

# Winning Space Race with Data Science

M. Ryan Sanjaya  
15 January 2025



# Outline

---

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

# Executive Summary

---

- We analyse SpaceX data to better understand space transportation industry.
- This report uses various Python libraries to make sense of the data.
- We observe 67% successful landings

# Introduction

---

- Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch
- Given this, we would like to know important factors that contributes to a successful landing

Section 1

# Methodology

# Methodology

---

## Executive Summary

Data collection methodology:

- Data collected from SpaceX API

Perform data wrangling

- We generate variables that capture landing outcomes

Perform exploratory data analysis (EDA) using visualization and SQL

- Perform interactive visual analytics using Folium and Plotly Dash

• Perform predictive analysis using classification models

- We use machine learning techniques to predict landing outcome based on the
  - most accurate model we have

# Data Collection

---

We import data using SpaceX API from this URL: <https://api.spacexdata.com/v4/launches/past>

The imported data is normalized and turn it to dataframe using Pandas

From the loaded data we can get various information such as booster versions and landing outcome

# Data Collection – SpaceX API

---

- We utilize get request
- The GitHub URL of the completed SpaceX API calls notebook is: <https://github.com/mryansanjaya/capstone-spacex/blob/main/jupyter-labs-spacex-data-collection-api-v2.ipynb>

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
[4] spacex_url="https://api.spacexdata.com/v4/launches/past"
✓ 0.0s
```

```
[5] response = requests.get(spacex_url)
✓ 0.9s
```

Check the content of the response

```
[6] print(response.content)
✓ 0.0s
...
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},
```

# Data Collection - Scraping

- Again we use get request to obtain Falcon 9 HTML Wiki page and create BeautifulSoup object
- The GitHub URL of the completed web scraping notebook is: <https://github.com/mryansanjaya/capstone-spacex/blob/main/jupyter-labs-webscraping.ipynb>

To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the [List of Falcon Heavy launches](#).

```
[3] static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=102280607"
```

Next, request the HTML page from the above URL and get a `response` object

**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.

```
[4] ▶ # 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`

```
[5] # Use BeautifulSoup() to create a BeautifulSoup object from a response text content  
soup = BeautifulSoup(response)
```

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

```
[6] # Use soup.title attribute  
soup.title
```

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

# Data Wrangling

---

The data were processed using Pandas and Numpy libraries

We can get, for example, the number of launches on each site

The GitHub URL of data wrangling related notebook is: <https://github.com/mryansanjaya/capstone-spacex/blob/main/labs-jupyter-spacex-Data%20wrangling-v2.ipynb>

## TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 39A **KSC LC 39A**. The location of each Launch Is placed in the column

Next, let's see the number of launches for each site.

Use the method `value_counts()` on the column `LaunchSite` to determine the nu

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

[3]

```
    ✓ 0.0s  
...  
LaunchSite  
CCAFS SLC 40      55  
KSC LC 39A        22  
VAFB SLC 4E       13  
Name: count, dtype: int64
```

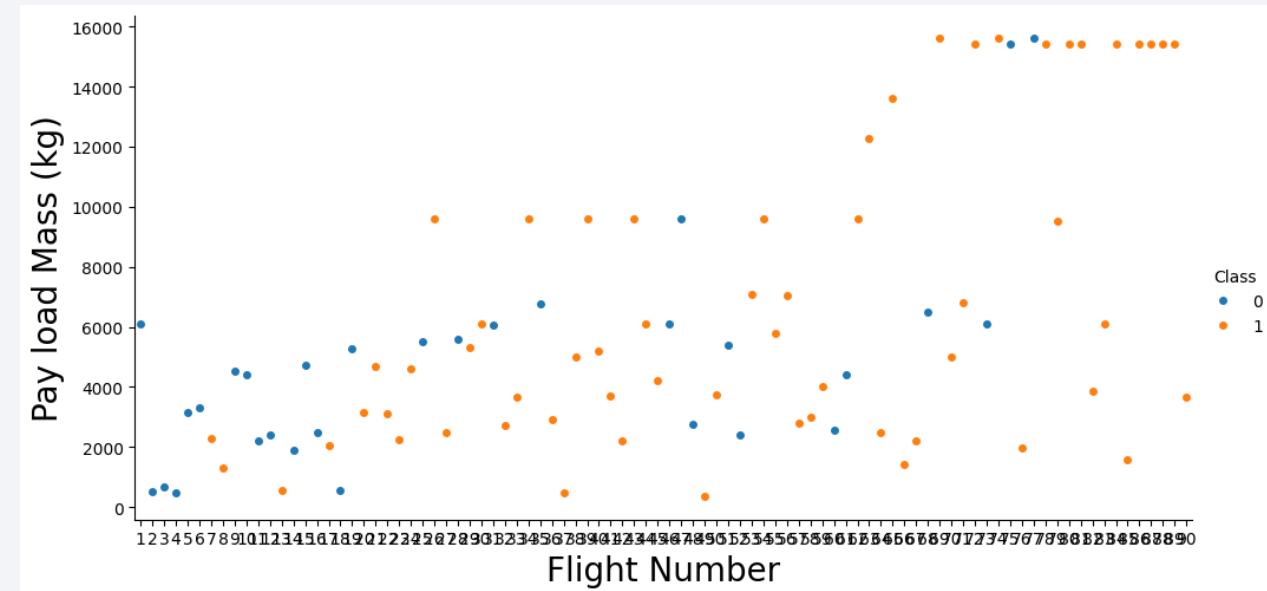
# EDA with Data Visualization

---

We use scatter plot of flight number and payload mass to make it easier to visualize the relationship between these variables

We find a positive relationship

The GitHub URL of EDA with data visualization notebook is: <https://github.com/mryansanjaya/capstone-spacex/blob/main/jupyter-labs-eda-dataviz-v2.ipynb>



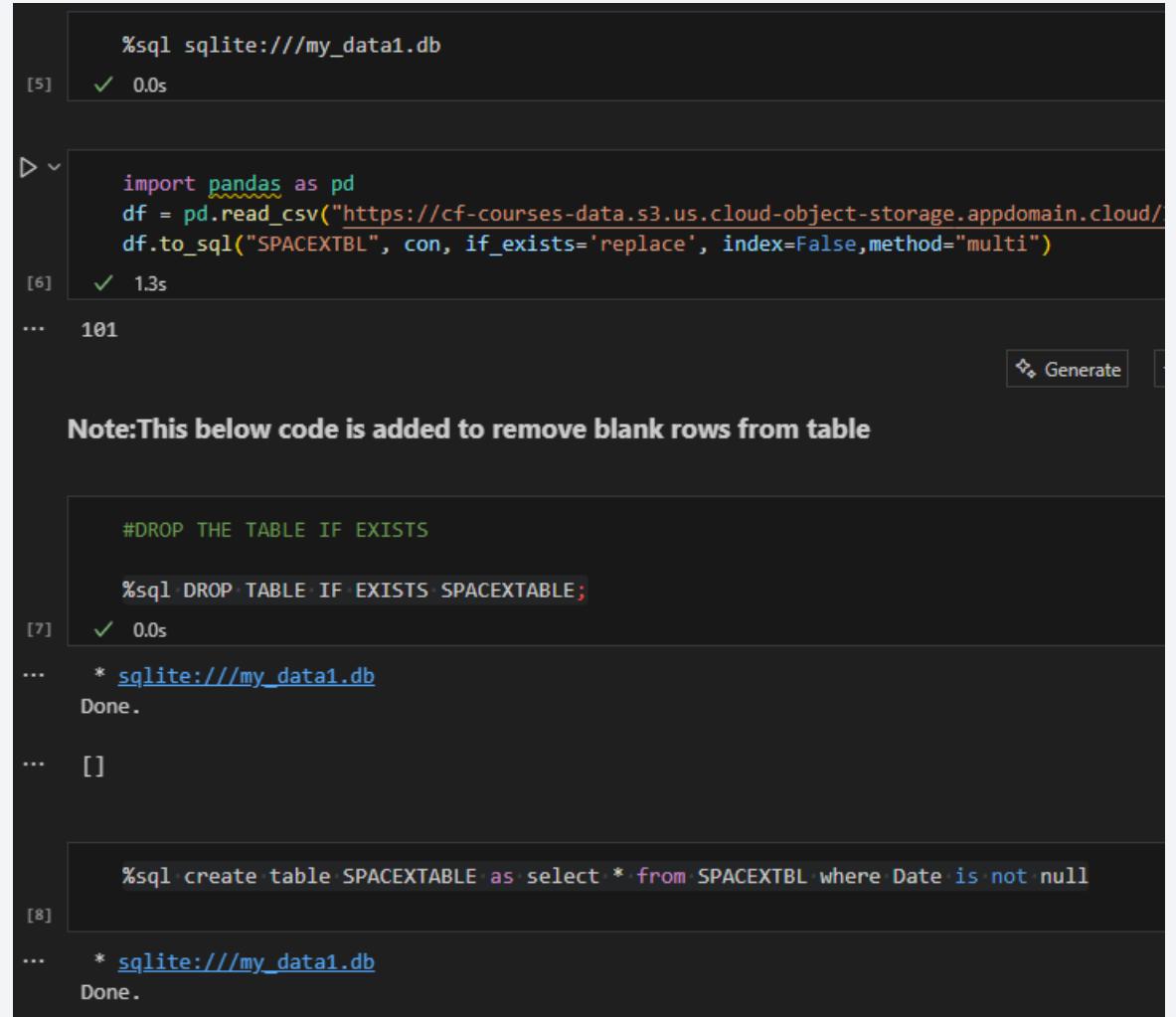
# EDA with SQL

SQL queries we did are:

```
%sql DROP TABLE IF EXISTS SPACEXTABLE;  
  
%sql create table SPACEXTABLE as select *  
from SPACEXTBL where Date is not null  
  
%sql SELECT * FROM SPACEXTBL LIMIT 20
```

And so on

The GitHub URL of EDA with SQL notebook is: [https://github.com/mryan-sanjaya/capstone-spacex/blob/main/jupyter-labs-eda-sql-coursera\\_sqlite.ipynb](https://github.com/mryan-sanjaya/capstone-spacex/blob/main/jupyter-labs-eda-sql-coursera_sqlite.ipynb)



A screenshot of a Jupyter Notebook interface showing two code cells. The first cell contains Python code to read a CSV file into a SQLite database:

```
%sql sqlite:///my_data1.db  
[5] ✓ 0.0s  
  
import pandas as pd  
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/...  
df.to_sql("SPACEXTBL", con, if_exists='replace', index=False, method="multi")  
[6] ✓ 1.3s  
... 101
```

The second cell contains a note and two more SQL commands:

Note: This below code is added to remove blank rows from table

```
#DROP THE TABLE IF EXISTS  
  
%sql DROP TABLE IF EXISTS SPACEXTABLE;  
[7] ✓ 0.0s  
... * sqlite:///my_data1.db  
Done.  
... []  
  
%sql create table SPACEXTABLE as select * from SPACEXTBL where Date is not null  
[8] ... * sqlite:///my_data1.db  
Done.
```

# Build an Interactive Map with Folium

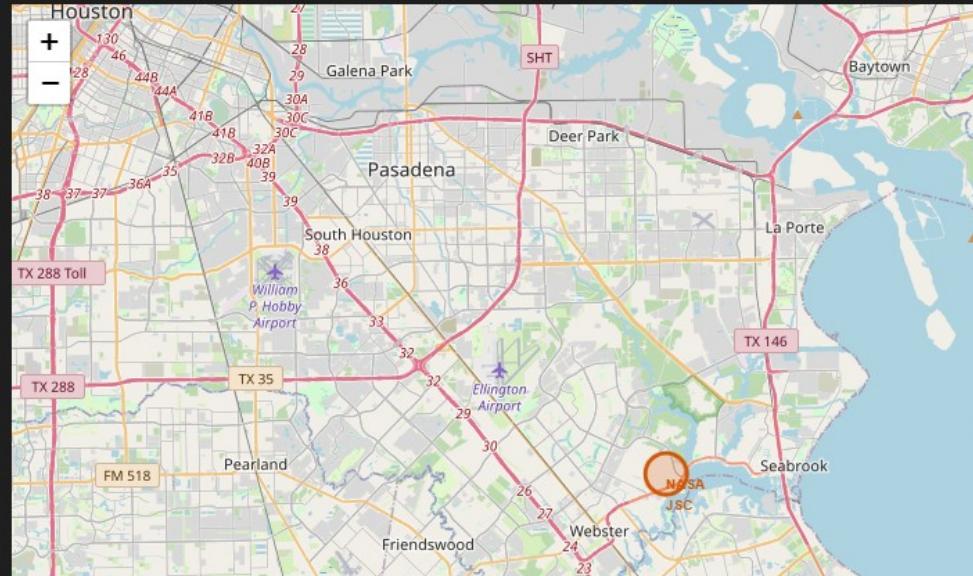
We use markers such as circle as map objects which is helpful to point out the important location

The GitHub URL of interactive map with Folium map is: <https://github.com/mryansanjaya/capstone-spacex/blob/main/lab-jupyter-launch-site-location-v2.ipynb>

We could use `folium.Circle` to add a highlighted circle area with a text label on a specific coordinate.

```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA JSC'))
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
    )
)
site_map.add_child(circle)
site_map.add_child(marker)
```

[8] ✓ 0.0s



The map displays the city of Houston, Texas, with various neighborhoods like Pasadena, South Houston, and Webster. A prominent blue circle highlights the location of NASA Johnson Space Center (JSC) in Seabrook. A text label "NASA JSC" is placed inside the circle. The map also shows major roads such as TX 288, TX 35, and FM 518, along with several airports including William P Hobby Airport and Ellington Airport.

# Build a Dashboard with Plotly Dash

---

Summarize what plots/graphs and interactions you have added to a dashboard

Explain why you added those plots and interactions

Add the GitHub URL of your completed Plotly Dash lab, as an external reference and peer-review purpose

# Predictive Analysis (Classification)

We built predictive models by first splitting data to train set and test set, then we use logistic regression and create the parameters dictionary

Best model is judged by using Grid Search technique

The GitHub URL of predictive analysis lab is: <https://github.com/mryansanjaya/capstone-spacex/blob/main/SpaceX-Machine-Learning-Prediction-Part-5-v1.ipynb>

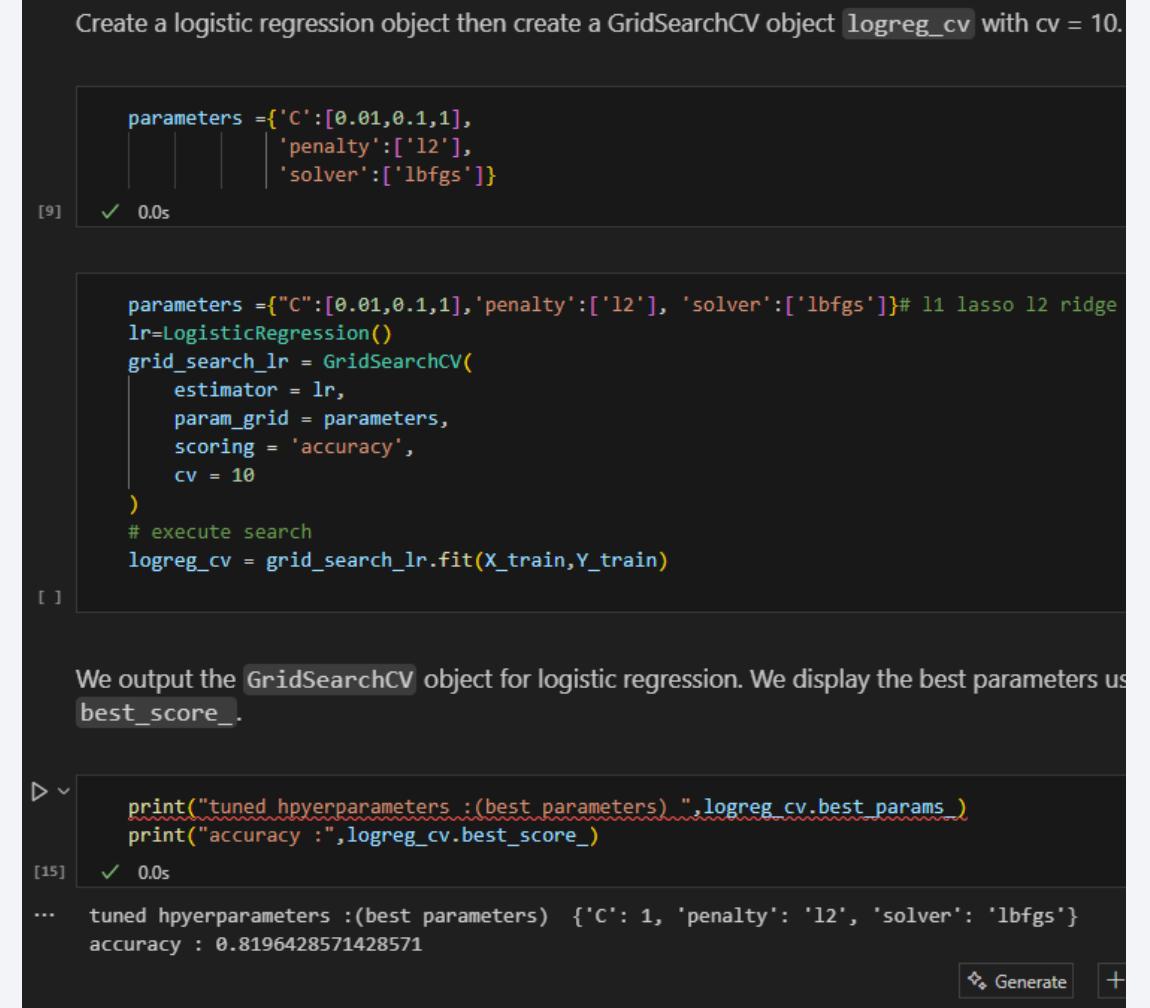
```
Create a logistic regression object then create a GridSearchCV object logreg_cv with cv = 10.

parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
[9]   ✓ 0.0s

parameters =[{"C": [0.01, 0.1, 1], "penalty": ["l2"], "solver": ["lbfgs"]}]# l1 lasso l2 ridge
lr=LogisticRegression()
grid_search_lr = GridSearchCV(
    estimator = lr,
    param_grid = parameters,
    scoring = 'accuracy',
    cv = 10
)
# execute search
logreg_cv = grid_search_lr.fit(X_train,Y_train)
[ ]
```

We output the GridSearchCV object for logistic regression. We display the best parameters using best\_params\_.

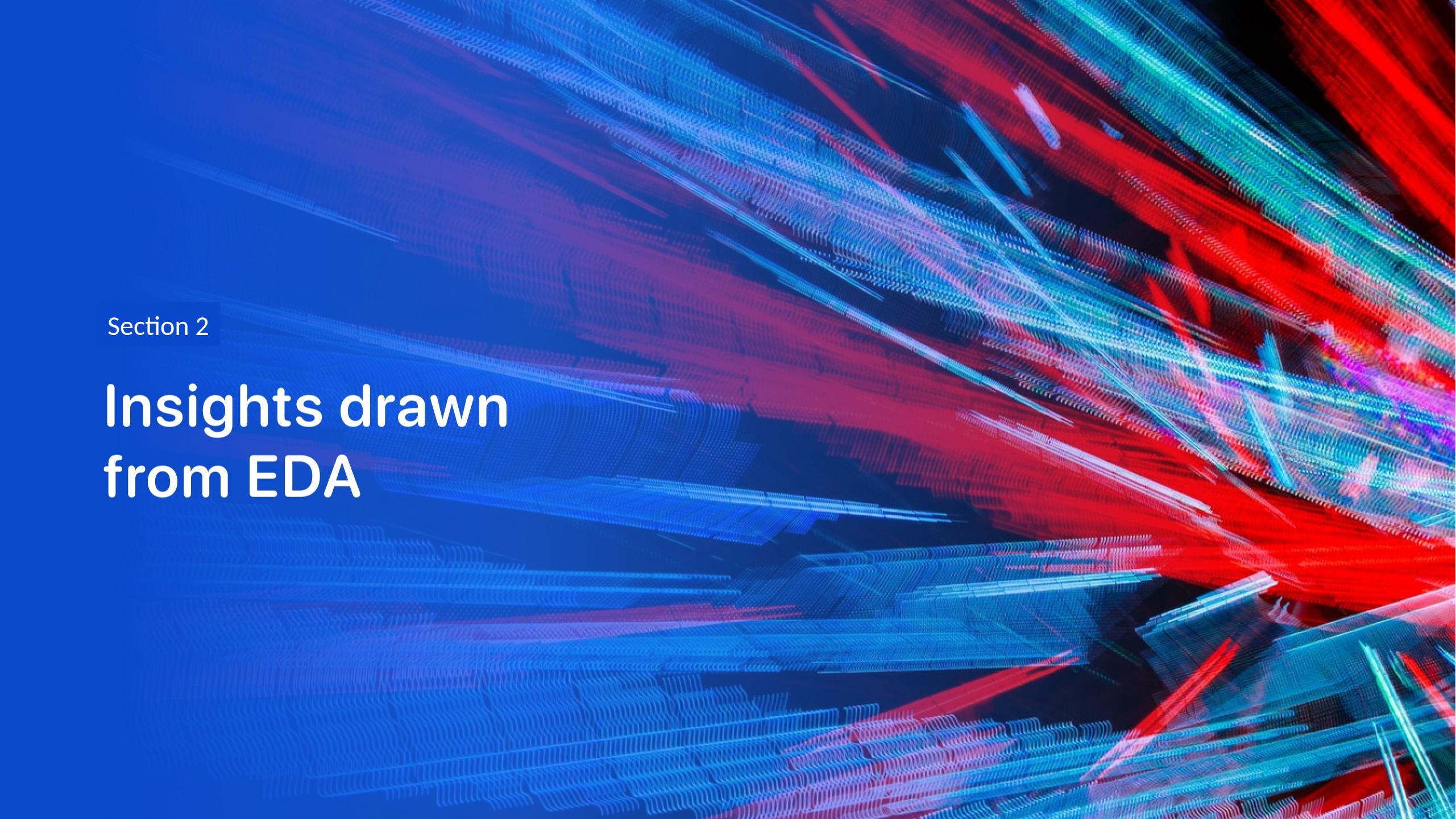
```
▷ ▾
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
[15]   ✓ 0.0s
... tuned hpyerparameters :(best parameters) {'C': 1, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8196428571428571
```



# Results

---

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

The background of the slide features a complex, abstract pattern of glowing lines. These lines are primarily blue and red, creating a sense of depth and motion. They appear to be composed of numerous small, individual points or pixels, giving them a granular texture. The lines curve and twist in various directions, some converging towards the center of the frame while others recede into the distance. The overall effect is reminiscent of a digital or quantum landscape.

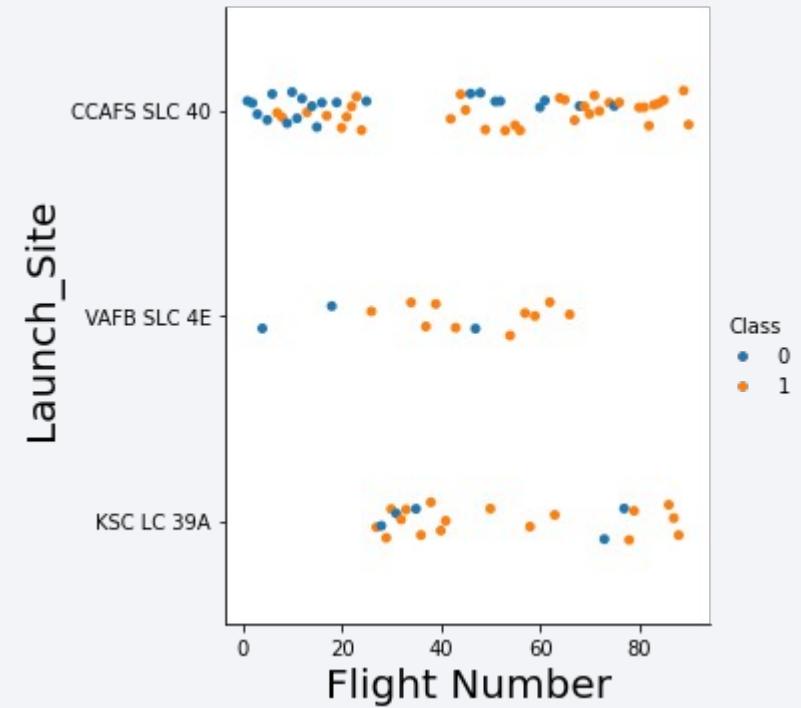
Section 2

## Insights drawn from EDA

# Flight Number vs. Launch Site

---

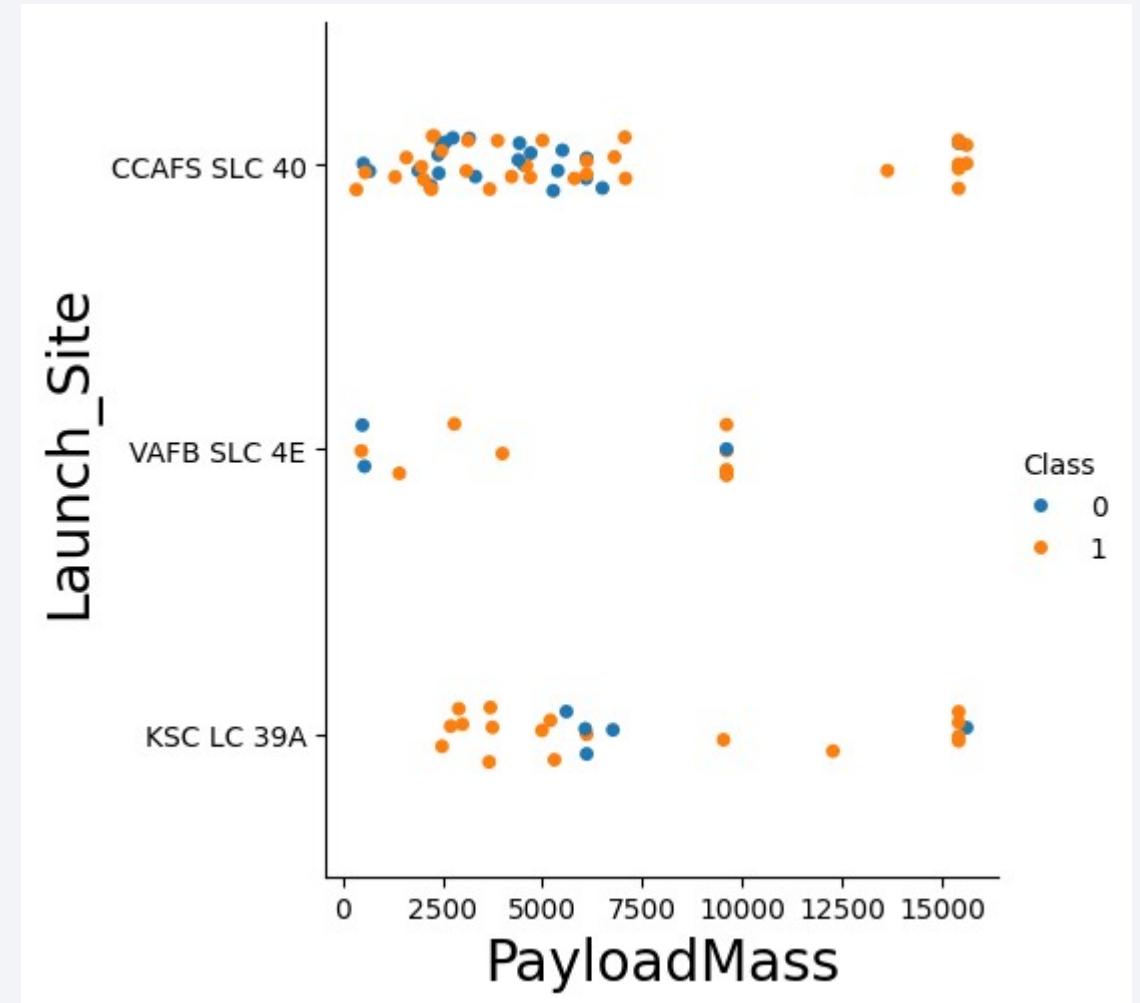
- We use scatter plot of flight number and launch site to make it easier to visualize the relationship between these variables
- We find a positive relationship: in any launch site, most recent flights tend to be much more successful than early flights



# Payload vs. Launch Site

---

- Show a scatter plot of Payload vs. Launch Site
- Large payload mass is more successful in recent flights



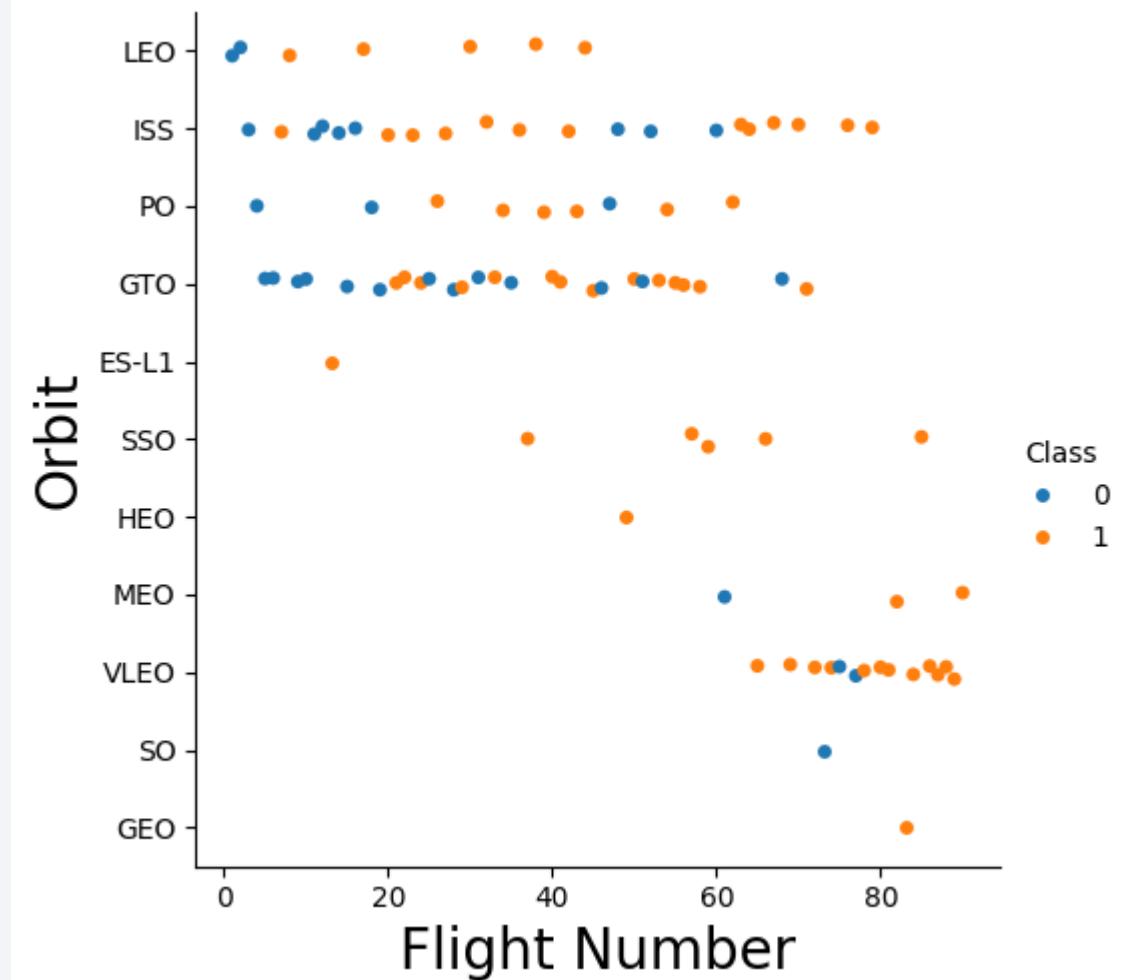
# Success Rate vs. Orbit Type

---

- Show a bar chart for the success rate of each orbit type
- Show the screenshot of the scatter plot with explanations

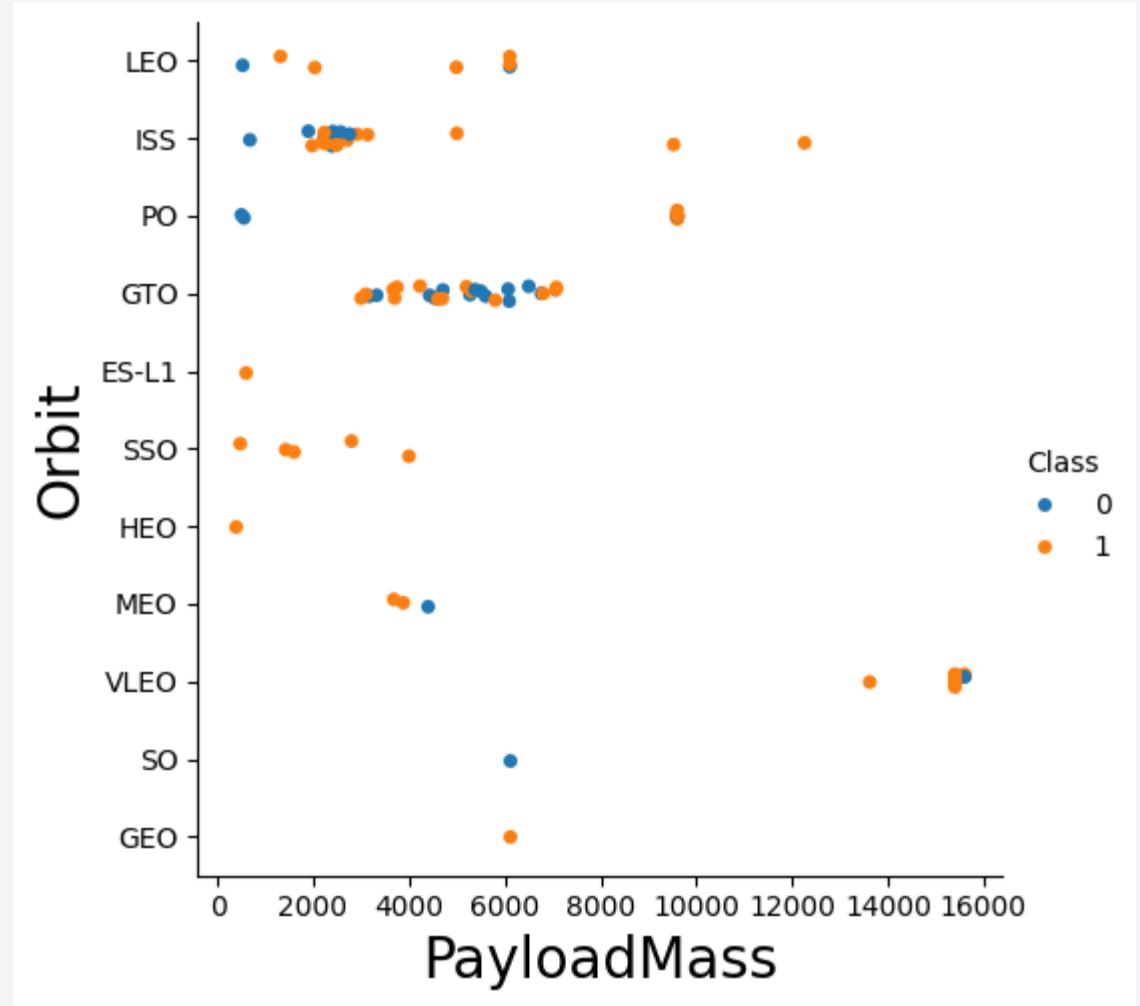
# Flight Number vs. Orbit Type

- Show a scatter point of Flight number vs. Orbit type
- In all orbit type, recent flights are more successful
- ES-L1, SSO, HEO, and GEO has the highest success rate but maybe because they only have few observations



# Payload vs. Orbit Type

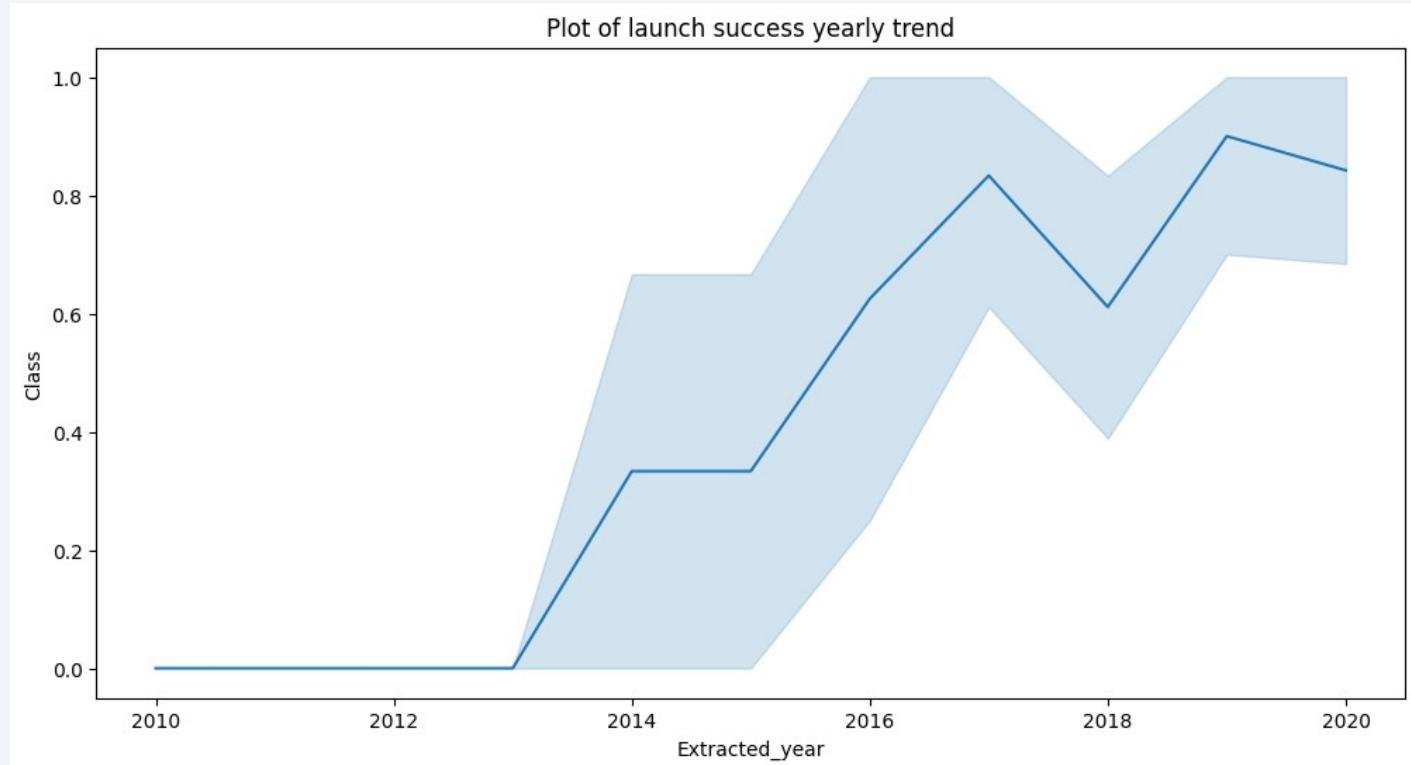
- Show a scatter point of payload vs. orbit type
- There's unclear relationship



# Launch Success Yearly Trend

---

- Show a line chart of yearly average success rate
- Success rate increases almost in every year



# All Launch Site Names

---

Find the names of the unique launch sites:

CCAFS LC-40

VAFB SLC-4E

KSC LC-39A

CCAFS SLC-40

We use: `SELECT DISTINCT Launch_Site FROM SPACEXTBL`

# Launch Site Names Begin with 'CCA'

---

We use this command:

```
SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

```
[9] %sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5
[9] ✓ 0.0s
...
... * sqlite:///my\_data1.db
Done.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2

# Total Payload Mass

---

We use this code:

```
SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass FROM  
SPACEXTBL WHERE Customer LIKE 'NASA (CRS)'
```

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass FROM SPACEXTBL WHERE Customer LIKE 'NASA (CRS)'  
[12]    ✓ 0.0s  
... * sqlite:///my\_data1.db  
Done.  
...  
Total_PayloadMass  
45596
```

# Average Payload Mass by F9 v1.1

---

We use this code:

```
SELECT AVG(PAYLOAD_MASS_KG_) AS Avg_PayloadMass FROM  
SPACEXTBL WHERE Booster_Version='F9 v1.1'
```

The screenshot shows a terminal window with the following output:

```
%sql SELECT AVG(PAYLOAD_MASS_KG_) AS Avg_PayloadMass FROM SPACEXTBL WHERE Booster_Version='F9 v1.1'  
[13] ✓ 0.0s  
... * sqlite:///my_data1.db  
Done.  
... Avg_PayloadMass  
2928.4
```

The command is executed successfully in 0.0s. The result is displayed in a table with one row, showing the average payload mass as 2928.4.

# First Successful Ground Landing Date

---

The code is:

```
SELECT MIN(Date) AS First_Success FROM SPACEXTBL WHERE  
Landing_Outcome='Success (ground pad)'
```

A screenshot of a terminal window showing the execution of an SQL query. The command entered is:

```
%sql SELECT MIN(Date) AS First_Success FROM SPACEXTBL WHERE Landing_Outcome='Success (ground pad)'
```

The output shows the result of the query:

```
[18]    ✓ 0.0s  
... * sqlite:///my\_data1.db  
Done.  
... First_Success  
2015-12-22
```

The result is highlighted with a gray box, showing the column name "First\_Success" and its value "2015-12-22".

## Successful Drone Ship Landing with Payload between 4000 and 6000

---

The code is:

```
SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome =  
'Success (drone ship)' AND PAYLOAD_MASS_KG_ > 4000 AND  
PAYLOAD_MASS_KG_ < 6000
```

```
[20] %sql SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS_KG_ > 4000 AND PAYLOAD_MASS_KG_ < 6000  
✓ 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

---

```
%sql SELECT COUNT(Mission_Outcome) AS Outcome_Success FROM SPACEXTBL WHERE Mission_Outcome LIKE 'Success%'  
[21] ✓ 0.0s  
... * sqlite:///my\_data1.db  
Done.  
  
... Outcome_Success  
100  
  
▷ %sql SELECT COUNT(Mission_Outcome) AS Outcome_Failure FROM SPACEXTBL WHERE Mission_Outcome LIKE 'Failure%'  
[22] ✓ 0.0s  
... * sqlite:///my\_data1.db  
Done.  
  
... Outcome_Failure  
1
```

# Boosters Carried Maximum Payload

---

List the names of the booster which have carried the maximum payload mass:

F9 B5 B1048.4

F9 B5 B1048.5

F9 B5 B1049.4

F9 B5 B1049.5

F9 B5 B1049.7

F9 B5 B1051.3

F9 B5 B1051.4

F9 B5 B1051.6

F9 B5 B1056.4

F9 B5 B1058.3

F9 B5 B1060.2

F9 B5 B1060.3

# 2015 Launch Records

---

List of failed landing\_outcomes in drone ship, their booster versions, and launch site names for in year 2015

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

Both failures happen in CCAFS LC-40 launch site.

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

---

This table shows the count of landing outcome from 2010-06-04 to 2017-03-20.

“No attempt” occurred most often.

Landing_Outcome	COUNT(Landing_Outcome)
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

The background of the slide is a photograph taken from space at night. It shows the curvature of the Earth against the dark void of space. City lights are visible as numerous small white and yellow dots, primarily concentrated in the lower right quadrant where the United States appears. In the upper left quadrant, the green and blue glow of the aurora borealis (Northern Lights) is visible in the upper atmosphere.

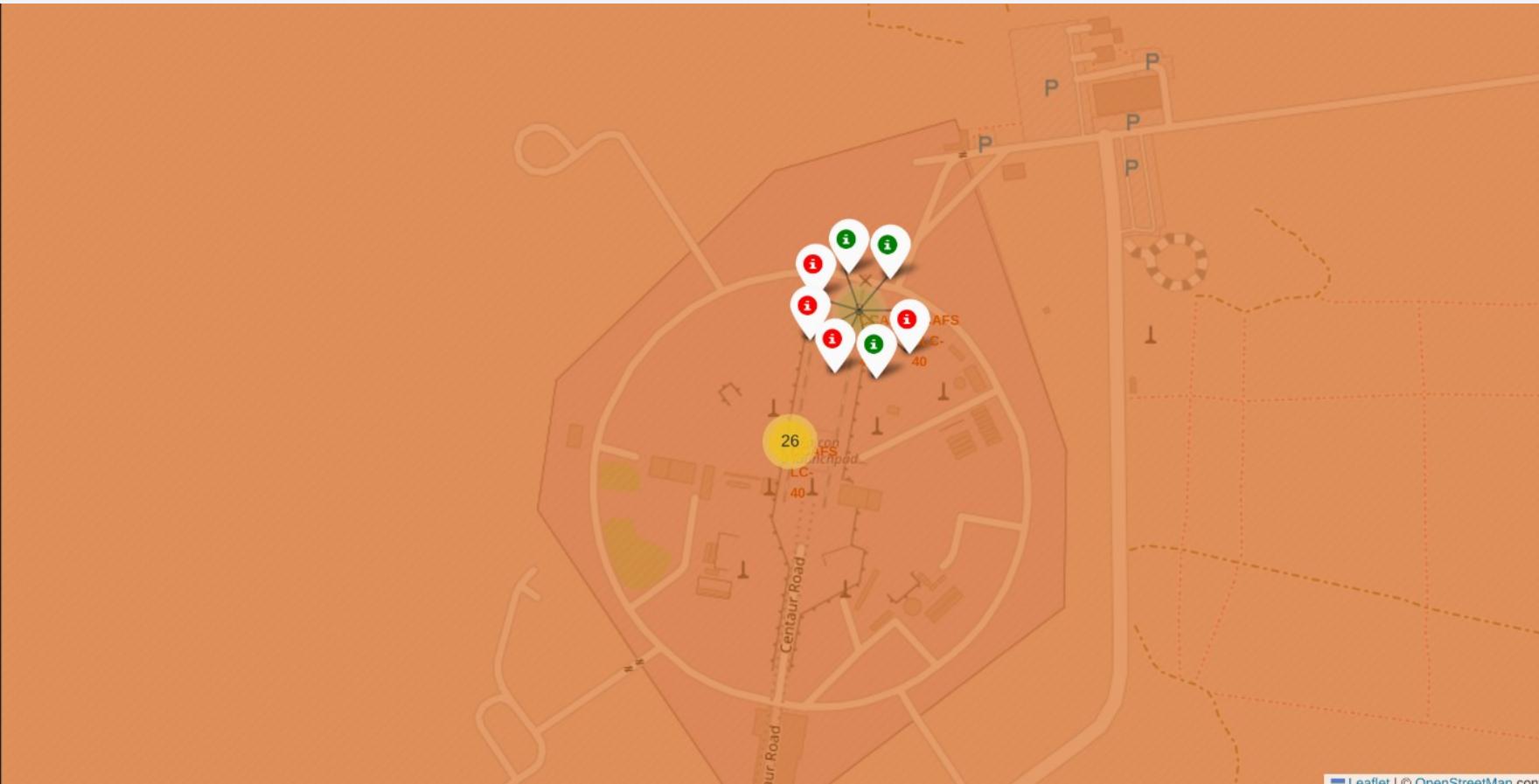
Section 3

# Launch Sites Proximities Analysis

# Launch Site Locations



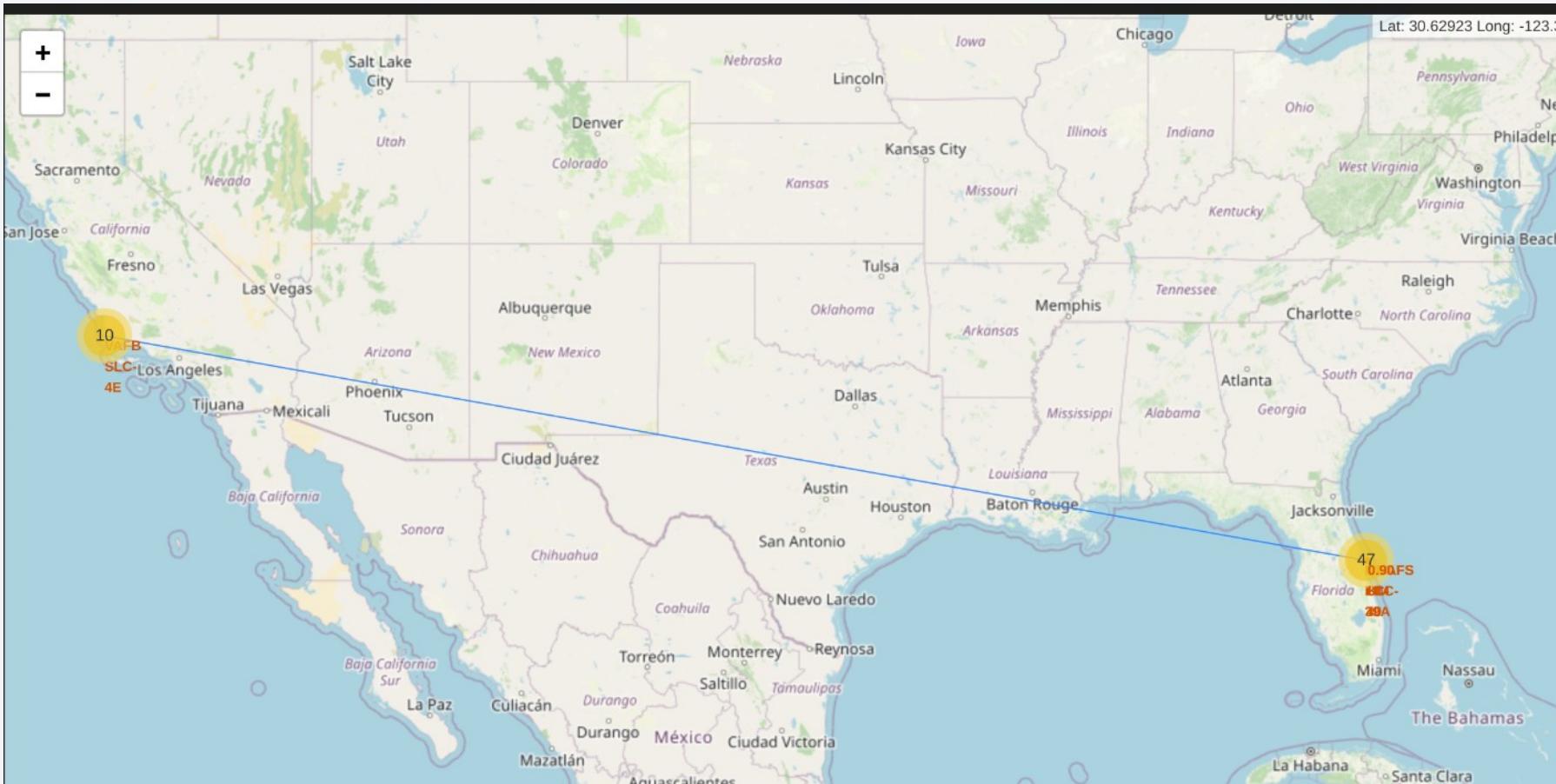
# Launch Sites and Outcomes



This is the launch outcomes from two locations in the East Coast.

Green means success, red means failure

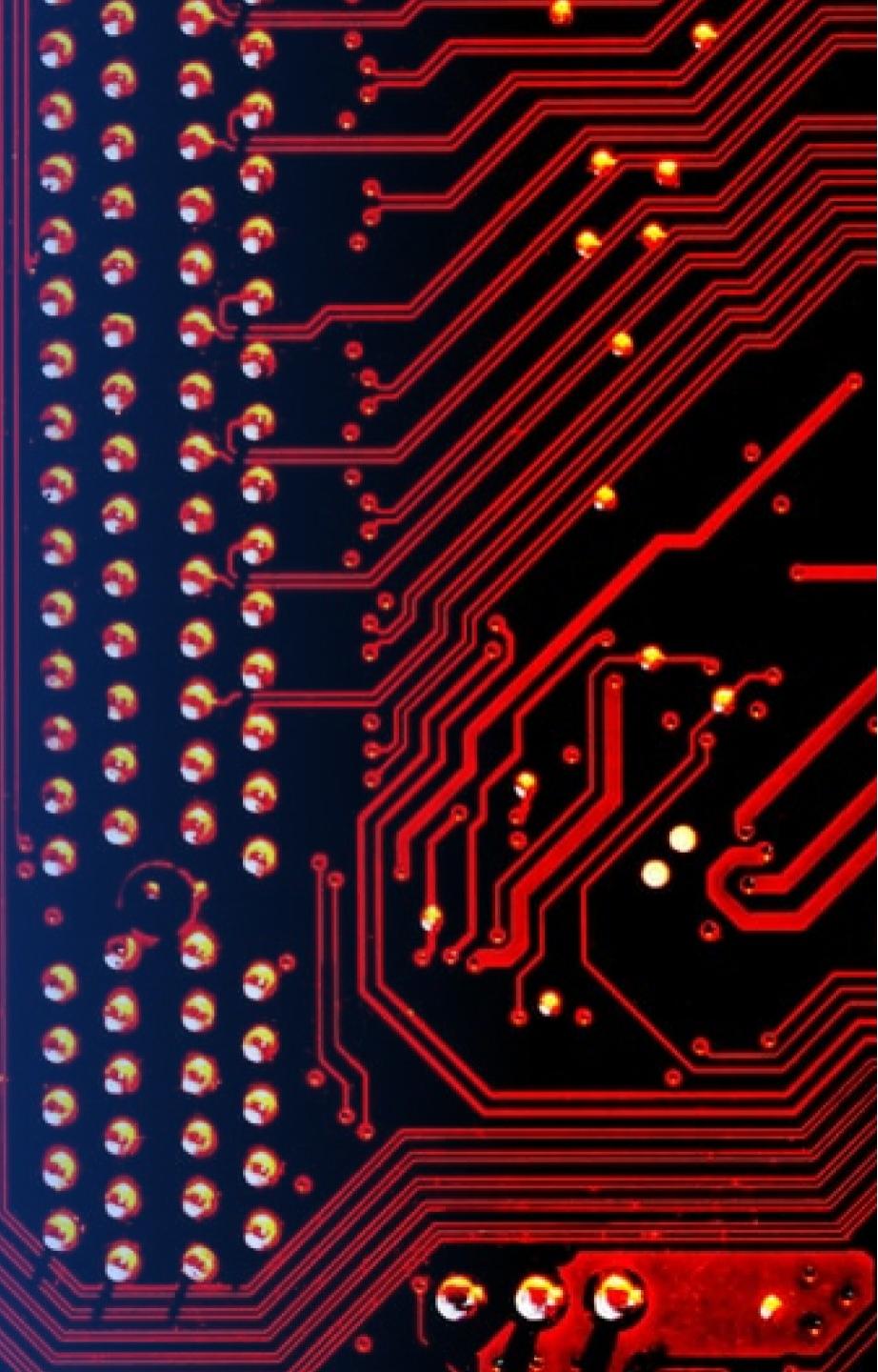
# <Folium Map Screenshot 3>



The distance between the two sites are quite far!

Section 4

# Build a Dashboard with Plotly Dash



# <Dashboard Screenshot 1>

---

Replace <Dashboard screenshot 1> title with an appropriate title

Show the screenshot of launch success count for all sites, in a piechart

Explain the important elements and findings on the screenshot

## <Dashboard Screenshot 2>

---

Replace <Dashboard screenshot 2> title with an appropriate title

Show the screenshot of the piechart for the launch site with highest launch success ratio

Explain the important elements and findings on the screenshot

# <Dashboard Screenshot 3>

---

Replace <Dashboard screenshot 3> title with an appropriate title

Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
grid_search_lr = GridSearchCV(
    estimator = lr,
    param_grid = parameters,
    scoring = 'accuracy',
    cv = 10
)
# execute search
logreg_cv = grid_search_lr.fit(X_train,Y_train)
[11] ✓ 7.2s
Outputs are collapsed ...
```

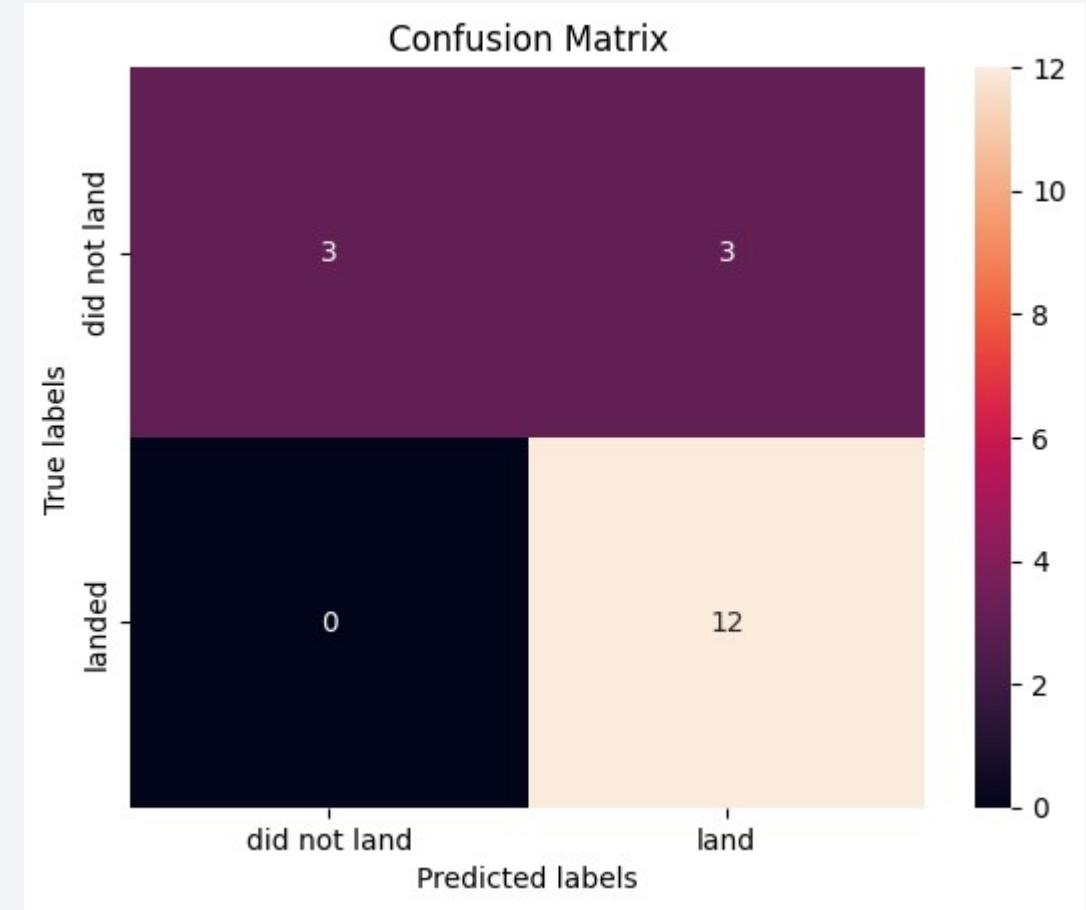
We output the `GridSearchCV` object for logistic regression. We display the best parameters using the data attribute `best_params_` and the validation data using the data attribute `best_score_`.

```
print("tuned hpyerparameters :(best parameters) ",logreg_cv.best_params_)
print("accuracy :",logreg_cv.best_score_)
[12] ✓ 0.0s
... tuned hpyerparameters :(best parameters) {'C': 1, 'penalty': 'l2', 'solver': 'lbfgs'}
accuracy : 0.8196428571428571
```

# Confusion Matrix

---

- We saw some misclassification, where there are 3 launches that failed to land but were predicted to land



# Conclusions

---

- We find a positive relationship: in any launch site, most recent flights tend to be much more successful than early flights
- ES-L1, SSO, HEO, and GEO has the highest success rate but maybe because they only have few observations
- Success rate increases almost in every year
- We saw some misclassification in the predictive model, where there are 3 launches that failed to land but were predicted to land

Thank you!

