



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

CARLOS FIGUEROA RIFO
22/12/2024



Outline

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

Executive Summary

- Summary of methodologies
 - Data Collection through API
 - Data Collection with Web Scraping
 - Data Wrangling
 - Exploratory Data Analysis with SQL
 - Exploratory Data Analysis with Data Visualization
 - Interactive Visual Analytics with Folium
 - Machine Learning Prediction
- Summary of all results
 - Exploratory Data Analysis result
 - Interactive analytics in screenshots
 - Predictive Analytics result

Introduction

- Space X's cost advantage relies on reusable first-stage rockets; this project aims to develop a machine learning pipeline to predict first-stage landing success and explore factors influencing it.
- Space Y, a new entrant, plans to leverage public data and multivariable analysis to guide strategic decisions in booster design, launch site selection, and cost estimation.
- Insights will help Space Y develop cost-efficient, reliable, and sustainable rockets to compete in the space exploration market.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
 - Data was collected using web scrapping techniques and data APIs from publicly available sites obtaining a large enough data set with multiple variables.
- Perform data wrangling
 - Collected data was enriched by creating a landing outcome label based on outcome data after summarizing and analyzing 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
 - Several models were developed using cross-validation and validated with test data to estimate accuracy and prevent overfitting. The data was normalized, split into training and test sets, and evaluated across four classification models with various parameter combinations.

Data Collection – SpaceX API

Step 1:

- Import libraries. Below we will define a series of helper functions that will help us use the API to extract information using identification numbers in the launch data. Let's start requesting rocket launch data from SpaceX API with the following URL:
`static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'`

Step 2:

- Request and parse the SpaceX launch data using the GET request
A lot of the data are IDs. For example the rocket column has no information about the rocket just an identification number.

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns rocket, payloads, launchpad, and cores. The data from these requests will be stored in lists and will be used to create a new dataframe.

Step 3:

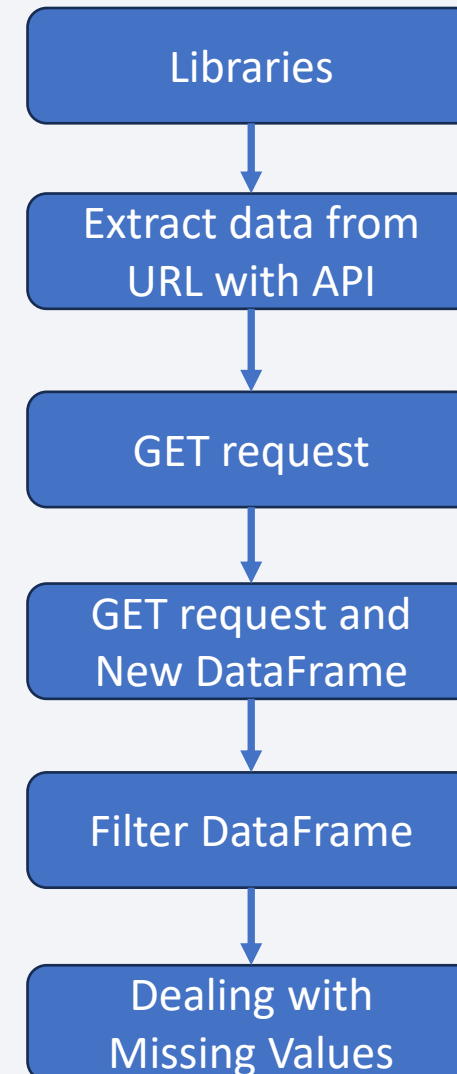
- Filter the dataframe to only include `Falcon 9` launches

Step 4

- Dealing with Missing Values

GitHub URL:

https://github.com/carlosfig1/UNAB_IBM_Data_Science/blob/cOd846b9ad38f342fab0c016d62ef71a40a6b47a/Applied%20Data%20Science%20Capstone/jupyter-labs-spacex-data-collection-api-v2.ipynb



Data Collection - Scraping

Web scrap Falcon 9 launch records with `BeautifulSoup`:

- Extract a Falcon 9 launch records HTML table from Wikipedia.
- Parse the table and convert it into a Pandas data frame.

Step 1:

Request the Falcon9 Launch Wiki page from its URL.

Step 2

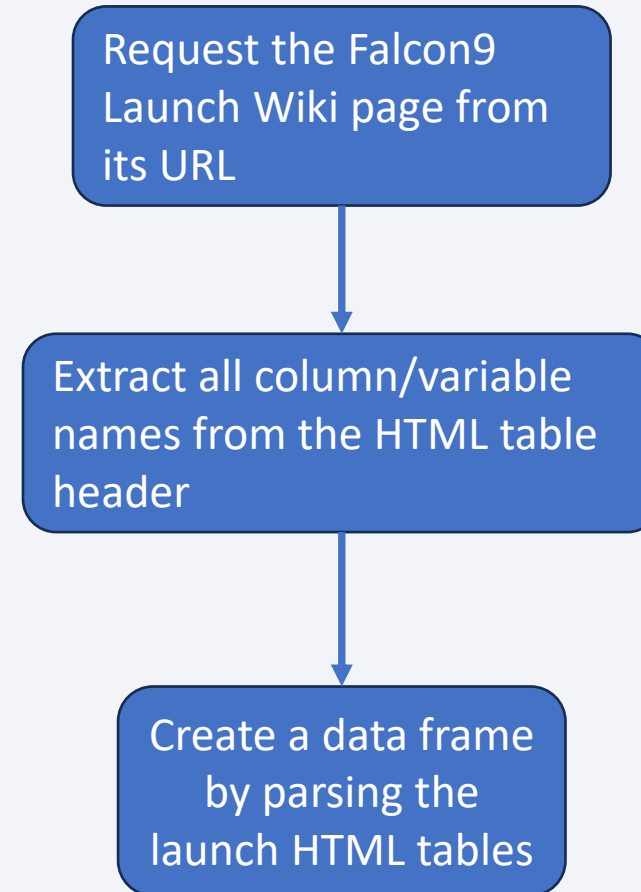
:Extract all column/variable names from the HTML table header.

Step3:

Create a data frame by parsing the launch HTML tables.

GitHub URL:

https://github.com/carlosfig1/UNAB_IBM_Data_Science/blob/c0d846b9ad38f342fab0c016d62ef71a40a6b47a/Applied%20Data%20Science%20Capstone/jupyter-labs-webscraping.ipynb



Data Wrangling

- Perform exploratory Data Analysis and determine Training Labels
 - Exploratory Data Analysis.
 - Determine Training Labels.

Step 1: Calculate the number of launches on each site.

Step 2: Calculate the number and occurrence of each orbit.

Step 3: Calculate the number and occurrence of mission outcome of the orbits.

Step 4: Create a landing outcome label from Outcome column.

GitHub URL:

https://github.com/carlosfig1/UNAB_IBM_Data_Science/blob/c0d846b9ad38f342fab0c016d62ef71a40a6b47a/Applied%20Data%20Science%20Capstone/labs-jupyter-spacex-Data%20wrangling-v2.ipynb

EDA with Data Visualization

- The following Charts were developed as we want to explore if there are any correlations between the different variables
 - ✓ “Relationship between Flight Number and Launch Site”
 - ✓ “Relationship between Payload and Launch Site”
 - ✓ “Relationship between success rate of each orbit type”
 - ✓ “Relationship between Flight Number and Orbit type”
 - ✓ “Relationship between Payload and Orbit type”
 - ✓ “Launch success yearly trend”

GitHub URL:

https://github.com/carlosfig1/UNAB_IBM_Data_Science/blob/c0d846b9ad38f342fab0c016d62ef71a40a6b47a/Applied%20Data%20Science%20Capstone/jupyter-labs-eda-dataviz-v2.ipynb

EDA with SQL

The following SQL queries were performed

- Names of the unique launch sites in the space mission: **select distinct Launch_Site from SPACEXTBL limit 100**
- Top 5 launch sites whose name begins with the string 'CCA': **select * from SPACEXTBL where Launch_Site LIKE 'CCA%' limit 5**
- Total payload mass carried by boosters launched by NASA (CRS): **select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer = 'NASA (CRS)'**
- Average payload mass carried by booster version F9 v1.1: **select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version = 'F9 v1.1'**
- Date when the first successful landing outcome in ground pad was achieved: **select min(Date) from SPACEXTBL where Landing_Outcome not like '%Failure%'**
- Names of the boosters which have success in drone ship and have payload mass between 4000 and 6000 kg: **select distinct Booster_Version from SPACEXTBL where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000 limit 1**
- Total number of successful and failure mission outcomes: **select distinct Mission_Outcome from SPACEXTBL**
- Names of the booster versions which have carried the maximum payload mass: **select Booster_Version, max(PAYLOAD_MASS__KG_) from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL) limit 1**
- Failed landing outcomes in droneship, their booster versions, and launch site names for in year 2015: **select substr(Date,6,2), Booster_Version, Launch_Site from SPACEXTBL where Landing_Outcome = 'Failure (drone ship)' and substr(Date,0,5)='2015'**
- Rank of the count of landing outcomes: **select Landing_Outcome, count(Landing_Outcome) as n from SPACEXTBL where Date between '2010-06-04' and '2017-03-20' group by Landing_Outcome order by n desc**

GitHub URL:

https://github.com/carlosfig1/UNAB_IBM_Data_Science/blob/c0d846b9ad38f342fab0c016d62ef71a40a6b47a/Applied%20Data%20Science%20Capstone/jupyter-labs-eda-sql-coursera_sqlite.ipynb

Build an Interactive Map with Folium

- Markers, circles, lines and marker clusters were used with Folium Maps
- Markers indicate points like launch sites;
- Circles indicate highlighted areas around specific coordinates, like NASA Johnson Space Center;
- Marker clusters indicates groups of events in each coordinate, like launches in a launch site; and
- Lines are used to indicate distances between two coordinates.

GitHub URL:

https://github.com/carlosfig1/UNAB_IBM_Data_Science/blob/c0d846b9ad38f342fab0c016d62ef71a40a6b47a/Applied%20Data%20Science%20Capstone/lab-jupyter-launch-site-location-v2.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
 - ✓ dropdown box was used to select the Site of interest. You can select any of the different Launch Sites or All if you want to analyze global information.
 - ✓ We plotted pie charts showing the total launches by a certain sites
 - ✓ We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

GitHub URL:

https://github.com/carlosfig1/UNAB_IBM_Data_Science/blob/c0d846b9ad38f342fab0c016d62ef71a40a6b47a/Applied%20Data%20Science%20Capstone/spacex_dash_app.py

Predictive Analysis (Classification)

- We implemented four classification models: Logistic Regression, Support Vector Machines, Decision Trees, and K-Neighbors. Data was split into 80% training and 20% testing sets. Hyperparameters were optimized using GridSearchCV with 10-fold cross-validation on the training data. Accuracy scores were calculated for both training and testing sets, and a confusion matrix was analyzed on the test data to ensure generalization. Using numpy and pandas, we loaded, transformed, and split the data. Feature engineering and algorithm tuning improved performance, and the best-performing classification model was identified.

URL Cloud Environment:

<https://github.com/carlosfig1/UNAB IBM Data Science/blob/c0d846b9ad38f342fab0c016d62ef71a40a6b47a/Applied%20Data%20Science%20Capstone/SpaceX Machine%20Learning%20Prediction Part 5 cloud.ipynb>

URL Locally with other parameters:

<https://github.com/carlosfig1/UNAB IBM Data Science/blob/c0d846b9ad38f342fab0c016d62ef71a40a6b47a/Applied%20Data%20Science%20Capstone/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb>

Results

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

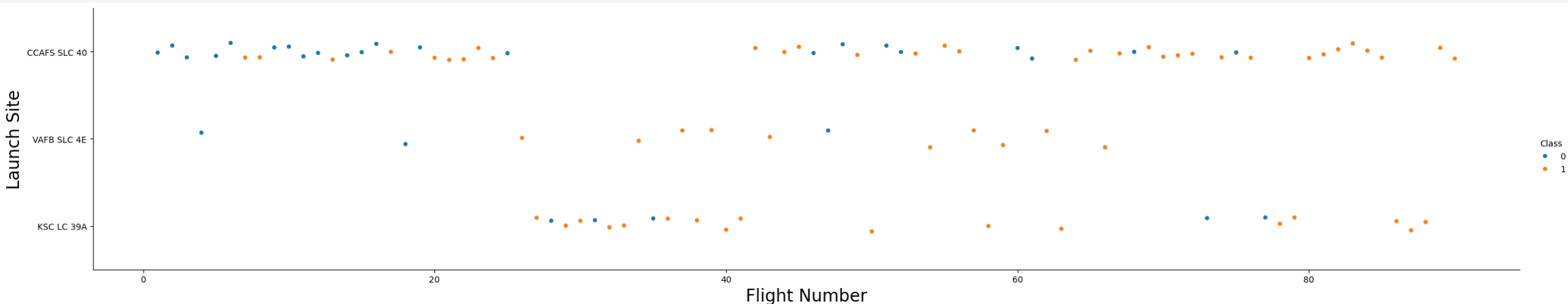
The background of the slide is an abstract composition. It features a dark blue base color. Overlaid on this are numerous diagonal streaks in shades of red and cyan. A faint, light blue grid pattern is also visible, particularly in the lower half of the image. The overall effect is dynamic and technological.

Section 2

Insights drawn from EDA

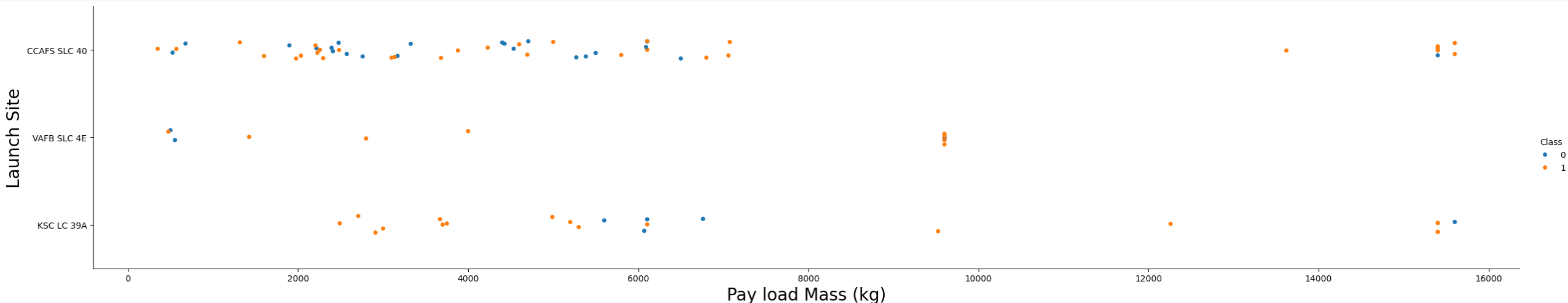
Flight Number vs. Launch Site

- CCAFS SLC-40: Used for the first 20+ launches with poor results, paused between flights 25-40, then resumed from flight 40 onward with improved outcomes.
- VAFB SLC 4E: Used less frequently compared to other sites.
- KSC LC-39A: Likely unavailable until flight 25, heavily used until flight 41, then used sporadically but with good results overall.



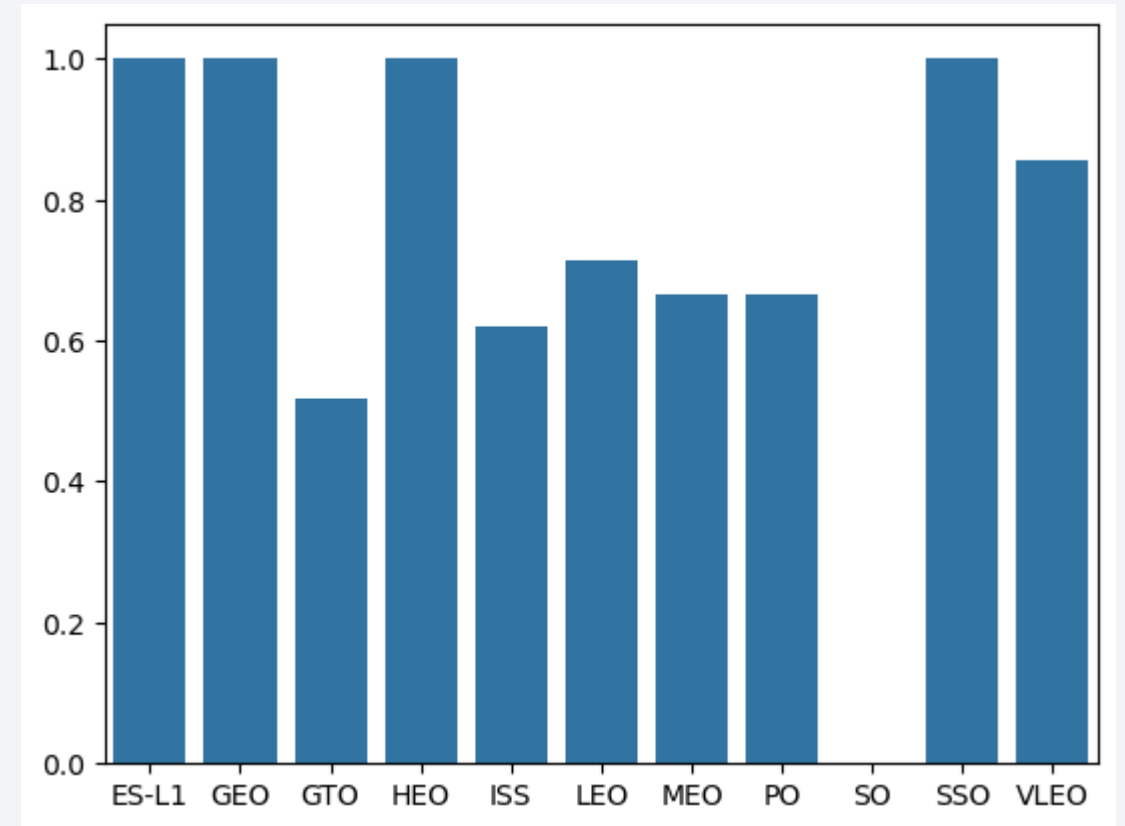
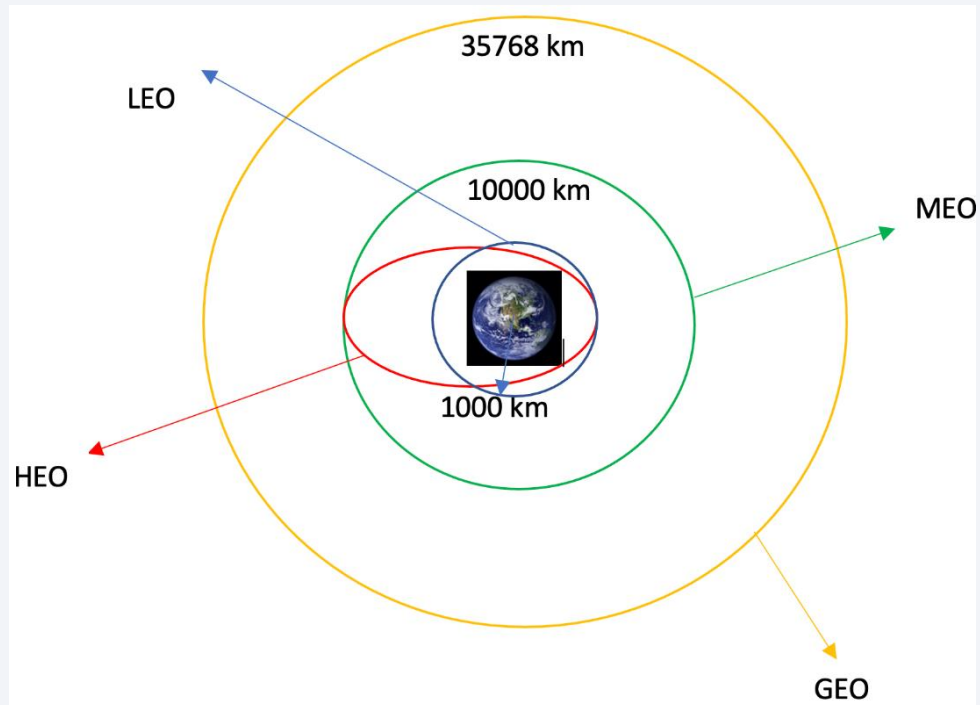
Payload vs. Launch Site

- VAFB SLC 4E: Used only for payloads under 10,000 kg.
- CCAFS SLC-40: Handles low and large payloads with modest success.
- KSC LC-39A: Used for intermediate and large payloads with high success rates.



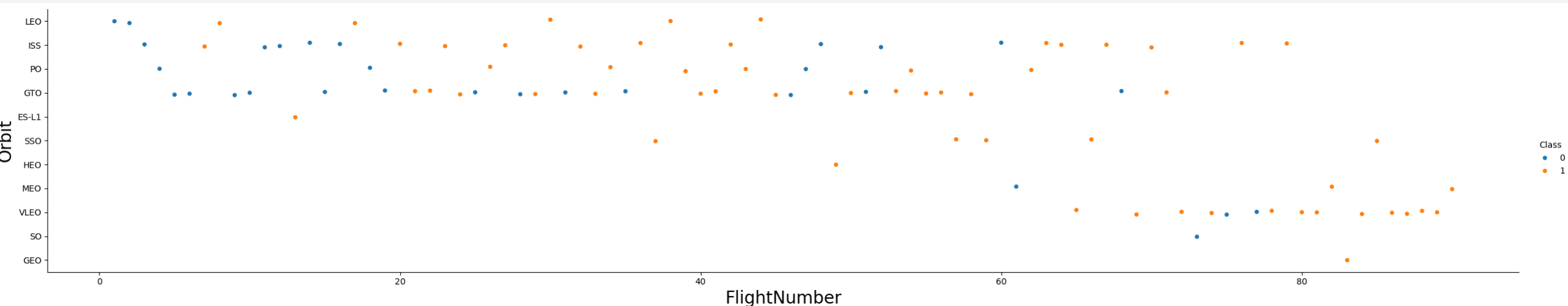
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- The orbits with the lowest success rates are SO and GTO



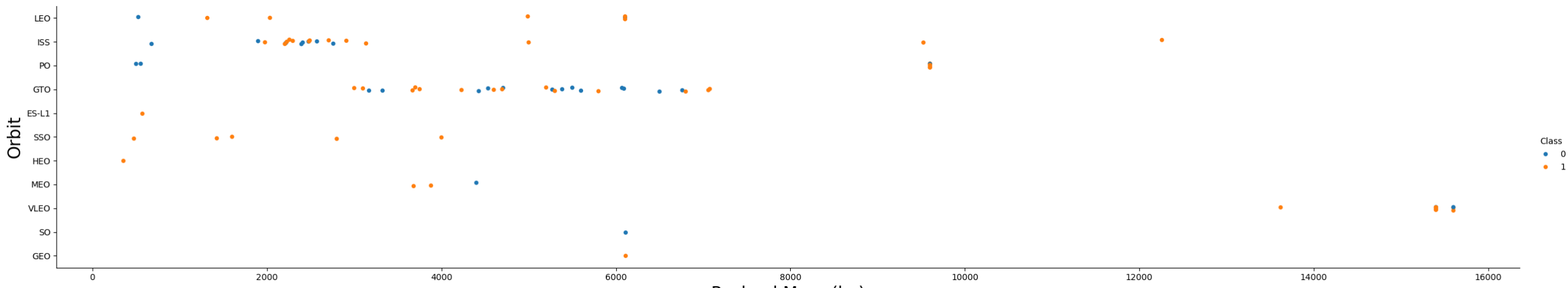
Flight Number vs. Orbit Type

- LEO Orbits: Success correlates with flight number, while GTO shows no such relationship.
- Orbit Accessibility: Certain orbits, like Very Low Earth Orbits, became viable only after flight 62 due to technological or strategic factors.
- Early Flights: Focused on GTO and LEO, likely for communication services (e.g., Starlink) and ISS servicing in partnership with NAS



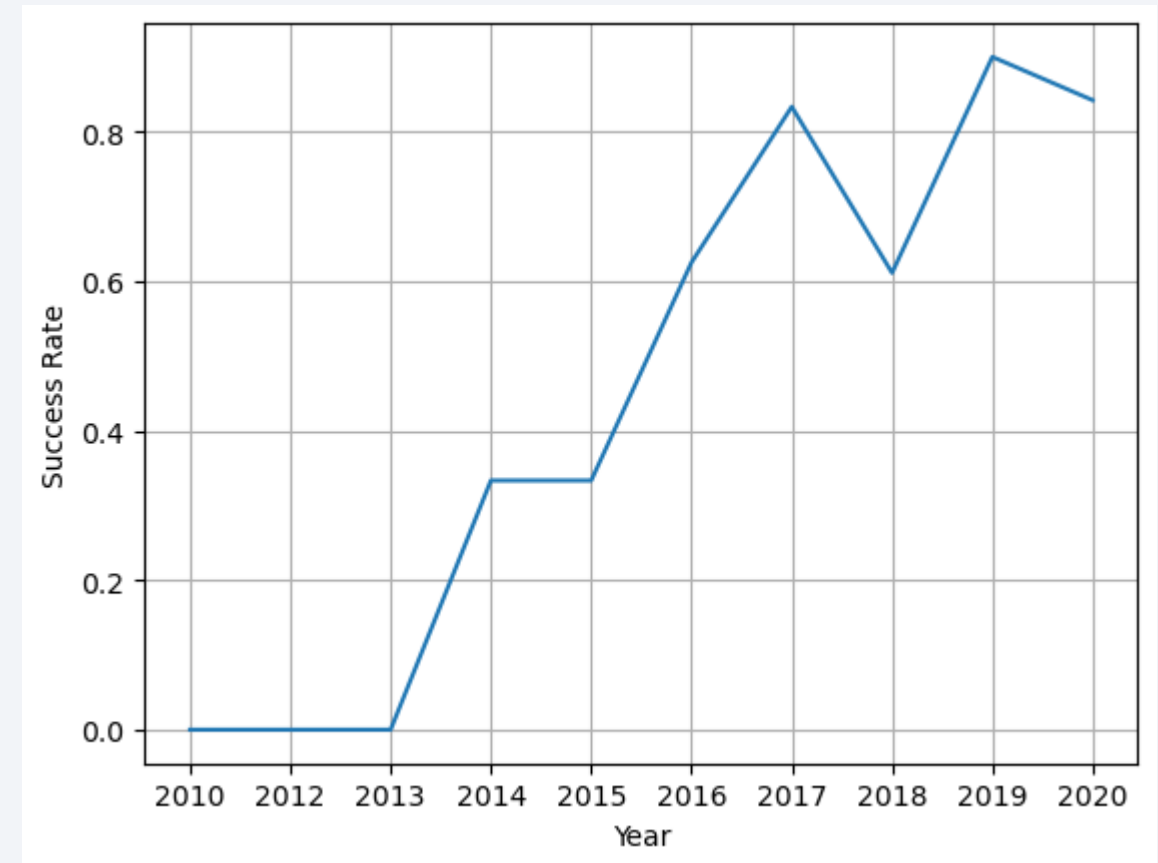
Payload vs. Orbit Type

- Large Payloads: Rarely launched, typically to specific orbits.
- Trend Analysis: A Payload Mass vs. Flight Number/Date chart could clarify trends, such as earlier booster mass limitations.
- Heavy Payloads: Successful landings are more frequent in PO, LEO, and ISS orbits.



Launch Success Yearly Trend

- 2013: Success rate was zero.
- 2013-2017: Steady increase, stabilizing in 2014.
- 2018: Drop, followed by recovery in 2019.
- 2020: Continued rise in success rate.



All Launch Site Names

- We use distinct to get unique values of Launch_Site from SPACEXTBL table and limit to display only 100 rows

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

%sql select distinct Launch_Site from SPACEXTBL limit 100
[14] ✓ 0.0s
... * 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'

- We use LIKE to get the rows where Launch_Site starts with 'CCA' from SPACEXTBL table and limit to display only 5 rows

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

```
%sql select * from SPACEXTBL where Launch_Site LIKE 'CCA%' limit 5
```

✓ 0.0s

* [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

- We use sum to get the total PAYLOAD_MASS__KG_ where Customer is 'NASA (CRS)' from SPACEXTBL table

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

%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer ='NASA (CRS)'
✓ 0.0s

* sqlite:///my\_data1.db
Done.

sum(PAYLOAD_MASS__KG_)
45596
```

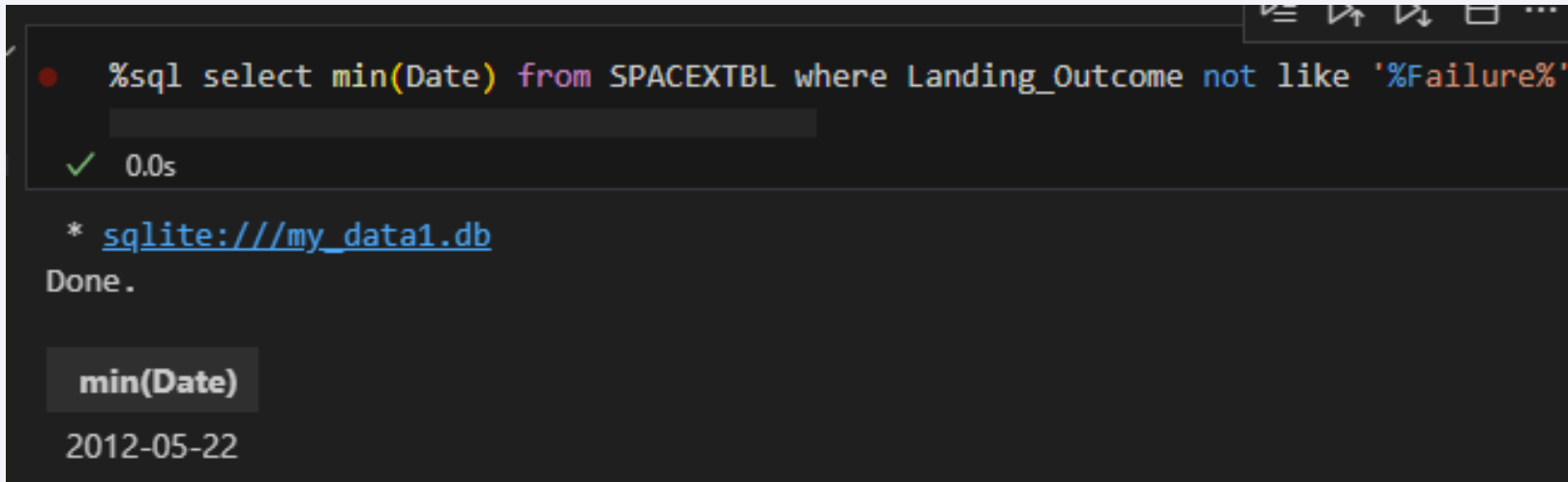
Average Payload Mass by F9 v1.1

- We use avg to get the average PAYLOAD_MASS_KG_ where Booster_Version is 'F9 v1.1' from SPACEXTBL table

```
▶ %sql select avg(PAYLOAD_MASS_KG_) from SPACEXTBL where Booster_Version = 'F9 v1.1'
[18] ✓ 0.0s Python
... * sqlite:///my\_data1.db
... Done.
... avg(PAYLOAD_MASS_KG_)
... 2928.4
```

First Successful Ground Landing Date

- We use min to get the earliest Date where Landing_Outcome does not contain 'Failure' from SPACEXTBL table



```
%sql select min(Date) from SPACEXTBL where Landing_Outcome not like '%Failure%'
✓ 0.0s

* sqlite:///my\_data1.db
Done.

min(Date)
2012-05-22
```

The screenshot shows a terminal window with a dark background. At the top, there's a prompt '%sql' followed by the SQL query 'select min(Date) from SPACEXTBL where Landing_Outcome not like '%Failure%'. Below the query, there's a green checkmark and '0.0s' indicating successful execution. Then, the terminal shows the connection path '* [sqlite:///my_data1.db](#)' and 'Done.'. Finally, the result is displayed in a table with one column 'min(Date)' and one row containing the date '2012-05-22'.

Successful Drone Ship Landing with Payload between 4000 and 6000

- We use distinct to get unique values of Booster_Version where Landing_Outcome is 'Success (drone ship)', PAYLOAD_MASS__KG_ is greater than 4000 and less than 6000 from SPACEXTBL table and limit to display only 10 rows

```
%sql select distinct Booster_Version from SPACEXTBL where Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000 limit 10
✓ 0.0s
* sqlite:///my\_data1.db
Done.
```

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

Total Number of Successful and Failure Mission Outcomes

- We use count to get the count of Mission_Outcome from SPACEXTBL table

```
%sql select Mission_Outcome,count(*) from SPACEXTBL group by Mission_Outcome
```

✓ 0.0s

* [sqlite:///my_data1.db](#)

Done.

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

Boosters Carried Maximum Payload

- We use max to get the maximum PAYLOAD_MASS__KG_ where PAYLOAD_MASS__KG_ is equal to the maximum PAYLOAD_MASS__KG_ from SPACEXTBL table and limit to display only 1 row

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

```
%sql select Booster_Version,max(PAYLOAD_MASS__KG_) from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL ) limit 1
```

✓ 0.0s

* [sqlite:///my_data1.db](#)

Done.

Booster_Version	max(PAYLOAD_MASS__KG_)
F9 B5 B1048.4	15600

2015 Launch Records

- We use substr to get the month part of Date, Booster_Version and Launch_Site where Landing_Outcome is 'Failure (drone ship)' and year part of Date is '2015' from SPACEXTBL table

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: SQLite 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.

```
%sql select distinct Landing_Outcome from SPACEXTBL where Landing_Outcome like '%drone%'
```

✓ 0.0s

* [sqlite:///my_data1.db](#)

Done.

Landing_Outcome

Failure (drone ship)

Precluded (drone ship)

Success (drone ship)

```
%sql select substr(Date, 6,2),Booster_Version, Launch_Site from SPACEXTBL where Landing_Outcome = 'Failure (drone ship)' and substr(Date,0,5)='2015'
```

✓ 0.0s

* [sqlite:///my_data1.db](#)

Done.

substr(Date, 6,2)	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

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

- We use count to get the count of Landing_Outcome where Date is between '2010-06-04' and '2017-03-20' from SPACEXTBL table and group by Landing_Outcome and order by n 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.

```
%sql select Landing_Outcome,count(Landing_Outcome) as n from SPACEXTBL where Date between '2010-06-04' and '2017-03-20' group by Landing_Outcome order by n desc
```

✓ 0.0s

* [sqlite:///my_data1.db](#)

Done.

Landing_Outcome	n
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

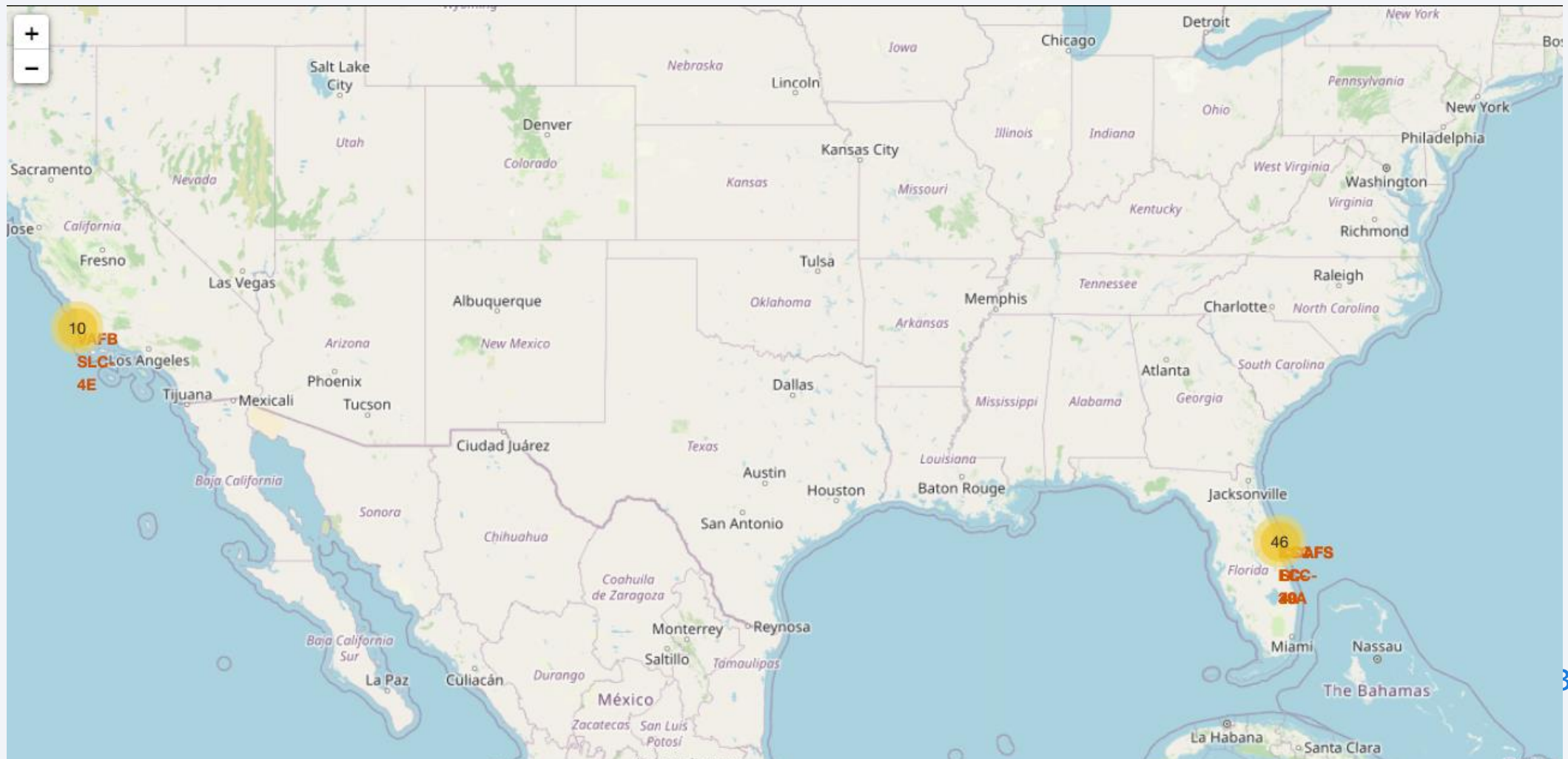
A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The background is a deep blue gradient.

Section 3

Launch Sites Proximities Analysis

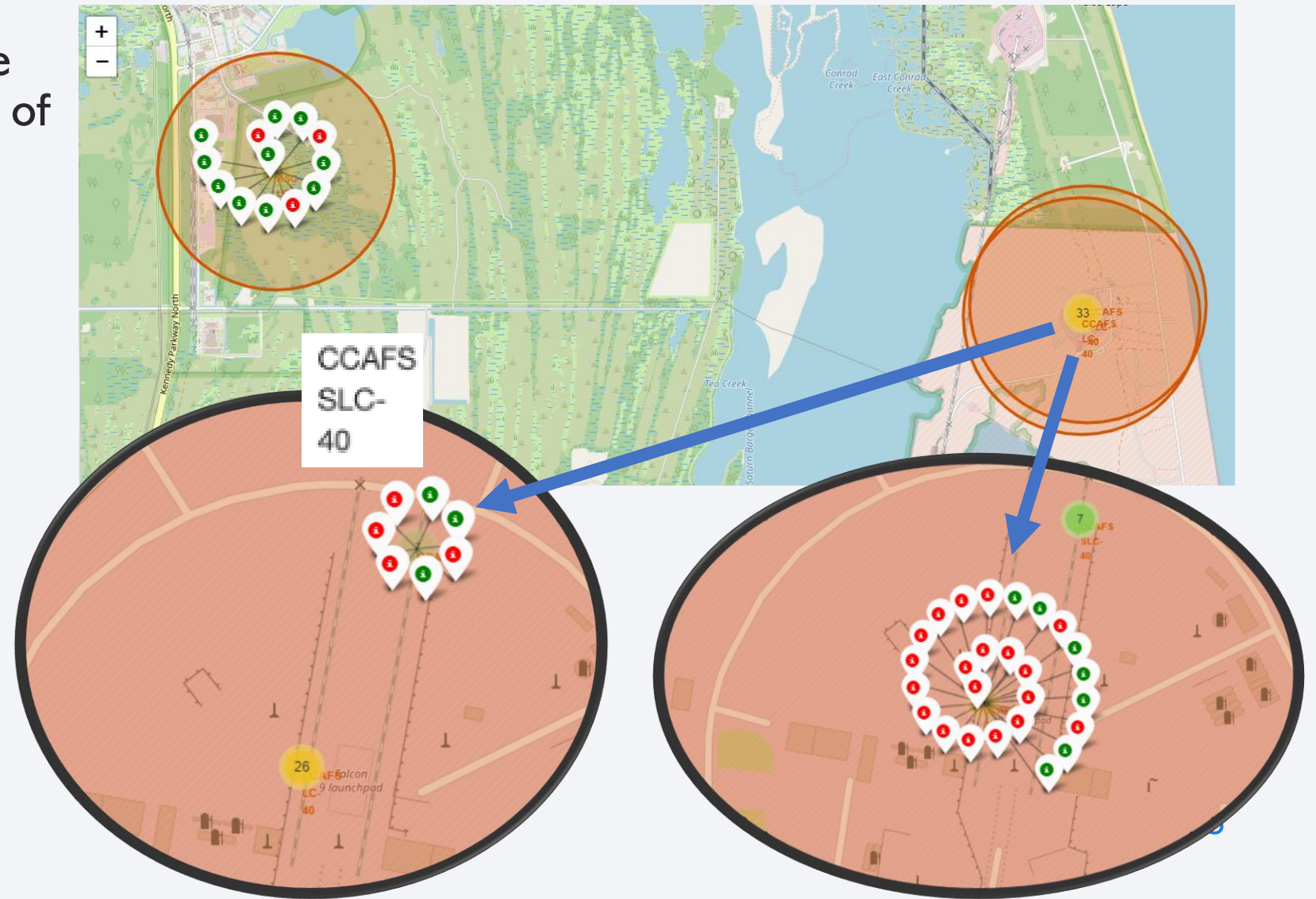
launch sites on a map

- Launch Sites are shown and are located near coastline in Florida and California



Mark the success/failed launches for each site on the map

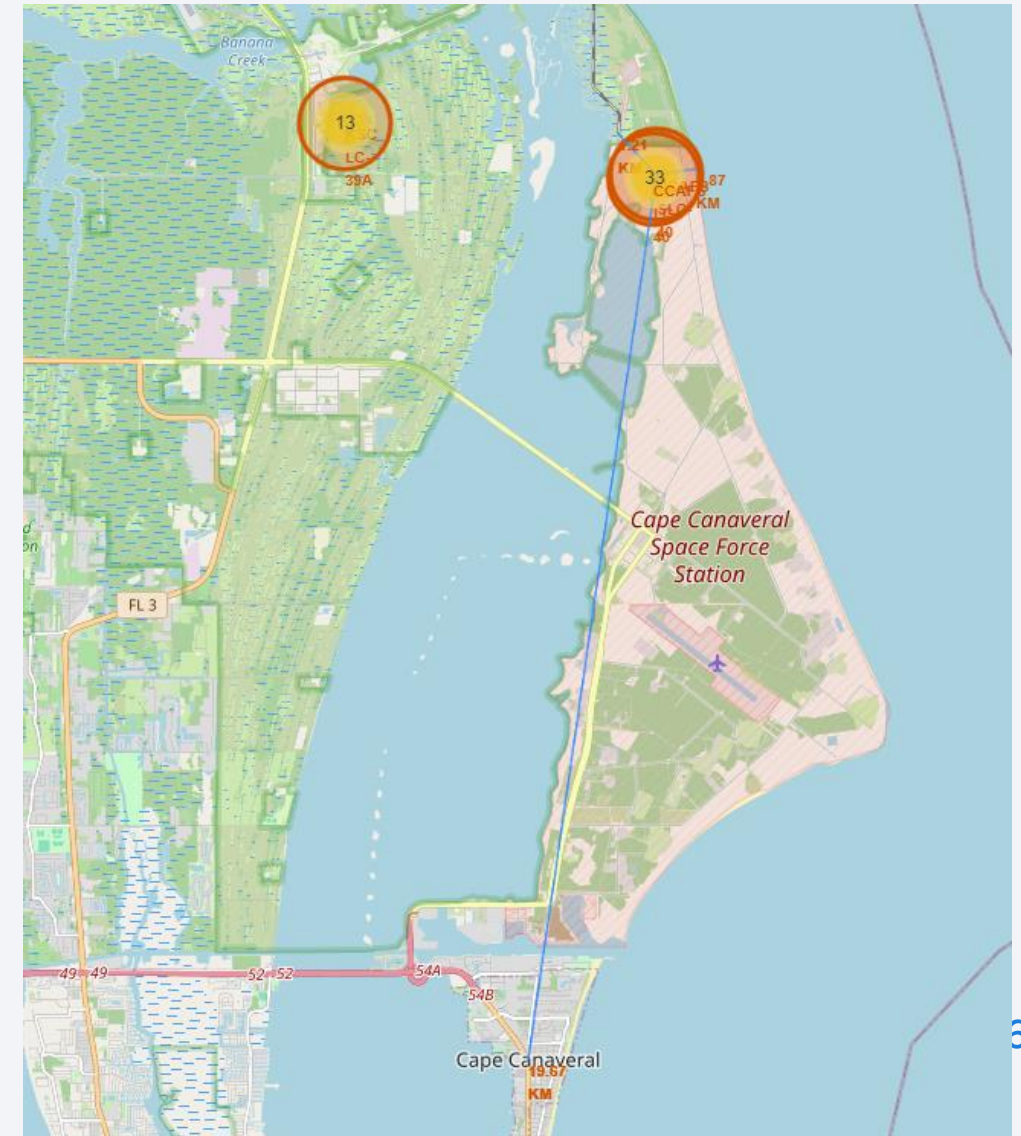
- Two nearby sites in Florida are shown with their total number of successful (green) and failed (red) launches
- KSC LC-39A has the best rate
- CCAFS SLC-40 has the worst rate



Distances between a launch site to its proximities

- Distance from site to railway and highway: 1.21 km
- Distance from site to Coastline: 0.87 km
- Distance from site to nearest city (Cape Canaveral) 19.67 km

The launch sites are very close to railway , highway and the Coastline. It is necessary take a safety considerations.

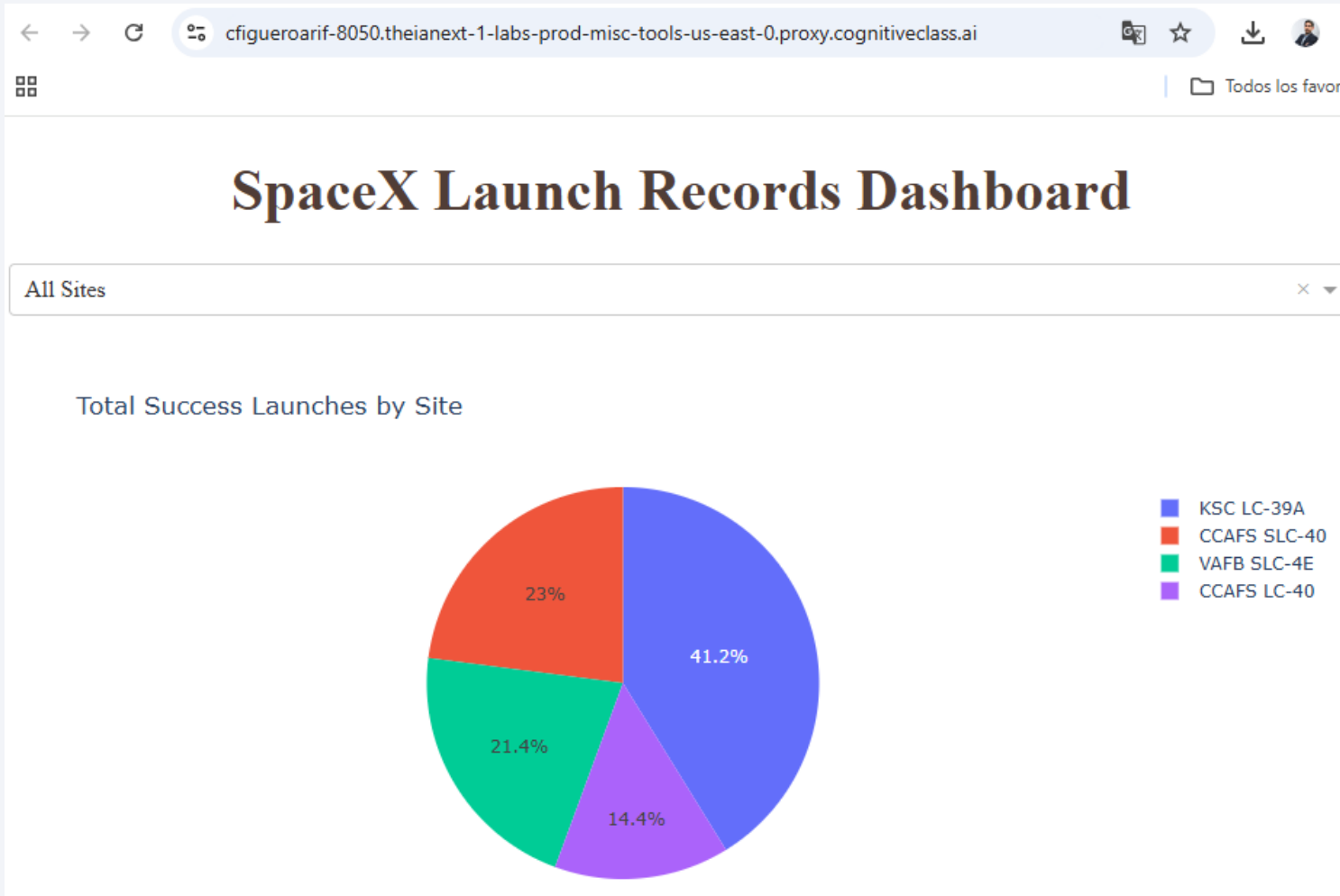




Section 4

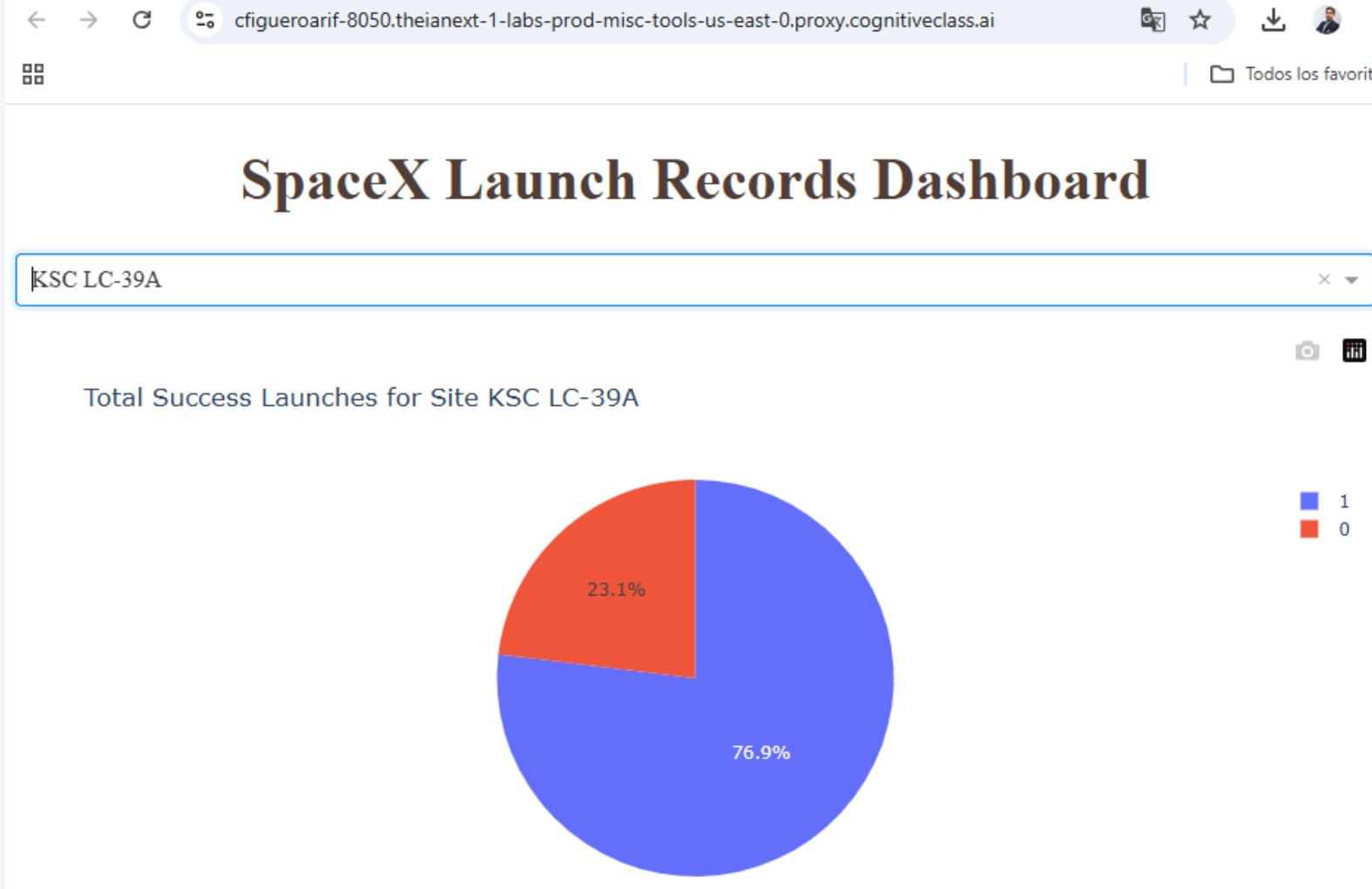
Build a Dashboard with Plotly Dash

Total Success Launches by Site



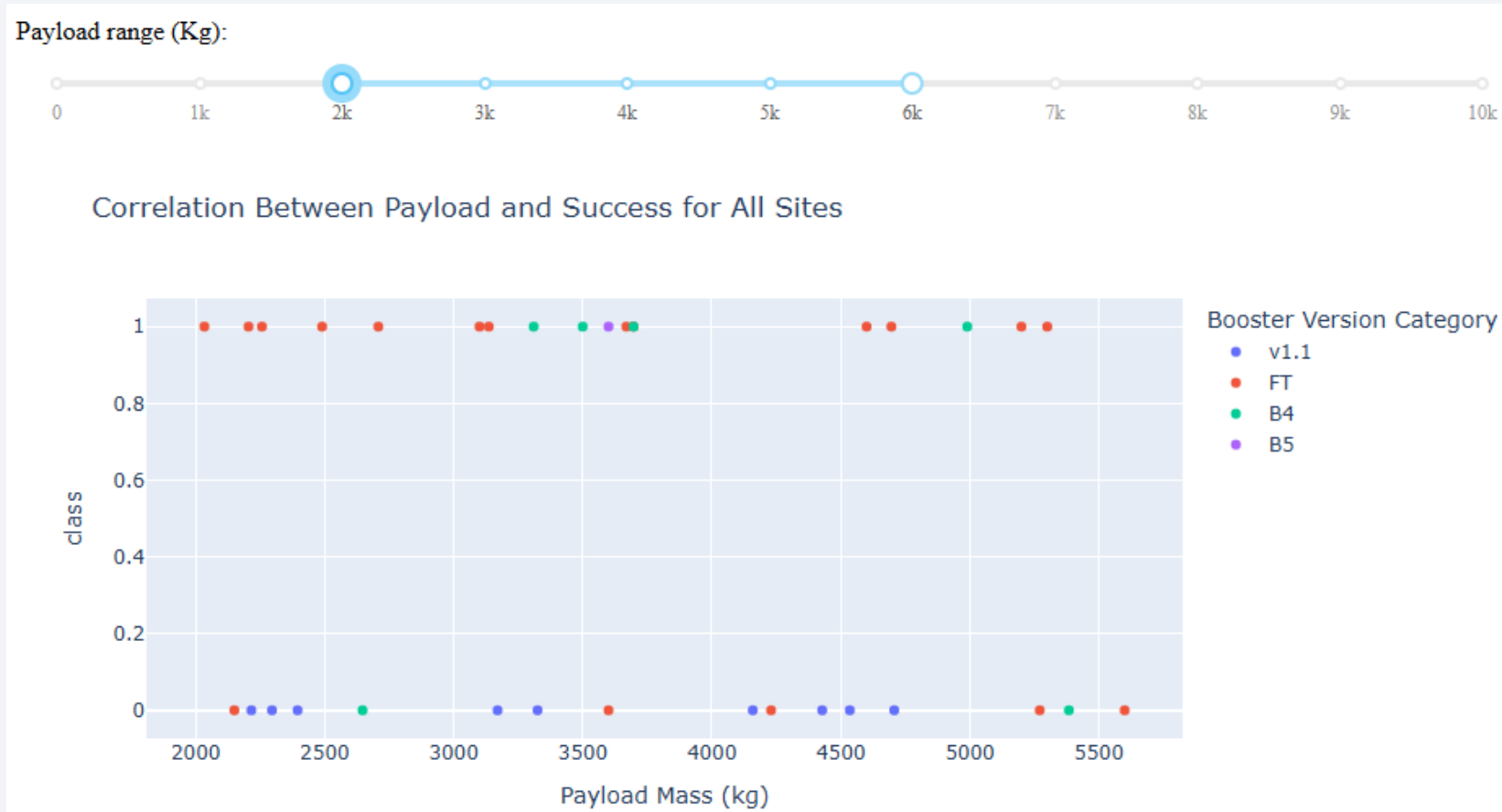
- The launch site with highest launch success ratio is KSC LC-39A with a 76.9% launch successful and the worst launch success ratio is CCAFS LC-40

Piechart for the launch site with highest launch success ratio



- The launch site with highest launch success ratio is KSC LC-39A with a 76.9% launch successful.

<Dashboard Screenshot 3>



- Payloads between 2000 and 5000 kg, Booster category FT was very successful.

Section 5

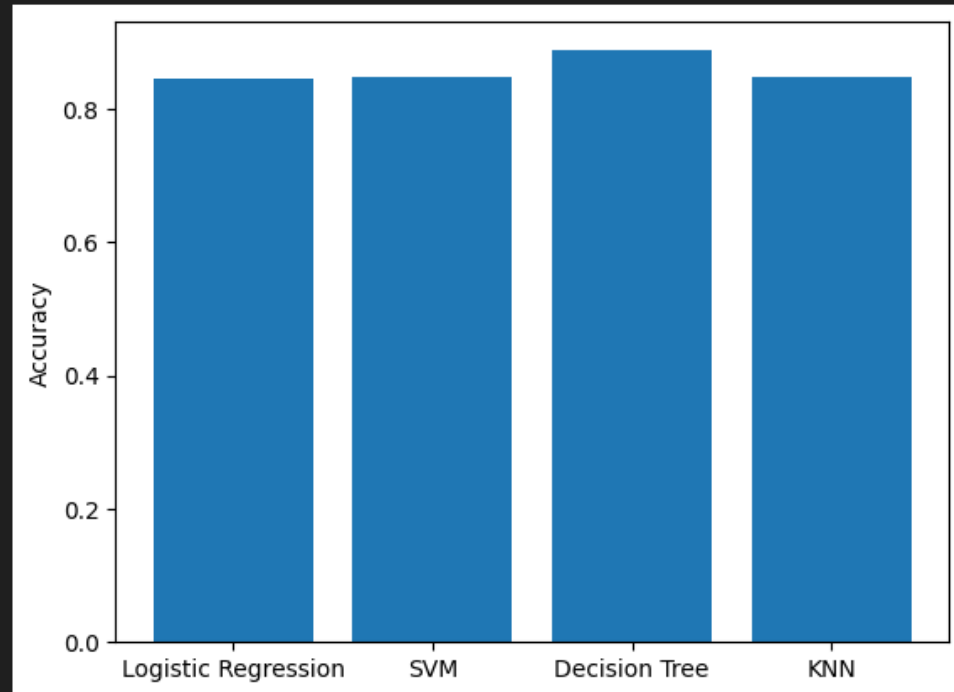
Predictive Analysis (Classification)

Classification Accuracy

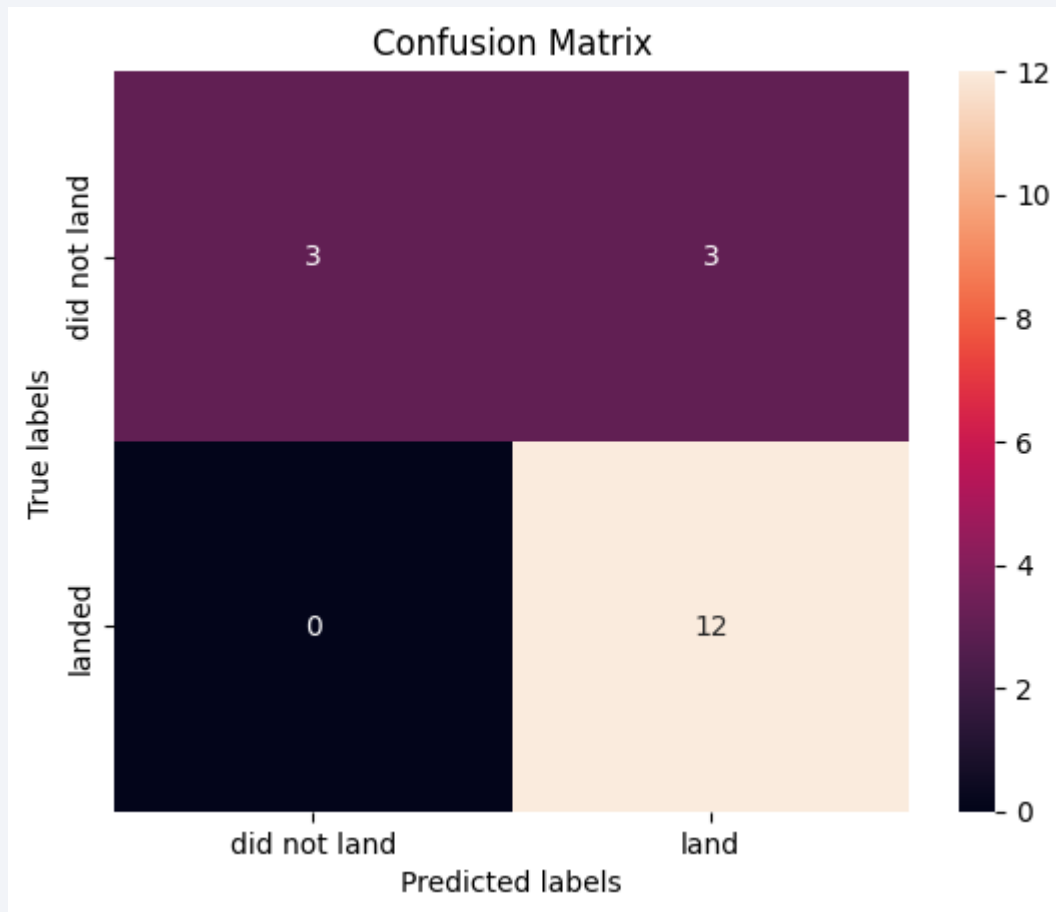
- The decision tree classifier is the model with the highest classification accuracy with 88.75% to Train Data. The accuracy to Test Data is 83.3%

```
accuracy = [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_]
models = ['Logistic Regression', 'SVM', 'Decision Tree', 'KNN']
plt.bar(models, accuracy)
plt.ylabel('Accuracy')
plt.show()
```

✓ 0.1s



Confusion Matrix



- **Parameter Explanation**

- 1. **True Positives (TP):**

- 1. The rocket landed, and the model correctly predicted it landed.
 - 2. **TP = 12.**

- 2. **False Negatives (FN):**

- 1. The rocket landed, but the model predicted it did not land.
 - 2. **FN = 0.**

- 3. **False Positives (FP):**

- 1. The rocket did not land, but the model predicted it landed.
 - 2. **FP = 3.**

- 4. **True Negatives (TN):**

- 1. The rocket did not land, and the model correctly predicted it did not land.
 - 2. **TN = 3.**

- **Accuracy (83.3%):** The model is correct in the majority of its predictions.
- **Precision (80%):** When the model predicts a rocket has landed, it is correct 80% of the time.
- **Recall (100%):** The model correctly identifies all actual landings.
- **F1-Score (88.9%):** A good balance between precision and recall.

The model performs well in identifying successful landings but could improve by reducing false positives.

Conclusions

- Success Rate Trends: Strong correlation with flight number, increasing steadily from 2013 to 2020.
- Key Launch Sites: KSC LC-39A had the highest success rate, with strategic site selection being crucial for logistics, safety, and performance.
- Successful Orbits: ES-L1, GEO, HEO, SSO, and VLEO had the highest success rates.
- Booster Performance: FT boosters (payloads $\leq 5000\text{kg}$) showed the best success rates.
- Model Accuracy: The classification model predicts mission outcomes with 83.3% accuracy, though false positives occur.
- Best Algorithm: The decision tree classifier is the most effective for this task.
- Future Steps: Further research and updated models are recommended to improve performance and insights.

Thank you!

