IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Name>
<Date>

Outline

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

Executive Summary

- Data was collected from the SpaceX API and Wikipedia page. A 'class' column labeled successful landings. Data exploration used SQL, visualizations, Folium maps, and dashboards. Features were selected, categorical variables were converted using one-hot encoding, and data was standardized. GridSearchCV identified optimal parameters for machine learning models, and accuracy scores were visualized.

- Models were developed using , Logistic Regression, Support Vector Machine, Decision Tree Classifier, and K Nearest Neighbors. All achieved similar accuracy around 83.33%, with a tendency to over-predict successful landings.

Introduction

•SpaceY, a competitor to SpaceX, uses SpaceX Falcon 9 rocket data to assess first-stage landing successes and determine launch costs to bid against SpaceX for rocket launches. SpaceX's Falcon 9 launch costs $62 million, compared to over $165 million for other companies.

•Python and Jupyter notebooks are used for data collection and analysis throughout the project. These notebooks and the final PDF report are stored in my GitHub repository.

•The report includes key areas: data collection methodology, data wrangling, exploratory data analysis (EDA), interactive data visualization, machine learning (ML) classification model development, and model evaluation.

•The accuracy of different ML algorithms in predicting Falcon 9 first-stage landing outcomes is compared.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data used in this project were collected from the SpaceX REST API and the Wikipedia launch table.
- Performed data wrangling
  - The data wrangling process involved cleaning, preparing the data for visualization, and extracting information for use in machine learning predictive models such as logistic regression, support vector machine (SVM), decision tree, and K-nearest neighbors (KNN).
- Performed exploratory data analysis (EDA) using visualization and SQL
- Performed interactive visual analytics using Folium and Plotly Dash
- Performed predictive analysis using classification models

Data Collection

The data collection process involved a combination of API requests from the SpaceX public API and web scraping data from a table in SpaceX's Wikipedia entry.

The API provided detailed data columns such as FlightNumber, Date, BoosterVersion, PayloadMass, Orbit, LaunchSite, Outcome, and more.

The web scraping process extracted columns like Flight No., Launch site, Payload, PayloadMass, Orbit, Customer, Launch outcome, Booster version, Booster landing, Date, and Time.

Data Collection – SpaceX API

Link to Github

1. Initialize API Connection:

Set up the connection to the SpaceX API to prepare for data requests.

2. Send API Request:

Formulate and send a request to the SpaceX API to retrieve the desired data.

3. Receive API Response:

Wait for the API to respond and receive the data.

4. Check Response Status:

Verify that the response status is OK. If not, log the error and resend the request.

5. Parse and Store Data:

Extract and parse the received data, then store it in a suitable format (e.g., DataFrame, database).

6. Validate Data:

Ensure the collected data is complete and accurate. If not, log the error and resend the request if necessary.

Data Collection - Scraping

- Link to Github - Web scraping

1. Library Imports and Setup:

Libraries (requests, BeautifulSoup, pandas) are imported.

2. Requesting the Wikipedia Page:

GET request is made to a static URL of the Wikipedia page to retrieve launch records.

3. Creating BeautifulSoup Object

4. Extracting, Converting and storing data into dataframe

Data Wrangling

- Various data wrangling steps are carried out, including handling missing values, converting data types, and creating new columns from existing data.
- Carried out visualization of launch sites, payload masses, and orbits to understand their distribution and impact on the landing outcome.
- Specific columns like FlightNumber, PayloadMass, Orbit, LaunchSite, and LandingOutcome are extracted and cleaned for use in machine learning models.
- The cleaned and processed DataFrame is saved for further analysis

Link to Github - Data Wrangling

EDA with Data Visualization

Data visualization using Matplotlib and Seaborn were conducted. The trends in Flight Number, Payload Mass, Launch Site, Orbit, Class are explored

Plots Used:

- Flight Number vs. Payload Mass
- Flight Number vs. Launch Site
- Payload Mass vs. Launch Site
- Orbit vs. Success Rate
- Flight Number vs. Orbit
- Payload vs. Orbit

Scatter plots, line charts, and bar plots were utilized to compare relationships between variables to determine if any relationships exist, which could then be used to train the machine learning model.

Data Visualisation - Github link

EDA with SQL

- Established a connection to a SQL database, where the SpaceX dataset is loaded and stored in a table
- Various SQL queries are executed to explore and analyze the dataset. Tasks include counting launches by site, calculating total payload mass for specific customers, and finding average payload mass for different booster versions.
- The analysis involved counting the total number of successful and failed mission outcomes and identifying boosters that have carried the maximum payload mass.

Github Link to Exploratory Data Analysis

Build an Interactive Map with Folium

1. Marked all Launch Sites:
Created a map using Folium and add markers with circles, popup labels, and text labels to each launch site using their longitude and latitude coordinates. This will display the geographical locations relative to the equator.

2. Indicated Success/Failure of Launches:
Used colored markers on the map to indicate successful or failed launches for each site.

3. Calculated Distances:
Computed the distance between each launch site and its proximities.

Github link
https://github.com/Sruthimohan05/IBM_data_science_capstonee/blob/main/Visualisation%20using%20Folium.ipynb

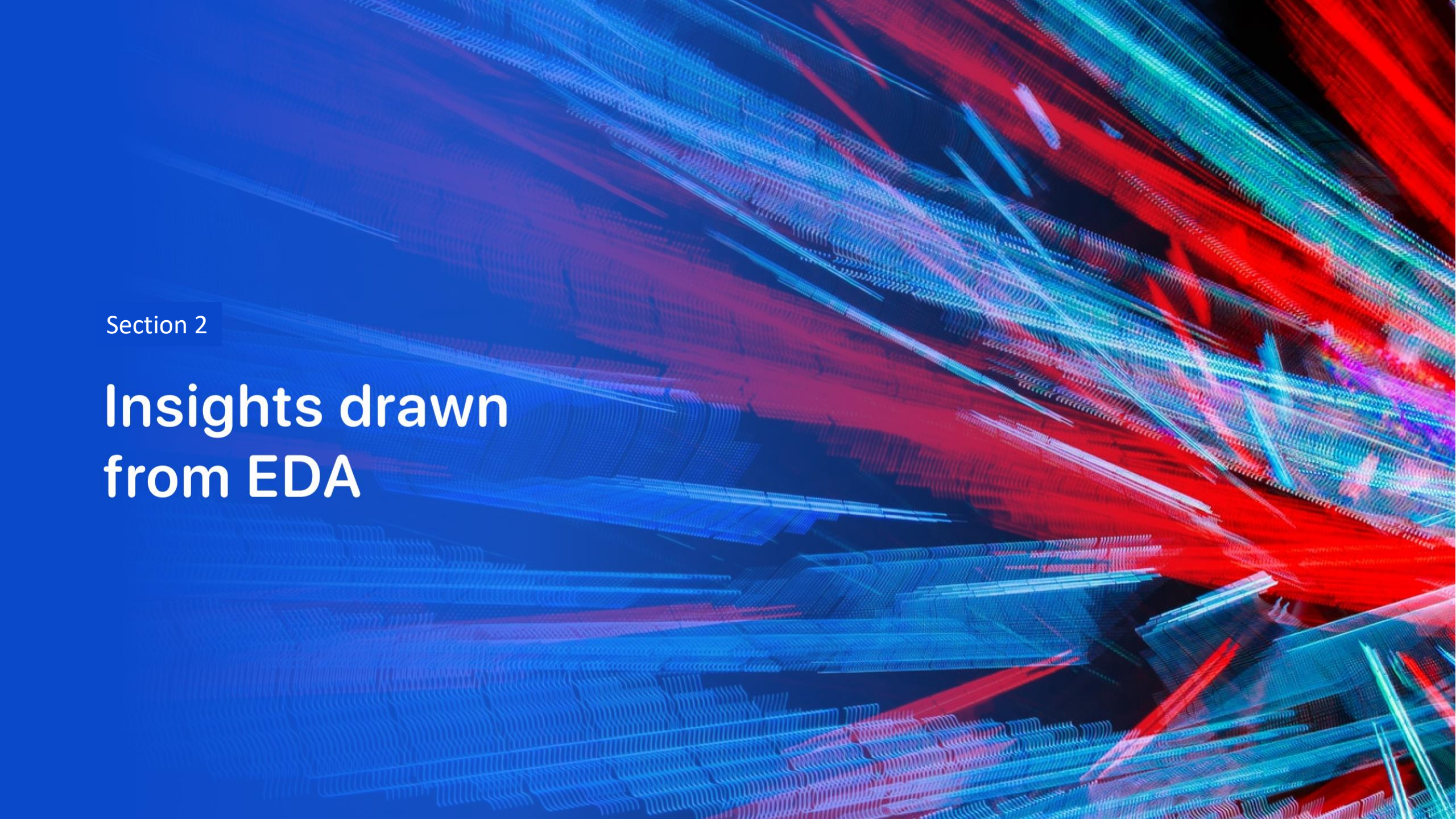
Build a Dashboard with Plotly Dash

The dashboard includes a pie chart and a scatter plot:

- The pie chart can display the distribution of successful landings across all launch sites or show the success rates for individual launch sites.
- The scatter plot accepts two inputs: selection of all sites or an individual site, and a payload mass range adjustable with a slider between 0 and 10,000 kg.
- The pie chart visualizes the success rates of launch sites.
- The scatter plot illustrates how success varies by launch sites, payload mass, and booster version category.

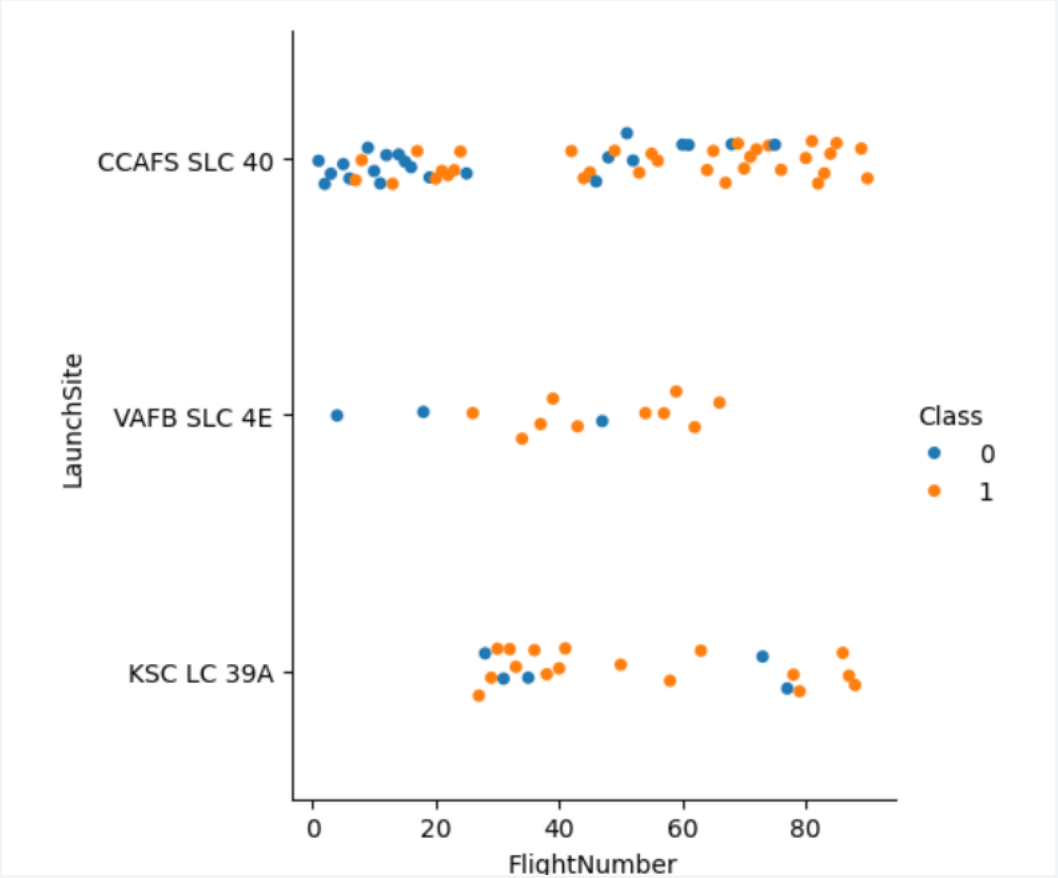
Github link

https://github.com/Sruthimohan05/IBM_data_science_capstonee/blob/main/Dash_Plotly.py


Section 2
Insights drawn from EDA

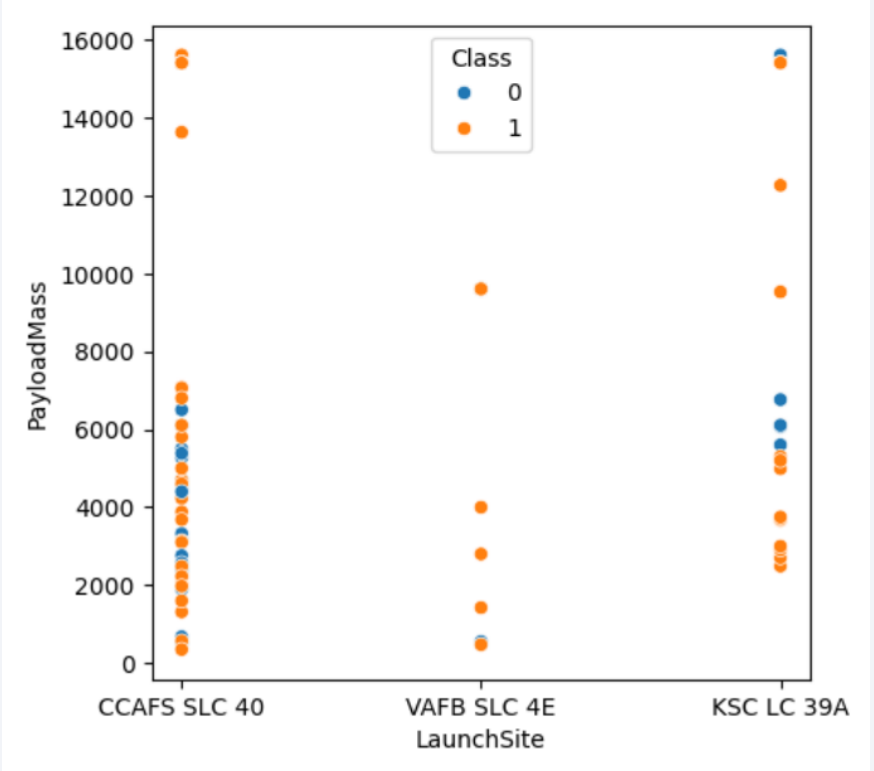
Flight Number vs. Launch Site

- Scatter plot of Flight Number vs. Launch Site
- More flights from launch site CCAFS SLC 40
- Newer flights have higher success rate than older ones



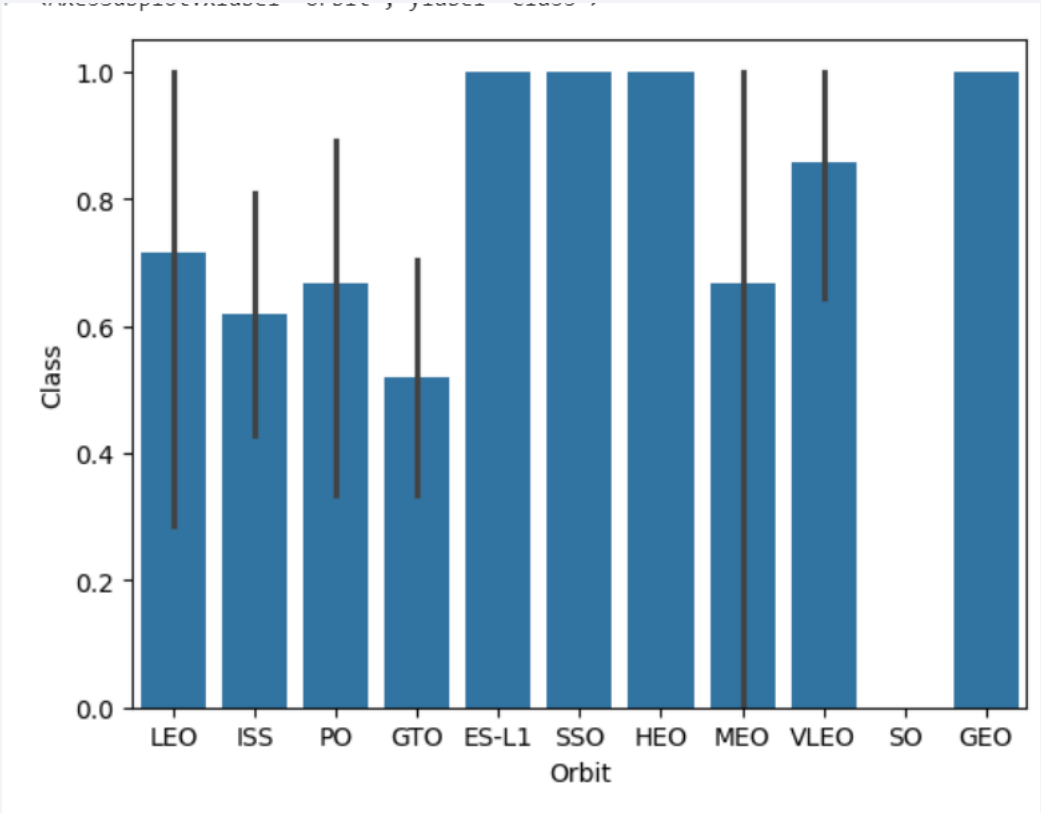
Payload vs. Launch Site

- Scatter plot of Payload vs. Launch Site
- VAFB SLC 4E launch site deploys payloads with mass less than 10000 kg and all launches were successful



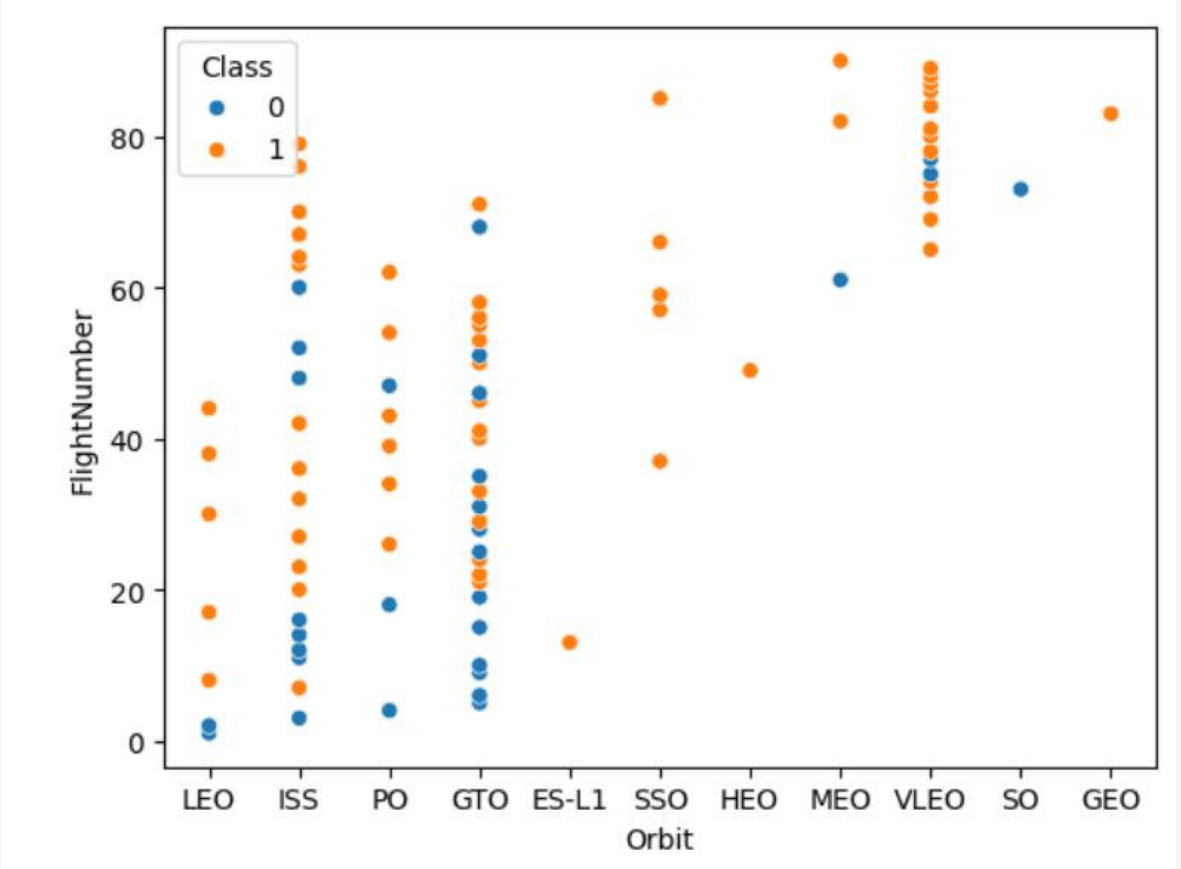
Success Rate vs. Orbit Type

- Bar chart for the success rate of each orbit type
- Four orbits have 100% success rate
- All orbits have success rate above 40% except SO



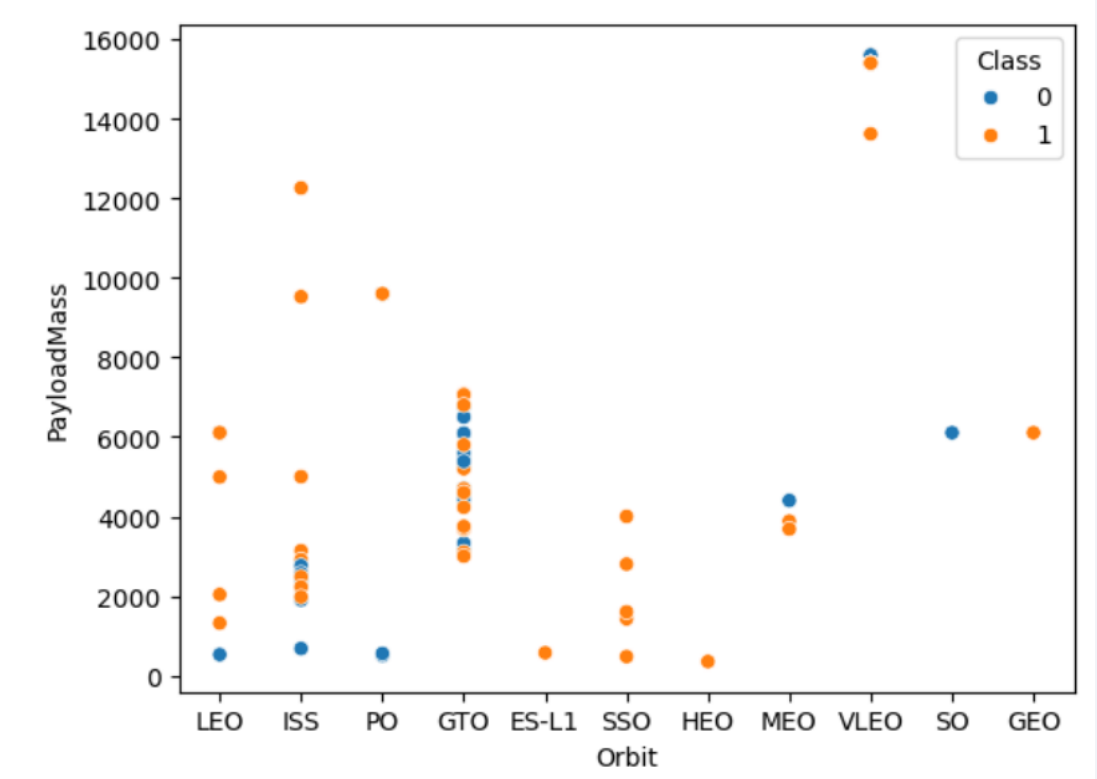
Flight Number vs. Orbit Type

- A scatter point of Flight number vs. Orbit type is shown
- The majority of the flights were launches to the ISS and GTO orbits



Payload vs. Orbit Type

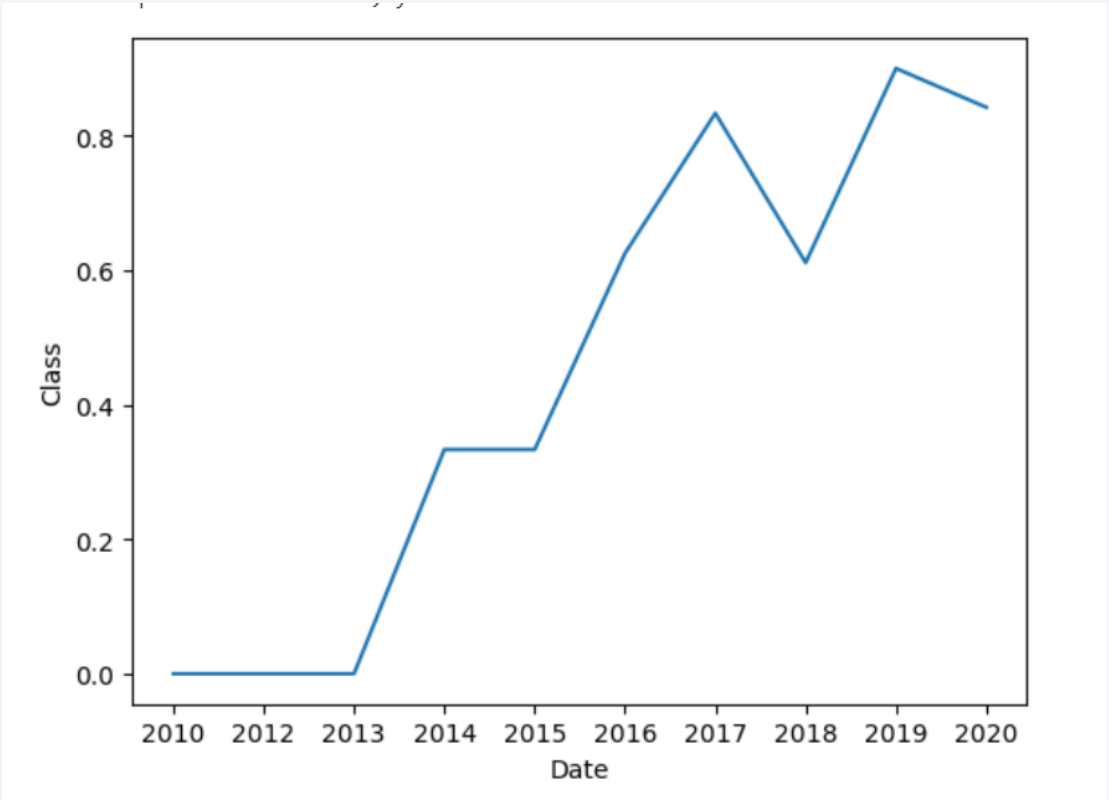
- A scatter point of payload vs. orbit type
- Payload mass above 800 kg is placed only in ISS, PO and VLEO



Launch Success Yearly Trend

- Line chart of yearly average success rate

- The success rate increases with time. Nearly 90% success rate in 2020



All Launch Site Names

- Names of unique launch sites
- There are four distinct launch sites

```
[15]: %sql select DISTINCT Launch_Site from SPACEXTABLE
 * sqlite:///my_data1.db
Done.
```

[15]:

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`

```
[16]: %sql select * from SPACEXTABLE where Launch_Site like 'CCA%' LIMIT 5
 * sqlite:///my_data1.db
Done.
```

[16]:

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
- The total payload mass carried is 99980 kg

```
[31]: %sql select sum(PAYLOAD_MASS__KG_) as total_payload_mass from SPACEXTABLE where customer like 'NASA%'
 * sqlite:///my_data1.db
Done.
[31]: total_payload_mass
------------------------
                 99980
```

Average Payload Mass by F9 v1.1

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

Task 4

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

```
[18]: %sql select avg(PAYLOAD_MASS__KG_) as total_payload_mass from SPACEXTABLE where Booster_Version like 'F9 v1.1'
```

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

[18]:

total_payload_mass
2928.4

First Successful Ground Landing Date

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

```
[41]: %sql select min(Date) as first_landing_date from SPACEXTABLE where Landing_Outcome like 'Success (ground pad)'
```

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

[41]:

first_landing_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

```
[43]: %sql select Booster_Version from SPACEXTABLE WHERE Landing_Outcome LIKE 'Success (drone ship)' AND PAYLOAD_MASS__KG_>4000 AND
```

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

[43]:

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes
- 98 successful missions and 1 failure missions

Task 7

List the total number of successful and failure mission outcomes

```
[54]: %sql select count(Mission_Outcome) from SPACEXTABLE where Mission_Outcome = 'Success'
```

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

[54]:

count(Mission_Outcome)
98

```
[19]: %sql select count(Mission_Outcome) from SPACEXTABLE where Mission_Outcome = 'Failure'
```

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

[19]:

count(Mission_Outcome)
0

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

```
[58]: %sql select Booster_version from SPACEXTABLE WHERE  PAYLOAD_MASS__KG_=(SELECT MAX(PAYLOAD_MASS__KG_)FROM SPACEXTABLE);
 * sqlite:///my_data1.db
Done.
```

[58]:

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

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.

[82]:
```
%%sql SELECT substr(Date, 6, 2) AS month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE
WHERE Landing_Outcome ='Failure (drone ship)' AND substr(Date, 0, 5) = '2015';
```

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

[82]:

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

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

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.

[83]:
```
%%sql SELECT Landing_Outcome, COUNT(*) AS outcome_count FROM SPACEXTABLE
WHERE Date >= '2010-06-04' AND Date <= '2017-03-20' GROUP BY Landing_Outcome
ORDER BY outcome_count DESC;
```

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

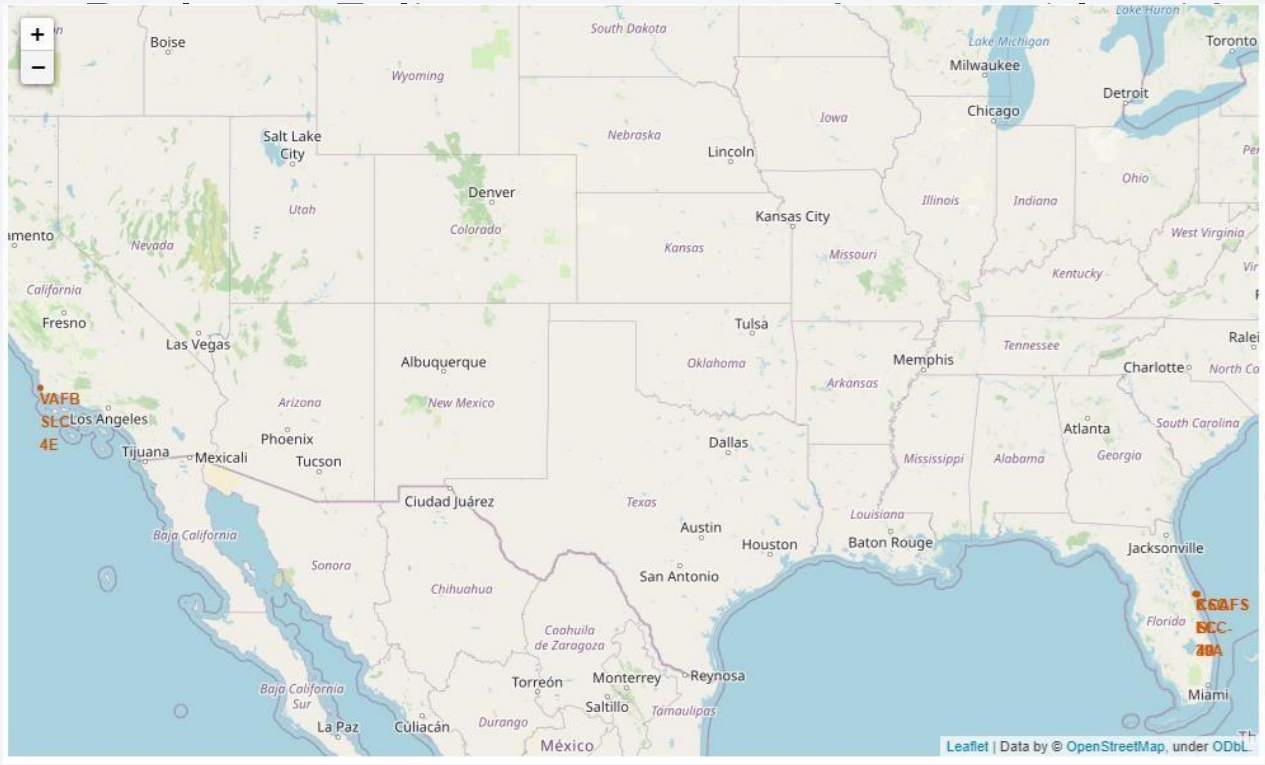
[83]:

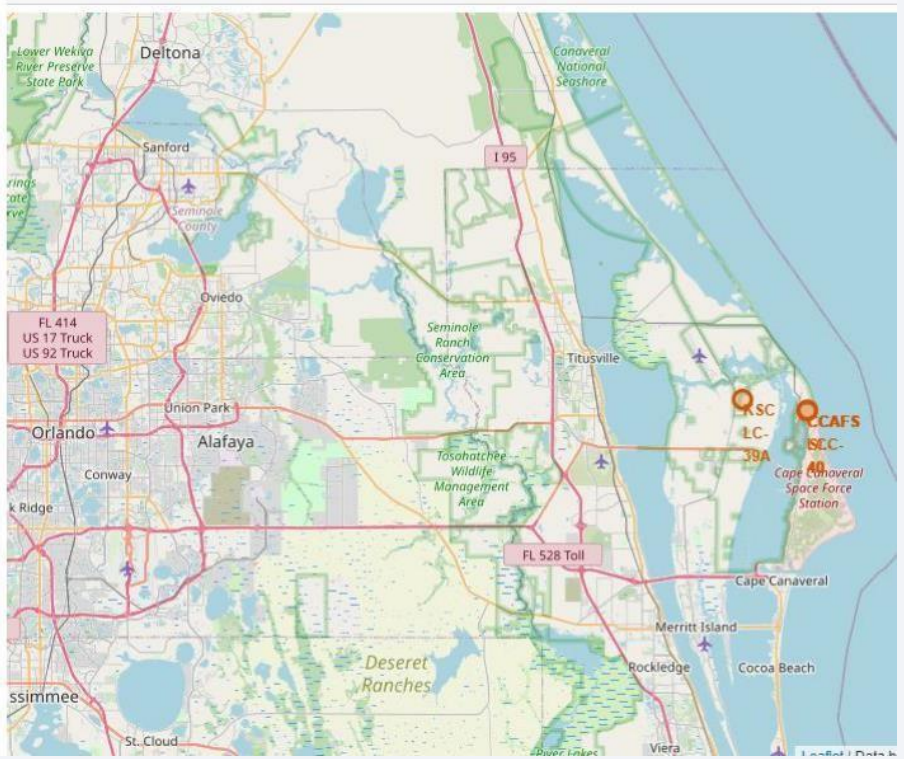
Landing_Outcome	outcome_count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

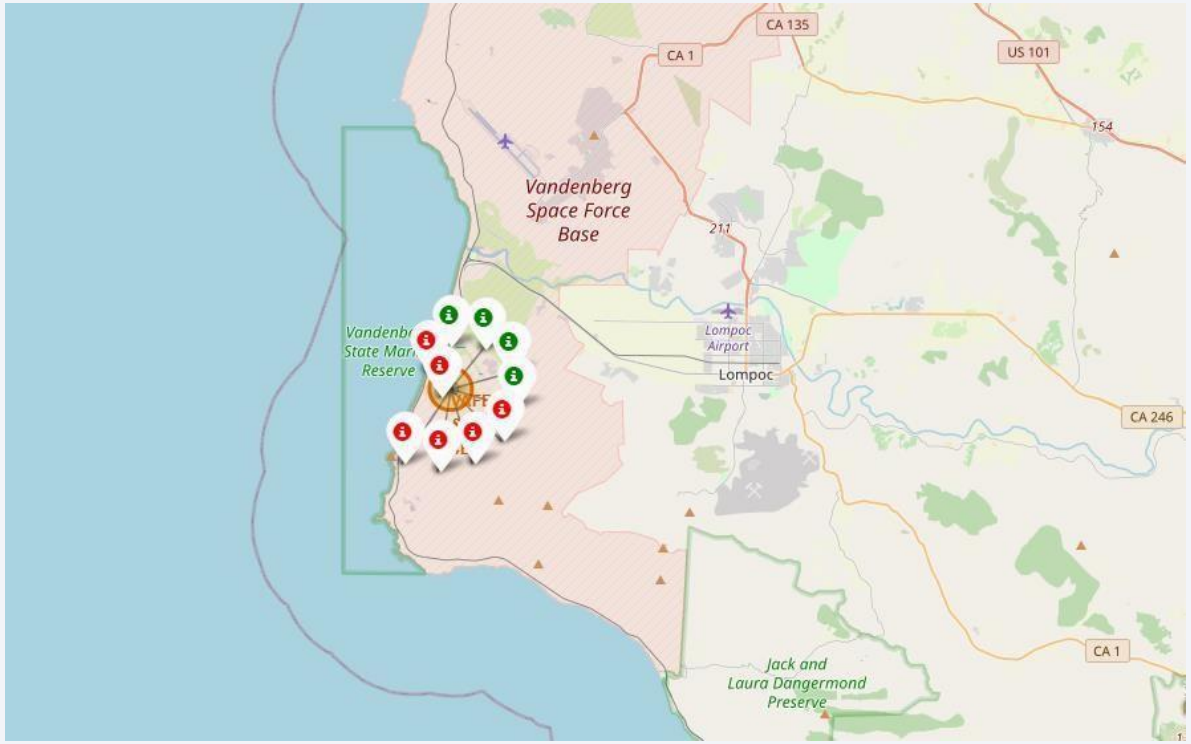
Launch sites





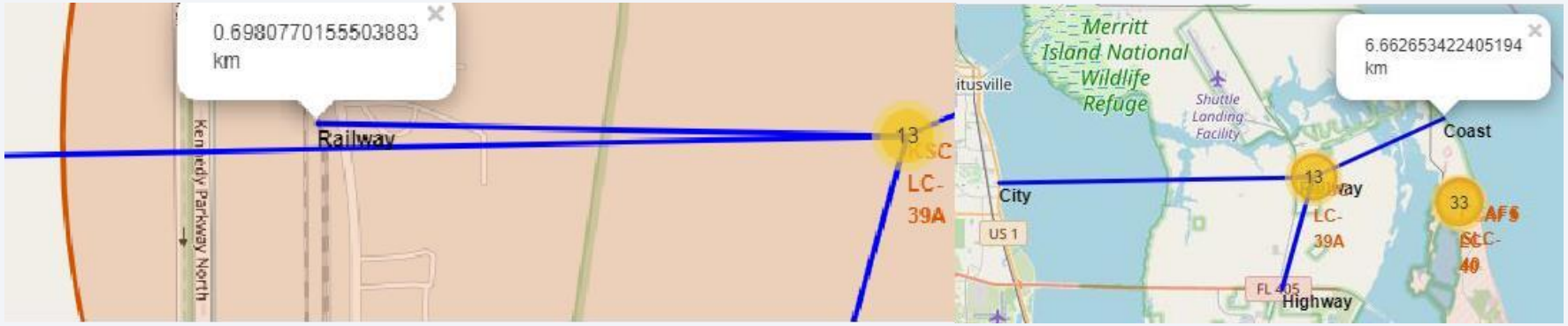
All launch sites are close to the ocean

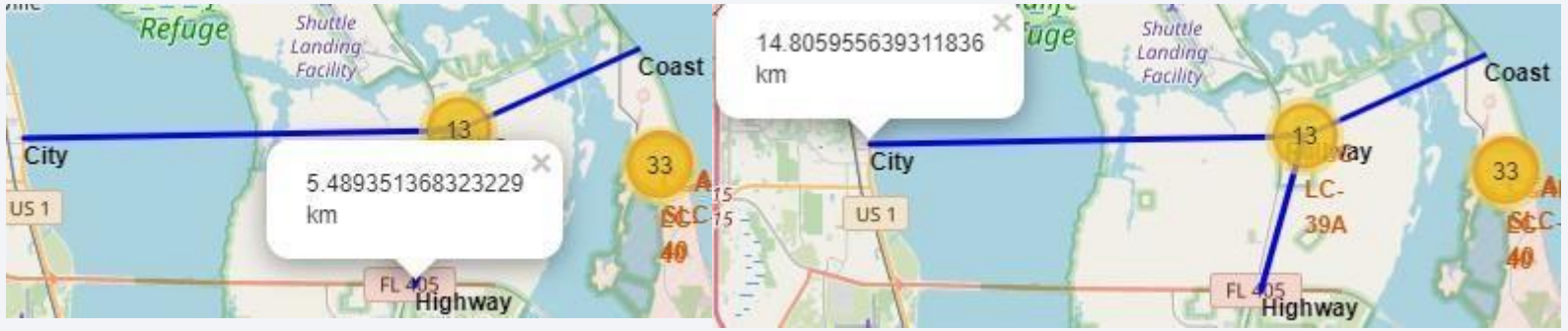
Color coded markers



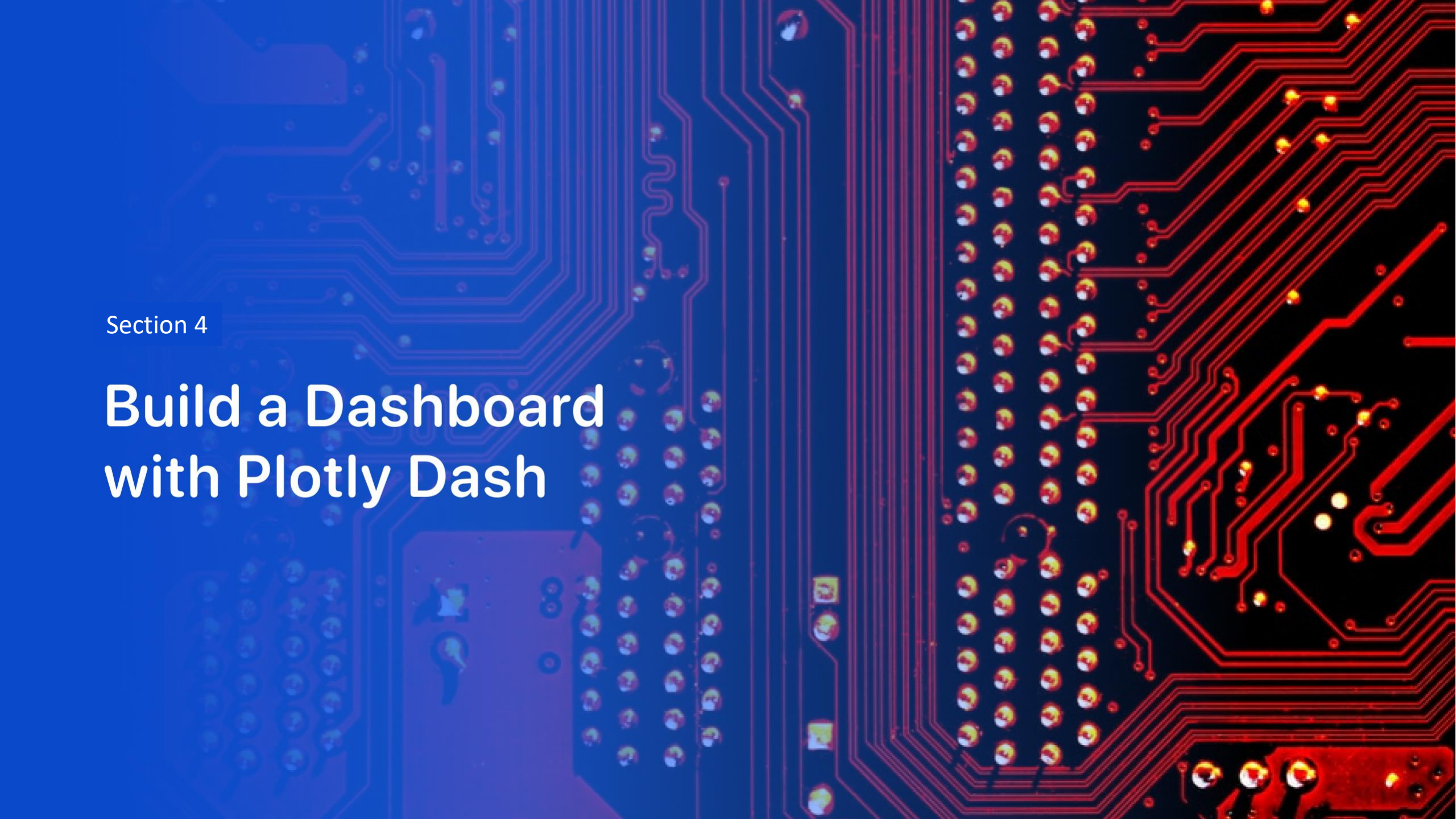
Successful launches are marked by green and unsuccessful ones are marked by red

Safe distance from railway lines

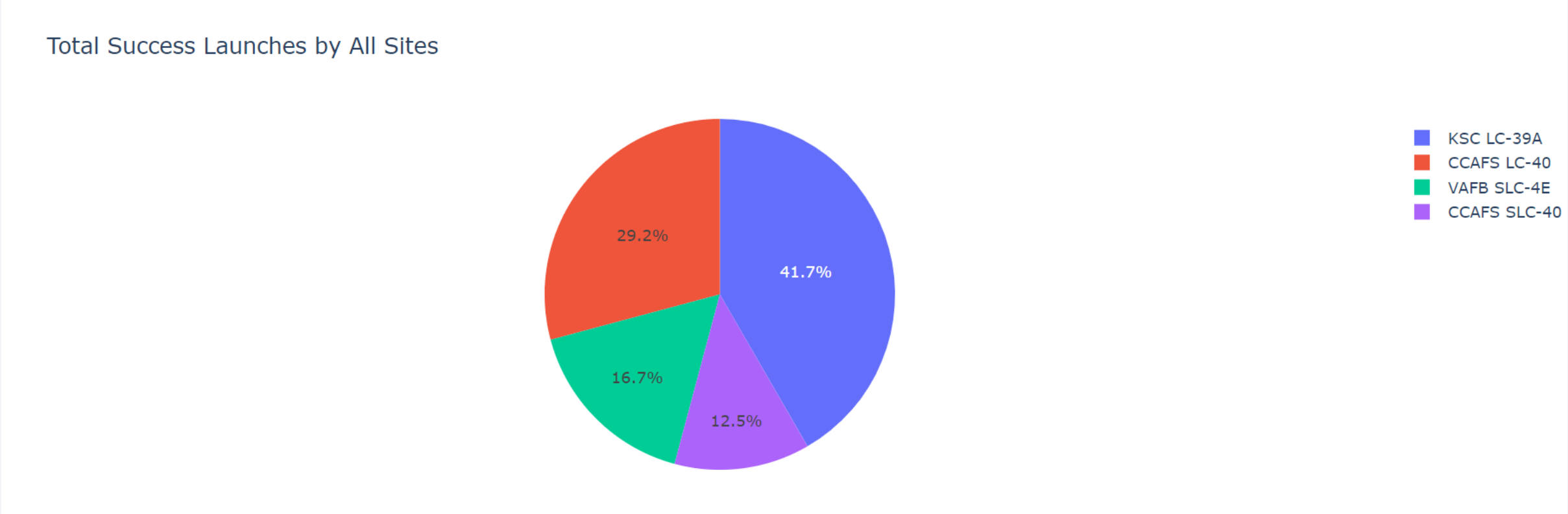




The launch sites are at safe distance from railway lines, city and highway.
They are close to the coast

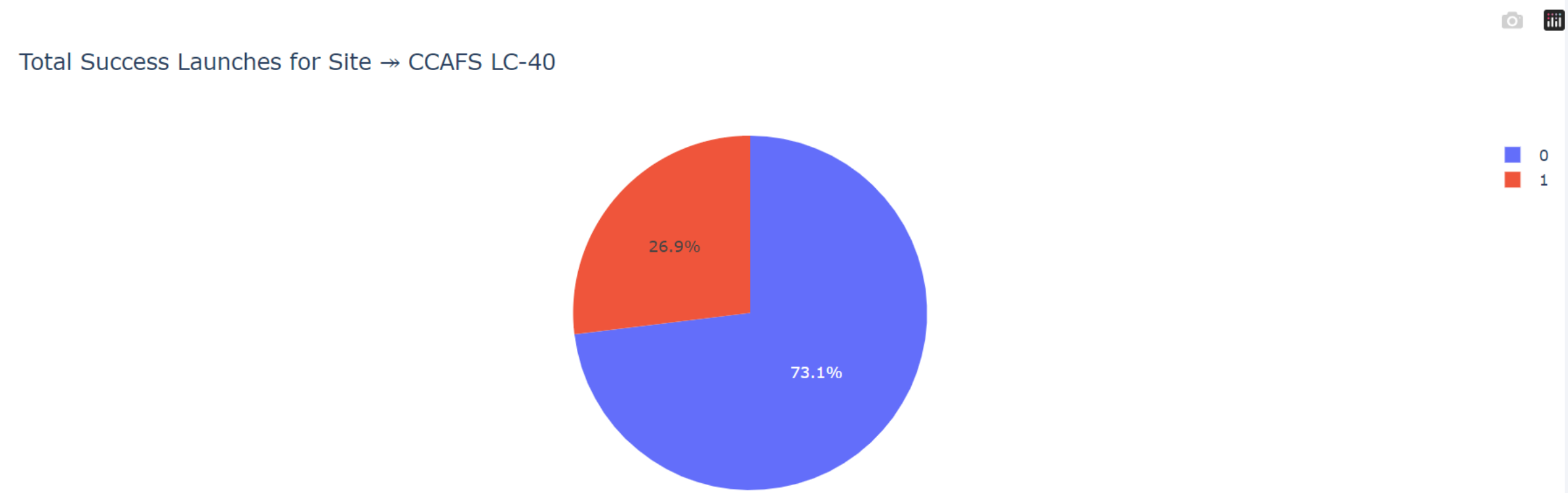

Section 4
Build a Dashboard with Plotly Dash

Total success launch by all sites



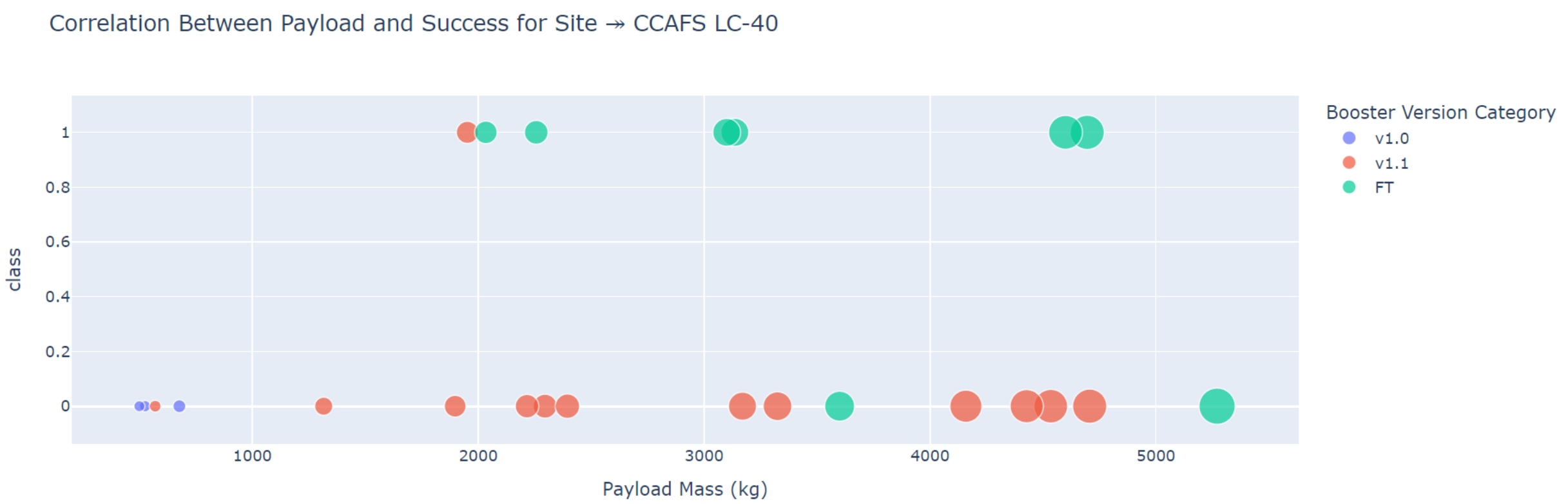
CCAFS LC -40 has maximum success

Highest Launch success ratio



It is for the site CCAFS - 40

Payload Vs launch Outcome



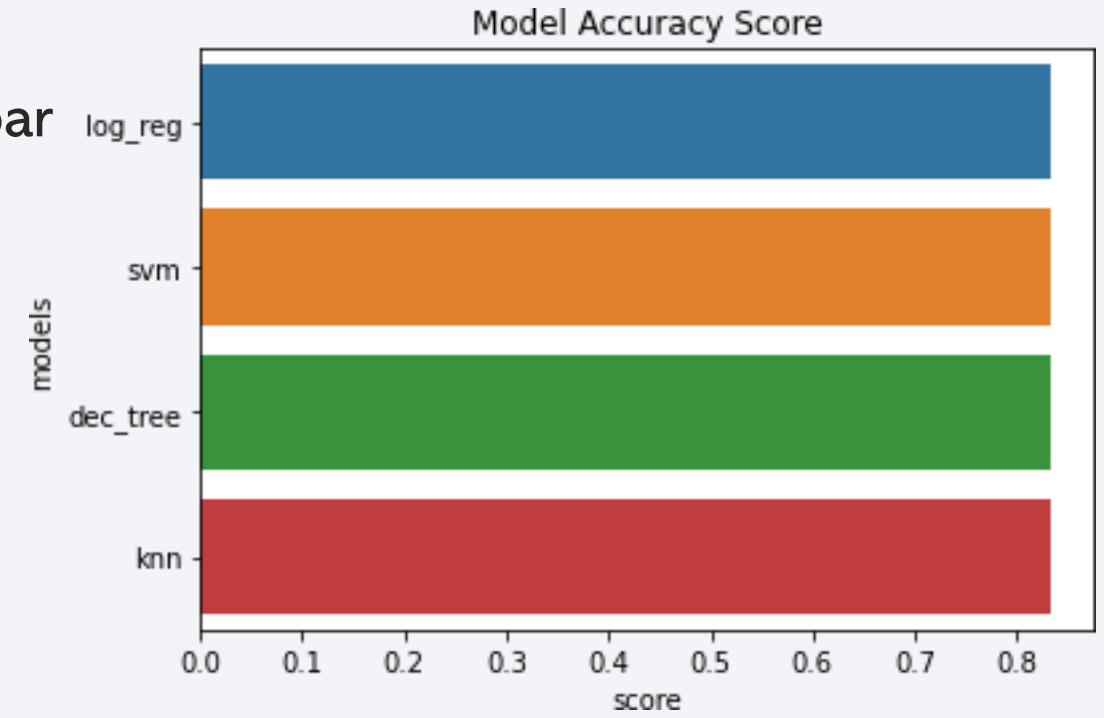
Section 5

Predictive Analysis (Classification)

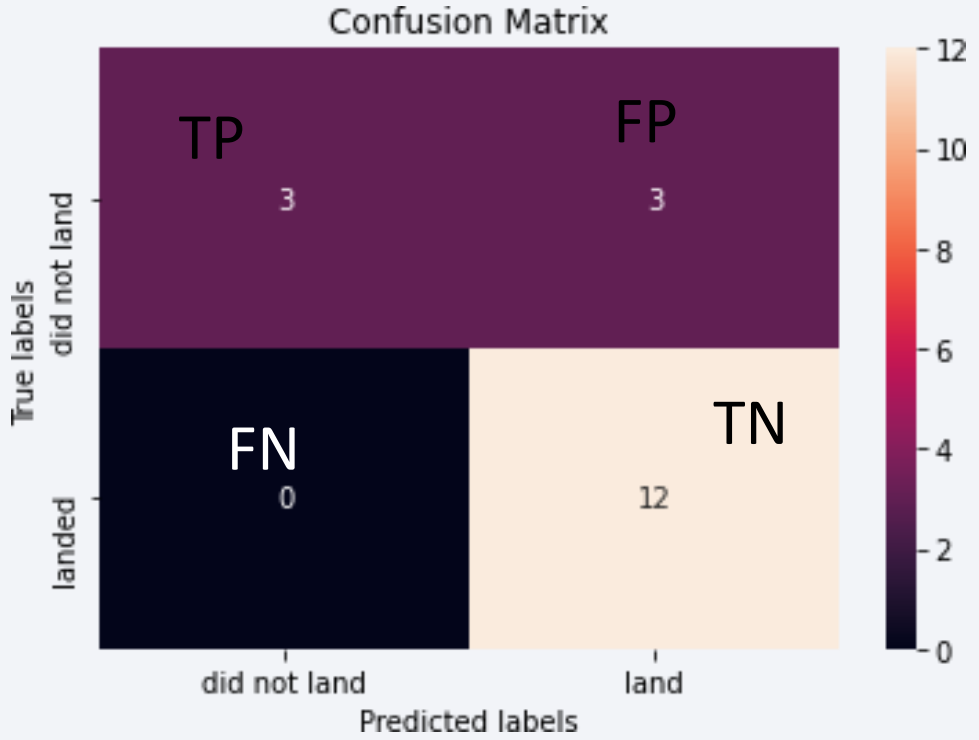
Classification Accuracy

- Figure shows the built model accuracy for all built classification models, in a bar chart

- All models have same classification accuracy



Confusion Matrix



15 outcomes are correctly predicted

Conclusions

- Machine learning model is developed for SpaceY to compete with SpaceX by predicting the successful landing of Stage 1, potentially saving around $100 million USD.
- Data was gathered from the SpaceX public API and by web scraping the SpaceX Wikipedia page, then labeled and stored in a DB2 SQL database.
- A dashboard was created for visualization purposes, helping to present the data effectively.
- We developed a machine learning model that achieved an accuracy rate of 83%.
- This model allows SpaceY's Allon Mask to predict with high accuracy whether a launch will successfully land Stage 1, guiding decisions on whether to proceed with the launch. Collecting more data could further improve the model's accuracy.

Appendix

- Github repository containing all codes;

https://github.com/Sruthimohan05/IBM_data_science_capstonee/tree/main

Thank you!