



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Alberto M. Palacio B.
March 2024



Outline

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

Executive Summary

Summary of methodologies

- Data collection with SpaceX's REST API was successful. Data from 90 Falcon 9 booster launches, from 2010 to 2020 was collected.
- In data wrangling a success rate of 66% was identified for all time Falcon 9 booster launches.
- In Exploratory data analysis (EDA) an increasing success rate was identified.

Summary of all results

Introduction

Project background and context

- Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch.

Problems you want to find answers

- What influences if the rocket will land successfully?
- What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.
- Predict if the first stage will land given the data from the preceding launches.

Section 1

Methodology

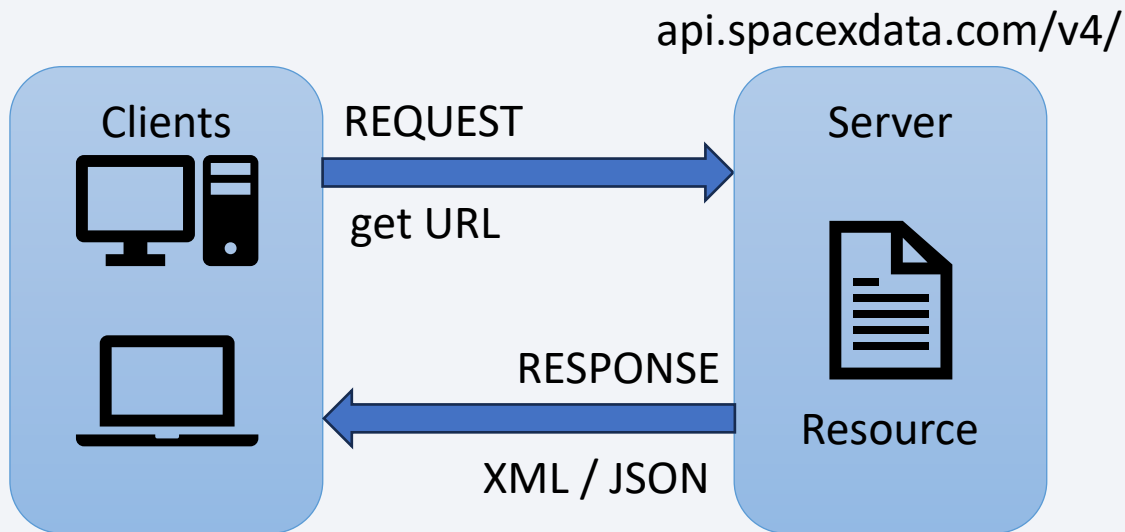
Methodology

Executive Summary

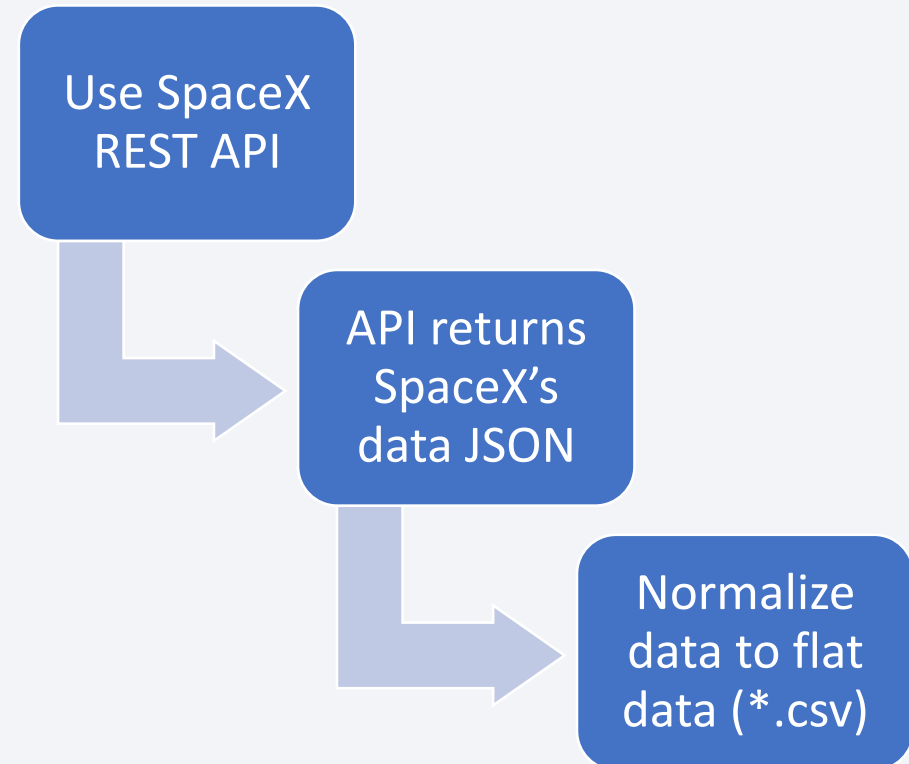
- Data collection methodology:
 - Data collection from SpaceX's REST API.
 - Data collection with web scrapping.
- Data wrangling methodology:
 - Data was processed using numpy and Pandas Python libraries.
- Performed exploratory data analysis (EDA) using SQL and Python visualization tools
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
 - How to build, tune, evaluate classification models

Data Collection

- Data was collected using SpaceX REST API:



- Data collection flowchart using SpaceX REST API:



Data Collection – SpaceX API

Data collection with SpaceX REST calls:

GitHub URL:

https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_collection/spacex-data-collection-api.ipynb

1. Getting response from API:

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

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

Check the content of the response

In [8]: print(response.content)

b'{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-los-t-launch.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":"2006-03-17T00:00:00.000Z","static_fire_date_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":3
```

2. Converting response to JSON and normalize it.

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

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

Using the dataframe data print the first 5 rows

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

Out[12]:

	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 – SpaceX API

Data collection with SpaceX REST calls:

GitHub URL:

https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-prediction/blob/main/data_collection/spacex-data-collection-api.ipynb

3. Create new dataframe removing unnecessary fields and data.

Finally lets construct our dataset using the data we have obtained. We combine the columns into a dictionary.

```
In [21]: launch_dict = {'FlightNumber': list(data['flight_number']),
                        'Date': list(data['date']),
                        'BoosterVersion': BoosterVersion,
                        'PayloadMass': PayloadMass,
                        'Orbit': Orbit,
                        'LaunchSite': LaunchSite,
                        'Outcome': Outcome,
                        'Flights': Flights,
                        'GridFins': GridFins,
                        'Reused': Reused,
                        'Legs': Legs,
                        'LandingPad': LandingPad,
                        'Block': Block,
                        'ReusedCount': ReusedCount,
                        'Serial': Serial,
                        'Longitude': Longitude,
                        'Latitude': Latitude}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

```
In [22]: # Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

Show the summary of the dataframe

```
In [23]: # Show the head of the dataframe
df.head(15)
```

```
Out[23]:
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	Na
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	Na
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	Na
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	Na
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1

Data Collection – SpaceX API

Data collection with SpaceX REST calls:

GitHub URL:

https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_collection/spacex-data-collection-api.ipynb

4. Filter the dataframe to include only Falcon 9 launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
In [24]: # Hint data['BoosterVersion']!= 'Falcon 1'
data_falcon9 = df[df['BoosterVersion']!='Falcon 1']
data_falcon9.head(15)
```

```
Out[24]:
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	La
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	
5	8	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	
6	10	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	
7	11	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	
8	12	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	
9	13	2014-01-06	Falcon 9	3325.0	GTO	CCSFS SLC 40	None None	1	False	False	False	

4. Export dataframe to flat .csv file

```
In [32]: data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection – Web Scraping

- Performed web scraping process using BeautifulSoup Python library.
- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_collection/spacex-data-collection-webscraping.ipynb

1. Request the Falcon9 Launch Wiki page from its URL

```
In [5]: # 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 [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, "html.parser")

In [7]: print(soup.prettify())

<!DOCTYPE html>
<html class="client-nojs vector-feature-language-in-header-enabled vector-feature-language-in-main-page-header-disabled vector-feature-sticky-header-disabled vector-feature-page-tools-pinned-disabled vector-feature-toc-pinned-clientpref-1 vector-feature-main-menu-pinned-disabled vector-feature-limited-width-clientpref-1 vector-feature-limited-width-content-enabled vector-feature-custom-font-size-clientpref-0 vector-feature-client-preferences-disabled vector-feature-client-preferences-pinned-disabled vector-feature-night-mode-disabled skin-night-mode-clientpref-0 vector-toc-available" dir="ltr" lang="en">
<head>
```

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

```
In [9]: # 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.findAll('table')

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

In [10]: # Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)

<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<br/>Booster</a> <sup class="reference" id="cite_ref-booster_11-0"><a href="#cite_note-booster-11">[b]</a></sup>
</th>
<th scope="col">Launch site
```

Data Collection – Web Scraping

- Performed web scraping process using BeautifulSoup Python library.
- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_collection/spacex-data-collection-webscraping.ipynb

3. Create a data frame by parsing the launch HTML tables

```
In [13]: launch_dict= dict.fromkeys(column_names)

# Remove an irrelevant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty List
launch_dict['Flight No.']= []
launch_dict['Launch site']= []
launch_dict['Payload']= []
launch_dict['Payload mass']= []
launch_dict['Orbit']= []
launch_dict['Customer']= []
launch_dict['Launch outcome']= []

# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

4: Export to flat .csv file

```
In [18]: df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

- Data was processed using NumPy and Pandas Python libraries.
- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_wrangling/spacex-Data%20wrangling.ipynb

1. Import required libraries and data from data collection step.

Import Libraries and Define Auxiliary Functions

We will import the following libraries.

```
In [1]: # Pandas is a software library written for the Python programming language for data manipulation and analysis.  
import pandas as pd  
# NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices,  
import numpy as np
```

Data Analysis

Load Space X dataset, from last section.

```
In [2]: df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset-1/spacex-Data%20wrangling.csv")  
df.head(10)
```

```
Out[2]:
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	

Data Wrangling

- Data was processed using NumPy and Pandas Python libraries.
- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_wrangling/spacex-Data%20wrangling.ipynb

2. Identify and calculate the percentage of the missing values in each attribute.

```
In [3]: df.isnull().sum()/len(df)*100
```



```
Out[3]: FlightNumber      0.000000  
Date                    0.000000  
BoosterVersion          0.000000  
PayloadMass             0.000000  
Orbit                   0.000000  
LaunchSite              0.000000  
Outcome                 0.000000  
Flights                 0.000000  
GridFins                0.000000  
Reused                  0.000000  
Legs                    0.000000  
LandingPad              28.888889  
Block                   0.000000  
ReusedCount             0.000000  
Serial                  0.000000  
Longitude               0.000000  
Latitude                0.000000  
dtype: float64
```

Data Wrangling

- Data was processed using NumPy and Pandas Python libraries.
- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_wrangling/spacex-Data%20wrangling.ipynb

3. Identify which columns are numerical and categorical.

```
In [4]: df.dtypes
```



```
Out[4]: FlightNumber    int64  
Date                  object  
BoosterVersion         object  
PayloadMass           float64  
Orbit                  object  
LaunchSite             object  
Outcome               object  
Flights               int64  
GridFins              bool  
Reused                bool  
Legs                  bool  
LandingPad            object  
Block                 float64  
ReusedCount           int64  
Serial                object  
Longitude             float64  
Latitude              float64  
dtype: object
```

Data Wrangling

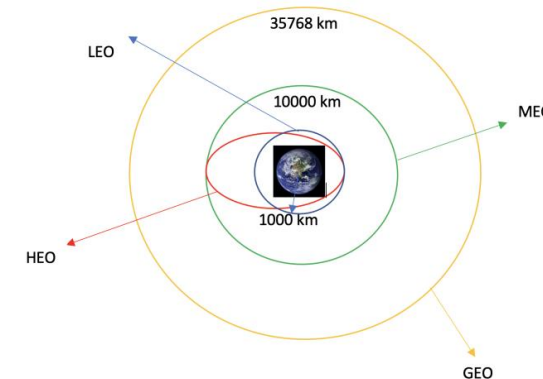
- Data was processed using NumPy and Pandas Python libraries.
- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_wrangling/spacex-Data%20wrangling.ipynb

4. Calculate the number of launches on each launching site.

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

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

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



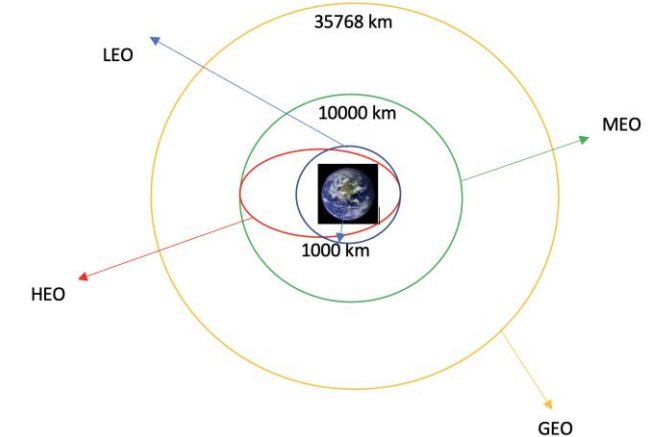
Data Wrangling

- Data was processed using NumPy and Pandas Python libraries.
- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_wrangling/spacex-Data%20wrangling.ipynb

5. Calculate the number and occurrence of each orbit.

```
In [8]: # Apply value_counts on Orbit column  
df['Orbit'].value_counts()
```

```
Out[8]: Orbit  
GTO      27  
ISS      21  
VLEO     14  
PO        9  
LEO        7  
SSO        5  
MEO        3  
ES-L1      1  
HEO         1  
SO          1  
GEO         1  
Name: count,
```



Data Wrangling

- Data was processed using NumPy and Pandas Python libraries.

- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_wrangling/spacex-Data%20wrangling.ipynb

6. Calculate the number and occurrence of mission outcome of the orbits.

```
In [9]: # landing_outcomes = values on Outcome column  
        landing_outcomes = df['Outcome'].value_counts()  
        landing_outcomes
```

```
Out[9]: Outcome  
True ASDS      41  
None None      19  
True RTLS      14  
False ASDS      6  
True Ocean      5  
False Ocean     2  
None ASDS       2  
False RTLS      1  
Name: count, dtype: int64
```


Data Wrangling

- Data was processed using NumPy and Pandas Python libraries.
- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/data_wrangling/spacex-Data%20wrangling.ipynb

7. Create a landing outcome label from Outcome column.

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

```
In [14]: # Landing_class = 0 if bad_outcome
# Landing_class = 1 otherwise
landing_class = list(df['Outcome'])
j=0
for i in landing_class:
    if i in bad_outcomes:
        landing_class[j] = 0
        j += 1
    else:
        landing_class[j] = 1
        j += 1
landing_class
```

8. Export data to a flat *.csv file.

```
In [21]: df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

- EDA and Data Visualization was completed using Numpy, Pandas, Matplotlib and Seaborn Python libraries.
 - GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/exploratory_data_analysis_EDA/eda-dataviz.ipynb
1. Visualize the relationship between Flight Number and Launch Site.
 2. Visualize the relationship between Payload and Launch Site.
 3. Visualize the relationship between success rate of each orbit type.
 4. Visualize the relationship between Flight Number and Orbit type.
 5. Visualize the relationship between Payload and Orbit type.
 6. Visualize the launch success yearly trend.
 7. Create dummy variables to categorical columns.
 8. Export data to a flat *.csv file.

EDA with SQL

- Applied EDA with SQL to get insights from the data.
- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/exploratory_data_analysis_EDA/spacex-eda-sql.ipynb
- Wrote queries to find out for the next instances:
 - The names of unique launch sites in the space mission.
 - The total payload mass carried by boosters launched by NASA (CRS)
 - The average payload mass carried by booster version F9 v1.1
 - The total number of successful and failure mission outcomes
 - The failed landing outcomes in drone ship, their booster version and launch site names.

Build an Interactive Map with Folium

- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/interactive-visual-analytics-and-dashboards/spacex_launch_site_location_analysis.ipynb
- Marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- Assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Identified which launch sites have relatively high success rate.
- Calculated the distances between a launch site to its proximities.

Build a Dashboard with Plotly Dash

- GitHub URL:
https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/spacex_dash_app.py
- Built an interactive dashboard with Plotly dash.
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

Predictive Analysis (Classification)

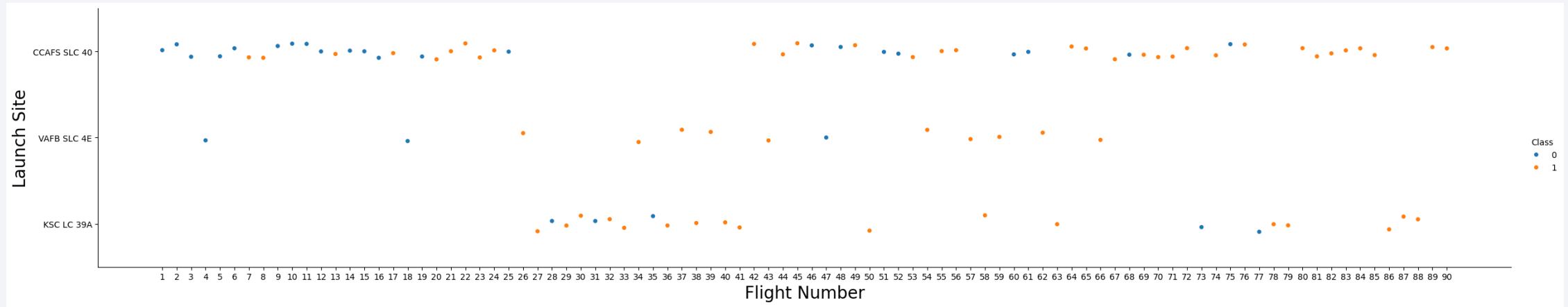
- GitHub URL:
[https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/predictive-machine-learning-model-analysis-%20classification/SpaceX Machine Learning Prediction.jupyterlite.ipynb](https://github.com/AlbertoMPalacioBastos/ML-model-for-SpaceX-booster-landing-success-prediction/blob/main/predictive-machine-learning-model-analysis-%20classification/SpaceX%20Machine%20Learning%20Prediction.jupyterlite.ipynb)
- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Found the best performing classification model.

The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower-left quadrant. The overall effect is dynamic and technological.

Section 2

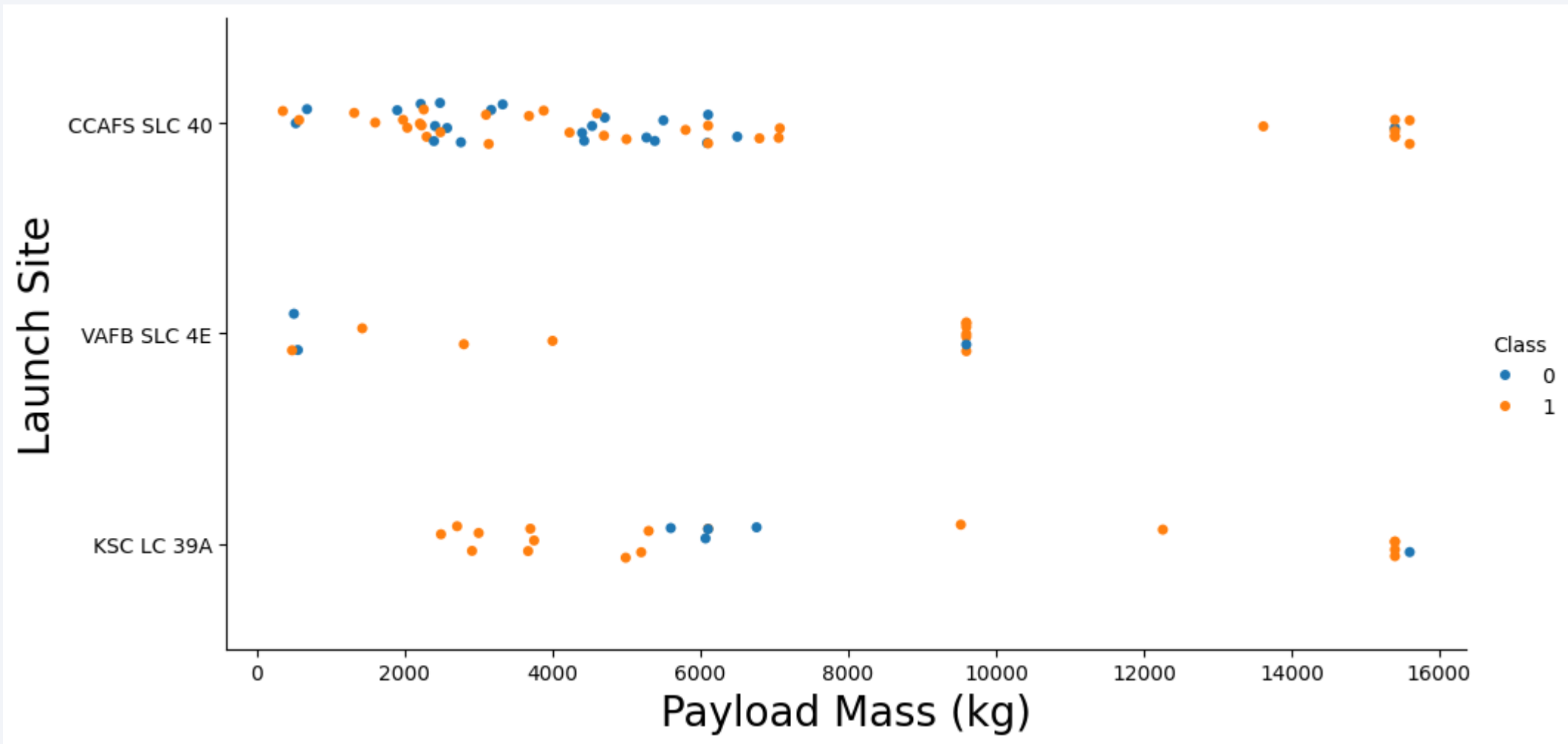
Insights drawn from EDA

Flight Number vs. Launch Site

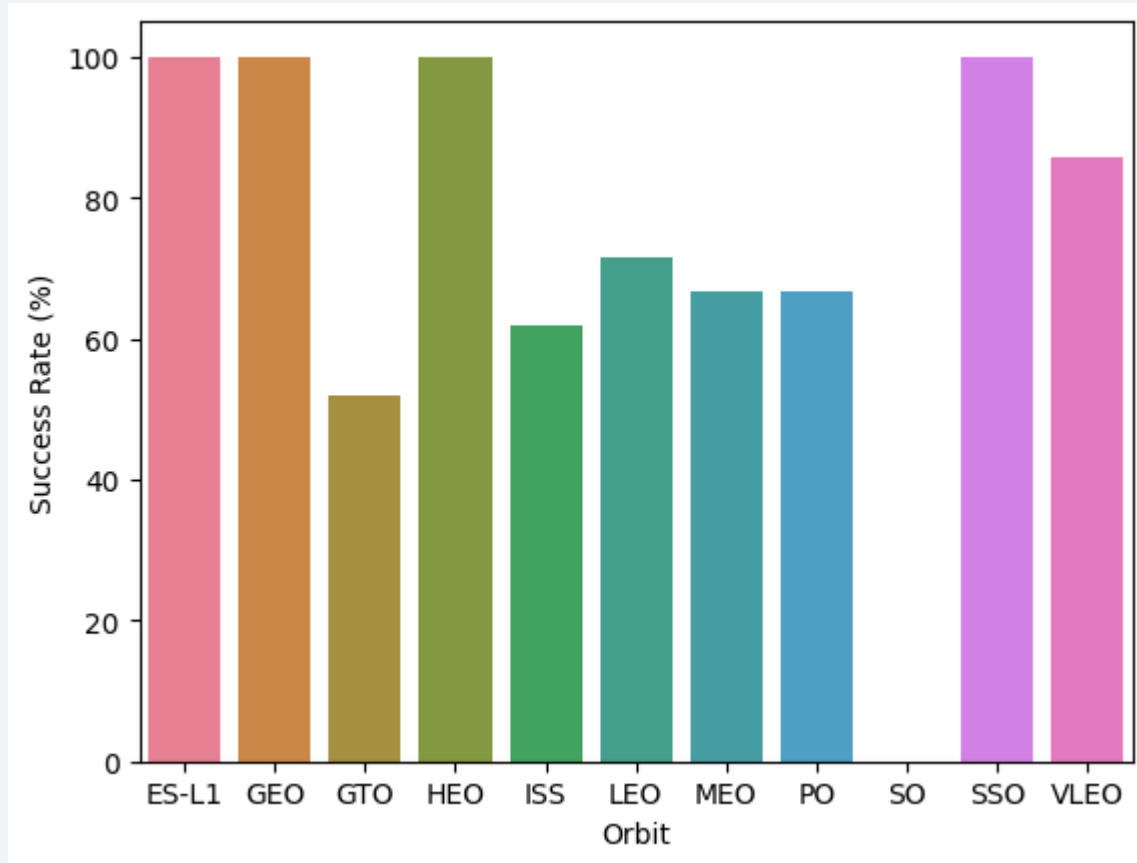


The larger the flight amount at a launch site, the greater the success rate at the launch site.

Payload vs. Launch Site

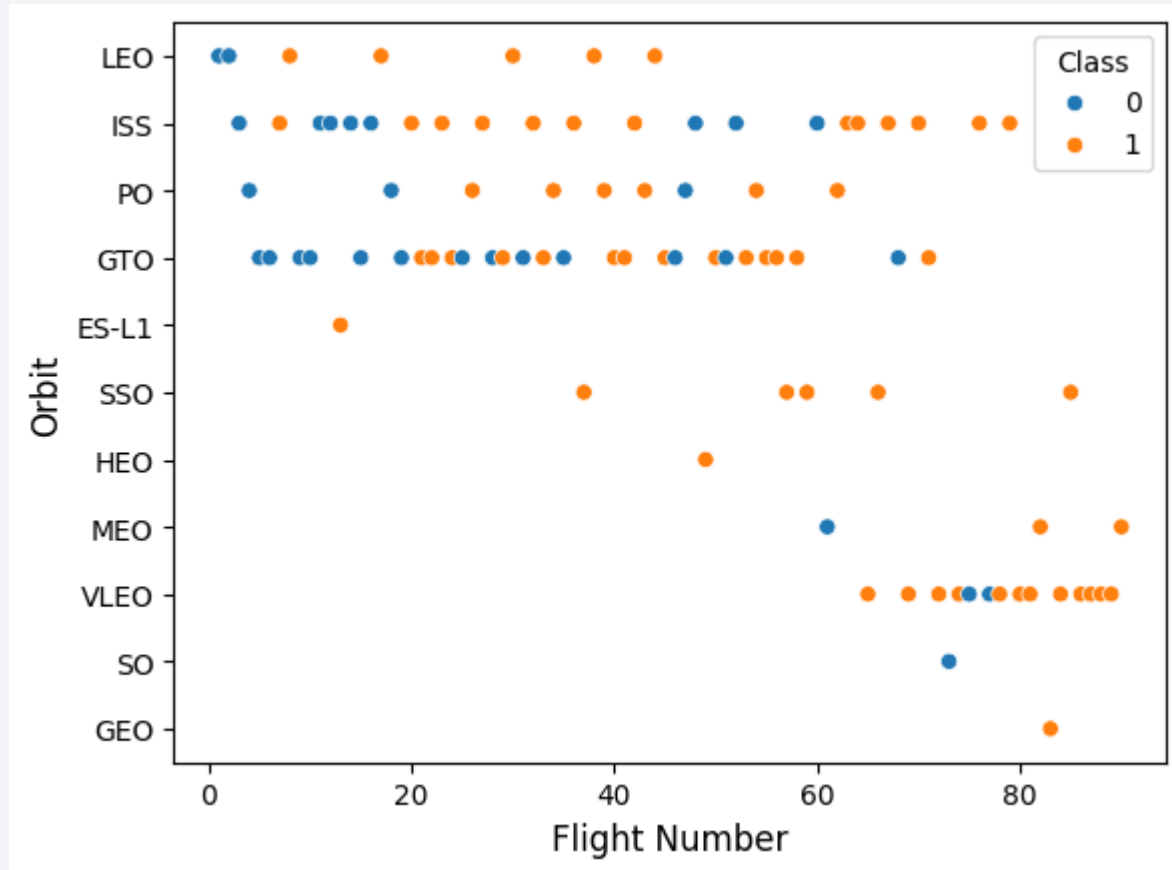


Success Rate vs. Orbit Type



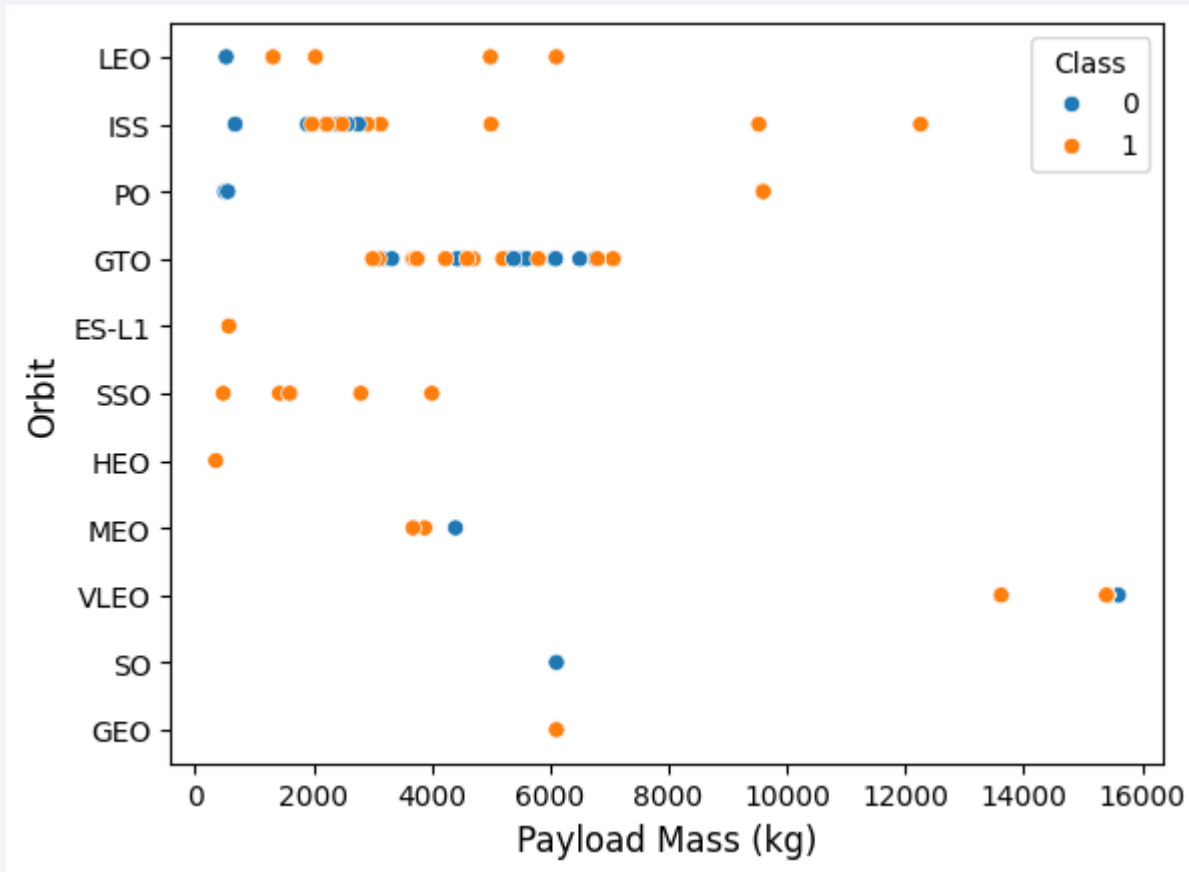
ES-L1, GEO, HEO, SSO and VLEO orbits had the most success rate with a 100%.

Flight Number vs. Orbit Type



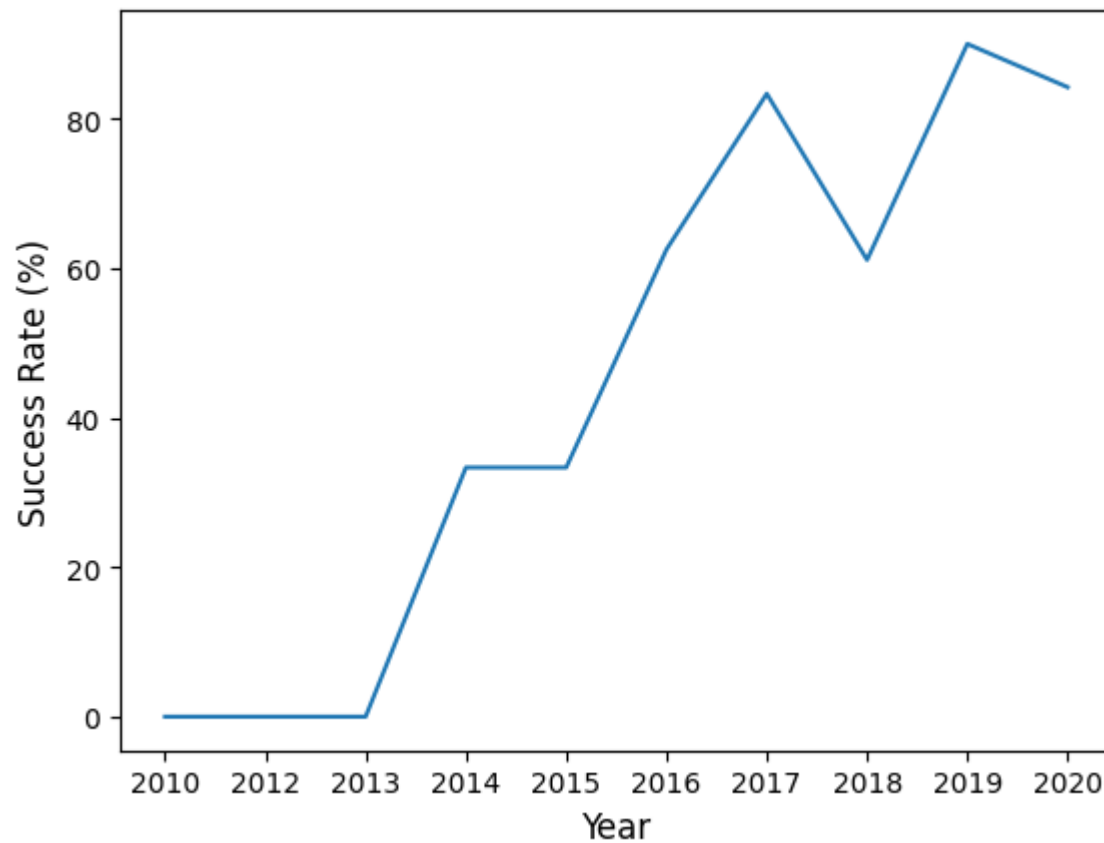
In the LEO orbit, success is related to the number of flights, whereas in the GTO orbit, there is no relationship between flight number and the orbit.

Payload vs. Orbit Type



With heavy payloads, the more successful is the landing attempt for PO, LEO and ISS orbits.

Launch Success Yearly Trend



As the SpaceX team learned with each landing attempt, the average yearly success rate kept increasing.

All Launch Site Names

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

```
In [9]: %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE
```

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

```
Out[9]: Launch_Site
```

```
CCAFS LC-40
```

```
VAFB SLC-4E
```

```
KSC LC-39A
```

```
CCAFS SLC-40
```

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

```
In [10]: %%sql
SELECT *
FROM SPACEXTABLE
WHERE "Launch_Site" LIKE "CCA%"
LIMIT 5

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

Out[10]:

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

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

```
In [11]: %%sql
SELECT SUM("PAYLOAD_MASS_KG_") AS "TOTAL PAYLOAD MASS (KG) launched by 'NASA (CRS)'"
FROM SPACEXTABLE
WHERE "Customer" LIKE "NASA (CRS)"

* sqlite:///my_data1.db
Done.

Out[11]: TOTAL PAYLOAD MASS (KG) launched by 'NASA (CRS)'
          45596
```

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

```
In [12]: %%sql
SELECT AVG("PAYLOAD_MASS_KG_") AS "AVERAGE PAYLOAD MASS (KG) carried by Falcon 9 V1.1"
FROM SPACEXTABLE
WHERE "Booster_Version" LIKE "F9 v1.1%"

* sqlite:///my_data1.db
Done.

Out[12]: AVERAGE PAYLOAD MASS (KG) carried by Falcon 9 V1.1
                2534.6666666666665
```

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

```
In [13]: %%sql
SELECT MIN("Date")
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE "%Success%"
```

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

```
Out[13]: MIN("Date")
2015-12-22
```


Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
In [21]: %%sql
SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE "%Success (drone %" AND ("PAYLOAD_MASS_KG_" BETWEEN 4000 AND 6000)
```

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

```
Done.
```

```
Out[21]: Booster_Version
```

```
F9 FT B1022
```

```
F9 FT B1026
```

```
F9 FT B1021.2
```

```
F9 FT B1031.2
```

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
In [25]: %%sql
SELECT DISTINCT "Mission_Outcome", COUNT(*)
FROM SPACEXTABLE
```

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

```
Out[25]:
```

Mission_Outcome	COUNT(*)
Success	101

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass

```
In [29]: %%sql
SELECT DISTINCT Booster_Version
FROM SPACEXTABLE
WHERE PAYLOAD_MASS_KG_ = (
    SELECT MAX(PAYLOAD_MASS_KG_)
    FROM SPACEXTABLE
)
```

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

```
Out[29]: Booster_Version
```

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

2015 Launch Records

- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
In [31]: %%sql
SELECT
    substr(Date, 6, 2) AS Month,
    Booster_Version,
    Launch_Site
FROM
    SPACEXTABLE
WHERE
    substr(Date, 0, 5) = "2015"
    AND Landing_Outcome LIKE "%Failure (drone ship)%"
```

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

```
Out[31]:
```

Month	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

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

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

```
In [32]: %%sql
SELECT
    Landing_Outcome,
    COUNT(*) AS Outcome_Count,
    RANK() OVER (ORDER BY COUNT(*) DESC) AS Outcome_Rank
FROM
    SPACE_TABLE
WHERE
    Date BETWEEN '2010-06-04' AND '2017-03-20'
    AND Landing_Outcome IN ('Failure (drone ship)', 'Success (ground pad)')
GROUP BY
    Landing_Outcome
ORDER BY
    Outcome_Count DESC
```

* sqlite:///my_data1.db

Done.

```
Out[32]:
```

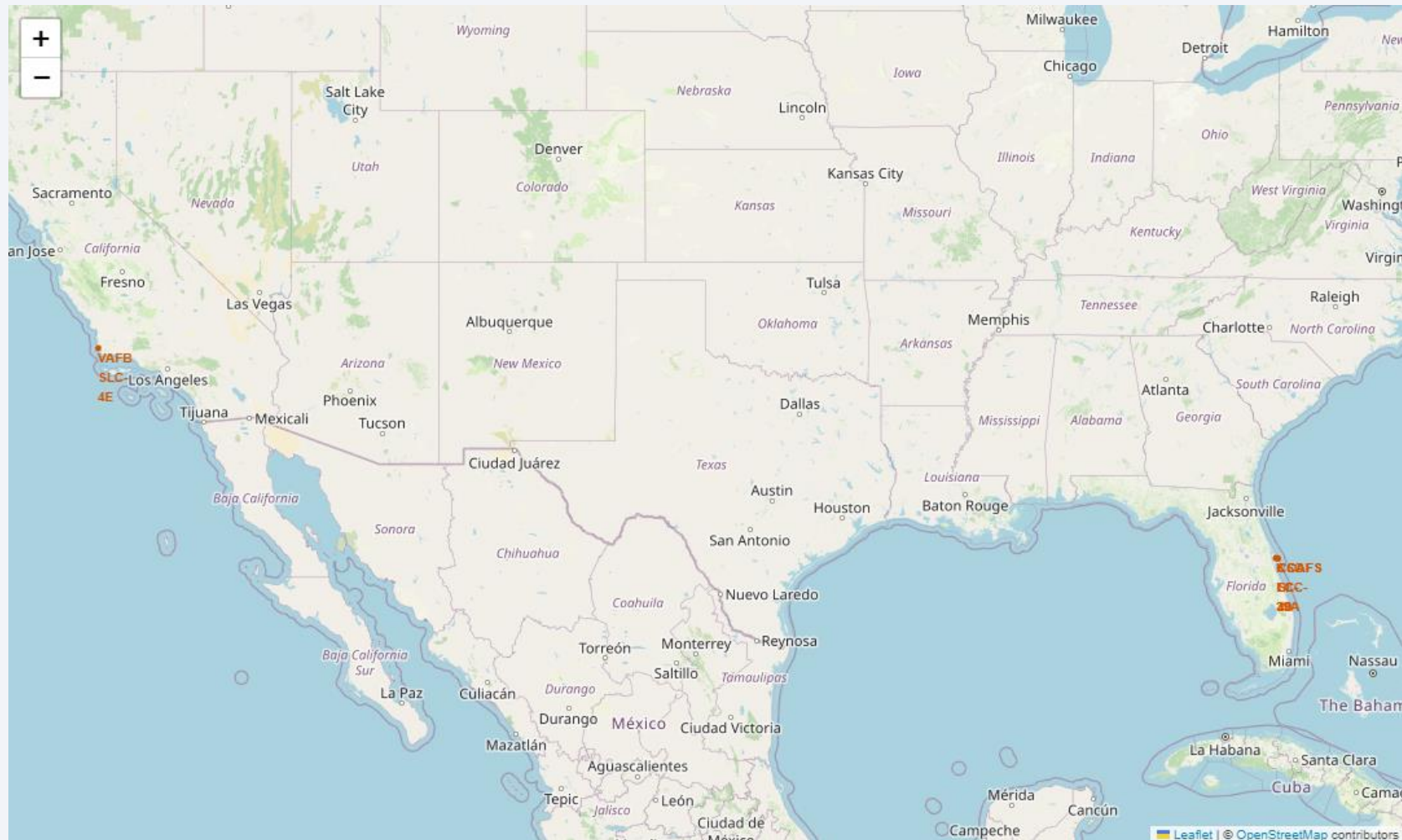
Landing_Outcome	Outcome_Count	Outcome_Rank
Failure (drone ship)	5	1
Success (ground pad)	3	2

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

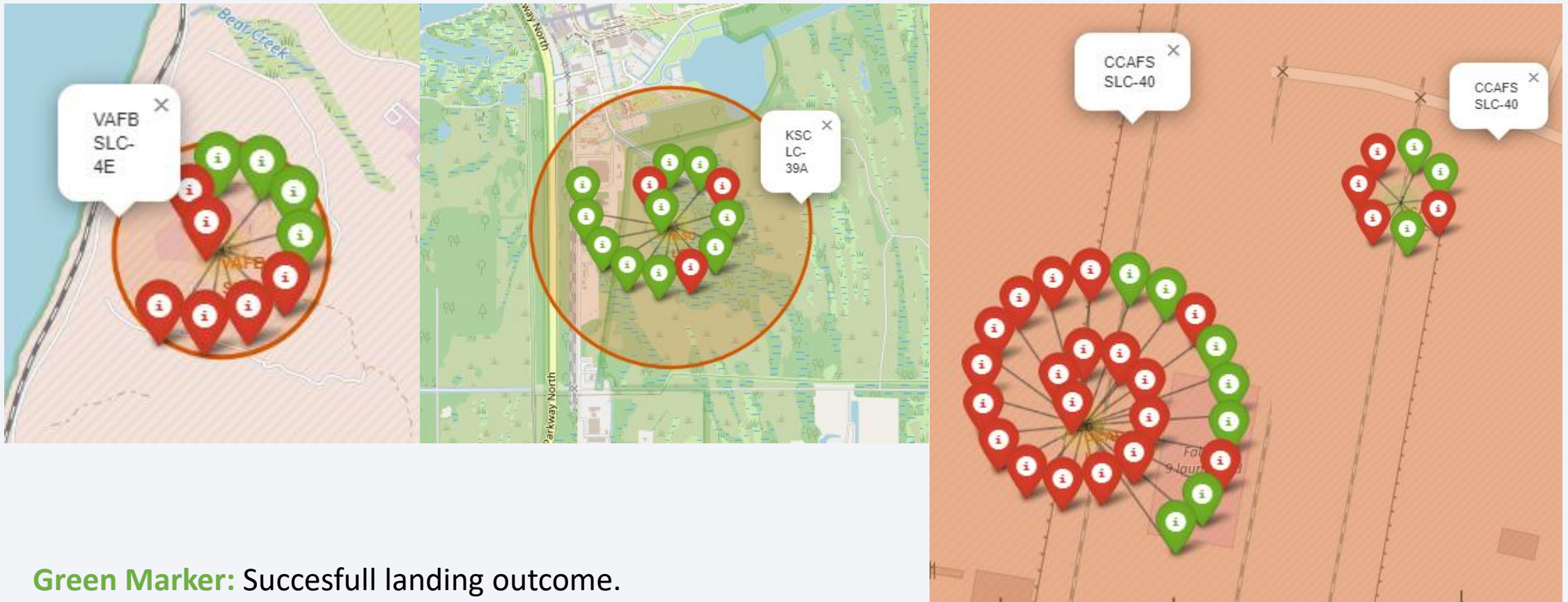
Section 3

Launch Sites Proximities Analysis

All launch sites' location



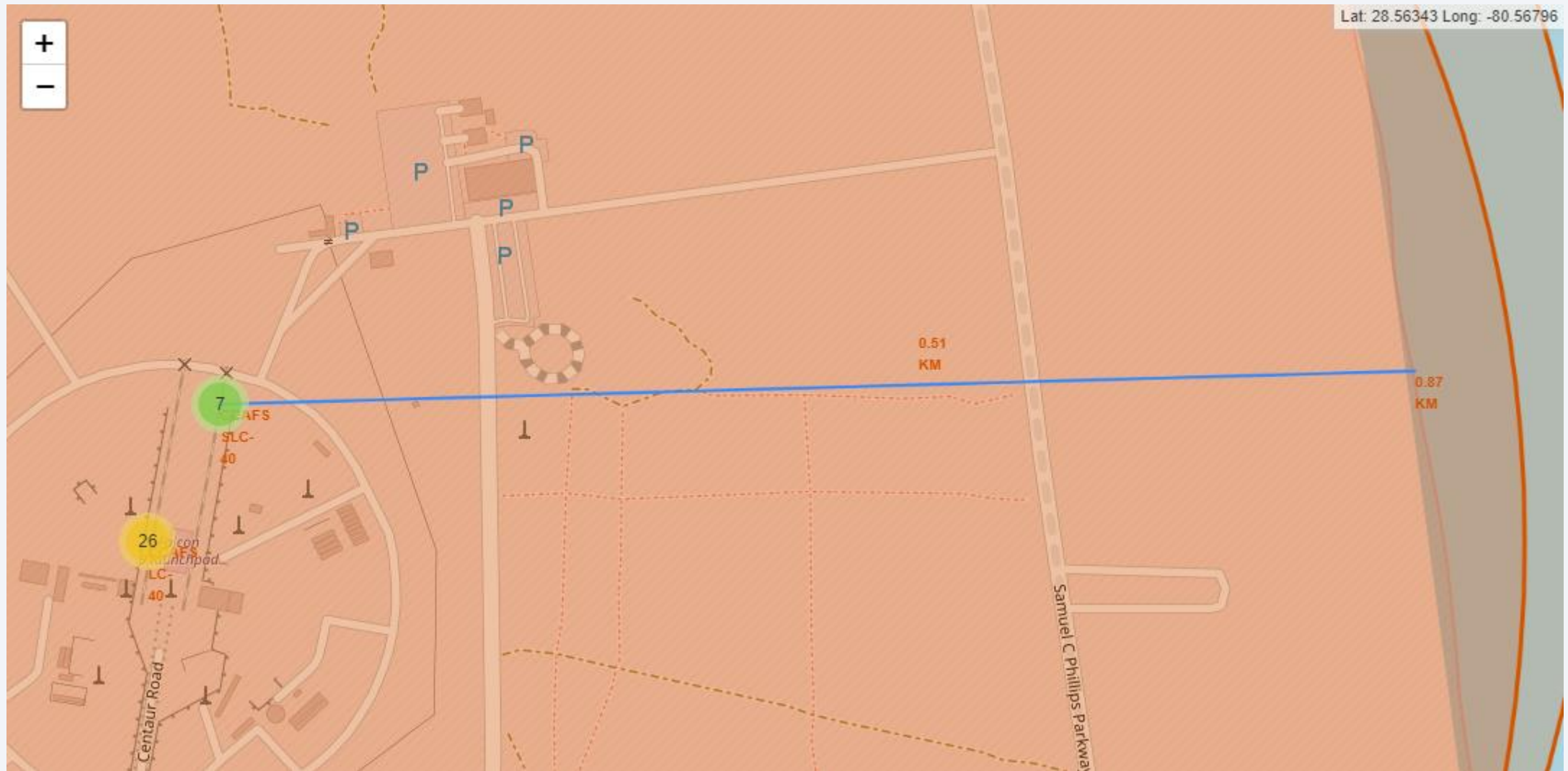
Markers showing launch sites with color labels



Green Marker: Succesfull landing outcome.

Red Marker: Unsuccesfull landing outcome.

Launch site proximity to coastline

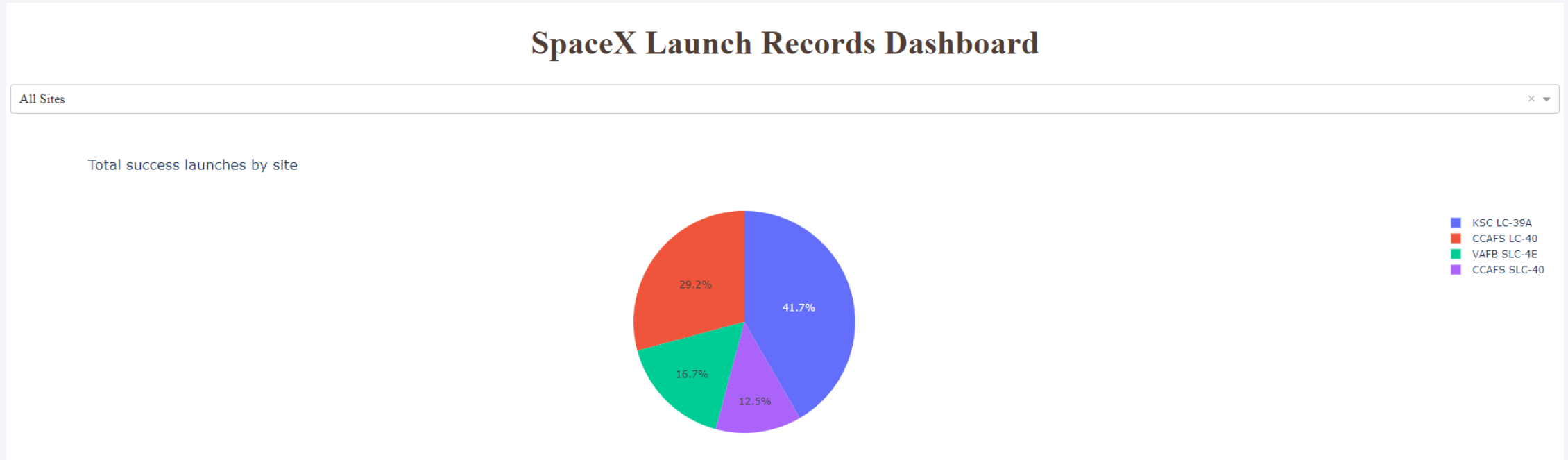




Section 4

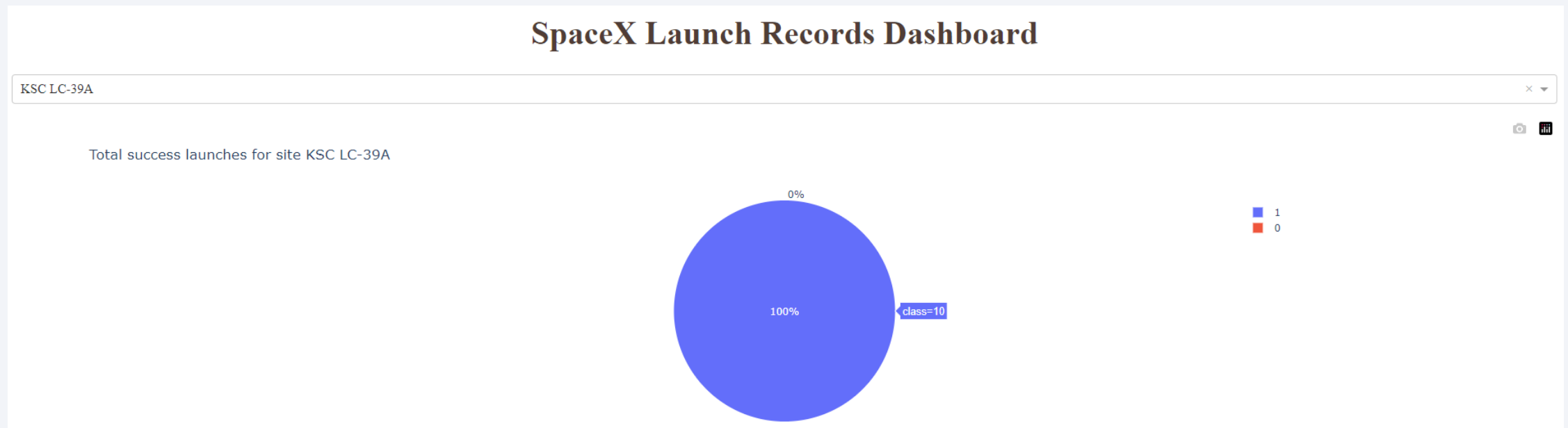
Build a Dashboard with Plotly Dash

Launch success count for all sites piechart



KSC LC-39A has the highest success rate of all launch sites with 41.7%

Piechart for the launch site with the highest launch success ratio

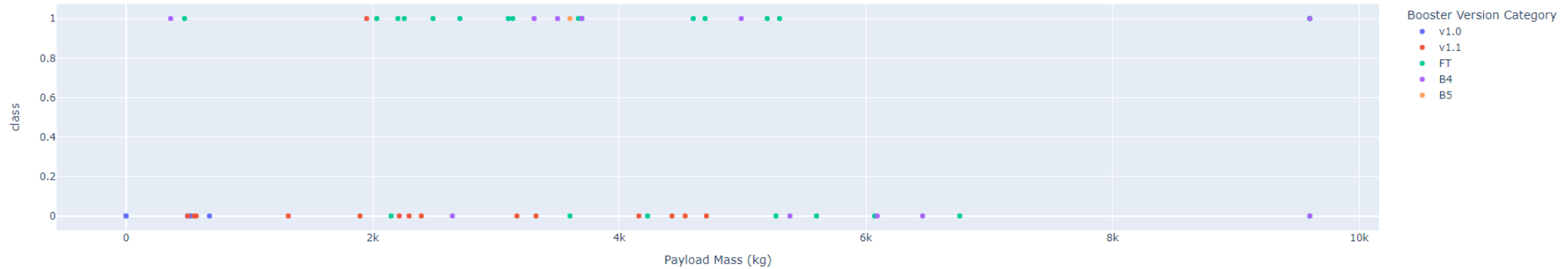


Payload vs. Launch Outcome scatter plot for all sites, with different payload ranges

Payload range (Kg):



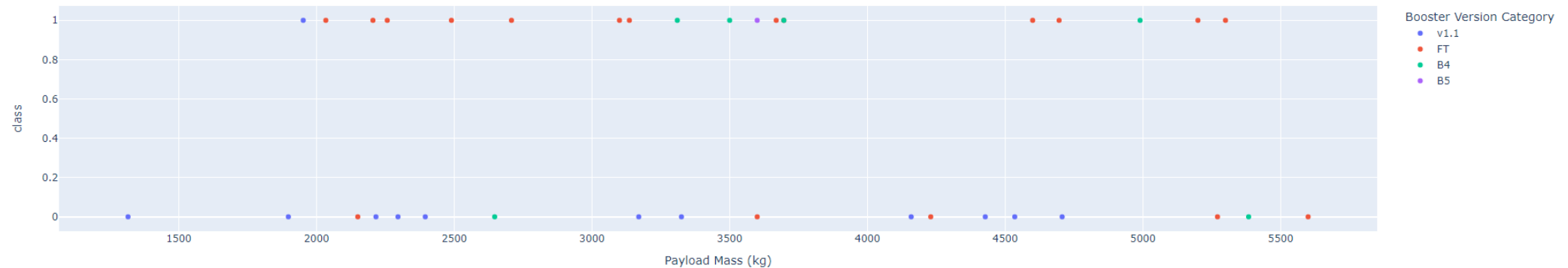
Correlation between Payload and Success for all Sites



Payload range (Kg):



Correlation between Payload and Success for all Sites



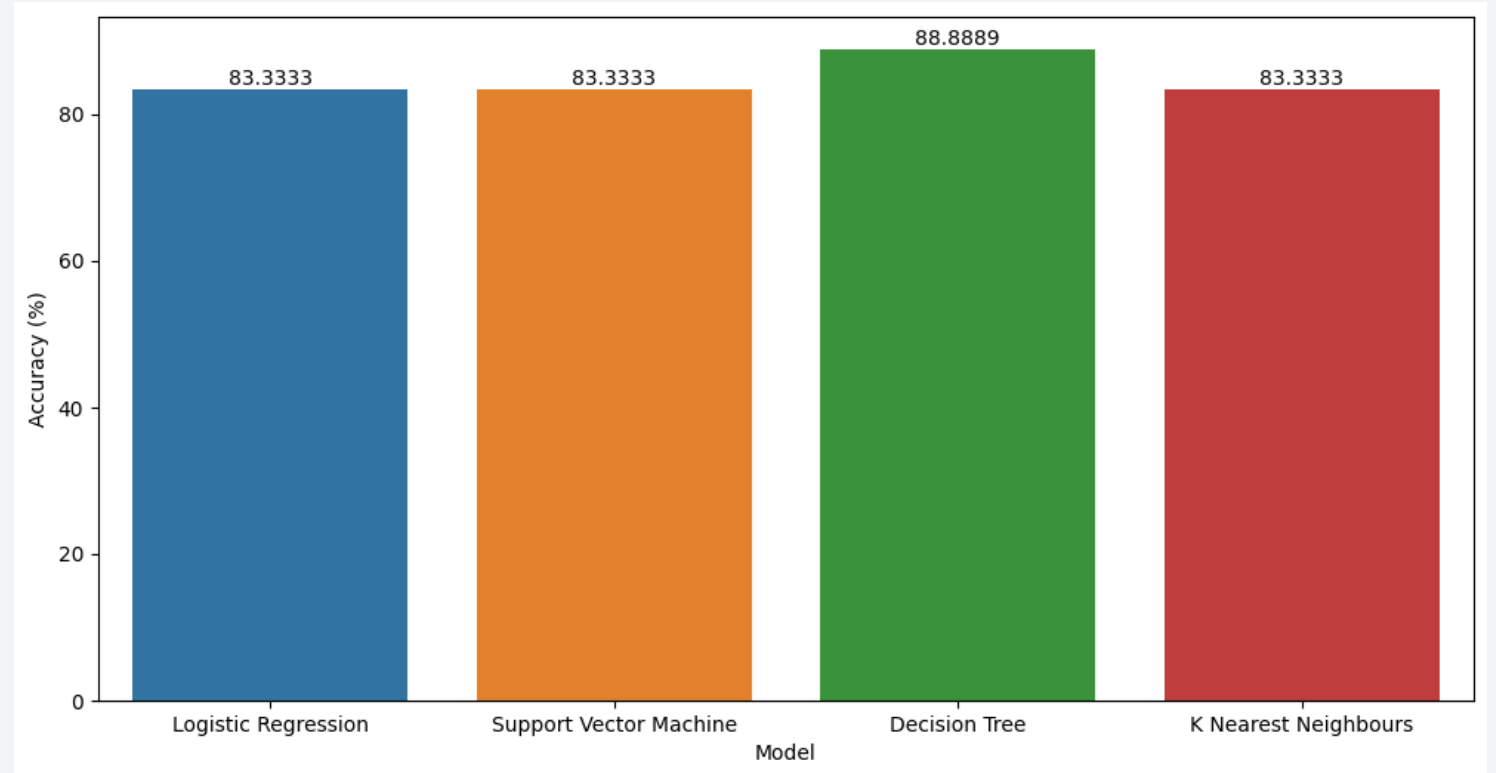


Section 5

Predictive Analysis (Classification)

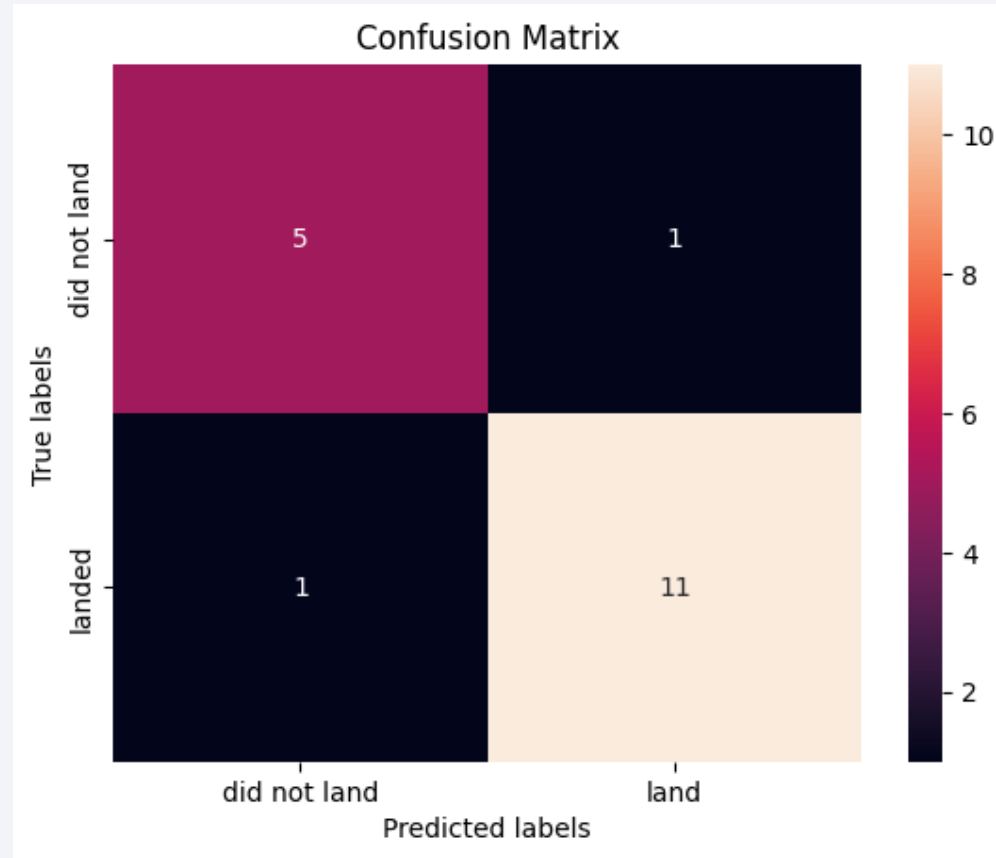
Classification Accuracy

	Model	Accuracy (%)
0	Logistic Regression	83.333333
1	Support Vector Machine	83.333333
2	Decision Tree	88.888889
3	K Nearest Neighbours	83.333333



The model with the highest accuracy score is the Decision Tree

Confusion Matrix



Tuned hyperparameters
(best parameters):
'criterion': 'entropy',
'max_depth': 4,
'max_features': 'sqrt',
'min_samples_leaf': 4,
'min_samples_split': 2,
'splitter': 'random'}
accuracy : 88.93 %

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes.

Conclusions

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!

