Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

Summary of methodologies:

-Data Collection API
-Data Collection with Web Scraping
-Data Wrangling
-EDA with SQL
-EDA with Visualization
-Interactive Visual Analytics with Folium
-Interactive Dashboard with Ploty Dash
-Machine Learning Prediction

Summary of all results:

-Exploratory Data Analysis result
-Interactive analytics in screenshots
-Predictive Analytics result

Introduction

- **Project background and context**

The goal of this project is to create a machine learning model that predicts whether the first stage of a SpaceX Falcon 9 rocket will successfully land. This prediction can help estimate launch costs and assist other companies in competing with SpaceX for launch contracts.

- **Problems you want to find answers**

1. What factors influence the likelihood of a rocket landing successfully?
2. How do different features interact to impact the success rate of a rocket landing?
3. What operating conditions are necessary to ensure the success of a rocket landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - For building the model, the data was split into training and testing sets. GridSearchCV was used for hyperparameter tuning, and the evaluation was performed using a confusion matrix and accuracy score.

Data Collection

Data collection involved:

1. Using SpaceX API with GET requests to retrieve data.
2. Converting API responses into a Pandas DataFrame via JSON normalization.
3. Cleaning data, handling missing values, and filling gaps.
4. Web scraping Falcon 9 launch records from Wikipedia with BeautifulSoup, extracting tables, and converting them to a Pandas DataFrame for analysis.

Data Collection – SpaceX API

Data Collection steps:

1. GET request to SpaceX API for Falcon 9 data
2. Response parsing using JSON normalization
3. Data cleaning and handling of missing values

GitHub Link: https://github.com/rewpak/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

GET request to SpaceX API for Falcon 9 data

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

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

Response parsing using JSON normalization

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

Data cleaning and handling of missing values

```
# Calculate the mean value of PayloadMass column
mean_payload_mass = launch_df['PayloadMass'].mean()
# Replace the np.nan values with its mean value
launch_df['PayloadMass'].replace(np.nan, mean_payload_mass, inplace=True)
```

Data Collection - Scraping

Scraping steps:

1.Web scraping was applied to extract Falcon 9 launch records using BeautifulSoup.

3.Extract all column/variable names from the HTML table header

2.The table was parsed and converted into a pandas DataFrame.

GitHub Link:https://github.com/rewpak/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

1.Perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response. Create a BeautifulSoup object from the HTML response

```
# 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")
```

Extract all column/variable names from the HTML table header

```
# 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')
print(f"Number of table: {len(html_tables)}")
for idx, table in enumerate(html_tables):
    headers = table.find_all('th')
    column_names = [header.text.strip() for header in headers]
    print(f"Table {idx + 1}: {column_names}")
```

Parsed table and converted it into a pandas dataframe.

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df.head()
```

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	CCAFS	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success\n	F9 v1.07B0003.18	Failure	4 June 2010	18:45

Data Wrangling

Missing Values: Calculated percentages for each attribute.
Column Types: Identified numerical and categorical columns.
Launch Sites: Counted launches per site.
Orbits: Analyzed orbit frequencies.
Mission Outcomes: Counted outcomes by orbit.
Landing Labels: Created "Landing Outcome" to classify success rates.

GitHub Link: https://github.com/rewpak/IBM-Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

Scatterplots, bar charts and line chart were created to analyze Falcon 9 launches. These include:

FlightNumber vs LaunchSite: To observe launch site usage patterns.

FlightNumber vs Orbit type: To explore orbit selection trends over time.

Payload Mass vs Orbit type/Launch Site: To examine payload distribution by orbit and launch site.

Orbit success rates: To evaluate success frequency for each orbit type.

Yearly launch success trends: To track performance improvements over time.

GitHub Link: https://github.com/rewpak/IBM-Applied-Data-Science-Capstone/blob/main/edadataviz.ipynb

EDA with SQL

Performed SQL queries:

-Retrieved unique launch sites.
-Filtered launch sites starting with 'CCA'.
-Calculated total payload for NASA (CRS).
-Computed average payload for F9 v1.1.
-Found date of first ground pad success.
-Listed boosters with drone ship success and payload 4000–6000 kg.
-Counted mission successes and failures.
-Identified boosters with max payload using a subquery.
-Displayed 2015 failures on drone ships with details.
-Ranked landing outcomes by count (2010–2017).

GitHub Link: https://github.com/rewpak/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

Map Objects Added to the Folium Map:

Markers: Pinpointed launch sites with labels.

Circles: Color-coded for launch outcomes (e.g., green for success, red for failure).

Lines: Connected launch sites to nearby features (coastlines, highways, cities).

Clusters: Grouped markers to show success rates.

Popups: Displayed detailed launch info interactively.

Purpose: To visualize launch site locations, success rates, and their proximities to infrastructure, making patterns and geographic relationships easier to analyze.

GitHub Link: https://github.com/rewpak/IBM-Applied-Data-Science-Capstone/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

The follwoing interactions make the dashboard intuitive and insightful for analyzing launch data:

Dropdown Menu: Filters data by launch site for focused analysis.

Pie Chart: Visualizes launch success rates for all sites or specific ones

Range Slider: Filters data by payload mass range for detailed exploration.

Scatter Plot: Shows correlation between payload mass and launch success, highlighting trends.

GitHub Link: https://github.com/rewpak/IBM-Applied-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

Summary and Flowchart steps:

-**Data Preparation**: Converted target to NumPy, standardized features.

-**Split Data**: Training (80%) and test (20%) sets.

-**Model Building**: Built Logistic Regression, SVM, Decision Tree, KNN models.

-**Optimization**: Tuned hyperparameters using GridSearchCV (cv=10).

-**Evaluation**: Compared models based on test accuracy

-**Best Model**: Selected KNN for its performance and simplicity.

GitHub Link: https://github.com/rewpak/IBM-Applied-Data-Science-Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

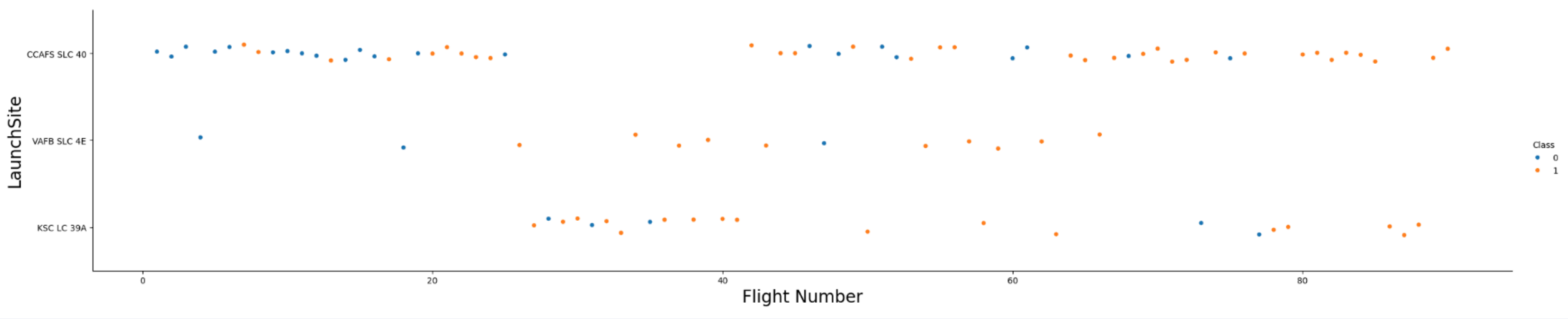
Results

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

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

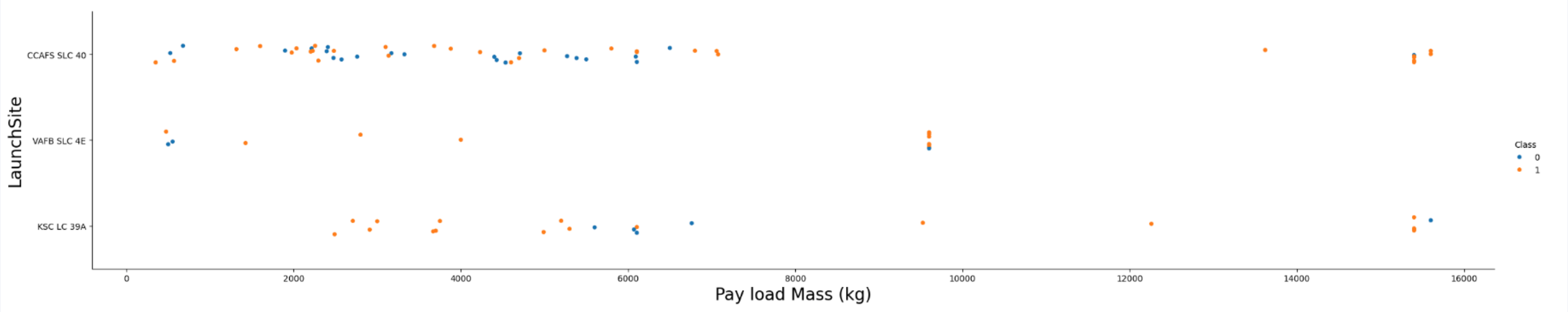


Based on the scatterplot we can conclude that:

CCAFS SLC 40 and KSC LC 39A have more launches and a higher success rate compared to **VAFB SLC 4E**.

Over time (with increasing flight number), the success rate improves, indicating better reliability of launches.

Payload vs. Launch Site



The scatterplot shows that:

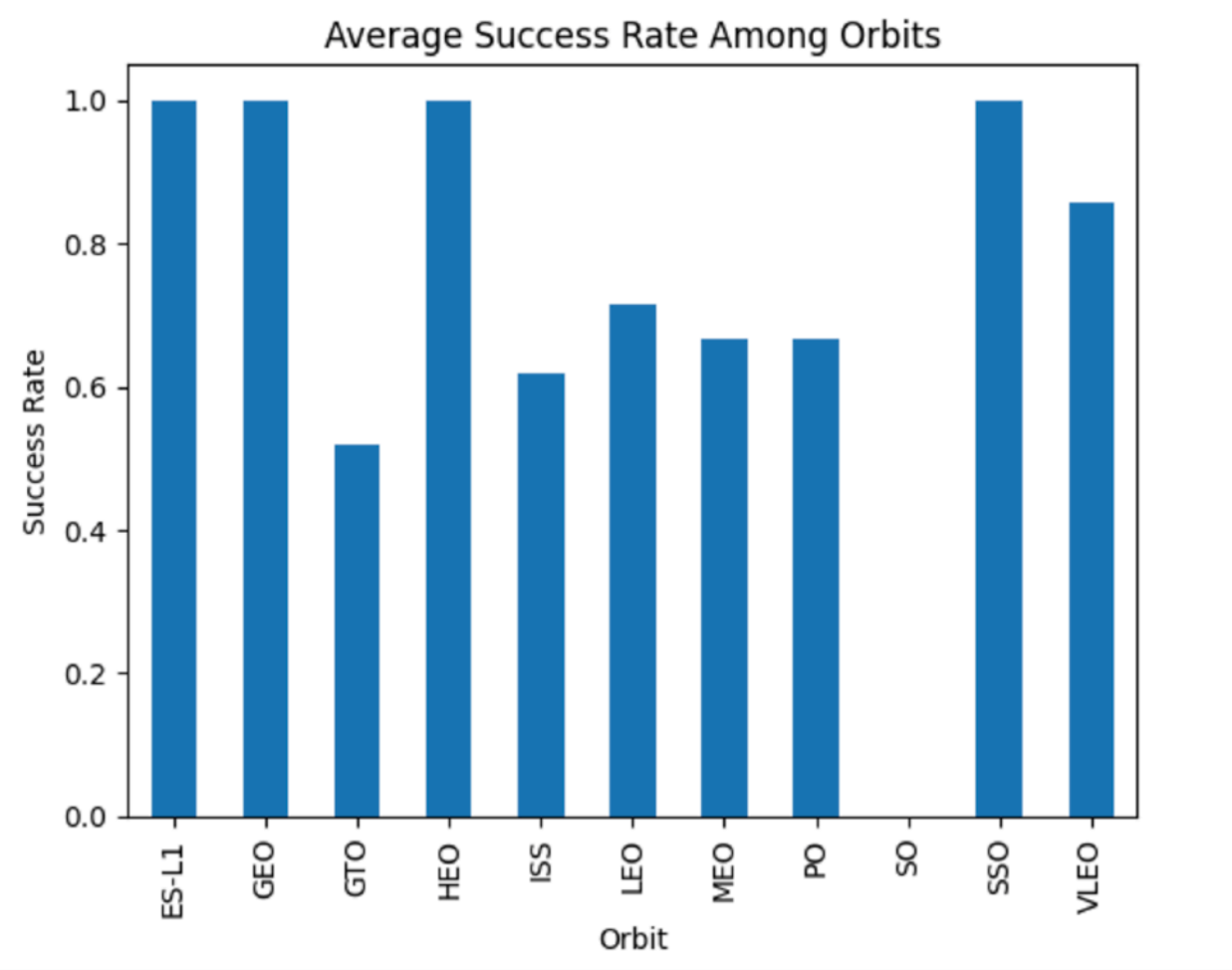
CCAFS SLC 40 and KSC LC 39A handle a wide range of payload masses, with both successes and failures across the spectrum.

VAFB SLC 4E has fewer launches and typically handles lower payload masses.

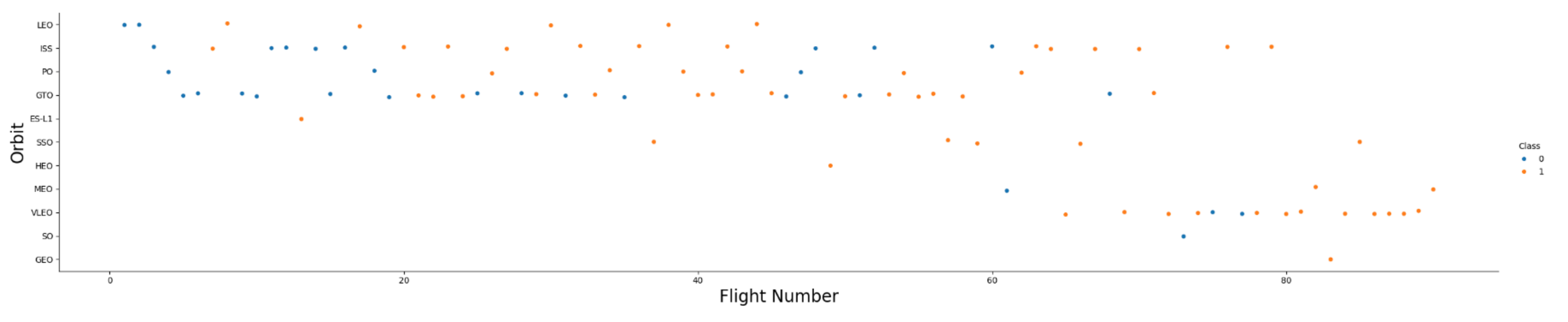
Higher payload masses (above 10,000 kg) generally show a higher success rate.

Success Rate vs. Orbit Type

From the plot, we can see that **ES-L1**, **GEO, HEO, SSO, VLEO** have the hightest success rate while the lowest rate is mentioned **GTO**



Flight Number vs. Orbit Type

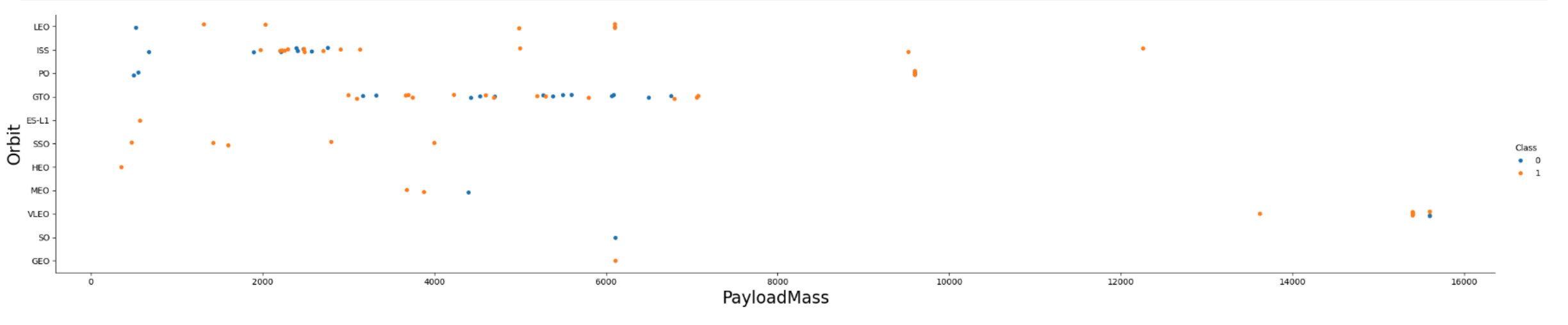


The scatterplot shows:

LEO, ISS, PO and GTO orbits dominate early launches, with a mix of successes (orange) and failures (blue).

Over time (higher flight numbers), the success rate improves for most orbits, particularly for **GTO**, which initially had more failures.

Payload vs. Orbit Type



The scatterplot shows:

LEO, ISS, and PO handle lower payload masses (below 4000 kg) with a mix of successes and failures.

SSO and GEO maintain high success rates regardless of payload mass.

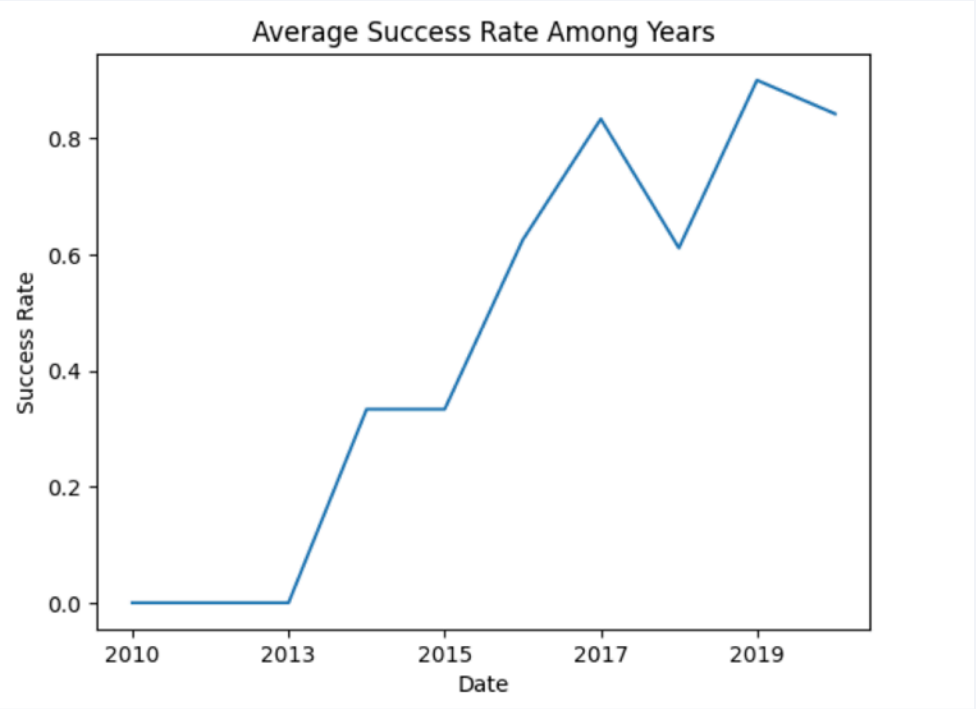
Overall, the success rate tends to increase with payload mass, especially for challenging orbits like GTO.

Launch Success Yearly Trend

The line chart demonstrates:

A gradual and significant increase in the average success rate starting around 2013.

Peaks in 2018–2019, showing a consistently high success rate above 80%.



All Launch Site Names

The **DISTINCT** was used to display only the unique launch sites

```
%sql SELECT DISTINCT "Launch_Site" FROM "SPACEXTABLE"
```

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

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

Launch Site Names Begin with 'CCA'

```
%sql SELECT "Launch_Site" FROM "SPACEXTABLE" WHERE "Launch_Site" LIKE 'CCA%' LIMIT 5
```

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

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

The "**%**" was used after CCA to display Launch Sites strarted on CCA

Total Payload Mass

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") FROM "SPACEXTABLE" WHERE "Customer" = 'NASA (CRS)'
```

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

SUM("PAYLOAD_MASS__KG_")
45596

The **SUM()** was used on PAYLOAD_MASS to display the total mass with Condition Customer = Nasa CRS

Average Payload Mass by F9 v1.1

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") FROM "SPACEXTABLE" WHERE "Booster_Version" = 'F9 v1.1'
```

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

AVG("PAYLOAD_MASS__KG_")
2928.4

The **AVG()** was used to display only the avarage mass

First Successful Ground Landing Date

```
%sql SELECT MIN(DATE) FROM "SPACEXTABLE" WHERE "Landing_Outcome" = 'Success (ground pad)'
```

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

MIN(DATE)
2015-12-22

The **MIN()** was used on DATE to display the First Successful Ground Landing Date

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING__OUTCOME='Success (drone ship)' and PAYLOAD_MASS__
```

```
 * ibm_db_sa://ktf76410:***@ba99a9e6-d59e-4883-8fc0-d6a8c9f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

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

We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome") FROM "SPACEXTABLE" GROUP BY "Mission_Outcome"
```

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

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

The "**COUNT()**" and "**GROUPED BY**" was used on MISSION_OUTCOME to display total number of successful and failire mission

Boosters Carried Maximum Payload

The **SUBQUIERY ,WHERE and MAX()** were used on Payload to display what Booster_version has max payload

```
%sql SELECT "Booster_Version" FROM "SPACEXTABLE" WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FRC
```

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

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

2015 Launch Records

```
%sql SELECT substr(Date, 6, 2) AS Month, "Landing_Outcome", "Booster_Version", "Launch_Site" FROM "SPACEXTABLE" W
```

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

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

WHERE clause, **LIKE**, **AND**, and **BETWEEN** conditions were used to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

```
%sql SELECT "Landing_Outcome", COUNT("Landing_Outcome") AS Outcome_Count FROM "SPACEXTABLE" WHERE "Date" BETWEEN
 * sqlite:///my_data1.db
Done.
```

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

We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.

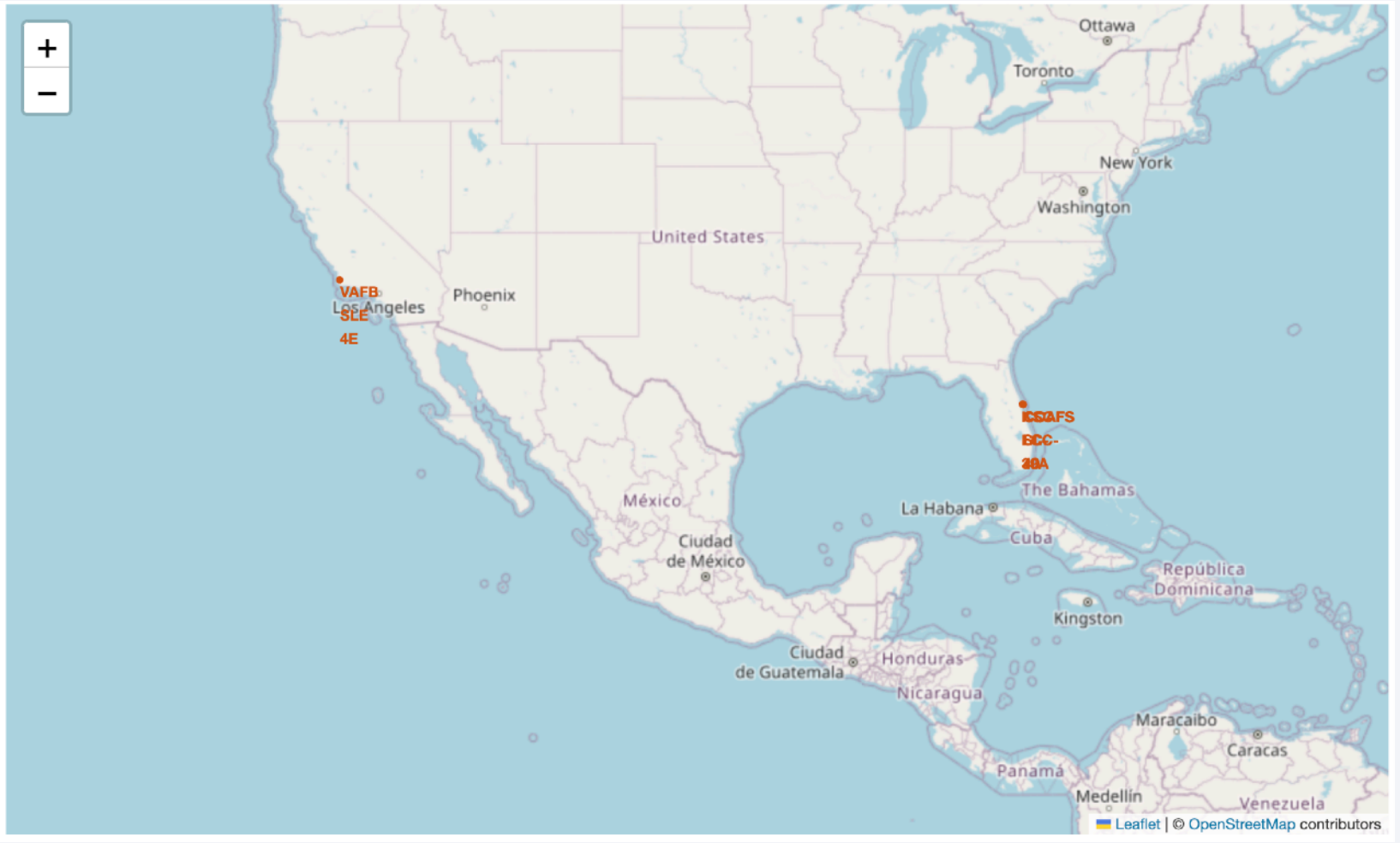
We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order

Section 3

Launch Sites Proximities Analysis

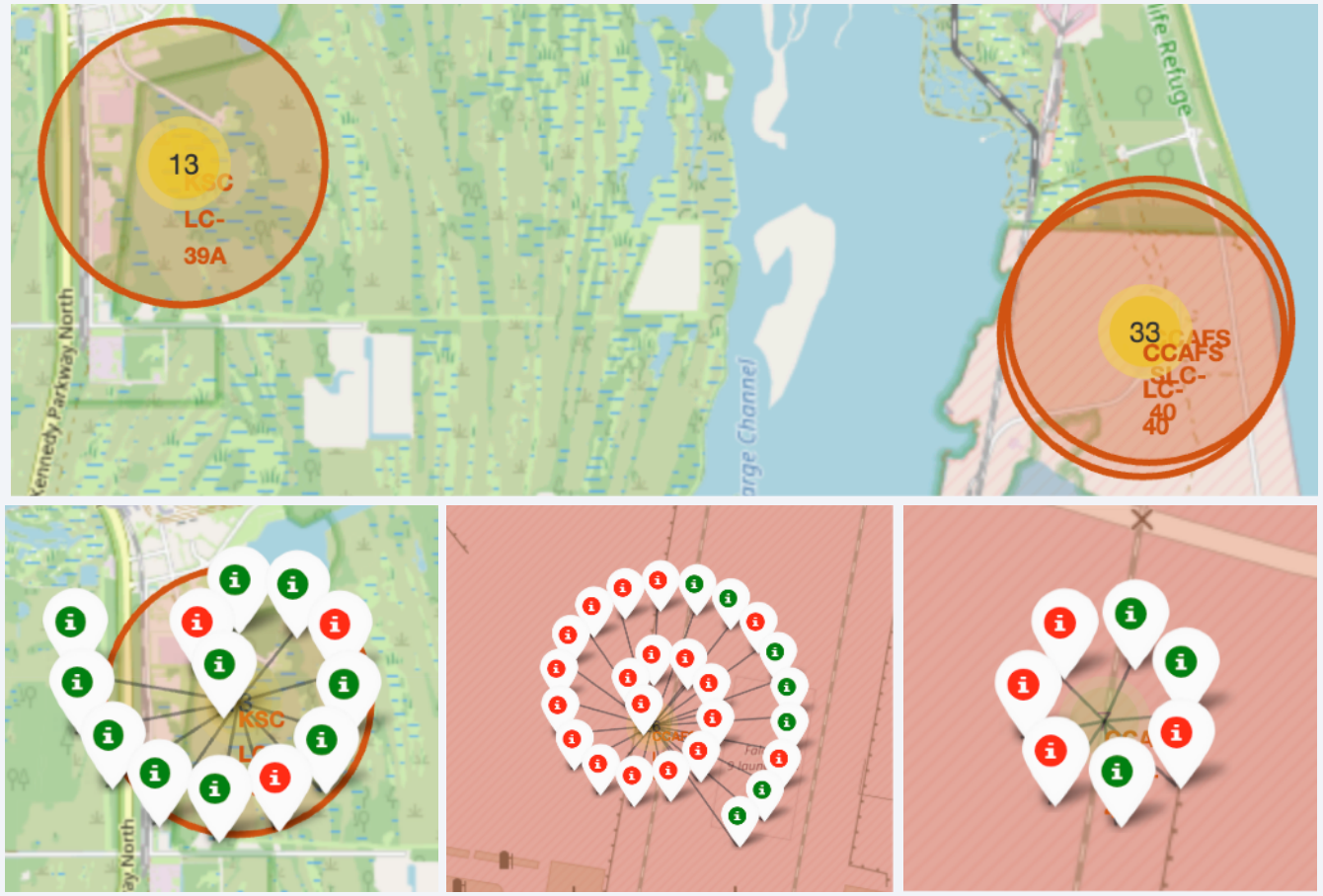
All launch sites on global map

The SpaceX launch sites are located along the coasts of the United States, specifically in Florida and California.

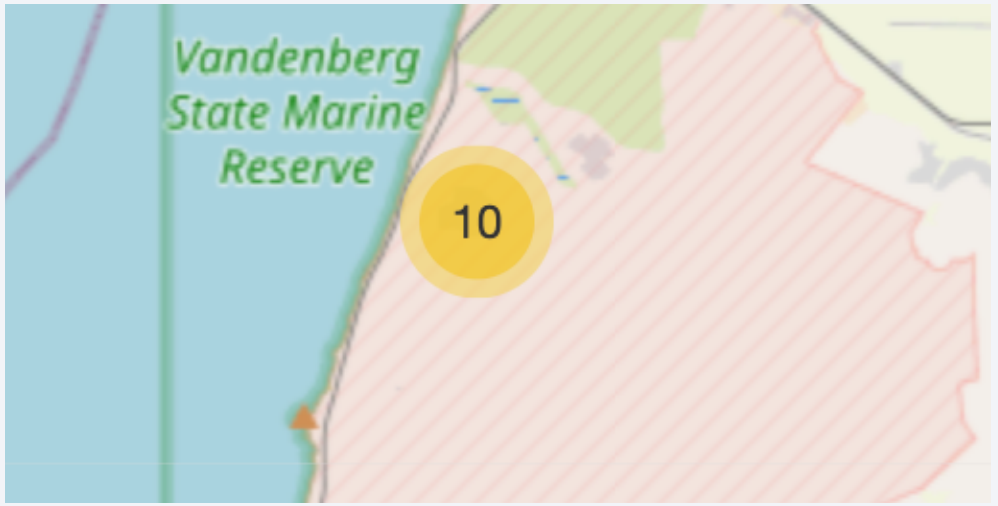


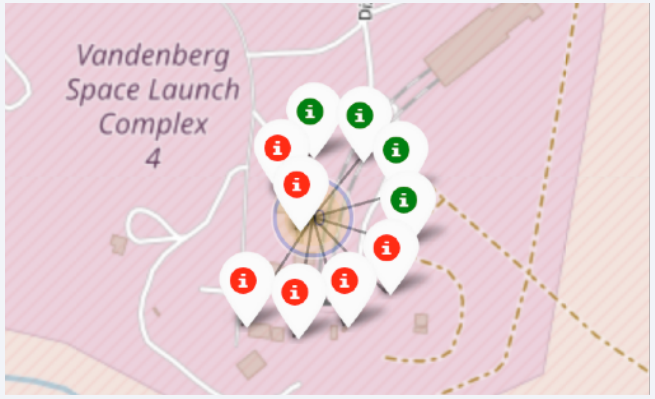
Launch Sites wiht Color labels

Florida Launches



California Launches





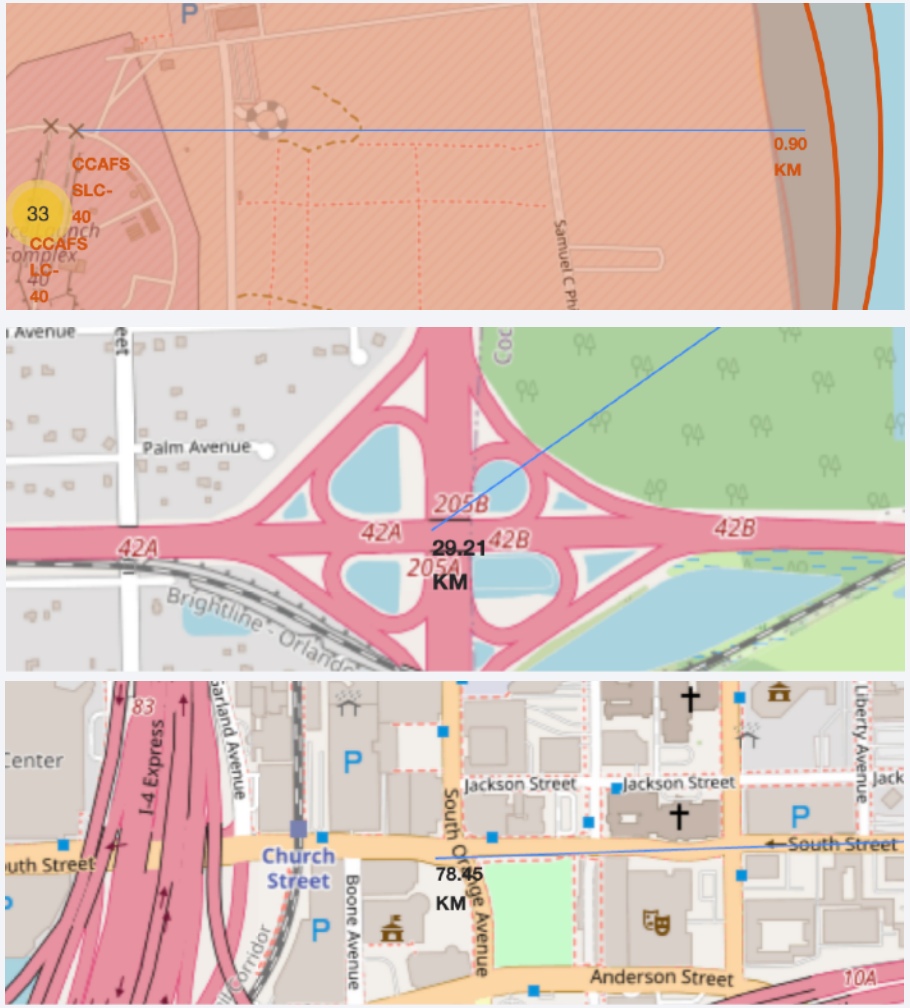
Green markers represent successful launches, while red markers indicate failures.

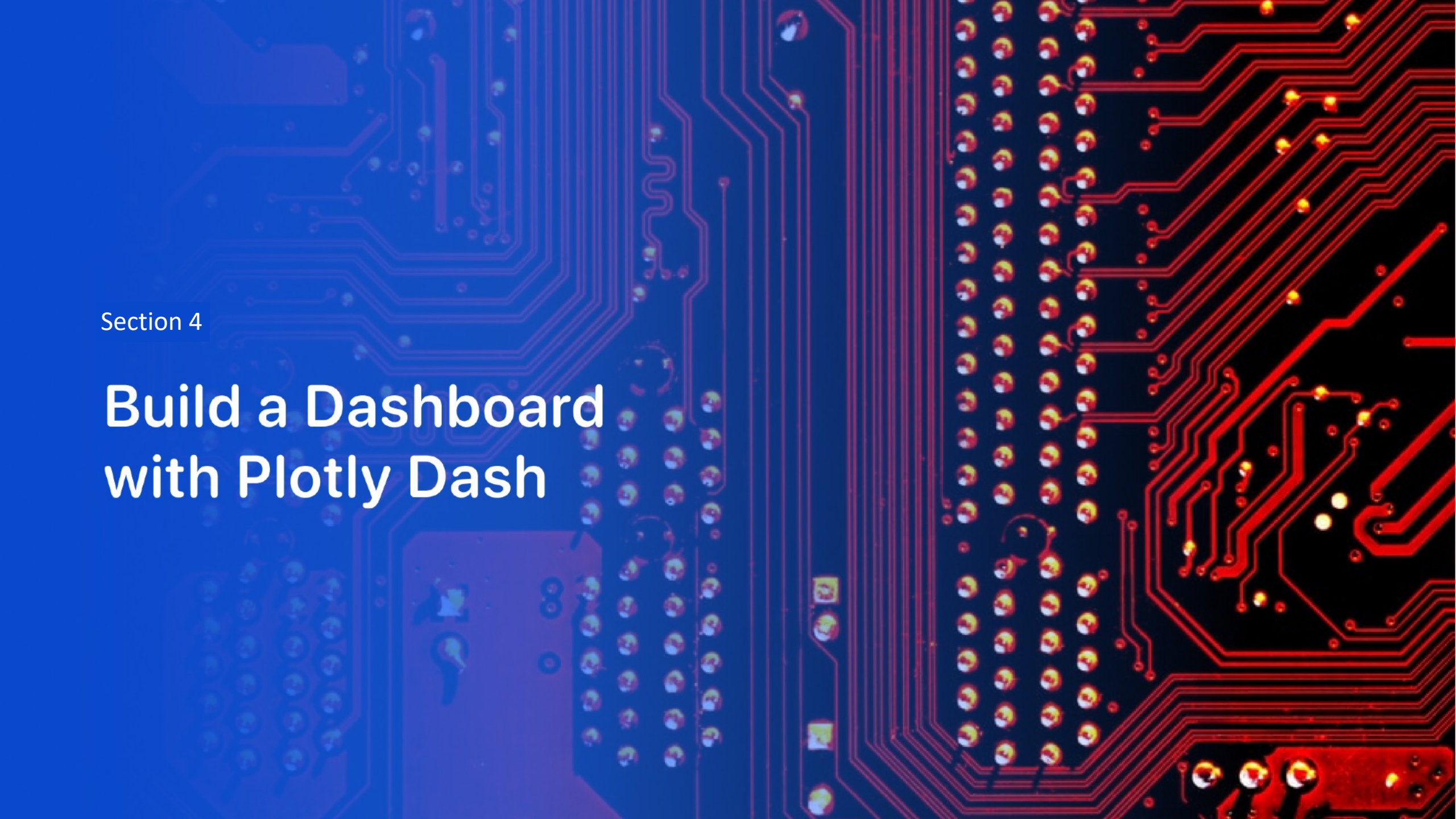
Launch Site distance to landmarks

Distance from Launch Site to Coastline is 0.90 km

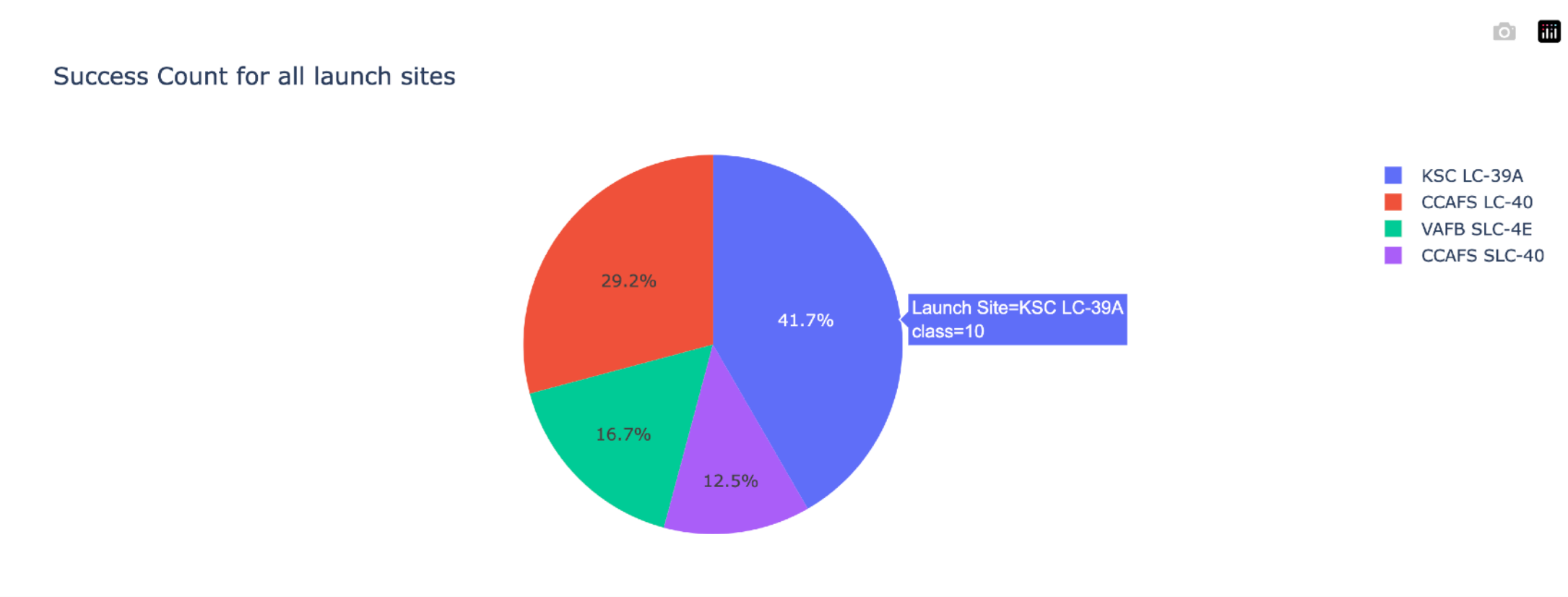
Distance from Launch Site to City is 78,45 km

Distance from Launch Site to Highwy is 29.21 km



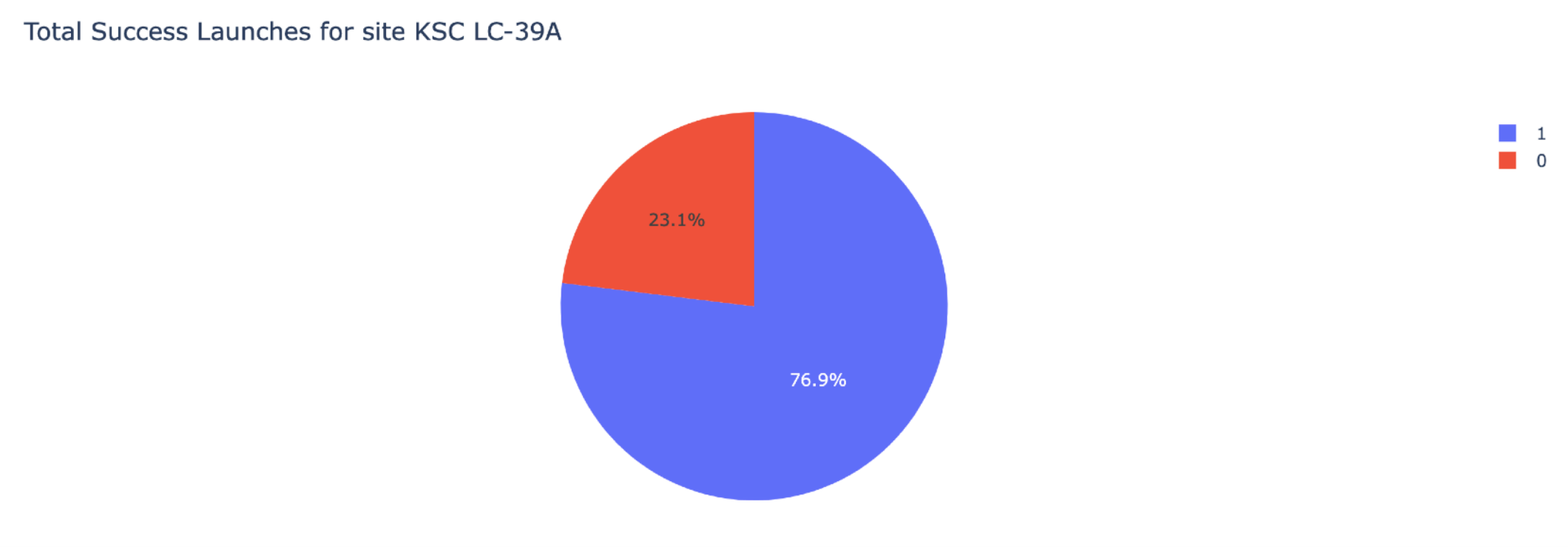

Section 4
Build a Dashboard with Plotly Dash

Successfull launches for all sites



- It is clear that KSC LC-39A has the most succssefull launches

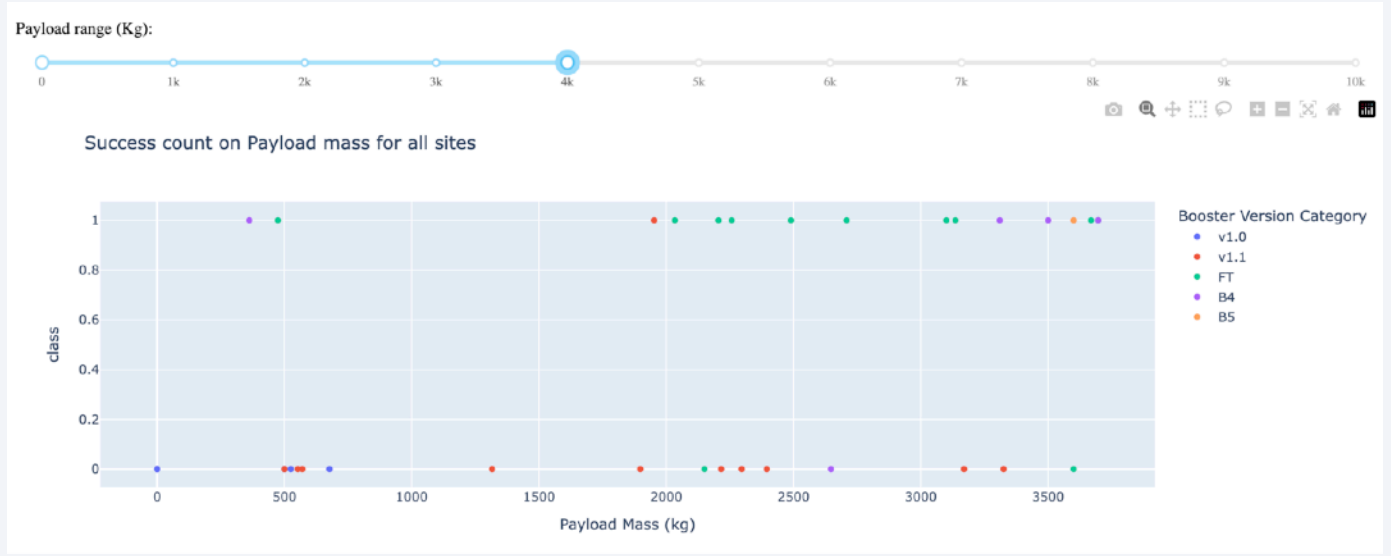
Proportion of Launches for most successful site



- The prortion of Successfull and Faliure launces is 76.9% and 23.1% respectively

Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

- It can be seen that the success rate for low weighted payloads is higher than the heavy weighted payloads



Section 5

Predictive Analysis (Classification)

Classification Accuracy

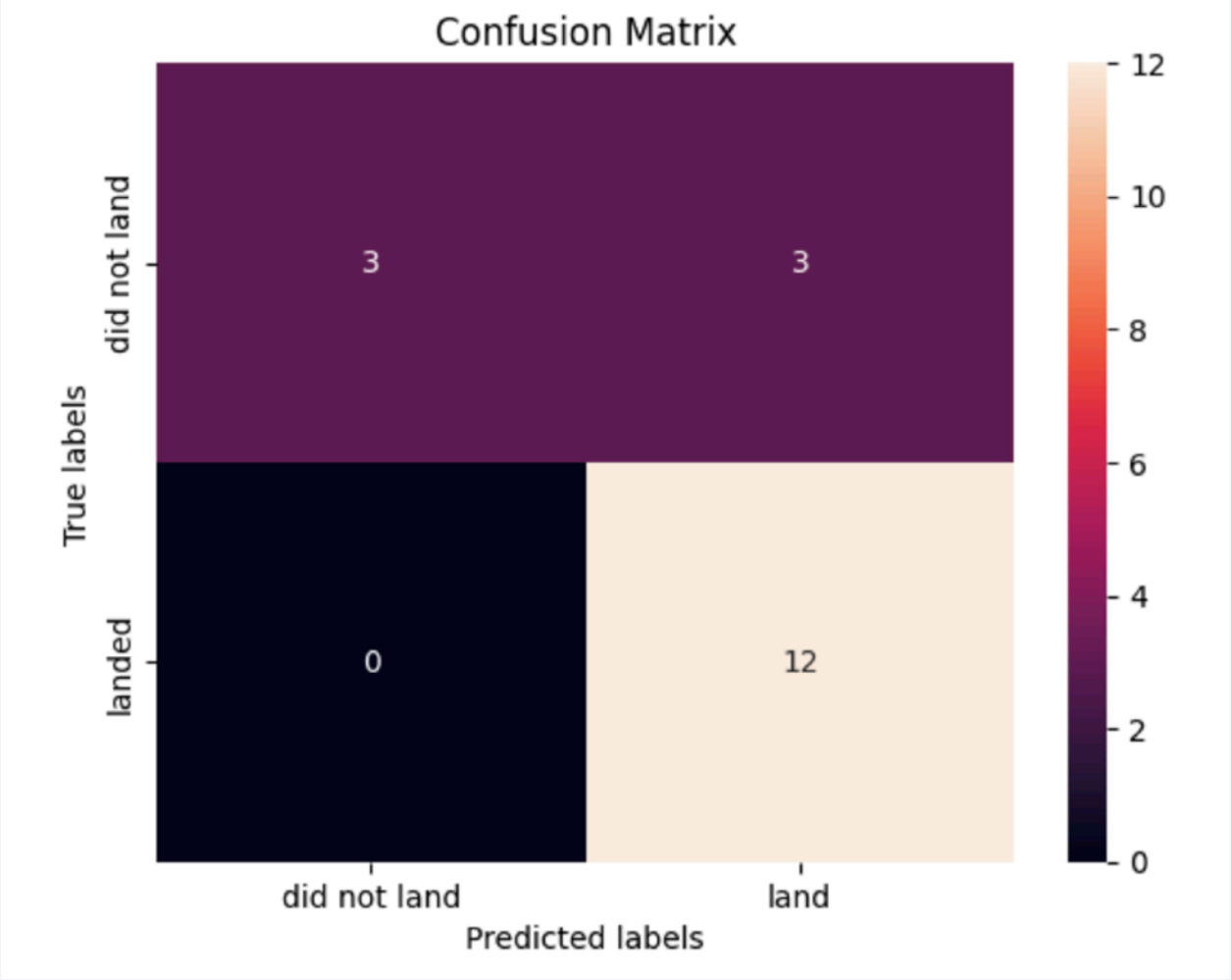
```
predictors = [knn_cv, svm_cv, logreg_cv, tree_cv]
best_predictor = None
best_result = 0


for predictor in predictors:
    result = predictor.score(X_test, Y_test)
    print(f"Model: {type(predictor.best_estimator_).__name__}, Accuracy: {result}")
    if result > best_result:
        best_result = result
        best_predictor = predictor


print(f"Best Predictor: {type(best_predictor.best_estimator_).__name__}, Accuracy: {best_result}")
```

```
Model: KNeighborsClassifier, Accuracy: 0.8333333333333334
Model: SVC, Accuracy: 0.8333333333333334
Model: LogisticRegression, Accuracy: 0.8333333333333334
Model: DecisionTreeClassifier, Accuracy: 0.8333333333333334
Best Predictor: KNeighborsClassifier, Accuracy: 0.8333333333333334
```

Although all models showed similar results, the best model was KNN, likely due to differences in decimal precision.

Confusion Matrix

- The confusion matrix for the decision tree classifier indicates its ability to differentiate between classes. However, a key issue is the presence of false positives, where unsuccessful landings are incorrectly classified as successful.



Conclusions

-Higher flight frequency at a launch site correlates with a higher success rate.
-Launch success rates steadily improved between 2013 and 2020.
-Orbits such as ES-L1, GEO, HEO, SSO, and VLEO demonstrated the highest success rates.
-KSC LC-39A recorded the most successful launches among all sites.
-The KNN Classifier proved to be the most effective machine learning algorithm for this analysis.

Thank you!