IBM Developer
SKILLS NETWORK

SPACEX FALCON 9

FIRST STAGE LANDING PREDICTION

NAME: HENRY SUÁREZ

DATE: 02/10/24

OUTLINE

Executive Summary

Introduction

Methodology

Conclusion

Appendix

EXECUTIVE SUMMARY

We will be working with SpaceX launch data obtained from the SpaceX REST API. This API provides comprehensive information about launches, including details about the rockets used, payloads delivered, launch specifications, landing requirements, and landing outcomes. Our objective is to analyze this data to predict whether SpaceX will attempt to land a rocket.

INTRODUCTION

Background and Context

SpaceX advertises Falcon 9 rocket launches on its website for a cost of $62 million, while other providers charge upwards of $165 million for similar services. A significant portion of these savings can be attributed to SpaceX's ability to reuse the first stage of the rocket. Consequently, if we can accurately predict whether the first stage will successfully land, we can better estimate the overall cost of a launch.

Problem

The goal is to train a machine learning model using publicly available information to predict whether SpaceX will reuse the first stage of the Falcon 9 rocket.

METHODOLOGY

Data Collection Methodology

1.Data Wrangling: Clean and preprocess the collected data to ensure it is structured and ready for analysis.

2.Exploratory Data Analysis (EDA): Conduct EDA using visualizations and SQL queries to uncover insights and patterns within the data.

3.Interactive Visual Analytics: Utilize Folium and Plotly Dash to create interactive visualizations that enhance data exploration and understanding.

4.Predictive Analysis: Apply classification models to perform predictive analysis, enabling us to forecast outcomes based on the data.

Data collection

- SpaceX REST API :

Data collected using SpaceX API

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cl
```

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())
```

Data collection

- Web scrapping data from Wikipedia using BeautifulSoup

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

Create a `BeautifulSoup` object from the HTML `response`

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text cont
soup = BeautifulSoup(response, 'html.parser')
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data wrangling

- Convert data into a more usable form
- Determine training label (1 means the booster successfully landed, 0 means it was unsuccessful)
- Preparing data feature engineering

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

```
# landing_class = 0 if bad_outcome
landing_class = [0 if x in bad_outcomes else 1 for x in df['Outcome']]
# landing_class = 1 otherwise
df['Class']=landing_class
```

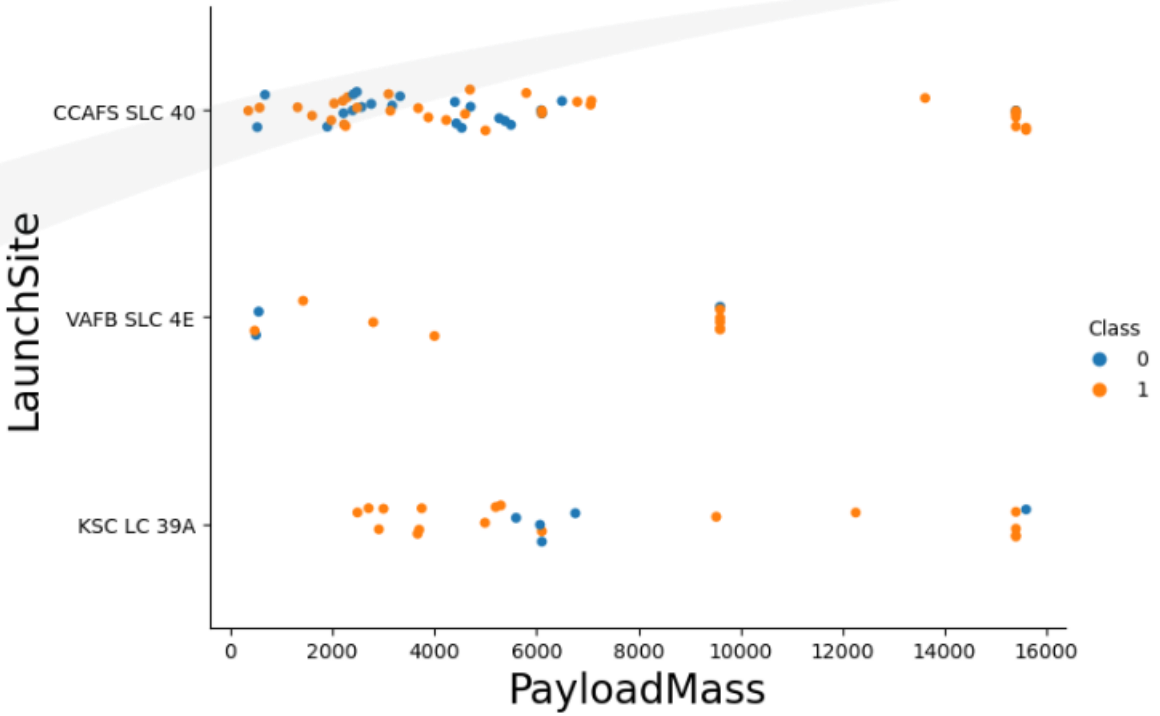
This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

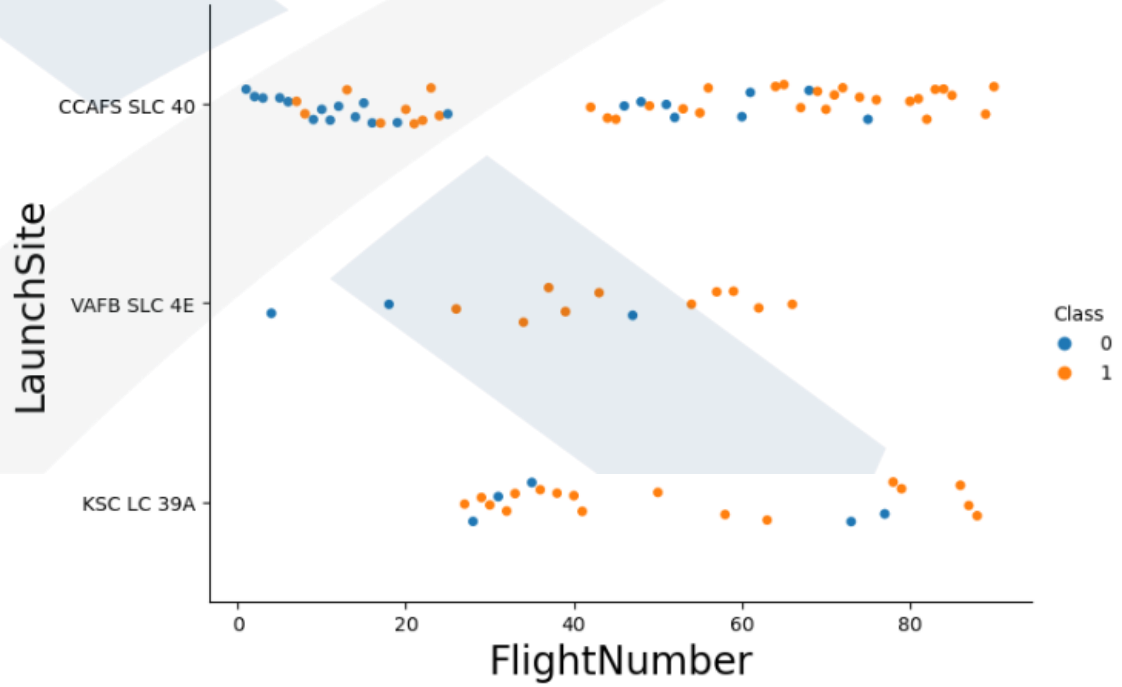
```
df['Class']=landing_class
df[['Class']].head(8)
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

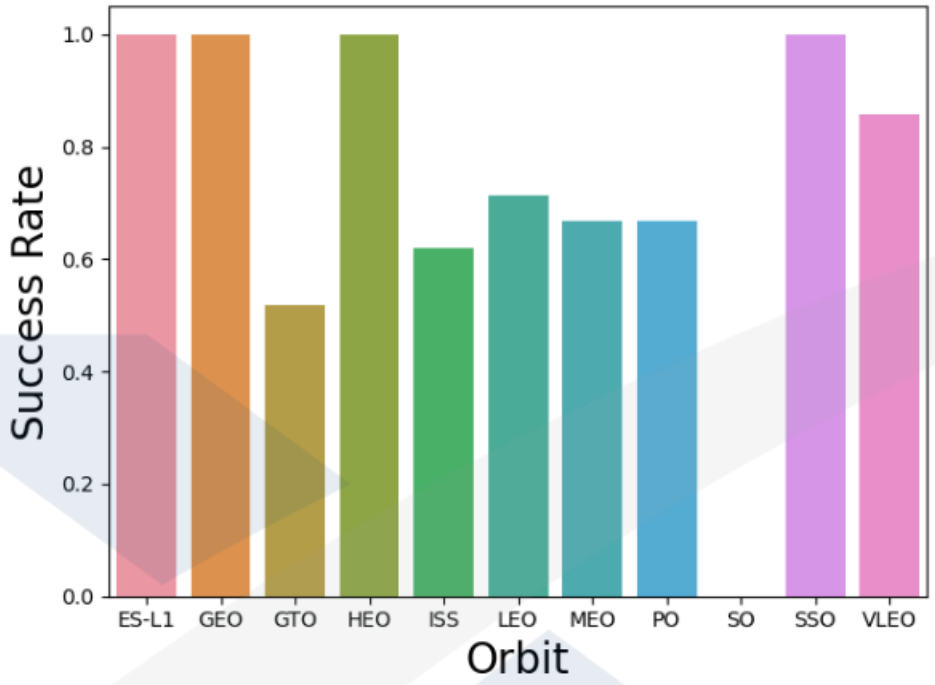
EDA with Data Visualization

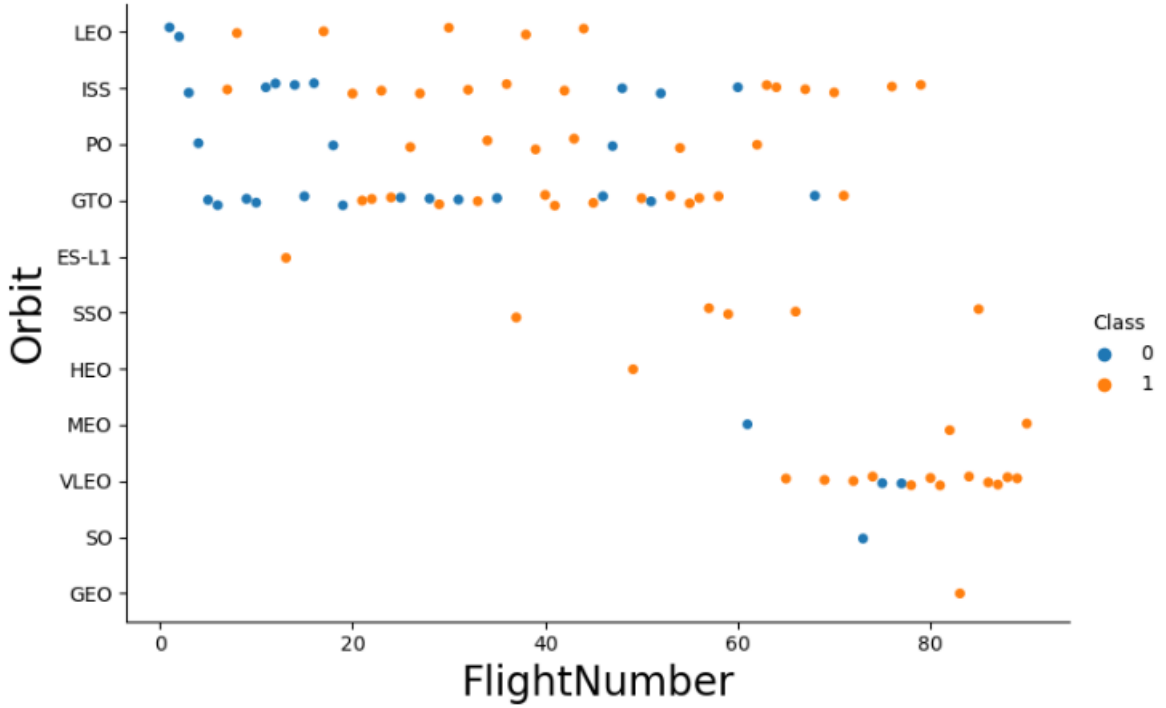
- Exploratory data analysis
- Preparing data feature engineering



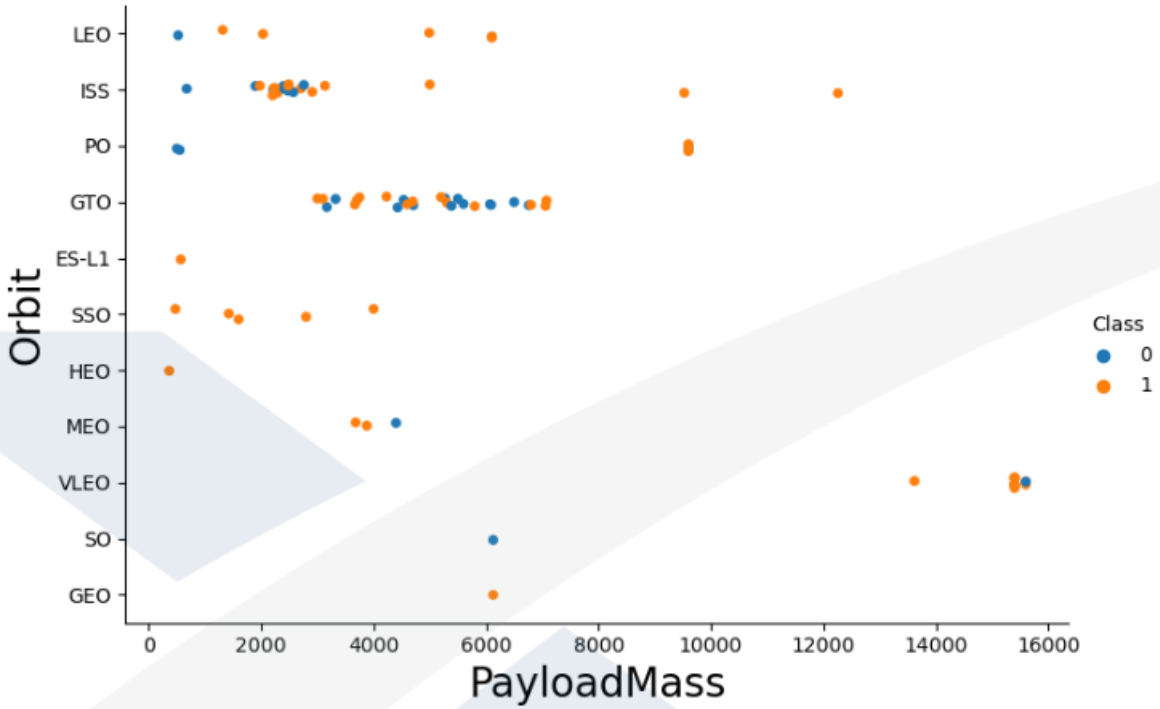


EDA with Data Visualization (Cont.)

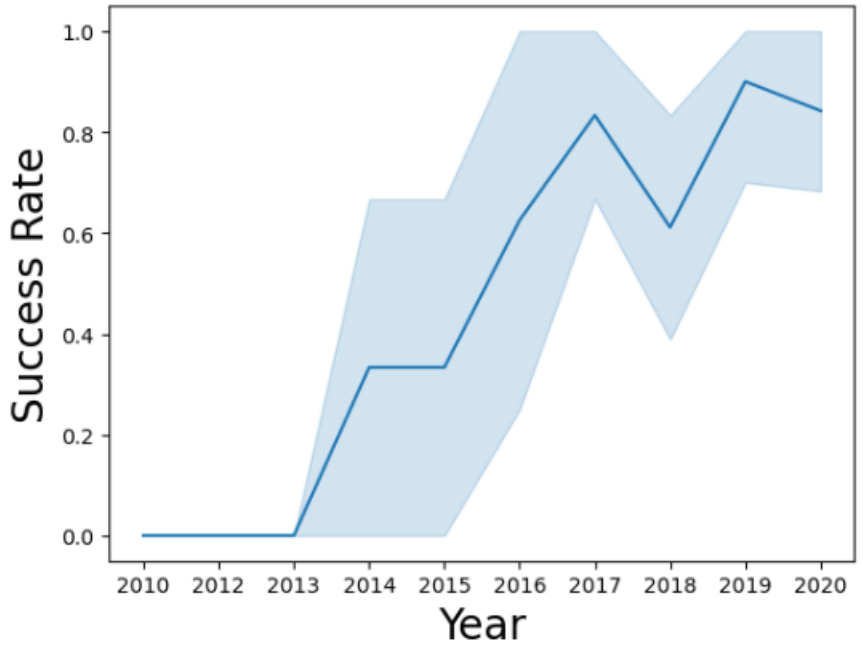




EDA with Data Visualization (Cont.)



Success rates have generally increased over time since 2013, with a slight dip observed in 2018. In recent years, the success rate has stabilized at approximately 80%.



EDA with SQL

Execute SQL queries to answer questions

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

```
%sql SELECT DISTINCT LAUNCH_SITE AS 'Launch Sites' FROM SPACEXTBL;
```

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

Launch Sites
CCAFS LC-40
VAFB SLC-4E
KSC LC-39A
CCAFS SLC-40

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

```
%sql SELECT * FROM SPACEXTBL 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	Landin
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure
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
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	

EDA with SQL (Cont.)

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS 'Payload Mass (Kgs)', Customer FROM SPACEXTBL WHERE Customer = 'N
```

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

Payload Mass (Kgs)	Customer
45596	NASA (CRS)

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS 'Payload Mass Kgs', Customer, Booster_Version FROM SPACEXTBL WHEF
```

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

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

EDA with SQL (Cont.)

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

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

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

MIN(DATE)
2015-12-22

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, Payload FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone sh
```

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

Booster_Version	Payload
F9 FT B1022	JCSAT-14
F9 FT B1026	JCSAT-16
F9 FT B1021.2	SES-10
F9 FT B1031.2	SES-11 / EchoStar 105

EDA with SQL (Cont.)

List the total number of successful and failure mission outcomes

```
%sql SELECT Mission_Outcome, COUNT(Mission_Outcome) AS Total FROM SPACEXTBL 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

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

```
%sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FRO
```

```
 * 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

EDA with SQL (Cont.)

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.

```
%sql SELECT SUBSTR(DATE,6,2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL whe
```

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

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 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(*) AS Outcome_Count FROM SPACEXTBL WHERE Date BETWEEN '2010-06-04' A
```

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

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

Interactive visual anaytics using Folium

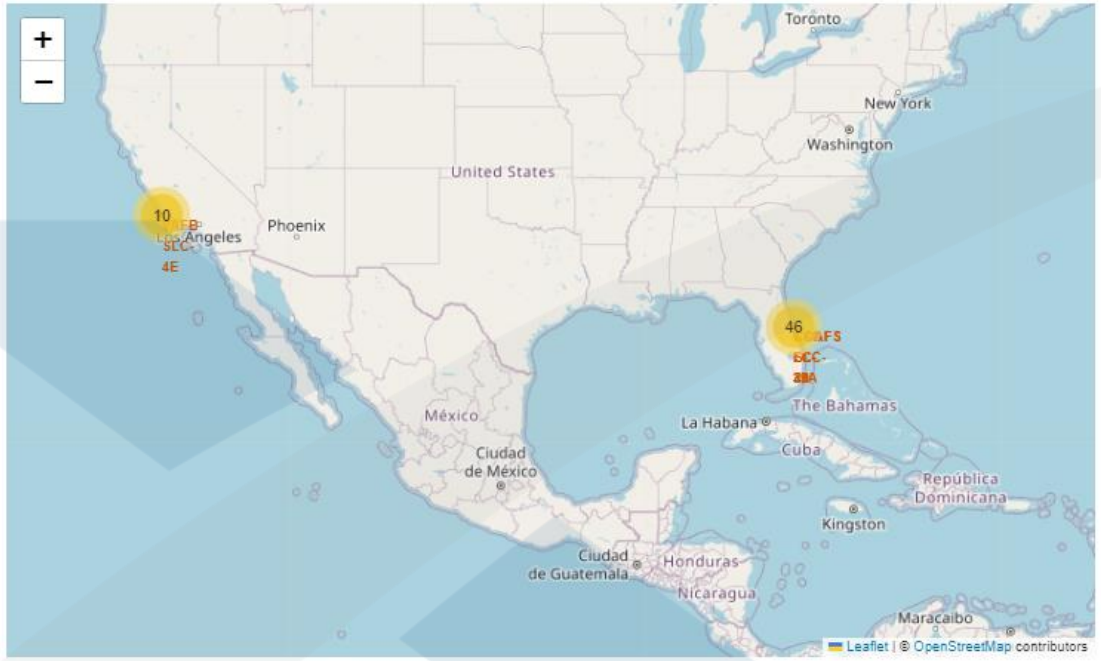
Adding Launch Sites to the Map with Folium

To visualize SpaceX launch sites on an interactive map, we can use **Folium** to add both circles and markers for each site. This allows us to clearly represent the locations and add pop-ups or labels for further details.

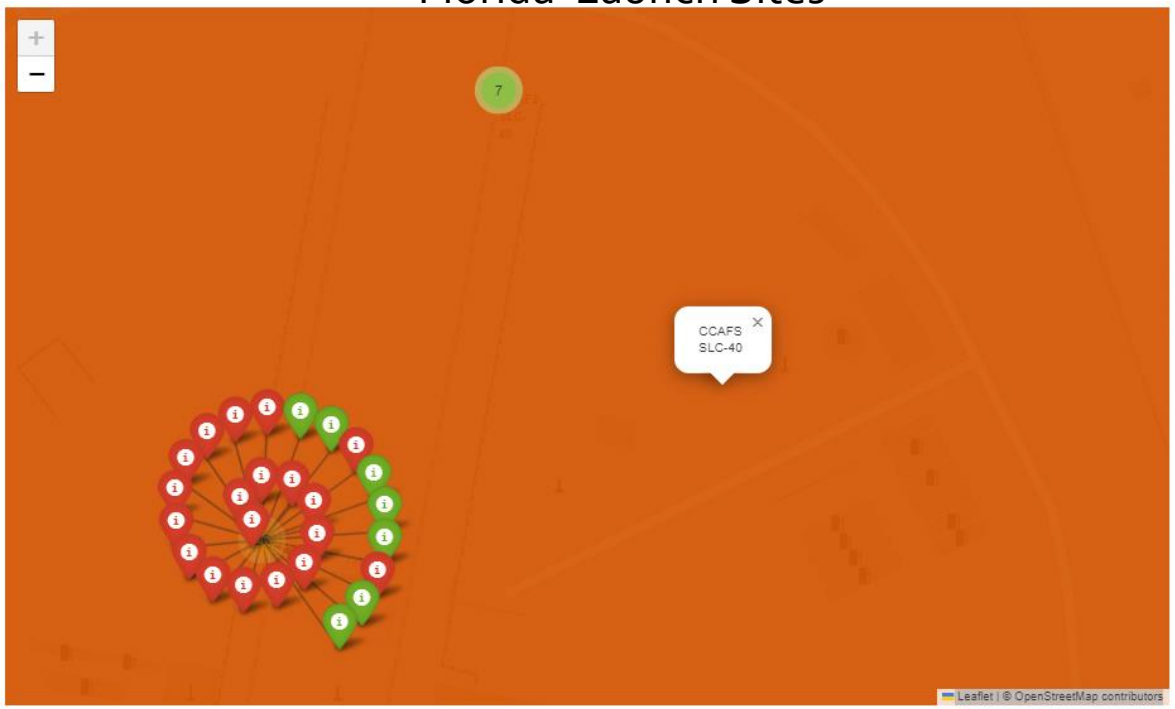
- **Circles** represent the launch sites, giving us a visual indication of their locations with customizable size and color.
- **Markers** provide labels or icons, enhancing interactivity by showing specific information when clicked.

Interactive visual anaytics using Folium (Cont.)

Launch Sites



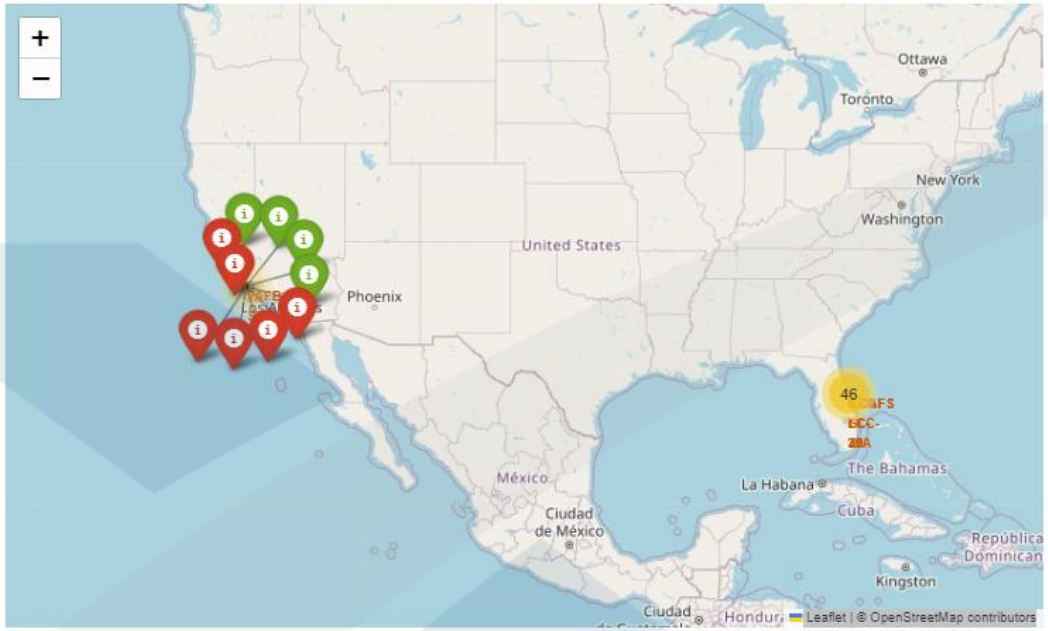
Florida Launch Sites



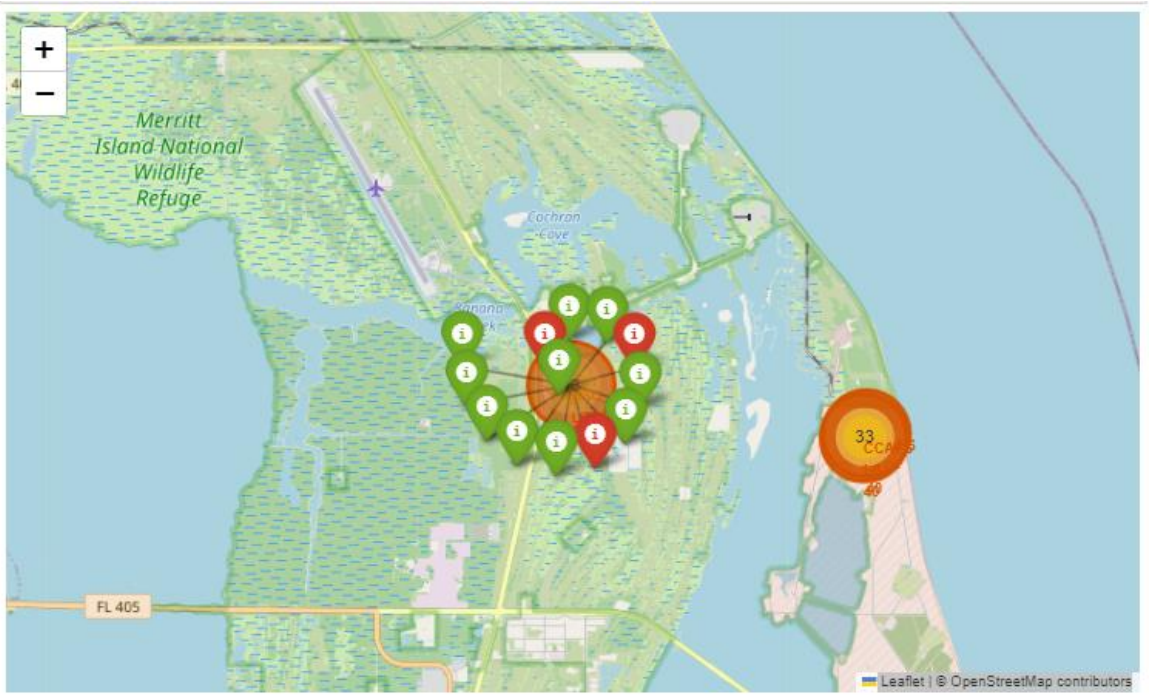
Interactive visual anaytics using Folium (Cont.)

Clusters on the Folium map are interactive and can be clicked to reveal individual landings. **Green icons** represent successful landings, while **red icons** indicate failed landings.

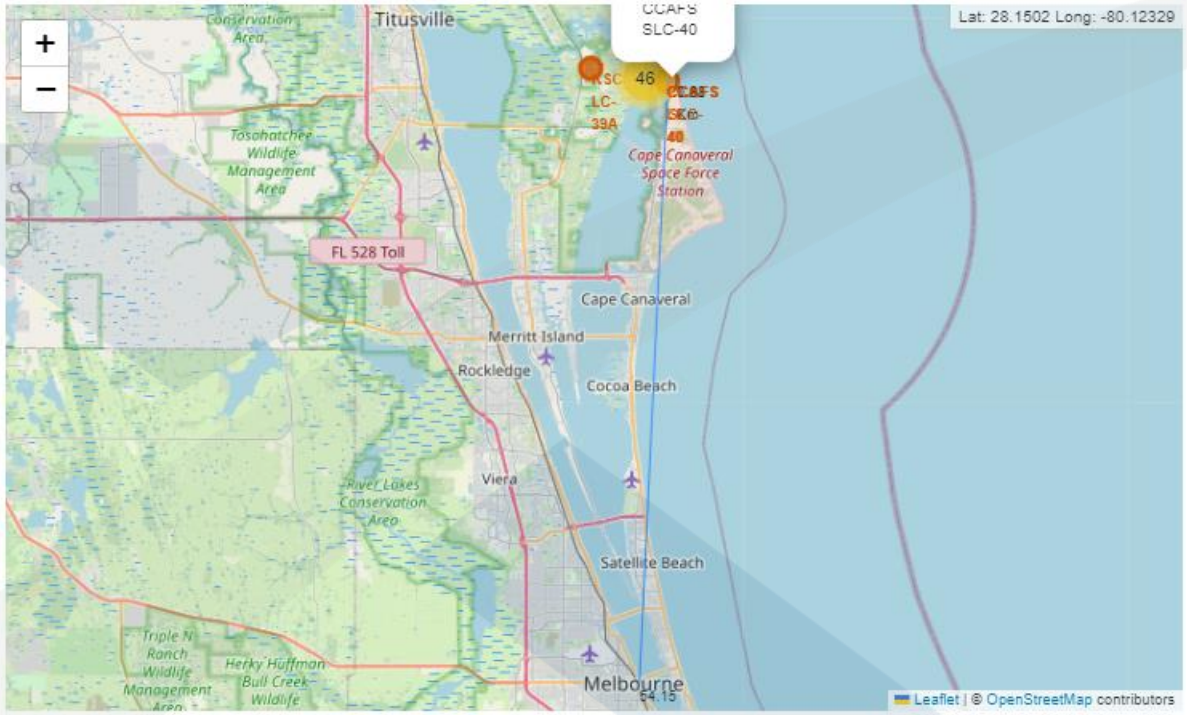
California Launch Sites



Florida Launch Sites

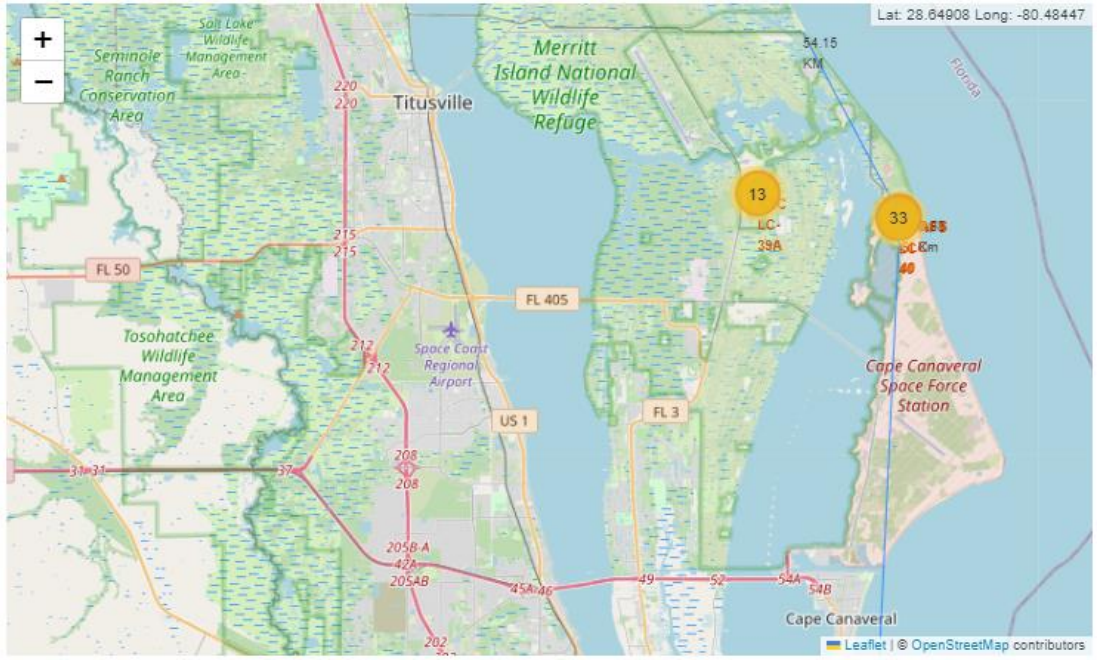


Interactive visual anaytics using Folium (Cont.)

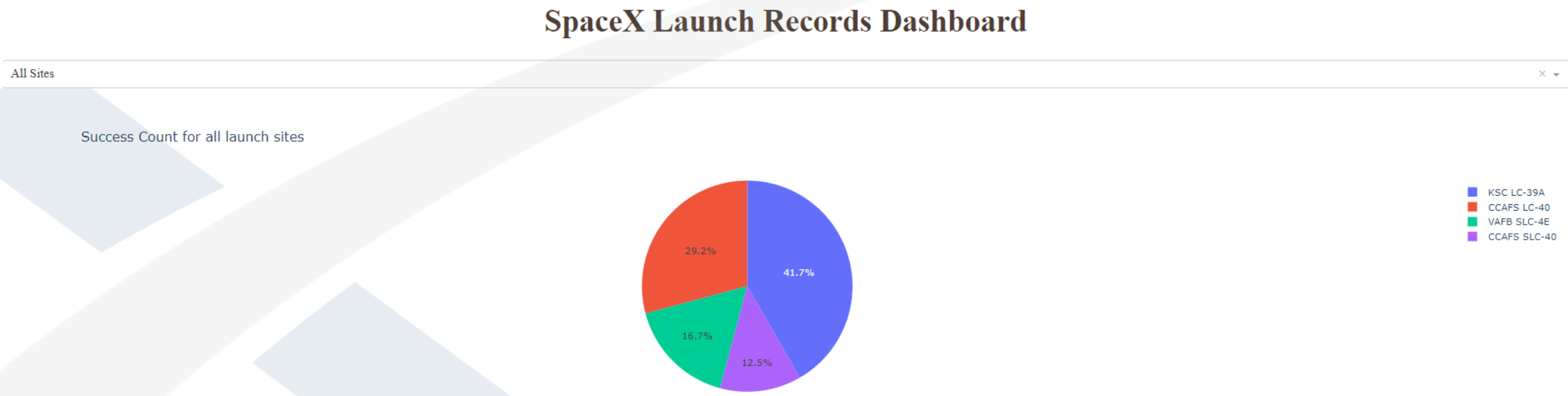


Proximity of Locations

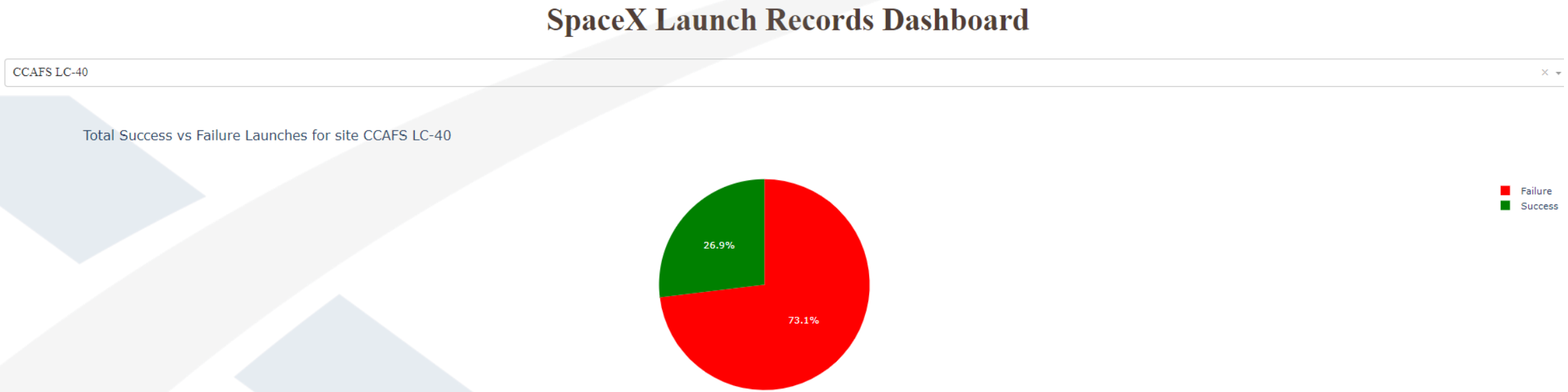
Using CCAFS SLC-40 as an example, launch sites are strategically located near railways to facilitate the transport of large equipment and supplies. Launch sites are situated near coasts and relatively far from cities, ensuring that in the event of a launch failure, debris can safely land in the sea, minimizing the risk to densely populated areas.



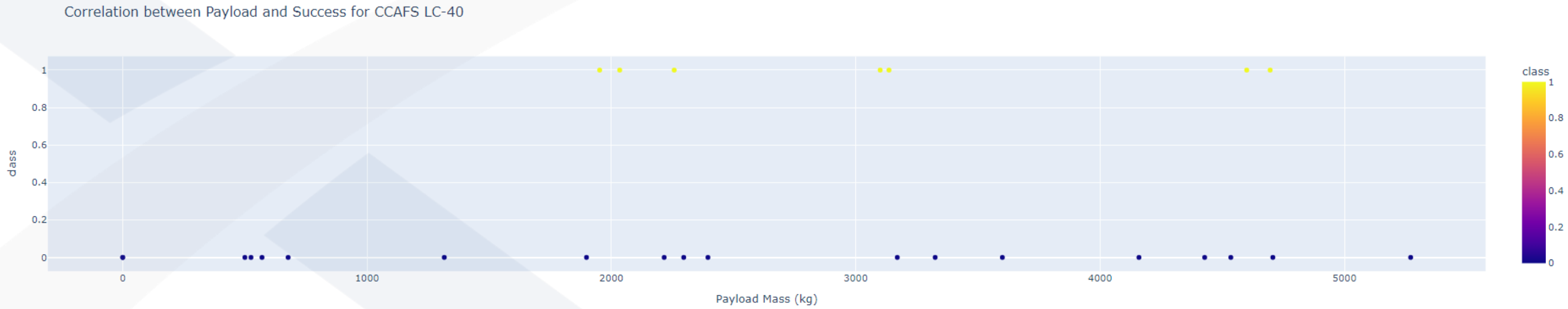
Interactive visual anaytics using Plotly Dash



Interactive visual anaytics using Plotly Dash



Interactive visual anaytics using Plotly Dash



Predictive analysis using classification models

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

```
print('Logistic Regression test data accuracy: ', logreg_cv.score(X_test, Y_test))
```

```
Logistic Regression test data accuracy:  0.8333333333333334
```

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

```
print('SVM test data accuracy: ', svm_cv.score(X_test, Y_test))
```

```
SVM test data accuracy:  0.8333333333333334
```

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

```
print('Decision Tree accuracy on test set: ', tree_cv.score(X_test, Y_test))
```

```
Decision Tree accuracy on test set:  0.7222222222222222
```

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

```
knn_cv.score(X_test, Y_test)
```

```
0.8333333333333334
```

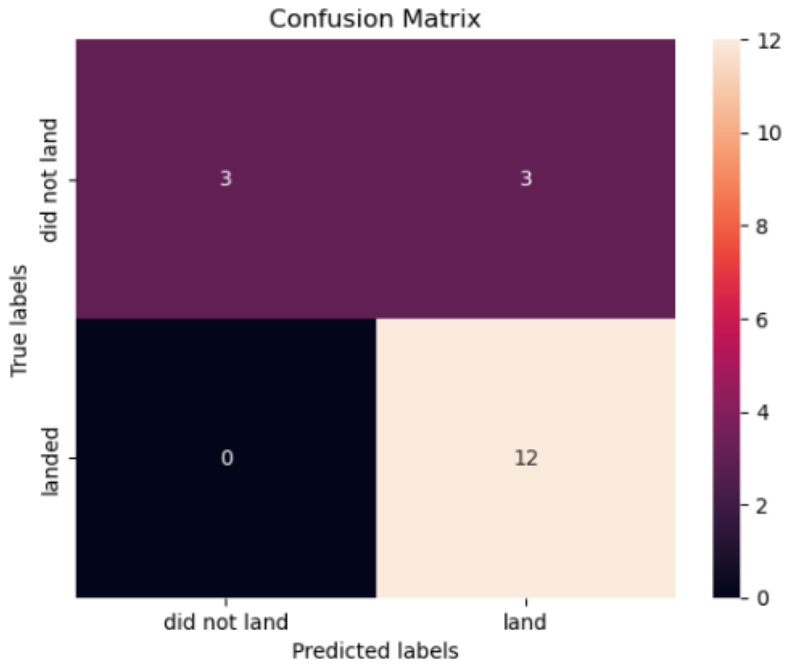
Predictive analysis using classification models

Method	Test Data Accuracy
Logistic_Reg	0.833333
SVM	0.833333
Decision Tree	0.833333
KNN	0.833333

Since all models performed similarly on the test set, the confusion matrix is identical across all models.

- The models correctly predicted **12 successful landings** when the true label indicated a successful landing.
- They also correctly predicted **3 unsuccessful landings** when the true label was unsuccessful.
- However, the models made **3 false positive predictions,** forecasting a successful landing when the actual outcome was unsuccessful.

Overall, the models tend to **overpredict successful landings,** which may indicate a bias that can be further addressed in future model tuning.



CONCLUSION

We built four ML models: Logistic Regression, SVM, Decision Tree, and KNN.

All models achieved an identical accuracy of 83.33% on the test data.

Orbits ES-L1, GEO, HEO and SSO have the highest success rates, while SO orbit has the lowest.

The success rate for the LEO orbit appears to be related to the number of flights.

APENDIX

GitHub repository

https://github.com/Kata10/Predicting-Falcon-9-First-Stage-Landing-Success