IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nguyễn Thanh Vy

26/07/2024

Presentation Contents

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

Executive Summary

This report evaluates machine learning models for predicting SpaceX's Falcon 9 rocket landings. It includes:

- Data preprocessing, model training, and hyperparameter tuning.
- Analysis of data from APIs and the web using EDA, visualizations, and SQL.
- Integration of insights into an interactive dashboard.
- Identification of the best-performing model, Decision Tree, and evaluation using key metrics.

Introduction

- The commercial space age has arrived with companies like Virgin Galactic, Rocket Lab, Blue Origin, and SpaceX.
- SpaceX has made space travel more affordable through reusable rockets.
- The goal is to develop a model to predict the landing success of SpaceX's Falcon 9 first stage
- Accurate predictions can inform cost estimates and optimize mission planning.

Introduction

Problem Statement:

01

SpaceX's cost-effective Falcon 9 launches rely on the reusability of the first stage.

02

Predicting successful landings is crucial for cost estimation.

Section 1

Methodology

Methodology

Data Collection:	Data Wrangling	Exploratory Data Analysis	Predictive Analysis
Data was sourced from the SpaceX API and additional web scraping.	Initial Data Cleaning and Feature Engineering	Visualization and Interactive Visual Analytics	Classificaion algorithms : Logistic Regression, SVM, Decision Tree and KNN

Data Collection

- **API:** Data was collected from the SpaceX REST API, specifically the endpoint api.spacexdata.com/v4/launches/past. This provided detailed information on past Falcon 9 launches, including rocket versions, payloads, and landing outcomes.

- **Web Scraping:** Additional data was obtained through web scraping of relevant Wiki pages and HTML tables to enrich the dataset with more comprehensive launch records.

Data Collection – SpaceX API

Request and Parse SpaceX Launch Data:

- Use GET request to fetch SpaceX data.
- Decode JSON response using .json() and convert it into a DataFrame using pd.json_normalize().
- Use API for detailed info on rocket, payloads, launchpad, and cores.

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	[]
								Successful first stage		

Data Collection – SpaceX API

Filter for Falcon 9 Launches :

- Remove Falcon 1 launches.
- Filter DataFrame by BoosterVersion for Falcon 9.
- Save to new DataFrame data_falcon9.

Github URL :

https://github.com/ViaThanh/MySubmission-IBM-Python-Test/blob/51b77aec79c9de5e4f0cb482d960758087f6f3fb/module%201-spacex-data-collection-api.ipynb

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
data_falcon9 = launch_df[launch_df['BoosterVersion'] != 'Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9 = data_falcon9.copy()
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9.head()
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Bl
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	

Data Collection - Web Scraping

Request the Page

- Use requests.get() with the provided static_url.
- Create a BeautifulSoup object from the HTML response.

Extract Column Names : from the HTML table header.

Create DataFrame : Parse the launch HTML tables into a DataFrame.

```
# Launch outcome
launch_outcome = list(row[7].strings)[0]
launch_dict['Launch outcome'].append(launch_outcome)

# Booster landing
booster_landing = landing_status(row[8])
launch_dict['Booster landing'].append(booster_landing)
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
# Create DataFrame from launch_dict
df = pd.DataFrame({key: pd.Series(value, dtype='object') for key, value in launch_dict.items()})
```

```
df.head()
```

	Flight No.	Launch site	Payload mass	Orbit	Customer	Launch outcome	Booster landing	Payload	Version, Booster	Date	Time
0	1	CCAFS	0	LEO	SpaceX	Success\n	Failure	Dragon Spacecraft Qualification Unit	F9 v1.0B0003.1	4 June 2010	18:45
1	2	CCAFS	0	LEO	NASA	Success	Failure	Dragon	F9 v1.0B0004.1	8 December 2010	15:43
2	3	CCAFS	525 kg	LEO	NASA	Success	No attempt\n	Dragon	F9 v1.0B0005.1	22 May 2012	07:44
3	4	CCAFS	4,700 kg	LEO	NASA	Success\n	No attempt	SpaceX CRS-1	F9 v1.0B0006.1	8 October 2012	00:35
4	5	CCAFS	4,877 kg	LEO	NASA	Success\n	No attempt\n	SpaceX CRS-2	F9 v1.0B0007.1	1 March 2013	15:10

Github URL :

https://github.com/ViaThanh/MySubmission-IBM-Python-Test/blob/51b77aec79c9de5e4f0cb482d960758087f6f3fb/module%201-spacex-webscraping.ipynb

Data Wrangling

Calculate Number of Launches by Site: Use value_counts() on LaunchSite.

- Counts: CCAFS SLC 40 (55), KSC LC 39A (22), VAFB SLC 4E (13).

Calculate Number of Each Orbit: Use value_counts() on Orbit.

Calculate Mission Outcome Counts: Use value_counts() on Outcome to find landing outcomes.

- Unsuccessful landings: bad_outcomes = {'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}

Create Landing Outcome Label: Generate landing_class list from Outcome, assigning 0 for bad outcomes and 1 otherwise.

Data Wrangling

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
True ASDS
None None
True RTLS
False ASDS
True Ocean
False Ocean
None ASDS
False RTLS
```

We create a set of outcomes where the second stage did not land successfully:

```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

```
{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

Use the method `.value_counts()` to determine the number and o

```
# Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

Github URL :

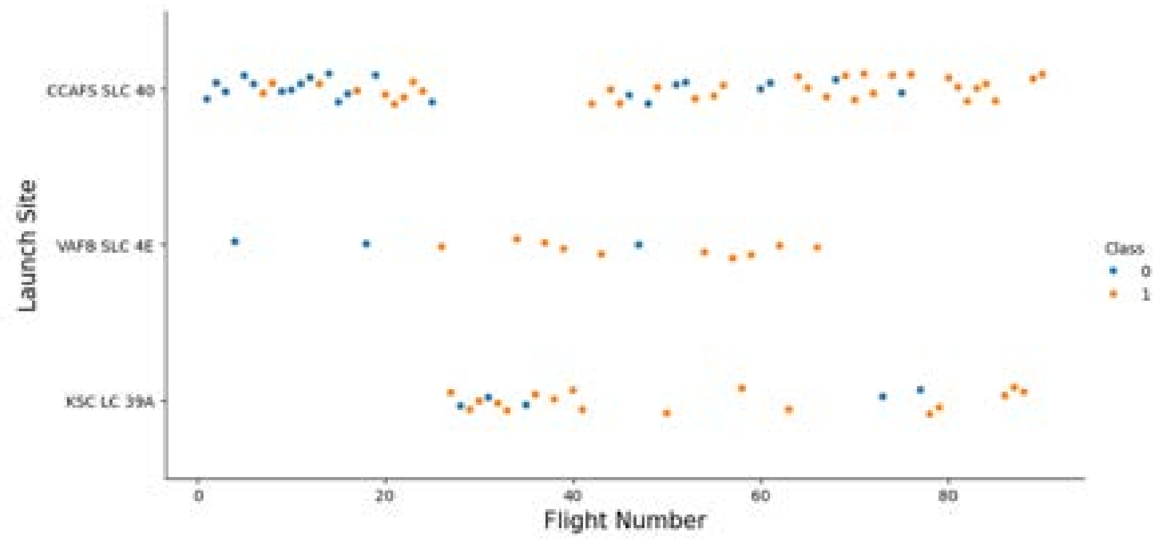
https://github.com/ViaThanh/MySubmission-IBM-Python-Test/blob/51b77aec79c9de5e4f0cb482d960758087f6f3fb/module%201-spacex-data%20wrangling.ipynb

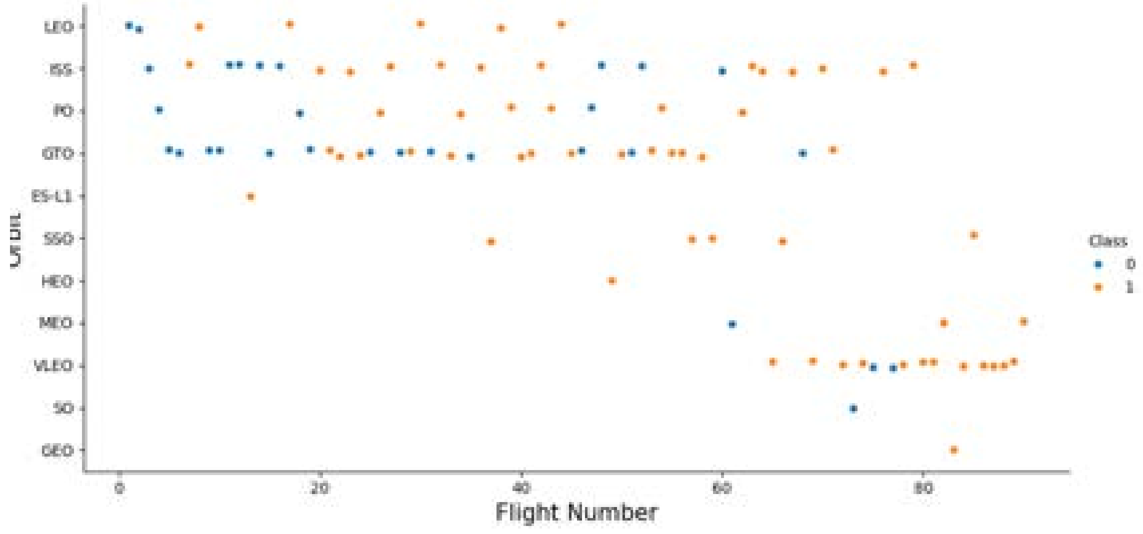
EDA with Data Visualization

01 Scatterplot for FlightNumber and Payload: Higher numbers and payloads often lead to successful landings.

02 Scatterplot for FlightNumber and Launch Site: Shows how sites handle different FlightNumbers.

03 Scatterplot for Payload Mass and Launch Site: Observes mass differences across sites.



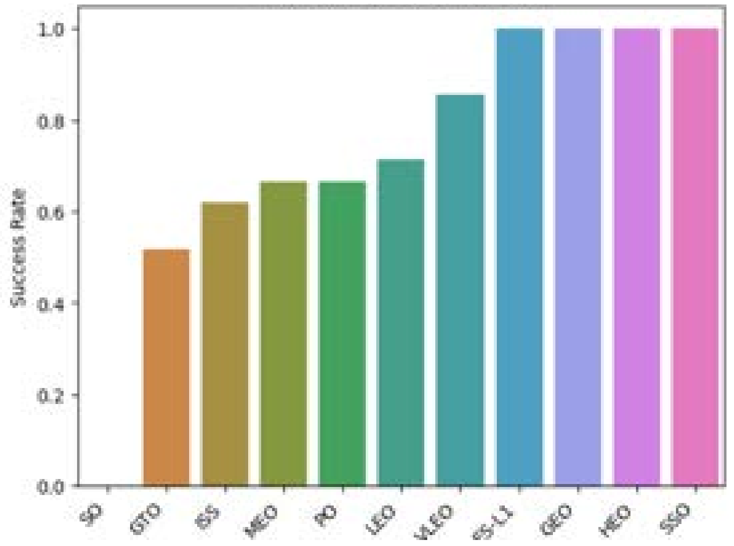


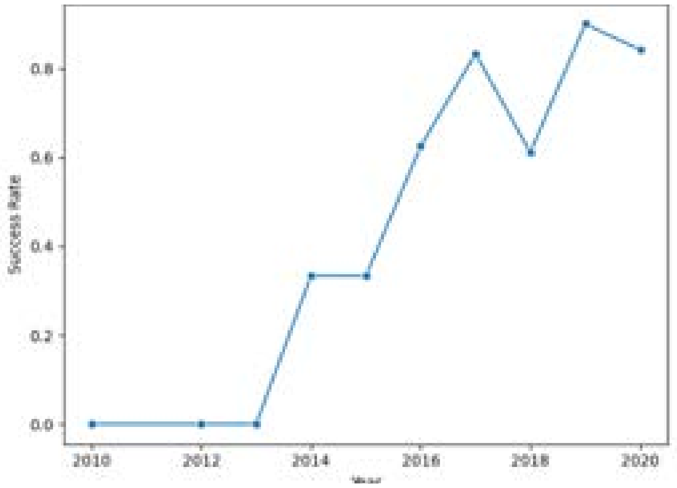
EDA with Data Visualization

04 Bar Plot for Success Rate by Orbit Type: Displays success patterns for orbits.

05 Scatterplot for FlightNumber and Orbit Type: Examines trends with orbits and payload mass.

06 Line Plot for Yearly Launch Success Trend: Shows success trends over time.





Github URL :

https://github.com/ViaThanh/MySubmission-IBM-Python-Test/blob/51b77aec79c9de5e4f0cb482d960758087f6f3fb/module%202-spacex-eda-data-vizualization.ipynb

EDA with SQL

Write and execute SQL queries to solve the assignment tasks.

- Display unique launch site names.
- Show 5 records of launch sites starting with 'CCA'.
- Display total payload mass carried by NASA (CRS) boosters.
- Display average payload mass for booster version F9 v1.1.
- List the date of the first successful ground pad landing.
- List booster names with successful drone ship landings and payloads between 4000 and 6000.

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

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

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

MIN("Date")
2015-12-22

EDA with SQL

- List total successful and failed mission outcomes.
- List booster versions with the maximum payload mass (use a subquery).
- List records for 2015 with month names, failed drone ship landings, booster versions, and launch sites.
- Rank landing outcomes (Failure on drone ship or Success on ground pad) between 2010-06-04 and 2017-03-20, in descending order.

Github URL :

https://github.com/ViaThanh/MySubmission-IBM-Python-Test/blob/51b77aec79c9de5e4f0cb482d960758087f6f3fb/module%202-spacex-eda-sql.ipynb

Build an Interactive Map with Folium

To perform interactive visual analytics using Folium, added some the following map objects:

- Map Initialization: Centered on NASA Johnson Space Center.
- Circles and Markers: Added a blue circle at NASA Johnson Space Center's coordinates with a popup label using *folium.Circle*. Also, added a blue marker at the same location with a label using *folium.Marker*.
- Mouse Position: Added to display coordinates when hovering over the map.
- Proximity Analysis: Drew lines to show distances between launch sites and points of interest using *folium.PolyLine*.

Github URL :

https://github.com/ViaThanh/MySubmission-IBM-Python-Test/blob/51b77aec79c9de5e4f0cb482d960758087f6f3fb/module%203-spacex-interactive-visual-folium.ipynb

Build a Dashboard with Plotly Dash

Plots or graphs and interactions have added to a dashboard:

- Add a dropdown list for selecting Launch Sites.
- Add a pie chart to display the total count of successful launches for all sites.
- Add a slider to choose the payload range.
- Add a scatter chart to illustrate the correlation between payload and launch success.

Github URL :

https://github.com/ViaThanh/MySubmission-IBM-Python-Test/blob/51b77aec79c9de5e4f0cb482d960758087f6f3fb/module%203-spacex-interactive-dash-plotly.py

Predictive Analysis (Classification)

Model Building and Tuning:

- Model : classification algorithms including Logistic Regression, SVM, Decision Tree Classifier, and KNN
- Grid Search: to identify the optimal settings for each model.

Evaluation:

- Training and Testing: split dataset into training and testing sets to evaluate model performance.
- Performance Metrics: The accuracy, precision, recall, F1-score and confusion matrix use to assess and compare model effectiveness.

Final Model:

- Selection: The model with the best performance metrics is Decision Tree as the final predictive model for Falcon 9 landing success.

Predictive Analysis (Classification)

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```

```
svm_cv = GridSearchCV(estimator=svm, param_grid=parameters, cv=10).fit(X_train, Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```

TASK 7

Calculate the accuracy on the test data using the method `score`:

```
accuracy_svm = svm_cv.score(X_test, Y_test)
```

```
accuracy_svm
```

```
0.8333333333333334
```

We can plot the confusion matrix

```
Yhat_svm =svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,Yhat_svm)
```

	Model	Accuracy	Precision	Recall	F1-Score
0	Logistic Regression	0.833333	0.8	1.0	0.888889
1	Support Vector Machine	0.833333	0.8	1.0	0.888889
2	Decision Tree	0.833333	0.8	1.0	0.888889
3	K-Nearest Neighbors	0.833333	0.8	1.0	0.888889

```
# Find the best model
models = {
    'LogisticRegression': logreg_cv.best_score_,
    'SupportVector': svm_cv.best_score_,
    'DecisionTree': tree_cv.best_score_,
    'KNeighbors': knn_cv.best_score_
}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm, 'with a score of', models[bestalgorithm])

# Print the best parameters for the best model
if bestalgorithm == 'LogisticRegression':
    print('Best params are:', logreg_cv.best_params_)
elif bestalgorithm == 'SupportVector':
    print('Best params are:', svm_cv.best_params_)
elif bestalgorithm == 'DecisionTree':
    print('Best params are:', tree_cv.best_params_)
elif bestalgorithm == 'KNeighbors':
    print('Best params are:', knn_cv.best_params_)
```

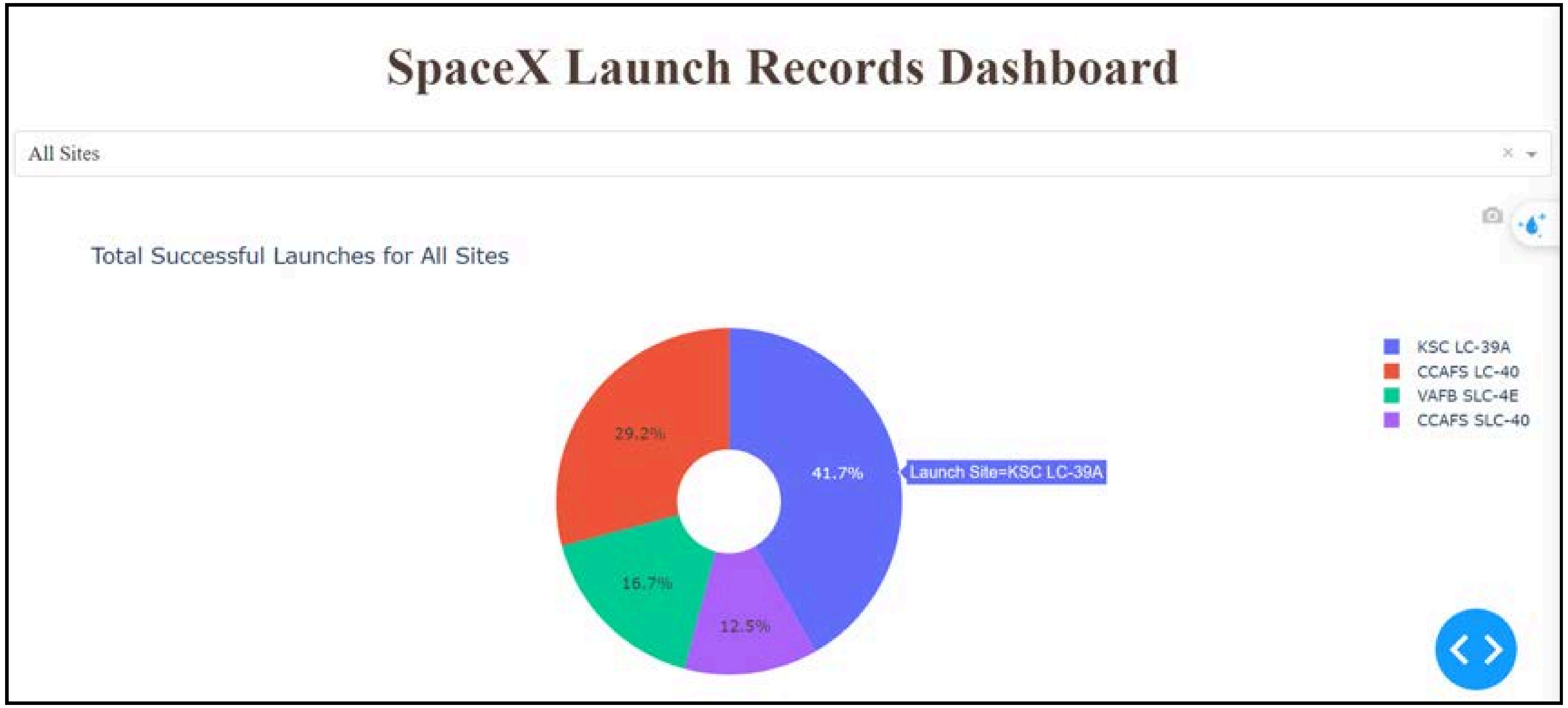
```
Best model is DecisionTree with a score of 0.875
Best params are: {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5,
'splitter': 'random'}
```

Github URL :

https://github.com/ViaThanh/MySubmission-IBM-Python-Test/blob/51b77aec79c9de5e4f0cb482d960758087f6f3fb/module%204-spacex-machine-learning-prediction.ipynb

Results

- As Flight Number increases, successful landings become more likely.
- Orbits ES-L1, GEO, HEO, and SSO have 100% success rates.
- Launch sites near the equator, railways, highways, coastlines, and far from city.
- Success rates have risen consistently since 2013.
- KSC LC-39A has the highest success rate.
- Payload range with the highest success is 2k-5.5k.
- The Decision Tree is the best model with the best score



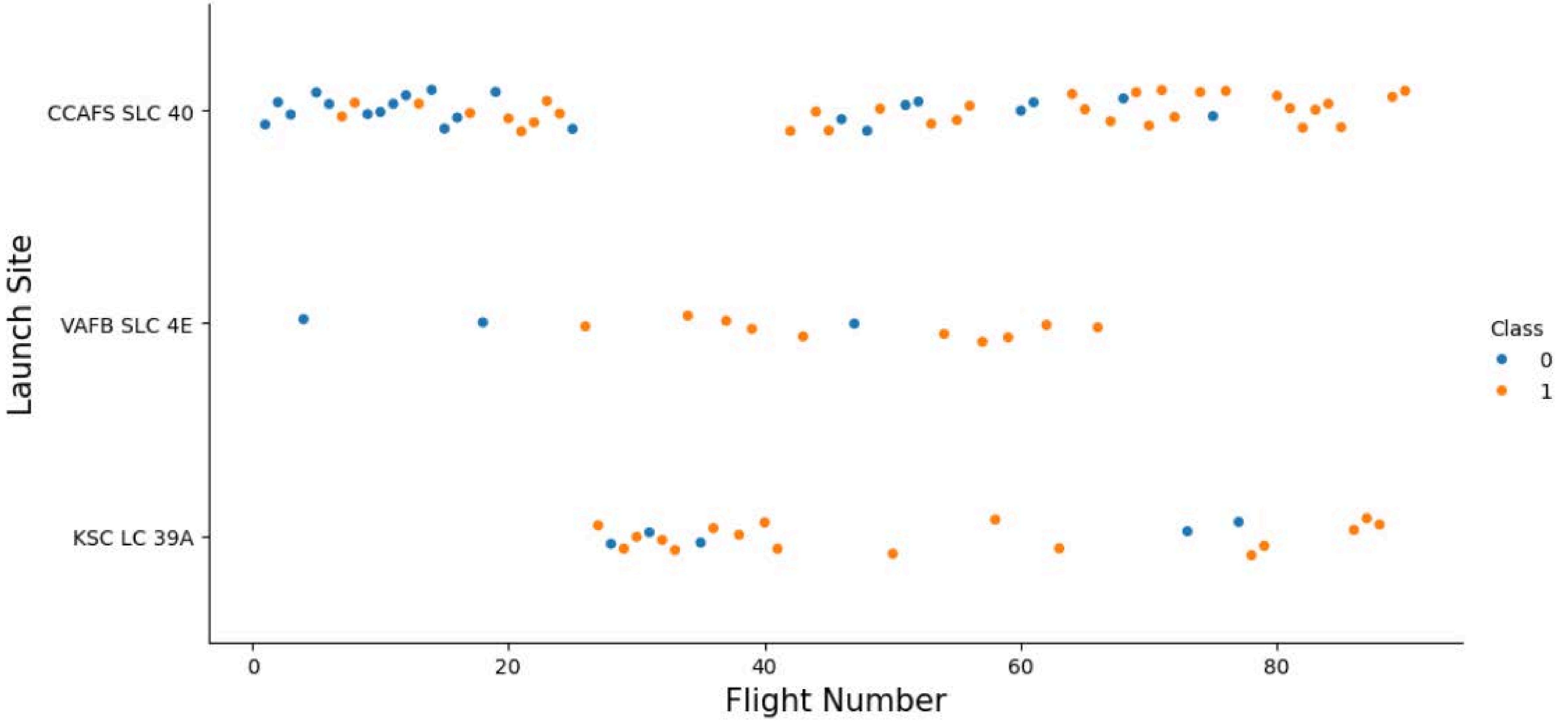
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

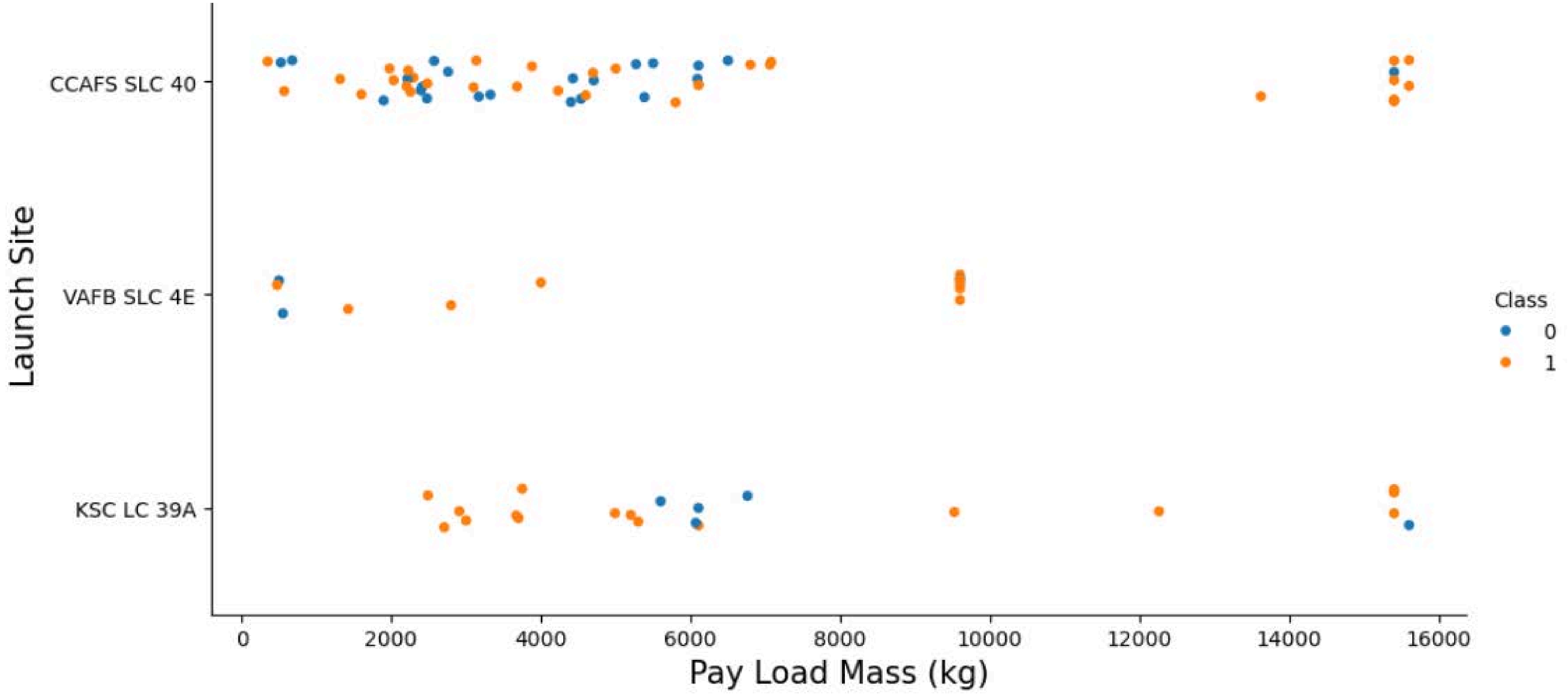
The analysis shows that as FlightNumber increases, successful landings become more likely. Launch Site also influences outcomes, with sites like CCAFS SLC 40, VAFB SLC 4E, and KSC LC 39A showing improved landing success with more flights.

Class 1 : land successful
Class 0 : land unsuccessful



Payload vs. Launch Site

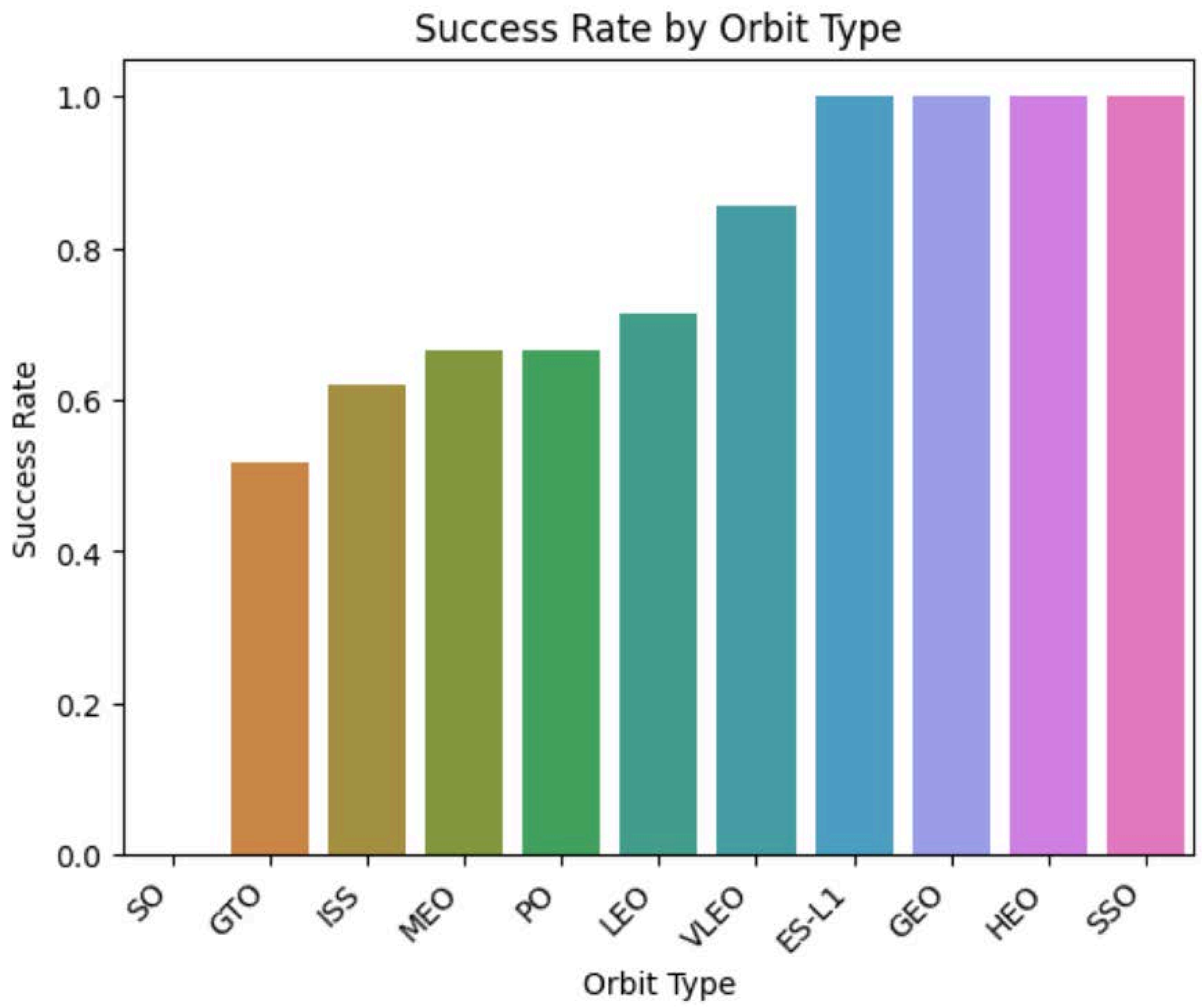
- For CCAFS SLC 40 and KSC LC 39A, as the payload mass increases, successful landings (Class 1) become more frequent.
- VAFB SLC 4E has fewer launches but a high success rate across various payload masses, but there are no rockets launched for heavypayload mass (greater than 10000)



Success Rate vs. Orbit Type

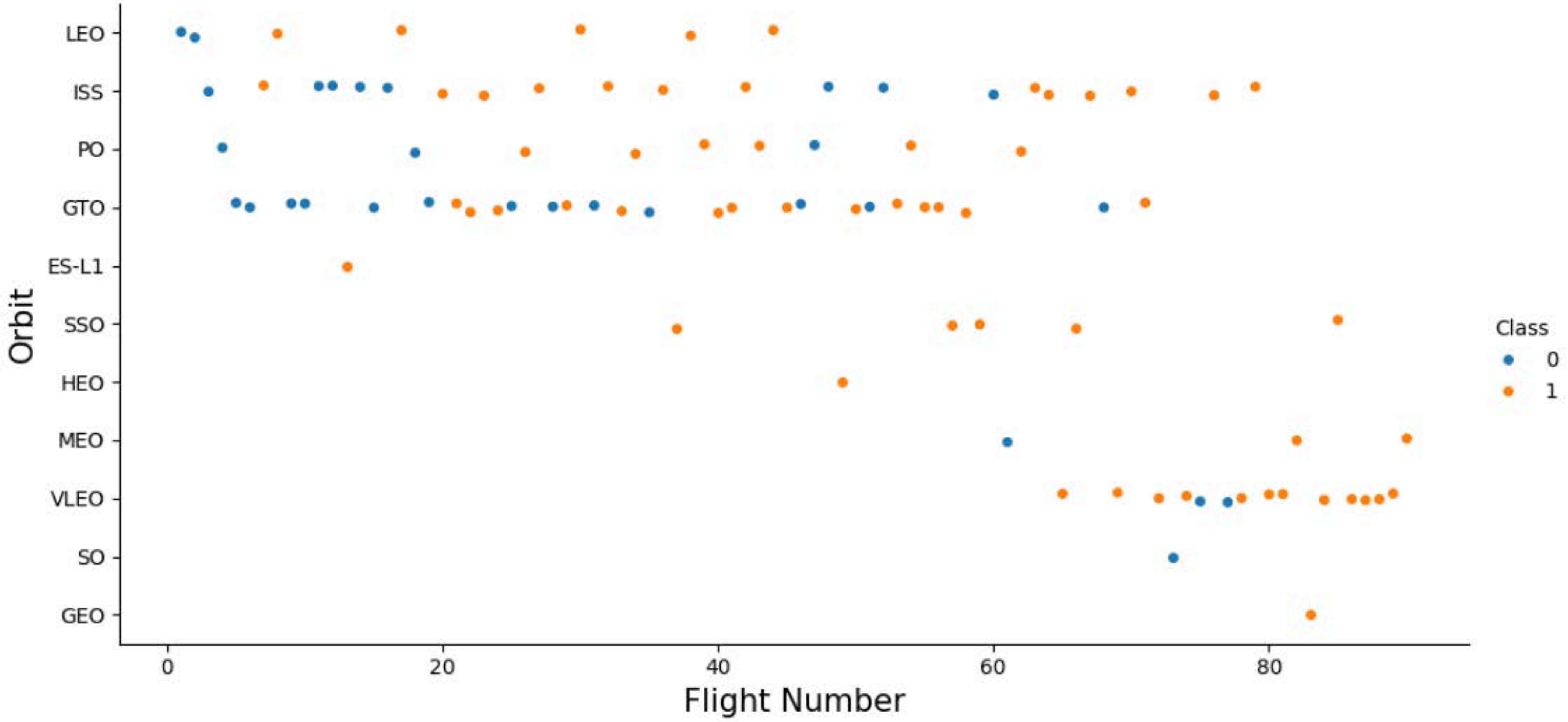
Orbits have the highest success rates (with 100%) :

- ES-L1
- GEO
- HEO
- SSO



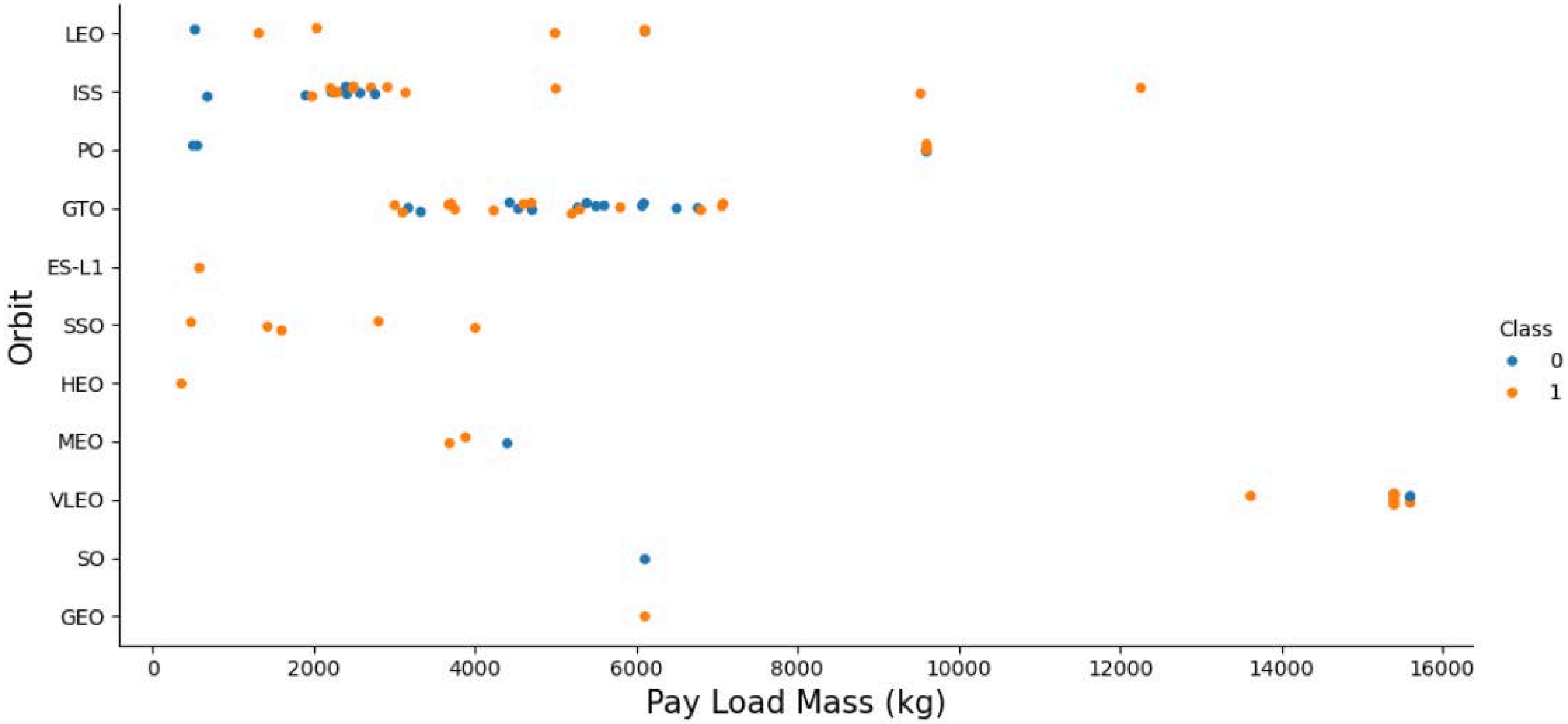
Flight Number vs. Orbit Type

- In the LEO orbit, success appears to be related to the number of flights.
- In contrast, there seems to be no relationship between flight number and success in the GTO orbit.



Payload vs. Orbit Type

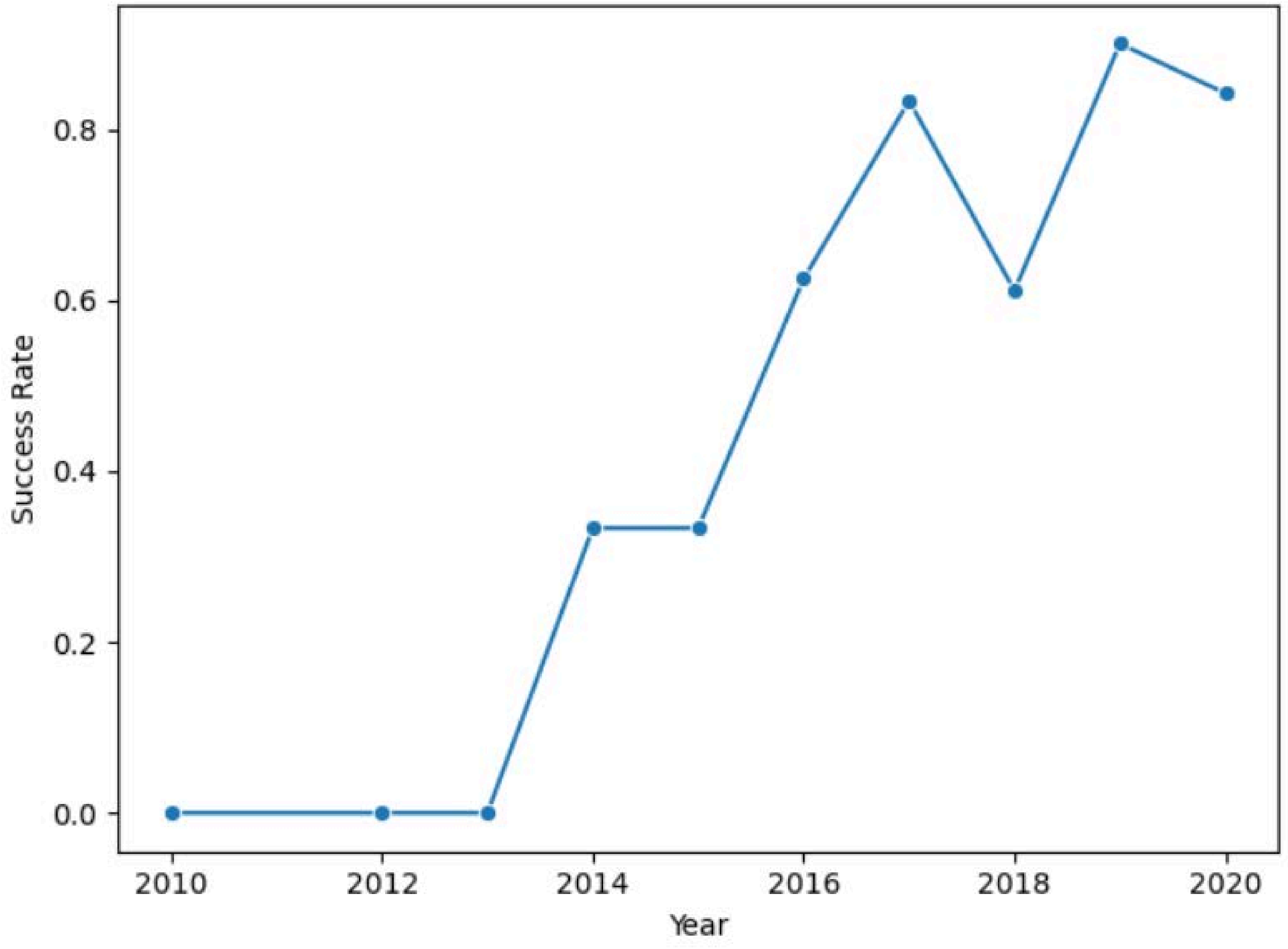
- For heavy payloads, successful landings or higher positive landing rates are observed in Polar, LEO, and ISS orbits.
- In contrast, for GTO orbits, distinguishing between successful and unsuccessful landings is challenging, as both outcomes are present.



Launch Success Yearly Trend

- Significant increase in 2013.
- 2013-2016: Steady rise to around 0.6.
- 2016-2017: Peak above 0.8.
- 2017-2019: Slight decline followed by an increase.

Since 2013, the success rate has consistently increased, indicating a strong positive trend.



All Launch Site Names

This query retrieves a list of unique launch sites from the SPACEXTABLE table. The result shows the distinct launch sites used:

- 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'

This query retrieves up to 5 rows from the SPACEXTABLE table where the Launch_Site begins with 'CCA'. This is useful for filtering records related to specific launch sites.

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

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

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

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

total_payload_mass
45596

- This query calculates the total payload mass carried by boosters launched for NASA (CRS).
- The total payload mass is **45596 kg.**

Average Payload Mass by F9 v1.1

- This query calculates the average payload mass carried by boosters with the version 'F9 v1.1'.
- The average payload mass is **2928.4 kg.**

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

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

avg_payload_mass
2928.4

First Successful Ground Landing Date

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

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

MIN("Date")
2015-12-22

The first successful landing outcome on a ground pad was achieved on **December 22, 2015.**

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%%sql SELECT DISTINCT(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
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

The boosters that have achieved a successful landing on a drone ship and have a payload mass between 4,000 kg and 6,000 kg are:

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

Total Number of Successful and Failure Mission Outcomes

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

- The total number of successful mission outcomes is 100 (all success outcomes).
- The total number of failure mission outcomes is 1.
- This query count the total number of occurrences for each mission outcome type where the outcome starts with 'Success' or 'Failure'.

Boosters Carried Maximum Payload

The query lists the booster versions that have carried the maximum payload mass. The booster versions with the highest payload mass are:

- 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

```
%%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

The query retrieves records from the year 2015 where the landing outcome was 'Failure (drone ship)'.

It displays the month names, booster versions, and launch sites. The results are ordered by month.

For 2015, the records are:

- January: Failure (drone ship), Booster Version: F9 v1.1 B1012, Launch Site: CCAFS LC-40
- April: Failure (drone ship), Booster Version: F9 v1.1 B1015, Launch Site: CCAFS LC-40

```
%%sql
SELECT
    CASE substr("Date", 6, 2)
        WHEN '01' THEN 'January'
        WHEN '02' THEN 'February'
        WHEN '03' THEN 'March'
        WHEN '04' THEN 'April'
        WHEN '05' THEN 'May'
        WHEN '06' THEN 'June'
        WHEN '07' THEN 'July'
        WHEN '08' THEN 'August'
        WHEN '09' THEN 'September'
        WHEN '10' THEN 'October'
        WHEN '11' THEN 'November'
        WHEN '12' THEN 'December'
    END AS month_name,
    "Landing_Outcome",
    "Booster_Version",
    "Launch_Site"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Failure (drone ship)' AND substr("Date", 1, 4) = '2015'
ORDER BY substr("Launch_Date", 6, 2);
```

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

month_name	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

```
%%sql
SELECT "Landing_Outcome", COUNT(*) AS outcome_count, DENSE_RANK() OVER (ORDER BY COUNT(*) DESC) AS rank
FROM SPACEXTABLE
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY outcome_count DESC;
```

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

Landing_Outcome	outcome_count	rank
No attempt	10	1
Success (drone ship)	5	2
Failure (drone ship)	5	2
Success (ground pad)	3	3
Controlled (ocean)	3	3
Uncontrolled (ocean)	2	4
Failure (parachute)	2	4
Precluded (drone ship)	1	5

The query ranks the count of different landing outcomes between June 4, 2010, and March 20, 2017, in descending order.

- No attempt: 10 occurrences: This is the most frequent outcome within the specified date range. It ranks 1st.

Section 3

Launch Sites Proximities Analysis

Launch sites' location on a global map

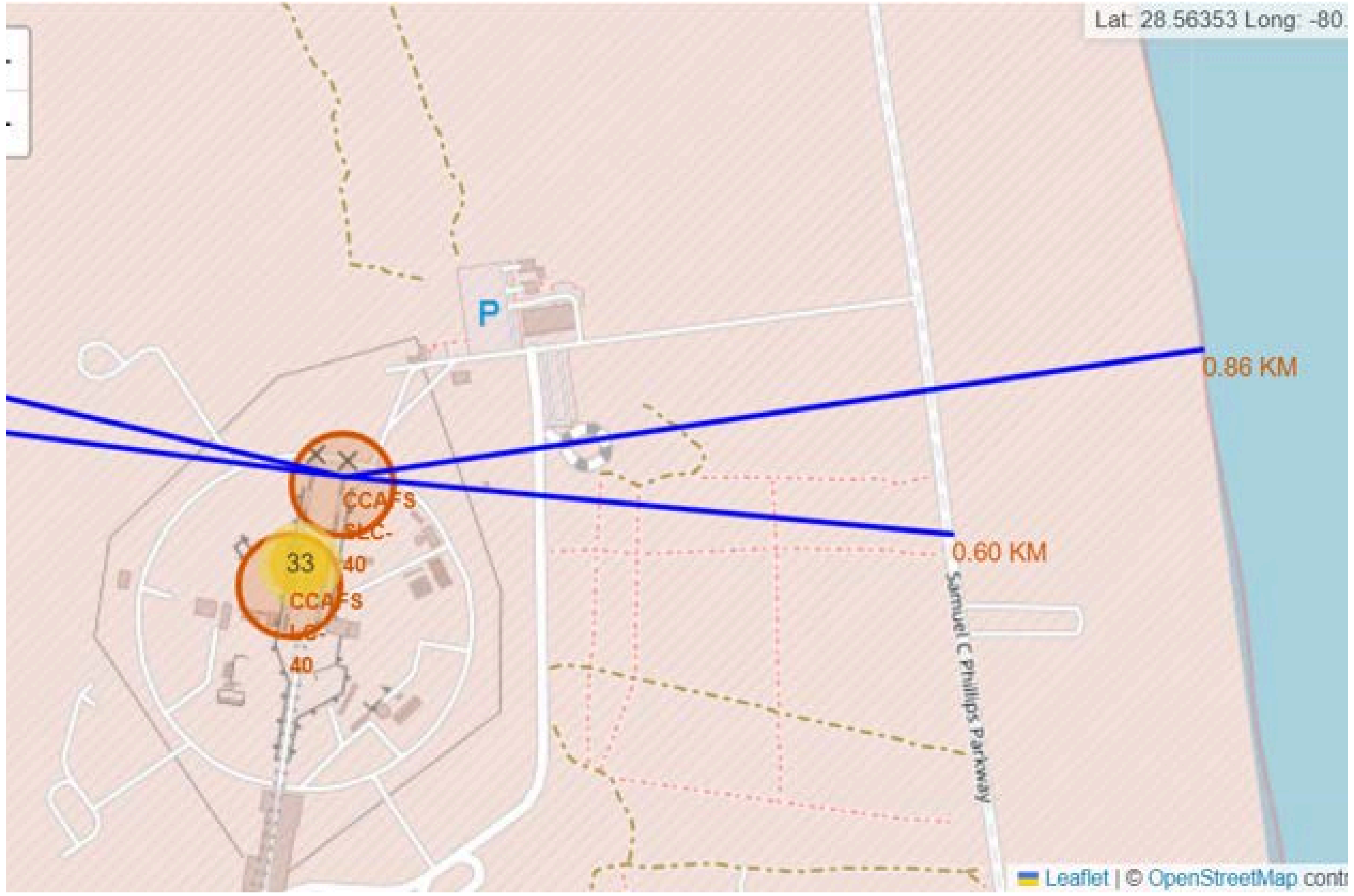
- Near Equator: Rockets gain additional velocity from Earth's rotation, optimizing fuel efficiency.
- Near Coast: Reduces risk to populated areas by allowing rocket debris to fall into the sea.
- Facilitates Transport: Coastal locations ease the transportation of large rocket components by sea.

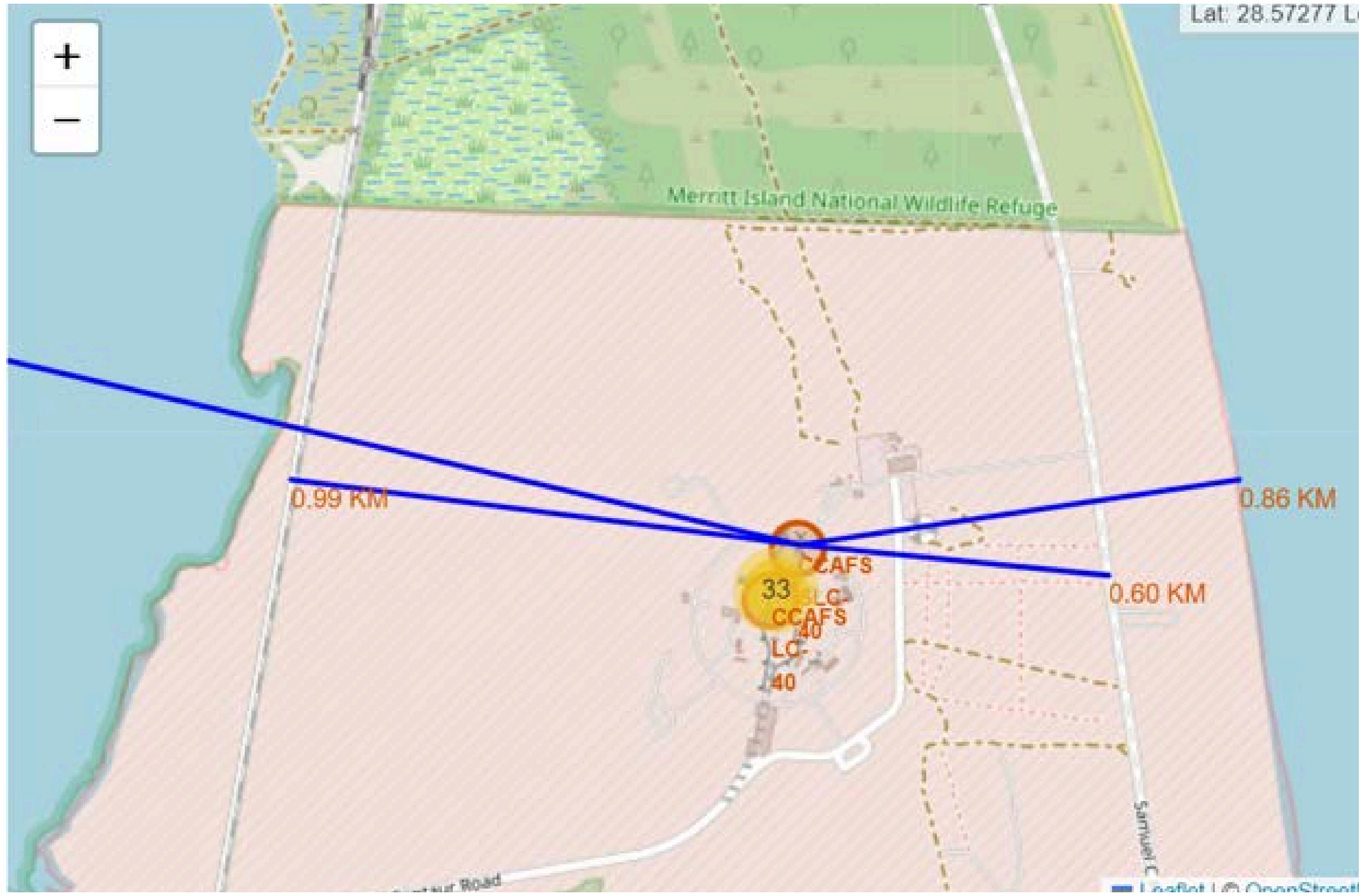
Mark the success/failed launches for each site

Green markers : successful launches
Red markers : failed launches

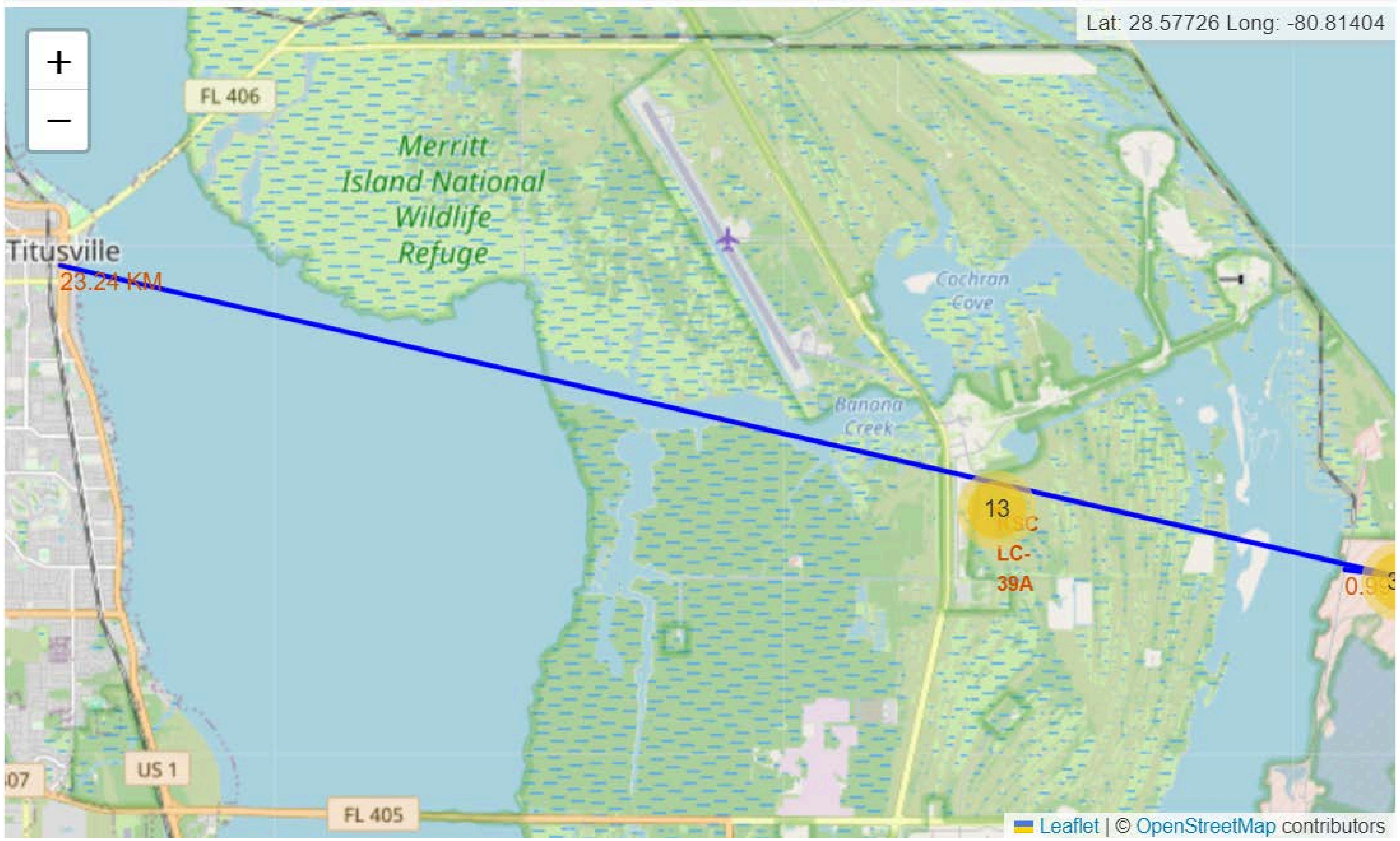
- From the color-labeled markers, we can easily identify which launch sites have relatively high success rates.
- In marker clusters, launch sites are concentrated in a spiral or circular shape, but will not exceed a certain range.

Distances of launch sites





Distances of launch sites

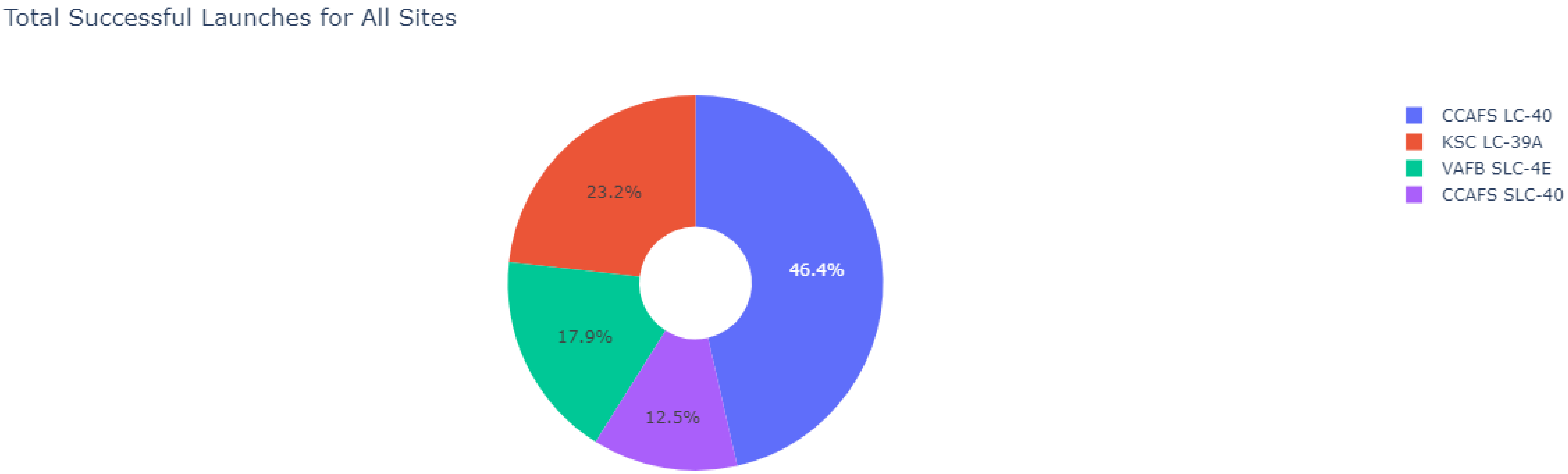


- **Near Railways and Highways:** Facilitates easy transportation of rocket components and equipment.
- **Near Coast:** Allows for safe launches over open water, reducing risk to populated areas.
- **Distance from Cities:** Minimizes noise and environmental impact on urban areas.

Section 4

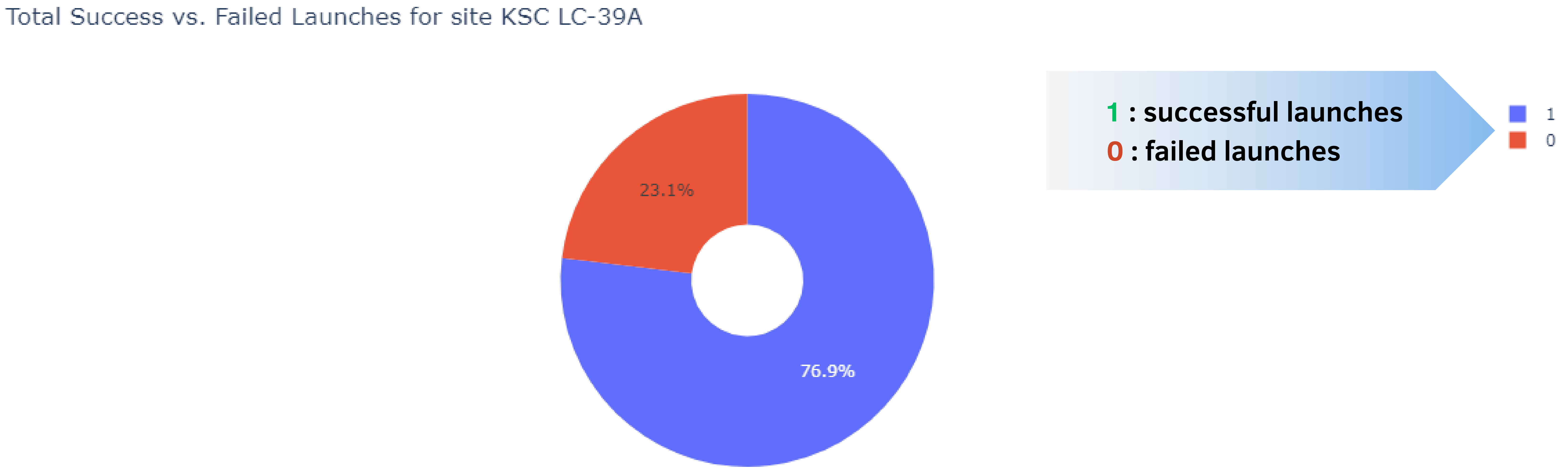
Build a Dashboard with Plotly Dash

Total successful launches for All sites



- Launch success count for all sites including CCAFS LC-40, CCAFS SLC-40, KSC LC-39A and VAFB SLC-4E.
- The launch site has the largest successful launches is KSC LC-39A with 46.4%

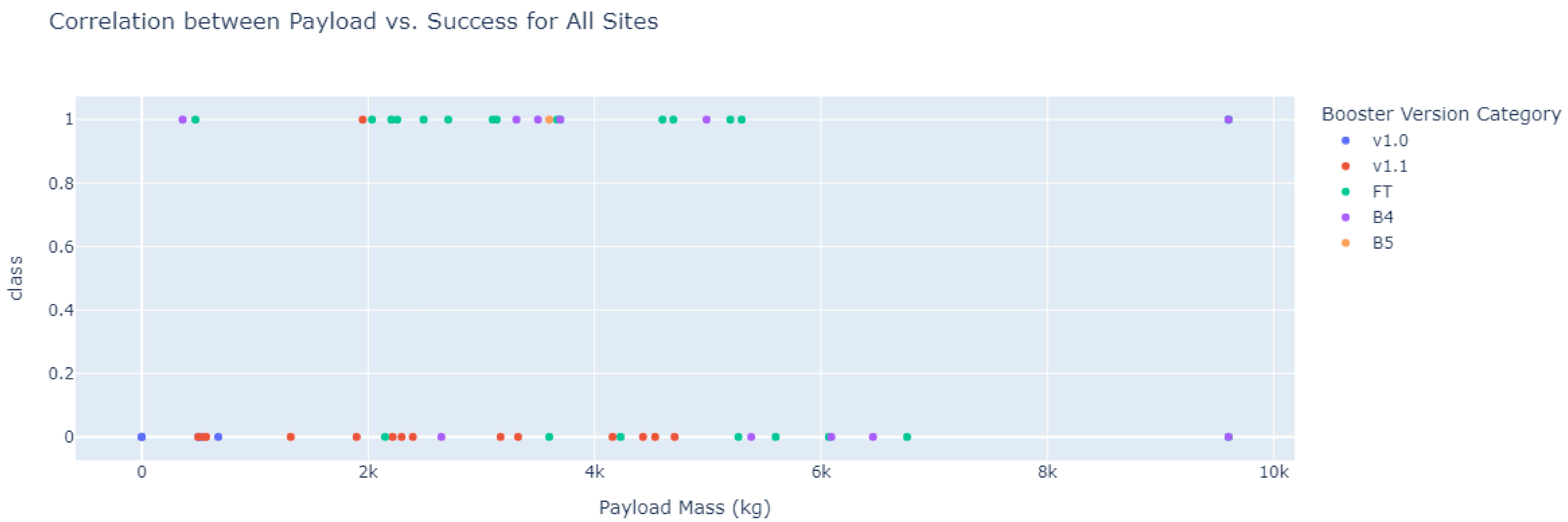
Highest launch success rate



1 : successful launches
0 : failed launches

The launch site has the highest successful launches is KSC LC-39A with 76.9%

Payload vs. Launch Outcome for all sites



- The payload range has the highest launch success rate is 2k-5,5k.
- The payload range has the lowest launch success rate is 2k-7k.
- FT is F9 Booster version has the highest launch success rate.

Section 5

Predictive Analysis (Classification)

Classification Accuracy - Results

- All models : same accuracy, precision, recall, and F1 score.
- We used best_score, a specific evaluation metric, to differentiate their performance.

The Decision Tree model achieved the highest best_score of 0.875, making it the best model

	Model	Accuracy	Precision	Recall	F1-Score
0	Logistic Regression	0.833333	0.8	1.0	0.888889
1	Support Vector Machine	0.833333	0.8	1.0	0.888889
2	Decision Tree	0.833333	0.8	1.0	0.888889
3	K-Nearest Neighbors	0.833333	0.8	1.0	0.888889

```python
# Find the best model
models = {
    'LogisticRegression': logreg_cv.best_score_,
    'SupportVector': svm_cv.best_score_,
    'DecisionTree': tree_cv.best_score_,
    'KNeighbors': knn_cv.best_score_
}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm, 'with a score of', models[bestalgorithm])

# Print the best parameters for the best model
if bestalgorithm == 'LogisticRegression':
    print('Best params are:', logreg_cv.best_params_)
elif bestalgorithm == 'SupportVector':
    print('Best params are:', svm_cv.best_params_)
elif bestalgorithm == 'DecisionTree':
    print('Best params are:', tree_cv.best_params_)
elif bestalgorithm == 'KNeighbors':
    print('Best params are:', knn_cv.best_params_)
```

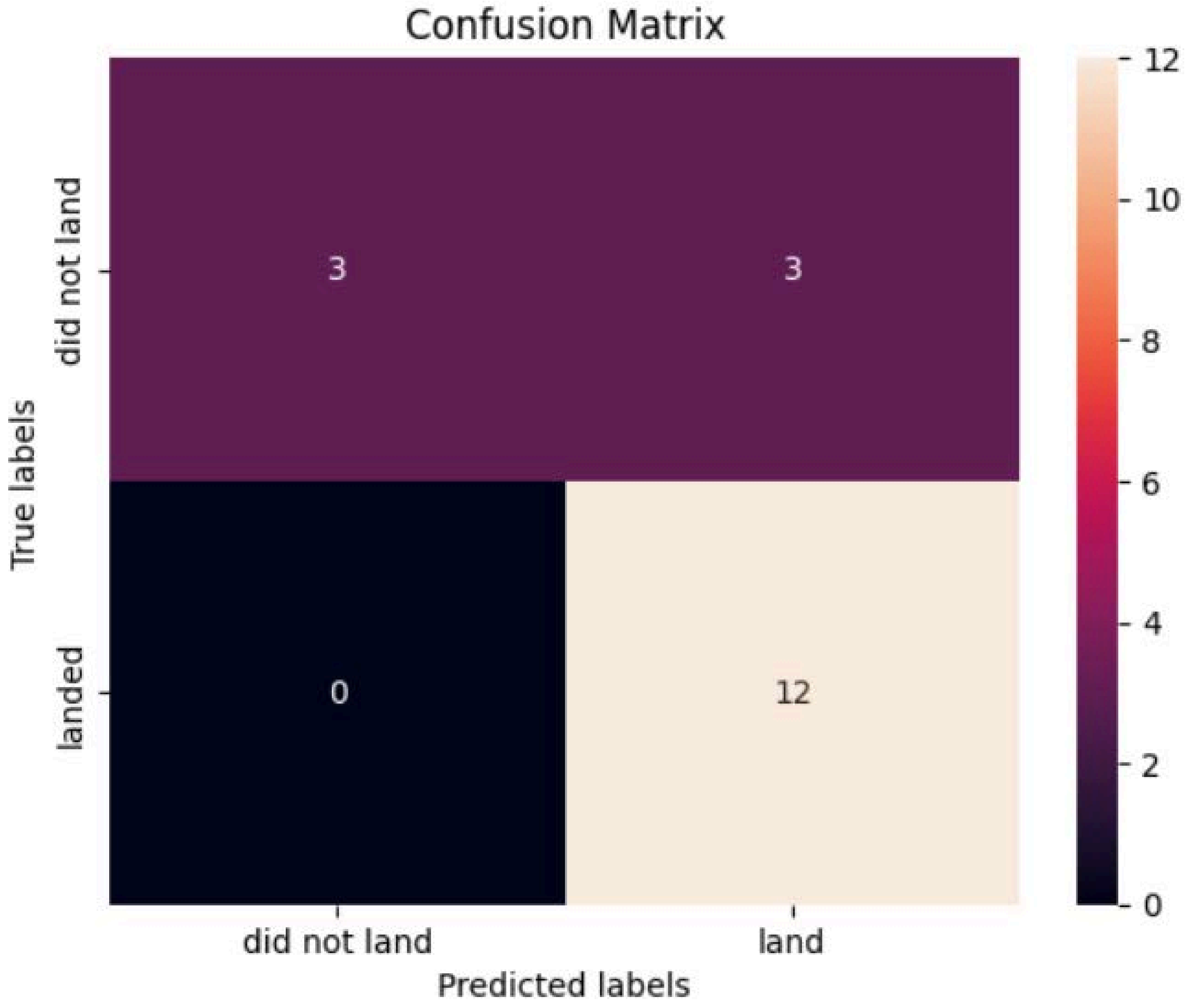
```
Best model is DecisionTree with a score of 0.875
Best params are: {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5,
'splitter': 'random'}
```

Confusion Matrix - Results

The confusion matrix indicates that the model is particularly strong at correctly identifying landings but has a moderate rate of false positives, predicting some "did not land" instances as "landed".

With :

- True Positives (TP): 12
- True Negatives (TN): 3
- False Positives (FP): 3
- False Negatives (FN): 0



Conclusions

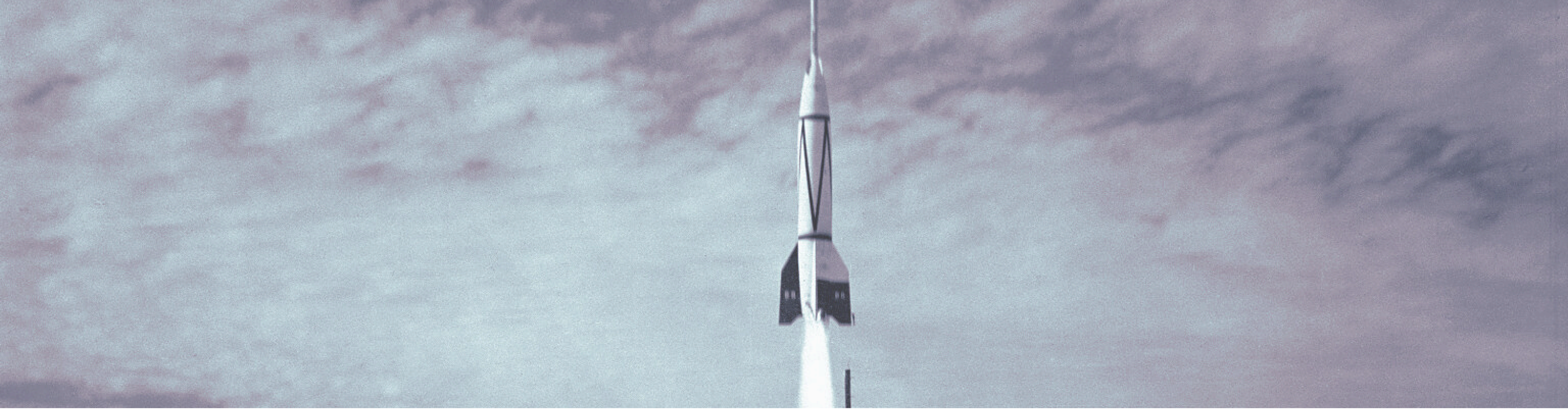
- As Flight Number increases, the likelihood of successful landings also rises

- Orbits have the highest success rates (100%) : ES-L1 , GEO , HEO , SSO
- Since 2013, the success rate has consistently increased, indicating a strong positive trend.

- Launch sites near the equator, railways and highways simplify transportation, sites near coastlines ensure safe water launches, and remote sites minimize noise and environmental impact.

- KSC LC-39A is the launch site has the highest successful launches The payload range has the highest launch success rate is 2k-5,5k.

- The Decision Tree is the best model with a best_score of 0.875



Discussion

- The project successfully created a predictive model for the Falcon 9's first stage landing success, underpinned by thorough data analysis and visualization.
- The provided files, plots, and interactive dashboards offer stakeholders an intuitive tool to explore data and insights.
- Future efforts will focus on refining the model and enhancing the dashboard's features.

Appendix

- **Data Sources:** Detailed information on the SpaceX API and web scraping sources.
- **Code and Algorithms:** Descriptions of the machine learning algorithms and configurations used.
- **Additional Figures:** Supplementary visualizations and data analyses

Thank you!