Winning Space Race with Data Science

Yan Cheung
January 2023

Outline

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

Executive Summary

To predict if the Falcon 9 first stage will land successfully, past data was used from two publicly available sources to gather information about past launches. The data obtained was then cleaned and processed. Exploratory data analysis (EDA) was performed to get an idea of the data. SQL, data visualization and Folium were all used to complete EDA. Machine learning algorithms were then deployed to test the models.

It was found that the more launches there are, the greater the success rate over the years. With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS orbits. Orbits GEO, HEO, SSO, ES-L1 has the best success rate. The KSC LC-39A site has the most successful launches, but if the mass is above 10,000 kg the success rate is 100% at the site CCAFS LC-40. The SVM classifier is the best machine learning algorithm for this dataset

Introduction

SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information will be used to bid against SpaceX for a rocket launch.

The rocket can be considered to be split into stage one, stage two and fairings. The payload is enclosed in the fairings. Stage two, or the second stage, helps bring the payload to orbit, but most of the work is done by the first stage and is much larger than the second stage.

Unlike other rocket providers, SpaceX's Falcon 9 can recover the first stage. Sometimes the first stage does not land, sometimes it will crash. Other times, Space X will sacrifice the first stage due to the mission parameters like payload, orbit, and customer.

We will predict if the Falcon 9 first stage will land successfully, so that it can be reused. We will then train a machine learning model and use public information to predict if SpaceX will reuse the first stage.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
- Perform data wrangling
- 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

SpaceX launch Data:
https://api.spacexdata.com/v4/launches/past

Publicly available SpaceX launch data that is gathered from SpaceX REST API.

- This API includes data about launches, rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.
- A get request is performed using the requests library to obtain the launch data, which will be used to get the data from the API. This result can be viewed by calling the .json() method.
- The json_normalize function will allow us to “normalize” the structured json data into a flat table.

Falcon 9 launch Data:
https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

- Falcon 9 Launch data is collected by using the Python BeautifulSoup package to web scrape HTML tables that contain valuable Falcon 9 launch records.
- The data with be parsed from those tables and converted them into a Pandas data frame for further visualization and analysis.

Data Collection – SpaceX API

GitHub URL to the notebook

1. Getting Response from API

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

```
response = requests.get(spacex_url)
```

2. Converting Response to .json

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

3. Use API again to get information about the launches and construct a dataset using the data

```
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}
```

4. Create a Pandas data frame from the dictionary

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

Data Collection - Scraping

GitHub URL to the notebook

1. Perform HTTP GET method to request from HTML

```
data  = requests.get(static_url).text
```

2. Use BeautifulSoup to find the third table on the Wiki page.

```
first_launch_table = html_tables[2]
print(first_launch_table)
```

3. Extract column names one by one

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

4. Create a dictionary with keys from the extractedcolumn names

```
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
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']=[]
```

5. Parse data and append to a table (refer to notebook)

6. Convert dictionary to a dataframe

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

Data Wrangling

GitHub URL to the notebook

In the data set, there are several different cases where the booster did not land successfully:

- True Ocean - successfully landed to a specific region of the ocean
- False Ocean - unsuccessfully landed to a specific region of the ocean.
- True RTLS - successfully landed to a ground pad
- False RTLS - unsuccessfully landed to a ground pad.
- True ASDS - successfully landed on a drone ship
- False ASDS - unsuccessfully landed on a drone ship

These were converted into Training Labels with:

- 1 meaning the booster successfully landed
- 0 means it was unsuccessful.

1. Calculate the number of launches on each site
2. Calculate the number and occurrence of each orbit
3. Calculate the number and occurrence of mission outcome per orbit type
4. Create a landing outcome label from Outcome column
5. Export to a CSV

EDA with Data Visualization

GitHub URL to the notebook

- Scatter Graph

Scatter graphs were drawn to show if there is any correlation between two variables and are suitable for continuous data.

  - Flight Number Vs. Pay Load Mass
  - Flight Number Vs. Launch Site
  - Pay Load Vs. Launch Site
  - Flight Number Vs. Orbit
  - Pay Load Mass Vs. Orbit

- Bar Chart

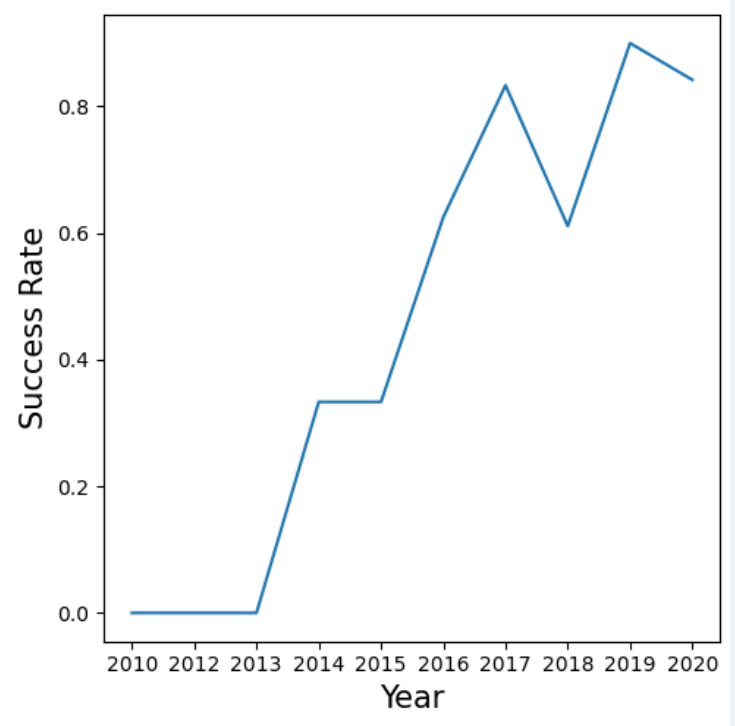
Bar charts makes it easy to compare sets of data between discrete variables and a good way to show relative sizes

  - Orbit Vs. Success Rate

- Line Chart

Line charts show information that changes over time.

Success Rate Vs. Year



EDA with SQL

GitHub URL to the notebook

SQL queries performed to gather information about the dataset

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first succesful landing outcome in ground pad was acheived.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- 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.
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Build an Interactive Map with Folium

GitHub URL to the notebook

Step 1 – Mark all launch sites on a map

- Firstly a folium Map object was created and a circle for each launch site in the data frame launch_sites
- Using folium.Circle to add a highlighted circle area with a text label on a specific coordinate

Step 2 – Mark the success/failed launches for each site on the map

- Next, we enhance the map by adding the launch outcomes for each site, and see which sites have high success rates. Marker clusters are used to simplify a map containing many markers having the same coordinate.
- If a launch was successful (class=1), then we use a green marker and if a launch was failed, we use a red marker (class=0)

Step 3 – Calculate the distances between a launch site to its proximities

- Finally a MousePosition was added to the map to get coordinate for a mouse over a point on the map, to easily find the coordinates of any points of interests (such as railway)
- Draw a PolyLine between a launch site to the selected point of interest and the distance was calculated

Build a Dashboard with Plotly Dash

GitHub URL to the notebook

- A Plotly Dash application was built for users to perform interactive visual analytics on SpaceX launch data in real-time.
- This dashboard application contains input components such as a
  - Dropdown list – With four different launch sites, a dropdown menu is used to select the different launch site and see which one has the largest success count and to check its check its detailed success rate
  - Pie chart – To show the total launches success count and detailed success rate
  - Range slider – to find if variable payload is correlated to mission outcome. From a dashboard point of view, we want to be able to easily select different payload range and see if we can identify some visual patterns.
  - Scatter point chart – Pay Load Vs. Outcome to we can visually observe how payload may be correlated with mission outcomes for selected site(s). In addition, we want to color-label the Booster version on each scatter point so that we may observe mission outcomes with different boosters.

Predictive Analysis (Classification)

GitHub URL to the notebook

At this stage, a machine learning pipeline was created to predict if the first stage will land given the data

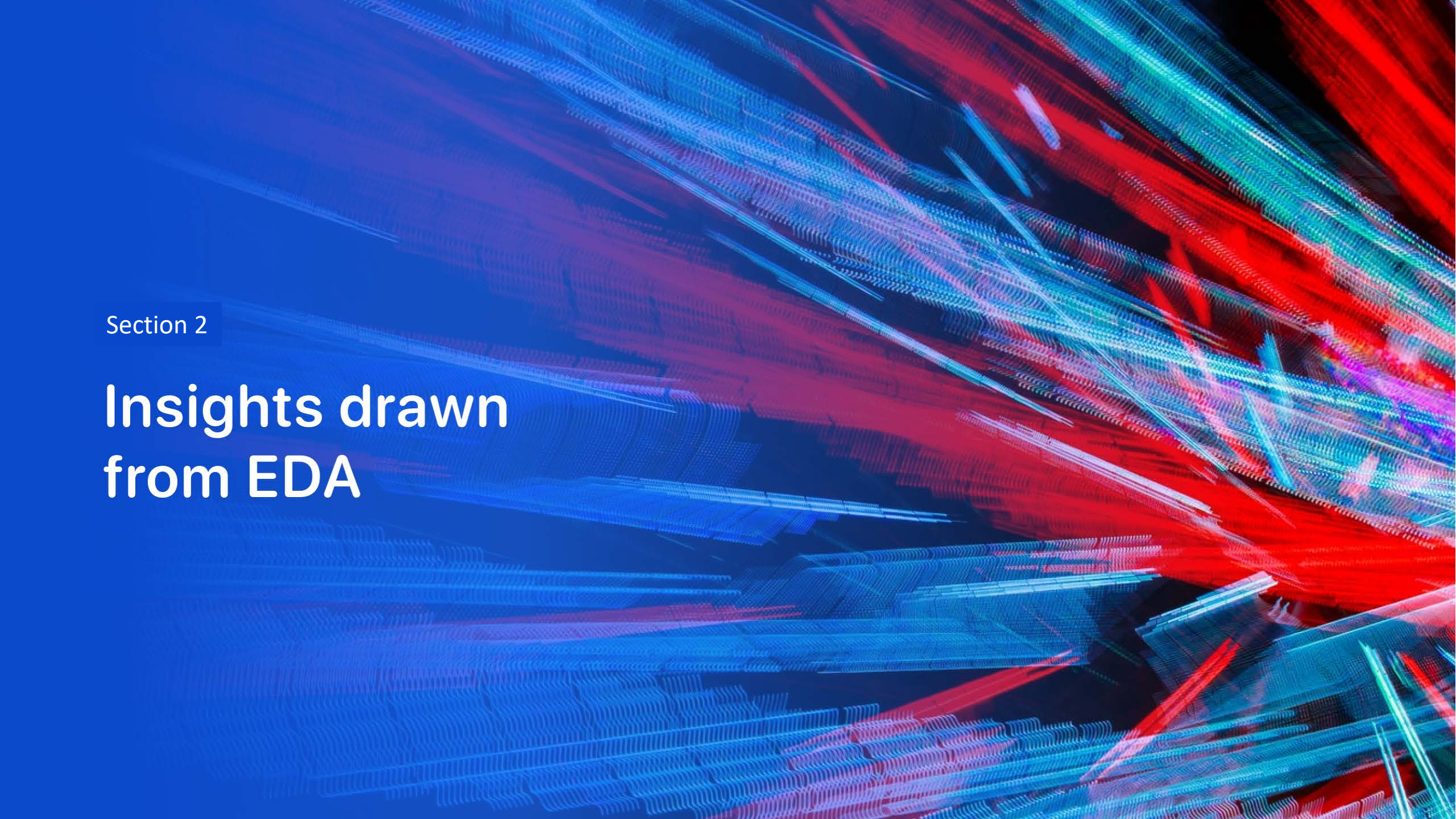
1. Perform exploratory Data Analysis and determine Training Labels
2. Create a column for the class and Standardize the data
3. Split into training data and test data
4. Use following models on the training data and utilise GridSearchCV to perform hyperparameter tuning in order to determine the optimal values for said model
    - Logistic Regression
    - Support Vector Machine (SVM)
    - Decision Tree Classification
    - K Nearest Neighbours
5. Plot confusion matrix
6. Calculate the accuracy of the model on the test data

Results

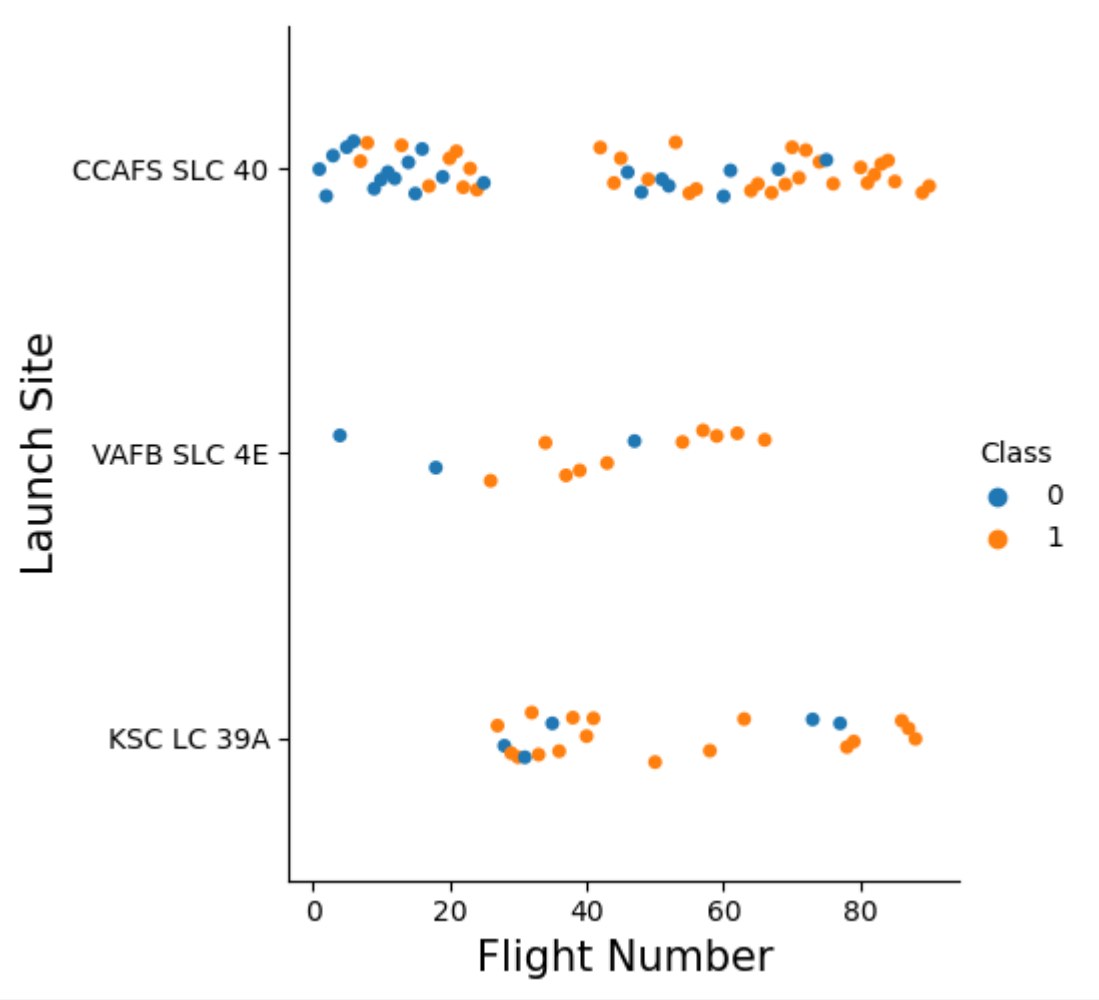
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA



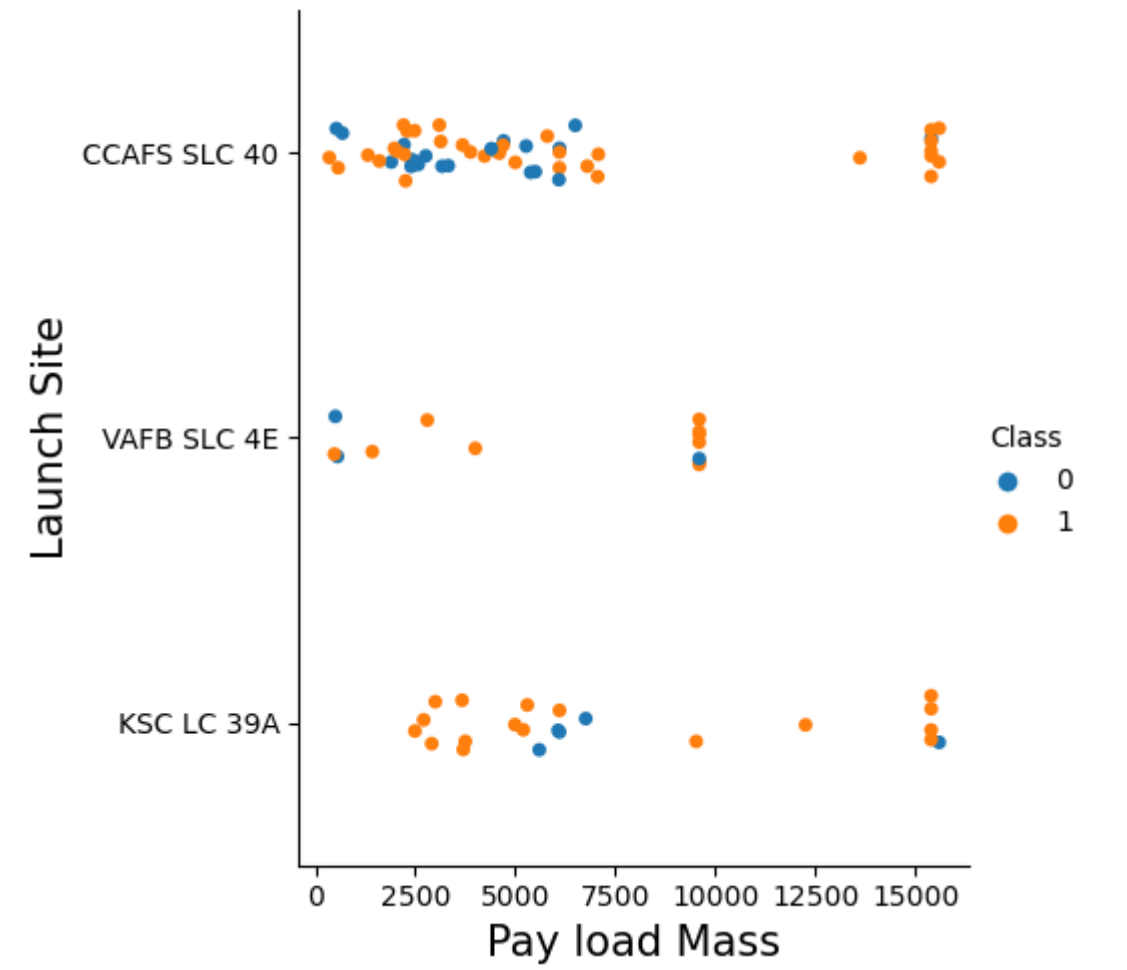
Flight Number vs. Launch Site



From this scatter graph we can see visually that KSC LC 39A and VAFB SLC 4E both appear to have a higher success rate then CCAFS SLC 40. A bar chart showing Launch Site Vs. Success Rate can confirm this.

We can also see there are more flights from CCAFS SLC 40 and there are a higher number of failures for early flights.

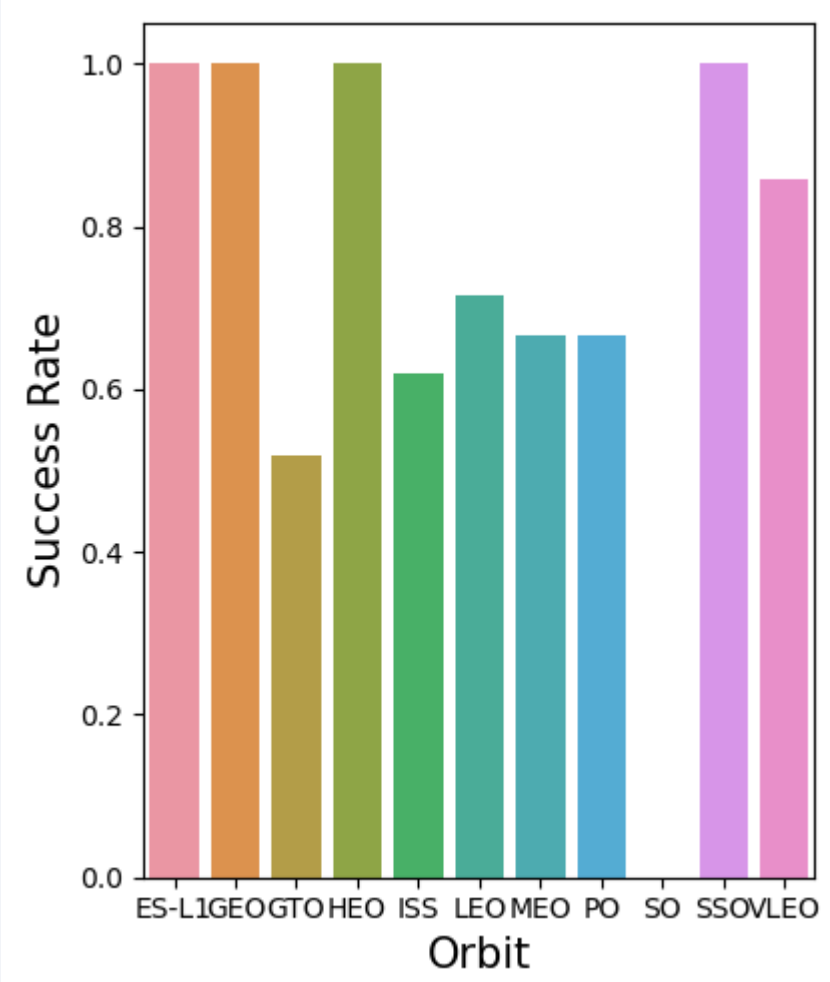
Payload vs. Launch Site



We see that CCAFS LC-40, has a success rate of 60%, but if the mass is above 10,000 kg the success rate is 100%.

There appears to be no correlation between pay load mass and the launch site.

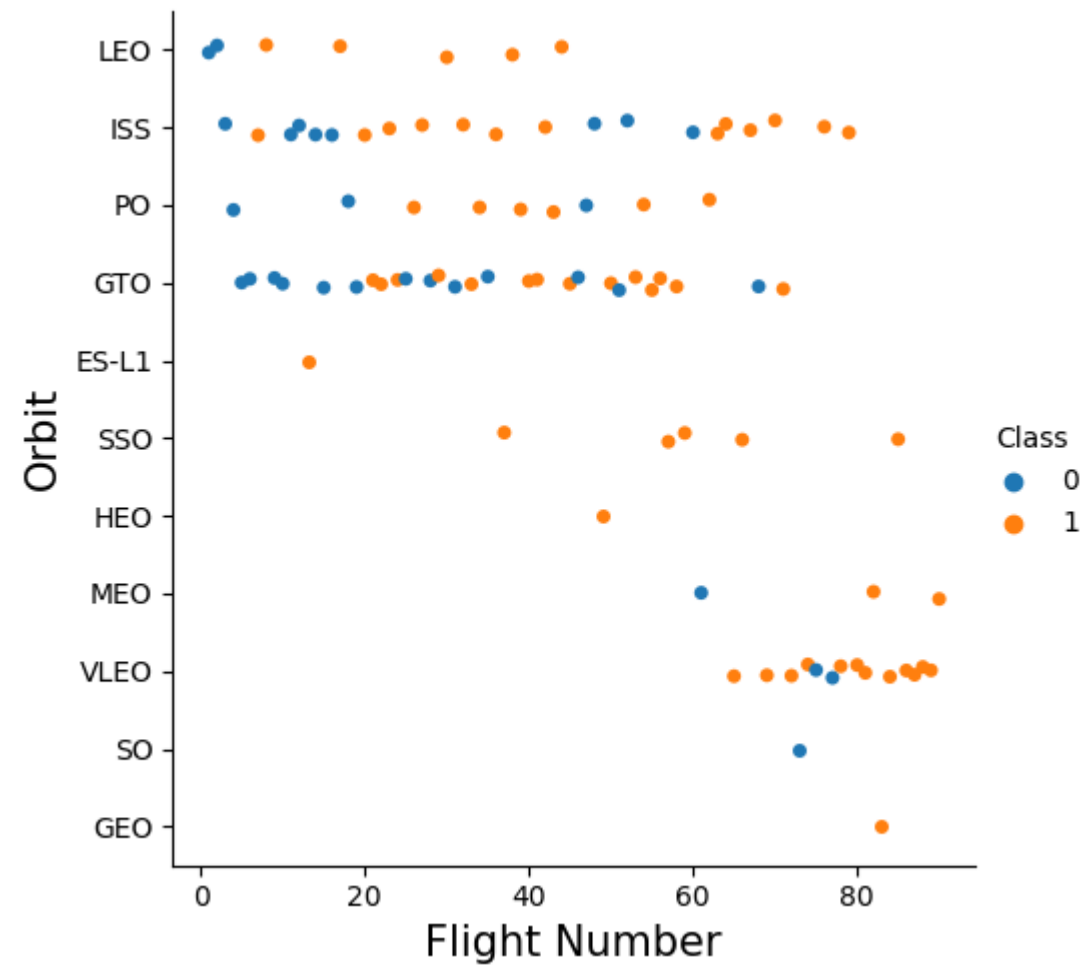
Success Rate vs. Orbit Type



Orbit GEO, HEO, SSO, ES-L1 has the best success rate.

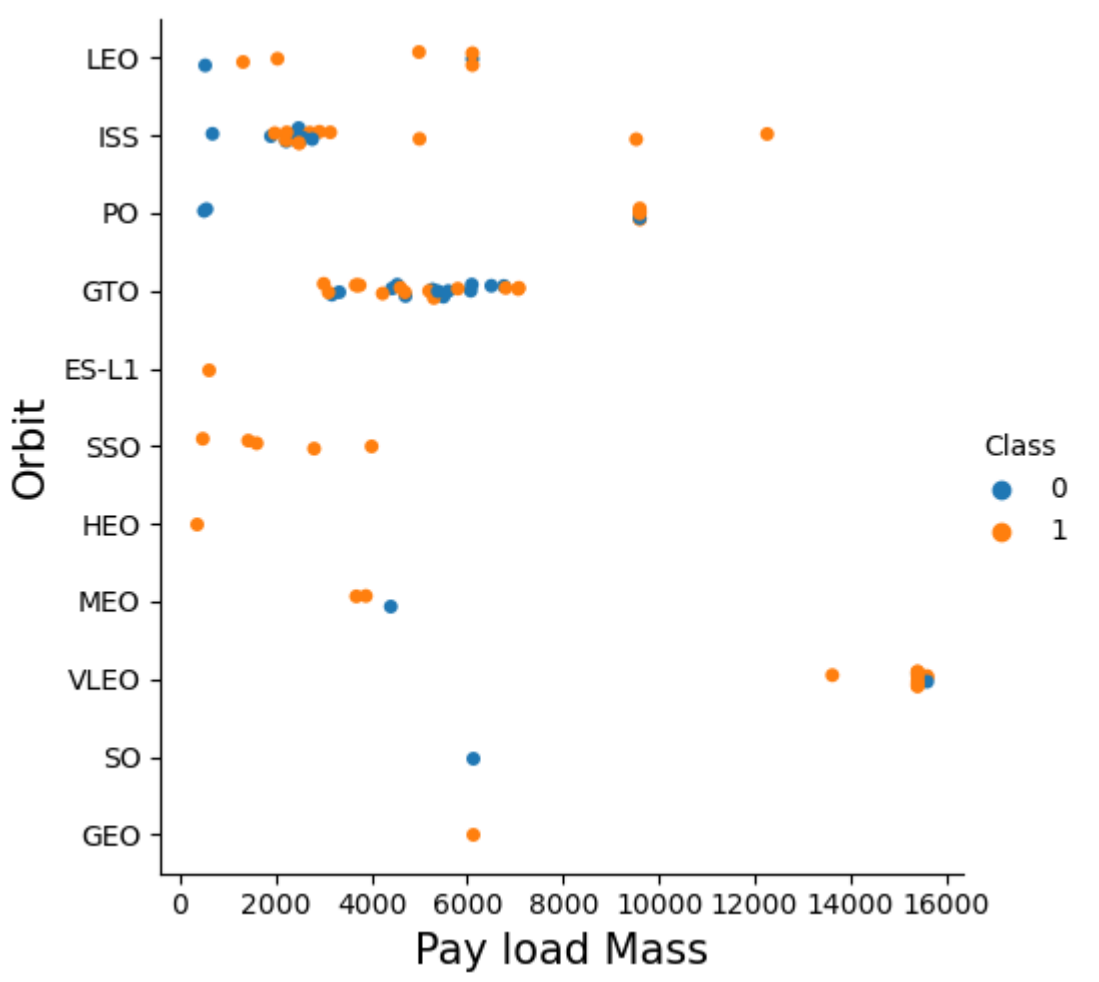
SO has a zero success rate and GTO has the next lowest success rate.

Flight Number vs. Orbit Type



We see that in 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.

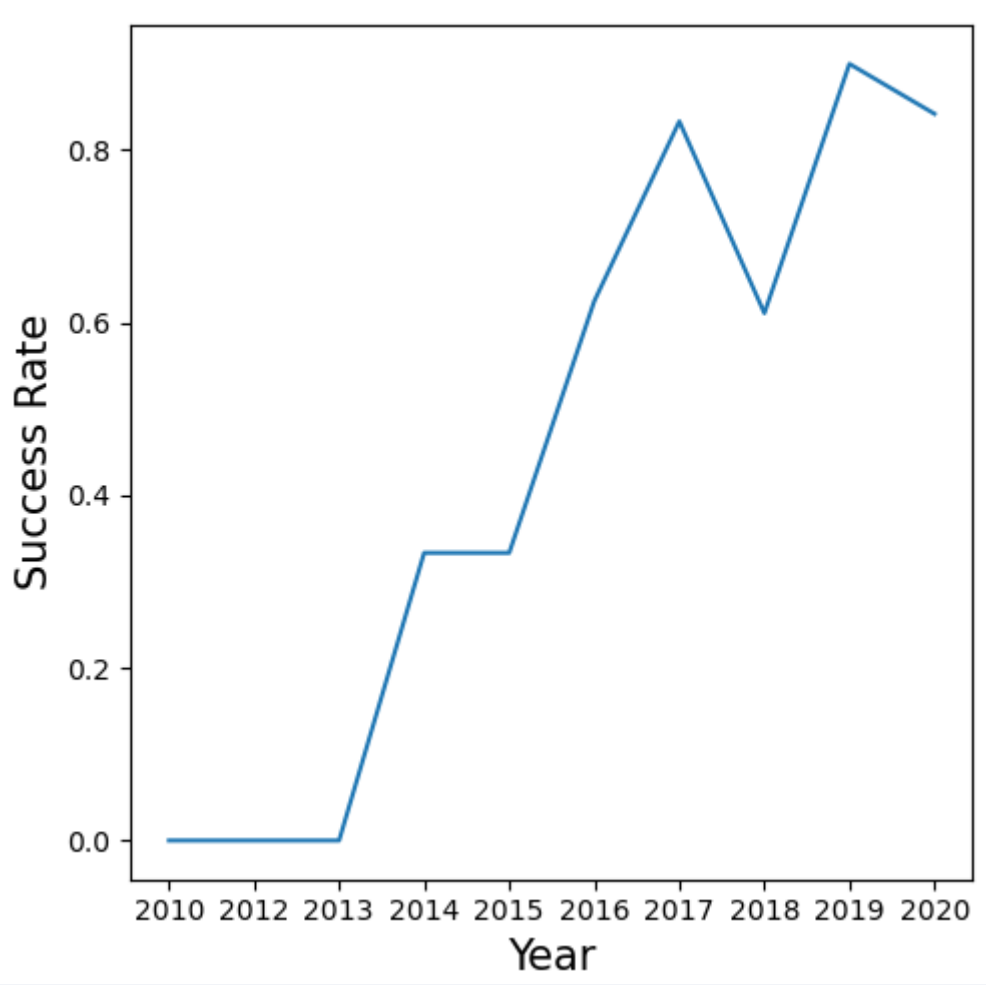
Payload vs. Orbit Type



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

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Launch Success Yearly Trend



We can observe that the success rate since 2013 kept increasing till 2020

All Launch Site Names

```
%sql select distinct "Launch_Site" from SPACEXTBL
```

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

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

DISTINCT function is used to specify that the statement is a query which returns unique values in specified columns.

Launch Site Names Begin with 'CCA'

```
%sql select * from SPACEXTBL 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
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

LIKE operator is used in a WHERE clause to search for a specified pattern in the column. The % are used as wildcards to find sites that begin with 'CCA'.

LIMIT is a clause to specify the maximum number of rows the result set must have

Total Payload Mass

```
%sql select SUM("PAYLOAD_MASS__KG_") as "Total Payload Mass" from SPACEXTBL \
    where "Customer" = "NASA (CRS)"
```

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

Total Payload Mass
45596

SUM function returns the total sum of a numeric column

Average Payload Mass by F9 v1.1

```
%sql select AVG("PAYLOAD_MASS__KG_") as "Average Payload Mass" from SPACEXTBL \
    where "Booster_Version" = "F9 v1.1"
```

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

Average Payload Mass
2928.4

AVG function returns the average value of a numeric column

First Successful Ground Landing Date

```
%sql select substr(Date,7,4) as Year, substr(Date,4,2) as Month, substr(Date,1,2) as Day from SPACEXTBL \
    where "Landing _Outcome"="Success (ground pad)" \
    limit 1
```

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

Year	Month	Day
2015	12	22

The first successful landing outcome was 22nd December 2015

MIN function returns the smallest value of the selected column.

SQLLite does not support dates so SUBSTR() was used to extract the date substring from a string

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select "Booster_Version" from SPACEXTBL \
    where "Landing _Outcome"="Success (drone ship)" \
    and PAYLOAD_MASS__KG_ between 4000 and 6000
```

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

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

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates and is inclusive

Total Number of Successful and Failure Mission Outcomes

```
%sql select substr(Mission_Outcome,1,7) as "Mission Outcome", count(*) as Total from SPACEXTBL  \
    group by 1
```

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

Mission Outcome	Total
Failure	1
Success	100

We can see that there were 100 successes and 1 failure.

The query used SUBSTR() to extract ‘Failure’ and ‘Success’.

COUNT is a function that takes the name of a column as argument and counts the number of rows then the column is not NULL

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

Boosters Carried Maximum Payload

```
%sql select distinct("Booster_Version") from SPACEXTBL \
    where "PAYLOAD_MASS__KG_" = (select max("PAYLOAD_MASS__KG_") from SPACEXTBL)
```

```
 * 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

A subquery is a query within another SQL query and embedded within the WHERE clause.

It is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

2015 Launch Records

```
%sql select substr(Date, 4, 2) as month, \
    "Landing _Outcome" as "Landing Outcome", \
    "Booster_Version" as "Booster Version", \
    "Launch_Site" as "Launch Site" from SPACEXTBL\
    where substr(Date,7,4)='2015' \
    and "Landing _Outcome" = "Failure (drone ship)"
```

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

month	Landing Outcome	Booster Version	Launch Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

SQLLite does not support monthnames. So substr(Date, 4, 2) was used as month to get the months and substr(Date,7,4)='2015' for year.

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

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%sql select "Date", "Landing _Outcome" as "Landing Outcome", \
    count("Landing _Outcome") as "Total" from SPACEXTBL \
    where substr(Date,7,4) || substr(Date,4,2) || substr(Date,1,2) between '20100604' and '20170320' \
    and substr("Landing _Outcome",1,8)="Success " \
    group by "Landing _Outcome" \
    order by 3 desc
```

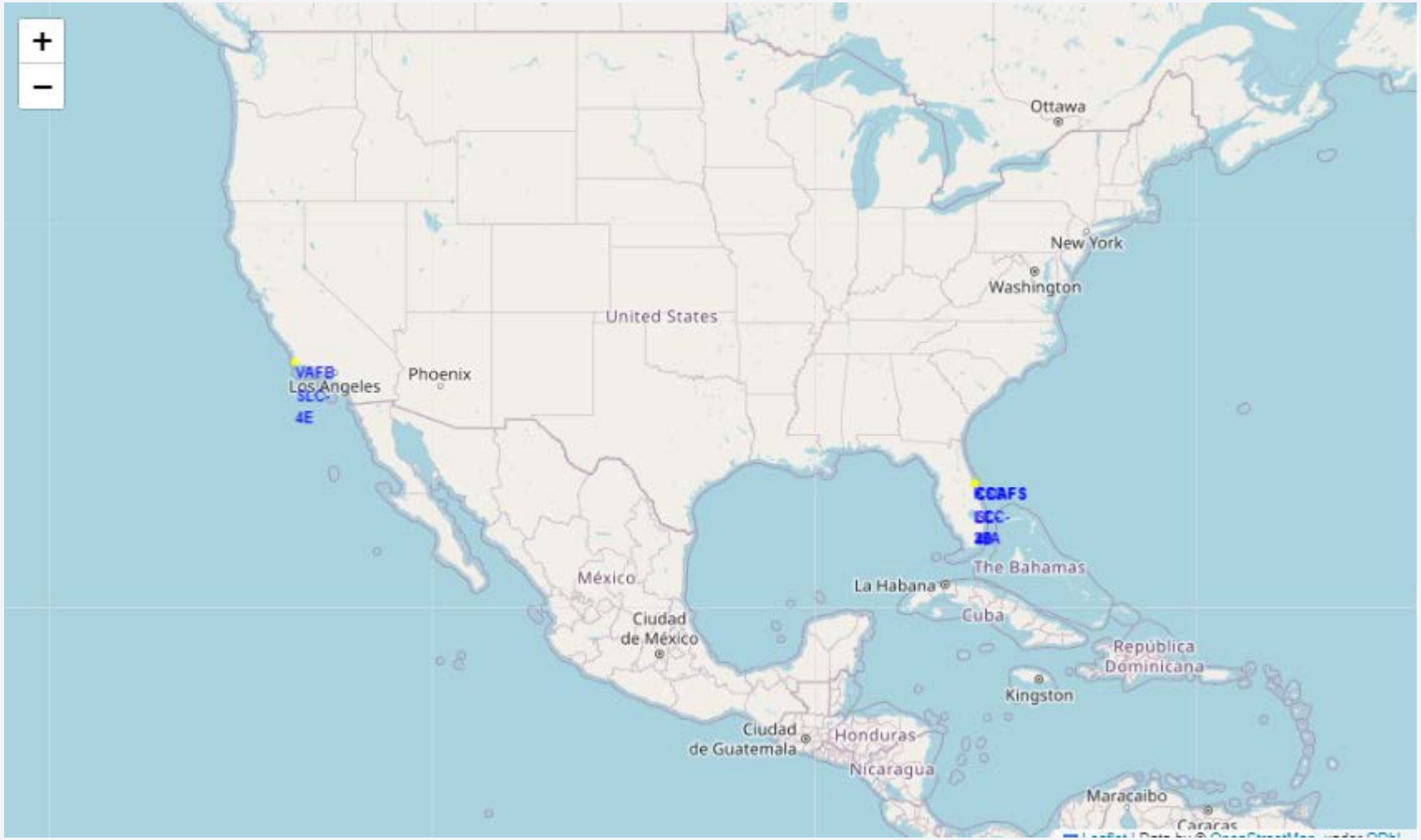
```
 * sqlite:///my_data1.db
Done.
```

Date	Landing Outcome	Total
08-04-2016	Success (drone ship)	5
22-12-2015	Success (ground pad)	3

We can see that between the dates of 2010-06-04 and 2017-03-20, there were 5 successful landings on a drone ship and 3 successful landings on a ground pad

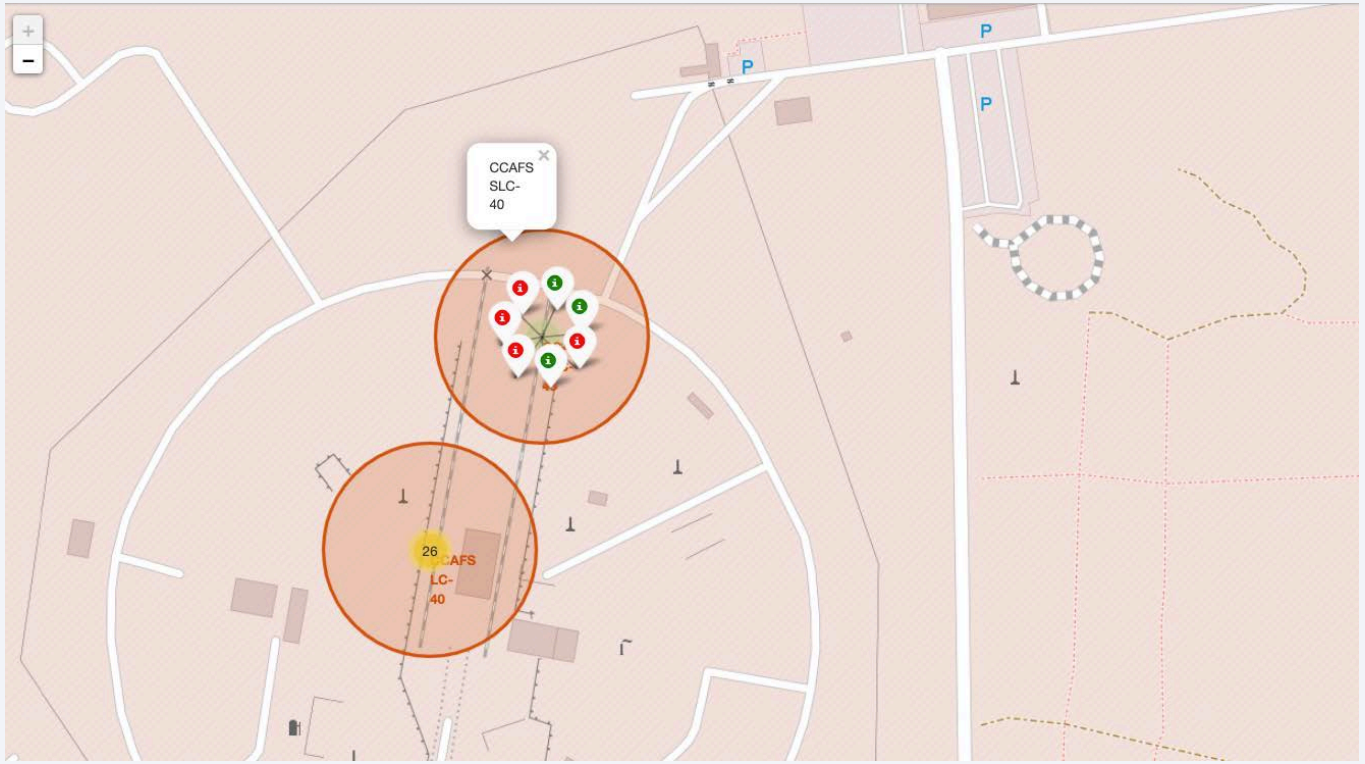

Section 3
Launch Sites
Proximities Analysis

Launch Site Location



Launch Site locations on the East and West cost of continental USA

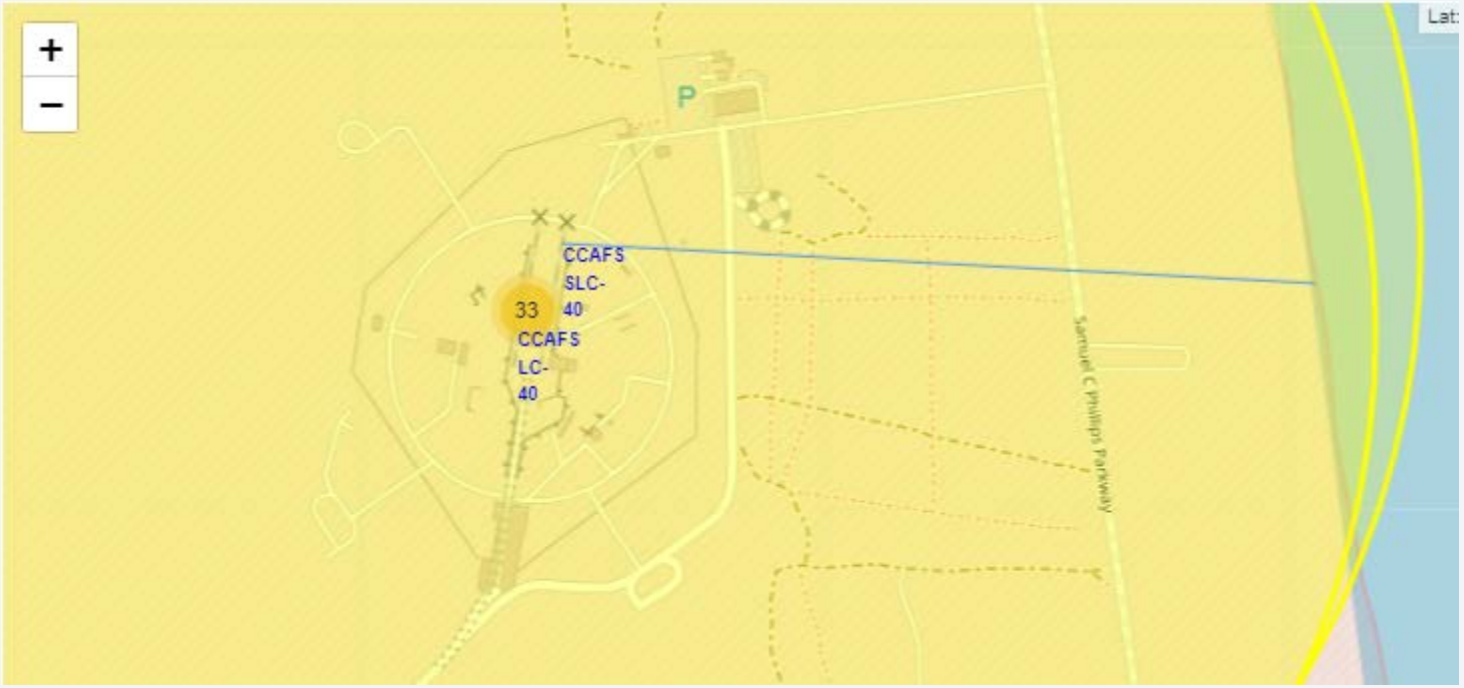
Cluster Markers



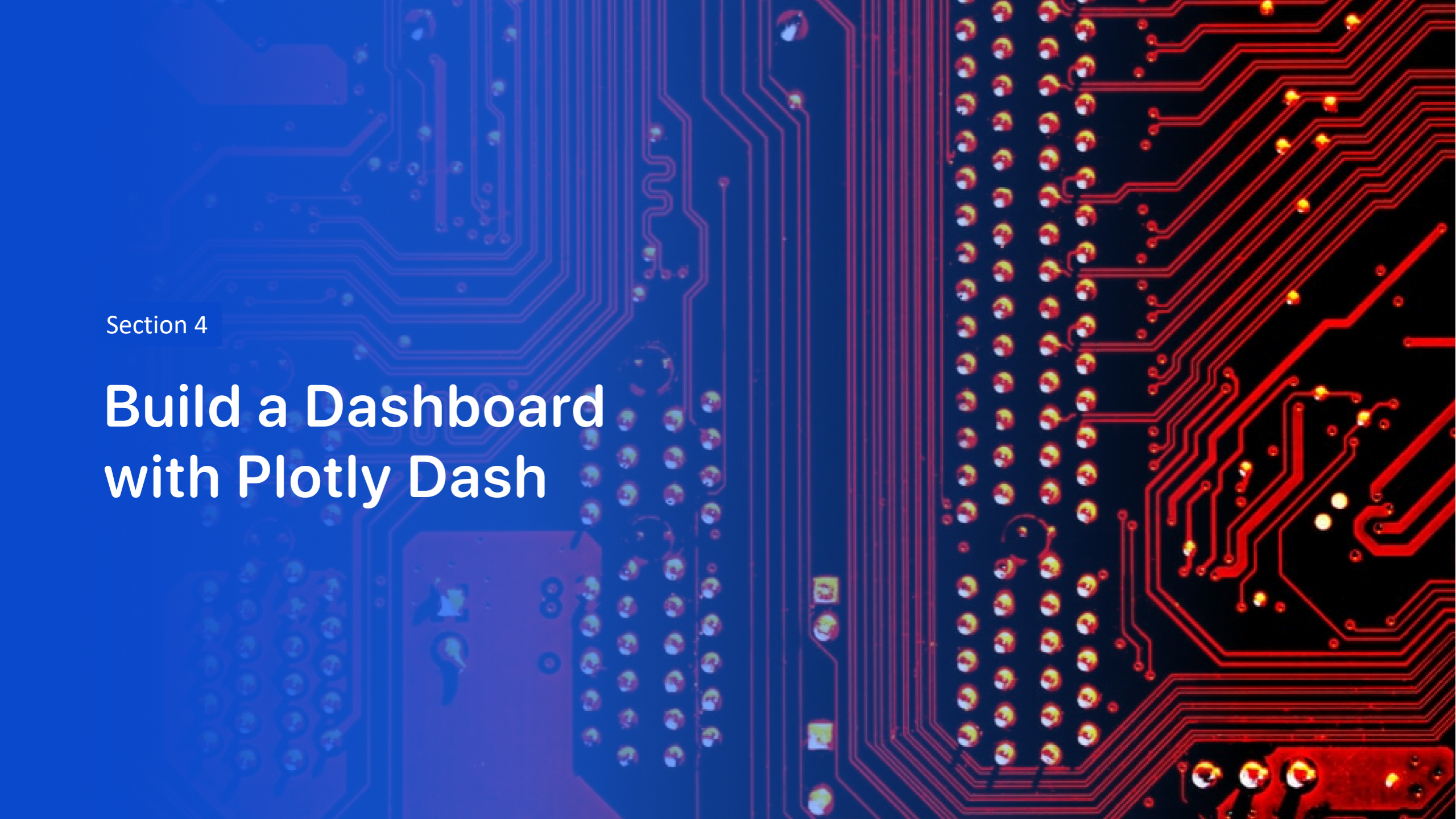
From the coloured labelled markers in marker clusters, we can easily identify which launch sites have relatively high success rates.

Green shows successful launches and red shows unsuccessful launches.

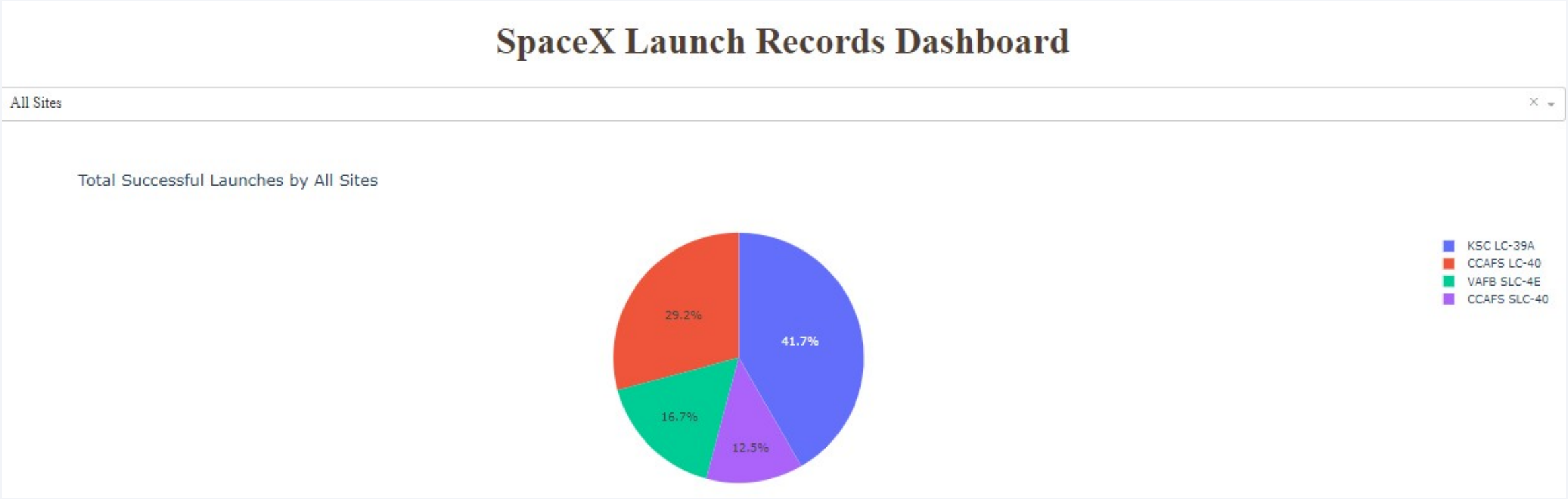
Distance to POI



- Are launch sites in close proximity to railways? no
- Are launch sites in close proximity to highways? no
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

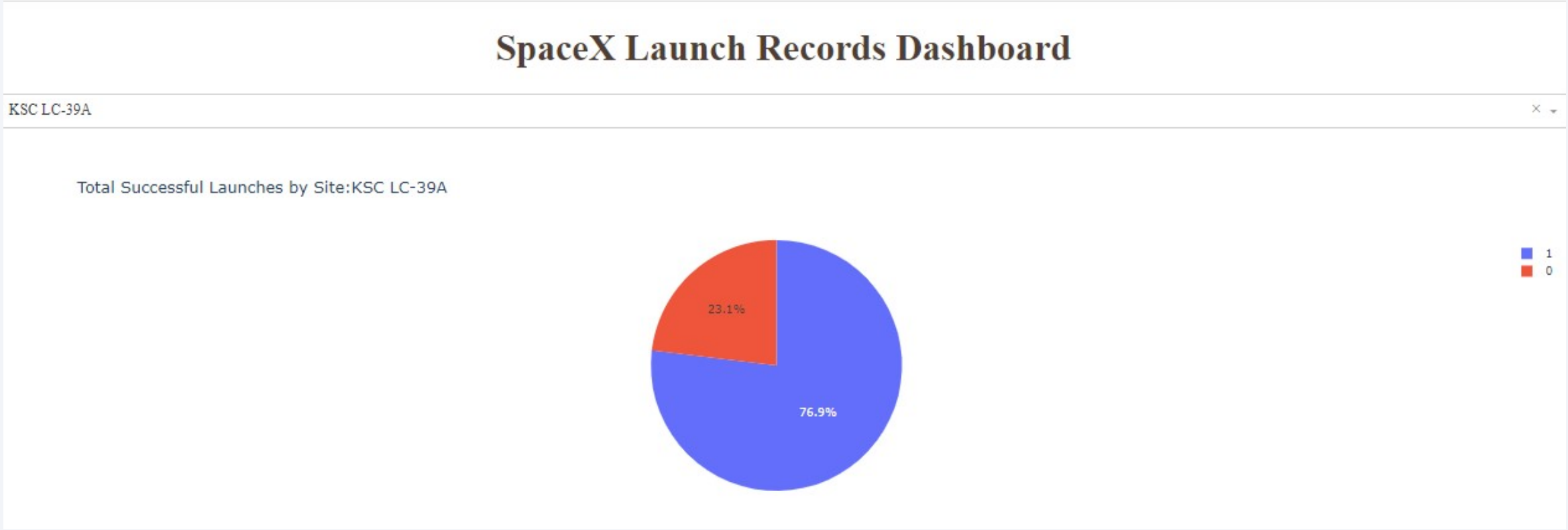

Section 4
Build a Dashboard with Plotly Dash

Total Successful Launches by All Sites



We can see that KSC LC-39A has the most successful launches from all the sites at 41.7%

Total Successful Launches from KSC LC-39A

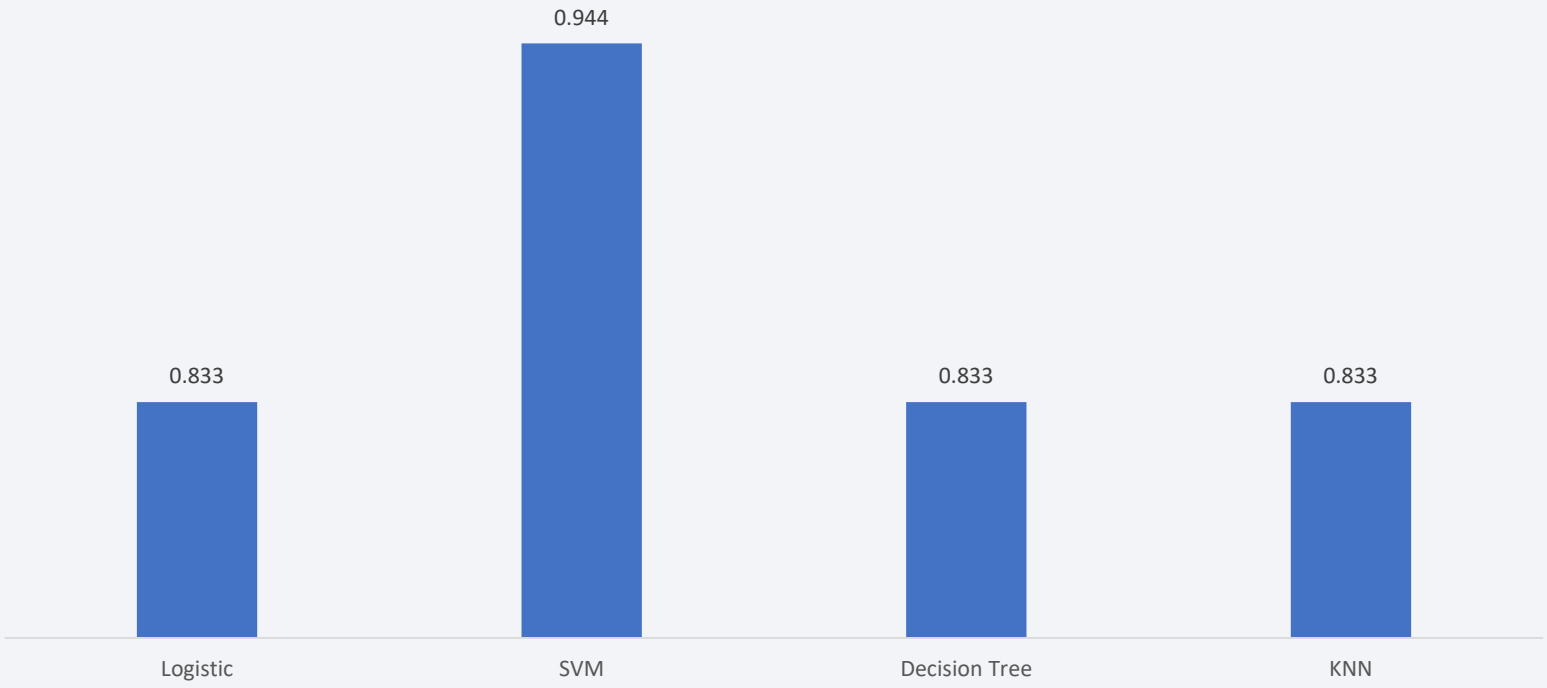


The KSC LC-39A launch site has a 76.9% successful launch rate and 23.1% failure rate.

Section 5

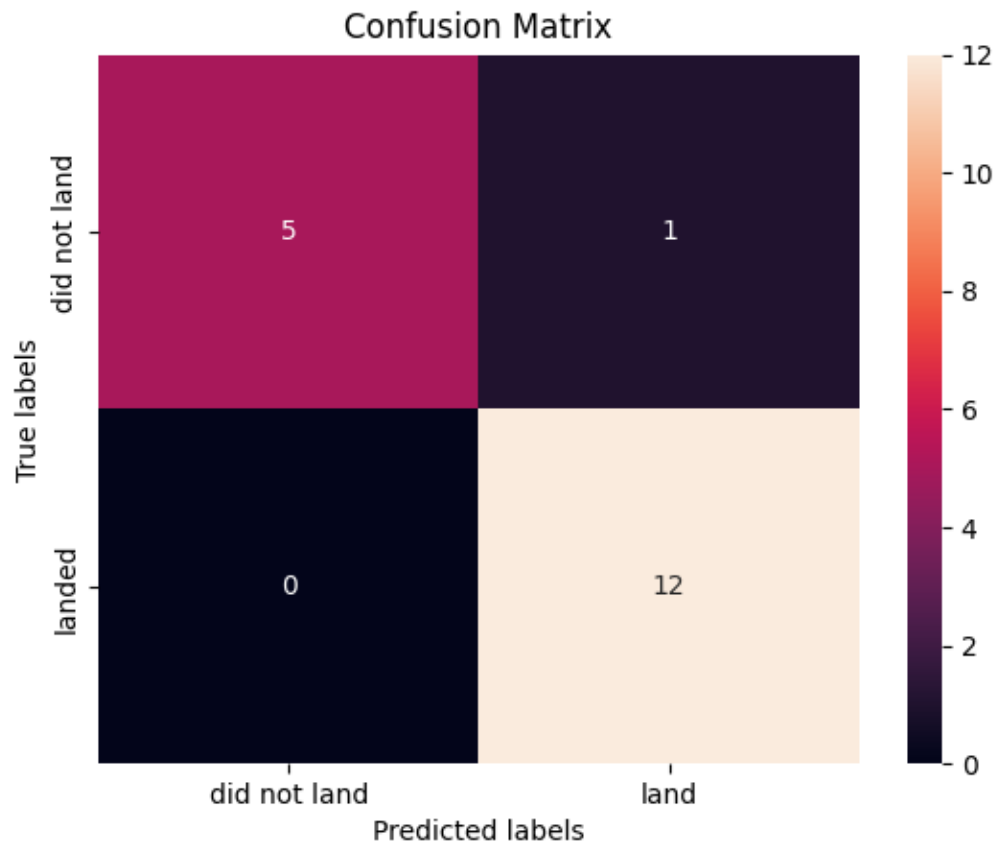
Predictive Analysis (Classification)

Classification Accuracy



SVM has the greatest accuracy when scoring on the test data at 0.944. The other models all returned a 0.833 score.

Confusion Matrix



Examining the confusion matrix, we see that SVM it does a very job at predicting both launches that land the first stage as well as those that did no land.

There is only 1 false positive.

Conclusions

- The more launches there are, the greater the success rate over the years.
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS orbits.
- Orbits GEO, HEO, SSO, ES-L1 has the best success rate.
- The KSC LC-39A site has the most successful launches, but if the mass is above 10,000 kg the success rate is 100% at the site CCAFS LC-40.
- The SVM classifier is the best machine learning algorithm for this dataset

Thank you!