

# Winning Space Race with Data Science

<Name>  
<Date>



# Outline

---

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

# Executive Summary

---

- Summary of methodologies
- Summary of all results

# Introduction

---

## Project background

Analyze rocket launching data of SpaceX company. We will explore the success rates, payload trends, , temporal patterns, landing outcomes, launch site, and rocket performance.

## Problems statement

- What success factor of SpaceX launches?
- How has payload mass and types evolved over time, and do they correlate with launch outcomes?
- What patterns exist in the outcomes of first stage landings?
- How do launch site choices affect mission success?
- Are there any notable trends or seasonality in launch history?

Section 1

# Methodology

# Methodology

---

## Executive Summary

- Data collection methodology:
  - Data collection by using SpaceX API and web scraping from wikipedia
- Perform data wrangling
  - Use One-hot feature 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

---

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using `.json()` function call and turn it into a pandas dataframe using `.json_normalize()`.
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

# Data Collection

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is <https://github.com/thawatchai-io/DS/blob/main/jupyter-labs-spacex-data-collection-api.ipynb>

1. Get request for rocket launch data using API

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"  
In [7]: response = requests.get(spacex_url)
```

2. Use json\_normalize method to convert json result to dataframe

```
In [12]: # Use json_normalize method to convert the json result into a dataframe  
# decode response content as json  
static_json_df = res.json()  
  
In [13]: # apply json_normalize  
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

```
In [30]: rows = data_falcon9['PayloadMass'].values.tolist()[0]  
  
df_rows = pd.DataFrame(rows)  
df_rows = df_rows.replace(np.nan, PayloadMass)  
  
data_falcon9['PayloadMass'][0] = df_rows.values  
data_falcon9
```

# Data Collection – SpaceX API

---

- Present your data collection with SpaceX REST calls using key phrases and flowcharts
- Add the GitHub URL of the completed SpaceX API calls notebook (must include completed code cell and outcome cell), as an external reference and peer-review purpose

Place your flowchart of SpaceX API calls here

# Data Collection - Scraping

- Use BeautifulSoup in order to web scrapping to Falcon 9 launch records
- We converted into a pandas dataframe.
- The link to the notebook is <https://github.com/thawatchai-io/DS/blob/main/jupyter-labs-webscraping.ipynb>

```
1. Apply HTTP Get method to request the Falcon 9 rocket launch page
```

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

```
In [5]: # 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
```

```
Out[5]: 200
```

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

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content  
soup = BeautifulSoup(html_data.text, '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]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

```
3. Extract all column names from the HTML table header
```

```
In [10]: column_names = []  
  
# Apply find_all() function with 'th' element on first_launch_table  
# Iterate each th element and apply the provided extract_column_from_header() to get a column name  
# Append the Non-empty column name ('if name is not None and len(name) > 0') into a list called column_names  
  
element = soup.find_all('th')  
for row in range(len(element)):  
    try:  
        name = extract_column_from_header(element[row])  
        if (name is not None and len(name) > 0):  
            column_names.append(name)  
    except:  
        pass
```

```
4. Create a dataframe by parsing the launch HTML tables
```

```
5. Export data to csv
```

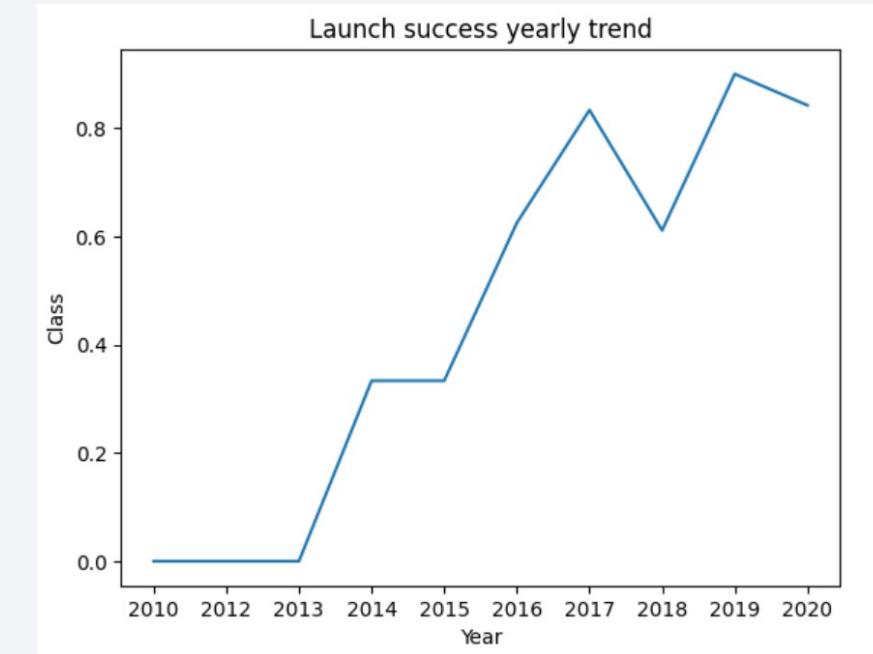
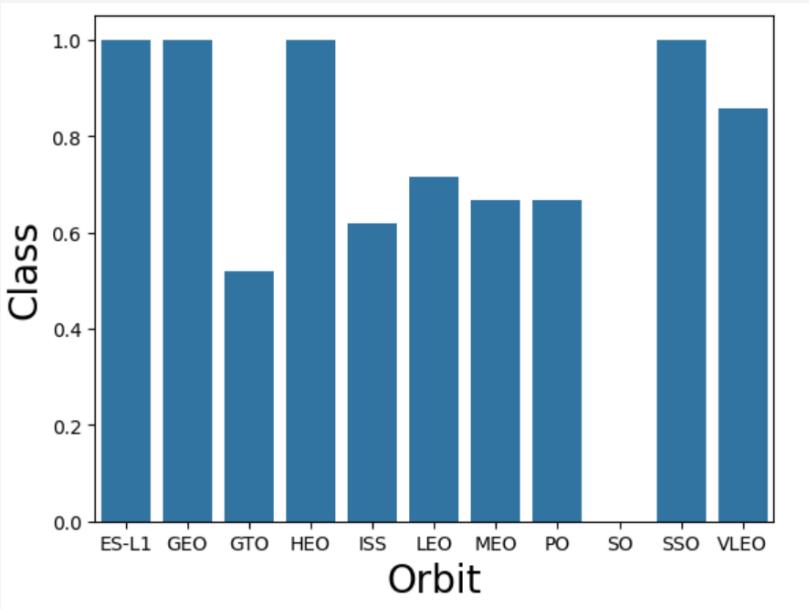
# Data Wrangling

---

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is  
<https://github.com/thawatchai-io/DS/blob/main/jupyter-labs-webscraping.ipynb>

# EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.



- The link to the notebook is  
<https://github.com/thawatchai-io/DS/blob/main/jupyter-labs-eda-dataviz.ipynb>

# EDA with SQL

---

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook is [https://github.com/thawatchai-io/DS/blob/main/jupyter-labs-eda-sql-coursera\\_sqlite.ipynb](https://github.com/thawatchai-io/DS/blob/main/jupyter-labs-eda-sql-coursera_sqlite.ipynb)

# Build an Interactive Map with Folium

---

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.

# Build a Dashboard with Plotly Dash

---

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is [https://github.com/thawatchai-io/DS/blob/main/spacex\\_dash\\_app.py](https://github.com/thawatchai-io/DS/blob/main/spacex_dash_app.py)

# Predictive Analysis (Classification)

---

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is [https://github.com/thawatchai-io/DS/blob/main/SpaceX\\_Machine\\_Learning\\_Prediction.ipynb](https://github.com/thawatchai-io/DS/blob/main/SpaceX_Machine_Learning_Prediction.ipynb)

# Results

---

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

The background of the slide features a complex, abstract digital visualization. It consists of numerous thin, glowing lines that create a sense of depth and motion. The lines are primarily blue and red, with some green and purple highlights. They form a grid-like structure that curves and twists across the frame, resembling a three-dimensional space or a network of data points. The overall effect is futuristic and dynamic.

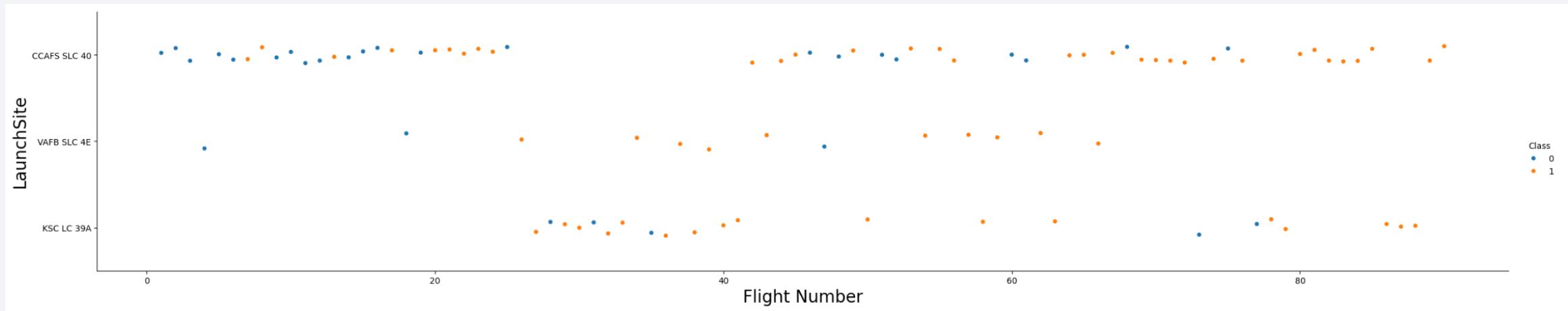
Section 2

## Insights drawn from EDA

# Flight Number vs. Launch Site

---

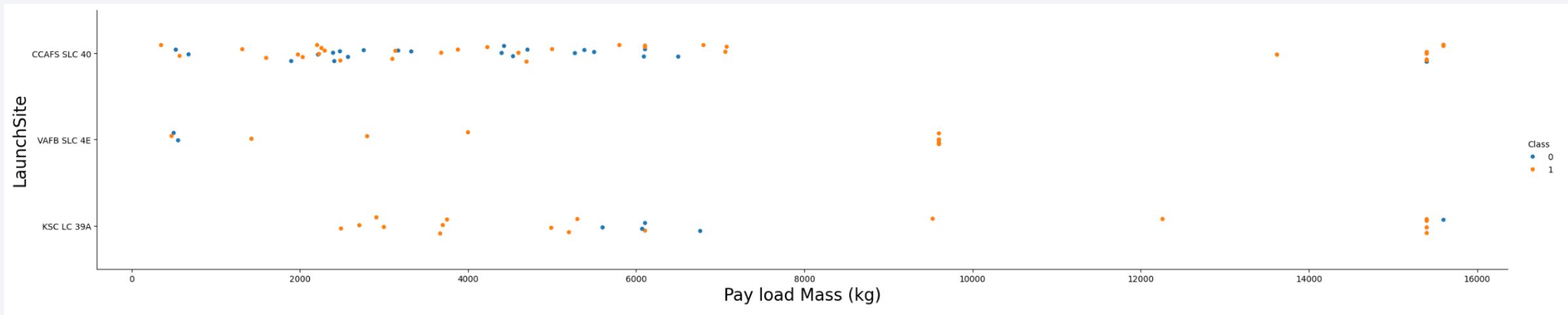
- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



# Payload vs. Launch Site

---

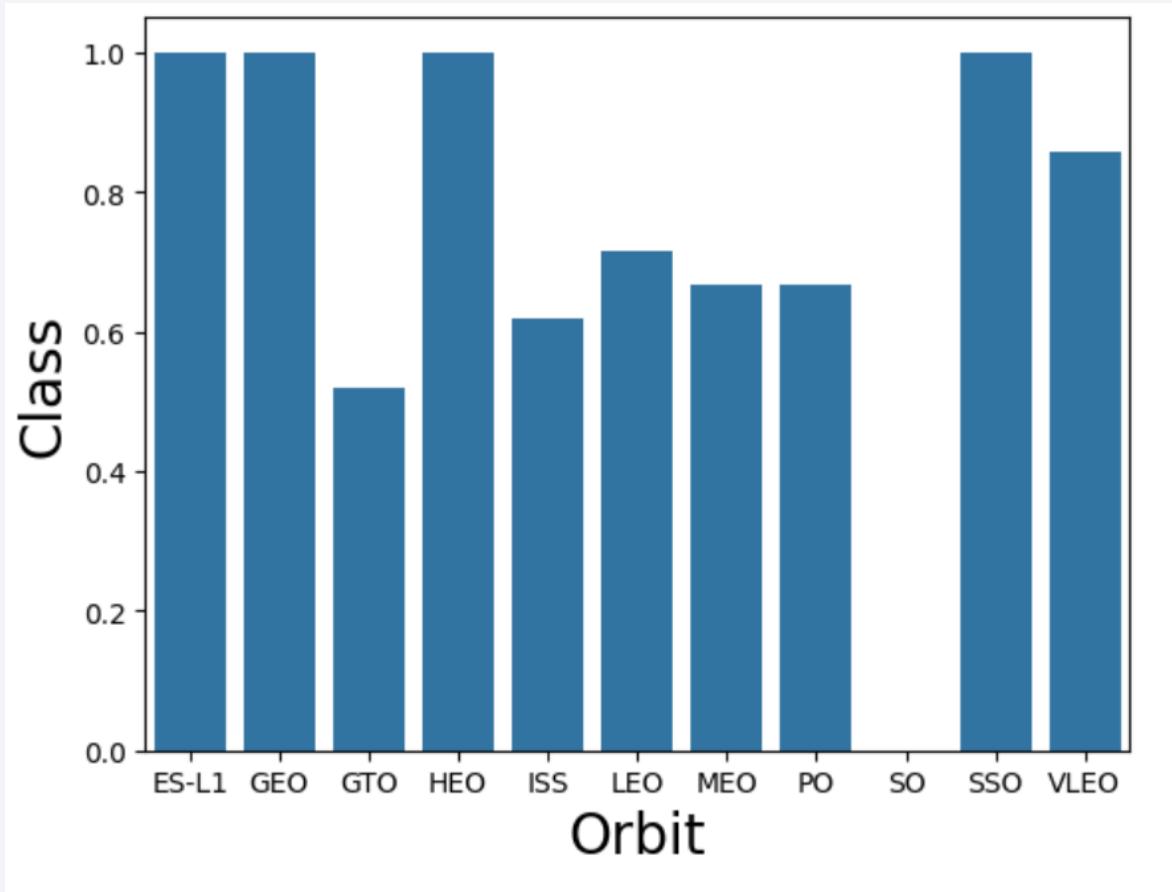
- The greater the payload mass for launch site CCAFS SLC 40 is higher success rate for rocket launch.



# Success Rate vs. Orbit Type

---

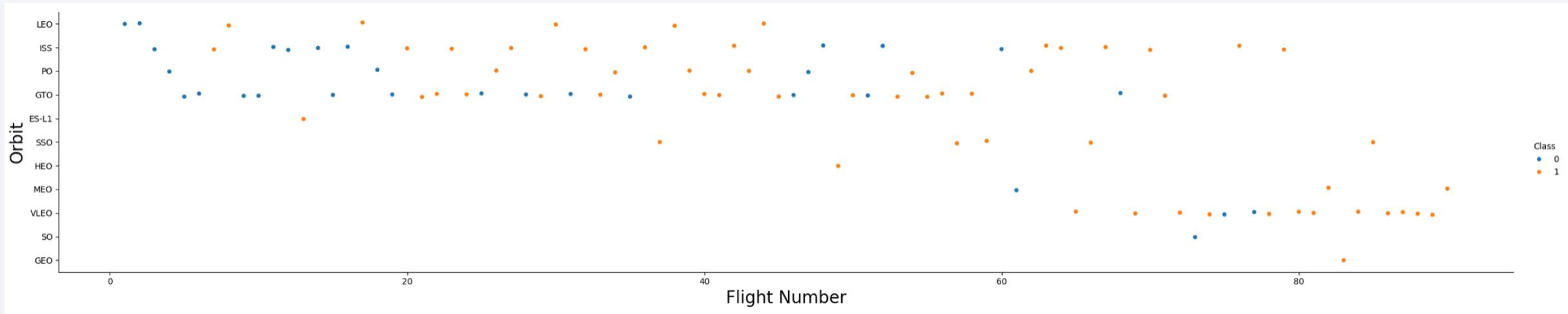
- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



# Flight Number vs. Orbit Type

---

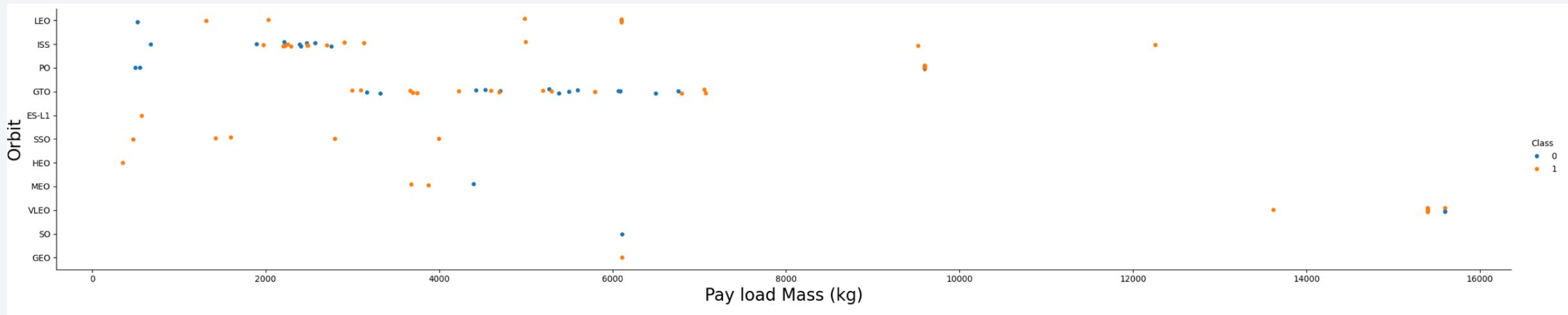
- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



# Payload vs. Orbit Type

---

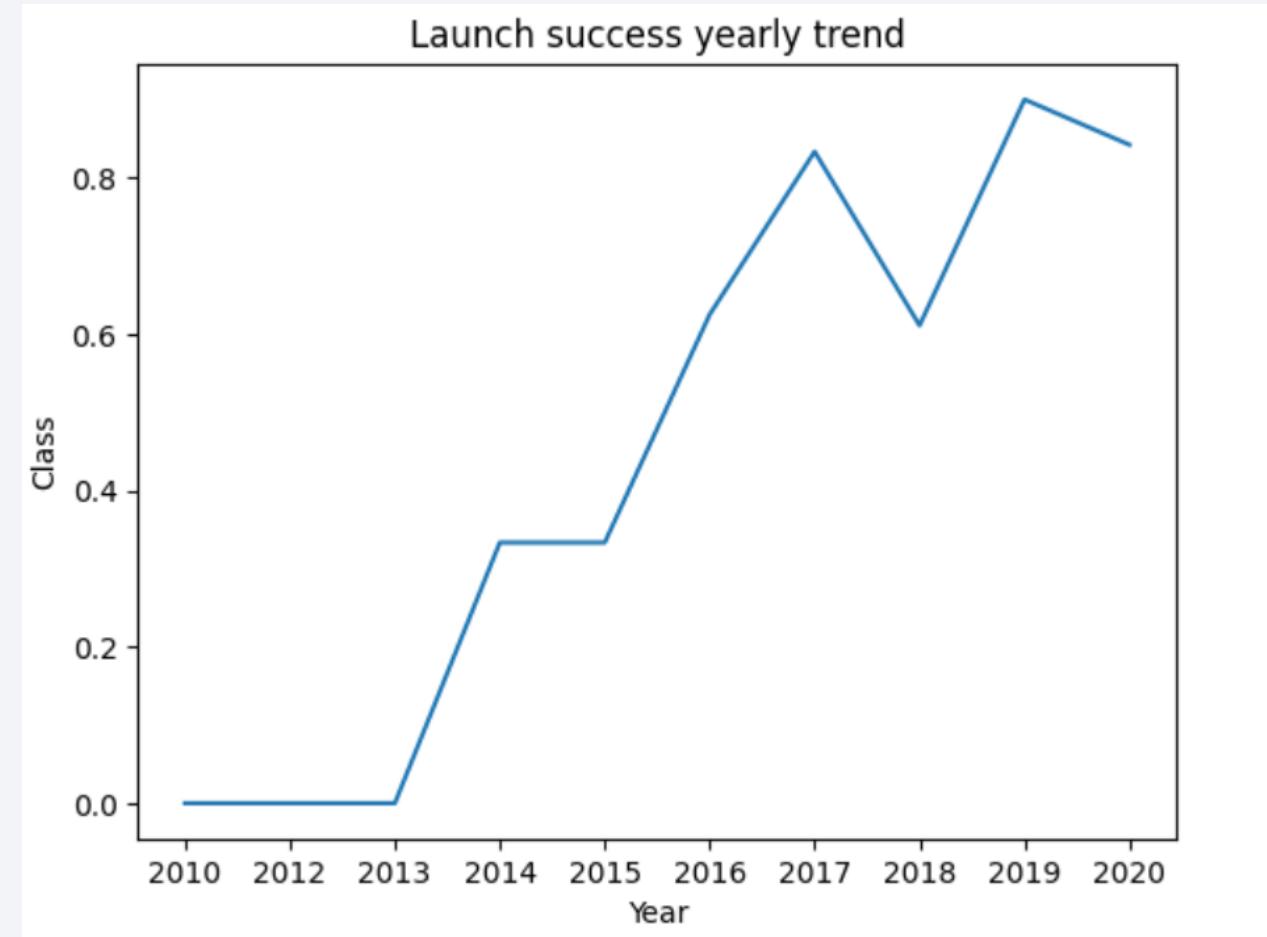
- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



# Launch Success Yearly Trend

---

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



# All Launch Site Names

---

```
%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

- Present your query result with a short explanation here
  - Use DISTINCT for select unique

# Launch Site Names Begin with 'CCA'

---

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

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	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

- Present your query result with a short explanation here
  - Use WHERE Launch\_Site LIKE 'CCA%' for select only begin with 'CCA'
  - Use LIMIT for limit output 5 records

# Total Payload Mass

---

- Calculate the total payload carried by boosters from NASA

```
: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer='NASA (CRS)'  
* sqlite:///my_data1.db  
Done.  
: SUM(PAYLOAD_MASS__KG_)  
-----  
45596
```

- Present your query result with a short explanation here
  - Use SUM function with PAYLOAD\_MASS\_\_KG\_\_
  - And WHERE Customer='NASA (CRS)' for select only from NASA

# Average Payload Mass by F9 v1.1

---

- Calculate the average payload mass carried by booster version 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 Like '%F9 v1.1%'
```

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

```
AVG(PAYLOAD_MASS__KG_)
```

```
2534.6666666666665
```

- Present your query result with a short explanation here
  - Use AVG function with PAYLOAD\_MASS\_\_KG\_\_
  - Use LIKE = '%F9 v1.1%' on Booster\_Version

# First Successful Ground Landing Date

---

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

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)'  
* sqlite:///my_data1.db  
Done.  
MIN(Date)  
2015-12-22
```

- Present your query result with a short explanation here
  - Use MIN function to find minimal date
  - WHERE Landing\_Outcome='Success (ground pad)' for select only successful landing outcome on ground pad

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

---

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

```
: %sql SELECT booster_version FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' and PAYLOAD_MASS_KG_ between 4000 and 6000
* sqlite:///my_data1.db
Done.
: Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2
```

- Present your query result with a short explanation here
  - Use BETWEEN 4000 and 6000
  - WHERE Landing\_Outcome = ‘Success (drone ship)’

# Total Number of Successful and Failure Mission Outcomes

---

- Calculate the total number of successful and failure mission outcomes

```
%sql SELECT Mission_Outcome, COUNT(*) FROM SPACEXTBL GROUP BY Mission_Outcome  
* sqlite:///my_data1.db  
Done.  


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


```

- Present your query result with a short explanation here
  - Use COUNT function and GROUP BY

# Boosters Carried Maximum Payload

---

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

```
: %sql SELECT Booster_Version FROM SPACEXTBL WHERE PAYLOAD_MASS_KG_ IN (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
```

- Present your query result with a short explanation here

- Use sub query and MAX function to find max payload value in order to query Booter\_Version

# 2015 Launch Records

---

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

```
%sql SELECT strftime('%m', Date) as Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE Landing_Outcome = 'Failure (drone ship)' and strftime('%Y', Date)='2015'  
* 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 |


```

- Present your query result with a short explanation here
  - Use WHERE Landing\_Outcome='Failure (drone ship)' and Year of Date = 2015

# 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(*) AS COUNT_LANDING_OUTCOME FROM SPACEXTBL WHERE date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY COUNT_LANDING_OUTCOME
* sqlite:///my_data1.db
Done.

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
```

- Present your query result with a short explanation here
  - Use COUNT and BETWEEN for select range date and GROUP BY for group count and ORDER for sorting

The background of the slide is a photograph taken from space at night. It shows the curvature of the Earth against a dark blue-black 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 right, the green and yellow glow of the aurora borealis is visible. The atmosphere of the Earth is thin and hazy, appearing as a light blue band near the horizon.

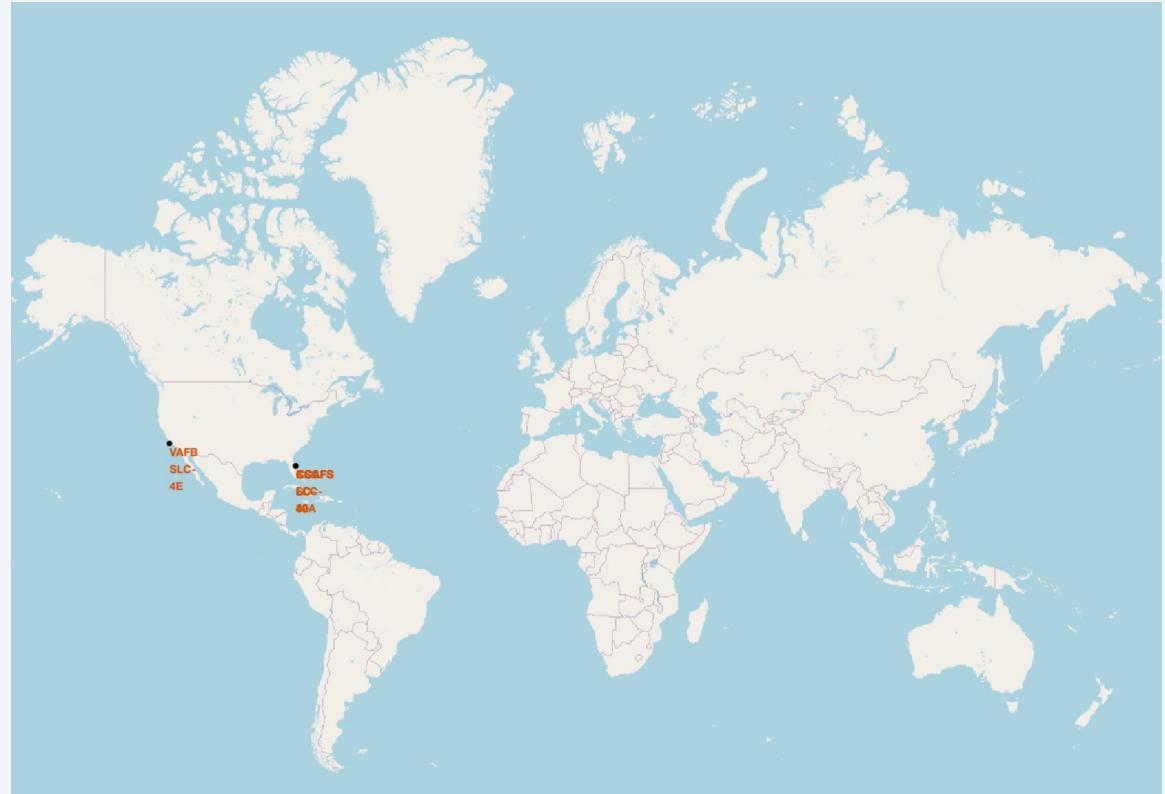
Section 3

# Launch Sites Proximities Analysis

# All launch site global map markers

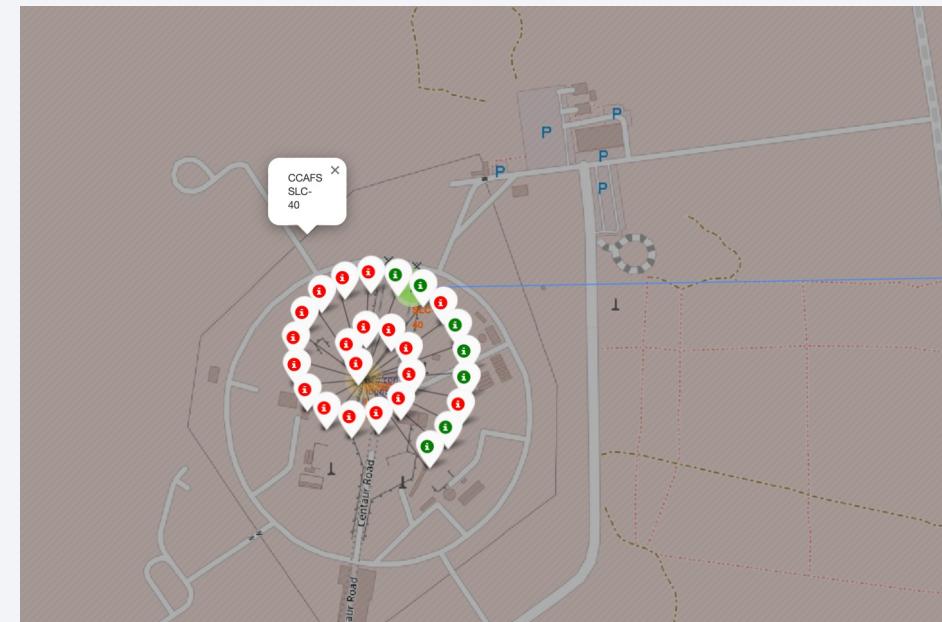
---

- SpaceX launch site are in the United States of America coasts. Florida and California



# Marker showing launch sites with color labels

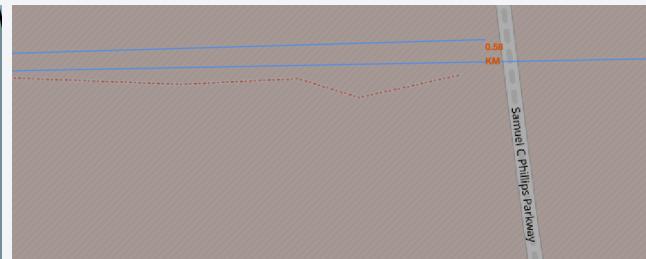
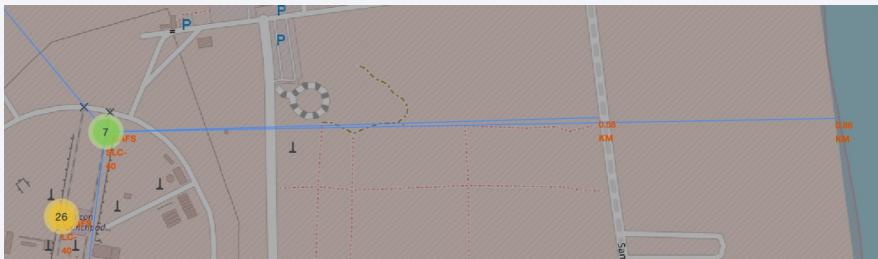
- Green color show successful launches
- Red color show failures



# Launch Site distance to landmarks

## From CCAFS SLC-40 site

- Distance from Melbourne Orlando Internation Airport 51.43KM
- Distance from NASA Railroad 1.28KM
- Distance from Samuel C Phillips Parkway 0.58KM
- Distance from Coastline 0.88KM



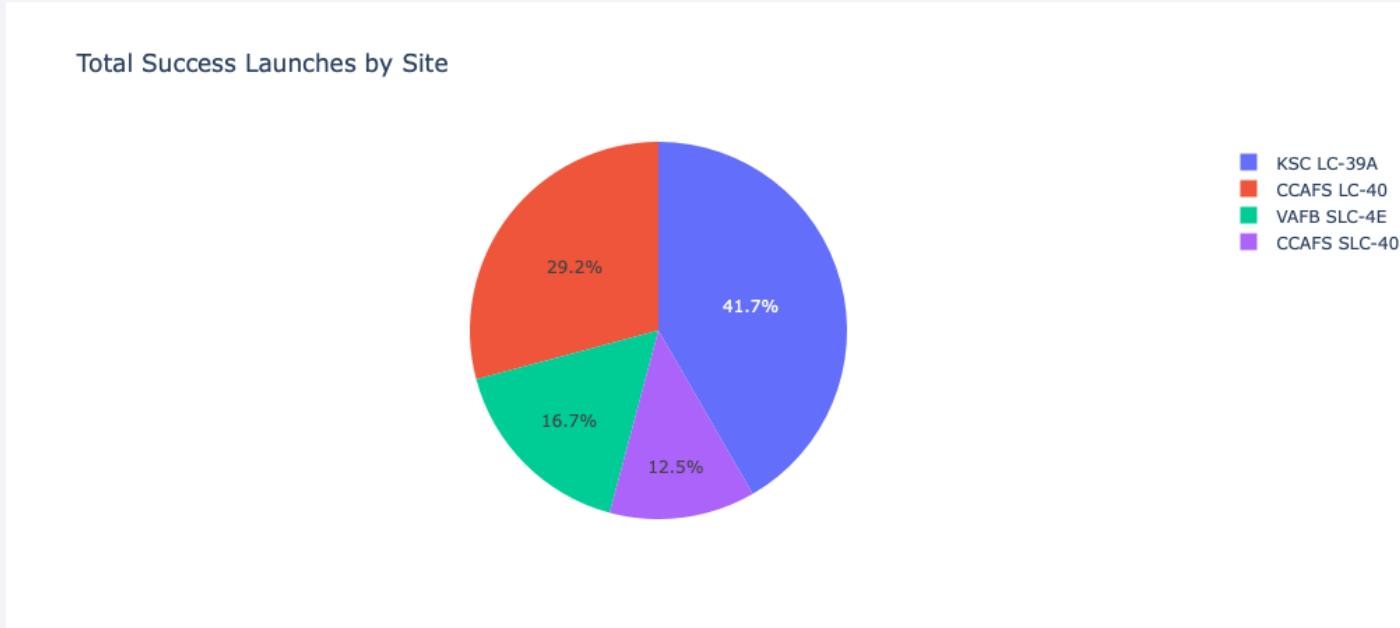
Section 4

# Build a Dashboard with Plotly Dash



# <Dashboard Screenshot 1>

---

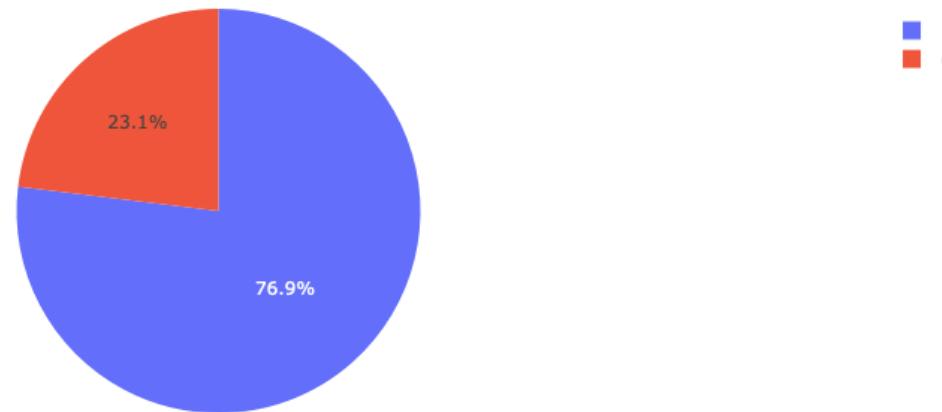


- KSC LC-39A site have number of success launches highest

## <Dashboard Screenshot 2>

---

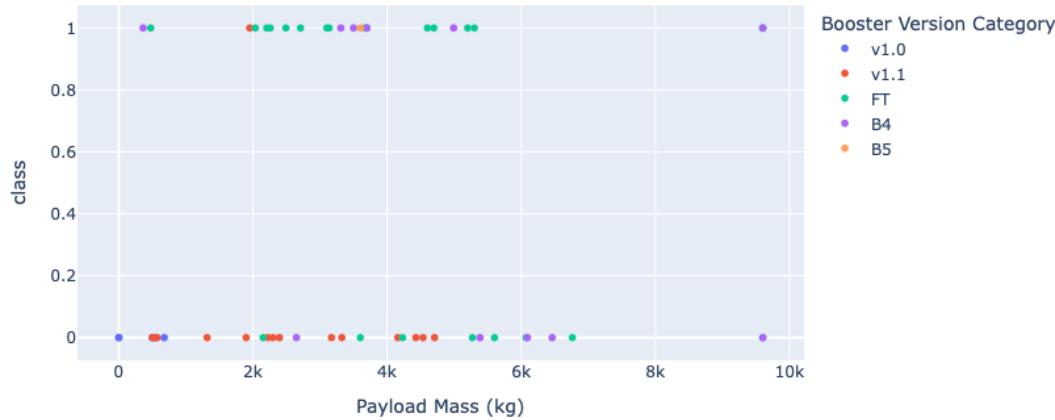
Total Success Launches for KSC LC-39A



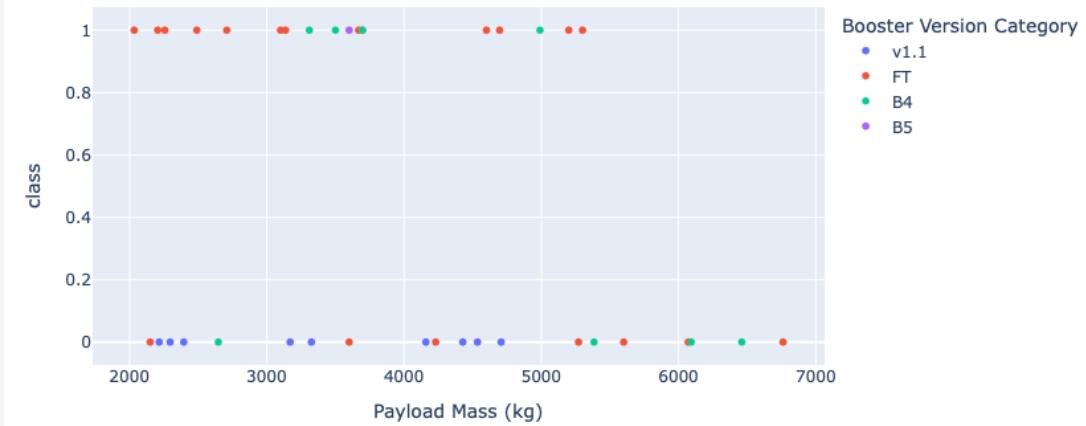
- KSC LC-39A has success rate 76.9%

# <Dashboard Screenshot 3>

Payload and Success for all Sites



Payload and Success for all Sites



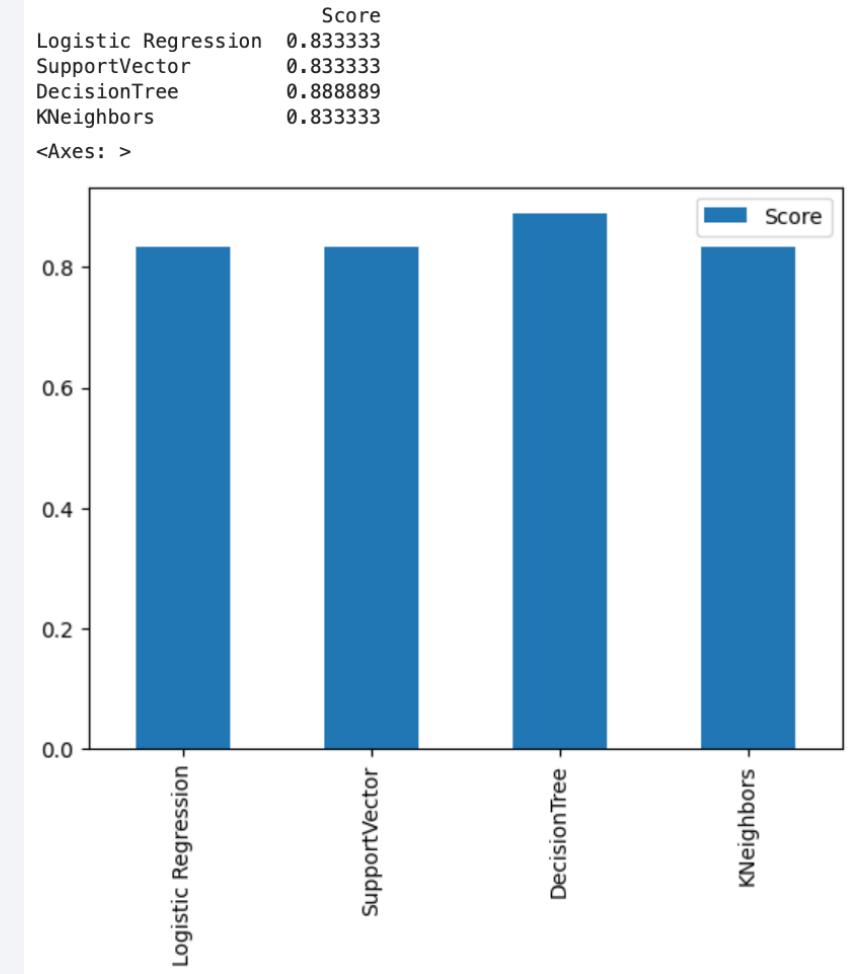
- Booster Version v1.0 will be at Payload Mass (Kg) between 0-1K

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

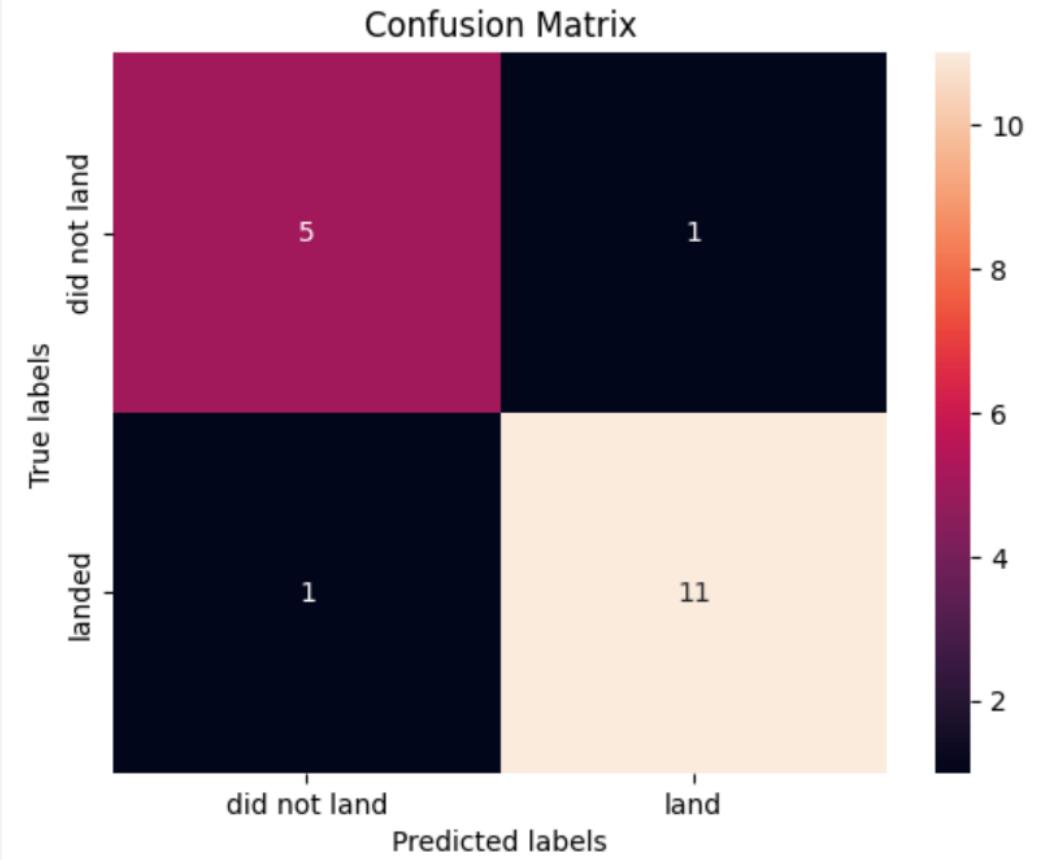
- From image,
  - The decision tree classifier is the model with the highest classification accuracy



# Confusion Matrix

- The confusion matrix for the decision tree classifier

```
yhat = tree_cv.predict(X_test)  
plot_confusion_matrix(Y_test,yhat)
```



# Conclusions

---

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

# Appendix

---

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!

