Winning Space Race with Data Science

Leonardo Barneschi
August 14, 2024

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection (SpaceX API)
  - Data Collection (Web Scraping)
  - Data Wrangling
  - EDA with SQL
  - EDA with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - EDA results with interactive analytics
  - Predictive Analysis

Introduction

- SpaceX has revolutionized the space industry by offering rocket launches, specifically the Falcon 9, for as low as $62 million, compared to other providers whose costs exceed $165 million per launch. This significant cost reduction is largely due to SpaceX's groundbreaking strategy of reusing the first stage of their rockets, landing them back for subsequent missions. By continually refining this process, SpaceX can further drive down launch prices.

- As a data scientist at a startup competing with SpaceX, the objective of this project is to develop a machine learning pipeline that predicts the landing outcome of the first stage. This project is essential for determining the optimal pricing strategy to effectively bid against SpaceX for rocket launches.

- The project objectives entail:
  - Identifying all factors that influence the landing outcome.
  - Identify eventual relationships between features and outcome.
  - Determining the best conditions to maximize the probability of a successful landing.

Section 1

Methodology

Methodology

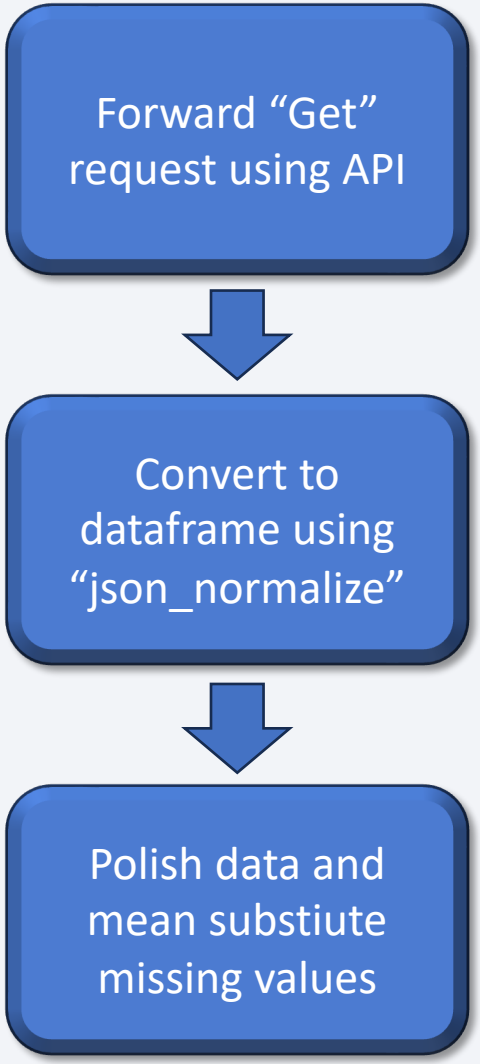
Executive Summary

- Data collection methodology:
  - SpaceX Rest API + web scraping
- Perform data wrangling
  - Data filtering and missing values
  - Featurization/encoding
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Building, training and evaluation of a Classification model

Data Collection

- Data collection is the process of gathering and measuring information on targeted variables in a structured system to answer relevant questions and evaluate outcomes. In this project, the dataset was obtained through two methods: REST API and web scraping from Wikipedia.
- For the REST API, we initiated the process with a GET request, decoded the JSON response, and transformed it into a pandas dataframe. Afterwards, we polished the data (missing values, handling gaps, etc.)
- For web scraping, we employed BeautifulSoup to extract launch records from HTML tables on Wikipedia. The extracted tables were parsed and converted into pandas dataframes, enabling further analysis.

Data Collection – SpaceX API




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

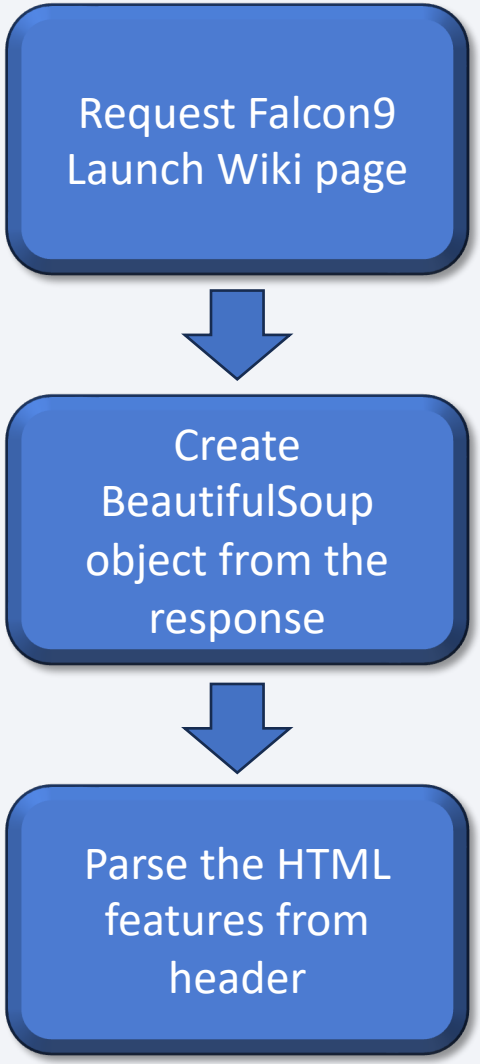
```
response = requests.get(spacex_url)
```

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

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and
# rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

- From: https://github.com/LeonardoBarneschi/DSCapstone/blob/main/00-data-collection-api.ipynb

Data Collection – Scraping




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

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html.text)
```

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

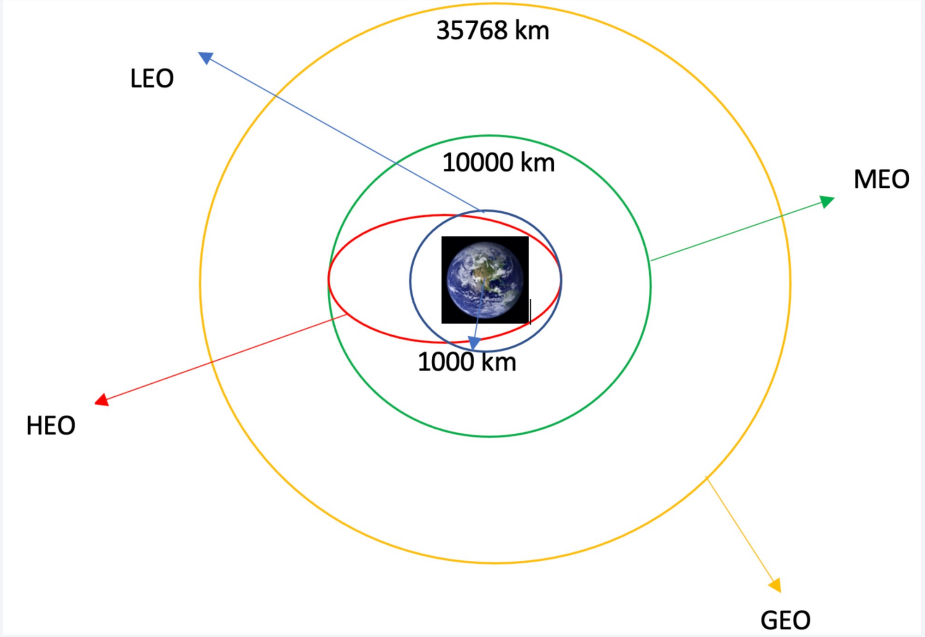
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = soup.find_all('table')
```

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into launch_dict with key `Flight No.`
            launch_dict["Flight No."].append(flight_number)
            #print(flight_number)
            datatimelist=date_time(row[0])
```

- From: https://github.com/LeonardoBarneschi/DSCapstone/blob/main/01-data-collection-with-web-scraping.ipynb

Data Wrangling

- Process of polishing the data to make it inputs to exploratory data analysis (EDA)
- In the following, the rocket launches, differing in launch site, outcome, orbits, etc. will be analyzed, and finally used to deploy a statistical model.
- To do this, data was first parsed (API + scraping) and prepared in a useful comma separated value (CSV) file.

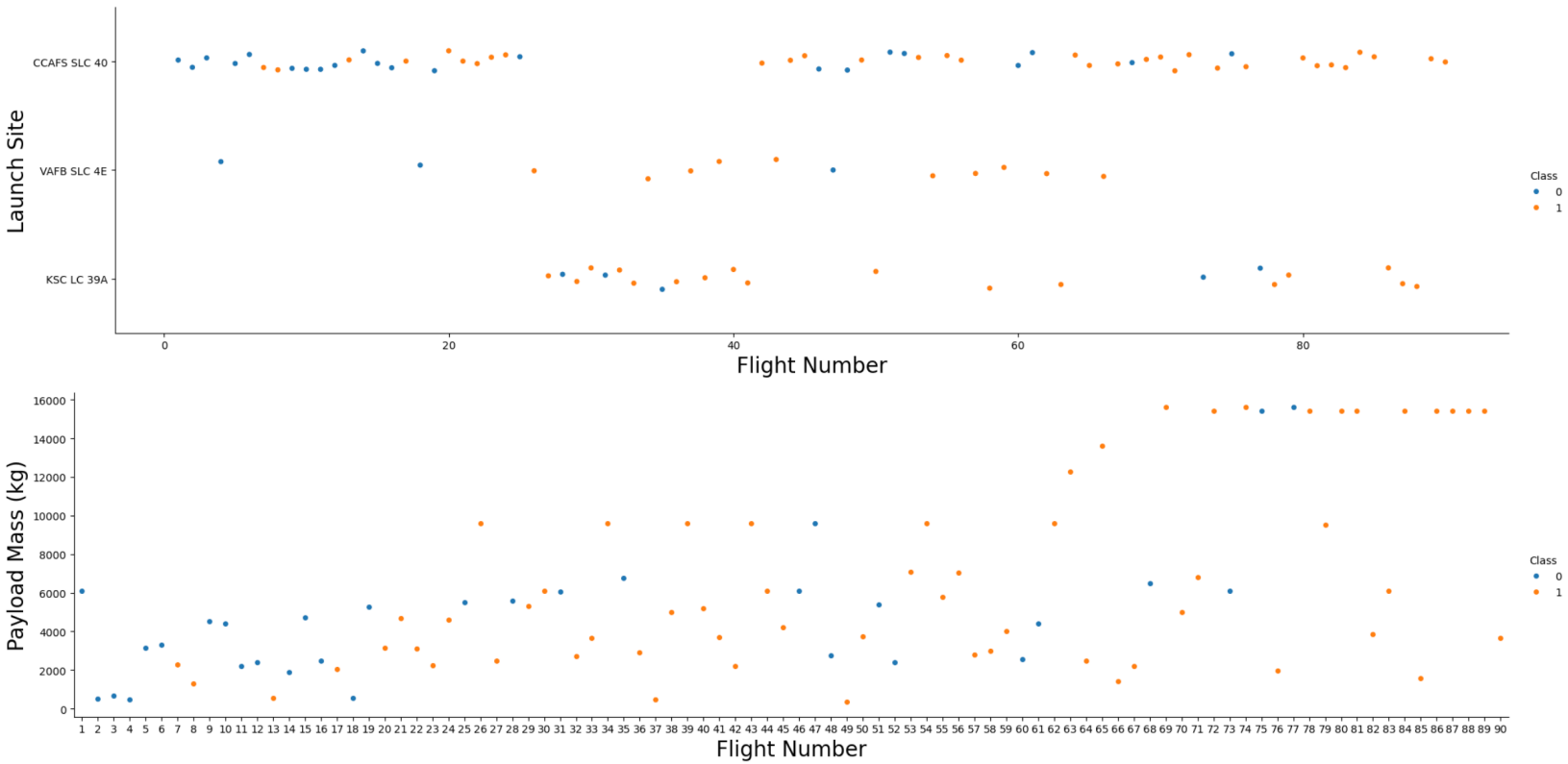



```
df=pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_1.csv")
df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	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
1	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
2	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
3	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
4	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
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1005	-80.577366	28.561857
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1006	-80.577366	28.561857
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1	False	False	True	NaN	1.0	0	B1007	-80.577366	28.561857
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1008	-80.577366	28.561857
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1	False	False	False	NaN	1.0	0	B1011	-80.577366	28.561857

- From: https://github.com/LeonardoBarneschi/DSCapstone/blob/main/02-data-wrangling.ipynb

EDA with Data Visualization



- Scatter graph as tool to find relationships/correlation between features
- Payload Mass, Flight Number, Launch site and Orbit site plotted against each other “combinatorially”
- This shows patterns that affect the most the launch outcome.

- From: https://github.com/LeonardoBarneschi/DSCapstone/blob/main/03-eda-dataviz.ipynb

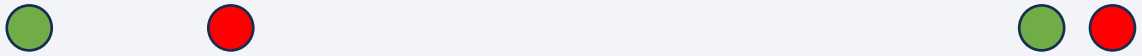
EDA with SQL

- Using SQL, we had performed many queries to get better understanding of the dataset, E.g.
  - Displaying the names of the launch sites.
  - Displaying 5 records where launch sites begin with the string 'CCA'.
  - Displaying the total payload mass carried by booster launched by NASA (CRS).
  - Displaying the average payload mass carried by booster version F9 v1.1.
  - Listing the date when the first successful landing outcome in ground pad was achieved.
  - Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
  - Listing 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 landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order

- From: https://github.com/LeonardoBarneschi/DSCapstone/blob/main/04-eda-sql.ipynb

Build an Interactive Map with Folium

- Launch data were visualize using an interactive map generated with Folium. Latitude and Longitude are used as x,y coordinates to position a maker object at launch site.



- The outcome (Success, Failure) are conveniently mapped to (1, 0) and the colors green and red with the MarkerCluster() construct.

- We calculated the distance from launch site and landmarks such as railway stations/lines, cities, highways and coastlines to answer questions such as:

    - The launch site can be chosen at random or are there specific prerequisites that the surrounding area should provide?

- From: https://github.com/LeonardoBarneschi/DSCapstone/blob/main/05-interactive-viz-analytics.ipynb

Build a Dashboard with Plotly Dash

- **Launch Sites Dropdown List:**
  - Added a dropdown list to enable Launch Site selection.
- **Pie Chart showing Success Launches (All Sites/Certain Site)**
  - Added a pie chart to show the total successful launches count for all sites and the Success vs. Failed counts for the site, if a specific Launch Site was selected.
- **Slider of Payload Mass Range:**
  - Include a Slider object to select the payload range to show the scatter
- **Scatter Chart of Payload Mass vs. Success Rate for the different Booster Versions:**
  - Include a Slider object to select the payload range to show the scatter

- From: https://github.com/LeonardoBarneschi/DSCapstone/blob/main/spacex_dash_app.py

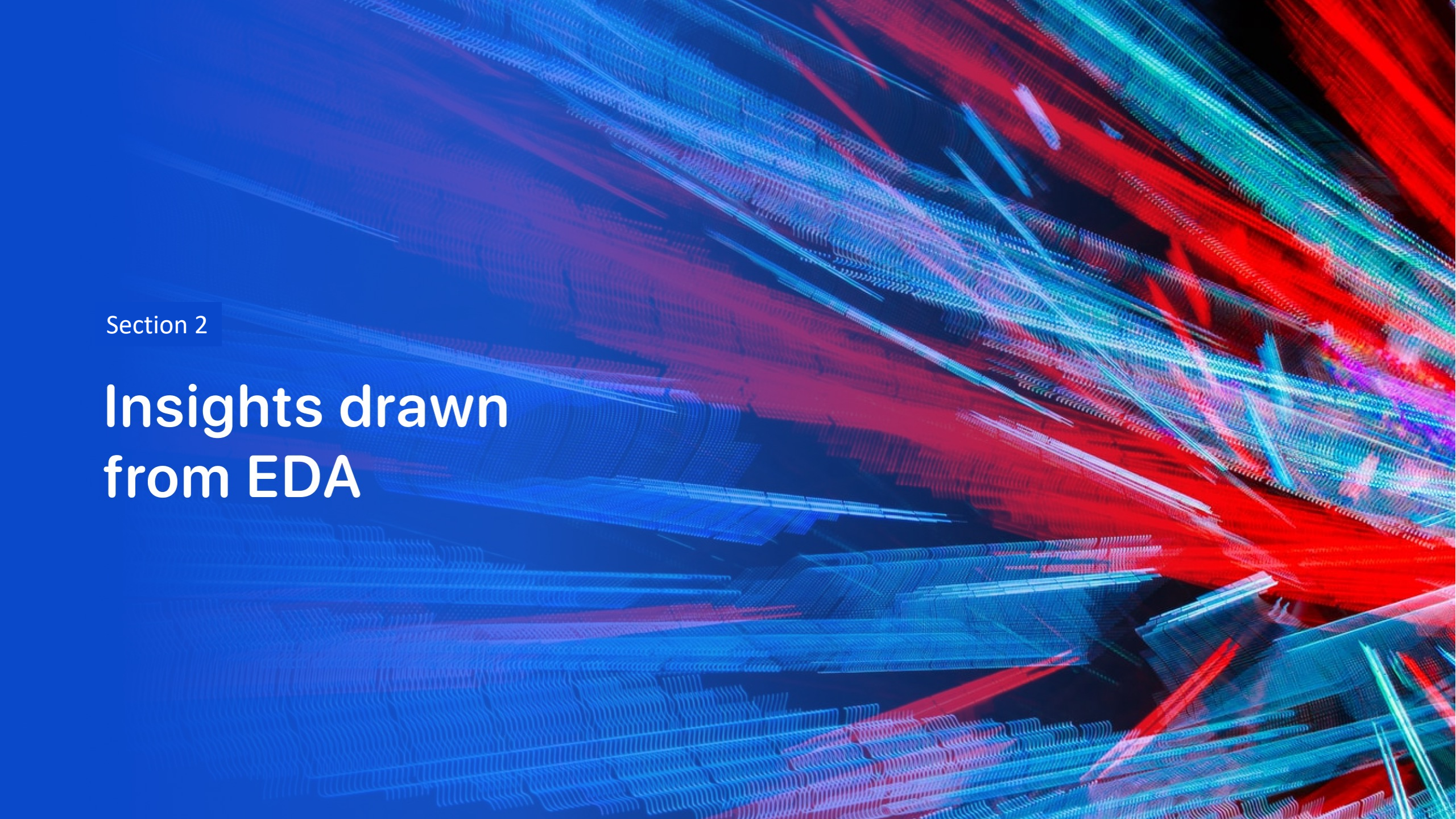
Predictive Analysis (Classification)

- Data Preparation
  - Load dataset into DataFrame
  - Normalize data
  - Train/Test split
- Model Preparation
  - Choose Algorithm (ML)
  - Set grid of parameters (GridSearchCV)
  - Plot Confusion Matrix (CM)
- Model Evaluation
  - Select hyperparameters for each model
  - Compute accuracy according to the desired metrics
- Model Comparison
  - Compare the models for accuracy
  - The model corresponding to the highest accuracy is selected

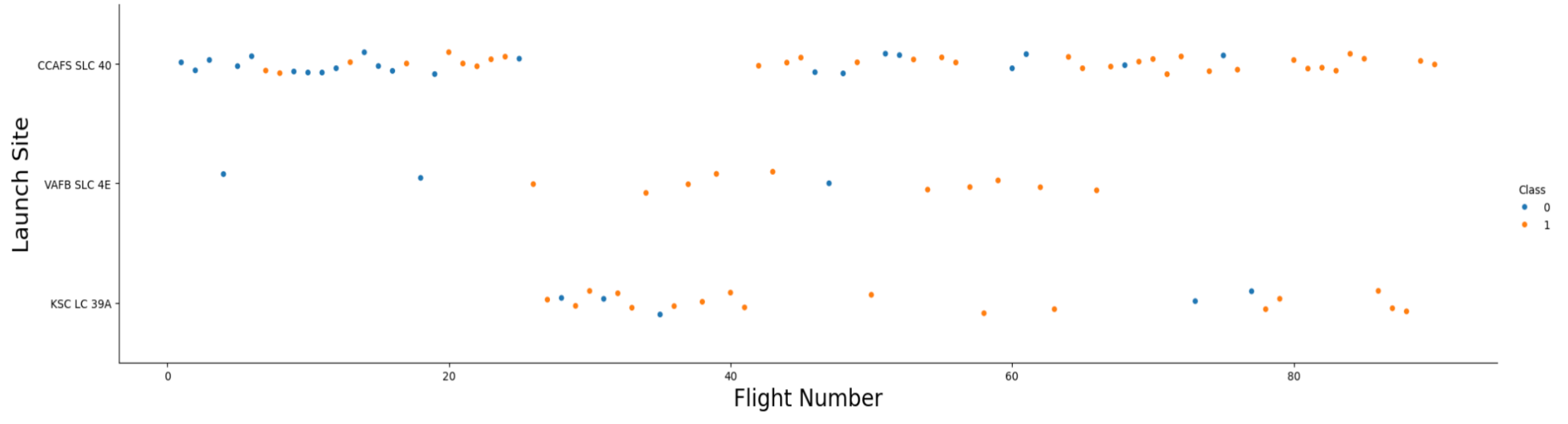
- From: https://github.com/LeonardoBarneschi/DSCapstone/blob/main/06-ml-pred.ipynb

Results

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

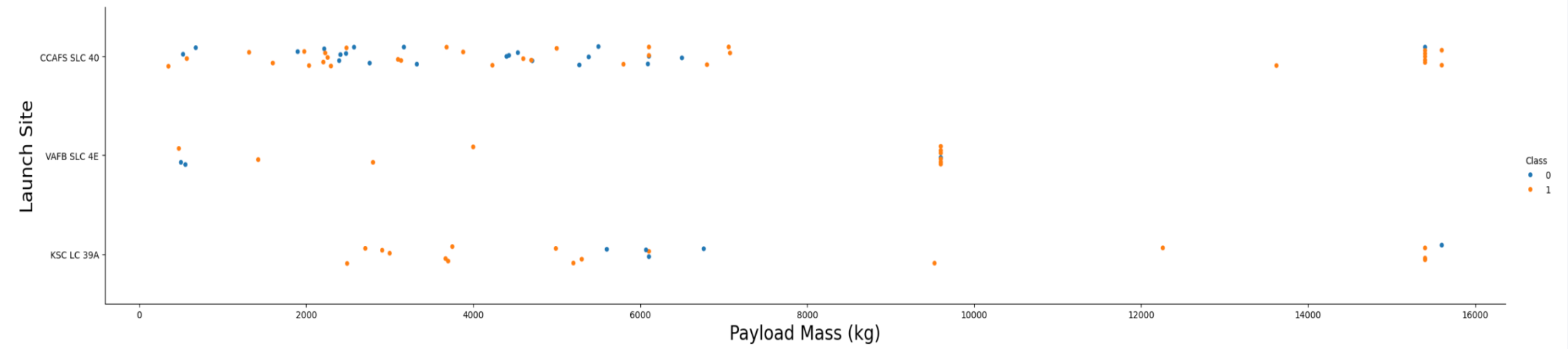

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



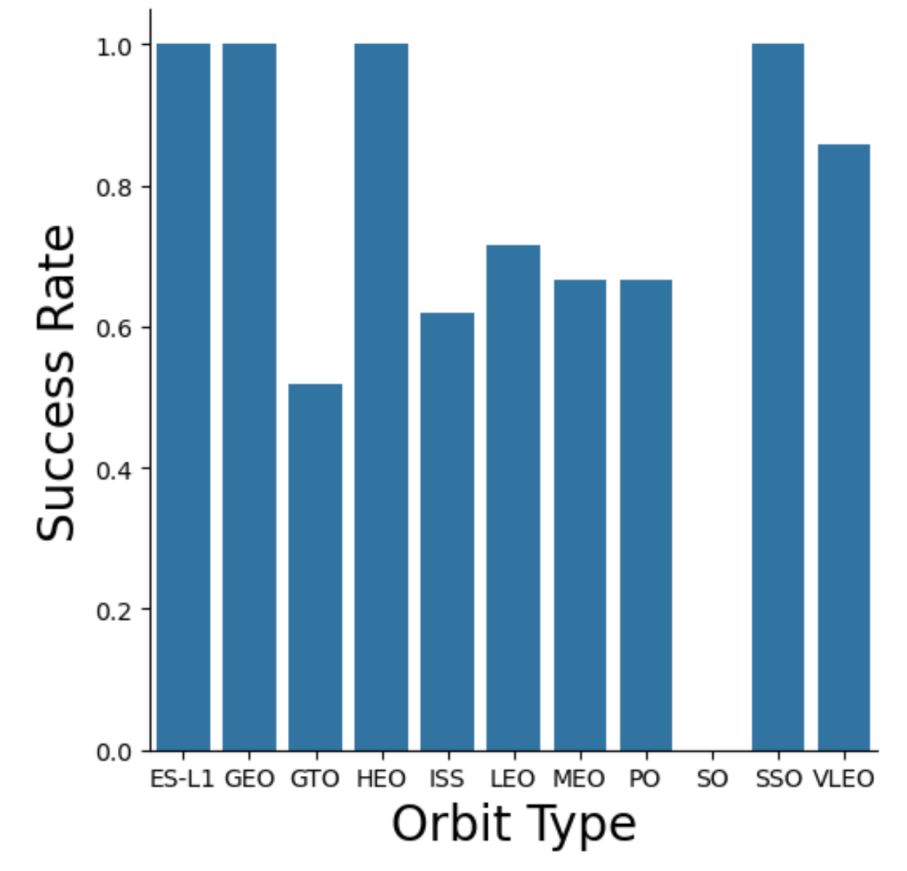
- A general trends seems to suggest that for all the sites the success rate increases with increased flight number.
- In general, VAFB SLC 4E and KSC LC 39A have higher success rate than CCAF5 SLC 40

Payload vs. Launch Site



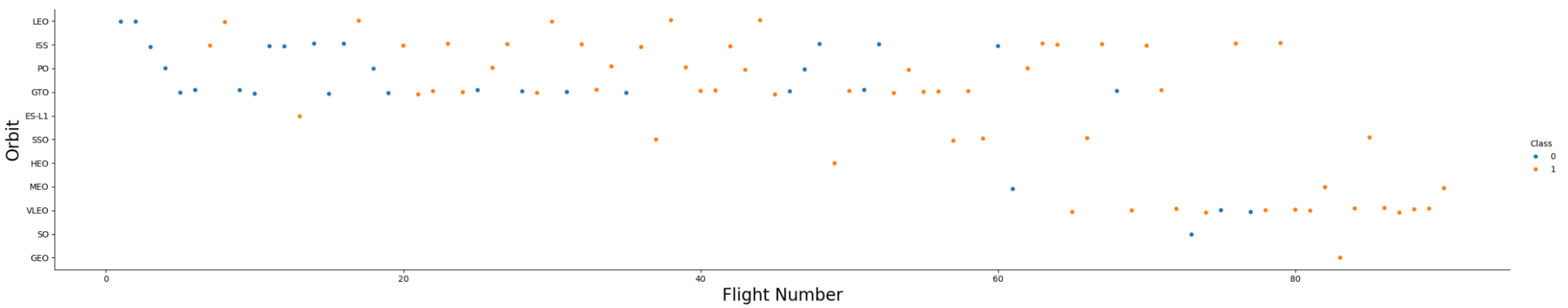
- It is straight forward to notice that increasing the Payload after a certain limit (7000 kg) seem to yield a vert high success rate

Success Rate vs. Orbit Type



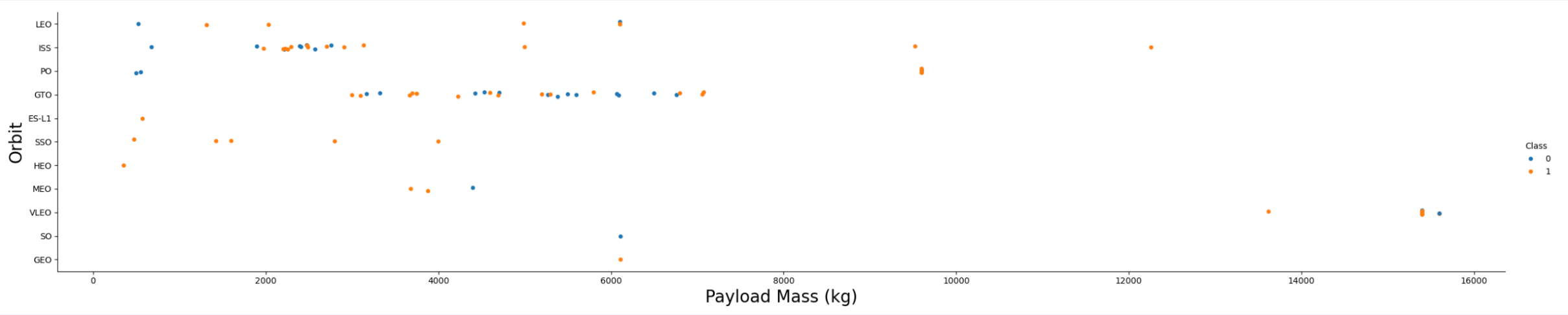
- The plot shows how the success rate is related to the type of orbit chosen/needed
- Clearly, ES-L1, GEO, HEO and SSO have 100% success rate as opposed to SO which has a 0% success rate

Flight Number vs. Orbit Type



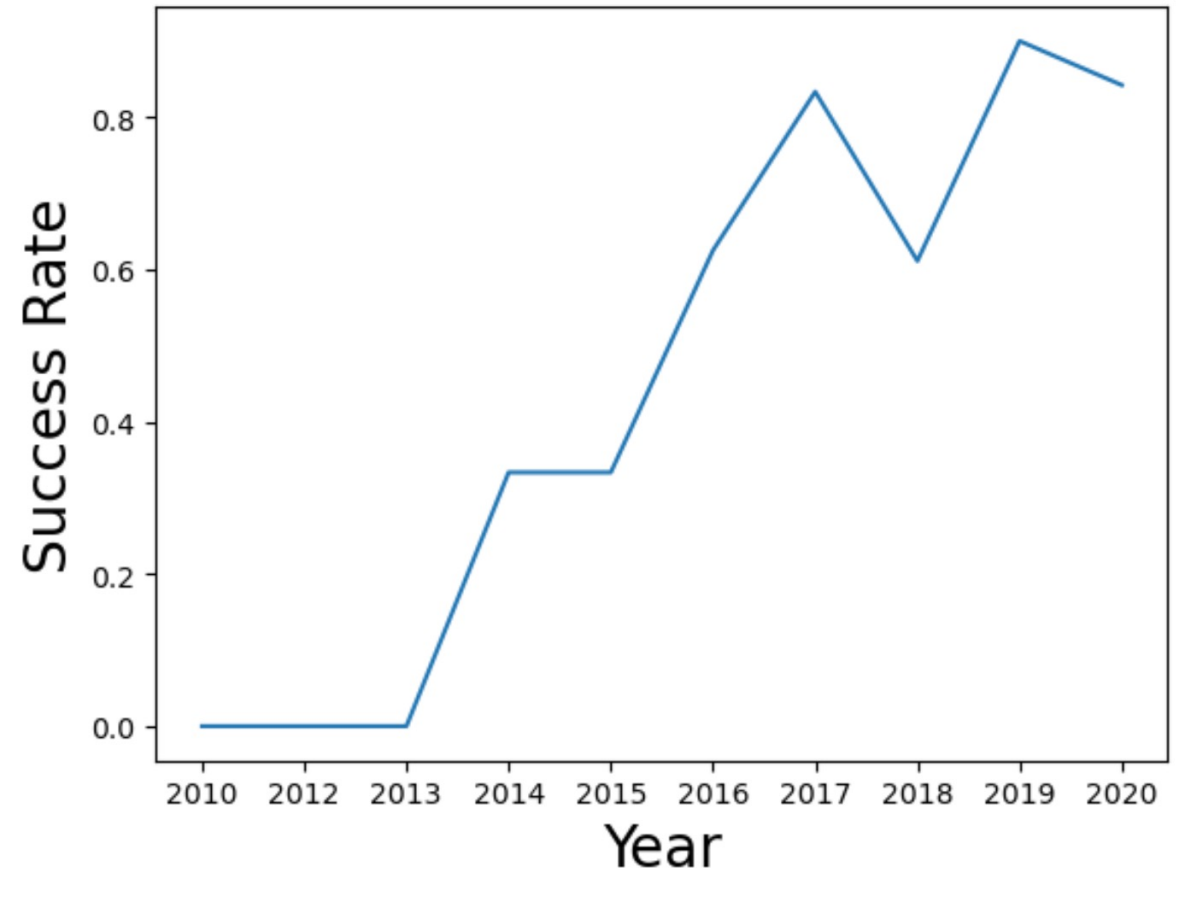
- We notice that the success rate increases with the number of flights for the LEO orbit. For some orbits like GTO, there is no relation between the success rate and the number of flights. But we can suppose that the high success rate of some orbits like SSO or HEO is due to the knowledge learned during former launches for other orbits.

Payload vs. Orbit Type



- The Payload, as noticed before has a significant influence on the success rate of rocket launches. In particular, here it is shown that this influence is also related to the type of orbit. Although the correlation is not striking, it is clear for some orbit it is necessary to increase the payload, while others, such as SSO, apparently do not depend on the payload at all.

Launch Success Yearly Trend



- I would say that this is rather self-explanatory

- The success rate is clearly tremendously increasing as years pass by, with a slight hiccup from 2019 to 2020

All Launch Site Names

- SQL QUERY -> SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL

Task 1

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

```
[23]: %sql select distinct launch_site from SPACEXTBL;
```

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

[23]:

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

- The command DISTINCT in the query, allows to take care of eventual duplicate of Launch_Site, as evidenced in the picture

Launch Site Names Begin with 'CCA'

Task 2

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

[24]:
```
%sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5;
```

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

[24]:

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

- Similar to a string comparison in any programming language.
- The main actor here is the command “where”, that check for substrings in string.
- Like performs the filtering
- Limit prints out only 5 entries of possibly more results

Total Payload Mass

Task 3

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

```
[25]: %sql select sum(payload_mass__kg_) as total_payload_mass from SPACEXTBL where customer = 'NASA (CRS)';
 * sqlite:///my_data1.db
Done.
```

[25]:

total_payload_mass
45596

- In short, this query defines a new variable “total_payload_mass” by first selecting from the SPACEXTBL the rows corresponding to the “NASA (CRS)” customer, then from each of this rows, payload_mass_kg is extracted and finally summed up

Average Payload Mass by F9 v1.1

Task 4

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

```
[33]: %sql select avg(payload_mass__kg_) as average_payload_mass from SPACEXTBL where booster_version like '%F9 v1.1%';
```

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

[33]:

average_payload_mass
2534.6666666666665

- Very similar to the previous query, the difference is that the rows filtering targeted the booster version rather than the customers and the payload was averaged instead of being summed.

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
[51]: %sql select min(date) as first_successful_landing from SPACEXTBL where landing_outcome = 'Success (ground pad)';
```

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

[51]:

first_successful_landing
2015-12-22

- Here the secret is to query for “landing_outcome” and filtering out “Success (ground pad)”. The landing outcome column is “verbose” enough to let us do this filtering first and then grabbing the index corresponding to the minium value (date format) in the column “date”.

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

[35]:
```
select booster_version from SPACEXTBL where Landing_Outcome = 'Success (drone ship)' and payload_mass__kg_ between 4000 and 6000;
```

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

[35]:

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

- This is a combination of 2 of the above. However, the output here is not a single value. First I filter the rows where “Success (drone ship)” is found, and out of this sub-table, I grab the rows where the payload_mass is between 4000 and 6000 kg.

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

[38]:
```
%sql select mission_outcome, count(*) as total_number from SPACEXTBL group by mission_outcome;
```

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

[38]:

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

- Use the function “count” on the query by “mission_outcome”. Pretty straightforward.

Boosters Carried Maximum Payload

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

[39]:
```
%sql select booster_version from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL);
```

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

[39]:

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. The main query uses subquery results and returns unique booster version (SELECT DISTINCT) with the heaviest payload mass.

2015 Launch Records

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

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

[47]:
```
%%sql select substr(Date,6,2) as month, date, booster_version, launch_site, landing_outcome from SPACEXTBL
      where landing_outcome = 'Failure (drone ship)' and substr(Date,0,5)='2015';
```

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

[47]:

month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

- This is just a double query (like a AND Boolean operation), to grab at the same times the rows where the “Date” contains “2015” (string comparison) and the landing outcome is a failed drone ship landing in the ocean.

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

Task 10

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

[50]:
```
%%sql select landing_outcome, count(*) as count_outcomes from SPACEXTBL
      where date between '2010-06-04' and '2017-03-20'
      group by landing_outcome
      order by count_outcomes desc;
```

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

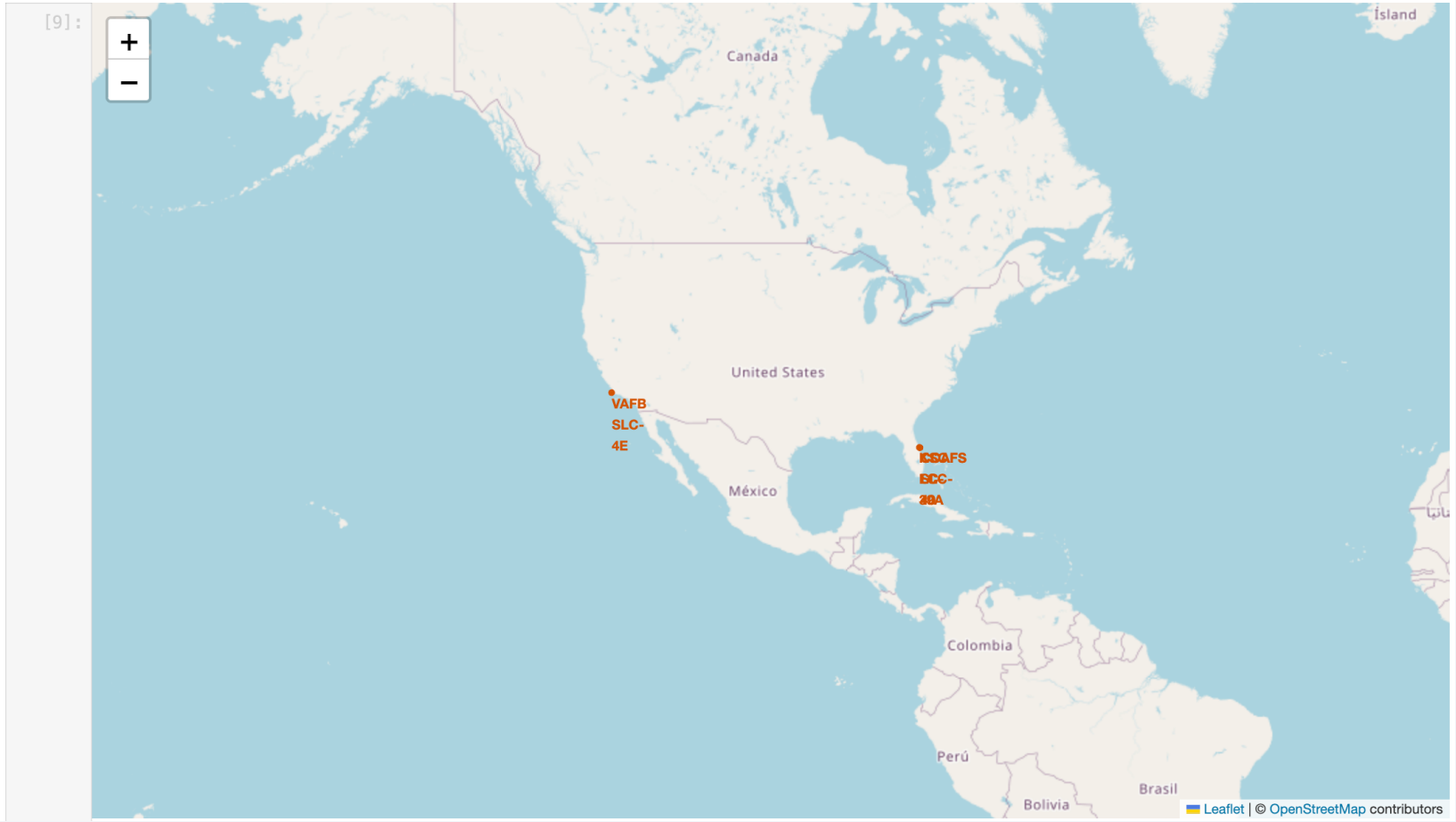
[50]:

Landing_Outcome	count_outcomes
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

- The query returns landing outcomes and their number where mission were both successful and date of launch is comprised between 04/06/2010 and 20/03/2017. The GROUP BY clause groups results by landing outcome and ORDER BY COUNT DESC just prints results in decreasing order.


Section 3
Launch Sites
Proximities Analysis

Launch Sites location

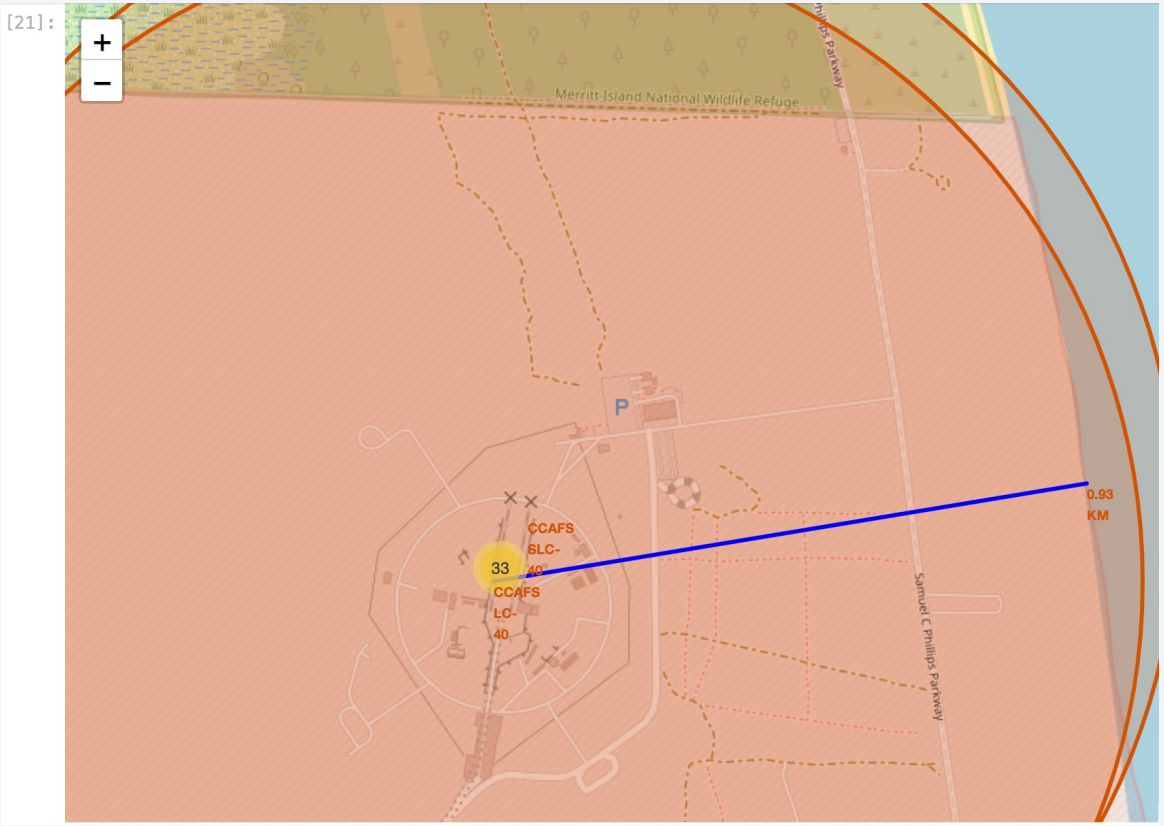


- All the launch sites are within the USA
- Those visibles are in Florida and in California (i.e. in coastal areas close to the equator)

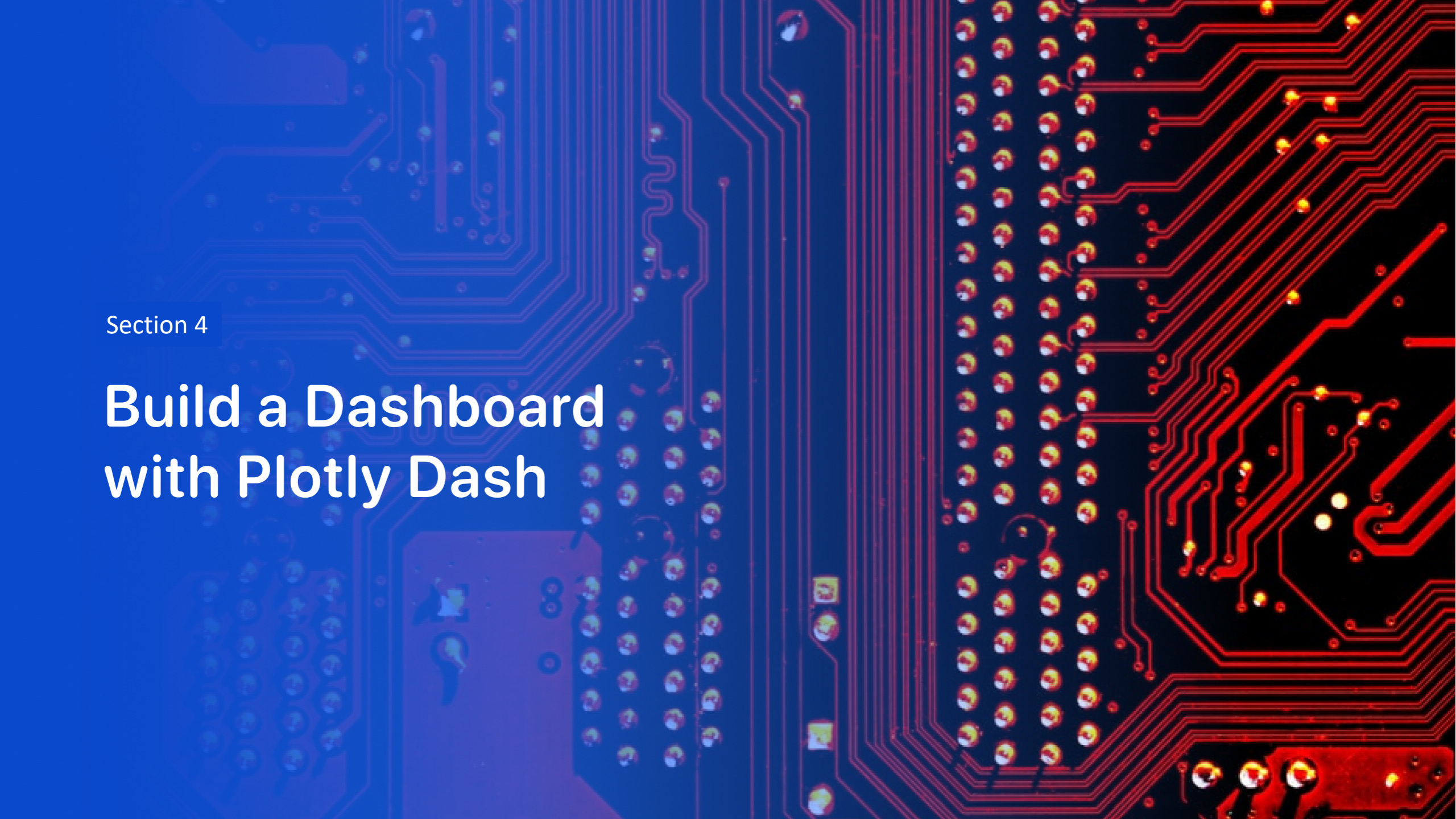
Launch Sites Visual Success Rate

- For a specific Launch Site we can add a marker object for each of the attempted launches, and for instance, as in the example, show with different colours the success/failure of each of the launches performed at the selected (zoomed in) selection.
- Here for instance, for this Launch site in Florida, we can see with Green Markers the succesfull launches and the opposite in red.

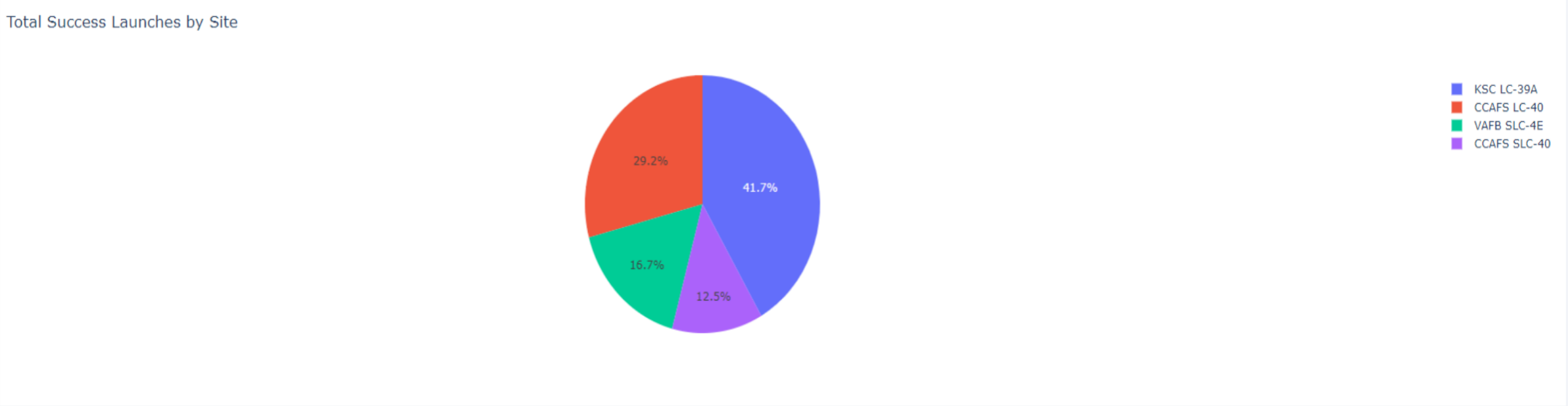
Launch Sites and Landmarks



- With Folium it is easy to also show the distances between different points on the map.
- For SpaceX, for instance, it could be important to have a look at how a certain launch site is far from the coast, as shown here.
- The blue line marks a distance of <1 km

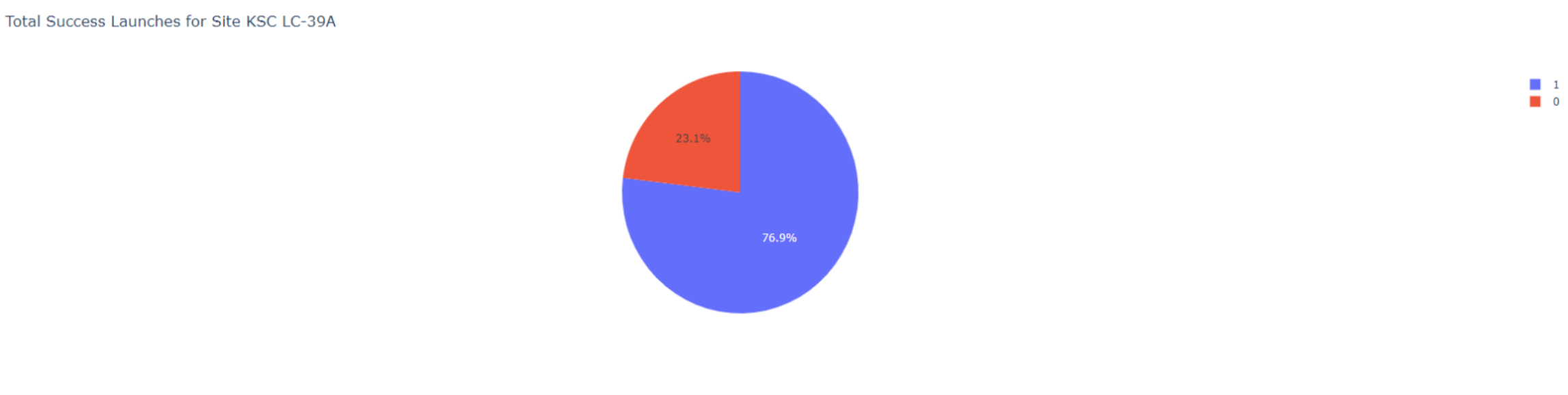

Section 4
Build a Dashboard with Plotly Dash

Success Rate by Launch Site – Pie Chart



We see that KSC LC-39A has the best success rate of launches.

Success Launches for a Specific Site



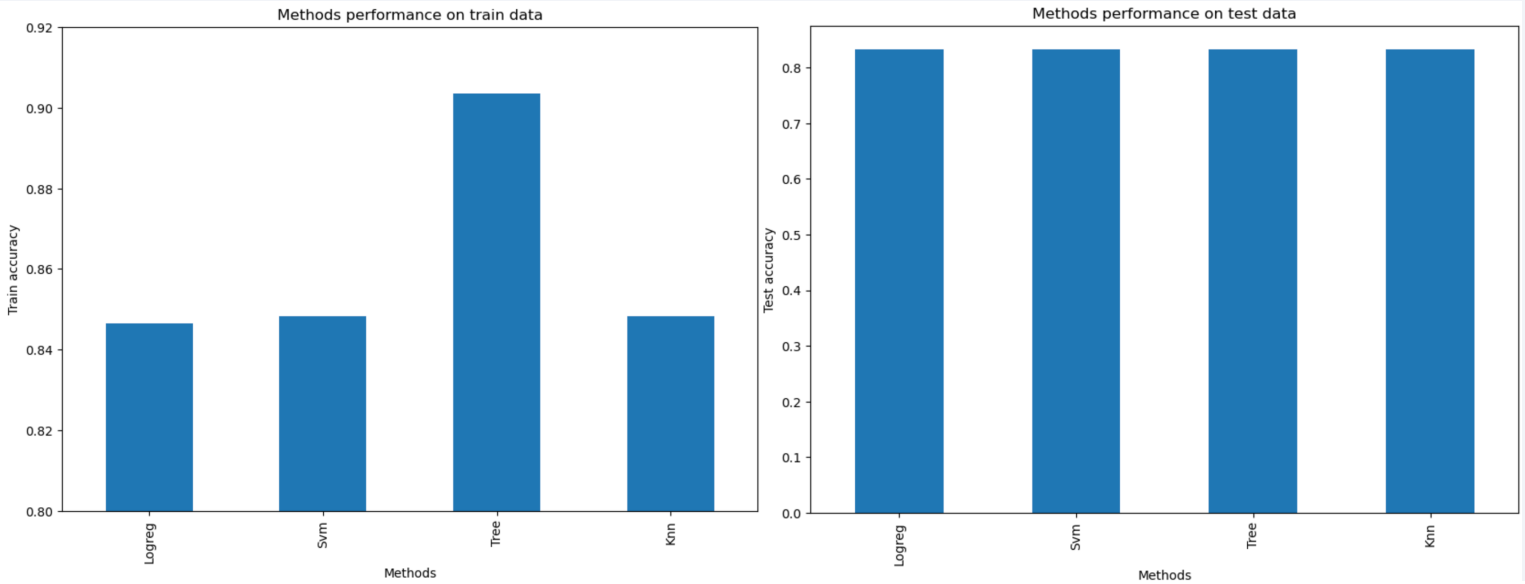
We see that KSC LC-39A has achieved a 76.9% success rate while getting a 23.1% failure rate.

Scatter Plot with Adjustable Range Dashboard

Section 5

Predictive Analysis (Classification)

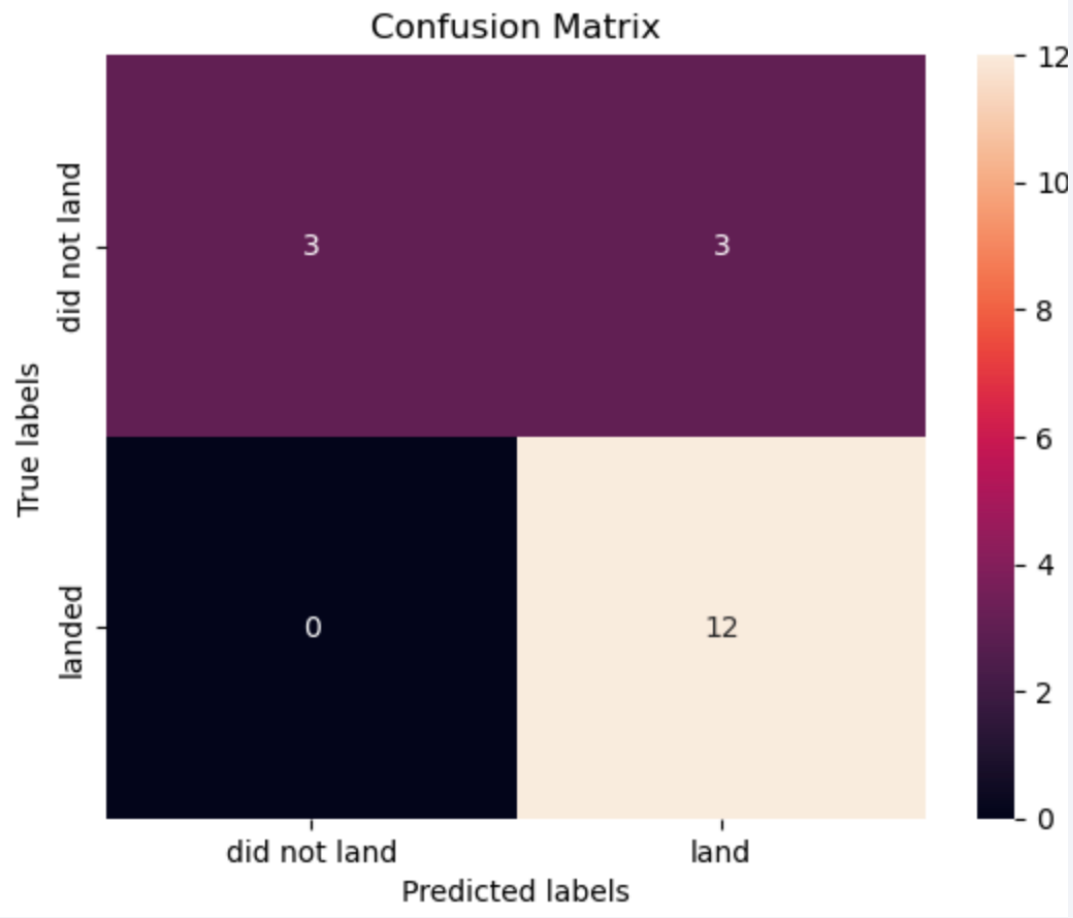
Classification Accuracy




```
[22]: print("tuned hyperparameters :(best parameters) ", tree_cv.best_params_)
      print("accuracy :", tree_cv.best_score_)

tuned hyperparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 16, 'max_features': 'sqrt', 'min_samples_leaf': 4,
'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.9035714285714287
```

- On the test set, which is the most important, the 4 tested model (classifiers) perform equally well. The decision tree classifier is the one that performs best on the training set. Ideally, we should increase the number or samples or personally I'd tray a 70/30 split to see which model generalizes better.

Confusion Matrix



- The interpretation is rather straightforward. For this decision tree classifier (hyperparams in the previous slide), the accuracy is of 100% in predicting the succesfull landings.
- On the other hand, it seems to perform less well in the prediction of the unsuccesful landings, where in 50% of the cases the model predicts a successful landing when it should be a failure instead.

Conclusions

- A successful landing correlates, especially, with the number of previous attempts (the more the merrier)
- Also, not less important, on the type of orbit (i.e. GEO, HEO, SSO, ES-L1) and the payload mass.

- KSC LC-39A seems to be the most succesfull launch site, but is not yet clear as to why

- Standard Classifier algorithm perform reasonably well with the current dataset, with a pitfall being false positives. Such algorithms (e.g. decision trees), would highly benefit from a larger dataset, and more importantly, more balanced dataset (50/50 success failures)

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!