IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

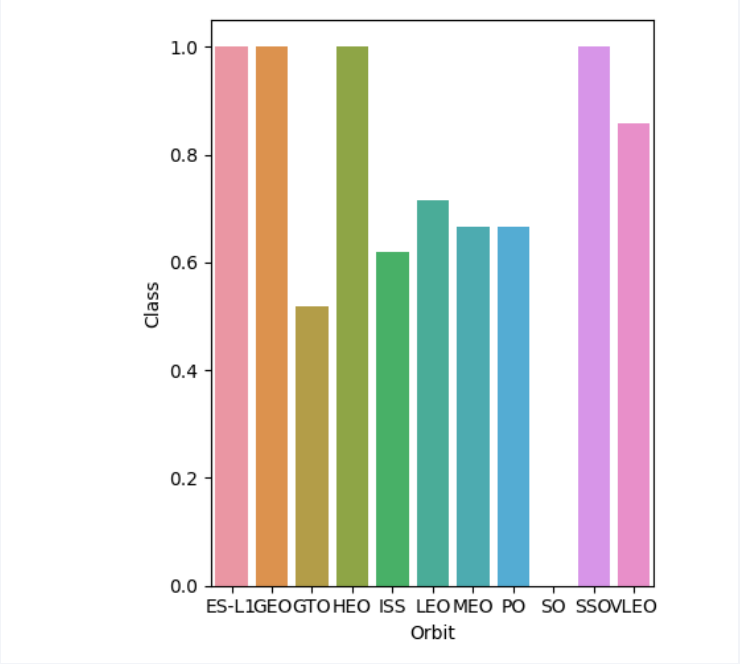
Preslav Kisyov
25/05/2023

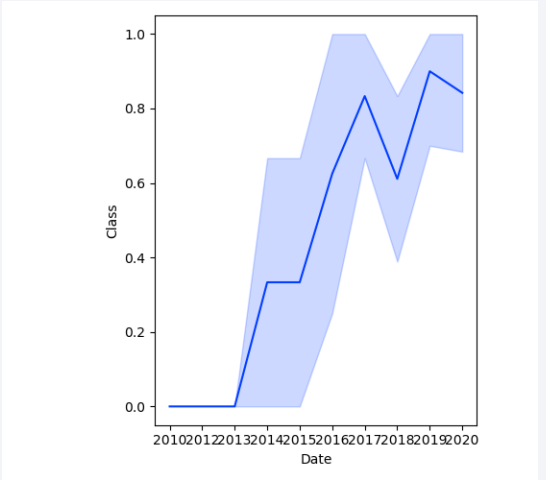
Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection & Wrangling
  - EDA
  - Interactive Visuals
  - Predictive Analysis
- Summary of all results
  - EDA Results
  - Geospatial Analytics
  - Dashboard
  - Predictive Analysis of Models





Introduction

- SpaceX conducts launches of its Falcon 9 rockets at an approximate price of $62 million. This cost is significantly lower compared to other providers who typically charge over $165 million for similar services. The primary reason behind this cost advantage is SpaceX's ability to successfully land and reuse the first stage of the rocket.

- By developing the capability to predict the landing outcome of the Falcon 9's first stage, we can estimate the overall cost of a launch. This predictive information becomes valuable when evaluating whether an alternative company should compete with SpaceX for a rocket launch contract. The ultimate goal of this project is to forecast the successful landing of SpaceX's Falcon 9 first stage.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Via REST API & Web Scraping
- Perform data wrangling
  - Using functions like fillna and value_counts to clean and fetermine the number of launches and occurences of each orbit.
  - Created labels of landing outcome 0-1.
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Using SQL, Pandas, and Matplotlib to visualize and analyse.
- Perform interactive visual analytics using Folium and Plotly Dash
  - Geospatial analytics using Folium and interactive dashboard with Dash
- Perform predictive analysis using classification models
  - We used Sklearn to train a model, split the data, analyse the models and find the best parameters.

Data Collection

In [7]:
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Out[7]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

In [17]:
```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_GROUND_LANDING FROM SPACEXTBL \
    WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

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

Out[17]:

FIRST_SUCCESSFUL_GROUND_LANDING
01/08/2018

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [19]:
```
%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.
```

Out[19]:

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

In [42]:
```
%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL \
    WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' \
    GROUP BY LANDING_OUTCOME \
    ORDER BY TOTAL_NUMBER DESC;
```

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

Out[42]:

Landing_Outcome	TOTAL_NUMBER
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

Data Collection – SpaceX API

1. GET Request to the API and convert the response to json
2. Clean data and construct a dataset using Pandas
3. Filter the data and reset the FlightNumber column

- https://github.com/PreslavKisyov/IBM-DataScience-Tools/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

We should see that the request was successfull with the 200 status response code

```
In [10]: response.status_code

Out[10]: 200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

```
In [11]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

Using the dataframe `data` print the first 5 rows

```
In [12]: # Get the head of the dataframe
         data.head(5)
```

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"

In [7]: response = requests.get(spacex_url)
```

Check the content of the response

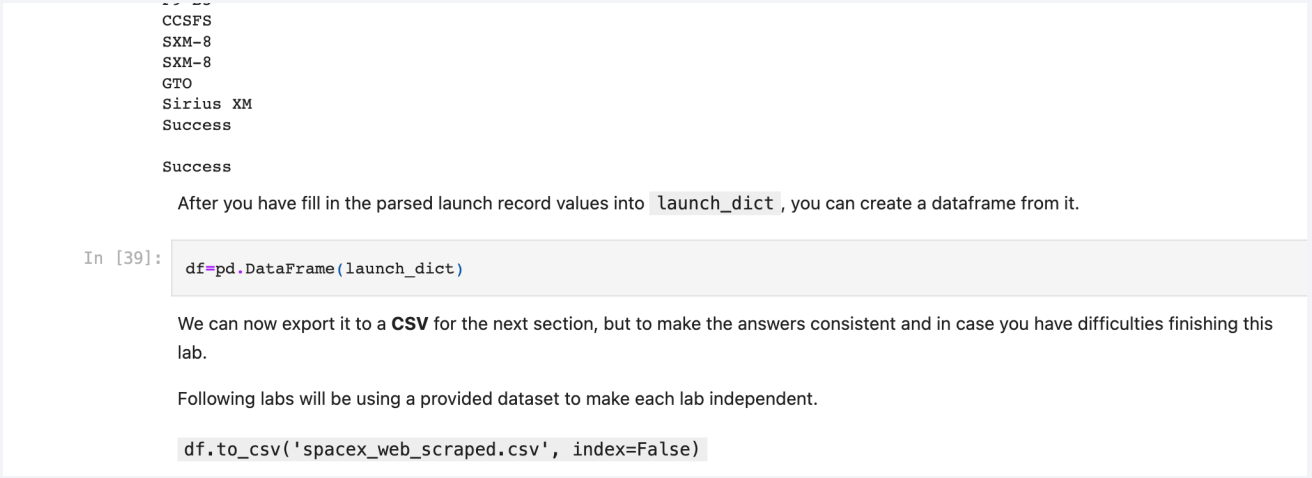
```
In [8]: print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"ht
tps://images2.imgbox.com/94/f2/NN6Ph45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":
{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"w
ebcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2
196-spacex-inaugural-falcon-1-rocket-lost-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static
_fire_date_utc":"2006-03-17T00:00:00.000Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d
0d95eda69955f709d1eb","success":false,"failures":[{"time":33,"altitude":null,"reason":"merlin engine failure"}],"de
tails":"Engine failure at 33 seconds and loss of vehicle","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b5b
6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flight_number":1,"name":"FalconSat","date_utc":"2006-03-2
4T22:30:00.000Z","date_unix":1143239400,"date_local":"2006-03-25T10:30:00+12:00","date_precision":"hour","upcomin
g":false,"cores":[{"core":"5e9e289df35918033d3b2623","flight":1,"gridfins":false,"legs":false,"reused":false,"landi
ng_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_update":true,"tbd":false,"launc
h_library_id":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reused":false,"recovery_attempt":false,"recovere
d":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/f9/4a/ZboXReNb_o.png","large":"https://i
```

Data Collection - Scraping

- Used beautifulSoup and statuc URL to get HTML data and extract tables.

- https://github.com/PreslavKisyov/IBM-DataScience-Tools/blob/main/jupyter-labs-webscraping.ipynb



```
CCSFS
SXM-8
SXM-8
GTO
Sirius XM
Success

Success
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

In [39]:
```
df=pd.DataFrame(launch_dict)
```

We can now export it to a **CSV** for the next section, but to make the answers consistent and in case you have difficulties finishing this lab.

Following labs will be using a provided dataset to make each lab independent.

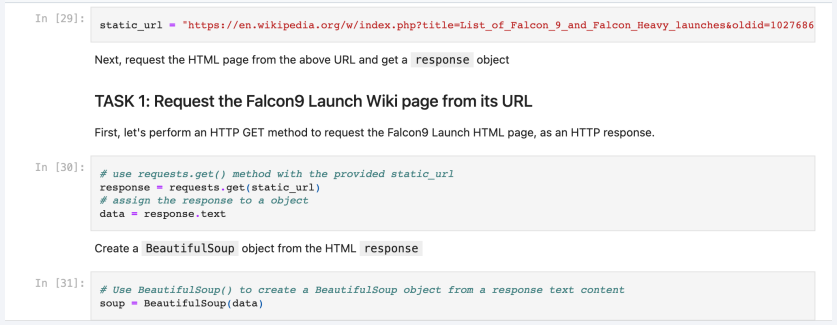
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

In [37]:
```
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']=[]
```

Next, we just need to fill up the `launch_dict` with launch records extracted from table rows.



In [29]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [30]:
```
# use requests.get() method with the provided static_url
response = requests.get(static_url)
# assign the response to a object
data = response.text
```

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

In [31]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data)
```

Data Wrangling

- Defined a set of unsuccessful outcomes and created a landing_list. In that way we could create the class column of 0-1 labels. Then I exported the .csv file.
- Used fillna() and value_counts() to clean and analyse the data
- https://github.com/PreslavKisyov/IBM-DataScience-Tools/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

EDA with Data Visualization

- We have used Scatter plots for Flight Number and Launch Site, Payload and Launch Site, Orbit and Flight Number and Payload and Orbit
- We have used Bar Charts for the Success Rate and Orbit Type
- We have used Line Charts for the Success Rate and Year
- https://github.com/PreslavKisyov/IBM-DataScience-Tools/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- Analysed the data by displaying records of different Launch Sites and Payload Mass per booster version
- List the date of first successful launch
- List the names of boosters that were successful
- List total number of successful and failed missions
- List boosters with maximum payload and failed landing outcomes
- Ranked the count of landing outcomes
- https://github.com/PreslavKisyov/IBM-DataScience-Tools/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

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.

- https://github.com/PreslavKisyov/IBM-DataScience-Tools/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

1. Pie chart showing the total successful launches per site
   - This makes it clear to see which sites are most successful
   - The chart could also be filtered to see the success/failure ratio for an individual site

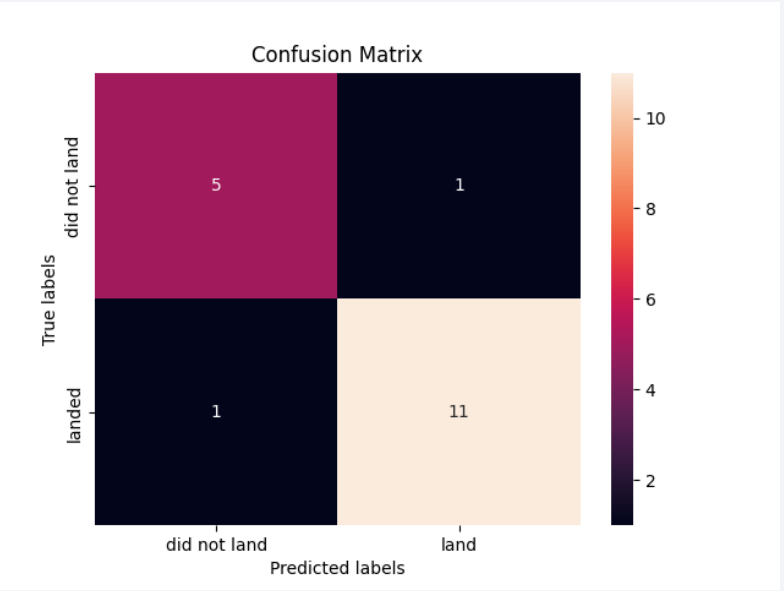
2. Scatter graph to show the correlation between outcome (success or not) and payload mass (kg)
   - This could be filtered by ranges of payload masses
   - It could also be filtered by booster version

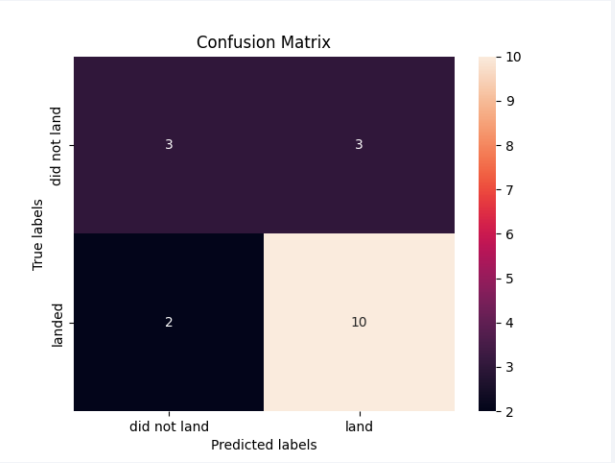
- https://github.com/PreslavKisyov/IBM-DataScience-Tools/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

1. Model Development
    1. Prepared the dataset by loading and transforming it
    2. Split the dataset into train/test
    3. For each algorithm created a GridSearchCV for parameter optimization
    4. Trained the model
2. Model Evaluation
    1. For each algorithm checked the performance and visualize it
    2. Tested using the test dataset

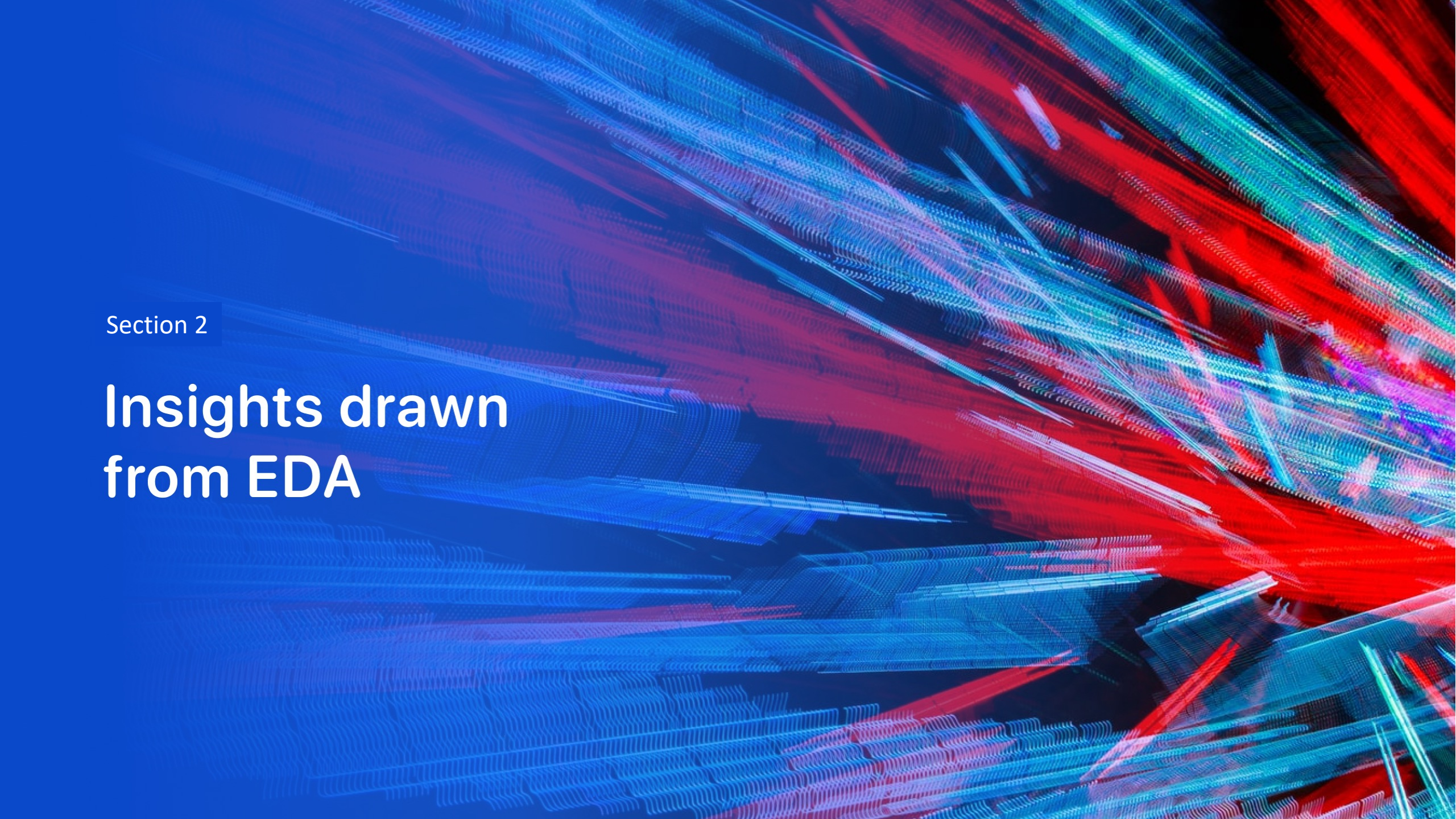
- https://github.com/PreslavKisyov/IBM-DataScience-Tools/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb





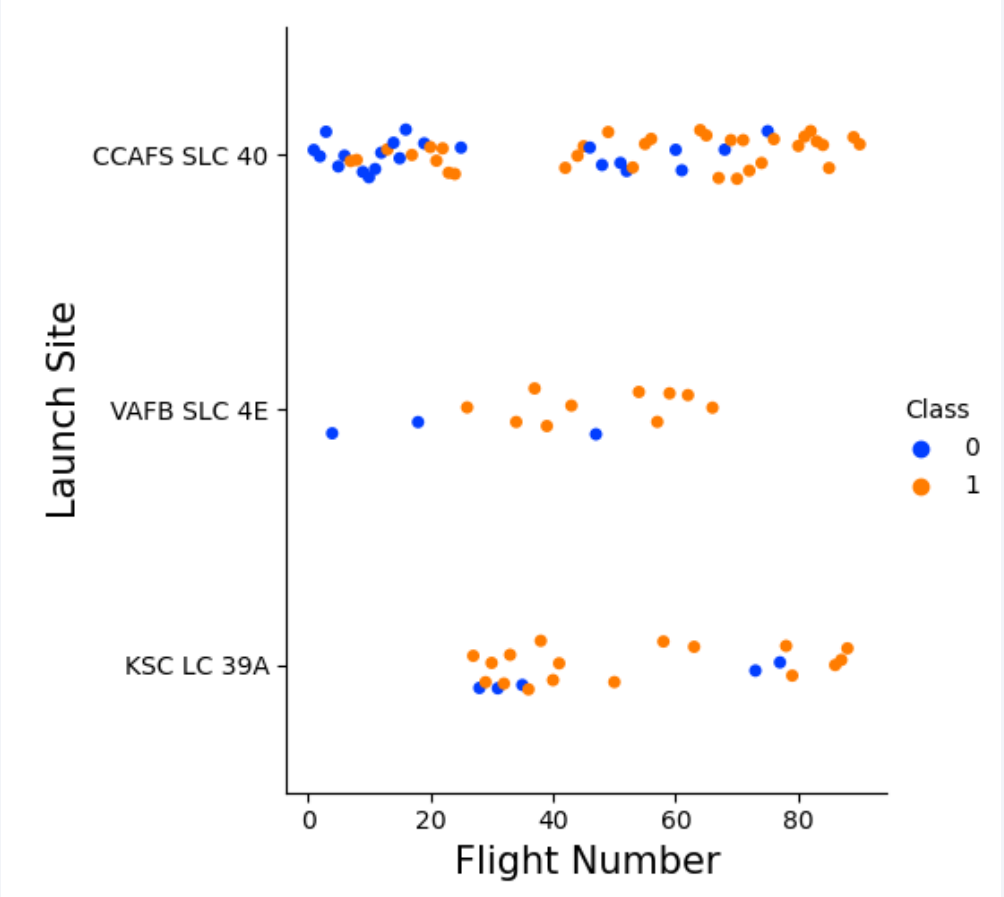
Results

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


Section 2
Insights drawn from EDA

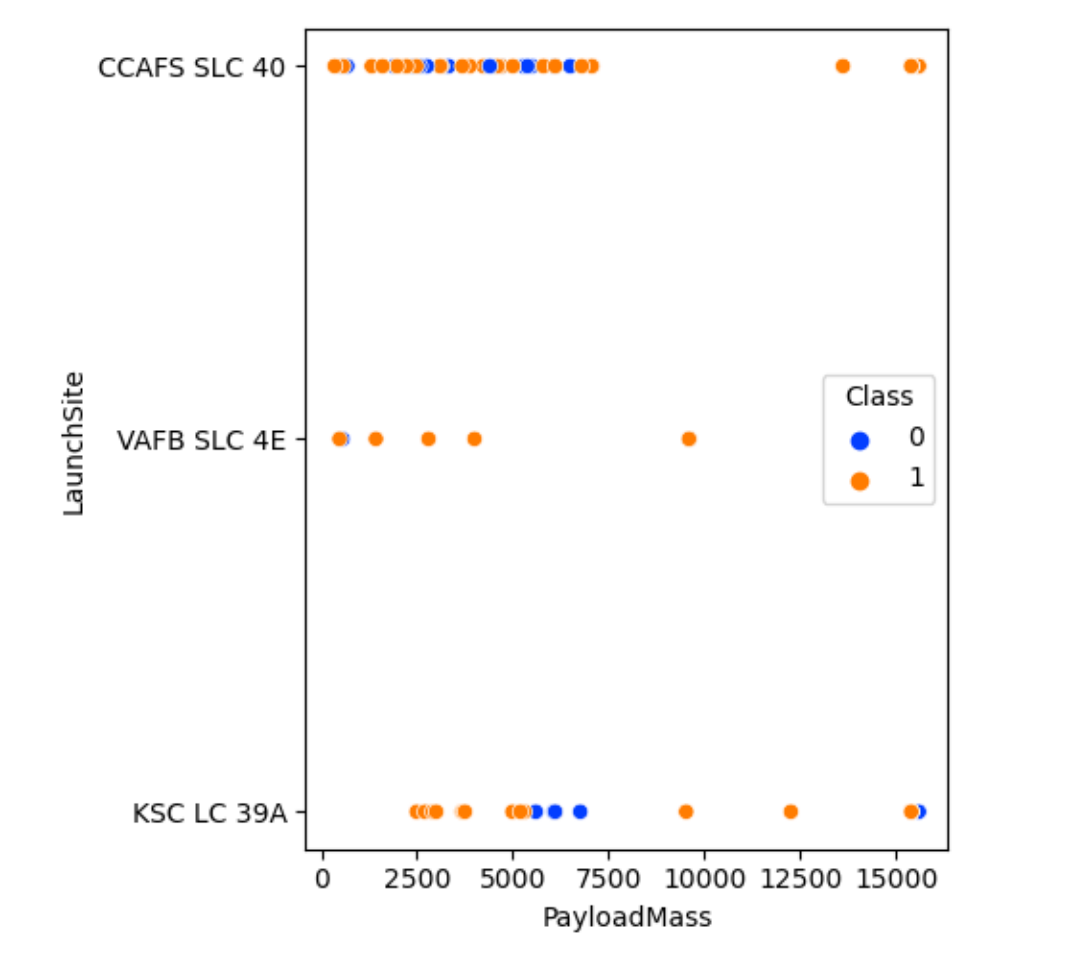
Flight Number vs. Launch Site

- As the number of flights increases, the launch site **success rate** increases. Most of the early flights (flight numbers **<** 30) **started** from CCAFS SLC **40** and were generally unsuccessful.
- The VAFB SLC 4E **flights** also show **that tendency** that earlier flights were less successful.
- **Previous** flights **have not been** launched from KSC LC 39A, so launches from this site are more successful.
- **After about 30 flights,** there are significantly more successful landings **(category** = 1).



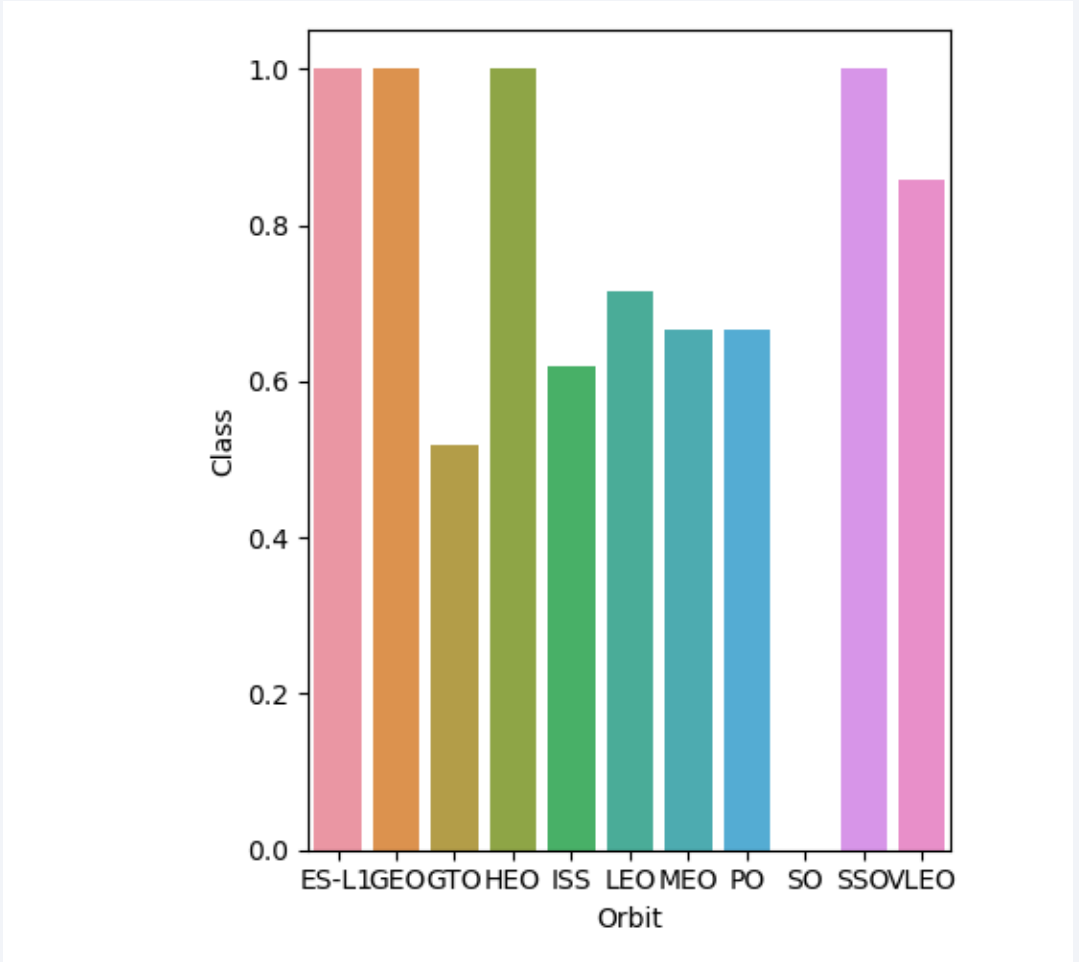
Payload vs. Launch Site

- **There** are very few **failed landings above 7000kg payload,** but there is also **much** less **information about those** heavier launches.
- There is no clear correlation between payload mass and **the** success **of** a **particular** launch site.
- All sites launched **different** payload masses, **and** most of the CCAFS SLC 40 **launches were relatively** lighter (with **a few** outliers).



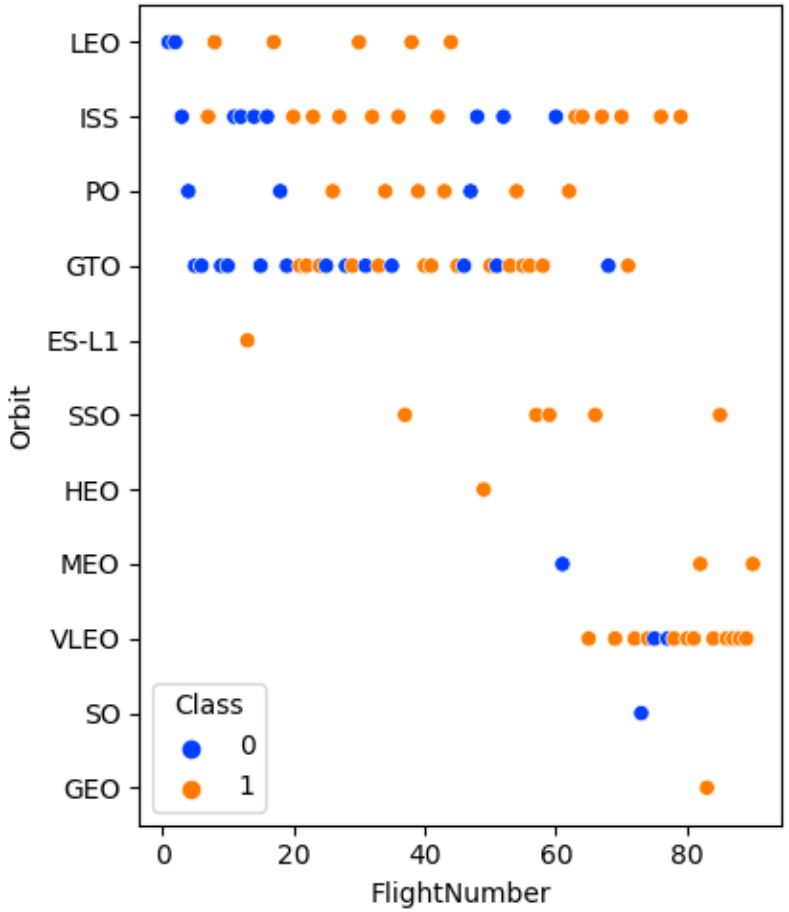
Success Rate vs. Orbit Type

- 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)



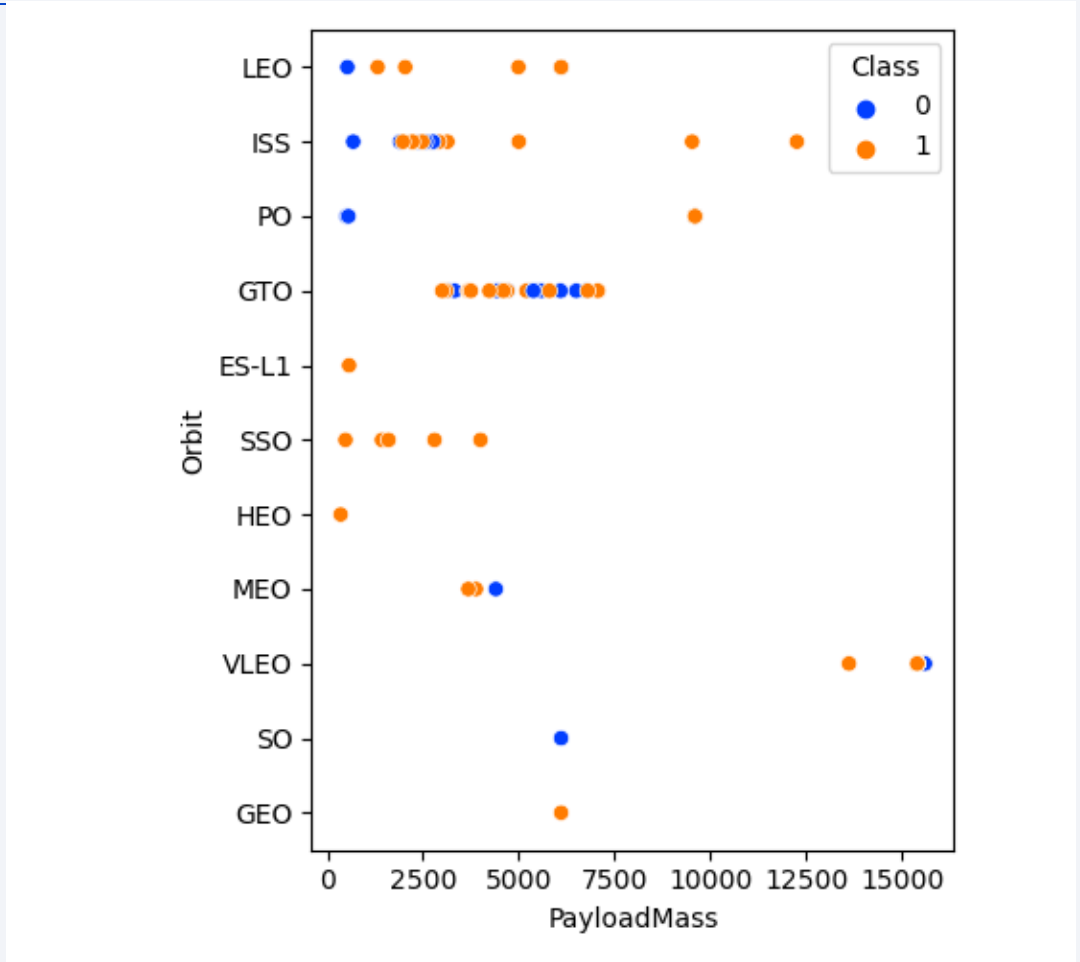
Flight Number vs. Orbit Type

- The 100% success rate of GEO, **HEO** and ES-L1 orbits can be explained by only **one** flight **to** the respective orbits. **SSO's** 100% success rate is more **impressive** with 5 successful flights.
- There is little **correlation** between **flight number** and **GTO success rate.**
- Generally, as **the number of flights** increases, the success rate increases. This is most extreme **in** LEO, where **failed** landings **occurred** only **in** low **flights** (early launches).



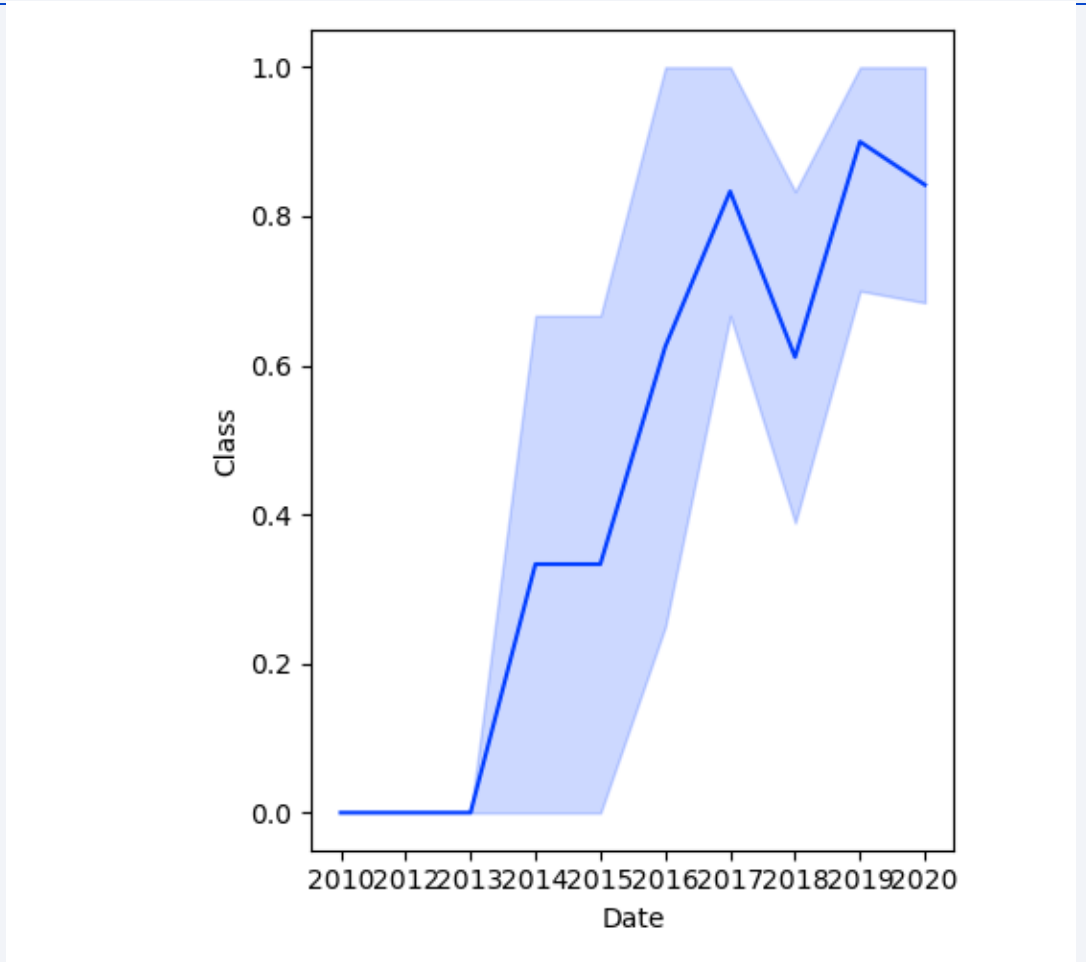
Payload vs. Orbit Type

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



Launch Success Yearly Trend

- 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

- The word `DISTINCT` returns only unique values from the `LAUNCH_SITE` column of the `SPACEXTBL` table.

Task 1

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

In [12]:
```
%sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
```

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

Out[12]:

Launch_Site
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40
None

Launch Site Names Begin with 'CCA'

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

Task 2

Display 5 records where launch sites begin with the string 'CCA'

In [13]:
```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

Out[13]:

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

Total Payload Mass

- 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).

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

In [14]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

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

Out[14]:

TOTAL_PAYLOAD_MASS
45596.0

Average Payload Mass by F9 v1.1

- 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.

Task 4

Display average payload mass carried by booster version F9 v1.1

In [15]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

Out[15]:

AVERAGE_PAYLOAD_MASS
2928.4

First Successful Ground Landing Date

- 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.

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

In [17]:
```
%sql SELECT MIN(DATE) AS FIRST_SUCCESSFUL_GROUND_LANDING FROM SPACEXTBL \
    WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

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

Out[17]:

FIRST_SUCCESSFUL_GROUND_LANDING
01/08/2018

Successful Drone Ship Landing with Payload 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.

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [19]:
```
%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.
```

Out[19]:

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

Total Number of Successful and Failure Mission Outcomes

- 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.

Task 7

List the total number of successful and failure mission outcomes

In [20]:
```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

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

Out[20]:

Mission_Outcome	TOTAL_NUMBER
None	0
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- 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.

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [21]:
```
%sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL \
    WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

Out[21]:

Booster_Version
F9 B5 B1048.4
F9 B5 B1049.4
F9 B5 B1051.3
F9 B5 B1056.4
F9 B5 B1048.5
F9 B5 B1051.4
F9 B5 B1049.5
F9 B5 B1060.2
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

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

Task 9

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [35]:
```
%sql SELECT DATE, BOOSTER_VERSION, LAUNCH_SITE, LANDING_OUTCOME FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Failure (
```

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

Out[35]:

Date	Booster_Version	Launch_Site	Landing_Outcome
01/10/2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
14/04/2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- 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.

Task 10

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

In [42]:
```
%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL \
    WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' \
    GROUP BY LANDING_OUTCOME \
    ORDER BY TOTAL_NUMBER DESC;
```

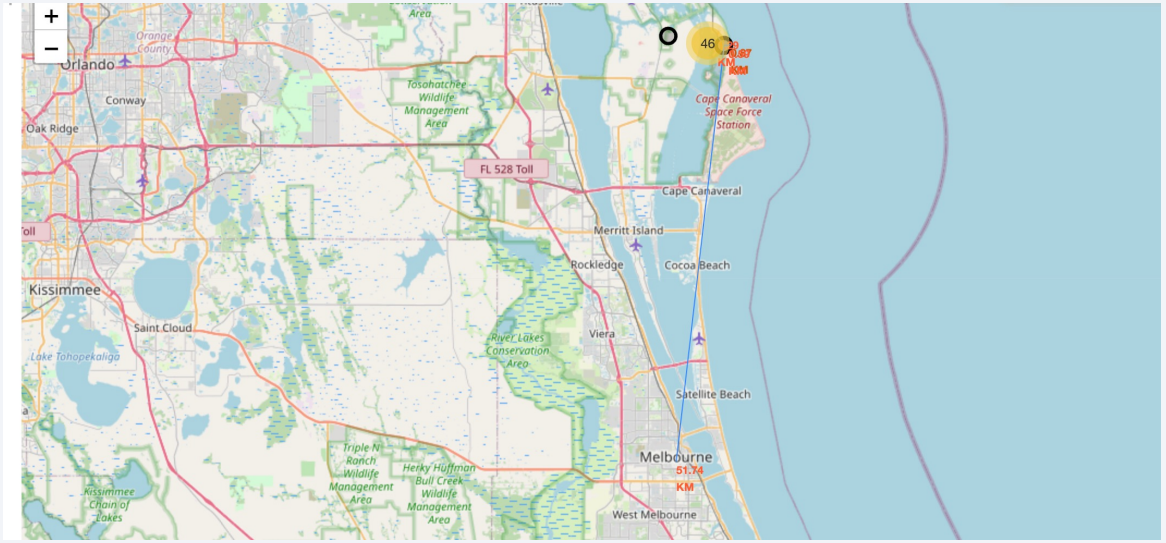
```
 * sqlite:///my_data1.db
Done.
```

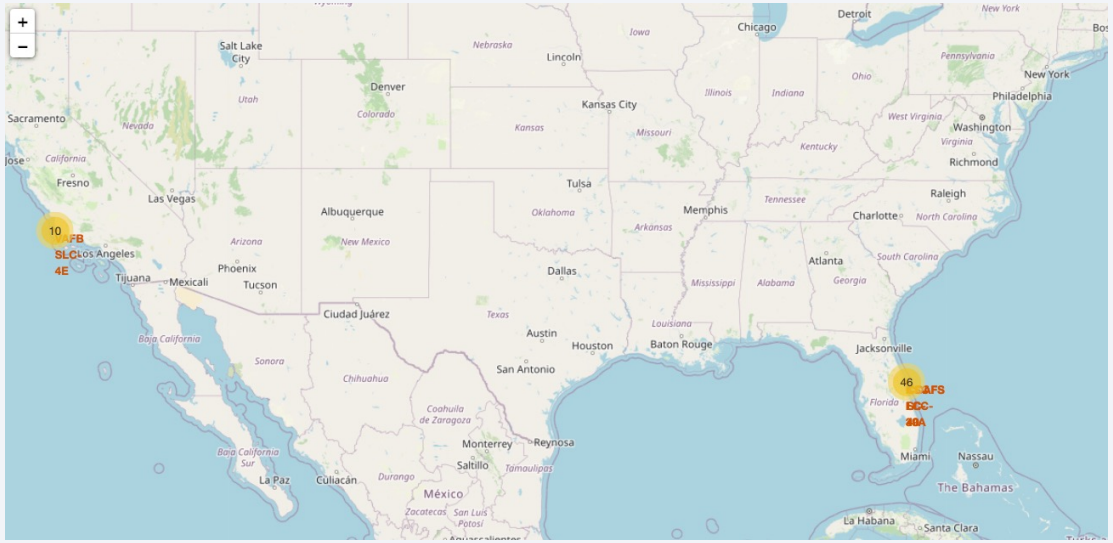
Out[42]:

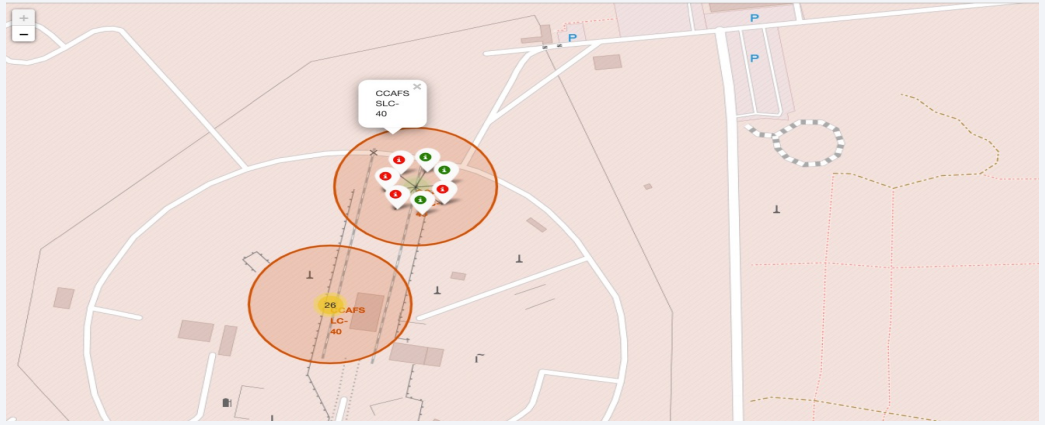
Landing_Outcome	TOTAL_NUMBER
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1


Section 3
Launch Sites
Proximities Analysis

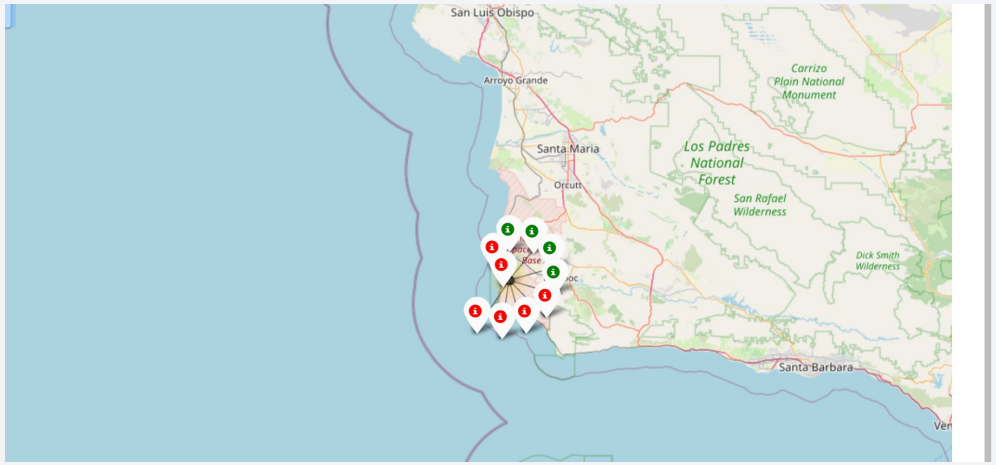
Mapping All Launch Sites

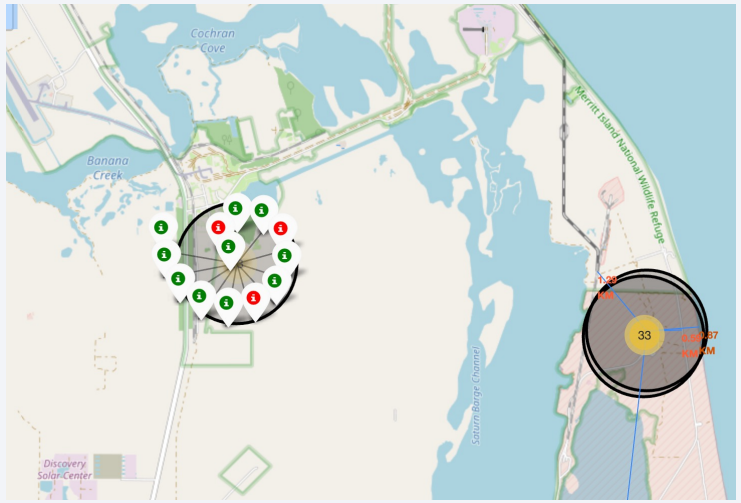




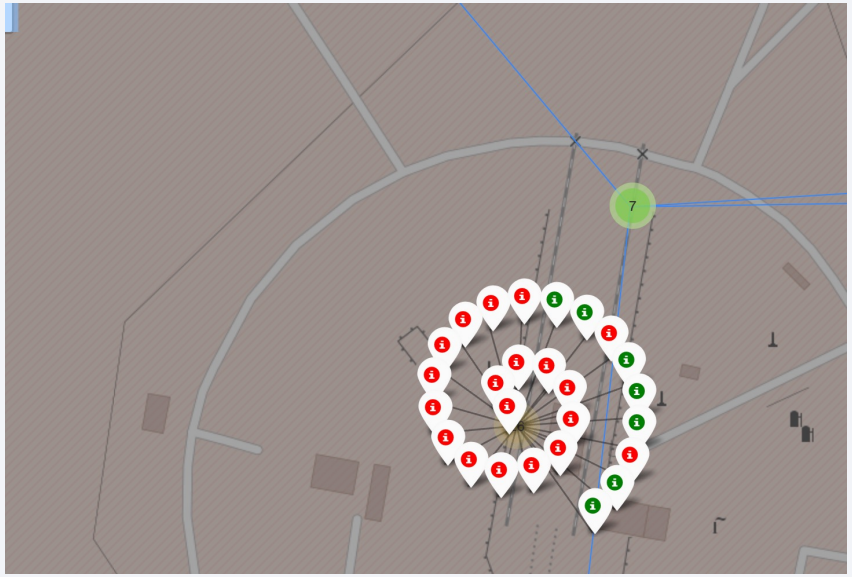


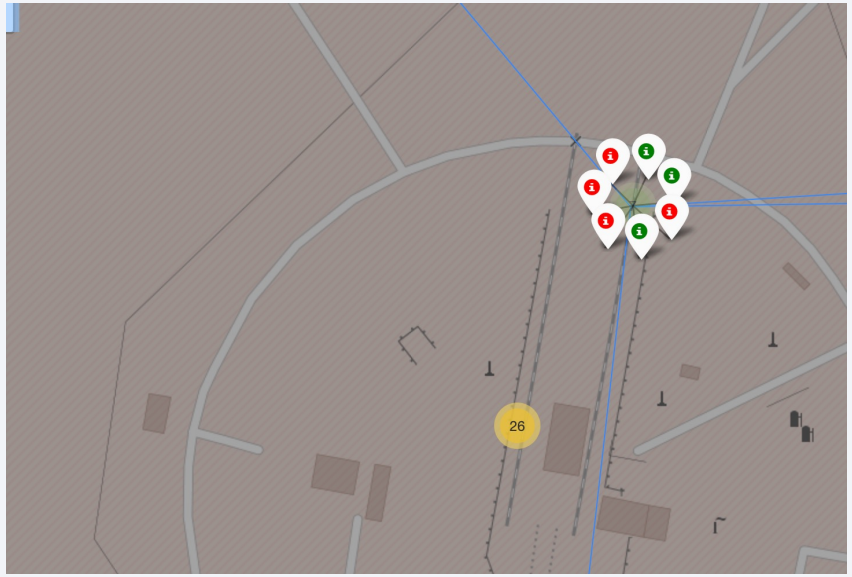
Successful and Failed Launches



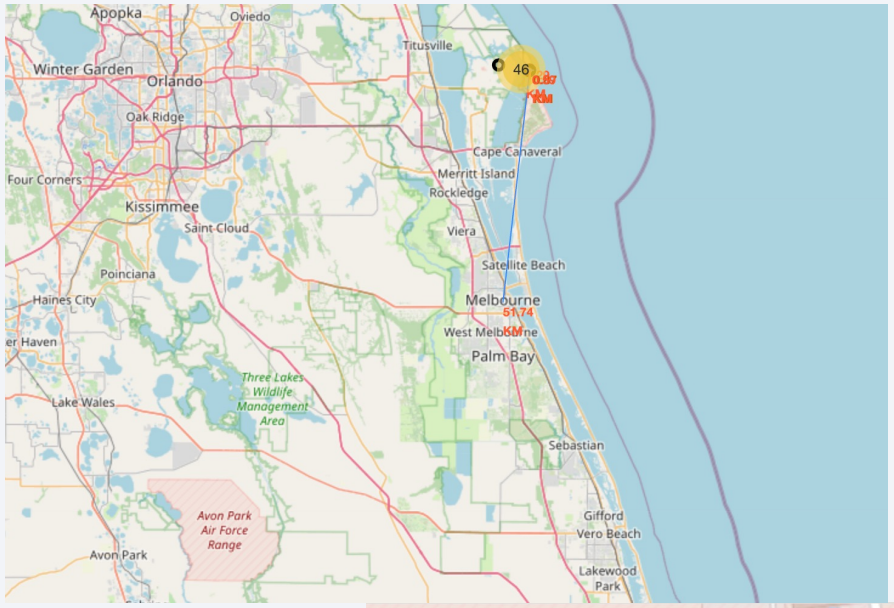


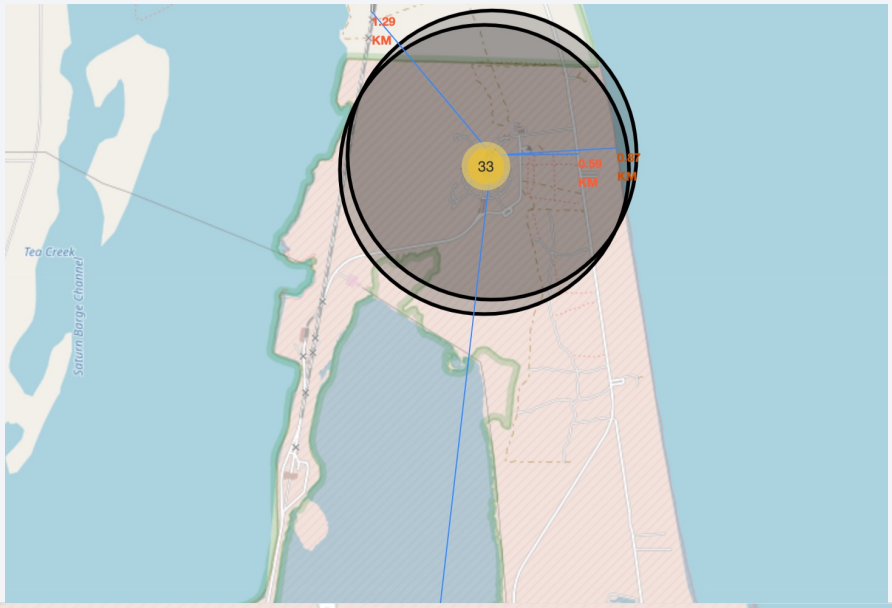
Launches have been grouped into clusters, and annotated with green icons for successful launches, and red icons for failed launches.

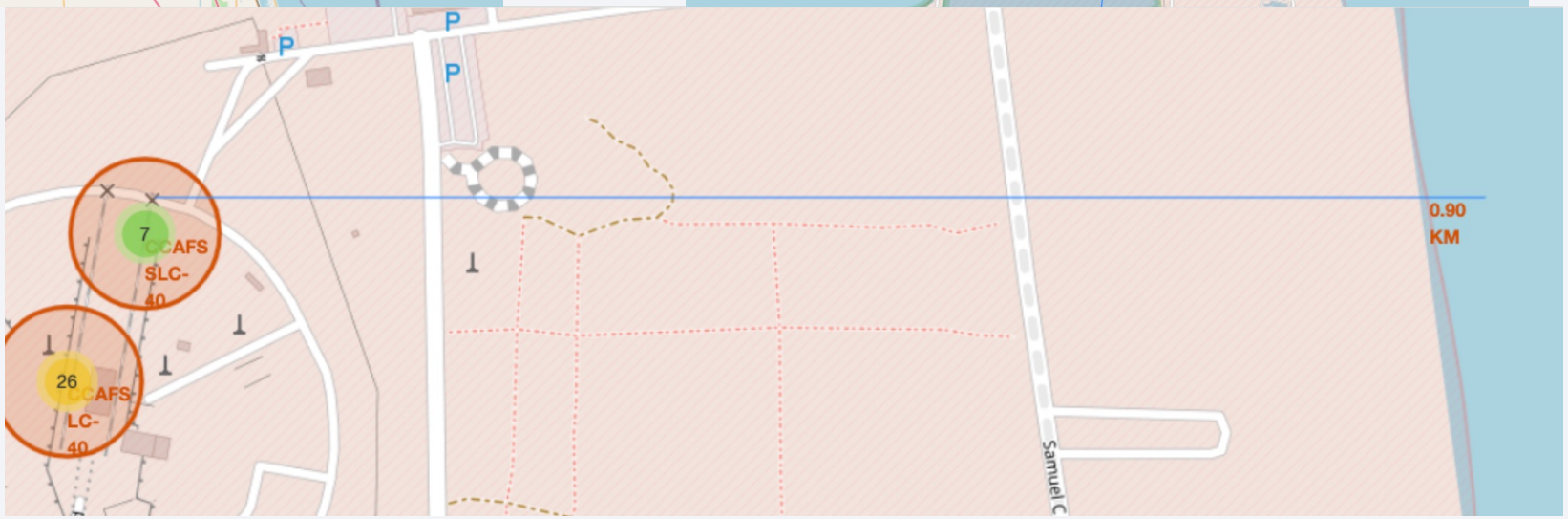




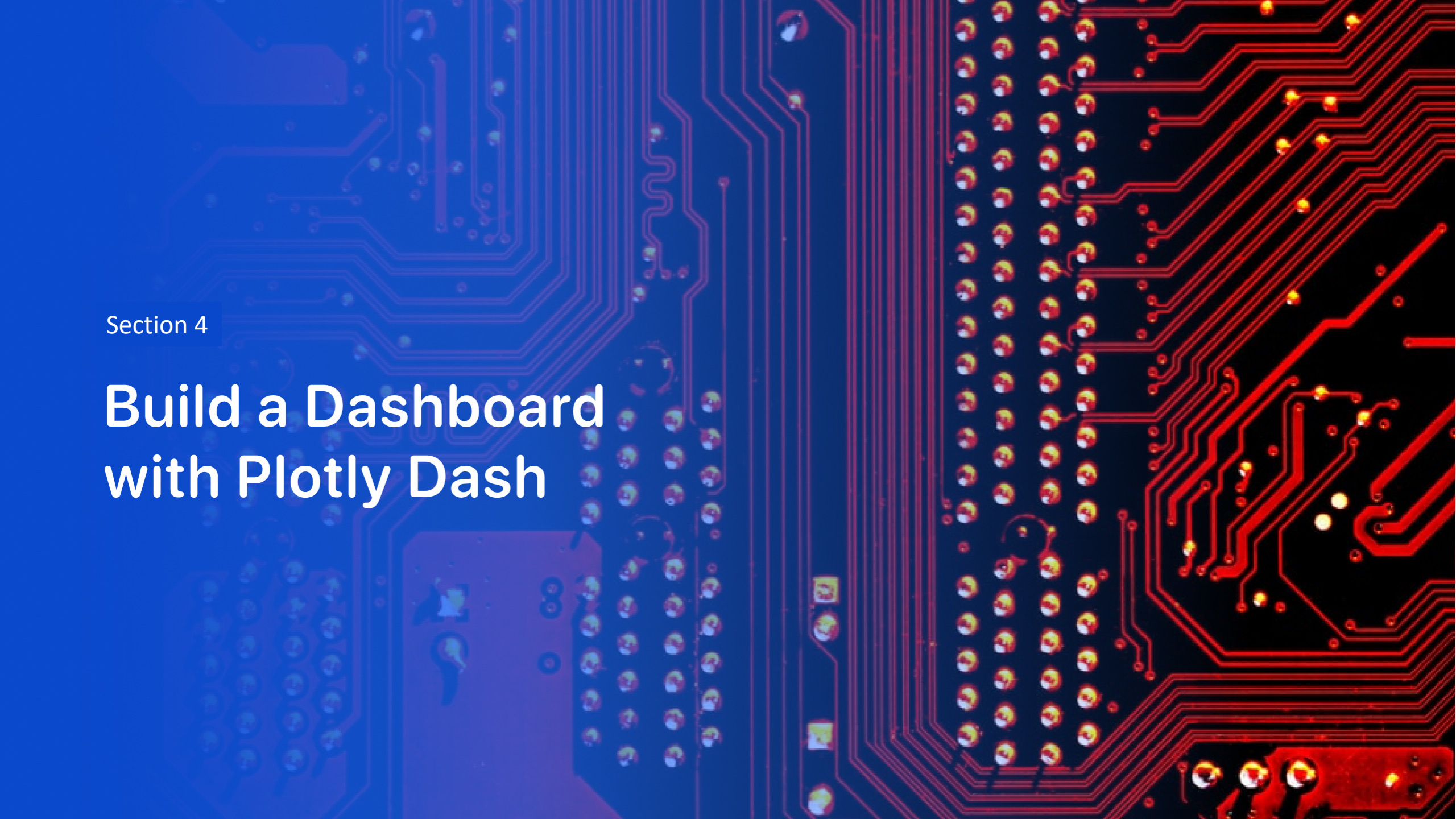
Launch Sites and their Proximity to other Points



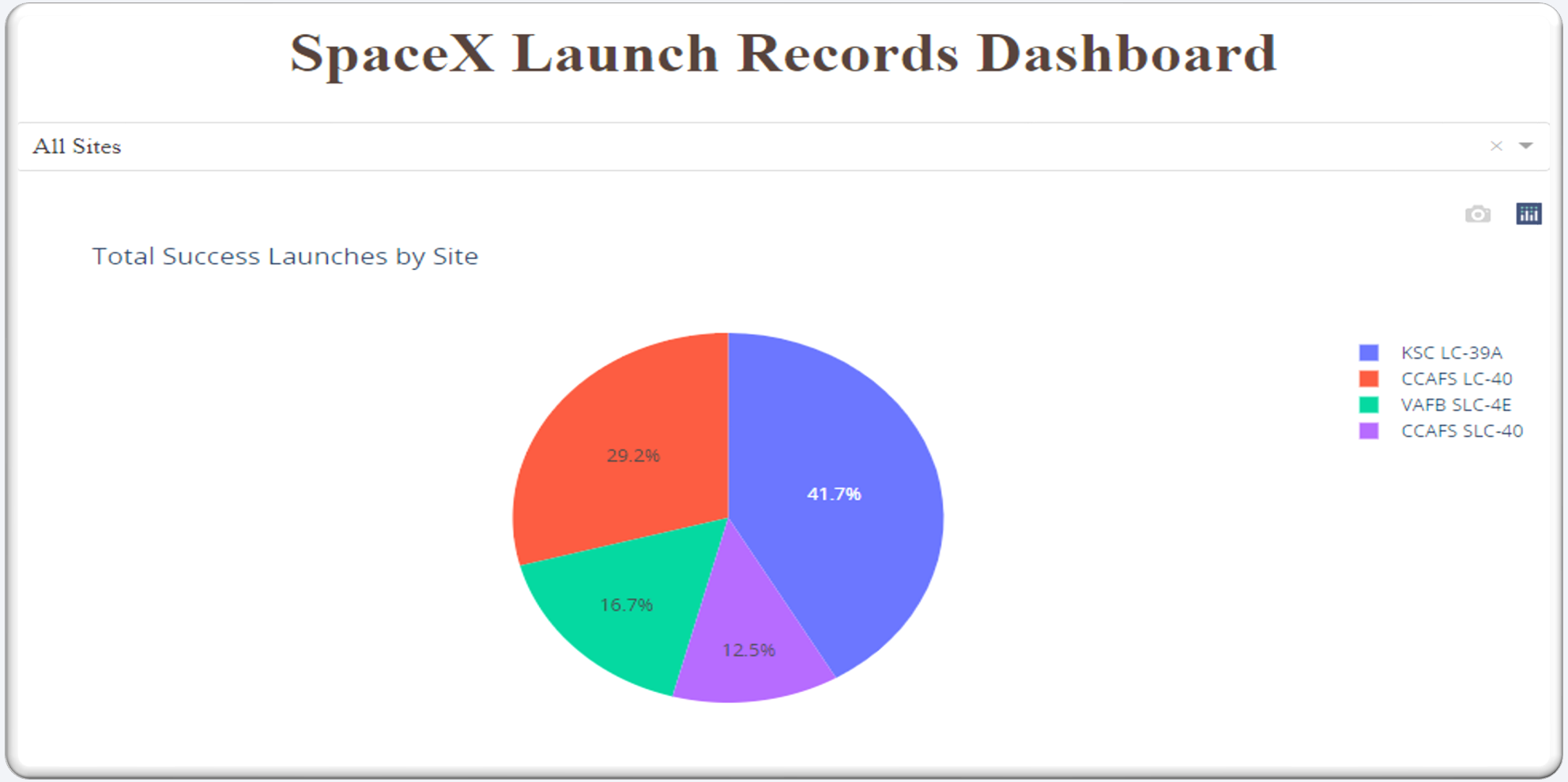




Using the CCAFS SLC-40 launch site as an example site, we can understand more about their placement.


Section 4
Build a Dashboard with Plotly Dash

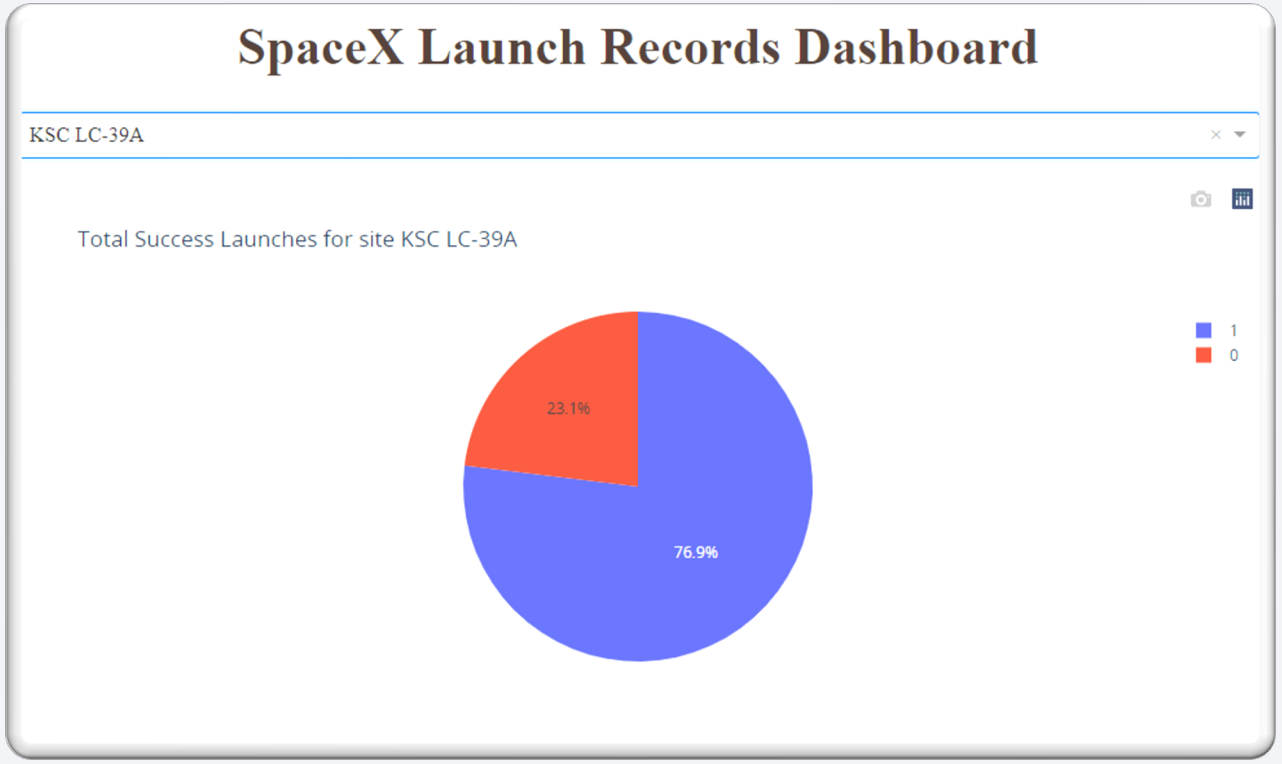
Success Count



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

Highest Success Launch Rate

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



Section 5

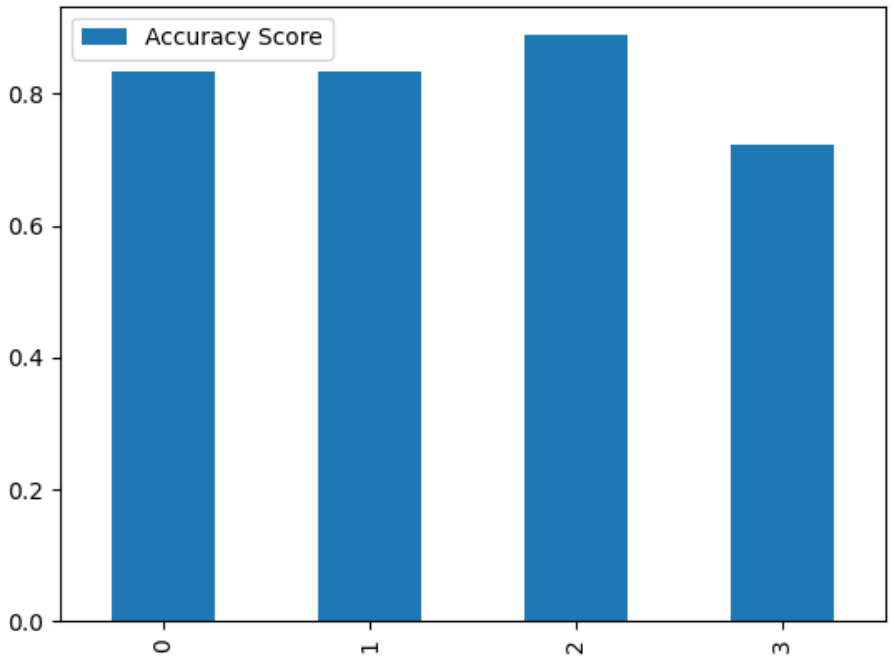
Predictive Analysis (Classification)

Classification Accuracy

Out[34]:

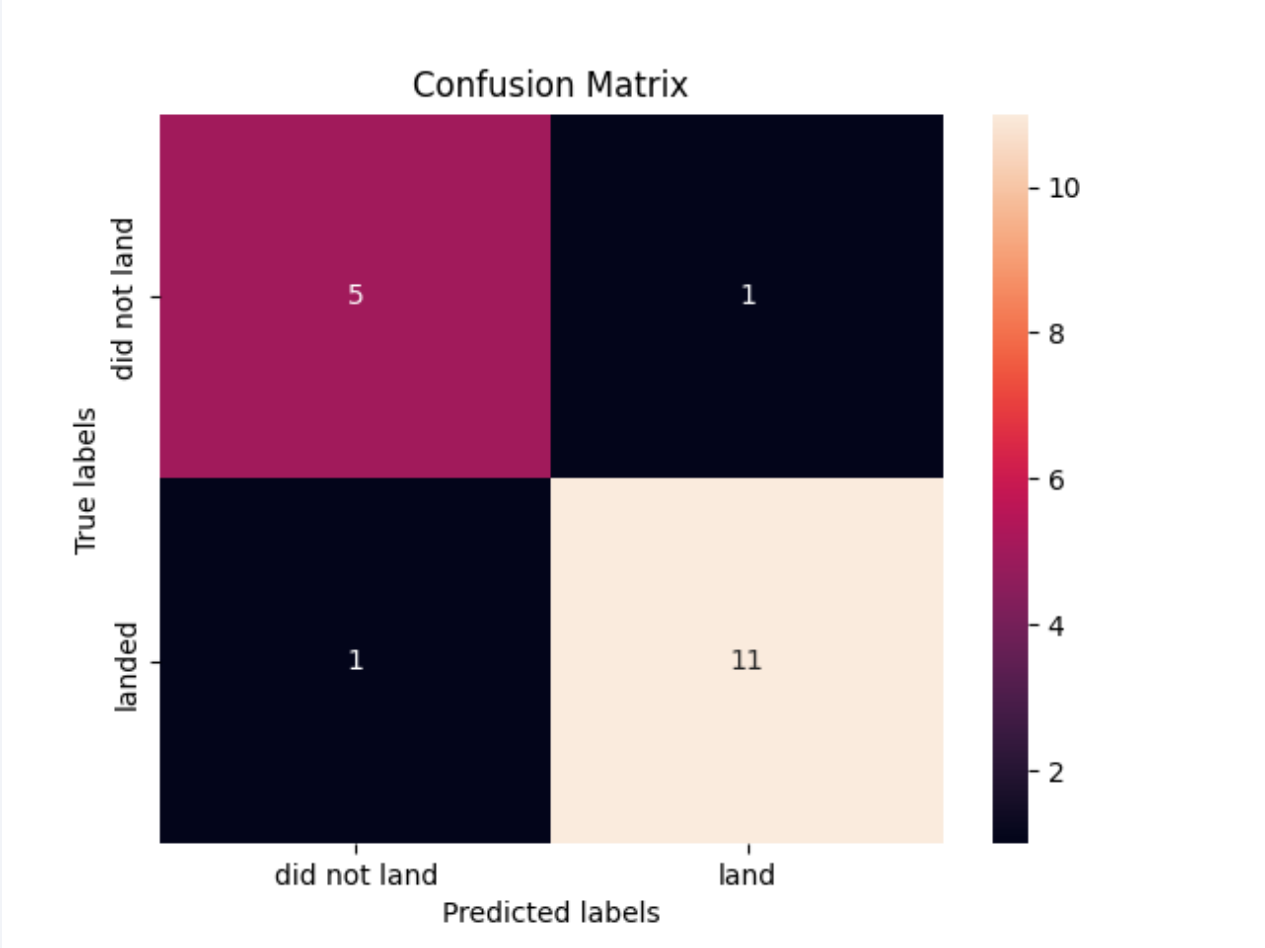
	Algorithm	Accuracy Score
0	Logistic Regression	0.833333
1	Support Vector Machine	0.833333
2	Decision Tree	0.888889
3	K Nearest Neighbours	0.722222

The best model was found to be the Decision Tree Algorithm



Confusion Matrix

- 2 samples were classified incorrectly with 1 being predicted as not landed where it landed and 1 predicted as landed where it did not land.



Conclusions

- With an increase in the number of flights, the likelihood of success at a launch site rises, and early flights tend to have a higher rate of failure.
- Orbit types such as ES-L1, GEO, HEO, and SSO boast a perfect success rate of 100%.
- The success rate for heavy payloads (weighing over 4000kg) is lower compared to that of lighter payloads.
- Among all launch sites, KSC LC-39 A has recorded the highest number of successful launches and the greatest rate of successful launches.
- The best performing classification model is the Decision Tree model.

Appendix

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
In [21]: %sql SELECT DISTINCT(BOOSTER_VERSION) FROM SPACEXTBL \
             WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
 * sqlite:///my_data1.db
Done.
```

Out[21]:

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

Task 9

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

```
In [35]: %sql SELECT DATE, BOOSTER_VERSION, LAUNCH_SITE, LANDING_OUTCOME FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Failure (
 * sqlite:///my_data1.db
Done.
```

Out[35]:

Date	Booster_Version	Launch_Site	Landing_Outcome
01/10/2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
14/04/2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

```
In [14]: %sql SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS FROM SPACEXTBL \
             WHERE CUSTOMER = 'NASA (CRS)';
 * sqlite:///my_data1.db
Done.
```

Out[14]:

TOTAL_PAYLOAD_MASS
45596.0

Task 10

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

```
In [42]: %sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL \
             WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017' \
             GROUP BY LANDING_OUTCOME \
             ORDER BY TOTAL_NUMBER DESC;
 * sqlite:///my_data1.db
Done.
```

Out[42]:

Landing_Outcome	TOTAL_NUMBER
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

Task 2

Display 5 records where launch sites begin with the string 'CCA'

```
In [13]: %sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
 * sqlite:///my_data1.db
Done.
```

Out[13]:

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

Thank you!