IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Anna Scherbakova
Apr 7, 2024

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data collection via API and Webscraping
  - Data wrangling
  - Exploratory Data Analysis (EDA) using visualization and SQL
  - Interactive visual analytics and Dashboards
  - Predictive analysis using classification models

- **Summary of all results**
  - Exploratory data analysis result
  - Interactive analytics – maps and dashboard
  - Predictive analytics result

Introduction

Companies such as SpaceX have revolutionized the commercial space industry. SpaceX's success stems from its capability to provide cost-effective rocket launches, primarily by reusing the first stage of its Falcon 9 rockets. This reuse drastically cuts launch costs, enhancing affordability and competitiveness in space access.

Space Y is a new rocket company aspiring to compete with SpaceX. The primary goal of the project is to determine the pricing strategy for Space Y's rocket launches. This involves gathering information about SpaceX's pricing model and analyzing data to understand the factors influencing launch costs. Additionally, the task is to predict whether SpaceX will reuse the first stage of its Falcon 9 rockets for each launch. This prediction is crucial for estimating the cost of launches accurately, as the reuse of the first stage significantly impacts pricing.

Key questions to be answered are:

- What are the primary indicators of a successful or unsuccessful landing?
- How do different rocket variables impact the likelihood of landing success or failure?
- What conditions are necessary for SpaceX to attain the highest landing success rate?

Section 1

Methodology

Methodology

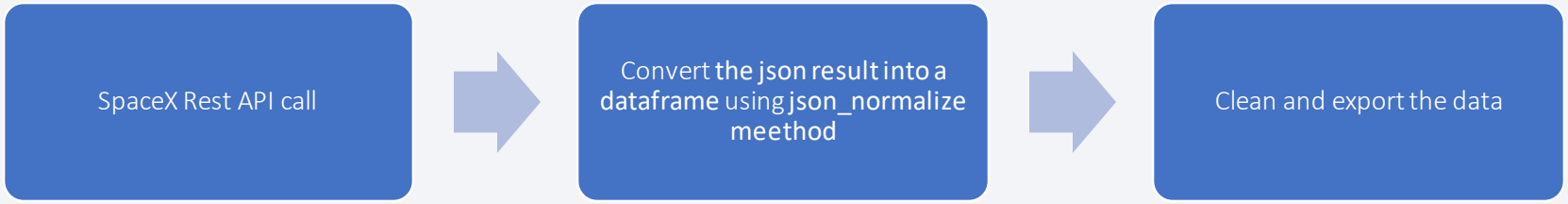
Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and web scrapping from Wikipedia
- Perform data wrangling
  - One-hot encoding is used for categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

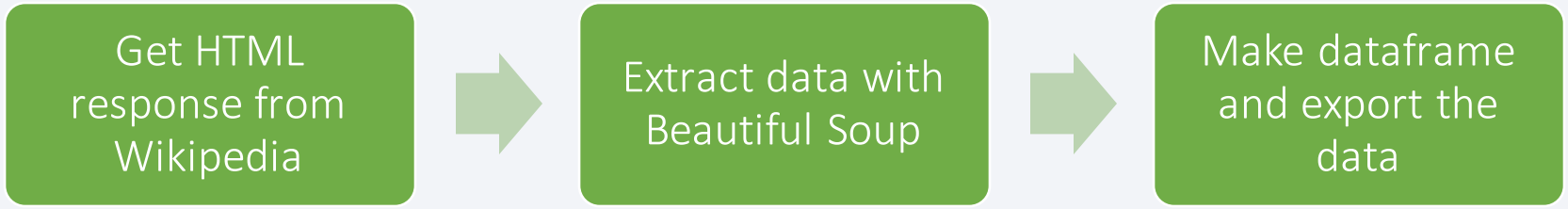
Data Collection

The dataset was collected using SpaceX REST API and Web Scrapping from Wikipedia

- REST API process:



- Web Scrapping process:



Data Collection – SpaceX API

Step 1: Get the response from SpaceX API

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

response = requests.get(spacex_url)
```

Step 2: Decode the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize()

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

Step 3: Clean the data

```
# 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']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that ha
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Step 4: Create dictionary

```
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}
```

Step 5: Create dataframe

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

Step 6: Filter the dataframe to only include Falcon 9 launches

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=df[df['BoosterVersion']!='Falcon 1']
```

[Github URL]

Data Collection – Scraping

Step 1: Request the Falcon9 Launch HTML page, as an HTTP response

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

Step 2: Create a BeautifulSoup object from the HTML response

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

Step 3: Find all tables on the wiki page

```
html_tables=soup.find_all('table')
```

Step 4: Extract column names

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

Step 5: Create dictionary

```
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
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']=[]
```

Step 6: Fill up the dictionary

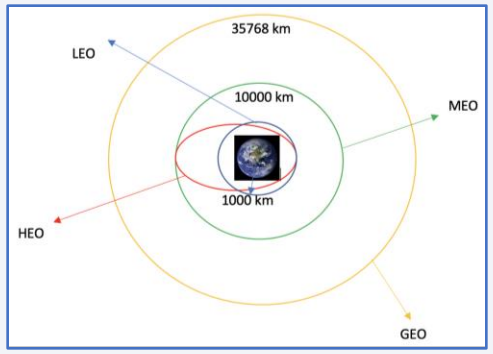
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            #print(flight_number)
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])
------------------
            # Date value
            # TODO: Append the date into launch_dict with key `Date`
            date = datatimelist[0].strip(',')
```

Step 7: Create dataframe

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df.to_csv('spacex_web_scraped.csv', index=False)
```

[Github URL]

Data Wrangling



Data wrangling is the process of converting raw data into a usable form

In the data set, there are several different cases where the booster did not land successfully. We will mainly convert different launches outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful

Step 1: Calculate the number of launches on each site

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

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

Step 2: Calculate the number and occurrence of each orbit

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

GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
```

Step 3: Calculate the number and occurrence of mission outcome of the orbits

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

True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

Step 4: Create a landing outcome label from Outcome column

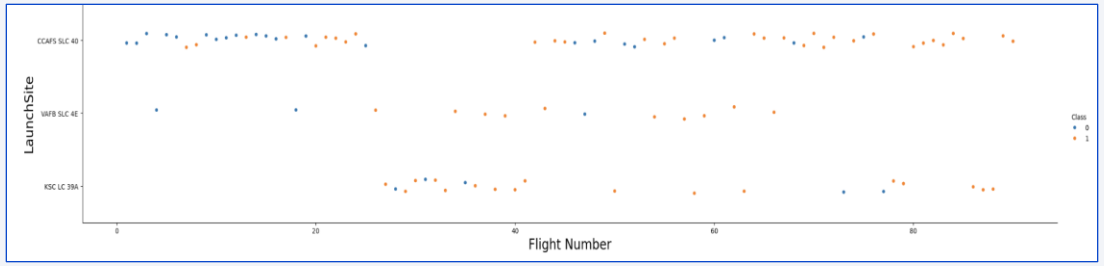
```
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

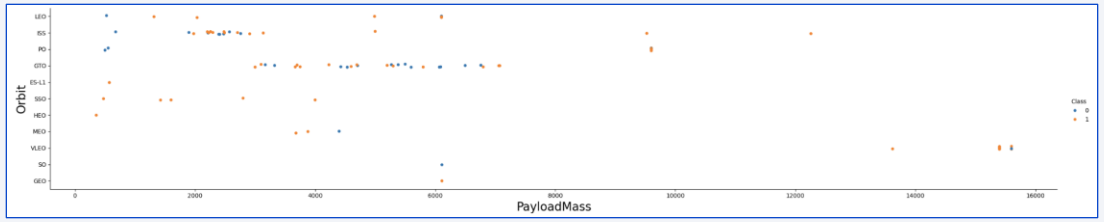
```
df['Class']=landing_class
```

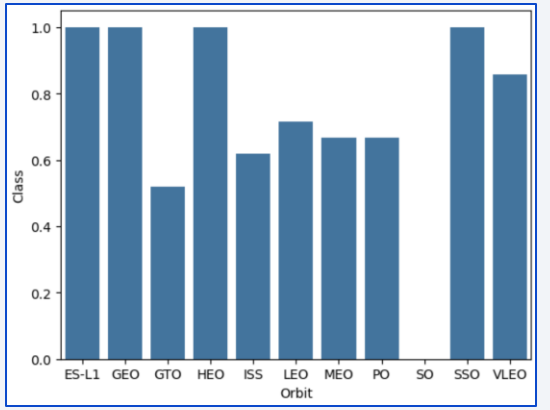
Github URL

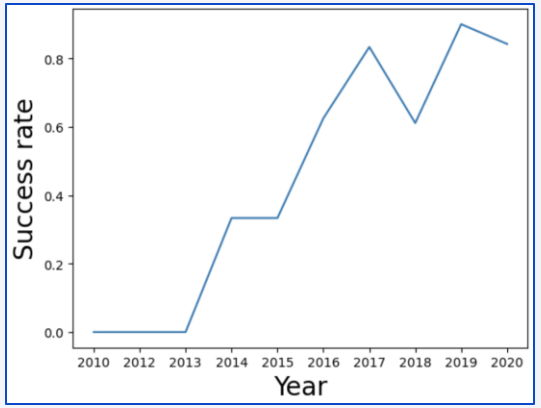
EDA with Data Visualization

- First, we plot **Scatter charts** to observe how the launch outcome is affected by:
  - Flight Number and Pay Load Mass
  - Launch Site and Flight Number
  - Launch Site and Pay Load Mass
  - Flight Number and Orbit type
  - Pay Load Mass and Orbit type
- Then we use **Bar chart** to visually check if there are any relationship between success rate and orbit type
- Last, **Line graph** allows us to see the changes in Success rate throughout the years









Github URL

EDA with SQL

Number of SQL queries were performed to get better understanding of SpaceX dataset:

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'CCA'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date when the first succesful landing outcome in ground pad was acheived
- Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Listing the total number of successful and failure mission outcomes
- Listing the names of the booster_versions which have carried the maximum payload mass using a subquery
- Listing the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015
- Ranking 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

Github URL

Build an Interactive Map with Folium

Folium Map object is a map with an initial center location to be NASA Johnson Space Center at Houston, Texas.

Following objects were added to the map for visualization purposes:

- 4 Launch sites with name labels, locations are marked with red circles;
- Launch outcomes for each site, represented by different color: **green** for success, **red** for failure;
- Lines were added for launch site KSC LC-39A to display it's distance to nearby railroad, highway, coastline and city

Github URL

Build a Dashboard with Plotly Dash

Dashboard application contains input components such as a dropdown list and a range slider to interact with a pie chart and a scatter point chart and explore and manipulate data in an real-time way

- Pie chart shows the total success and the total failure for the certain launch site
- Rangeslider allows a user to select a payload mass in a fixed range
- Scatter chart shows the correlation between Success outcome and Payload Mass

Github URL

Predictive Analysis (Classification)

Building the model

- Load the dataset
- Transform the data and then split it into training and test sets
- Select ML algorithms
- Set the parameters and algorithms to GridSearchCV and fit it to dataset.

Evaluating the model

- Get best hyperparameters for each type of model
- Calculate the accuracy for each model
- Plot Confusion Matrix

Improving the Model

- Use Feature Engineering and Algorithm Tuning

Finding the Best Model

- Compare the accuracy of each model
- The model with the highest accuracy will be best performing model

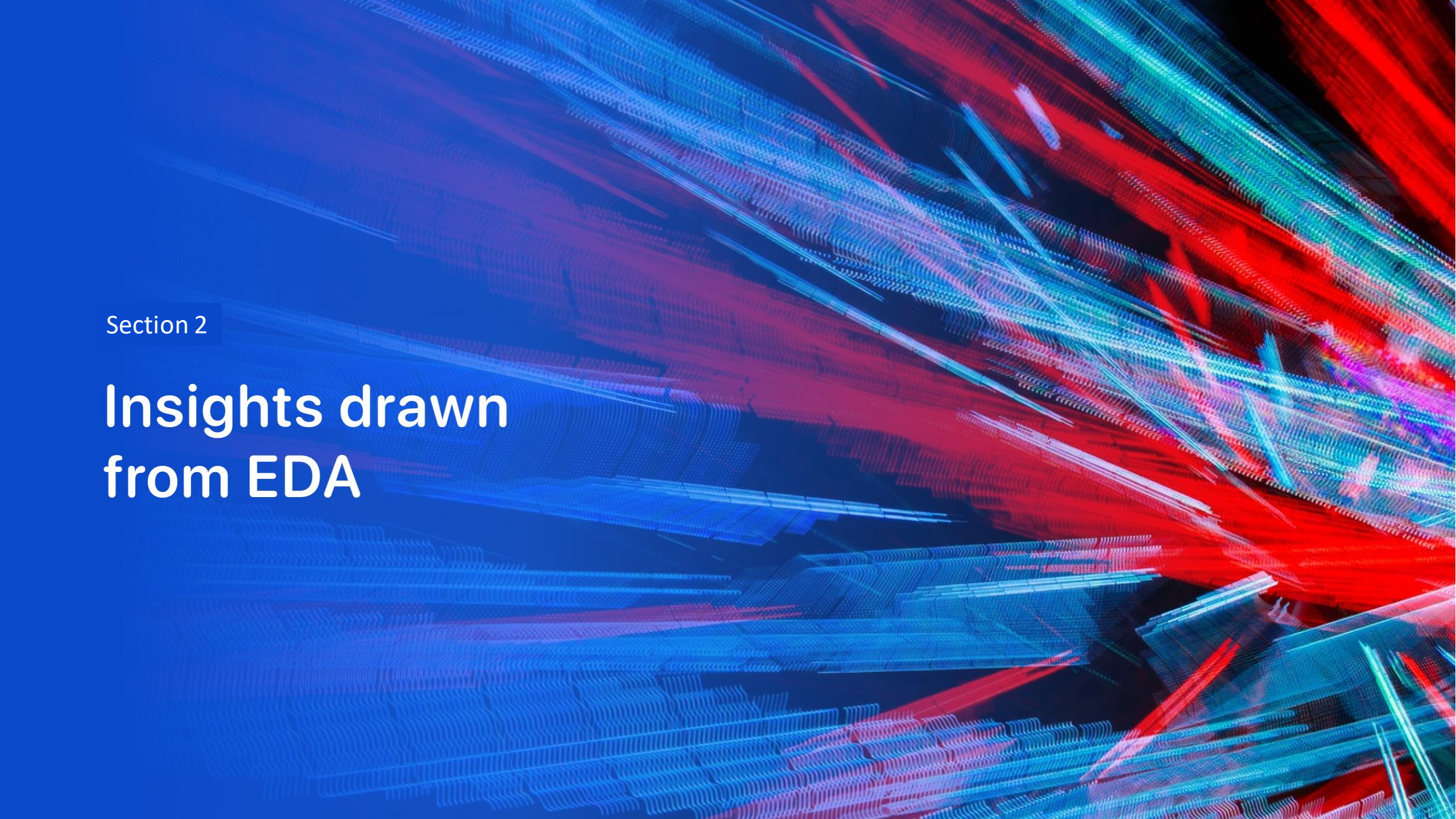
Github URL

Results

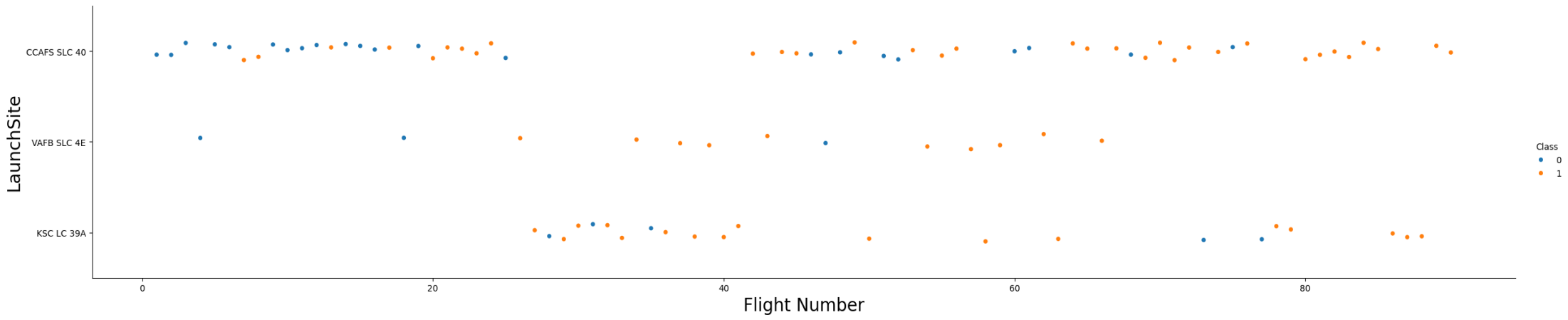
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

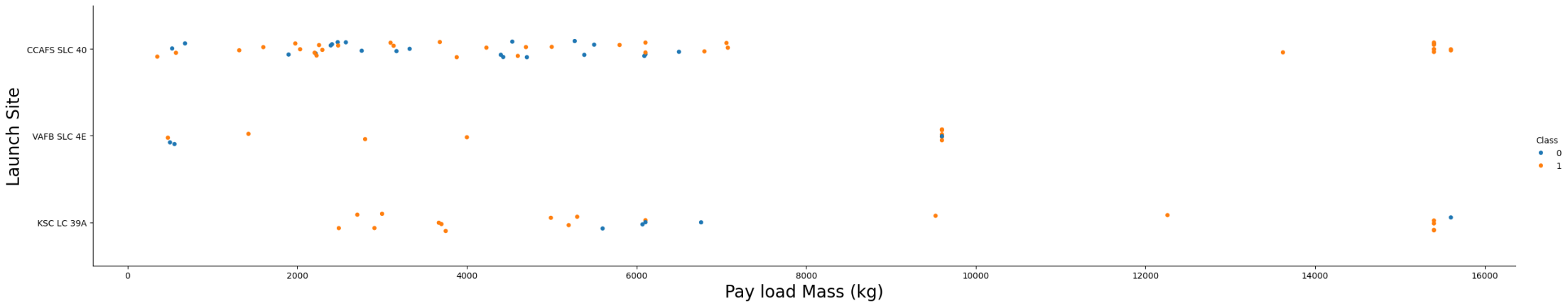


Flight Number vs. Launch Site



We observe that with an increase in the number of flights, success rate also rises significantly.

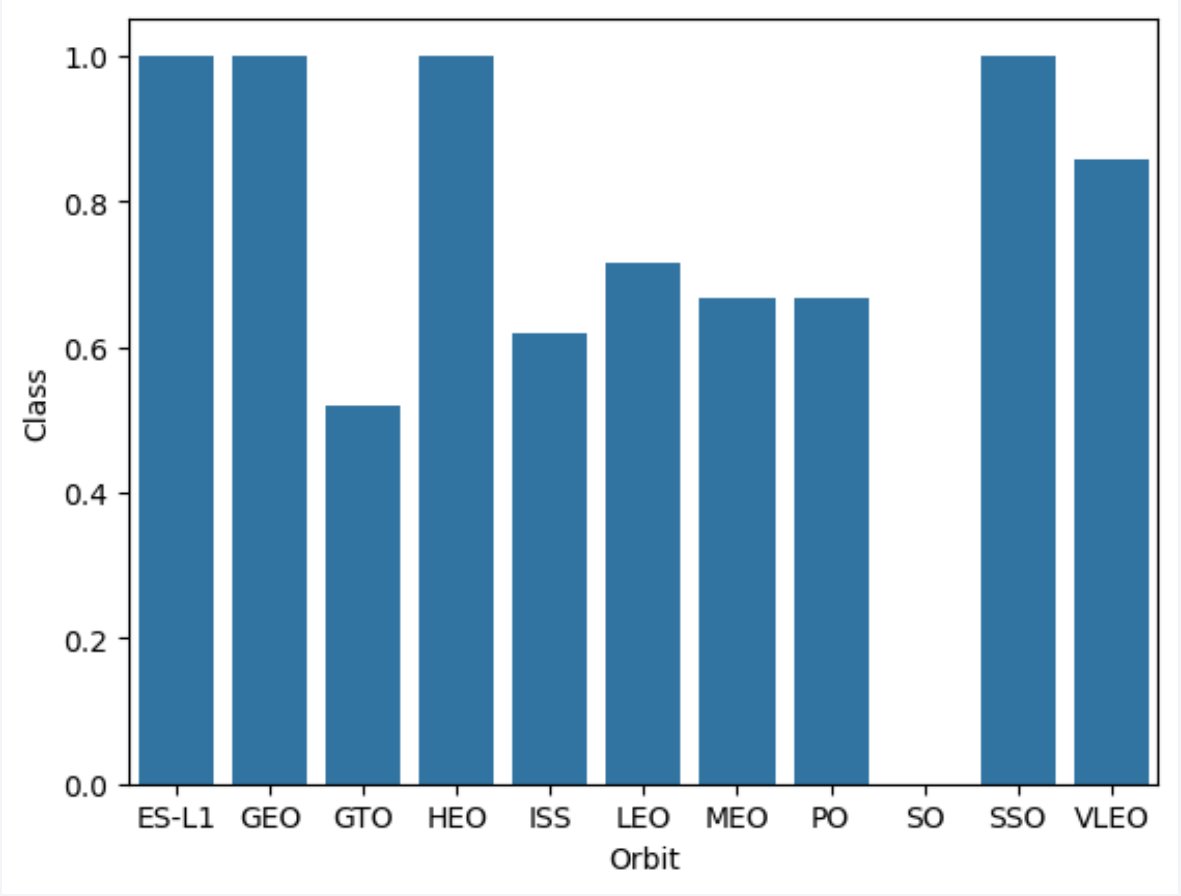
Payload vs. Launch Site



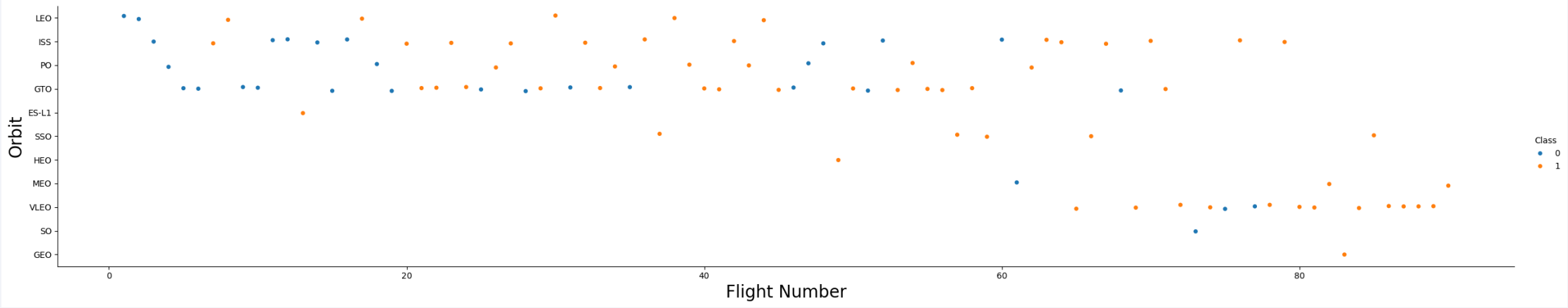
- There is no strong correlation between pay load mass and launch outcome. However probability of success increases significantly on a launch site CCAFS SLC 40 once pay load mass exceeds 7000kg. On the other hand, launch site KSC LC 39A better supports lighter launches, where pay load mass is between 1000 and 5000kg

Success Rate vs. Orbit Type

- The chart shows relationship between success rate and orbit type
- ES-L1, GEO, HEO and SSO orbits have 100% success rate
- SO orbit has zero success rate

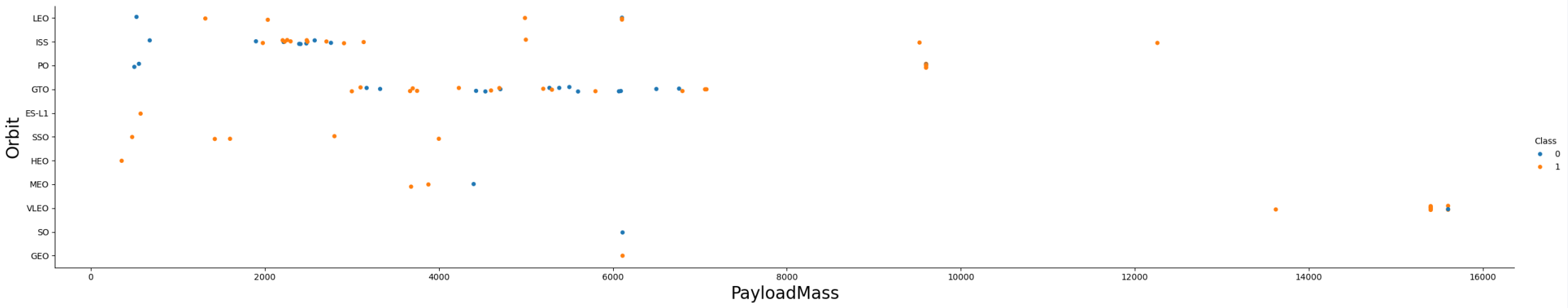


Flight Number vs. Orbit Type



- In the LEO orbit the Success rate increases with the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.
- 4 orbits ES-L1, HEO, SO and GEO have only 1 data point which is insufficient for correlation analysis

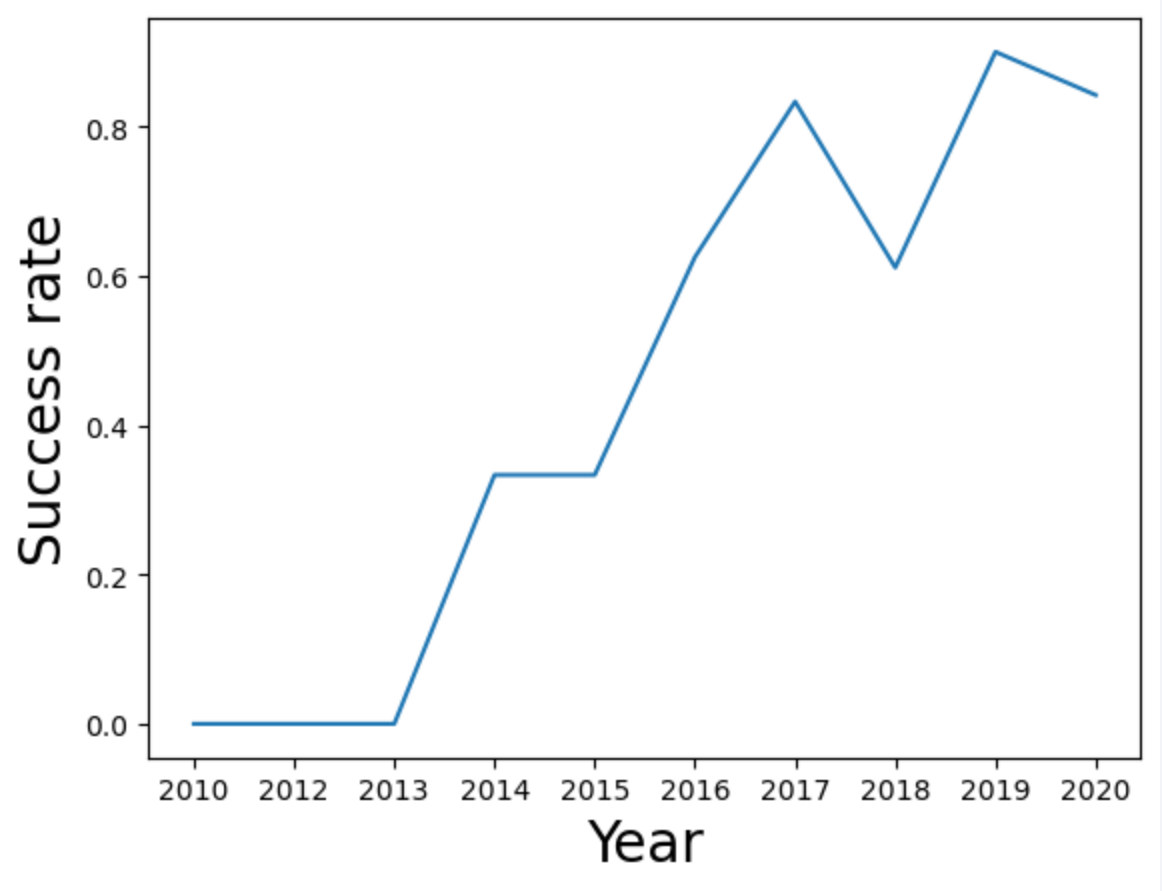
Payload vs. Orbit Type



- LEO, PO and ISS orbits have higher successful landing rate with heavy payloads
- However for GTO there is no correlation between outcome and weight

Launch Success Yearly Trend

The success rate of Space X launches is increasing since 2013



All Launch Site Names

The SELECT **DISTINCT** statement is used to return only unique names of Lauch Sites

```
%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'

```
%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	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

The code executes a SQL query to retrieve the first 5 rows from the table **SPACEXTABLE** where the **Launch_Site** column starts with the string **'CCA'**

Total Payload Mass

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE WHERE Customer like 'NASA (CRS)%'
```

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

sum(PAYLOAD_MASS__KG_)
48213

This query returns the sum of the payload masses (in kilograms) for the rows in the **SPACEXTABLE** table where the **Customer** column starts with the string **'NASA (CRS)'**

Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE WHERE Booster_Version like 'F9 v1.1%'
```

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

avg(PAYLOAD_MASS__KG_)
2534.6666666666665

The query returns the average payload mass (in kilograms) for the rows in the **SPACEXTABLE** table where the **Booster_Version** column starts with the string **'F9 v1.1'**

First Successful Ground Landing Date

```
%sql select min(Date) from SPACEXTABLE WHERE Landing_Outcome like '%ground pad%' AND Landing_Outcome like '%success%'
```

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

min(Date)
2015-12-22

This query selects the minimum (earliest) Date from the SPACEXTABLE table where the Landing_Outcome column contains both "ground pad" and "success". Using the % wildcard ensures that the words "ground pad" and "success" can appear anywhere within the Landing_Outcome column.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select Payload from SPACEXTABLE WHERE (PAYLOAD_MASS__KG_ between 4000 and 6000) and Landing_Outcome='Success (drone ship)'
```

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

Payload
JCSAT-14
JCSAT-16
SES-10
SES-11 / EchoStar 105

In this query we use the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

```
%sql select(select count(Mission_Outcome) from SPACEXTABLE WHERE Mission_Outcome like '%Success%') as Success, \
(select count(Mission_Outcome) from SPACEXTABLE WHERE Mission_Outcome like '%Failure%') as Failure
```

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

Success	Failure
100	1

This query calculates the counts of mission outcomes categorized as "Success" and "Failure" from the **SPACEXTABLE** table and presents them as separate columns named "Success" and "Failure".

Boosters Carried Maximum Payload

This query retrieves the **Booster_Version** from the **SPACEXTABLE** where the **PAYLOAD_MASS__KG_** is equal to the maximum payload mass found in the **PAYLOAD_MASS__KG_** column of the **SPACEXTABLE**.

```
%sql select Booster_Version from SPACEXTABLE where PAYLOAD_MASS__KG_=(select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)
```

```
 * 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

```
%sql select (substr(Date, 6,2)) as Month, Booster_Version, Launch_Site from SPACEXTABLE\
where Landing_Outcome='Failure (drone ship)' and Date like '2015%'
```

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

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

This query retrieves the month, booster version, and launch site from the **SPACEXTABLE** where the landing outcome is specified as "Failure (drone ship)" and the date starts with "2015".

substr(Date, 6, 2) extracts the month from the **Date** column, starting from the 6th character, and 2 characters long. It aliases this extracted month as "Month".

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

This query retrieves and counts the occurrences of different landing outcomes from the **SPACEXTABLE** within a specific date range.

It then presents the results grouped by landing outcome, sorted in descending order based on the total count of each landing outcome.

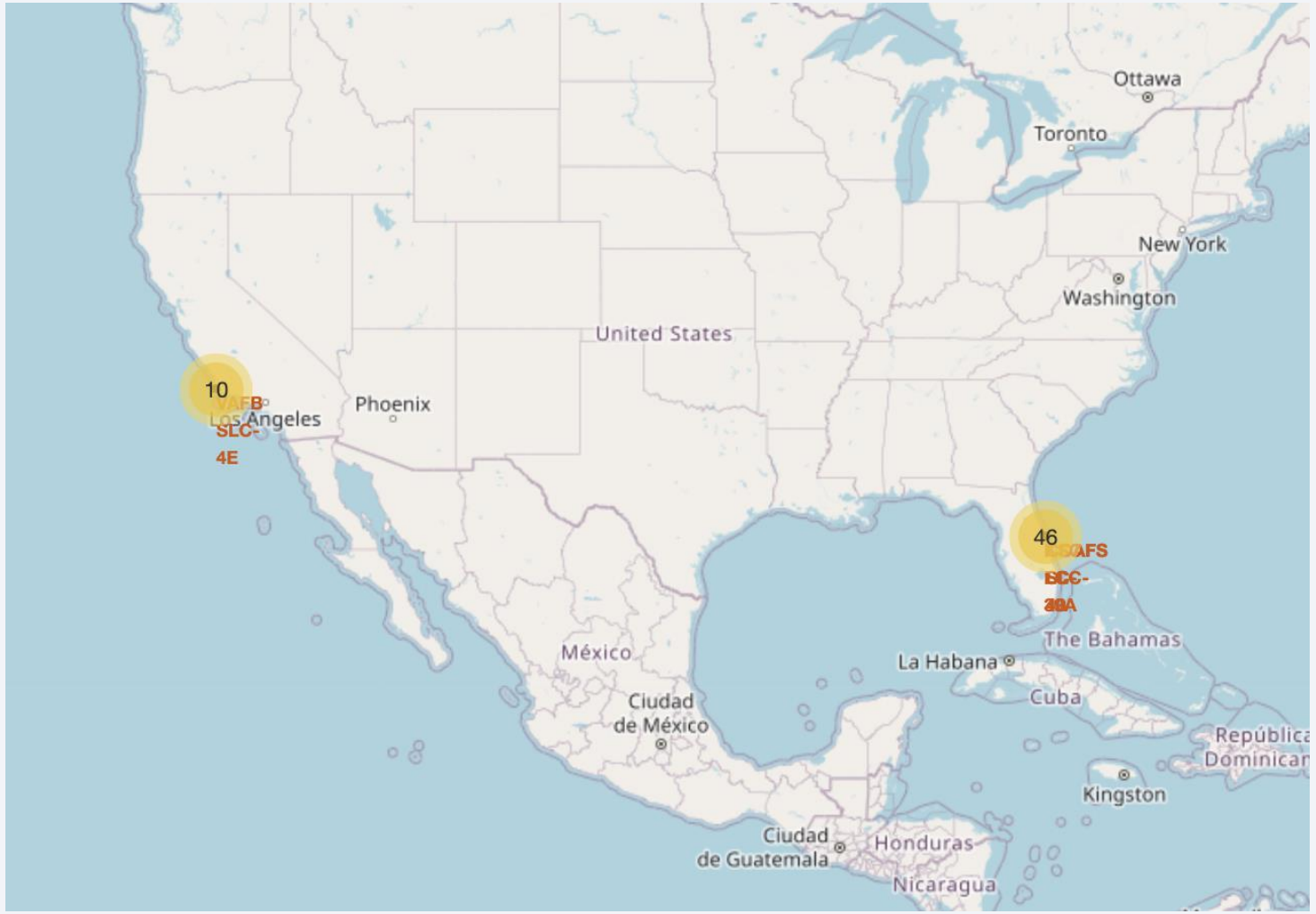
```
%sql select Landing_Outcome as "Landing Outcome", \
count(Landing_Outcome) as "Total Count" from SPACEXTABLE\
where Date between '2010-06-04' and '2017-03-20' \
group by Landing_Outcome\
order by count(Landing_Outcome) desc
```

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

Landing Outcome	Total 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

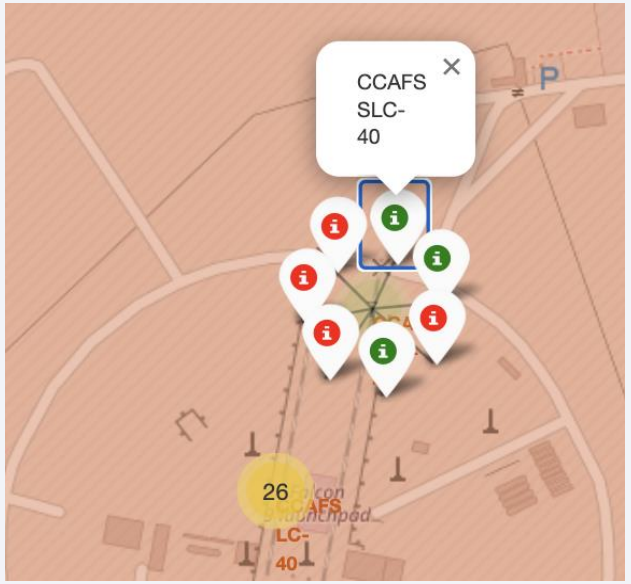

Section 3
Launch Sites
Proximities Analysis

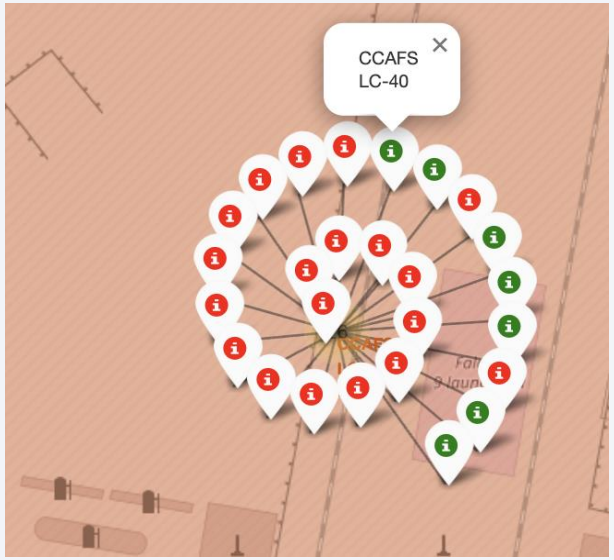
Locations of Launch Sites

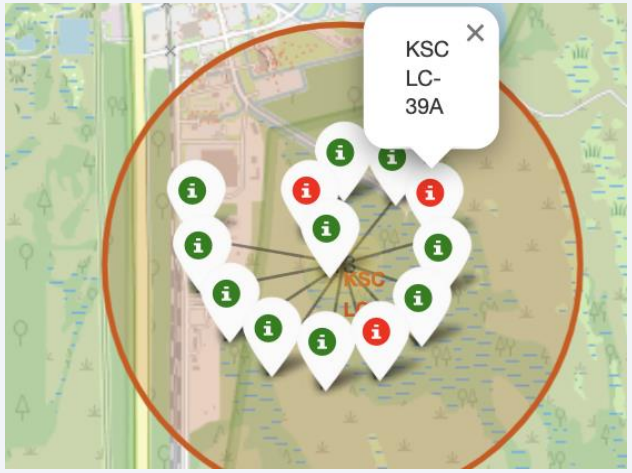


All Space X launch sites are located on US territory

Launch Sites - Launch outcomes





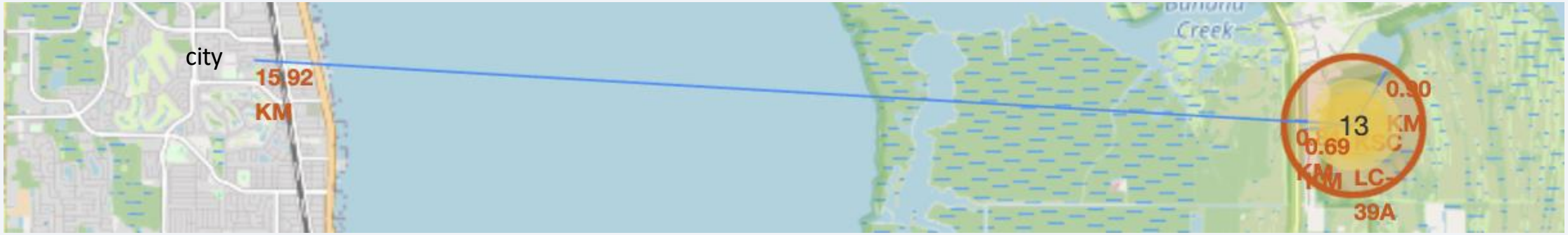


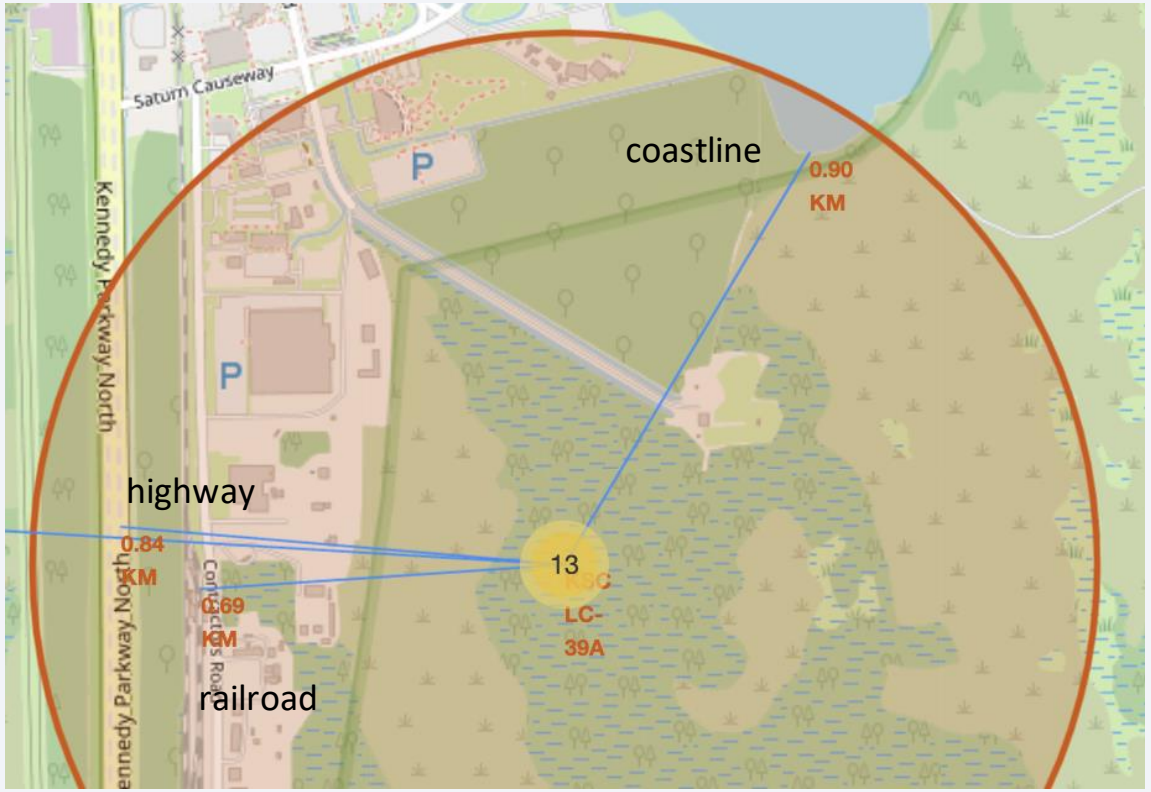
On this map Green markers represent successful launches, Red markers – failures

Lauch site CCAFS LC-40 has the highest number of launch attempts with the lowest rate of success

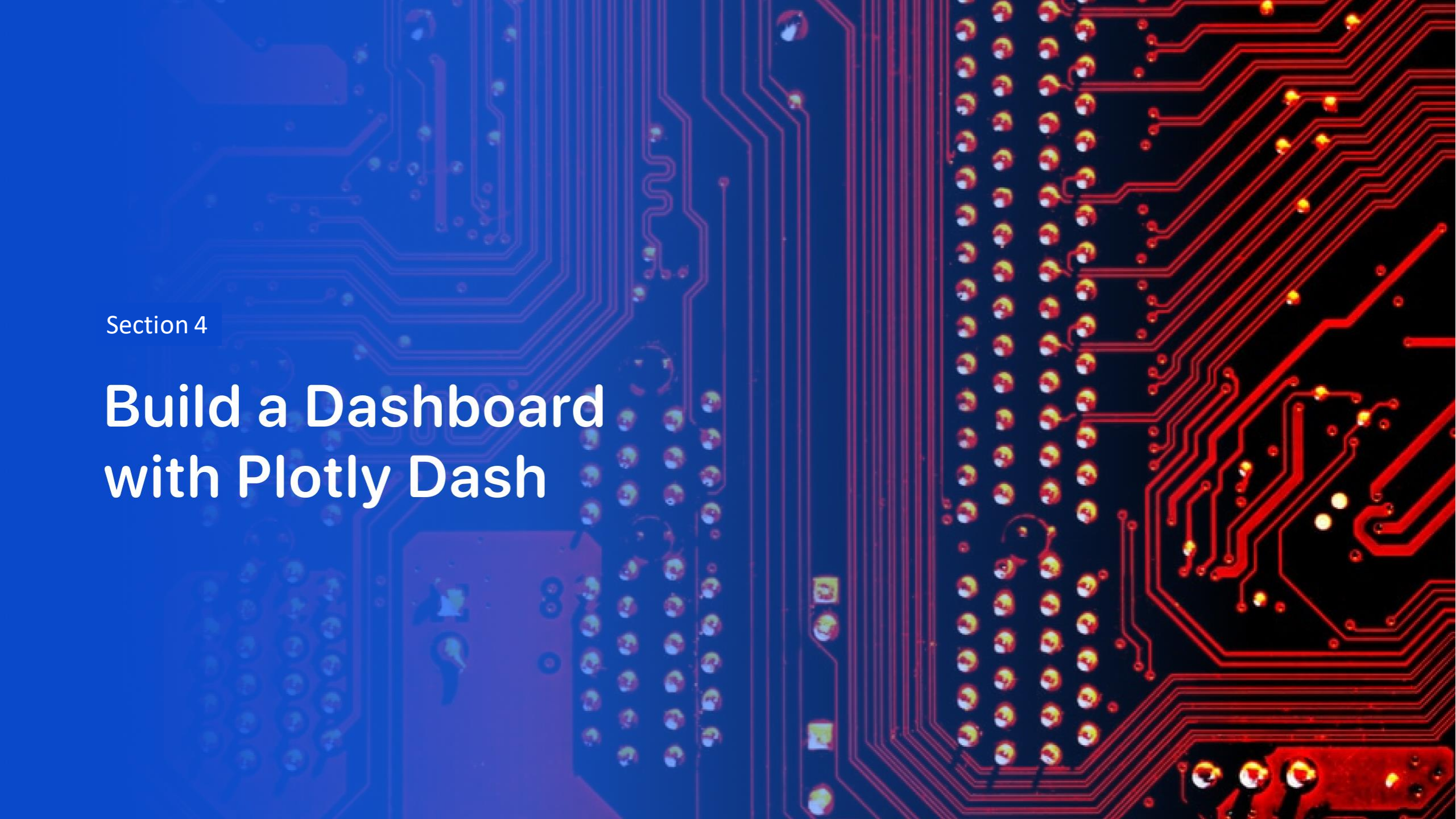
Out of 4 launch sites, KSC LC-39A has the highest overall success rate

KSC LC-39A distance to railroad, highway, coastline and city

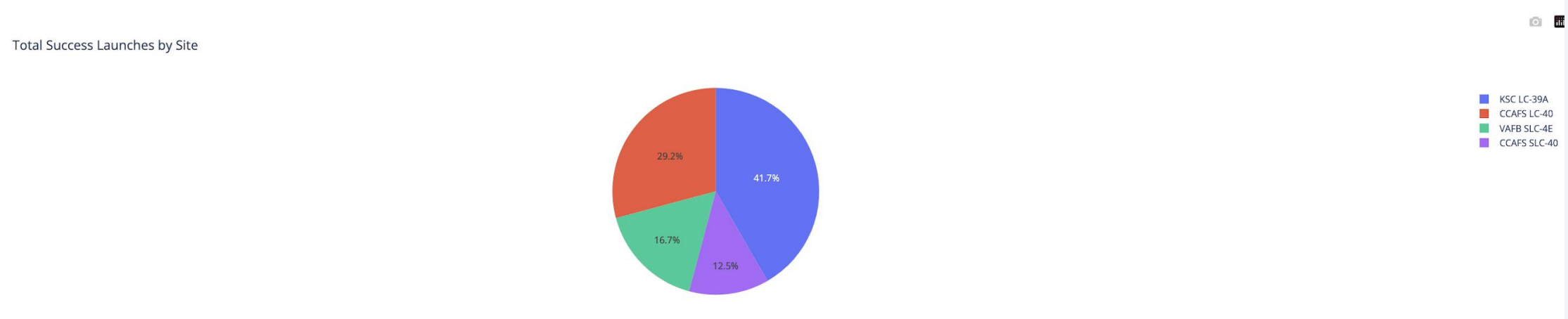




- Are launch sites in close proximity to railways? - Yes
- Are launch sites in close proximity to highways? - Yes
- Are launch sites in close proximity to coastline? - Yes
- Do launch sites keep certain distance away from cities? - Yes

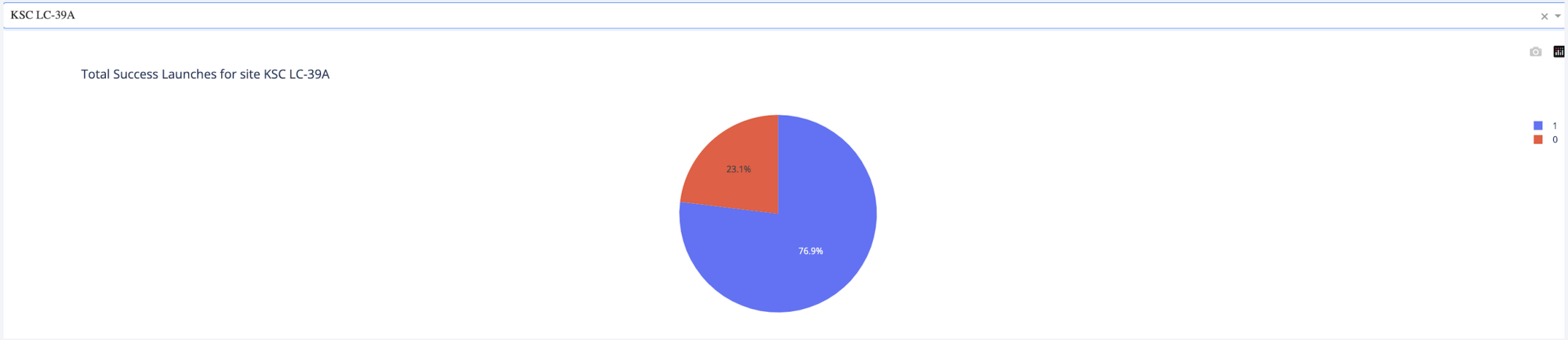

Section 4
Build a Dashboard with Plotly Dash

Success percentage by each site



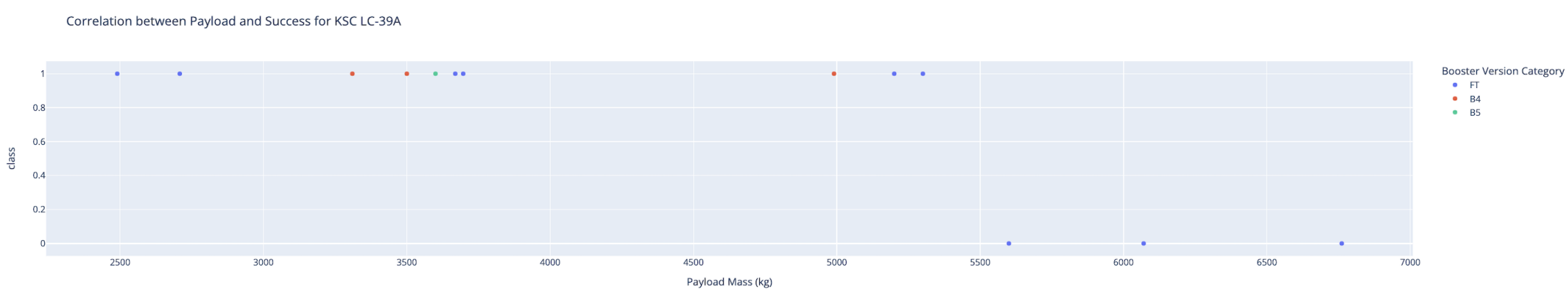
Out of four launch sites, KSC LC-39A has the highest rate of success

Highest launch success ratio: KSC LC-39A



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

Payload vs. Launch Outcome for KSC LC-39A



KSC LC-39A has the most success when payload mass is below 5500kg

Section 5

Predictive Analysis (Classification)

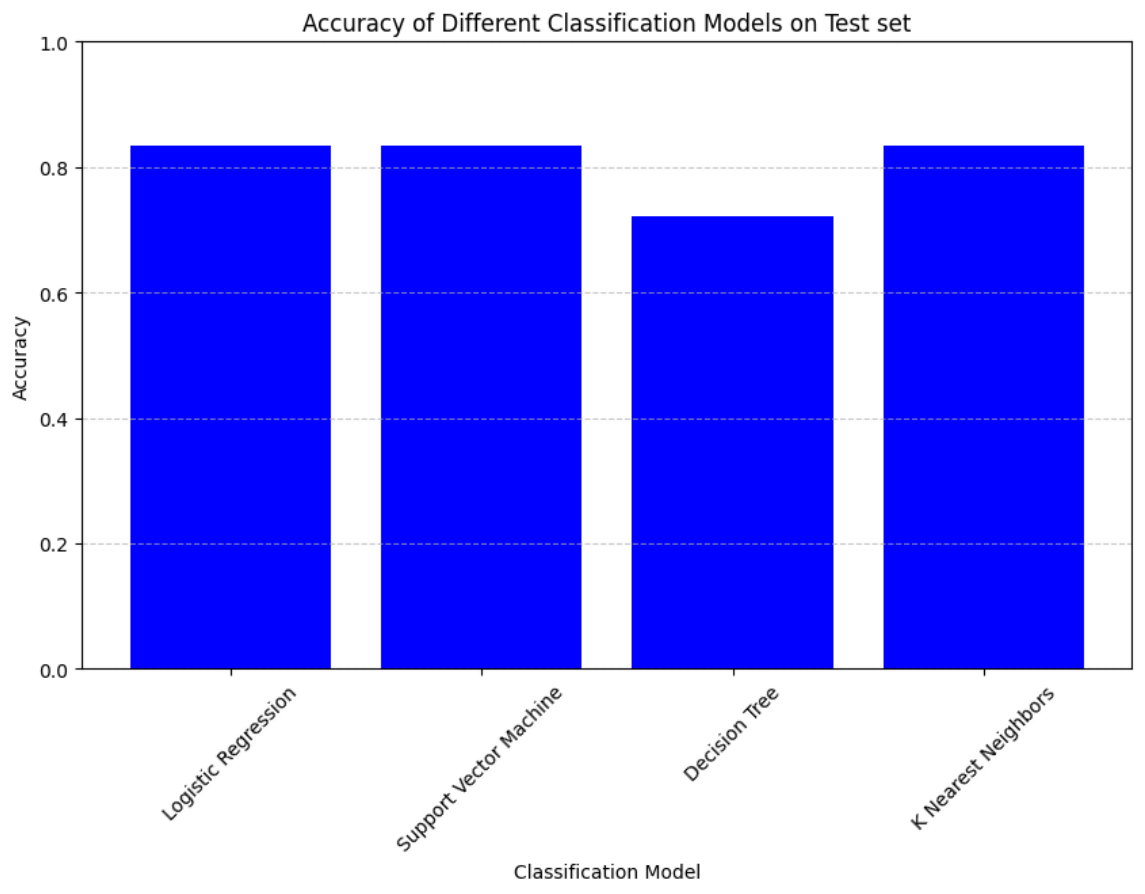
Classification Accuracy

```
print('Accuracy for Logistics Regression method:', logreg_cv.score(X_test, Y_test))
print( 'Accuracy for Support Vector Machine method:', svm_cv.score(X_test, Y_test))
print('Accuracy for Decision tree method:', tree_cv.score(X_test, Y_test))
print('Accuracy for K nearest neighbors method:', knn_cv.score(X_test, Y_test))
```

```
Accuracy for Logistics Regression method: 0.8333333333333334
Accuracy for Support Vector Machine method: 0.8333333333333334
Accuracy for Decision tree method: 0.7222222222222222
Accuracy for K nearest neighbors method: 0.8333333333333334
```

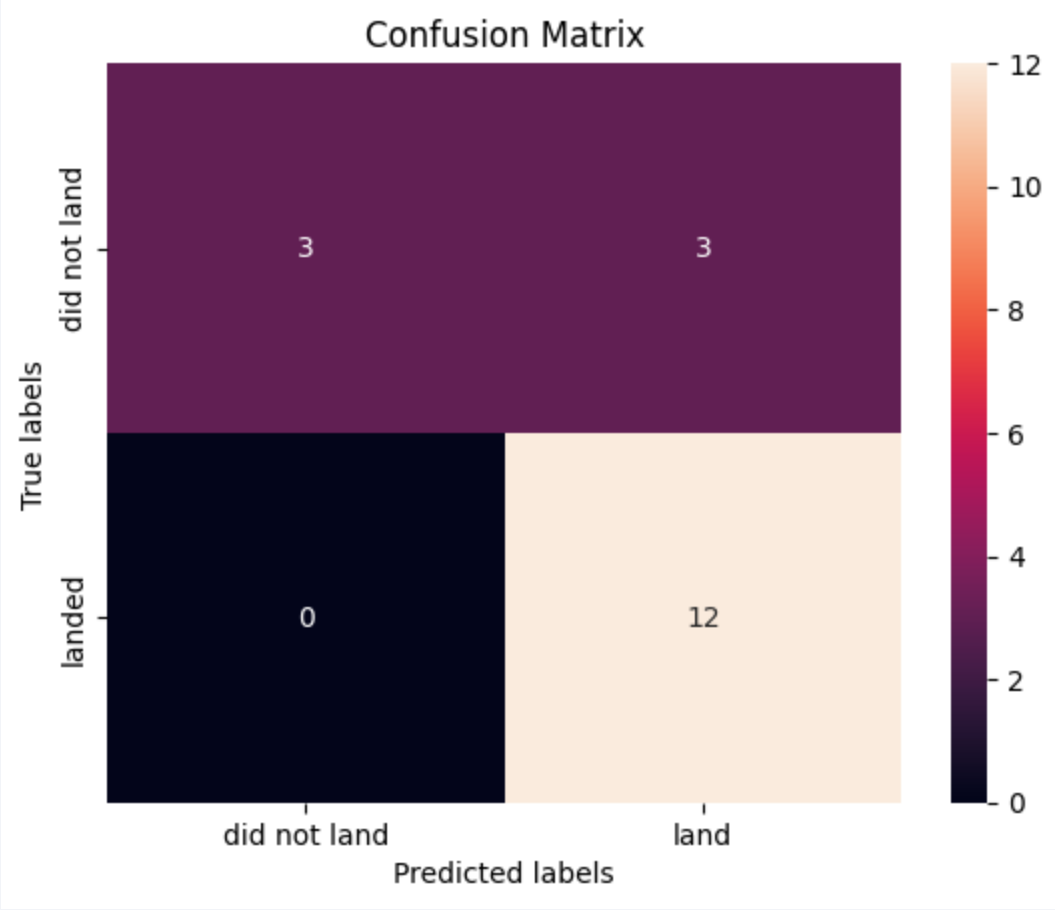
3 models show the best score of 0.83 on test data:

- Logistics Regression
- Support Vector Machine
- K nearest neighbors



Confusion Matrix

Confusion matrix is the same for all 3 best performing models



Conclusions

- The outcome of a mission is influenced by various factors, including the launch site, orbital trajectory, and notably, the number of prior launches. It's reasonable to conclude that accumulated knowledge from past launches has contributed to transitioning from launch failures to successes.
- Out of 4 launch sites, KSC LC-39A has the highest overall success rate of 76.9%
- There is no strong correlation between pay load mass and launch outcome. However probability of success increases significantly on a launch site CCAFS SLC 40 once pay load mass exceeds 7000kg. On the other hand, launch site KSC LC 39A better supports lighter launches, where pay load mass is between 1000 and 5000kg
- Depending on the specific orbits, the payload mass emerges as a crucial consideration for mission success. Different orbits necessitate varying payload masses, with some favoring lighter payloads over heavier ones.
- Out of all orbits with more than 1 data point, only SSO orbit has 100% success rate

Thank you!