboard showing launch successes and the relationship between payload mass and launch success
- **Perform predictive analysis using classification models**
  - Multiple classification models were built and tuned and were compared for accuracy

Data Collection – SpaceX API

[Github URL of Completed Notebook]

Request rocket launch data from SpaceX REST API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

Decode response content as a JSON and turn it into a Pandas Dataframe

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

Take a subset of the data and apply custom functions to gain information about the launches from IDs given for each launch

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
```

```
# Call getBoosterVersion
getBoosterVersion(data)
```

```
# Call getLaunchSite
getLaunchSite(data)
```

```
# Call getPayloadData
getPayloadData(data)
```

```
# Call getCoreData
getCoreData(data)
```

Construct final dataset and filter the data to keep only the relevant features (Falcon 9 launches) for our analysis

```
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}
```

```
# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=df[df['BoosterVersion'] != 'Falcon 1']
data_falcon9.head()
```

Export Data to CSV

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection – Web Scraping

[Github URL of Completed Notebook]

Use HTTP GET method to request the Falcon9 launch data from Wikipedia

```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data= requests.get(static_url).text
```

Create a BeautifulSoup object from the HTML response

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup=BeautifulSoup(html_data,'html.parser')
```

Collect relevant column names from the HTML table header

```
column_names = []
# 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
columns = first_launch_table.find_all('th')
for column in columns:
    col = extract_column_from_header(column)
    if (col is not None and len(col) > 0):
        column_names.append(col)
```

Create a dictionary from the column names and parse the HTML tables

```
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']=[]
```

Convert dictionary into a Pandas dataframe

```
df=pd.DataFrame(launch_dict)
```

Export dataframe to a CSV

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

[Github URL of Completed Notebook]

Identify missing values and replace them with the mean of the values

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

Calculate the number of launches on each site

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

Calculate the number and occurrence of each orbit

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

Calculate the number and occurrence of mission outcome per orbit type

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
```

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

Create a landing outcome label from Outcome column to determine the label for training supervised models

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for i in df['Outcome']:
    if i in bad_outcomes:
        landing_class.append(1)
    else:
        landing_class.append(0)
print(landing_class[0:5])
```

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

Output to CSV

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

EDA with Data Visualization

Github URL of Completed Notebook

To determine the relationship between two given variables, scatter/categorical plots were used:

Flight Number vs. Payload Mass

Flight Number vs. Launch Site

Payload vs. Launch Site

Flight Number vs. Orbit Type

Payload vs. Orbit Type

A Bar Chart was used to visualize the relationship between success rate and orbit type

A Line Chart was used to visualize the trend of launch successes between the years 2010-2020

EDA with SQL

Github URL of Completed Notebook

- Display the names of the unique launch sites in the space mission: **select distinct("Launch_Site") from spacextable**
- Display 5 records where launch sites begin with the string 'CCA': **select * from spacextable where "Launch_Site" like 'CCA%' limit 5**
- Display the total payload mass carried by boosters launched by NASA: **select sum(payload_mass__kg_) as "Total Payload Mass (kg)" from spacextable where customer = "NASA (CRS)"**
- Display average payload mass carried by booster version F9 v1.1: **select avg(payload_mass__kg_) from spacextable where "Booster_Version" like "F9 v1.1"**
- List the date when the first successful landing outcome in ground pad was achieved: **select min(date) as 'Earliest date of successful ground pad outcome' from spacextable where "Mission_Outcome" = "Success" and "Landing_Outcome" = "Success (ground pad)"**
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000: **select "Booster_Version" from spacextable where "Landing_Outcome" = "Success (drone ship)" and payload_mass__kg_ between 4000 and 6000**
- List the total number of successful and failure mission outcomes: **select "Mission_Outcome" as "Outcome", count(*) as "Total" from spacextable group by "Mission_Outcome"**
- List the names of the booster_versions which have carried the maximum payload mass: **select "Booster_Version" from spacextable where payload_mass__kg_ = (select max(payload_mass__kg_) from spacextable)**
- 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: **select substr(date, 6, 2) as "Month", "Booster_Version", "Launch_Site" from spacextable where "Landing_Outcome" = "Failure (drone ship)" and substr(date,1,4)="2015"**
- Rank the count of landing outcomes (such as Failure(droneship) or Success (ground pad) between the date 2010-06-04 and 2017-03-20, in descending order: **select "Landing_Outcome", count(*) as "Total" from spacextable where date between "2010-06-04" and "2017-03-02" group by "Landing_Outcome" order by "Total" desc**

Build an Interactive Map with Folium

Github URL of Completed Notebook

Circles were created using launch site coordinates to visualize their locations on the Folium map

Within the circles, colored markers were added through a marker cluster to visualize the launch outcomes for each site. Green markers indicated successes while red markers indicated failures.

Lines were created to measure the distance of the launch site from landmarks such as cities, railways, highways, and coastlines.

Launch sites were found to be in close proximity to coastlines while maintaining some distance from major cities, highways, and railways.

- This can be due to the safety risk of the damages that can be caused by a launch failure. The close proximity to the coastline provides an area for the rockets to crash safely while avoiding populated areas.

Build a Dashboard with Plotly Dash

Github URL of Completed Notebook

- A pie chart was added to show the total launches by a specific/all sites
  - The user can filter between sites using a dropdown selector
  - For all sites, the pie chart shows the proportion of the total successful launches for all sites
  - For individual sites, the pie chart shows the proportion of successful vs failed launches per site
- A scatter plot was added to show the relationship between launch outcome and payload mass for the different booster versions
  - The user can filter between launch sites using a dropdown selector
  - The user can select a range of payload mass using an interactive slider bar
  - Booster version categories are separated by color on the scatter plot

Predictive Analysis (Classification)

[Github URL of Completed Notebook]

Load the dataframe

```
URL2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_
resp2 = await fetch(URL2)
text2 = io.BytesIO((await resp2.arrayBuffer()).to_py())
X = pd.read_csv(text2)
```

Create NumPy array from the column “Class” and assign it to variable Y

```
Y = data['Class'].to_numpy()
```

Standardize the data

```
# students get this
transform = preprocessing.StandardScaler()
X = preprocessing.StandardScaler().fit(X).transform(X)
```

Split X and Y data into training and test data

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.2,random_state=2)
```

Create predictive models (LogReg, Tree, SVM, KNN) and use GridsearchCV to find the best parameters

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}
lr=LogisticRegression()
logreg_cv= GridSearchCV(lr,parameters, cv=10)
logreg_cv.fit(X_train,Y_train)
```

Calculate the accuracy score

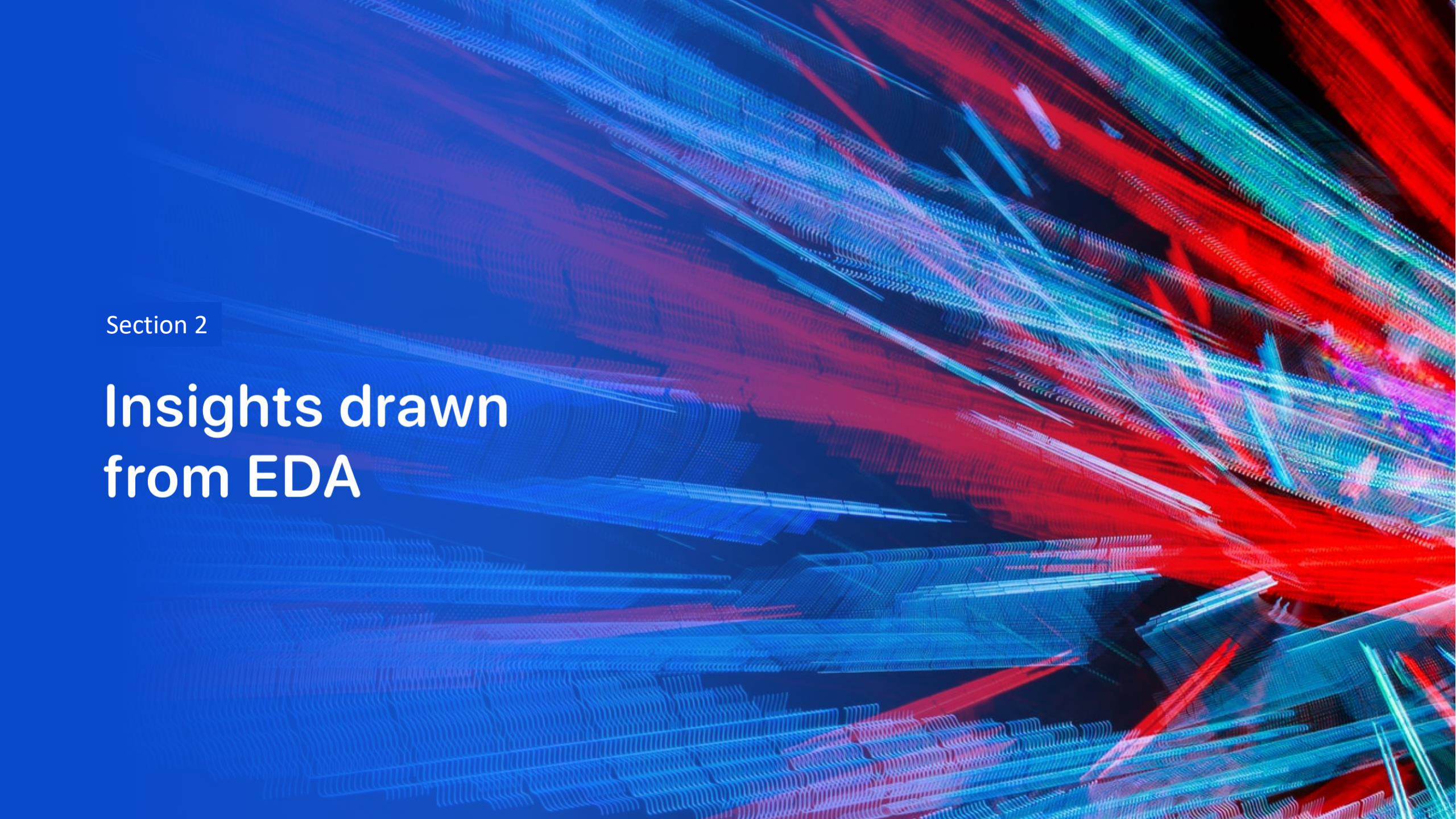
```
print('Logistic Regression test data accuracy:',logreg_cv.score(X_test,Y_test))
```

Display the confusion matrix

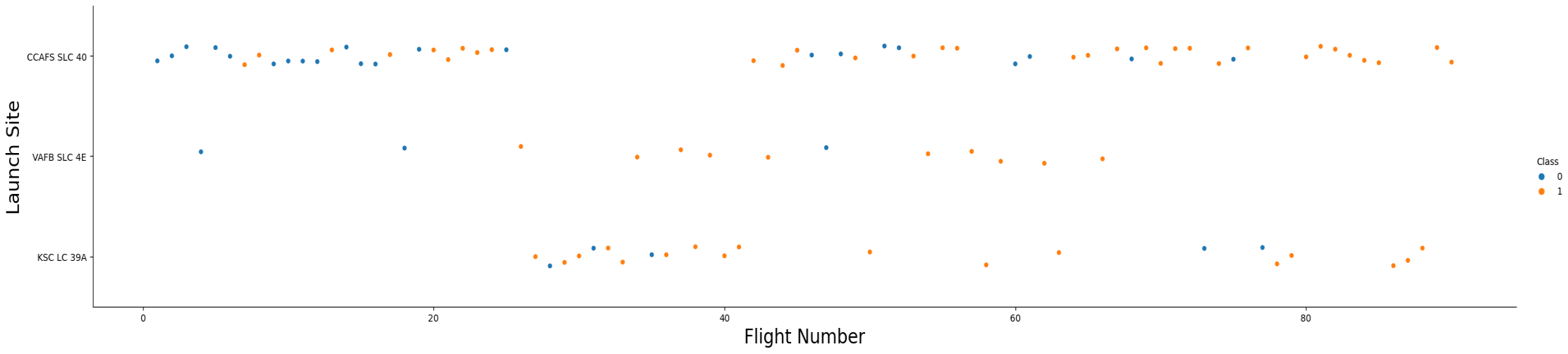
```
yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Results

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

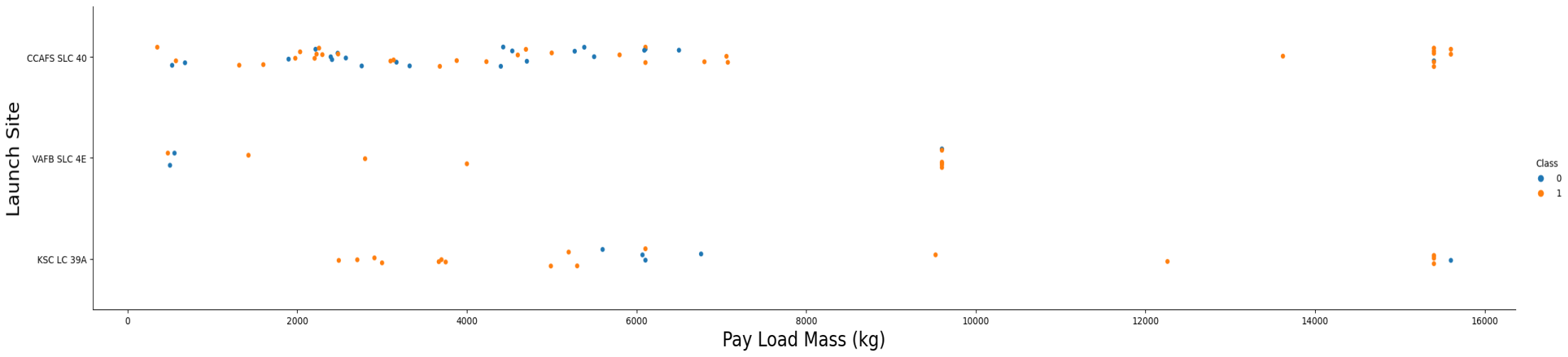

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



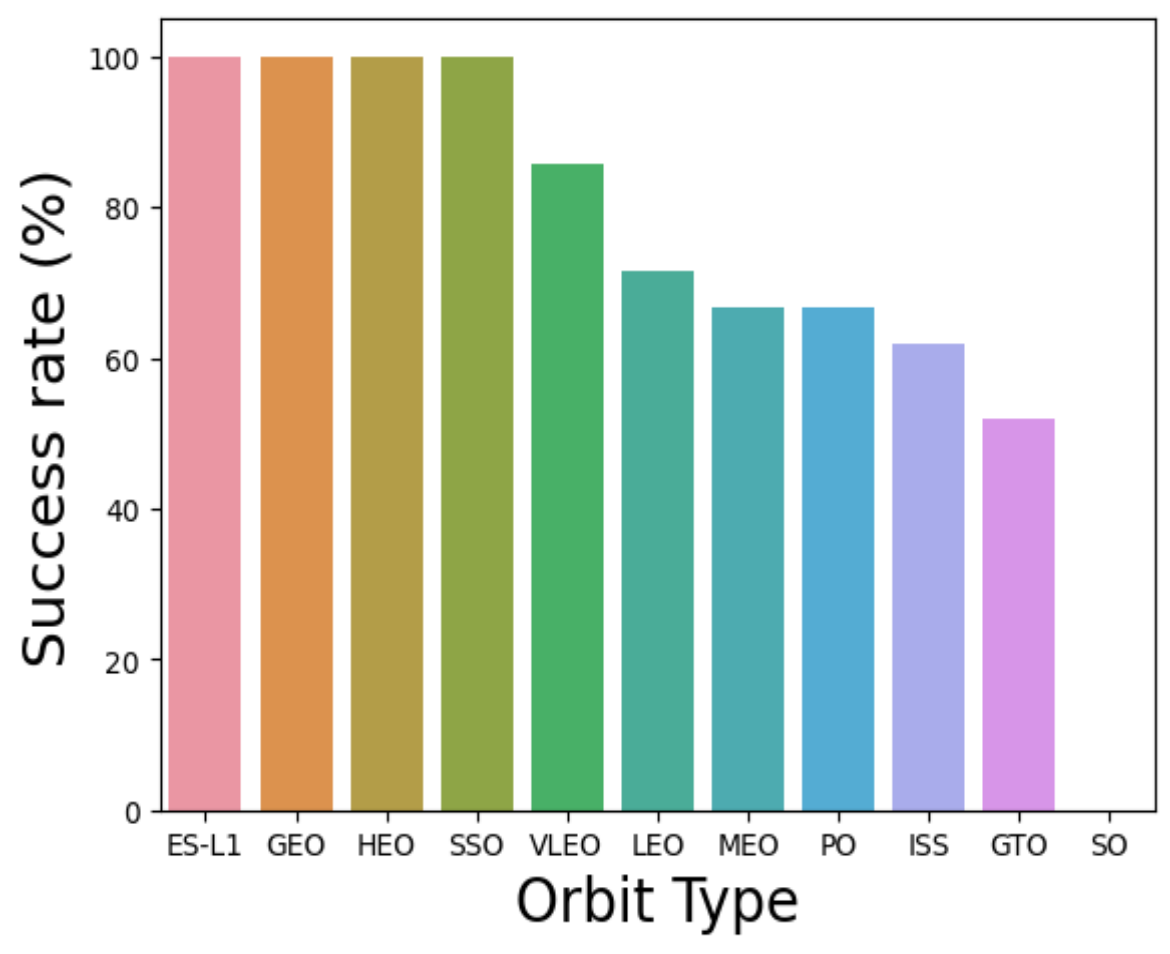
It can be observed that for each site, the number of successful launches increases as the number of launches increases

Payload vs. Launch Site



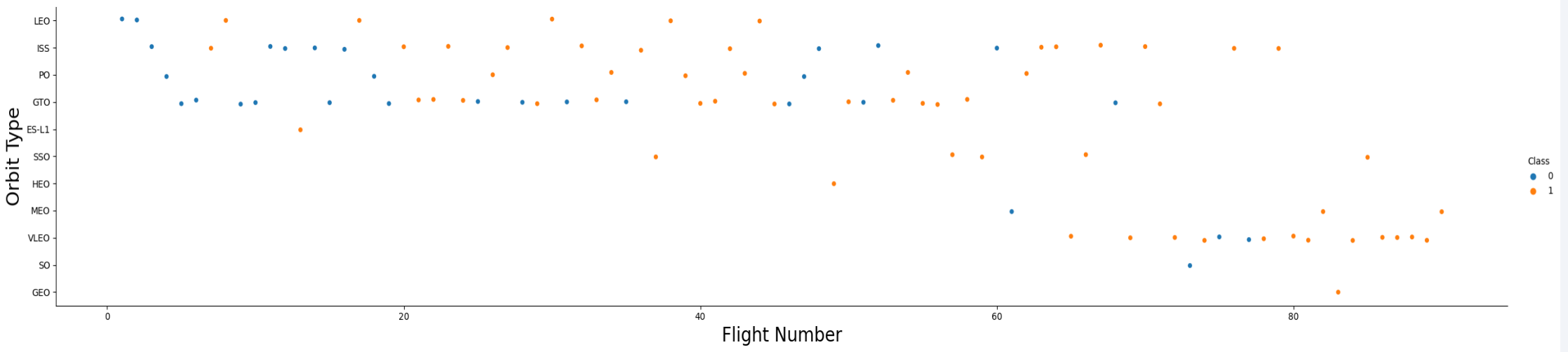
- Site CCAFS SLC 40 has the greatest number of successes at a large payload mass.
- Site VAFB SLC 4E has very few failures, so it's difficult to determine a pattern. However, this site doesn't have launches that have a payload mass greater than 10000kg.
- KSC LC 39A has a greater number of successes at a lower payload mass than at a high payload mass

Success Rate vs. Orbit Type



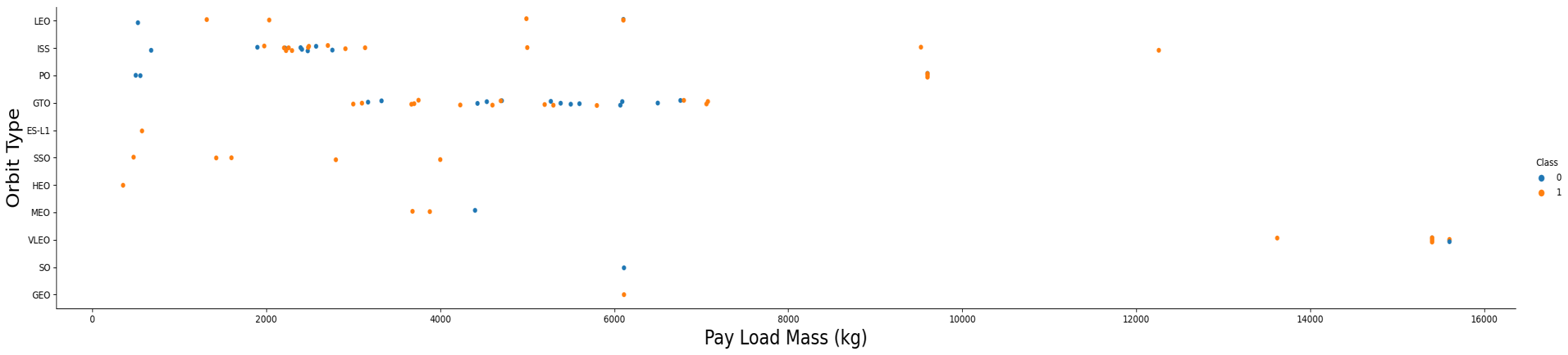
- The ES-L1, GEO, HEO, and SSO all have a success rate of 100%.
- The orbit type with the lowest success rate is GTO

Flight Number vs. Orbit Type



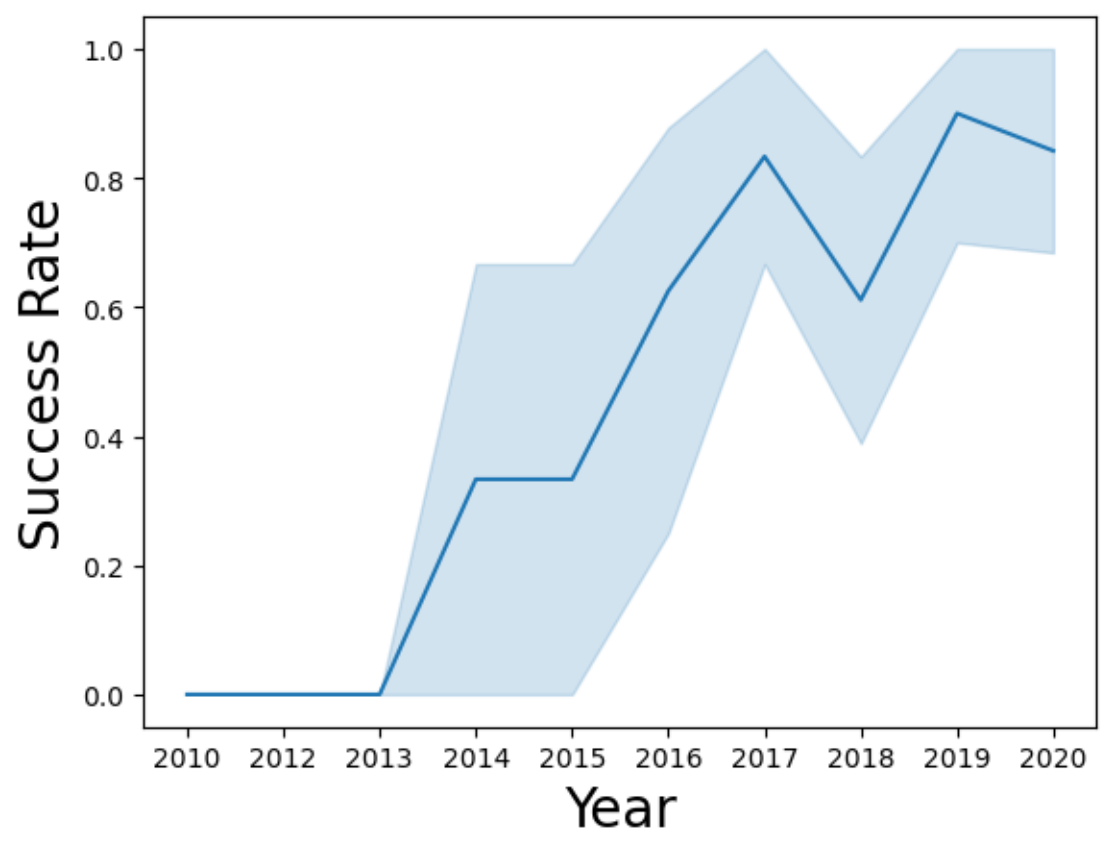
- The LEO and ISS orbit type appear to have more successes with an increasing number of flights
- There seems to be no discernable relationship between flight number and successes for orbit type GTO

Payload vs. Orbit Type



- LEO, ISS, and PO have increased successes as payload mass increases
- For GTO, increasing the payload mass seems to have contribute to the increased failures.
- SSO seems to perform well with lower payload masses
- VLEO has a high success rate under high payload masses

Launch Success Yearly Trend



- The success rate started to increase at year 2013 and has continued to increase as the years went on.
- However, between 2017 and 2018, the success rate experienced a sharp decline, but increased once more the year after.

All Launch Site Names

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

SQL Query

select distinct("Launch_Site") from spacextable

Explanation

Distinct() shows only the unique values of the designated column

Result:

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

Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with `CCA`

SQL Query

- **select * from spacextable where “Launch_Site” like ‘CCA%’ limit 5**

Result:

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

Explanation

The LIKE operator is used in a WHERE clause to search for a specific string pattern in a column. This operator is used in conjunction with the percent sign wildcard

Total Payload Mass

Calculate the total payload carried by boosters from NASA

SQL Query

```
select sum(payload_mass__kg_) as "Total Payload Mass
(kg)" from spacextable where customer="NASA (CRS)"
```

Result:

Total Payload Mass (kg)
45596

Explanation

The SUM function returns the total sum of a numeric column (payload_mass__kg_)

The WHERE clause is used to extract the records that fulfill a specified condition (NASA(CRS))

Average Payload Mass by F9 v1.1

Calculate the average payload mass carried by booster version F9 v1.1

SQL Query

```
select avg(payload_mass__kg_) as "Average Payload
Carried by F9 v1.1" from spacextable where
"Booster_Version" like "F9 v1.1"
```

Result:

Average Payload Carried by F9 v1.1
2928.4

Explanation

The AVG function returns the average value of a numeric column (payload_mass__kg_)

The WHERE clause is used to extract the records that fulfill a specified condition (F9 v1.1)

First Successful Ground Landing Date

Find the dates of the first successful landing outcome on ground pad

SQL Query

```
select min(date) as 'Earliest date of successful ground pad
outcome' from spacextable where "Mission_Outcome" =
"Success" and "Landing_Outcome" = "Success (ground
pad)“
```

Result:

Earliest date of successful ground pad outcome
2015-12-22

Explanation

The MIN function returns the smallest value of the selected column

The WHERE clause is used to extract records that fulfill a specified condition (Success(ground pad))

Successful Drone Ship Landing with Payload between 4000 and 6000

List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

SQL Query

```
select "Booster_Version" from spacextable where
"Landing_Outcome" = "Success (drone ship)" and
payload_mass__kg_ between 4000 and 6000
```

Explanation

The BETWEEN operator selects values within a given range(4000,6000)

Result:

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

Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcomes

SQL Query

```
select "Mission_Outcome" as "Outcome", count(*) as "Total"
from spacextable group by "Mission_Outcome“
```

Explanation

The COUNT function returns the number of rows that match a specified criterion (Outcome)

The GROUP BY statement is used with COUNT to group the result-set by one or more columns (Mission_Outcome)

Result:

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

Boosters Carried Maximum Payload

List the names of the booster which have carried the maximum payload mass

SQL Query

```
select "Booster_Version" from spacextable where
payload_mass__kg_ = (select max(payload_mass__kg_)
from spacextable)
```

Explanation

A subquery is a query within another SQL query and embedded within the WHERE clause. It returns data that will be used in the main query as a condition for the WHERE clause (max(payload_mass__kg_))

Result:

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

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

SQL Query

```
select substr(date, 6, 2) as "Month", "Booster_Version",
"Launch_Site" from spacextable where "Landing_Outcome"
= "Failure (drone ship)" and substr(date,1,4)="2015“
```

Explanation

The SUBSTR() function extracts a substring from a string (the month and year were extracted from the date)

Result:

Month	Booster_Version	Launch_Site
10	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

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

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 Query

```
select "Landing_Outcome", count(*) as "Total" from spacextable where date between "2010-06-04" and "2017-03-02" group by "Landing_Outcome" order by "Total" desc
```

Explanation

The ORDER BY keyword is used to sort the result-set in ascending or descending order. DESC is used to sort in descending order.

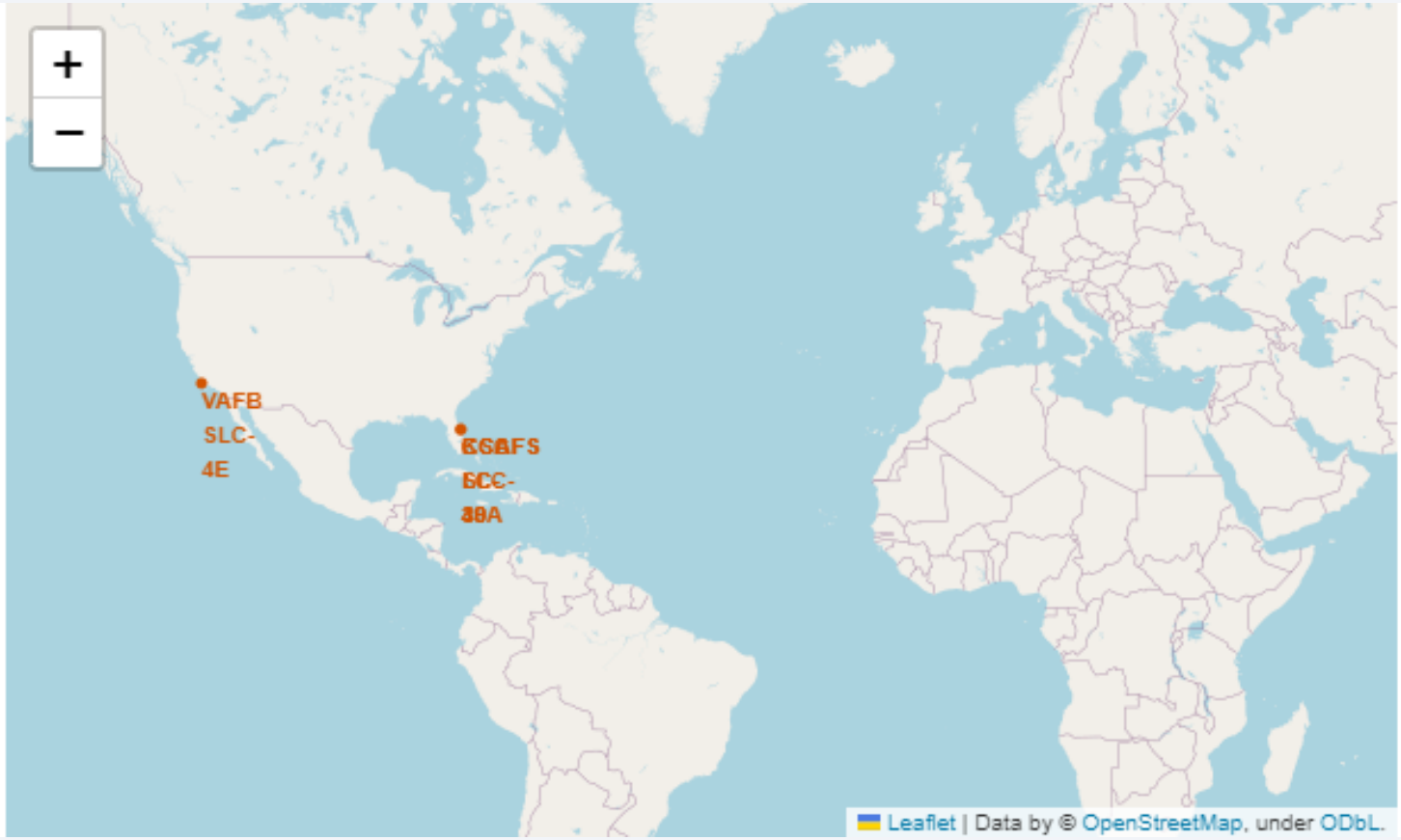
Result:

Landing_Outcome	Total
No attempt	9
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	4
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

Section 3

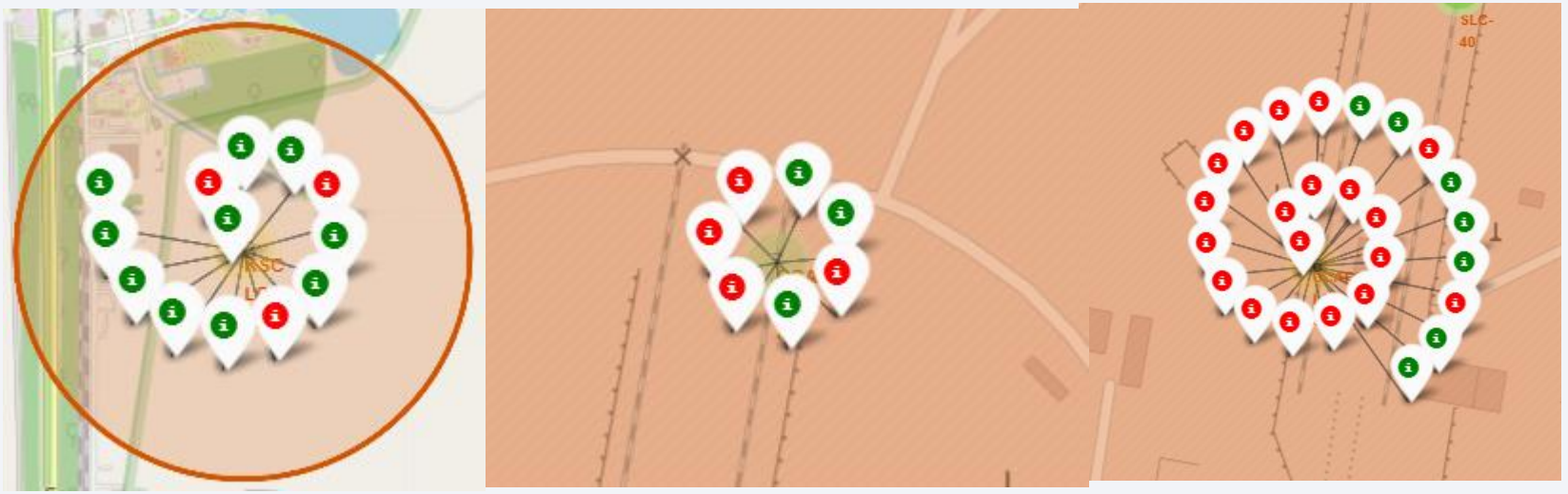
Launch Sites Proximities Analysis

Launch Sites' Location Markers on Global Map

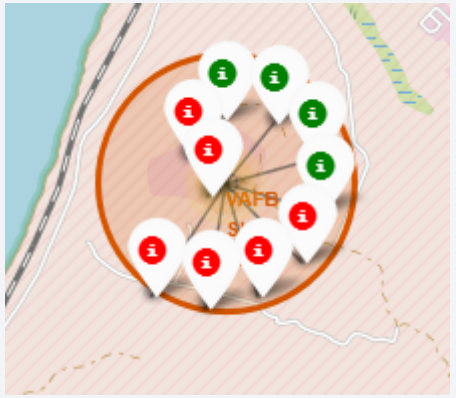


The SpaceX Launch Sites are found in the coastal areas of the United States

Color Labeled Launch Outcomes



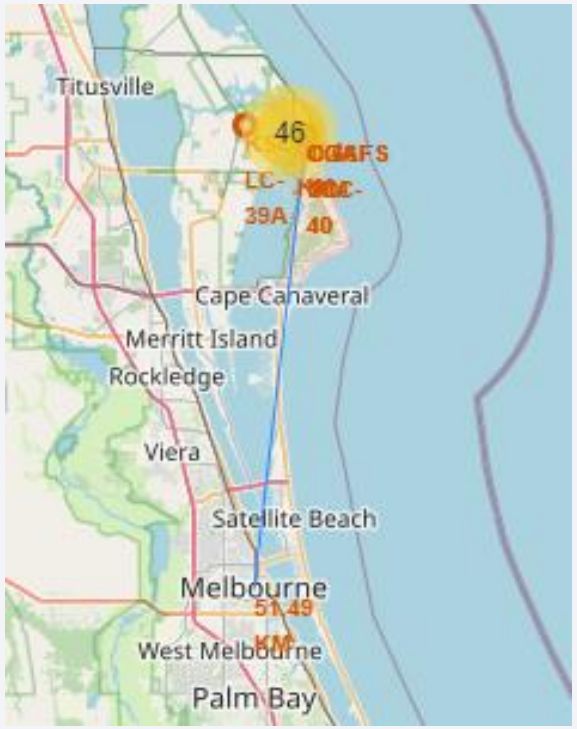
Florida Launch Sites



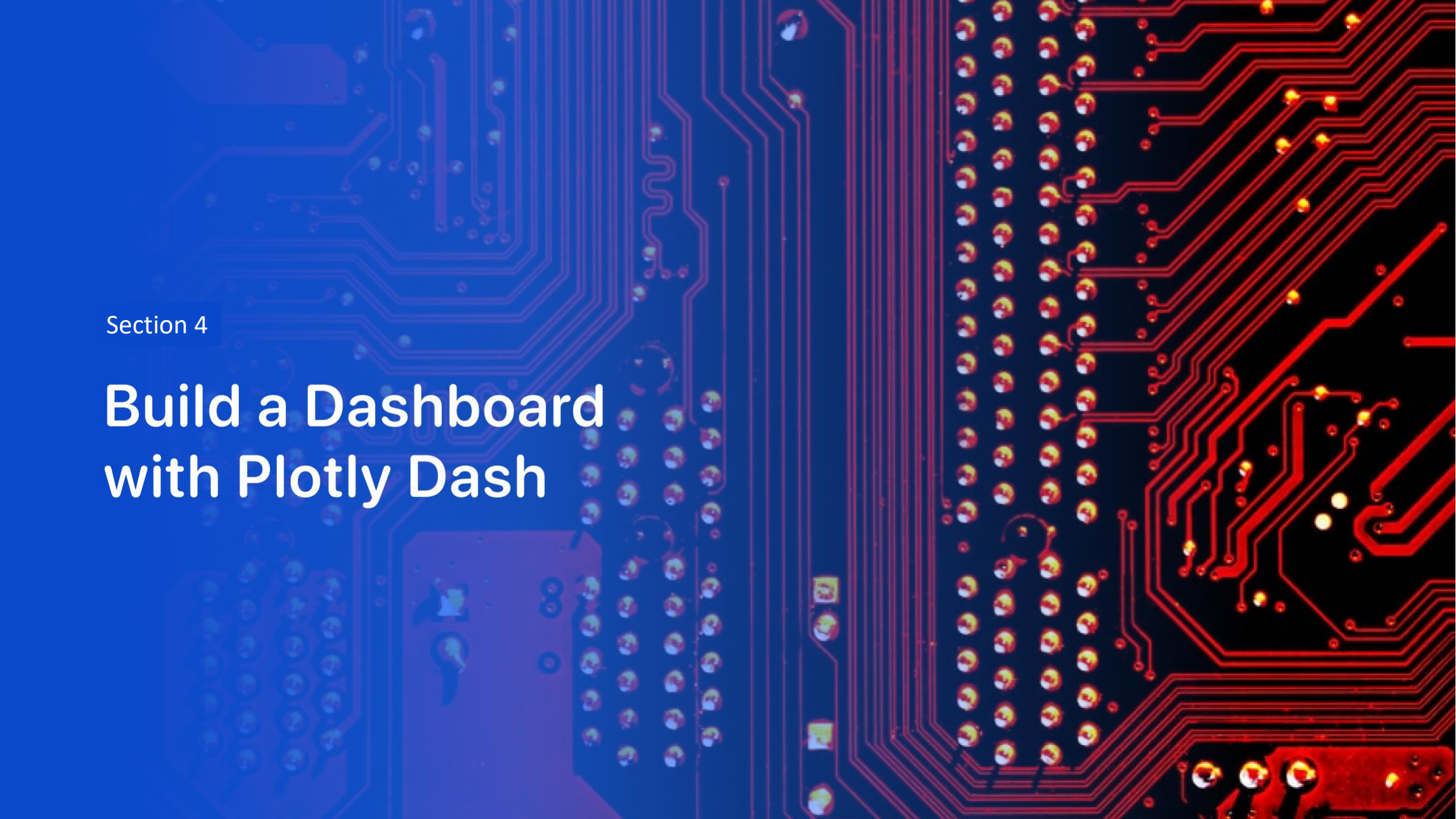
California Launch Site

The Green markers indicate Success while the red markers indicate Failure

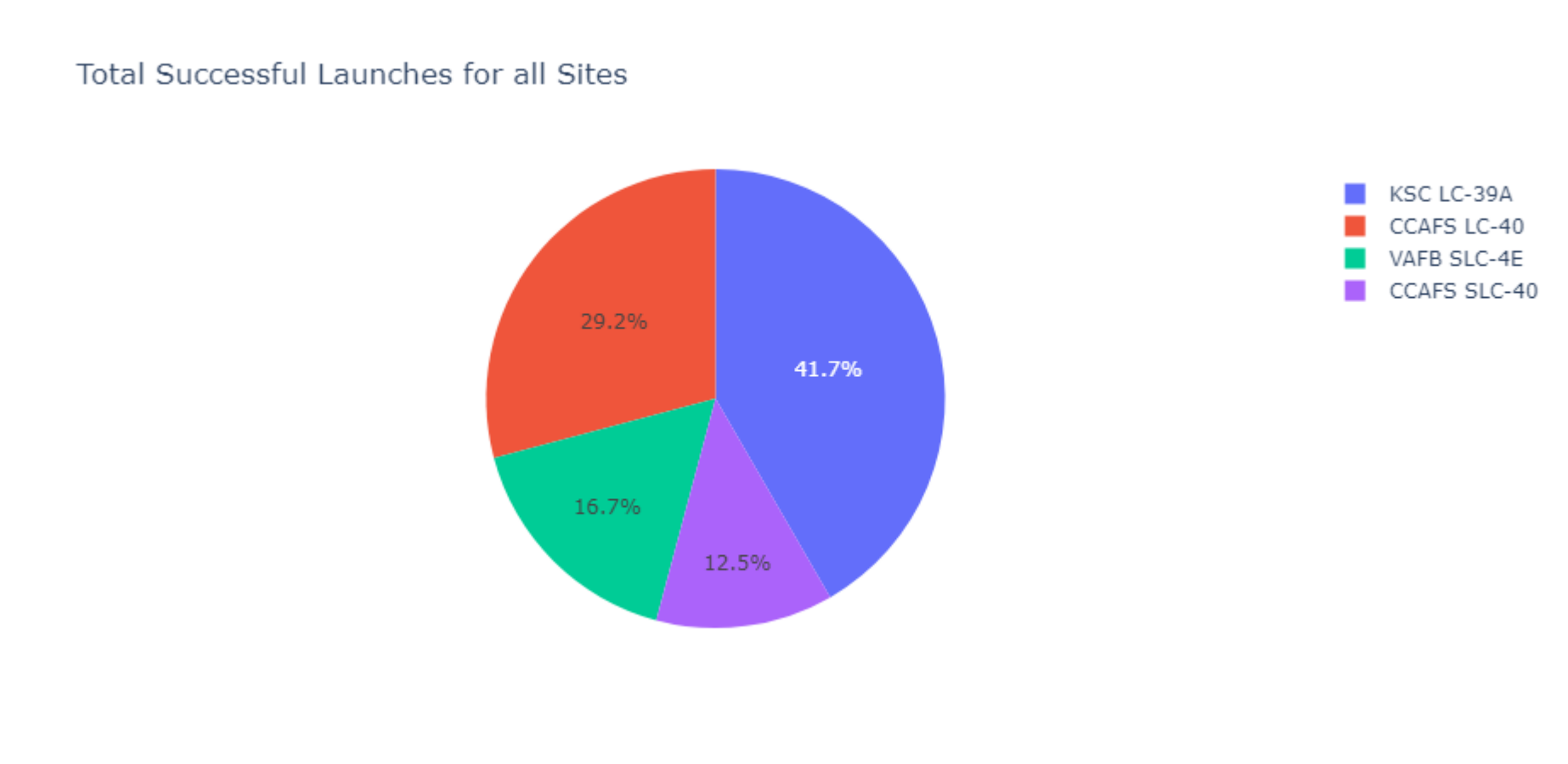
Proximity of Launch Site to Landmarks



The Launch Sites in Florida maintains a great distance(>52km) away from the major city while being close to the coastline (<1km) and non-major transportation routes

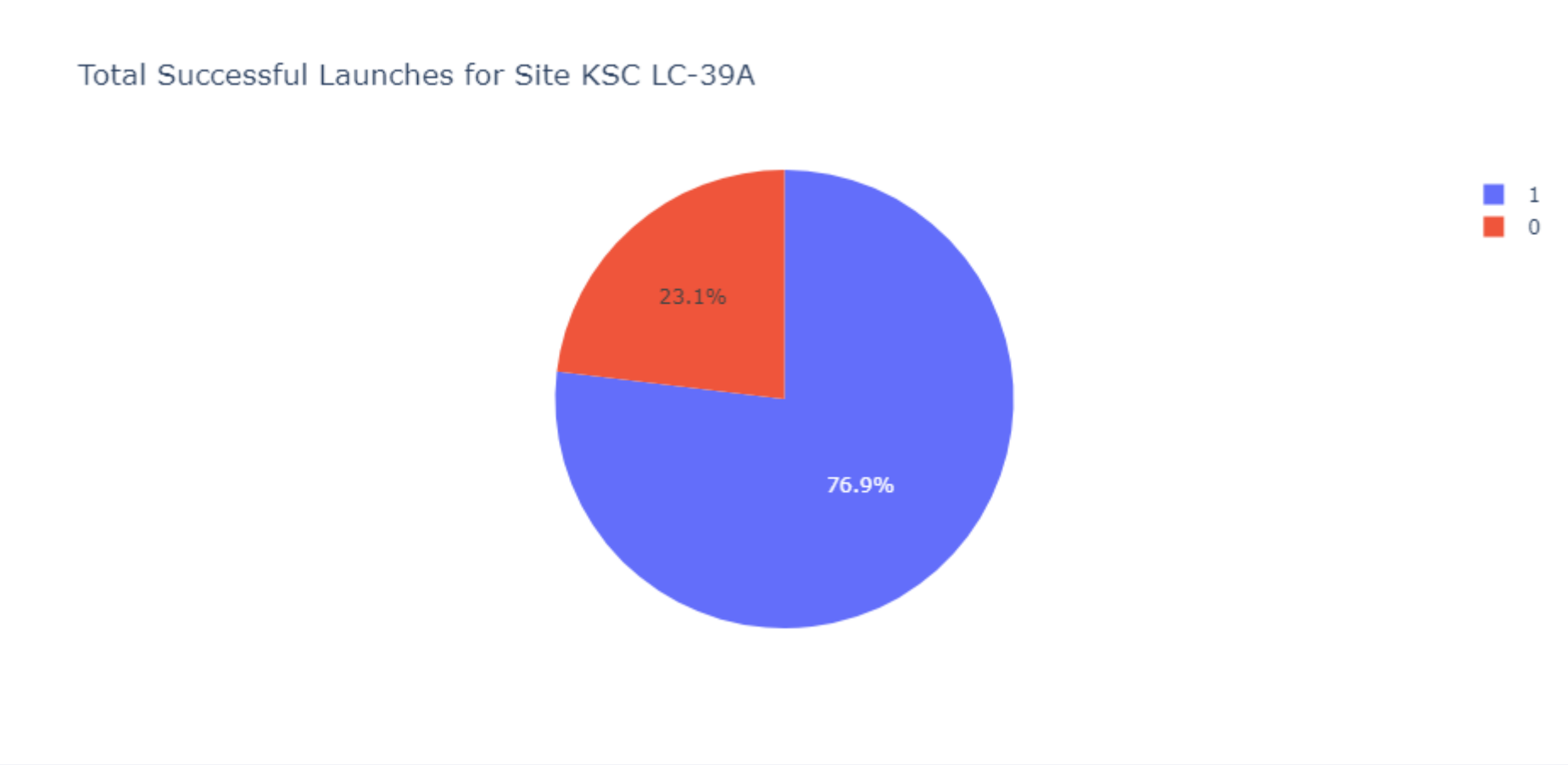

Section 4
Build a Dashboard with Plotly Dash

Piechart for Launch Success Count for All Sites



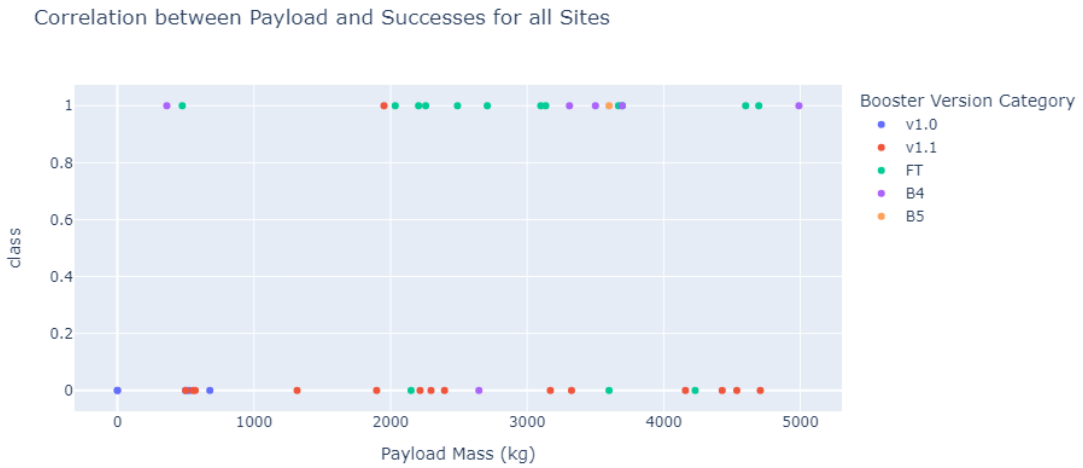
From this pie chart, we can see that KSC LC-39A had the highest number of successes out of all the launch sites

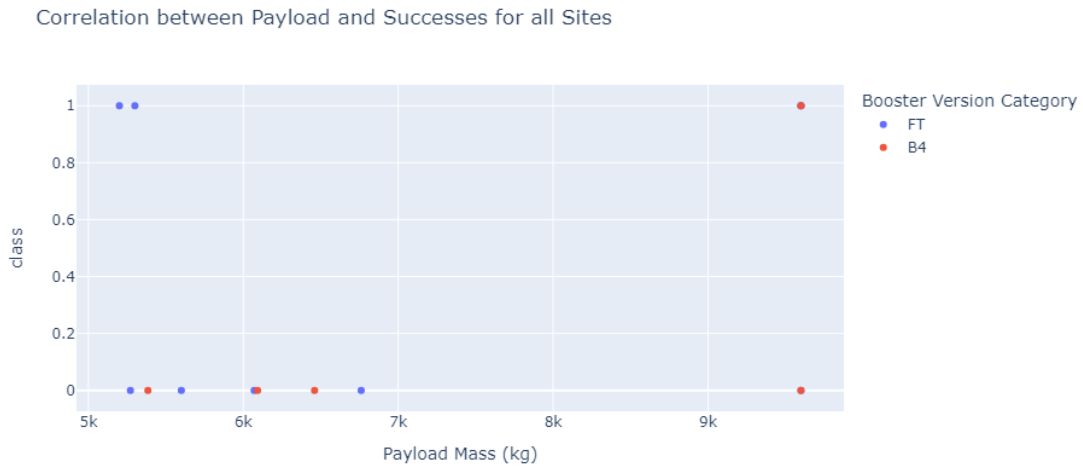
<Dashboard Screenshot 2>



Site KSC LC-39A had a success rate of 76.9% and a failure rate of 23.1%

Payload vs. Launch Outcome for all Sites





Booster version FT had the highest number of successes for low payload mass, while booster v1.1 had the highest number of failures for low payload mass. Booster FT also had the highest successes for high payload mass

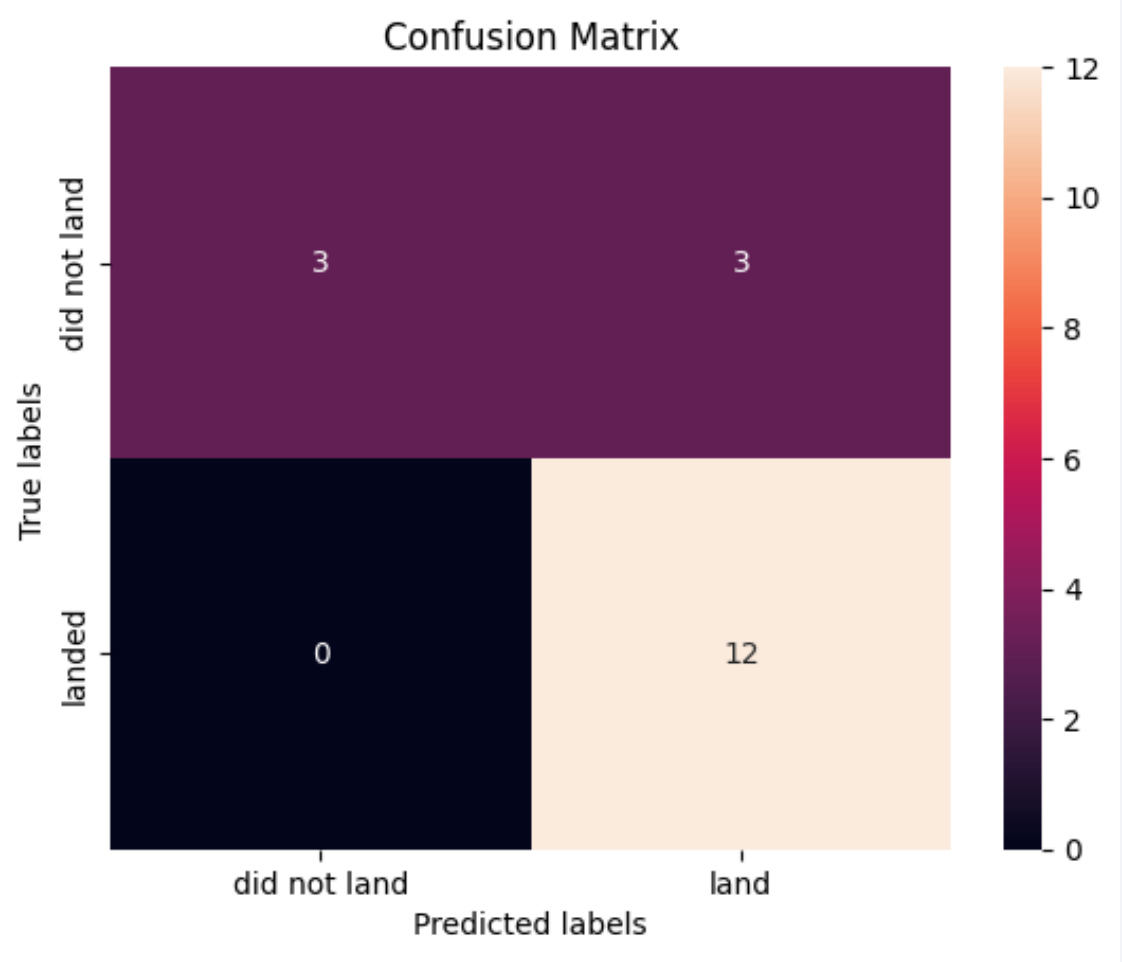
Section 5

Predictive Analysis (Classification)

Classification Accuracy

All four models outputted similar accuracy scores of 0.8333333333333334

Confusion Matrix



All predictive models output the same confusion matrix. This shows that all models can distinguish between the two classes. However, as we can see, there is an error with the model giving false positive results.

Conclusions

- All predictive models have the same accuracy score of 83%
- KSC LC-39A had the highest number of successes out of all the launch sites
- KSC LC 39A has a greater number of successes at a lower payload mass than at a high payload mass
- The ES-L1, GEO, HEO, and SSO orbits all have a success rate of 100%.
- The success rate of the launches increase as the number of launches increase over time
- Booster version FT had the highest number of successful launches

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!