IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Aviwe Dlepu>
<16/04/2025>

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with Data Visualization
  - Exploratory Data Analysis with SQL
  - Interactive Visual Analytics with Folium
  - Predictive Analysis with Machine Learning
- Summary of all results
  - Exploratory Data Analysis
  - Interactive Analytics in Screenshots
  - Predictive Analytics Results

Introduction

- Project background and context

SpaceX advertises Falcon 9 rocket launches at $62 million, compared to competitors charging upwards of $165 million. A major reason for this cost efficiency is SpaceX's ability to reuse the first stage of the rocket.

If we can predict whether the first stage will land successfully, we can estimate the true cost of a launch. This insight would be valuable to companies looking to compete with or bid against SpaceX.

The goal of this project is to build a machine learning pipeline that predicts the likelihood of a first-stage landing success.

- Problems you want to find answers
  - What factors determine whether the rocket lands successfully?
  - How do different features interact to influence the outcome?
  - What operational conditions are required to ensure a successful 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
  - How to build, tune, evaluate classification models

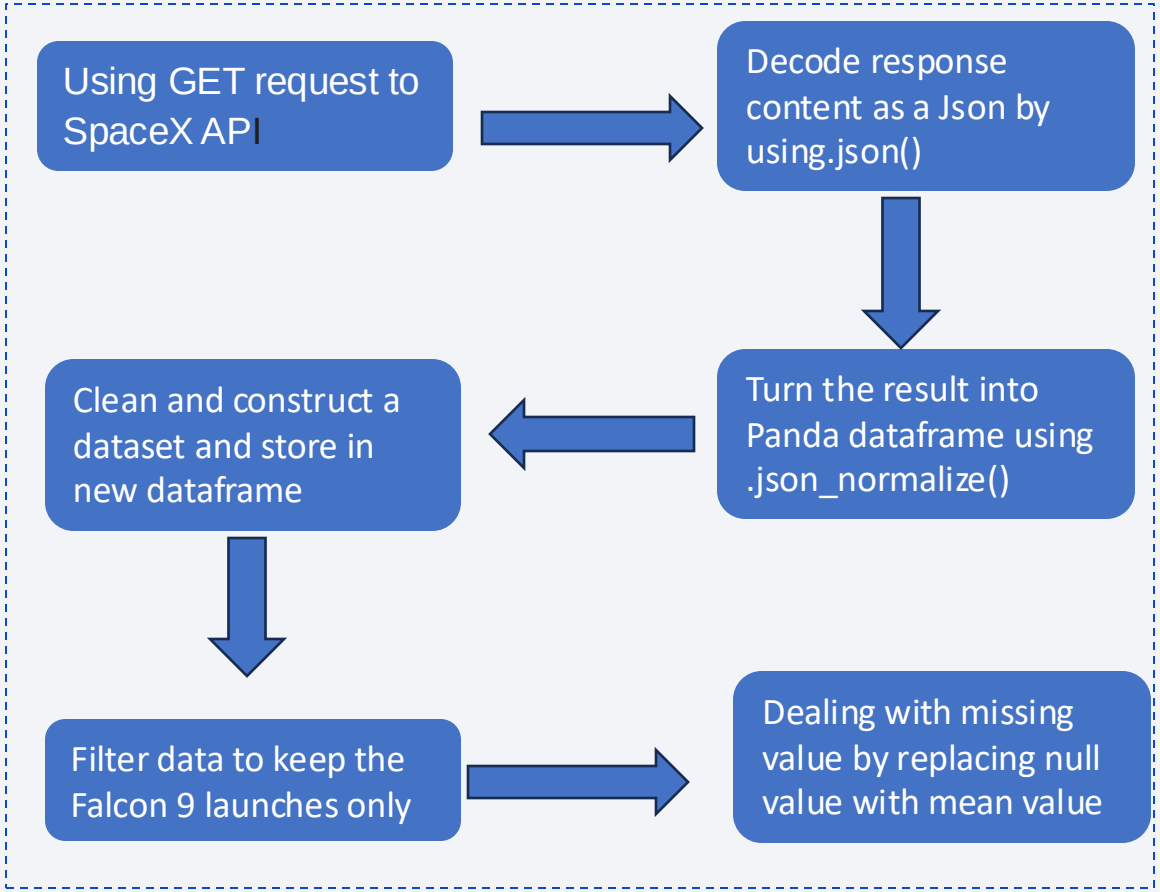
Data Collection

- Describe how data sets were collected.
  - First, we used a GET request to fetch launch data from the SpaceX API.
  - The response content was decoded as JSON using the .json() function and then normalized into a Pandas DataFrame with .json_normalize().
  - We cleaned the dataset by checking for missing values and filled in gaps where necessary to prepare it for analysis.
  - Additionally, we performed web scraping using BeautifulSoup to extract Falcon 9 launch records from Wikipedia.
  - The goal was to parse the HTML launch table and convert it into a structured DataFrame for further analysis.

Data Collection – SpaceX API

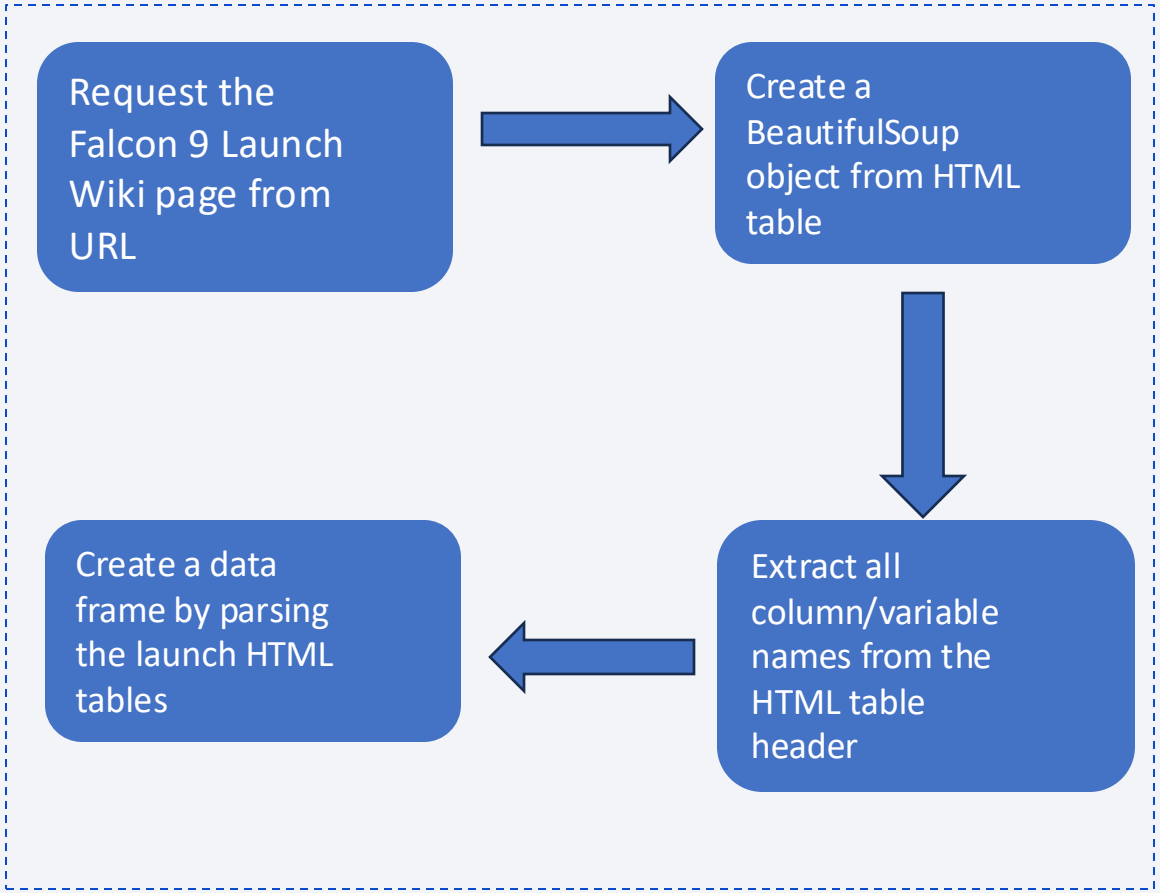
- We used the GET request to the SpaceX API to collect data, clean the requested data and did some data wrangling and formatting, finally store it in a data frame.

- The link to the notebook is: https://github.com/AvyDle/Comprehensive-Data-Science-Study-on-SpaceX-Launches-and-Landings/blob/main/jupyter-labs-webscraping.ipynb



Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is: https://github.com/AvyDle/Comprehensive-Data-Science-Study-on-SpaceX-Launches-and-Landings/blob/main/jupyter-labs-webscraping.ipynb



Data Wrangling

- We calculated the number of launches per launch site and analyzed the frequency of different orbit types.
- From the outcome column, we created a new target label indicating whether the first stage landing was successful.
- The cleaned and labeled dataset was then exported to a CSV file for use in model training and further analysis.
- The link to the notebook is: https://github.com/AvyDle/Comprehensive-Data-Science-Study-on-SpaceX-Launches-and-Landings/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.

EDA with SQL

- Loaded the SpaceX dataset into a PostgreSQL database directly from the Jupyter notebook.
- Performed SQL-based exploratory data analysis to extract insights from the data.
- Key queries included:
  - Retrieved the names of all unique launch sites.
  - Calculated the total payload mass for boosters launched under NASA (CRS) missions.
  - Found the average payload mass for booster version F9 v1.1.
  - Counted the total number of successful and failed mission outcomes.
  - Identified failed drone ship landings, along with their booster versions and launch site names.

- The link to the notebook is: https://github.com/AvyDle/Comprehensive-Data-Science-Study-on-SpaceX-Launches-and-Landings/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- Plotted all SpaceX launch sites on an interactive Folium map.
- Added map objects such as:
  - Markers to indicate each launch location.
  - Circles to represent launch outcomes.
  - Lines to connect launch sites with landing locations (if applicable).
- Assigned landing outcome classes:
  - 0 for failure
  - 1 for success
- Used color-coded marker clusters to visually highlight which launch sites had higher success rates.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

Build a Dashboard with Plotly Dash

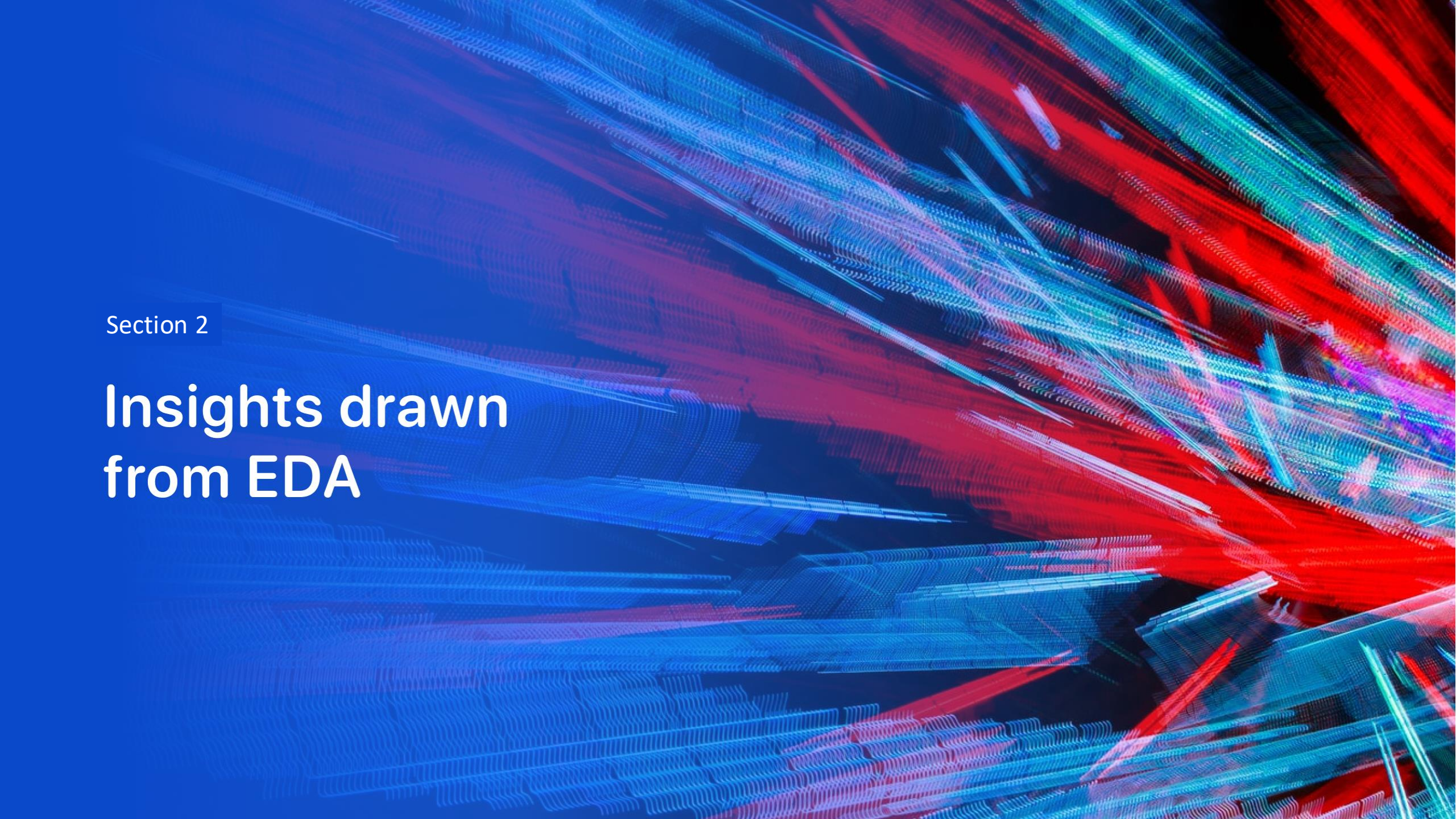
- Built an interactive dashboard using Plotly Dash for dynamic data exploration.
- Created pie charts to display total launch counts by site.
- Developed a scatter plot to show the relationship between launch outcome and payload mass (kg) across different booster versions.
- Enabled interactive filters for users to explore launch patterns by site, payload range, and booster type.
- The link to the notebook is: https://github.com/AvyDle/Comprehensive-Data-Science-Study-on-SpaceX-Launches-and-Landings/blob/main/spacex-dash-app.py

Predictive Analysis (Classification)

- The data was loaded using NumPy and Pandas, and then transformed and split into training and testing datasets.
- Different machine learning models were built and evaluated, with hyperparameter tuning performed using GridSearchCV.
- Accuracy was used as the performance metric for model evaluation.
- The model was iteratively improved through feature engineering and algorithm tuning.
- The best performing classification model was identified through these iterative processes.

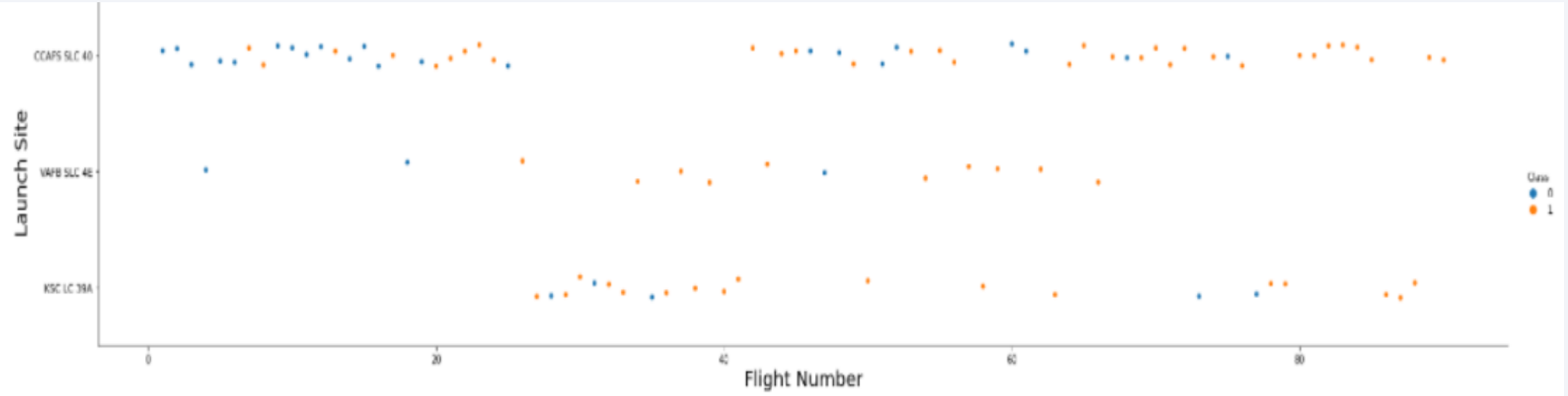
Results

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


Section 2
Insights drawn from EDA

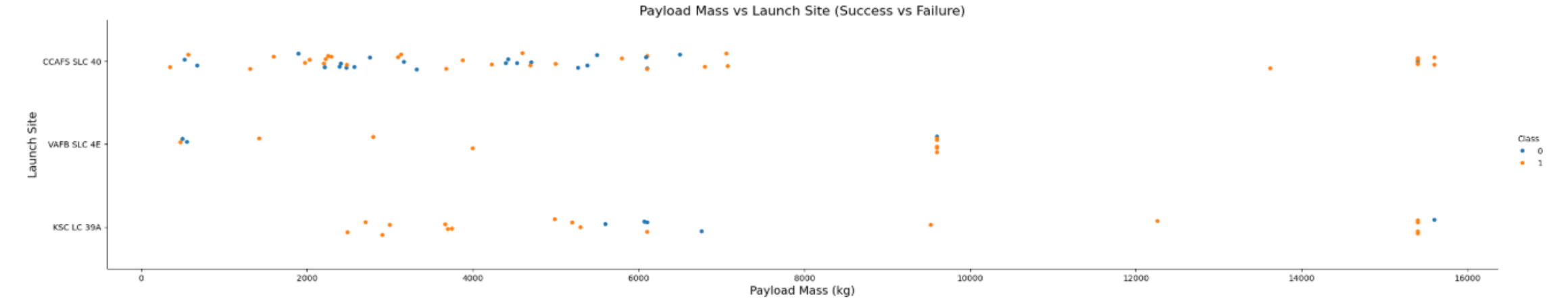
Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



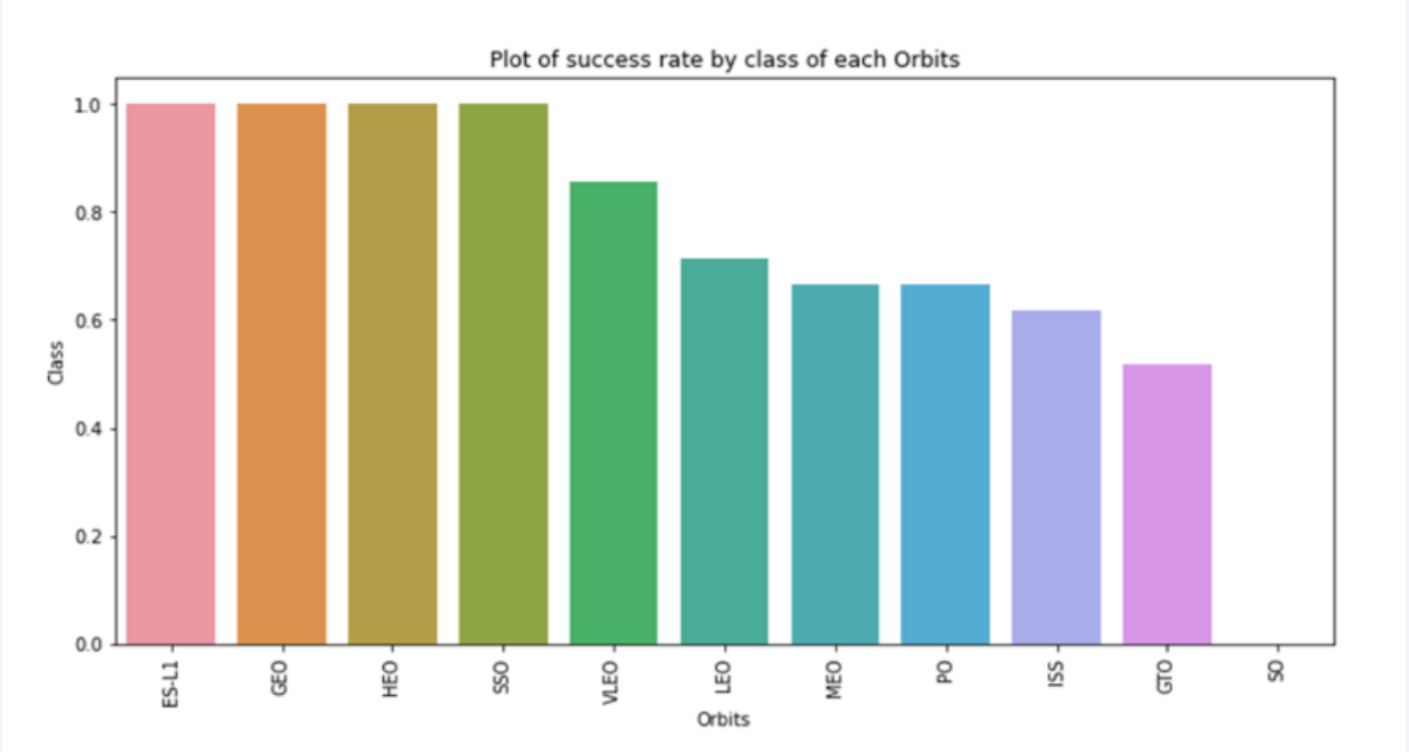
Payload vs. Launch Site

- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



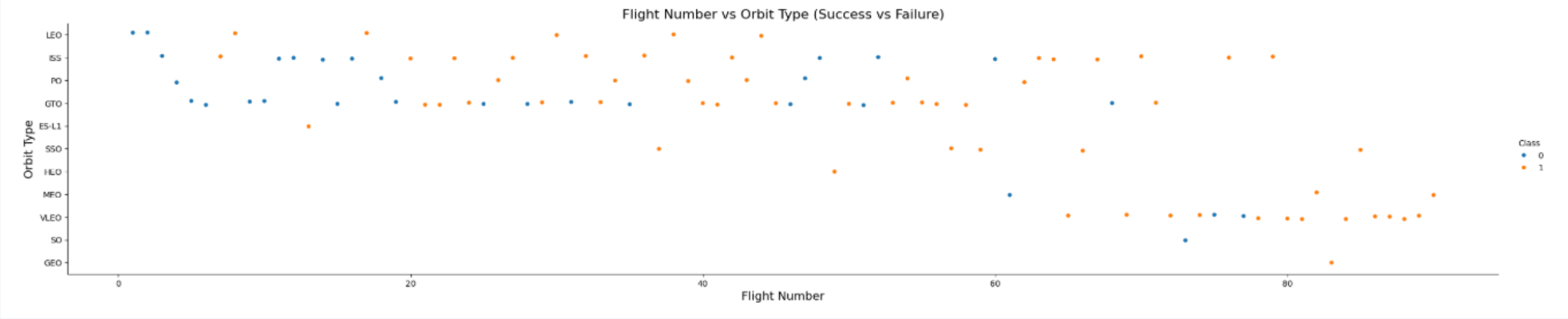
Success Rate vs. Orbit Type

- From the plot, we can see that ES L1, GEO, HEO, SSO, VLEO had the most success rate.



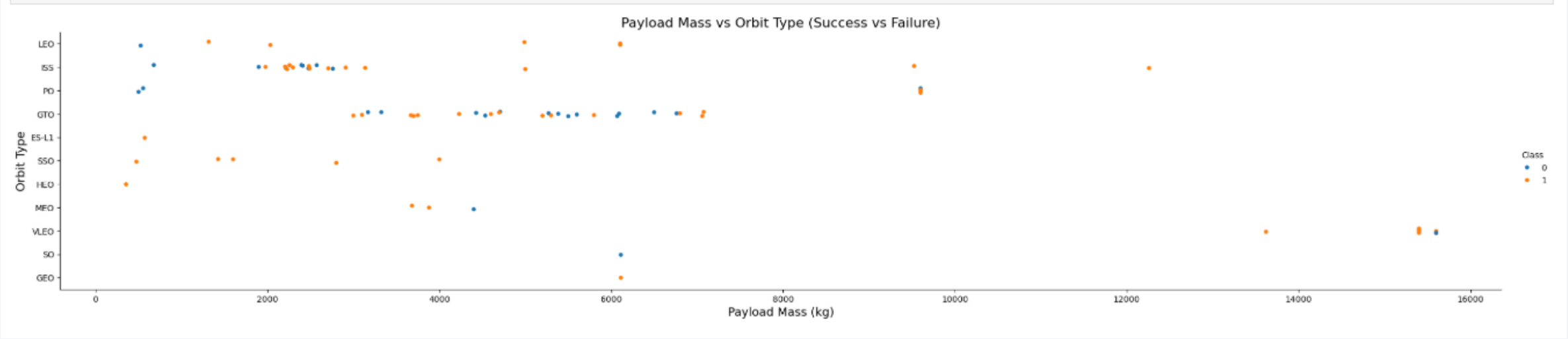
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



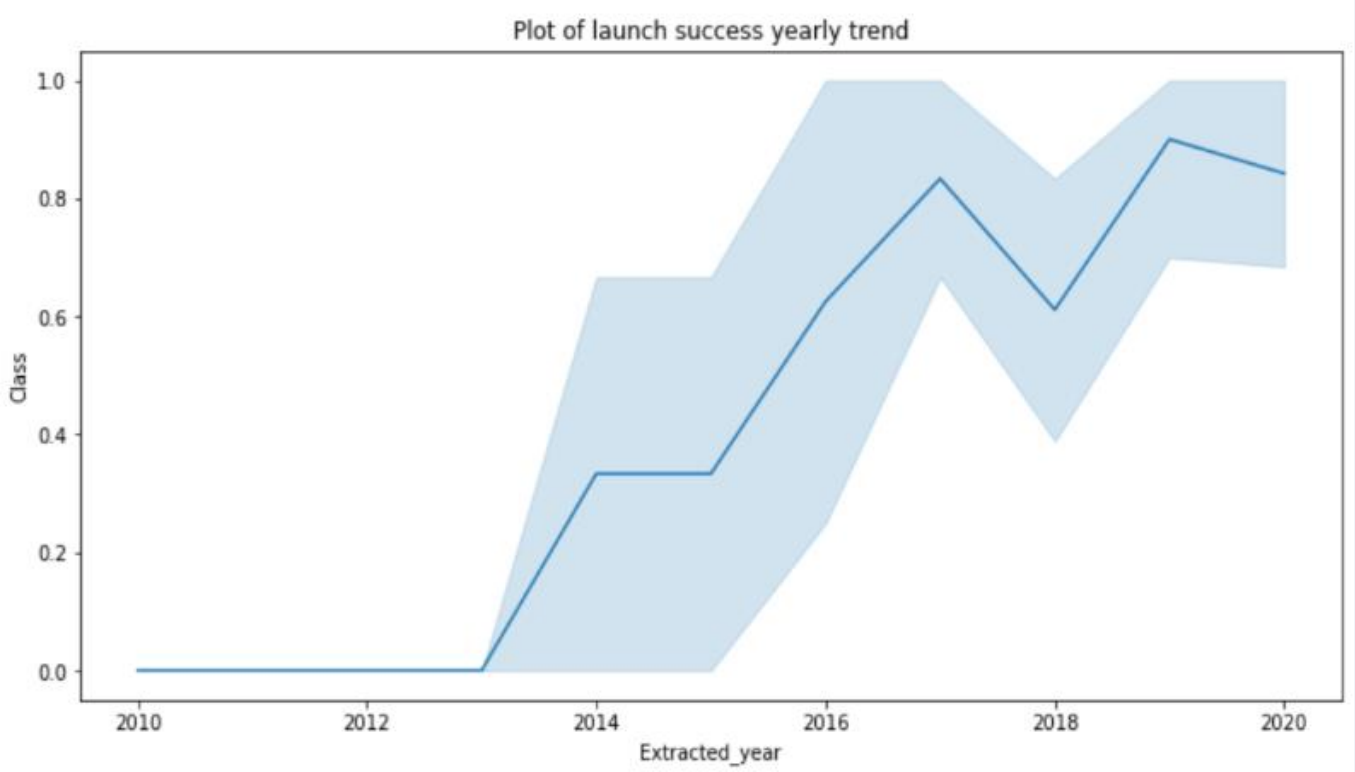
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

- We used the query above to display 5 records where launch sites begin with `CCA`

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

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596using the query below

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

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

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

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

In [14]:

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- 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

In [15]:

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like '%' to filter for WHERE MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

Out[16]:

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

	failureoutcome
0	1

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

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

In [17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

- We used a combinations of the WHERE clause, LIKE , AND , and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

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

In [18]:
```
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- 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.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

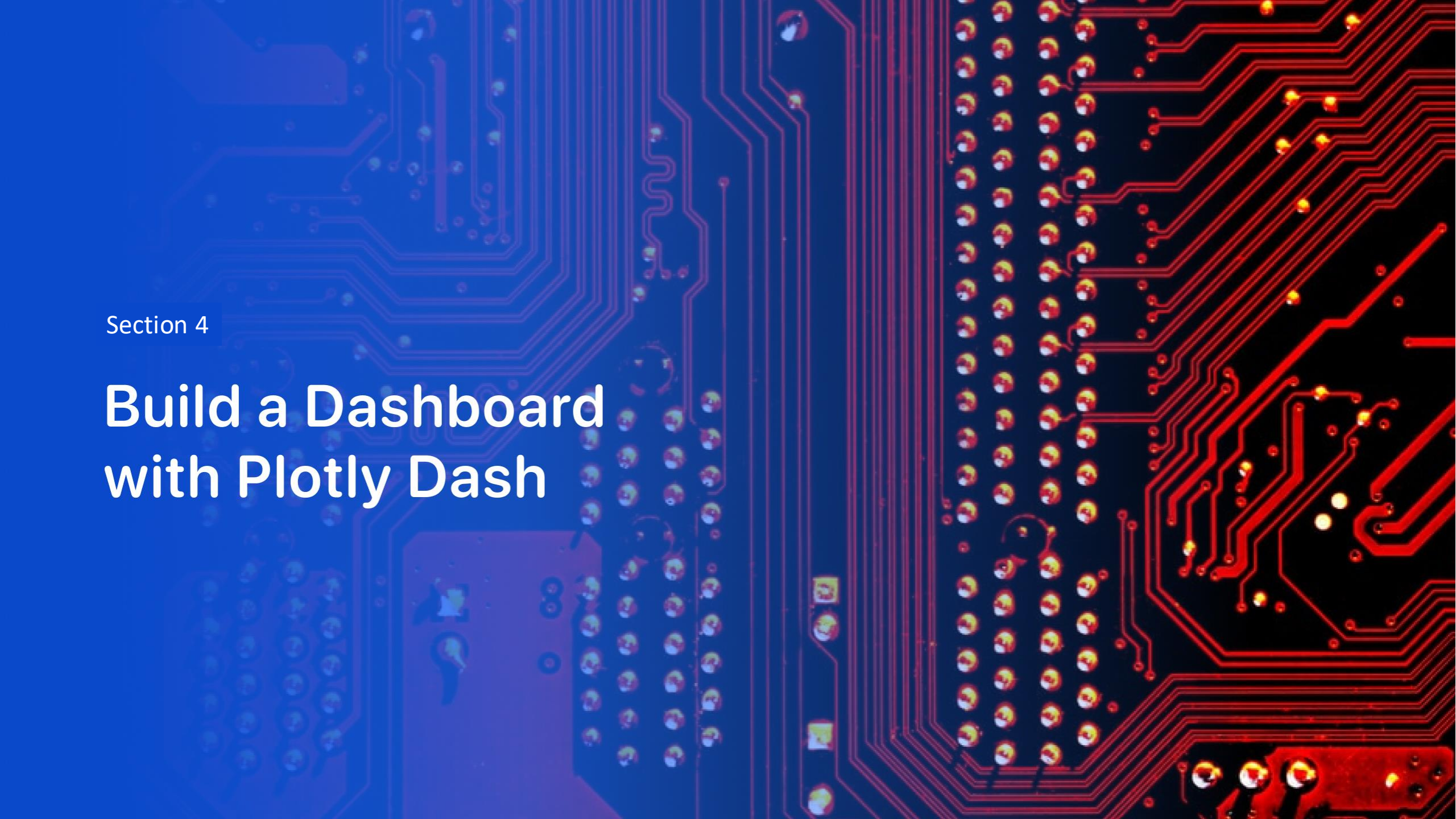
- All SpaceX launch sites are located along the United States of America coasts, specifically in Florida and California.

<Folium Map Screenshot 2>

- Green Markers represent successful launches, while Red Markers indicate failures.

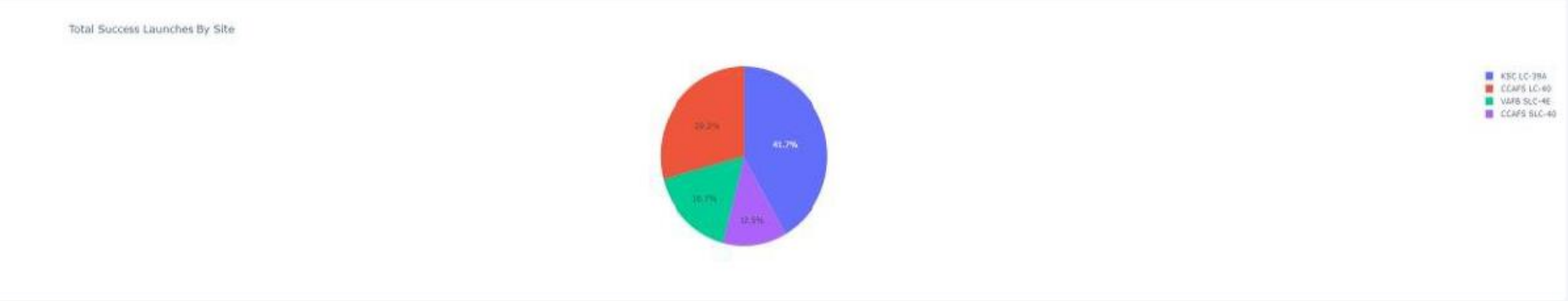
<Folium Map Screenshot 3>

- Launch sites are located relatively close to railways and highways to support transportation needs.


Section 4
Build a Dashboard with Plotly Dash

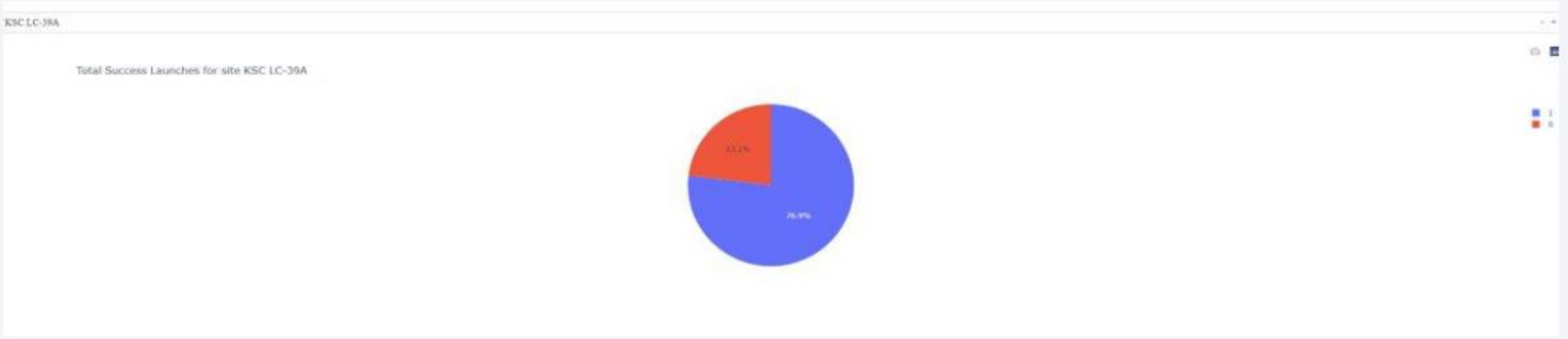
<Dashboard Screenshot 1>

- KSC LC-39A recorded the highest number of successful launches among all the sites.



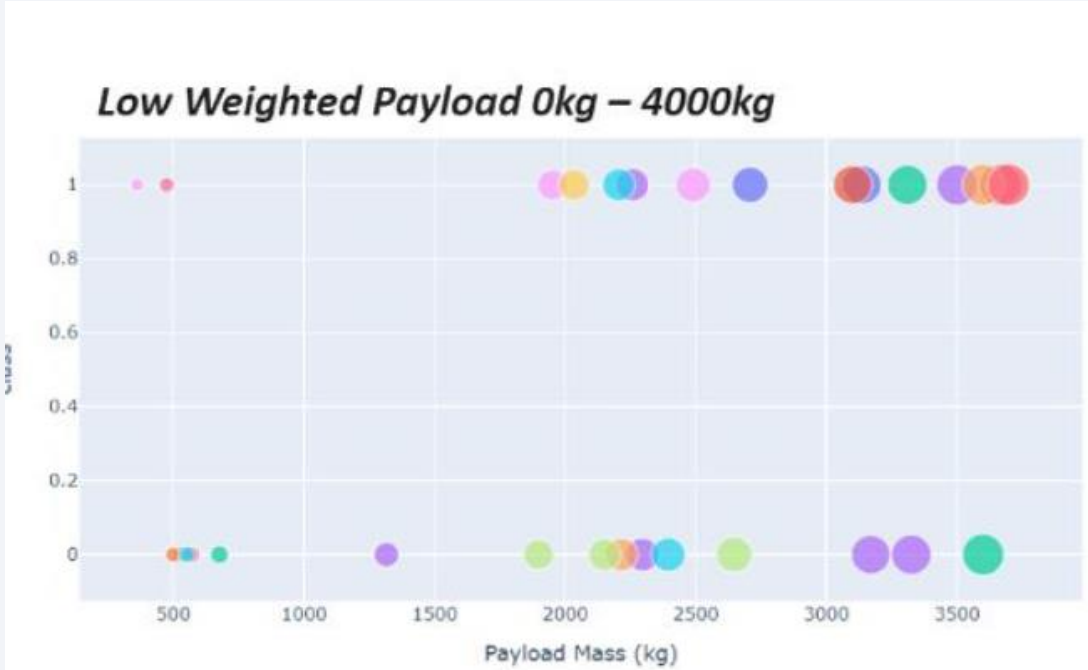
<Dashboard Screenshot 2>

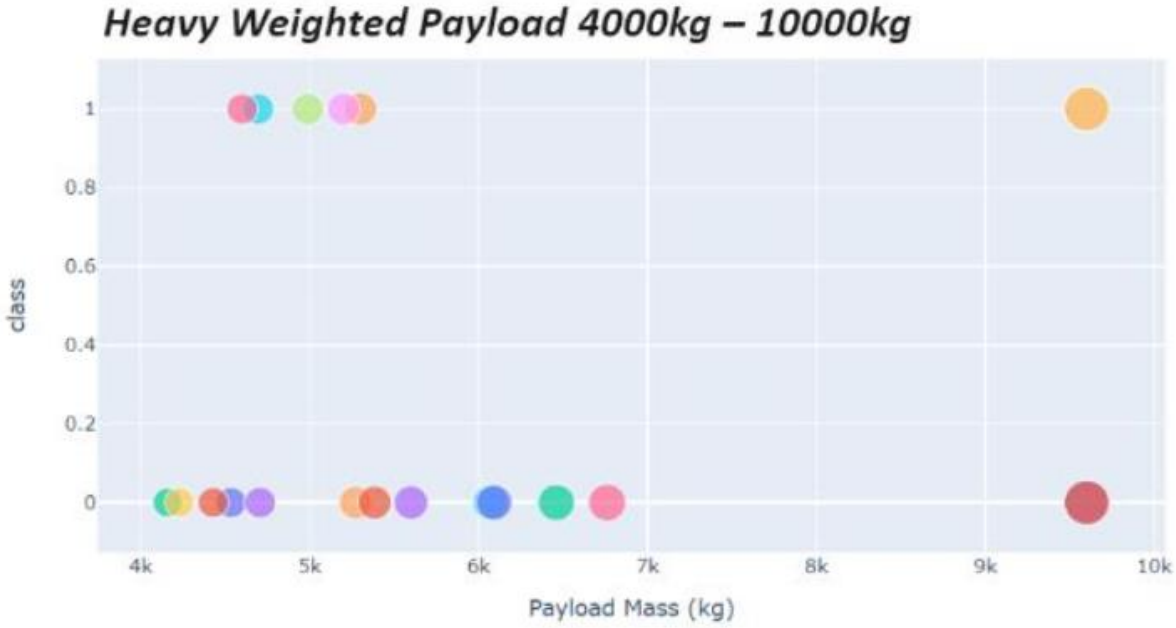
- KSC LC-39A achieved a success rate of 76.9%, with a failure rate of 23.1%.



<Dashboard Screenshot 3>

- Success rates are higher for lower payload masses compared to heavier payloads.





Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The Decision Tree Classifier achieved the highest classification accuracy among all the models.

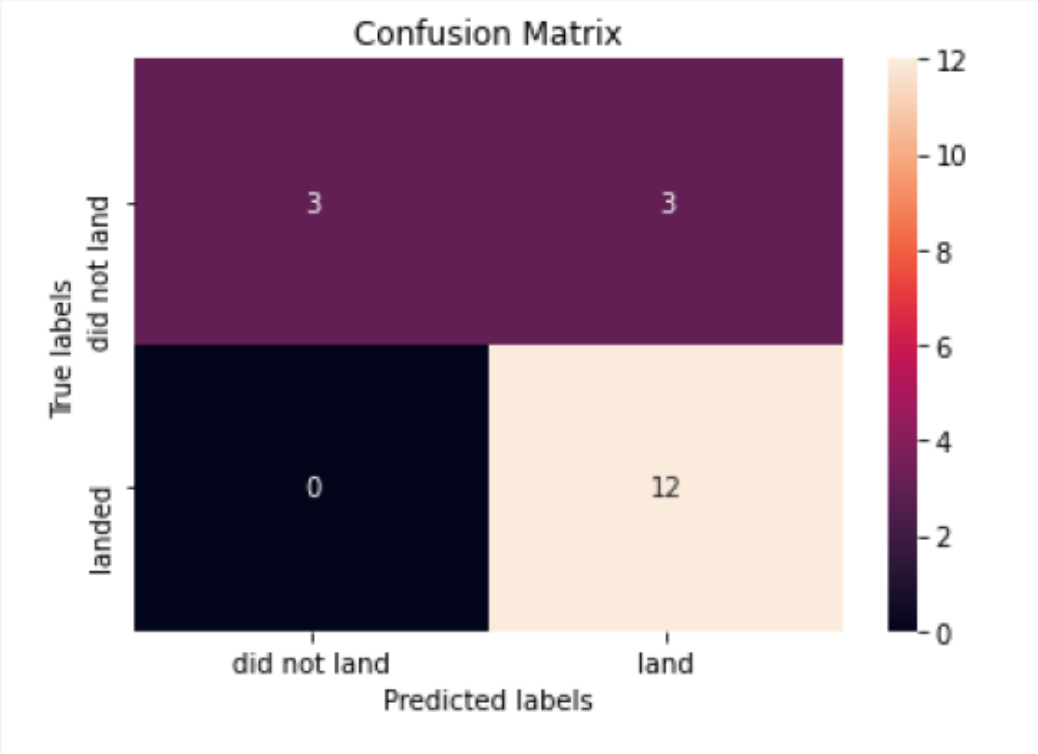
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

- The confusion matrix for the Decision Tree Classifier indicates that the model can differentiate between classes. However, it struggles with false positives, misclassifying unsuccessful landings as successful ones.



Conclusions

- Launch sites with a higher number of flights tend to have higher success rates.
- Launch success rates showed a steady increase from 2013 to 2020.
- Orbits such as **ES-L1, GEO, HEO, SSO, and VLEO** had the highest success rates.
- **KSC LC-39A** recorded the most successful launches among all sites.
- The **Decision Tree Classifier** outperformed other models and proved to be the most effective for this prediction task.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!