IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jasper Robbins
December 2024

Outline

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

Executive Summary

Summary of Methodologies

- **Data Collection:** SpaceX APIs and web scraping public sources (e.g., Wikipedia)
- **Data Wrangling:** Cleaned and structured data
- **Data Analysis:** SQL, visualization, and exploratory methods
- **Geospacial Data Analysis:** Interactive mapping with Folium
- **Dashboarding:** Dynamic dashboards with Plotly Dash
- **Predictive Modeling:** Applied Logistic Regression, SVM, Decision Trees, and KNN for classification based predictions

Summary of Results

- Collected & Structured data from SpaceX APIs and public sources for Analysis
- Identified key trends using SQL queries, geospatial mapping, and dynamic dashboards
- Achieved ~83% accuracy with our best predictive model

Introduction

Project background and context

In competition with SpaceX, a rival rocket company ‘SpaceY’, aims to leverage data driven insights to enhance its understanding about SpaceX Falcon 9 rocket first-stage landings. The goal is to analyze publically available data to make informed predictions about the success of Falcon 9 first-stage landings, helping ‘SpaceY’ strategize and potentially improve its own systems.

Problems to answer

- How do key factors such as orbit, payload mass, launch site, and other variables influence the success rate of Falcon 9 first-stage landings?
- Can advanced data visualization techniques reveal patterns and insights that are not readily apparent through traditional coding or statistical analysis?
- Can predictive machine learning models accurately forecast the outcomes of future Falcon 9 first-stage landings based on historical data?

Section 1

Methodology

Methodology

Data Collection

- Gathered historical data from SpaceX APIs and supplemented it with web-scraped information from public sources (e.g., Wikipedia).
- Combined and integrated data into unified datasets for improved analysis.

Data Wrangling & Processing

- Cleaned missing values, resolved inconsistencies, standardized data formats, and engineered features to improve downstream analysis.

Exploratory Data Analysis

- Utilized SQL and Python-based visualizations to investigate relationships between variables like orbit type, payload mass, and landing success.
- Identified key trends, such as the impact of orbit type and launch site on landing outcomes.

Interactive Visual Analytics

- Developed geospatial maps with Folium to examine launch and landing site patterns and assess the importance of location as a feature.
- Created a Plotly Dash dashboard for dynamic exploration of all launch sites, including specific success ratios and metrics per launch site.

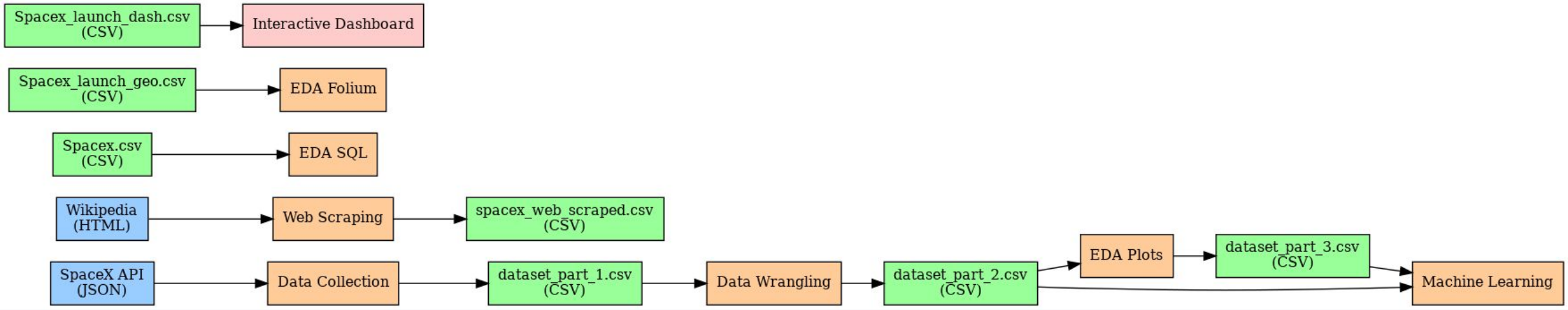
Predictive Modeling

- Assessed various classification algorithms (Logistic Regression, SVM, Decision Trees, KNN) for predictive analysis.
- Optimized models through hyperparameter tuning using grid search.

Data Collection

Data Sources

- A dataset retrieved from an IBM-hosted API response, containing launch data in JSON format.
- A publicly available Wikipedia page with launch data in HTML tables.
- Supplementary datasets in CSV format, provided as part of the course materials.



Data Collection – SpaceX API

SpaceX API Call: Retrieve launch data using Python's requests library, accessing relevant SpaceX API endpoints.

Create DataFrame: Convert nested JSON data into a tabular format using pandas.json_normalize().

Filter Data: Filter the DataFrame to include only Falcon 9 launches using boolean indexing.

Handle Missing Values: Address missing values in PayloadMass by imputation or row removal.

Save Processed Data: Export the cleaned dataset to dataset_part_1.csv using .to_csv().

Github: Data Collection

Data Collection - Scraping

Request Call: Fetch the HTML content of the Wikipedia page using Python's requests library.

Extract Relevant Data: Identify and extract data from the HTML tables relevant to Falcon 9 launches using BeautifulSoup.

Parse and Create DataFrame: Parse the launch data table and convert it into a structured DataFrame with relevant columns.

Save Extracted Data: Export the processed DataFrame to spacex_web_scraped.csv using to_csv().

Github: Web Scraping

Data Wrangling

Input File: Start with dataset_part_1.csv containing cleaned and structured launch data.

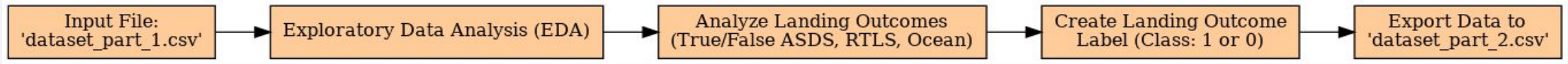
Exploratory Data Analysis (EDA): Perform analysis to identify patterns and relationships in the data.

Analyze Landing Outcomes: Examine outcomes such as True/False ASDS, RTLS, and Ocean landings.

Create Landing Outcome Label: Generate a Class column where 1 indicates successful landings and 0 indicates unsuccessful landings.

Export Processed Data: Save the updated dataset with labels to dataset_part_2.csv.

Github: Data Wrangling



EDA with Data Visualization

Input File: Start with dataset_part_2.csv containing cleaned data from the previous step.

Scatter Plots: Create scatter plots to analyze relationships:

- Flight Number vs Payload Mass.
- Flights vs Launch Site.
- Payload Mass vs Launch Site.

Bar Chart: Plot success rate vs orbit type to analyze orbit-specific performance.
Scatter Plots: Visualize additional relationships:

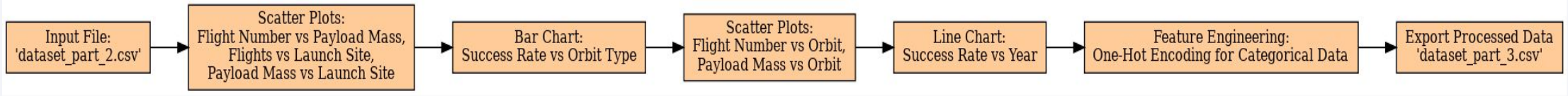
- Flight Number vs Orbit.
- Payload Mass vs Orbit.

Line Chart: Plot success rate trends over years to observe historical patterns.

Feature Engineering: Perform one-hot encoding to prepare categorical data for modeling.

Export Processed Data: Save the final dataset to dataset_part_3.csv.

Github: EDA Visualization

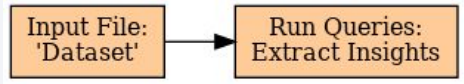


EDA with SQL

Input File: Start with the dataset as the source for query operations.

Run Queries: Perform SQL or data-frame-based queries to extract specific insights and answer targeted questions.

Github: EDA SQL



Build an Interactive Map with Folium

Input File: Used course given file spacex_launch_geo.csv with site coordinates and outcomes.

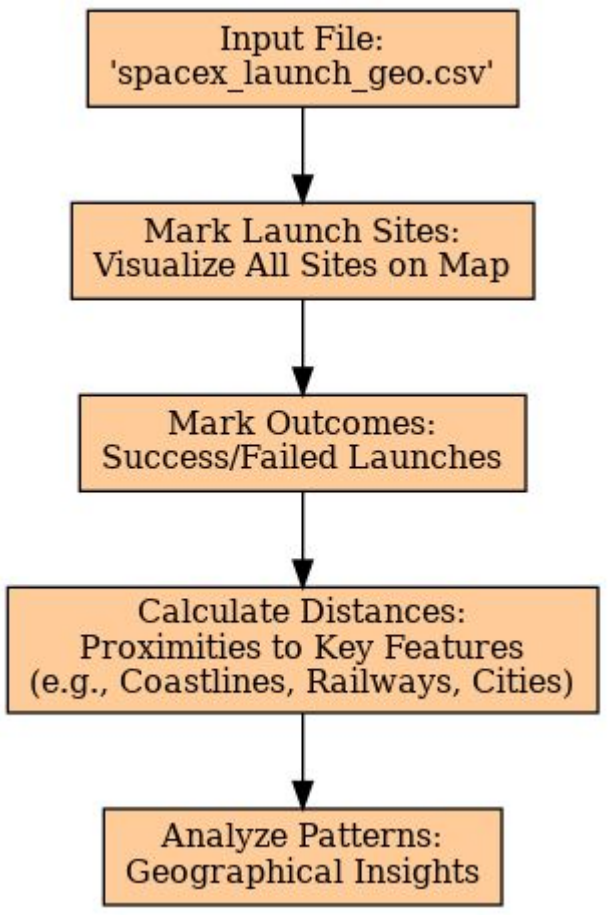
Launch Site Markers: Added circles and labeled markers to pinpoint launch sites, enabling clear geographic visualization.

Launch Outcomes: Used color-coded markers (green for success, red for failure) to show site-specific performance.

Distance Lines: Added lines and distance markers to key features (coastlines, cities, etc.) for proximity analysis.

Geographical Patterns: Derived insights on site suitability based on proximity to infrastructure and coastal areas.

Github: Folium



Build a Dashboard with Plotly Dash

Dashboard Title: Displayed as "SpaceX Launch Records Dashboard" with center alignment and custom styling.

Dropdown Menu: Allows selection of a launch site (or all sites) for filtering the data, with a default value of "All Sites".

Pie Chart: Displays the distribution of launch successes and failures, dynamically updated based on the selected site.

Range Slider: Enables filtering of data by payload mass within a specified range, defaulting to the dataset's min and max payload values.

Scatter Plot: Visualizes the correlation between payload mass and launch success, color-coded by booster version and filtered by site and payload range.

Github: Dashboard

Predictive Analysis (Classification)

Input Files: Utilize dataset_part_2.csv and dataset_part_3.csv for model training and evaluation.

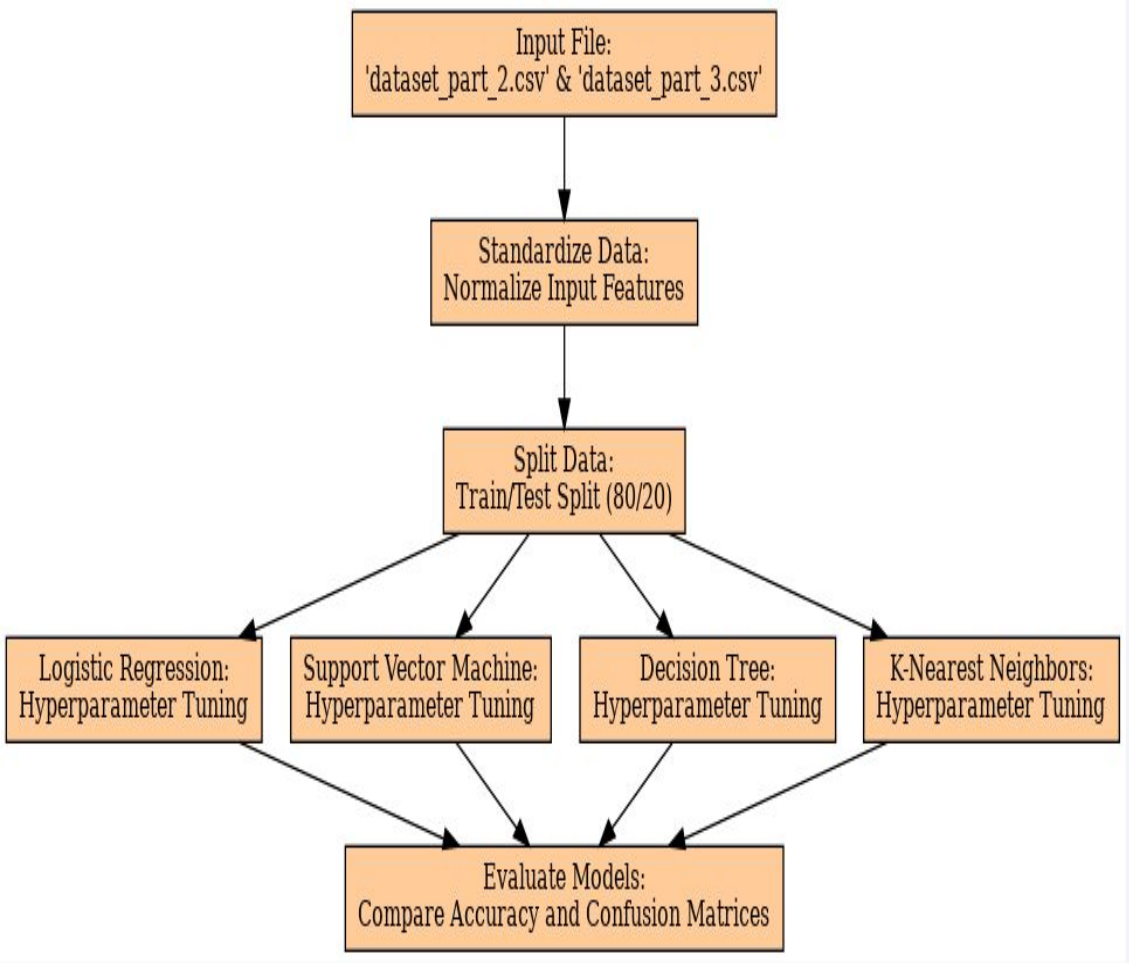
Standardization: Normalize input features to ensure uniform scale for better model performance.

Split Data: Divide the dataset into training and test sets with an 80/20 ratio.

Model Training: Perform hyperparameter tuning for the following algorithms:

- Logistic Regression
- Support Vector Machine
- Decision Tree
- K-Nearest Neighbors

Evaluation: Compare model accuracies and confusion matrices to determine the best-performing algorithm.



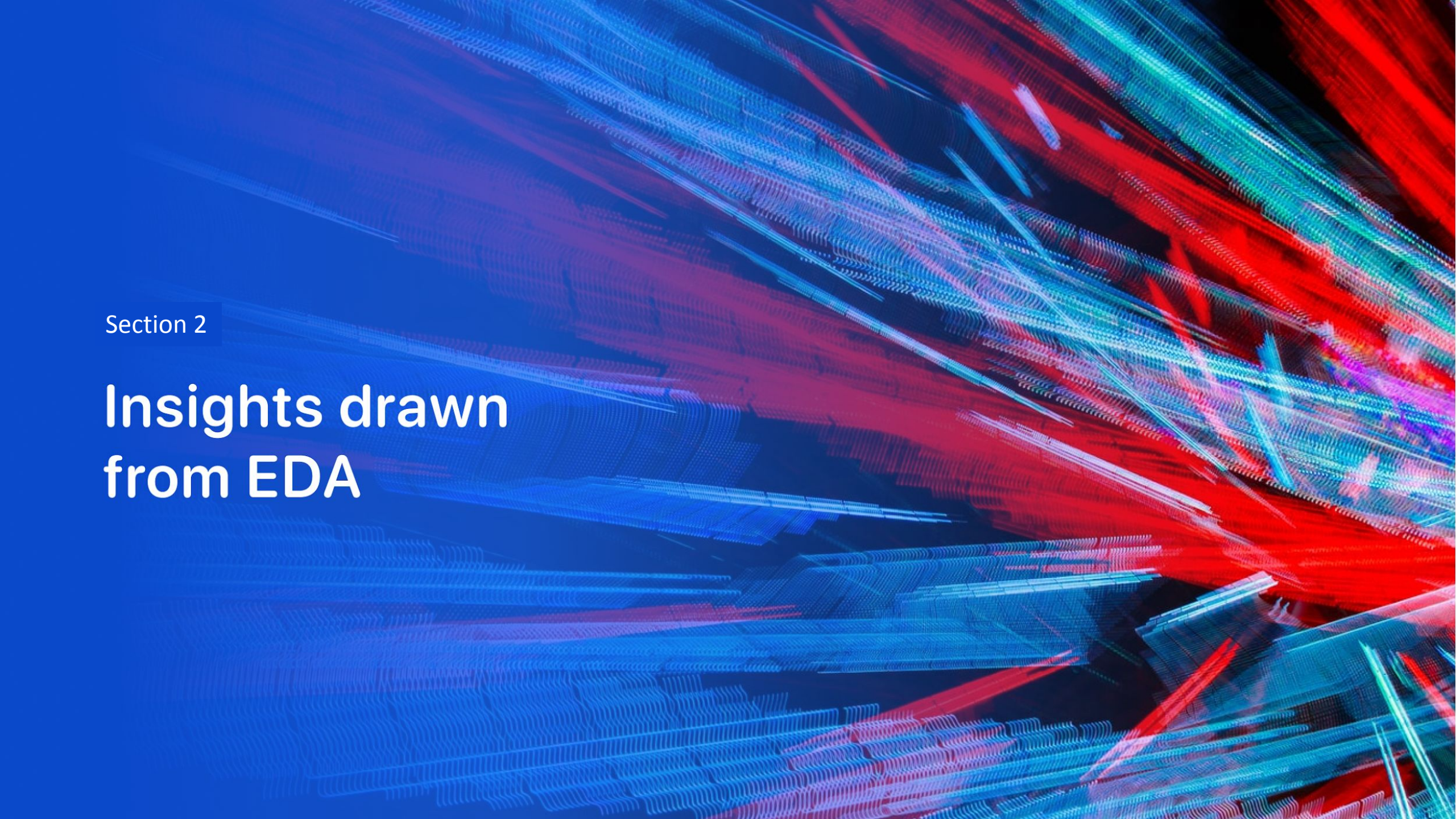
Github: Machine Learning

Results

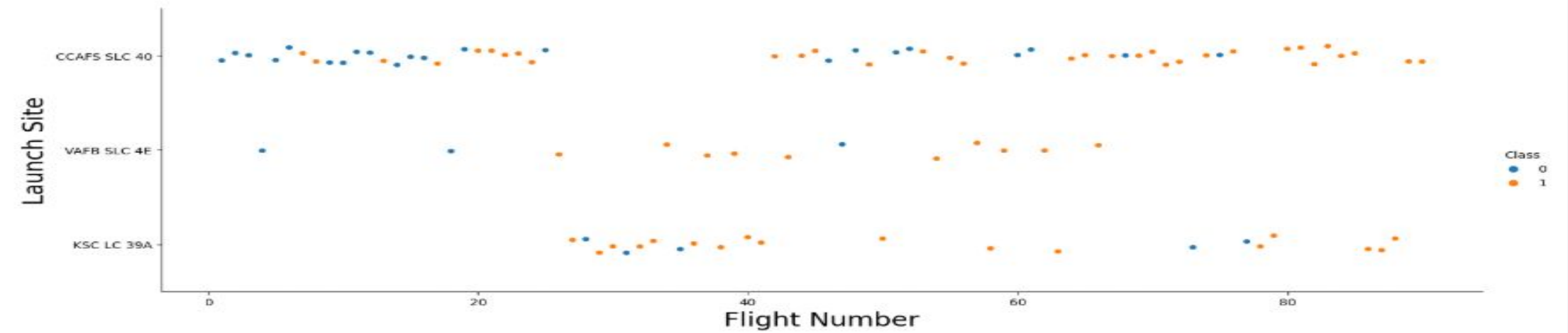
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

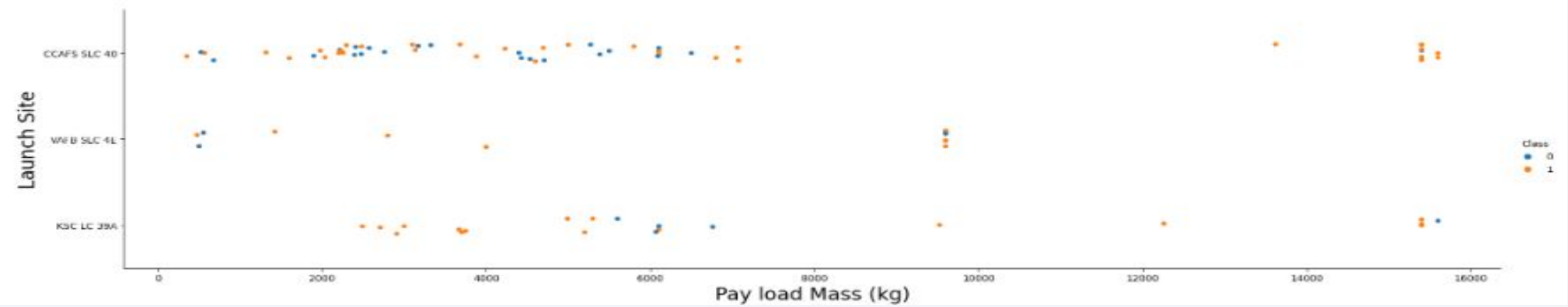


Flight Number vs. Launch Site



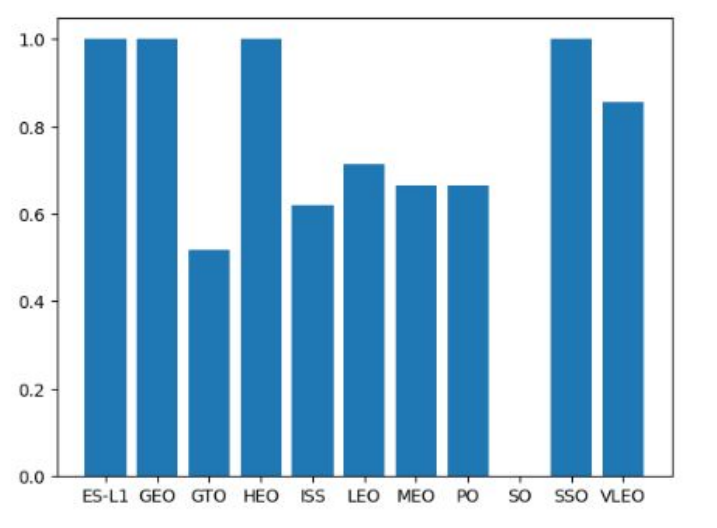
- CCAFS SLC 40 contributed to the most launches, with significant improvements the more flights were recorded.
- VAFB SLC 4E contributed the least launches, with majority success, although a small sample size.
- KSC LC 39A contributed a moderate amount of launches, with a general trend of increasing success rate.

Payload vs. Launch Site



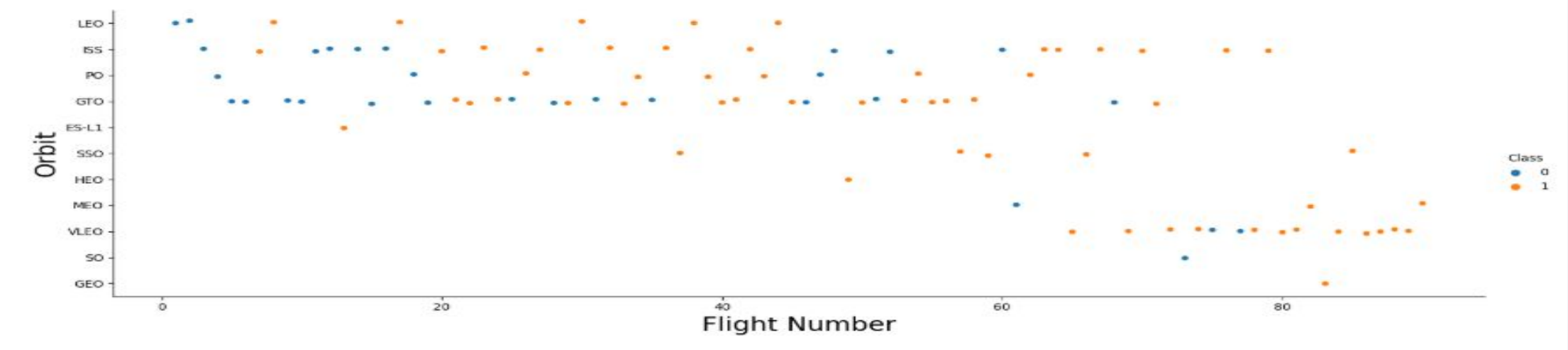
- CCAFS SLC 40 included the most variety in terms of payload mass, with most failures being at lighter weights.
- VAFB SLC 4E included no surprising trends, similar failure rates compared to other sites, with far less launches total.
- KSC LC 39A followed a similar trend to CCAFS SLC 40, with success growing with higher payload mass.

Success Rate vs. Orbit Type



- The largest outlier being SO, which indicates 0 successful missions.
- Other insights, we see ES-L1, HEO, SSO all with 100% success rate, VLEO not far behind, with about a 90% success rate.
- GTO, ISS, LEO, MEO, PO, all contain moderate success, ranging between 50% and 70% success rate.

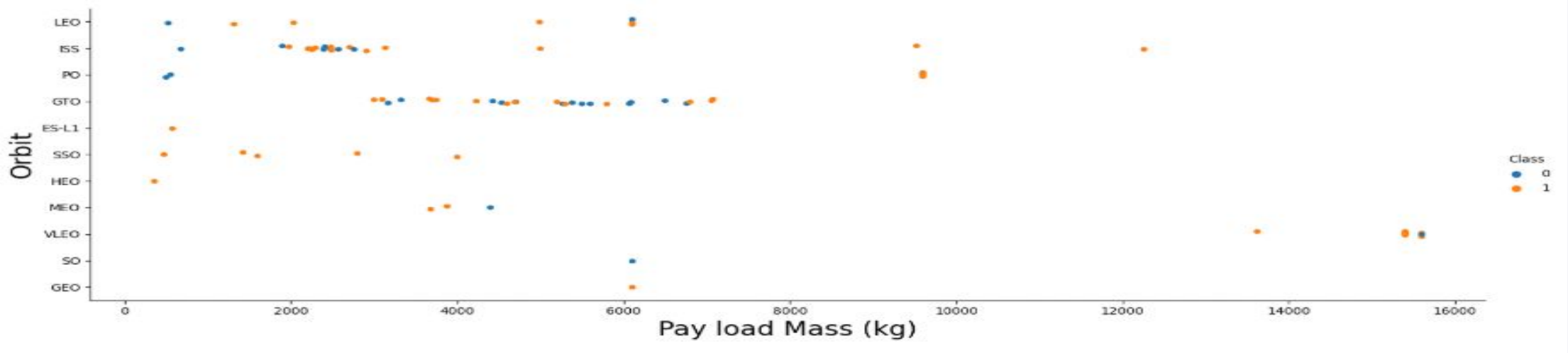
Flight Number vs. Orbit Type



- While not surprising, we see most failures being in the earlier flight numbers, meaning as more flights occur, the higher rate of success.

- This trend is most apparent with VLEO and LEO orbit types, where we see very few failures as they grow in flight numbers.

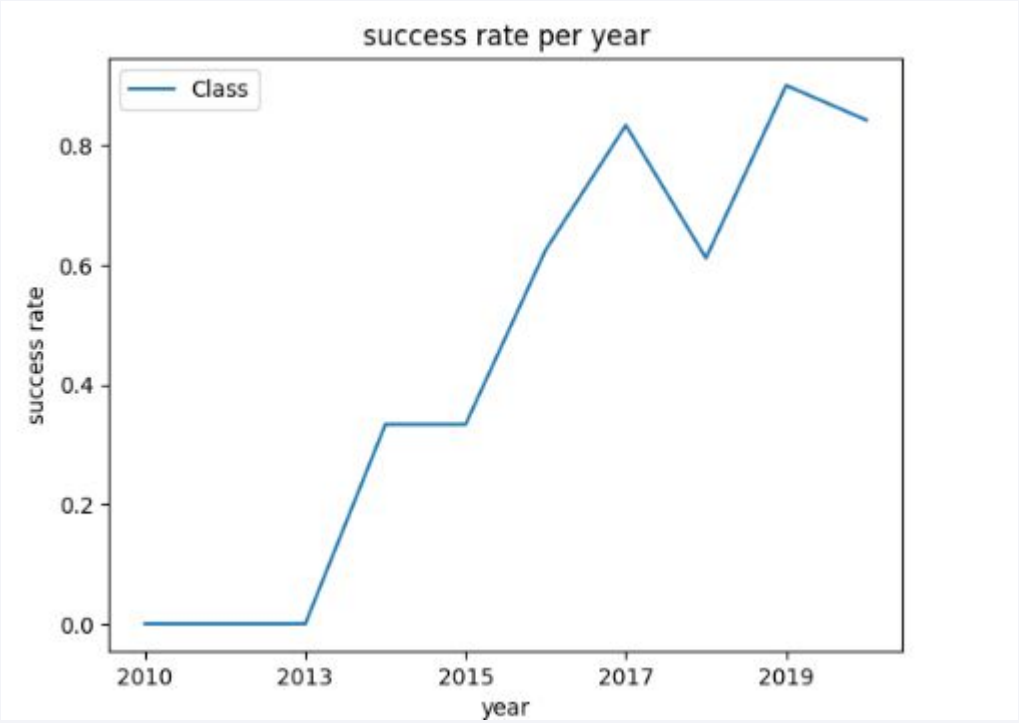
Payload vs. Orbit Type



- We see a high negative impact of mass on GTO orbit

- We see many attempts of the same mass to ISS, indicating some external factors that may be keeping SpaceX from going heavier for that orbit, although likely that is just due to ISS being a popular orbit.

Launch Success Yearly Trend



- As years increase, we see general improvement, even though drops do occur, this indicates strong and steady improvements in the field.

- We do see drops, as a matter of fact we see the data ends in a drop, although given the trajectory of the graph,it is likely the trend will fix itself.

All Launch Site Names

```
[11]: %sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
 * sqlite:///my_data1.db
Done.
[11]:
```

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

- We use DISTINCT to locate unique launch sites

Launch Site Names Begin with 'CCA'

```
[15]: %sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE "CCA%" LIMIT 5;
 * sqlite:///my_data1.db
Done.
```

[15]:

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

- Given names will only start with CCA, then have potentially anything else, we use LIKE to locate just those that start with it. We use LIMIT as the original prompt tells us to just grab 5.

Total Payload Mass

- We first make sure we are grabbing the correct values, then minimize it to just calculate the sum of the payload masses.

```
%sql SELECT Date, Booster_Version, Launch_Site, Payload, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE CUSTOMER = "NASA (CRS)";
```

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

Date	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_
2012-10-08	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500
2013-03-01	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677
2014-04-18	F9 v1.1	CCAFS LC-40	SpaceX CRS-3	2296
2014-09-21	F9 v1.1 B1010	CCAFS LC-40	SpaceX CRS-4	2216
2015-01-10	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395
2015-04-14	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898
2015-06-28	F9 v1.1 B1018	CCAFS LC-40	SpaceX CRS-7	1952
2016-04-08	F9 FT B1021.1	CCAFS LC-40	SpaceX CRS-8	3136
2016-07-18	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257
2017-02-19	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490
2017-06-03	F9 FT B1035.1	KSC LC-39A	SpaceX CRS-11	2708
2017-08-14	F9 B4 B1039.1	KSC LC-39A	SpaceX CRS-12	3310
2017-12-15	F9 FT B1035.2	CCAFS SLC-40	SpaceX CRS-13	2205
2018-04-02	F9 B4 B1039.2	CCAFS SLC-40	SpaceX CRS-14	2647
2018-06-29	F9 B4 B1045.2	CCAFS SLC-40	SpaceX CRS-15	2697
2018-12-05	F9 B5B1050	CCAFS SLC-40	SpaceX CRS-16	2500
2019-05-04	F9 B5B1056.1	CCAFS SLC-40	SpaceX CRS-17, Starlink v0.9	2495
2019-07-25	F9 B5 B1056.2	CCAFS SLC-40	SpaceX CRS-18, AMOS-17	2268
2020-03-07	F9 B5 B1059.2	CCAFS SLC-40	SpaceX CRS-20, Starlink 5 v1.0	1977
2020-12-06	F9 B5 B1058.4	KSC LC-39A	SpaceX CRS-21	2972

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER="NASA (CRS)";
```

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

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE "F9 V1.1%";
 * sqlite:///my_data1.db
Done.
AVG(PAYLOAD_MASS__KG_)
2534.6666666666665
```

- While we could use part of the prompt from the last slide and divide by total valid entries, SQL offers an AVG line that simplifies this entire thing.

First Successful Ground Landing Date

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

Hint:Use min function

```
36]: %sql SELECT DATE FROM SPACEXTBL WHERE LANDING_OUTCOME = "Success" ORDER BY DATE ASC LIMIT 1;
```

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

36]:

Date
2018-07-22

- Sorting dates and grabbing just 1

Successful Drone Ship Landing with Payload between 4000 and 6000

```
[41]: sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' AND Payload_MASS__KG_ BETWEEN 4000 AND 6000;
 * sqlite:///my_data1.db
Done.
```

[41]:

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

- Successful drone ship landings will always have the outcome ‘Success (drone ship)’, therefore we can query that exactly as well as confirm the mass to be within range.

Total Number of Successful and Failure Mission Outcomes

```
[20]: %sql SELECT Mission_Outcome, COUNT(Mission_Outcome) FROM SPACEXTBL Group by Mission_Outcome;
 * sqlite:///my_data1.db
Done.
```

[20]:

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

- Use Count to count missions of each type

Boosters Carried Maximum Payload

```
[42]: %sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ IN (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
 * sqlite:///my_data1.db
Done.
```

[42]:

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

- Nested Query to find which boosters were carrying max payload.

2015 Launch Records

```
[72]: %sql SELECT strftime('%m', Date) AS month, Date, Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTBL WHERE Landing_Outcome = 'Failure (drone sh
```

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

[72]:

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)

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.

```
nth, Date, Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTBL WHERE Landing_Outcome = 'Failure (drone ship)' AND strftime('%Y', Date) = '2015';
```

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

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)

- SQLite doesnt have access to MONTHNAME, so we just grab month number instead, but inside the query we just isolate the values we need, primarily month and year, and we query as if they were strings

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

```
[75]: %sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count FROM SPACEXTBL WHERE DATE BETWEEN "2010-06-04" AND "2017-03-20" GROUP BY Landing_Outcome ORDER BY
 * sqlite:///my_data1.db
Done.
```

[75]:

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

- This is full query as it was cut off

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


Section 3
Launch Sites
Proximities Analysis

Where are Stations?

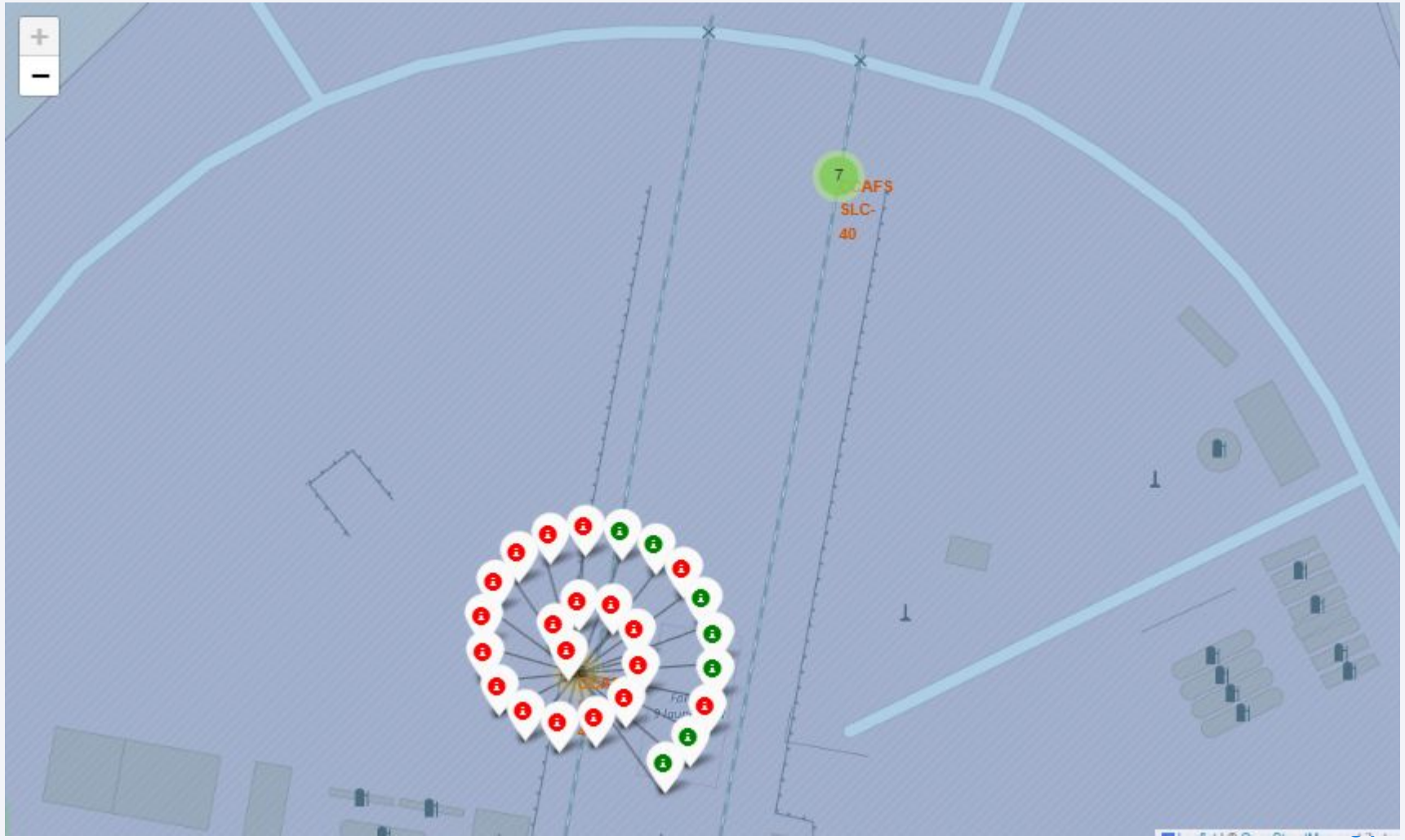
- Here we can see that stations are purely on coasts within the USA, with most being in Florida.
- Looking closer, I have color coated each station to its according type.

Zooming In

- Just to illustrate color coating of station types

Success/Failiures per Station

- We can see success in green, Failure in red, per station.
- This specific station (CCAFS SLC-40) tends to have mediocre/poor results, although the station above shows moderate to good results as it is green, indicating majority green within the bubble of 7 entries.

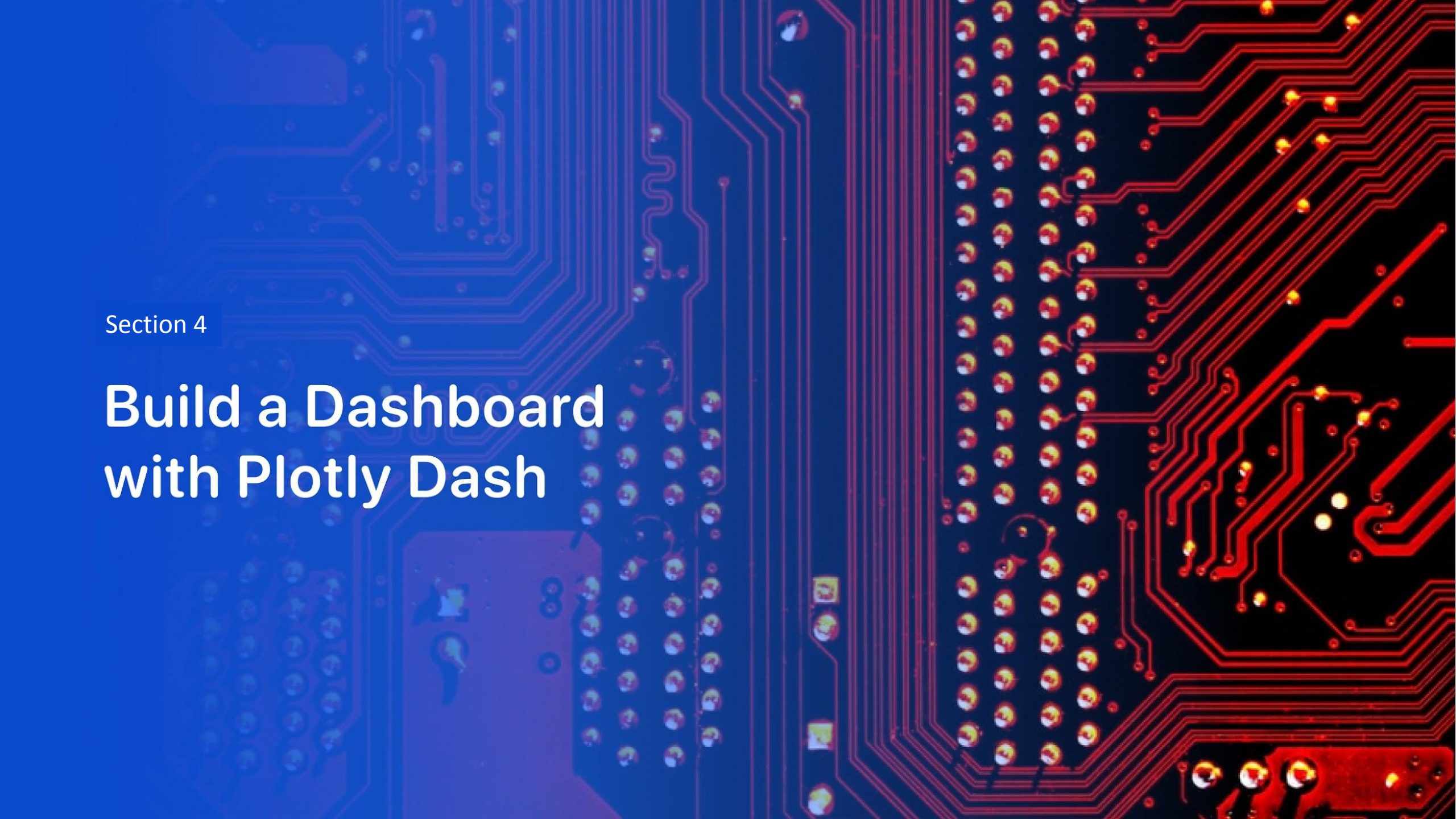


Location Importance of Stations

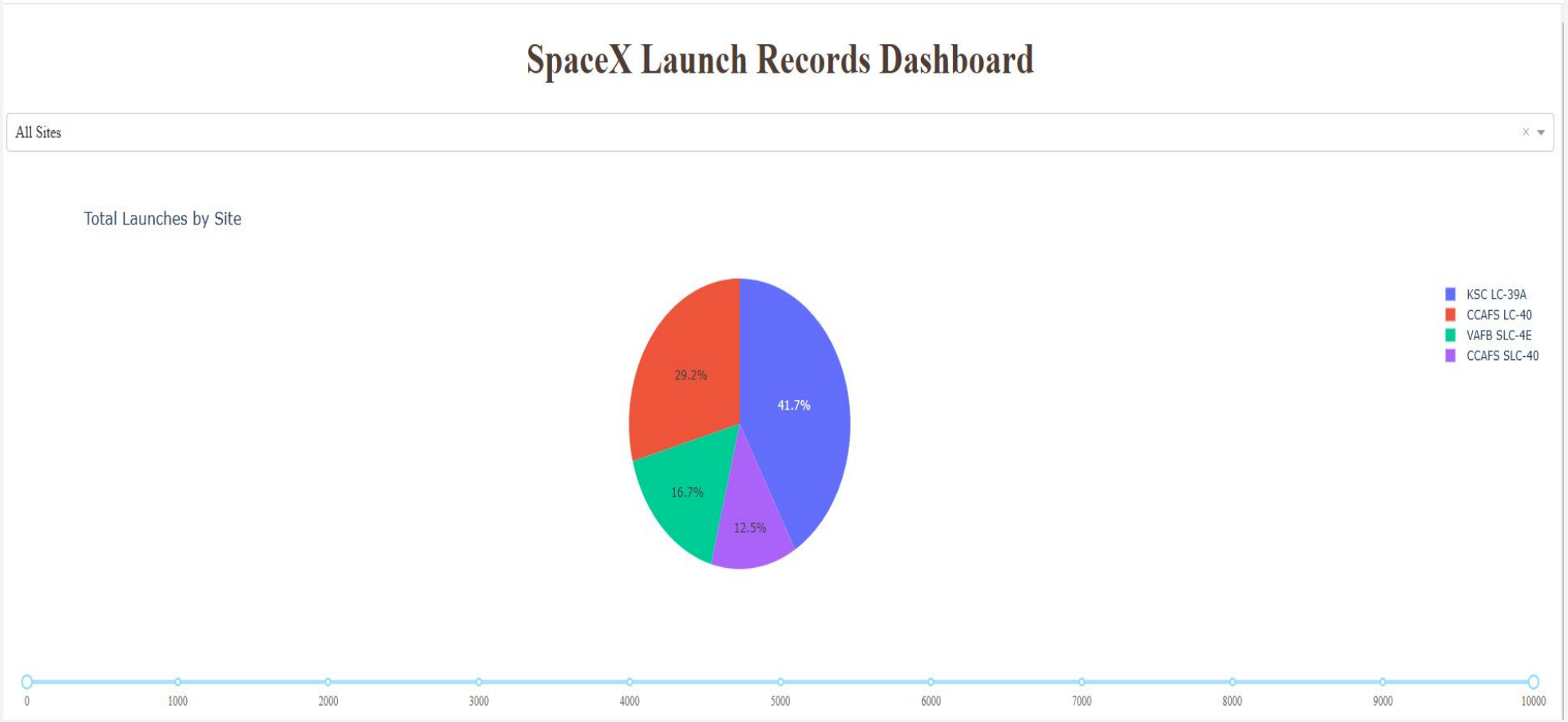
We can see that surrounding stations, there tends to be not a lot, 1-2 roads in, no railways, and most importantly, no cities.

This is important as derailed rockets can cover much ground fast, and it would be a horrible tragedy for a rocket to accidentally land on a city.

As we can see we are 1.25 KM from a railroad, 0.65 KM from a road, and 0.95 KM from the closest coast.


Section 4
Build a Dashboard with Plotly Dash

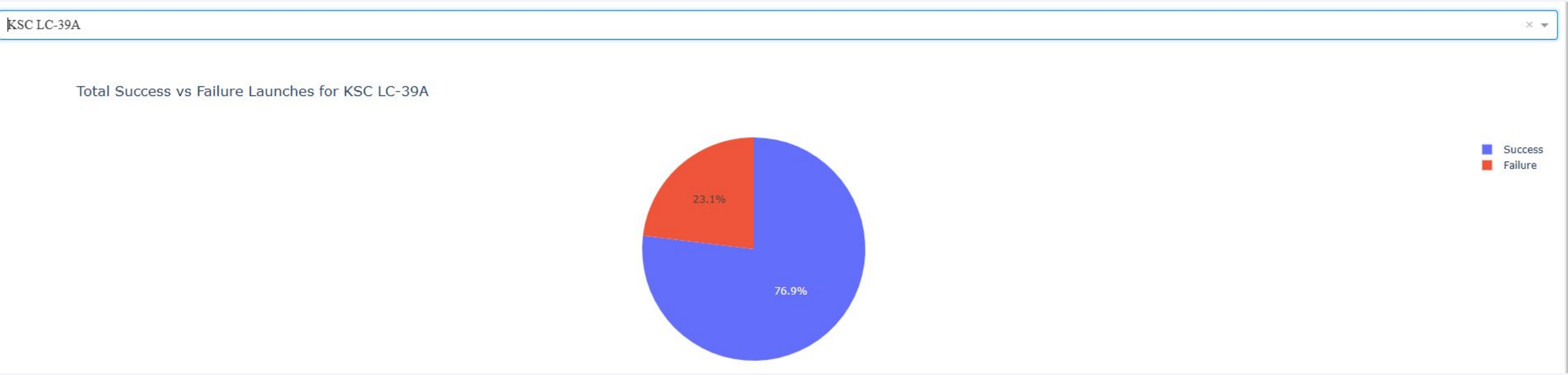
Success Count for All Sites



Success Count for All Sites

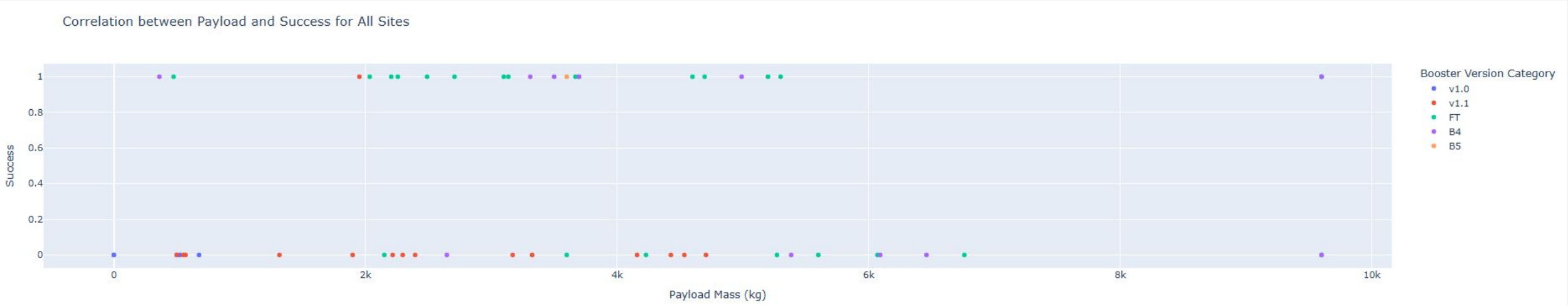
- On the previous slide we saw a pie chart, where majority of launches came from KSC LC -39A, followed by CCAFS LC-40,which are both florida based launch sites.

Highest Success Rate of a Station



- Looking closer at launch site KSC LC-39A, we see our highest success rate, at 76.9% . This is also one of the most popular sites for launches.

Payload Mass vs Success



- We see a stark rise in success explicitly between payload masses of 2k and ~5.7k. This seems to be the “golden child” payload mass range.

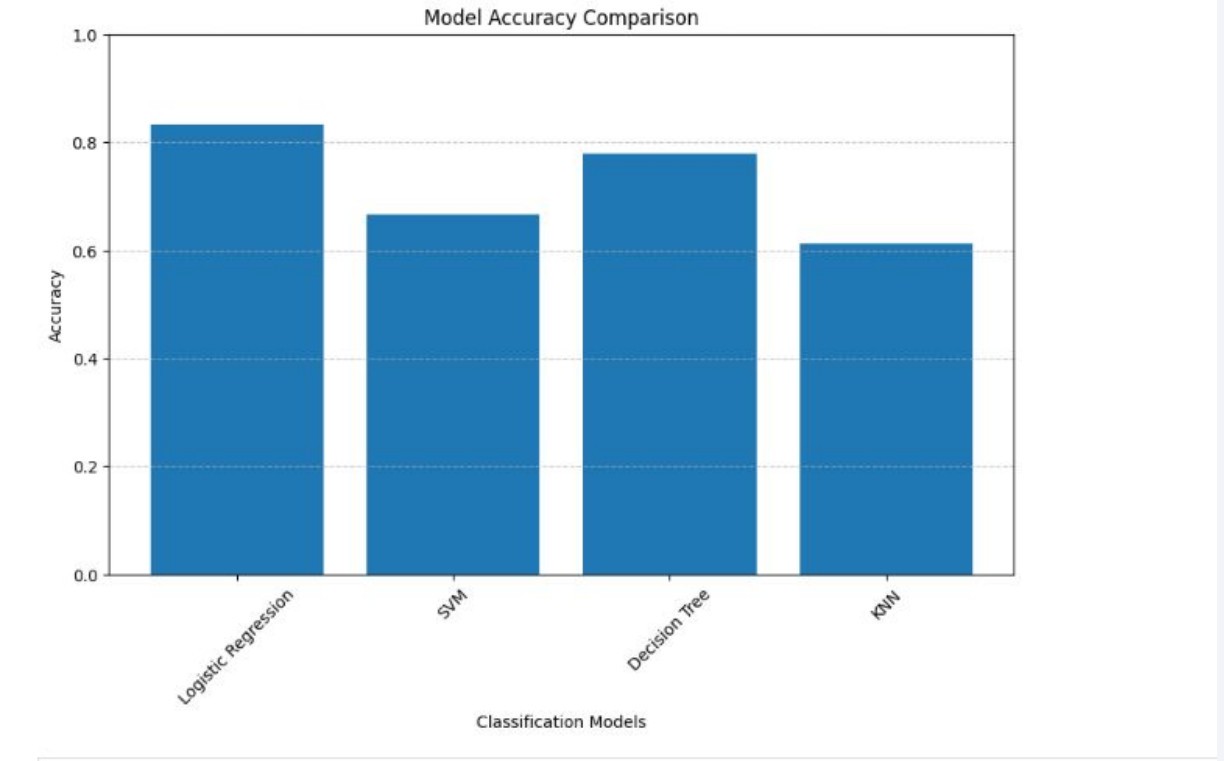
Section 5

Predictive Analysis (Classification)

Classification Accuracy

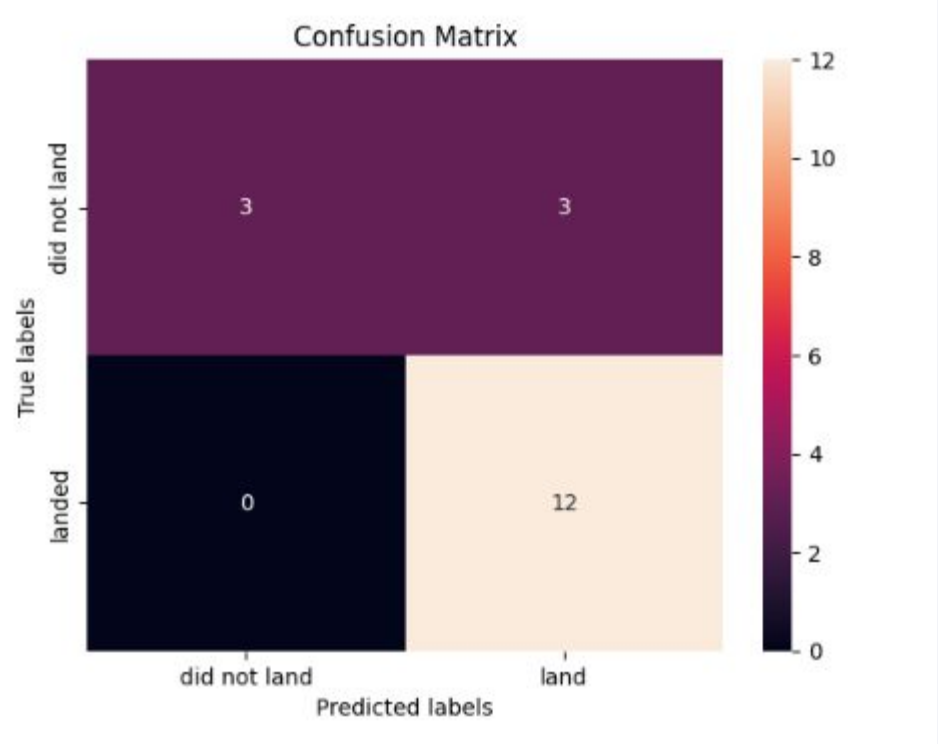
- We can see generally “Okay” results, with better results for Logistic Regression and Decision Tree. Given more data, I believe we could do better.

- Logistic Regression was the overall best.



Confusion Matrix

- Here we see our Confusion Matrix for Logistic Regression, in which we attained a ~83% success rate.
- For note, top left is “True Negative”, top right is “False Positive”, bottom left is “False Negative”, and bottom right is “True Positive”, so the higher values we get in top left and bottom right, the better performance we have.



Conclusions

- **Model Performance:**
  - **Logistic Regression** yielded the best results, achieving over 80% accuracy.
  - **Decision Tree** closely followed, also achieving 80%+ accuracy, but Logistic Regression linear assumptions aligned better with the dataset's characteristics.
- **Model Insights:**
  - Logistic Regression performs well when there is a **linear relationship** between features and target variables. Visualizations and statistical analysis confirmed such relationships in the dataset, explaining its superior performance.
- **Launch Site Analysis:**
  - Launches from **Florida-based regions** showed better success rates, though the **sample size** should be considered.
  - With only two clusters of launch sites in the U.S., their success rates are relatively consistent with each other.
- **Success Over Time:**
  - Regardless of the launch site, the **success rate of launches has steadily improved** over the years, with no indications of decline.
- **Orbit-Specific Success:**
  - Missions to **SO Orbit** showed a 0% success rate.
  - Orbits such as **ES-L1, GEO, HEO, and SSO** demonstrated a **100% mission success rate**.
  - Payload masses for missions to the **ISS** were observed to be consistent, reflecting standardized requirements for these missions.

Appendix

While I reused the same code file after it generated my plots to avoid clutter, I used this library to build my flow charts.

Github: Graphviz

Thank you!