IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

MONICA CHECA

AGOSTO 2024

# Outline

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

# Executive Summary

- Summary of methodologies
  - Data Collection though API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Visual Analytics with Folium (interactive)
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis Results
  - Interactive analytics with screenshots
  - Predictive Analytics results

# Introduction

- Project background and context

  The goal of the project is to create a model, trained with machine learning, capable of predict in the fist stage of a space rocket will land successfully, because there is a big difference between the cost published in the website of SpaceX and other providers.

  SpaceX tell us about the cost of 62 million dollars with its Falcon 9 rocket launches meanwhile the cost published by others providers is about 165 millions dollars. A lot of the savings is because the reuse of the first stage in SpaceX launches, so, if we can determine if this first stage will land, we probably can determine the cost of a launch.

- Problems you want to find answers

  - First of all, we need to know what factors determine if the rocket will land successfully.
  - What is the relation between various features that determine the success rate in the landing?
  - Which conditions are importants in the operative to success in landing?

Section 1

# Methodology

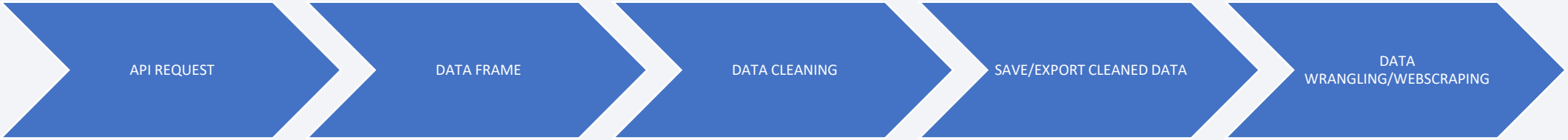
# Methodology

## Executive Summary

- Data collection methodology:
  - Data collection was from wiki dataset through SpaceX API and Webscraping.
- Perform data wrangling
  - It was used one-hot encoding to categorical 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
  - How to build, tune, evaluate classification models

# Data Collection

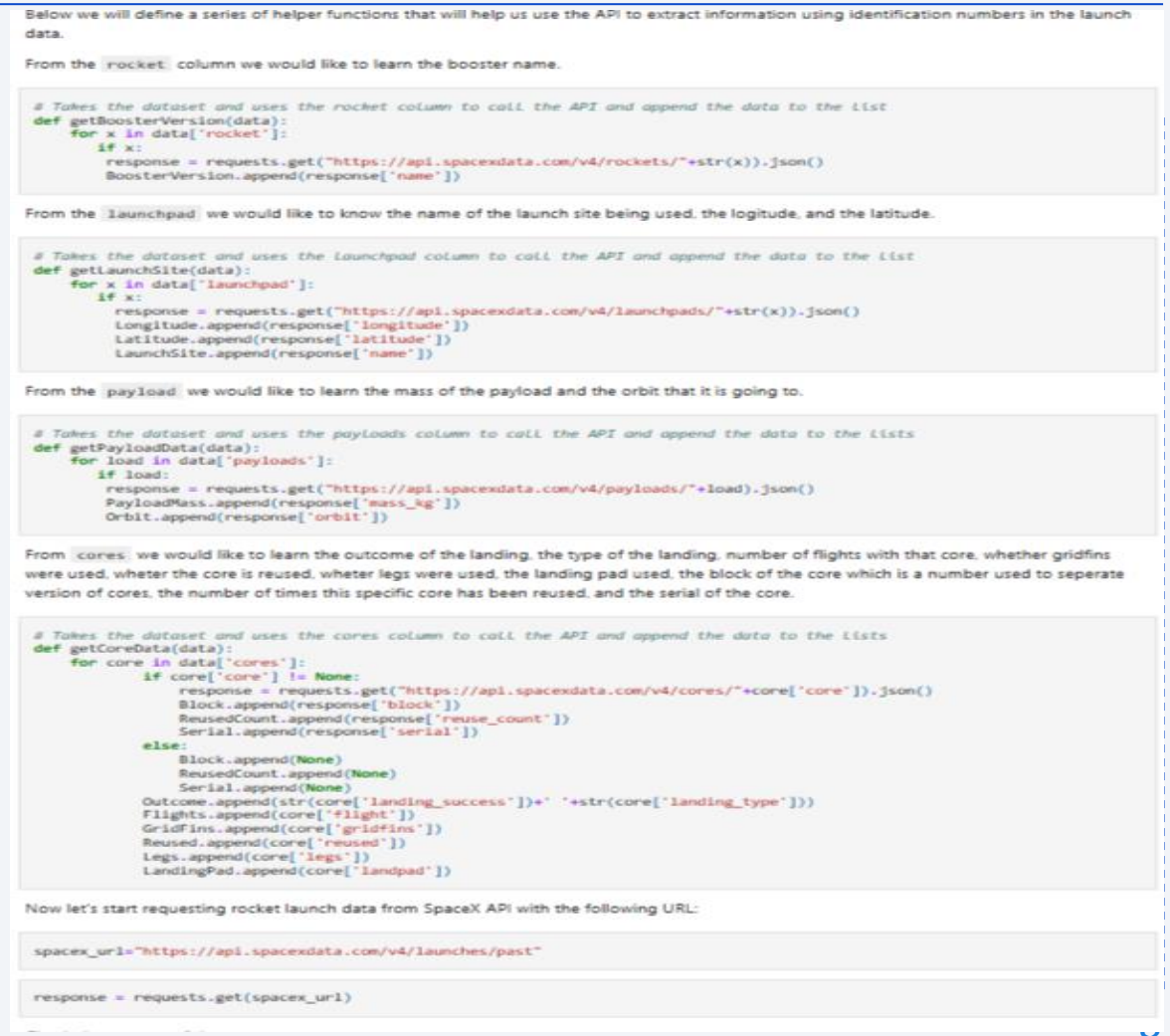
- Data sets were collected using the request into SpaceX API
- After get the data set, we turn it into a pandas dataframe and normalized and cleaned the requested data, checked for missing values with .isnull() and replaced with .replace() when was necessary.
- It was fundamental the use of pandas and sqlite in the jupyter notebooks to realize all the task.
- We used data wrangling and webscraping with BeautifulSoup in Wikipedia for search Falcon 9 records.



# Data Collection – SpaceX API

- We used the get request in SpaceX API to collect the data requested, clean, and we did data wrangling and formatting some data.

- The GitHub URL of the completed SpaceX notebook is: [SpaceXCapstone/jupyter-labs-spacex-data-collection-api.ipynb at main · MonChe5/SpaceXCapstone (github.com)]



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.

From the `rocket` column we would like to learn the booster name.

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
            BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
        if x:
            response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
            Longitude.append(response['longitude'])
            Latitude.append(response['latitude'])
            LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
        if load:
            response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
            PayloadMass.append(response['mass_kg'])
            Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

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

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

# Data Collection - Scraping

- We perform an HTTP GET method to request the Falcon 9 Launch HTML page and create a BeautifulSoup object.
- Then, we parse the table and convert it to a pandas dataframe.
- The GitHub URL of the completed web scraping notebook, is: [SpaceXCapstone/jupyter-labs-webscraping.ipynb at main · MonChe5/SpaceXCapstone (github.com)]

### 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
html_data = requests.get(static_url)
html_data.status_code
```

```
200
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text)
```

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

```
# Use soup.title attribute
soup.title
```

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

# Data Wrangling

- We did some EDA to find some patterns in the data: calculate the number of launches in each site and the number and occurrence in each orbit(with .value_counts() method),
- Then, we calculated the number and occurrence  of mission outcome with no success and labeled using Outcome, finally we determine the success rate.
- You need to present your data wrangling process using key phrases and flowcharts
- This the GitHub URL of my data wrangling related notebook: [SpaceXCapstone/labs-jupyter-spacex-Data wrangling.ipynb at main · MonChe5/SpaceXCapstone (github.com)]

**TASK 3: Calculate the number and occurence of mission outcome of the orbits**

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes` .Then assign it to a variable landing_outcomes.

```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

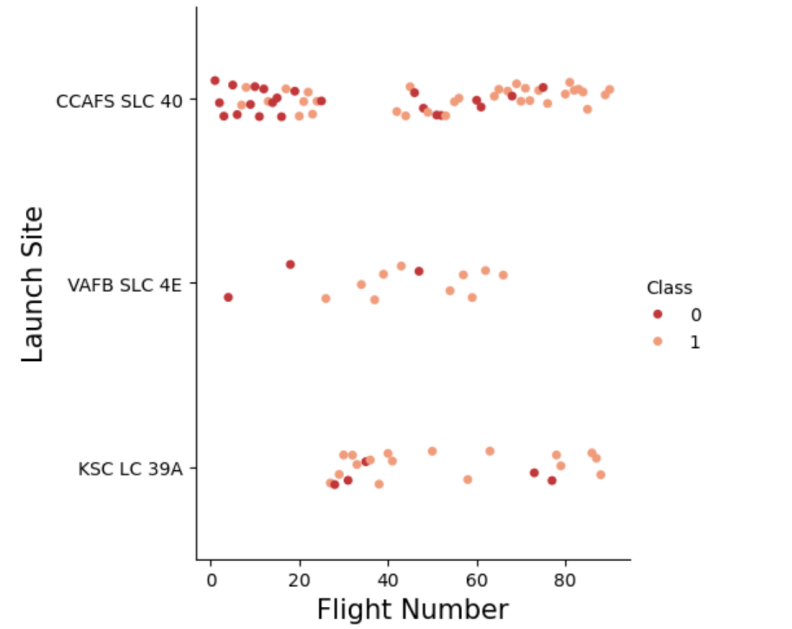
```
True ASDS       41
None None       19
True RTLS       14
False ASDS       6
True Ocean       5
False Ocean      2
None ASDS        2
False RTLS       1
Name: Outcome, dtype: int64
```

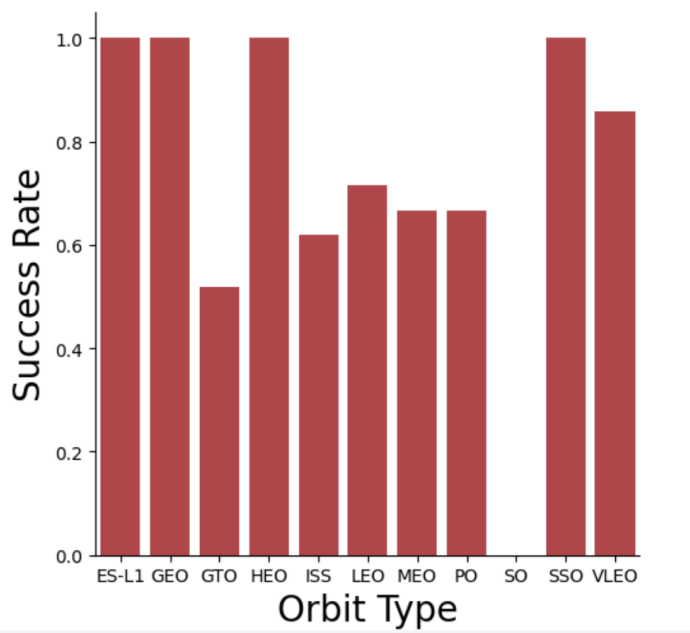
`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

# EDA with Data Visualization

- We used matplotlib and seaborn for visual representations, very important to see relationships between different features: scatter point plots and bart chart clarified a lot the launch site and the orbits success rate and the line chart to see the success rate through the years.

- The GitHub URL of my completed EDA is: [SpaceXCapstone/edadataviz (1).ipynb at main · MonChe5/SpaceXCapstone (github.com)]



# EDA with SQL

- Once connect to the data base, the SQL queries that I performed were:
  - Names of Unique Launch sites in the space mission
  - Five records where launch sites begin with string '**CCA**'
  - Total payload mass carried by booster launched by **NASA (CRS)**
  - Average payload mass carried by booster version **F9 v1.1**
  - Date when the first  successful landing outcome in ground pad was achieved.
  - List of success in drone ship with payload mass **>4000 but <6000**
  - Total number of successful and failure mission outcomes.
  - List of Booster versions with maximum payload mass.
  - List of Month names of failure landing outcomes in dron ships in year 2015.
  - 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.
- The GitHub URL of my completed EDA with SQL notebook is: SpaceXCapstone/jupyter-labs-eda-sql-coursera_sqllite (1).ipynb at main · MonChe5/SpaceXCapstone (github.com)

# Build an Interactive Map with Folium

- To make more interactive the map with Folium, I added:
  - A Marker with folium.circle with a pop up label to pin all specific coordenates for each launch sites on a map
  - A Marker color with the success/failed launches for each site on the map
  - A Mouse position to get the coordenates over the map
  - A Distance Marker line To Calculate the distances between a launch site to its proximities

- The GitHub URL of my completed interactive map with Folium map is:

SpaceXCapstone/lab_jupyter_launch_site_location (1).ipynb at main · MonChe5/SpaceXCapstone (github.com)

# Build a Dashboard with Plotly Dash

- I did a Dashboard with Plotly Dash in which I indroduced a dropdown list and a range slider to allow interaction with the pie chart and the scatter point chart that I generated.


- The GitHub URL of my Plotly Dash lab is:

SpaceXCapstone/spacex_dash_app.py at main · MonChe5/SpaceXCapstone (github.com)

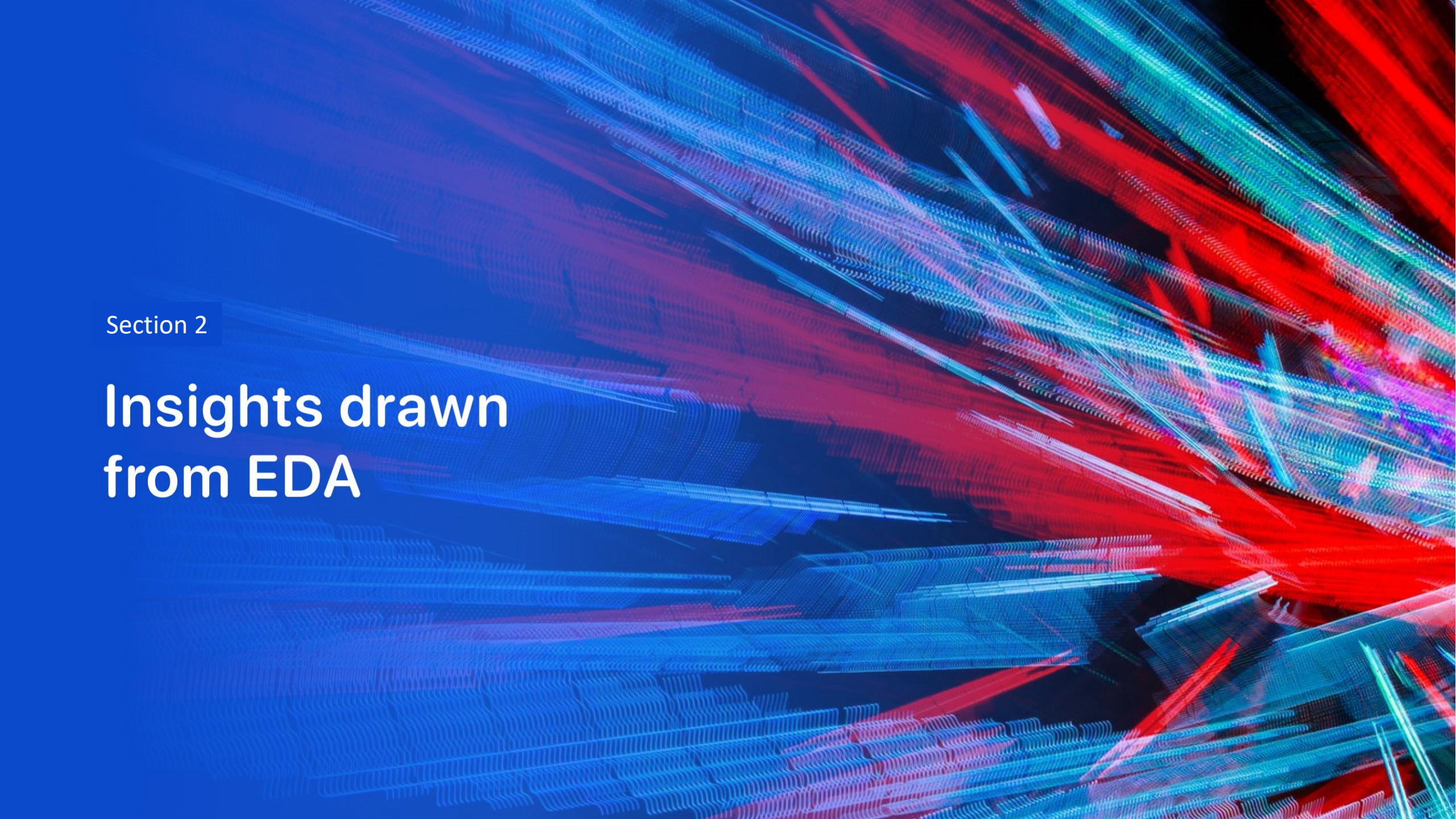
# Predictive Analysis (Classification)

- The cleaned data was obtained, and it was assigned the features to a variable X, standardized with preprocessing.StandardScaler and target to the variable Y by creating a numpy array with to_numpy. I scaled the features with sklearn scaler().
- Then I used “train_test_split” with an initial 20% to testing and first, I created a logistic regression object to fit in.
- I used the score method to calculate the accuracy of the different models (decision tree, K nearest neighbour, Support Vector Machine) in the test data and see the results in a confusion matrix.


- The GitHub URL is: SpaceXCapstone/SpaceX_Machine Learning Prediction_Part_5 (1).ipynb at main · MonChe5/SpaceXCapstone (github.com)

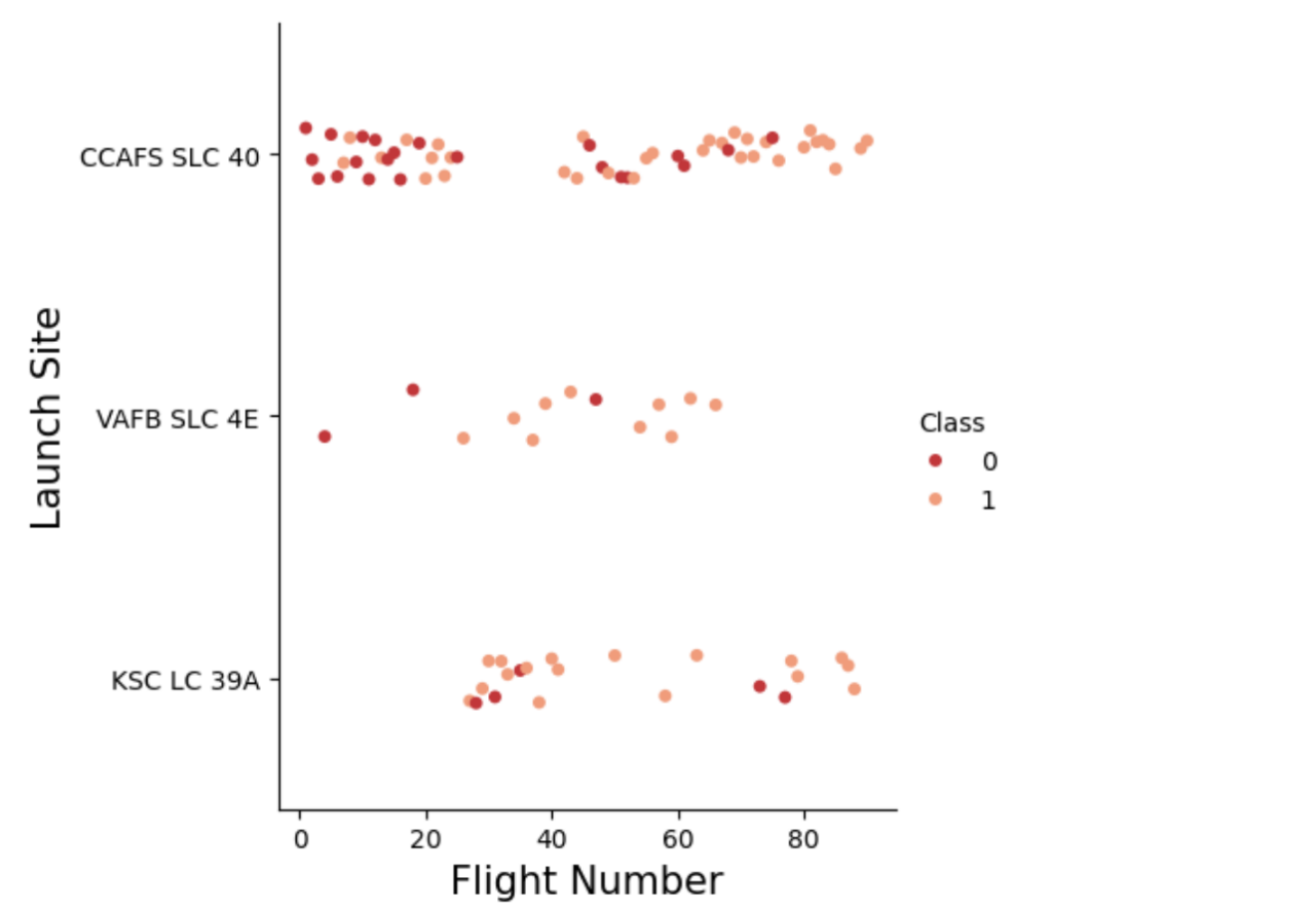
# Results

- Exploratory data analysis results NASA (CRS) has a total “payload mass” of 45.596 kg, average payload_mass of 2928.4 kg in ‘booster version F9 v1.1’; first landing outcome in land with success in ‘2015-12-22’; there are 5 booster versions: JCSAT-14,JCSAT-16,SES-10,SES-11 / EchoStar 105with a succesfull landing on drone ship (with payload_mass between 4000 and 6000 kg), and a total of 99 success mission_outcome from SpaceX F9.
- Interactive analytics demo in screenshots show higher success rate for CCAFS SLC 40 in the increase of flights, higher success rate in Orbit type “ES-L1,ESO,HEO and GEO of Falcon F9 first stage landing over time (2010-2020)
- Predictive analysis results:
  - Accuracy for Logistics Regression method: 0.8333333333333334
  - Accuracy for Support Vector Machine method: 0.8333333333333334
  - Accuracy for Decision tree method: 0.7222222222222222
  - Accuracy for K nearest neighbours method: 0.8333333333333334

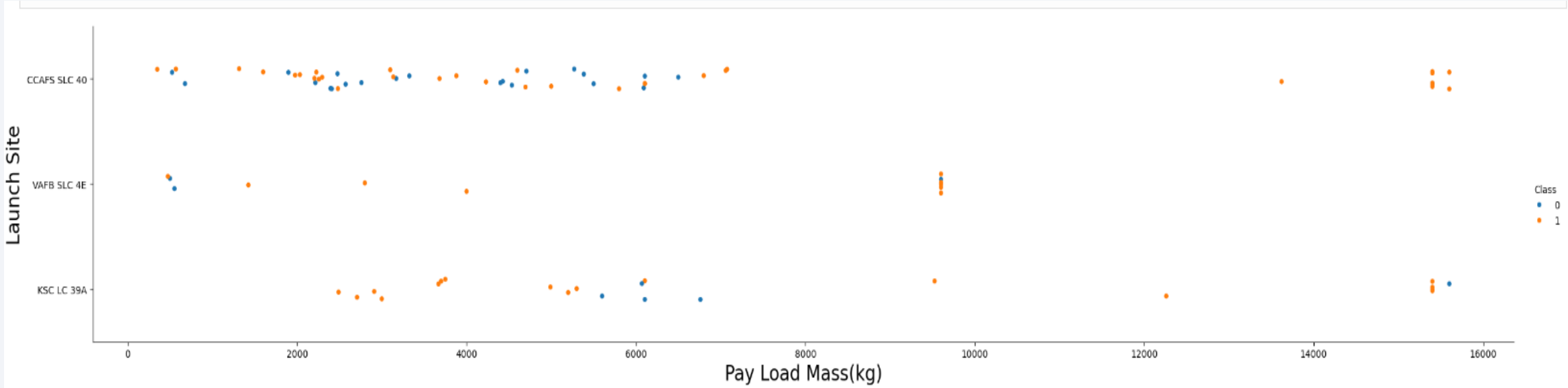

Section 2
Insights drawn from EDA

# Flight Number vs. Launch Site

- The relationship between the Flight Number vs. Launch Site is that with lower flight number, as 20, there's no failure or success for KSC LC 39A, but 2 cases of failure y VAFB SLC 4E.
- If the flight number increases over 80, there are more success in CCAFS SLC 40 and none in VAFB SLC 4E.



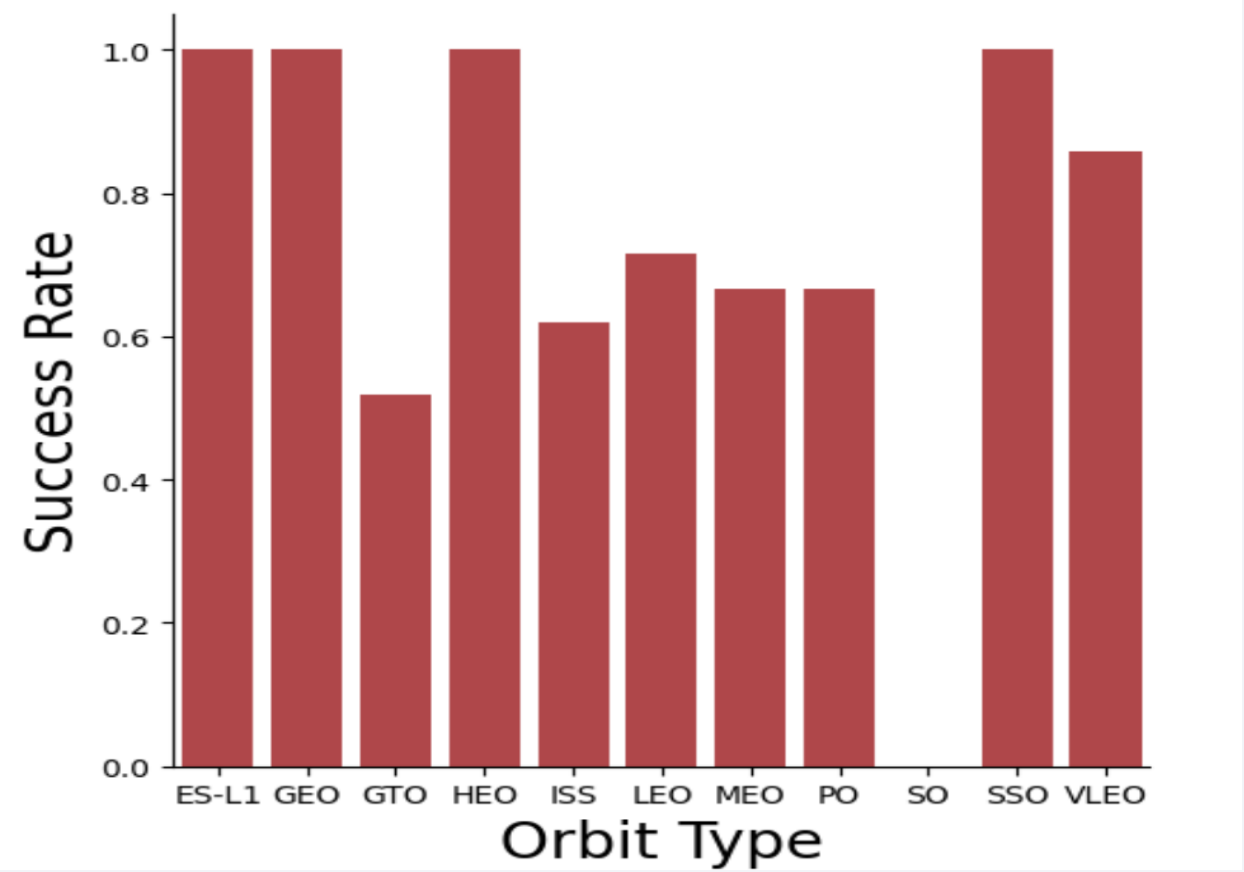
# Payload vs. Launch Site



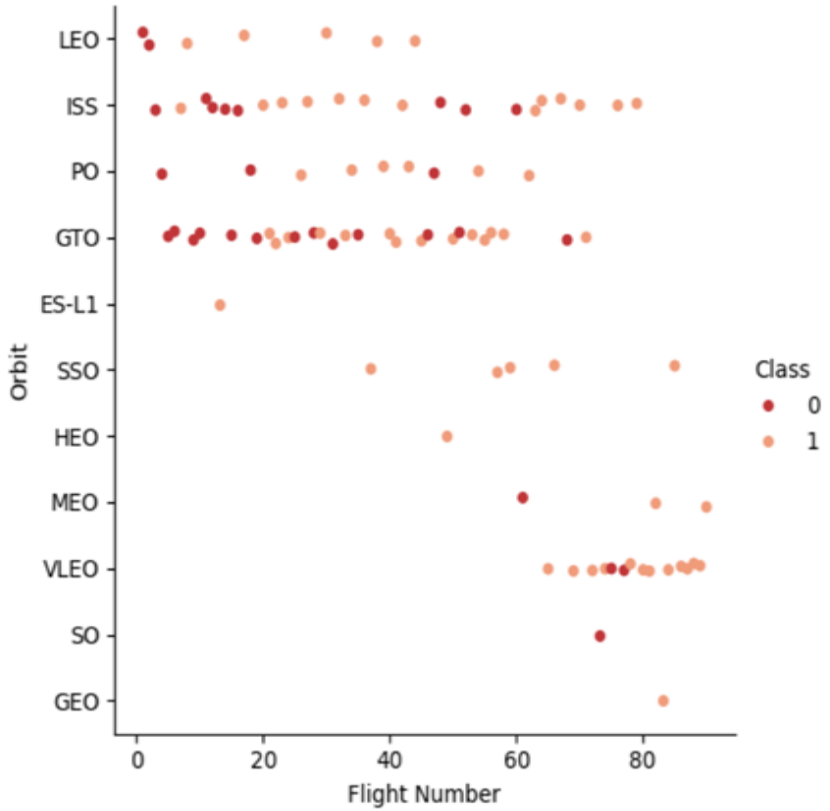

Now if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

# Success Rate vs. Orbit Type

Four Orbit Type has the highest success rate of 100%: ES-L1, GEO, HEO and SSO

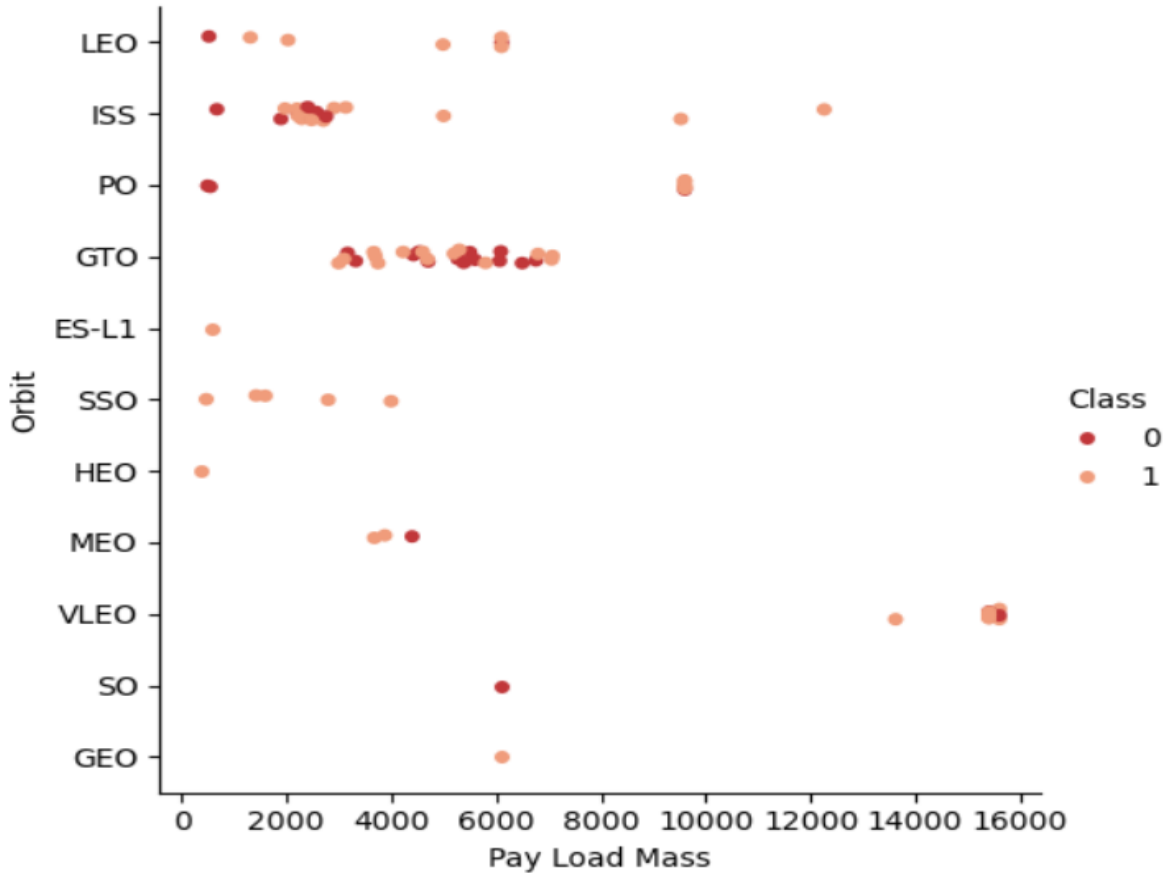


# Flight Number vs. Orbit Type




You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

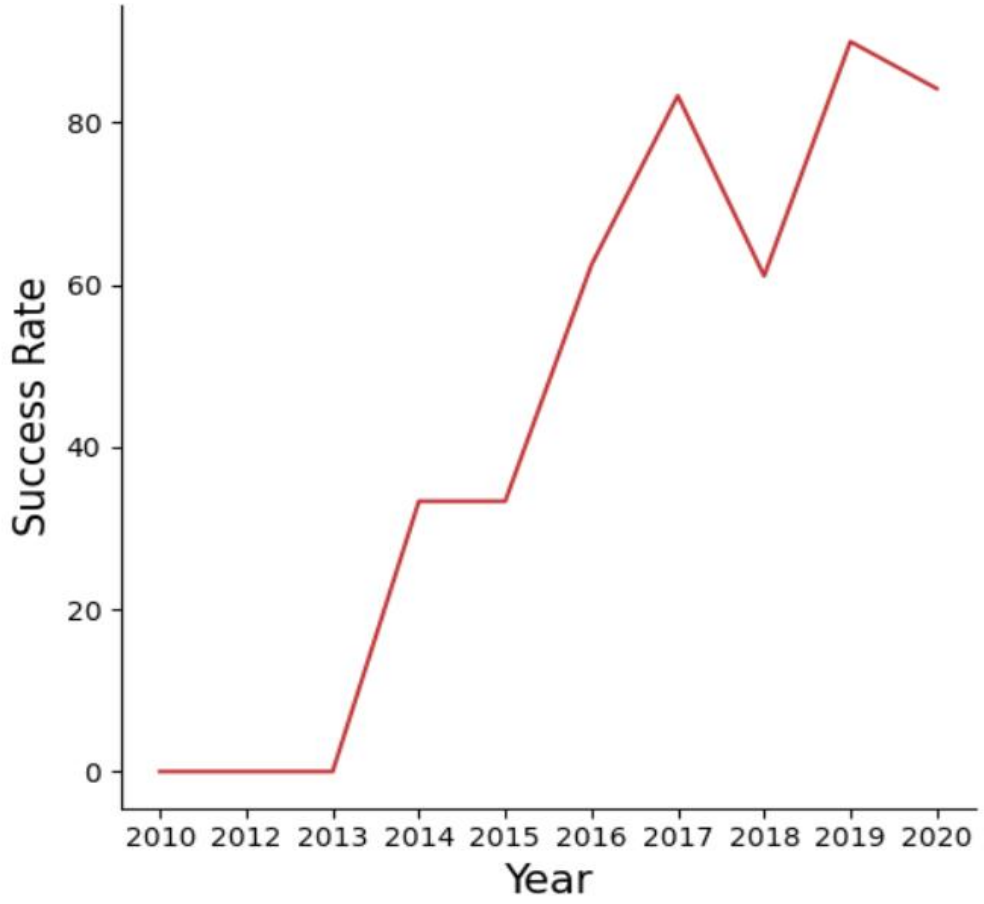
# Payload vs. Orbit Type




With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

# Launch Success Yearly Trend




you can observe that the sucess rate since 2013 kept increasing till 2020

# All Launch Site Names

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

```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL;
```

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

| Launch_Site |
| --- |
| CCAFS LC-40 |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |

The name of unique launch site are:
**CCAFS LC-40**: Space Launch Complex 40, located at north Cape Cañaveral, Florida (Wikipedia)
**VAFB SLC-4E**: Vanderberg AFB Space Launch Complex 4, in California. It has two pads, both are used by SpaceX for Falcon 9, one for launch operations and other as landing zone (Wikipedia)
**KSC LC-39A**: Kennedy Space Center Launch Complex 39ª, located at NASA's Kennedy Space Center in Merrit Island, Florida, with three launch pads (Wikipedia)

# Launch Site Names Begin with 'CCA'

Below is presented the list of Launch Site which names begin with ‘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 | 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

## Task 3

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

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

| SUM(PAYLOAD_MASS__KG_) |
|---|
| 45596 |

# Average Payload Mass by F9 v1.1

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

| AVG(PAYLOAD_MASS__KG_) |
|---|
| 2928.4 |

# First Successful Ground Landing Date

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 |

# 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 PAYLOAD
FROM SPACEXTBL
WHERE LANDING_OUTCOME = 'Success (drone ship)'
AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

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

| Payload |
| --- |
| JCSAT-14 |
| JCSAT-16 |
| SES-10 |
| SES-11 / EchoStar 105 |

# Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

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

| Mission_Outcome | TOTAL_NUMBER |
|---|---|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

# Boosters Carried Maximum Payload

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

```
%%sql
SELECT DISTINCT BOOSTER_VERSION
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (
    SELECT MAX(PAYLOAD_MASS__KG_)
    FROM SPACEXTBL);
```

```
 * 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

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, DATE,BOOSTER_VERSION, LAUNCH_SITE, [Landing_Outcome]
FROM SPACEXTBL
where Landing_Outcome = 'Failure (drone ship)' and substr(Date,0,5)='2015';
```

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

| month | Date | Booster_Version | Launch_Site | Landing_Outcome |
|---|---|---|---|---|
| 01 | 2015-01-10 | F9 v1.1 B1012 | CCAFS LC-40 | Failure (drone ship) |
| 04 | 2015-04-14 | F9 v1.1 B1015 | CCAFS LC-40 | Failure (drone ship) |

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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%%sql
SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING_OUTCOME
ORDER BY TOTAL_NUMBER DESC
```

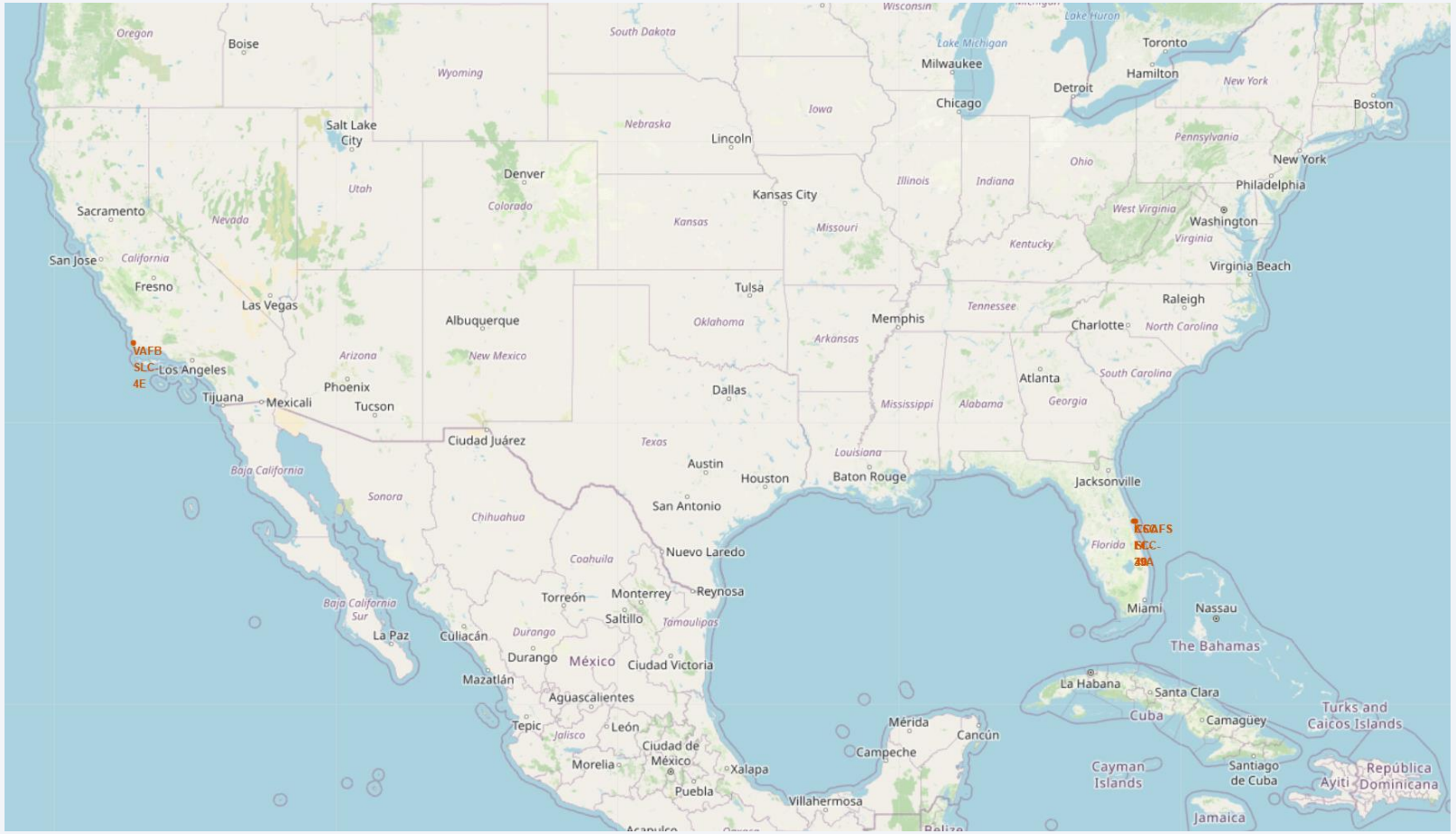
```
 * sqlite:///my_data1.db
Done.
```

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

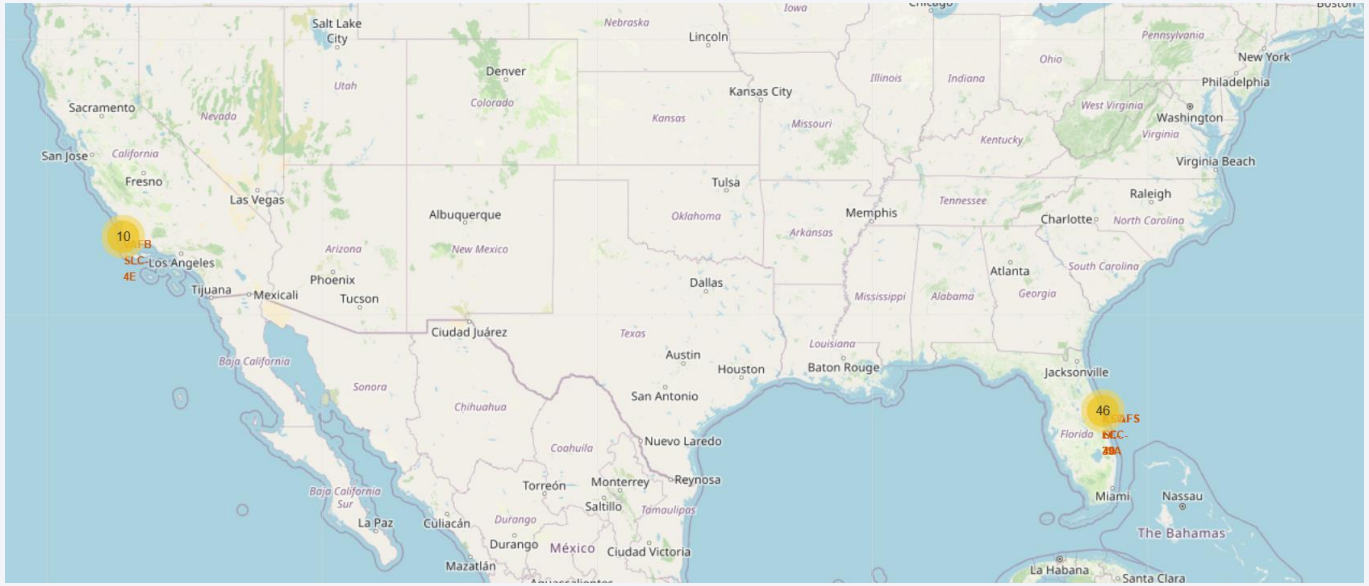
Section 3

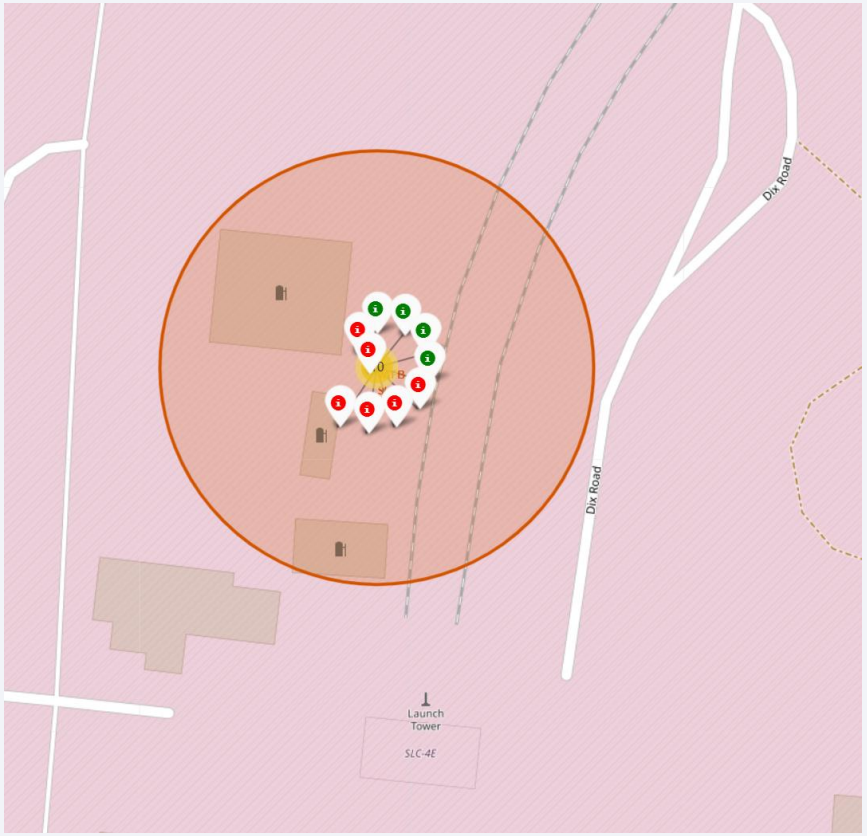
# Launch Sites Proximities Analysis

# Launch Site Location Analysis with Folium

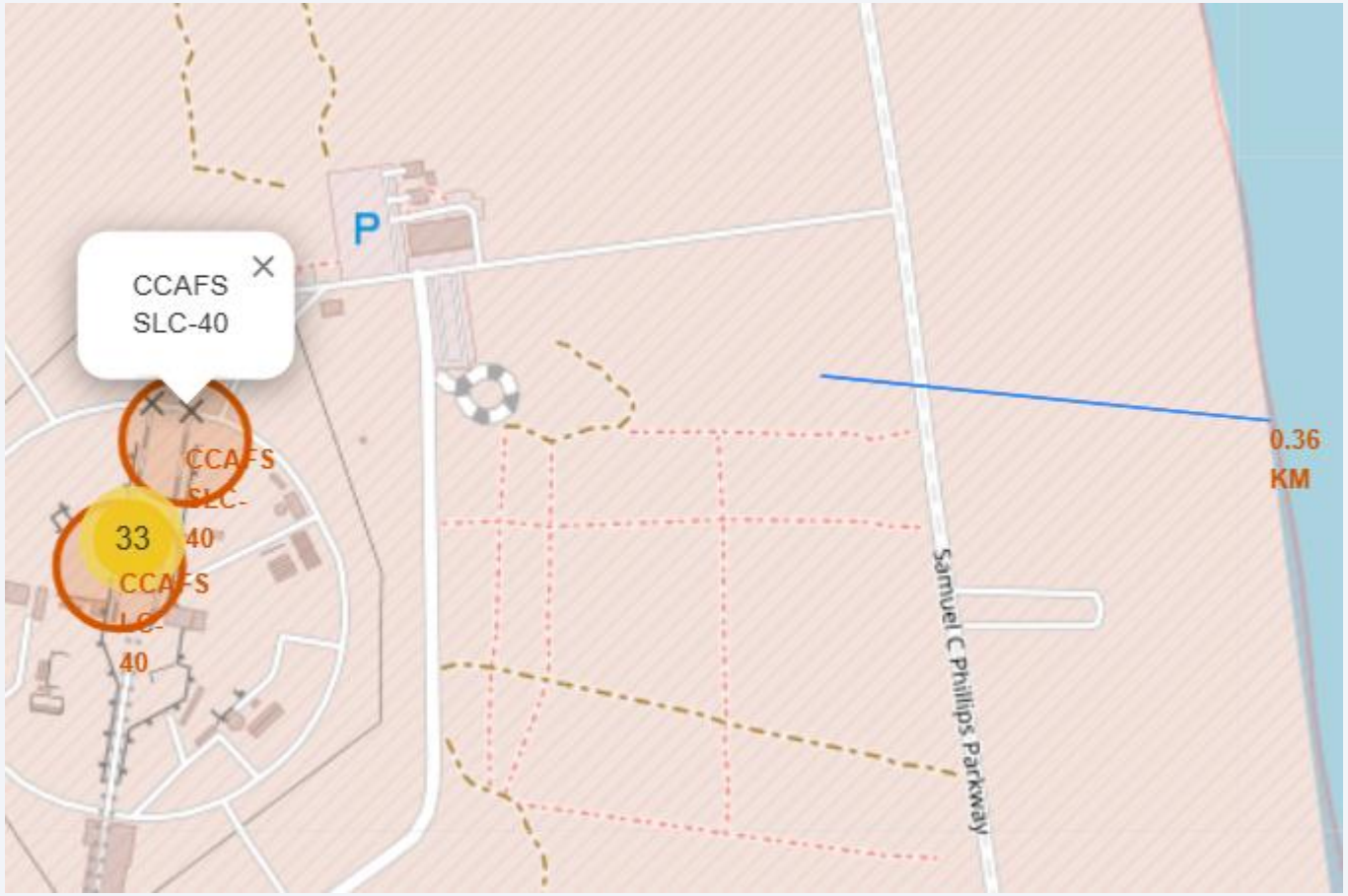


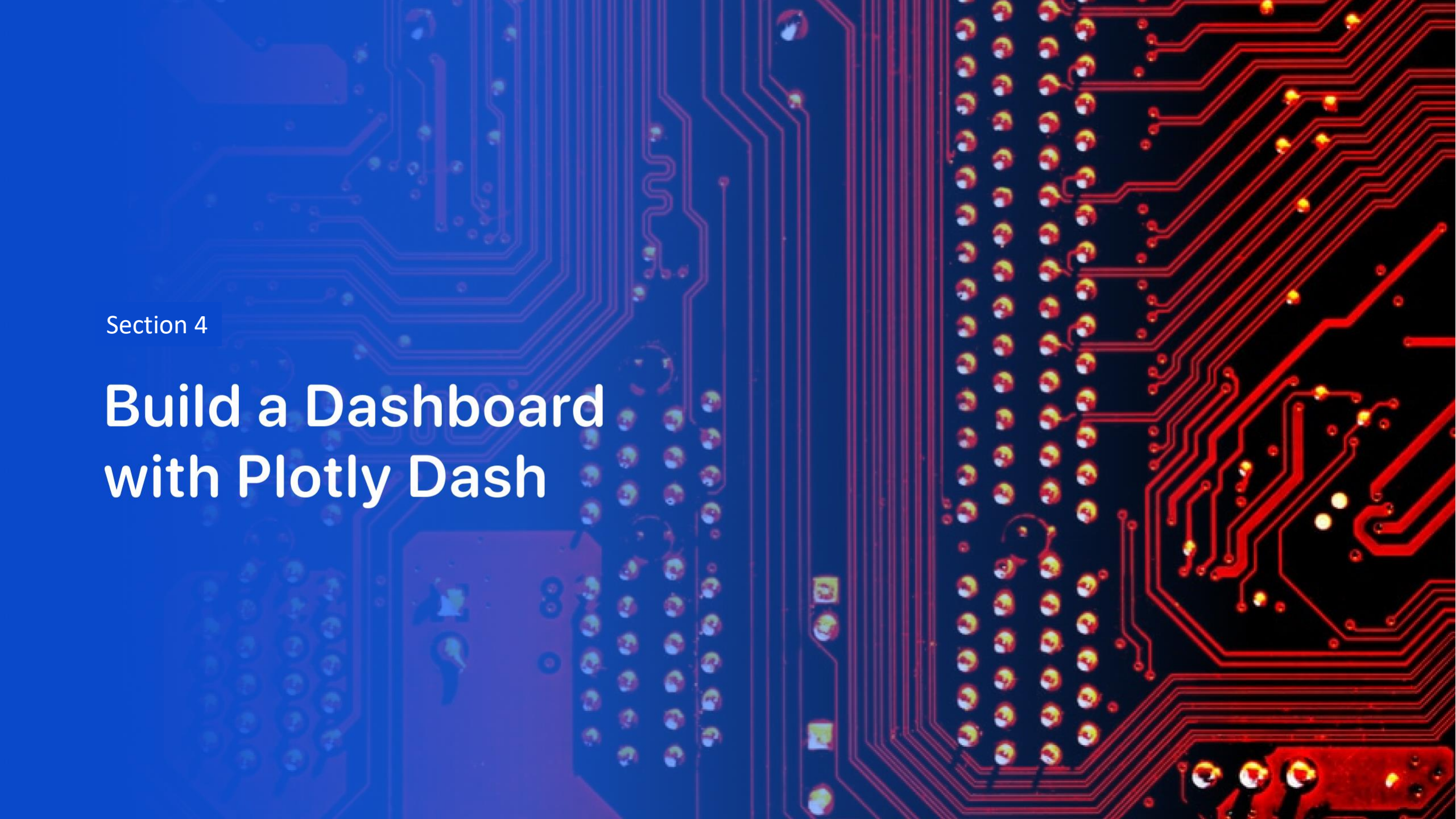
# Launch sites ubication and labeled



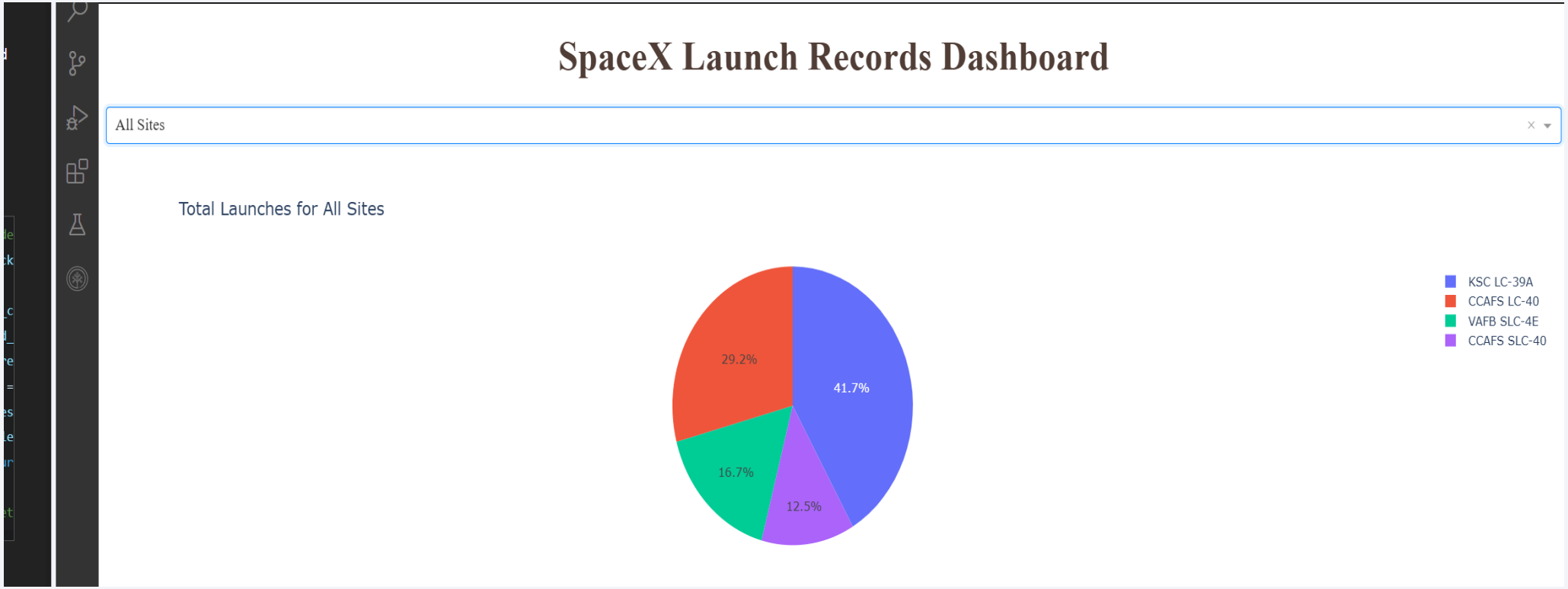



# Launch sites markers distance to railway, highway and coastline

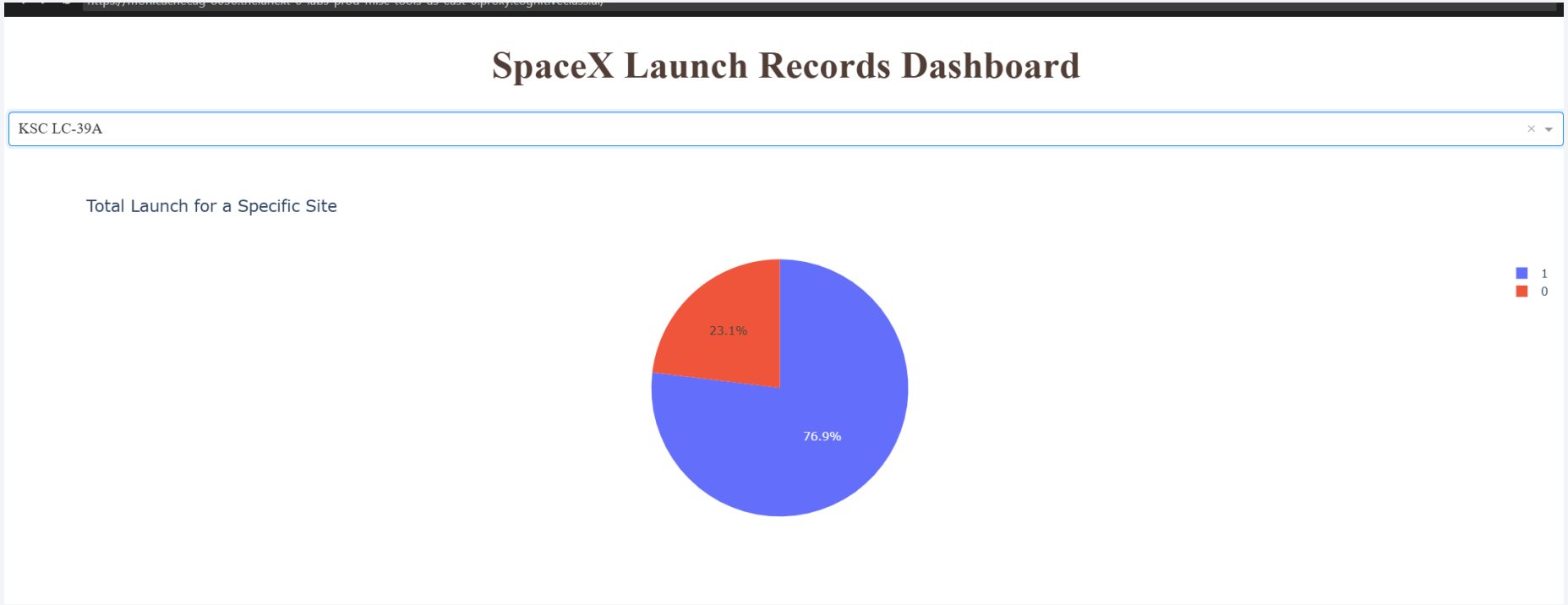



Section 4
Build a Dashboard with Plotly Dash

# Pie Chart with all the launch success sites

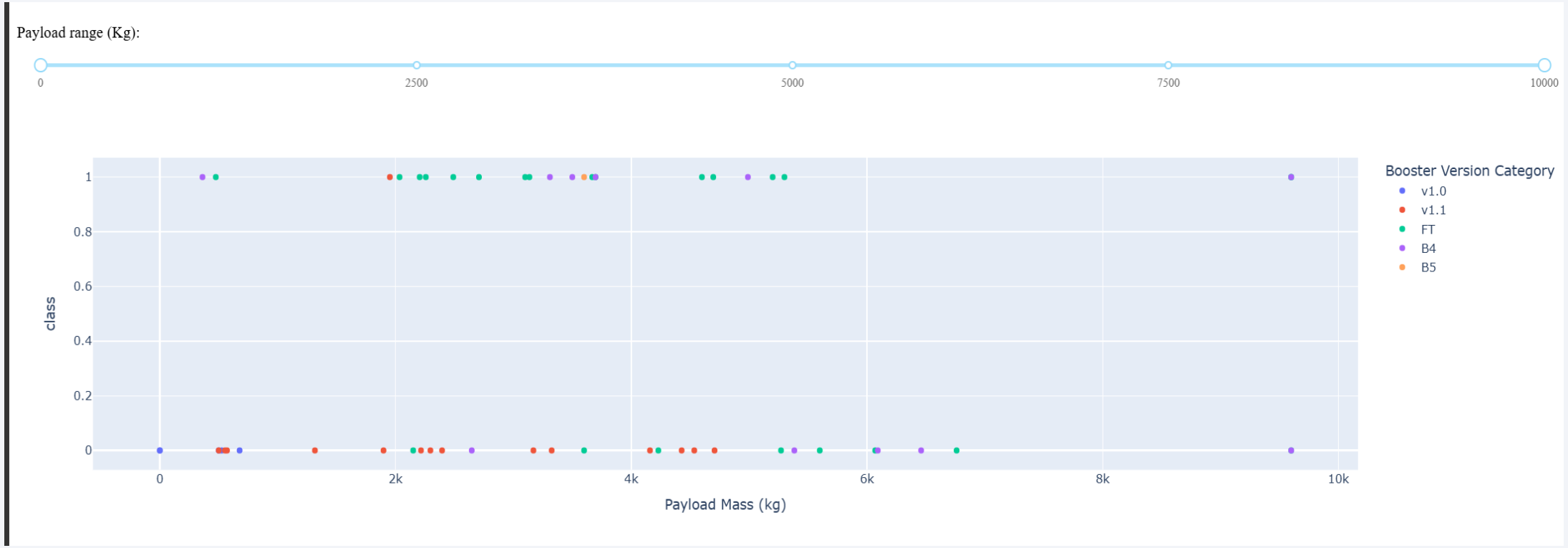


# Pie chart with the highest launch success ratio




KSC LC-39 A has highest launch ratio of 76,9%

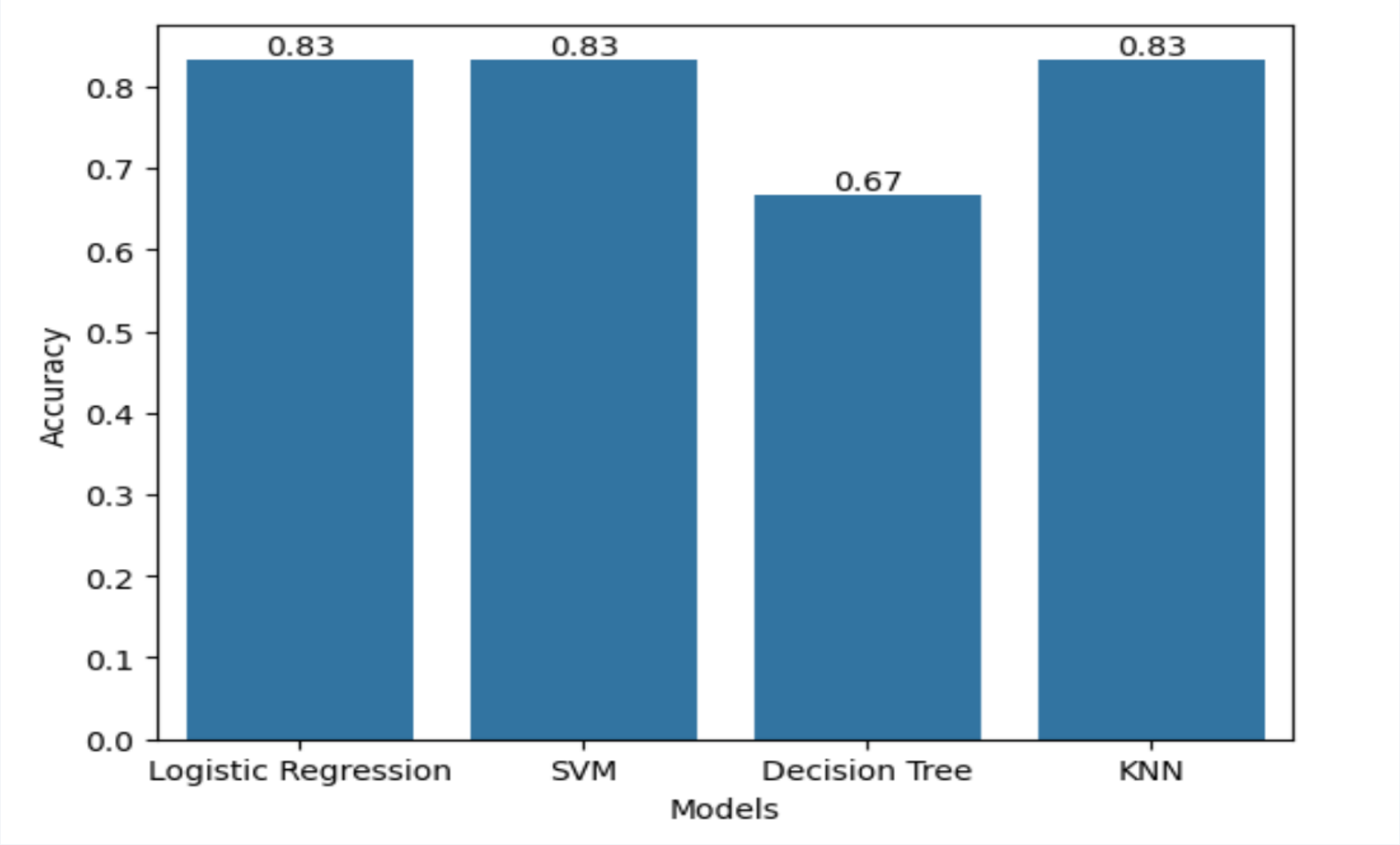
# Payload vs. Lauch Outcome scatter plot



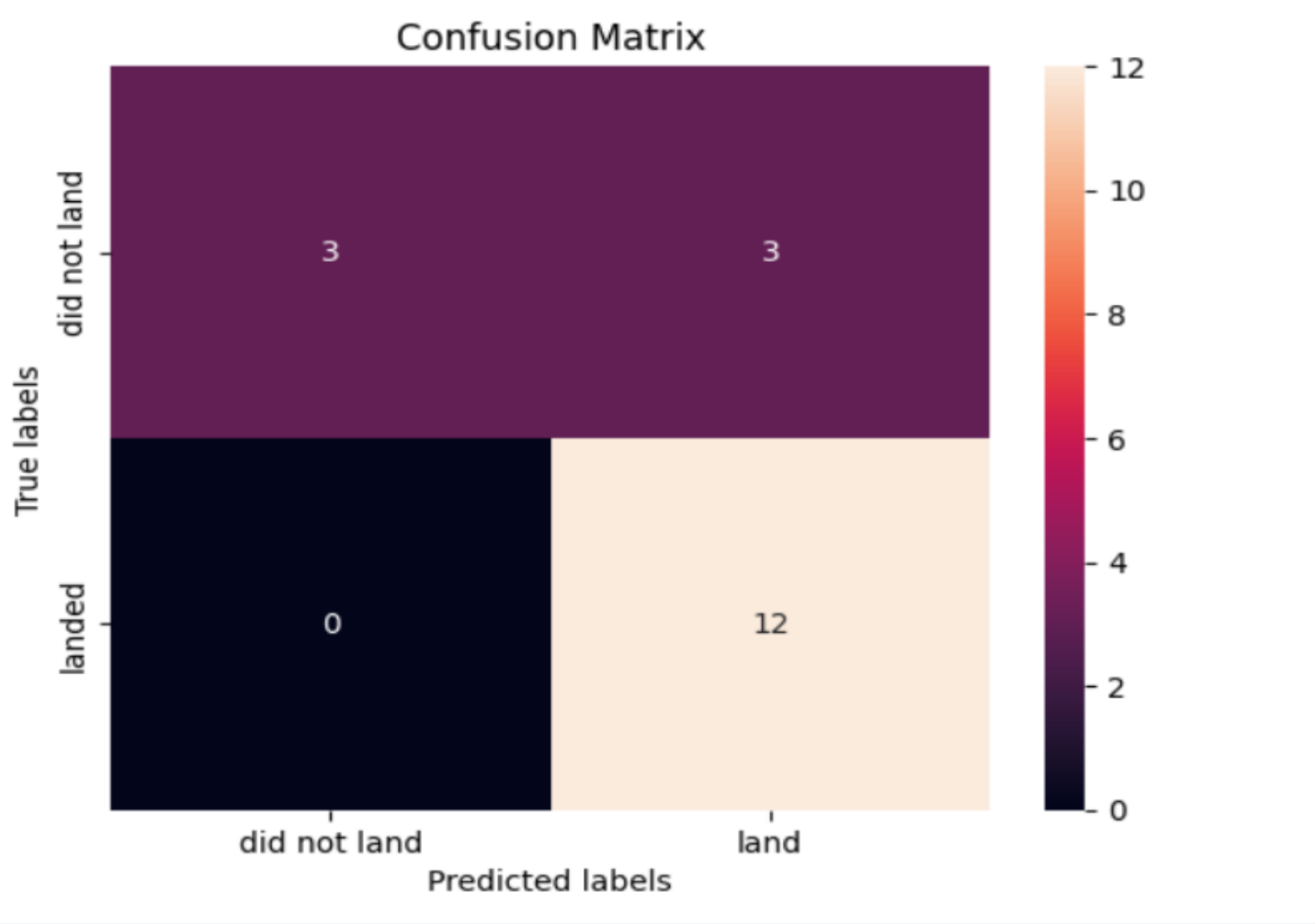
Section 5

# Predictive Analysis (Classification)

# Classification Accuracy



# Confusion Matrix




In this case, the knn_cv has an accuracy of 83,3% with 3 false positive and 0 false negative while recall is 100%.

# Conclusions

- We can observe that flight number increases when more success rate for Launch Site CCAFS SLC 40 and the same situation ocurrs for payload mass.
- The success rate for Falco 9 landing is increased since 2010 to 2020.
- There is a 100% of success rate of landing on Orbit type ES-L1, SSO, HEO and GEO
- We can predict with our model the outcome of the landing with an accuracy of 83,3%.

# Appendix

- The relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that I have created during this project are all in the repository of my GitHut:
- MonChe5/SpaceXCapstone: Data Science Applied (github.com)

# Thank you!