Winning Space Race with Data Science

<Jiping Wang>
<25/02/25>

Outline

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

Executive Summary

Summary of methodologies:

- Methodologies Overview
- Data Collection via API
- Data Collection through Web Scraping
- Data Wrangling
- Exploratory Data Analysis
- SQL-Based EDA
- Data Visualization for EDA
- Interactive Visual Analytics with Folium
- Machine Learning Prediction
- Summary of Predictive Analytics Results

Introduction

SpaceX's Disruption in the Space Industry

SpaceX has revolutionized the space industry by offering Falcon 9 rocket launches at a significantly lower cost—starting at $62 million—compared to competitors, who charge upwards of $165 million per launch. This cost reduction is largely driven by SpaceX's innovative approach of reusing the first stage of the rocket, allowing it to be relaunched for future missions. As this process is repeated, costs are expected to decrease even further.

As a data scientist working for a startup competing with SpaceX, the goal of this project is to develop a machine learning pipeline that predicts the landing success of the rocket's first stage. This prediction is crucial for determining competitive bid pricing against SpaceX for rocket launches.

Key Problems to Address:

- Identifying factors that influence landing success.
- Analyzing relationships between variables and their impact on landing outcomes.
- Determining optimal conditions to maximize the probability of a successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data Collection Methodology:
  - Data gathered via SpaceX REST API and web scraping from Wikipedia.
  - Data wrangling and cleaning procedures applied.
- Data Processing: Categorical features processed using one-hot encoding.
- Exploratory Data Analysis (EDA): Visual analysis and SQL-based exploration conducted.
- Interactive Visual Analytics: Interactive dashboards created using Folium and Plotly Dash.
- Predictive Analysis: Classification models built, tuned, and evaluated for predictive insights.

Data Collection

- **Data Collection Process**
- **Purpose:** Gathering and measuring information on targeted variables to answer key questions and evaluate outcomes.
- **Methods Used:**
  - **REST API:**
    - Initiated with a GET request.
    - Decoded the response as JSON and converted it into a Pandas dataframe using json_normalize().
    - Cleaned the data, checked for missing values, and filled them where necessary.
  - **Web Scraping (BeautifulSoup):**
    - Extracted launch records as an HTML table.
    - Parsed the table and converted it into a Pandas dataframe for further analysis.

Data Collection – SpaceX API

Steps Summary :

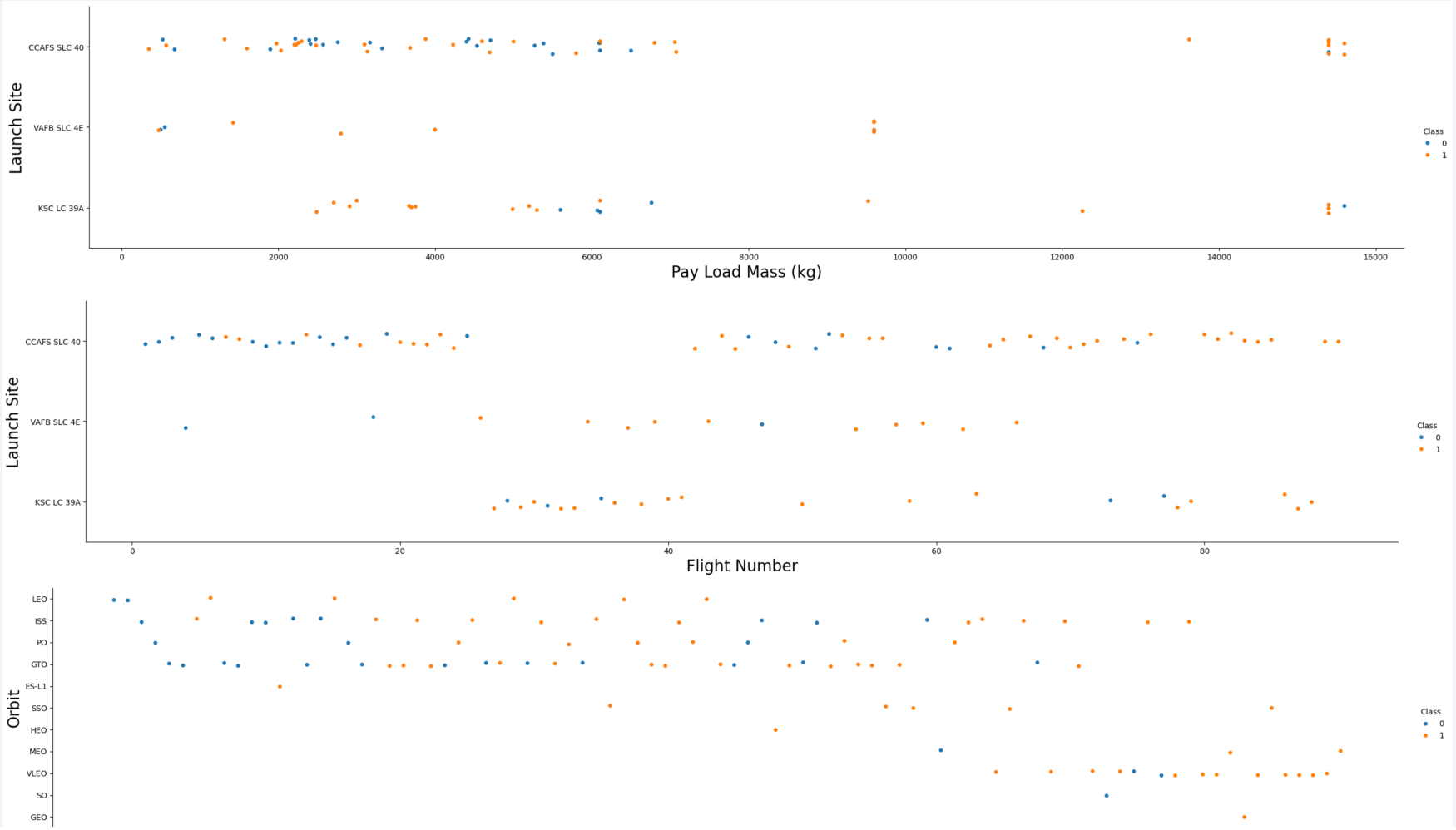
- API Request: Retrieve past launch data from the SpaceX API using a GET request.
- Normalize JSON: Convert the JSON response into a Pandas DataFrame using pd.json_normalize().
- Data Subsetting: Select relevant columns (rocket, payloads, launchpad, cores, flight_number, date_utc).
- Data Cleaning: Filter out rows with multiple cores or payloads, and extract single values from lists in cores and payloads.
- Date Conversion & Filtering: Convert date_utc to a date format and filter launches to include only those before or on November 13, 2020.

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

response = requests.get(spacex_url)

[10] static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da

[12] # Use json_normalize meethod to convert the json result into a dataframe
     response = requests.get(static_json_url)
     data = pd.json_normalize(response.json())

[13] # 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
     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
     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)]
```

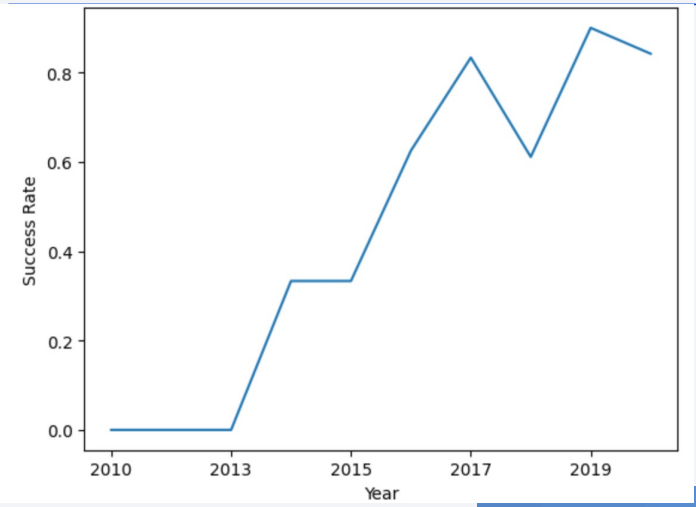
Data Collection & Data Wrangling

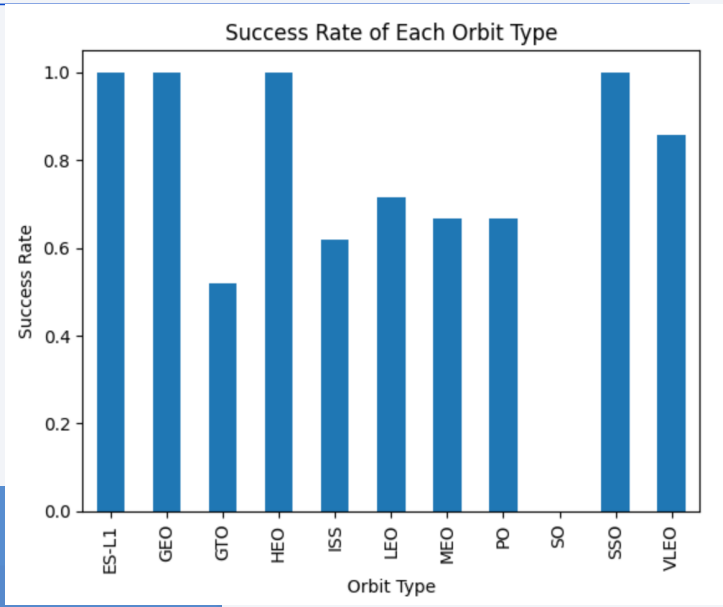
- Data Wrangling & Processing
- Data Loading: Imported into Pandas DataFrames.
- Target Variable Creation: Extracted 'Class' column into Y (NumPy array).
- Feature Standardization: Applied StandardScaler to normalize data.
- Data Splitting: Divided into training and test sets for machine learning.

EDA with Data Visualization



EDA with Data Visualization





EDA with Data Visualization

Initial Exploration with Scatter Plots:

We began by using scatter plots to explore relationships between key attributes:

- Payload vs. Flight Number
- Flight Number vs. Launch Site
- Payload vs. Launch Site
- Flight Number vs. Orbit Type
- Payload vs. Orbit Type

Scatter plots helped identify patterns and dependencies between variables, providing insights into factors affecting landing success.

Further Analysis with Bar and Line Graphs:

- **Bar Graphs:** Used to analyze categorical relationships, such as identifying which orbits have the highest success probability.
- **Line Graphs:** Utilized to observe trends over time, specifically tracking the yearly success rate of launches.

Feature Engineering:

We applied feature engineering to prepare for success prediction, creating dummy variables for categorical columns to use in future modeling.

EDA with SQL

- Task 1: Displayed unique Launch Site names.
- Task 2: Shown 5 records where launch sites start with 'CCA'.
- Task 3: Calculated total Payload Mass for NASA (CRS) launches.
- Task 4: Computed average Payload Mass for Booster Version F9 v1.1.
- Task 5: Identified the first successful ground pad landing date.
- Task 6: Listed Boosters with successful drone ship landings and payload mass between 4000-6000 kg.
- Task 7: Counted total successful and failed missions.
- Task 8: Displayed Booster Versions carrying the maximum payload mass using subqueries.
- Task 9: Listed records for failure drone ship landings in 2015 with month, booster version, and launch site.
- Task 10: Ranked Landing Outcomes between 2010-06-04 and 2017-03-20 in descending order

- More work see: Github

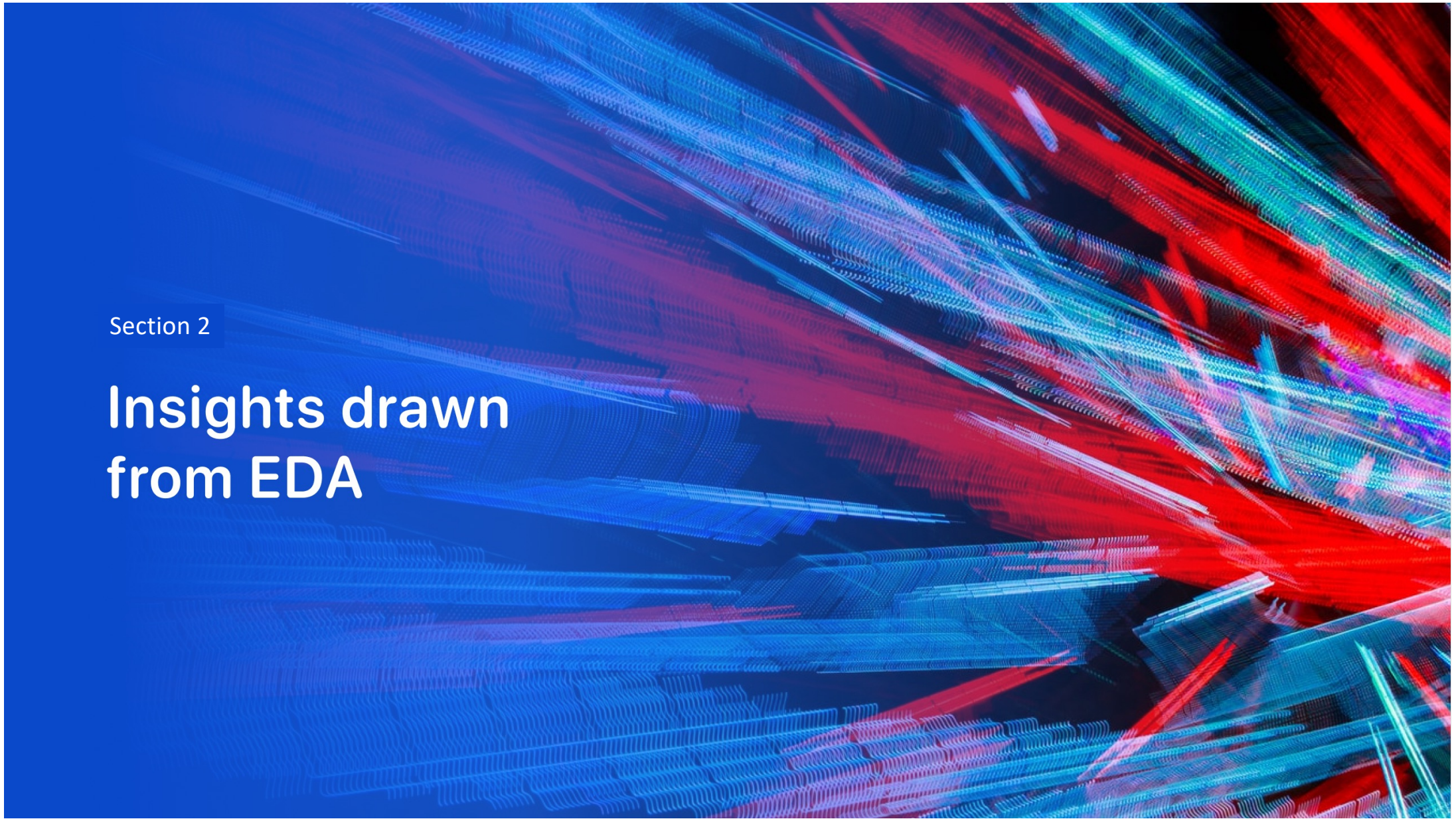
Predictive Analysis (Classification)

- Building the Model

1. Load and Transform Data:
    - Import the dataset into NumPy and Pandas.
    - Preprocess and transform the data, then split into training and test datasets.
2. Select Machine Learning Approach:
    - Decide on the type of machine learning model to use (e.g., classification, regression).
3. Model Training:
    - Set hyperparameters and algorithms for GridSearchCV.
    - Fit the model to the dataset.

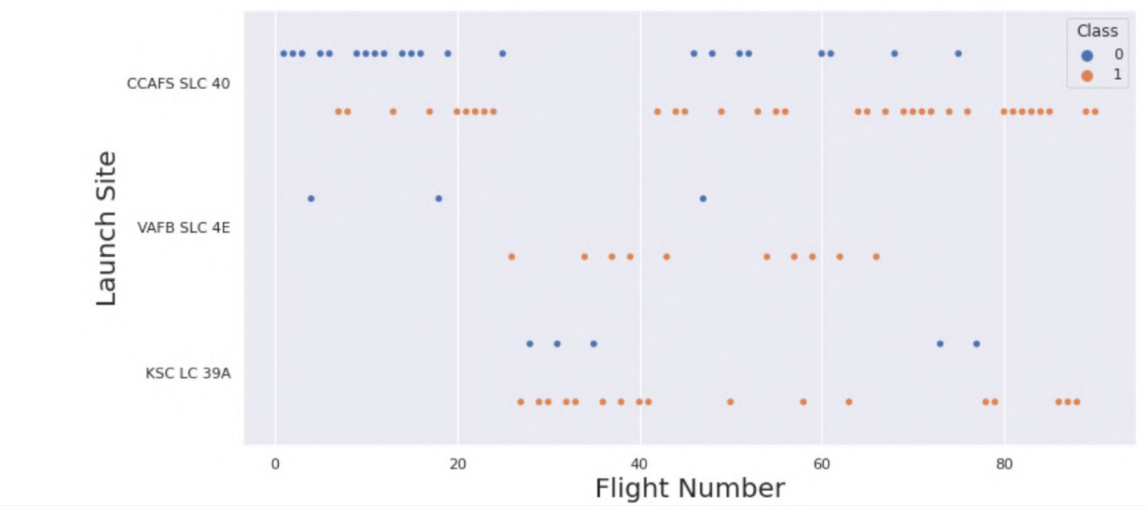
Results

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


Section 2
Insights drawn from EDA

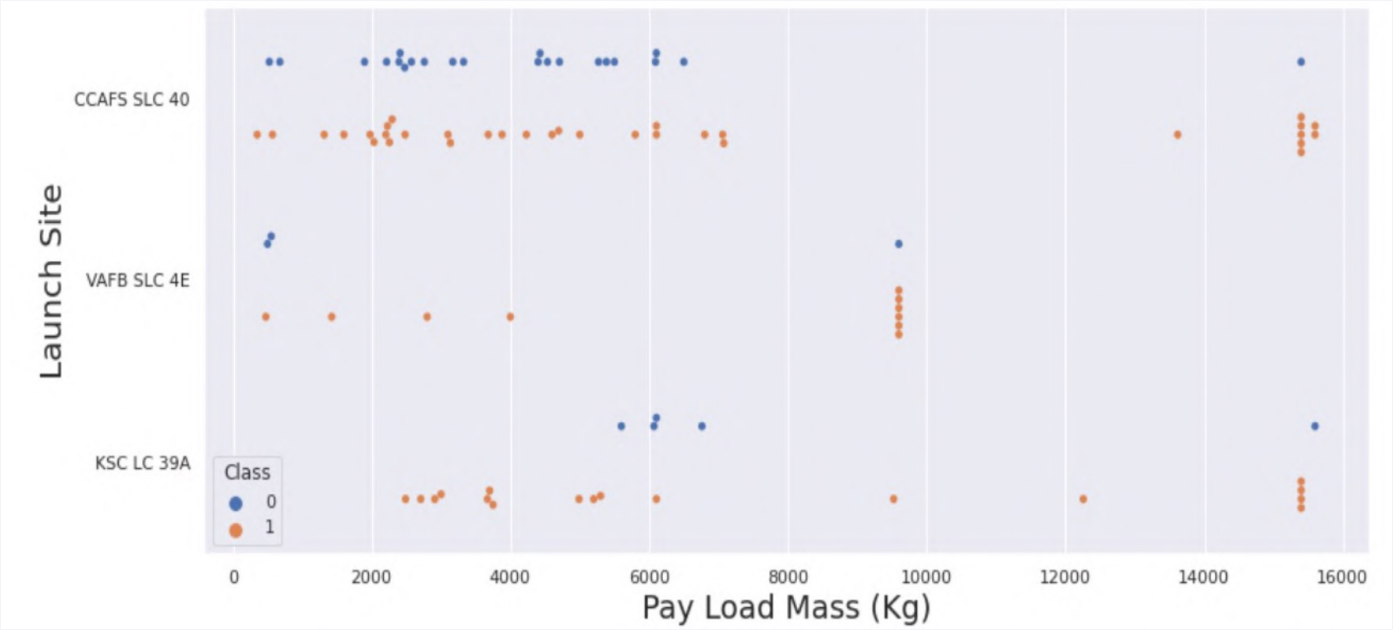
Flight Number vs. Launch Site

Scatter Plot Analysis:
The plot shows a positive correlation between the number of flights at a launch site and its success rate. However, CCAFS SLC40 deviates from this trend, showing the lowest correlation.



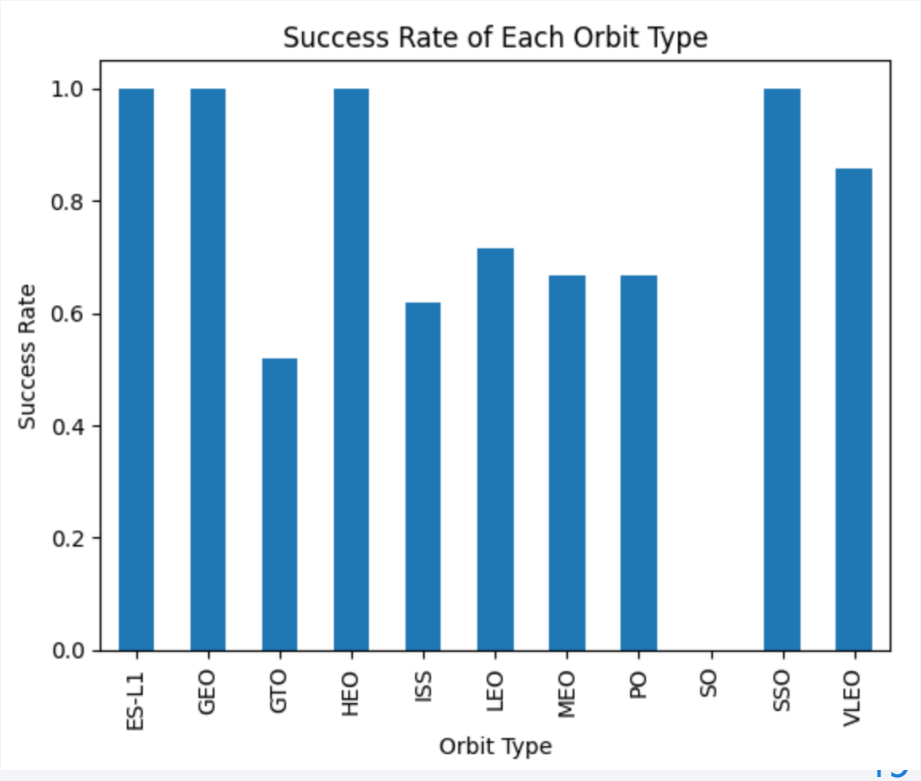
Payload vs. Launch Site

- As payload mass exceeds 7,000 kg, the success rate significantly increases.

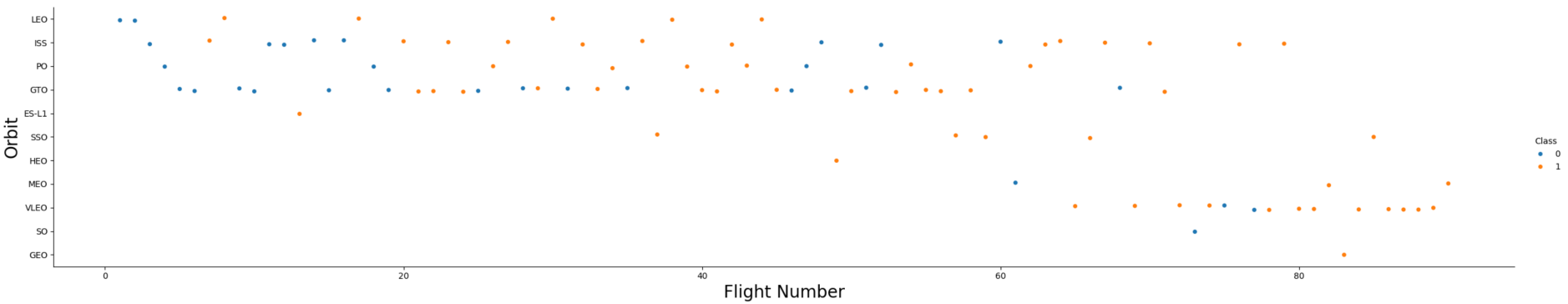


Success Rate vs. Orbit Type

- Orbit Influence on Landing Outcomes
- This figure shows that certain orbits, like SSO, HEO, GEO, and ES-L1, have a 100% success rate, while SO orbit has a 0% success rate. However, further analysis reveals that some orbits (e.g., GEO, SO, HEO, and ES-L1) only have one occurrence, indicating that more data is needed to identify patterns or trends before drawing conclusions.



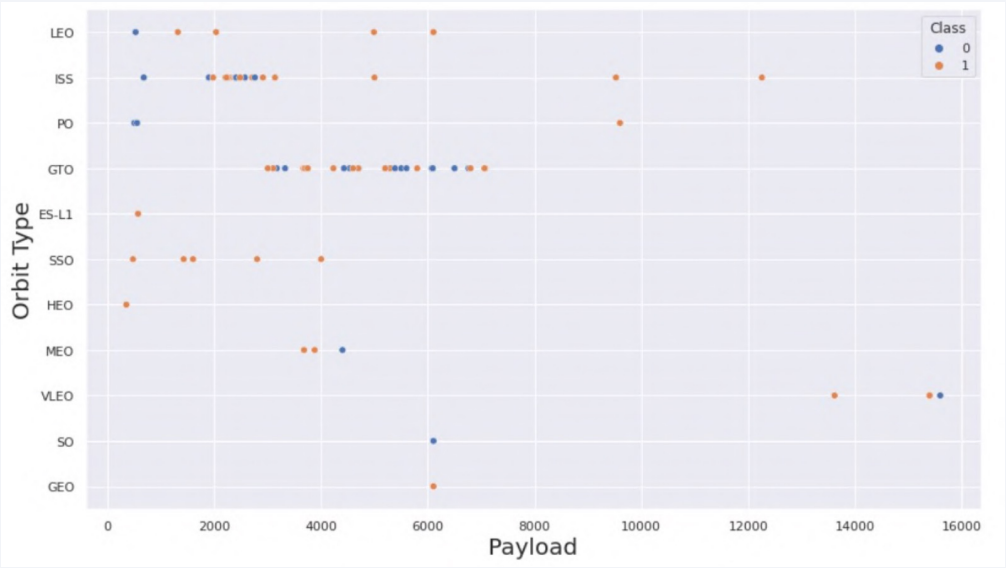
Flight Number vs. Orbit Type



- **Scatter Plot Insights:**
- The scatter plot reveals that, generally, higher flight numbers correlate with greater success rates, particularly in LEO orbits. However, GTO orbits show no clear relationship between flight number and success rate. Orbits with only one occurrence should be excluded due to insufficient data.

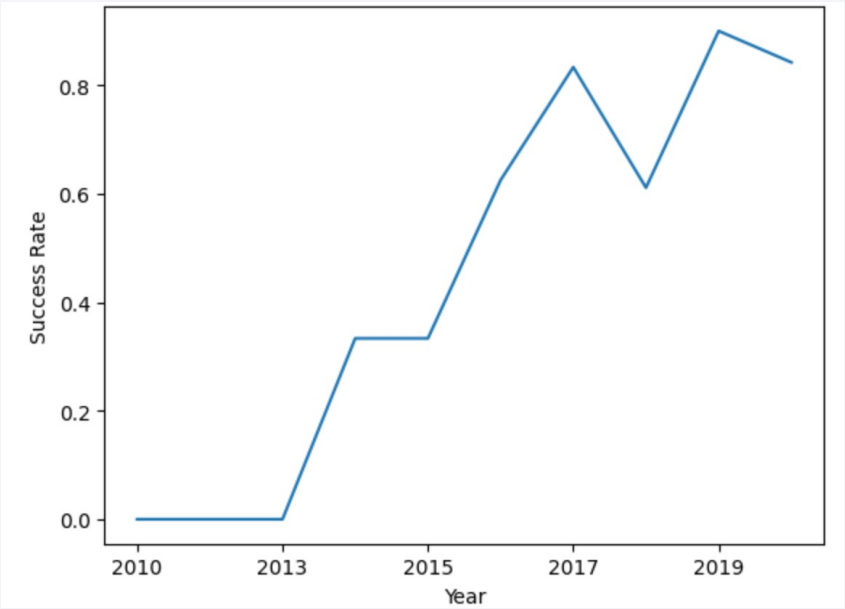
Payload vs. Orbit Type

- Payload Impact on Orbits
- Heavier payloads positively affect LEO, ISS, and PO orbits.
- Negative impact observed on MEO and VLEO orbits.
- No clear relation for GTO orbit.
- More data needed to identify trends for SO, GEO, and HEO orbits.



Launch Success Yearly Trend

- The figure shows a clear upward trend from 2013 to 2020. If this trend continues, the success rate will steadily approach 100% in the coming years.



All Launch Site Names

- Find the names of the unique launch sites: Launch_Site
- CCAFS LC-40
- VAFB SLC-4E
- KSC LC-39A
- CCAFS SLC-40

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

%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'

- Find 5 records where launch sites begin with `CCA`

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

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

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

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

- Calculate the total payload carried by boosters from NASA

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

%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE Customer LIKE 'NASA (CRS)';
```

```
 * sqlite:///my_data1.db
Done.
SUM(PAYLOAD_MASS__KG_)
45596
```

Average Payload Mass by F9 v1.1

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

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

```
[13] # Display average payload mass carried by booster version F9 v1.1

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

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

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

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

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

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

MIN(DATE)
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

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

```
[15] %sql SELECT Booster_Version FROM SPACEXTABLE WHERE "Landing _Outcome" = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

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

Booster_Version

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

```
[16] #  List the total number of successful and failure mission outcomes

     %sql SELECT Mission_Outcome, COUNT(*) AS Total FROM SPACEXTABLE GROUP BY Mission_Outcome;
```

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

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

Boosters Carried Maximum Payload

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

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

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

```
 * 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

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

[34] Generated code may be subject to a licence | AAhdm/Capstone-

```
%sql SELECT substr(Date, 6, 2) AS Month, "Landing _Outcome", Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date, 0, 5) = '2015' AND landing_outcome = 'Failure
```

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

Month	"Landing _Outcome"	Booster_Version	Launch_Site
01	Landing _Outcome	F9 v1.1 B1012	CCAFS LC-40
04	Landing _Outcome	F9 v1.1 B1015	CCAFS LC-40

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

- 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

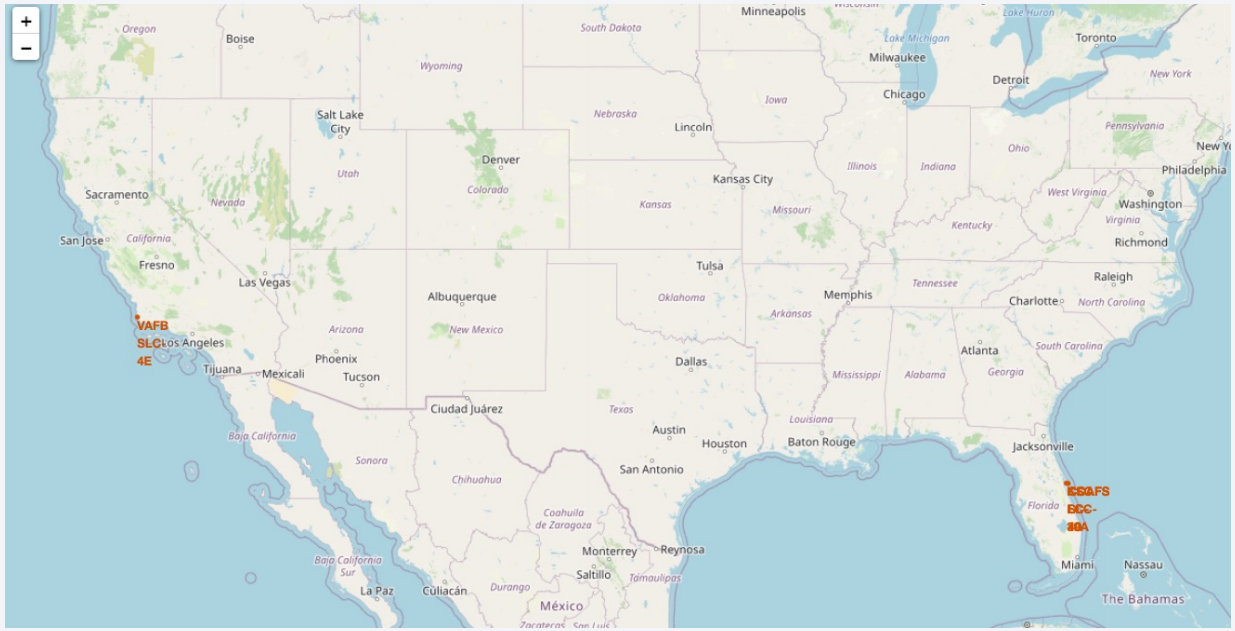
```
Generated code may be subject to a licence |
# prompt: Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

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

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

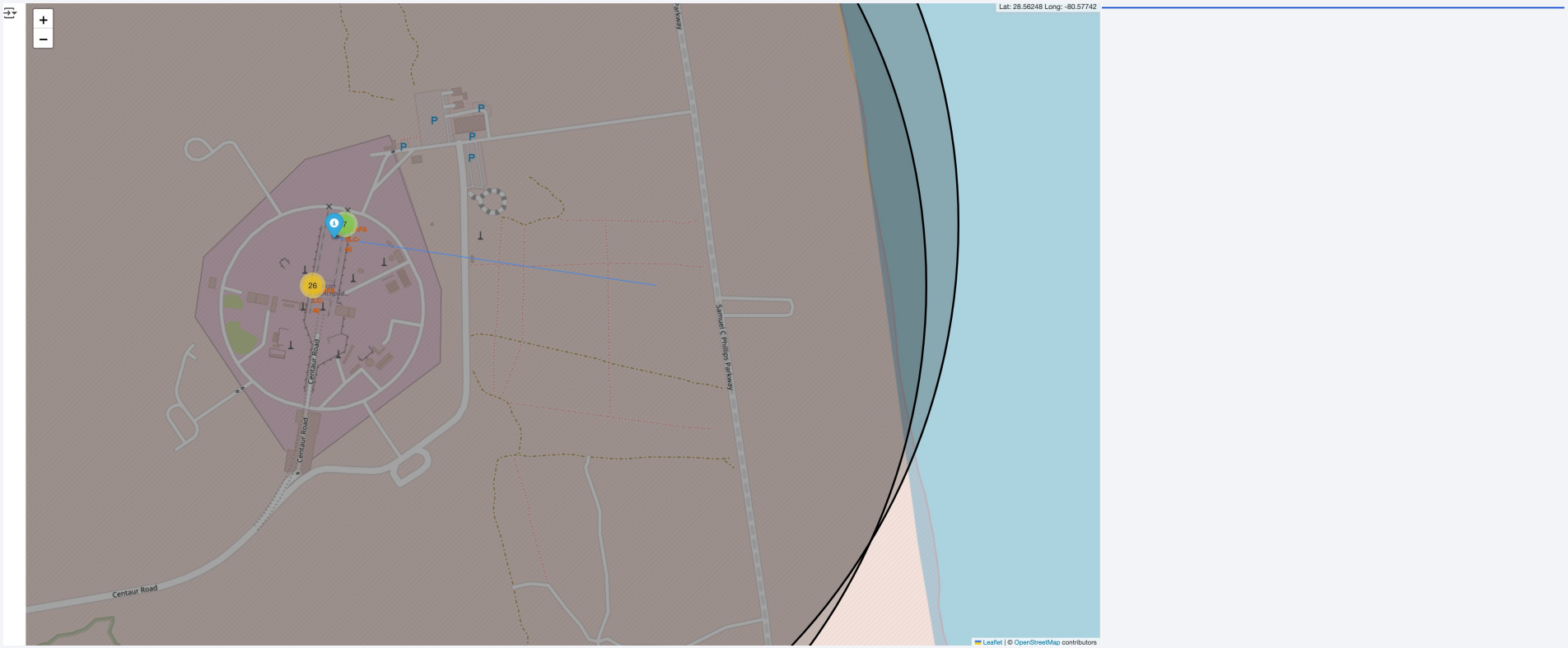
Landing _Outcome	Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1


Section 3
Launch Sites
Proximities Analysis

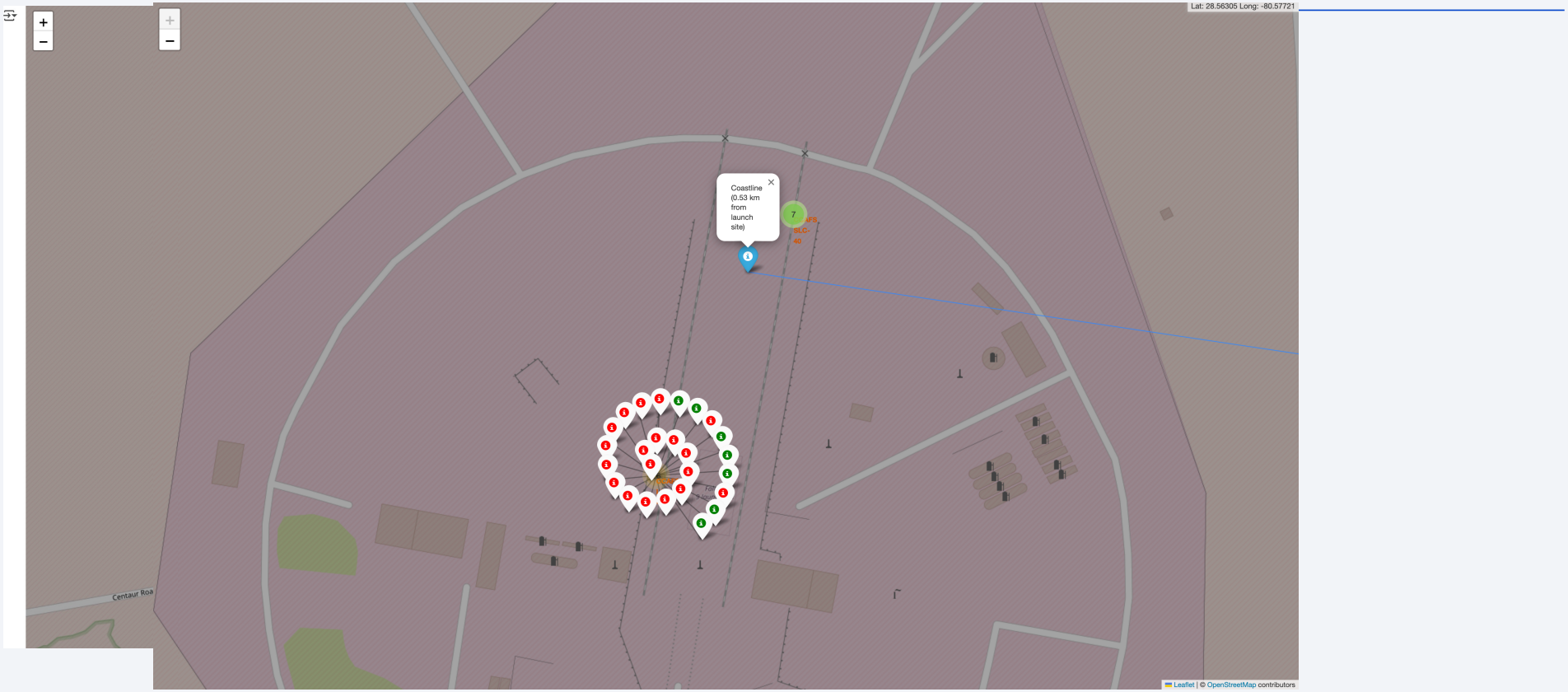
Location of all the Launch Sites



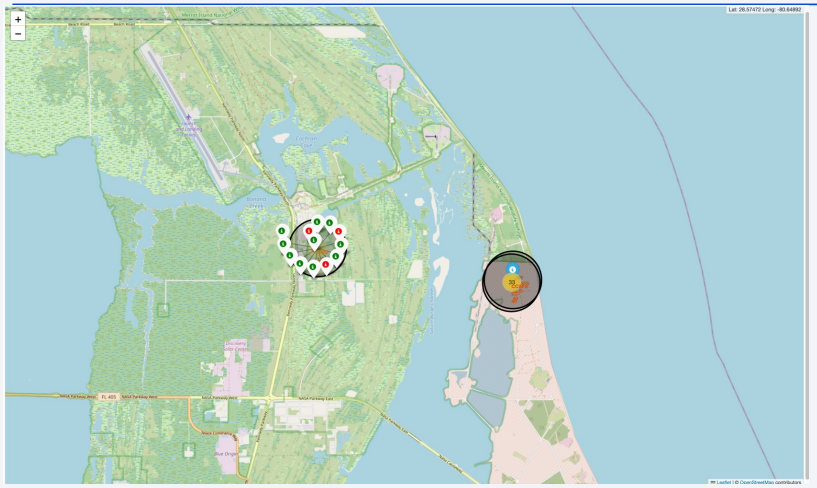
Launch Sites

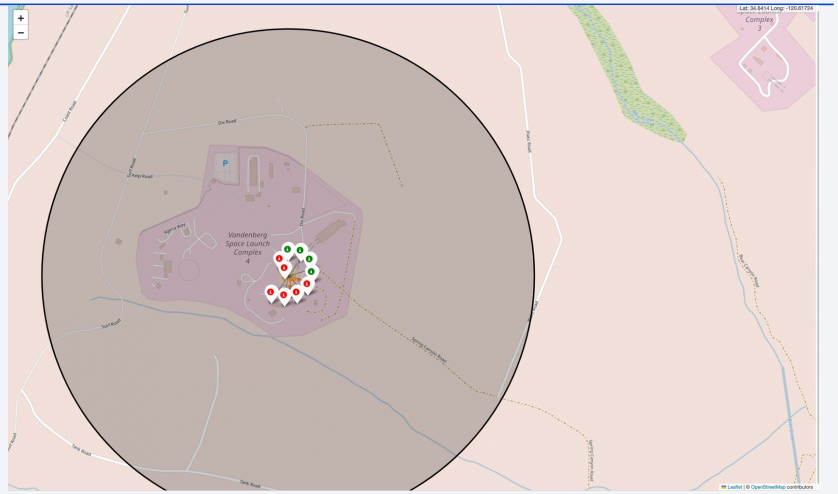


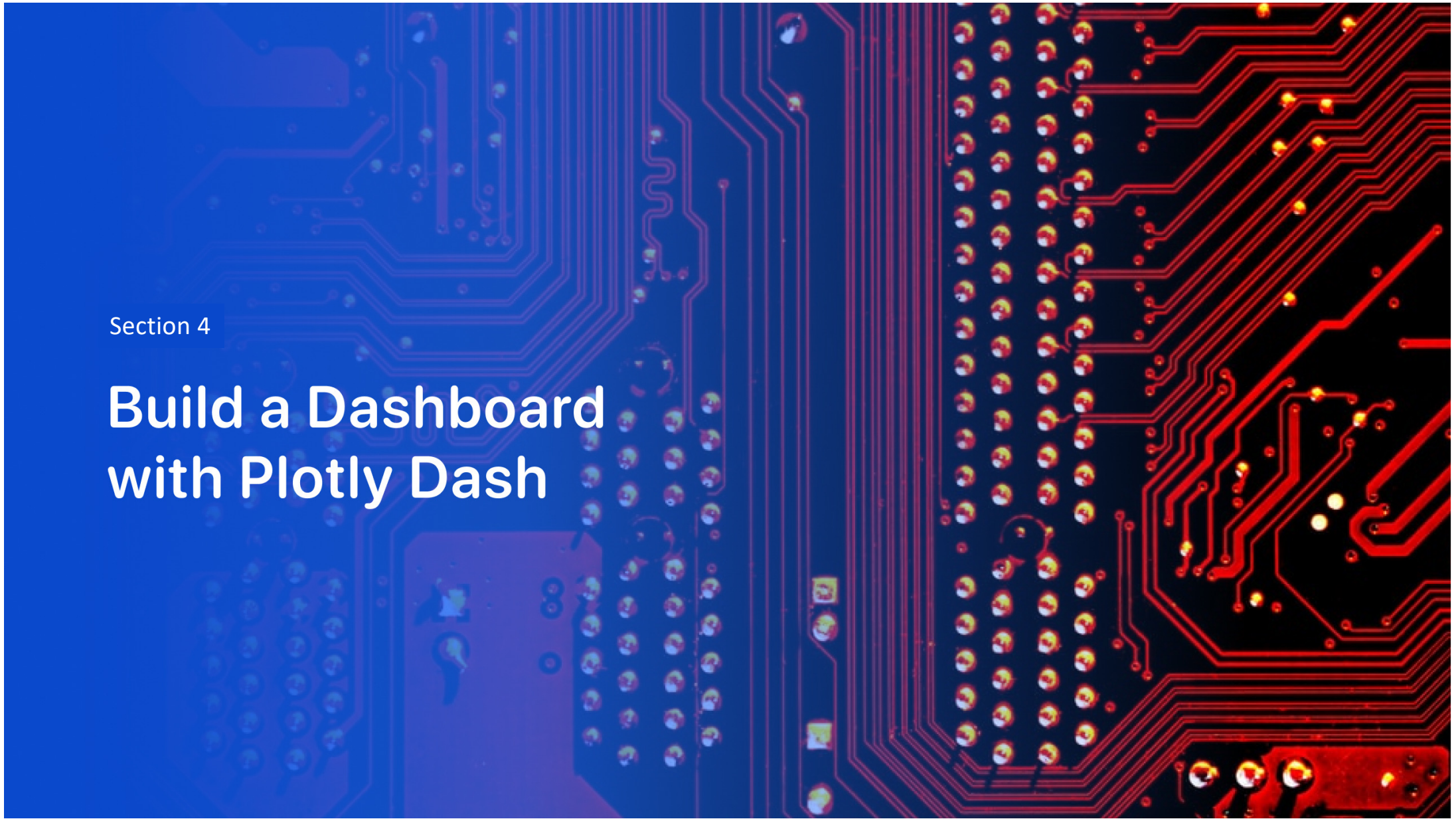
Launch Sites



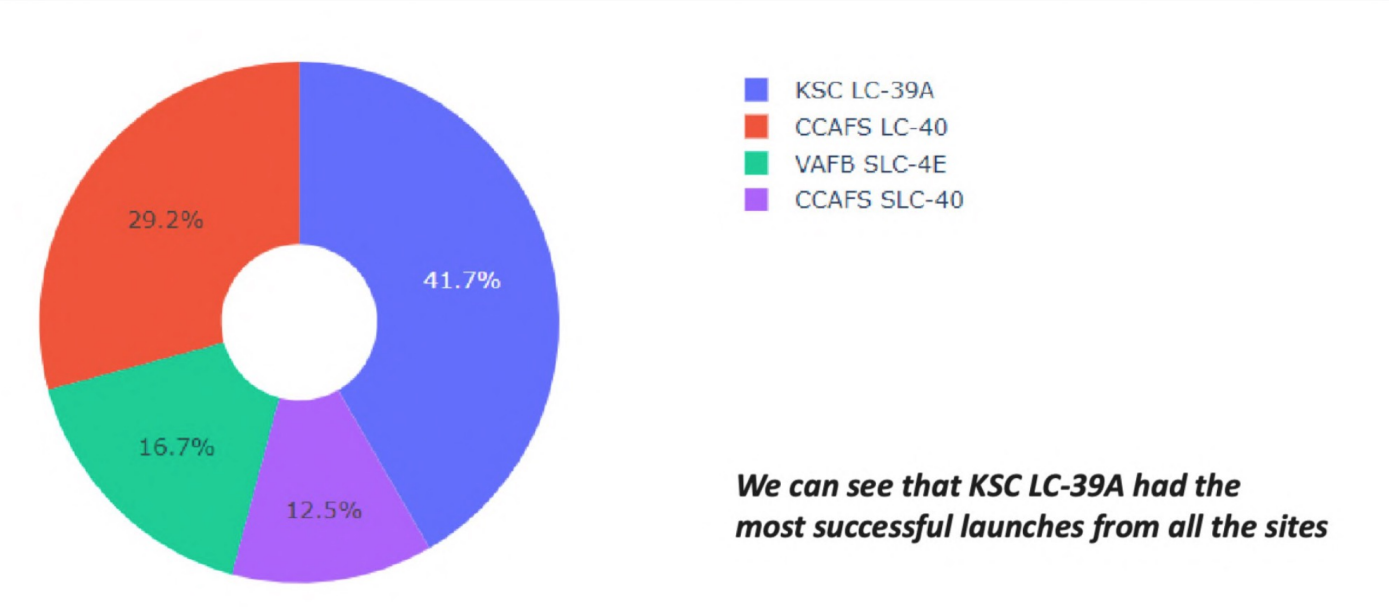
Launch Sites



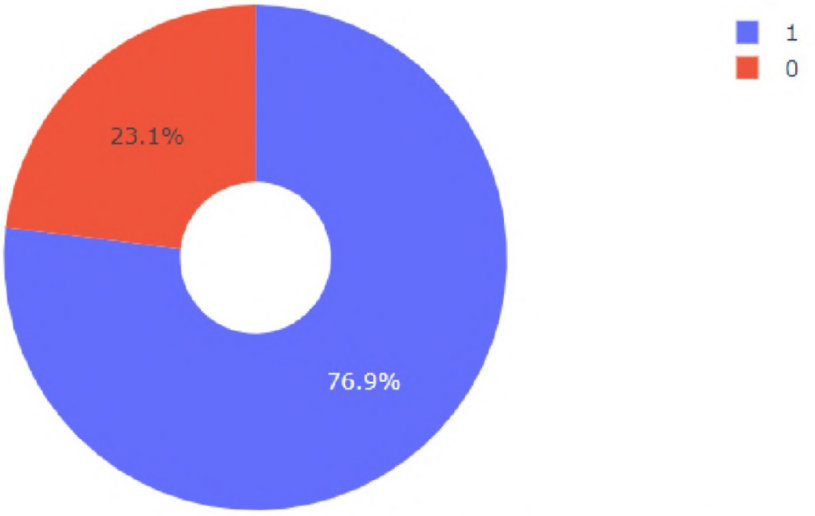



Section 4
Build a Dashboard with Plotly Dash

Launch success count for all sites, in a piechart



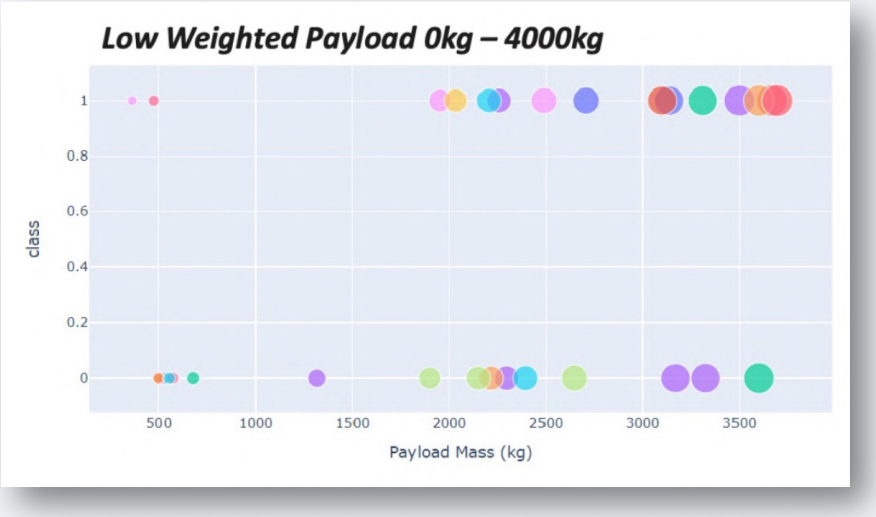
Piechart for the launch site with highest launch success ratio

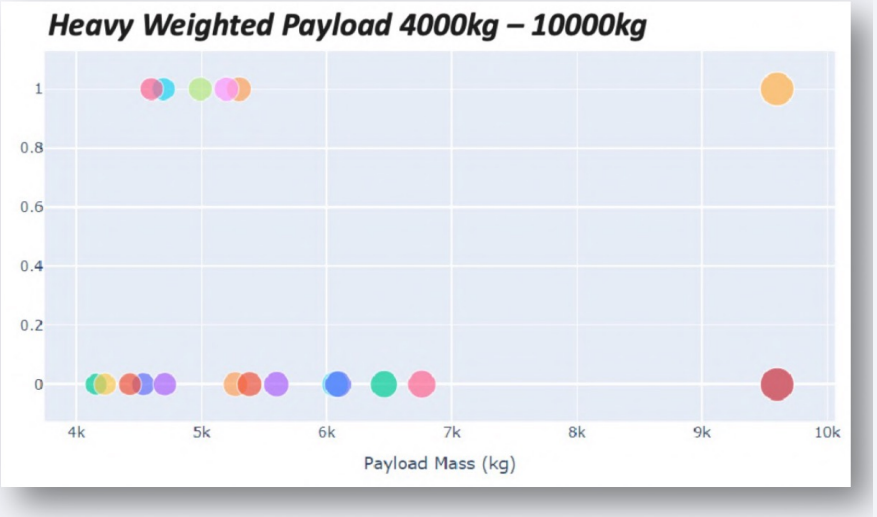


KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs. Launch Outcome scatter plot for all sites

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





Section 5

Predictive Analysis (Classification)

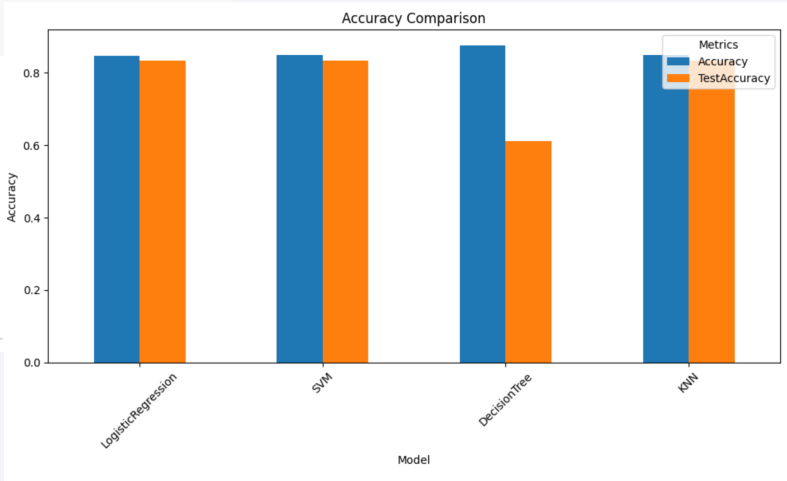
Classification Accuracy

- DecisionTree has the highest classification accuracy

```
Generated code may be subject to a licence |

print("Model\t\tAccuracy\tTestAccuracy")
print("{}\t\t{}\t\t{}".format('LogisticRegression',logreg_cv.best_score_,logreg_cv.score(X_test, Y_test)))
print("{}\t\t{}\t\t{}".format('SVM',svm_cv.best_score_,svm_cv.score(X_test, Y_test)))
print("{}\t\t{}\t\t{}".format('DecisionTree',tree_cv.best_score_,tree_cv.score(X_test, Y_test)))
print("{}\t\t{}\t\t{}".format('KNN',knn_cv.best_score_,knn_cv.score(X_test, Y_test)))

data = {'Model': ['LogisticRegression', 'SVM', 'DecisionTree', 'KNN'],
        'Accuracy': [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_],
        'TestAccuracy': [logreg_cv.score(X_test, Y_test), svm_cv.score(X_test, Y_test), tree_cv.score(X_test, Y_test), knn_cv.score(X_test, Y_test)]}
df = pd.DataFrame(data)
df
```

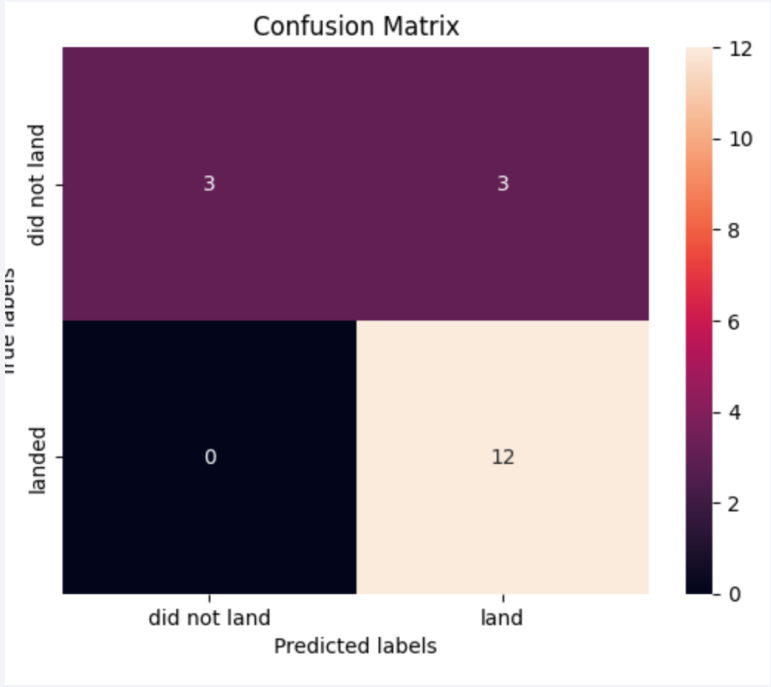
```
Model           Accuracy        TestAccuracy
LogisticRegression              0.8464285714285713              0.8333333333333334
SVM             0.8482142857142856              0.8333333333333334
DecisionTree            0.875           0.6111111111111112
KNN             0.8482142857142858              0.8333333333333334
```

	Model	Accuracy	TestAccuracy
0	LogisticRegression	0.846429	0.833333
1	SVM	0.848214	0.833333
2	DecisionTree	0.875000	0.611111
3	KNN	0.848214	0.833333



Confusion Matrix

- The decision tree classifier's confusion matrix shows it can distinguish between classes, but false positives (unsuccessful landings marked as successful) remain a major issue.



Conclusions

- In conclusion:
- The Tree Classifier Algorithm is the best for this dataset.
- Lighter payloads (≤4000kg) performed better than heavier ones.
- SpaceX's success rate increased from 2013 to 2020, with future improvements expected.
- KSC LC-39A had the highest success rate (76.9%).
- SSO orbit had a 100% success rate, with multiple occurrences

Thank you!