
# Winning Space Race with Data Science

Pranali Patil
10/10/2023

https://github.com/pranalipatil1208/ds-capstone

# Outline


Executive Summary

Introduction

Methodology

Results

Conclusion

Appendix


# Executive Summary

## Summary of Methodologies:

- This project follows these steps:
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis
  - Interactive Visual Analytics
  - Predictive Analysis (Classification)

## Summary of Results:

- This project produced the following outputs and visualizations:
  - Exploratory Data Analysis (EDA) results
  - Geospatial analytics
  - Interactive dashboard
  - Predictive analysis of classification models

# Introduction

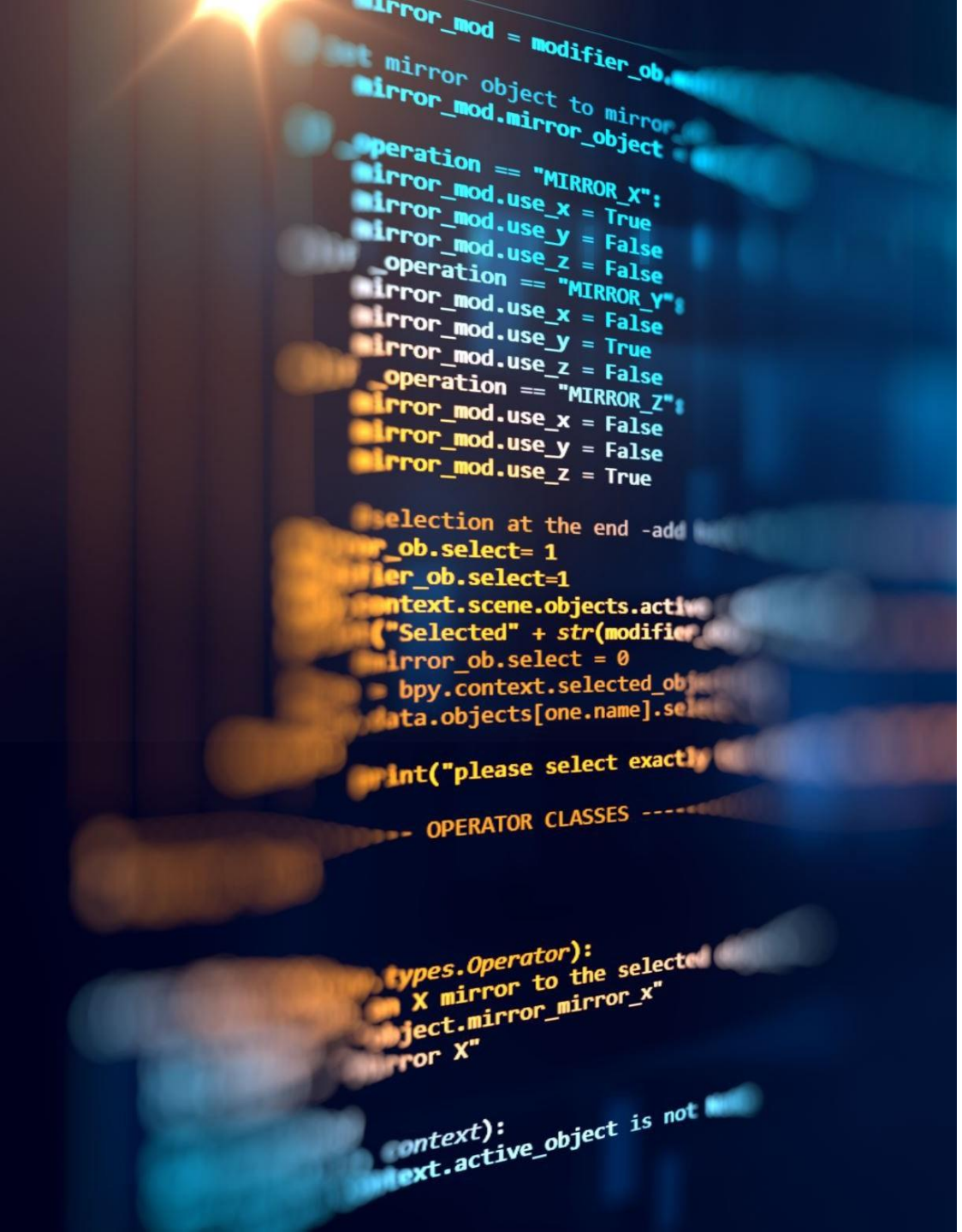
- SpaceX launches Falcon 9 rockets at a cost of around $62m. This is considerably cheaper than other providers (which usually cost upwards of $165m), and much of the savings are because SpaceX can land, and then re-use the first stage of the rocket.
- If we can make predictions on whether the first stage will land, we can determine the cost of a launch, and use this information to assess whether or not an alternate company should bid and SpaceX for a rocket launch.
- This project will ultimately predict if the Space X Falcon 9 first stage will land successfully.

Section 1

# Methodology

# Methodology

1. Data Collection
   - Making GET requests to the SpaceX REST API
   - Web Scraping
2. Data Wrangling
   - Using the `.fillna()` method to remove NaN values
   - Using the `.value_counts()` method to determine the following:
     - Number of launches on each site
     - Number and occurrence of each orbit
     - Number and occurrence of mission outcome per orbit type
   - Creating a landing outcome label that shows the following:
     - 0 when the booster did not land successfully
     - 1 when the booster did land successfully
3. Exploratory Data Analysis
   - Using SQL queries to manipulate and evaluate the SpaceX dataset
   - Using Pandas and Matplotlib to visualize relationships between variables, and determine patterns
4. Interactive Visual Analytics
   - Geospatial analytics using Folium
   - Creating an interactive dashboard using Plotly Dash
5. Data Modelling and Evaluation
   - Using Scikit-Learn to:
     - Pre-process (standardize) the data
     - Split the data into training and testing data using `train_test_split`
     - Train different classification models
     - Find hyperparameters using `GridSearchCV`
   - Plotting confusion matrices for each classification model
   - Assessing the accuracy of each classification model




# Data Collection

Data collection process involved a combination of API requests from Space X public API and web scraping data from a table in Space X's Wikipedia entry.

The next slide will show the flowchart of data collection from API and the one after will show the flowchart of data collection from webscraping.

Space X API Data Columns:

- FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, Flights, GridFins,
- Reused, Legs, LandingPad, Block, ReusedCount, Serial, Longitude, Latitude.

Wikipedia Webscrape Data Columns:

- Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Version Booster, Booster landing, Date, Time.

# Data Collection - SpaceX API

Using the SpaceX API to retrieve data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

1
- Make a GET response to the SpaceX REST API
- Convert the response to a .json file then to a Pandas DataFrame

2
- Use custom logic to clean the data (see Appendix)
- Define lists for data to be stored in
- Call custom functions (see Appendix) to retrieve data and fill the lists
- Use these lists as values in a dictionary and construct the dataset

3
- Create a Pandas DataFrame from the constructed dictionary dataset

4
- Filter the DataFrame to only include Falcon 9 launches
- Reset the FlightNumber column
- Replace missing values of PayloadMass with the mean PayloadMass value

```
#Global variables
BoosterVersion = []
PayloadMass = []
Orbit = []
LaunchSite = []
Outcome = []
Flights = []
GridFins = []
Reused = []
Legs = []
LandingPad = []
Block = []
ReusedCount = []
Serial = []
Longitude = []
Latitude = []
```

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

```
the list has now been update
```

```
BoosterVersion[0:5]
```

```
['Falcon 1', 'Falcon 1',

we can apply the rest of the fu
```

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

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

```
# Call getCoreData
getCoreData(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}
```

# Data Collection - Scraping

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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

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

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

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

Place your flowchart of web scraping here

```
# Use soup.title attribute
soup.title
```

```
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']=[]
```

Web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.

1. Request the HTML page from the static URL
   - Assign the response to an object
2. Create a BeautifulSoup object from the HTML response object
   - Find all tables within the HTML page
3. Collect all column header names from the tables found within the HTML page
4. Use the column names as keys in a dictionary
   - Use custom functions and logic to parse all launch tables (see Appendix) to fill the dictionary values
5. Convert the dictionary to a Pandas DataFrame ready for export

Github:https://github.com/IbtihalAlbalawi/testrepo/blob/child_branch/jupyter-labs-webscraping(1).ipynb

# Data Wrangling- Pandas

**Context:**

- **The SpaceX dataset contains several Space X launch facilities, and each location is in the LaunchSite column.**
- **Each launch aims to a dedicated orbit, and some of the common orbit types are shown in the figure below. The orbit type is in the Orbit column.**

**Initial Data Exploration:**

- **Using the .value_counts() method to determine the following:**
  - **Number of launches on each site**
  - **Number and occurrence of each orbit**
  - **Number and occurrence of landing outcome per orbit type**

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

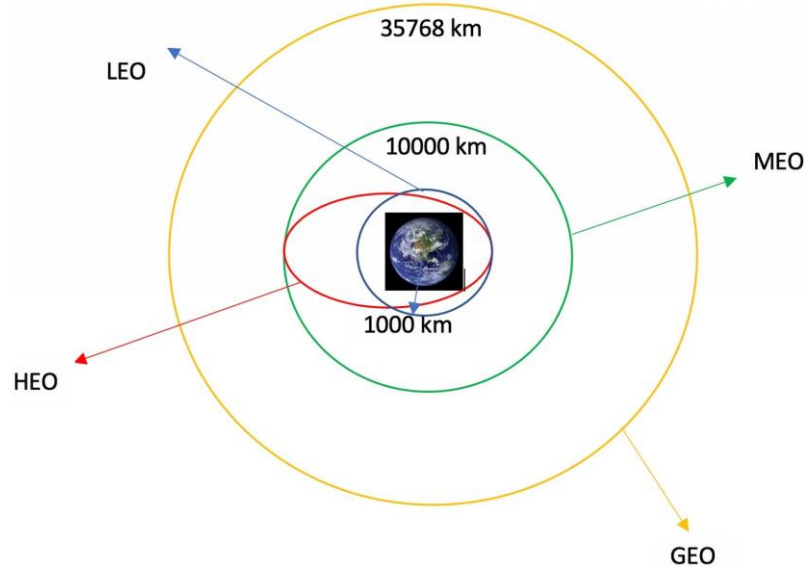
```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

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

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

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

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



# Data Wrangling-Pandas

Context:

- The landing outcome is shown in the `Outcome` column:

`True Ocean` – the mission outcome was successfully landed to a specific region of the ocean

`False Ocean` – the mission outcome was unsuccessfully landed to a specific region of the ocean.

`True RTLS` – the mission outcome was successfully landed to a ground pad

`False RTLS` – the mission outcome was unsuccessfully landed to a ground pad.

`True ASDS` – the mission outcome was successfully landed to a drone ship

`False ASDS` – the mission outcome was unsuccessfully landed to a drone ship.

`None ASDS` and `None None` – these represent a failure to land.

Data Wrangling:

- To determine whether a booster will successfully land, it is best to have a binary column, i.e., where the value is 1 or 0, representing the success of the landing.
- This is done by:
  1. Defining a set of unsuccessful (bad) outcomes, `bad_outcome`
  2. Creating a list, `landing_class`, where the element is 0 if the corresponding row in `Outcome` is in the set `bad_outcome`, otherwise, it's 1.
  3. Create a `Class` column that contains the values from the list `landing_class`
  4. Export the DataFrame as a .csv file.

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

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

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

| | Class |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 1 |
| 7 | 1 |

```
df.head(5)
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
def onehot(item):
    if item in bad_outcomes:
        return 0
    else:
        return 1
landing_class = df["Outcome"].apply(onehot)
landing_class
```

```
0     0
1     0
2     0
3     0
4     0
     ..
85    1
86    1
87    1
88    1
89    1
Name: Outcome, Length: 90, dtype: int64
```

# EDA with Data Visualization

## SCATTER CHARTS

Scatter charts were produced to visualize the relationships between:

- Flight Number and Launch Site
- Payload and Launch Site
- Orbit Type and Flight Number
- Payload and Orbit Type

Scatter charts are useful to observe relationships, or correlations, between two numeric variables.

## LINE CHARTS

Line charts were produced to visualize the relationships between:

- Success Rate and Year (i.e. the launch success yearly trend)

Line charts contain numerical values on both axes, and are generally used to show the change of a variable over time.

## BAR CHARTS

A bar chart was produced to visualize the relationship between:

- Success Rate and Orbit Type

Bar charts are used to compare a numerical value to a categorical variable. Horizontal or vertical bar charts can be used, depending on the size of the data.

# EDA with SQL

- To gather some information about the dataset, some SQL queries were performed.
- The SQL queries performed on the data set were used to:
  - 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 the average payload mass carried by booster version F9 v1.1
  - List the date when the first successful landing outcome on a ground pad was achieved
  - List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
  - List the total number of successful and failed mission outcomes
  - List the names of the booster versions which have carried the maximum payload mass
  - List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
  - 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

# Build an Interactive Map with Folium

**The following steps were taken to visualize the launch data on an interactive map:**

### 1- Mark all launch sites on a map

- Initialise the map using a Folium Map object
- Add a folium.Circle and folium.Marker for each launch site on the launch map

### 2- Mark the success/failed launches for each site on a map

- As many launches have the same coordinates, it makes sense to cluster them together.
- Before clustering them, assign a marker colour of successful (class = 1) as green, and failed (class = 0) as red.
- To put the launches into clusters, for each launch, add a folium.Marker to the MarkerCluster() object.
- Create an icon as a text label, assigning the icon_color as the marker_colour determined previously.

### 3- Calculate the distances between a launch site to its proximities

- To explore the proximities of launch sites, calculations of distances between points can be made using the Lat and Long values.
- After marking a point using the Lat and Long values, create a folium.Marker object to show the distance.
- To display the distance line between two points, draw a folium.PolyLine and add this to the map.

# Build a Dashboard with Plotly Dash

- **The following plots were added to a Plotly Dash dashboard to have an interactive visualisation of the data:**

1. **Pie chart (`px.pie()`) showing the total successful launches per site**
   - This makes it clear to see which sites are most successful
   - The chart could also be filtered (using a `dcc.Dropdown()` object) to see the success/failure ratio for an individual site

2. **Scatter graph (`px.scatter()`) to show the correlation between outcome (success or not) and payload mass (kg)**
   - This could be filtered (using a `RangeSlider()` object) by ranges of payload masses
   - It could also be filtered by booster version

# Predictive Analysis (Classification)

The following steps were taking to develop, evaluate, and find the best performing classification model:

## Model Development

- To prepare the dataset for model development:
  - Load dataset
  - Perform necessary data transformations (standardise and pre-process)
  - Split data into training and test data sets, using `train_test_split()`
  - Decide which type of machine learning algorithms are most appropriate
- For each chosen algorithm:
  - Create a `GridSearchCV` object and a dictionary of parameters
  - Fit the object to the parameters
  - Use the training data set to train the model

## Model Evaluation

- For each chosen algorithm:
  - Using the output GridSearchCV object:
    - Check the tuned hyperparameters (`best_params_`)
    - Check the accuracy (`score` and `best_score_`)
  - Plot and examine the Confusion Matrix

## Finding the Best Classification Model

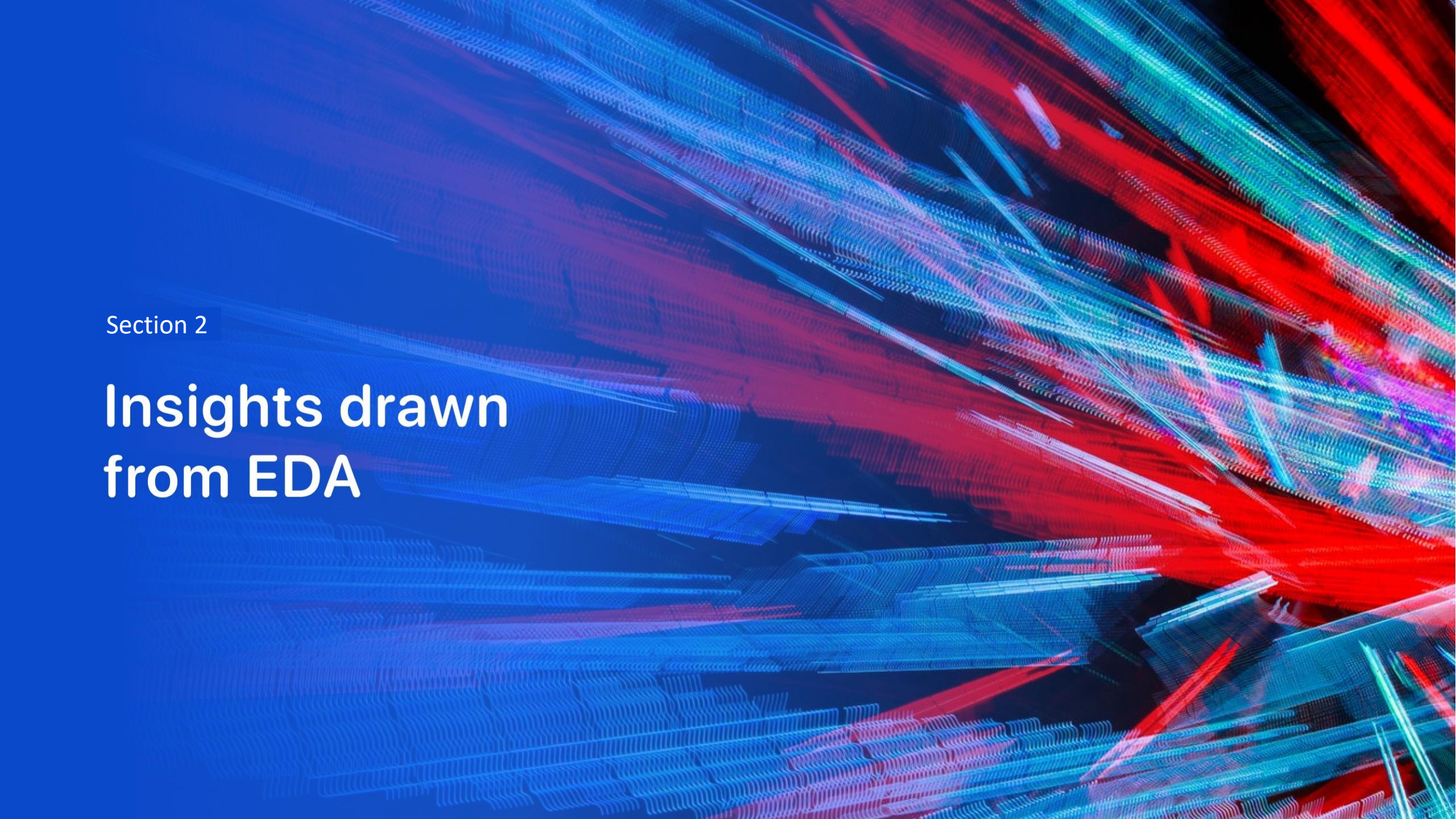
- Review the accuracy scores for all chosen algorithms
- The model with the highest accuracy score is determined as the best performing model

# Results

EXPLORATORY DATA ANALYSIS RESULTS

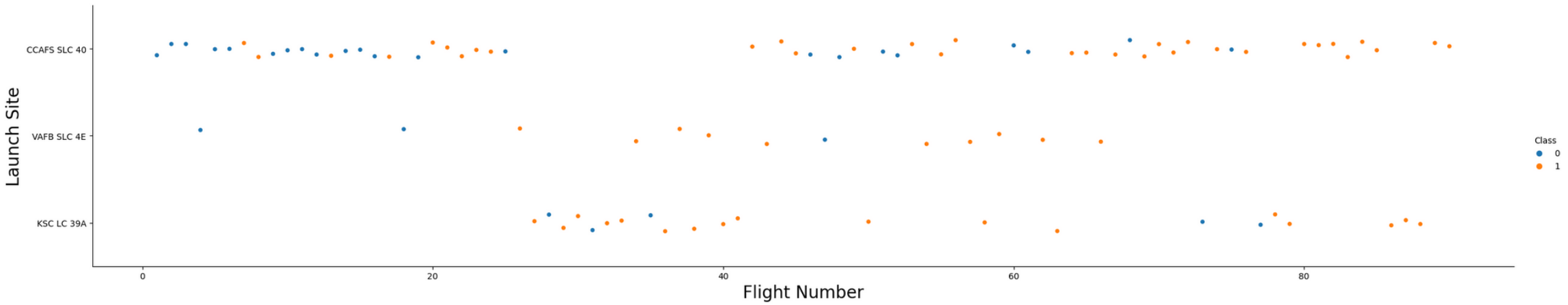
INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS

PREDICTIVE ANALYSIS RESULTS



Section 2

# Insights drawn from EDA




# Flight Number vs. Launch Site

**The scatter plot of Launch Site vs. Flight Number shows that:**

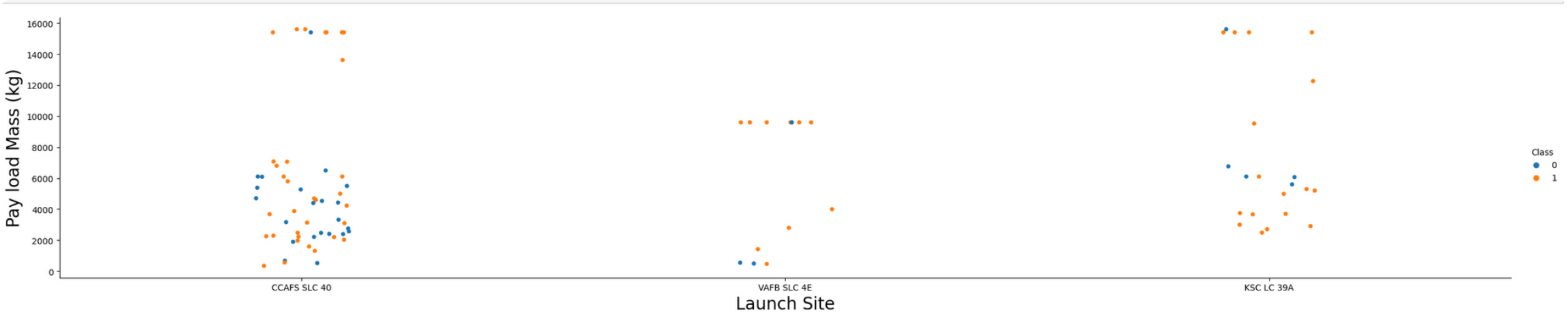
-As the number of flights increases, the rate of success at a launch site increases.

-Most of the early flights (flight numbers < 30) were launched from CCAFS SLC 40, and were generally unsuccessful.

-The flights from VAFB SLC 4E also show this trend, that earlier flights were less successful.

-No early flights were launched from KSC LC 39A, so the launches from this site are more successful.

-Above a flight number of around 30, there are significantly more successful landings (Class = 1).




# Payload vs. Launch Site

**The scatter plot of Launch Site vs. Payload Mass shows that:**

- **Above a payload mass of around 7000 kg, there are very few unsuccessful landings, but there is also far less data for these heavier launches.**
- **There is no clear correlation between payload mass and success rate for a given launch site.**
- **All sites launched a variety of payload masses, with most of the launches from CCAFS SLC 40 being comparatively lighter payloads (with some outliers).**

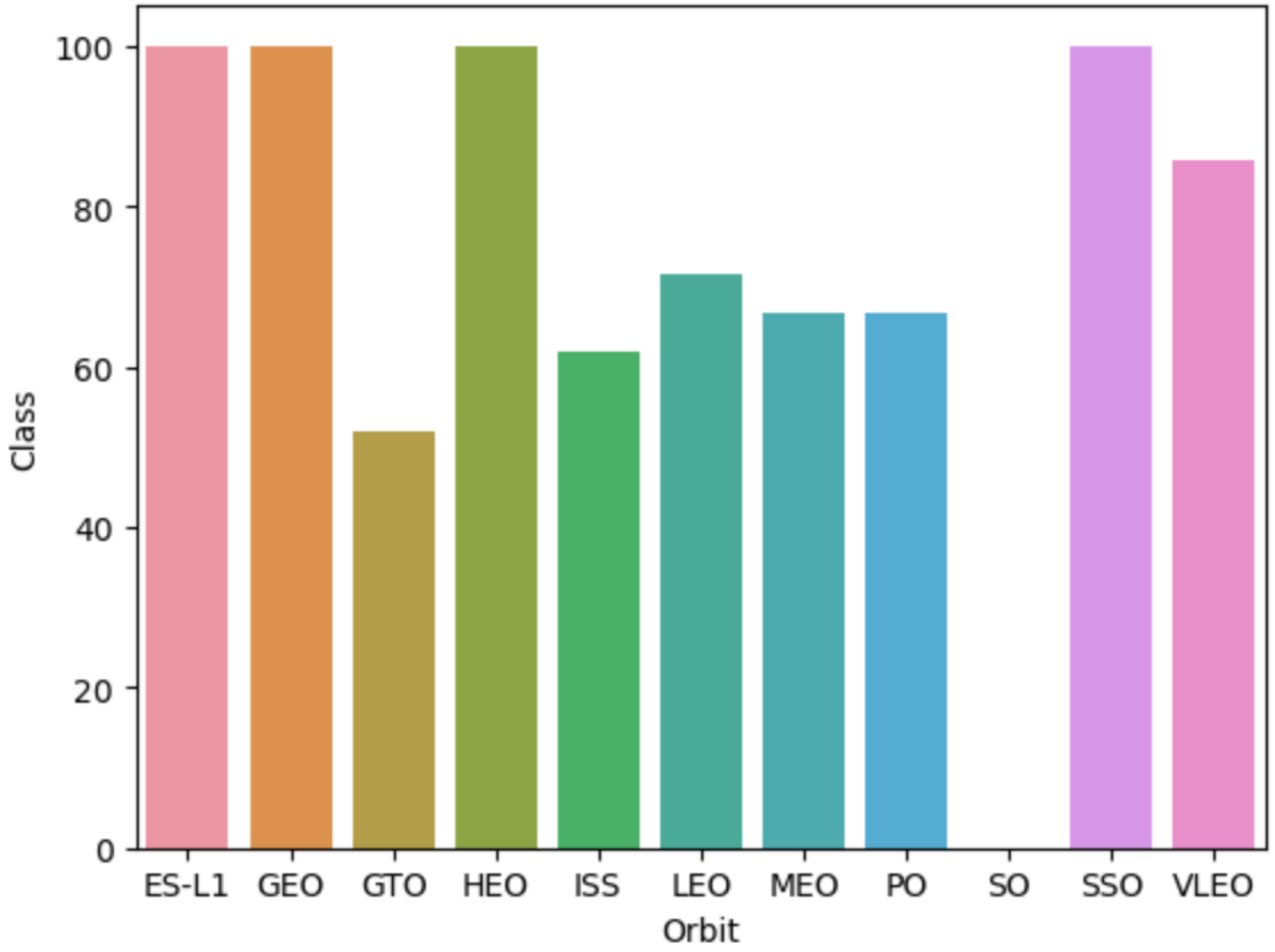
# Success Rate vs. Orbit Type

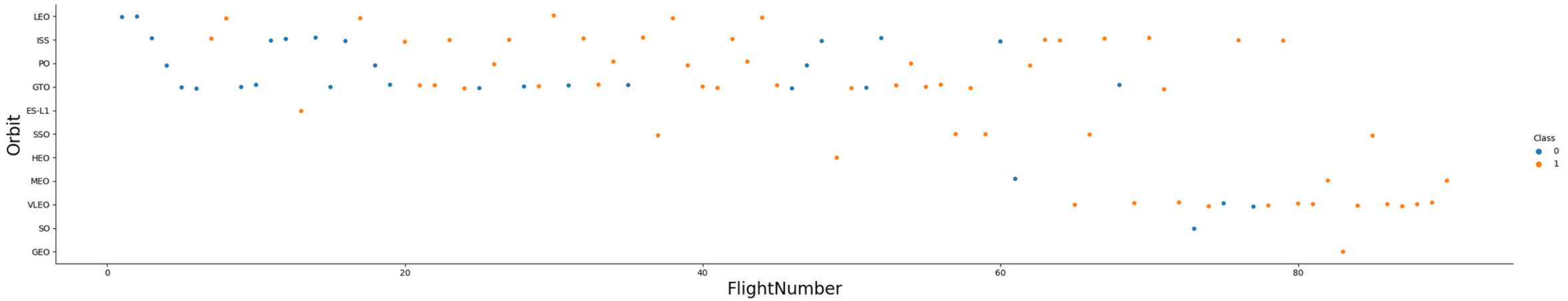
The bar chart of Success Rate vs. Orbit Type shows that the following orbits have the highest (100%) success rate:

- -ES-L1 (Earth-Sun First Lagrangian Point)
- -GEO (Geostationary Orbit)
- -HEO (High Earth Orbit)
- -SSO (Sun-synchronous Orbit)

The orbit with the lowest (0%) success rate is:

- -SO (Heliocentric Orbit)

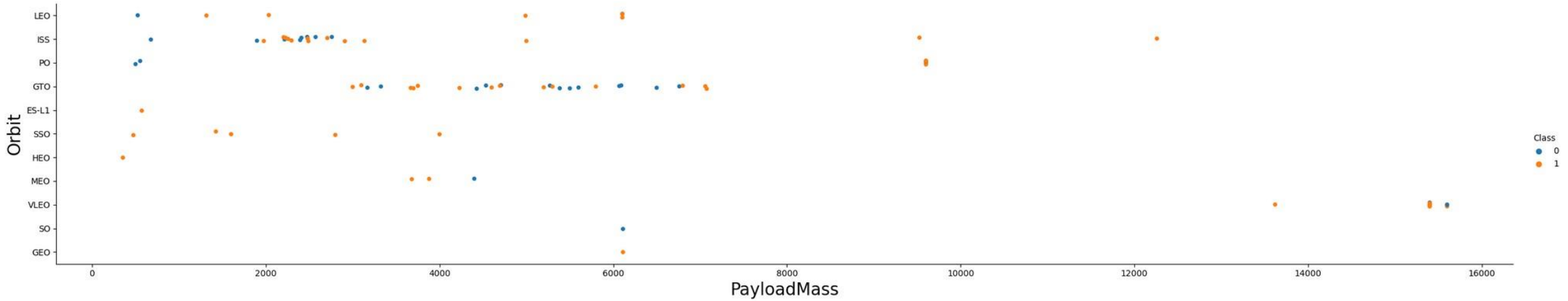





# Flight Number vs. Orbit Type

This scatter plot of Orbit Type vs. Flight number shows a few useful things that the previous plots did not, such as:

- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is little relationship between Flight Number and Success Rate for GTO.
- Generally, as Flight Number increases, the success rate increases. This is most extreme for LEO, where unsuccessful landings only occurred for the low flight numbers (early launches).




# Payload vs. Orbit Type

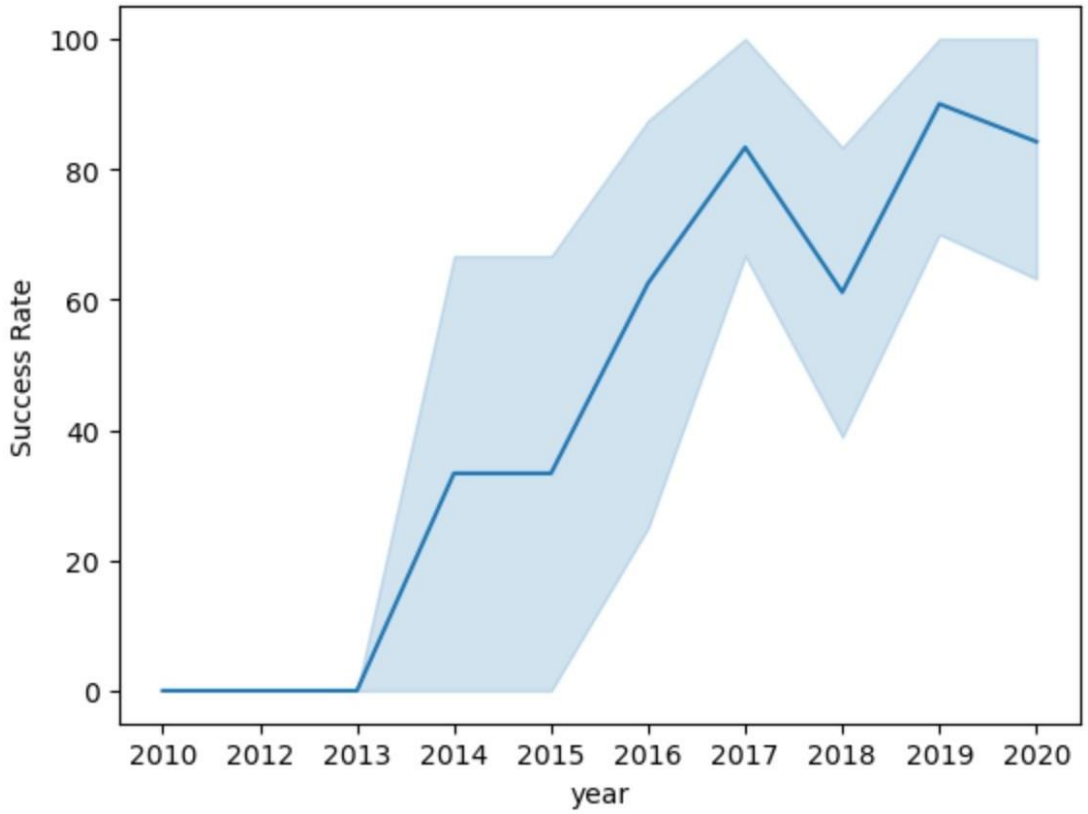
This scatter plot of Orbit Type vs. Payload Mass shows that:

- The following orbit types have more success with heavy payloads:
  - PO (although the number of data points is small)
  - ISS
  - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.

# Launch Success Yearly Trend

The line chart of yearly average success rate shows that:

- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.



# All Launch Site Names

Find the names of the unique launch sites.

```
%sql ibm_db_sa://yyy33800:dwNKg8J3L0IBd6CP@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:3
%sql SELECT Unique(LAUNCH_SITE) FROM SPACEXTBL;
```

The word UNIQUE returns only unique values from the LAUNCH_SITE column of the SPACEXTBL table.

# Launch Site Names Begin with 'CCA'

Find 5 records where launch sites begin with ‘CCA’.

```
%sql SELECT * \
    FROM SPACEXTBL \
    WHERE LAUNCH_SITE LIKE'CCA%' LIMIT 5;
```

| Launch_Site |
| --- |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |
| CCAFS LC-40 |

`LIMIT 5` fetches only 5 records, and the `LIKE` keyword is used with the wild card `‘CCA%’` to retrieve string values beginning with ‘CCA’.

# Total Payload Mass

Calculate the total payload carried by boosters from NASA.

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

| SUM(PAYLOAD_MASS__KG_) |
| --- |
| 45596 |

The SUM keyword is used to calculate the total of the LAUNCH column, and the SUM keyword (and the associated condition) filters the results to only boosters from NASA (CRS).

# Average Payload Mass by F9 v1.1

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

| AVG(PAYLOAD_MASS__KG_) |
| --- |
| 2928.4 |

The AVG keyword is used to calculate the average of the PAYLOAD_MASS__KG_ column, and the WHERE keyword (and the associated condition) filters the results to only the F9 v1.1 booster version.

# First Successful Ground Landing Date

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

```
%sql SELECT MIN(DATE) \
FROM SPACEXTBL \
WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

The MIN keyword is used to calculate the minimum of the DATE column, i.e. the first date, and the WHERE keyword (and the associated condition) filters the results to only the successful ground pad landings.

# 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 SELECT PAYLOAD \
FROM SPACEXTBL \
WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

The WHERE keyword is used to filter the results to include only those that satisfy both conditions in the brackets (as the AND keyword is also used). The BETWEEN keyword allows for 4000 < x < 6000 values to be selected.

# Total Number of Successful and Failure Mission Outcomes

Calculate the total number of successful and failure mission outcome.

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

| Mission_Outcome | total_number |
|---|---|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

The COUNT keyword is used to calculate the total number of mission outcomes, and the GROUPBY keyword is also used to group these results by the type of mission outcome.

# Boosters Carried Maximum Payload

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

```
%sql SELECT BOOSTER_VERSION \
FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

| 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 used here. The `SELECT` statement within the brackets finds the maximum payload, and this value is used in the `WHERE` condition. The `DISTINCT` keyword is then used to retrieve only distinct /unique booster versions.

# 2015 Launch Records

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

```
%sql SELECT substr(Date,4,2) as month, DATE,BOOSTER_VERSION, LAUNCH_SITE, [Landing _Outcome] \
FROM SPACEXTBL \
where [Landing _Outcome] = 'Failure (drone ship)' and substr(Date,7,4)='2015';
```

| Launch_Site | Landing _Outcome |
|---|---|
| CCAFS LC-40 | Failure (drone ship) |
| CCAFS LC-40 | Failure (drone ship) |

The WHERE keyword is used to filter the results for only failed landing outcomes, AND only for the year of 2015.

# 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 SELECT [Landing _Outcome], count(*) as count_outcomes \
FROM SPACEXTBL \
WHERE DATE between '04-06-2010' and '20-03-2017' group by [Landing _Outcome] order by count_outcomes DESC;
```

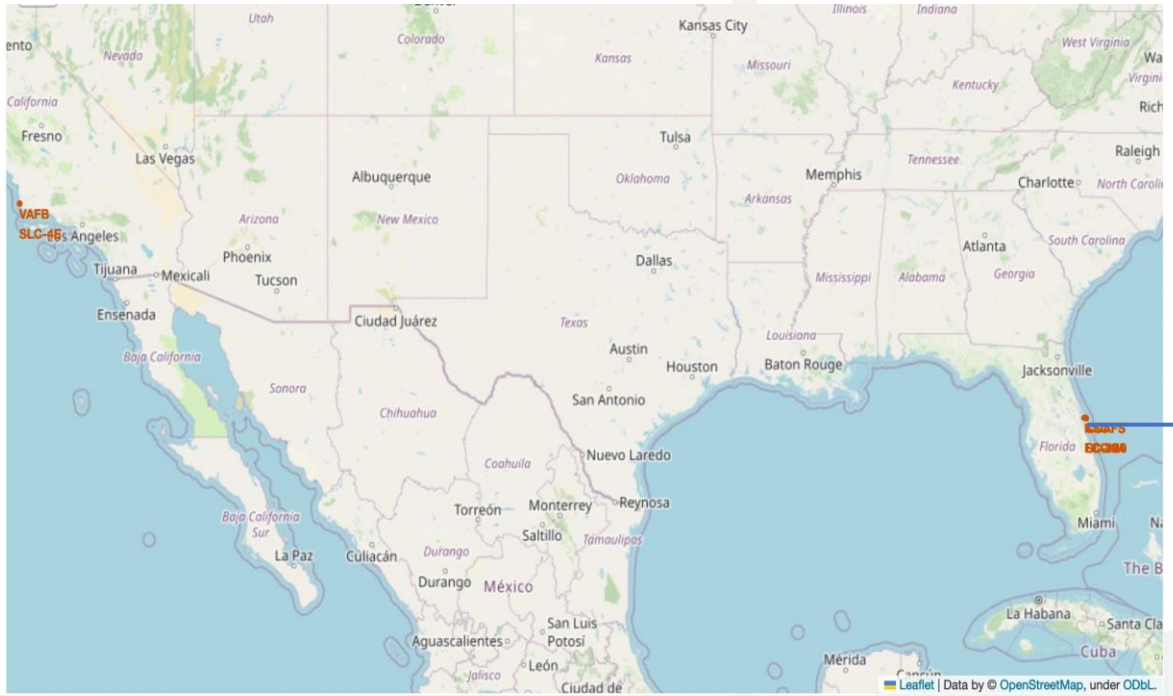
| Landing _Outcome | count_outcomes |
|---|---|
| Success | 20 |
| No attempt | 10 |
| Success (drone ship) | 8 |
| Success (ground pad) | 6 |
| Failure (drone ship) | 4 |
| Failure | 3 |
| Controlled (ocean) | 3 |
| Failure (parachute) | 2 |
| No attempt | 1 |

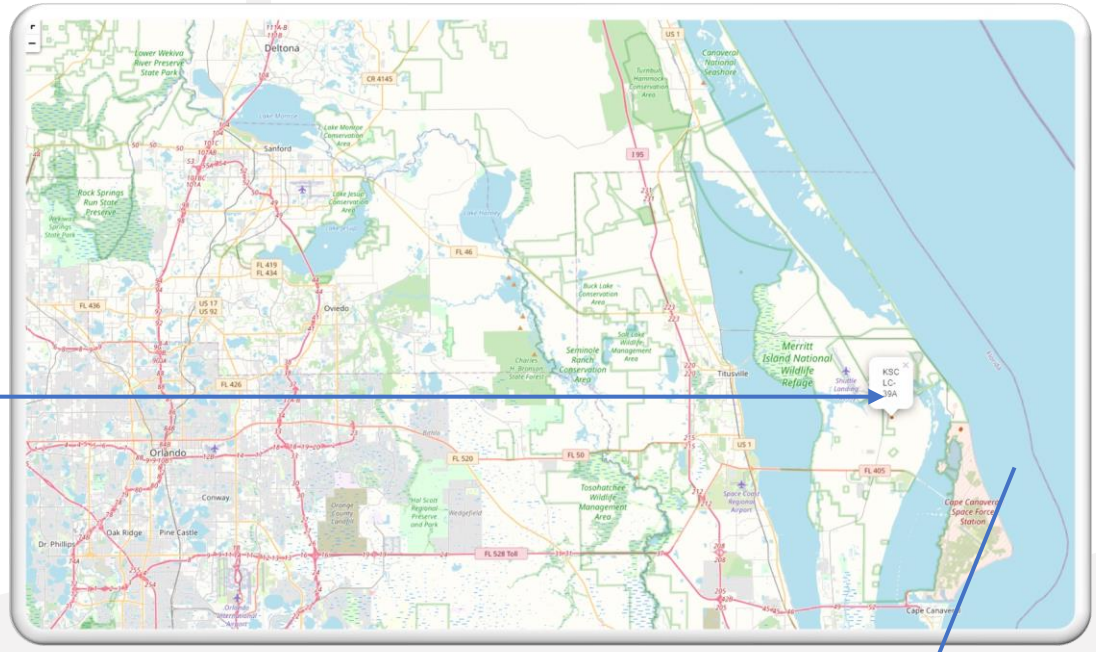
The `WHERE` keyword is used with the `BETWEEN` keyword to filter the results to dates only within those specified. The results are then grouped and ordered, using the keywords `GROUP BY` and `ORDER BY`, respectively, where `DESC` is used to specify the descending order.

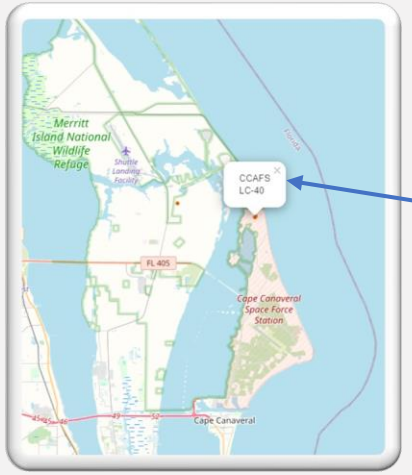
Section 3

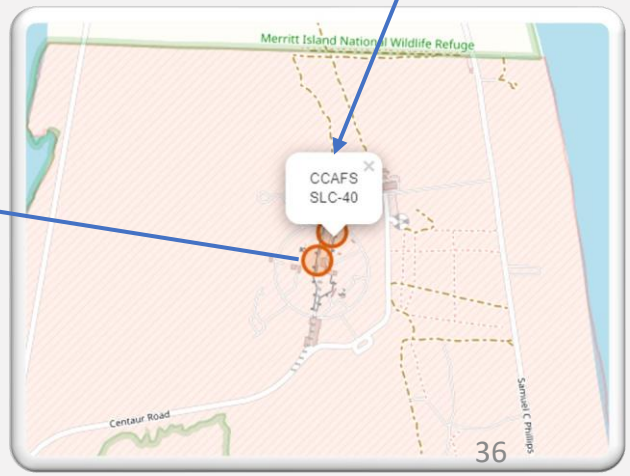
# Launch Sites Proximities Analysis

# ALL LAUNCH SITES ON A MAP



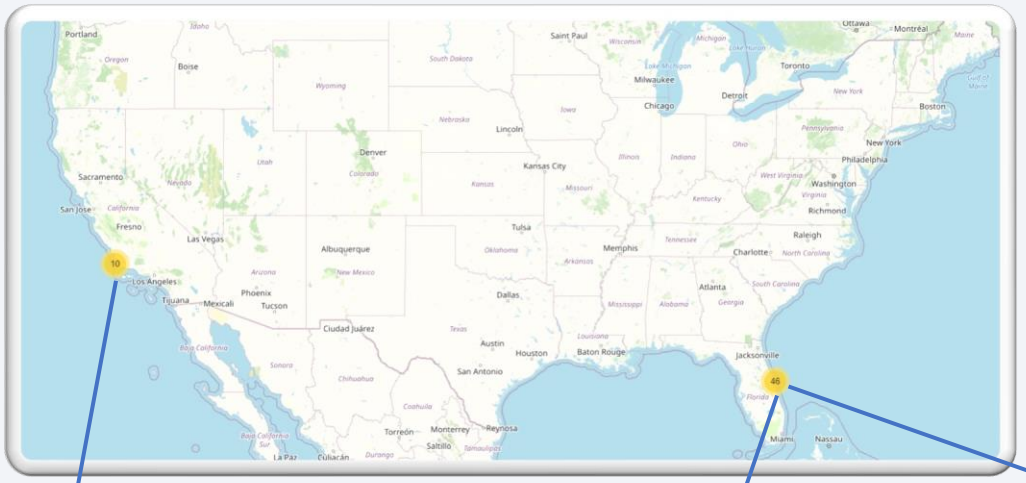



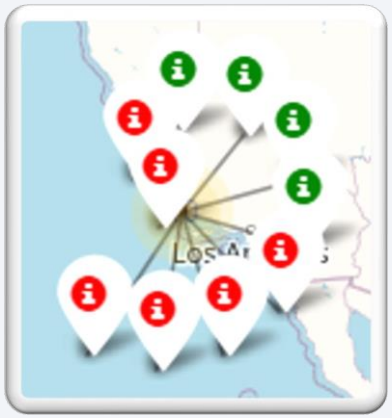

All SpaceX launch sites are on coasts of the United States of America, specifically Florida and California.

# SUCCESS/FAILED LAUNCHES FOR EACH SITE

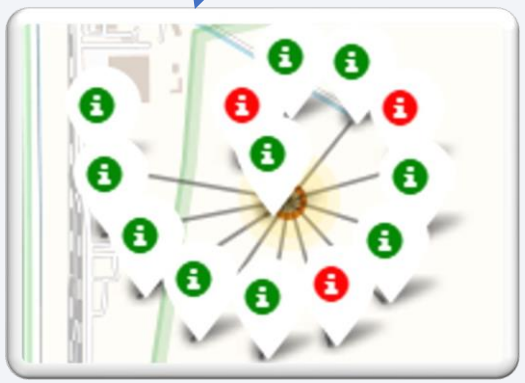



- Launches have been grouped into clusters, and annotated with green icons for successful launches, and red icons for failed launches.

VAFB SLC-4E

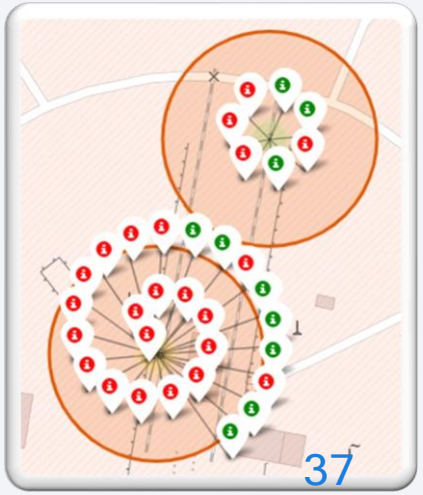


KSC LC-39A



CCAFS SLC-40 and CCAFS LC-40

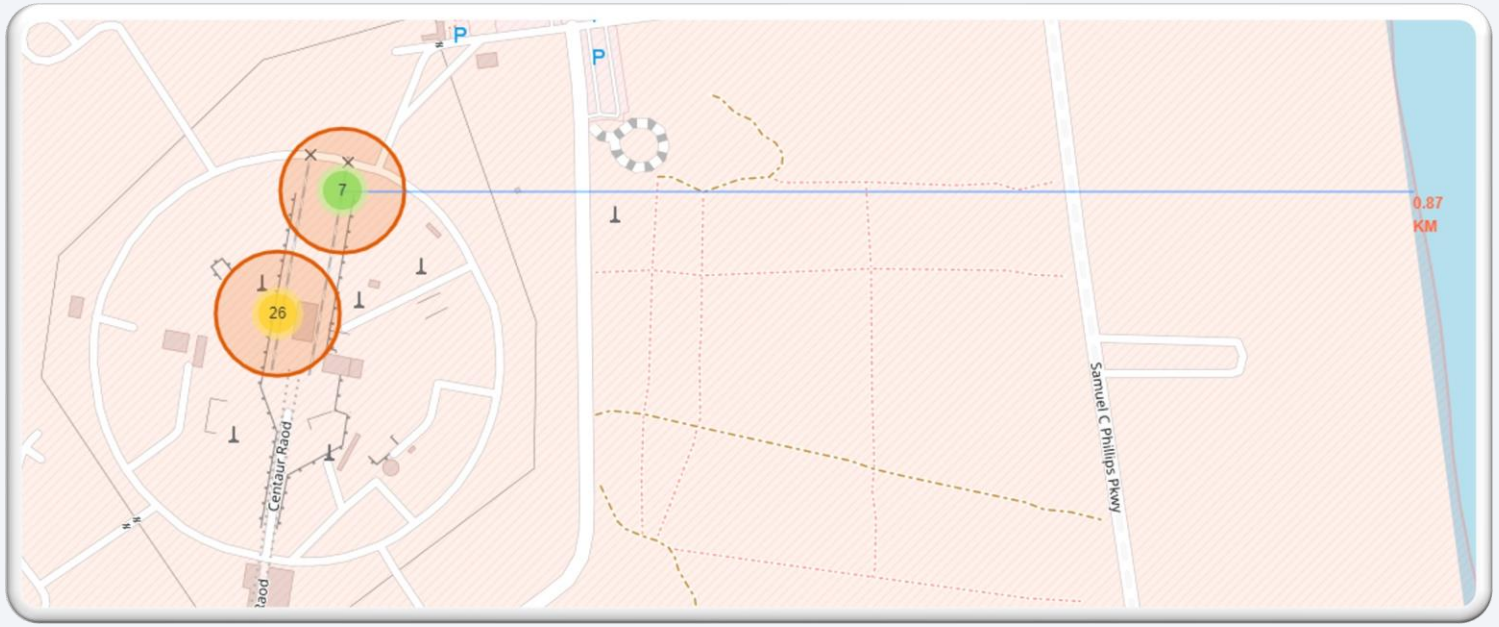

=

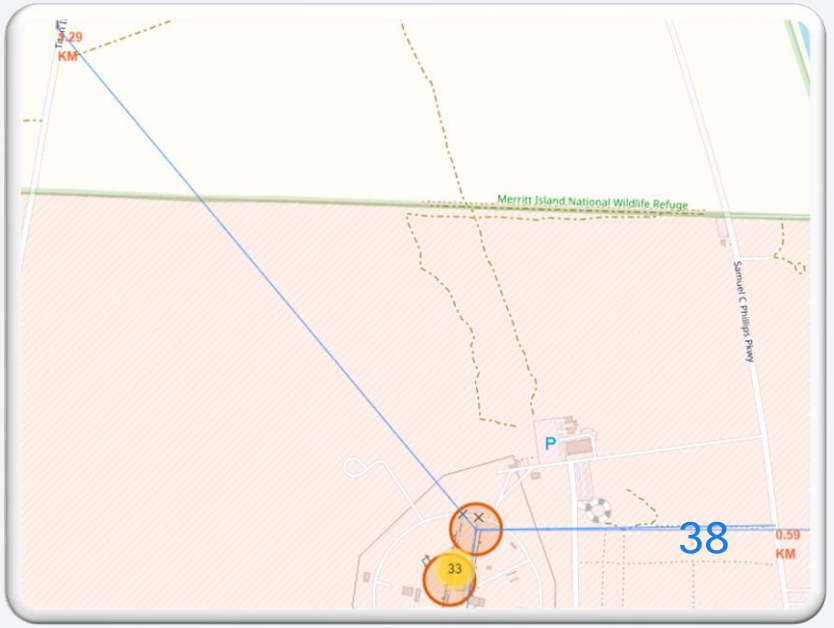


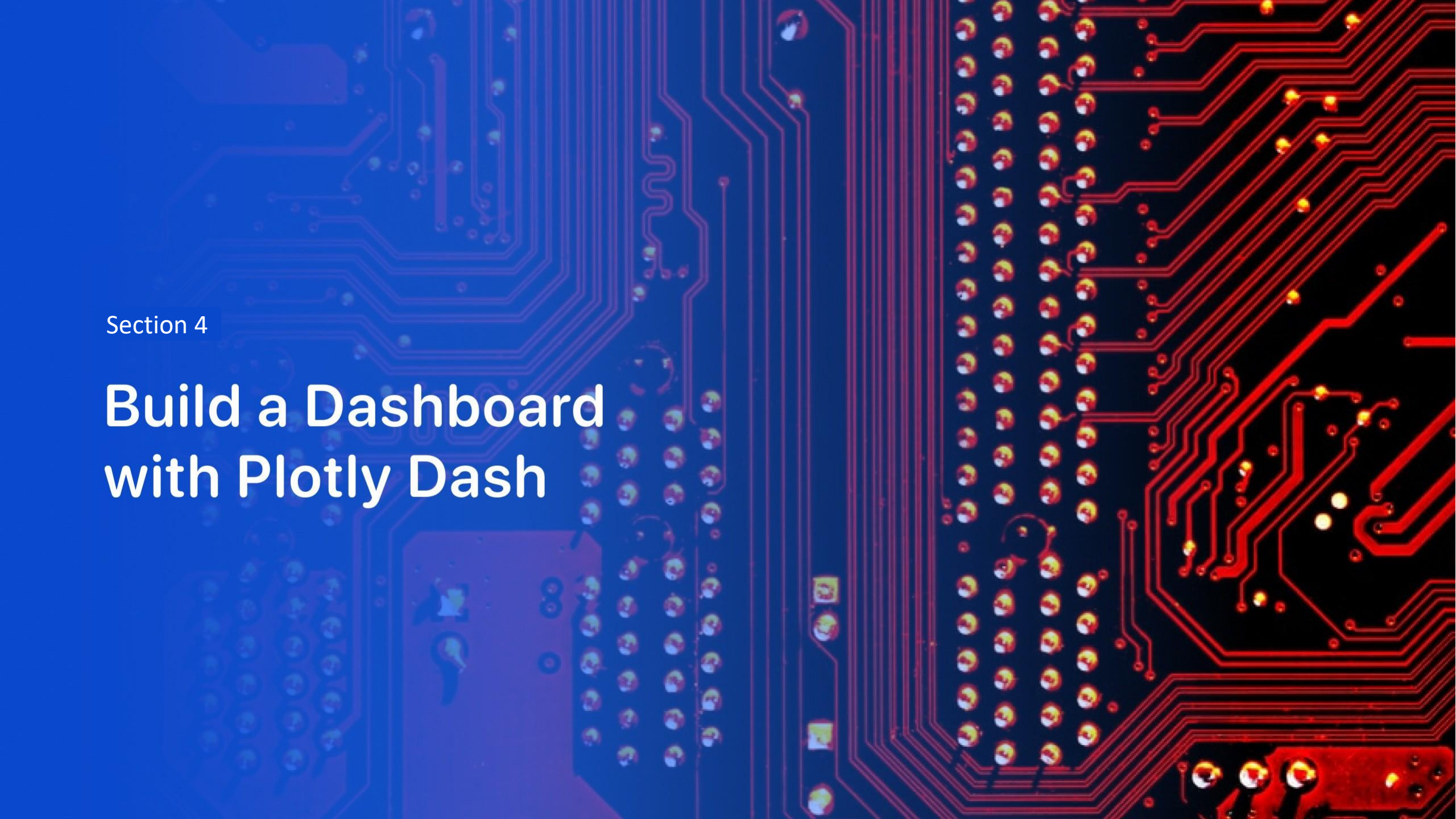
# PROXIMITY OF LAUNCH SITES TO OTHER POINTS OF INTEREST

Using the CCAFS SLC-40 launch site as an example site, we can understand more about the placement of launch sites.


- Are launch sites in close proximity to railways?
- YES. The coastline is only 0.87 km due East.
- Are launch sites in close proximity to highways?
- YES. The nearest highway is only 0.59km away.
- Are launch sites in close proximity to railways?
- YES. The nearest railway is only 1.29 km away.
- Do launch sites keep certain distance away from cities?
- YES. The nearest city is 51.74 km away.

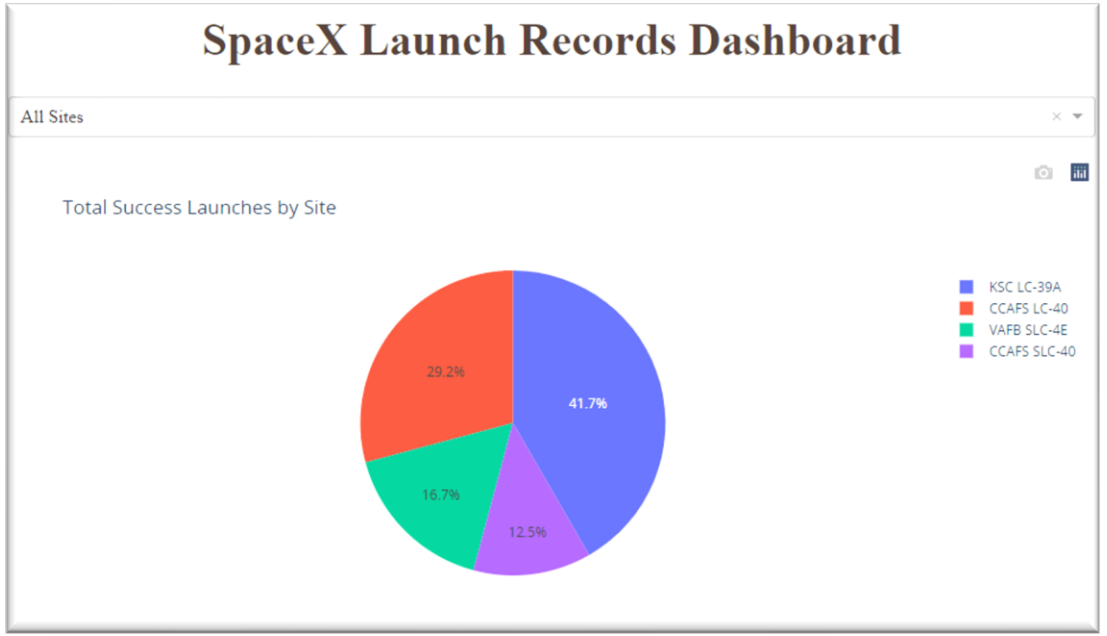



Section 4
Build a Dashboard with Plotly Dash

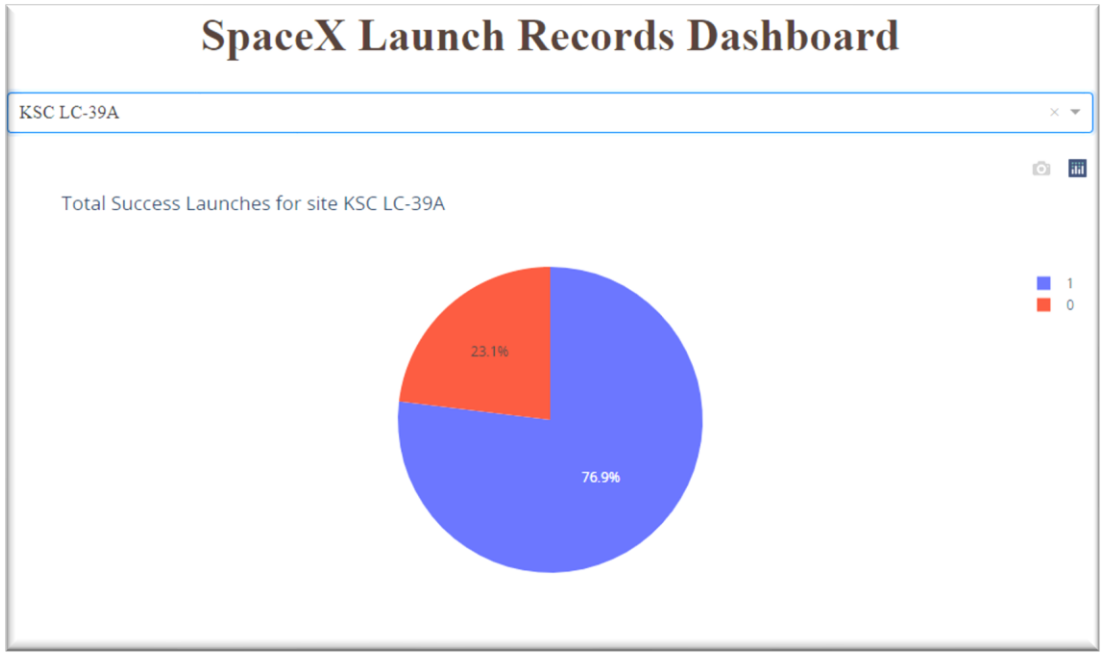
# Launch success count for all sites

The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches.



# Pie chart for the launch site with highest launch success ratio

The launch site KSC LC-39 A also had the highest rate of successful launches, with a 76.9% success rate.






- Plotting the launch outcome vs. payload for all sites shows a gap around 4000 kg, so it makes sense to split the data into 2 ranges:
  - 0 – 4000 kg (low payloads)
  - 4000 – 10000 kg (massive payloads)
- From these 2 plots, it can be shown that the success for massive payloads is lower than that for low payloads.
- It is also worth noting that some booster types (v1.0 and B5) have not been launched with massive payloads.



# Launch Outcome VS. Payload scatter plot for all sites

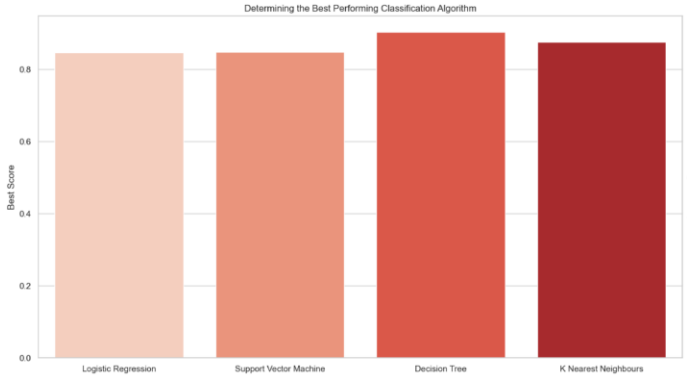
Section 5

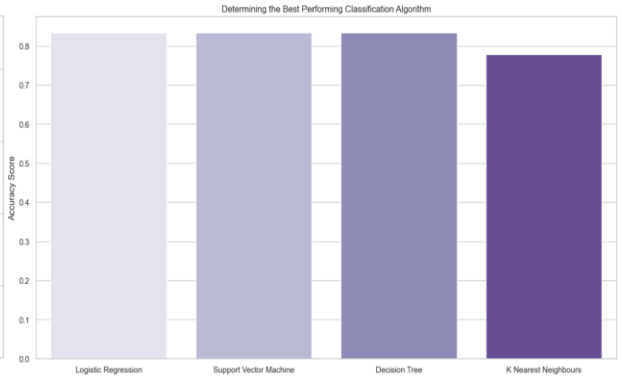
# Predictive Analysis (Classification)

# Classification Accuracy

Plotting the Accuracy Score and Best Score for each classification algorithm produces the following result:

- The Decision Tree model has the highest classification accuracy
  - The Accuracy Score is 94.44%
  - The Best Score is 90.36%

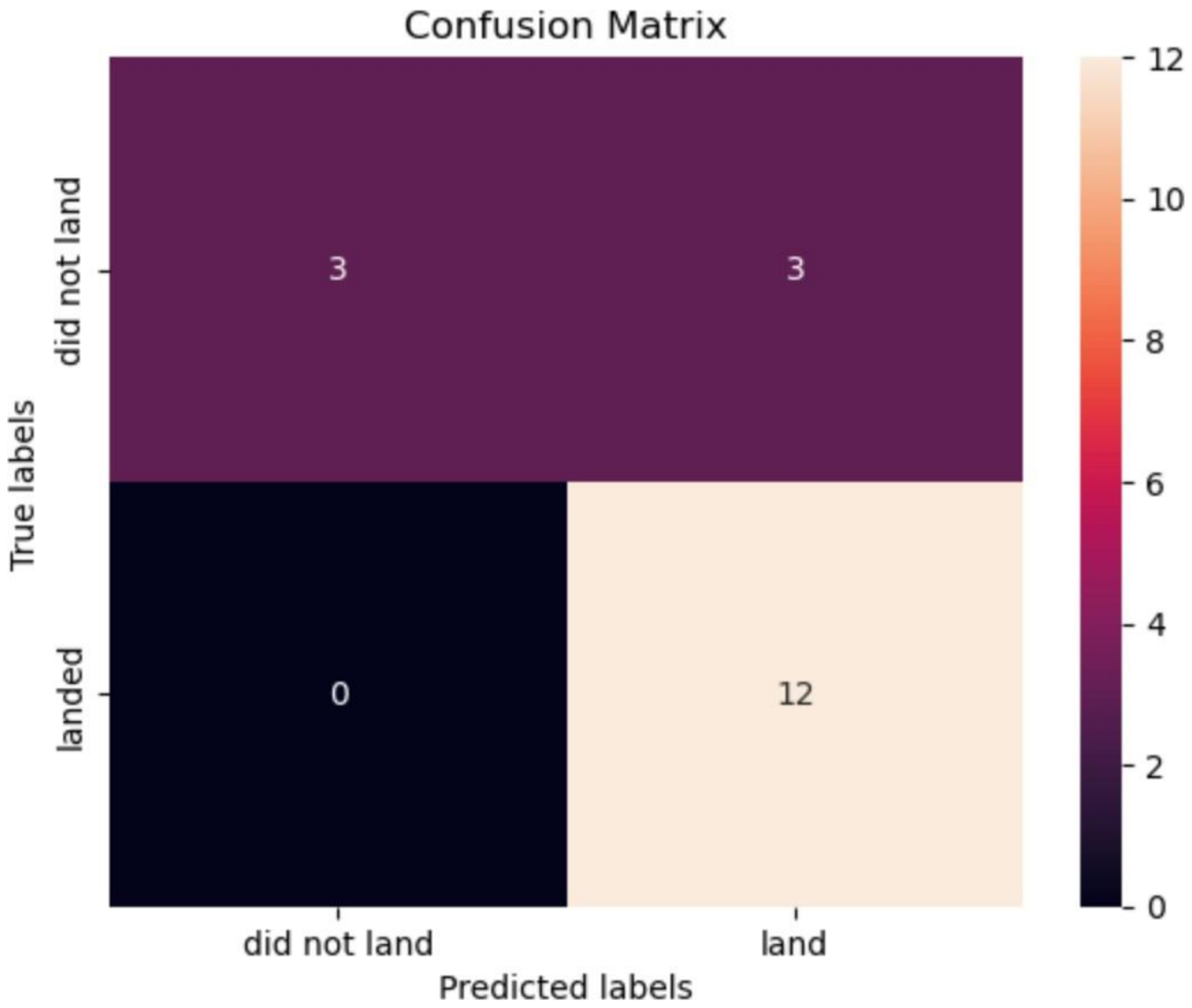



| | Algorithm | Accuracy Score | Best Score |
|---|---|---|---|
| 0 | Logistic Regression | 0.833333 | 0.846429 |
| 1 | Support Vector Machine | 0.833333 | 0.848214 |
| 2 | Decision Tree | 0.833333 | 0.903571 |
| 3 | K Nearest Neighbours | 0.777778 | 0.876786 |

# Confusion Matrix

- As shown previously, best performing classification model is the Decision Tree model, with an accuracy of 94.44%.
- This is explained by the confusion matrix, which shows only 1 out of 18 total results classified incorrectly (a false positive, shown in the top-right corner).
- The other 17 results are correctly classified (5 did not land, 12 did land).



# Conclusions

- As the number of flights increases, the rate of success at a launch site increases, with most early flights being unsuccessful. I.e. with more experience, the success rate increases.
  - Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
  - After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
  - After 2016, there was always a greater than 50% chance of success.

- Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.
    - The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
    - The 100% success rate in SSO is more impressive, with 5 successful flights.
    - The orbit types PO, ISS, and LEO, have more success with heavy payloads:
    - VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.

- The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches, and also the highest rate of successful launches, with a 76.9% success rate.
- The success for massive payloads (over 4000kg) is lower than that for low payloads.
- The best performing classification model is the Decision Tree model, with an accuracy of 94.44%.

# Appendix

https://github.com/pranalipatil1208/ds-capstone

# Thank you!