**IBM Developer**
**SKILLS NETWORK**

# Winning Space Race with Data Science

Jairo Rodriguez Calume
Octubre 2024

# Outline

- Executive Summary

- Introduction

- Methodology

- Results

- Conclusion

- Appendix

# Executive Summary

- Summary of methodologies

  - SpaceX Data Collection using SpaceX API

  - SpaceX Data Collection with Web Scraping

  - SpaceX Data Wrangling

  - SpaceX Exploratory Data Analysis using SQL

  - Space-X EDA DataViz Using Python Pandas and Matplotlib

  - Space-X Launch Sites Analysis with Folium-Interactive Visual Analytics and Ploty Dash

  - SpaceX Machine Learning Landing Prediction

- Summary of all results

  - EDA results

  - Interactive Visual Analytics and Dashboards

  - Predictive Analysis(Classification)

# Introduction
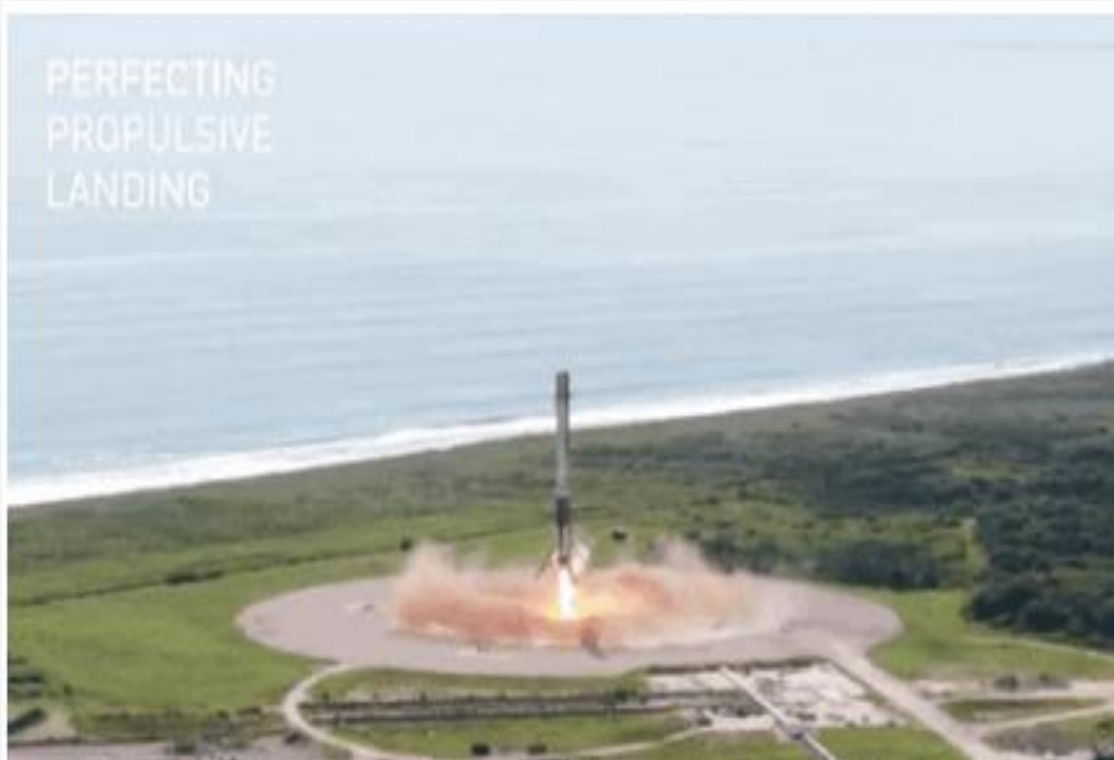


- Project background and context

  SpaceX 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 SpaceX 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 SpaceX for a rocket launch.

- Problems you want to find answers

  In this capstone, we will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website.

Section 1

# Methodology

# Methodology

Executive Summary

- Data collection methodology:

  - Describes how data sets were collected

- Perform data wrangling

  - Describes how data were processed

- 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

- Description of how SpaceX Falcon9 data was collected.

  - Data was first collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API. This was done by first defining a series helper functions that would help in the use of the API to extract information using identification numbers in the launch data and then requesting rocket launch data from the SpaceX API url.

  - Finally to make the requested JSON results more consistent, the SpaceX launch data was requested and parsed using the GET request and then decoded the response content as a Json result which was then converted into a Pandas data frame.

  - Also performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches of the launch records are stored in a HTML. Using BeautifulSoup and request Libraries, I extract the Falcon 9 launch HTML table records from the Wikipedia page, Parsed the table and converted it into a Pandas data frame

# Data Collection – SpaceX API

- Data collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API then requested and parsed the SpaceX launch data using the GET request and decoded the response content as a Json result which was then converted into a Pandas data frame

- Here is the GitHub URL of the completed SpaceX API calls notebook https://github.com/jmrocalume/Ciencia_Dat_Capstone/blob/main/1.%20Space−X%20Data%20Collection%20API.ipynb



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.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

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
respjson = response.json()
data = pd.json_normalize(respjson)
```

# Data Collection - WebScraping

- Performed web scraping to collect Falcon 9 historical launch records from a Wikipedia using BeautifulSoup and request, to extract the Falcon 9 launch records from HTML table of the Wikipedia page, then created a data frame by parsing the launch HTML.

- Here is the GitHub URL of the completed web scraping notebook.

- (*hhttps://github.com/jmrocalume/ Ciencia_Dat_Capstone/blob/main/2.% 20Space−X%20Web%20scraping% 20Falcon%209%20and%20Falcon% 20Heavy%20Launches%20Records% 20from%20Wikipedia.ipynb*)

In [4]: `static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"`

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.

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, 'html.parser')
```

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

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]: `List of Falcon 9 and Falcon Heavy launches - Wikipedia`

## TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external re this lab

In [10]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
```

# Data Wrangling

- After obtaining and creating a Pandas DF from the collected data, data was filtered using the *BoosterVersion* column to only keep the Falcon 9 launches, then dealt with the missing data values in the *LandingPad* and *PayloadMass* columns. For the *PayloadMass* , missing data values were replaced using mean value of column.

- Also performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models

- Here is the GitHub URL of the completed data wrangling related notebooks.

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 variable `landing_class` :

```
# Landing_class = 0 if bad_outcome
# Landing_class = 1 otherwise
df['Class'] = df['Outcome'].apply(lambda x: 0 if x in bad_outcomes else 1)
df['Class'].value_counts()
```

```
1    60
0    30
Name: Class, dtype: int64
```

This variable will represent the classification variable that represents the outcome of each launch. If the first stage landed Successfully

```
landing_class=df['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

- Performed data Analysis and Feature Engineering using Pandas and Matplotlib.i.e.

    - Exploratory Data Analysis

    - Preparing Data Feature Engineering

- Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, FlightNumber and Orbit type, Payload and Orbit type.

- Used Bar chart to Visualize the relationship between success rate of each orbit type

- Line plot to Visualize the launch success yearly trend.

- Here is the GitHub URL of your completed EDA with data visualization notebook, (hhttps://github.com/jmrocalume/Ciencia_Dat_Capstone/blob/main/5.%20Space-X%20EDA%20DataViz%20Using%20Pandas%20and%20Matplotlib%20-%20SpaceX.ipynb)

# EDA with Data Visualization (Plots Cont....)
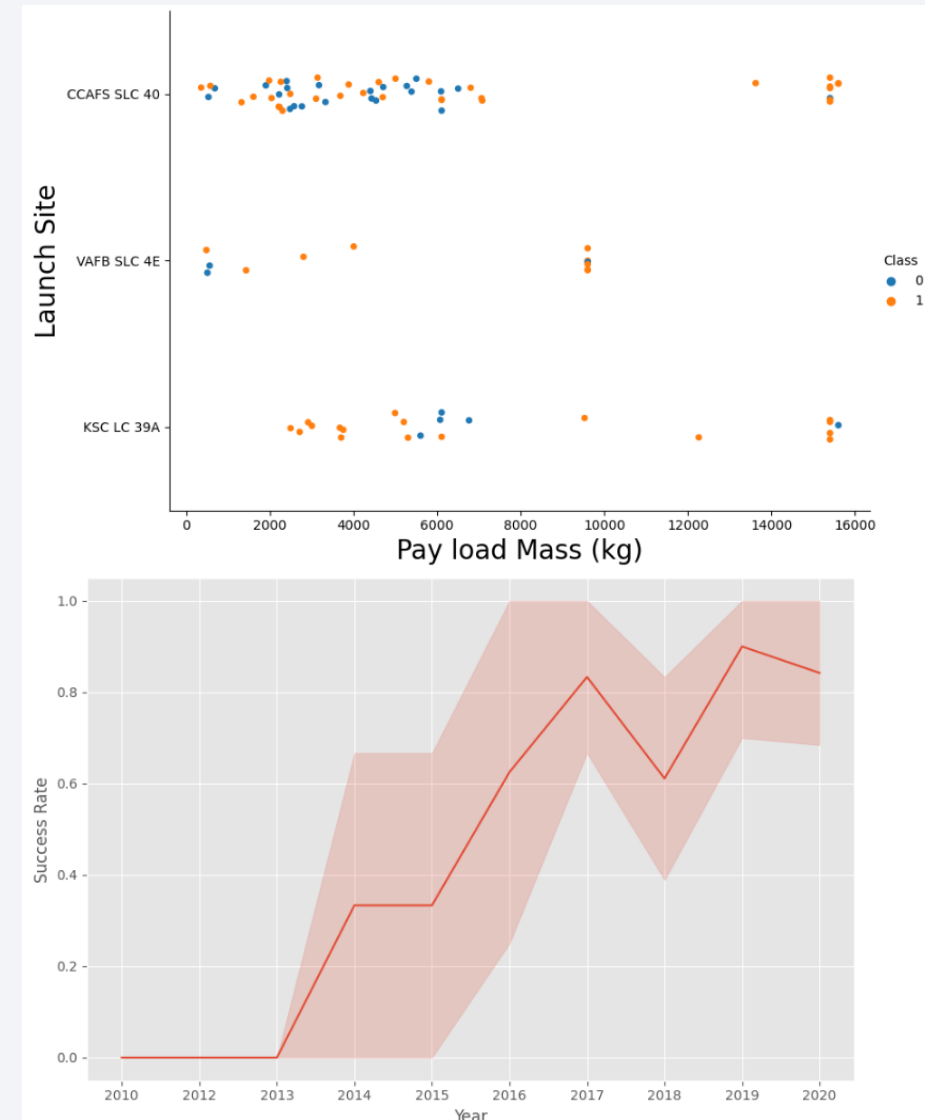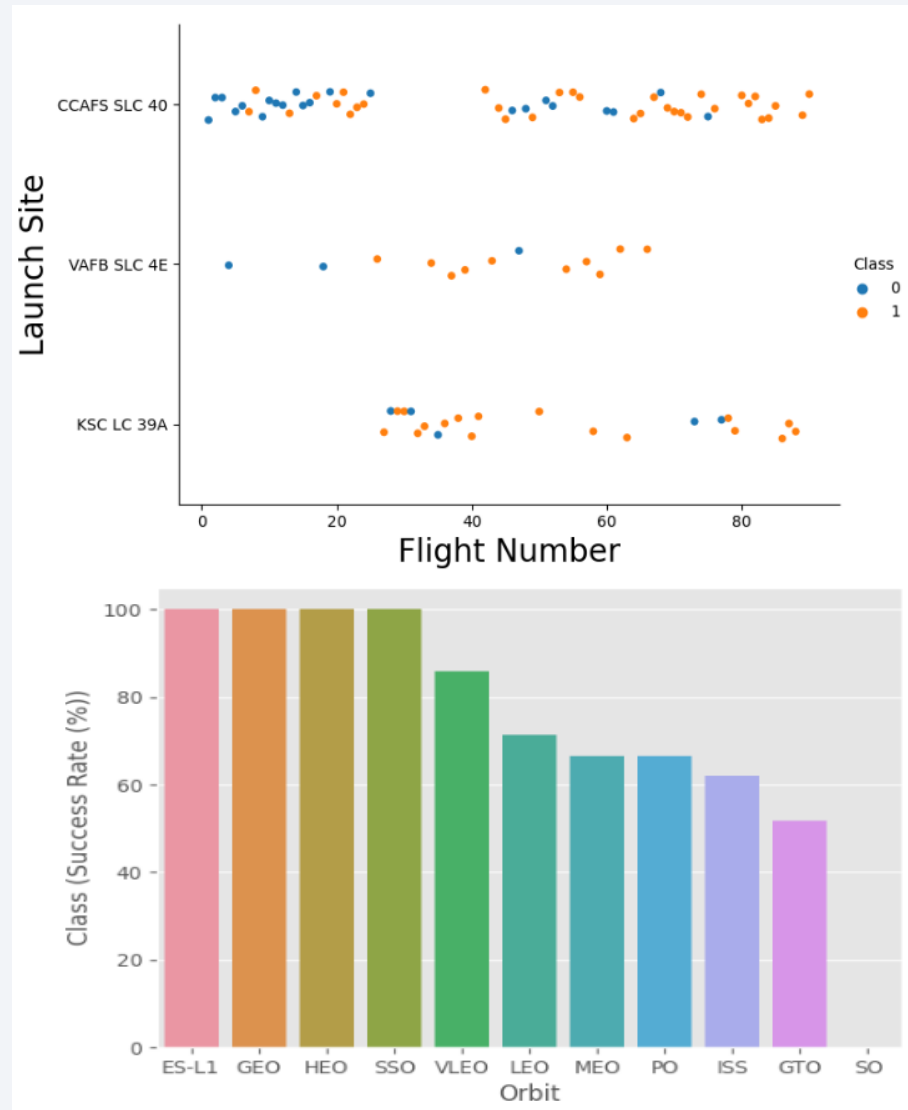
# EDA with SQL

- The following SQL queries were performed for EDA

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

    ```
    %sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
    ```

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

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

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

    ```
    %sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = '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' WHERE Booster_Version LIKE 'F9 v1.1%';
    ```

# EDA with SQL (Cont.…)

- List the date when the first successful landing outcome in ground pad was achieved

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

- 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 ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;)*

- 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";
```

- [Here](https://github.com/jmrocalume/Ciencia_Dat_Capstone/blob/main/4.%20Space-X%20EDA%20Using%20SQL.ipynb) is the GitHub URL of your completed EDA with SQL notebook. (*https://github.com/jmrocalume/Ciencia_Dat_Capstone/blob/main/4.%20Space−X%20EDA%20Using%20SQL.ipynb*)

# Build an Interactive Map with Folium

- Created folium map to marked all the launch sites, and created map objects such as markers, circles, lines to mark the success or failure of launches for each launch site.

- Created a launch set outcomes (failure=0 or success=1).

- Here is the GitHub URL of the completed interactive map with Folium map, as an external reference and peer-review purpose ($https://github.com/jmrocalume/$ $Ciencia\_Dat\_Capstone/blob/main/6.Space-X\%20Launch\%20Sites\%20Locations\%$ $20Analysis\%20with\%20Folium-Interactive\%20Visual\%20Analytics.ipynb$)

# Build a Dashboard with Plotly Dash

- Built an interactive dashboard application with Plotly dash by:

  - Adding a Launch Site Drop-down Input Component

  - Adding a callback function to render success-pie-chart based on selected site dropdown

  - Adding a Range Slider to Select Payload

  - Addeng a callback function to render the success-payload-scatter-chart scatter plot

- [Here is the GitHub URL](https://github.com/jmrocalume/Ciencia_Dat_Capstone/blob/main/7.%20Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash%20-%20spacex_dash_app.py) of your completed Plotly Dash lab
  (*hhttps://github.com/jmrocalume/Ciencia_Dat_Capstone/blob/main/7.%20Build%20an%20Interactive%20Dashboard%20with%20Ploty%20Dash%20-%20spacex_dash_app.py*)

# SpaceX Dash App
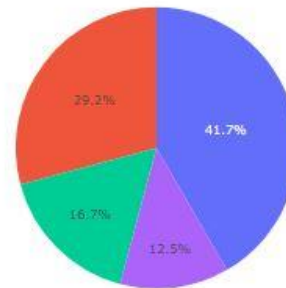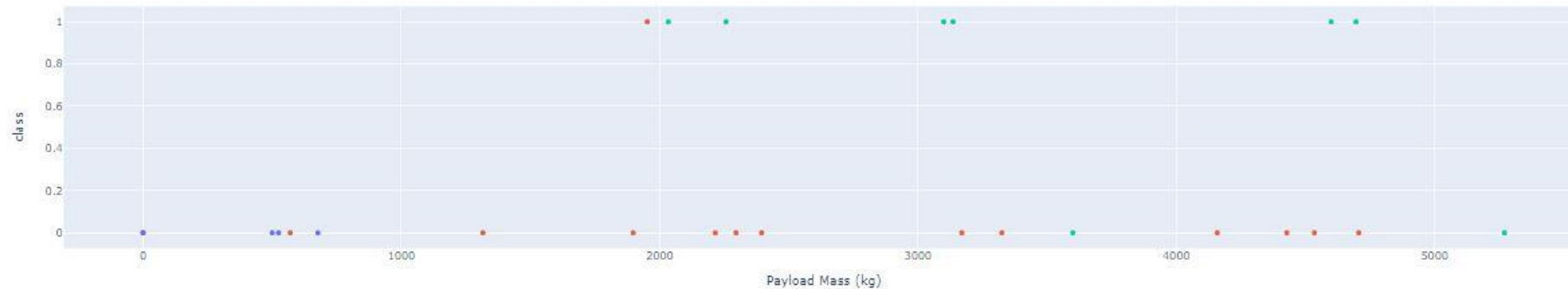
# Predictive Analysis (Classification)

- Summary of how I built, evaluated, improved, and found the best performing classification model

- After loading the data as a Pandas Dataframe, I set out to perform exploratory Data Analysis and determine Training Labels by;

  - creating a NumPy array from the column Class in data, by applying the method to_numpy() then assigned it to the variable Y as the outcome variable.

  - Then standardized the feature dataset (x) by transforming it using preprocessing.StandardScaler() function from Sklearn.

  - After which the data was split into training and testing sets using the function train_test_split from sklearn.model_selection with the test_size parameter set to 0.2 and random_state to 2.

# Predictive Analysis (Classification)

- In order to find the best ML model/ method that would performs best using the test data between SVM, Classification Trees, k nearest neighbors and Logistic Regression;

  - First created an object for each of the algorithms then created a GridSearchCV object and assigned them a set of parameters for each model.

  - For each of the models under evaluation, the GridsearchCV object was created with cv=10, then fit the training data into the GridSearch object for each to Find best Hyperparameter.

  - After fitting the training set, we output GridSearchCV object for each of the models, then displayed the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.

  - Finally using the method score to calculate the accuracy on the test data for each model and plotted a confussion matrix for each using the test and predicted outcomes.

# Predictive Analysis (Classification)

- The table below shows the test data accuracy score for each of the methods comparing them to show which performed best using the test data between SVM, Classification Trees, k nearest neighbors and Logistic Regression;

| Out[68]: | 0 |
|---|---|
| Method | Test Data Accuracy |
| Logistic_Reg | 0.833333 |
| SVM | 0.833333 |
| Decision Tree | 0.833333 |
| KNN | 0.833333 |

- GitHub URL of the completed predictive analysis lab (*https://github.com/jmrocalume/Ciencia_Dat_Capstone/blob/main/8.%20SpaceX%20Machine%20Learning%20Prediction.ipynb*)

# Results

- Exploratory data analysis results

- Interactive analytics demo in screenshots
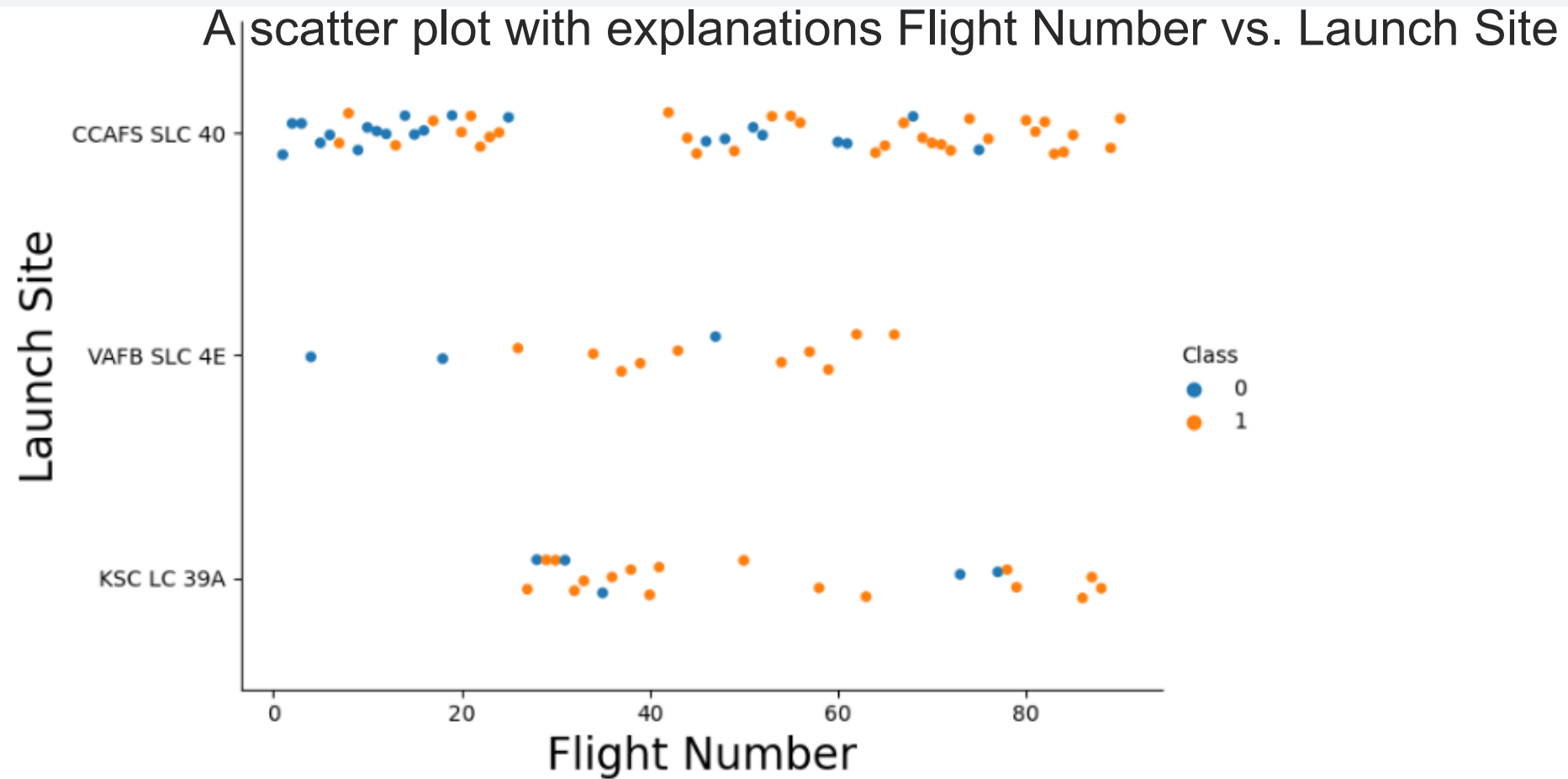
- Predictive analysis results

Section 2

# Insights drawn from EDA

# Flight Number vs. Launch Site

A scatter plot of Flight Number vs. Launch Site

# Flight Number vs. Launch Site with explanations
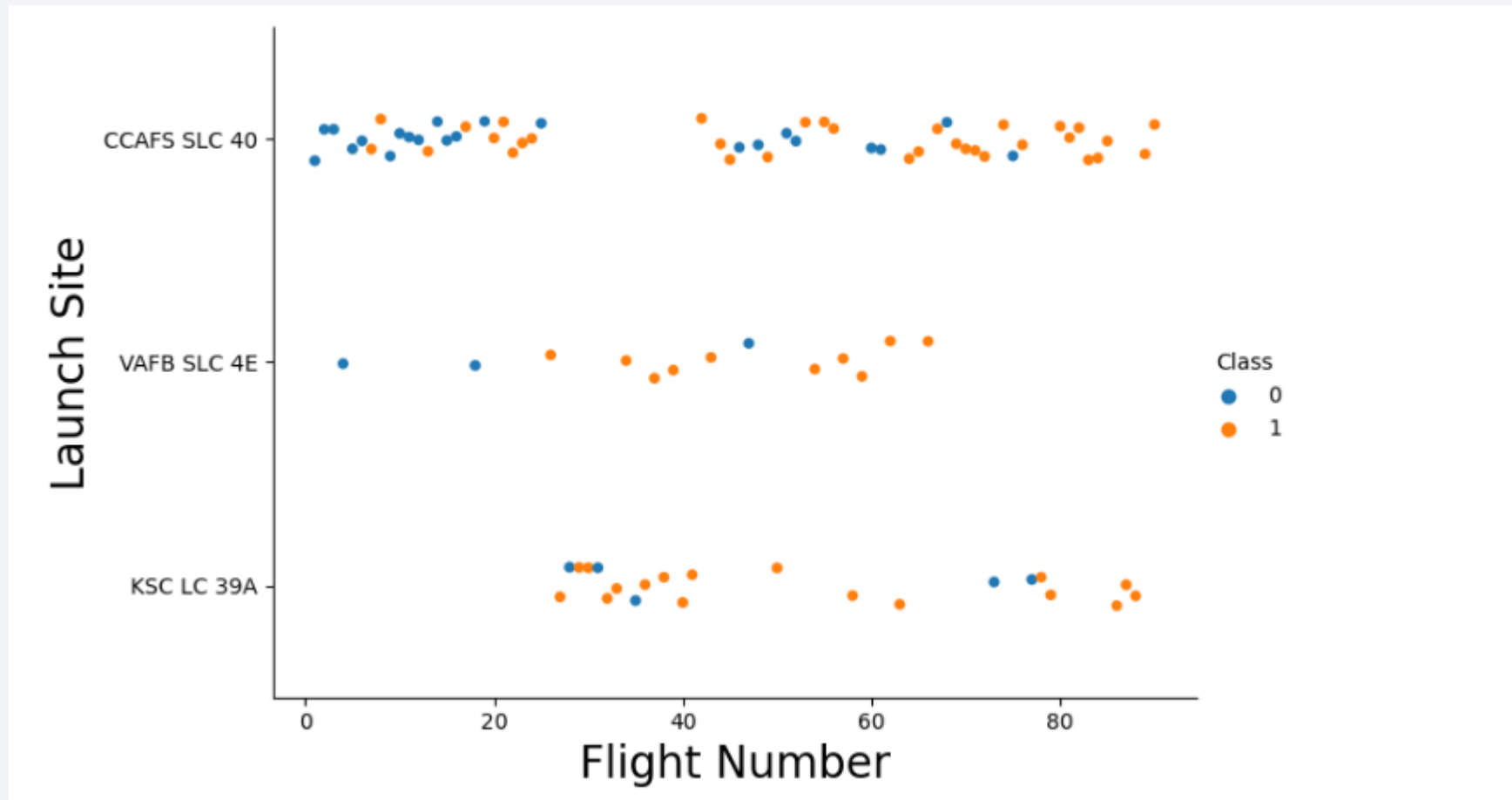


A scatter plot with explanations Flight Number vs. Launch Site

Now try to explain the patterns you found in the Flight Number vs. Launch Site scatter point plots.

We can deduce that, as the flight number increases in each of the 3 launcg sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80th flight.
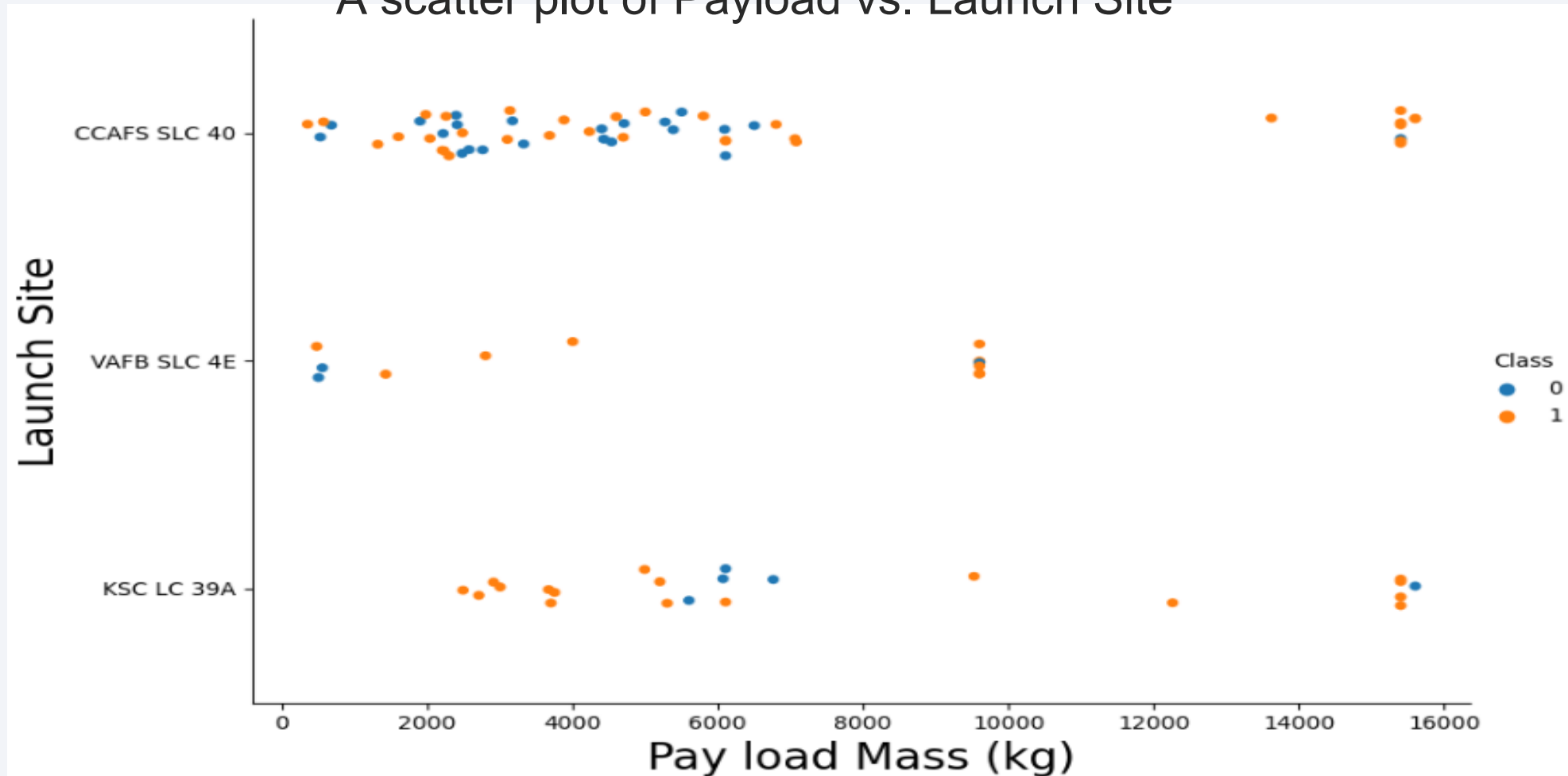
# Payload vs. Launch Site

A scatter plot of Payload vs. Launch Site
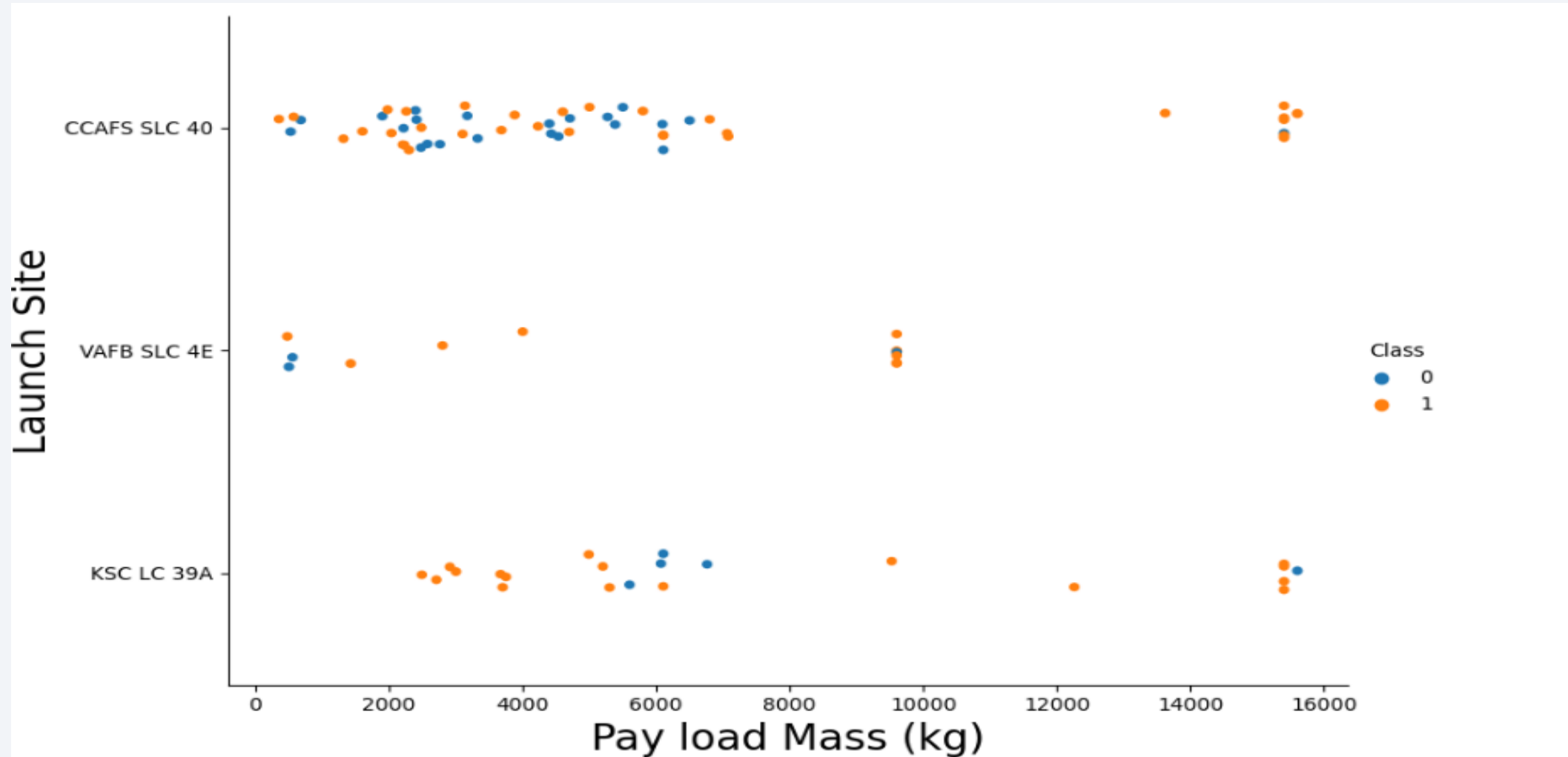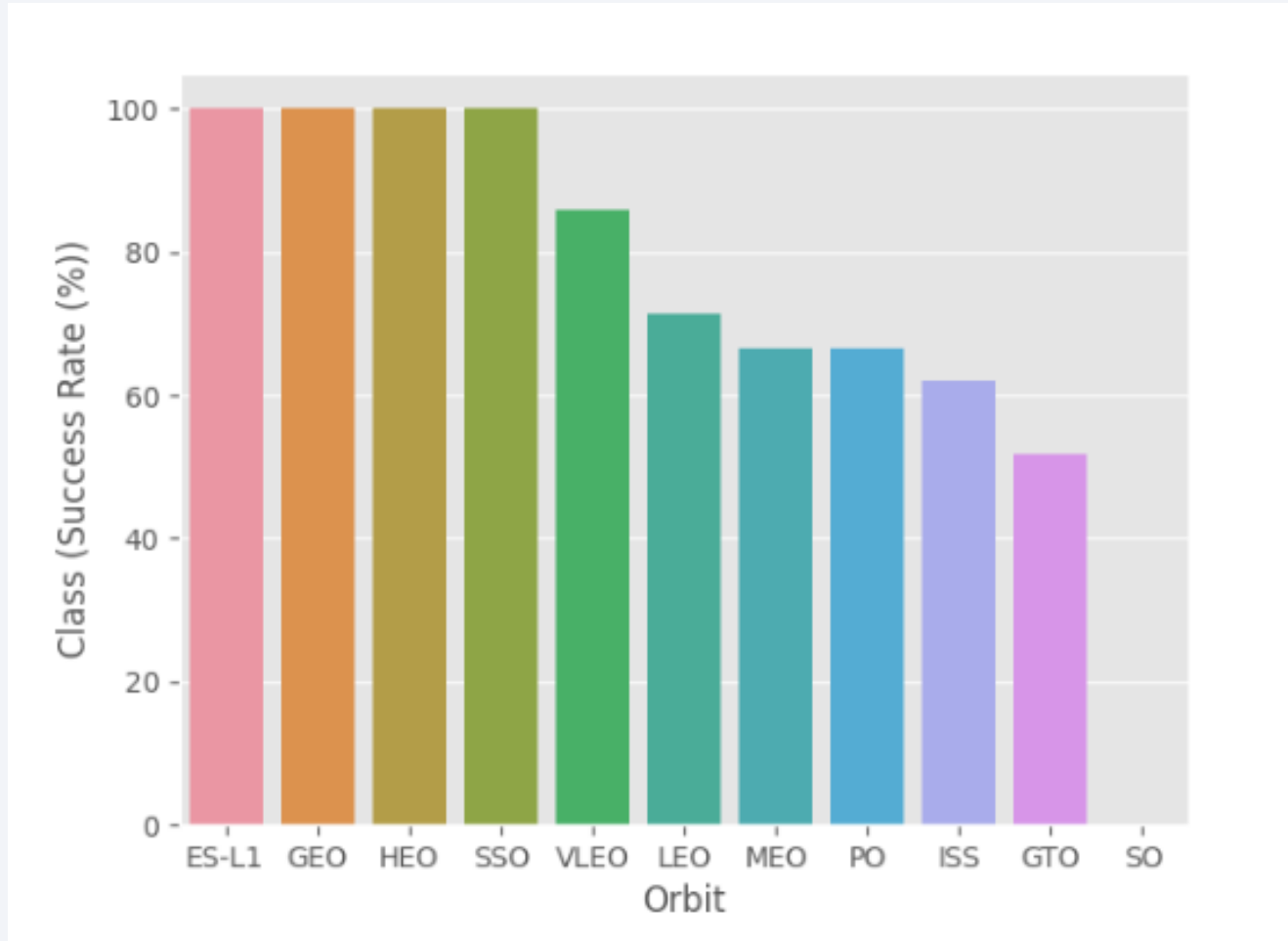
# Payload vs. Launch Site with explanations

A scatter plot of Payload vs. Launch Site

Now if you observe Payload 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).
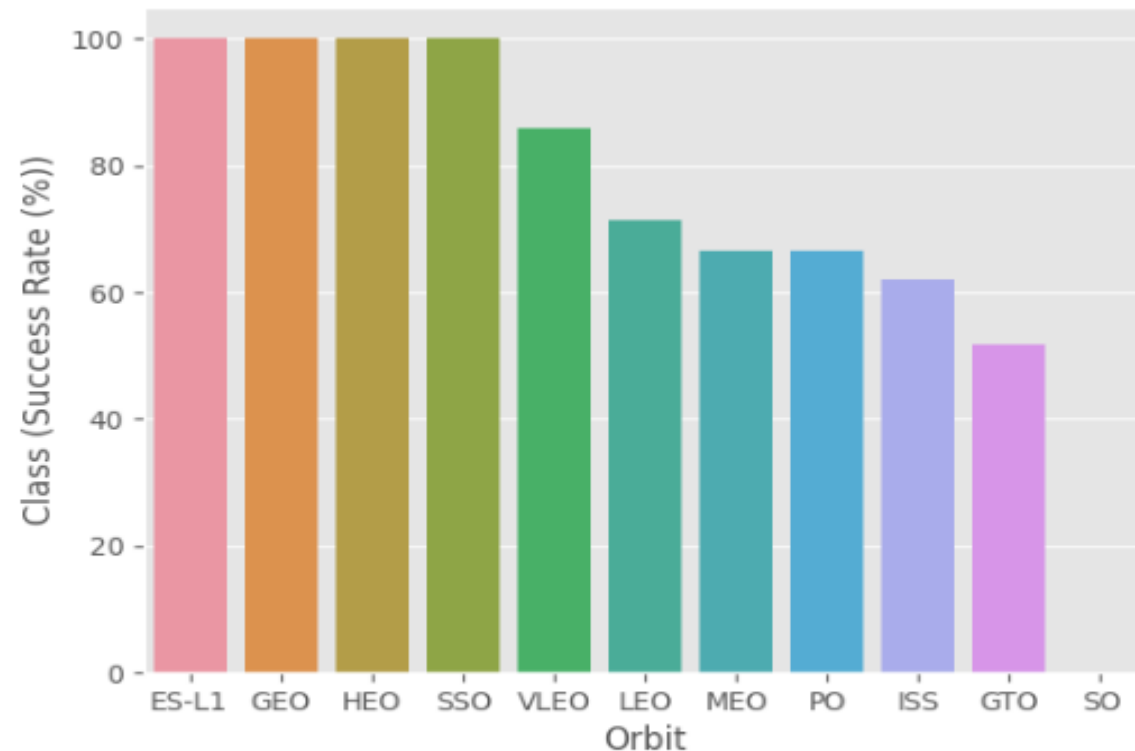
# Success Rate vs. Orbit Type

- Show a bar chart for the success rate of each orbit type

# Success Rate vs. Orbit Type with explanations

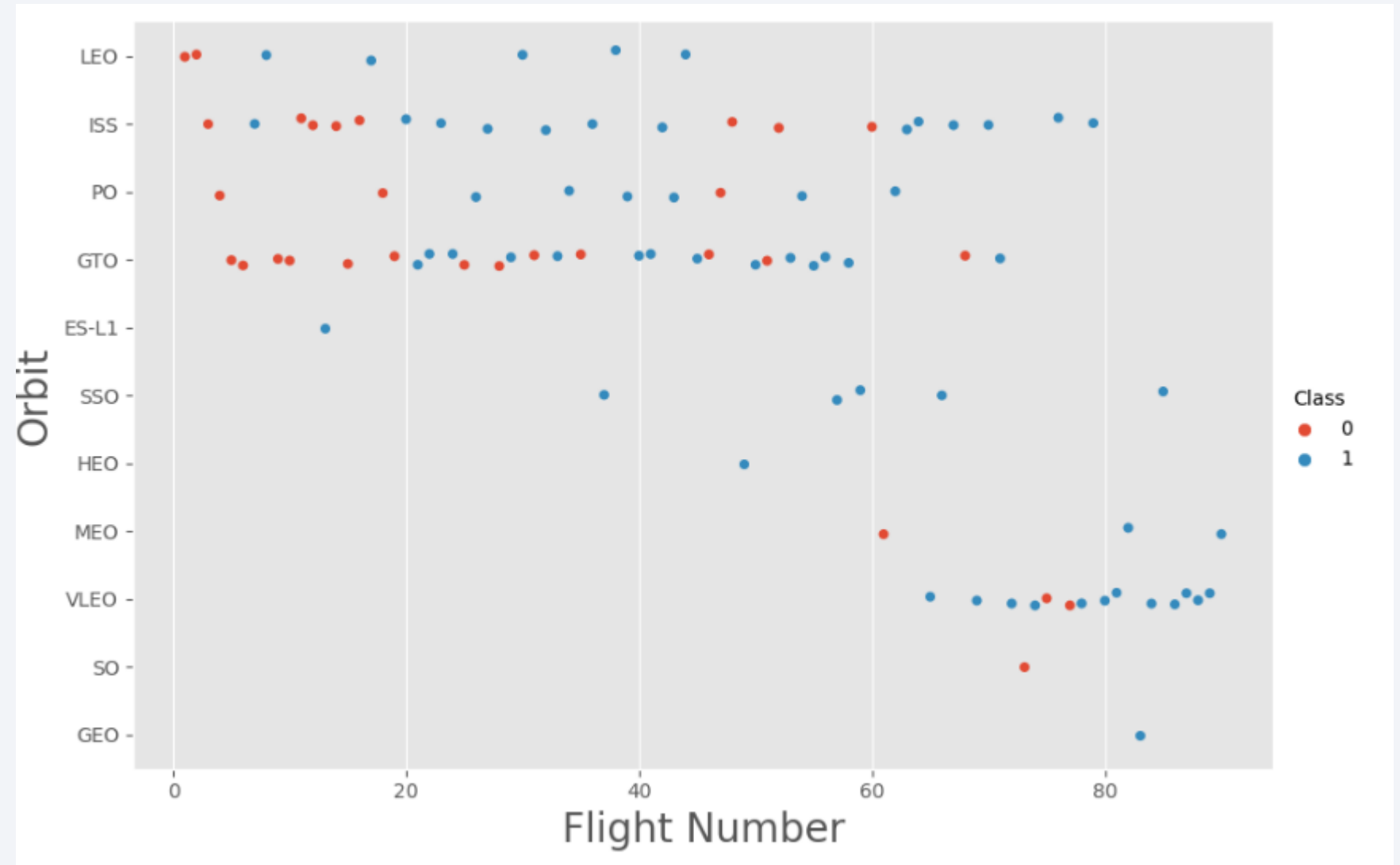- Show the screenshot of the bar chart with explanations



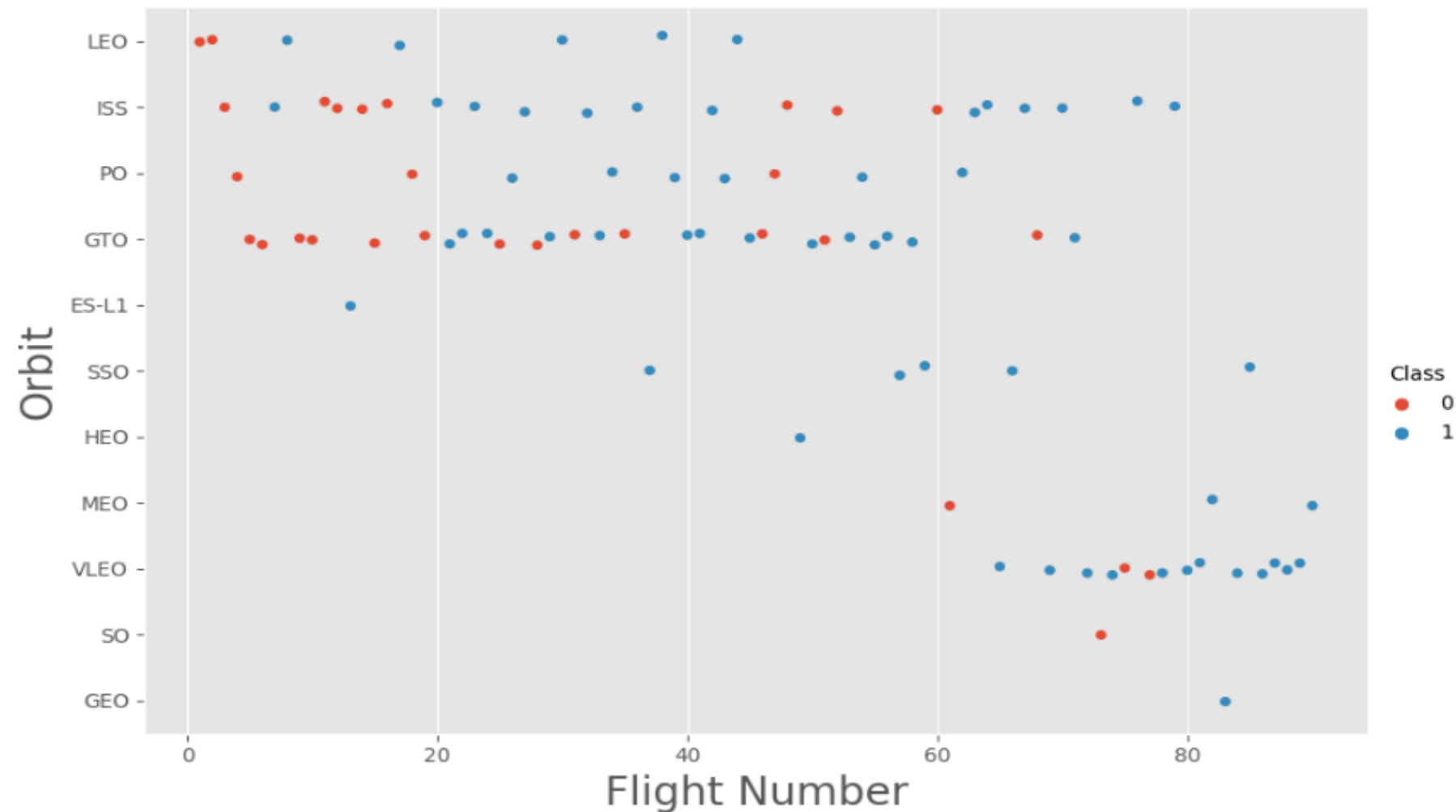Analyze the ploted bar chart try to find which orbits have high sucess rate.

Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.

# Flight Number vs. Orbit Type

- A scatter point of Flight number vs. Orbit type

# Flight Number vs. Orbit Typewith explanations

You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

# Payload vs. Orbit Type

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

- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) both have near equal chances.

# Launch Success Yearly Trend

- Since 2013, the success rate kept going up till 2020

A line chart of yearly average success rate
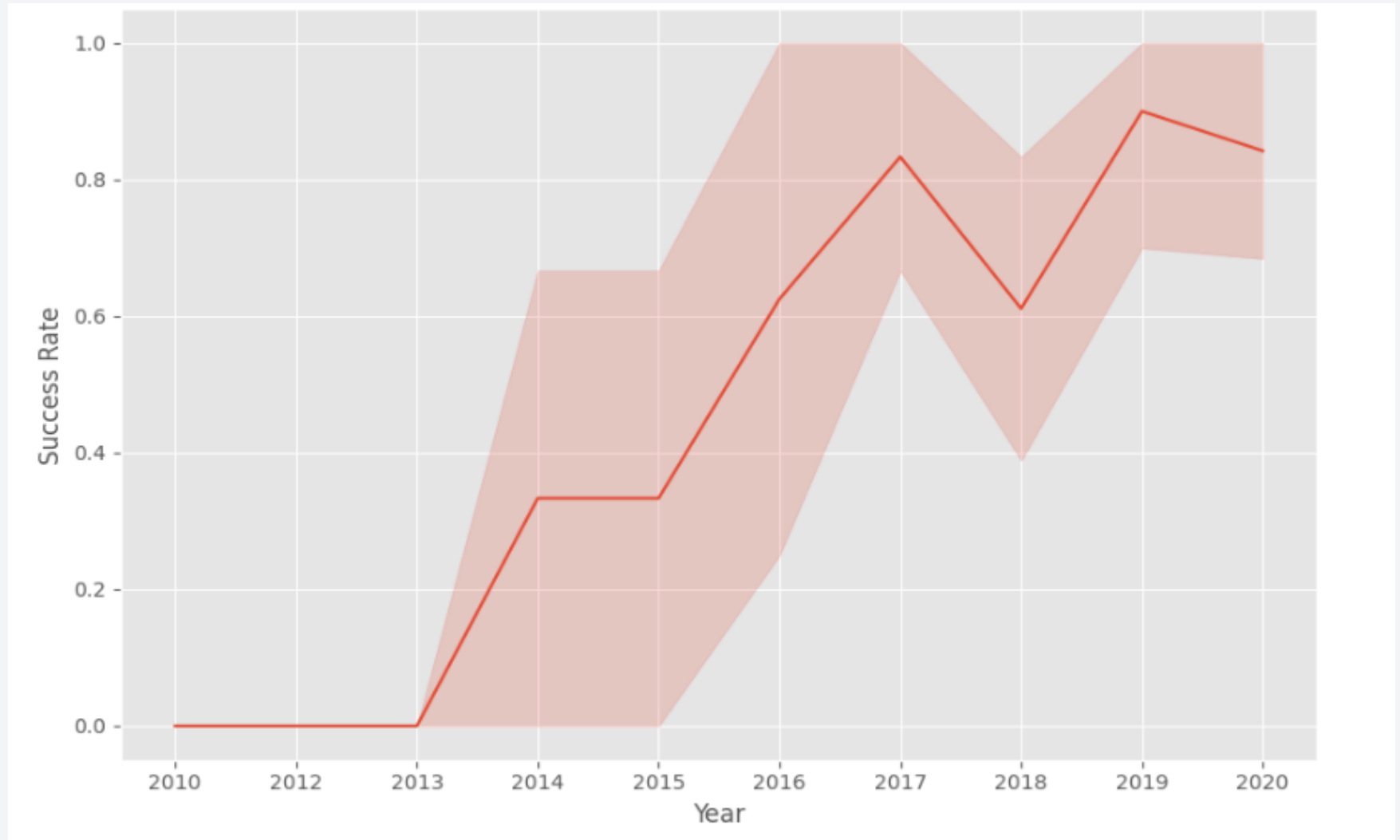
# All Launch Site Names

- Find the names of the unique launch sites

- Used 'SELECT DISTINCT' statement to return only the unique launch sites from the 'LAUNCH_SITE' column of the SPACEXTBL table

## Task 1

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

In [31]:
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

* sqlite:///my_data1.db
Done.

Out[31]:

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

# Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`



## Task 2

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

```
In [72]:   %sql SELECT * FROM 'SPACEXTBL' WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[72]:

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing _Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 04-06-2010 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 08-12-2010 | 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) |
| 22-05-2012 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 08-10-2012 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 01-03-2013 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

- Used 'LIKE' command with '%' wildcard in 'WHERE' clause to select and dispay a table of all records where launch sites begin with the string 'CCA'

34

# Total Payload Mass

- Calculate and Display the total payload carried by boosters from NASA

## Task 3

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

```
In [17]:   %sql SELECT SUM(PAYLOAD_MASS__KG_) as "Total Payload Mass(Kgs)", Customer FROM 'SPACEXTBL' WHERE Customer = 'NASA (CRS)';
```

 * sqlite:///my_data1.db
Done.

Out[17]: 

| Total Payload Mass(Kgs) | Customer |
|---|---|
| 45596 | NASA (CRS) |

- Used the 'SUM()' function to return and dispaly the total sum of 'PAYLOAD_MASS_KG' column for Customer 'NASA(CRS'

# Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

## Task 4

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' WHERE Booster_Version LIKE 'F9 v1.1%';
```

 * sqlite:///my_data1.db
Done.

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

- Used the 'AVG()' function to return and dispaly the average payload mass carried by booster version F9 v1.1

# First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad

## Task 5

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)**

01-05-2017

- Used the 'MIN()' function to return and dispaly the first (oldest) date when first successful landing outcome on ground pad '*Success (ground pad)*' happened.

# Successful Drone Ship Landing with Payload between 4000 and 6000

- List of Boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000



## Task 6

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 * FROM 'SPACEXTBL'
```

```
]:   %sql SELECT DISTINCT Booster_Version, Payload FROM SPACEXTBL WHERE "Landing _Outcome" = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOA
```

```
     * 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 |

- Used 'Select Distinct' statement to return and list the 'unique' names of boosters with operators >4000 and <6000 to only list booster with payloads btween 4000-6000 with landing outcome of 'Success (drone ship)'.

# Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

## Task 7

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 |

- Used the 'COUNT()' together with the 'GROUP BY' statement to return total number of missions outcomes

# Boosters Carried Maximum Payload

- List of the boosters which have carried the maximum payload mass

```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS__KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS__KG_" = (SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL)
```

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

| Booster_Version | Payload | PAYLOAD_MASS__KG_ |
|---|---|---|
| F9 B5 B1048.4 | Starlink 1 v1.0, SpaceX CRS-19 | 15600 |
| F9 B5 B1049.4 | Starlink 2 v1.0, Crew Dragon in-flight abort test | 15600 |
| F9 B5 B1051.3 | Starlink 3 v1.0, Starlink 4 v1.0 | 15600 |
| F9 B5 B1056.4 | Starlink 4 v1.0, SpaceX CRS-20 | 15600 |
| F9 B5 B1048.5 | Starlink 5 v1.0, Starlink 6 v1.0 | 15600 |
| F9 B5 B1051.4 | Starlink 6 v1.0, Crew Dragon Demo-2 | 15600 |
| F9 B5 B1049.5 | Starlink 7 v1.0, Starlink 8 v1.0 | 15600 |
| F9 B5 B1060.2 | Starlink 11 v1.0, Starlink 12 v1.0 | 15600 |
| F9 B5 B1058.3 | Starlink 12 v1.0, Starlink 13 v1.0 | 15600 |
| F9 B5 B1051.6 | Starlink 13 v1.0, Starlink 14 v1.0 | 15600 |
| F9 B5 B1060.3 | Starlink 14 v1.0, GPS III-04 | 15600 |
| F9 B5 B1049.7 | Starlink 15 v1.0, SpaceX CRS-21 | 15600 |

- Using a Subquerry to return and pass the Max payload and used it list all the boosters that have carried the Max payload of 15600kgs

# 2015 Launch Records

- List of failed landing outcomes in drone ship, with their booster versions, and launch site names in 2015

## Task 9

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.

```sql
%sql SELECT substr(Date,7,4), substr(Date, 4, 2),"Booster_Version", "Launch_Site", Payload, "PAYLOAD_MASS__KG_", "Mission_Outcome", "Landing _Outcome"
```

* sqlite:///my_data1.db
Done.

| substr(Date,7,4) | substr(Date, 4, 2) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Mission_Outcome | Landing _Outcome |
|---|---|---|---|---|---|---|---|
| 2015 | 01 | F9 v1.1 B1012 | CCAFS LC-40 | SpaceX CRS-5 | 2395 | Success | Failure (drone ship) |
| 2015 | 04 | F9 v1.1 B1015 | CCAFS LC-40 | SpaceX CRS-6 | 1898 | Success | Failure (drone ship) |

- Used the 'subsrt()' in the select statement to get the month and year from the date column where substr(Date,7,4)='2015' for year and Landing_outcome was 'Failure (drone ship)' and return the records nmatching the filter.

41

# 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



## Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```sql
%sql SELECT * FROM SPACEXTBL WHERE "Landing _Outcome" LIKE 'Success%' AND (Date BETWEEN '04-06-2010' AND '20-03-2017') ORDER BY Date DESC;
```

* sqlite:///my_data1.db
Done.

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing _Outcome |
|------|-----------|-----------------|-------------|---------|-------------------|-------|----------|-----------------|------------------|
| 19-02-2017 | 14:39:00 | F9 FT B1031.1 | KSC LC-39A | SpaceX CRS-10 | 2490 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 18-10-2020 | 12:25:57 | F9 B5 B1051.6 | KSC LC-39A | Starlink 13 v1.0, Starlink 14 v1.0 | 15600 | LEO | SpaceX | Success | Success |
| 18-08-2020 | 14:31:00 | F9 B5 B1049.6 | CCAFS SLC-40 | Starlink 10 v1.0, SkySat-19, -20, -21, SAOCOM 1B | 15440 | LEO | SpaceX, Planet Labs, PlanetIQ | Success | Success |
| 18-07-2016 | 04:45:00 | F9 FT B1025.1 | CCAFS LC-40 | SpaceX CRS-9 | 2257 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 18-04-2018 | 22:51:00 | F9 B4 B1045.1 | CCAFS SLC-40 | Transiting Exoplanet Survey Satellite (TESS) | 362 | HEO | NASA (LSP) | Success | Success (drone ship) |

Section 3
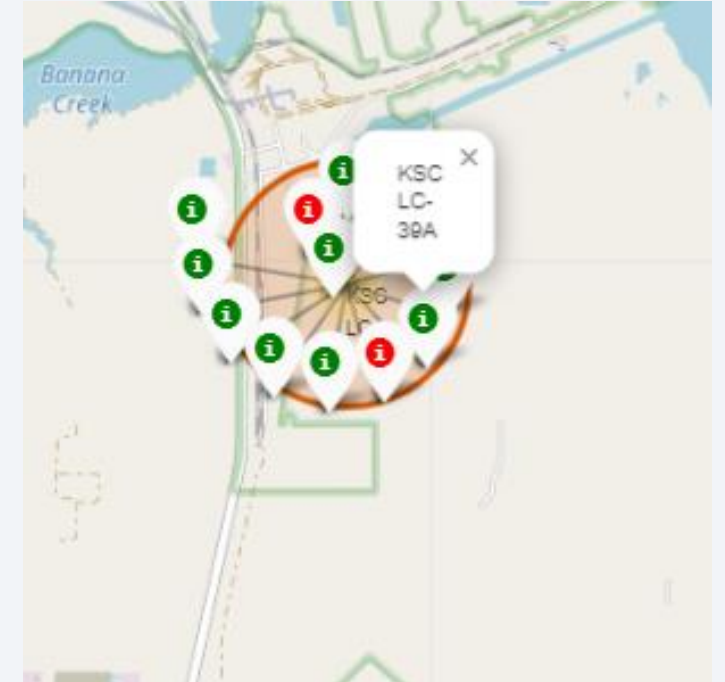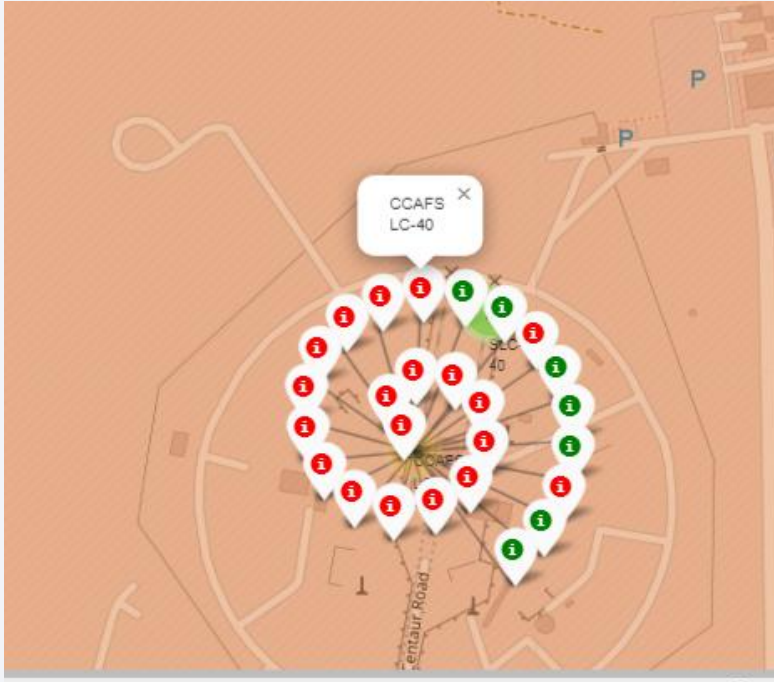
# Launch Sites
# Proximities Analysis

# Markers of all launch sites on global map
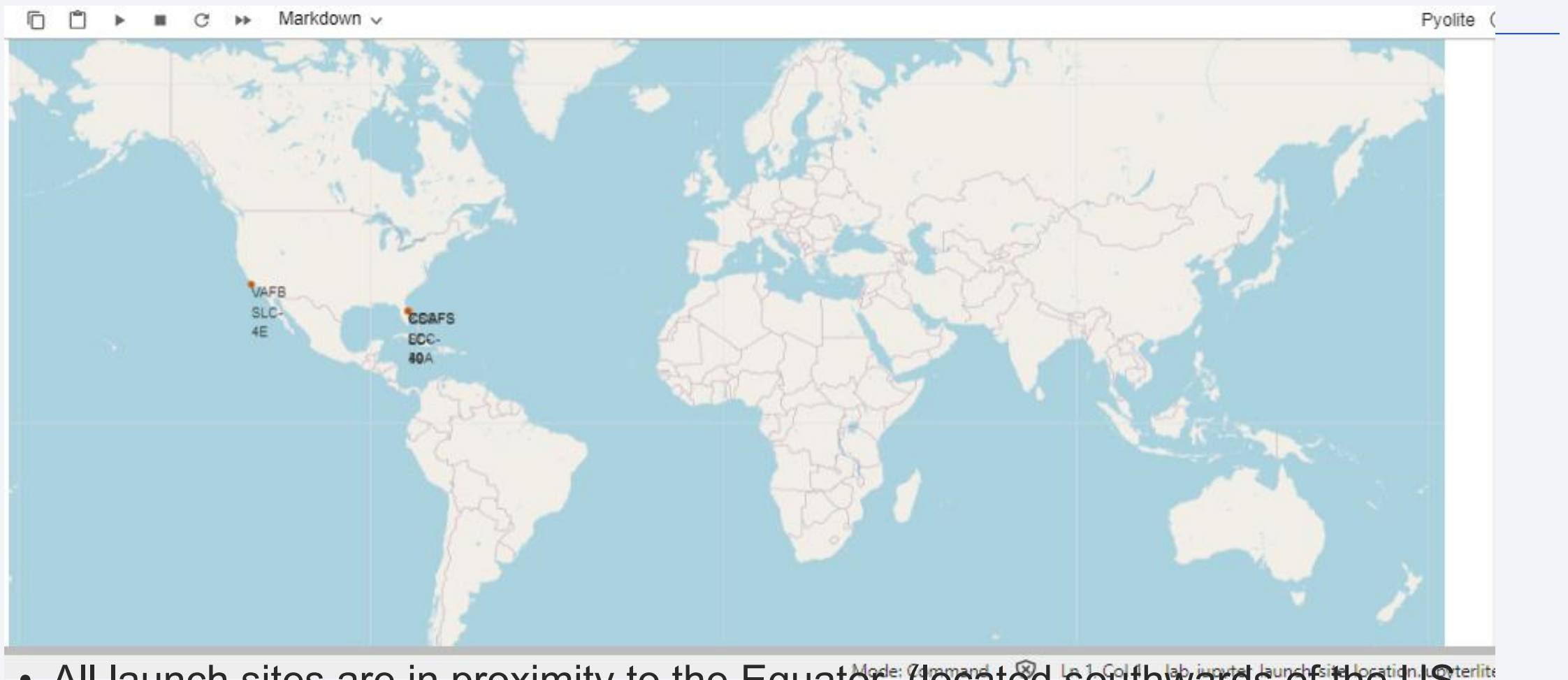


- All launch sites are in proximity to the Equator, (located southwards of the US map). Also all the laumch sites are in very close proximity to the coast.

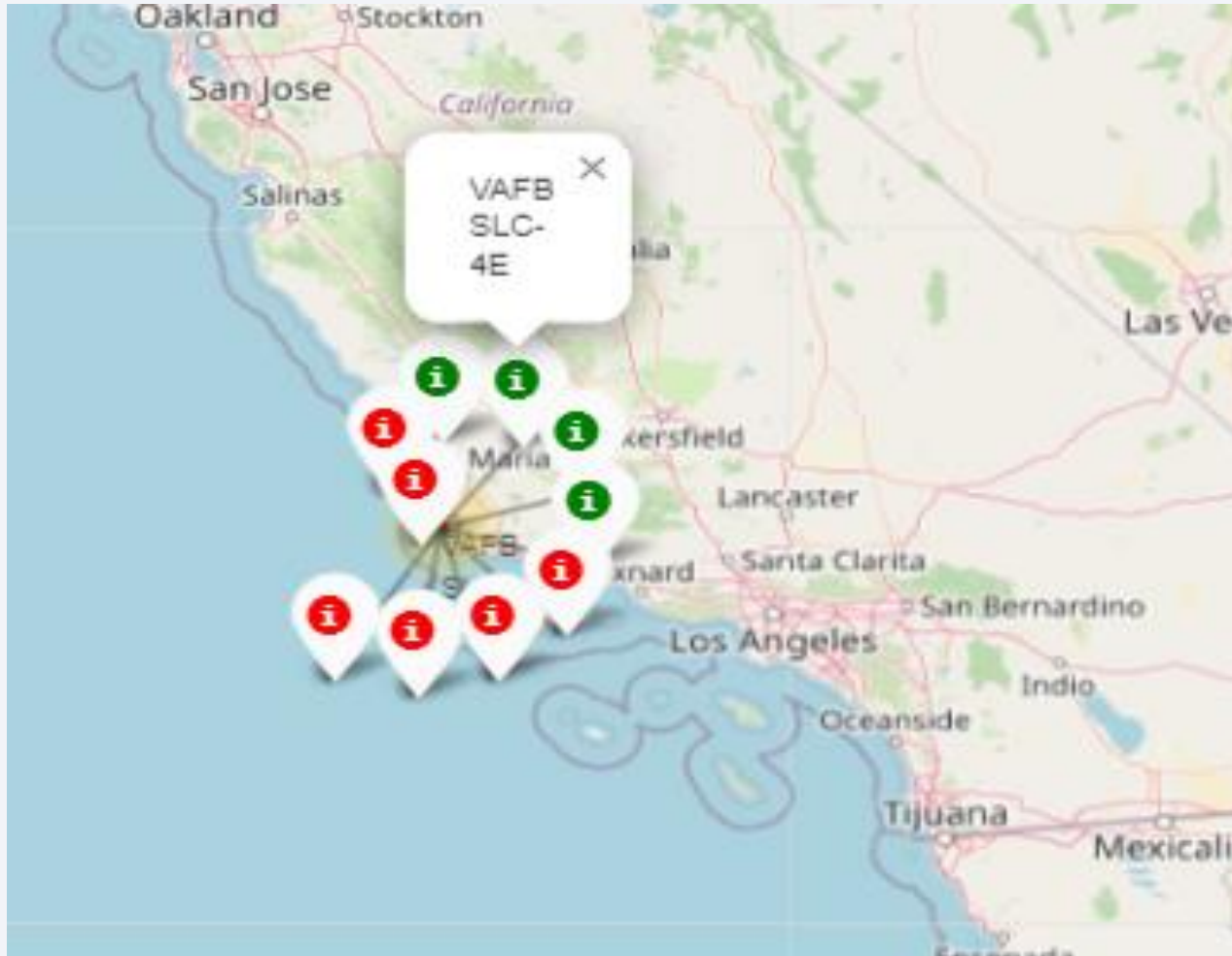# Launch outcomes for each site on the map With Color Markers

## Florida Sites



- In the Eastern coast (Florida) Launch site KSC LC-39A has relatively high success rates compared to CCAFS SLC-40 & CCAFS LC-40.

# Launch outcomes for each site on the map With Color Markers
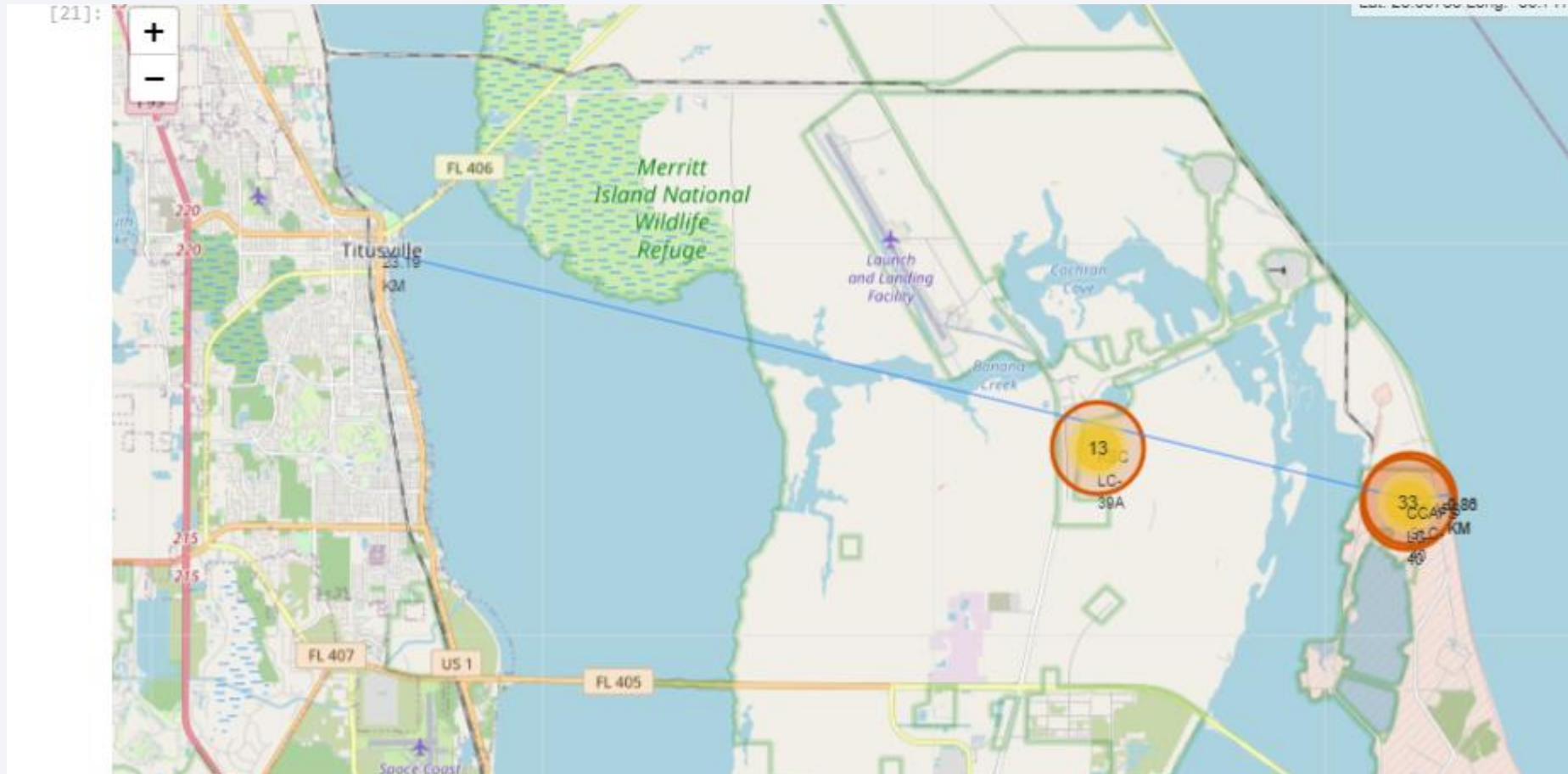
## West Coast/ Carlifonia



- In the West Coast (Californai) Launch site VAFB SLC-4E has relatively lower success rates 4/10 compared to KSC LC-39A launch site in the Eastern Coast of Florida.

# Distances between a launch site to its proximities



- Launch site **CCAFS SLC-40** proximity to coastline is 0.86km
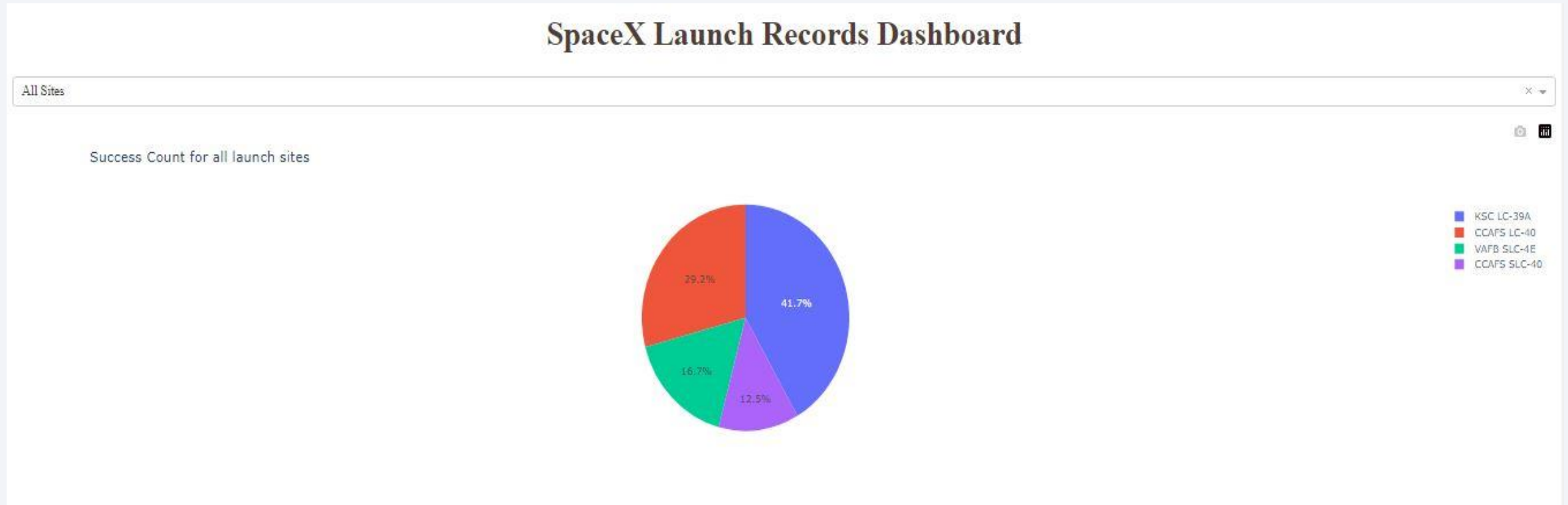
# Distances between a launch site to its proximities



- Launch site CCAFS SLC-40 closest to highway (Washington Avenue) is 23.19km
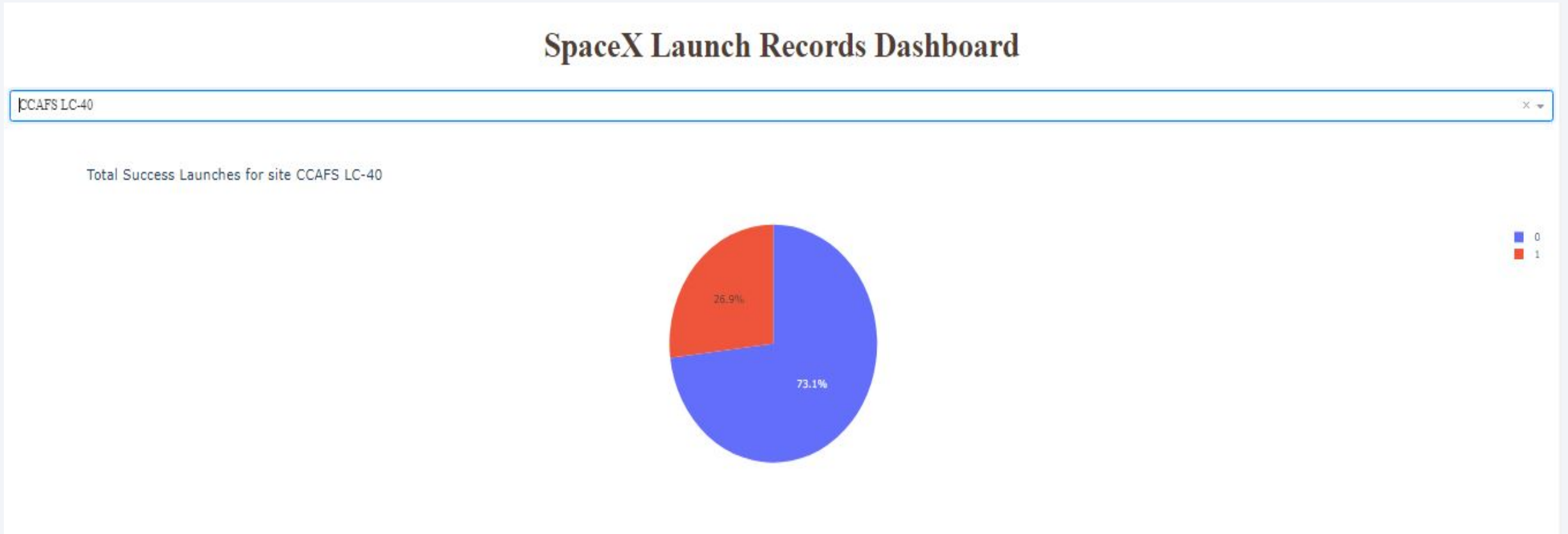
48

# Build a Dashboard with Plotly Dash

# Pie-Chart for launch success count for all sites



- Launch site KSC LC-39A has the highest launch success rate at 42% followed by  CCAFS LC-40 at 29%, VAFB SLC-4E at 17% and lastly launch site CCAFS SLC-40 with a success rate of 13%

# Pie chart for the launch site with 2nd highest launch success ratio



**SpaceX Launch Records Dashboard**

CCAFS LC-40

Total Success Launches for site CCAFS LC-40

26.9%

73.1%

0
1

- Launch site CCAFS LC-40 had the 2nd highest success ratio of 73% success against 27% failed launches

# Payload vs. Launch Outcome scatter plot for all sites
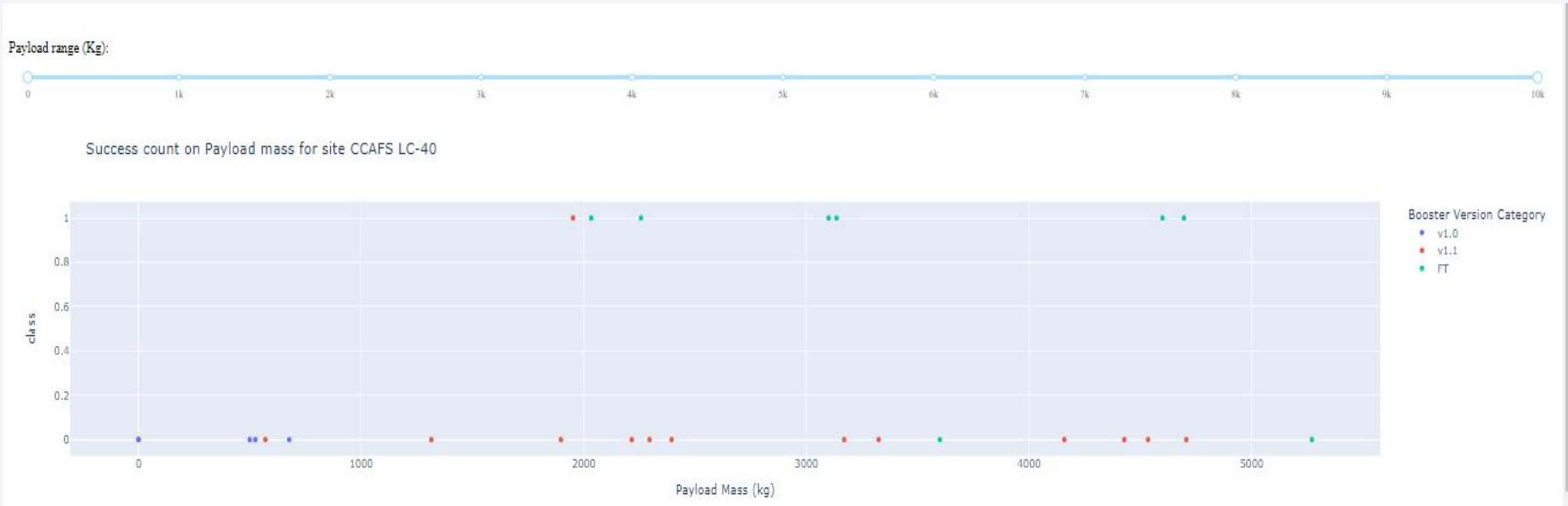


- For Launch site CCAFS LC-40 the booster version FT has the largest success rate from a payload mass of >2000kg

Section 5

# Predictive Analysis (Classification)

# Classification Models Accuracy

Out[68]:

| | 0 |
| --- | --- |
| Method | Test Data Accuracy |
| Logistic_Reg | 0.833333 |
| SVM | 0.833333 |
| Decision Tree | 0.833333 |
| KNN | 0.833333 |

*All the methods perform equally on the test data: i.e. They all have the same accuracy of 0.833333 on the test Data*

# Confusion Matrix

- All the 4 classification model had the same confusion matrixes and were able equally distinguish between the different classes. The major problem is false positives for all the models.
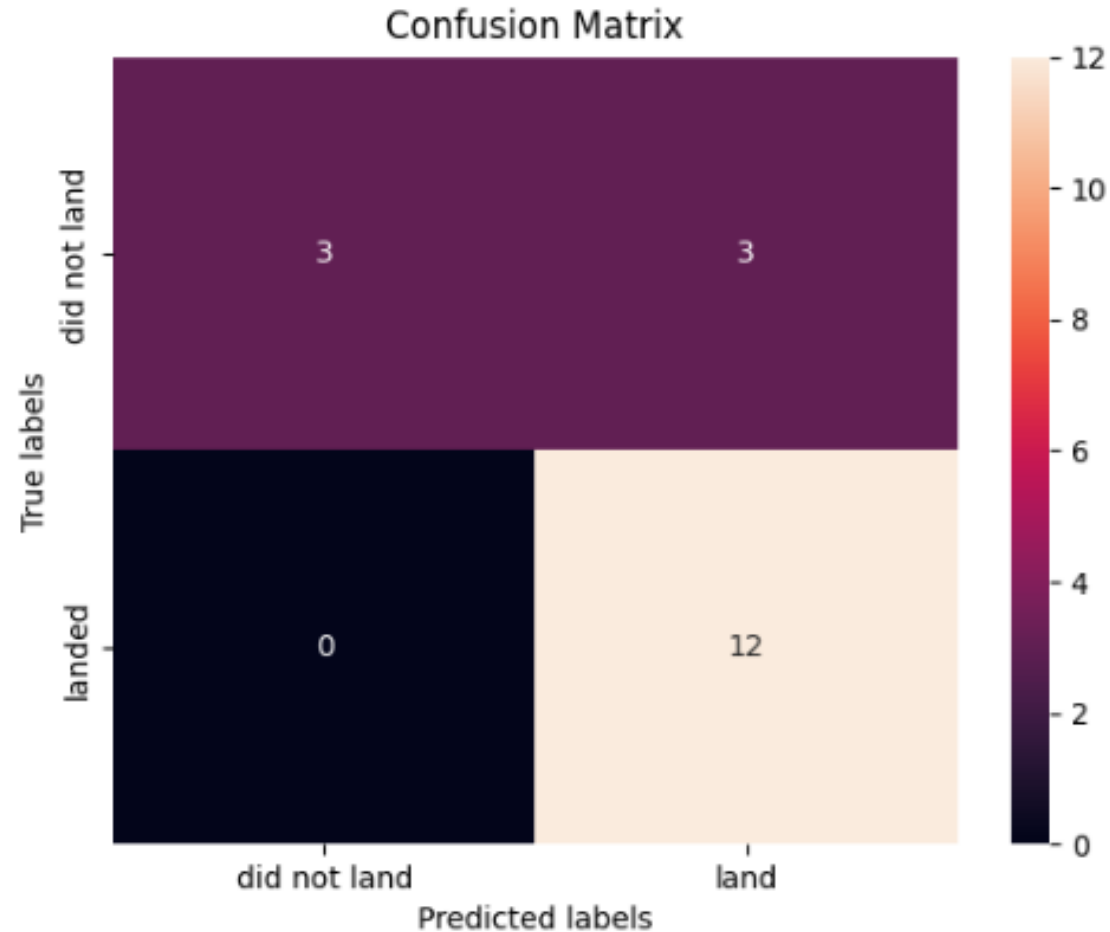
# Conclusions

- Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.

- We can deduce that, as the flight number increases in each of the 3 launcg sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80th flight

- If you observe Payload 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).

- Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate.

- LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit

# Conclusions Cont….

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS. However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here

- Anf finally the sucess rate since 2013 kept increasing till 2020.

Thank you!