IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Heda Zilli Tomita
1 November 2024

Outline

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

Executive Summary

- **Summary of methodologies**
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis
- **Summary of all results**
  - Attributes correlate with successful landings.
  - Exploratory data analysis results
  - Machine Learning models on SpaceX data

Introduction - Project background and context

The space industry has experienced a remarkable transformation in recent years, driven by innovative private enterprises like SpaceX, which has emerged as a frontrunner in this new era.

- By developing the Falcon 9 rocket, SpaceX has successfully lowered launch costs significantly, thanks to its ability to land and reuse the first stage.
- Falcon 9 rocket at a cost of $62 million; other providers cost upward of $165 million each.

Introduction - Project background and context

In this competitive landscape, Space Y aims to position itself as a viable competitor to SpaceX.

Objective

- Gather data about SpaceX's operations and use machine learning model

Challenges:

- Understanding the factors that influence the success of first-stage recovery
- Can we predict landing success based on historical data?
- What are the conditions to get best results and ensure the best succesful landing rate?

Section 1

Methodology

Methodology

Executive Summary

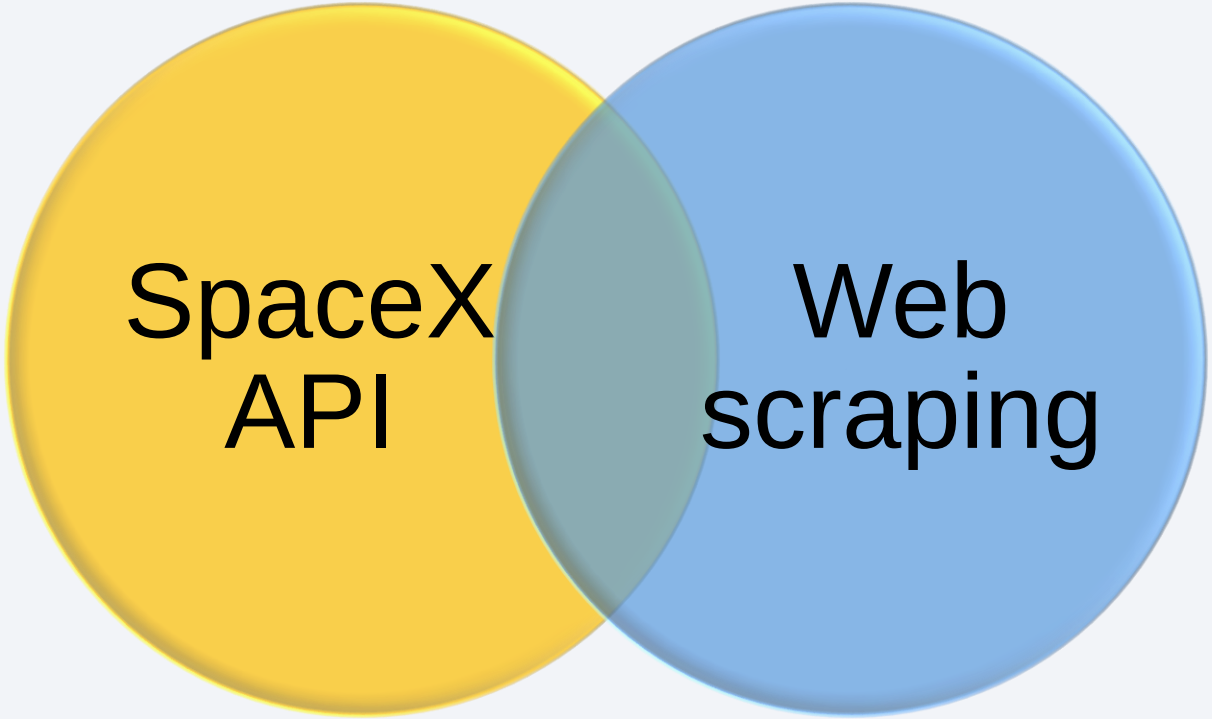
- Data collection methodology:
  - SpaceX’s public API to gather data about launches.
  - Web scraping from wiki pages.
- Perform data wrangling
  - Preparing the dataset for analysis by standardizing and converting relevant attributes.
  - The data collected (JSON object and HTML tables) was converted into Pandas dataframe for visualization and analysis.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Use of machine learning to determine if the first stage of Falcon 9 will land succesfully

Data Collection

Request data

Convert to JSON file

Normalize data into CSV



Request data from Wikipedia

Web data extraction using BeautifulSoup

Normalize data into CSV

Data Collection – SpaceX API

- Import libraries
- SpaceX API for data retrival
  - use the GET method to retrieve data.
- Data processing
  - Capture the response from the API, which will be in JSON format.
  - Convert JSON response to DataFrame.
  - Clean data and apply getBoosterVersion function
- Filter to only include Falcon 9 and dealing with missig values
- Exporting to a CSV

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

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

```
# Decode the response content as JSON
data_json = response.json()
# Convert the JSON result into a DataFrame using json_normalize
data = pd.json_normalize(data_json)
```

```
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}
```

Notebook: [Data Collection API -GitHub]

Data Collection – Scraping

- Import libraries
- Wikipedia data retrival
  - Make an HTTP request to the URL for Falcon launches.
  - Create BeutifulSoup objet from HTML response
- Extract data
  - Finding all tables
  - Extract column names
  - Create dictionary
  - Store data in dictionary
- Convert the dictionary into a Pandas DataFrame
- Exporting to CSV

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

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

```
# 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')
```

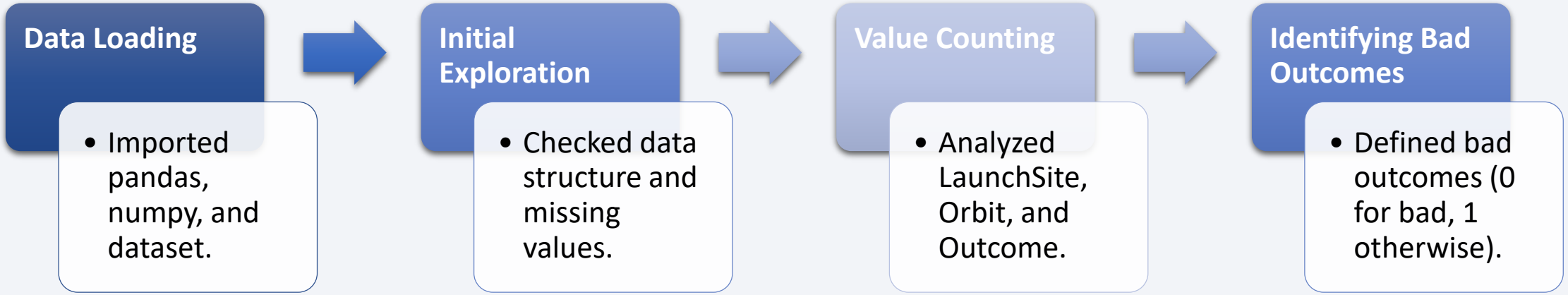
```
# Let's print the third table and check its content
first_launch_table = html_tables[2]
print(first_launch_table)
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant 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']=[]
```

Notebook: Data Collection Webscraping -GitHub

Data Wrangling

Preparing the dataset for analysis by standardizing and converting relevant attributes, especially the outcome variable

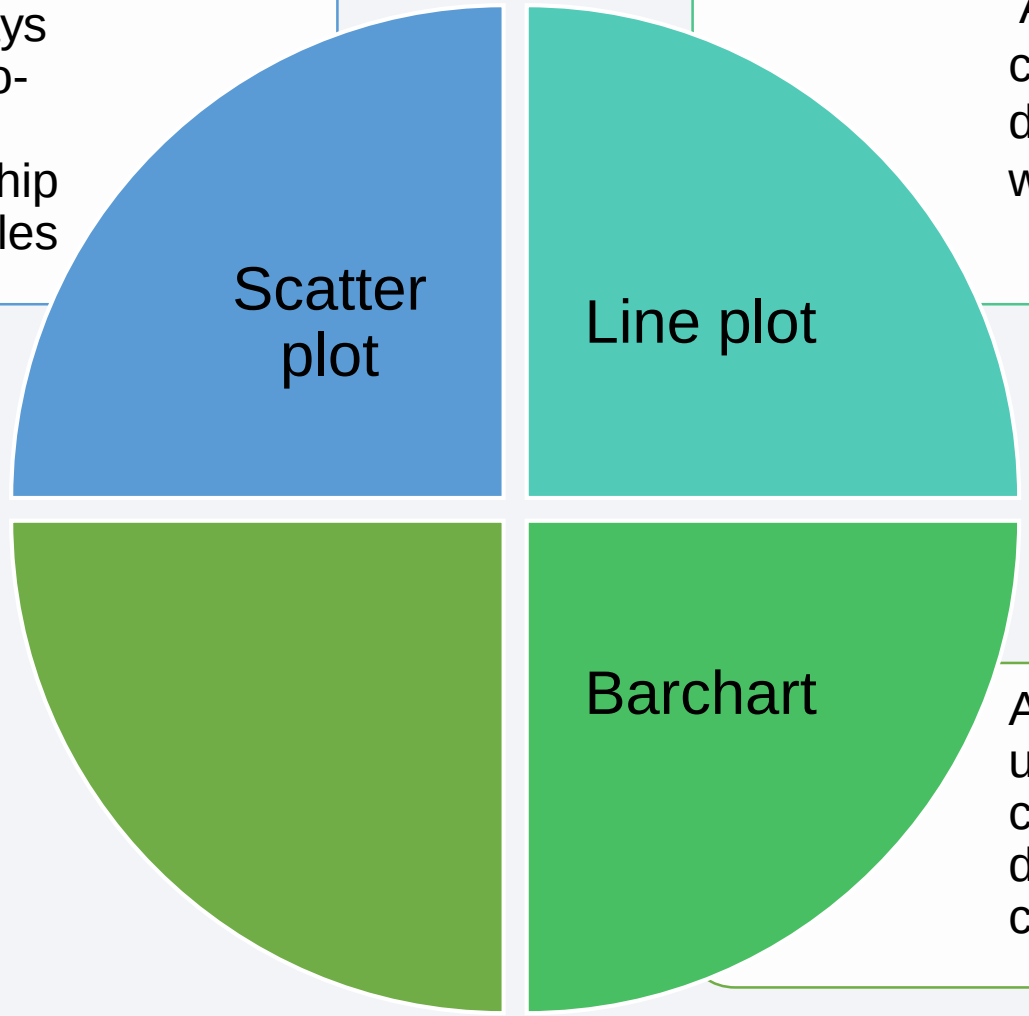


Notebook: [Data Wrangling-GitHub]

EDA with Data Visualization

A scatter plot displays data points on a two-dimensional grid. It shows the relationship between two variables

- Flight Number vs Launch site
- PayLoad vs Launch site



A line plot connects data points with lines.

- Year vs Success rate

A bar chart uses bars to compare different categories.

- Orbit Type vs Success rate

Notebook: [Data Visualization-GitHub]

EDA with SQL

Setup:	• Connect to SQLite database and read CSV data into a DataFrame.
Table Creation and Management:	• Drop an existing table (SPACEXTABLE) and create it from SPACEXTBL, filtering out records with null dates.
Distinct Launch Sites:	• Query for distinct launch sites from the SPACEXTBL table.
Payload Mass Calculations:	• Calculate total and average payload mass for specific customers
Landing Outcome Analysis:	• Determine the earliest launch date, count successful missions
Monthly and Date-Based Queries:	• Count landing outcomes over a specified date range

Notebook: Data EDA with SQL-GitHub

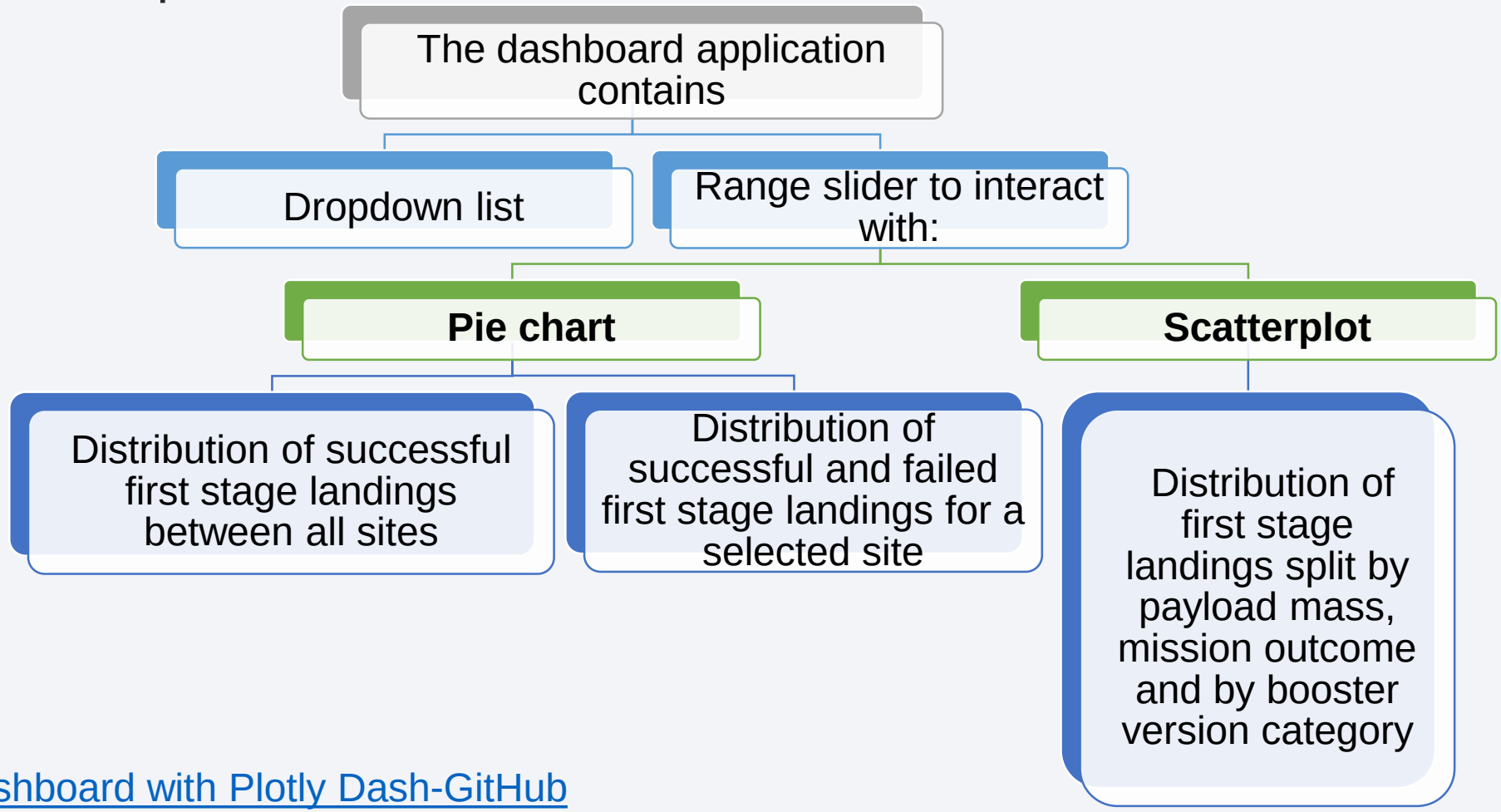
Build an Interactive Map with Folium

- **Using FOLIUM for Launch site analysis for SpaceX launch dataset**
- Finding an optimal location for building a launch site
- In a Folium map, you can create various map objects:
  - **Markers**: pinpoint specific locations points of interest, or data collection sites.
    - Mark all launch sites on a map
    - Mark the success/failed launches for each site on the map
  - **Circles**: represent areas of interest or influence
  - **Lines (PolyLines)**: show paths, routes, or connections between different locations.
    - Calculate the distances between a launch site to its proximities.
- Using Folium provided valuable insights into the spatial distribution of launch sites.

Notebook: Data Map Folium-GitHub

Build a Dashboard with Plotly Dash

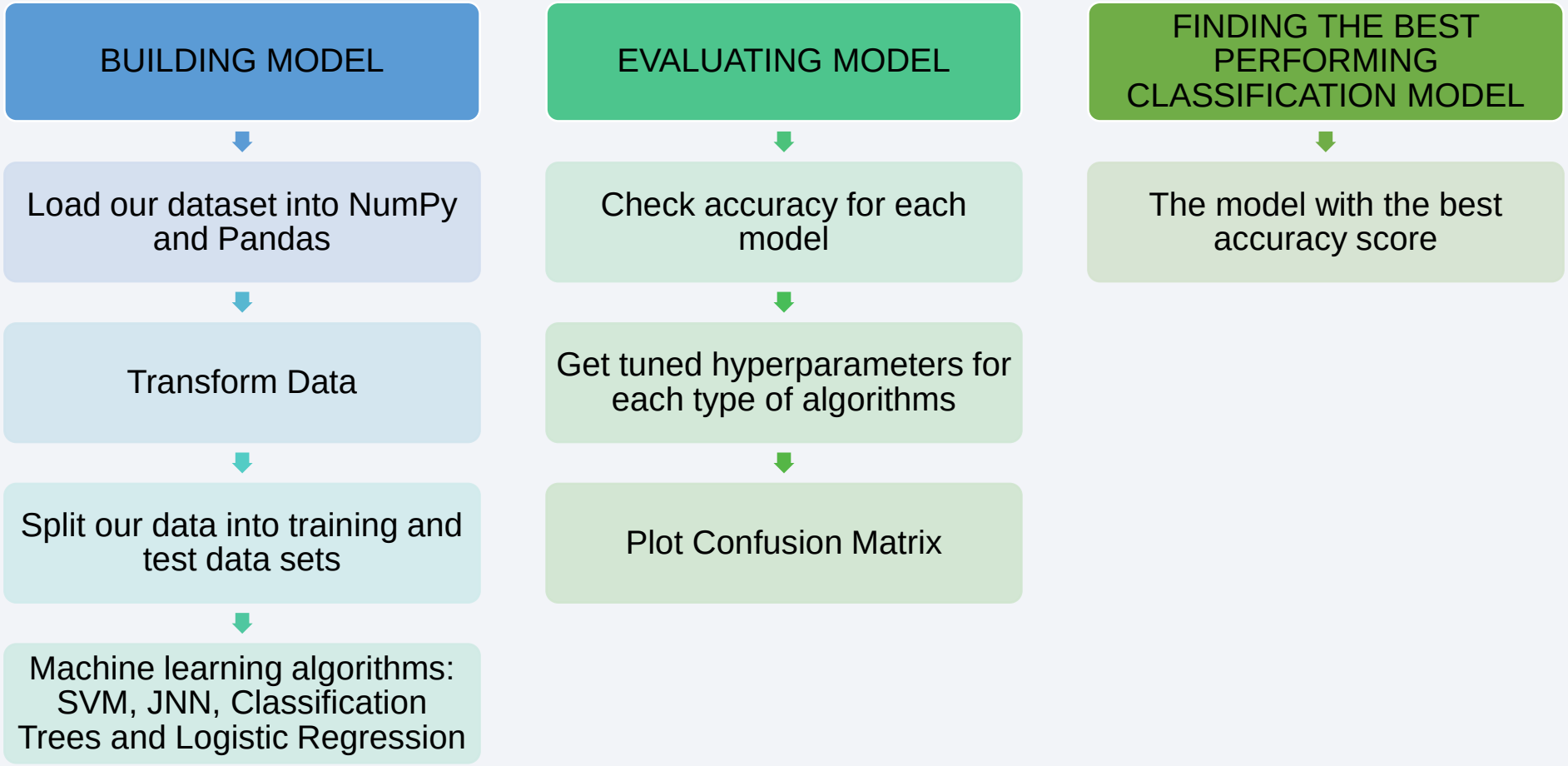
Build a **Plotly Dash application** for users to perform interactive visual analytics on SpaceX launch data in real-time



Notebook: Dashboard with Plotly Dash-GitHub

Predictive Analysis (Classification)

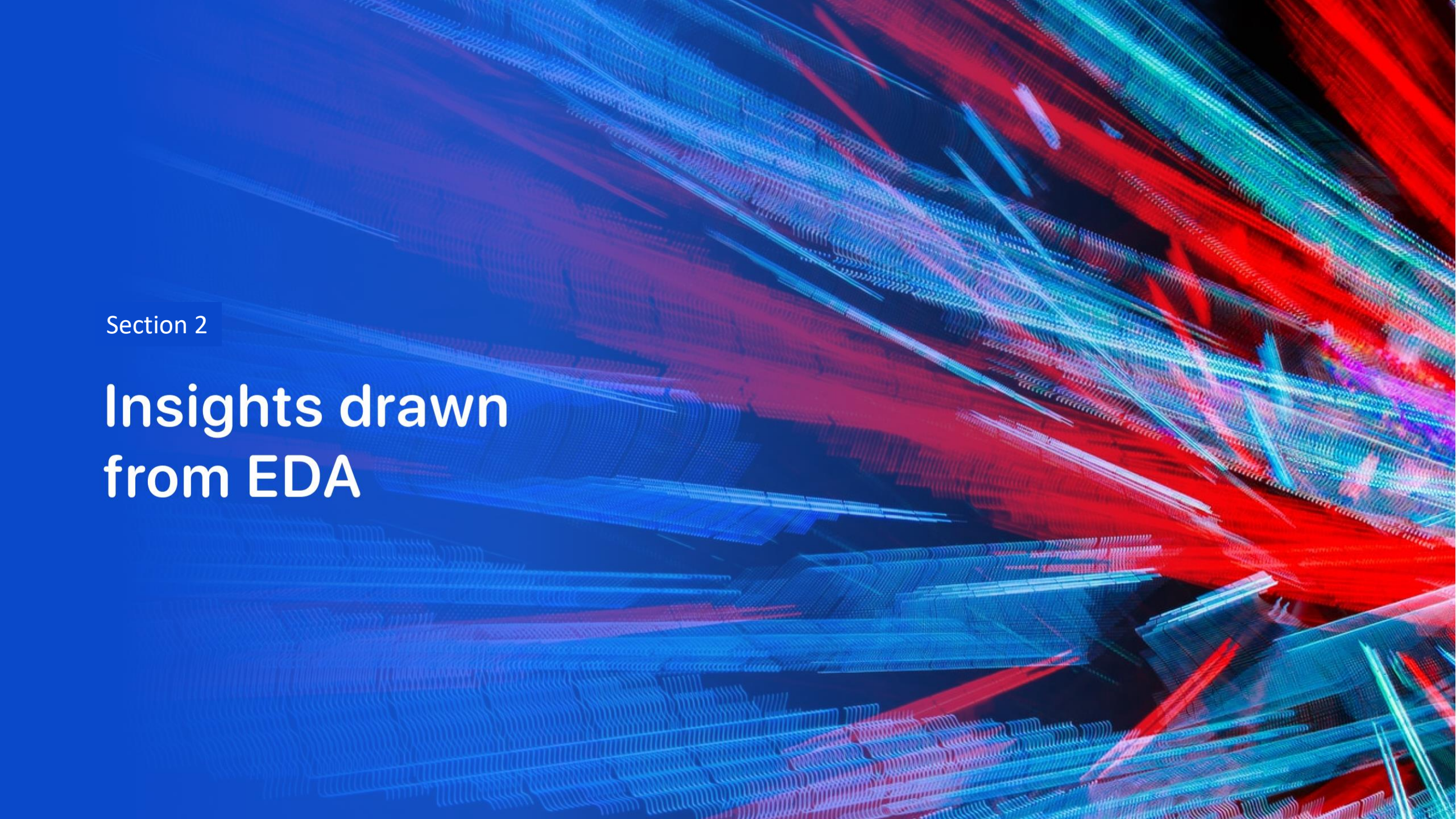
Perform exploratory Data Analysis and determine Training Labels



Notebook: [Data Machine learning-GitHub]

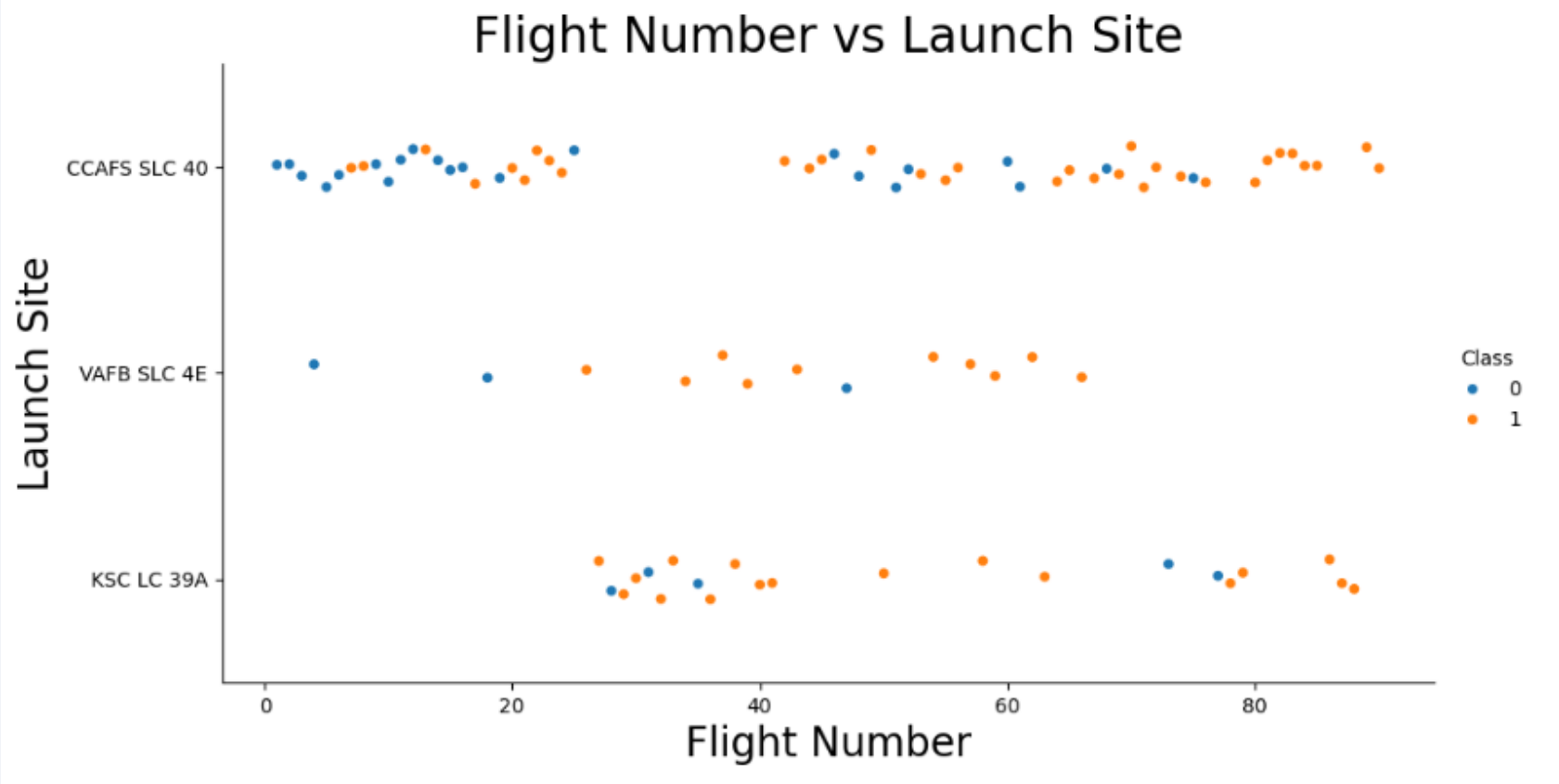
Results

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


Section 2
Insights drawn from EDA

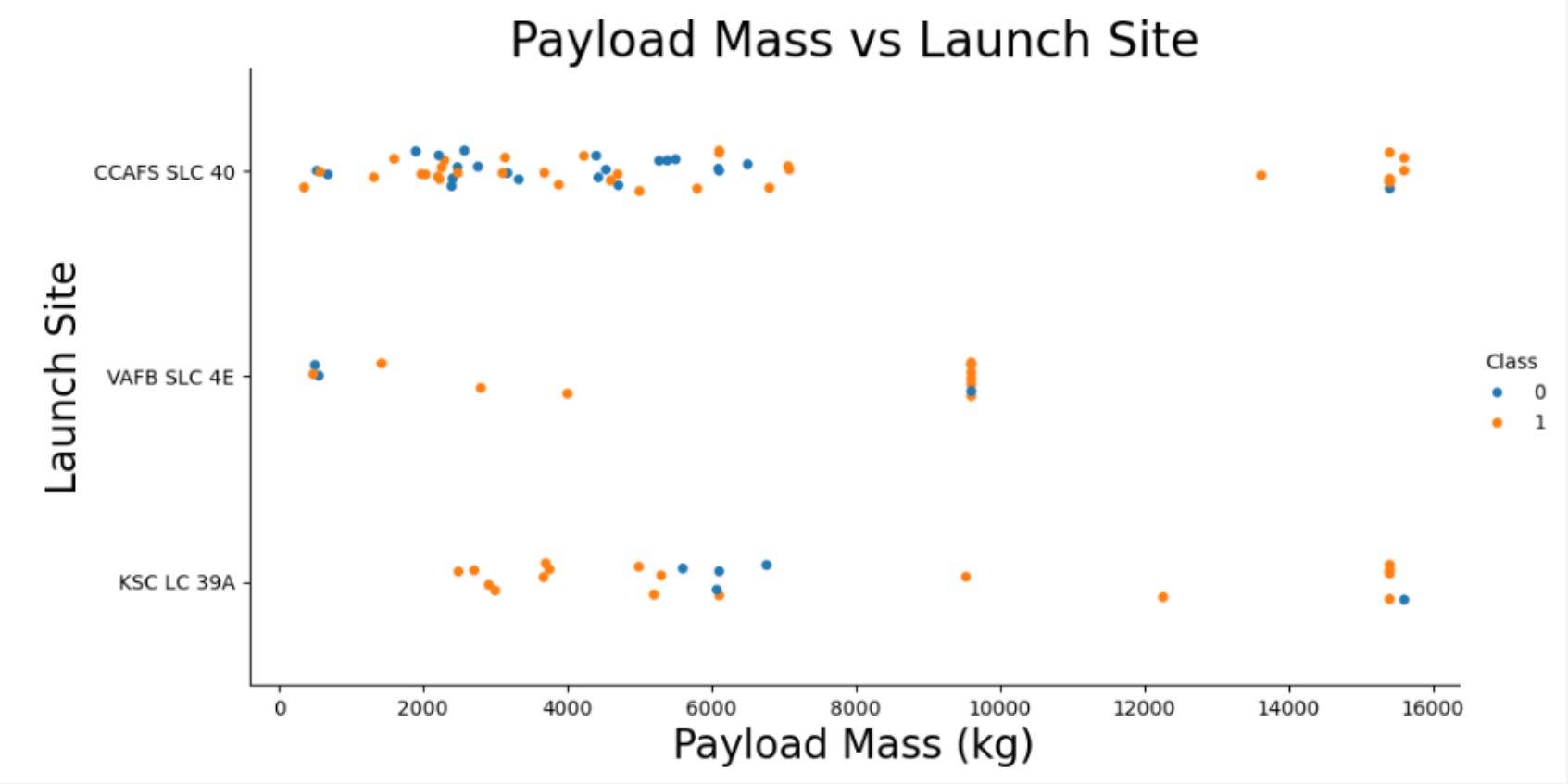
Flight Number vs. Launch Site

- **Relationship between Flight Number and Launch Site**
- Success rate varies with launch site.



- Class 0 (blue) = unsuccessful launch
- Class 1 (orange)= successful launch

Payload vs. Launch Site

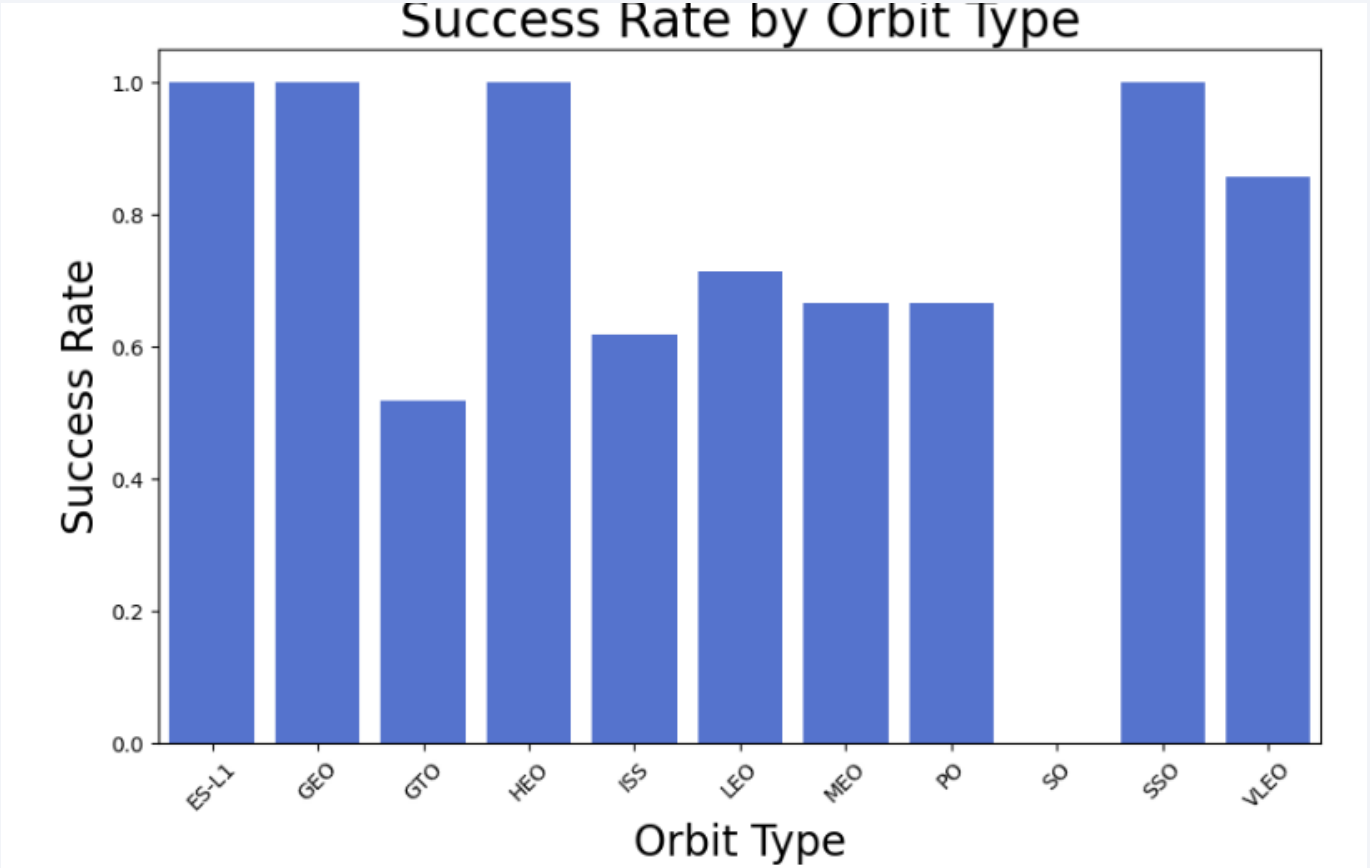


- **Relationship between Payload Mass and Launch Site**
- For VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

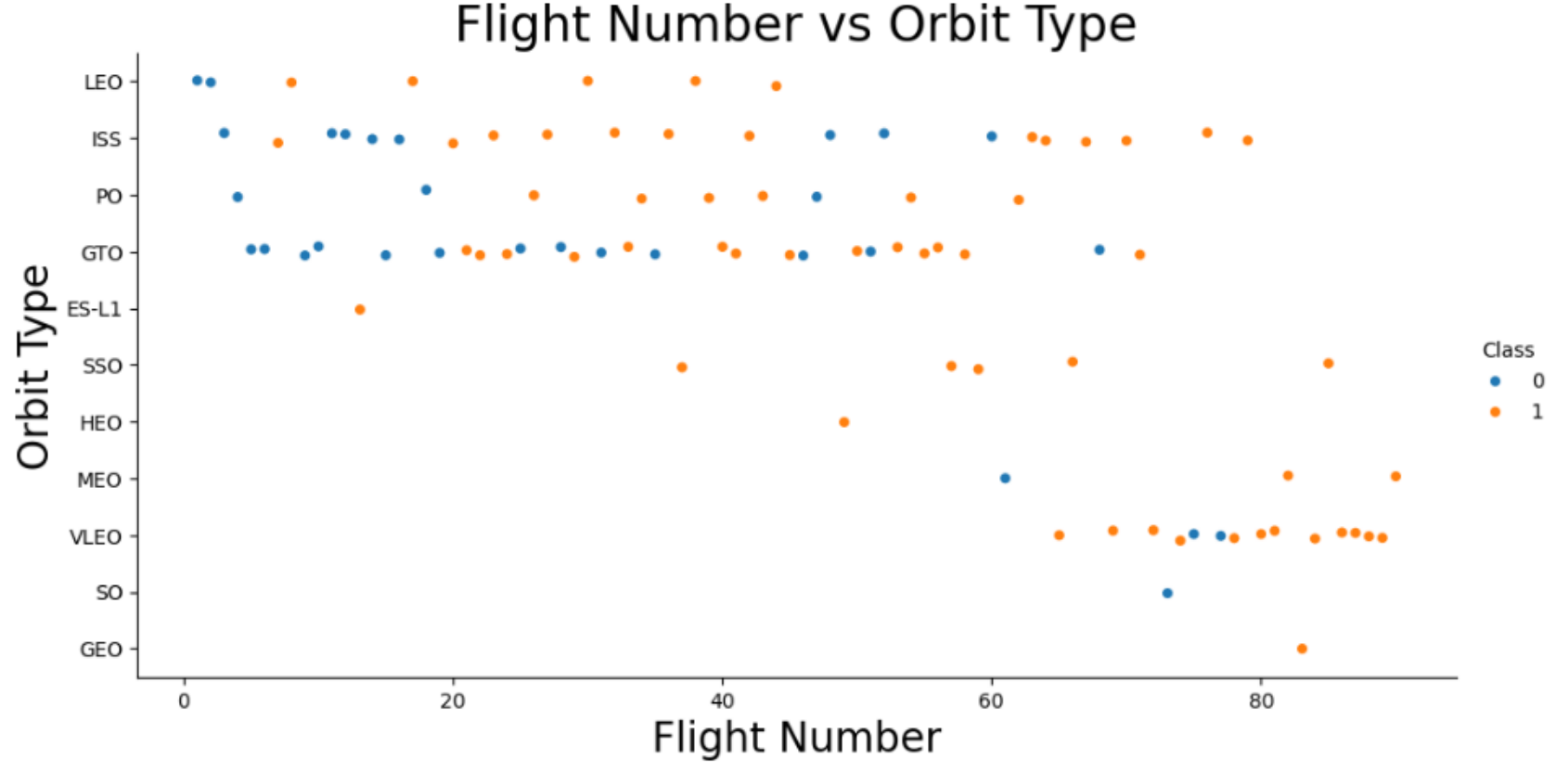
- Class 0 (blue) = unsuccessful launch
- Class 1 (orange)= successful launch

Success Rate vs. Orbit Type

- **Relationship between success rate of each orbit type**
- The orbits with highest success rates are:
  - ES-L1
  - GEO
  - HEO
  - SSO
- SO orbits have no successful first stage landings



Flight Number vs. Orbit Type

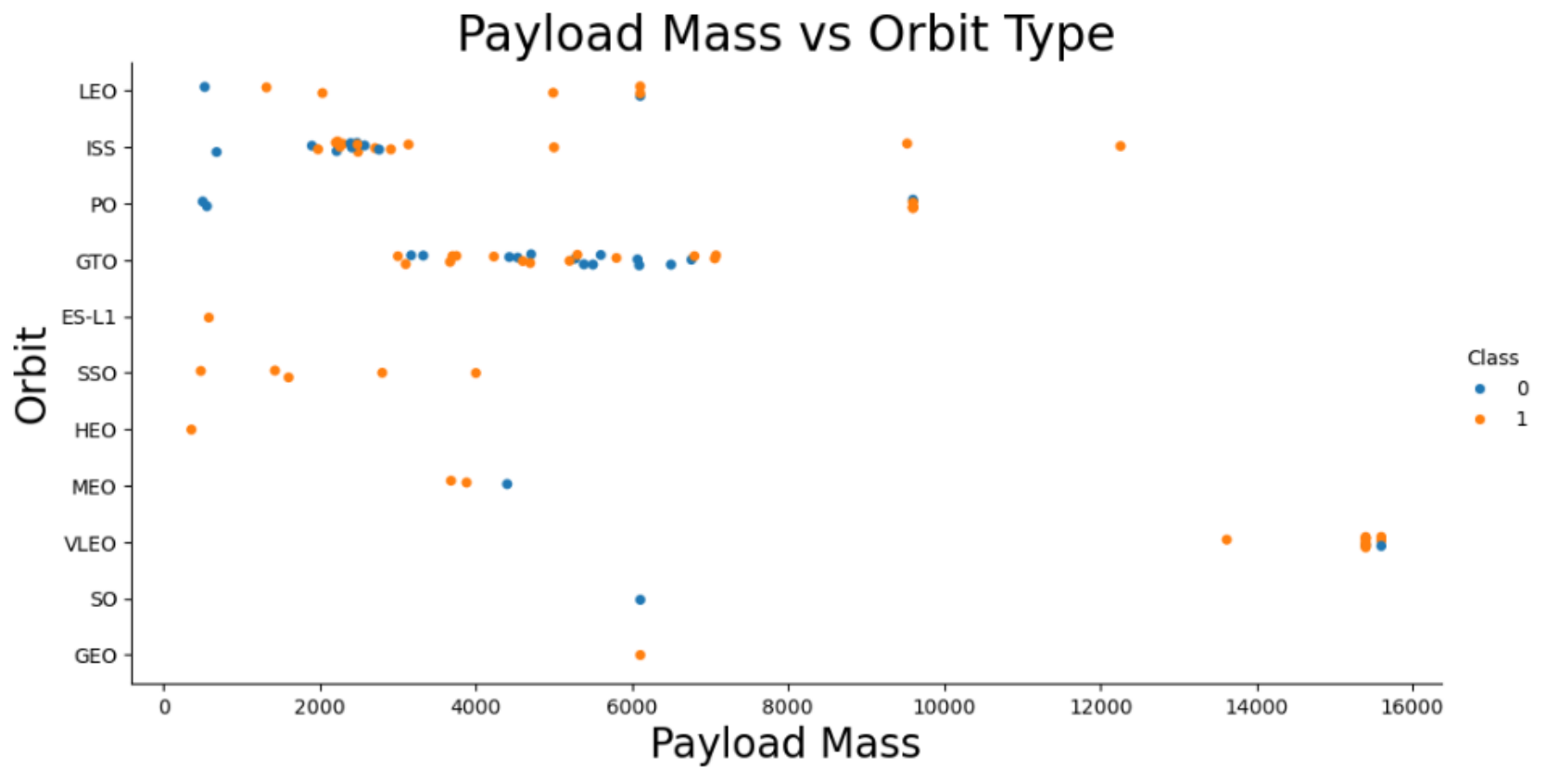


- **Relationship between FlightNumber and Orbit type**
- For the LEO orbit, success seems to be related to the number of flights.
- In the GTO orbit appears to be no relationship between flight number and success.

- Class 0 (blue) = unsuccessful launch
- Class 1 (orange)= successful launch

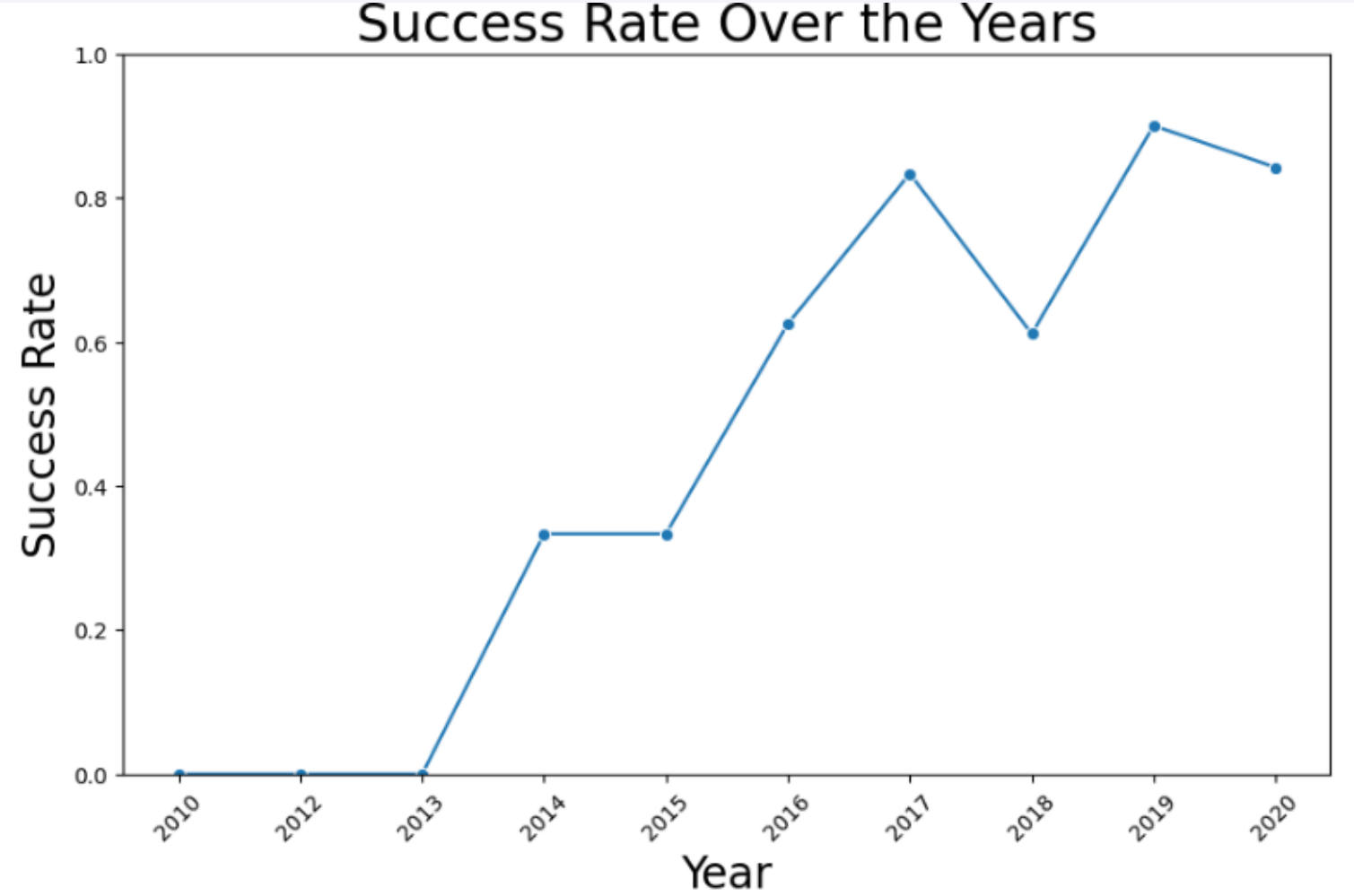
Payload vs. Orbit Type

- **Relationship between Payload Mass and Orbit type**
- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
  - For GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.
- Successful launch appears to have no obvious correlation with payload mass



- Class 0 (blue) = unsuccessful launch
- Class 1 (orange)= successful launch

Launch Success Yearly Trend



- **Visualize the launch success yearly trend**
- Success rate since 2013 kept increasing till 2020

All Launch Site Names

SQL query

- SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;

Explanation

- Using DISTINCT in the query will only show unique values

Output

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

Launch Site Names Begin with 'CCA'

SQL query

- SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5;

Explanation

- Display 5 records where launch sites begin with the string 'CCA‘ using WHERE and LIKE

Output

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

SQL query

- SELECT SUM("PAYLOAD_MASS__KG_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE Customer = 'NASA (CRS)';

Explanation

- Calculates the total payload carried by boosters from NASA

Output

Total_Payload_Mass
45596

Average Payload Mass by F9 v1.1

SQL query

- SELECT AVG("PAYLOAD_MASS__KG_") AS Avarage_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" LIKE 'F9 v1.1%';

Explanation

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

Output

Avarage_Payload_Mass
2534.6666666666665

First Successful Ground Landing Date

SQL query

- SELECT MIN("DATE") AS LaunchDate FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (ground pad)';

Explanation

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

Output

LaunchDate
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

SQL query

- SELECT DISTINCT "Booster_Version", "PAYLOAD_MASS__KG_" FROM SPACEXTABLE WHERE "Landing_Outcome" = 'Success (drone ship)‘ AND "PAYLOAD_MASS__KG_" > 4000 AND "PAYLOAD_MASS__KG_" < 6000;

Explanation

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

Output

Booster_Version	PAYLOAD_MASS__KG_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Total Number of Successful and Failure Mission Outcomes

SQL query

- SELECT "MISSION_OUTCOME", COUNT(*) AS Total FROM SPACEXTABLE WHERE "Mission_Outcome" = 'Success'

Explanation

- Calculates the total number of successful and failure mission outcomes

Output

Mission_Outcome	Total
Success	98

Boosters Carried Maximum Payload

SQL query

- SELECT "Booster_Version", "PAYLOAD_MASS__KG_" FROM SPACEXTABLE WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTABLE);'

Explanation

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

Output

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

SQL query

Explanation

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

```
%%sql SELECT
    CASE substr("Date", 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS Month,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Failure (drone ship)'
AND substr("Date", 0, 5) = '2015';
```

Output

Month	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

SQL query

Explanation

- Ranks the count of landing outcomes between the date 2010-06-04 and 2017-03-20, in descending order

```
%%sql
    SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count
    FROM SPACEXTABLE
        WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY "Landing_Outcome"
        ORDER BY Outcome_Count DESC;
```

Output

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

Section 3

Launch Sites Proximities Analysis

Launch Site Locations

- Location on map using site's latitude and longitude

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

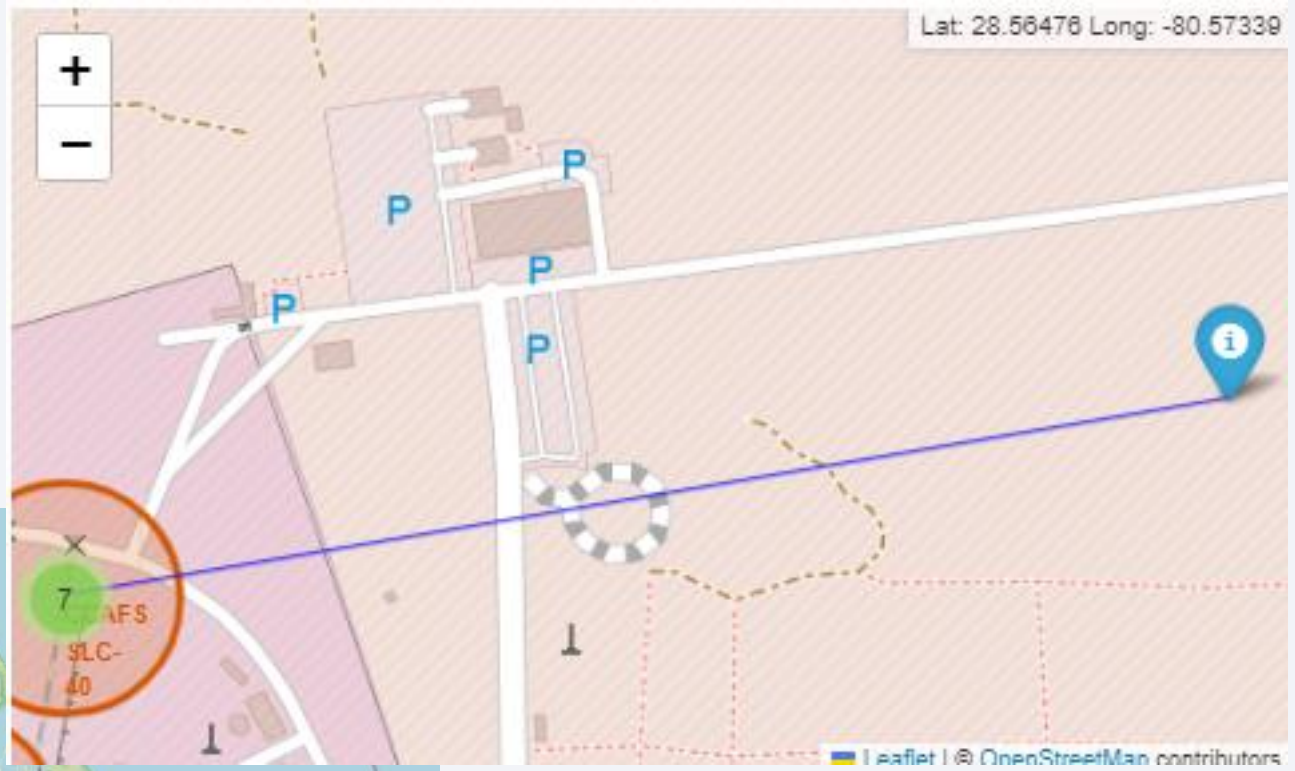
Map Markers of Success/Failed Landings

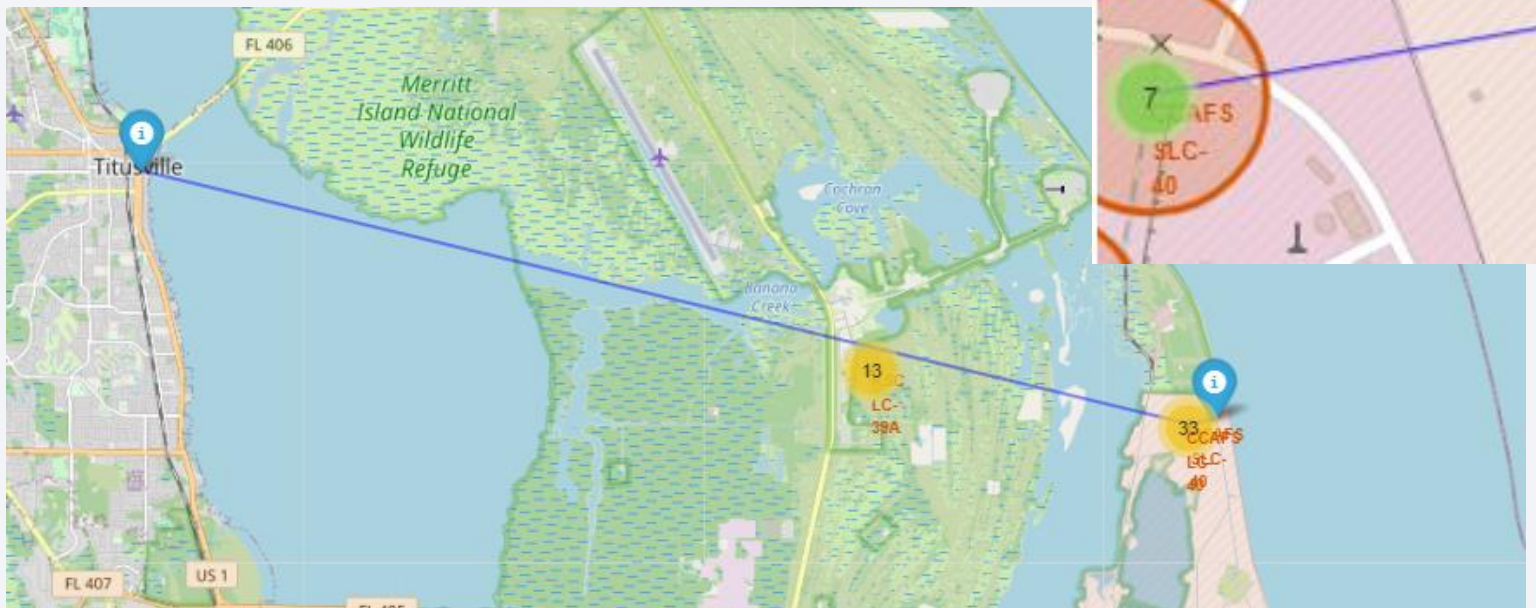
- Marker clusters are use to simplify the map containing many markers with the same coordinate
- The markers display the mission Success or Failure for Falcon 9 first stage landings.
- Image is of CCAFS SLC-40 launch site

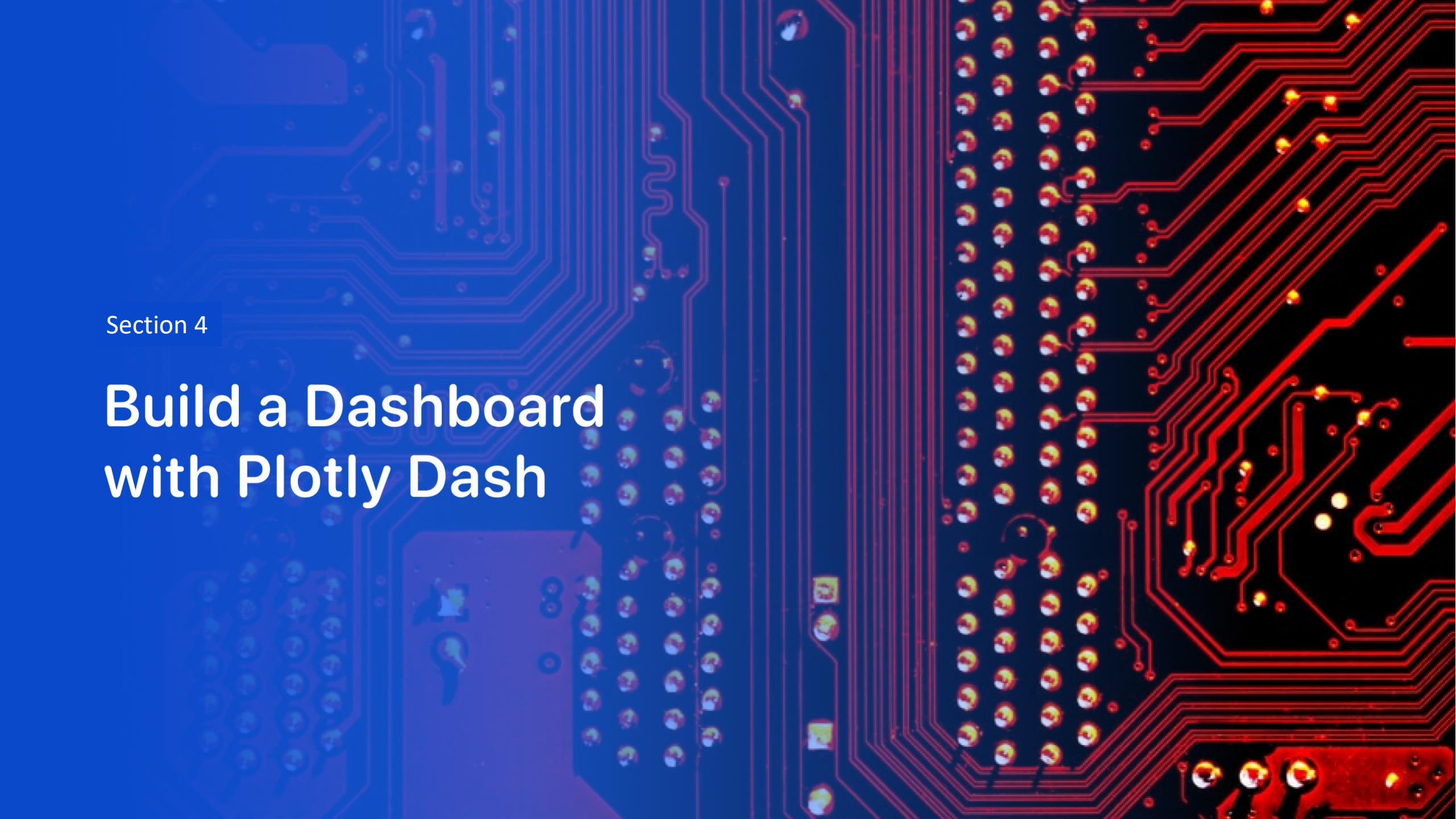
- Red = unsuccessful launch
- Green= successful launch

Distance from Launch Site to Proximities

- The generated folium map shows the launch site to its proximities to a coastline and a railway, with distance calculated and displayed

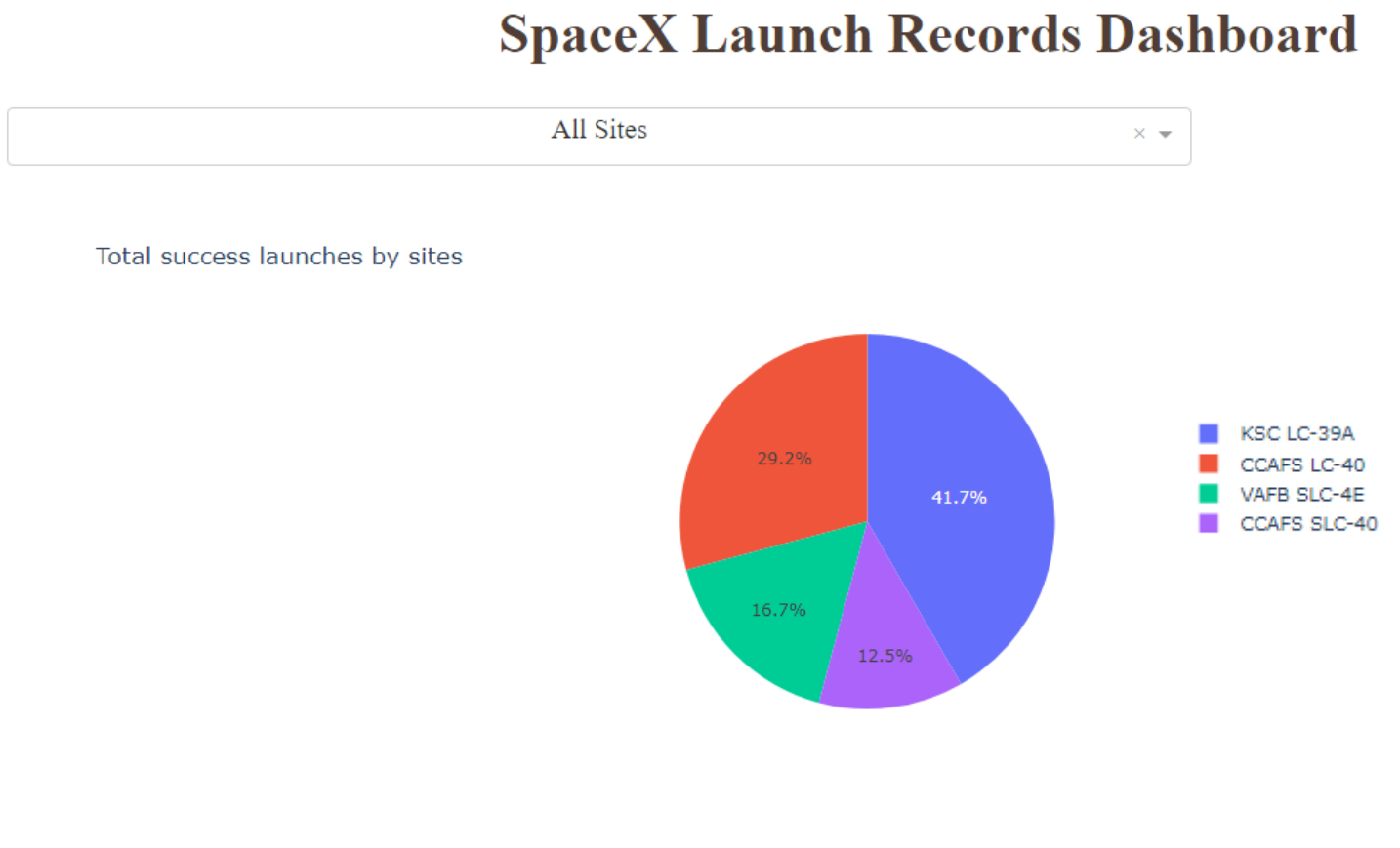




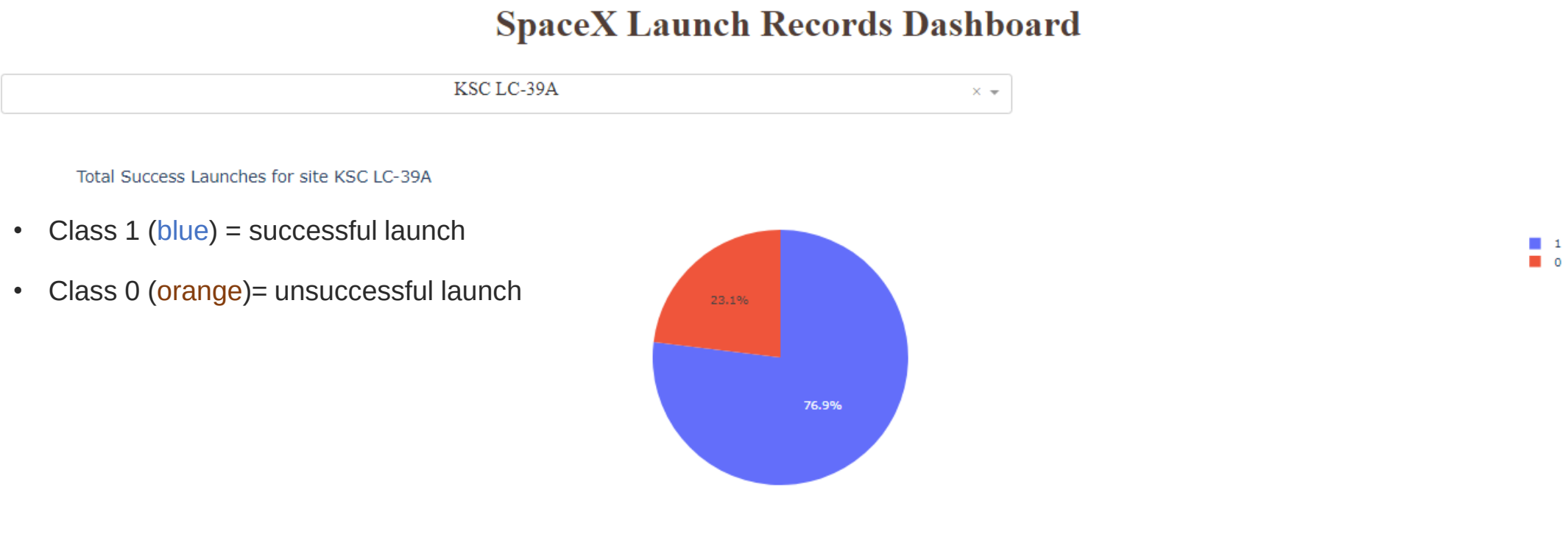

Section 4
Build a Dashboard with Plotly Dash

Total success launches all sites

- We have four different launch sites
  - KSC LC-39A **41.7%**
  - CCAFS LC-40 **29.2%**
  - VAFB SLC-4E **16.7%**
  - CCAFS SLC-40 **12.5%**



Total success lauches for KSC LC-39A



- Class 1 (blue) = successful launch
- Class 0 (orange)= unsuccessful launch

- KSC LC-39A is the launch site with highest launch success ratio
- When in the dropdown a specific launch site is selected, a pie chart is show for the success (class=1) and failed (class=0) lauches.

Payload vs. Launch Outcome

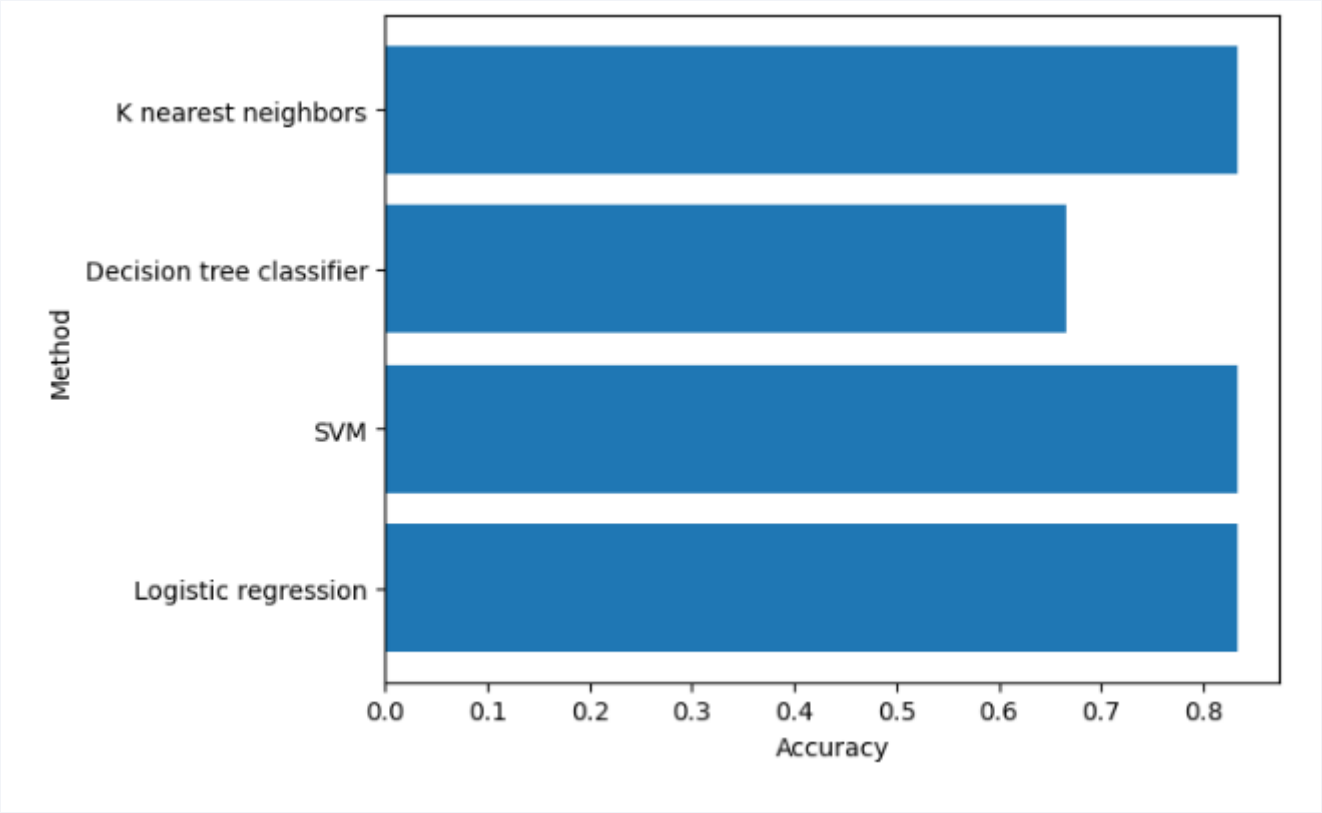
- Scatter plot for all sites with Payload range
  - 0-3k
  - 3k-6k
  - 6k-10k
- Range slider to select different payload range

Section 5

Predictive Analysis (Classification)

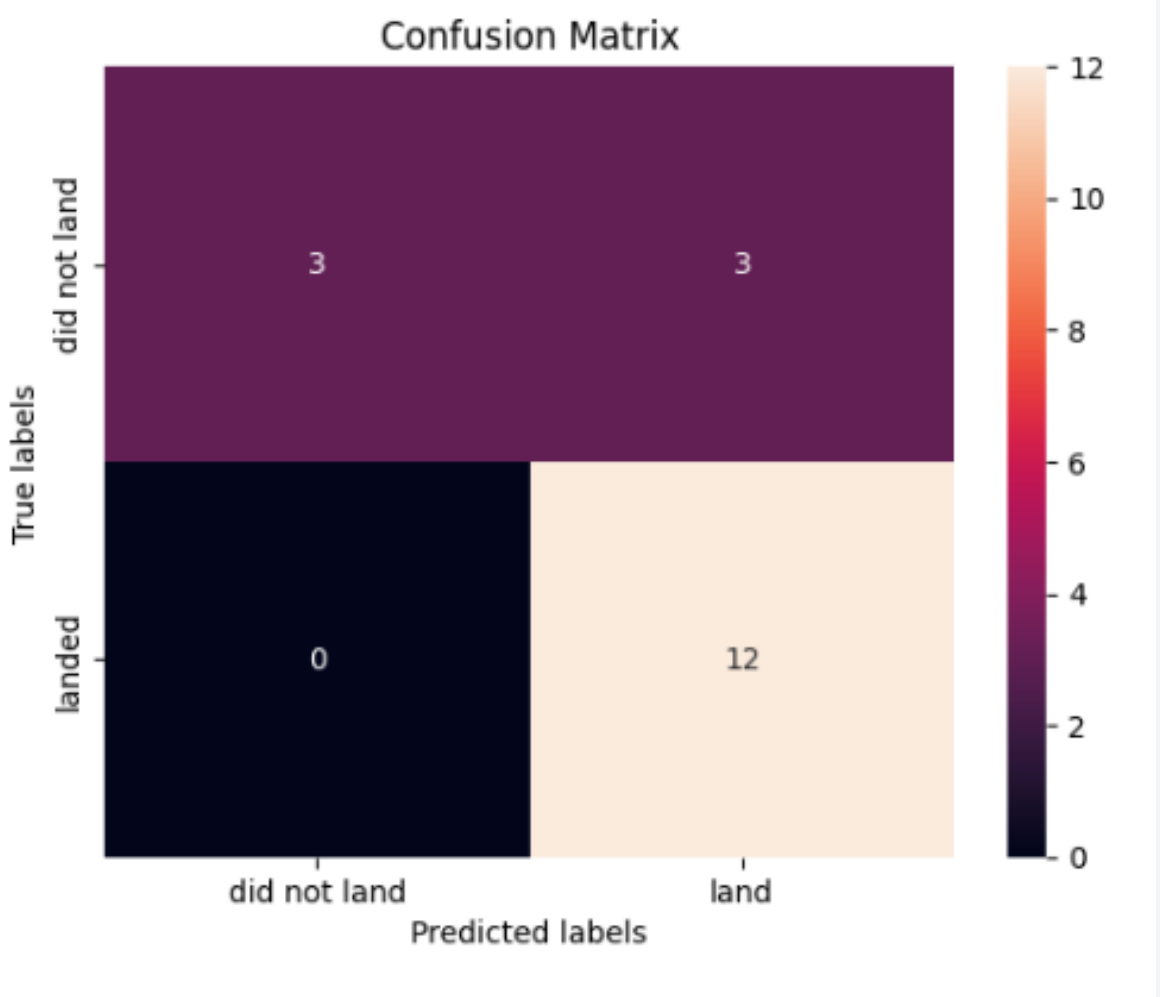
Classification Accuracy

- Bar chart of the accuracy for all built classification models
- The best performing model is: Logistic regression with an accuracy of 0.83
- Decision Tree model performed poorly relative to the other models



Confusion Matrix

- Visualization of the results using a confusion matrix to assess performance in terms of true positives, false positives, etc.
- The logistic regression can distinguish between the different classes. We see that the problem is false positives.
  - True Postive - 12
  - False Postive - 3



Conclusions

- The best performing model is: Logistic regression with an accuracy of 0.83
  - While the logistic regression model shows good performance with a reasonably high number of true positives, the false positives indicate an area for improvement.
- Success rate varies with launch site.
- This study has provided a comprehensive examination of the data science process, from data gathering to predictive modeling.

Appendix

- **Jupyter Notebooks**
  - Notebook: Data Collection API -GitHub
  - Notebook: Data Collection Webscraping – GitHub
  - Notebook: Data Wrangling-GitHub
  - Notebook: Data Visualization-GitHub
  - Notebook: Data EDA with SQL-GitHub
  - Notebook: Data Map Folium-GitHub
  - Notebook: Dashboard with Plotly Dash-GitHub

Thank you!