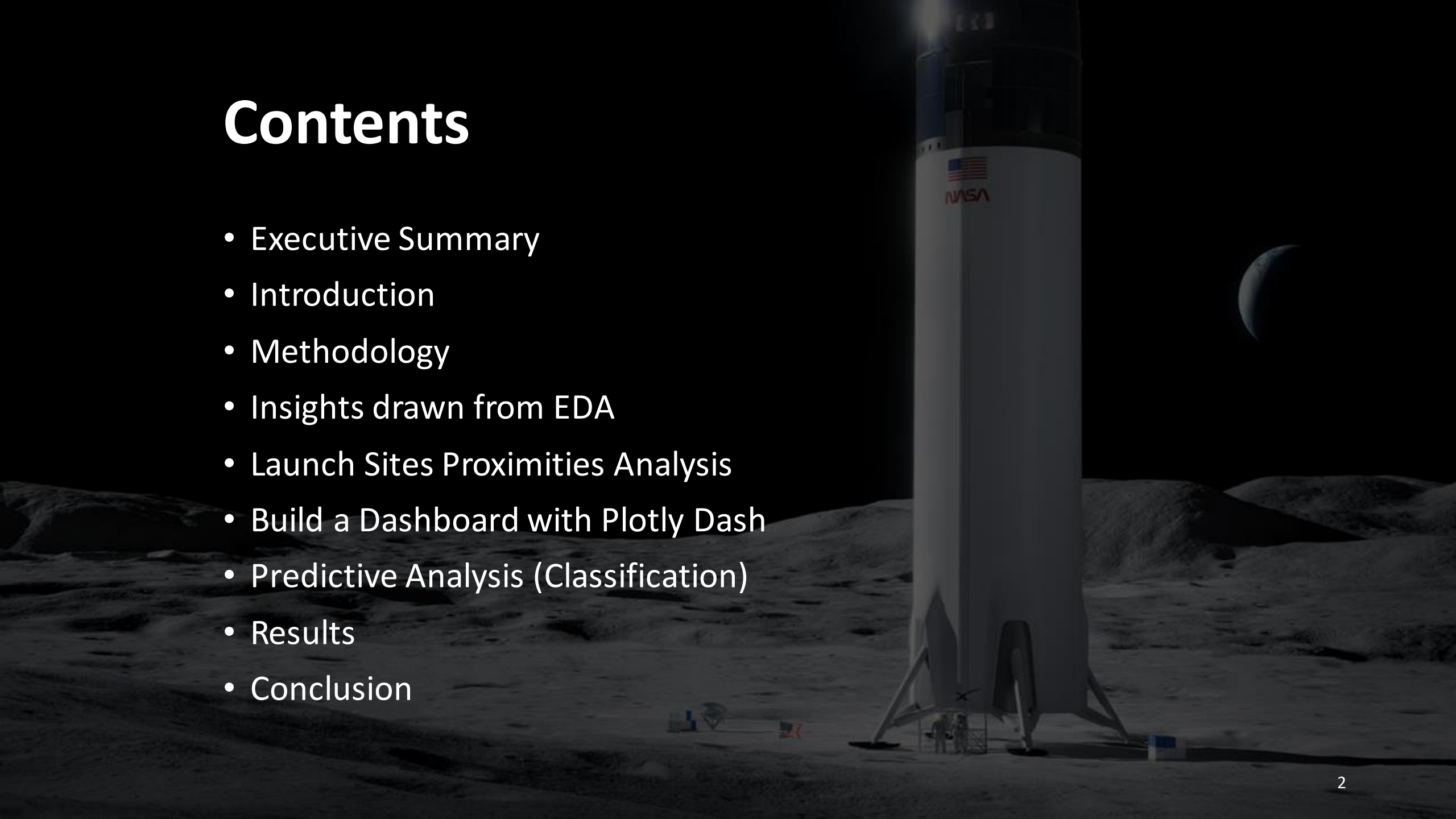
Predicting Falcon 9 First Stage Landing for Cost-Efficient Rocket Launches

Name: Shruti Prakash Karmarkar

Date: 28th January 2024

Contents



Executive Summary

Summary of methodologies

In this capstone project, I utilized a combination of data collection techniques, including API calls to the SpaceX REST API and web scraping using Python's BeautifulSoup package. The collected data underwent thorough wrangling, involving the normalization of structured JSON data into a tabular format. To enhance the dataset's relevance, I filtered out Falcon 1 launches and addressed missing values, specifically in the PayloadMass column. The subsequent analysis included exploratory data analysis (EDA), which uncovered insights into factors influencing successful rocket landings.
For predictive modeling, I employed machine learning algorithms, including logistic regression, support vector machines (SVM), k-nearest neighbors (KNN), and decision trees. The decision tree emerged as the most effective model, achieving an accuracy of 94%.

Summary of all results

Exploratory Data Analysis (EDA):

- Flight number correlates with a higher likelihood of successful first-stage landings.
- Payload mass inversely affects landing success, with heavier payloads decreasing success rates.
- Launch site "CCAFS LC-40" exhibits the highest success rate, with success increasing over time.
- Notable success in Polar, LEO, and ISS orbits, while GTO orbit outcomes are less distinguishable.

•Predictive Modeling:

- Decision tree outperforms logistic regression, SVM, and KNN, achieving an accuracy of 94%.
- Booster version "FT" demonstrates the most success, particularly for payload masses between 2k – 4k.
- Considerable challenges observed for booster version v1.1, with most launches ending unsuccessfully.

These outcomes underscore the importance of factors such as payload mass, launch site, and booster version in predicting successful rocket landings. The decision tree model stands out as a robust tool for forecasting outcomes, providing valuable insights for future space missions.

Introduction

- **Project background and context**

In the rapidly advancing field of space exploration, the reusability of rocket components has emerged as a game-changer. SpaceX, at the forefront of this revolution, employs reusable first stages for its Falcon 9 rockets, offering a cost-effective alternative to traditional launch methods. With each successful landing, SpaceX not only achieves a technological feat but also drives down the overall expense of rocket launches. This capstone project centers around predicting the outcome of Falcon 9 first stage landings, aiming to find the critical factors influencing success. By doing so, we aim to contribute insights that could reshape the landscape of cost-efficient and sustainable space travel.

- **Problems you want to find answers**

The core challenge we tackle in this capstone revolves around predicting the success of Falcon 9 first stage landings. The ability to reliably forecast whether the first stage will land successfully is of paramount importance, as it directly impacts the cost-effectiveness of rocket launches. By addressing this challenge, we seek to provide a comprehensive understanding of the variables and conditions that contribute to successful landings.

Section 1
Methodology

Methodology

Executive Summary

- **Data collection methodology:**
  - We're using SpaceX REST API to access the data, the file will be in the JSON format, therefore, I will use json_normalize function. This function will allow us to “normalize” the structured json data into a flat table. Our primary data source, the SpaceX REST API, provided comprehensive information about launches, encompassing details such as rocket specifics, payload data, launch specifications, landing details, and landing outcomes. The API's endpoints, notably "api.spacexdata.com/v4/launches/past," facilitated the extraction of past launch data.
  - Implemented web scraping techniques on Wiki pages to extract Falcon 9 launch records from HTML tables using BeautifulSoup.
  - A lot of columns have IDs as Data, therefore, I performed additional API requests to gather specific information, such as: "rocket" column to learn about the booster name.
- **Perform data wrangling**
  - Calculate the number of launches on each site, number and occurrence of each orbit, the number and occurence of mission outcome of the orbits.
  - We create a set of outcomes where the second stage did not land successfully
- **Perform exploratory data analysis (EDA) using visualization and SQL**
- **Perform interactive visual analytics using Folium and Plotly Dash**
- **Perform predictive analysis using classification models**
  - How to build, tune, evaluate classification models

Data Collection

Start requesting rocket launch data from SpaceX API with the following URL: "https://api.spacexdata.com/v4/launches/past"

Use json_normalize method to convert the json result into a dataframe

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.

Following details have been extracted:

- Booster name -> "rocket"
- payload mass / orbit -> "payload"
- launch site/latitude/longitude -> launchpad
- Landing outcomes/ types/ flights/ gridfins/ reuse status/ legs/ landing pads/ block numbers/ reuse counts/ and serials -> 'cores'.
- This data, stored in lists, was instrumental in creating a refined dataframe for further analysis.

Finally I have constructed our new dataset using the data we have obtained by combining the columns into a dictionary. Then, we need to create a Pandas data frame from the dictionary.

Filter the dataframe to only include Falcon 9 launches

Data Collection – SpaceX API

https://github.com/Shrutikarmarkar/IBM-Capstone-Project/blob/main/jupyter_labs_spacex_data_collection_api.ipynb

1. Request rocket launch data from SpaceX API with the following URL:
   - spacex_url= "https://api.spacexdata.com/v4/launches/past"
   - response = requests.get(spacex_url)

2. To make the requested JSON results more consistent, we will use the following static response object for this project:
   - static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
   - response = requests.get(static_json_url)

3. Now we decode the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize()
   - data = pd.json_normalize(response.json())

4. Takes the dataset and uses the rocket column to call the API and append the data to the list
   - def getBoosterVersion(data):
     for x in data['rocket']:
     if x:
     response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
     BoosterVersion.append(response['name'])

5. Takes the dataset and uses the launchpad column to call the API and append the data to the list
   - **def** getLaunchSite(data):
   - **for** x **in** data['launchpad']:
   - **if** x:
   - response **=** requests**.**get("https://api.spacexdata.com/v4/launchpads/"**+**str(x))**.**json()
   - Longitude**.**append(response['longitude'])
   - Latitude**.**append(response['latitude'])
   - LaunchSite**.**append(response['name'])

6. Takes the dataset and uses the payloads column to call the API and append the data to the lists
   - `def getPayloadData(data):`
     `for load in data['payloads']:`
       `if load:`
         `response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()`
         `PayloadMass.append(response['mass_kg'])`
         `Orbit.append(response['orbit'])`

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

Data Collection – Scraping

https://github.com/Shrutikarmarkar/IBM-Capstone-Project/blob/main/jupyter-labs-webscraping%20(1).ipynb

1. To keep the lab tasks consistent, I will scrape the data from a snapshot of the List of Falcon 9 and Falcon Heavy launches Wikipage updated on 9th June 2021
   - static_url = https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922

2. Perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.
   - response = requests.get(static_json_url)

3. Create a BeautifulSoup object from the HTML response
   - soup = bs(response.text, 'html.parser')

4. Extract all column/variable names from the HTML table header.
   - Use the find_all function in the BeautifulSoup object, with element type `table`
     - html_tables = soup.find_all("table")
   - Iterate through the <th> elements and extract column name one by one.

5. Create an empty dictionary "launch_dict" with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe.
   - df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })

6. Finally, convert the Dataframe into csv file.
   - df.to_csv('spacex_web_scraped.csv', index=False)

Data Wrangling

https://github.com/Shrutikarmarkar/IBM-Capstone-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

Calculate the number of launches on each site

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

Calculate the number and occurrence of each orbit

- Use the method .value_counts() to determine the number and occurrence of each orbit in the column Orbit

Calculate the number and occurence of mission outcome of the orbits

- Use the method .value_counts() on the column Outcome to determine the number of landing_outcomes.Then assign it to a variable landing_outcomes.

We create a set of outcomes where the second stage did not land successfully:

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

This variable will represent the classification variable that represents the outcome of each launch.

- df['Class']=landing_class

We can now export it to a CSV for the next section

- df.to_csv("dataset_part_2.csv", index=False)

Section 2
Insights Drawn from EDA

EDA with Data Visualization

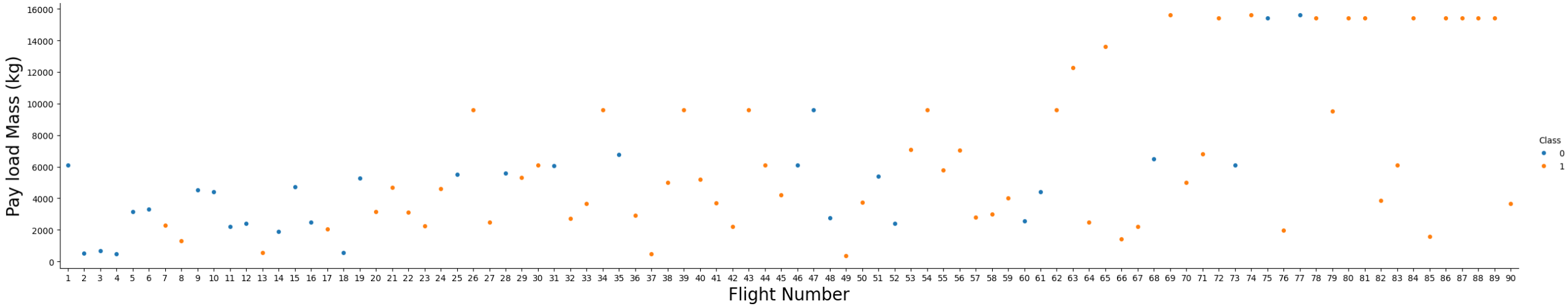
We will visualize the relationships between following variables:

- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Launch Site vs. Payload Mass
- Relationship between success rate and orbit type.
- Flight Number vs. Orbit
- Payload Mass vs. Orbit
- Visualize the launch success yearly trend

https://github.com/Shrutikarmarkar/IBM-Capstone-Project/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

Flight Number vs. Payload Mass

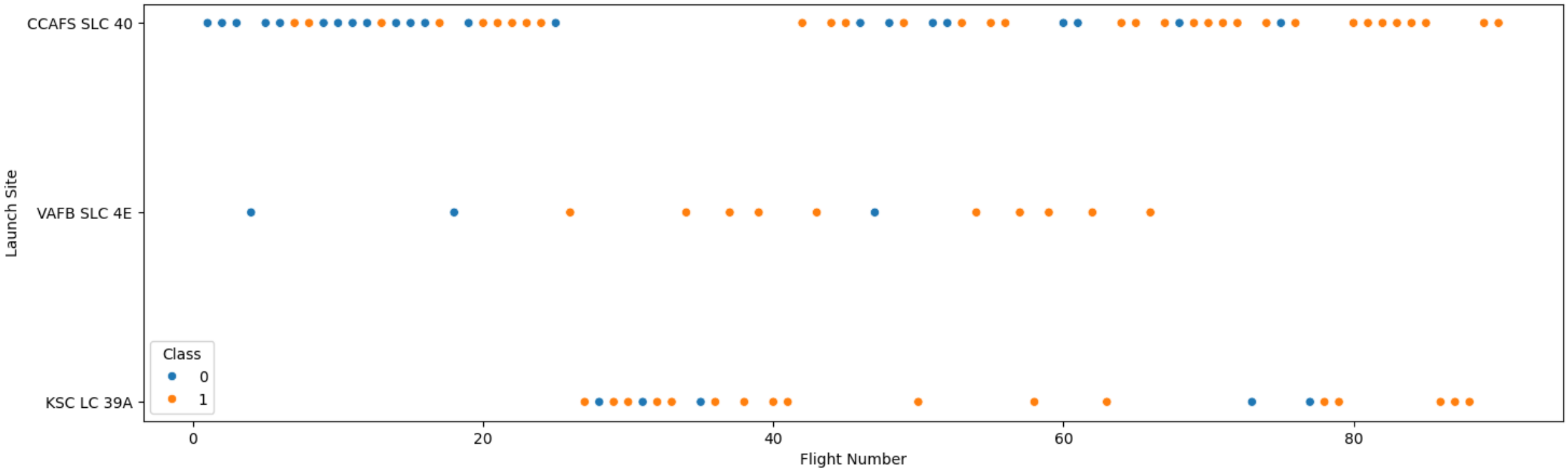
We will observe how the Flight Number (indicating the continuous launch attempts.) and Payload variables would affect the launch outcome.



We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

Flight Number vs. Launch Site

We will observe the success rate of each sites by observing the scatter plot between Flight Number vs. Launch Site.

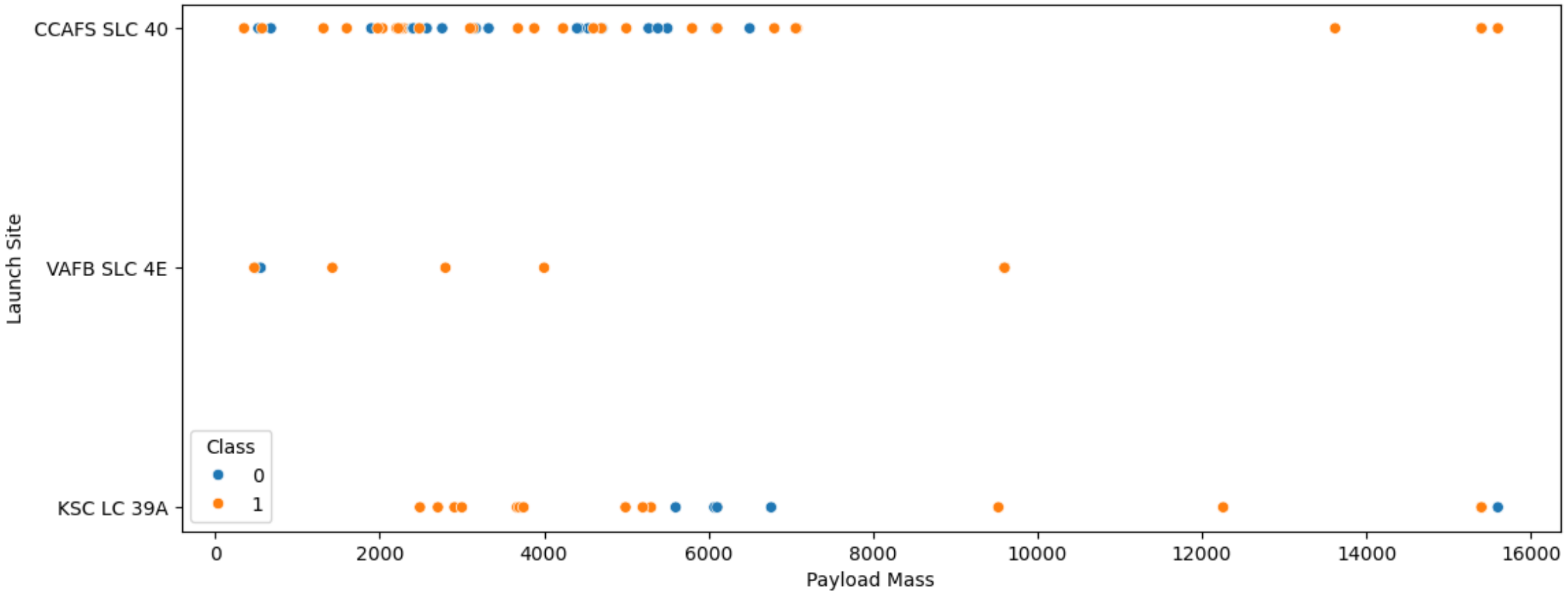


The plot above represents that:

- Most of the flights have been launched from site "CCAFS LC-40"
- Launch Site "CCAFS LC-40" has the highest success rate as compared to the other two
- The Lauch outcomes becomes more successful as the number of flights increases

Launch Site vs. Payload Mass

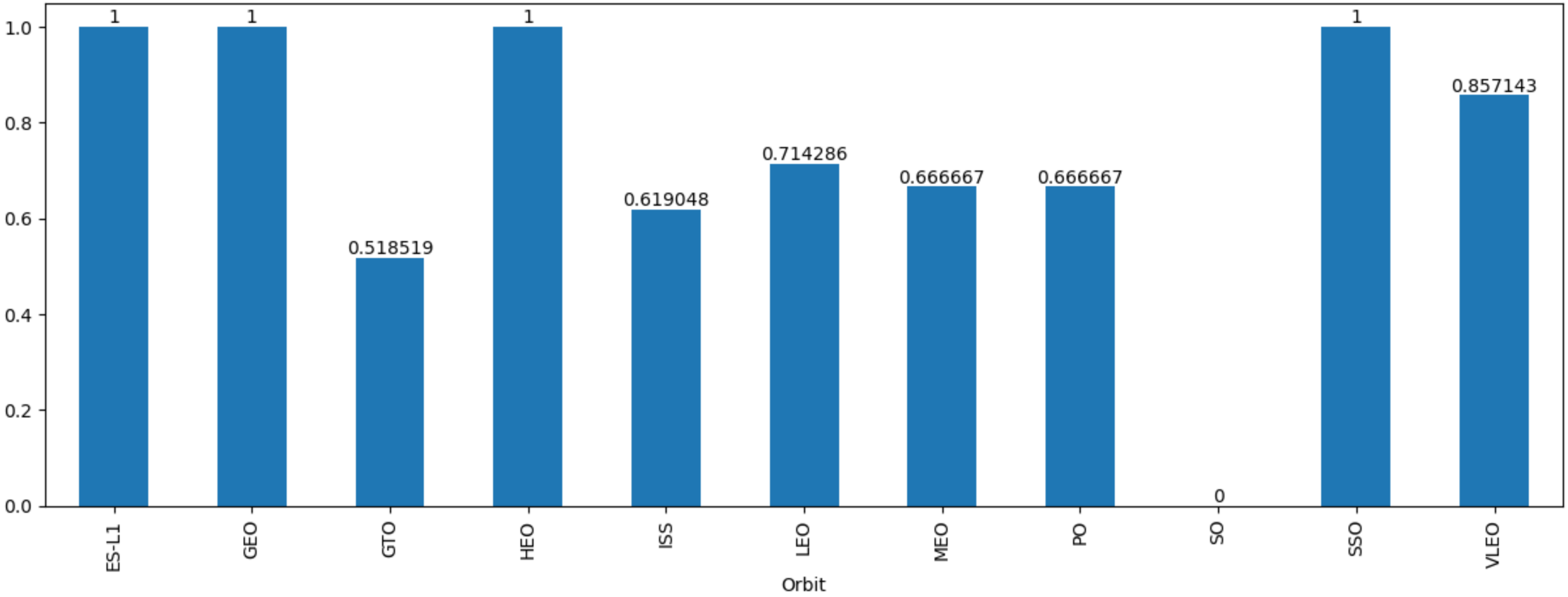
We will observe if the Payload Mass has any effect on the success of a launch at a particular Launch Site.



In Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch site there are no rockets launched for heavy payload mass(greater than 10000).

Relationship between success rate and orbit type.

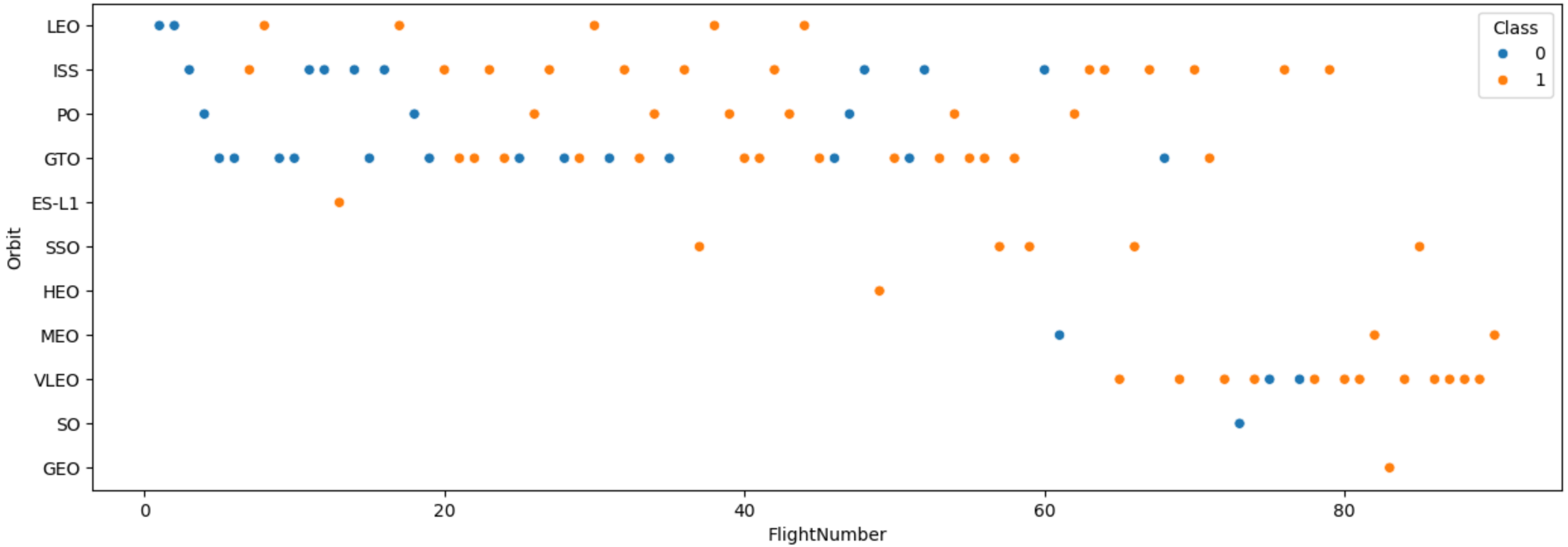
We want to visually check if there are any relationship between success rate and orbit type.



The above bar plot illustrates that four orbits namely, ES-L1, GEO, HEO, and SSO have highest success rates, whereas the orbit SO has lowest i.e., zero success rate.

Flight Number vs. Orbit

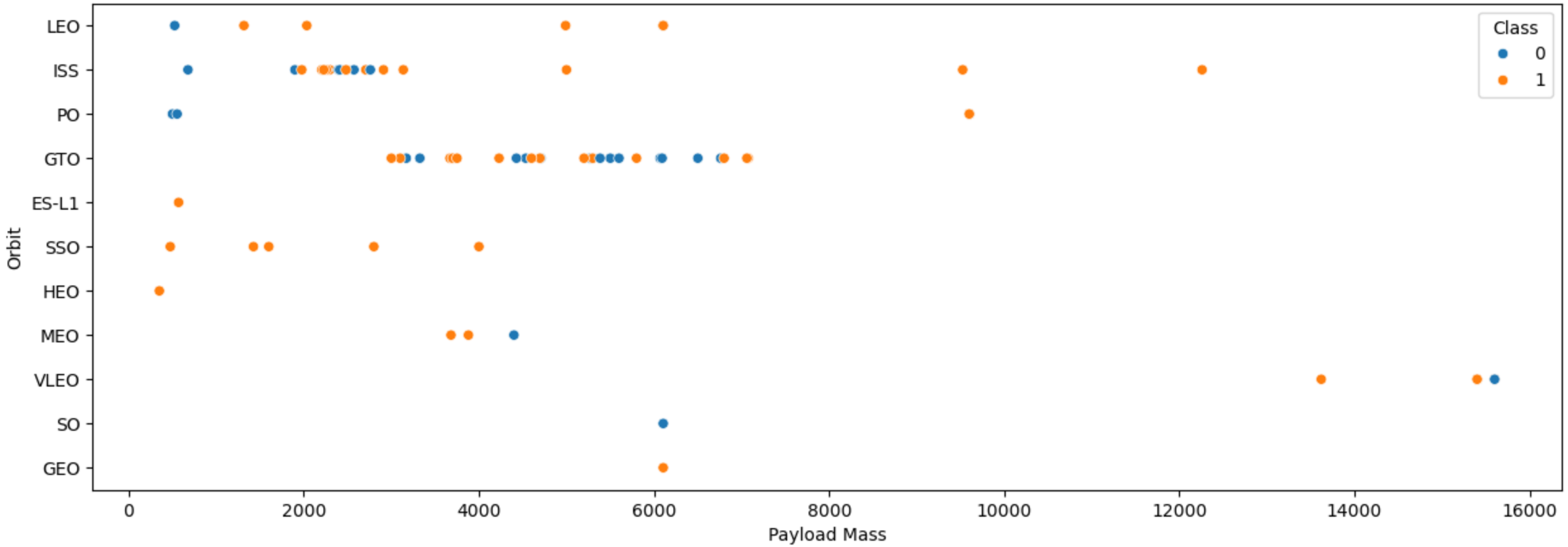
For each orbit, we want to see if there is any relationship between Flight Number and Orbit type.



You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload Mass vs. Orbit

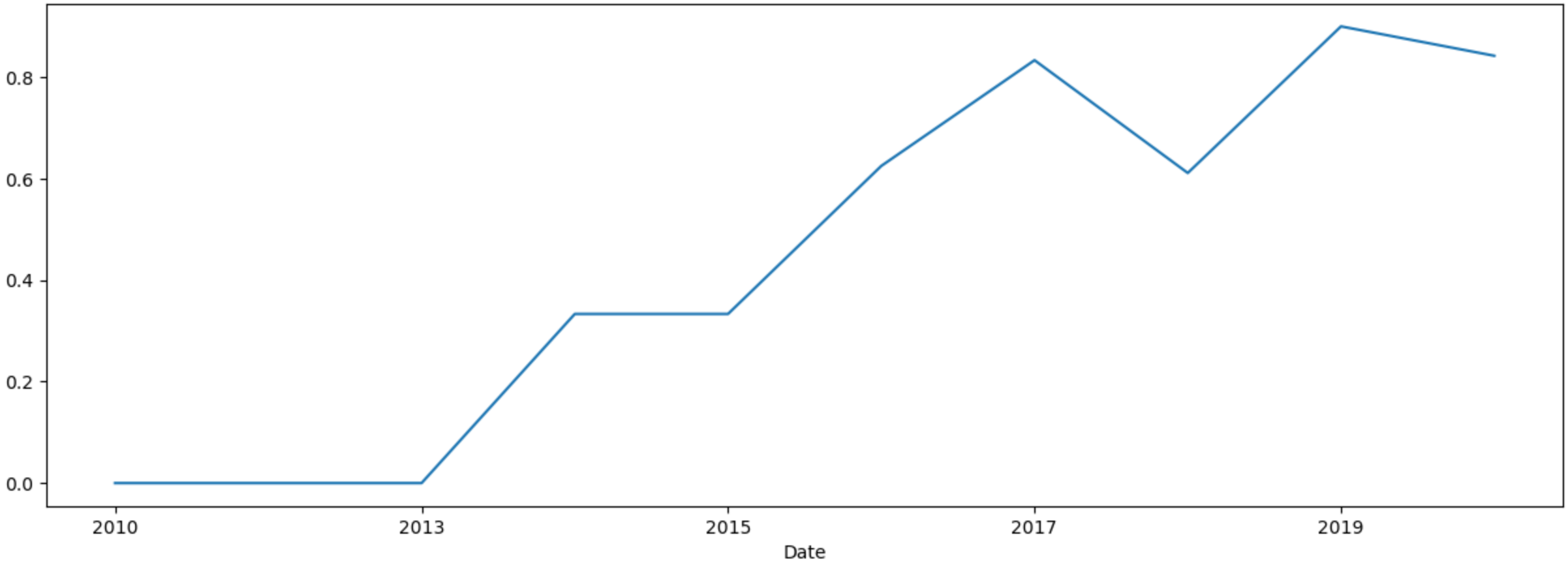
We can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type



With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.
However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.

Visualize the launch success yearly trend

We can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type



For the above line plot you can observe that the success rate since 2013 kept increasing till 2020

EDA with SQL

https://github.com/Shrutikarmarkar/IBM-Capstone-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

All Launch Site Names

```
%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

1. **Distinct Launch Sites**: Applied a **DISTINCT** query to extract unique launch site names from the SPACEXTABLE.

2. **Query Syntax**: Employed the SQL query **%sql select distinct Launch_Site from SPACEXTABLE** for this purpose.

Launch Site Names Begin with 'CCA'

```
%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
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (
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 (
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	N
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	N
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	N

1.Employed the SQL query **%sql select * from SPACEXTABLE where Launch_Site like "CCA%" limit 5** to filter and display five records from the SPACEXTABLE.

2.Utilized the **WHERE** clause to specify the condition for filtering, focusing on launch sites that start with the acronym 'CCA.'

3.The **LIKE** operator was applied to match launch site names containing the specified prefix ('CCA') while allowing for flexibility in the matching pattern.

4.The **LIMIT 5** constraint ensured the retrieval of a limited set of records, offering a concise overview of the relevant data.

5.This query aids in examining a subset of the dataset associated with launch sites beginning with 'CCA,' facilitating targeted analysis and insights into missions related to these specific sites.

Total Payload Mass

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer like "NASA (CRS)"
```

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

sum(PAYLOAD_MASS__KG_)
45596

1.Executed the SQL query **%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTABLE where Customer like "NASA (CRS)".**

2.Applied the SUM function to calculate the total payload mass (PAYLOAD_MASS__KG_) for missions associated with the customer "NASA (CRS)."

3.The WHERE clause with the condition Customer like "NASA (CRS)" filtered the data, focusing exclusively on missions conducted for NASA under the Commercial Resupply Services (CRS) program.

Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version like "F9 v1.1"
```

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

avg(PAYLOAD_MASS__KG_)
2928.4

1. Executed the SQL query **%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTABLE where Booster_Version like "F9 v1.1".**
2. Utilized the AVG function to calculate the average payload mass (PAYLOAD_MASS__KG_) for missions associated with the booster version "F9 v1.1."
3. The WHERE clause with the condition Booster_Version like "F9 v1.1" filtered the data, focusing exclusively on missions where the specified booster version was used.

First Successful Ground Landing Date

```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome like "Success (ground pad)"
```

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

min(Date)
2015-12-22

1. Executed the SQL query **`%sql select min(Date) from SPACEXTABLE where Landing_Outcome like "Success (ground pad)"`**.
2. Utilized the **`MIN`** function to retrieve the earliest date (**`Date`**) associated with successful ground pad landings.
3. The **`WHERE`** clause with the condition **`Landing_Outcome like "Success (ground pad)"`** filtered the data, focusing specifically on missions where the landing outcome indicated a successful landing on a ground pad.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select Booster_Version from SPACEXTABLE where Landing_Outcome like "Success (drone ship)" and PAYLOAD_MASS__KG
```

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

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

1.Executed the SQL query **%sql select Booster_Version from SPACEXTABLE where Landing_Outcome like "Success (drone ship)" and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000**.

2.Utilized the SELECT statement to retrieve the Booster_Version information.

3.The FROM clause specifies the table SPACEXTABLE from which the data is selected.

4. The WHERE clause includes multiple conditions:
   - Landing_Outcome like "Success (drone ship)": Filters for missions with a successful landing on a drone ship.
   - PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000: Further refines the selection to missions where the payload mass is between 4000 kg and 6000 kg.

Total Number of Successful and Failure Mission Outcomes

```
%sql select  Mission_Outcome, count(*) from SPACEXTABLE group by Mission_Outcome
```

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

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

1. Executed the SQL query %sql select Mission_Outcome, count(*) from SPACEXTABLE group by Mission_Outcome.
2. Applied the SELECT statement to retrieve information about Mission_Outcome and the count of occurrences.
3. The FROM clause specifies the table SPACEXTABLE from which the data is selected.
4. Used the GROUP BY clause with Mission_Outcome to group the results based on mission outcomes.
5. The COUNT(*) function calculates the number of occurrences for each unique mission outcome.

Boosters Carried Maximum Payload

```
%sql select Booster_Version, PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS_
```

```
 * 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

1.Utilized the SQL query **%sql select Booster_Version, PAYLOAD_MASS__KG_ from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE)**.

2.The inner query (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE) identifies the maximum payload mass recorded in the dataset.

3.The outer query selects the Booster_Version and PAYLOAD_MASS__KG_ columns for entries where the payload mass matches the maximum value.

4.This query is designed to retrieve information about the booster version associated with the highest payload mass in the dataset.

2015 Launch Records

```
%sql select substr(Date, 6,2) as Month_Names, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE where
```

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

Month_Names	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

1. Employed the SQL query **%sql select substr(Date, 6,2) as Month_Names, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE where Landing_Outcome like 'Failure (drone ship)' and substr(Date,0,5)='2015'**.
2. Utilized the **substr** function to extract the month information from the **Date** column and renamed it as **Month_Names**.
3. Filtered results to include entries where the landing outcome is categorized as 'Failure (drone ship)' and the year is 2015.
4. Selected columns including the month names, landing outcome, booster version, and launch site for analysis.
5. This query is designed to provide details about instances where drone ship landings failed in the year 2015, including associated booster versions, launch sites, and month information.

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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%sql select Landing_Outcome, count(*) as Count from SPACEXTABLE where Date between "2010-06-04" and "2017-03-20" group
```

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

Landing_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

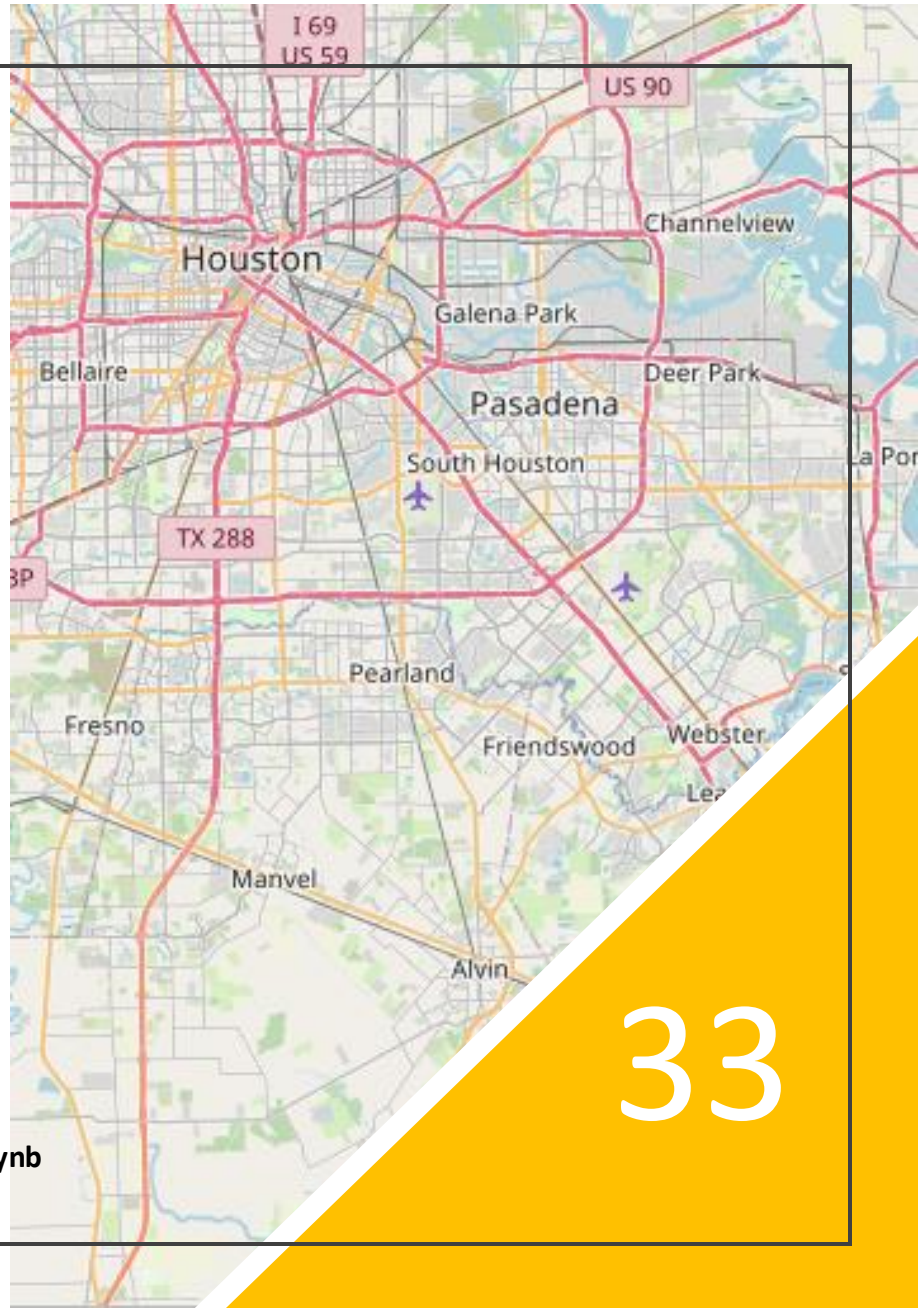
Employed the SQL query **%sql select Landing_Outcome, count(*) as Count from SPACEXTABLE where Date between "2010-06-04" and "2017-03-20" group by Landing_Outcome order by Count desc;**

This SQL query retrieves the count of different landing outcomes from the "SPACEXTABLE" dataset for entries with dates falling between "2010-06-04" and "2017-03-20." The results are grouped by the "Landing_Outcome" column and sorted in descending order based on the count. The query aims to provide insights into the distribution of landing outcomes during the specified time range.

Section 3
Launch Sites Proximities Analysis

Build an Interactive Map with Folium

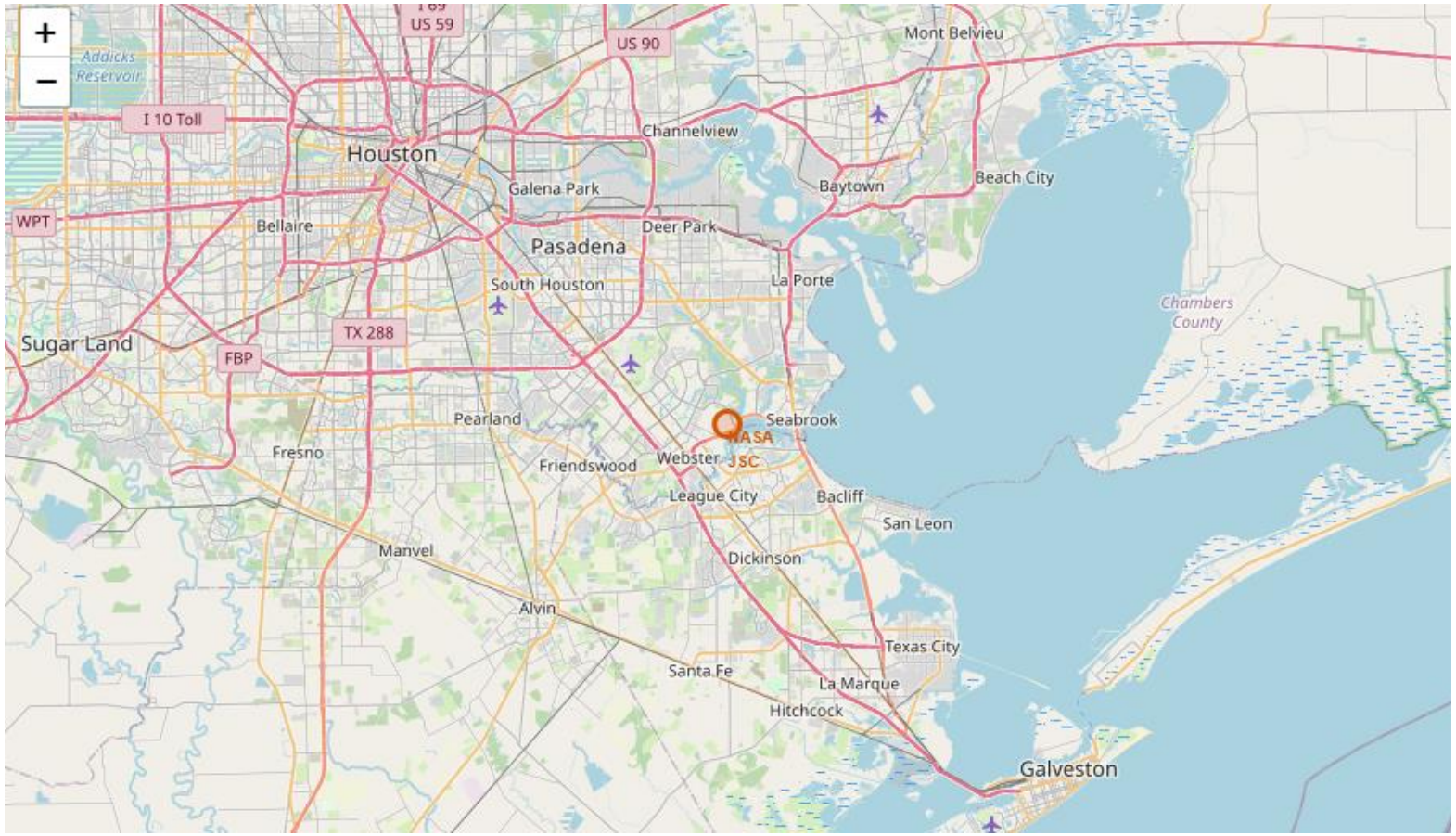
- We will perform the following operations:
- Mark all launch sites on a map
- Mark the success/failed launches for each site on the map
- Calculate the distances between a launch site to its proximities
- Calculate the distance between the coastline point and the launch site.
- Distance to a closest city, railway, highway



https://github.com/Shrutikarmarkar/IBM-Capstone-Project/blob/main/lab_jupyter_launch_site_location.jupyterlite%20(1).ipynb

Mark all launch sites on a map

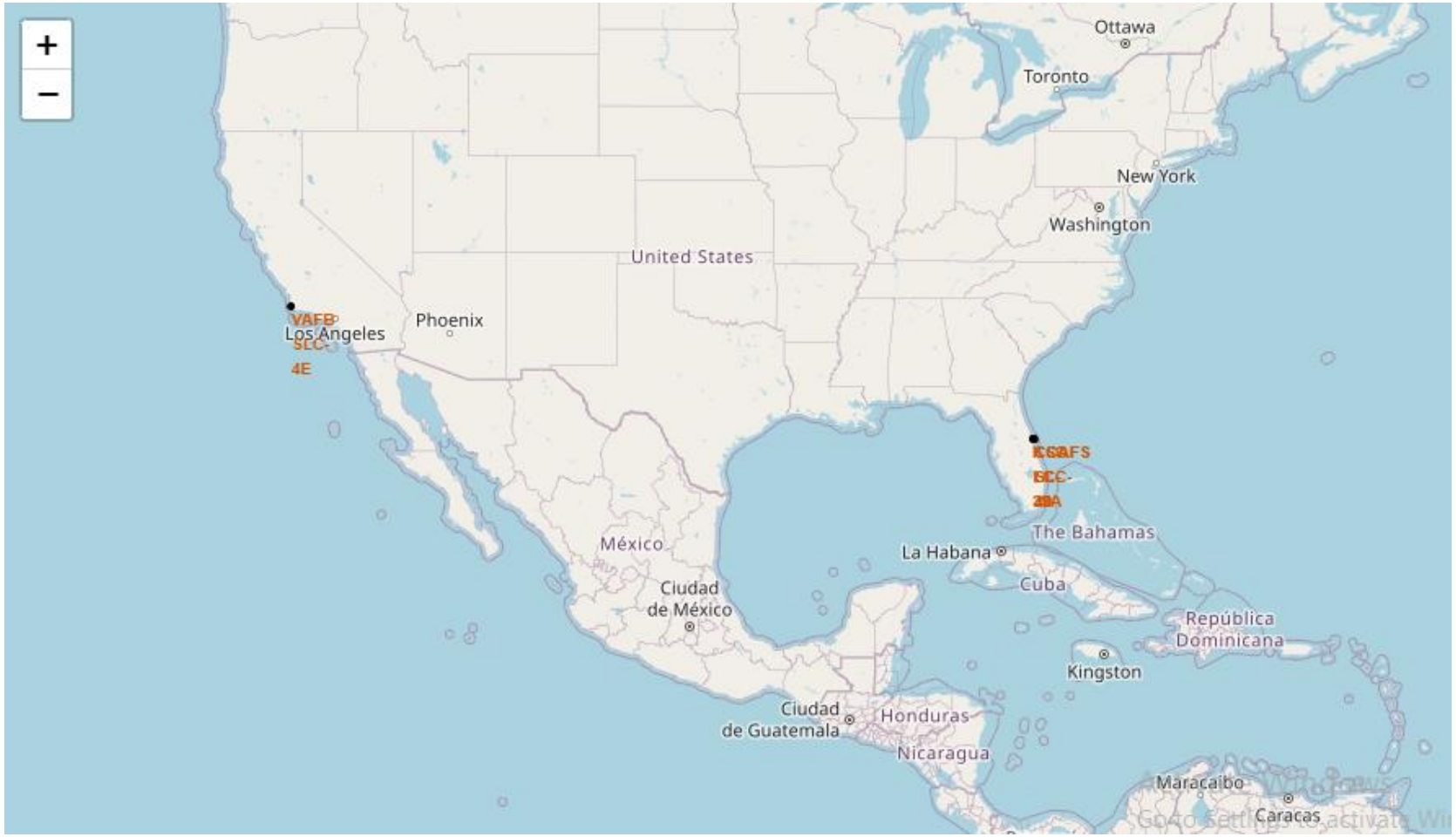
spacex_launch_geo.csv dataset is an augmented dataset with latitude and longitude added for each site. We will add each site's location on a map using site's latitude and longitude coordinates However, the coordinates are just plain numbers that cannot give you any intuitive insights about where are those launch sites. Therefore, we will visualize those locations by pinning them on a map. We could use **folium.Circle** to add a highlighted circle area with a text label on a specific coordinate.



You should find a small yellow circle near the city of Houston. That location is 'NASA Johnson Space Center'

Mark all launch sites on a map

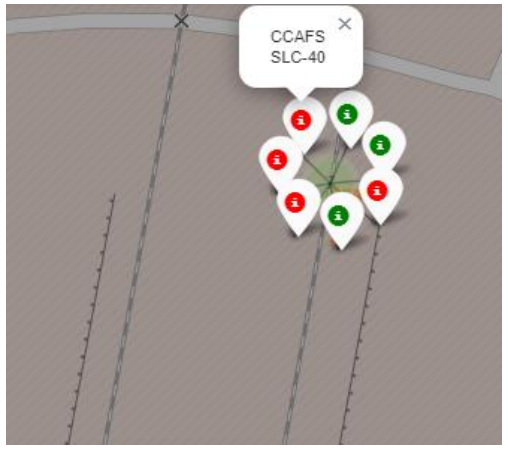
Added **folium.Circle** and **folium.Marker** for each launch site on the site map

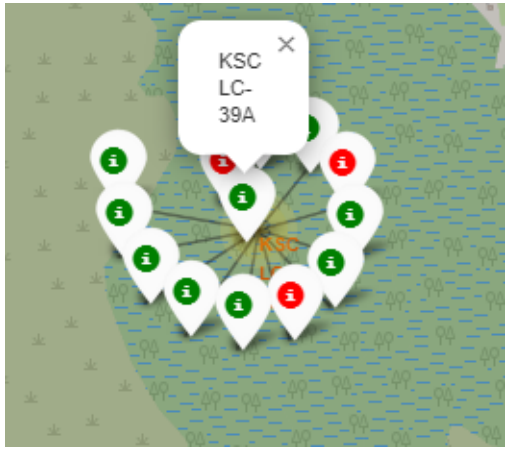


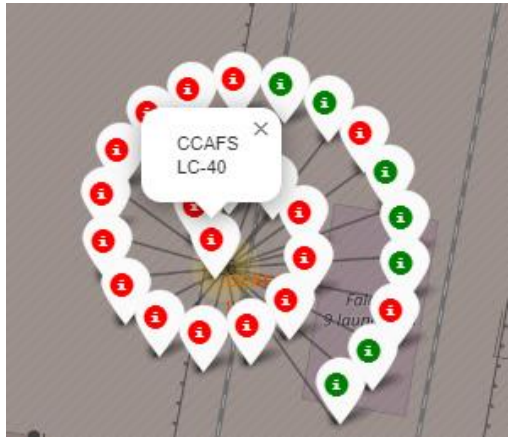
- All launch sites are in very close proximity to the coast.
- Launch sites CCAFS LC-40, CCAFS SLC-40, KSC LC-39A are very close to each other, whereas,, site VAFB SLC-4E is completely at the opposite direction

Mark the success/failed launches for each site on the map

- Added the launch outcomes for each site, and see which sites have high success rates.
- Created markers for all launch records. If a launch was successful (class=1), then we use a green marker and if a launch was failed, we use a red marker (class=0)
- launch only happens in one of the four launch sites, which means many launch records will have the exact same coordinate. **Marker clusters** can be a good way to simplify a map containing many markers having the same coordinate.







From the color-labeled markers in marker clusters, you should be able to easily identify which launch sites have relatively high success rates which is: KSC LC- 39A

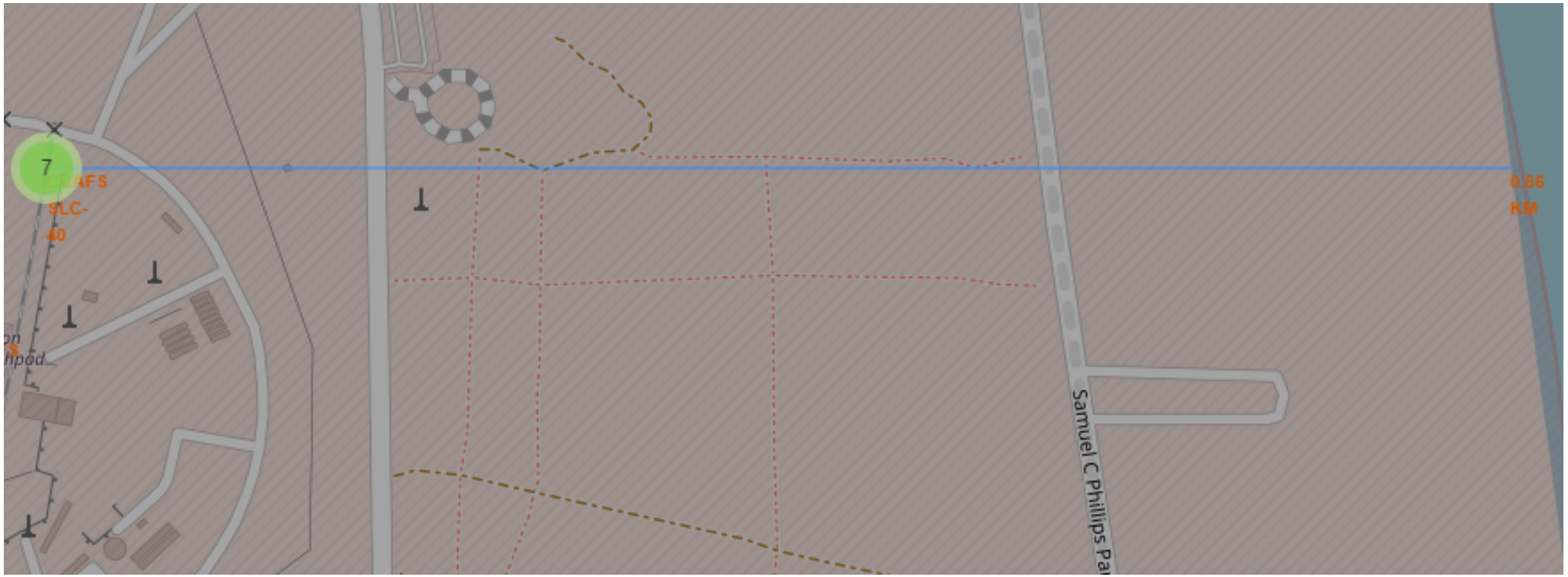
Calculate the distances between a launch site to its proximities

We need to explore and analyze the proximities of launch sites. Added a **`MousePosition`** on the map to get coordinate for a mouse over a point on the map. As such, while we are exploring the map, we can easily find the coordinates of any points of interests (such as railway)

After zooming in to a launch site and exploring its proximity to we can find railway, highway, coastline, etc. We can move the mouse to these points and mark down their coordinates (shown on the top-left) in order to calculate the distance to the launch site.

Calculate the distance between the coastline point and the launch site.

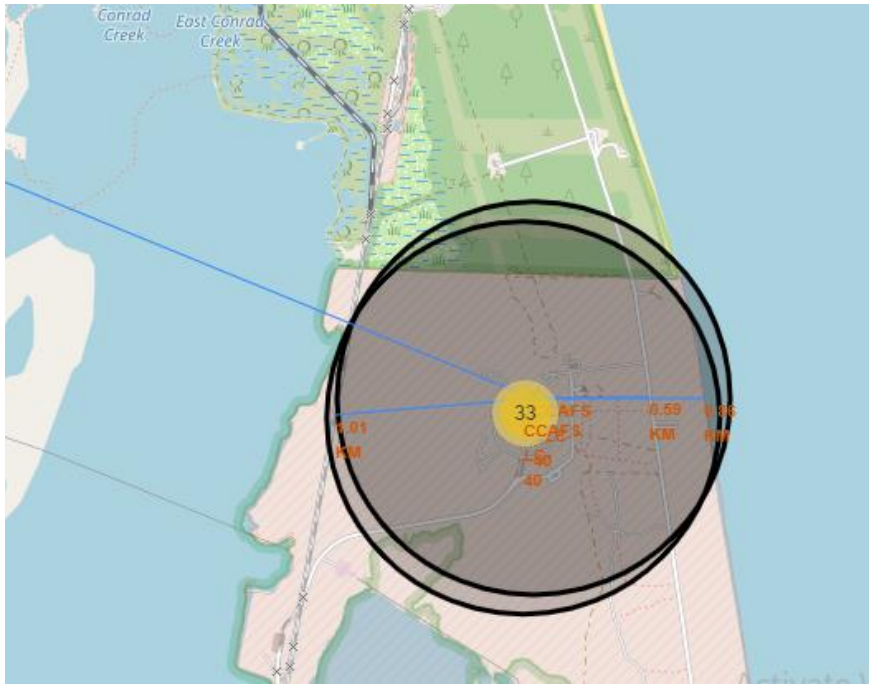
- Find coordinate of the closet coastline using Mouse Pointer and put a marker on it. Then define a function to calculate the distance between the coastline and the Launch site.
- Draw a **PolyLine** between a launch site to the selected coastline point

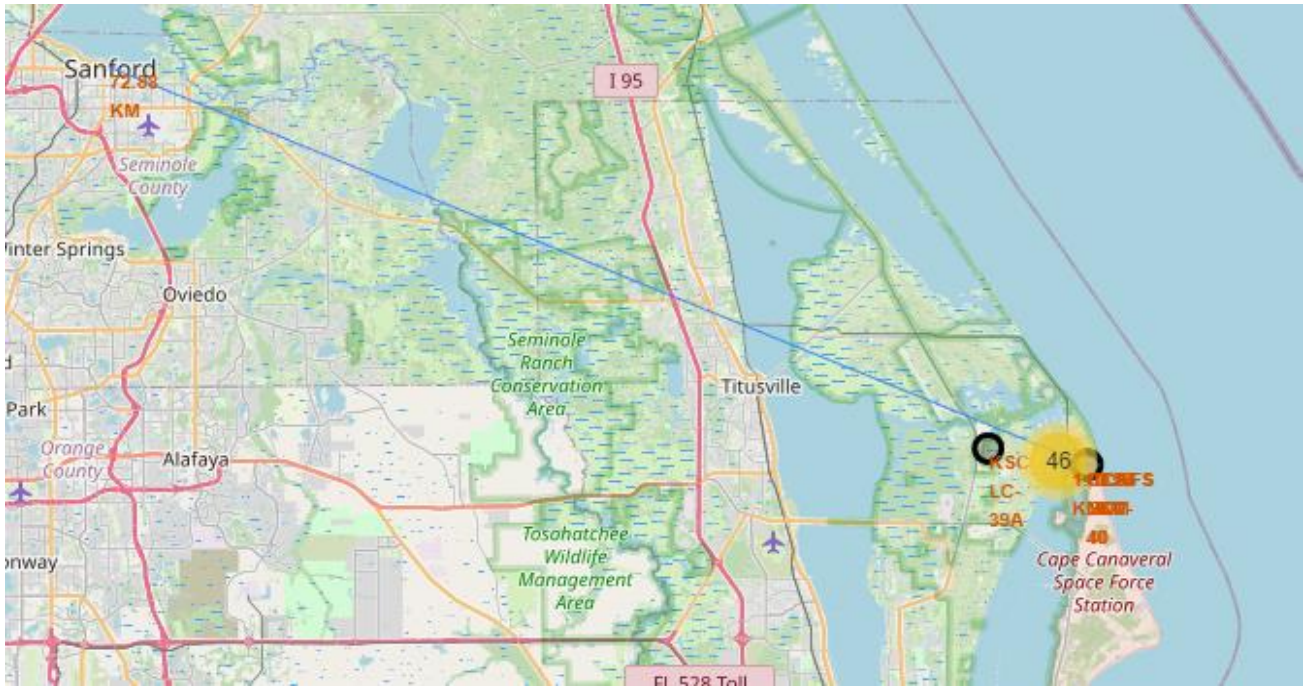


The nearest coastline is 0.86 km away, therefore, we can say that the Launch site is at a close proximity to the coastline.

Distance to a closest city, railway, highway

- Find coordinate of the closet coastline using Mouse Pointer and put a marker on it. Then define a function to calculate the distance between the coastline and the Launch site.
- Draw a **PolyLine** between a launch site to the selected coastline point



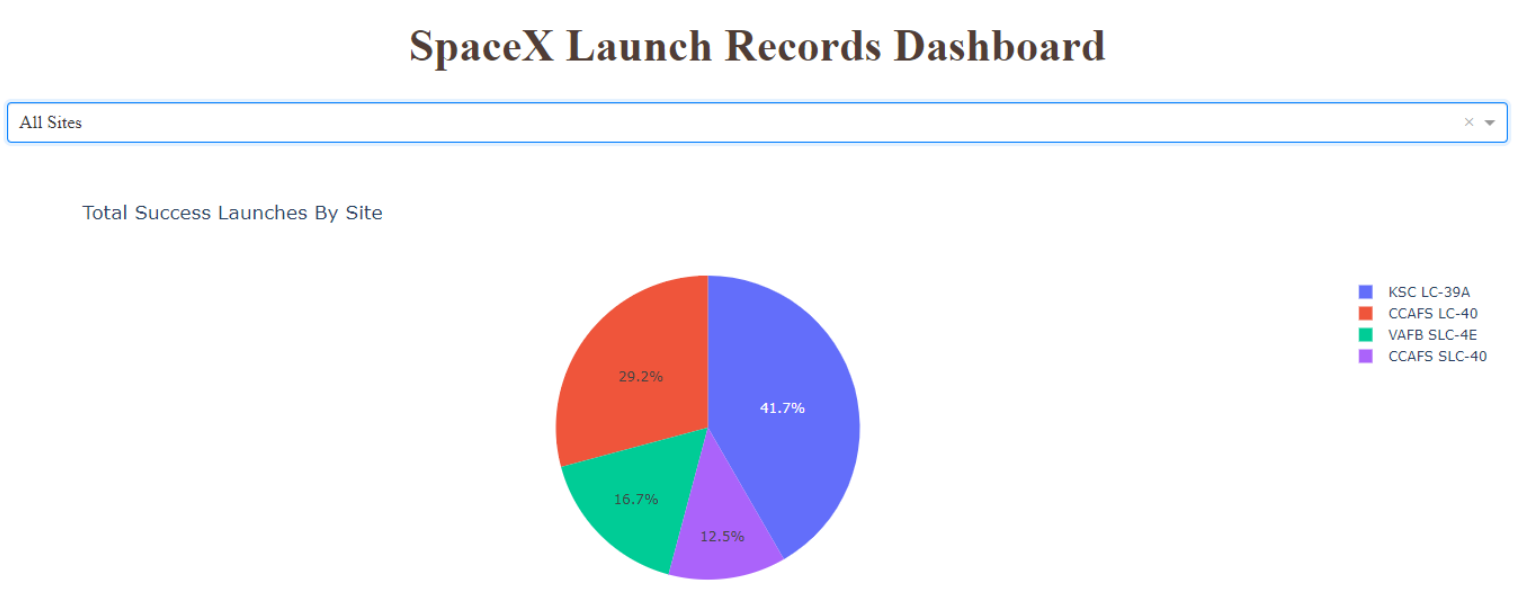


- Are launch sites in close proximity to railways?
  - Yes
- Are launch sites in close proximity to highways?
  - Yes
- Are launch sites in close proximity to coastline?
  - Yes
- Do launch sites keep certain distance away from cities?
  - Yes

Section 4
Build a Dashboard with Plotly Dash

https://github.com/Shrutikarmarkar/IBM-Capstone-Project/blob/main/spacex_dash_app.py

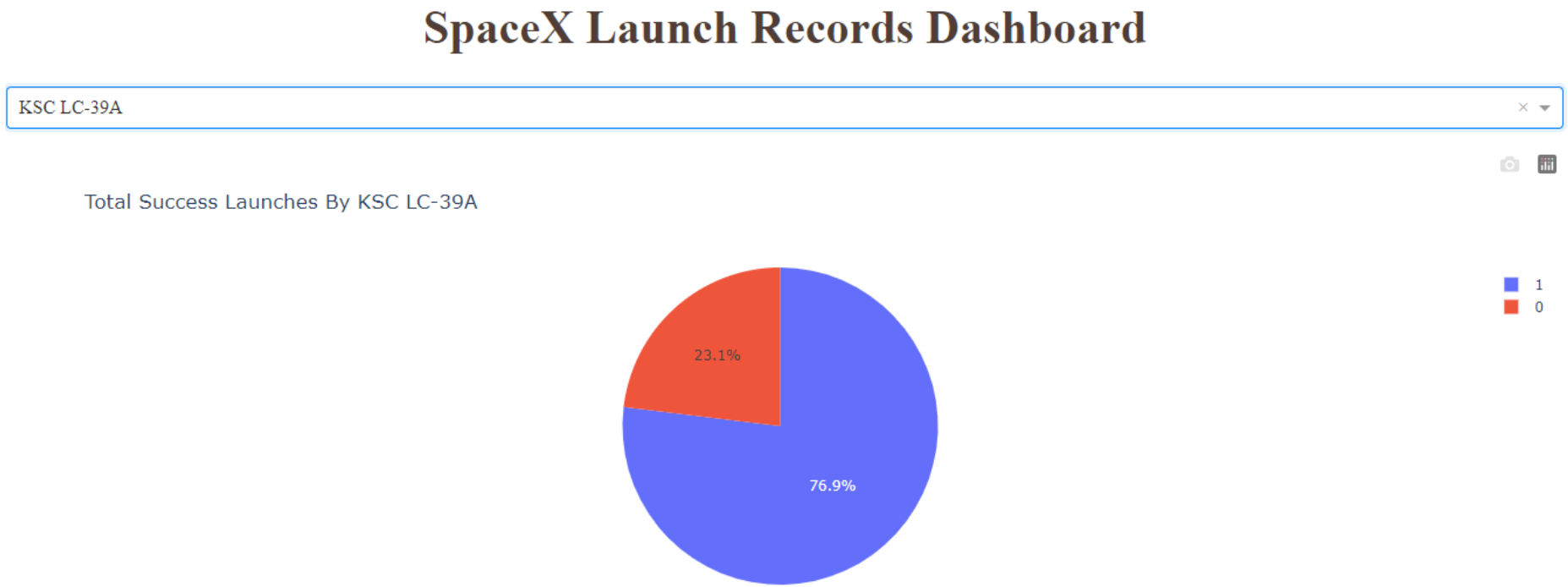
Launch Success Count for All Sites



A dropdown menu was implemented using the dcc.Dropdown component in Plotly Dash for selecting different launch sites. The dropdown includes options for each launch site available in the spacex_df dataframe, along with a default 'All Sites' option. Users can choose a specific launch site or select 'All Sites' to view aggregated data. The dropdown is designed to be searchable, allowing users to easily locate and select launch sites of interest.

Analysis: The success rates for SpaceX Falcon 9 rocket launches vary across different launch sites. Kennedy Space Center's Launch Complex 39A stands out with a success rate of 41.7%, indicating a relatively high proportion of successful landings. In contrast, Cape Canaveral Air Force Station's Space Launch Complex 40 has a lower success rate of 12.5%, while Launch Complex 40 at the same station achieves a moderate success rate of 29.2%. Vandenberg Air Force Base's Space Launch Complex 4E shows a success rate of 16.7%.

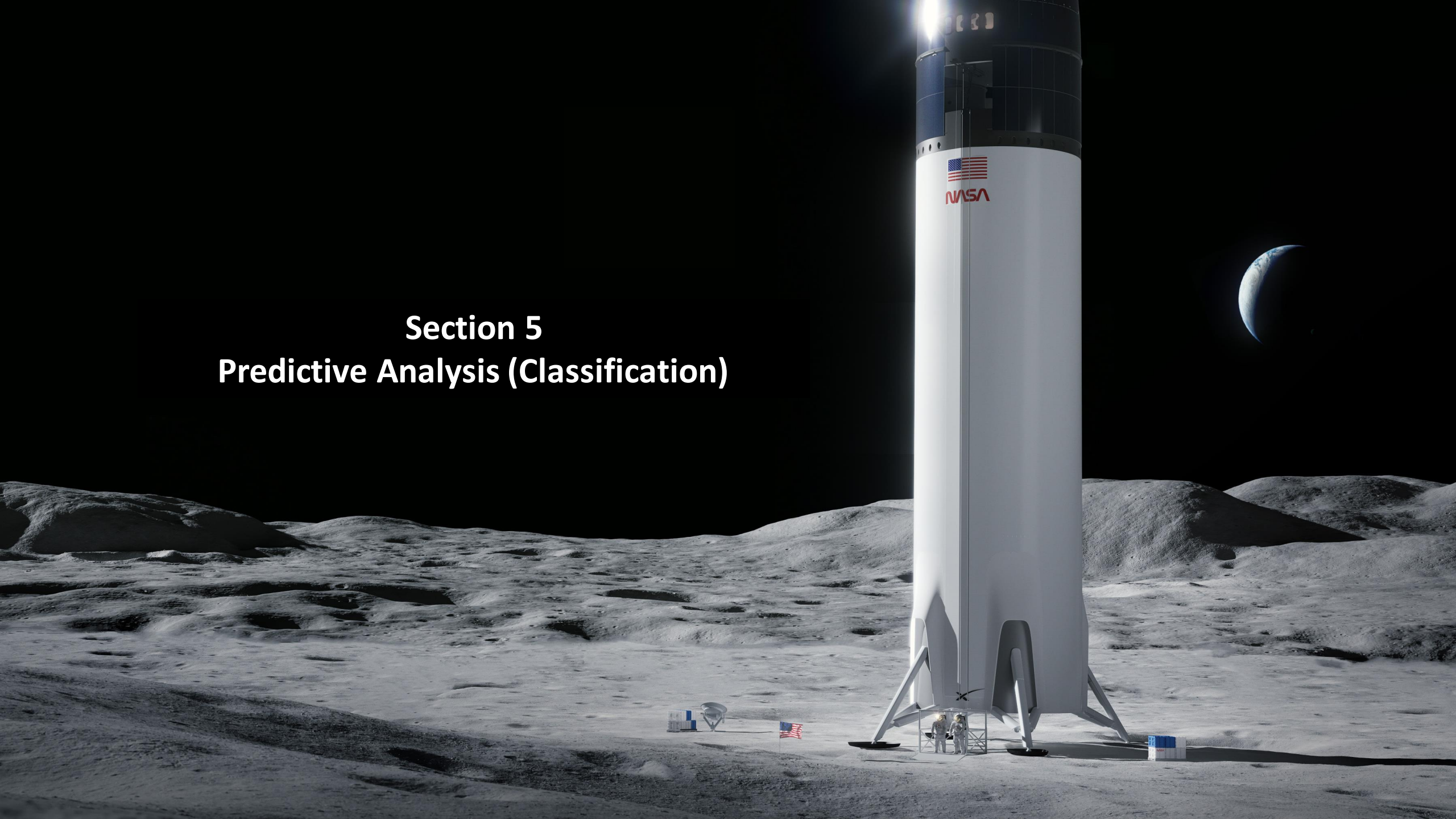
Launch site with highest launch success ratio



Analysis: The launch site with the highest launch success ratio is associated with a class 1 success rate of 76.9% and a class 0 failure rate of 23.1%. This indicates a significant success predominance, suggesting that the specified launch site has a strong track record of successful Falcon 9 rocket landings. The disparity between the success and failure percentages emphasizes the reliability and effectiveness of operations at this particular launch site in ensuring positive mission outcomes.

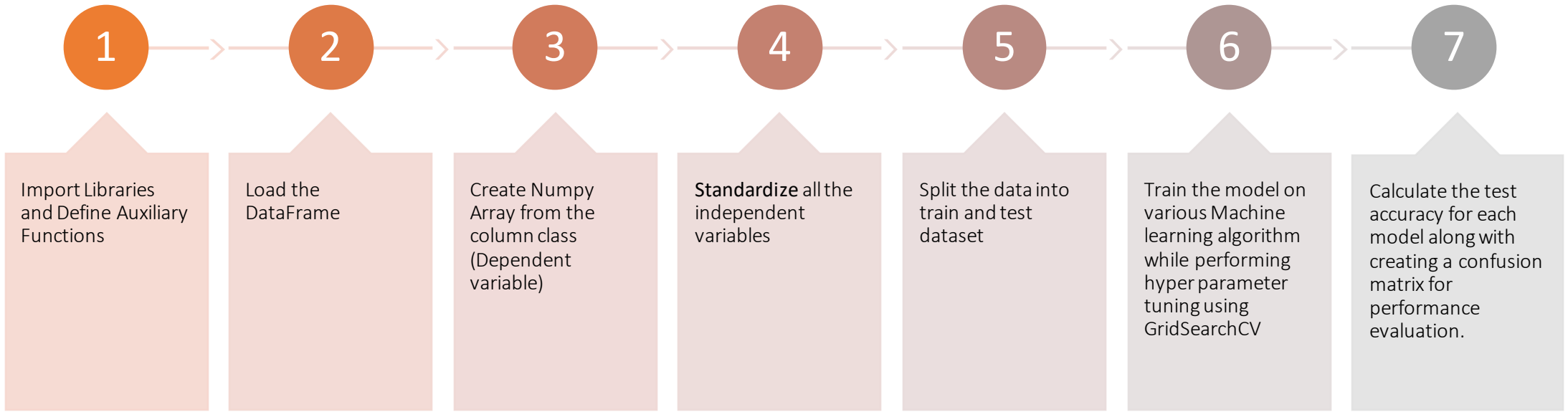
Payload vs. Launch Outcome scatter plot for all sites

Analysis: The analysis reveals that the Booster version "FT" exhibits the highest success rate across all launch sites, especially within the payload range of 2k – 4k. In contrast, launches using the booster version v1.1 have shown a consistent trend of being unsuccessful, indicating a potential reliability or performance issue associated with this specific booster version. This insight can guide future decisions on booster selection, emphasizing the importance of considering historical performance when planning launches.

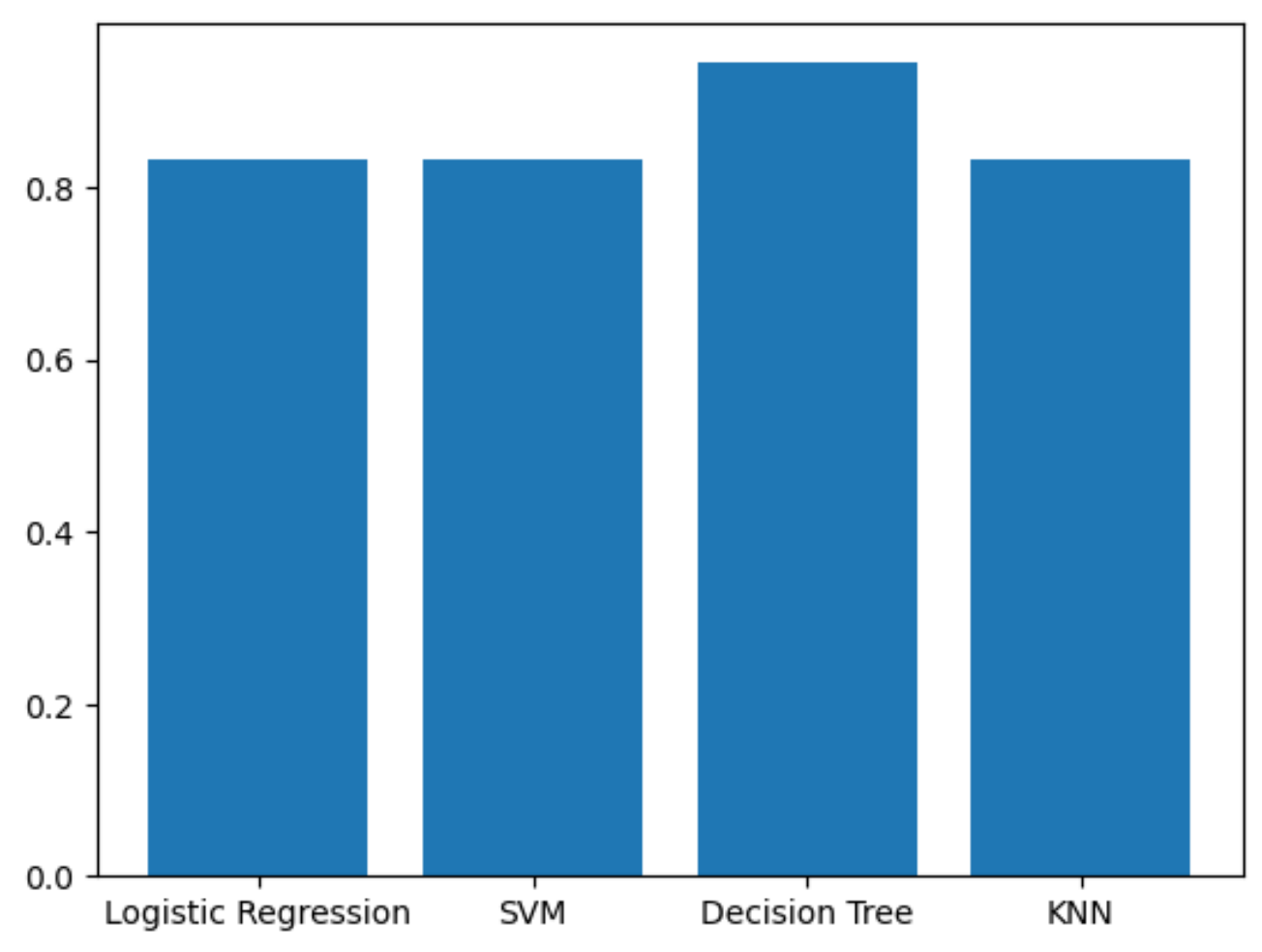


Section 5
Predictive Analysis (Classification)

Machine Learning Workflow



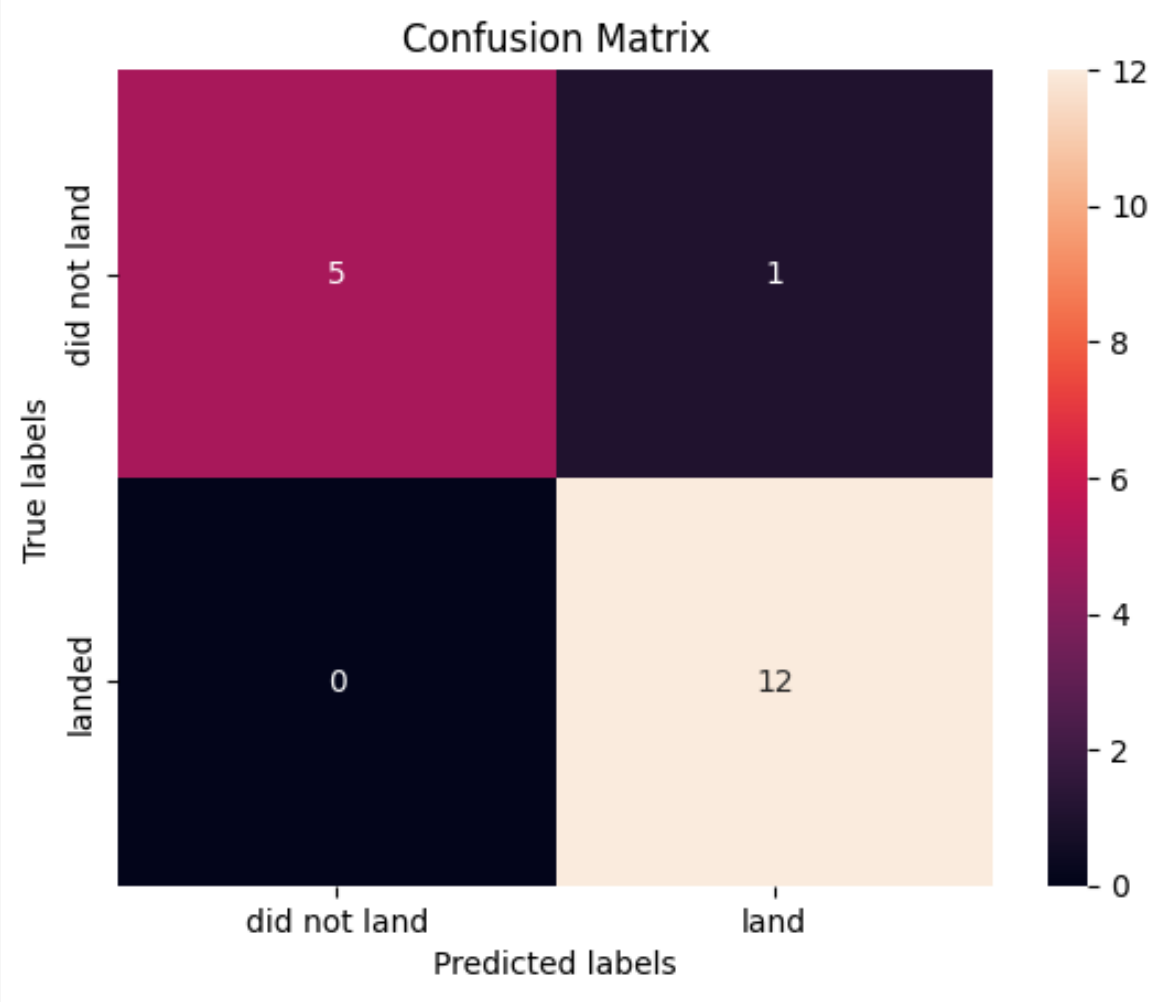
https://github.com/Shrutikarmarkar/IBM-Capstone-Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite%20(1).ipynb



Classification Accuracy

Analysis: The **decision tree** emerges as the top-performing model among logistic regression, SVM, and KNN, boasting an impressive accuracy rate of **94%.** This signifies that the decision tree algorithm is particularly effective in capturing the underlying patterns and relationships within the given data, outperforming its counterparts in making accurate predictions.

The choice of the decision tree as the preferred model could be attributed to its ability to handle complex decision boundaries and adapt well to the nature of the dataset, showcasing its suitability for the specific task at hand.



Confusion Matrix for Decision Tree

In a confusion matrix with "Did Not Land" and "Landed" as two outcomes, each cell of the matrix represents a different combination of predicted and actual classes. Here's a breakdown:

- True Positive (TP): The model correctly predicted all 12 "Landed" when the actual outcome was "Landed."
- True Negative (TN): The model correctly predicted 5/6 "Did Not Land" when the actual outcome was "Did Not Land."
- False Positive (FP): The model incorrectly predicted 1/6 "Landed" when the actual outcome was "Did Not Land" (Type I error).
- False Negative (FN): The model did not incorrectly predicted "Did Not Land" when the actual outcome was "Landed" (Type II error).

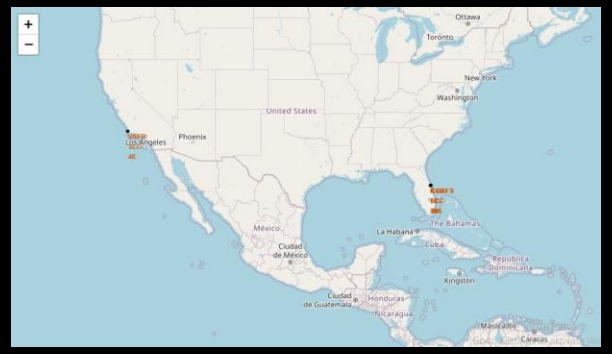
Results

1. **Exploratory Data Analysis Results from Visualization:**
   - Positive correlation observed between increasing flight numbers and successful first-stage landings.
   - Payload mass inversely related to first-stage return success, indicating heavier payloads might impact successful landings negatively.
   - "CCAFS LC-40" exhibits the highest success rate compared to other launch sites.
   - Success rates generally improve with a higher number of flights.
   - Four orbits (ES-L1, GEO, HEO, SSO) demonstrate the highest success rates.
   - Orbit "SO" has a zero success rate.
   - In LEO orbit, success appears correlated with the number of flights.
   - No discernible relationship between flight number and success in GTO orbit.
   - Positive landing rates higher for Polar, LEO, and ISS with heavy payloads.
   - Line plot indicates a consistent increase in success rates from 2013 to 2020.

2. **Exploratory Data Analysis Results from SQL:**
   - There are four distinct launch sites: CCAFS LC-40, KSC LC-39A, VAFB SLC-4E, CCAFS SLC-40.
   - The total sum of Payload mass for NASA (CRS) is 45596.
   - The average Payload mass for booster version "F9 v1.1" is 2928.4
   - The first successful ground landing was on date: 2015-12-22
   - F9 FT B1022, F9 FT B1026, F9 FT B1021.2, and F9 FT B1031.2 are the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
   - There were 1 failed mission outcome, whereas 100 successful mission outcomes.
   - Following booster versions had maximum payload mass of 15600 kgs: 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 B5B1051.6 , F9 B5 B1060.3 , F9 B5 B1049.7
   - Top three landing outcomes were No Attempts (10), Success (drone ship) (5), Failure (drone ship) (5) between the date 2010-06-04 and 2017-03-20, in descending order.

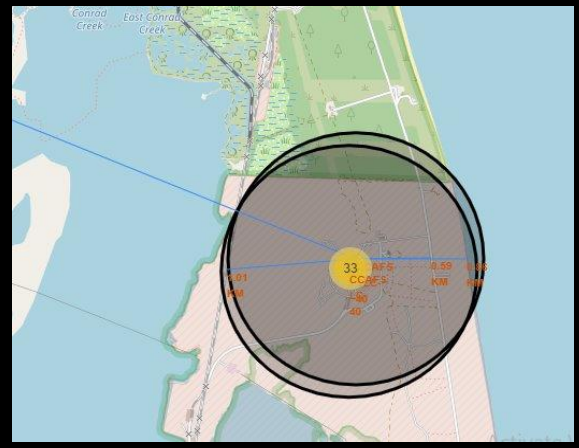
3. **Predictive analysis results:**
   - The predictive analysis results reveal that the Decision Tree model outperformed Logistic Regression, SVM, and KNN, achieving an accuracy of 94%.

4. Interactive analytics demo in screenshots

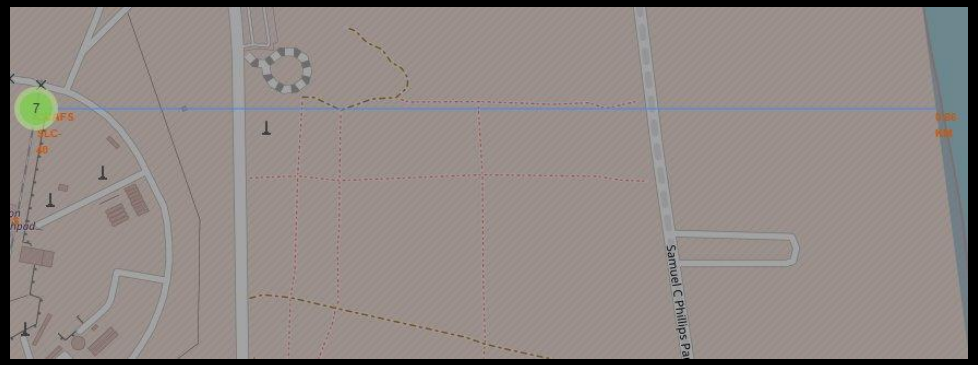


- All launch sites are in very close proximity to the coast.
- Launch sites CCAFS LC-40, CCAFS SLC-40, KSC LC-39A are very close to each other, whereas,, site VAFB SLC-4E is completely at the opposite direction

From the color-labeled markers in marker clusters, it is easy to identify that KSC LC-39A launch sites has relatively highest success rate.



Launch sites are strategically located in proximity t
railways, highways, and coastlines while maintainin
a safe distance from urban areas.



The nearest coastline is 0.86 km away, therefore, we can say that the Launch site is at a close proximity to the coastline.

Conclusion

1. **Launch Success Factors:** The analysis indicates that higher flight numbers positively correlate with successful first-stage landings. However, a heavier payload decreases the likelihood of a successful return.

2. **Launch Site Influence:** The majority of flights occur at CCAFS LC-40, boasting the highest success rate among the launch sites. Success rates tend to increase with the number of flights from a specific site.

3. **Orbit Success Rates:** Certain orbits, such as ES-L1, GEO, HEO, and SSO, demonstrate higher success rates, while the SO orbit shows no successes. The relationship between success and flight numbers varies in different orbits.

4. **Payload Impact:** Successful landings or positive landing rates are more prominent for Polar, LEO, and ISS with heavy payloads. However, distinguishing success in GTO orbits is challenging due to a mix of positive and negative landing outcomes.

5. **Temporal Trends:** The line plot shows a consistent increase in success rates from 2013 to 2020, suggesting overall improvement in mission outcomes over the years.

6. **Distinct Launch Sites:** A DISTINCT query highlights four launch sites: CCAFS LC-40, VAFB SLC-4E, KSC LC-39A, and CCAFS SLC-40.

7. **Filtering and Display:** Specific SQL queries enable filtering and display of relevant data, such as limiting records to launch sites starting with "CCA" or summarizing payload masses for NASA (CRS) missions.

8. **Statistical Queries:** Utilizing SQL, various statistical queries were executed, including sum and average calculations for payload masses, identifying launch sites with specific outcomes, and determining the earliest successful ground pad landing date.

9. **Model Performance:** In predictive analysis, the Decision Tree model emerged as the most accurate among the tested models, achieving a remarkable 94% accuracy.

10. **Infrastructure Proximity:** Launch sites exhibit close proximity to railways, highways, and coastlines. Additionally, they maintain a certain distance from cities, aligning with safety and operational considerations.


IBM Developer
SKILLS NETWORK
Thank you