IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Akash Bhattacharya
October 06, 2025

Outline

Executive Summary

- In this project, we have tried to predict whether the Falcon 9 1st stage will land successfully, based on various parameters.
- These parameters include launch site, payload mass, target orbit and more.
- We have used data from SpaceX's API and from Wikipedia.
- We have visualized the data and analyzed it using different machine learning models.
- We have created a model which successfully predicts whether the Falcon 9 1st stage will land with ~ 84% accuracy.

Introduction 1

- This is the Data Science and Machine Learning Capstone Project of the IBM Data Science Professional Certificate (Course # IBM DS0720EN on edX).
- (Space Exploration Technologies Corp.) is a private American aerospace company founded by Elon Musk in 2002. It designs, manufactures, and launches rockets and spacecraft, aiming to reduce space transportation costs and enable human colonization of Mars.
- SpaceX advertised costs are ~ $62M, as comparted to >~$165M for other vendors. These cost savings are based on reusability of the 1st stage of the Falcon 9 rocket.
- This capstone project requires the student to analyze publicly available launch data on the SpaceX Falcon 9 system to determine if the first stage will land successfully.

Introduction 2

- The Falcon 9 is a reusable, two-stage medium-lift launch vehicle developed by SpaceX. It is designed to transport payloads and humans to Earth orbit and beyond. Its first stage can land vertically for reuse, significantly reducing launch costs. (Ref: SpaceX - Falcon 9)
- The problem addressed in this study is to predict the success of a Falcon 9 launch. This is done by analyzing the available data on Falcon9 launches and building a predictive model.

The Falcon 9 system

Falcon 9 Performance Summary and Comparison

•**Performance**:
- Payload to Low Earth Orbit (LEO): 22,800 kg
- Payload to Geostationary Transfer Orbit (GTO): 8,300 kg
- Reusability: First stage is reusable with vertical landing capability. [www.spacex.com]

•**Comparison**:
- **Falcon 9 vs. Falcon Heavy**: Falcon Heavy has ~3x the payload capacity of Falcon 9 (up to 63.8 tons to LEO) due to its additional boosters. [impulso.space]
- **Falcon 9 vs. Ariane 6**: Ariane 6 (Europe) offers up to 10.9 tons to LEO and is expendable, making Falcon 9 more cost-effective due to reusability. [en.wikipedia.org]
- **Falcon 9 vs. Atlas V**: Atlas V (USA) is also expendable and less frequently launched; Falcon 9 has a higher launch cadence and lower cost per launch. [en.wikipedia.org]

- **Low cost is key:**
  - The Falcon 9 has relatively low base launch costs as compared to other launch vehicles. The Falcon 9 1st stage is designed to be recovered and reused, which can further lower costs. (Ref: Rocket Launch Costs (2020-2030): How Cheap Is Space Travel Becoming? (Latest Pricing Data) | PatentPC)

Section 1

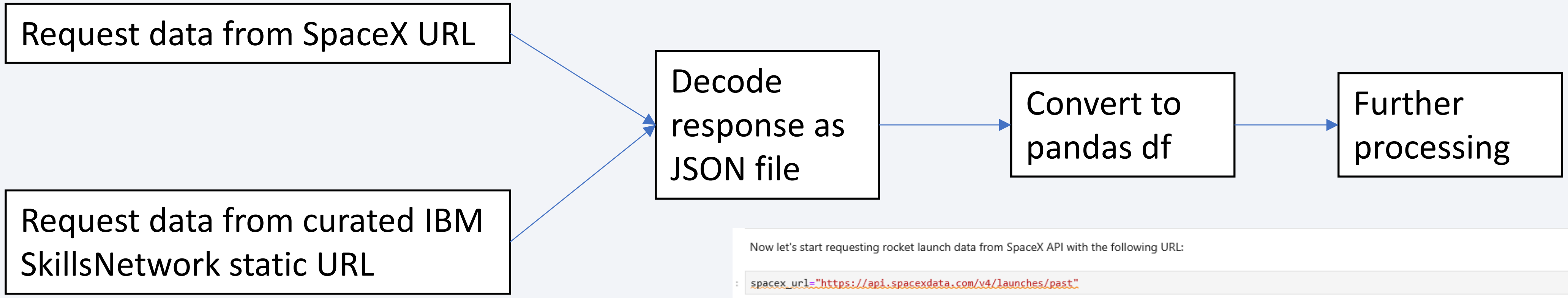
Methodology

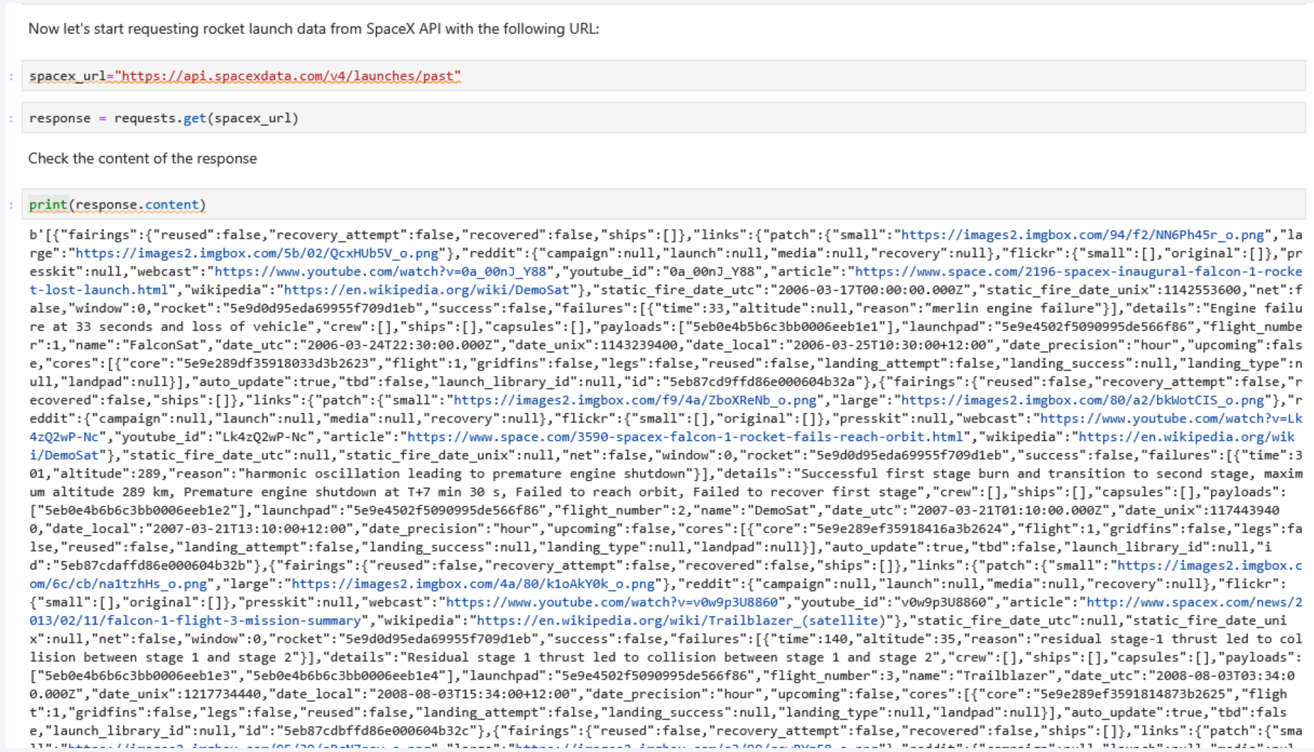
Methodology

Executive Summary

- Data collection methodology:
  - Request data from SpaceX API
  - Collect data from Wikipedia
  - Request data from course's static URLs
- Perform data wrangling
  - Done with numpy and pandas to convert landing data outcomes into training labels.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Several models were tested: Logistic, SVM, Classification Tree and KNN

Data Collection


Request data from SpaceX URL
Request data from curated IBM SkillsNetwork static URL
Decode response as JSON file
Convert to pandas df
Further processing



Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

Check the content of the response

```
print(response.content)
```

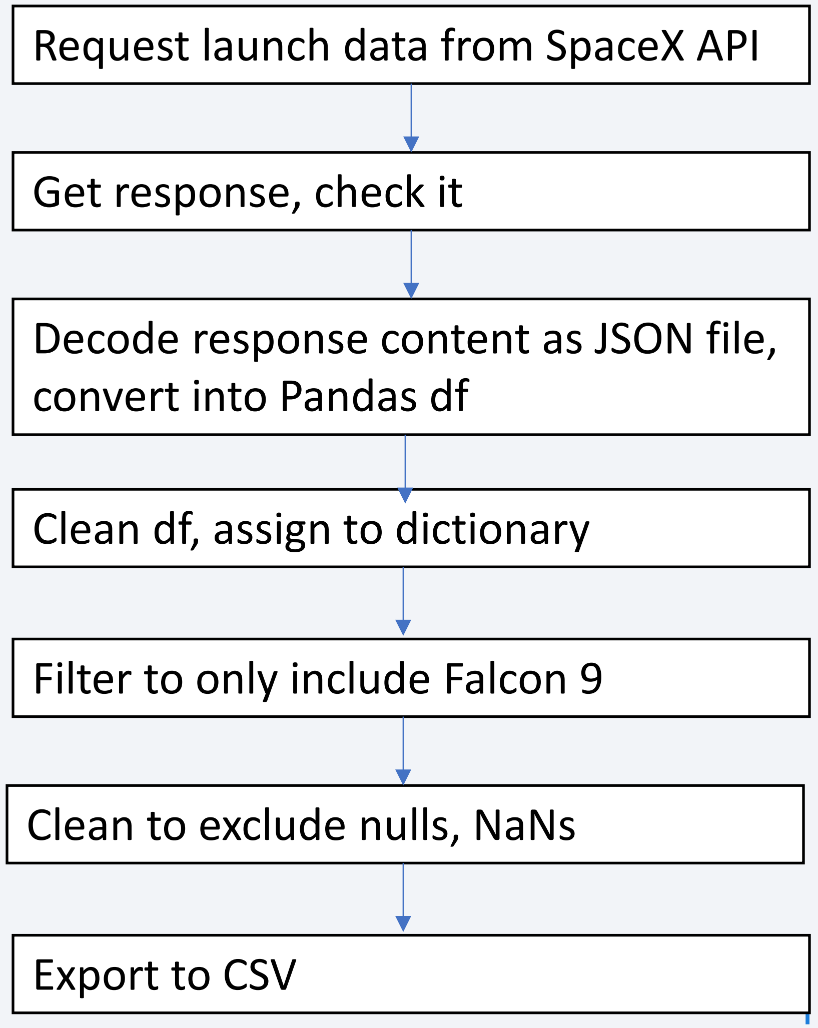
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.space.com/2196-spacex-inaugural-falcon-1-rocket-lost-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":33,"altitude":null,"reason":"merlin engine failure"}],"details":"Engine failure at 33 seconds and loss of vehicle","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e4502f5090995de566f86","flight_number":1,"name":"FalconSat","date_utc":"2006-03-24T22:30:00.000Z","date_unix":1143239400,"date_local":"2006-03-25T10:30:00+12:00","date_precision":"hour","upcoming":false,"cores":[{"core":"5e9e289df35918033d3b2623","flight":1,"gridfins":false,"legs":false,"reused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_update":true,"tbd":false,"launch_library_id":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/f9/4a/ZboXReNb_o.png","large":"https://images2.imgbox.com/80/a2/bkWotCIS_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=Lk4zQ2wP-Nc","youtube_id":"Lk4zQ2wP-Nc","article":"https://www.space.com/3590-spacex-falcon-1-rocket-fails-reach-orbit.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fire_date_utc":null,"static_fire_date_unix":null,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":301,"altitude":289,"reason":"harmonic oscillation leading to premature engine shutdown"}],"details":"Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b6b6c3bb0006eeb1e2"],"launchpad":"5e9e4502f5090995de566f86","flight_number":2,"name":"DemoSat","date_utc":"2007-03-21T01:10:00.000Z","date_unix":1174439400,"date_local":"2007-03-21T13:10:00+12:00","date_precision":"hour","upcoming":false,"cores":[{"core":"5e9e289ef35918416a3b2624","flight":1,"gridfins":false,"legs":false,"reused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_update":true,"tbd":false,"launch_library_id":null,"id":"5eb87cdaffd86e000604b32b"},{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/6c/cb/na1tzhHs_o.png","large":"https://images2.imgbox.com/4a/80/k1oAkY0k_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=v0w9p3U8860","youtube_id":"v0w9p3U8860","article":"http://www.spacex.com/news/2013/02/11/falcon-1-flight-3-mission-summary","wikipedia":"https://en.wikipedia.org/wiki/Trailblazer_(satellite)"},"static_fire_date_utc":null,"static_fire_date_unix":null,"net":false,"window":0,"rocket":"5e9d0d95eda69955f709d1eb","success":false,"failures":[{"time":140,"altitude":35,"reason":"residual stage-1 thrust led to collision between stage 1 and stage 2"}],"details":"Residual stage 1 thrust led to collision between stage 1 and stage 2","crew":[],"ships":[],"capsules":[],"payloads":["5eb0e4b6b6c3bb0006eeb1e3","5eb0e4b6b6c3bb0006eeb1e4"],"launchpad":"5e9e4502f5090995de566f86","flight_number":3,"name":"Trailblazer","date_utc":"2008-08-03T03:34:00.000Z","date_unix":1217734440,"date_local":"2008-08-03T15:34:00+12:00","date_precision":"hour","upcoming":false,"cores":[{"core":"5e9e289ef3591814873b2625","flight":1,"gridfins":false,"legs":false,"reused":false,"landing_attempt":false,"landing_success":null,"landing_type":null,"landpad":null}],"auto_update":true,"tbd":false,"launch_library_id":null,"id":"5eb87cdbffd86e000604b32c"},{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"sma

Screenshot: showing request from SpaceX URL

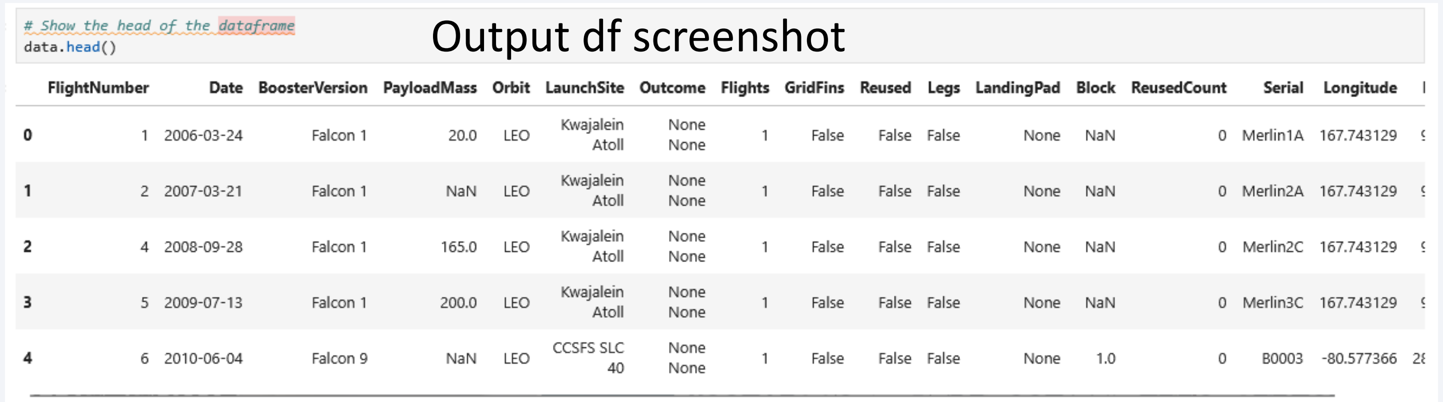
Data Collection – SpaceX API

```
# Get data

spacex_url=https://api.spacexdata.com/v4/launches/past

response = requests.get(spacex_url)

# Use json_normalize method to convert the json result into a dataframe

data =pd.json_normalize(response.json())

# Get the head of the dataframe

data.head()
```

- Github: https://github.com/iab149/IBM-DS0720EN-Data-Science-and-Machine-Learning-Capstone-Project/blob/cc6b300c3ef2ad57605ac10f6c0ca29508e2b6db/IBM-DS0720EN-AB-spacex-data-collection-api.ipynb


Request launch data from SpaceX API
Get response, check it
Decode response content as JSON file, convert into Pandas df
Clean df, assign to dictionary
Filter to only include Falcon 9
Clean to exclude nulls, NaNs
Export to CSV

Output df screenshot



```
# Show the head of the dataframe
data.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366

Data Collection - Scraping

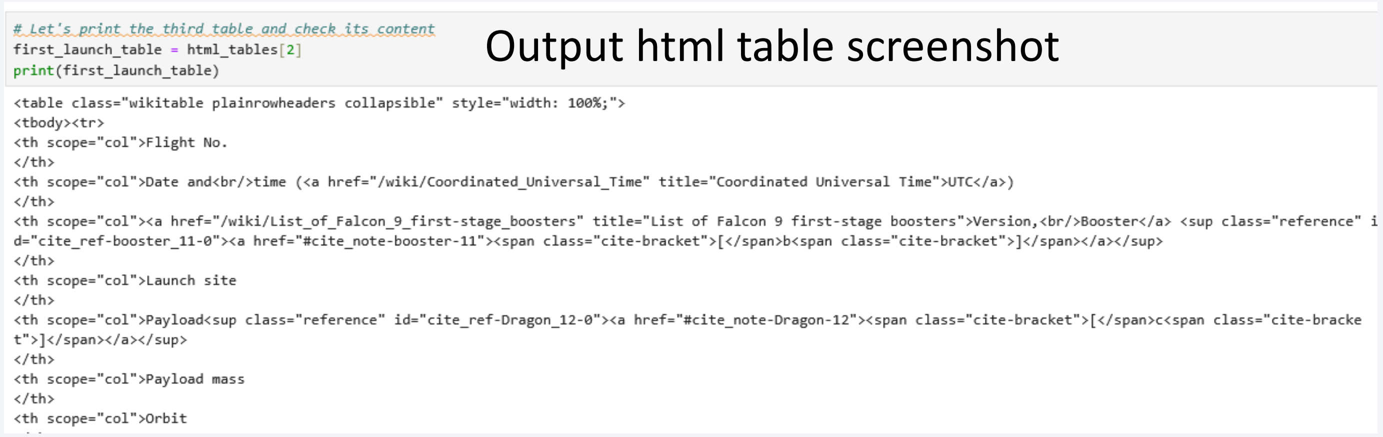
static_url =
"https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=102768692
2

headers = {   "User-Agent": "IBM-DS0720EN-DataScience-CapstoneProject"}

# use requests.get() method with the provided static_url
# assign the response to a object

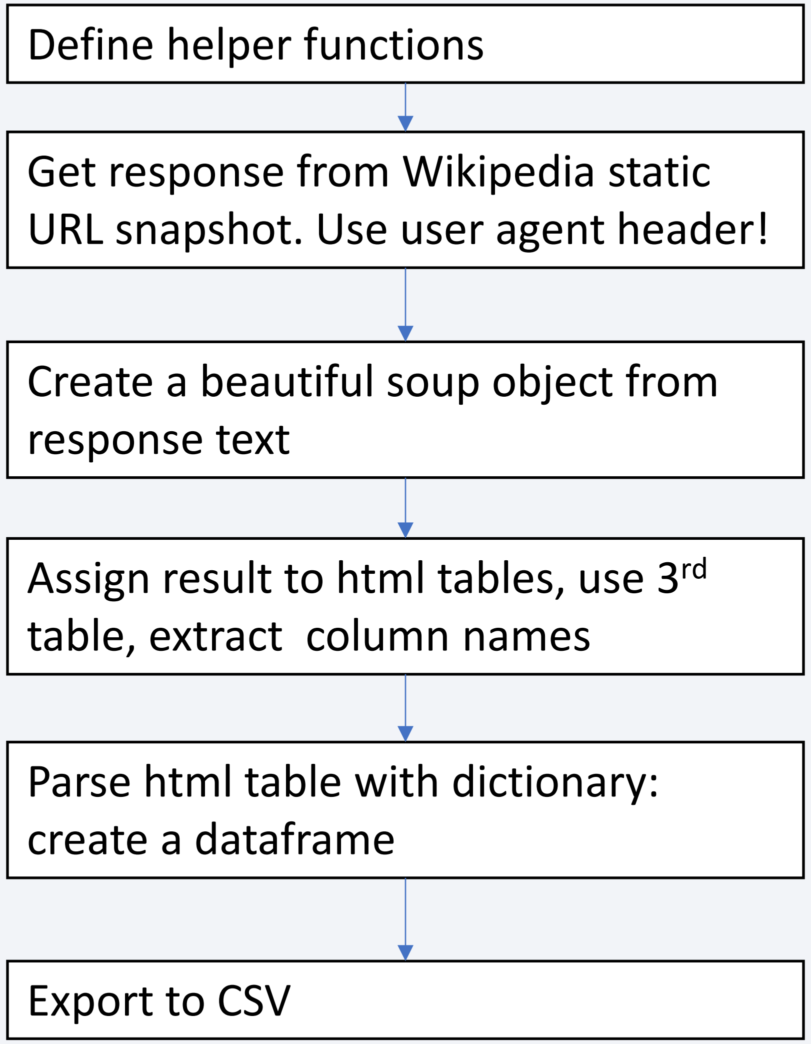
response = requests.get(static_url, headers=headers)
print(response.status_code)
print(response.content)

- Github: https://github.com/iab149/IBM-DS0720EN-Data-Science-and-Machine-Learning-Capstone-Project/blob/cc6b300c3ef2ad57605ac10f6c0ca29508e2b6db/IBM-DS0720EN-AB-jupyter-labs-webscraping.ipynb


Output html table screenshot

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

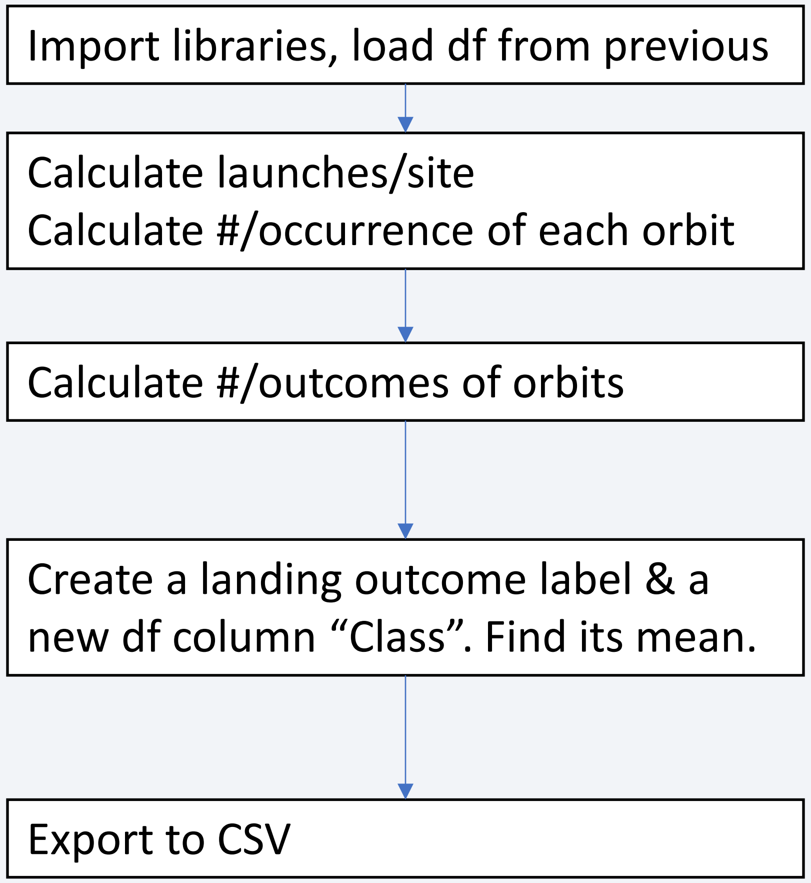
```
<table class="wikitable plainrowheaders collapsible" style="width: 100%;">
<tbody><tr>
<th scope="col">Flight No.
</th>
<th scope="col">Date and<br/>time (<a href="/wiki/Coordinated_Universal_Time" title="Coordinated Universal Time">UTC</a>)
</th>
<th scope="col"><a href="/wiki/List_of_Falcon_9_first-stage_boosters" title="List of Falcon 9 first-stage boosters">Version,<br/>Booster</a> <sup class="reference" i
d="cite_ref-booster_11-0"><a href="#cite_note-booster-11"><span class="cite-bracket">[</span>b<span class="cite-bracket">]</span></a></sup>
</th>
<th scope="col">Launch site
</th>
<th scope="col">Payload<sup class="reference" id="cite_ref-Dragon_12-0"><a href="#cite_note-Dragon-12"><span class="cite-bracket">[</span>c<span class="cite-bracke
t">]</span></a></sup>
</th>
<th scope="col">Payload mass
</th>
<th scope="col">Orbit
```


Define helper functions
Get response from Wikipedia static URL snapshot. Use user agent header!
Create a beautiful soup object from response text
Assign result to html tables, use 3rd table, extract column names
Parse html table with dictionary: create a dataframe
Export to CSV

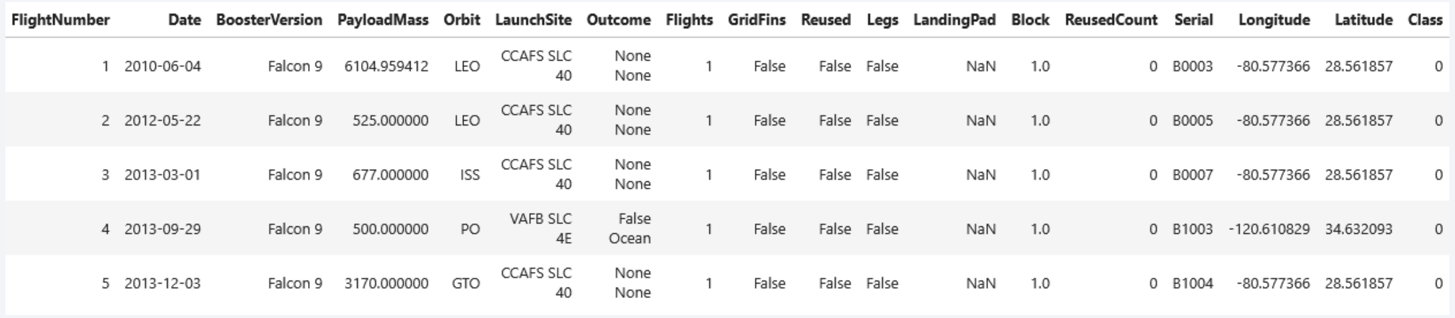
Data Wrangling

```
# Apply value_counts on Orbit column
# Count the number of launches by orbit type excluding 'GTO'orbit_
counts = df['Orbit'].value_counts()
print(orbit_counts)
# Remove 'GTO' from the result
orbit_counts_no_GTO = orbit_counts.drop('GTO', errors='ignore’)
# Display the result
print("Number of launches by orbit type (excluding GTO):")
print(orbit_counts_no_GTO)
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
print(landing_outcomes)
```

Github: https://github.com/iab149/IBM-DS0720EN-Data-Science-and-Machine-Learning-Capstone-Project/blob/135a0e310d7165b54918bdf26bb42a6331f4a067/IBM-DS0720EN-AB-labs-jupyter-spacex-data%20wrangling_jupyterlite.ipynb


Import libraries, load df from previous
Calculate launches/site
Calculate #/occurrence of each orbit
Calculate #/outcomes of orbits
Create a landing outcome label & a new df column “Class”. Find its mean.
Export to CSV

Screenshot of new df head with Class column



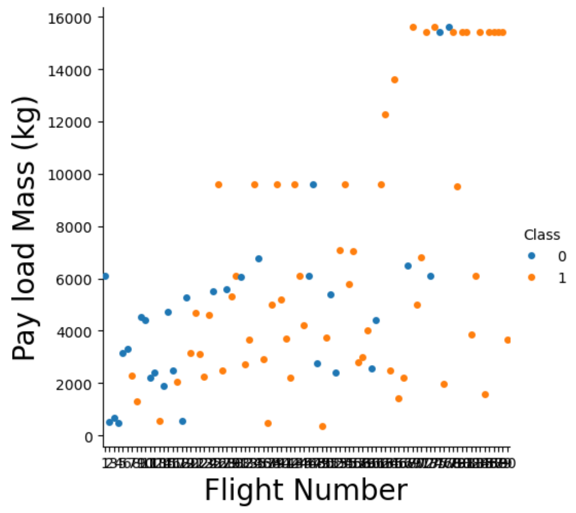
FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude	Class
1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0003	-80.577366	28.561857	0
2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0005	-80.577366	28.561857	0
3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B0007	-80.577366	28.561857	0
4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1	False	False	False	NaN	1.0	0	B1003	-120.610829	34.632093	0
5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1004	-80.577366	28.561857	0

EDA with Data Visualization

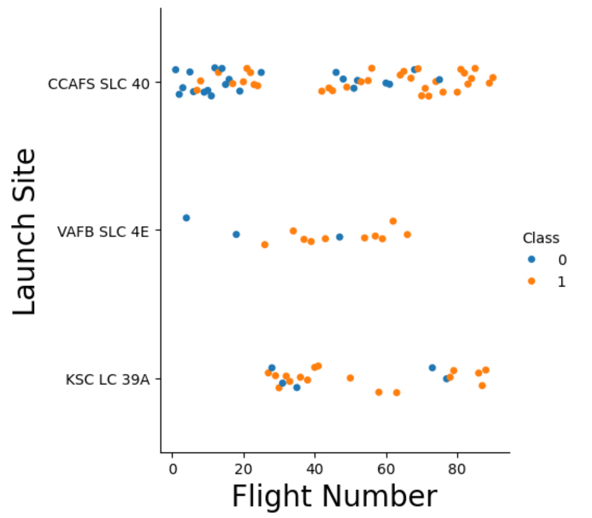
Various charts were plotted showing success (1, Orange) and Failure (0, Blue)

Matplotlib and Seaborn were used for plotting.

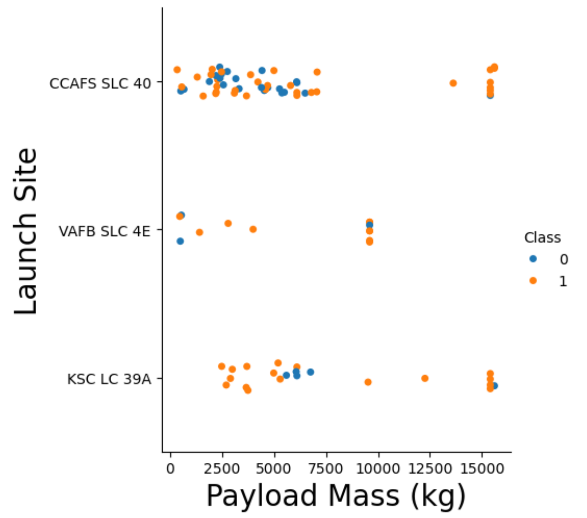
Github: https://github.com/iab149/IBM-DS0720EN-Data-Science-and-Machine-Learning-Capstone-Project/blob/37c5fce23030653318b020122deb621b87bfca84/IBM-DS0720EN-AB-edadataviz.ipynb



Payload vs Flight #



Launch Site vs Flight #

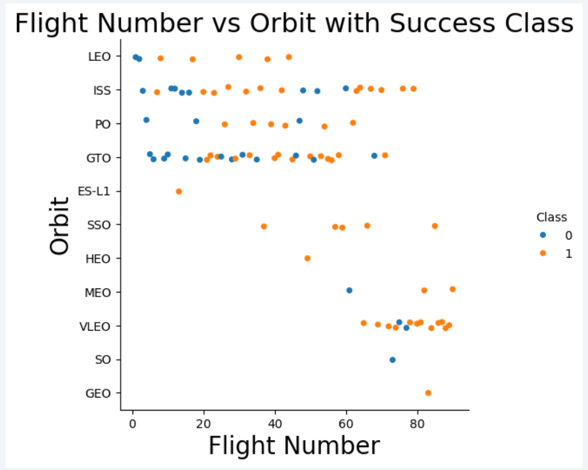


Launch Site vs Payload Mass

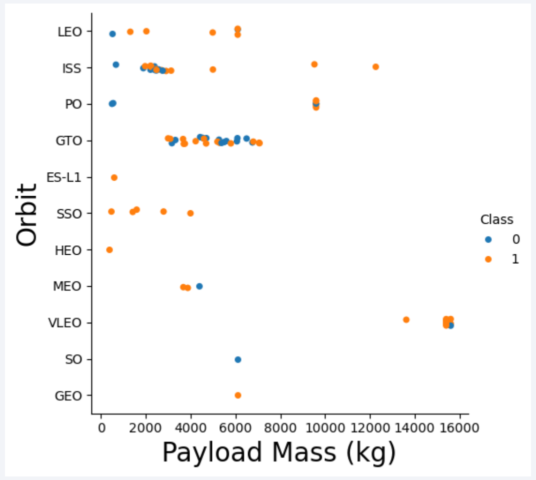
- Success improves with flight #
- Success improves with Payload Mass
- VAFB is not used to launch heavy payloads (> 10000 kg)

EDA with Data Visualization

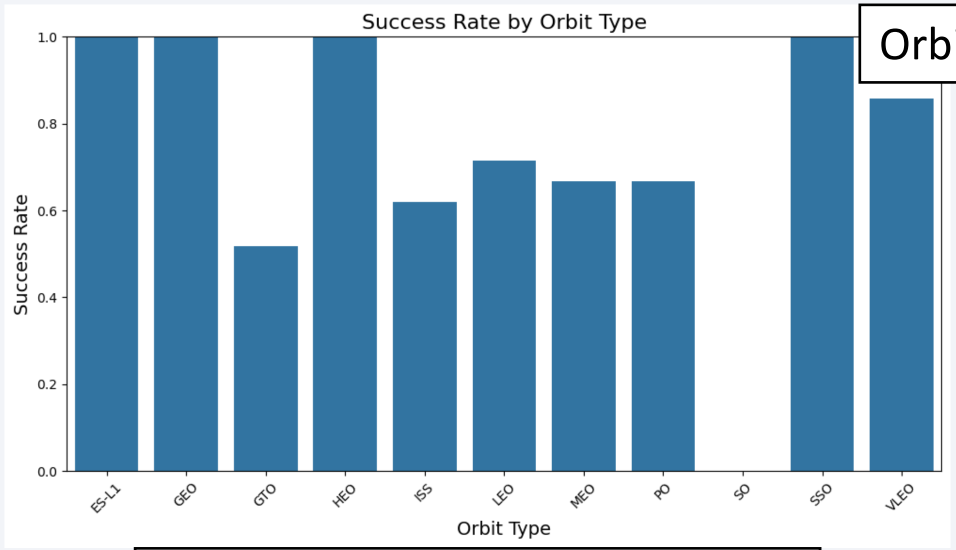
Various charts were plotted showing success (1, Orange) and Failure (0, Blue)



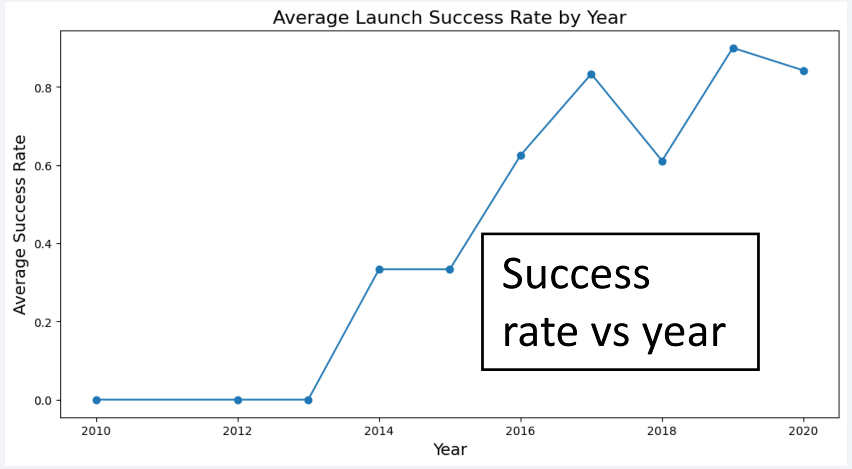
Orbit vs Flight #



Orbit vs Payload Mass



Success rate vs orbit type



Success rate vs year

EDA with SQL

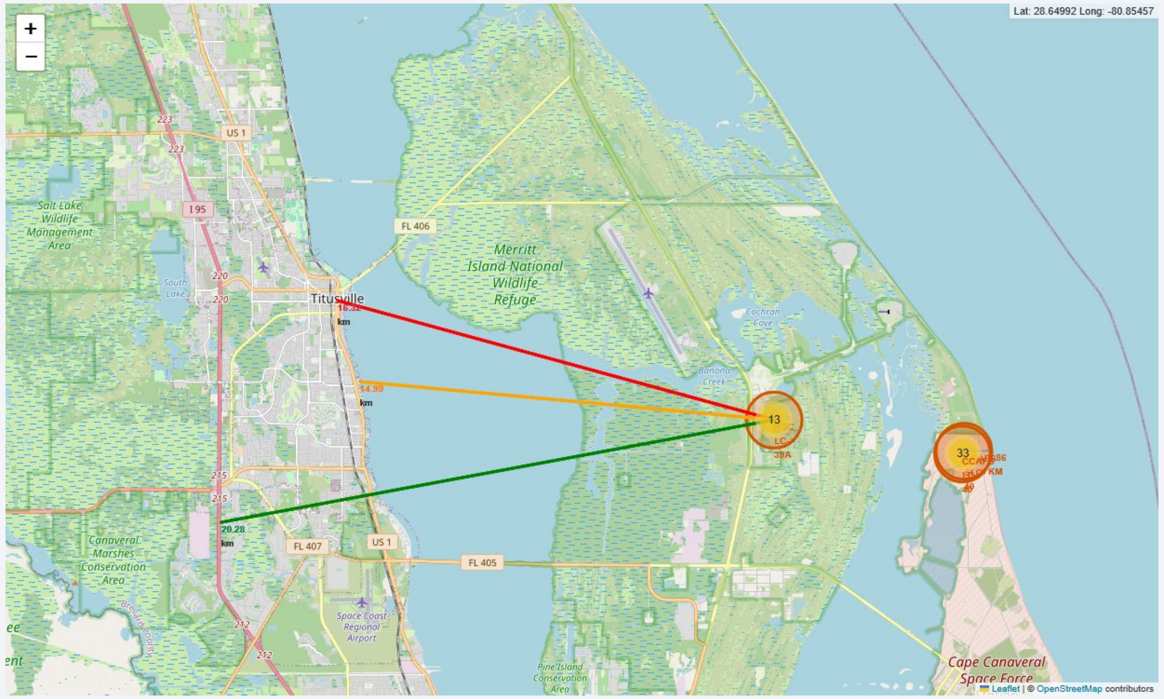
- **Github: https://github.com/iab149/IBM-DS0720EN-Data-Science-and-Machine-Learning-Capstone-Project/blob/2161480fac3234a1f32232d8febbc2dc035554d5/IBM-DS0720EN-AB-jupyter-labs-eda-sql-edx_sqllite.ipynb**
- **Display the names of the unique launch sites in the space mission**
- **Display 5 records where launch sites begin with the string 'KSC'**
- **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 where the succesful landing outcome in drone ship was achieved.**
- **List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000 kg**
- **List all the booster_versions that have carried the maximum payload mass (using subquery).**
- **List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017**
- **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. Code snippet for this query:**

  -
    ```
    query = "“”
    SELECT "Landing_Outcome", COUNT(*) AS OutcomeCount
    FROM SPACEXTABLE
    WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20’
    GROUP BY "Landing_Outcome“
    ORDER BY OutcomeCount DESC""“
    cur.execute(query)
    results = cur.fetchall()
    # Print each landing outcome with its count
    for outcome, count in results:
        print(f"{outcome}: {count}")
    ```

Build an Interactive Map with Folium

- Download and read the `spacex_launch_geo.csv`
- Select relevant sub-columns: `Launch Site`, `Lat(Latitude)`, `Long(Longitude)`, `class`
- Create a map with starting location as NASA Johnson Space Center
- Mark it with a blue circle and popup
- Create and add folium.Circle and folium.Marker for each launch site on the site map
- Mark the success/failed launches for each site on the map
- Add Mouse Position to get the coordinate (Lat, Long) for a mouse over on the map
- Calculate the distances between a launch site to its proximities
- Why? This tells us that site KSC LC 39A is:
  - 16.3 km to the city Titusville
  - 14.99 km to the mainland coastline
  - 20.28 km to the highway I95

- Github: https://github.com/iab149/IBM-DS0720EN-Data-Science-and-Machine-Learning-Capstone-Project/blob/7fe3ed1f1d76b74e5d5248dda1ad1add235ff7d5/IBM-DS0720EN-AB-lab_jupyter_launch_site_location.jupyterlite.ipynb

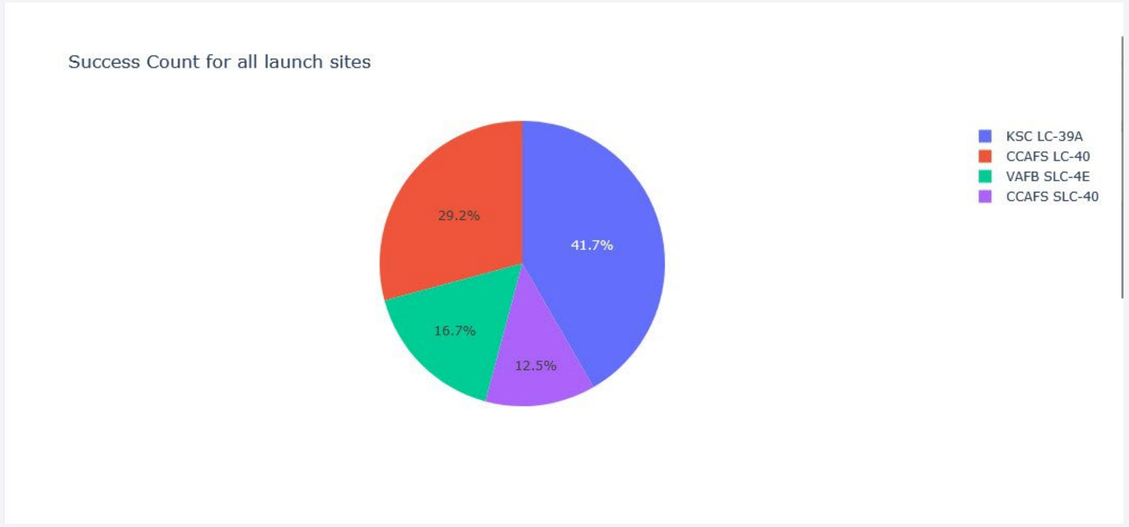


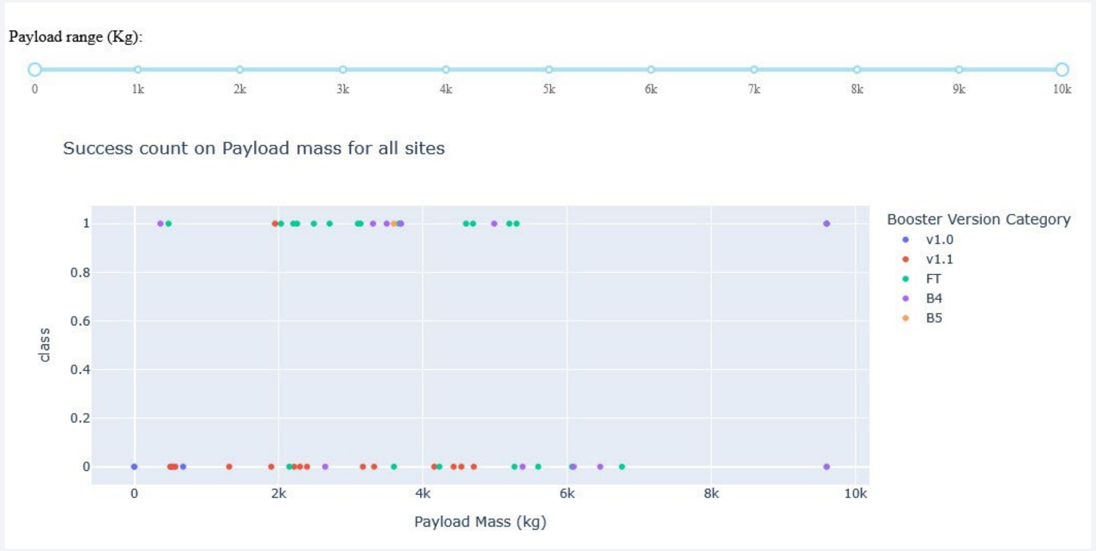
Lat: 28.64992 Long: -80.85457
Titusville
14.99 km
20.28 km
13
33
Merritt Island National Wildlife Refuge
Canaveral Marshes Conservation Area
Space Coast Regional Airport
Cape Canaveral Space Force
Leaflet | © OpenStreetMap contributors

Build a Dashboard with Plotly Dash

Purpose: enable users to perform interactive visual analytics on SpaceX launch data in real-time.

- Read data, download template or skeleton dashboard
- Task 1: Add a dropdown to select sites
- Task 2: Add a pie chart to show the total successful launches count for all sites
- Task 3: Add a slider to select payload range
- Add a scatter chart to show the correlation between payload and launch success
- Github: https://github.com/iab149/IBM-DS0720EN-Data-Science-and-Machine-Learning-Capstone-Project/blob/c89a701a8b9896d7b09363b19595c8553a4a3ba4/IBM-DS0720EN-AB-spacex-dash-app.py

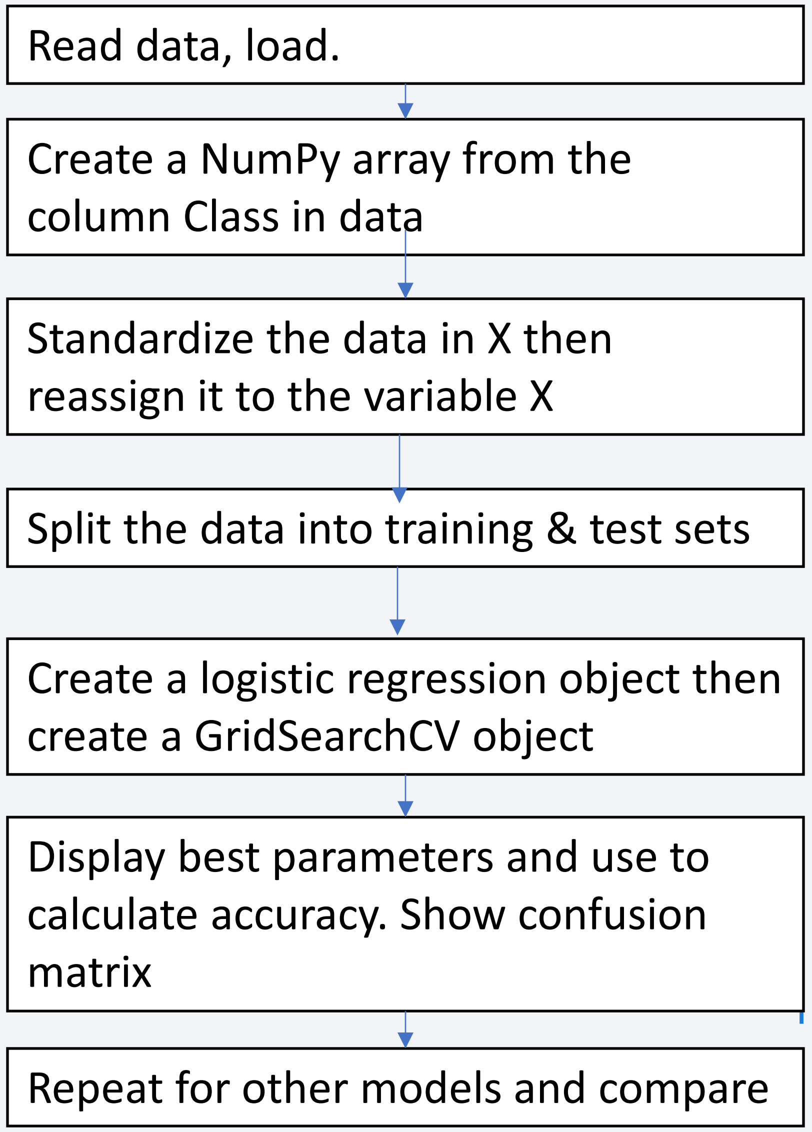




Predictive Analysis (Classification)

- Motivation: use data from previous labs to build a model which will predict whether the Falcon 9 1st stage will successfully land.
- Load data, standardize it. Use the column “Class” as the output variable Y. Use other variables as input. Call it X.
- Split into training and test datasets.
- Using the following models, find best parameters, calculate accuracy, show confusion matrix.
  - Logistic regression
  - Support vector machine
  - Decision tree classifier
  - k nearest neighbors
- Flowchart is shown:

- Github: https://github.com/iab149/IBM-DS0720EN-Data-Science-and-Machine-Learning-Capstone-Project/blob/c994afb29cdde8e281151c4d0bff16ba3d0cdb0d/IBM-DS0720EN-AB-SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb



Read data, load.

Create a NumPy array from the column Class in data

Standardize the data in X then reassign it to the variable X

Split the data into training & test sets

Create a logistic regression object then create a GridSearchCV object

Display best parameters and use to calculate accuracy. Show confusion matrix

Repeat for other models and compare

Results

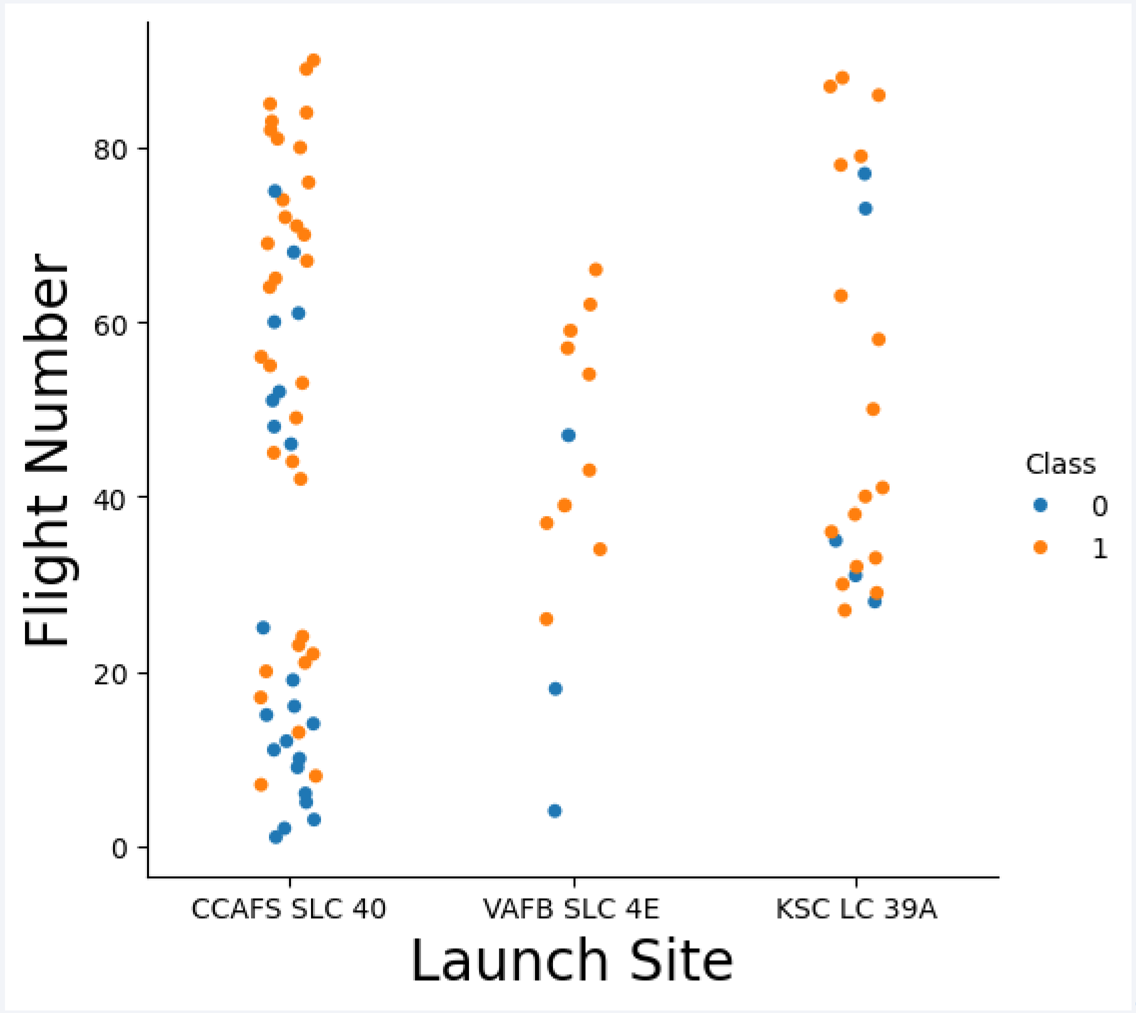
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

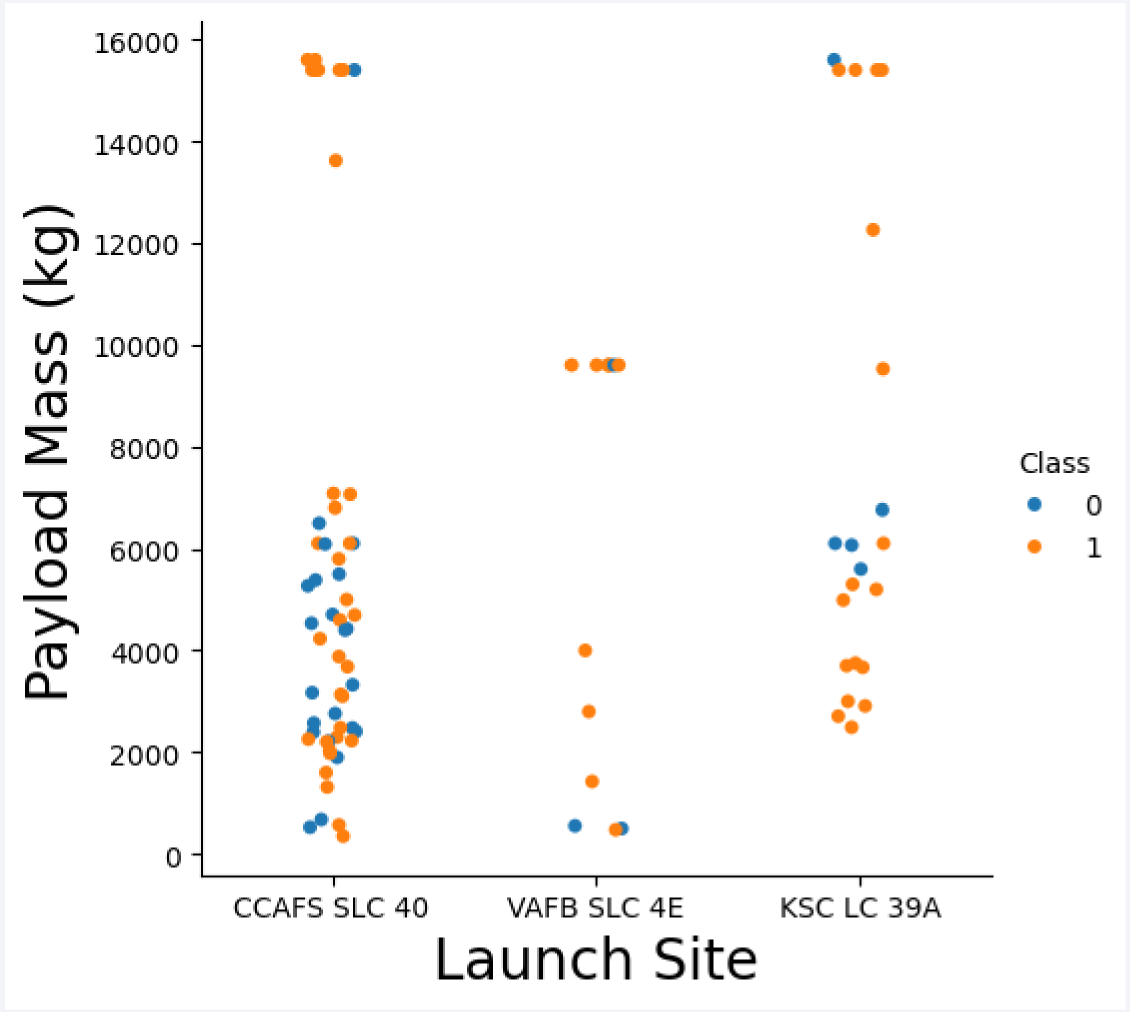
Flight Number vs. Launch Site

- Site CCAFS SLC40 is most often used
- Sites KSC LC39A and VAFB SLC4E are more often successful than site CCAS SLC40
- Success increases with later flight #s



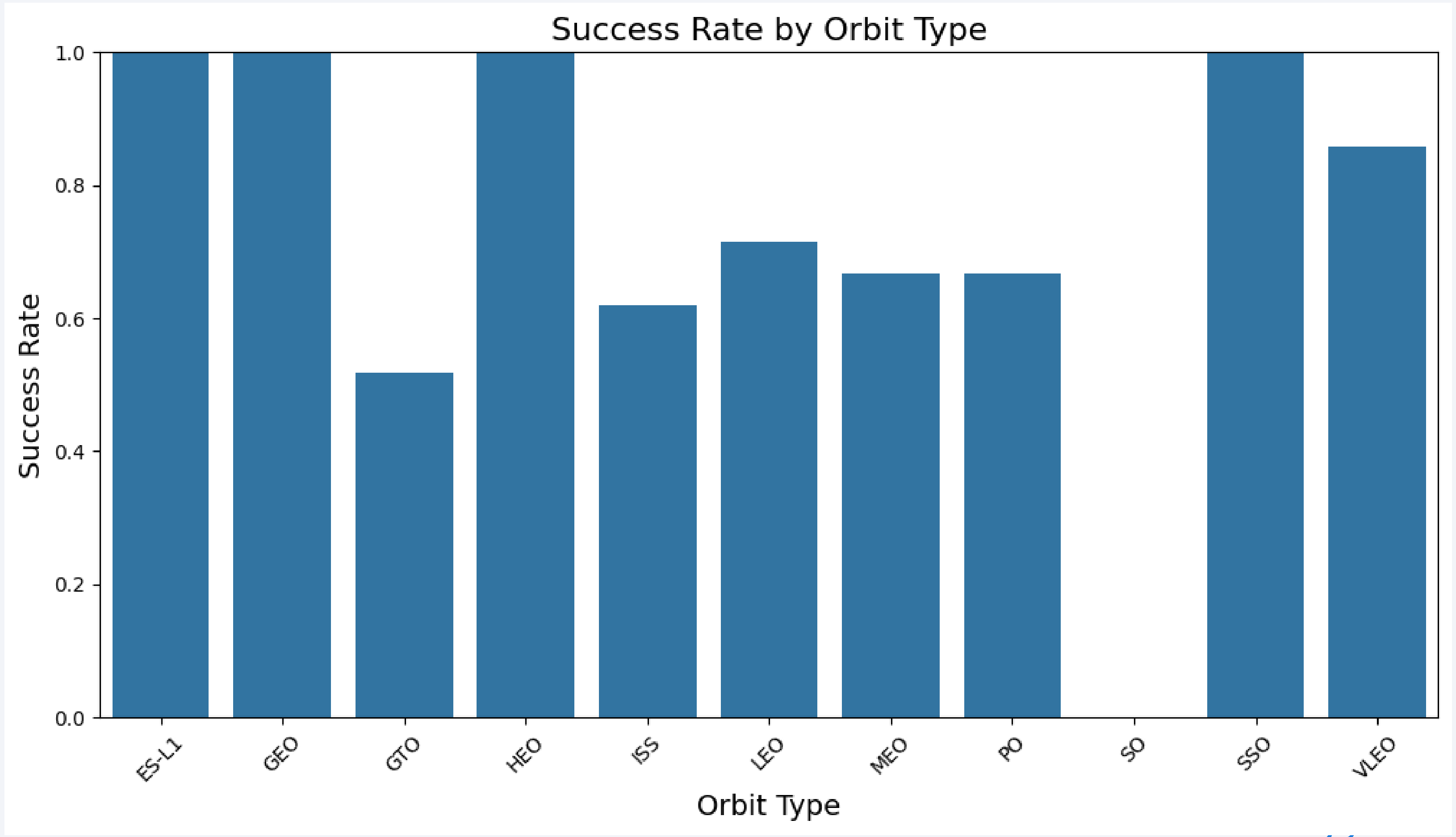
Payload vs. Launch Site

- Most launches with payload > 8000 kg have been successful
- Site KSC LC 39A has been 100% successful for launch payloads < 4000 kg
- Site VAFB SLC4E has not been used for heavy payload missions (> 10000 kg)



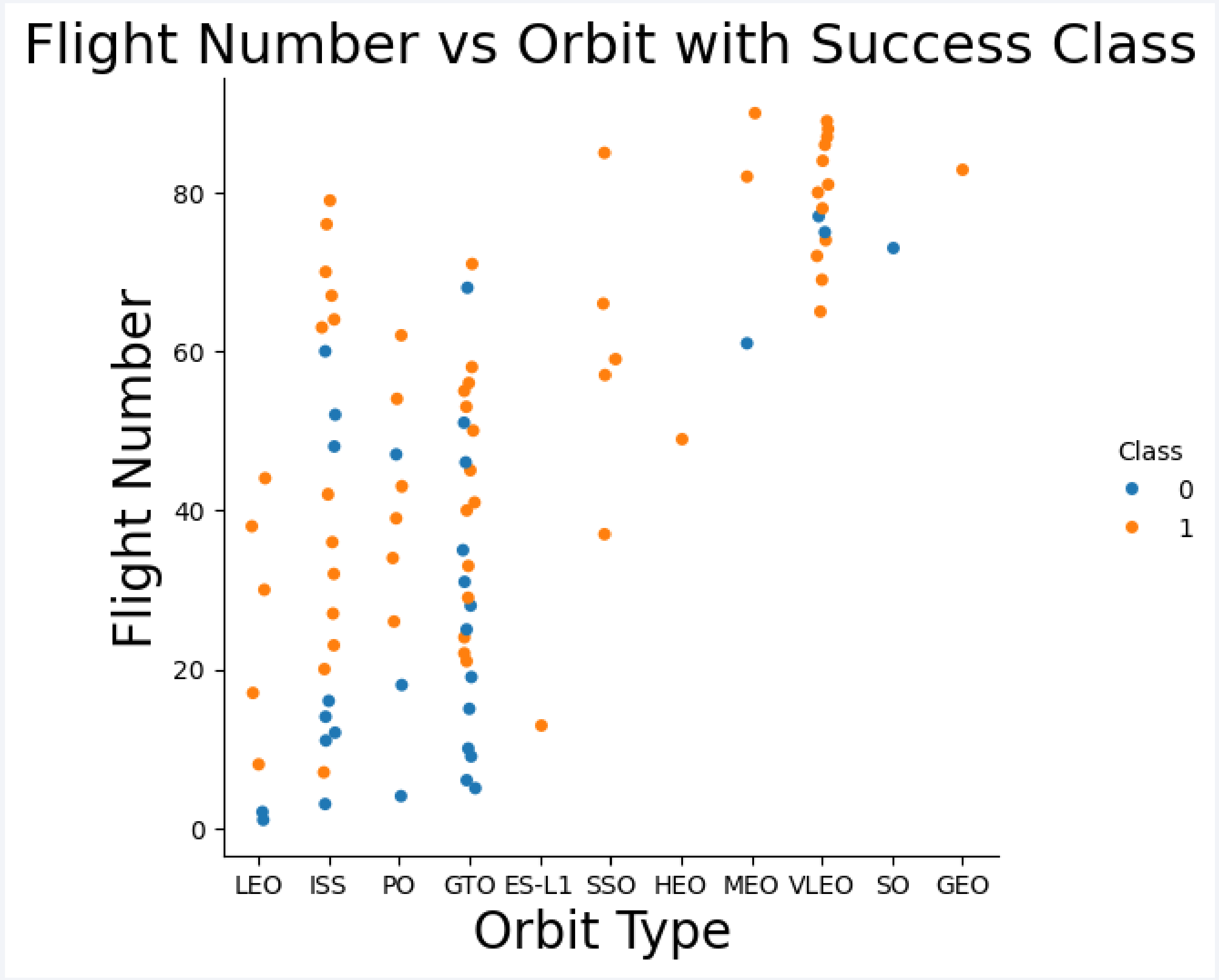
Success Rate vs. Orbit Type

- Orbits ES-L-1, GEO, HEO, SSO and VLEO are 100% successful
- Orbits GTO, ISS, LEO, MEO, PO and VLEO have between 50% and 90% success rates
- The SO orbit has been a complete failure with 0% success rate



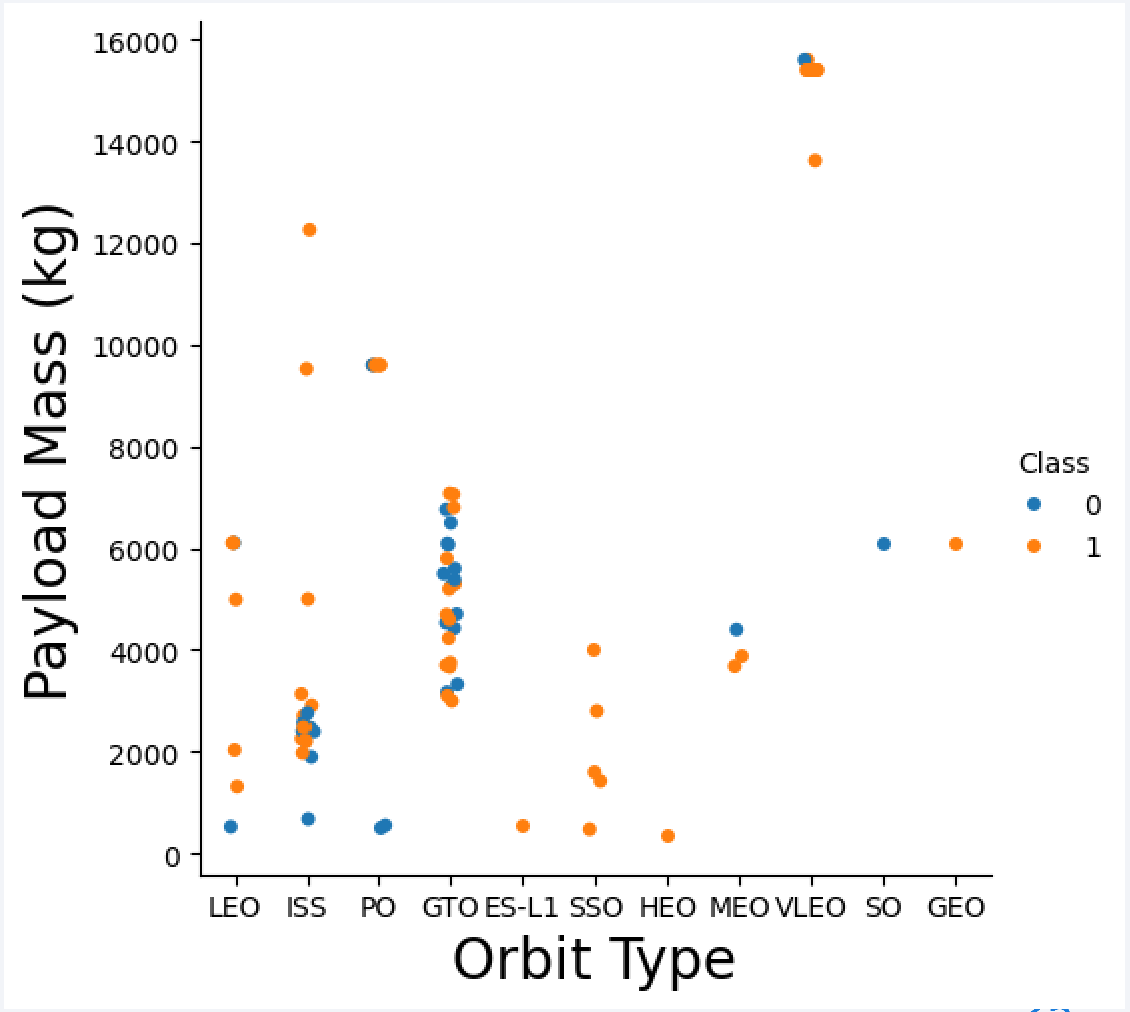
Flight Number vs. Orbit Type

- Most of the flights were to the ISS or GTO orbits
- ISS, GTO, LEO orbits have been the mission profiles from the start of Falcon 9 use.
- Later mission profiles (flight #s) have included VLEO orbits as well.



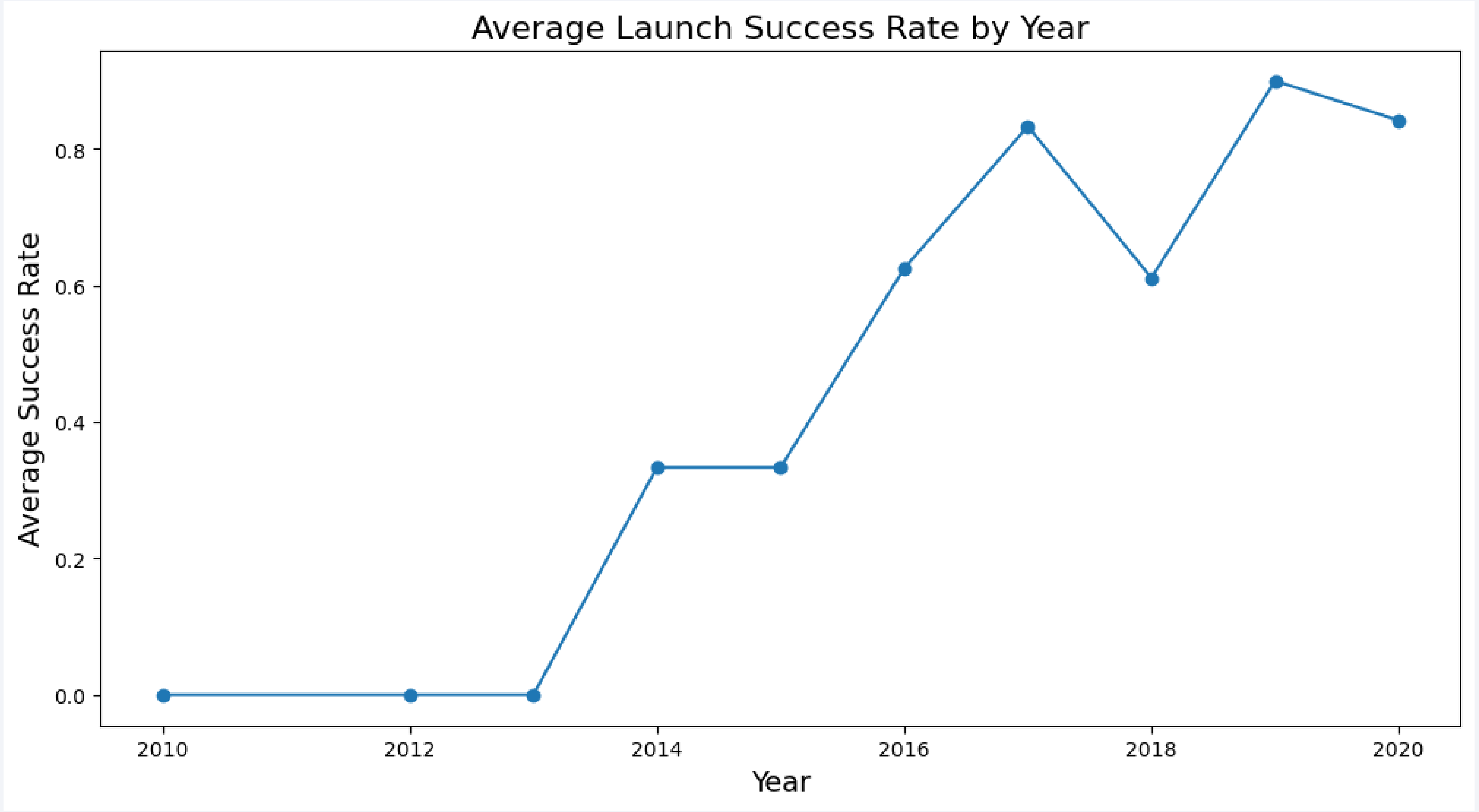
Payload vs. Orbit Type

- Payloads above 14000 kg are sent to VLEO
- Payloads between 8000 and 14000 kg have been sent to ISS, PO and VLEO
- Payloads between 4000 and 8000 kg mostly go to GTO
- Payloads below 4000 kg mostly go to ISS



Launch Success Yearly Trend

- Data is available starting 2010
- The years between 2010 and 2013 were marked by failures
- From 2013 to 2020, there has been a marked linear improvement with success rates around 80% in 2020
- There is a dip in success rate to 60% in 2018



All Launch Site Names

```
('CCAFS LC-40',)
('VAFB SLC-4E',)
('KSC LC-39A',)
('CCAFS SLC-40',)
```

```
# the equivalent sql command is given below. I am using
python since %sql did not work
# %sql SELECT DISTINCT "LaunchSite" FROM SPACEXTABLE;



query = """
    SELECT DISTINCT launch_site
    FROM SPACEXTABLE
    """


cur.execute(query)
items = cur.fetchall()

for i in items:
    print(i)
```

Launch Site Names Begin with 'KSC'

```
('2017-02-19', '14:39:00', 'F9 FT B1031.1', 'KSC LC-39A', 'SpaceX
CRS-10', 2490, 'LEO (ISS)', 'NASA (CRS)', 'Success', 'Success
(ground pad)')

-------------------------------------------------
('2017-03-16', '6:00:00', 'F9 FT B1030', 'KSC LC-39A', 'EchoStar
23', 5600, 'GTO', 'EchoStar', 'Success', 'No attempt')

-------------------------------------------------
('2017-03-30', '22:27:00', 'F9 FT  B1021.2', 'KSC LC-39A', 'SES-
10', 5300, 'GTO', 'SES', 'Success', 'Success (drone ship)')

-------------------------------------------------
('2017-05-01', '11:15:00', 'F9 FT B1032.1', 'KSC LC-39A', 'NROL-
76', 5300, 'LEO', 'NRO', 'Success', 'Success (ground pad)')

-------------------------------------------------
('2017-05-15', '23:21:00', 'F9 FT B1034', 'KSC LC-39A',
'Inmarsat-5 F4', 6070, 'GTO', 'Inmarsat', 'Success', 'No attempt')
```

```
query = """
    SELECT *
    FROM SPACEXTABLE
    WHERE "Launch_Site" LIKE 'KSC%'
    LIMIT 5
    """

cur.execute(query)
items = cur.fetchall()

for i in items:
    print(f"{i}\n")
```

Total Payload Mass

```
Now let us focus on the NASA
(CRS) lines only and sum the
payloads
Total Payload Mass for NASA CRS
missions: 48213 kg
```

```
# Now let us focus on the NASA (CRS) lines only and sum the payloads
print("Now let us focus on the NASA (CRS) lines only and sum the payloads")
query = """
SELECT SUM("PAYLOAD_MASS__KG_") AS TotalPayloadMass
FROM SPACEXTABLE
WHERE "Customer" LIKE '%NASA (CRS)%'
"""
cur.execute(query)
result = cur.fetchone()
print(f"Total Payload Mass for NASA CRS missions: {result[0]} kg")
```

Average Payload Mass by F9 v1.1

```
Average Payload Mass for F9 v1.1: 2928.4 kg
```

```
query = """
SELECT AVG("PAYLOAD_MASS__KG_") AS AveragePayloadMass
FROM SPACEXTABLE
WHERE "Booster_Version" = 'F9 v1.1'
"""
cur.execute(query)
result = cur.fetchone()
print(f"Average Payload Mass for F9 v1.1: {result[0]} kg")
```

First Successful Ground Landing Date

```
query = """
SELECT "Date"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE  '%Success (drone ship)%
""""
cur.execute(query)
dates = cur.fetchall()
# Print each matching date
for d in dates:
                print(d[0])
print(f"The earliest such succesful outcome was in:\n{min(dates)}")
```

```
The earliest such succesful outcome was in:
('2016-04-08',)
```

Successful Drone Ship Landing with Payload between 4000 and 6000

```
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1
```

```
query = """
SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE '%Success (ground pad)%'
  AND "PAYLOAD_MASS__KG_" > 4000
  AND "PAYLOAD_MASS__KG_" < 6000
"""
cur.execute(query)
boosters = cur.fetchall()

# Print each matching booster version
for b in boosters:
    print(b[0])
```

Total Number of Successful and Failure Mission Outcomes

```
Failure (in flight): 1
Success: 98
Success : 1
Success (payload status unclear): 1
```

```
query = """
SELECT "Mission_Outcome", COUNT(*) AS OutcomeCount
FROM SPACEXTABLE
GROUP BY "Mission_Outcome"
"""
cur.execute(query)
outcomes = cur.fetchall()

# Print each outcome and its count
for outcome in outcomes:
    print(f"{outcome[0]}: {outcome[1]}")
```

Boosters Carried Maximum Payload

```
Booster Version: F9 B5 B1048.4, Payload Mass: 15600 kg
Booster Version: F9 B5 B1049.4, Payload Mass: 15600 kg
Booster Version: F9 B5 B1051.3, Payload Mass: 15600 kg
Booster Version: F9 B5 B1056.4, Payload Mass: 15600 kg
Booster Version: F9 B5 B1048.5, Payload Mass: 15600 kg
Booster Version: F9 B5 B1051.4, Payload Mass: 15600 kg
Booster Version: F9 B5 B1049.5, Payload Mass: 15600 kg
Booster Version: F9 B5 B1060.2 , Payload Mass: 15600 kg
Booster Version: F9 B5 B1058.3 , Payload Mass: 15600 kg
Booster Version: F9 B5 B1051.6, Payload Mass: 15600 kg
Booster Version: F9 B5 B1060.3, Payload Mass: 15600 kg
Booster Version: F9 B5 B1049.7 , Payload Mass: 15600 kg
```

```
query = """
SELECT DISTINCT "Booster_Version", "PAYLOAD_MASS__KG_"
FROM SPACEXTABLE
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_")
    FROM SPACEXTABLE
)
"""
cur.execute(query)
results = cur.fetchall()

# Print each booster version with its payload mass
for booster, payload in results:
    print(f"Booster Version: {booster}, Payload Mass: {payload}
kg")
```

2017 Launch Records

```
Month: 02, Landing Outcome: Success (ground pad), Booster
Version: F9 FT B1031.1, Launch Site: LaunchSite
Month: 05, Landing Outcome: Success (ground pad), Booster
Version: F9 FT B1032.1, Launch Site: LaunchSite
Month: 06, Landing Outcome: Success (ground pad), Booster
Version: F9 FT B1035.1, Launch Site: LaunchSite
Month: 08, Landing Outcome: Success (ground pad), Booster
Version: F9 B4 B1039.1, Launch Site: LaunchSite
Month: 09, Landing Outcome: Success (ground pad), Booster
Version: F9 B4 B1040.1, Launch Site: LaunchSite
Month: 12, Landing Outcome: Success (ground pad), Booster
Version: F9 FT  B1035.2, Launch Site: LaunchSite
```

```
query = """
SELECT
    substr("Date", 6, 2) AS Month,
    "Landing_Outcome",
    "Booster_Version",
    "LaunchSite"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE '%Success (ground pad)%'
  AND substr("Date", 1, 4) = '2017'
"""
cur.execute(query)
records = cur.fetchall()

# Print each matching record
for r in records:
    print(f"Month: {r[0]}, Landing Outcome: {r[1]}, Booster
Version: {r[2]}, Launch Site: {r[3]}")
```

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

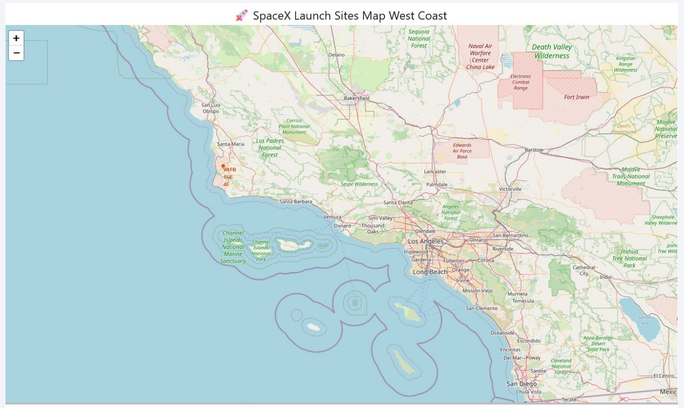
```
No attempt: 10
Success (drone ship): 5
Failure (drone ship): 5
Success (ground pad): 3
Controlled (ocean): 3
Uncontrolled (ocean): 2
Failure (parachute): 2
Precluded (drone ship): 1
```

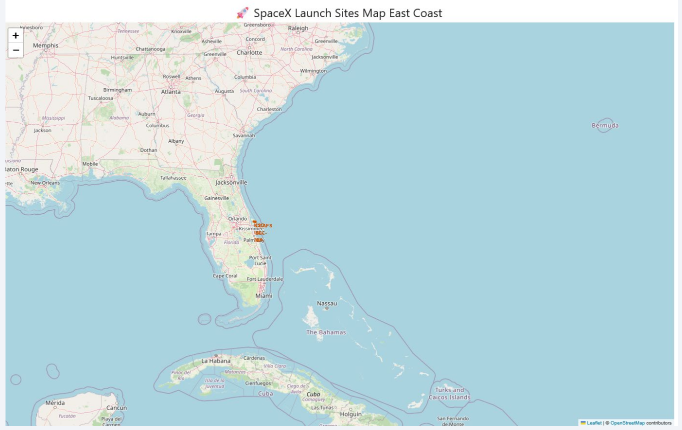
```
query = """
SELECT "Landing_Outcome", COUNT(*) AS OutcomeCount
FROM SPACEXTABLE
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY OutcomeCount DESC
"""
cur.execute(query)
results = cur.fetchall()

# Print each landing outcome with its count
for outcome, count in results:
    print(f"{outcome}: {count}")
```

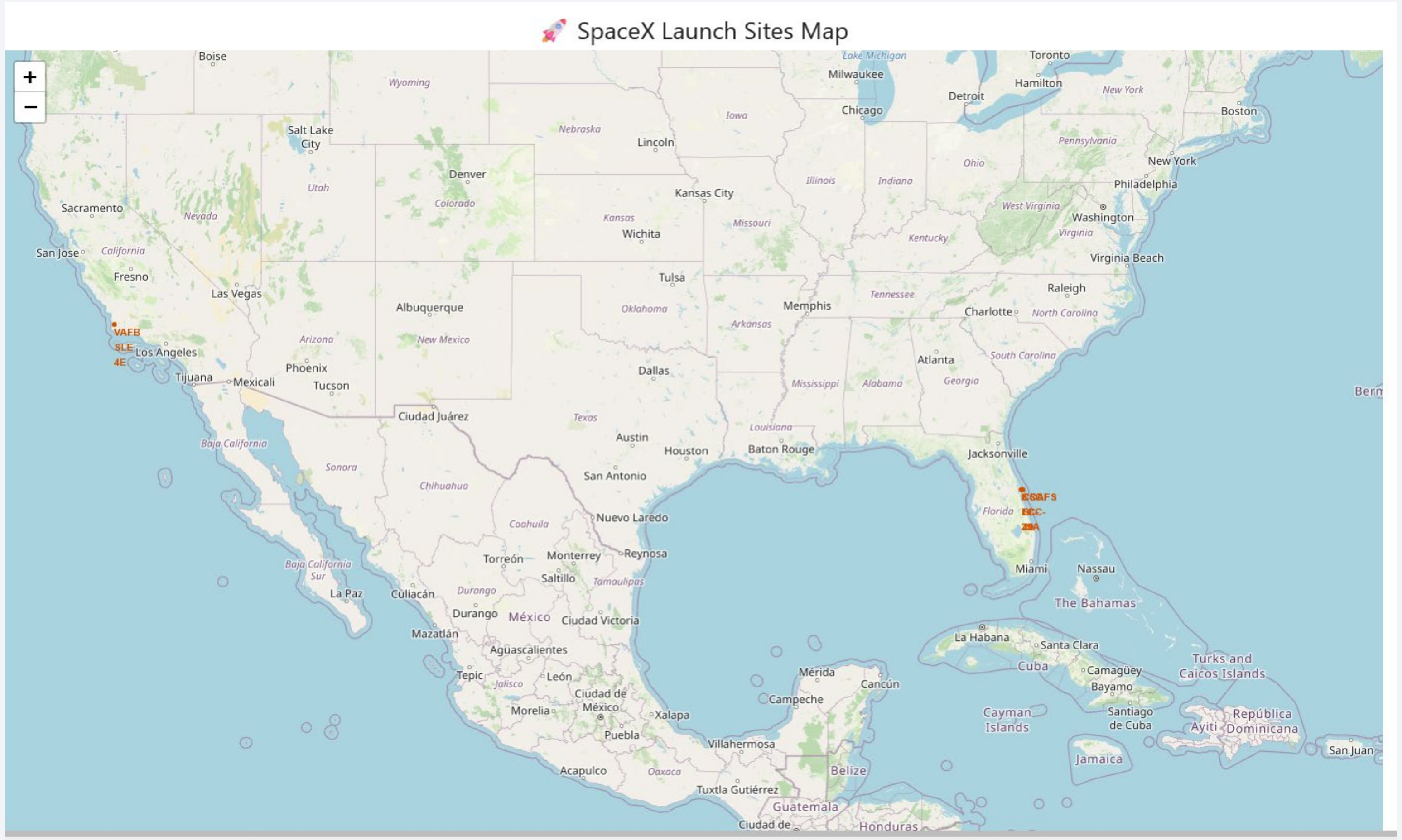
Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>



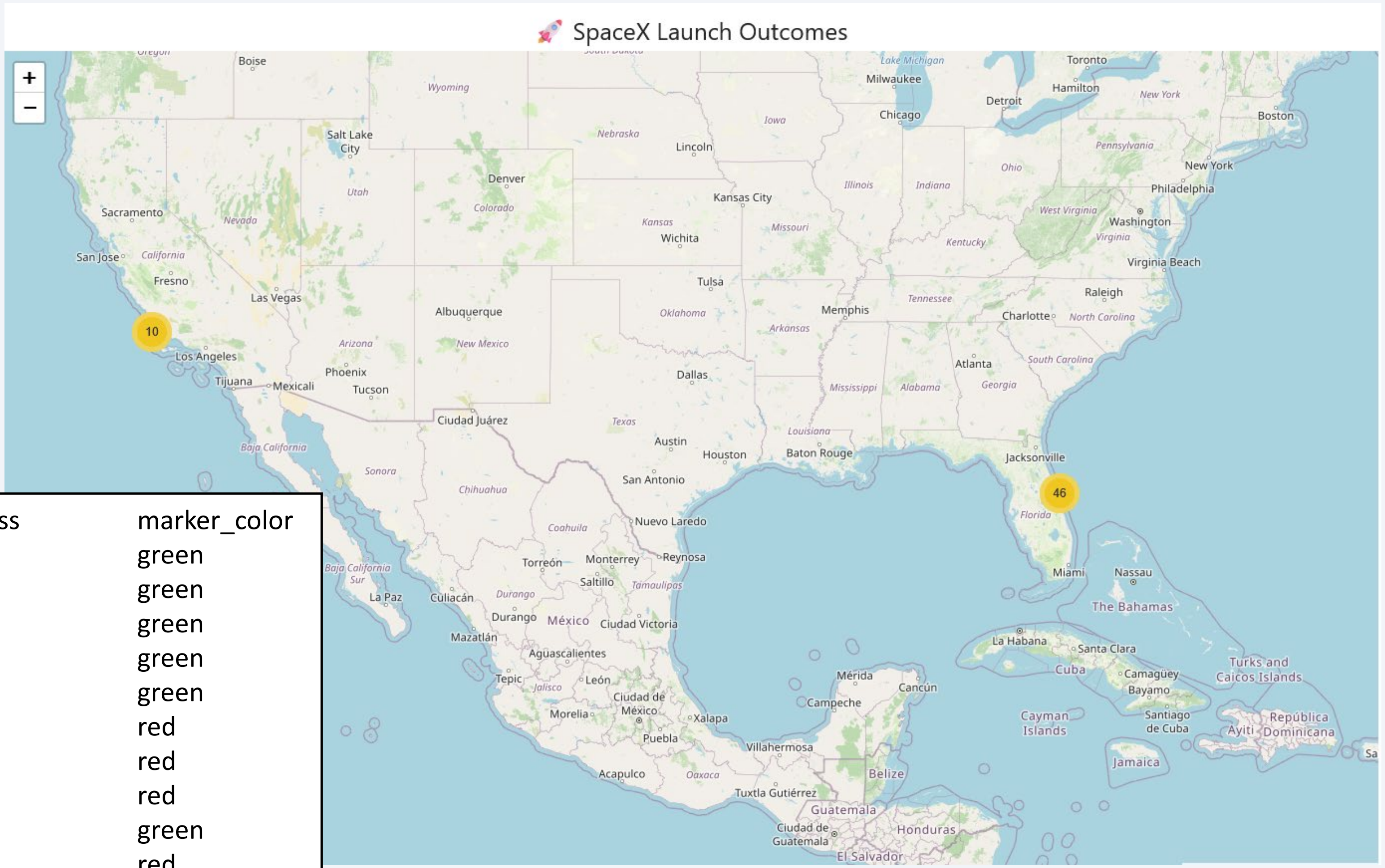


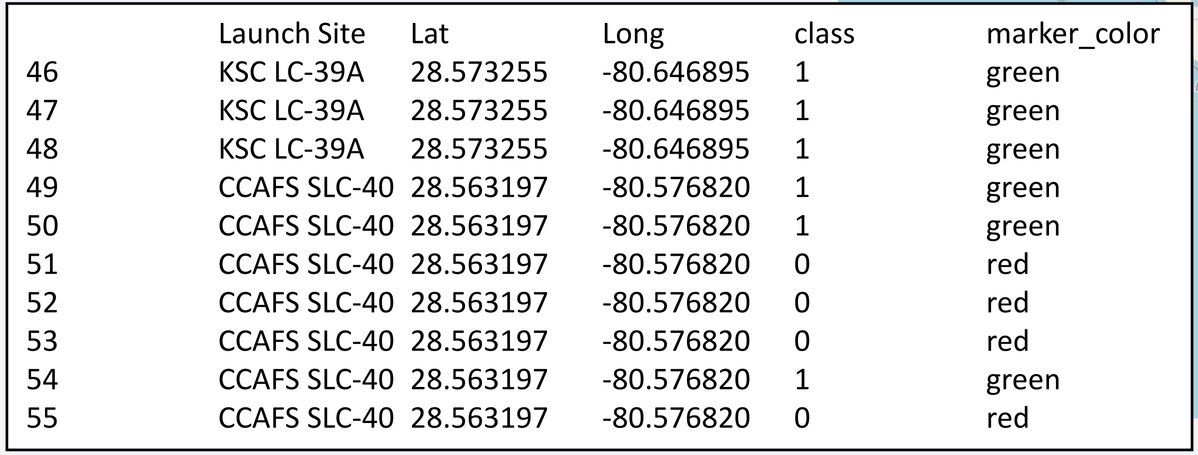


- Two locations are chosen: on East and West Coast of the US
- Both locations are close to the equator – this helps in the tangential velocity vector at launch

- Both locations are also proximal to the coast to allow splashdown of debris into the ocean in case of catastrophic failure

<Folium Map Screenshot 2>



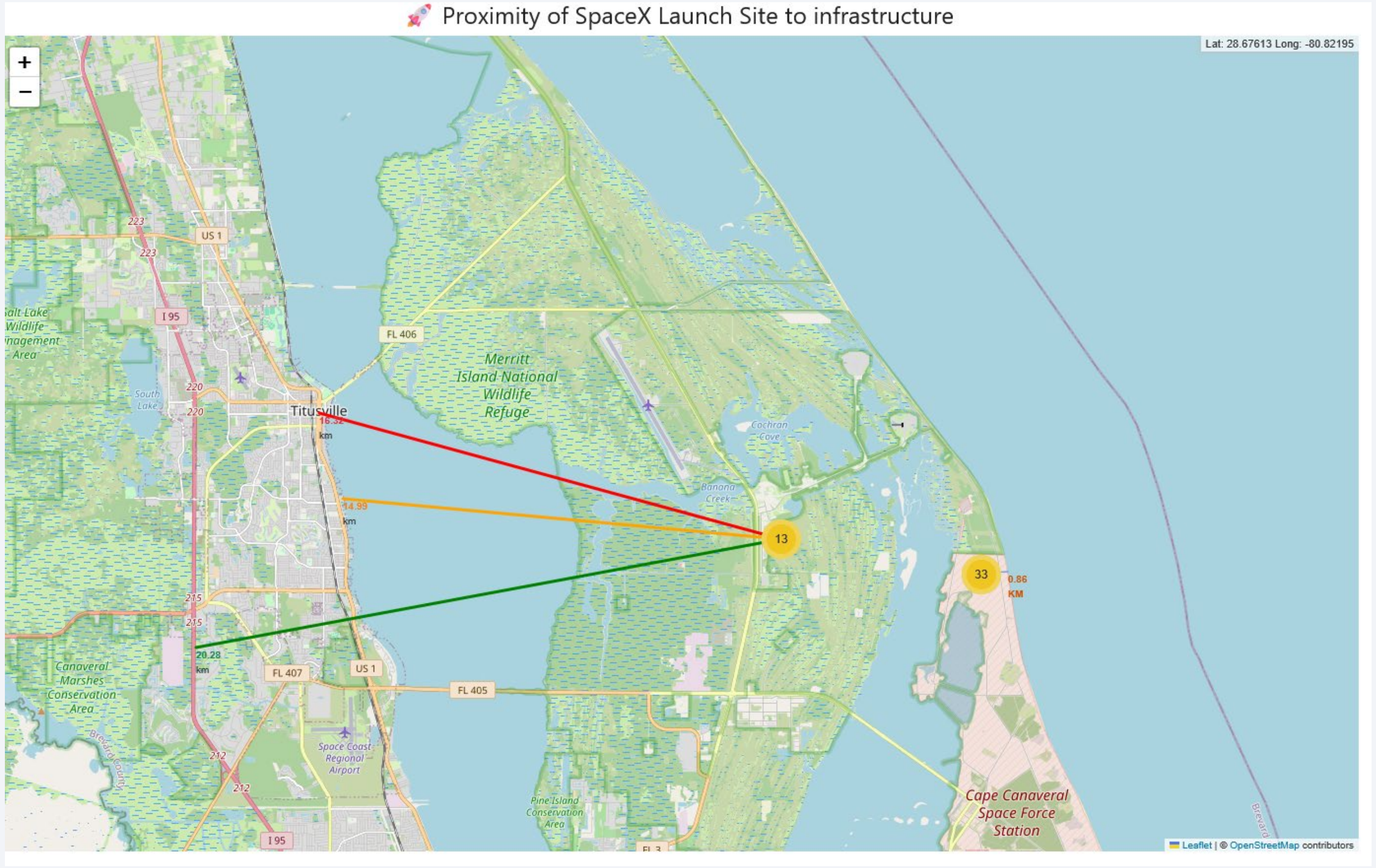


	Launch Site	Lat	Long	class	marker_color
46	KSC LC-39A	28.573255	-80.646895	1	green
47	KSC LC-39A	28.573255	-80.646895	1	green
48	KSC LC-39A	28.573255	-80.646895	1	green
49	CCAFS SLC-40	28.563197	-80.576820	1	green
50	CCAFS SLC-40	28.563197	-80.576820	1	green
51	CCAFS SLC-40	28.563197	-80.576820	0	red
52	CCAFS SLC-40	28.563197	-80.576820	0	red
53	CCAFS SLC-40	28.563197	-80.576820	0	red
54	CCAFS SLC-40	28.563197	-80.576820	1	green
55	CCAFS SLC-40	28.563197	-80.576820	0	red

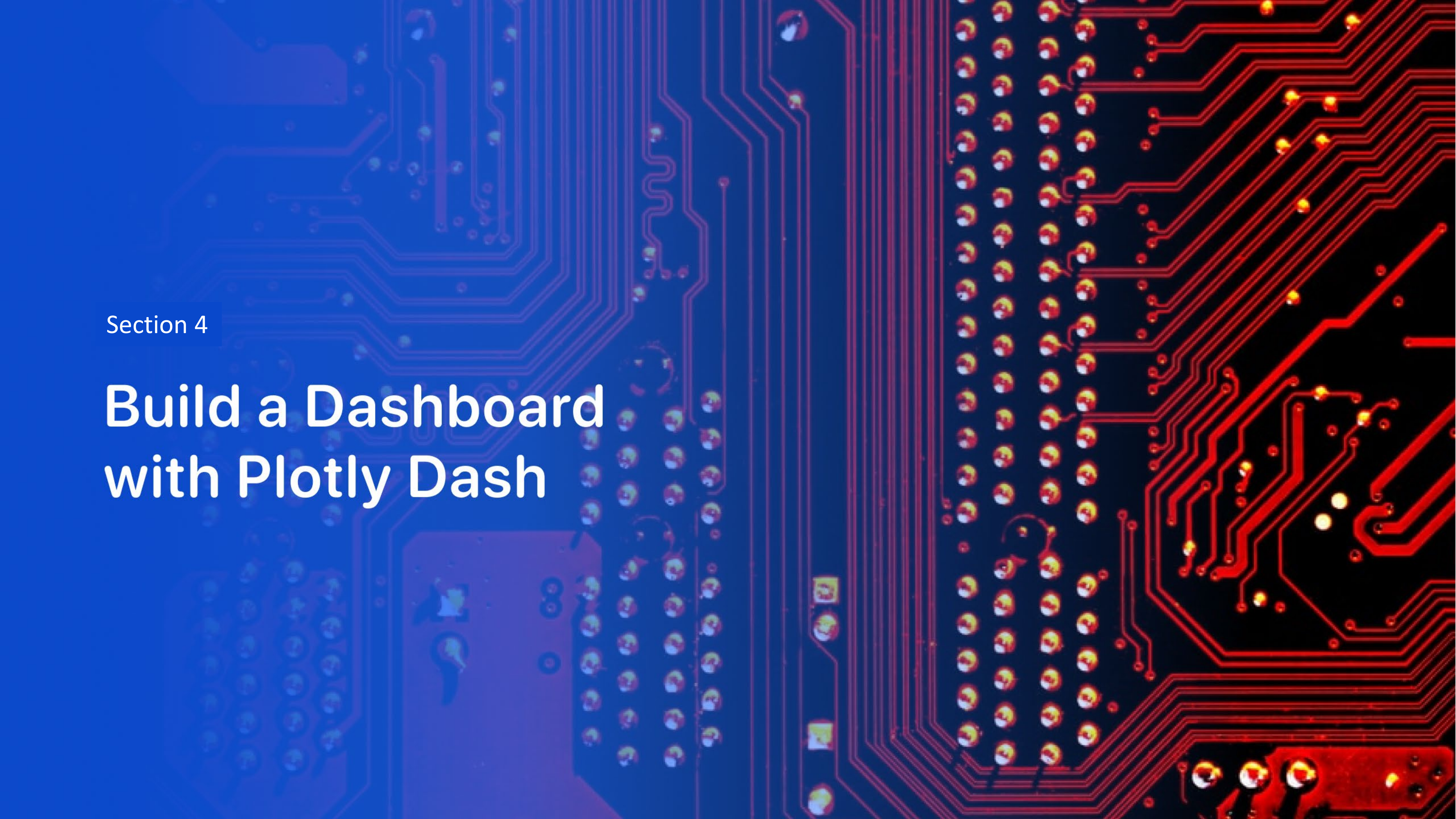
<Folium Map Screenshot 3>

This tells us that site KSC LC 39A is:

- 16.3 km to the city Titusville
- 14.99 km to the mainland coastline
- 20.28 km to the highway I95



🚀 Proximity of SpaceX Launch Site to infrastructure
Lat: 28.67613 Long: -80.82195
Titusville
16.32 km
14.99 km
20.28 km
13
33
0.86 KM
Merritt Island National Wildlife Refuge
Cape Canaveral Space Force Station
Canaveral Marshes Conservation Area
Space Coast Regional Airport
Leaflet | © OpenStreetMap contributors



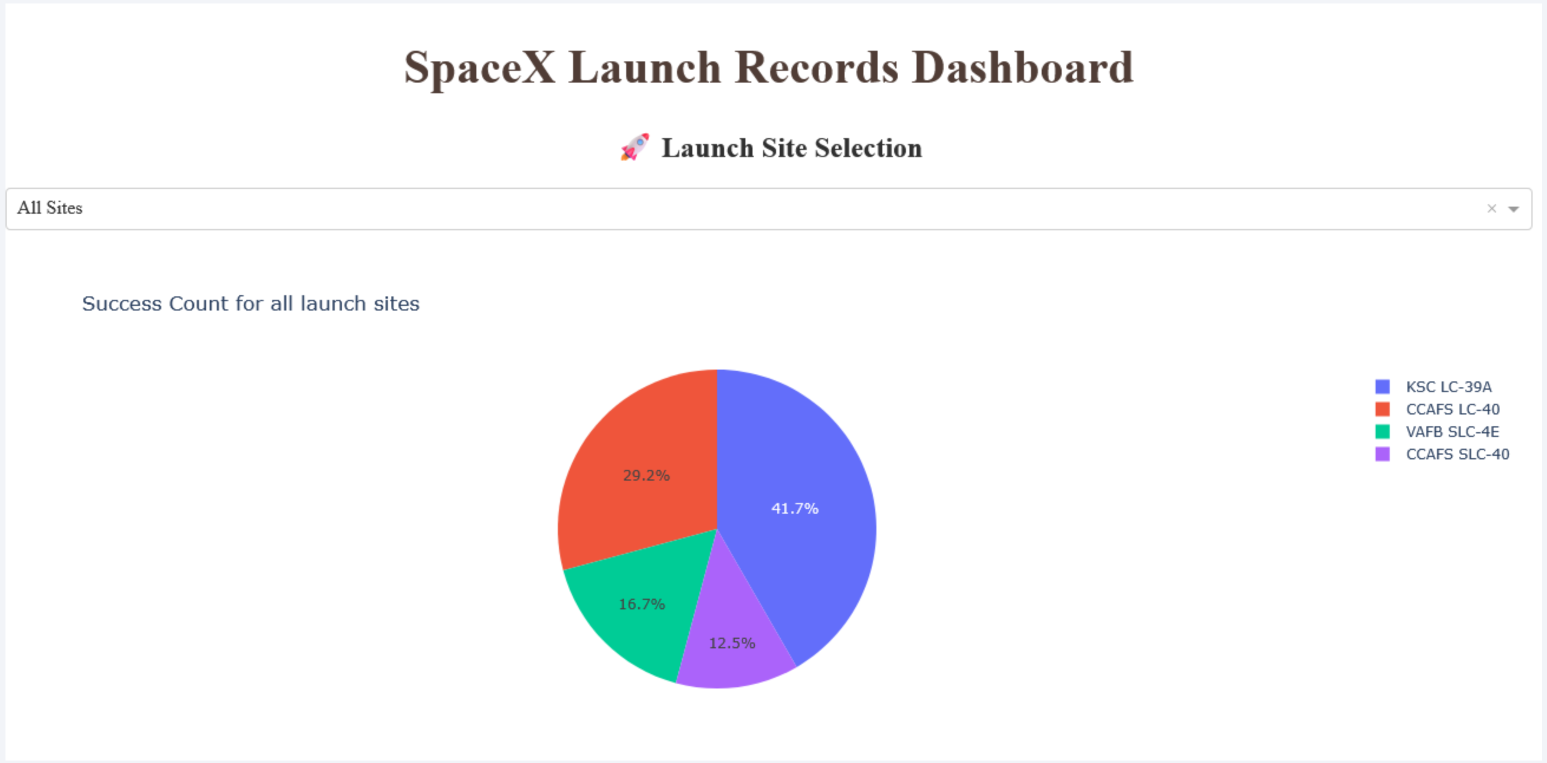
Section 4

Build a Dashboard with Plotly Dash

<Dashboard Screenshot 1>

Most successful launch sites:

1. KSC LC-39A: 41.7%
2. CCAFS LC-40: 29.2%


SpaceX Launch Records Dashboard

🚀 Launch Site Selection

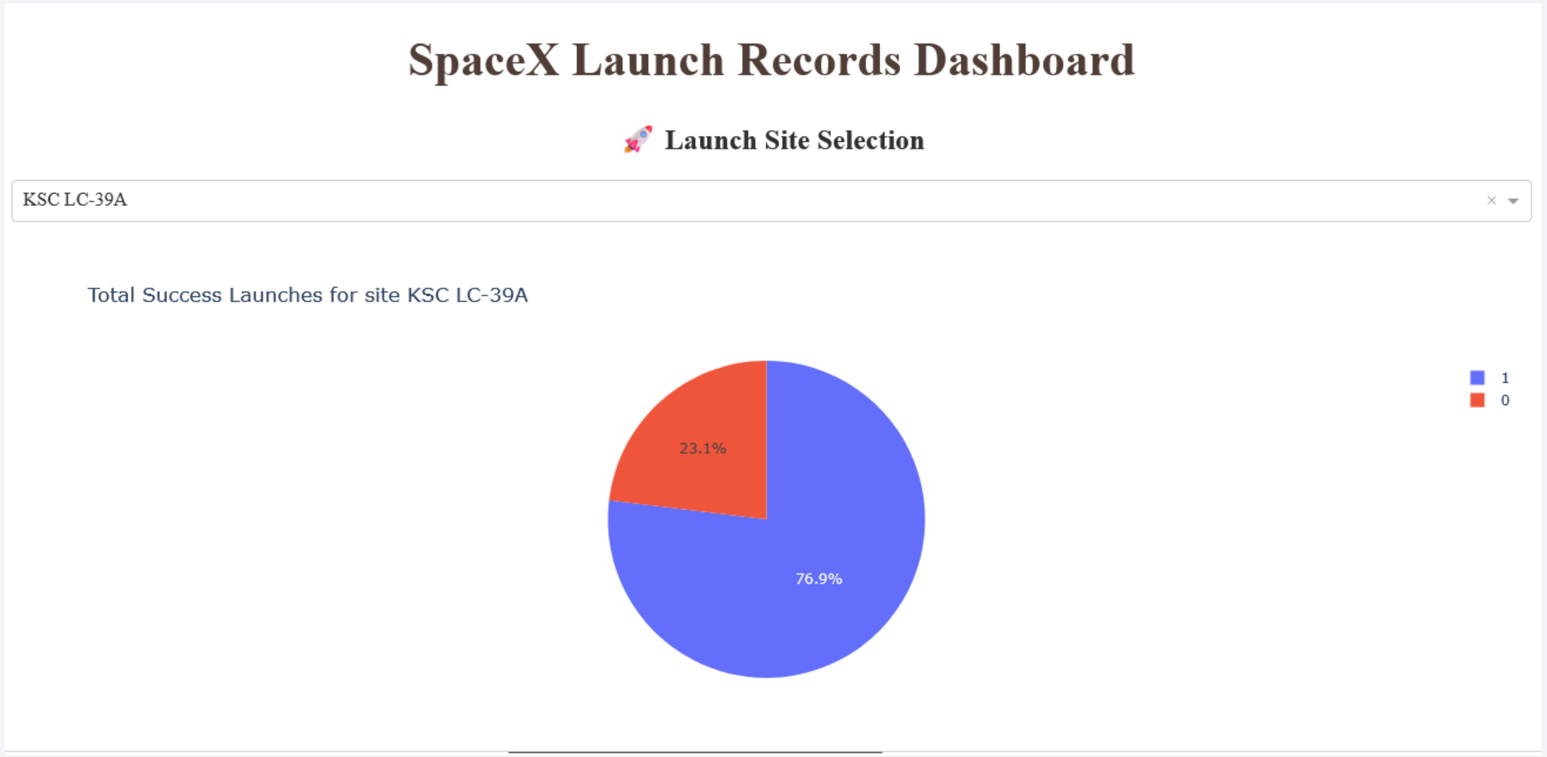
All Sites

Success Count for all launch sites

<Dashboard Screenshot 2>

KSC LC-39A launch site:

1. Success: 76.9%
2. Failure: 23.1%


SpaceX Launch Records Dashboard

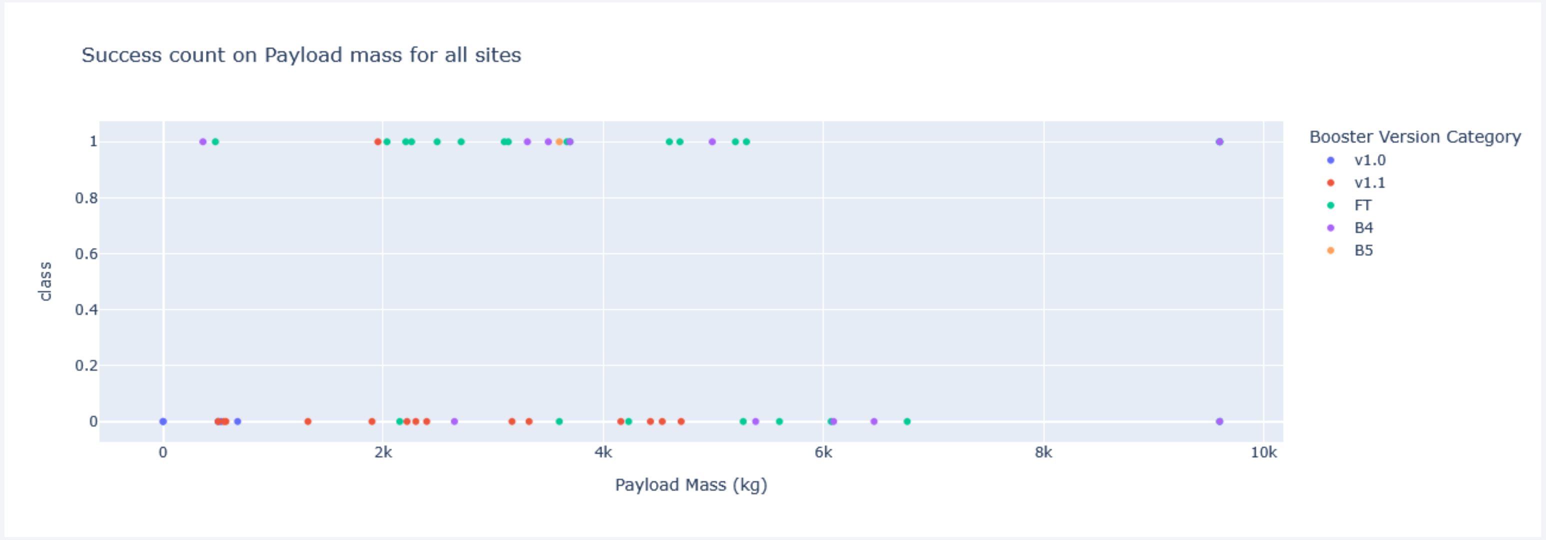
🚀 Launch Site Selection

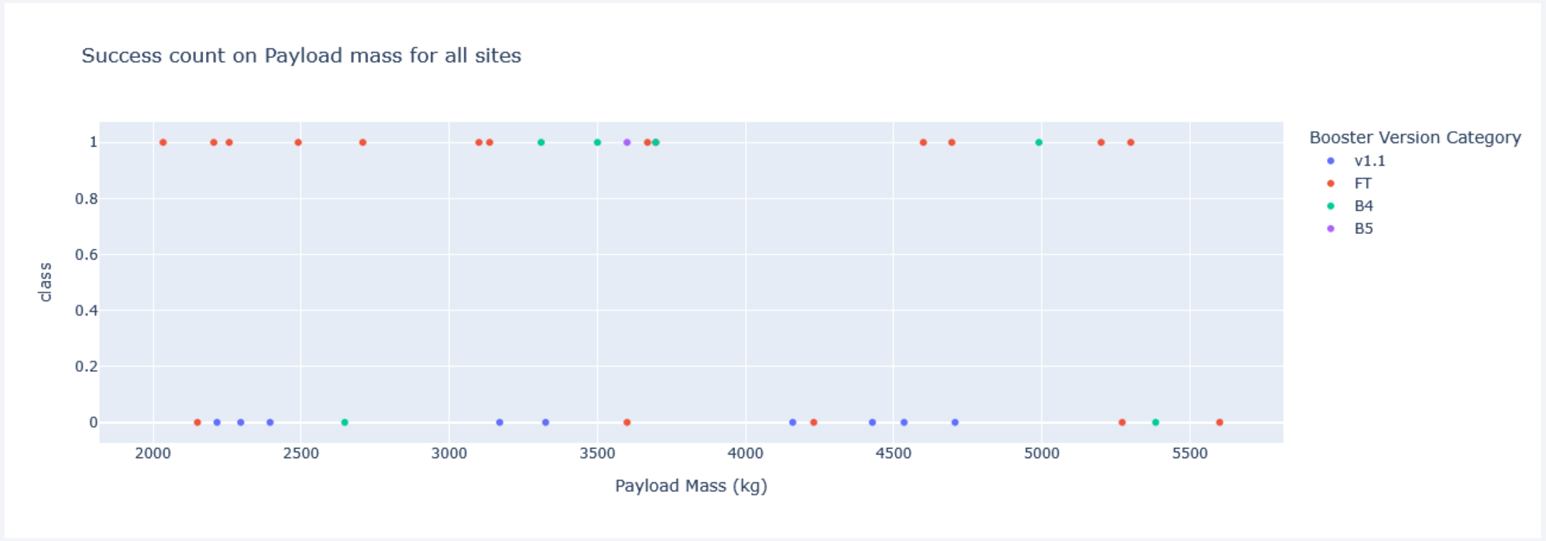
KSC LC-39A

Total Success Launches for site KSC LC-39A

<Dashboard Screenshot 3>

- Most of the successful launches have payloads between 2000 and 5500 kg
- This is shown in the zoomed plot below
- Most of the successful launches are also with the FT booster category (green marker)





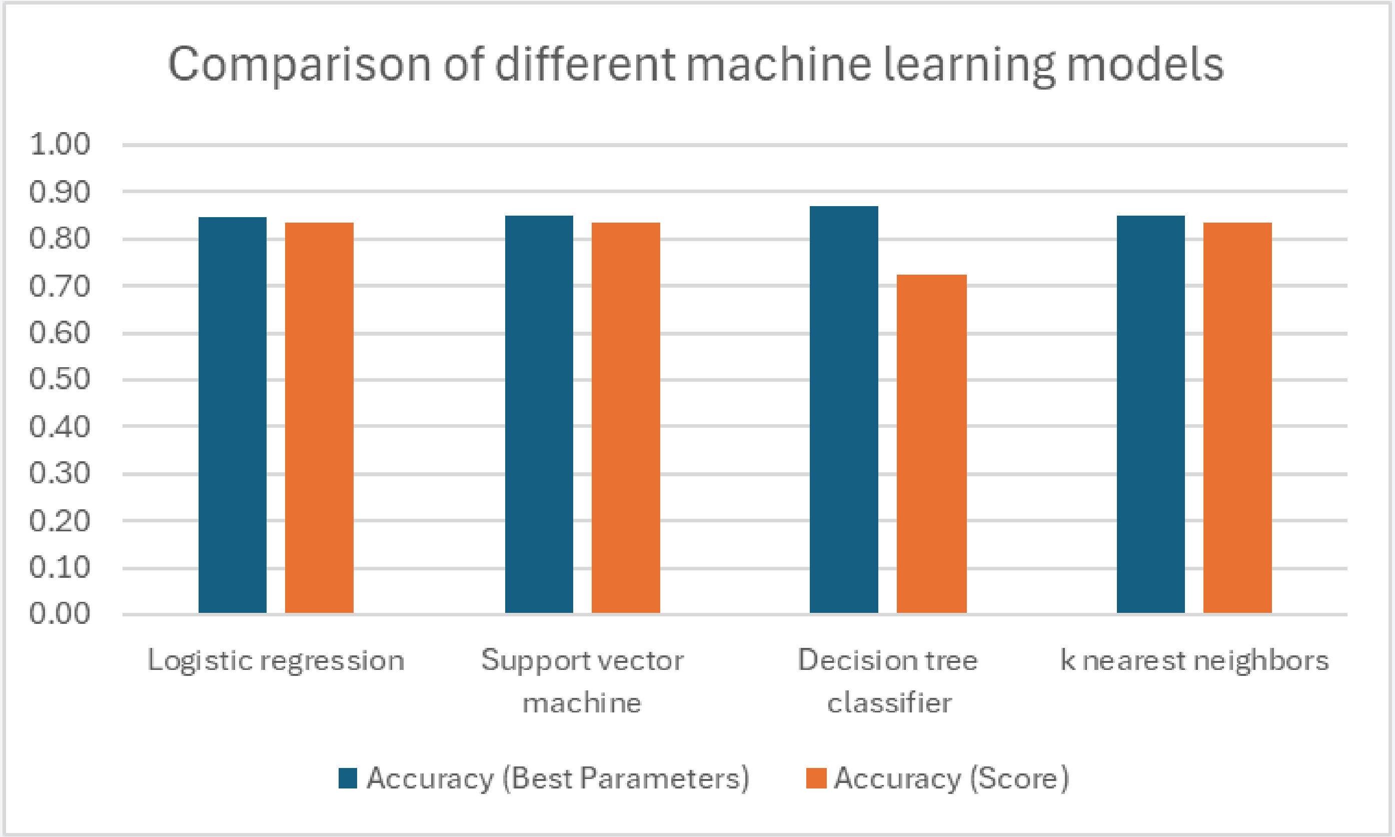
Section 5

Predictive Analysis (Classification)

Classification Accuracy

The best scores are as follows:

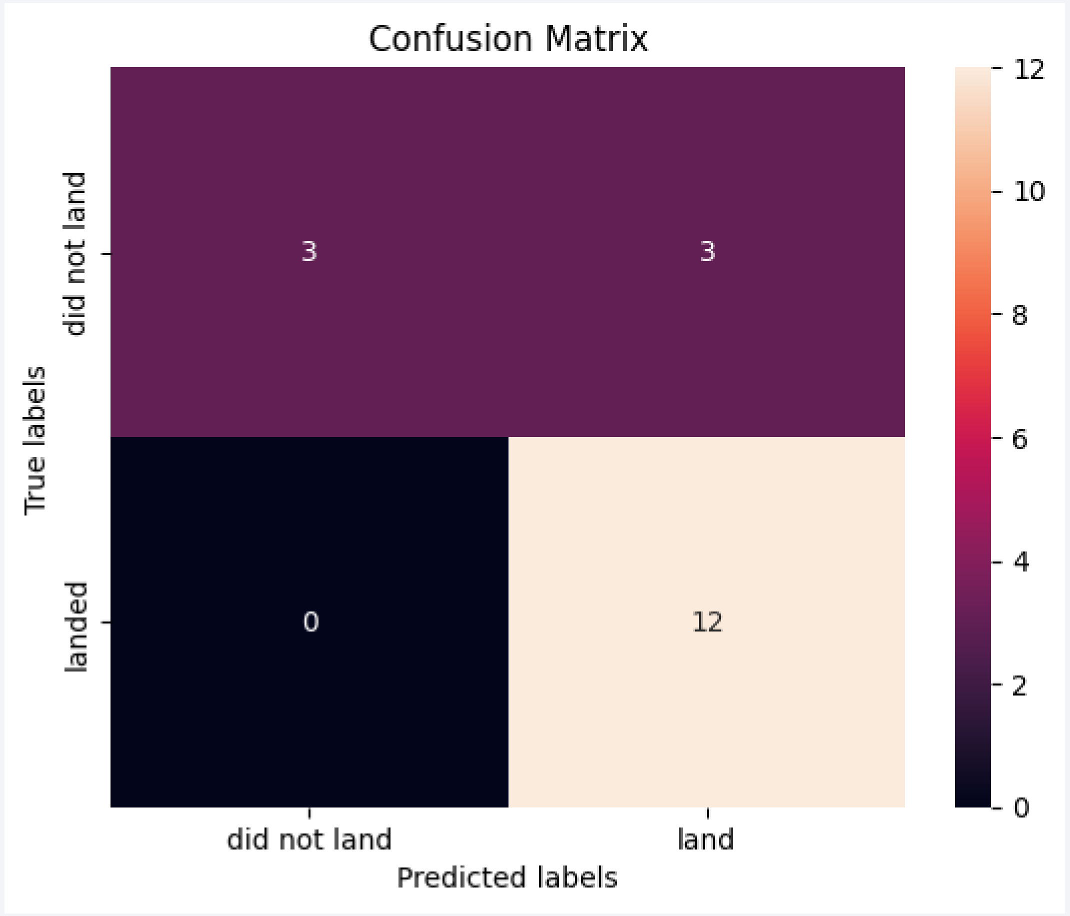
- Logistic = 8.464e-01
- SVM: 8.482e-01
- Tree: 8.714e-01
- KNN: 8.482e-01
- All the methods are more or less similar, delivering a best score between 84.8% and 87.1%
- The Decision Tree delivers the best score using optimized parameters



Confusion Matrix

- This confusion matrix is representative of:
  - Logistic regression
  - SVM
  - KNN

- There are 12 true positives (land / predicted to land)
- There are 3 true negatives (did not land / predicted to not land)
- There are 3 false positives (did not land / predicted to not land)
- There are 0 false negatives (landed / predicted to not land)



Conclusions

- Data is available from 2010 to 2020.
- From 2010 to 2013, there have been mostly failures.
- From 2013 to 2020, success has been increasing linearly with a dip in 2018.
- There is a correlation between launch site and success: the most successful launch site is KSC LC-39A with 41.7% of all successful launches coming from there. Its independent success rate is 76.9%
- Most of the successful launches have payloads between 2000 and 5500 kg.
- Most of the successful launches are also with the FT booster category.
- Several machine learning models were used. They all yield comparable results with the decision tree being best at 87.14% accuracy.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!