IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Rajeev Sarma Sunnam
17-Jan-2024

Outline

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

Executive Summary

Summary of methodologies:

1. **Data Collection:**
    1. Utilize SpaceX API for real-time mission, launch, and landing data.
    2. Implement web scraping techniques to gather additional details from diverse sources.
2. **Data Wrangling:**
    1. Clean and format the collected data to handle missing values and ensure consistency.
3. **Exploratory Data Analysis (EDA) using SQL:**
    1. Leverage SQL queries to extract meaningful insights and patterns from the dataset.
4. **Data Visualization with Python (Pandas and Matplotlib):**
    1. Use Pandas for data manipulation and Matplotlib for creating informative visualizations.
5. **Launch Site Analysis with Folium and PlotlyDash:**
    1. Employ Folium for interactive maps and PlotlyDash for dynamic dashboards.
    2. Explore geographical patterns and trends related to SpaceX launch sites.
6. **Machine Learning Landing Prediction:**
    1. Build a machine learning model based on historical data to predict landing outcomes.
    2. Incorporate various features for a comprehensive predictive analysis.

Executive Summary continue…

- **Summary of all results**
- **Exploratory Data Analysis (EDA) Results:**
    1. Unearthed mission success trends and crucial statistical insights.
    2. Provided a deeper understanding of mission characteristics influencing success rates.
- **Interactive Visual Analytics and Dashboards:**
    1. Developed dynamic interfaces with Folium and PlotlyDash for engaging data exploration.
    2. Enhanced user interaction through informative and visually appealing interactive maps and dashboards.
- **Predictive Analysis (Classification):**
    1. Implemented a robust classification model predicting landing outcomes.
    2. Evaluated model effectiveness using key metrics, offering insights into future mission success.

Introduction

Project Background and Context:

•SpaceX, a prominent player in the space industry, offers Falcon 9 rocket launches at a significantly lower cost of 62 million dollars compared to other providers' prices, which can be as high as 165 million dollars.

•The cost advantage stems from SpaceX's ability to reuse the first stage of the Falcon 9 rocket, showcasing their commitment to innovative and cost-effective space exploration.

•The focus of this project lies in leveraging data from Falcon 9 rocket launches, as advertised on SpaceX's website, to predict the successful landing of the first stage. This prediction is crucial in determining the overall cost of a launch, as the ability to reuse the first stage significantly impacts expenses.

Problems to Address:

•The primary question revolves around predicting the successful landing of the Falcon 9 first stage.

•By answering this question, the project aims to provide insights into the cost-effectiveness of a launch, given the ability to reuse the first stage.

•The predictive analysis becomes a valuable tool for potential competitors or alternate companies looking to bid against SpaceX for rocket launches. Understanding the likelihood of a successful landing allows for informed decision-making in the bidding process.

Section 1

Methodology

Data Collection

The SpaceX launch data was collected using a combination of web scraping from the SpaceX website and utilizing the SpaceX API. The key steps in the data collection process are summarized below:

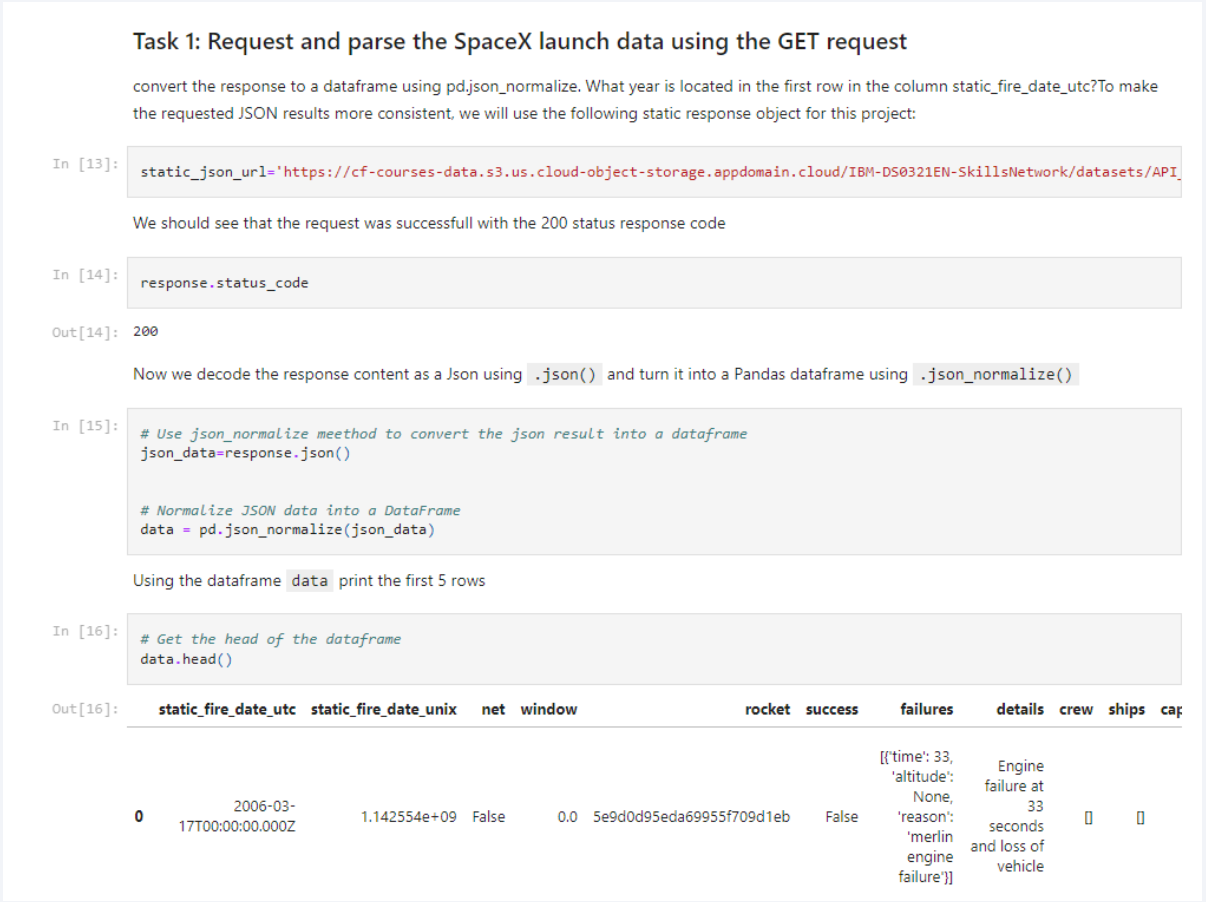
Web Scraping: A GET request was made to SpaceX's official website to retrieve information on Falcon 9 rocket launches. BeautifulSoup and requests Python libraries were employed for efficient web scraping. The data was extracted from relevant HTML tags and parsed.

API Integration: SpaceX API was utilized to obtain real-time and detailed data on rocket launches. API endpoints were accessed to retrieve information such as launch outcomes, success criteria, and reusable stage details. Authentication protocols were implemented to ensure secure access to SpaceX data.

Quality Control and Cleaning: Data quality checks were implemented to identify and address anomalies or missing values. Pandas and data cleaning techniques were applied to standardize and clean the dataset. Data validation ensured the accuracy and reliability of the collected information.

Data Collection – SpaceX API

- The SpaceX API facilitated real-time data retrieval on Falcon 9 rocket launches, offering detailed information via specific endpoints. With authentication for secure access, the API provided dynamic, structured data, contributing to a more accurate and up-to-date dataset compared to traditional web scraping.
- https://github.com/Rajeev-1809/Ds-certification/blob/main/jupyter-labs-spacex-data-collection-api.ipynb



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

convert the response to a dataframe using pd.json_normalize. What year is located in the first row in the column static_fire_date_utc?To make the requested JSON results more consistent, we will use the following static response object for this project:

In [13]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
```

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

In [14]:
```
response.status_code
```

Out[14]: 200

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

In [15]:
```
# Use json_normalize meethod to convert the json result into a dataframe
json_data=response.json()


# Normalize JSON data into a DataFrame
data = pd.json_normalize(json_data)
```

Using the dataframe `data` print the first 5 rows

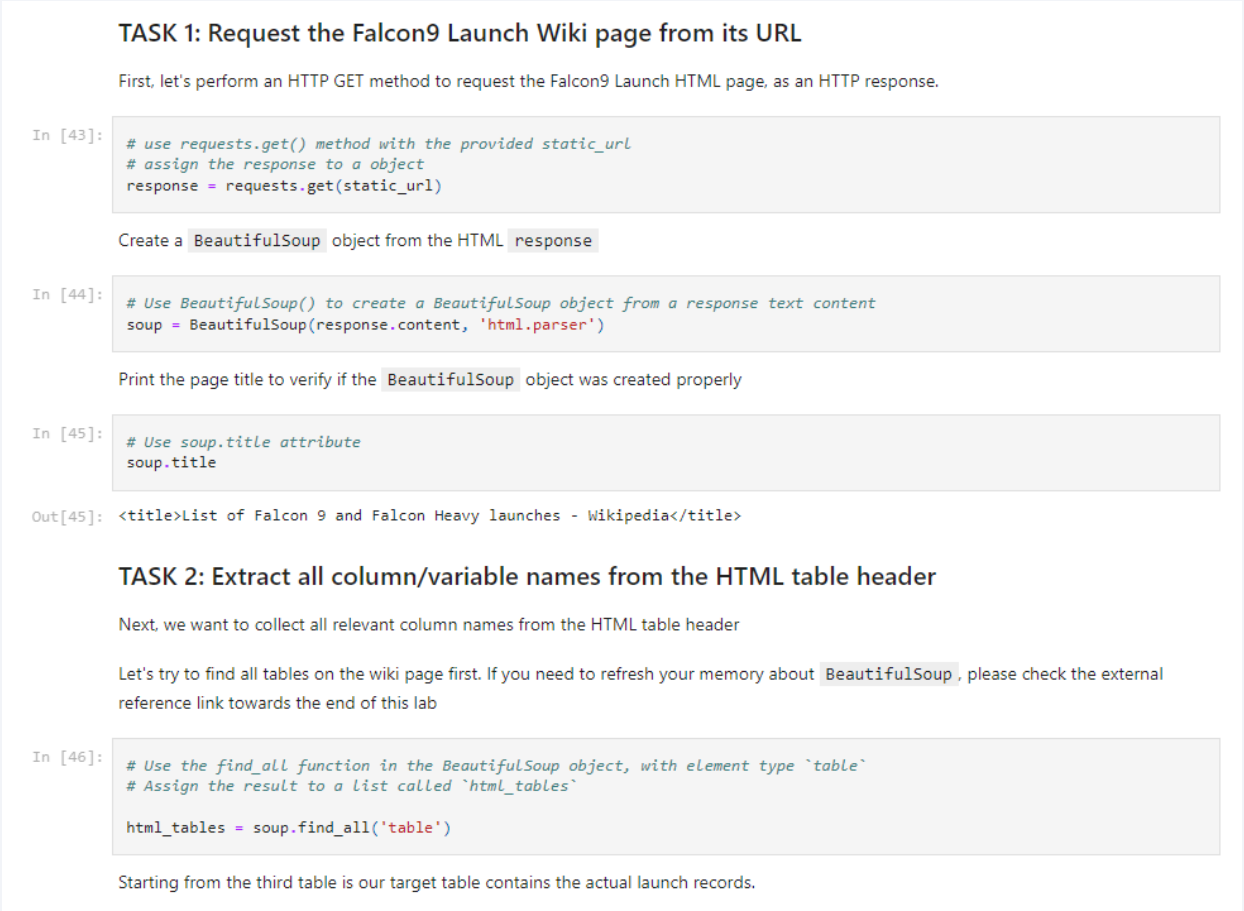
In [16]:
```
# Get the head of the dataframe
data.head()
```

Out[16]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	cap
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]	

Data Collection - Scraping

- Web scraping involved a GET request to SpaceX's website using Python's BeautifulSoup and requests libraries. The HTML content was parsed to extract relevant data on Falcon 9 rocket launches. This method enabled the collection of specific information directly from the website's source code.
- https://github.com/Rajeev-1809/Ds-certification/blob/main/jupyter-labs-webscraping.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 [43]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

In [44]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, 'html.parser')
```

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

In [45]:
```
# Use soup.title attribute
soup.title
```

Out[45]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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

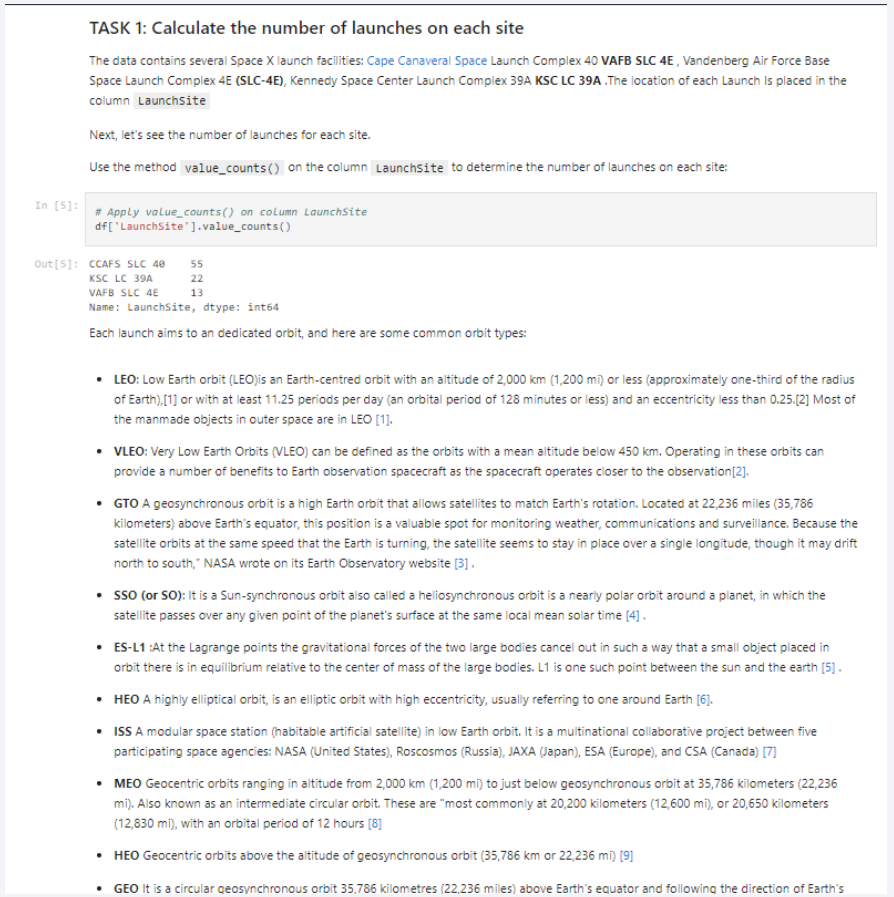
Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [46]:
```
# 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.

Data Wrangling

- Data wrangling involved addressing missing values in the SpaceX dataset. The mean payload mass was calculated and used to replace NaN values. This process ensured data completeness. The cleaned dataset was then exported to a CSV file for further analysis, ensuring consistency and reliability in subsequent tasks.
- https://github.com/Rajeev-1809/Ds-certification/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

In [5]:
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

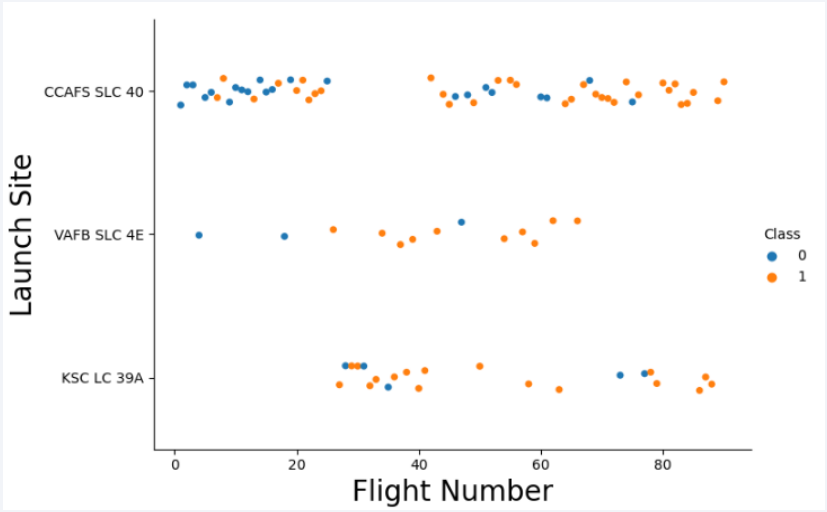
Out[5]:
```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: LaunchSite, dtype: int64
```

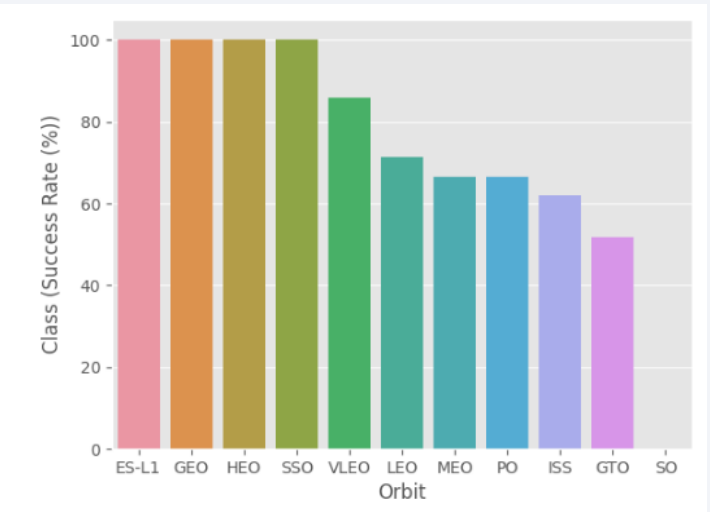
Each launch aims to an dedicated orbit, and here are some common orbit types:

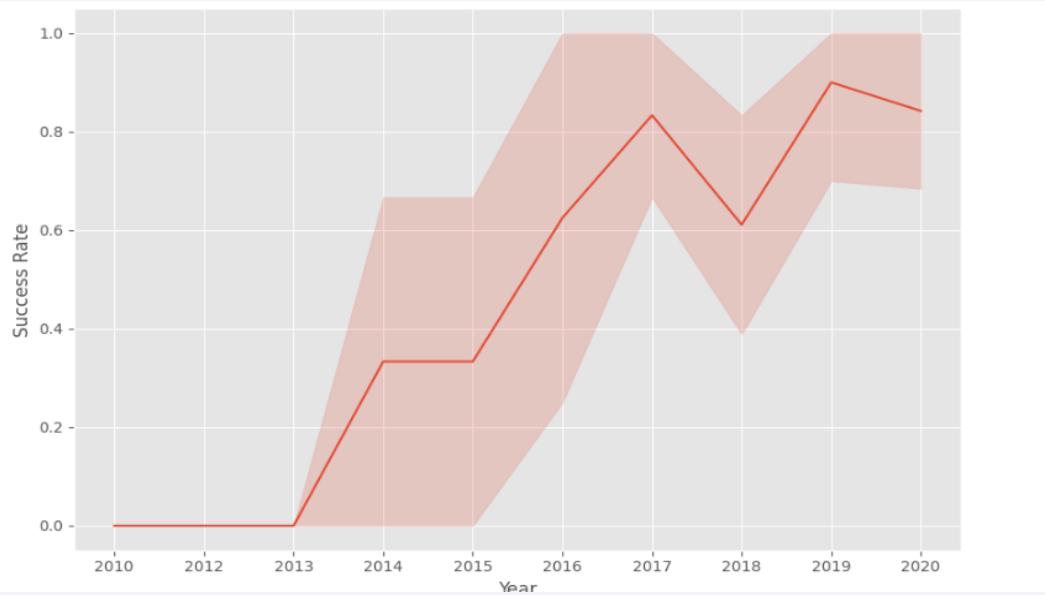
- **LEO:** Low Earth orbit (LEO)is an Earth-centred orbit with an altitude of 2,000 km (1,200 mi) or less (approximately one-third of the radius of Earth),[1] or with at least 11.25 periods per day (an orbital period of 128 minutes or less) and an eccentricity less than 0.25.[2] Most of the manmade objects in outer space are in LEO [1].
- **VLEO:** Very Low Earth Orbits (VLEO) can be defined as the orbits with a mean altitude below 450 km. Operating in these orbits can provide a number of benefits to Earth observation spacecraft as the spacecraft operates closer to the observation[2].
- **GTO** A geosynchronous orbit is a high Earth orbit that allows satellites to match Earth's rotation. Located at 22,236 miles (35,786 kilometers) above Earth's equator, this position is a valuable spot for monitoring weather, communications and surveillance. Because the satellite orbits at the same speed that the Earth is turning, the satellite seems to stay in place over a single longitude, though it may drift north to south," NASA wrote on its Earth Observatory website [3] .
- **SSO (or SO):** It is a Sun-synchronous orbit also called a heliosynchronous orbit is a nearly polar orbit around a planet, in which the satellite passes over any given point of the planet's surface at the same local mean solar time [4] .
- **ES-L1** :At the Lagrange points the gravitational forces of the two large bodies cancel out in such a way that a small object placed in orbit there is in equilibrium relative to the center of mass of the large bodies. L1 is one such point between the sun and the earth [5] .
- **HEO** A highly elliptical orbit, is an elliptic orbit with high eccentricity, usually referring to one around Earth [6].
- **ISS** A modular space station (habitable artificial satellite) in low Earth orbit. It is a multinational collaborative project between five participating space agencies: NASA (United States), Roscosmos (Russia), JAXA (Japan), ESA (Europe), and CSA (Canada) [7]
- **MEO** Geocentric orbits ranging in altitude from 2,000 km (1,200 mi) to just below geosynchronous orbit at 35,786 kilometers (22,236 mi). Also known as an intermediate circular orbit. These are "most commonly at 20,200 kilometers (12,600 mi), or 20,650 kilometers (12,830 mi), with an orbital period of 12 hours [8]
- **HEO** Geocentric orbits above the altitude of geosynchronous orbit (35,786 km or 22,236 mi) [9]
- **GEO** It is a circular geosynchronous orbit 35,786 kilometres (22,236 miles) above Earth's equator and following the direction of Earth's

EDA with Data Visualization

- Conducted exploratory data analysis and feature engineering using Pandas and Matplotlib. This involved analyzing relationships between variables such as Flight Number and Launch Site, Payload and Launch Site, Flight Number and Orbit type. Utilized scatter plots, bar charts, and line plots to visualize patterns, success rates, and launch trends over the years. Prepared and engineered data features for enhanced analysis.
- https://github.com/Rajeev-1809/Ds-certification/blob/main/5.%20Space-X%20EDA%20DataViz%20Using%20Pandas%20and%20Matplotlib%20-%20SpaceX.ipynb

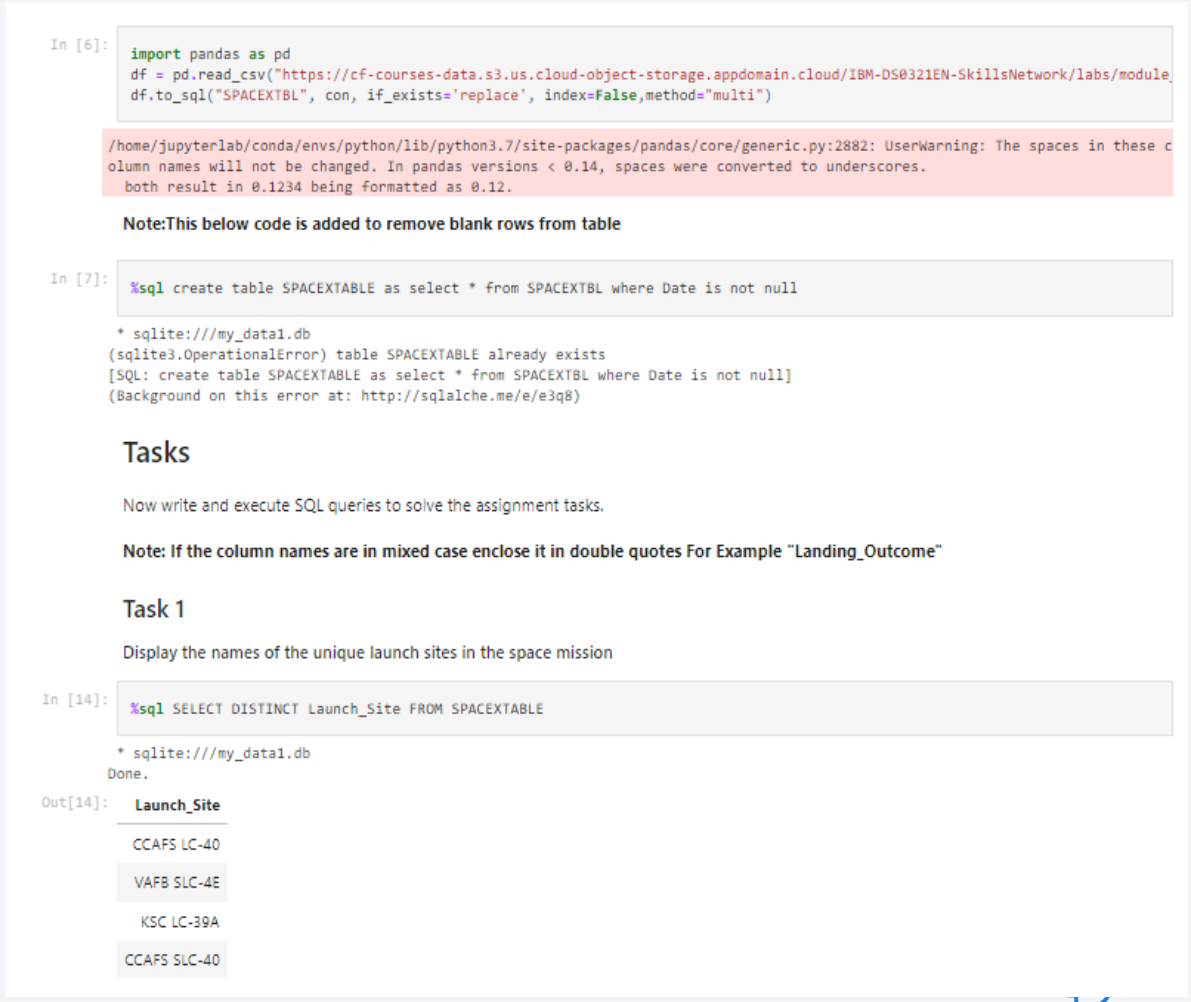






EDA with SQL

- Displayed unique launch site names and etrieved 5 records where launch sites start with 'CCA'.
- Calculated total payload mass carried by NASA (CRS) boosters.
- Determined average payload mass of booster version F9 v1.1.
- Listed date of the first successful ground pad landing.
- Listed names of boosters with drone ship success and payload mass between 4000 and 6000.
- Summarized total number of successful and failure mission outcomes.
- Identified booster versions carrying maximum payload mass using a subquery.
- Displayed records with month names, failure drone ship landings, booster versions, and launch sites for 2015.
- Ranked count of landing outcomes between June 4, 2010, and March 20, 2017, in descending order.
- https://github.com/Rajeev-1809/Ds-certification/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb



In [6]:
```
import pandas as pd
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/labs/module_
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False,method="multi")
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/generic.py:2882: UserWarning: The spaces in these c
olumn names will not be changed. In pandas versions < 0.14, spaces were converted to underscores.
  both result in 0.1234 being formatted as 0.12.
```

Note:This below code is added to remove blank rows from table

In [7]:
```
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) table SPACEXTABLE already exists
[SQL: create table SPACEXTABLE as select * from SPACEXTBL where Date is not null]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Tasks

Now write and execute SQL queries to solve the assignment tasks.

Note: If the column names are in mixed case enclose it in double quotes For Example "Landing_Outcome"

Task 1

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

In [14]:
```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE
```

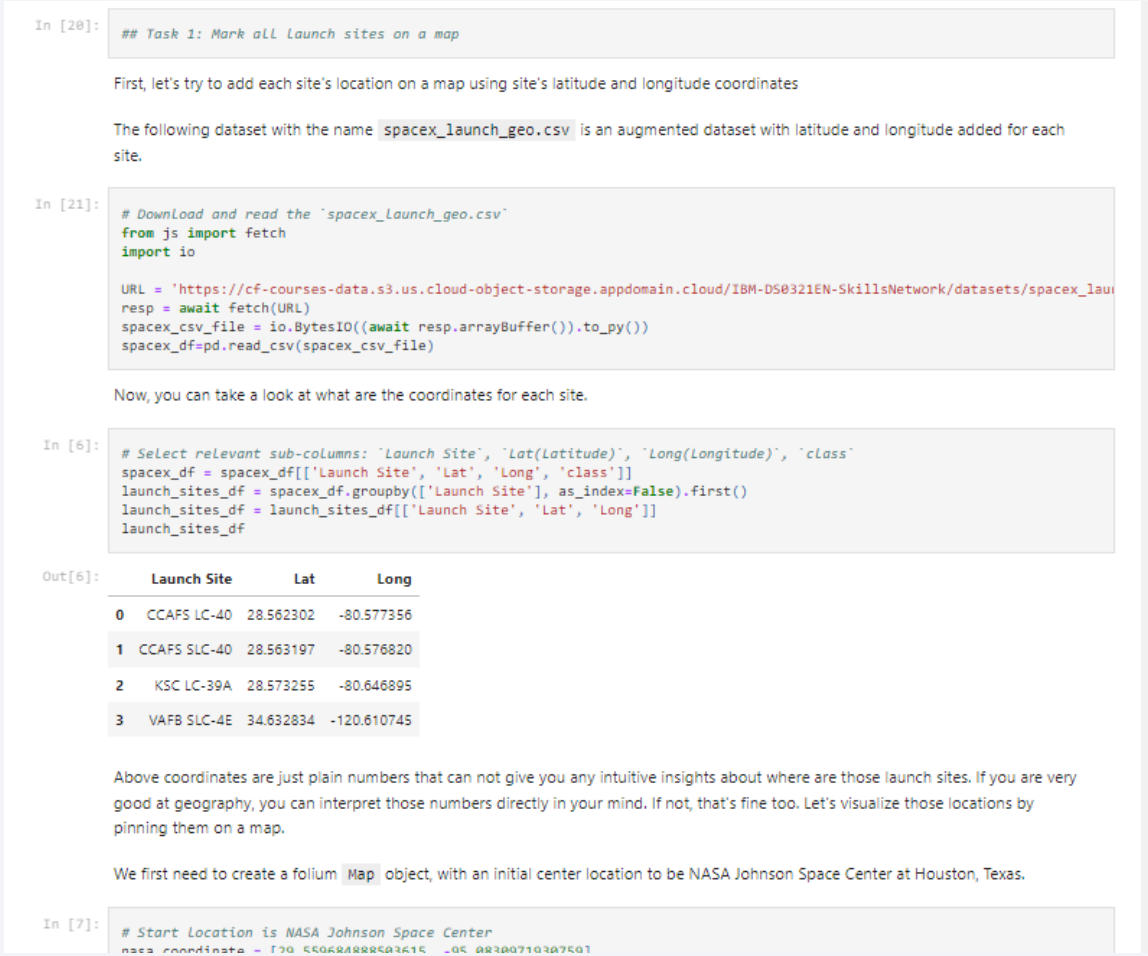
```
 * sqlite:///my_data1.db
Done.
```

Out[14]:

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

Build an Interactive Map with Folium

- Markers were used to represent launch sites, colored by success or failure. Circles highlighted launch areas and displayed distances. Lines connected sites to coastline and proximity points. MousePosition showed coordinates. MarkerClusters grouped markers at the same location. DivIcon customized labels with distance information, ensuring an informative and interactive map.
- Markers highlighted launch sites and outcomes, providing visual clarity. Circles and lines conveyed geographical relationships, aiding proximity analysis. MarkerClusters reduced map clutter. DivIcon labels enhanced information display for better user understanding.
- https://github.com/Rajeev-1809/Ds-certification/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb



In [20]:
```
## Task 1: Mark all launch sites on a map
```

First, let's try to add each site's location on a map using site's latitude and longitude coordinates

The following dataset with the name `spacex_launch_geo.csv` is an augmented dataset with latitude and longitude added for each site.

In [21]:
```
# Download and read the `spacex_launch_geo.csv`
from js import fetch
import io

URL = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/spacex_lau
resp = await fetch(URL)
spacex_csv_file = io.BytesIO((await resp.arrayBuffer()).to_py())
spacex_df=pd.read_csv(spacex_csv_file)
```

Now, you can take a look at what are the coordinates for each site.

In [6]:
```
# Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
spacex_df = spacex_df[['Launch Site', 'Lat', 'Long', 'class']]
launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]
launch_sites_df
```

Out[6]:

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

Above coordinates are just plain numbers that can not give you any intuitive insights about where are those launch sites. If you are very good at geography, you can interpret those numbers directly in your mind. If not, that's fine too. Let's visualize those locations by pinning them on a map.

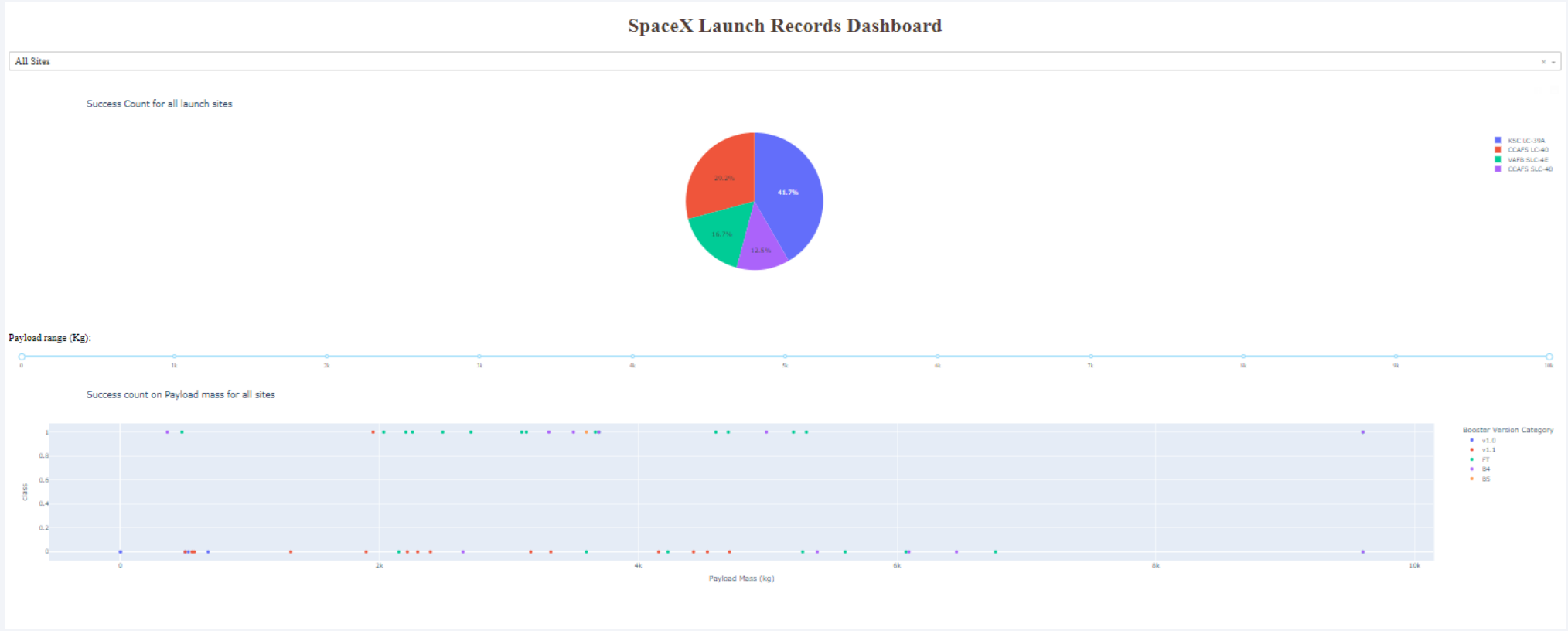
We first need to create a folium `Map` object, with an initial center location to be NASA Johnson Space Center at Houston, Texas.

In [7]:
```
# Start Location is NASA Johnson Space Center
```

Build a Dashboard with Plotly Dash

- The SpaceX Launch Records Dashboard includes a dropdown for selecting launch sites, displaying a pie chart showing success counts for all sites or a selected site. Additionally, a payload range slider is incorporated, updating a scatter chart depicting the correlation between payload mass and launch success for all sites or a chosen site. The dropdown and pie chart provide an overview of launch success at different sites. The payload range slider and scatter chart allow users to explore the correlation between payload mass and launch success.

- https://github.com/Rajeev-1809/Ds-certification/blob/main/spacex_dash_app.py



Predictive Analysis (Classification)

- In this classification model development, I loaded SpaceX launch data, performed exploratory data analysis, and created a machine learning pipeline. I standardized the data, split it into training and test sets, and utilized GridSearchCV to optimize hyperparameters for Logistic Regression, SVM, Decision Tree, and KNN models. The models were evaluated and compared on test data, revealing similar accuracy of 83.33%.
- https://github.com/Rajeev-1809/Ds-certification/blob/main/8.%20SpaceX%20Machine%20Learning%20Prediction.ipynb

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size to 0.2 and random_state to 2. The training data and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

In [21]:
```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

In [27]:
```
print ('Train set:')
print('X-train= ', X_train.shape,'Y-train= ',Y_train.shape)
print ('Test set:')
print('X-test= ', X_test.shape,  'Y-test= ',Y_test.shape)
```

```
Train set:
X-train=  (72, 83) Y-train=  (72,)
Test set:
X-test=  (18, 83) Y-test=  (18,)
```

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to find the best parameters from the dictionary `parameters`.

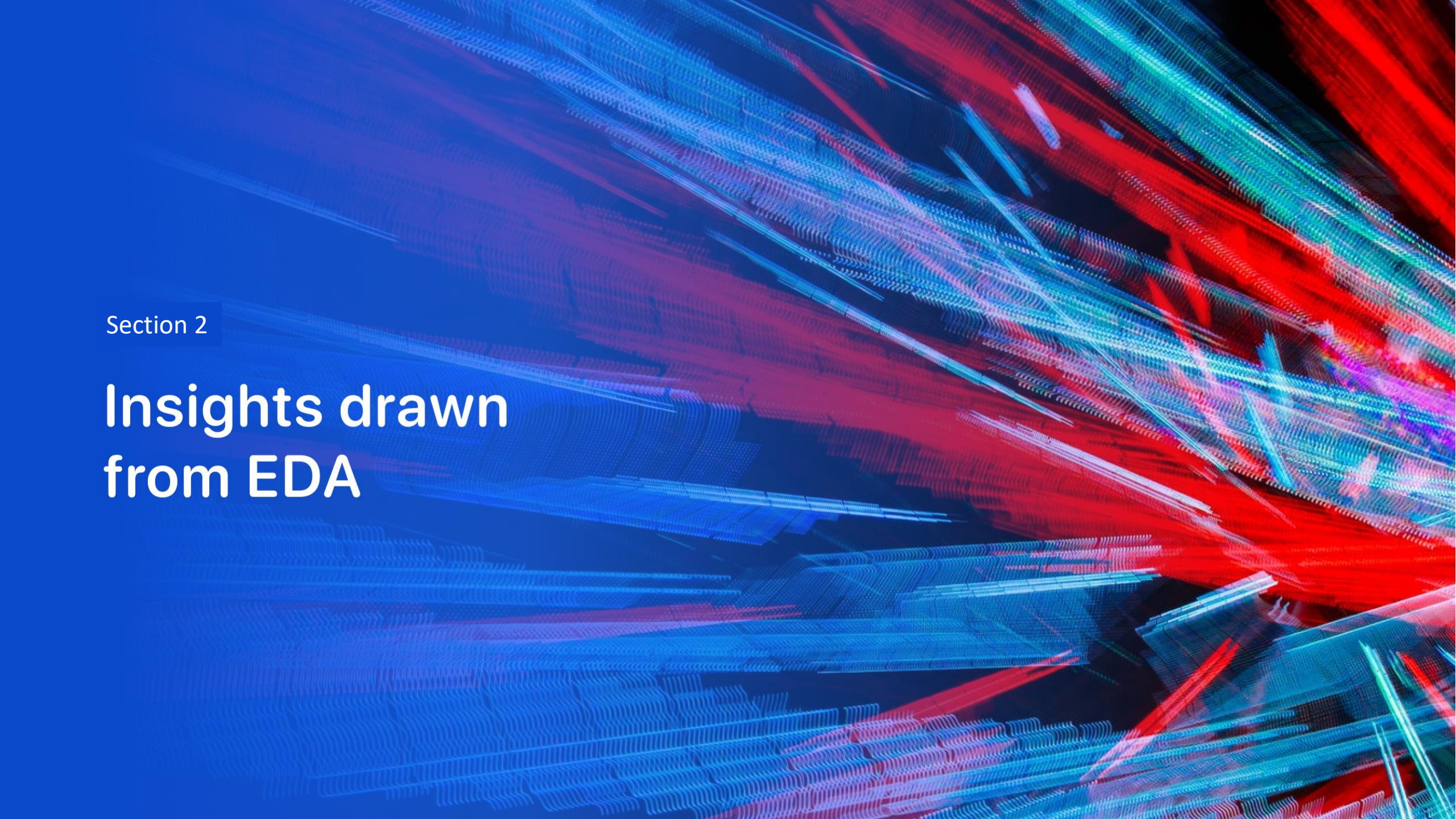
In [ ]:
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

In [28]:
```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
# Create a logistic regression object
lr=LogisticRegression()

# Create a GridSearchCV object logreg_cv
logreg_cv = GridSearchCV(lr, parameters, cv=10)

#Fit the training data into the GridSearch object
logreg_cv.fit(X_train, Y_train)
```

Out[28]:
```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

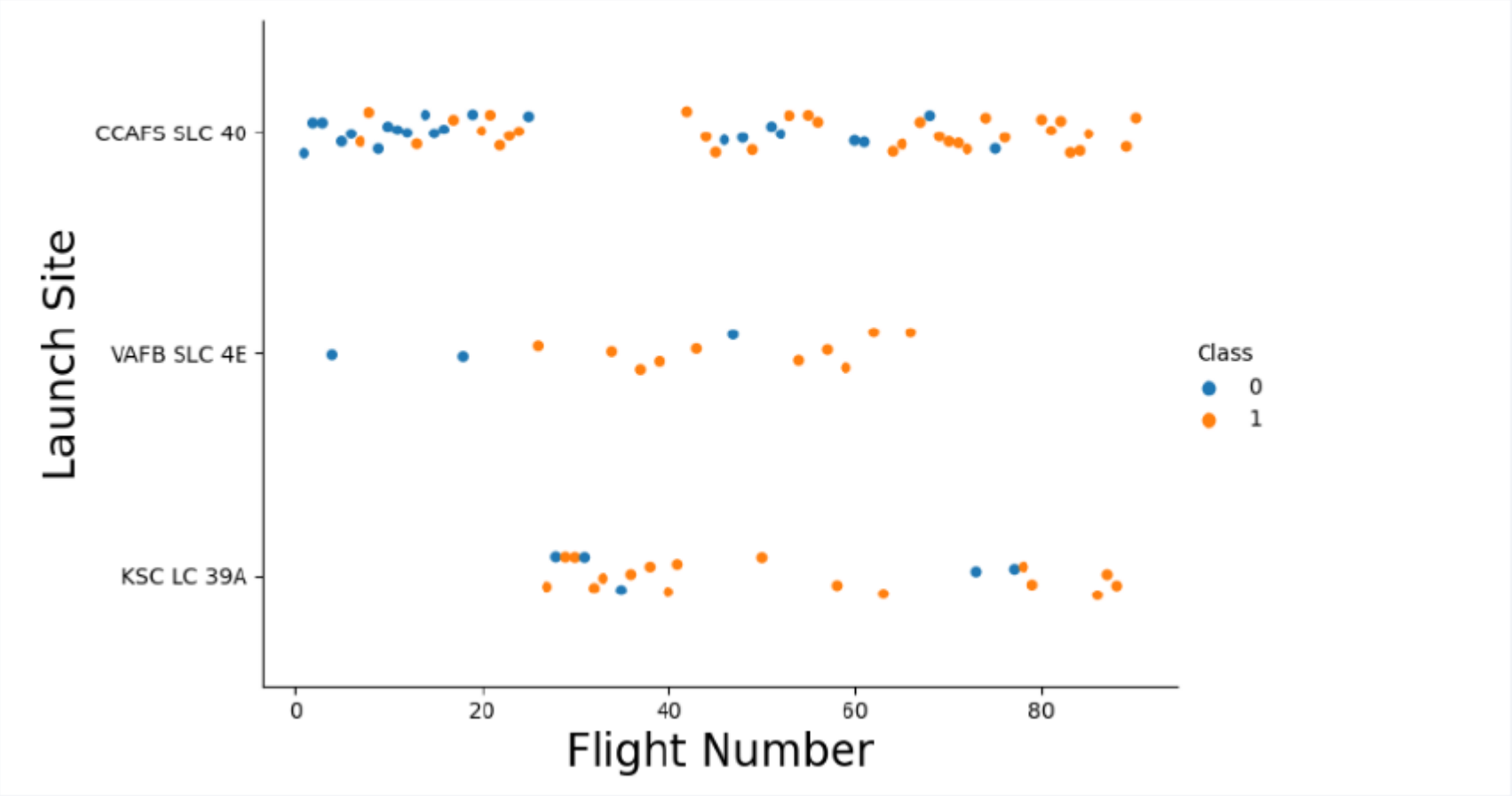
Results

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

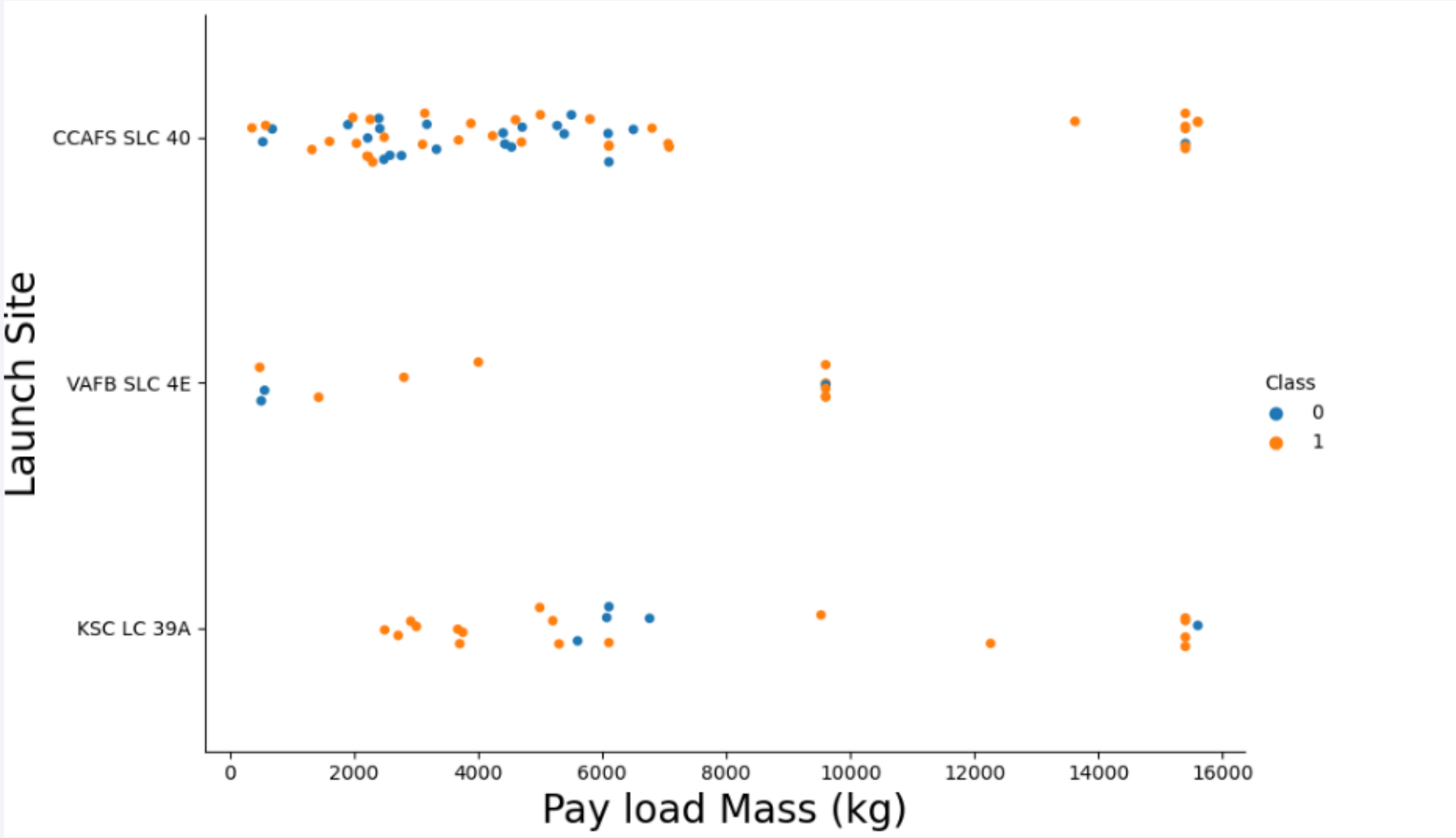

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

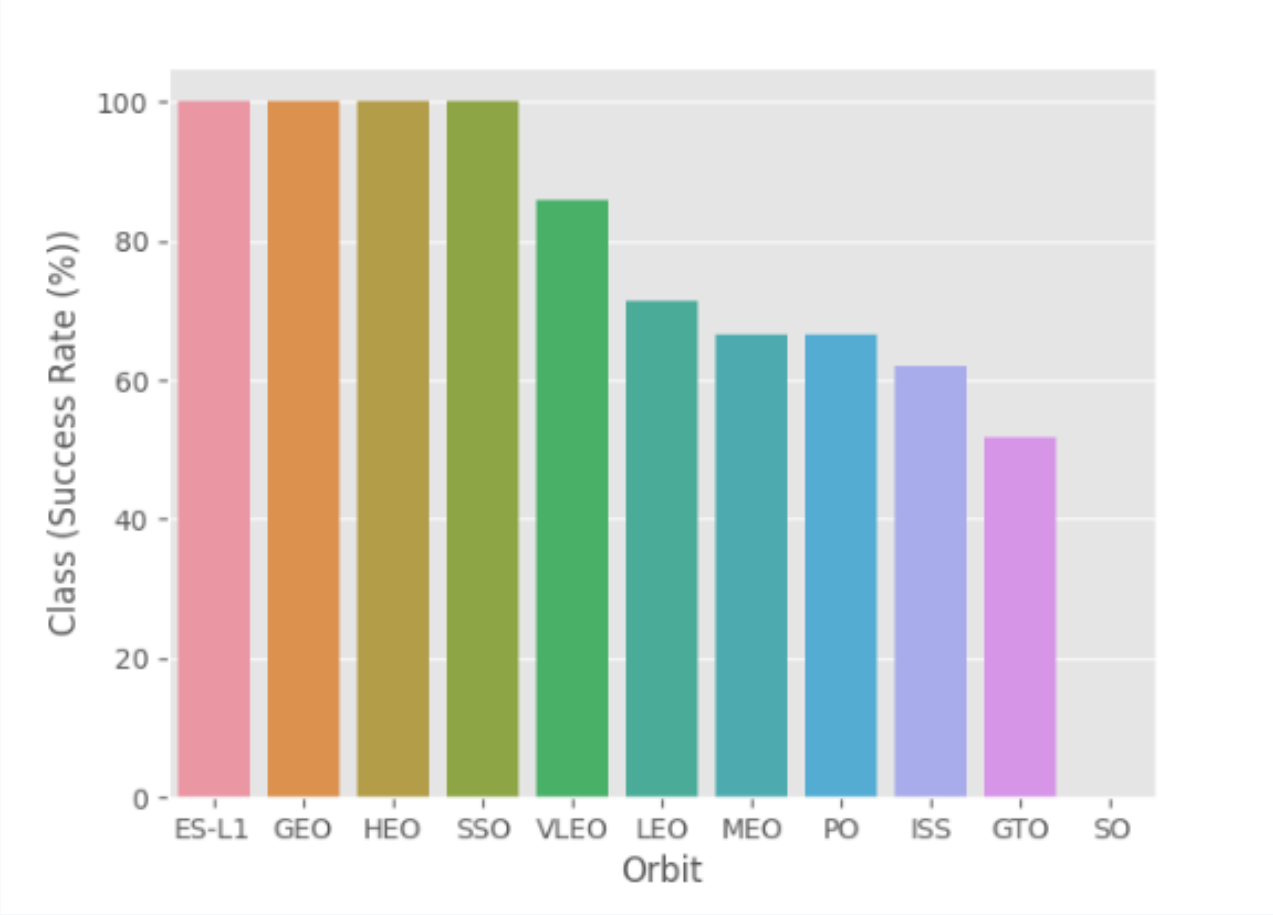
A scatter plot of Flight Number vs. Launch Site



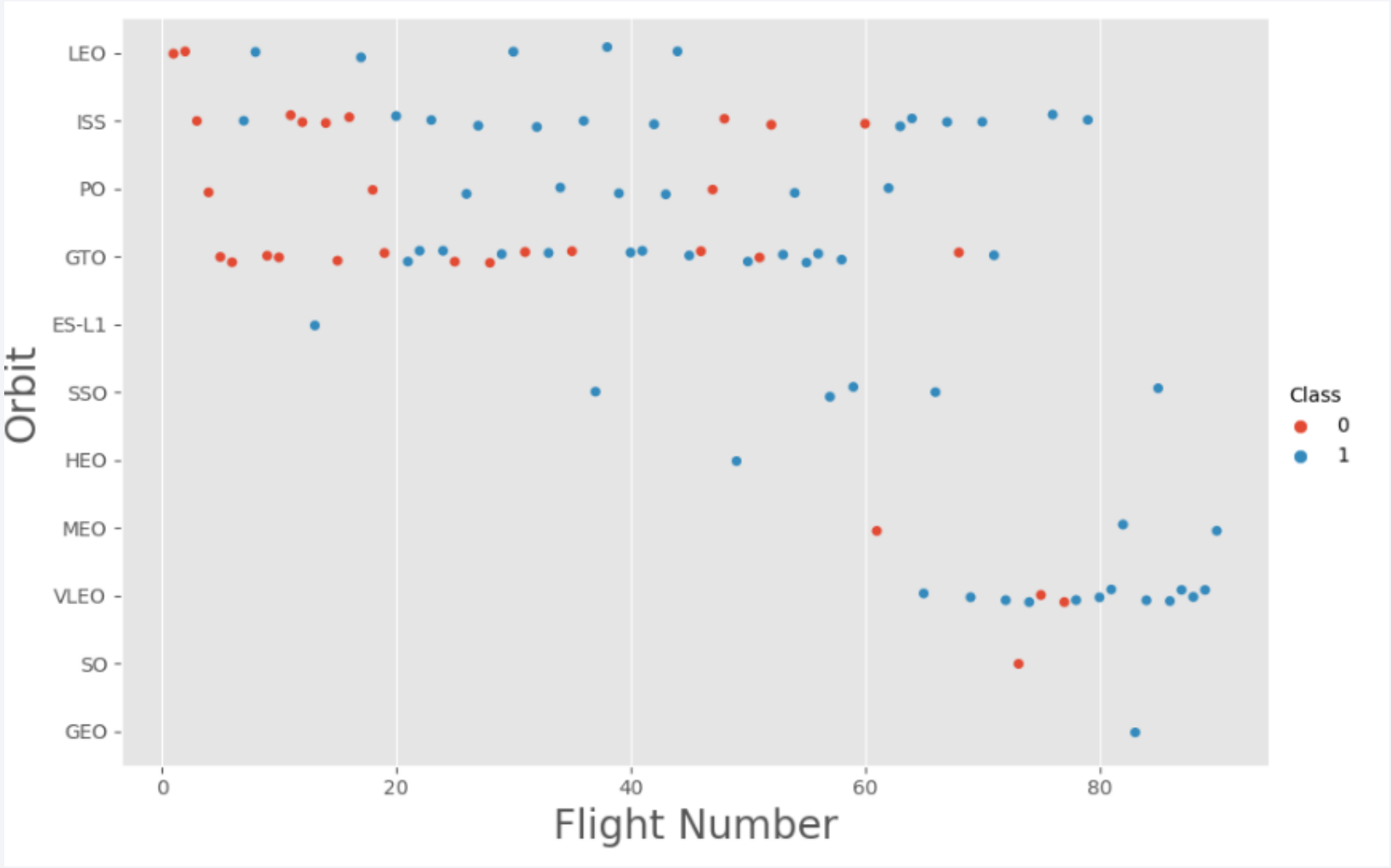
Payload vs. Launch Site



Success Rate vs. Orbit Type

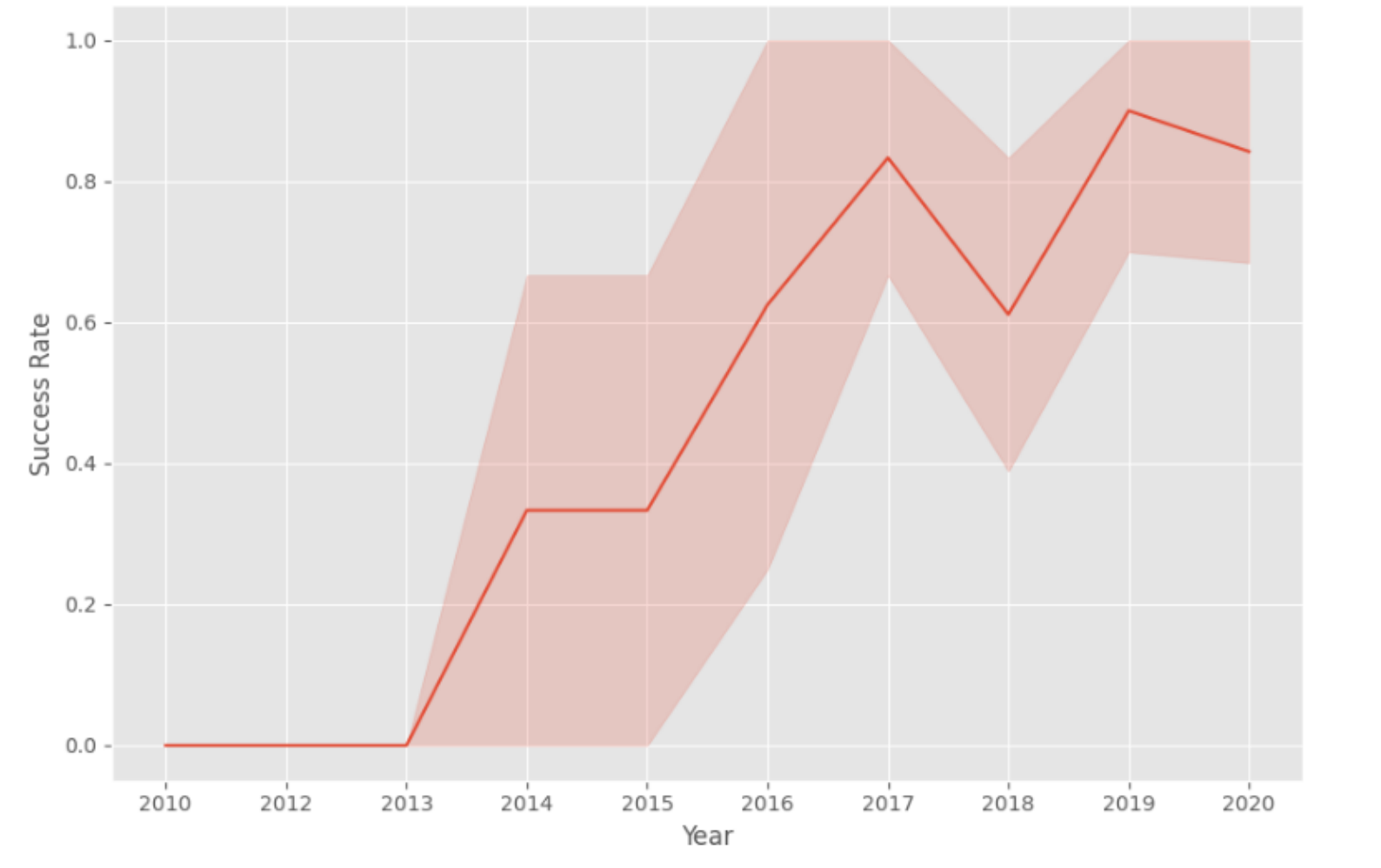


Flight Number vs. Orbit Type



Payload vs. Orbit Type

Launch Success Yearly Trend



All Launch Site Names

Task 1

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

In [31]:
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

Out[31]:

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

Launch Site Names Begin with 'CCA'

Task 2

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

In [72]:
```
%sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[72]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Task 3

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

In [17]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

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

Out[17]:

Total Payload Mass(Kgs)	Customer
45596	NASA (CRS)

Average Payload Mass by F9 v1.1

Task 4

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
%sql SELECT MIN(DATE) FROM 'SPACEXTBL' WHERE "Landing _Outcome" = "Success (ground pad)";
```

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

MIN(DATE)
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
# %sql SELECT * FROM 'SPACEXTBL'
```

```
%sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOA
```

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

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") as Total FROM SPACEXTBL GROUP BY "Mission_Outcome";
```

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

Mission_Outcome	Total
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

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

Booster_Version	Payload	PAYLOAD_MASS__KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

2015 Launch Records

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.

```
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome", "Landing _Outcome"
```

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

substr(Date,7,4)	substr(Date, 4, 2)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Mission_Outcome	Landing _Outcome
2015	01	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	Success	Failure (drone ship)
2015	04	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	Success	Failure (drone ship)

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

Task 10

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

```
%sql SELECT * FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORDER BY Date DESC;
```

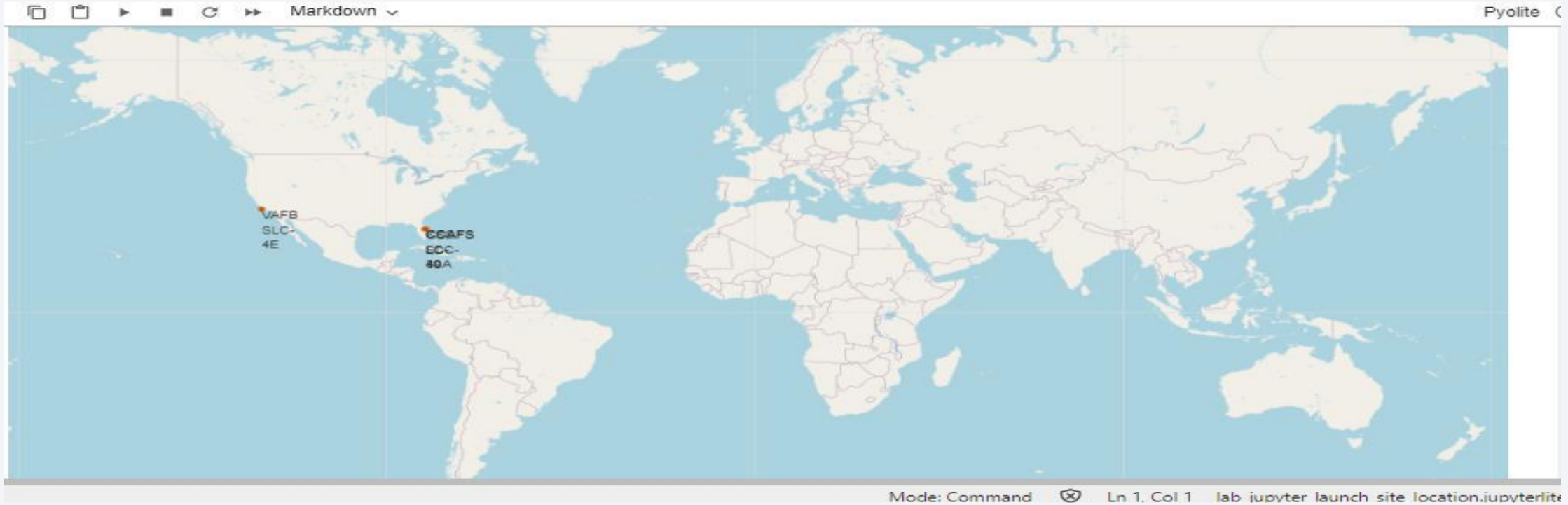
```
 * sqlite:///my_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-10-2020	12:25:57	F9 B5 B1051.6	KSC LC-39A	Starlink 13 v1.0, Starlink 14 v1.0	15600	LEO	SpaceX	Success	Success
18-08-2020	14:31:00	F9 B5 B1049.6	CCAFS SLC-40	Starlink 10 v1.0, SkySat-19, -20, -21, SAOCOM 1B	15440	LEO	SpaceX, Planet Labs, PlanetIQ	Success	Success
18-07-2016	04:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
18-04-2018	22:51:00	F9 B4 B1045.1	CCAFS SLC-40	Transiting Exoplanet Survey Satellite (TESS)	362	HEO	NASA (LSP)	Success	Success (drone ship)

Section 3

Launch Sites Proximities Analysis

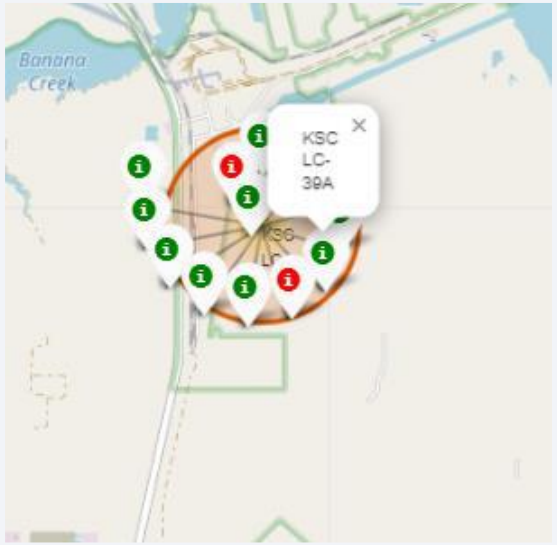
<Folium Map Screenshot 1>



Each launch site is strategically situated near the Equator, situated southward on the US map. Furthermore, all launch sites are positioned in close proximity to coastlines. This geographical arrangement near the Equator and coastal regions contributes to optimal launch conditions and facilitates efficient access to different orbital trajectories.

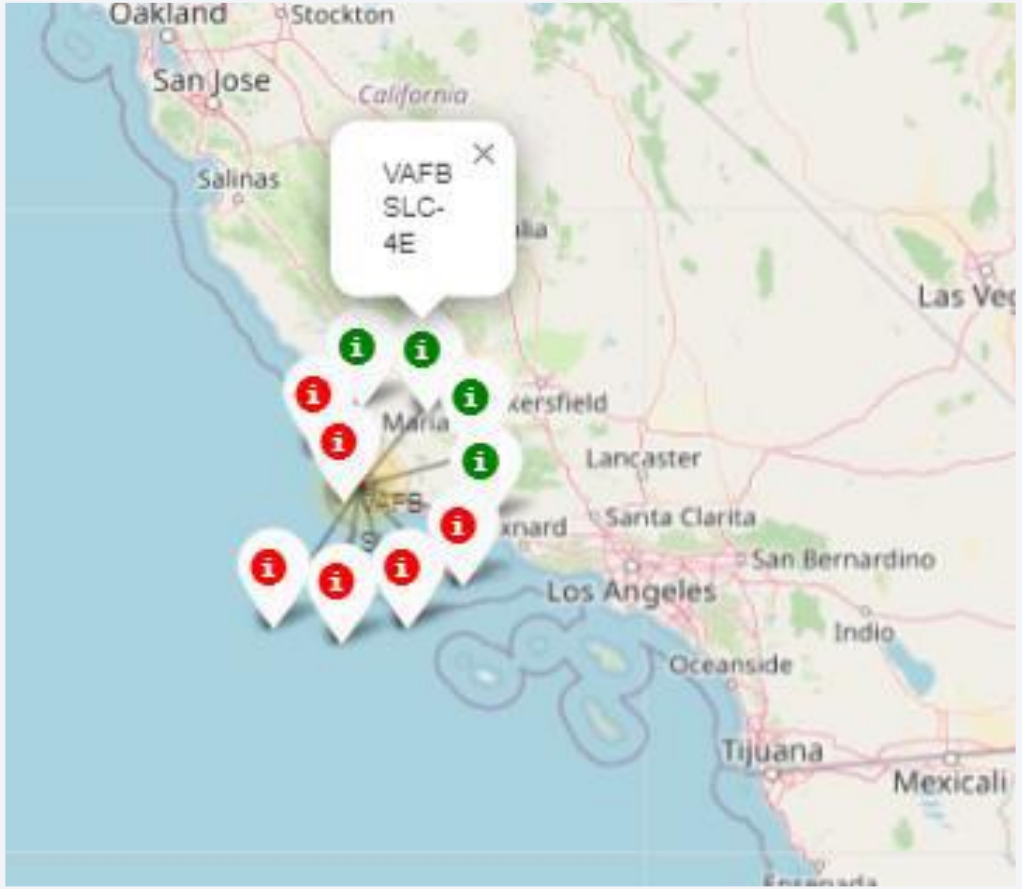
<Folium Map Screenshot 2>

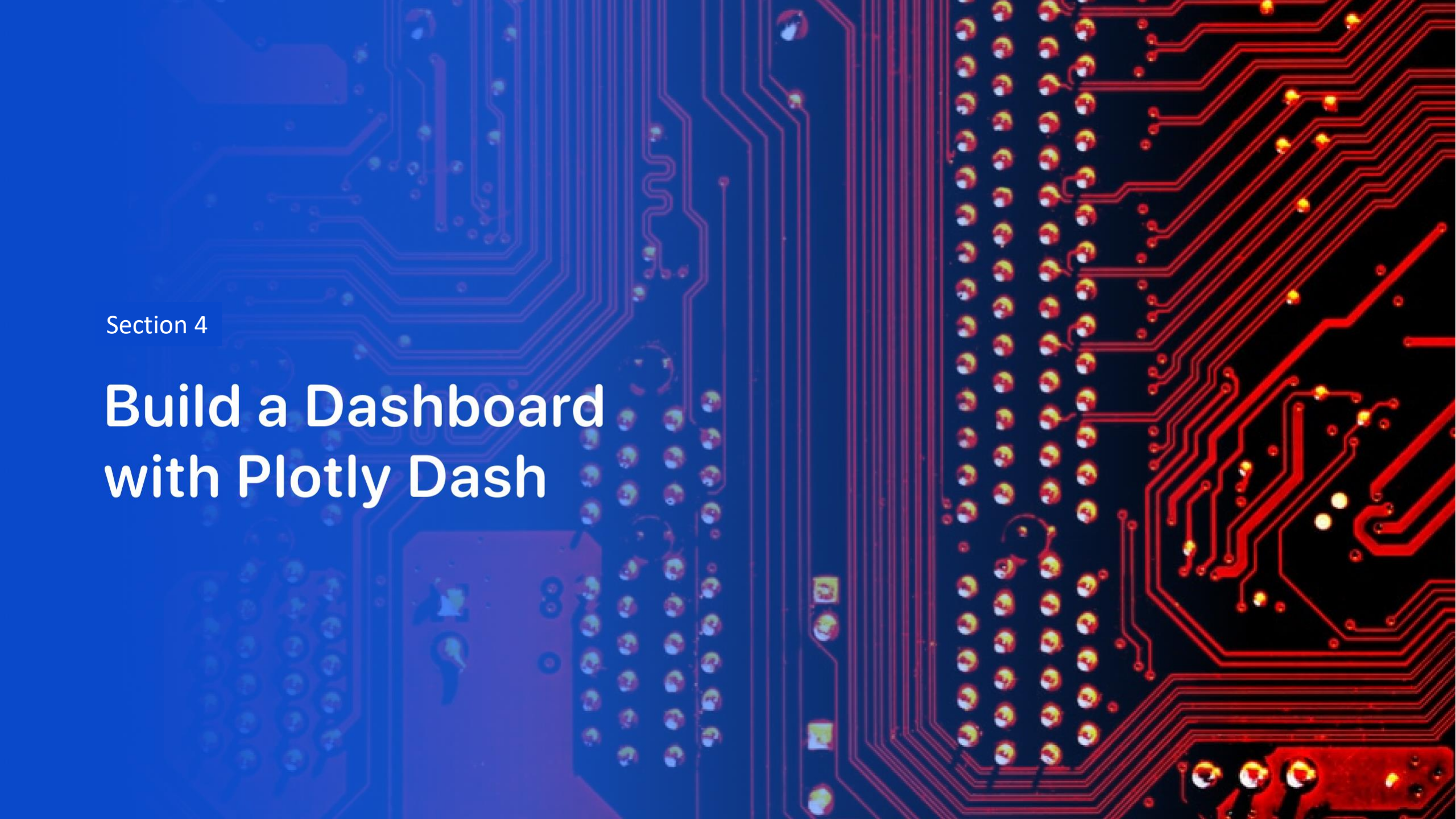
On the Eastern coast in Florida, Launch site KSC LC-39A exhibits notably higher success rates in comparison to its counterparts, CCAFS SLC-40 and CCAFS LC-40. This suggests that KSC LC-39A demonstrates a more favorable track record in achieving successful launches within the region.



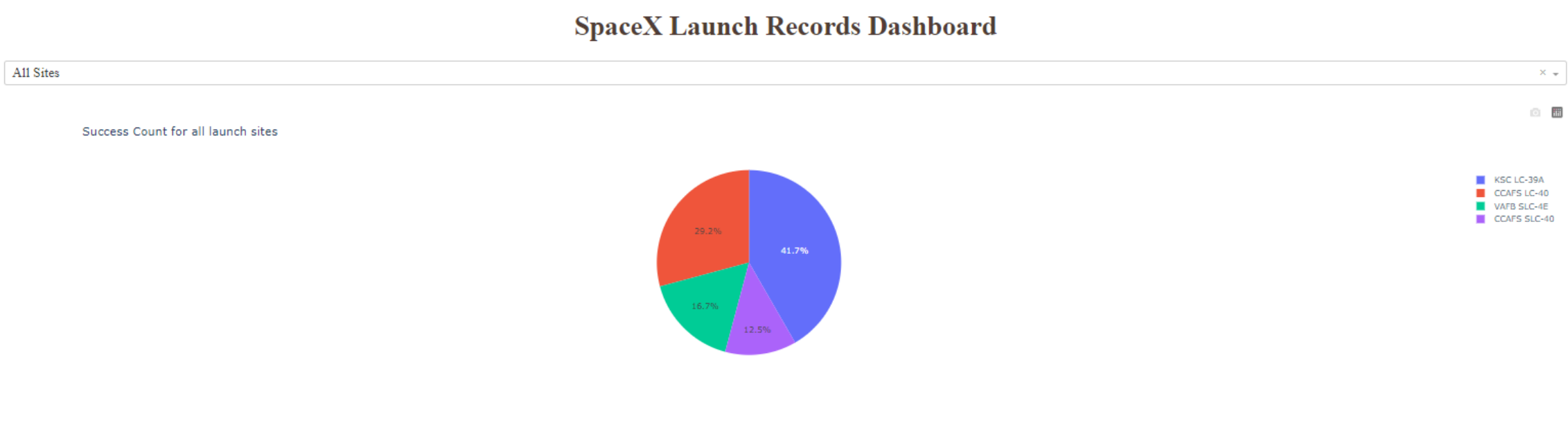
<Folium Map Screenshot 3>

On the West Coast at VAFB SLC-4E in California, the success rate is comparatively lower at 4 out of 10 launches. In contrast, the Launch site KSC LC-39A on the Eastern Coast of Florida exhibits a higher success rate. This discrepancy emphasizes regional variations in launch success within the United States.



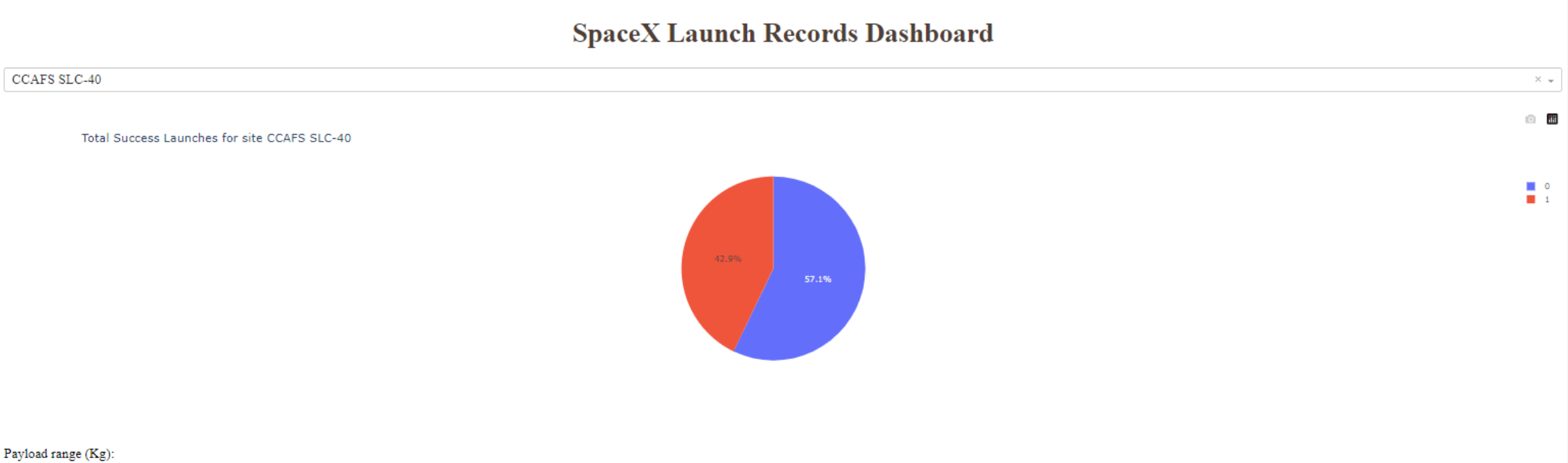

Section 4
Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>



KSC LC-39A boasts the highest launch success rate at an impressive 42%, surpassing CCAFS LC-40, which follows closely with a commendable 29% success rate. This distinction highlights KSC LC-39A as the leading launch site, reflecting a substantial track record of successful missions compared to CCAFS LC-40.

<Dashboard Screenshot 2>



CCAFS SLC-40 boasts the highest success rate among launch sites, standing at an impressive 42.9%. This stellar performance positions it as a standout choice, demonstrating remarkable reliability and success in space missions compared to other launch sites.

<Dashboard Screenshot 3>

Within Launch site CCAFS LC-40, booster version FT stands out with the highest success rate, particularly evident for payload masses exceeding 2000 kg. This observation underscores the significance of booster version FT in achieving successful launches, especially when handling payloads of substantial mass.

Section 5

Predictive Analysis (Classification)

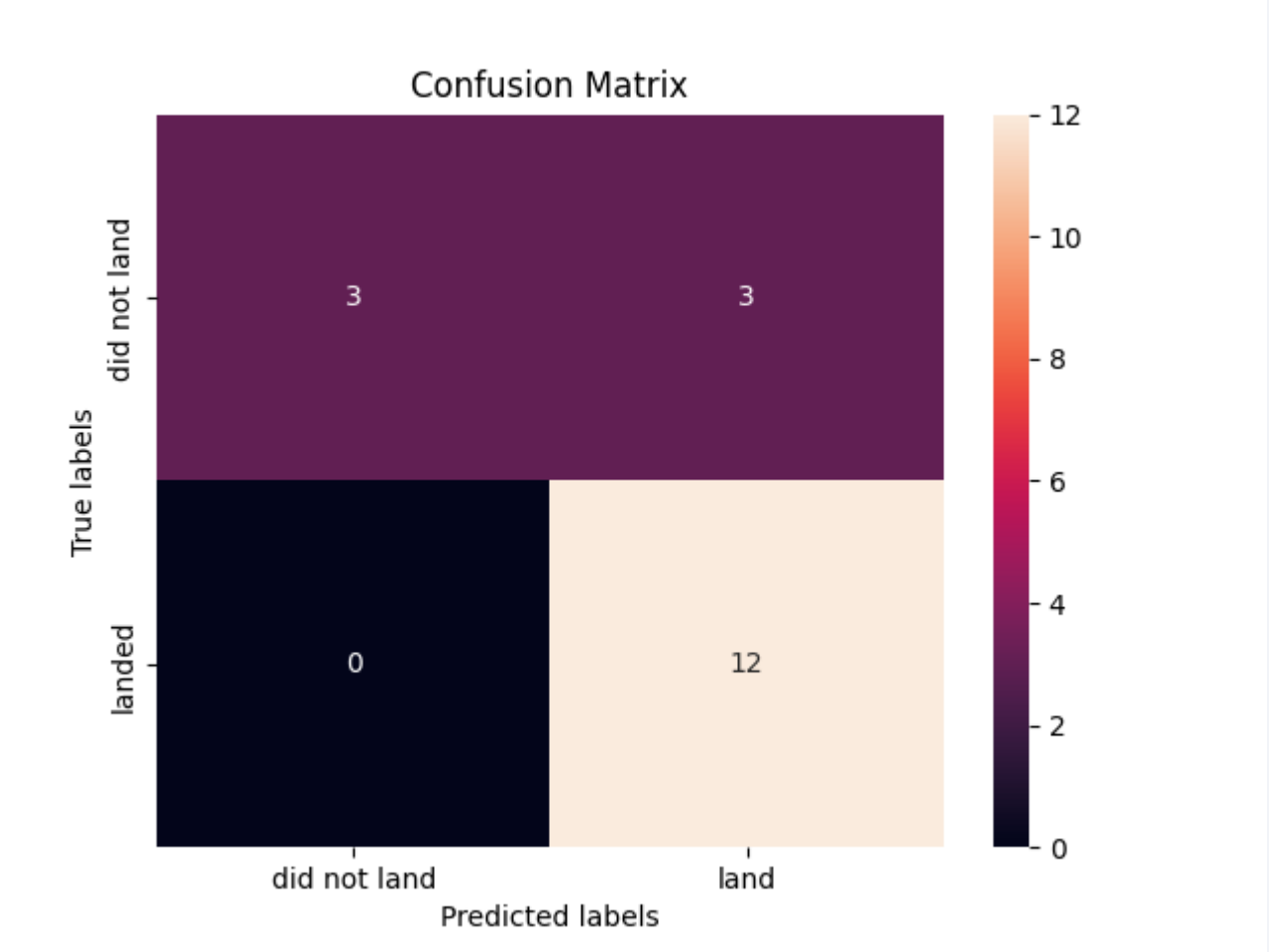
Classification Accuracy

All the models performed similarly.

	0
Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

Confusion Matrix

All four classification models exhibited identical confusion matrices, effectively distinguishing between classes. However, a common issue across all models was false positives. Despite their equal performance, false positives emerged as a shared challenge, signaling the need for further refinement in minimizing misclassifications.



Conclusions

- CCAFS LC-40 has a success rate of 60%, while KSC LC-39A and VAFB SLC 4E boast a success rate of 77%.
- Success rates tend to increase with higher flight numbers, reaching 100% for VAFB SLC 4E after Flight 50, and for KSC LC 39A and CCAFS SLC 40 after the 80th flight.
- Payload vs. Launch Site analysis indicates higher success rates for heavy payloads in VAFB-SLC, particularly for Polar, LEO, and ISS, but distinguishing GTO outcomes is challenging.
- Success rates have consistently increased since 2013 until 2020, with no launches for heavy payloads (> 10000 kg).
- Orbits ES-L1, GEO, HEO & SSO boast 100% success rates, while SO orbit has the lowest success rate at ~50%, with 0% success in some cases.
- In LEO orbit, success seems related to the number of flights, but there's no apparent relationship between flight number and success in GTO orbit.

Thank you!