IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Aditya Dadasaheb Lawand
25-07-2023

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:

1. Exploratory Data Analysis (EDA) results
2. Geospatial analytics
3. Interactive dashboard
4. 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

Methodology

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 was collected from the following two sources:
  - Space X API - https://api.spacexdata.com/v4/rockets/
  - Wikipedia - https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

Data Collection – SpaceX API

- SpaceX offers a public API from where data can be obtained and then used.
- Using the SpaceX API to retrieve data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

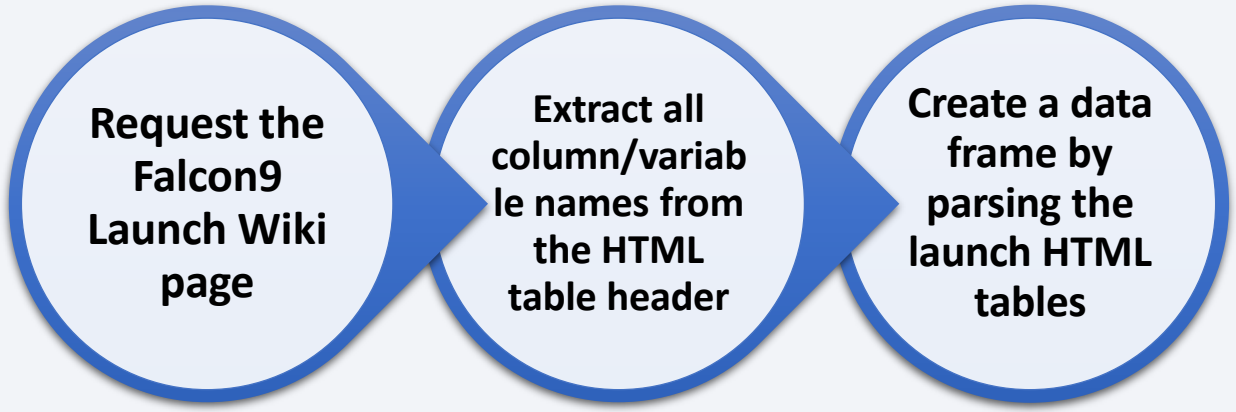
GitHub

Data Collection – SpaceX API (contd.)

- Steps:
  - Make a GET response to the SpaceX REST API.
  - Convert the response to a .json file then to a Pandas DataFrame.
  - Use custom logic to clean the data.
  - Define lists for data to be stored in.
  - Call custom functions to retrieve data and fill the lists.
  - Use these lists as values in a dictionary and construct the dataset.
  - Create a Pandas DataFrame from the constructed dictionary dataset.
  - Filter the DataFrame to only include Falcon 9 launches
  - Reset the FlightNumber column
  - Replace missing values of PayloadMass with the mean PayloadMass value.

Data Collection – Web Scraping

- Data about SpaceX launches can also be obtained from Wikipedia.
- Web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.



GitHub

Data Collection – Web Scraping (contd.)

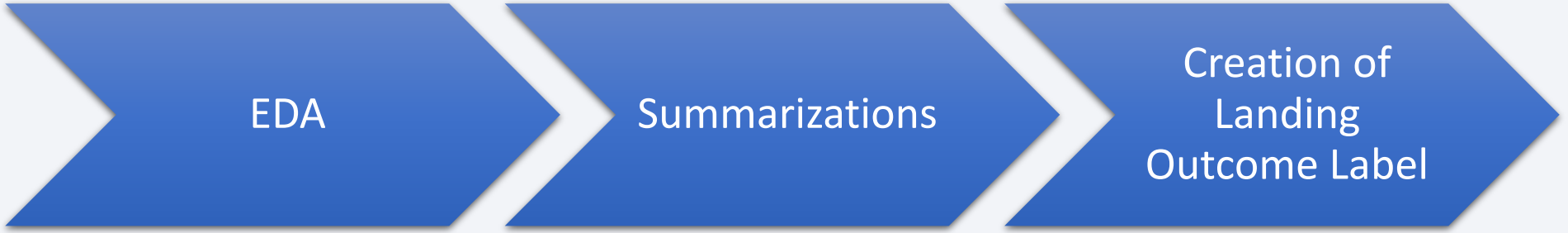
- Steps:
  - Request the HTML page from the static URL
  - Assign the response to an object.
  - Create a BeautifulSoup object from the HTML response object .
  - Find all tables within the HTML page.
  - Collect all column header names from the tables found within the HTML page.
  - Use the column names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

Data Wrangling

- Initially, some Exploratory Data Analysis (EDA) was performed on the dataset.
- Then, the summaries of launches per site, occurrences of each orbit, and commences of mission outcome per orbit type were calculated.
- Finally, the landing outcome label was created from the Outcome column.
- 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:
  - Defining a set of unsuccessful (bad) outcomes, bad_outcome
  - 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.
  - Create a Class column that contains the values from the list landing_class
  - Export the DataFrame as a .csv file.

GitHub

Data Wrangling (contd.)



EDA with Data Visualization

Scatter Plots

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.

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.

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.

GitHub

EDA with SQL

The SQL queries performed on the data set were used to:

1. Display the names of the unique launch sites in the space mission.
2. Display 5 records where launch sites begin with the string ‘CCA’.
3. Display the total payload mass carried by boosters launched by NASA (CRS).
4. Display the average payload mass carried by booster version F9 v1.1.
5. List the date when the first successful landing outcome on a ground pad was achieved.
6. List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg.
7. List the total number of successful and failed mission outcomes.
8. List the names of the booster versions which have carried the maximum payload mass.
9. List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015.
10. Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

GitHub

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.

GitHub

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

GitHub

Predictive Analysis (Classification)

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

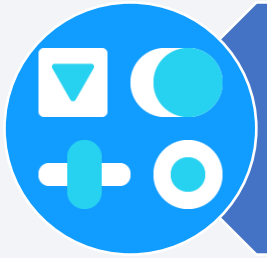
Finding the Best Classification Model

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

GitHub

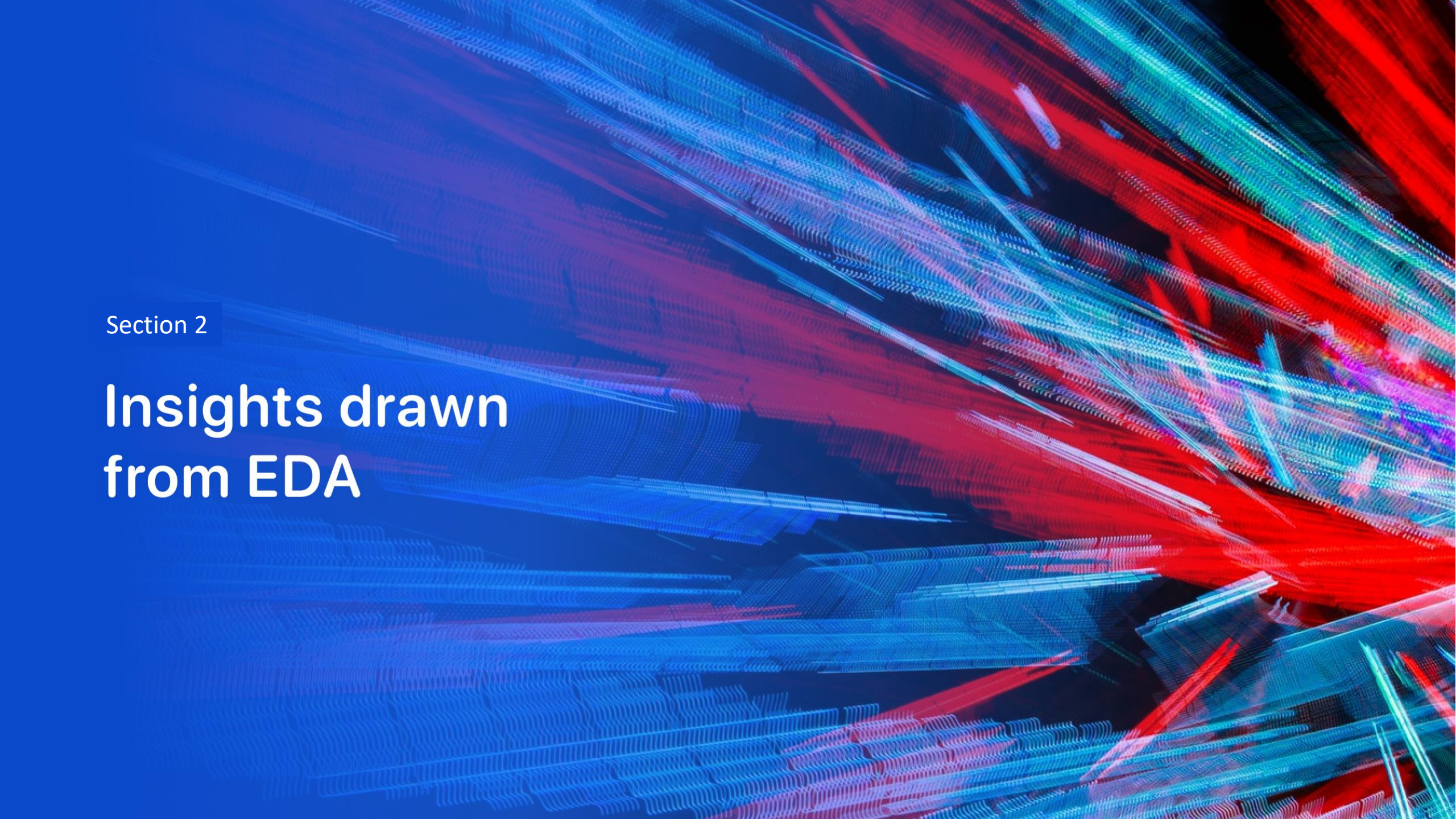
Results

Exploratory Data Analysis



Interactive Analytics

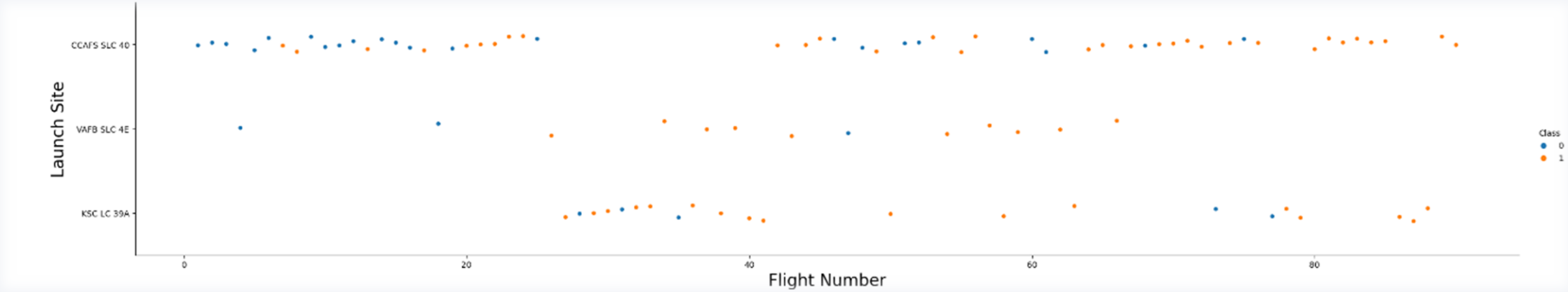
Predictive Analysis


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

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

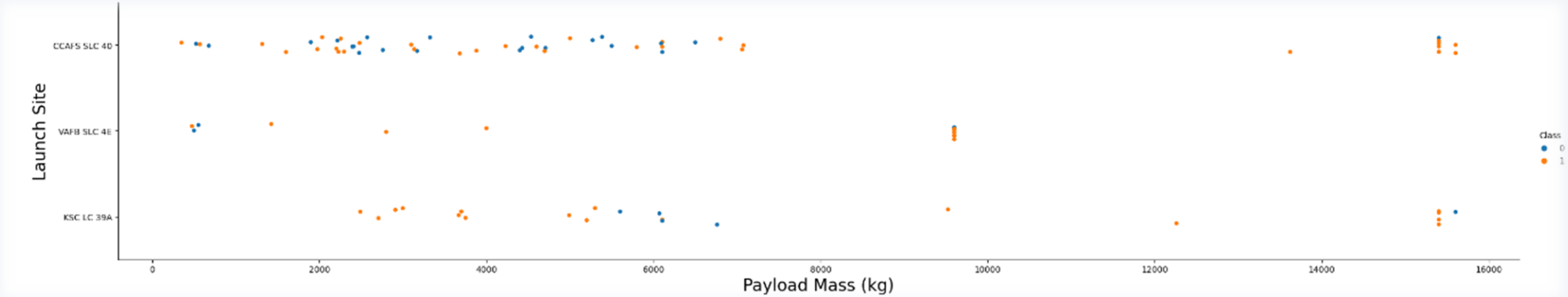
- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight numbers < 30) were launched from CCAFS SLC 40, and were generally unsuccessful.
- The flights from VAFB SLC 4E also show this trend, that earlier flights were less successful.
- No 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).



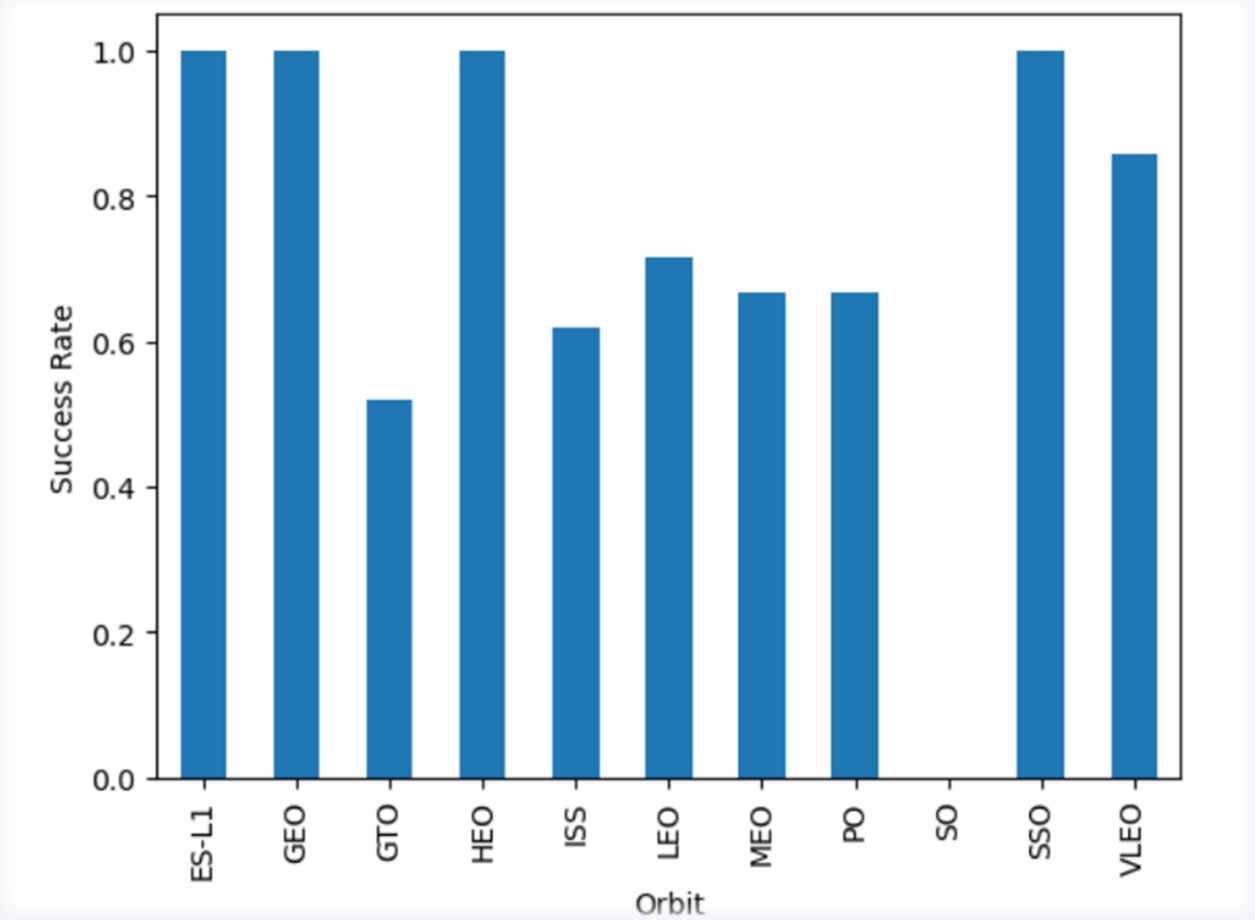
Success Rate vs. Orbit Type

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:

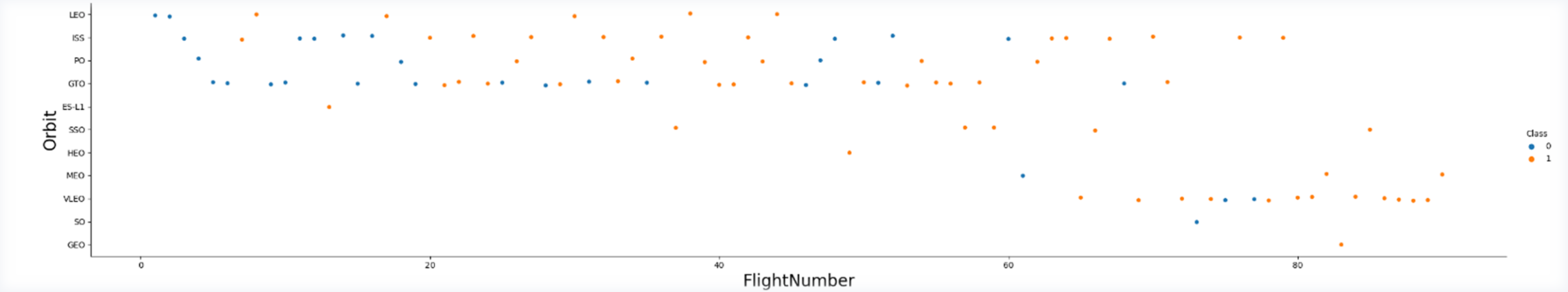
- SO (Heliocentric Orbit)



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:

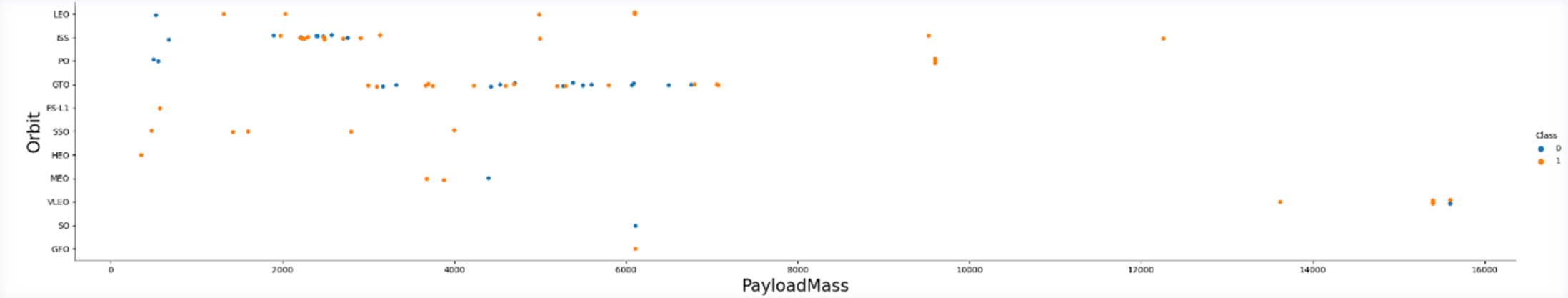
- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is little relationship between Flight Number and Success Rate for GTO.
- Generally, as Flight 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:

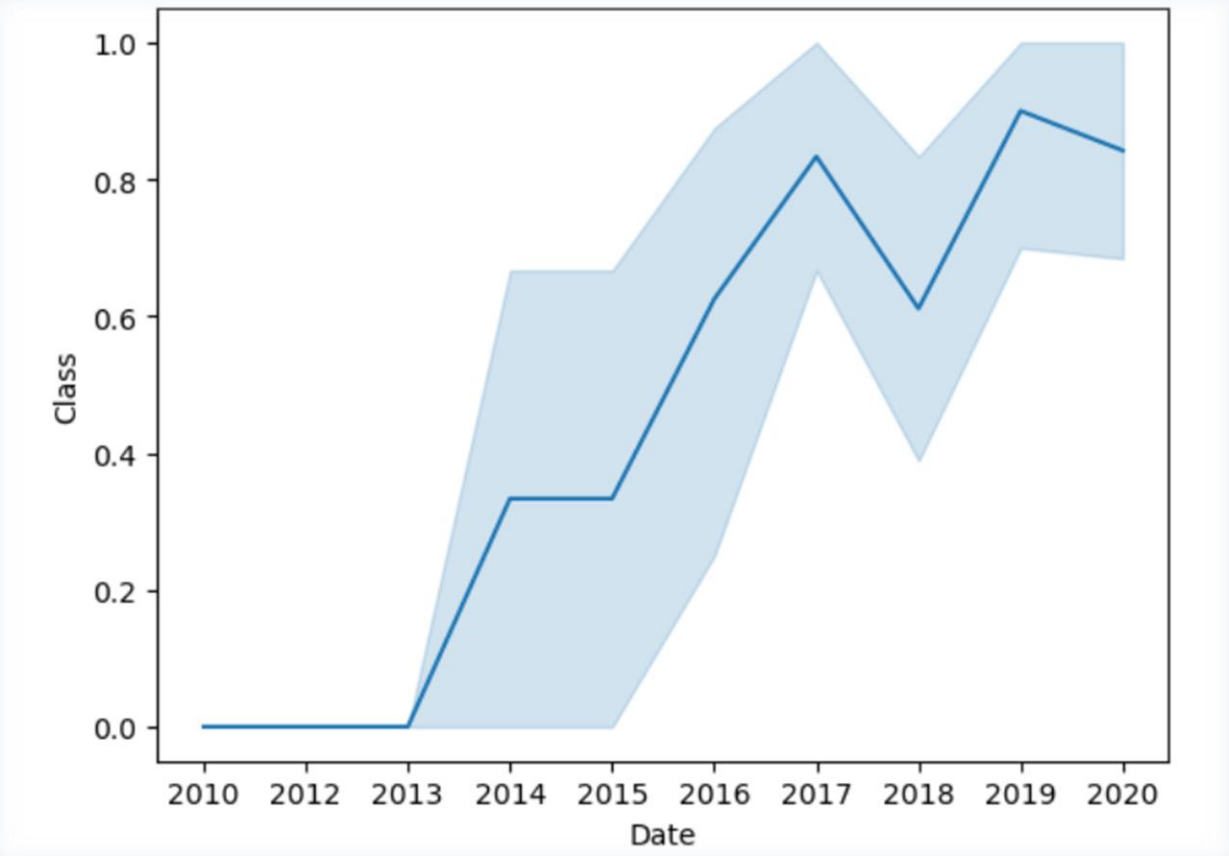
- The following orbit types have more success with heavy payloads:
  - PO (although the number of data points is small)
  - ISS
  - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes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

```
In [5]: %sql select distinct(launch_site) from spacextbl;
 * sqlite:///my_data1.db
Done.
```

Out[5]:

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

The word `DISTINCT` returns only distinct 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`

```
In [6]: %sql select * from spacextbl where launch_site like "CCA%" limit 5;
 * sqlite:///my_data1.db
Done.
```

Out[6]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

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

```
In [7]: %sql select sum(payload_mass__kg_) as "Total Payload Mass (NASA CRS)" from spacextbl where customer="NASA (CRS)";
         * sqlite:///my_data1.db
        Done.
```

Out[7]:

Total Payload Mass (NASA CRS)
45596.0

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

```
In [8]: %sql select avg(payload_mass__kg_) as "Avergae Payload Mass (F9 v1.1)" from spacextbl where booster_version = "F9 v1.1"
 * sqlite:///my_data1.db
Done.
```

Out[8]:

Avergae Payload Mass (F9 v1.1)
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

```
In [9]: %sql select min(date) as "First Success(Ground Pad)" from spacextbl where landing_outcome = "Success (ground pad)";
 * sqlite:///my_data1.db
Done.
```

Out[9]:

First Success(Ground Pad)
01/08/2018

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.

In [10]:
```
%sql select distinct booster_version from spacextbl where landing_outcome = "Success (drone ship)" and payload_mass__kg_ \
between 4000 and 6000;
```

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

Out[10]:

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

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 outcomes.

```
In [17]: %sql select mission_outcome, count(*) as Number from spacextbl group by mission_outcome;
 * sqlite:///my_data1.db
Done.
```

Out[17]:

Mission_Outcome	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.

```
In [18]: %sql select booster_version from spacextbl where payload_mass__kg_ = (select max(payload_mass__kg_) from spacextbl);
 * sqlite:///my_data1.db
Done.
```

Out[18]:

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.

```
In [20]: %sql select booster_version, launch_site from spacextbl where landing_outcome = "Failure (drone ship)" and date like "%2015";
 * sqlite:///my_data1.db
Done.
```

Out[20]:

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

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.

```
In [23]: %sql SELECT landing_outcome, COUNT(landing_outcome) AS qty FROM spacextbl WHERE date BETWEEN '01-06-2010' AND '20-03-2017' \
         GROUP BY landing_outcome;
```

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

Out[23]:

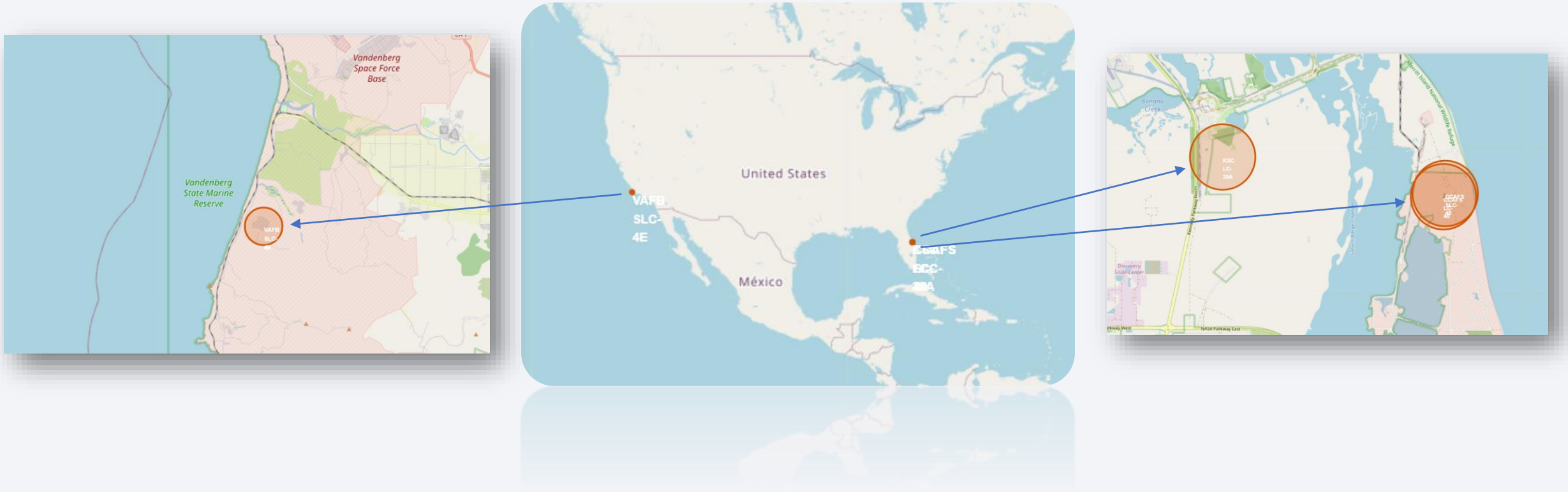
Landing_Outcome	qty
Controlled (ocean)	3
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
No attempt	14
No attempt	1
Success	24
Success (drone ship)	8
Success (ground pad)	8

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

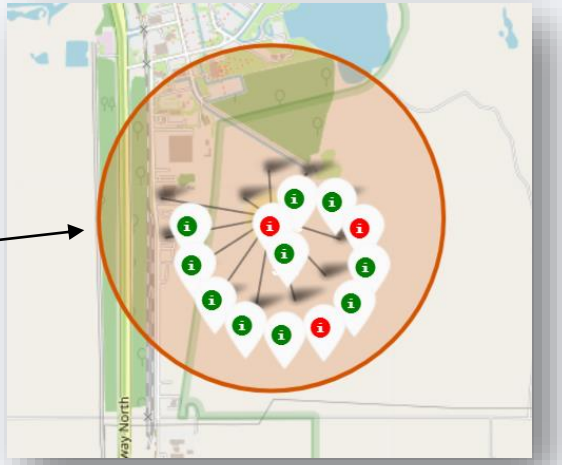


All SpaceX launch sites are on coasts of the United States of America, specifically Florida and California.

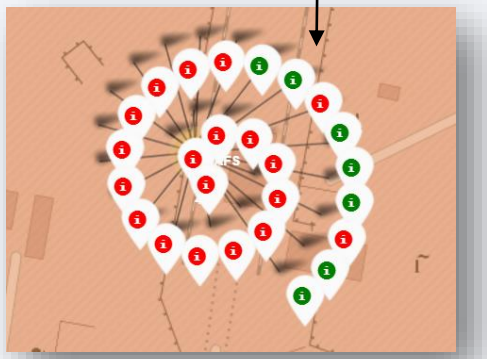
Launch Outcomes by Site

VAFB SLC-4E

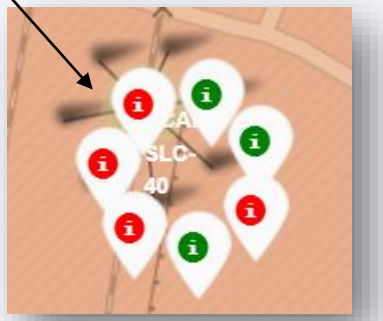
KSC LC-39A



CCAFS SLC-40



CCAFS LC-40



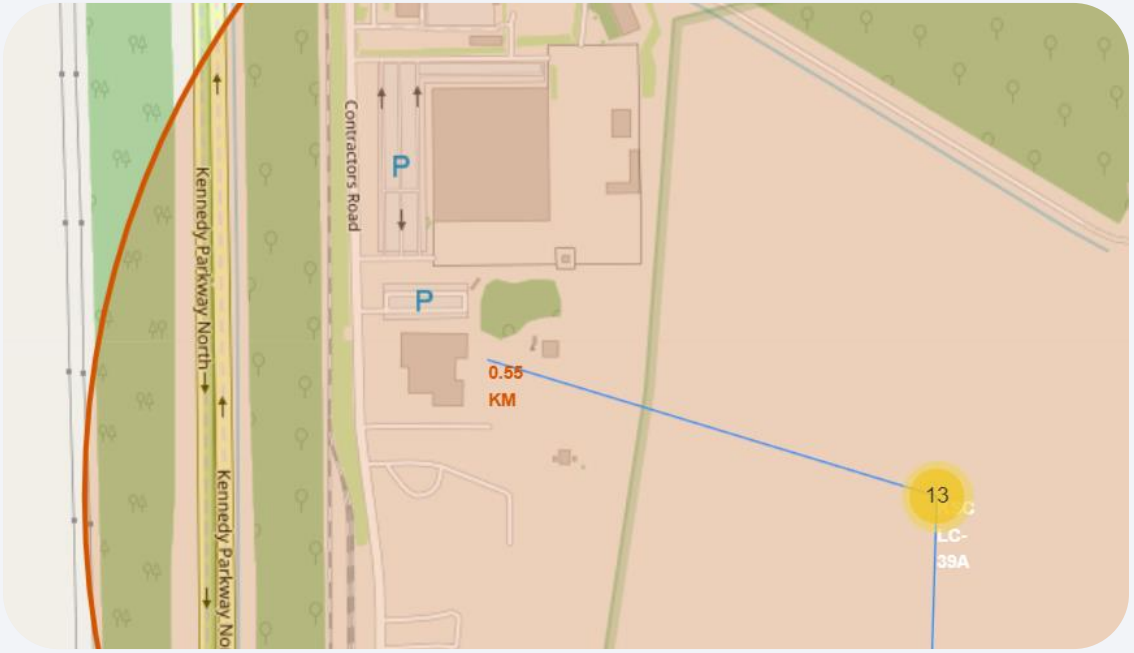
Launches have been grouped into clusters, and annotated with green icons for successful launches, and red icons for failed launches.

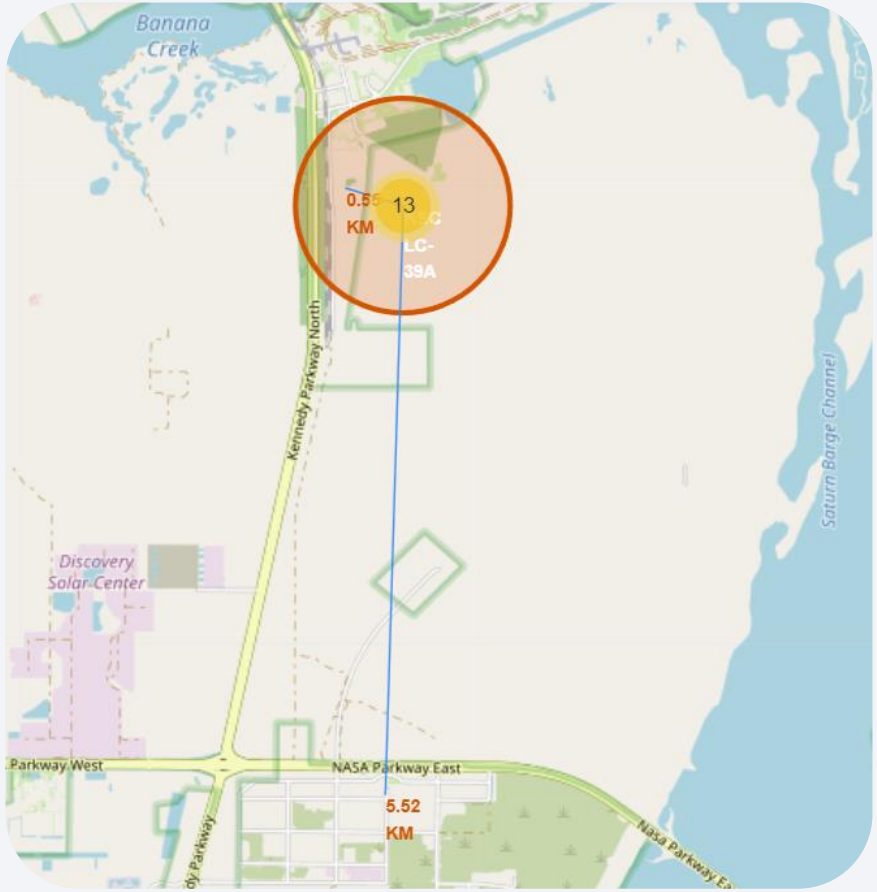
Logistics and Safety

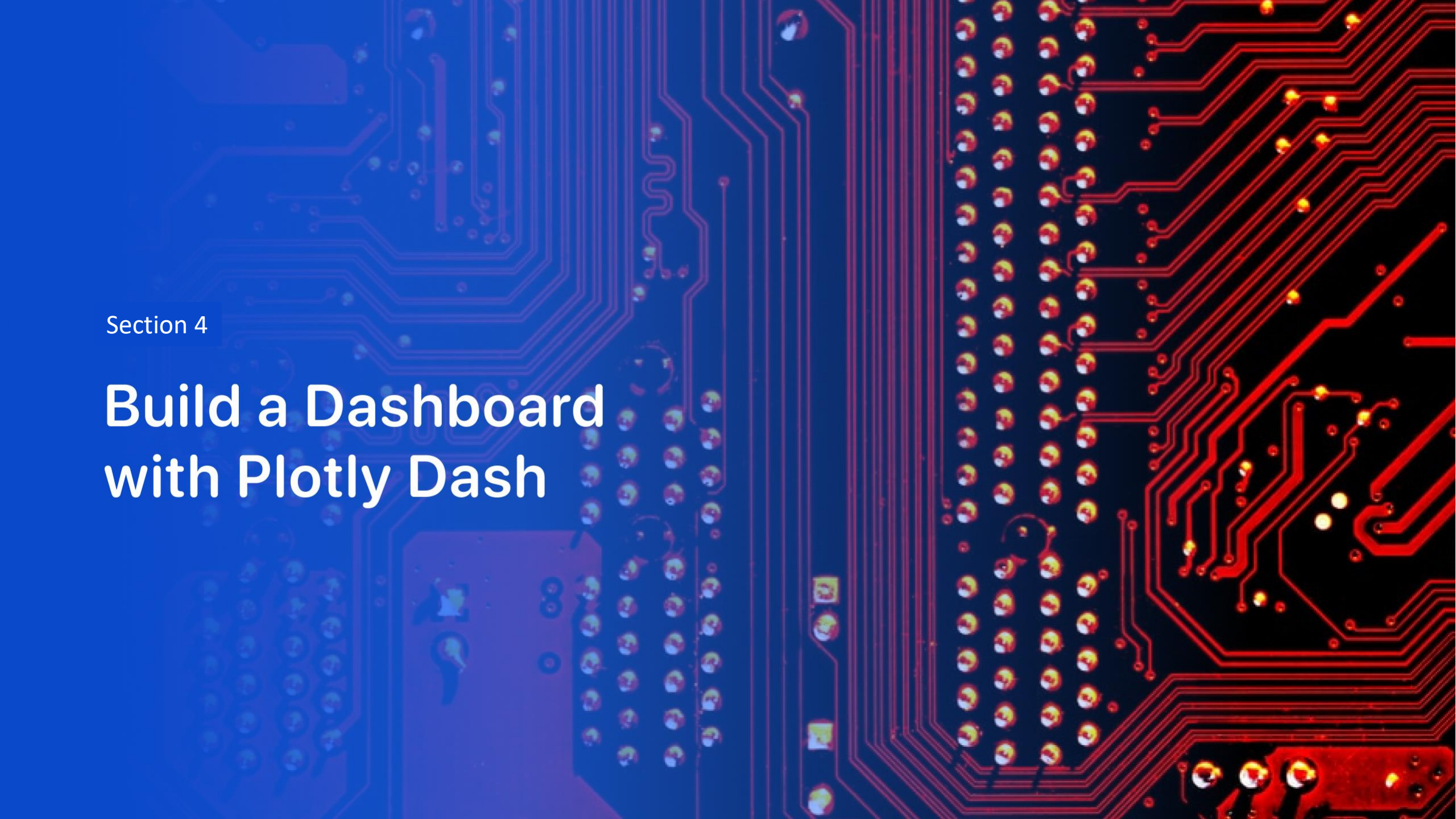
Using the KSC LC-39A launch site as an example site, we can understand more about the placement of launch sites.

Are launch sites in close proximity to railways?

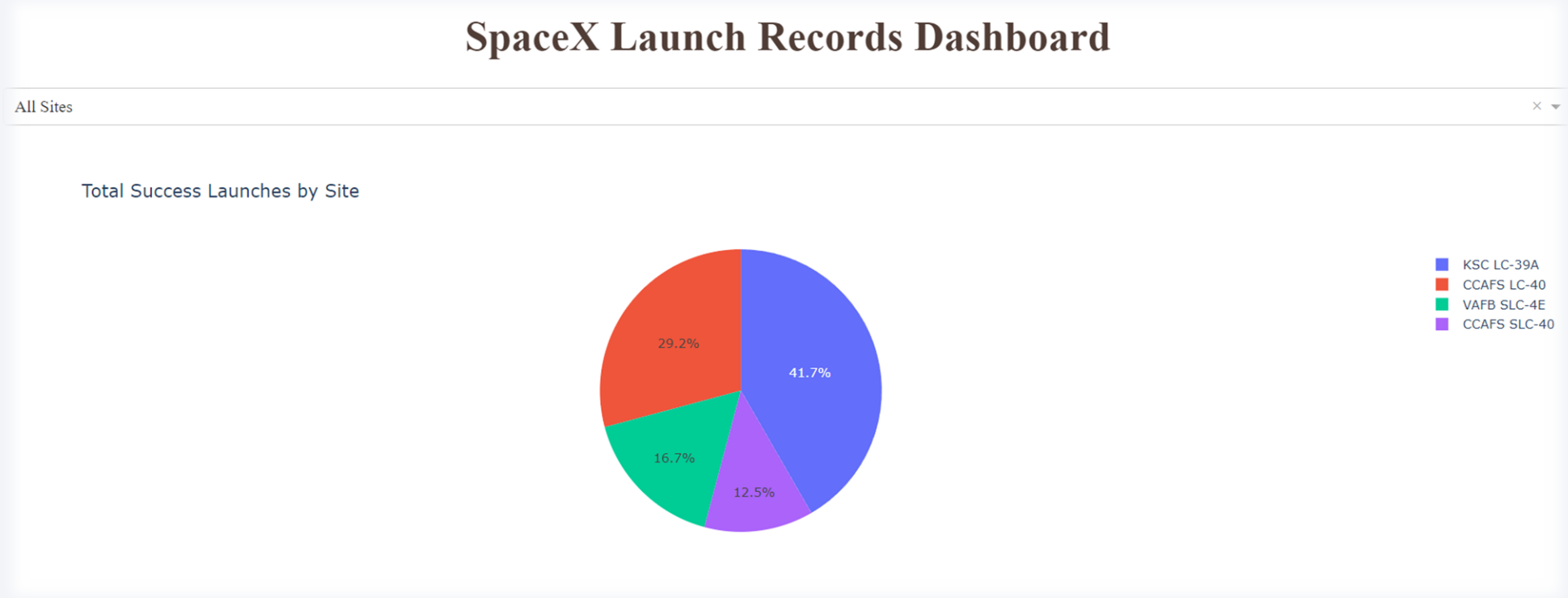
- YES. The nearest railway is only 0.59 km away.





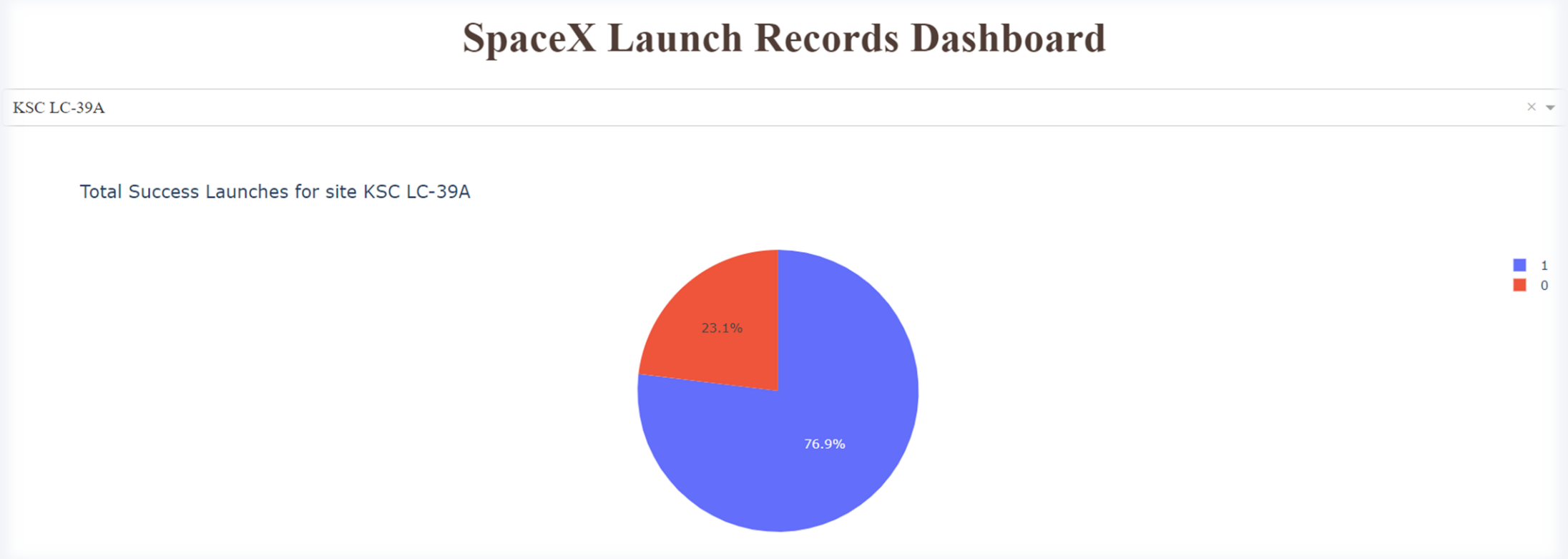

Section 4
Build a Dashboard with Plotly Dash

Successful Launches by Site



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

Launch Success Ratio for KSC LC-39A

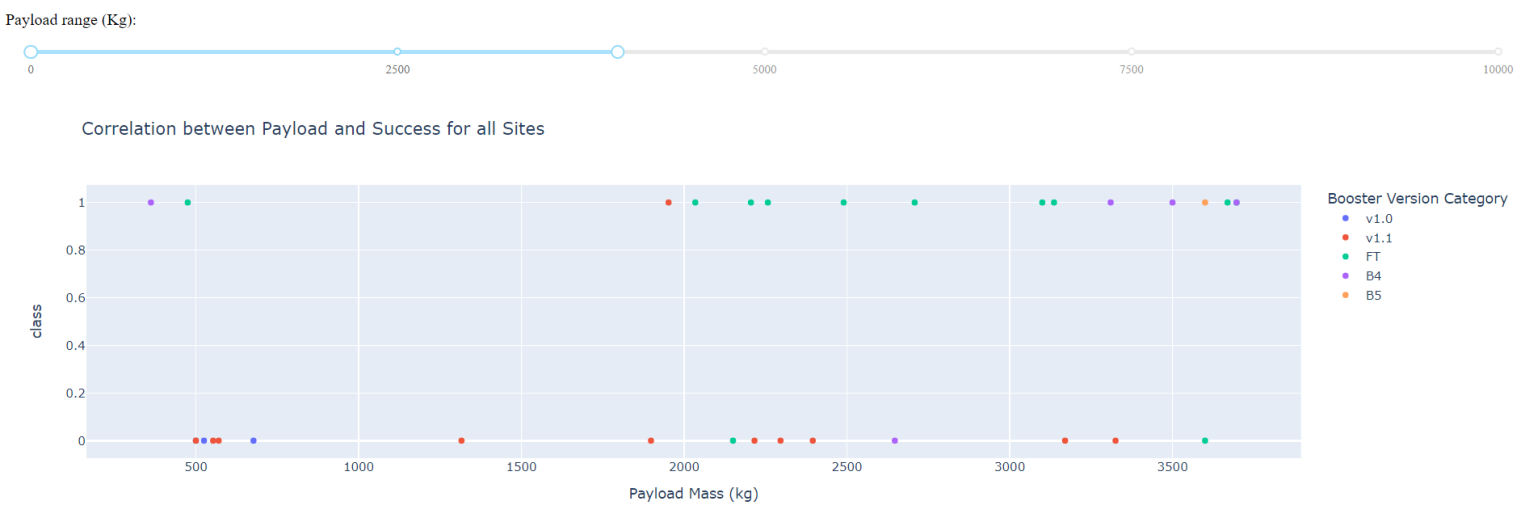


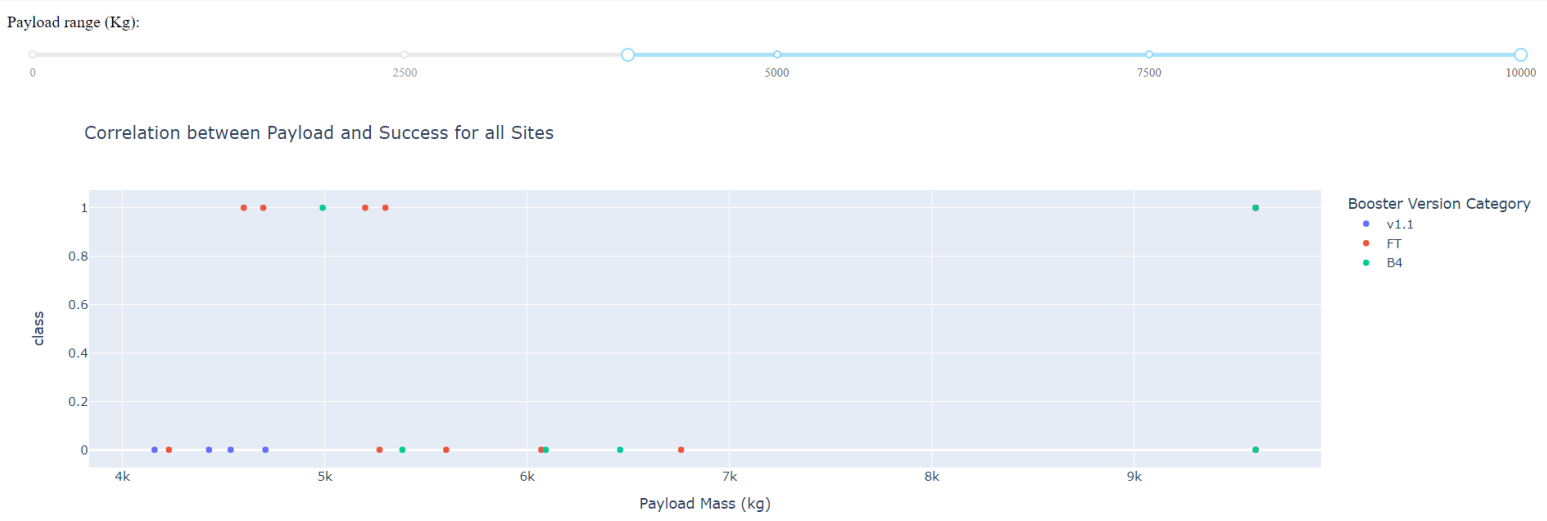
The launch site KSC LC-39 A also had the highest rate of successful launches, with a 76.9% success rate.

Payload vs. Launch Outcome

Payloads under 6,000kg and FT boosters are the most successful combination.

Payload vs. Launch Outcome





- 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.

Section 5

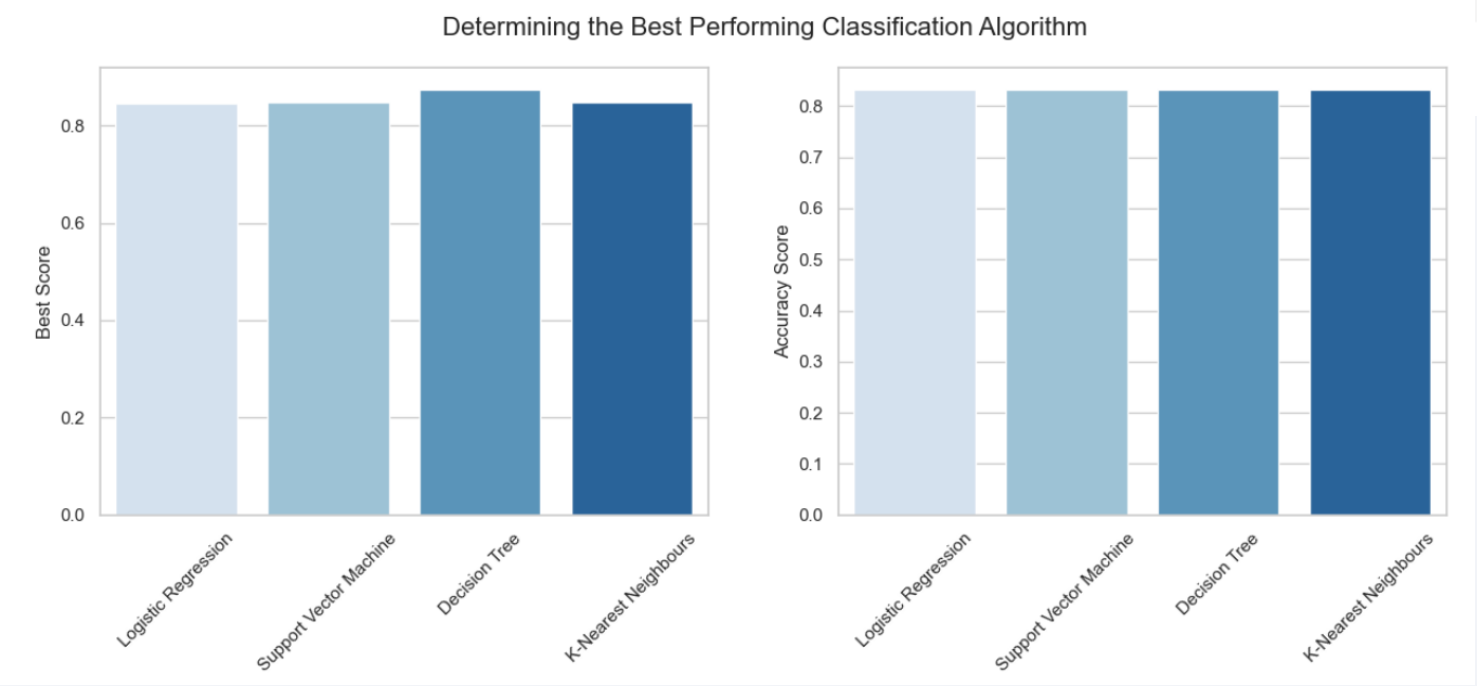
Predictive Analysis (Classification)

Classification Accuracy

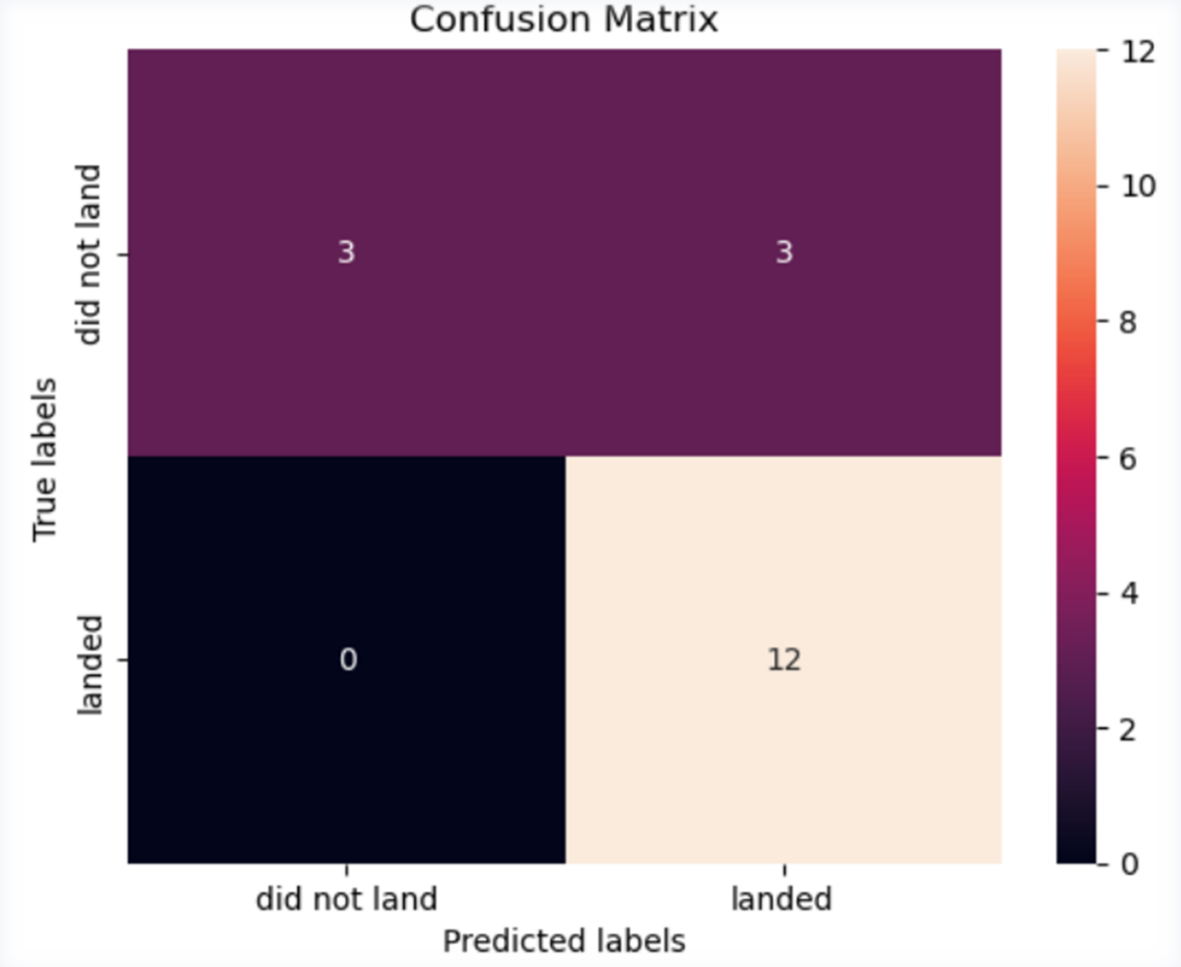
Plotting the Accuracy Score and Best Score for each classification algorithm produces the following result:

- The Decision Tree model has the highest classification accuracy
  - The Accuracy Score is 83.33%
  - The Best Score is 87.50%

	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	0.875000
3	K-Nearest Neighbours	0.833333	0.848214



Confusion Matrix



- As shown previously, best performing classification model is the Decision Tree model, with an accuracy of 83.33%.
- This is explained by the confusion matrix, which shows only 3 out of 18 total results classified incorrectly (a false positive, shown in the top-right corner).
- The other 15 results are correctly classified (3 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 83.33%.

Appendix

Custom functions to retrieve the required information.

```
def getBoosterVersion(data):
    for x in data["rocket"]:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
            BoosterVersion.append(response["name"])
```

```
def getLaunchSite(data):
    for x in data["launchpad"]:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
            Longitude.append(response["longitude"])
            Latitude.append(response["latitude"])
            LaunchSite.append(response["name"])
```

```
def getPayloadData(data):
    for load in data["payloads"]:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response["mass_kg"])
        Orbit.append(response["orbit"])
```

```
def getCoreData(data):
    for core in data["cores"]:
        if core["core"] != None:
            response = requests.get("https://api.spacexdata.com/v4/cores/"+core["core"]).json()
            Block.append(response["block"])
            ReusedCount.append(response["reuse_count"])
            Serial.append(response["serial"])
        else:
            Block.append(None)
            ReusedCount.append(None)
            Serial.append(None)
        Outcome.append(str(core["landing_success"]) + " " + str(core["landing_type"]))
        Flights.append(core["flight"])
        GridFins.append(core["gridfins"])
        Reused.append(core["reused"])
        Legs.append(core["legs"])
        LandingPad.append(core["landpad"])
```

```
extracted_row = 0
for table_number, table in enumerate(soup.find_all("table","wikitable plainrowheaders collapsible")):
    for rows in table.find_all("tr"):
        if rows.th:
            if rows.th.string:
                flight_number = rows.th.string.strip()
                flag = flight_number.isdigit()
        else:
            flag = False

        row = rows.find_all("td")

        if flag:
            extracted_row += 1
            launch_dict["Flight No."].append(flight_number)
            datetimelist = date_time(row[0])

            date = datetimelist[0].strip(",")
            launch_dict["Date"].append(date)

            time = datetimelist[1]
            launch_dict["Time"].append(time)

            bv = booster_version(row[1])
            if not(bv):
                bv = row[1].a.string
            launch_dict["Version Booster"].append(bv)

            launch_site = row[2].a.string
            launch_dict["Launch site"].append(launch_site)

            payload = row[3].a.string
            launch_dict["Payload"].append(payload)

            payload_mass = get_mass(row[4])
            launch_dict["Payload mass"].append(payload_mass)

            orbit = row[5].a.string
            launch_dict["Orbit"].append(orbit)

            if (row[6].a != None):
                customer = row[6].a.string
            else:
                customer = ""
            launch_dict["Customer"].append(customer)

            launch_outcome = list(row[7].strings)[0]
            launch_dict["Launch outcome"].append(launch_outcome)

            booster_landing = landing_status(row[0])
            launch_dict["Booster landing"].append(booster_landing)
```

Thank you!