IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

João Tomás
26-09-2025

Outline

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

Executive Summary

- **Summary of methodologies:**
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis (EDA)
  - Interactive Visual Analytics
  - Predictive Analysis
- **Summary of results:**
  - EDA results
  - Geospatial analytics
  - Interactive Dashboard
  - Predictive Analysis of classification models

Introduction

- SpaceX lists the price of a Falcon 9 launch on its website at 62 million dollars, whereas competing providers charge upwards of 165 million dollars. A large portion of this cost advantage comes from SpaceX’s ability to reuse the rocket’s first stage.
- This means that by knowing whether the first stage will successfully land, we can estimate the overall cost of a launch. Such insight would be valuable for another company looking to compete with SpaceX when bidding for launch contracts.
- The goal of this project is to predict if the SpaceX Falcon 9 will land successfully

Section 1

Methodology

Methodology

Data collection methodology:

- GET requests to the SpaceX REST API & Web Scraping
- Perform data wrangling
  - NaN values were removed using the .fillna( ) method
  - The method .value_counts ( ) was used to determine
    - the number of launches per site
    - the number of each orbit
    - the number of mission outcomes per orbit
  - Creating labels for landing outcomes (1 for successful landing, 0 for unsuccessful landing)

- Perform exploratory data analysis (EDA) using visualization and SQL
  - Manipulate and evaluate SpaceX dataset using SQL
  - Using Pandas and MatplotLib libraries to visualize relationships between variables and find patterns
- Perform interactive visual analytics using Folium and Plotly Dash
  - Using Folium for geospatial analytics
  - Creating a dashboard using Plotly Dash
- Perform predictive analysis using classification models
  - Using Scikit-Learn:
    - Standardize data
    - Split data into training and test datasets
    - Train classification models
    - Use GridSearchCV to find hyperparameters
  - Plot confusing matrices and verify the accuracy of the models

Data Collection

- Data was collected using the SpaceX API and Web Scraping
- The GET response from the REST API was decoded as a JSON using the .json() function and turned into a Pandas dataframe using the .json_normalize()
- The data was cleaned and checked for missing values
- BeautifulSoup was used for Web Scraping from Wikipedia for records on the Falcon9 launches.
- The data was converted into a dataframe.

Data Collection – SpaceX API

- The GET request was used to collect data from the SpaceX API, then the data was cleaned. Lists for the data were defined and the functions to retrieve data and fill lists were called.
- The lists were used to construct the dataset and to create the Pandas Dataframe

- Github link:

https://github.com/JoaoMTomas/IBM-Data-Science-Capstone-Project/blob/main/Lab%201%20Collecting%20the%20data.ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

[10]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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

[11]:
```
response=requests.get(static_json_url)
```

[12]:
```
response.status_code
```

[12]:
```
200
```

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

[13]:
```
data = response.json()
data = pd.json_normalize(data)
```

Using the dataframe `data` print the first 5 rows

[32]:
```
# Get the head of the dataframe
print(data.head(1))
```

```
                      rocket                  payloads  \
0  5e9d0d95eda69955f709d1eb  5eb0e4b5b6c3bb0006eeb1e1

                  launchpad  \
0  5e9e4502f5090995de566f86

                                                                                                                                                                          cores  \
0  {'core': '5e9e289df35918033d3b2623', 'flight': 1, 'gridfins': False, 'legs': False, 'reused': False, 'landing_attempt': False, 'landing_success': None, 'landing_type': None, 'landpad': None}

   flight_number                  date_utc        date
0              1  2006-03-24T22:30:00.000Z  2006-03-24
```

You will notice that a lot of the data are IDs. For example the rocket column has no information about the rocket just an identification number.

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

[15]:
```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

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

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

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

# Using the date we will restrict the dates of the launches
```

Data Collection - Scraping

- Web Scraping for the Falcon9 launch records was performed using BeautifulSoup
- The HTML table was parsed and converted to a dataframe
- Github link: https://github.com/JoaoMTomas/IBM-Data-Science-Capstone-Project/blob/main/Lab%201Web%20scraping%20Falcon%209%20and%20Falcon%20Heavy%20Launches%20Records%20from%20Wikipedia.ipynb

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 [5]:
```
# use requests.get() method with the provided static_url and headers
# assign the response to a object
response = requests.get(static_url, headers=headers)
print(response.status_code)
```

```
200
```

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

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

In [7]:
```
print(soup.title.string)
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [8]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

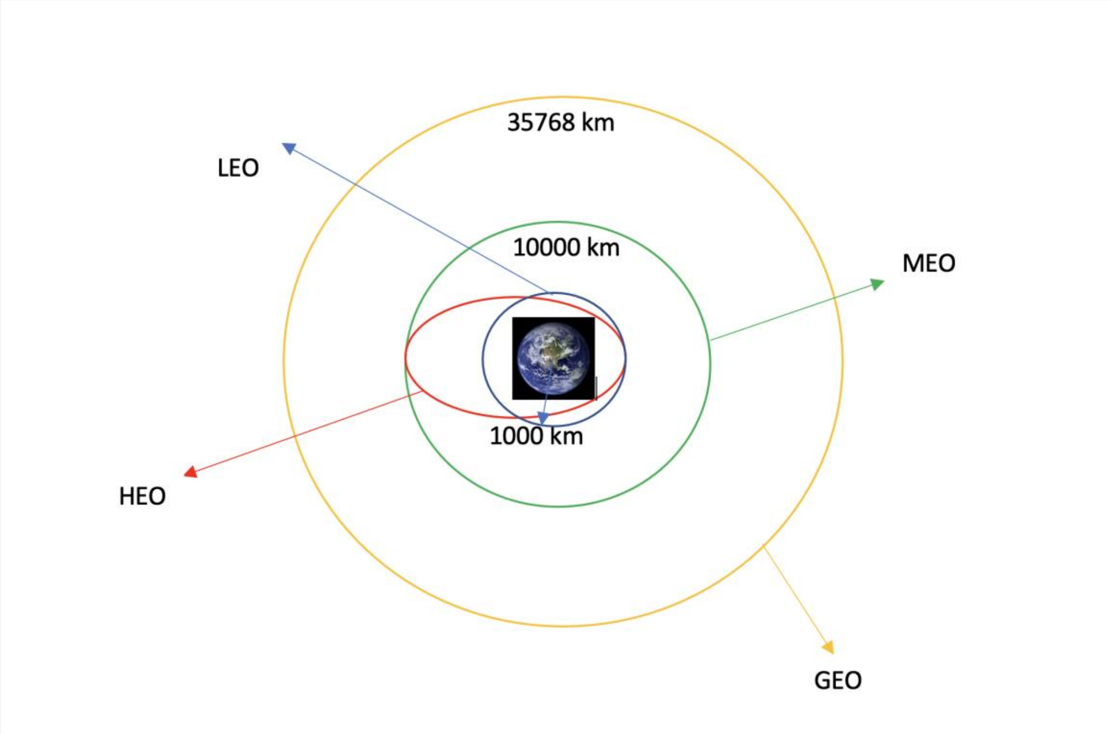
Starting from the third table is our target table contains the actual launch records.

In [9]:
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
```

Data Wrangling

- First it was performed the exploratory data analysis nad the training labels were defined.
- Then the number of launches per site, number of orbits and the outcomes were calculated
- The landing outcome label was exported to a csv

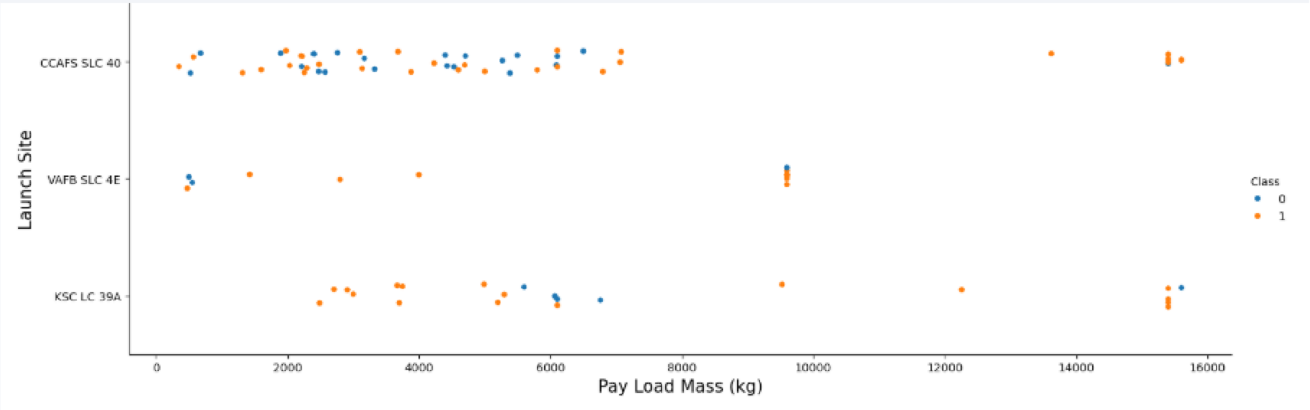


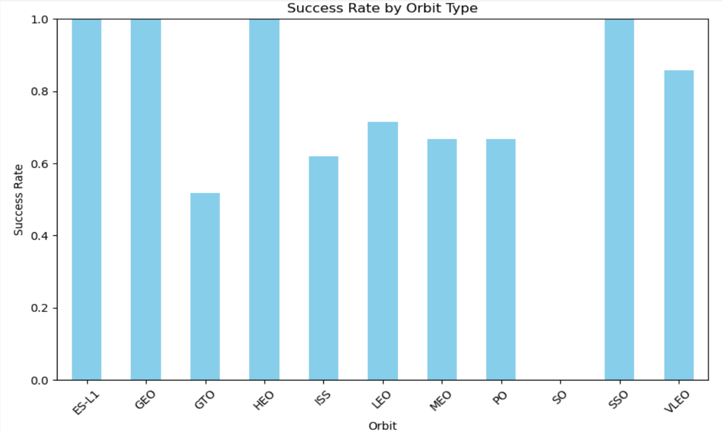
- Github link:

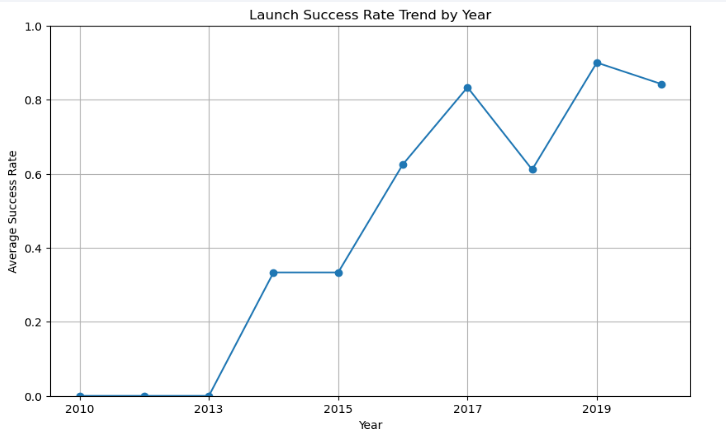
https://github.com/JoaoMTomas/IBM-Data-Science-Capstone-Project/blob/main/Lab%202%20Data%20wrangling.ipynb

EDA with Data Visualization

- The data was explored by visualizing the relationship between flight number and launch sites, payload and launch sites, success rate of each orbit type, flight number and orbit type, the launch success yearly trend, using scatter charts, bar charts and line charts.







- Github link:

- https://github.com/JoaoMTomas/IBM-Data-Science-Capstone-Project/blob/main/Hands-on%20Lab%20Complete%20the%20EDA%20with%20Visualization.ipynb

EDA with SQL

- SQL queries were used to get insights from the data, namely:
  - Display the unique launch site names.
  - Display 5 records where launch sites begin with `'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 of the **first successful ground pad landing**.
  - List the booster versions with **successful drone ship landings** and payload mass between **4001 and 5999 kg**.
  - Count the total number of **successful** and **failure mission outcomes**.
  - List booster versions that carried the **maximum payload mass**.
  - Display **month, failure drone ship landings, booster version, and launch site** for the year **2015**.
  - Rank the **landing outcomes** (count of each type) between **2010-06-04 and 2017-03-20**, in descending order.

- Github link:

https://github.com/JoaoMTomas/IBM-Data-Science-Capstone-Project/blob/main/Assignment%20SQL%20Notebook%20for%20Peer%20Assignment.ipynb

Build an Interactive Map with Folium

- All launch sites were marked, and map objects such as **markers, circles, and lines** were added on a **Folium map** to indicate the success or failure of launches at each site. Launch outcomes were encoded as **0 for failure** and **1 for success**. Using **color-labeled marker clusters**, sites with relatively high success rates were identified. Distances from each launch site to nearby features were calculated, addressing questions such as:
    - Are launch sites near railways, highways, and coastlines?
    - Do launch sites maintain a certain distance from cities?
- An **interactive map** was built using **Folium**.

- Github link:

https://github.com/JoaoMTomas/IBM-Data-Science-Capstone-Project/blob/main/Launch%20Sites%20Locations%20Analysis%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- An interactive dashboard was built using **Plotly Dash**.
- It features **pie charts** showing the total launches by different sites and a **scatter plot** illustrating the relationship between mission outcomes and payload mass (kg) across various booster versions.

- Github link:

https://github.com/JoaoMTomas/IBM-Data-Science-Capstone-Project/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

- The data was **loaded using NumPy and Pandas**, then transformed and split into **training and testing sets**.
- Various **machine learning models** were built, with hyperparameters tuned using **GridSearchCV**. **Accuracy** was used as the evaluation metric, and the models were improved through **feature engineering** and **algorithm tuning**.
- The **best-performing classification model** was identified.
- Github link:

https://github.com/JoaoMTomas/IBM-Data-Science-Capstone-Project/blob/main/Hands%20on%20Lab%20Complete%20the%20Machine%20Learning%20Prediction%20lab.ipynb

Results

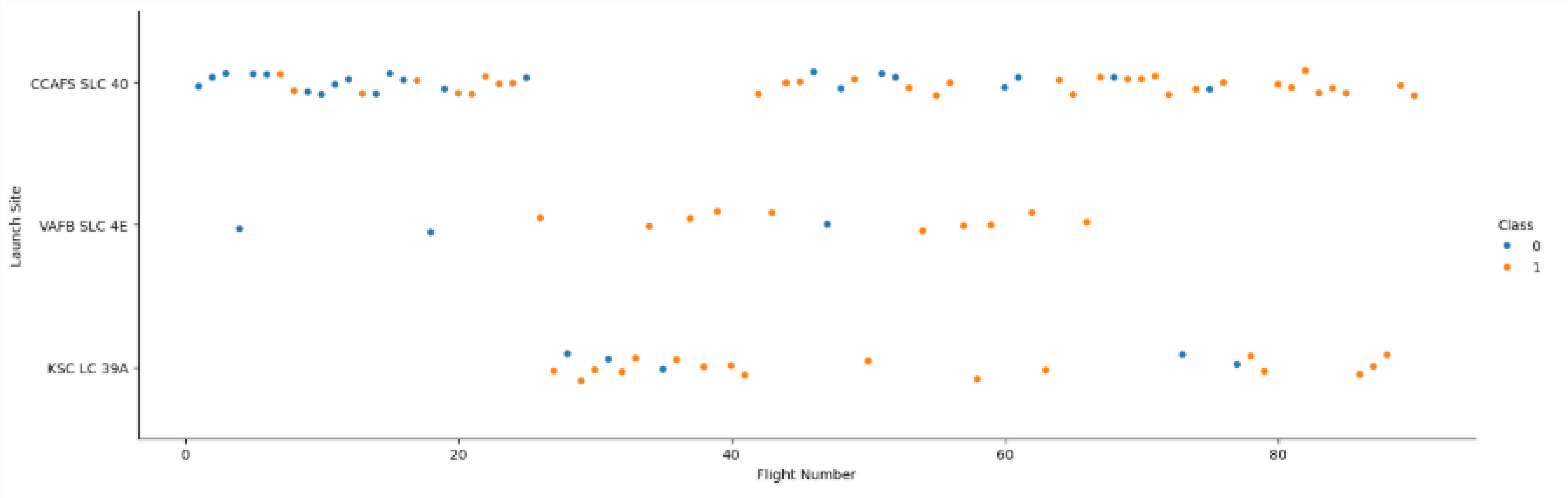
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

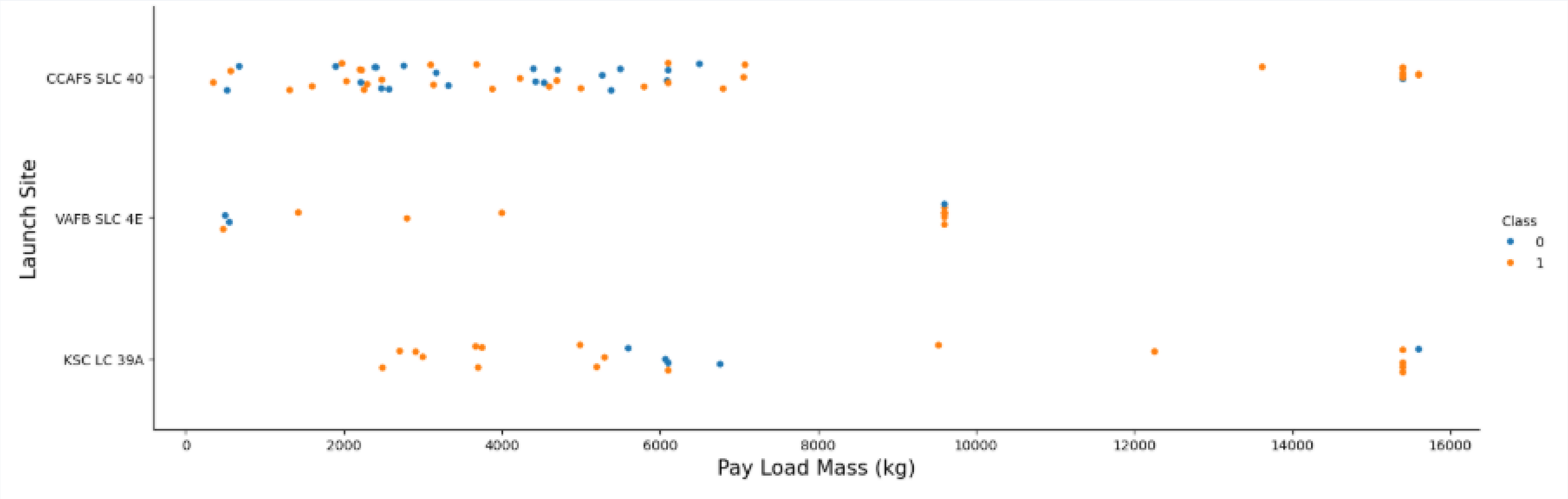
Flight Number vs. Launch Site

- The graph demonstrates that success rate at a launch site increases with the number of flights at that site



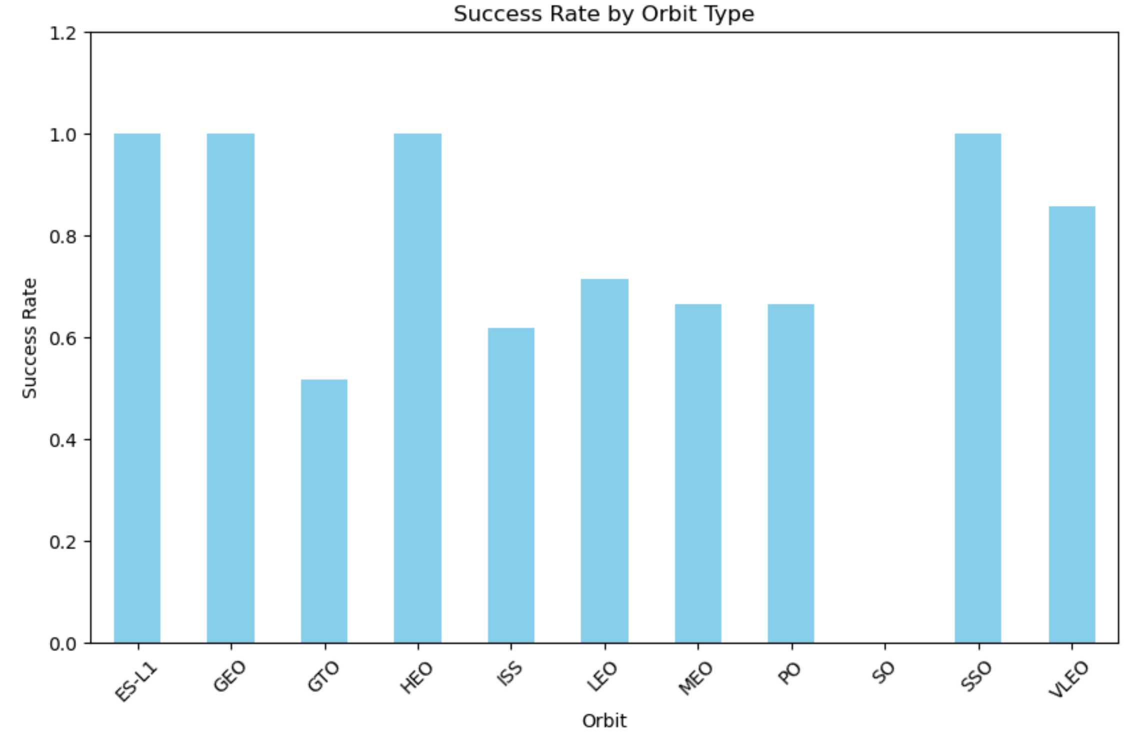
Payload vs. Launch Site

- The graph shows that the increase of payload translates to less successful landings, however there is no correlation between payload and the success rate of a specific launch site



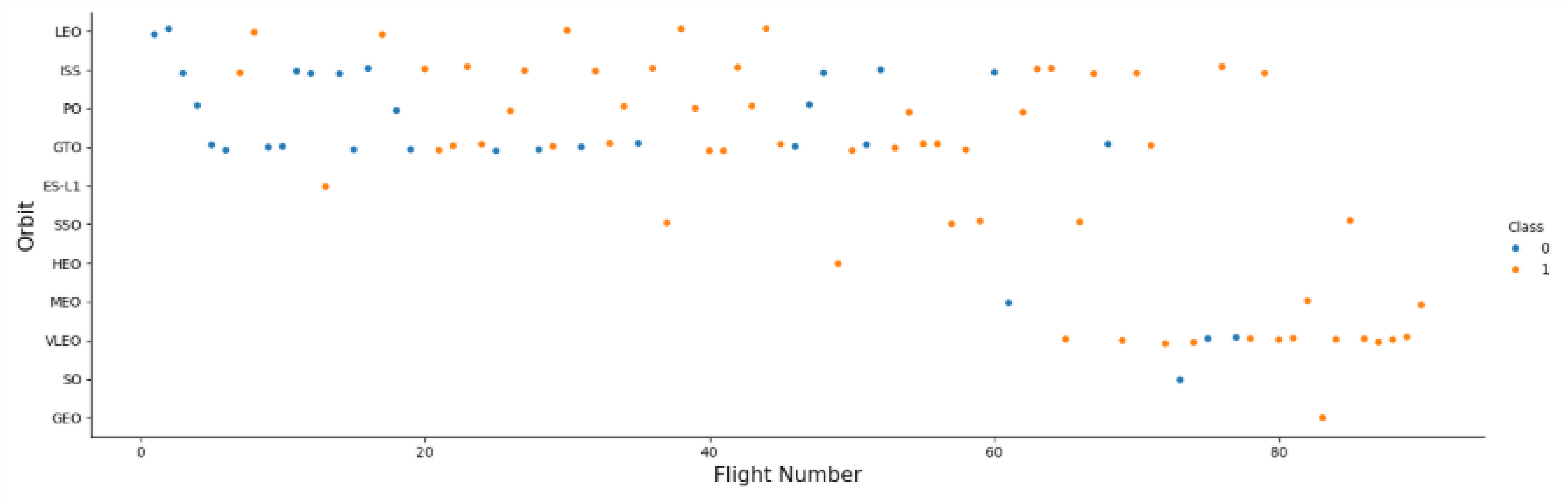
Success Rate vs. Orbit Type

- The bar chart shows that 4 orbits present a 100% success rate, with one having a 0% success rate



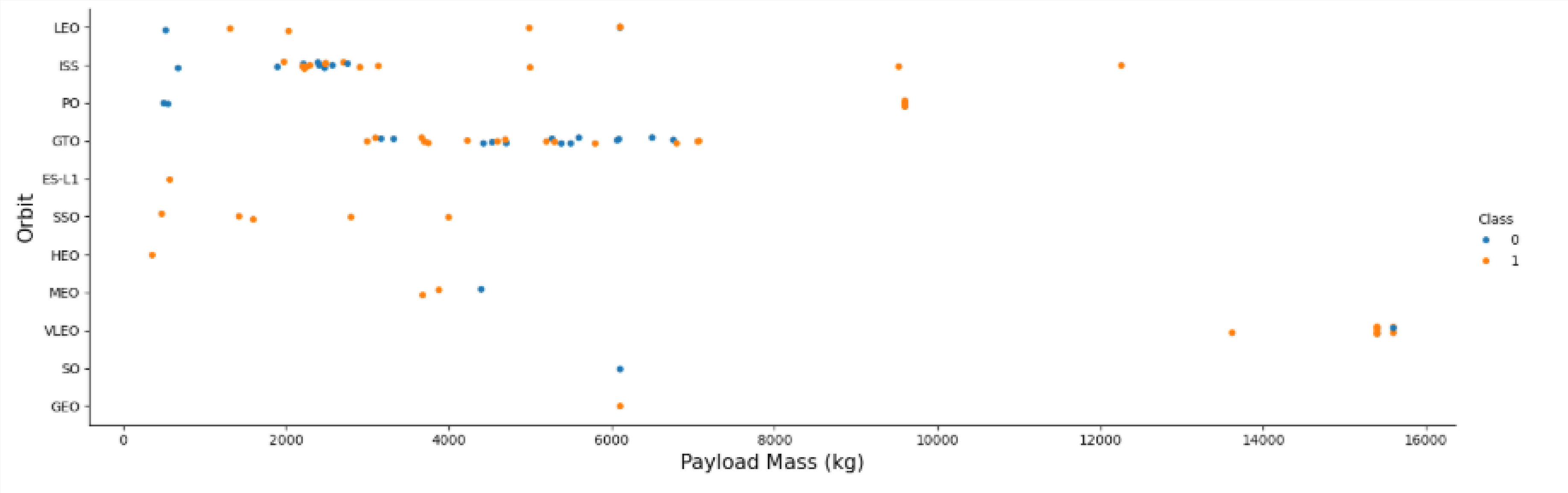
Flight Number vs. Orbit Type

- The graph indicates that the 100% success rate of the orbits GEO, HEO and ES-L1 is due to only having 1 flight whilst the SSO as 100% success rate with 5 flights.



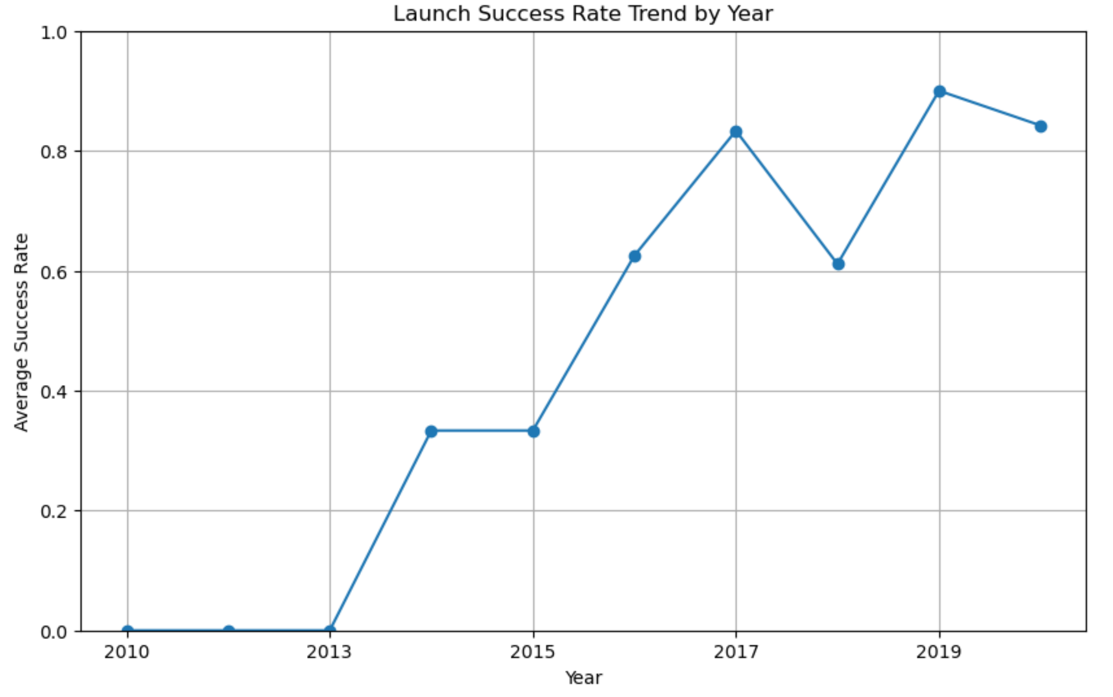
Payload vs. Orbit Type

- The chart demonstrates that the PO, ISS and LEO have the higher success rate with heavy payloads



Launch Success Yearly Trend

- The line graph indicates a success rate of 0% between 2010 and 2013 followed by an increase over the following years
- After 2016 the success rate stayed always above 50%



All Launch Site Names

- The query retrieves all the unique values from the Launch_Site column

```
%%sql
SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;
```

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

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

Launch Site Names Begin with 'CCA'

- The LIKE keyword followed by CCA% retrieves the values that begin with CCA independently of the rest of the value
- The LIMIT 5 limits the records to the first 5 in the table

```
%%sql
SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The SUM keyword calculates the sum of the values in the Payload_Mass_kg_ column, the WHERE keyword limits to when the Customer is NASA (CRS)

```
%%sql
SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass
FROM SPACEXTABLE
WHERE "Customer" = 'NASA (CRS)';
```

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

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

- The AVG keyword calculates the average value of the Payload_Mass_kg_ column, the WHERE keyword limits the values to where the Booster version is the F9 v1.1

Task 4

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

```
%%sql
SELECT AVG("Payload_Mass__kg_") AS Average_Payload_Mass
FROM SPACEXTABLE
WHERE "Booster_Version" Like 'F9 v1.1%';
```

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

Average_Payload_Mass
2534.6666666666665

First Successful Ground Landing Date

- To find the smaller date (hence the first date), the MIN keyword is used to retrieve the smaller value.
- The WHERE keyword is used to limit the values to when the Landing Outcome is successful.

```
%%sql
SELECT MIN("Date") AS First_Successful_Landing
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (ground pad)';
```

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

First_Successful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The query retrieves the Successful Drone Ship Landing with a Payload between 4000 and 6000 by:
  - Using WHERE to limit the values to the Successful (drone ship)
  - Followed by AND and BETWEEN that add the condition for the payload to be between the 4000 and 6000 values

```
%%sql
SELECT "Booster_Version" FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (drone ship)' AND "Payload_Mass__kg_" BETWEEN 4001 AND 5999;
```

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

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

Total Number of Successful and Failure Mission Outcomes

- SUM (CASE WHEN keyword is used to calculate the number of mission outcomes for each condition

```
%%sql
SELECT
    SUM(CASE WHEN "Mission_Outcome" LIKE 'Success%' THEN 1 ELSE 0 END) AS Success_Count,
    SUM(CASE WHEN "Mission_Outcome" LIKE 'Failure%' THEN 1 ELSE 0 END) AS Failure_Count
FROM SPACEXTABLE
```

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

Success_Count	Failure_Count
100	1

Boosters Carried Maximum Payload

- The query retrieves the boosters that have carried the maximum payload, using keywords as seen above

```
%%sql
SELECT "Booster_Version","Payload_Mass__kg_"
FROM SPACEXTABLE
WHERE "Payload_Mass__kg_" = (SELECT MAX("Payload_Mass__kg_")
    FROM SPACEXTABLE);
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- The query retrieves the failed landing outcomes in drone ships for the year 2015
- The keyord As Month translates the dates to the months

```
%%sql
SELECT substr(Date, 6,2) AS Month, "Landing_Outcome", "Booster_Version","Launch_Site"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE 'Failure (drone ship)%' AND substr(Date,0,5)='2015';
```

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

Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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
- The keywords GROUP BY and DESC order the landing_outcomes by count, retrieving the rank of landing outcomes

```
%%sql
SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count
FROM SPACEXTABLE
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY Outcome_Count DESC;
```

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

Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

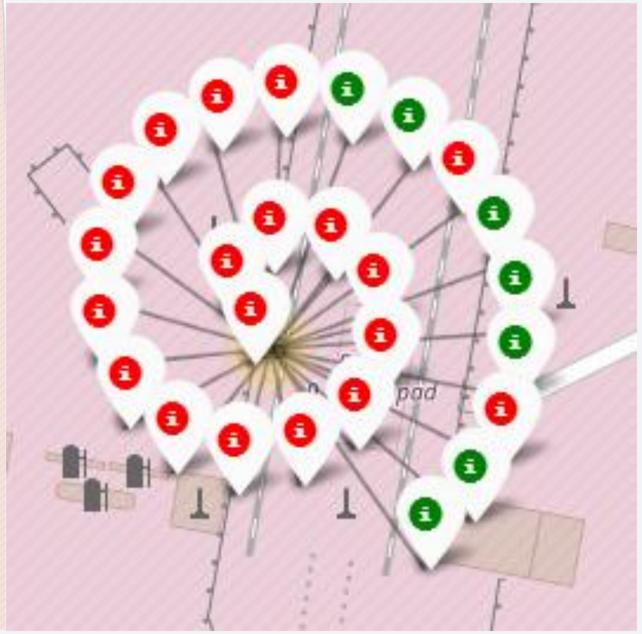
Launch Sites Proximities Analysis

Global map – Launch Sites

- All SpaceX launches are on the east and west coasts of the USA only

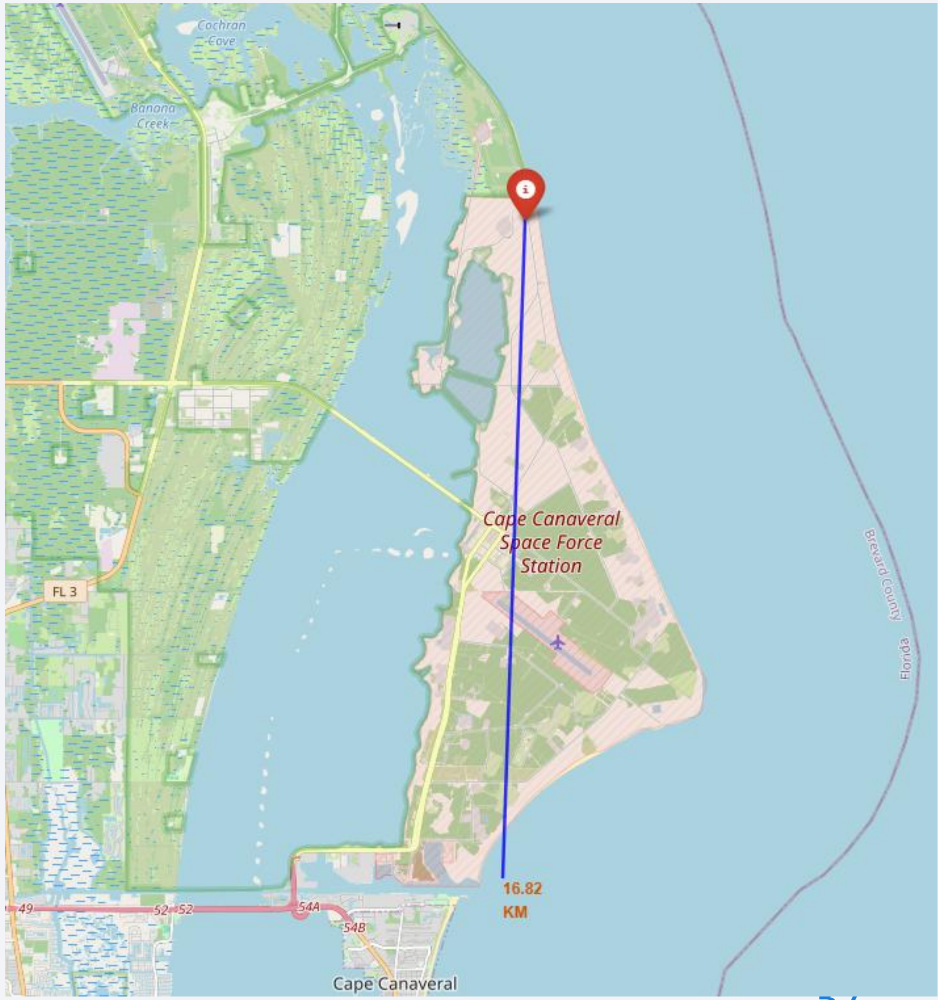
Success/Failed launches marker identification

- The markers are shown in clusters and when zoomed they reveal the successful and failed launches



Proximity and distance calculation of launch site to a defined coastline

The image shows the distance between the launch site and a defined point in the coastline, revealing a straight line between the points and the distance shown in the map

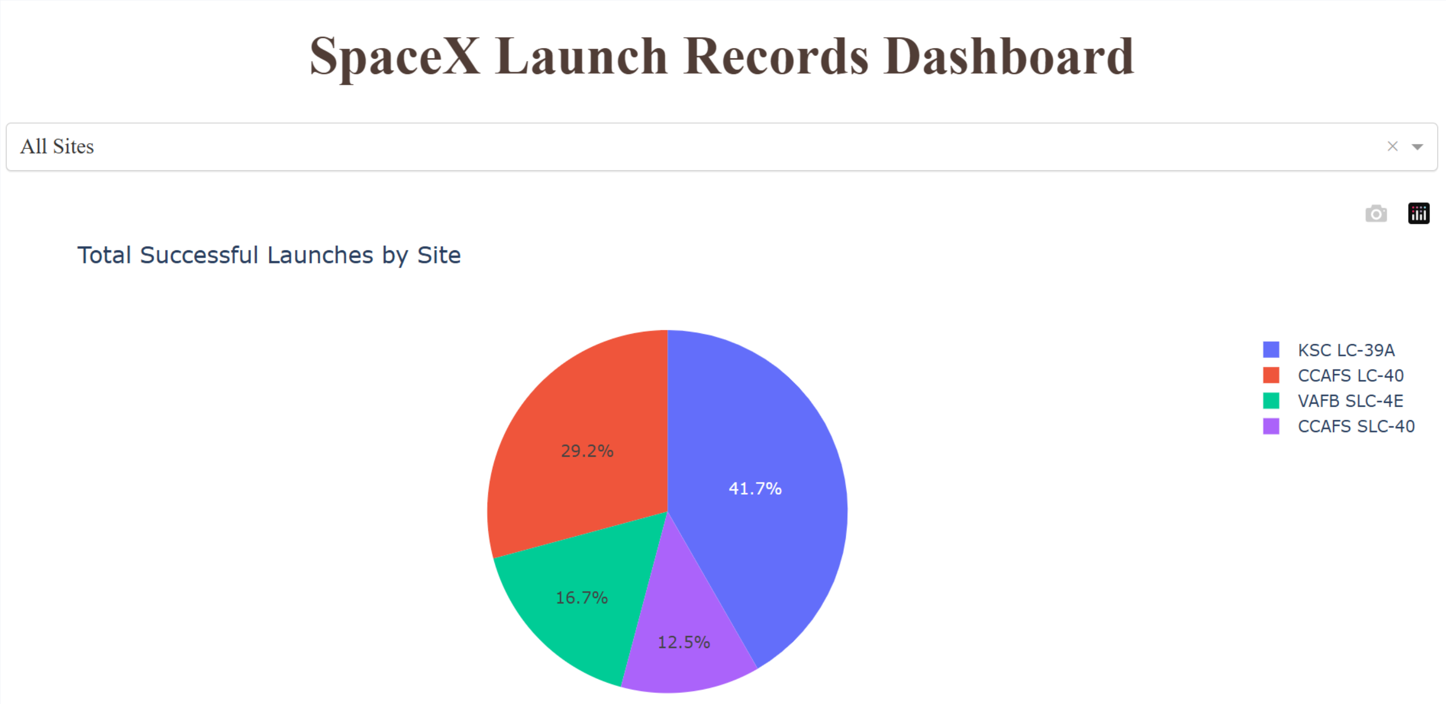


Section 4

Build a Dashboard with Plotly Dash

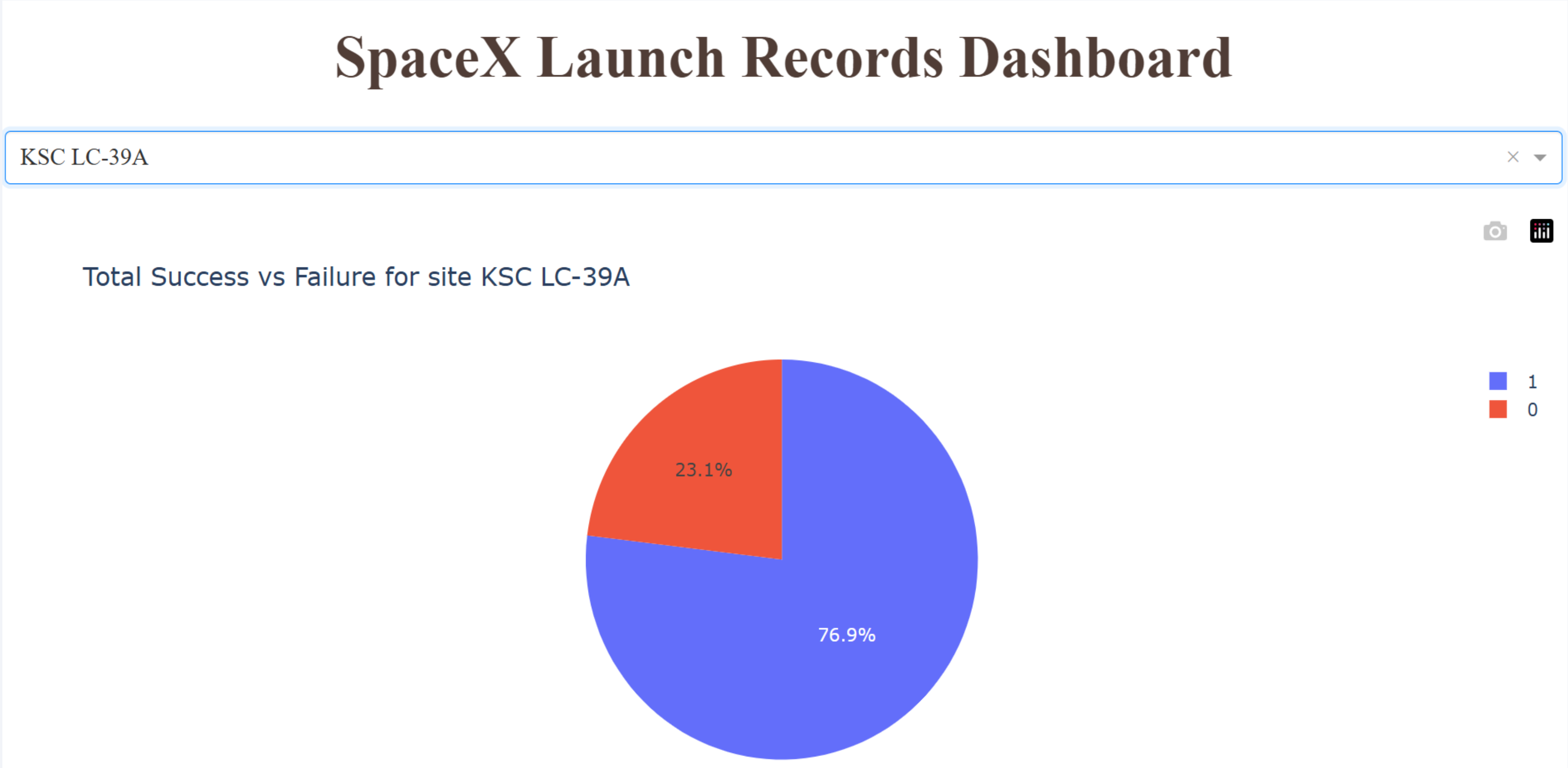
Launch Success count for all sites

- The graph shows the distribution of successful launch sites across all launch sites with KSC LC-39A being the most successful



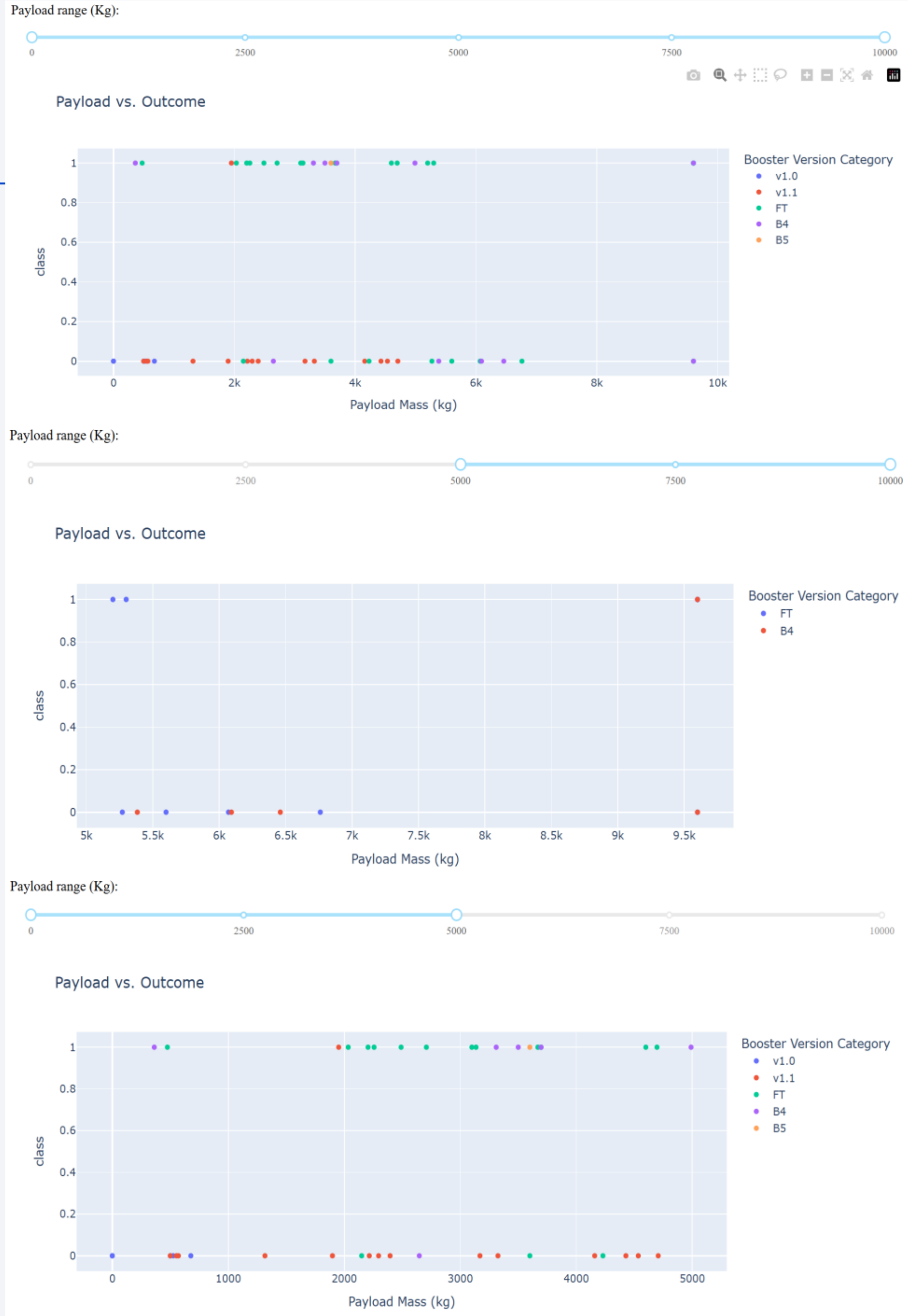
Pie Chart for the launch site with the highest launch rate

- As seen before, the launch site KSC LC-39A is the most successful launch site with a rate of 76.9%



Launch Outcome vs Payload

- The graphs demonstrate the variation of outcome compared to the payload
- It can be seen that lighter payloads are have a more successful outcome

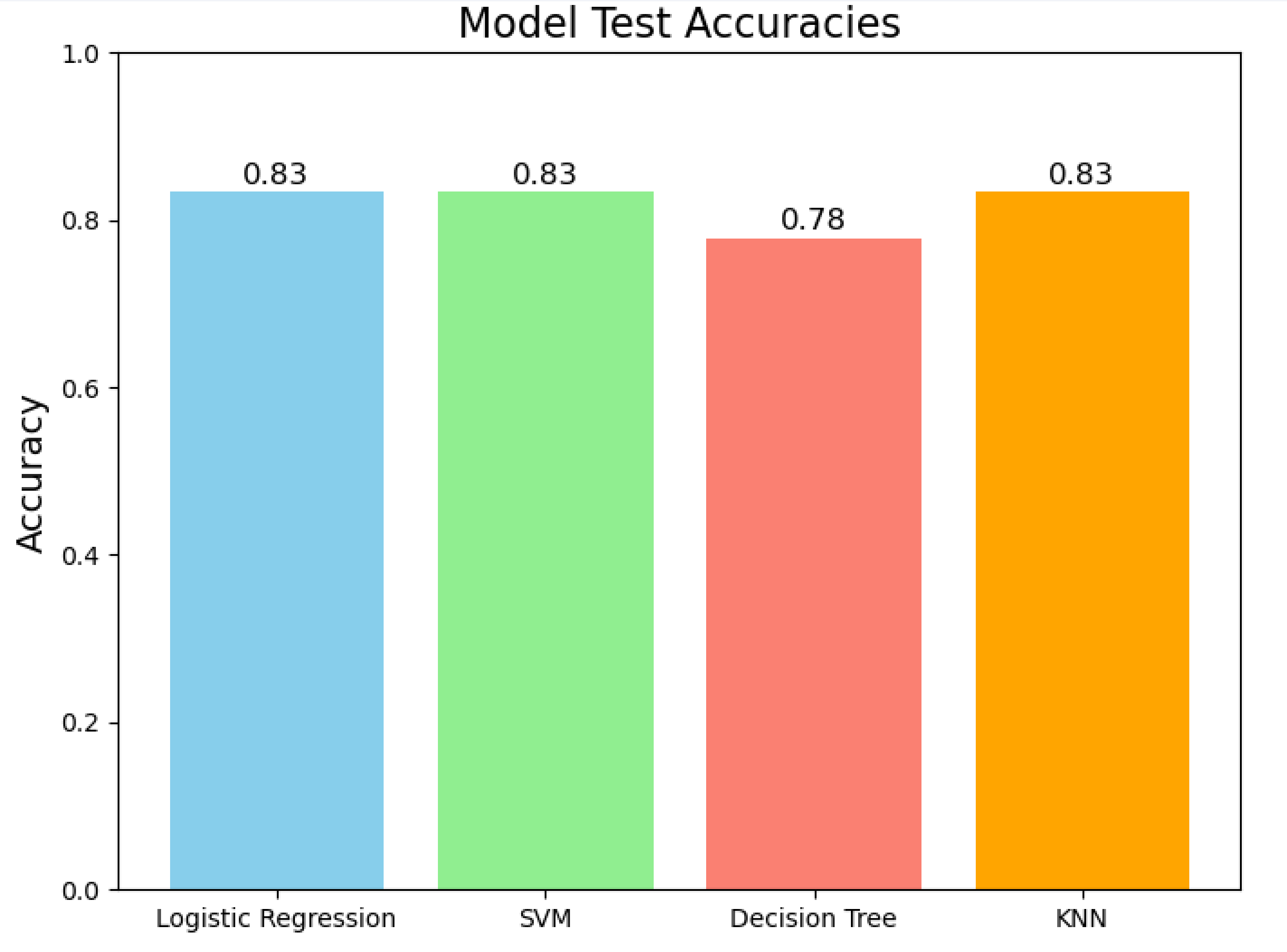


Section 5

Predictive Analysis (Classification)

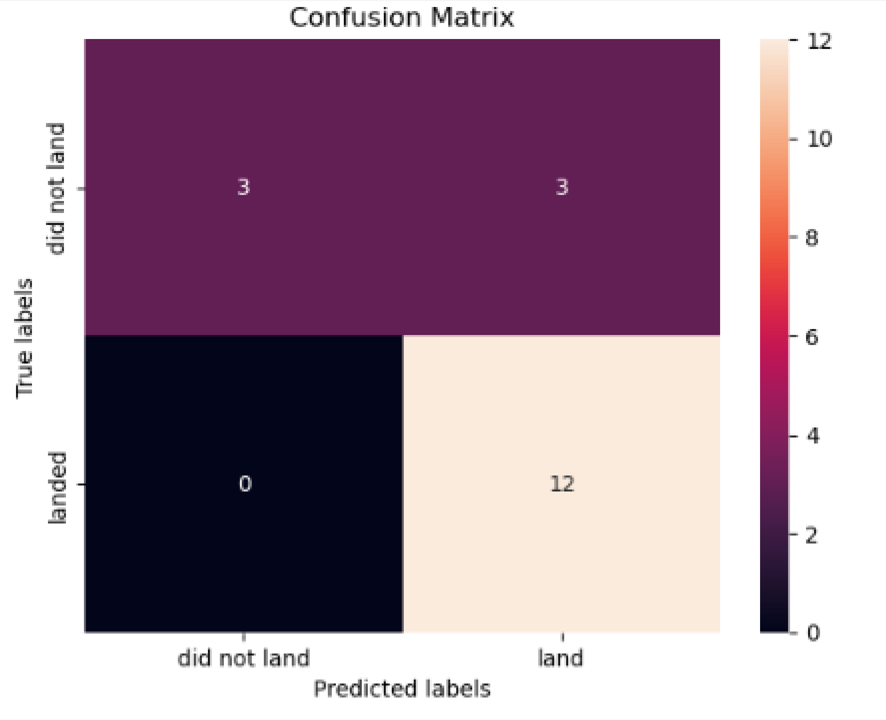
Classification Accuracy

- As it can be seen in the bar chart, the models present the same accuracy to predict the outcome of a launch with the exception of the Decision Tree



Confusion Matrix

- The confusion matrix for the three equally accurate models is the same across them, with only 3 false positives, and 0 false negatives



Conclusions

- The higher the number of flights at a launch site, the greater the launch success rate observed at that site.
- Launch success rates began improving around 2013 and continued to increase through 2020.
- Orbits such as ES-L1, GEO, HEO, SSO, and VLEO showed the highest success rates.
- KSC LC-39A recorded the most successful launches among all launch sites.
- Considering accuracy and F1-score, **Logistic Regression, SVM, and KNN classifiers all performed equally well** for predicting launch success.

Thank you!