IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Batool Sherif Mohamed
11/6/2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis results
  - Interactive maps and dashboard
  - Predictive results

Introduction

SpaceX is a revolutionary company who has disrupt the space industry by offering a rocket launches specifically Falcon 9 as low as 62 million dollars; while other providers cost upward of 165 million dollar each. Most of this saving thanks to SpaceX astounding idea to reuse the first stage of the launch by re-land the rocket to be used on the next mission. Repeating this process will make the price down even further. As a data scientist of a startup rivaling SpaceX, the goal of this project is to create the machine learning pipeline to predict the landing outcome of the first stage in the future. This project is crucial in identifying the right price to bid against SpaceX for a rocket launch.

The problems included:

- Identifying all factors that influence the landing outcome.
- The relationship between each variables and how it is affecting the outcome.
- The best condition needed to increase the probability of successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and web scrapping from Wikipedia
  - Perform data wrangling
  - Data was processed using one-hot encoding for categorical features
- 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

Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevant questions and evaluate outcomes. As mentioned, the dataset was collected by REST API and Web Scrapping from Wikipedia

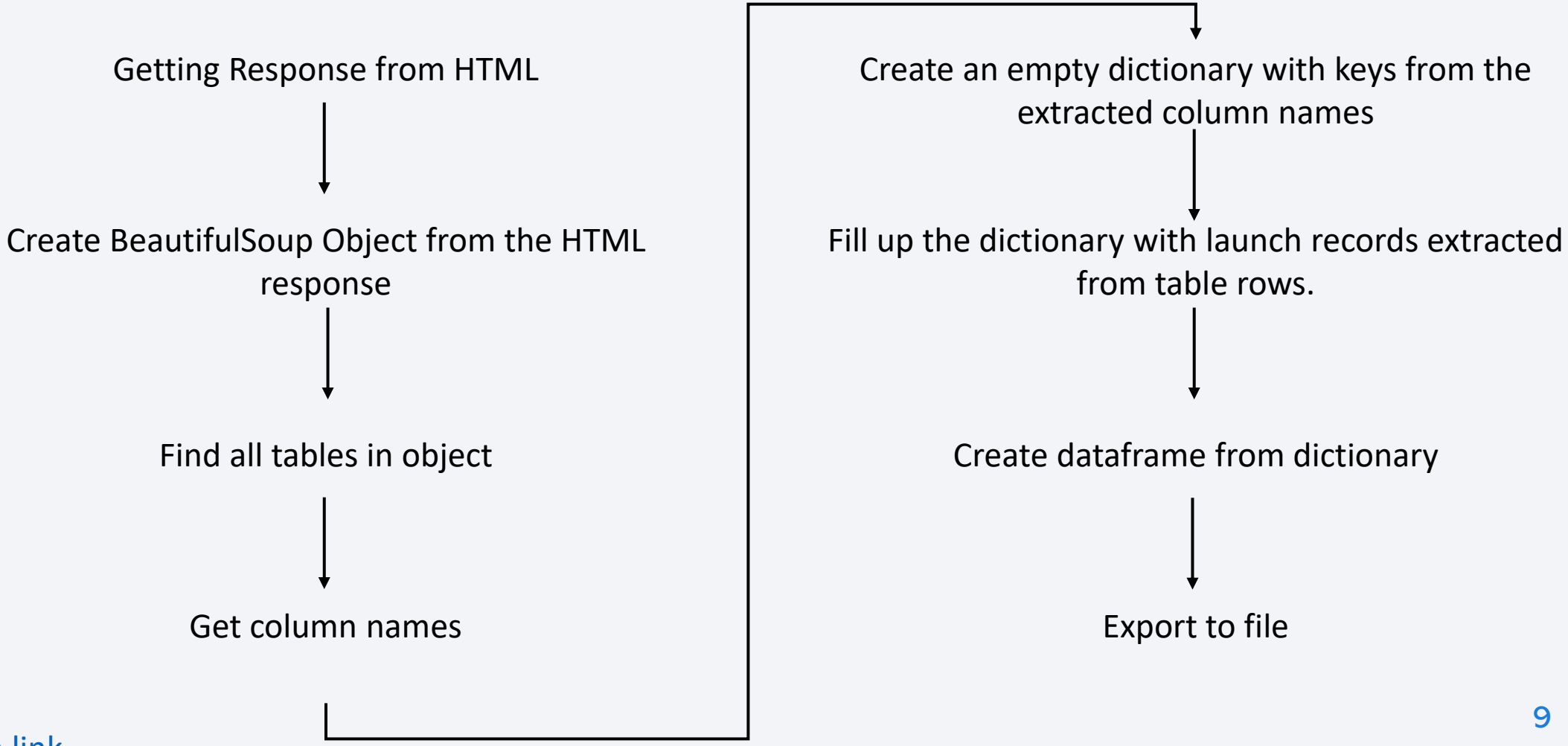
For REST API, its started by using the get request. Then, we decoded the response content as Json and turn it into a pandas dataframe using json_normalize(). We then cleaned the data, checked for missing values and fill with whatever needed.

For web scrapping, we will use the BeautifulSoup to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for further analysis

Data Collection – SpaceX API

Github link

Data Collection - Scraping



Github link

Data Wrangling

Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).

In the dataset, there are several cases where the booster did not land successfully.

- True Ocean, True RTLS, True ASDS means the mission has been successful.
- False Ocean, False RTLS, False ASDS means the mission was a failure.

We need to transform string variables into categorical variables where 1 means the mission has been successful and 0 means the mission was a failure.

Calculate launches number for each site

↓

Calculate the number and occurrence of each orbit

↓

Calculate number and occurrence of mission outcome per orbit type

↓

Create landing outcome label from Outcome column

↓

Export to file

[Github link]

EDA with Data Visualization

Scatter Graphs

Show relationship between variables.

- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Orbit vs. Flight Number
- Payload vs. Orbit Type

Bar graph

Show the relationship between numeric and categoric variables.

- Success rate vs. Orbit

Line Graph

Show trends or pattern of the attribute over time.

- Success rate vs. Year

Github link

EDA with SQL

Using SQL, we had performed many queries to get better understanding of the dataset, Ex:

- Displaying the names of the unique launch sites in the space mission.
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS).
- Display average payload mass carried by booster version F9 v1.1.
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
- List the total number of successful and failure mission outcomes.
- Listing the names of the booster_versions which have carried the maximum payload mass.- Listing the failed landing_outcomes in drone ship, their booster versions, and launch sites names for in year 2015.
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Github link

Build an Interactive Map with Folium

To visualize the launch data into an interactive map. We took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.

We then assigned the dataframe launch_outcomes (failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().

We then used the Haversine's formula to calculated the distance of the launch sites to

- various landmark to find answer to the questions of:
- How close the launch sites with railways, highways and coastlines?
- How close the launch sites with nearby cities?

Github link

Build a Dashboard with Plotly Dash

Dashboard has dropdown, pie chart, range slider and scatter plot .

- Dropdown→ Dropdown allows a user to choose the launch site or all launch sites.
- Pie chart → shows the total success and the total failure for the launch site chosen with the dropdown component.
- Range slider→ allows a user to select a payload mass in a fixed range.
- Scatter chart→ shows the relationship between two variables, in particular Success vs Payload Mass.

Github Link.

Predictive Analysis (Classification)

Data preparation

- Load dataset
- Normalize data
- Split data into training and test sets.

Model preparation

- Selection of machine learning algorithms
- Set parameters for each algorithm to GridSearchCV
- Training GridSearchModel models with training dataset

Model evaluation

- Get best hyperparameters for each type of model
- Compute accuracy for each model with test dataset
- Plot Confusion Matrix

Model comparison

- Comparison of models according to their accuracy
- The model with the best accuracy will be chosen (see Notebook for result)

[Github Link.]

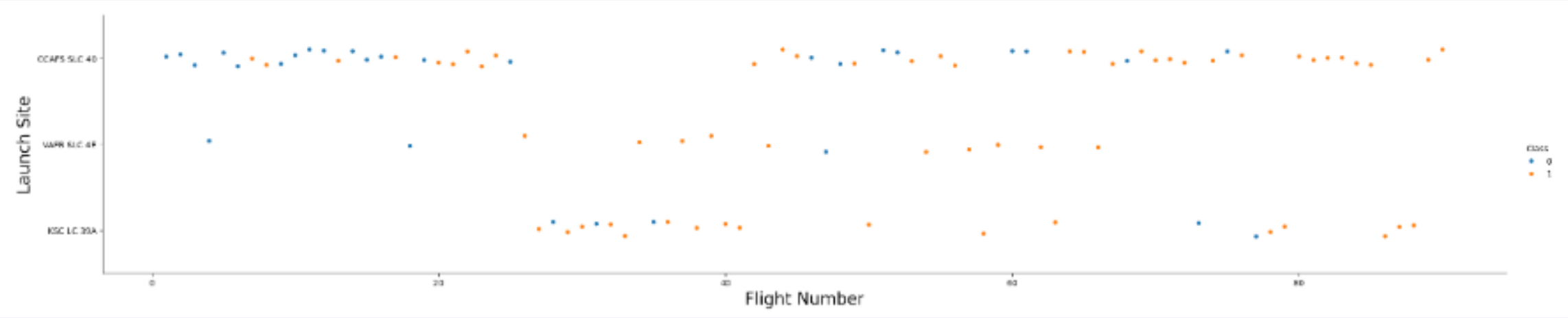
Results

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

Section 2

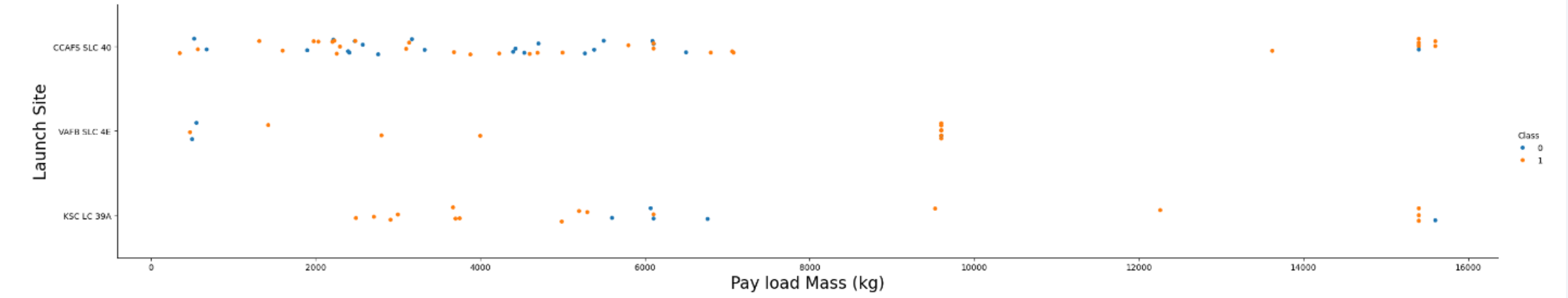
Insights drawn from EDA

Flight Number vs. Launch Site



•**CCAFS SLC 40**: This launch site is used consistently across a wide range of flight numbers, from the earliest flights to the more recent ones.
•**KSC LC 39A**: This site starts being used later, around flight number 20, and continues to be used for many subsequent flights.
•**VAFB SLC 4E**: This site is used intermittently across the flight numbers, with no clear pattern of increasing or decreasing use over time.

Payload vs. Launch Site



Depending on the launch site, a heavier payload may be a consideration for a successful landing. On the other hand, a too heavy payload can make a landing fail

Success Rate vs. Orbit Type

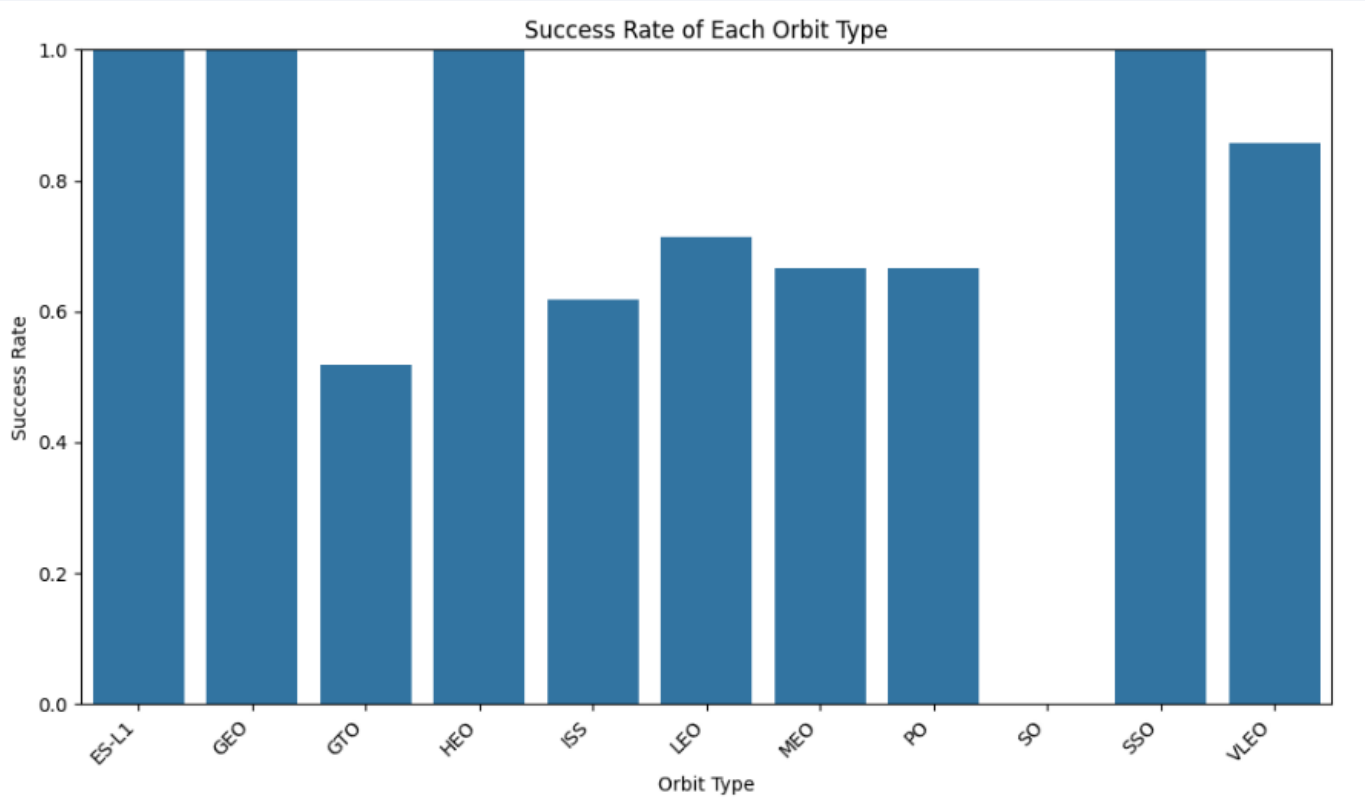
Orbit Types with 100% Success Rate:
- ES-L1 GEO SSO

Orbit Types with High Success Rate:
- VLEO

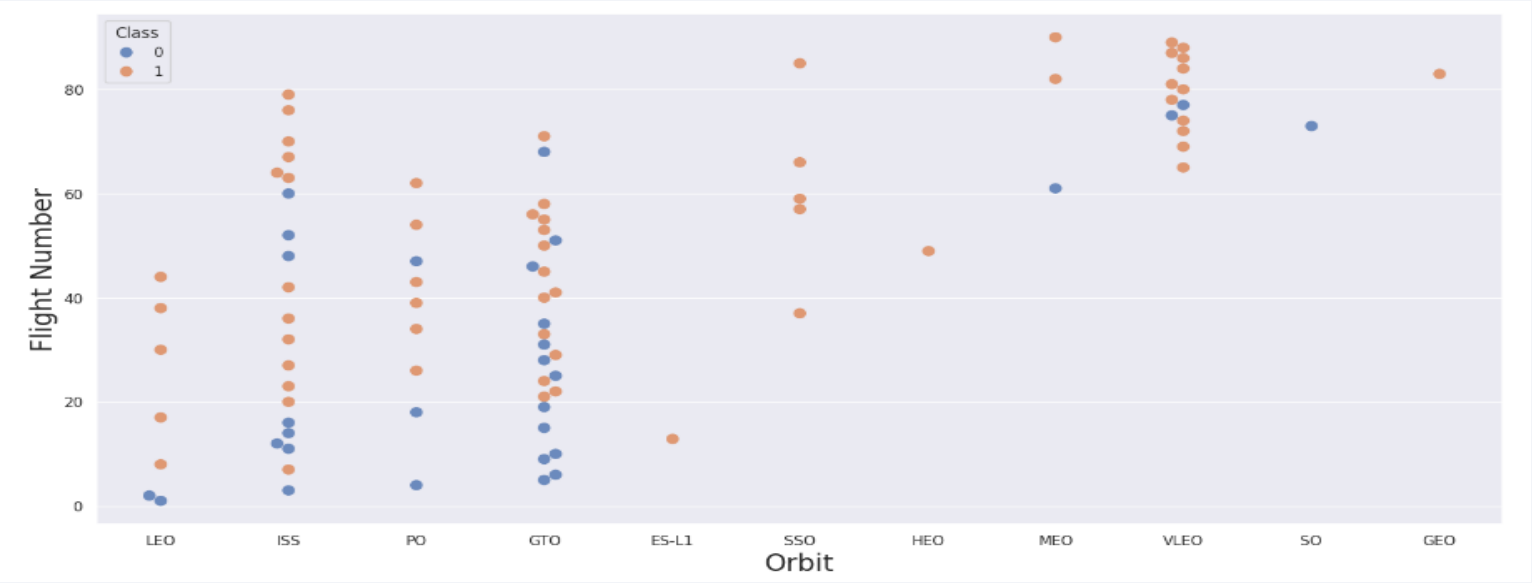
Orbit Types with Moderate Success Rate:
- HEO, LEO, MEO, PO

Orbit Types with Lower Success Rate:
- ISS GTO

Orbit Type with 0% Success Rate:
- SO

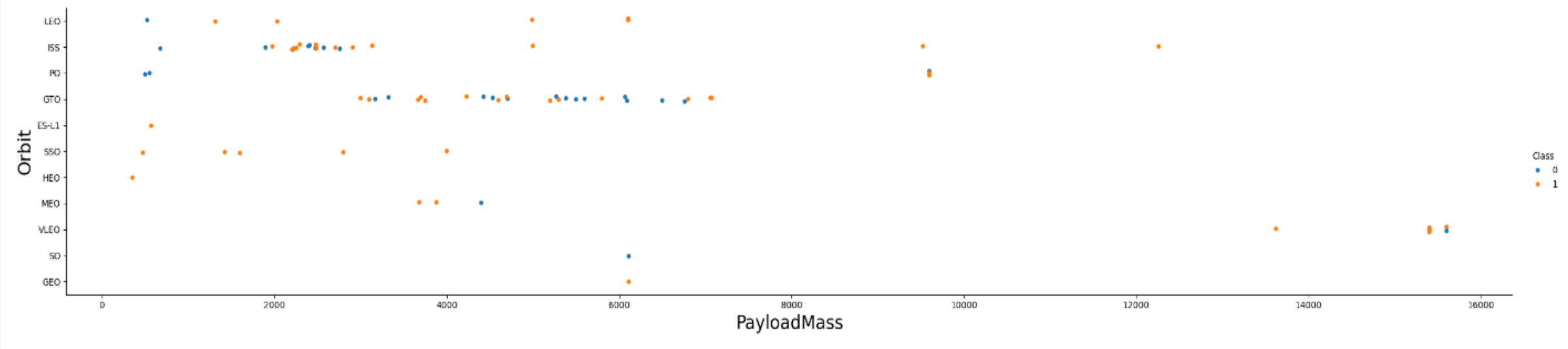


Flight Number vs. Orbit Type



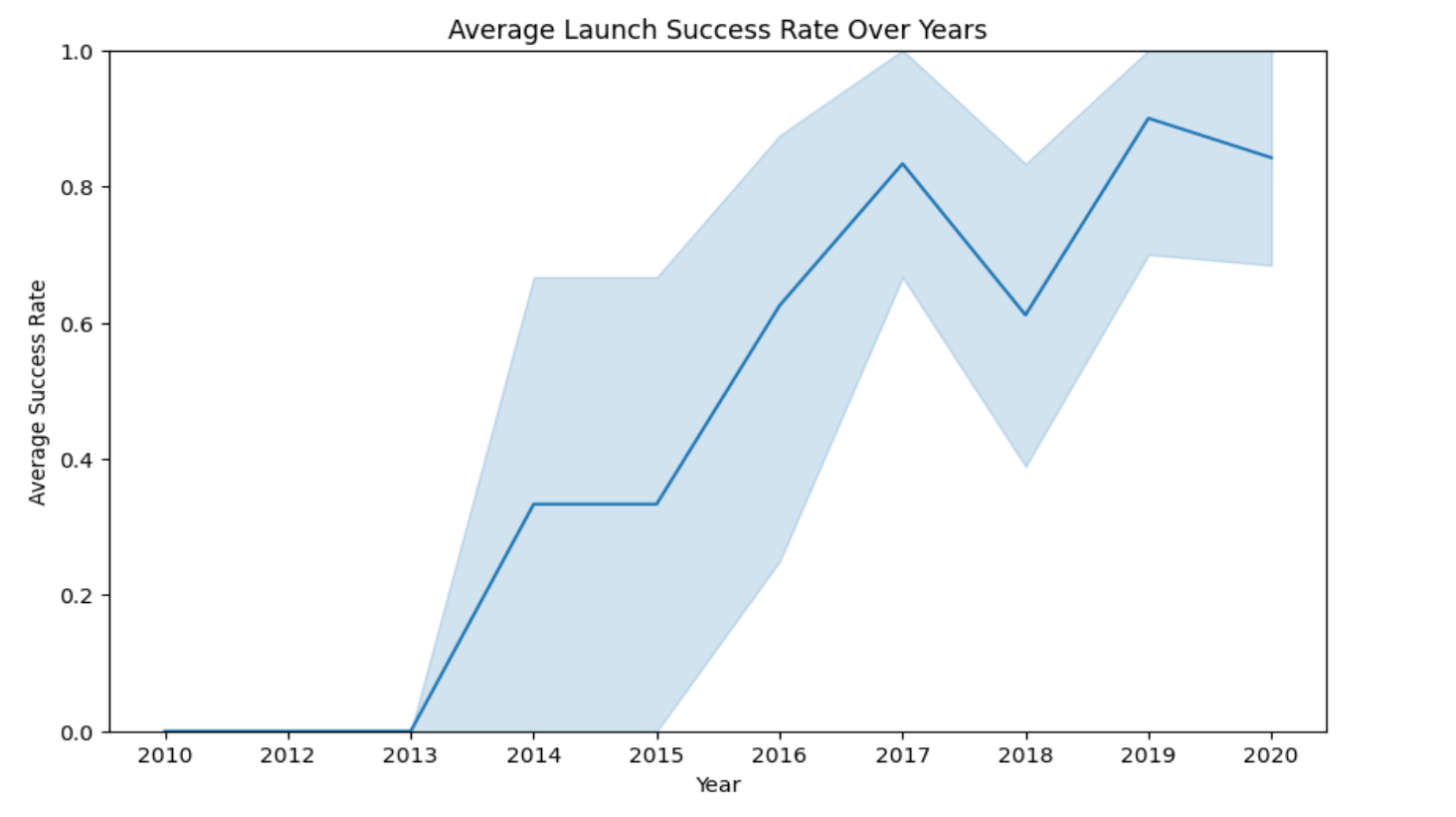
LEO and ISS: Mix of successes and failures across flight numbers
PO and GTO: Both successes and failures, with more successes in later flights for PO.
ES-L1, SSO, HEO, GEO: Only successes observed.
MEO: Mix of successes and failures.
VLEO: Higher concentration of successes, especially in higher flight numbers.
SO: Only failures observed.

Payload vs. Orbit Type



Heavier payload has positive impact on LEO, ISS and P0 orbit, it has negative impact on MEO and VLEO orbit.
GTO orbit seem to depict no relation between the attributes.
SO, GEO and HEO orbit need more dataset to see any pattern or trend

Launch Success Yearly Trend



There is an increase in the Space X Rocket success rate starting from year 2013.

All Launch Site Names

The key word DISTINCT to show only unique launch sites from the SpaceX data.

```
# Query the unique launch sites
result = %sql SELECT DISTINCT Launch_Site FROM SPACEXTABLE
print(result)
```

```
 * 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'

```
# Query 5 records where launch sites begin with the string 'CCA'
result = %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' LIMIT 5

import pandas as pd
from IPython.display import display
# Convert the result to a DataFrame
result_df = result.DataFrame()

# Display the DataFrame
display(result_df)
```

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

	Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

The WHERE clause followed by LIKE clause filters launch sites that contain the substring CCA. LIMIT 5 shows 5 records from filtering.

Total Payload Mass

```
result = %sql SELECT SUM(PAYLOAD_MASS__KG_) as Total_Payload_Mass FROM SPACEXTABLE WHERE Customer LIKE '%NASA (CRS)%'
print(result)
```

```
 * sqlite:///my_data1.db
Done.
+--------------------+
| Total_Payload_Mass |
+--------------------+
|       48213        |
+--------------------+
```

- **SELECT SUM(PAYLOAD_MASS__KG_) as Total_Payload_Mass :** This part calculates the sum of the (PAYLOAD_MASS__KG_) column and labels the result as Total_Payload_Mass.

- **WHERE Customer LIKE '%NASA (CRS)%'** : This filters the results to only include rows where the Customer column contains the string "NASA (CRS)".

Average Payload Mass by F9 v1.1

```
result_avg_payload = %sql SELECT AVG(PAYLOAD_MASS__KG_) as Average_Payload_Mass FROM SPACEXTABLE WHERE Booster_Version = 'F9 v1.1'
print(result_avg_payload)
```

```
 * sqlite:///my_data1.db
Done.
+----------------------+
| Average_Payload_Mass |
+----------------------+
|        2928.4        |
+----------------------+
```

This query returns the average of all payload masses where the booster version contains the substring F9 v1.1.

First Successful Ground Landing Date

```
result_first_success_landing = %sql SELECT MIN(Date) as First_Successful_Landing_Date FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)'
print(result_first_success_landing)
```

```
 * sqlite:///my_data1.db
Done.
+-------------------------------+
| First_Successful_Landing_Date |
+-------------------------------+
|           2015-12-22          |
+-------------------------------+
```

SELECT MIN(Date) as First_Successful_Landing_Date: This part of the query selects the minimum (earliest) date from the Date column and labels the result as First_Successful_Landing_Date.

WHERE Landing_Outcome = 'Success (ground pad)': This filters the results to only include rows where the Landing_Outcome is 'Success (ground pad)'.

Successful Drone Ship Landing with Payload between 4000 and 6000

```
result_boosters_drone_ship = %sql SELECT Booster_Version FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000
print(result_boosters_drone_ship)
```

```
 * sqlite:///my_data1.db
Done.
+-----------------+
| Booster_Version |
+-----------------+
|   F9 FT B1022   |
|   F9 FT B1026   |
|  F9 FT  B1021.2 |
|  F9 FT  B1031.2 |
+-----------------+
```

We used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

With the first SELECT, we show the subqueries that return results.
The first subquery counts the successful mission.
The second subquery counts the unsuccessful mission.
The WHERE clause followed by LIKE clause filters mission outcome.
The COUNT function counts records filtered.

```
# Query for successful missions
successful_mission_query = """
SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission"
FROM SPACEXTABLE
WHERE MISSION_OUTCOME LIKE 'Success%';
"""

# Query for failure missions
failure_mission_query = """
SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission"
FROM SPACEXTABLE
WHERE MISSION_OUTCOME LIKE 'Failure%';
"""

# Execute the queries and fetch the results
successful_mission_result = %sql $successful_mission_query
failure_mission_result = %sql $failure_mission_query

# Display the results together in one table
results = successful_mission_result.DataFrame().join(failure_mission_result.DataFrame())
results
```

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

	Successful Mission	Failure Mission
0	100	1

Boosters Carried Maximum Payload

```
result_max_payload_booster = %sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE)
print(result_max_payload_booster)
```

```
 * sqlite:///my_data1.db
Done.
+-----------------+
| Booster_Version |
+-----------------+
|  F9 B5 B1048.4  |
|  F9 B5 B1049.4  |
|  F9 B5 B1051.3  |
|  F9 B5 B1056.4  |
|  F9 B5 B1048.5  |
|  F9 B5 B1051.4  |
|  F9 B5 B1049.5  |
|  F9 B5 B1060.2  |
|  F9 B5 B1058.3  |
|  F9 B5 B1051.6  |
|  F9 B5 B1060.3  |
|  F9 B5 B1049.7  |
+-----------------+
```

We used a subquery to filter data by returning only the heaviest payload mass with MAX function.

2015 Launch Records

```
# Query the records for the specified conditions
result = %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'

# Display the result as a DataFrame
result_df = result.DataFrame()
print(result_df)
```

```
 * sqlite:///my_data1.db
Done.
  Month       Landing_Outcome Booster_Version  Launch_Site
0    01  Failure (drone ship)   F9 v1.1 B1012  CCAFS LC-40
1    04  Failure (drone ship)   F9 v1.1 B1015  CCAFS LC-40
```

This query returns month, booster version, launch site where landing was unsuccessful and landing date took place in 2015.
Substr function process date in order to take month or year.

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

```
# Query to rank the count of landing outcomes between the specified dates
result_landing_outcomes = %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

# Display the result as a DataFrame
result_landing_outcomes_df = result_landing_outcomes.DataFrame()
print(result_landing_outcomes_df)
```

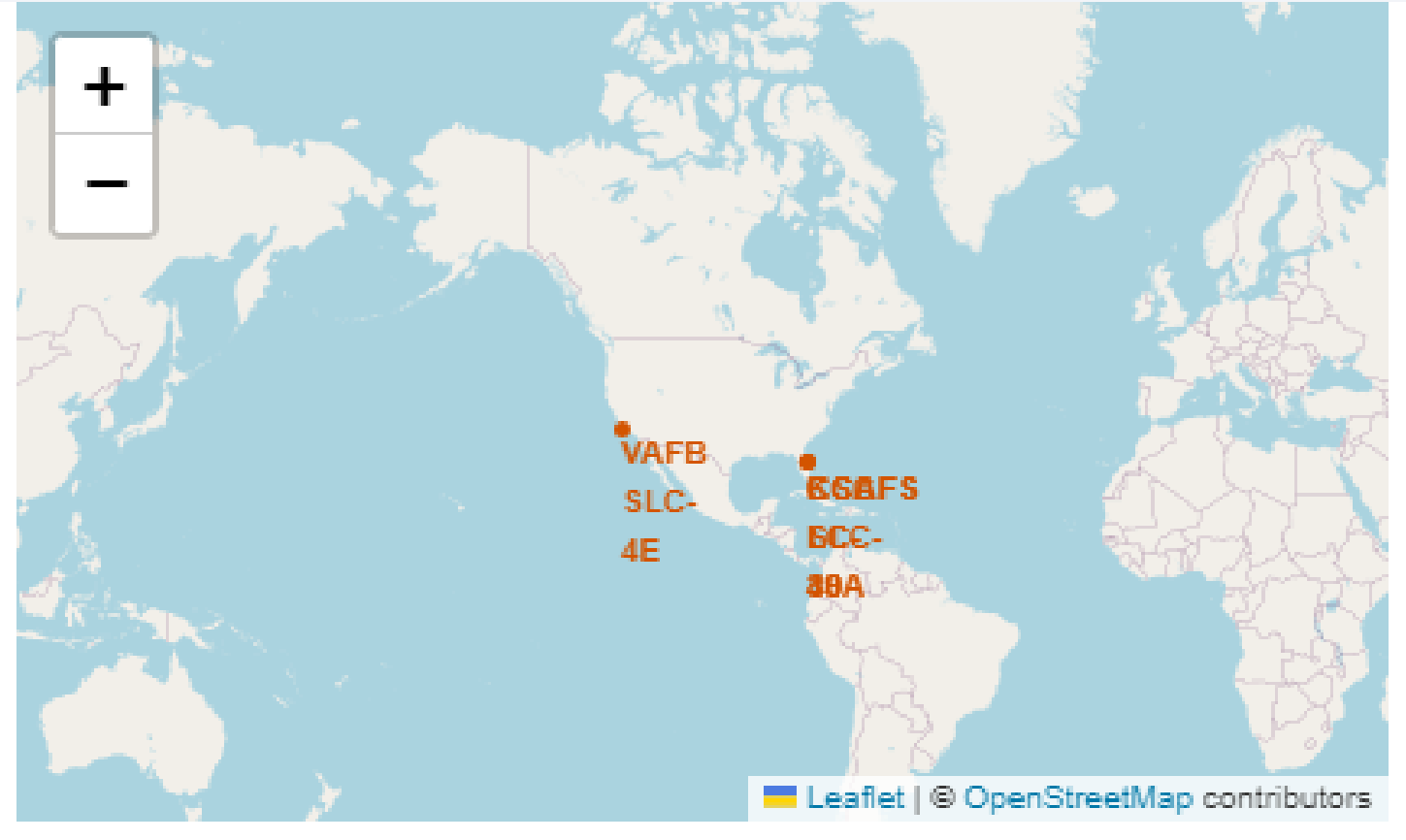
```
 * sqlite:///my_data1.db
Done.
          Landing_Outcome  Count
0              No attempt     10
1   Success (drone ship)      5
2   Failure (drone ship)      5
3   Success (ground pad)      3
4     Controlled (ocean)      3
5   Uncontrolled (ocean)      2
6    Failure (parachute)      2
7  Precluded (drone ship)      1
```

This query returns landing outcomes and their count where mission was successful and date is between 04/06/2010 and 20/03/2017. The GROUP BY clause groups results by landing outcome and ORDER BY COUNT DESC shows results in decreasing order

Section 3

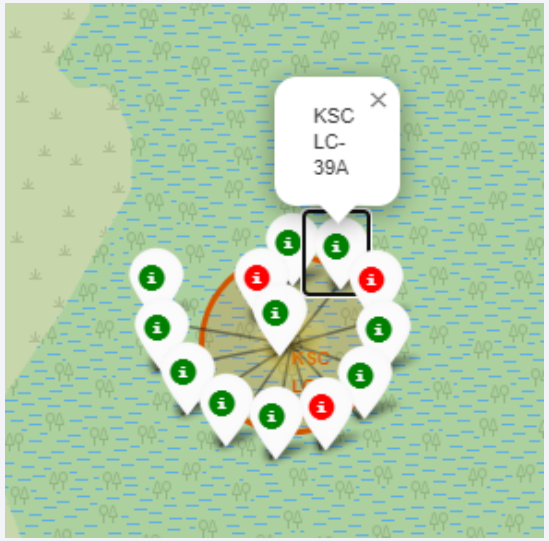
Launch Sites Proximities Analysis

Location of all the Launch Sites



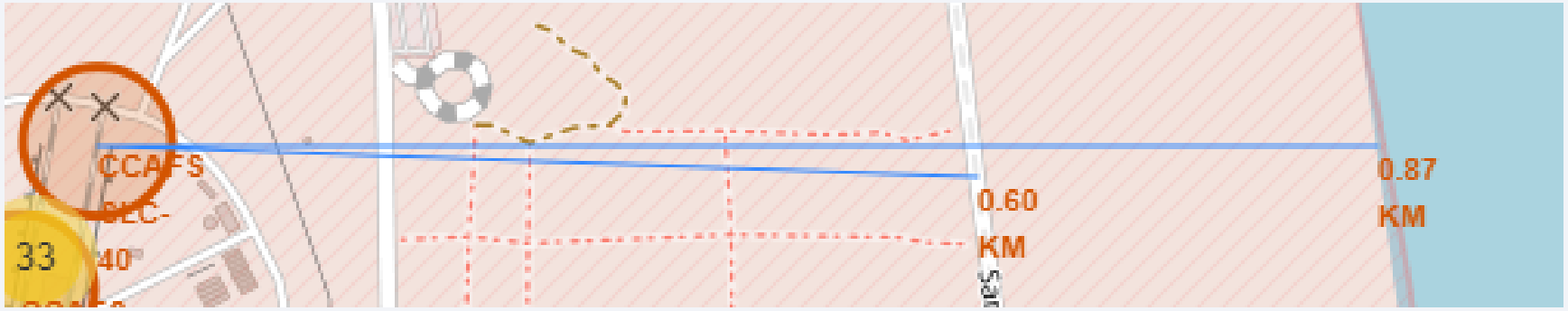
Space X launch sites are located on the coast of the United States

Folium map – Color Labeled Markers

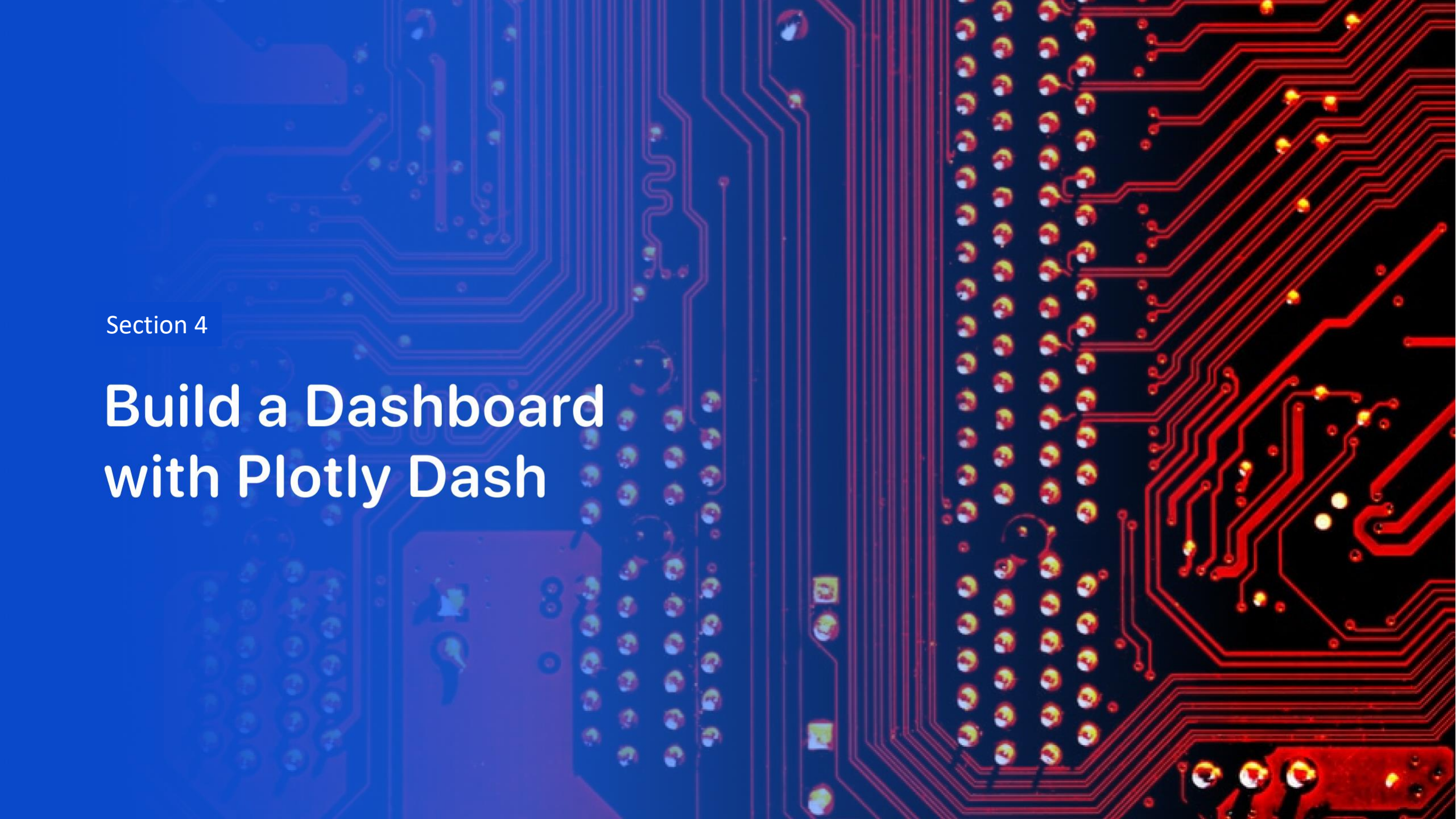


Green marker represents successful launches.
Red marker represents unsuccessful launches.
We note that KSC LC-39A has a higher launch success rate.

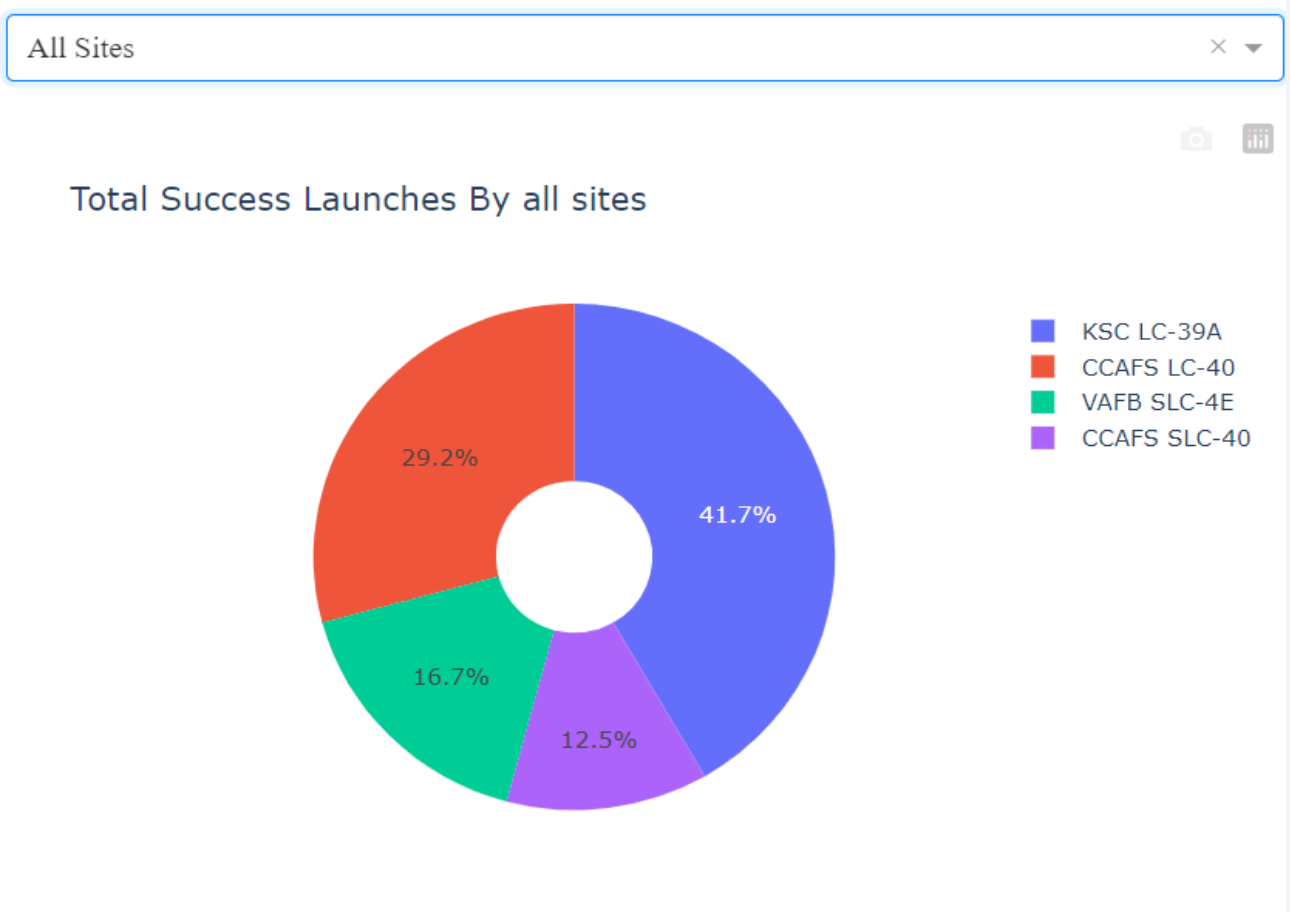
Folium Map – Distances between CCAFS SLC-40 and its proximities



Is CCAFS SLC-40 in close proximity to railways ? Yes
Is CCAFS SLC-40 in close proximity to highways ? Yes
Is CCAFS SLC-40 in close proximity to coastline ? Yes
Do CCAFS SLC-40 keeps certain distance away from cities ? No

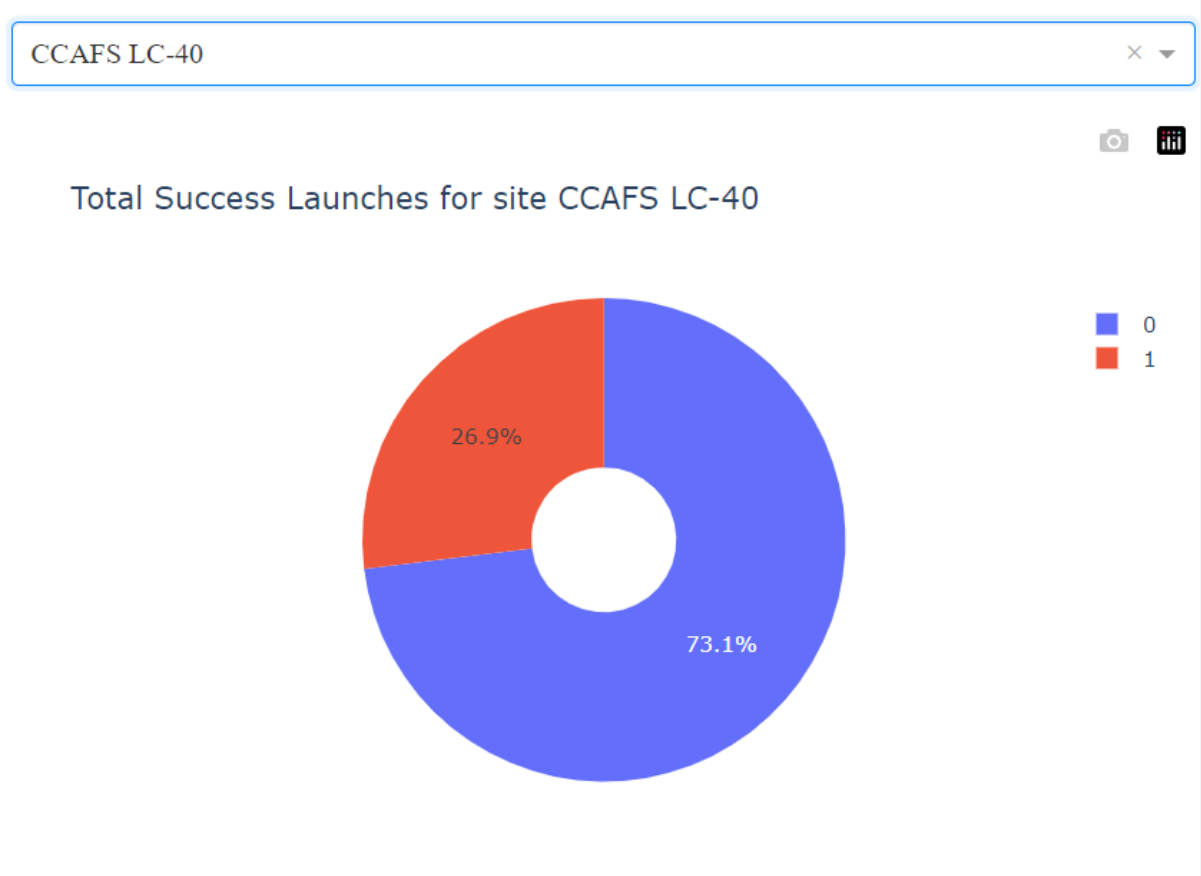

Section 4
Build a Dashboard with Plotly Dash

Dashboard – Total success by Site



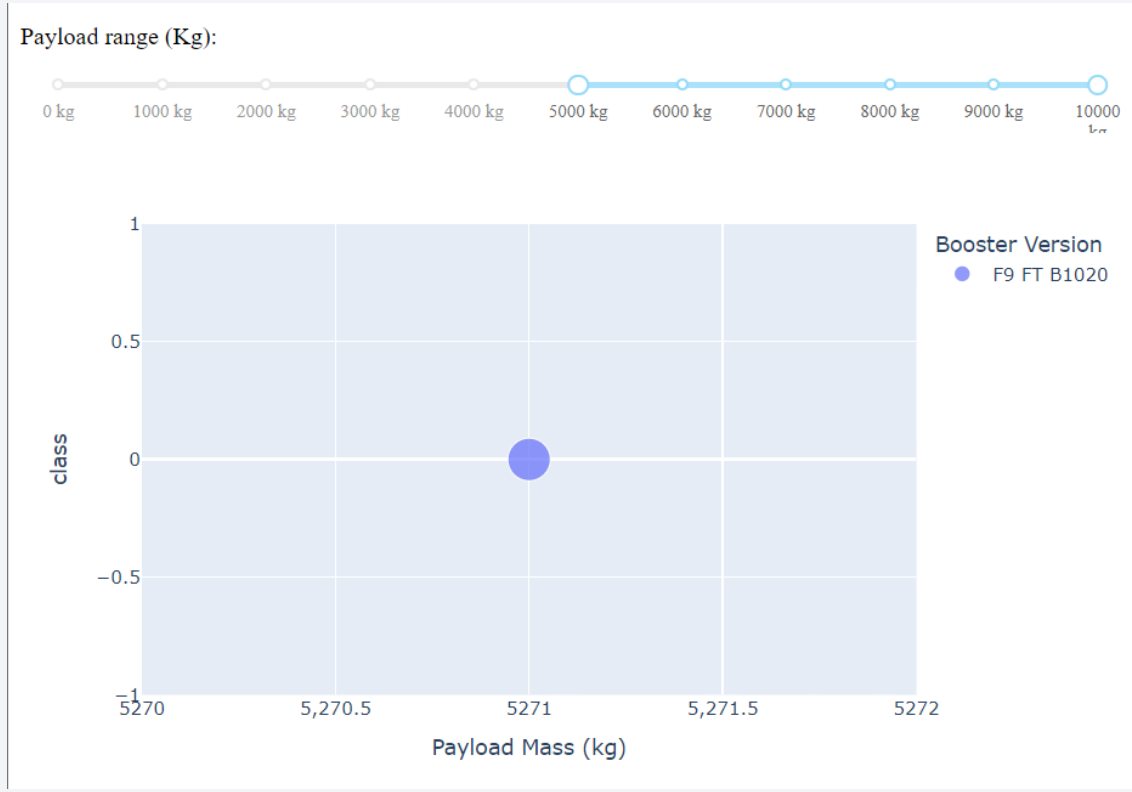
KSC LC-39A had the most successful launches from all sites.

Dashboard – Total success launches for Site CCAFS LC-40



CCAFS LC-40 has 73.1% success rate and 26.7% failure rate.

<Dashboard Screenshot 3>



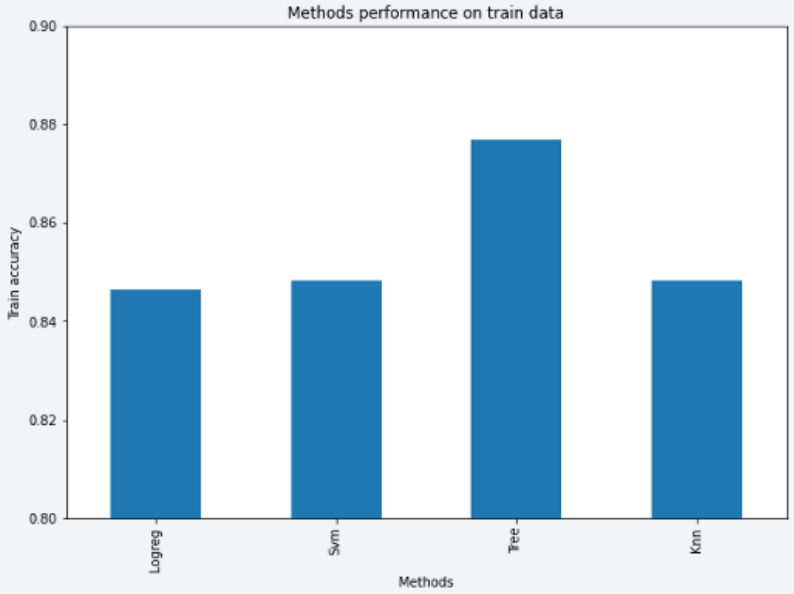
The success rate for low weighted payload is higher than heavy weighted payload

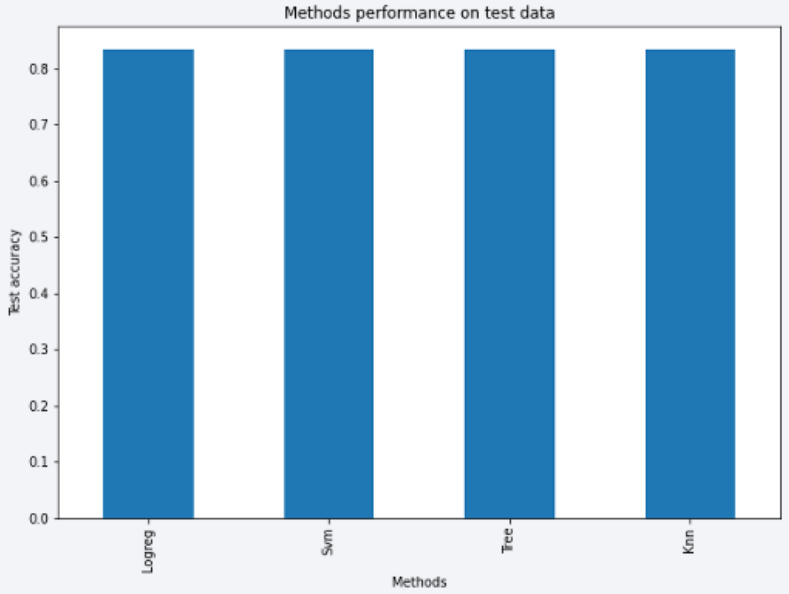
Section 5

Predictive Analysis (Classification)

Classification Accuracy

	Accuracy Train	Accuracy Test
Tree	0.876786	0.833333
Knn	0.848214	0.833333
Svm	0.848214	0.833333
Logreg	0.846429	0.833333





In our accuracy test, all methods demonstrated comparable performance. To make a definitive choice between them, additional test data would be beneficial. However, if an immediate decision is required, we would opt for the decision tree method.

Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier

		Actual values	
		1	0
Predicted values	1	TP	FP
	0	FN	TN

Conclusions

- The Tree Classifier Algorithm is the best machine learning approach for this dataset.

- Since 2013, the success rate for SpaceX launches has increased, with a direct proportional relationship to the passing years up to 2020, suggesting a trend towards perfecting launches in the future.

- The SSO orbit has the highest success rate, at 100%, with more than one occurrence.

- The success of a mission can be attributed to factors such as the launch site, the orbit, and especially the number of previous launches, indicating a gain in knowledge from launch failures to successes.

- The orbits with the best success rates are GEO, HEO, SSO, and ES-L1.

- Depending on the orbit, payload mass can be a significant factor for mission success, with lighter payloads generally performing better than heavier ones.

Thank you!