



# Winning Space Race with Data Science

Nishita  
20 May 2025



# Outline

---

- Resumen ejecutivo
- Introducción
- Metodología
- Resultados
- Conclusión
- Apéndice

# Resumen ejecutivo

---

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

# Introduction

---

- Project background and context
- 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. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.
- Problems you want to find answers
  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Section 1

# Methodology



# Methodology

---

## Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied 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

---

- Describe how data sets were collected.
  - 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.
- You need to present your data collection process use key phrases and flowcharts

# 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

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 - Scraping

- Present your web scraping process using key phrases and flowcharts
- Add the GitHub URL of the completed web scraping notebook, as an external reference and peer-review purpose

```
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=1027686921"

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 = []

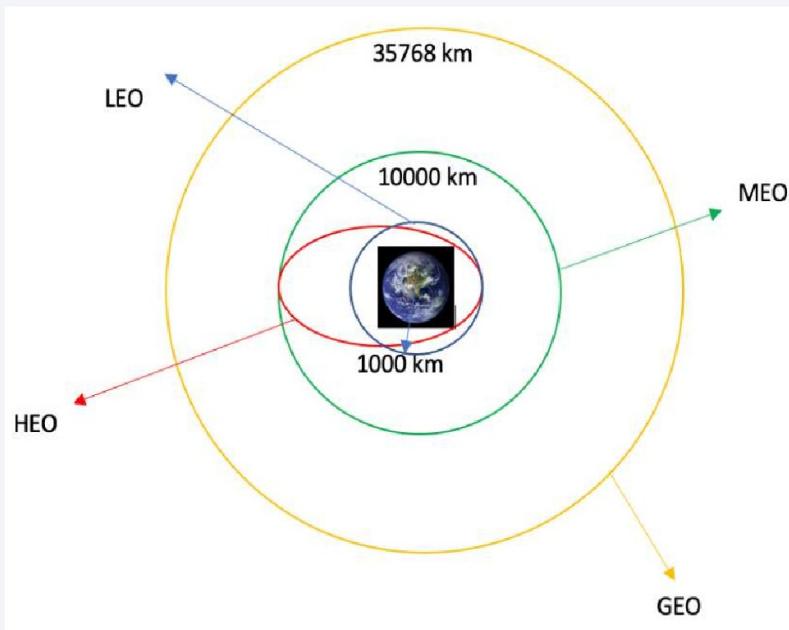
        # Apply find_all() function with "th" element on first_launch_table
        # Iterate each th element and apply the provided extract_column_from_header() to get a column name
        # Append the non-empty column name ('if name is not None and len(name) > 0') into a list called column_names

        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

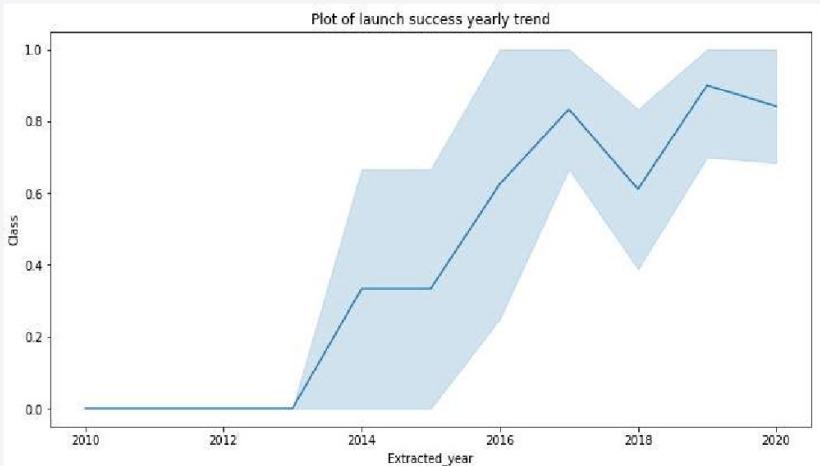
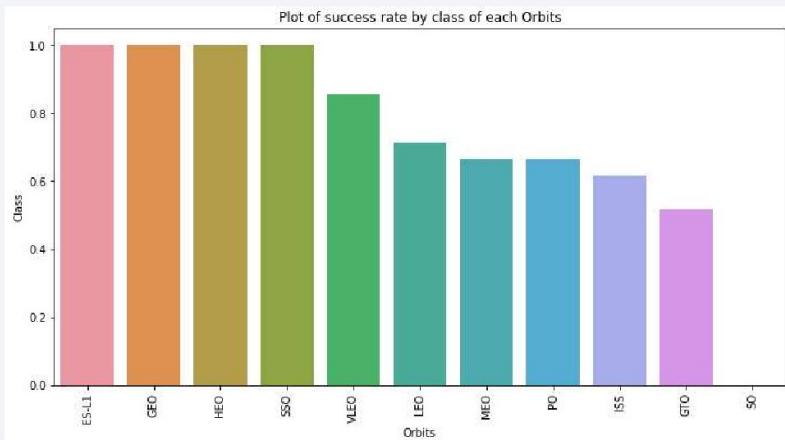
---



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.

# 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.



# 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.

# 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.

# Predictive Analysis (Classification)

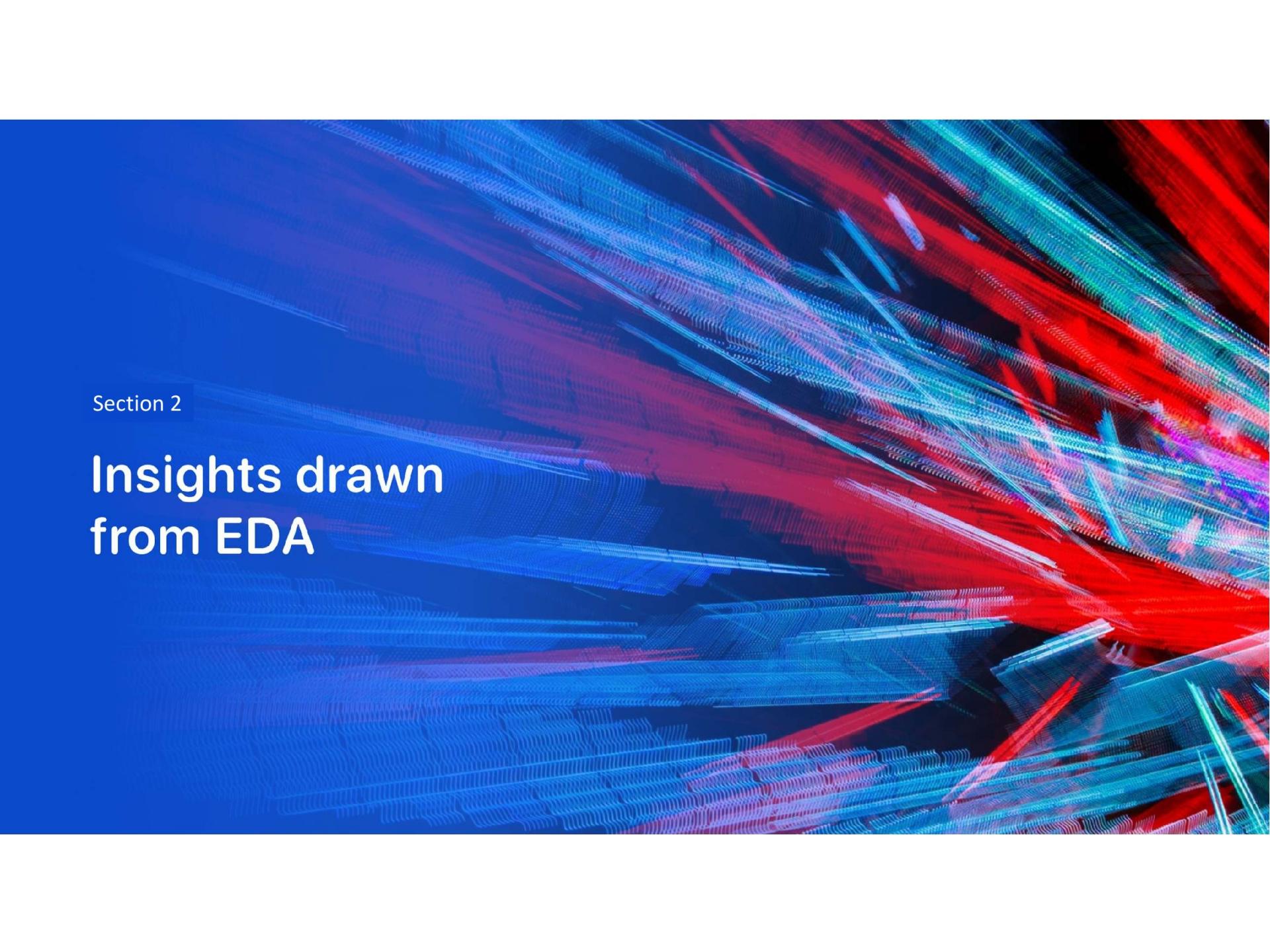
---

- 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

# Results

---

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

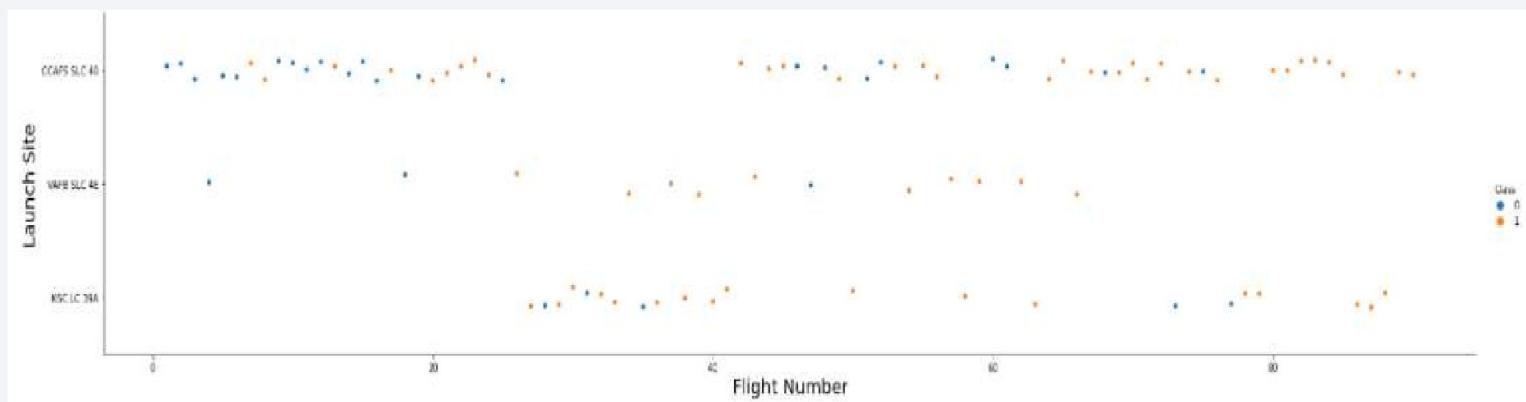
The background of the slide features a complex, abstract pattern of glowing lines. These lines are primarily blue and red, with some green and white highlights. They form a dense, flowing network that resembles a digital or quantum landscape. The lines are thick and appear to be composed of many small, glowing particles, creating a sense of depth and motion.

Section 2

## Insights drawn from EDA

# Flight Number vs. Launch Site

---

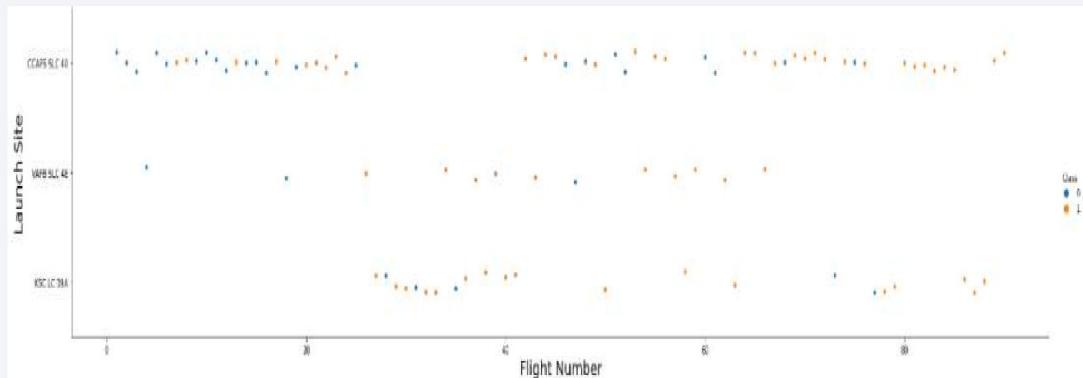


# Payload vs. Launch Site

- Show a scatter plot of Payload vs. Launch Site
- Show the screenshot of the scatter plot with explanations



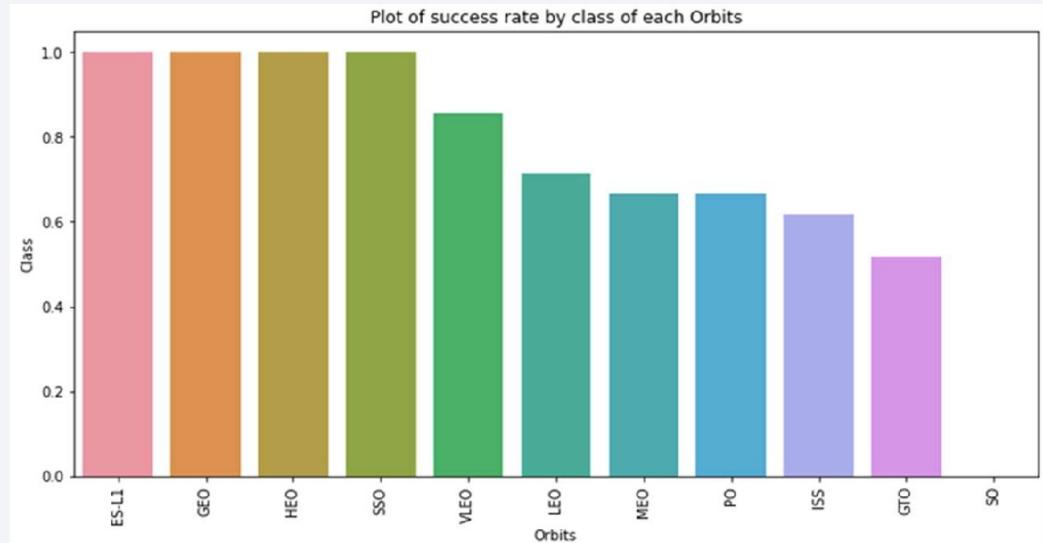
The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



# Success Rate vs. Orbit Type

---

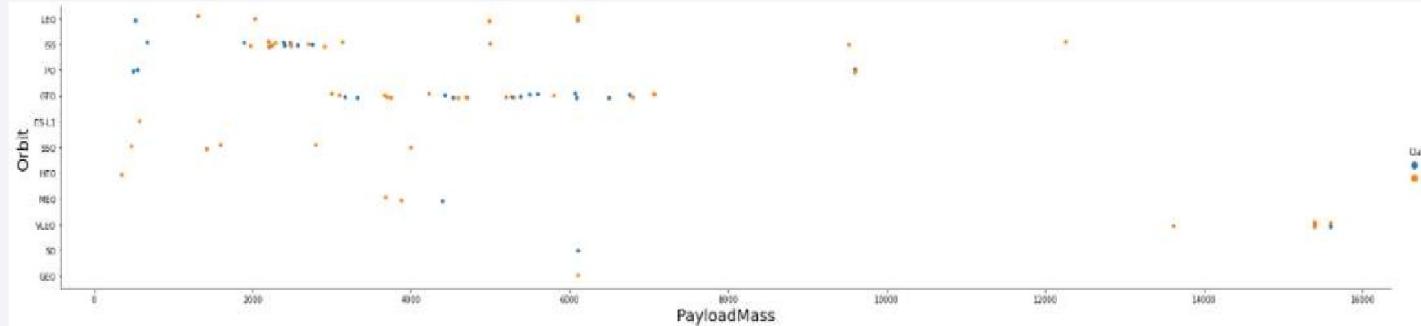
- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



# Flight Number vs. Orbit Type

---

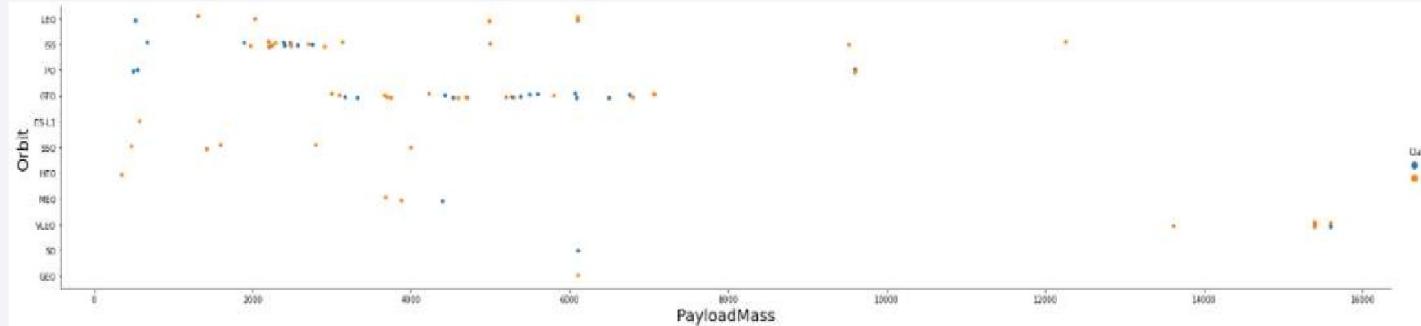
- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



# Payload vs. Orbit Type

---

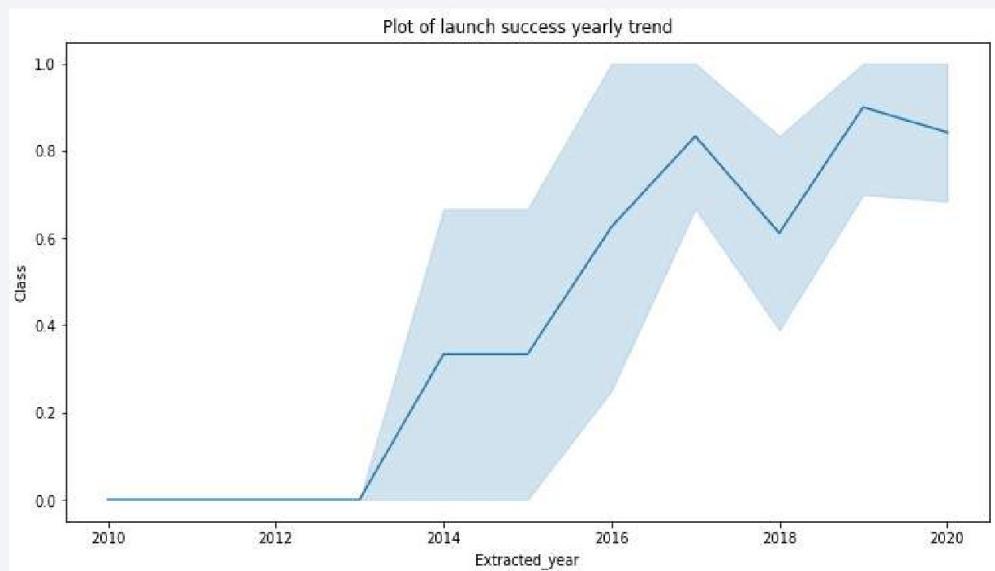
- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



# Launch Success Yearly Trend

---

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



# All Launch Site Names

---

- We used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

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

In [10]: task_1 = """
        SELECT DISTINCT LaunchSite
        FROM SpaceX
        """

create_pandas_df(task_1, database=conn)

Out[10]:   launchsite
0      KSC LC-39A
1    CCAFS LC-40
2    CCAFS SLC-40
3    VAFB SLC-4E
```

# Launch Site Names Begin with 'CCA'

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

```
In [11]: task_2 = """
    SELECT *
    FROM SpaceX
    WHERE LaunchSite LIKE 'CCA%'
    LIMIT 5
"""
create_pandas_df(task_2, database=conn)
```

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- We used the query above to display 5 records where launch sites begin with `CCA`

# Total Payload Mass

---

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

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

In [12]: task_3 = """
            SELECT SUM(PayloadMassKG) AS Total_PayloadMass
            FROM SpaceX
            WHERE Customer LIKE 'NASA (CRS)'
            """
            create_pandas_df(task_3, database=conn)

Out[12]: total_payloadmass
          0      45596
```

# Average Payload Mass by F9 v1.1

---

- Calculate the average payload mass carried by booster version F9 v1.1
  - Present your query result with a short explanation here
- 
- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

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

```
In [13]: task_4 = """
    SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
    FROM SpaceX
    WHERE BoosterVersion = 'F9 v1.1'
"""

create_pandas_df(task_4, database=conn)
```

Out[13]: avg\_payloadmass

0	2928.4
---	--------

# First Successful Ground Landing Date

---

- We observed that the dates of the first successful landing outcome on ground pad was 22<sup>nd</sup> December 2015

```
In [14]: task_5 = """
    SELECT MIN(Date) AS FirstSuccessfull_landing_date
    FROM SpaceX
    WHERE LandingOutcome LIKE 'Success (ground pad)'
"""

create_pandas_df(task_5, database=conn)
```

```
Out[14]: firstsuccessfull_landing_date
0           2015-12-22
```

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

```
In [15]: task_6 = """
    SELECT BoosterVersion
    FROM SpaceX
    WHERE LandingOutcome = 'Success (drone ship)'
        AND PayloadMassKG > 4000
        AND PayloadMassKG < 6000
    """
create_pandas_df(task_6, database=conn)
```

```
Out[15]: boosterversion
0 F9 FT B1022
1 F9 FT B1026
2 F9 FT B1021.2
3 F9 FT B1031.2
```

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

# Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
In [16]: task_7a = """
    SELECT COUNT(MissionOutcome) AS SuccessOutcome
    FROM SpaceX
    WHERE MissionOutcome LIKE 'Success%'
    """

task_7b = """
    SELECT COUNT(MissionOutcome) AS FailureOutcome
    FROM SpaceX
    WHERE MissionOutcome LIKE 'Failure%'
    """

print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

The total number of successful mission outcome is:

successoutcome
0 100

The total number of failed mission outcome is:

```
Out[16]: failureoutcome
0 1
```

- We used wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

# Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

```
List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
In [17]: task_8 = """
    SELECT BoosterVersion, PayloadMassKG
    FROM SpaceX
    WHERE PayloadMassKG = (
        SELECT MAX(PayloadMassKG)
        FROM SpaceX
    )
    ORDER BY BoosterVersion
"""
create_pandas_df(task_8, database=conn)

Out[17]: boosterversion payloadmasskg
0 F9 B5 B1048.4 15600
1 F9 B5 B1048.5 15600
2 F9 B5 B1049.4 15600
3 F9 B5 B1049.5 15600
4 F9 B5 B1049.7 15600
5 F9 B5 B1051.3 15600
6 F9 B5 B1051.4 15600
7 F9 B5 B1051.6 15600
8 F9 B5 B1056.4 15600
9 F9 B5 B1058.3 15600
10 F9 B5 B1060.2 15600
11 F9 B5 B1060.3 15600
```

# 2015 Launch Records

---

- We used combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

```
List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

In [18]: task_9 = """
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        """
create_pandas_df(task_9, database=conn)

Out[18]:   boosterversion  launchsite  landingoutcome
0      F9 v1.1 B1012  CCAFS LC-40  Failure (drone ship)
1      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))

```
In [19]: task_10 = ...  
        SELECT LandingOutcome, COUNT(LandingOutcome)  
        FROM SpaceX  
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'  
        GROUP BY LandingOutcome  
        ORDER BY COUNT(LandingOutcome) DESC  
        ...  
  
create_pandas_df(task_10, database=conn)
```

```
Out[19]:   landingoutcome  count  
0          No attempt     10  
1    Success (drone ship)     6  
2    Failure (drone ship)     5  
3    Success (ground pad)     5  
4    Controlled (ocean)      3  
5    Uncontrolled (ocean)      2  
6  Precluded (drone ship)      1  
7    Failure (parachute)      1
```

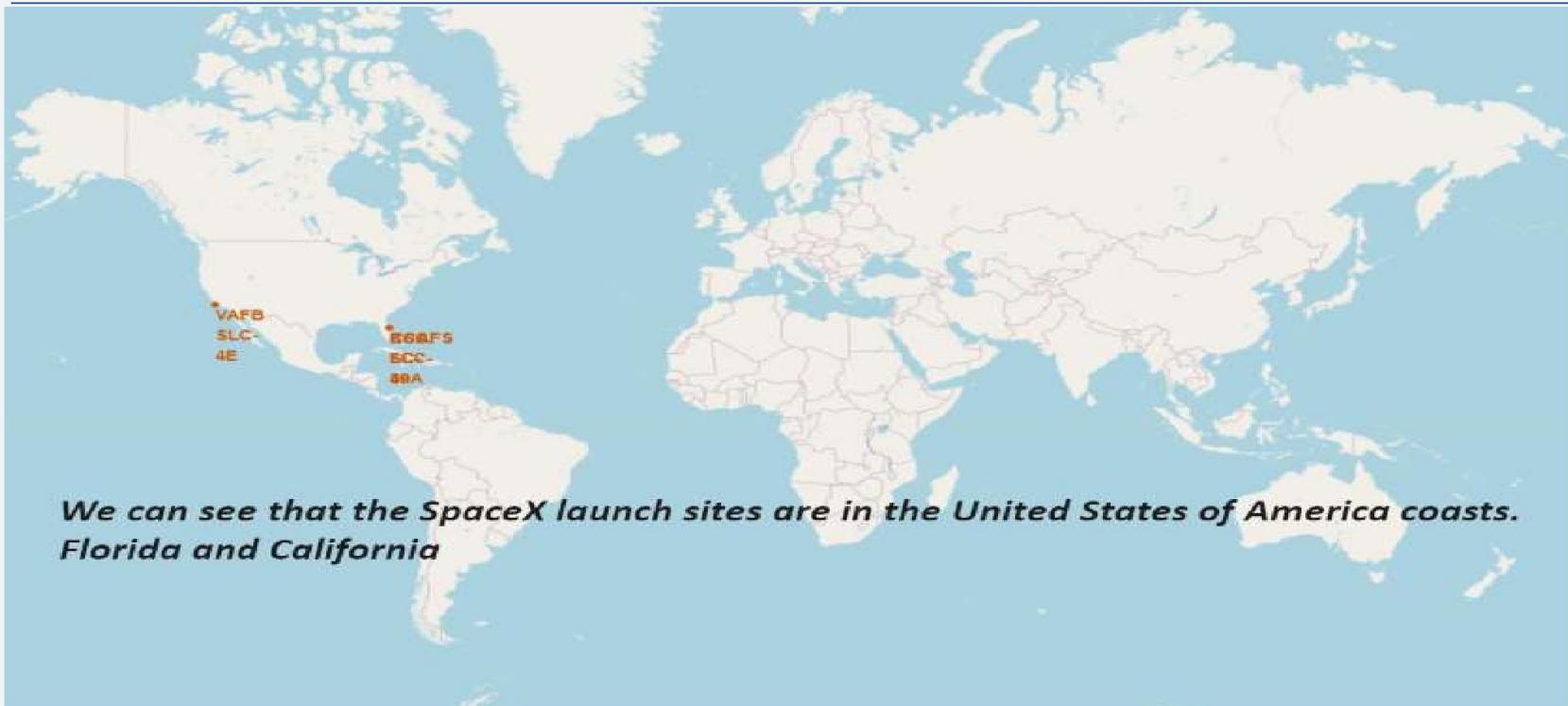
- We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.
- We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

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 half of the image. In the upper right quadrant, there is a bright green glow, likely representing the Aurora Borealis or a similar atmospheric phenomenon.

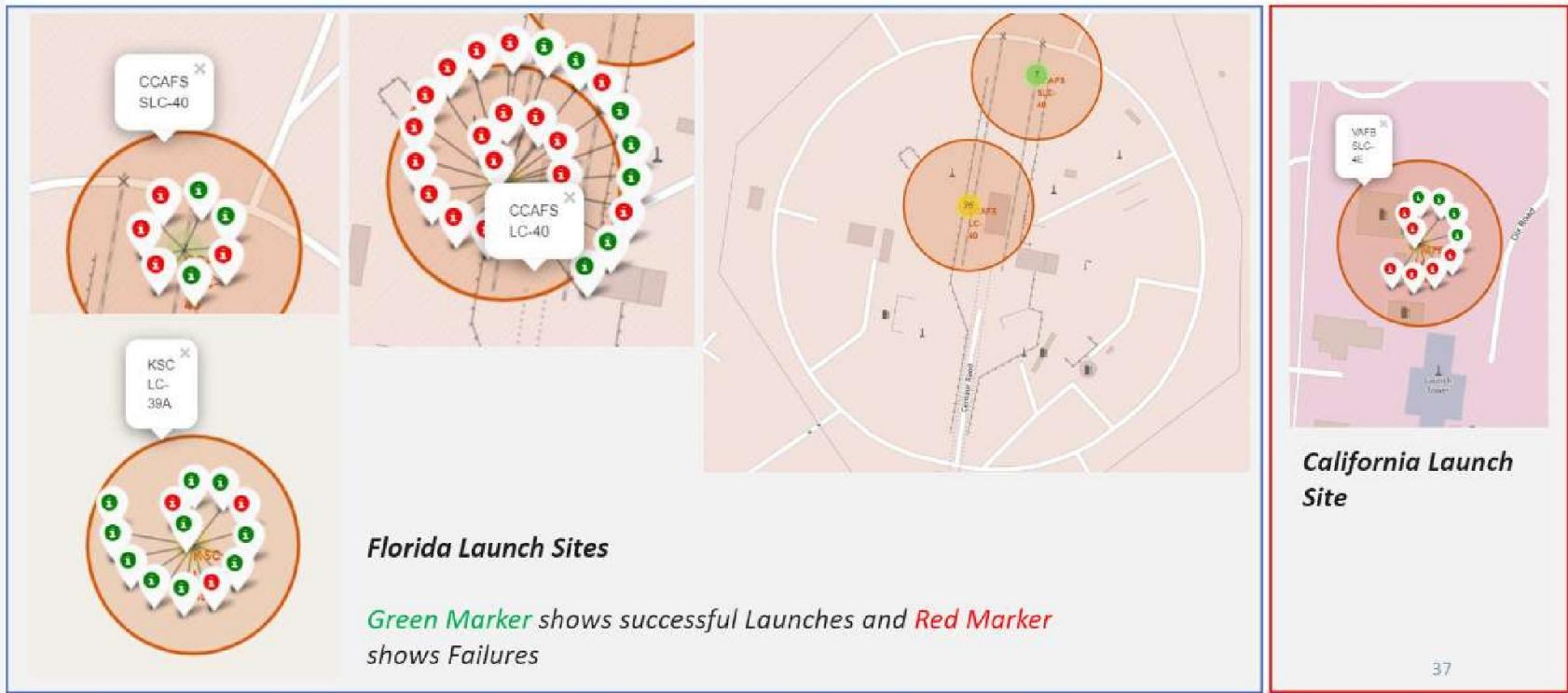
Section 3

# Launch Sites Proximities Analysis

## All launch sites global map markers



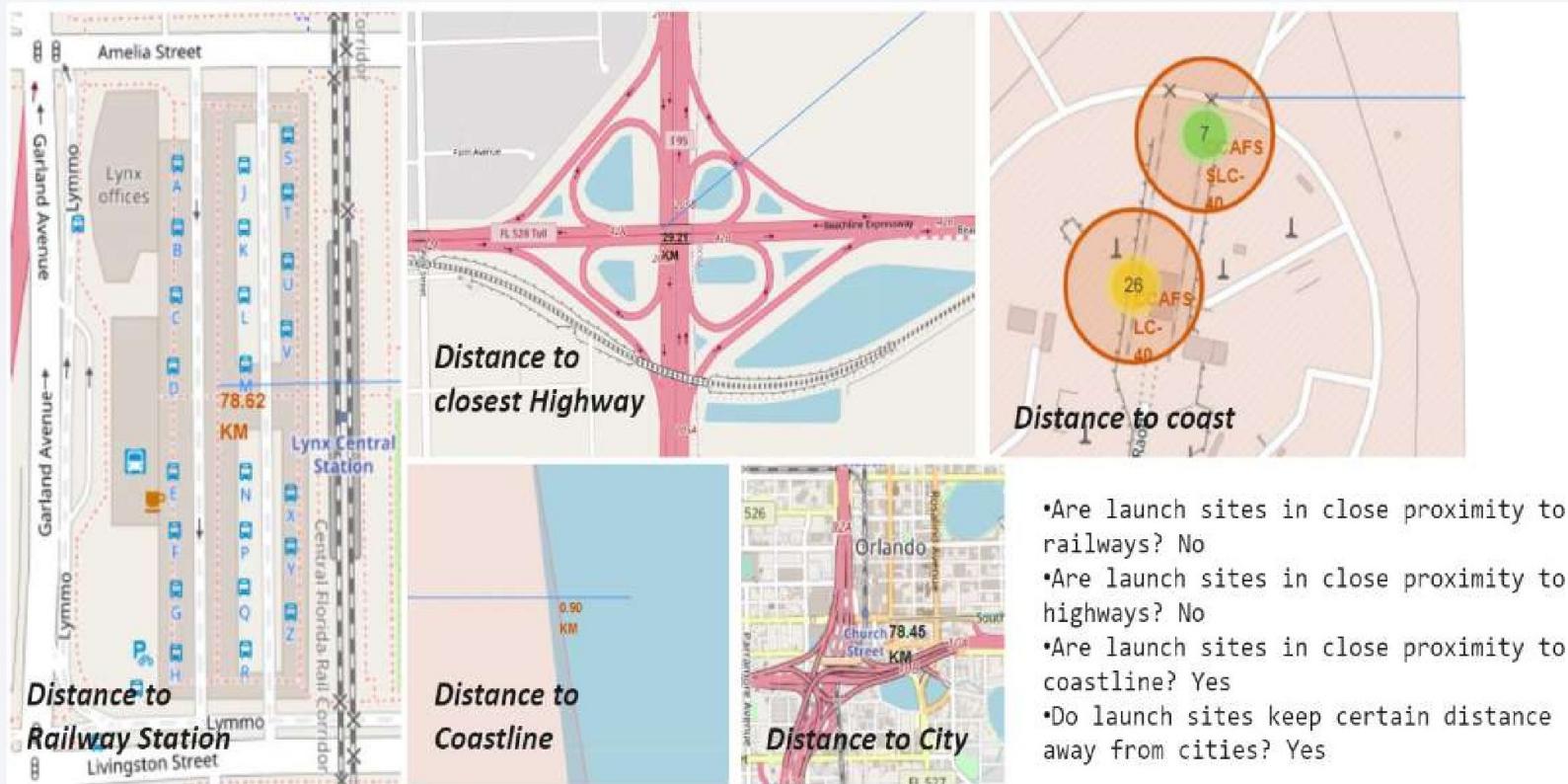
# Markers showing launch sites with color labels



37

36

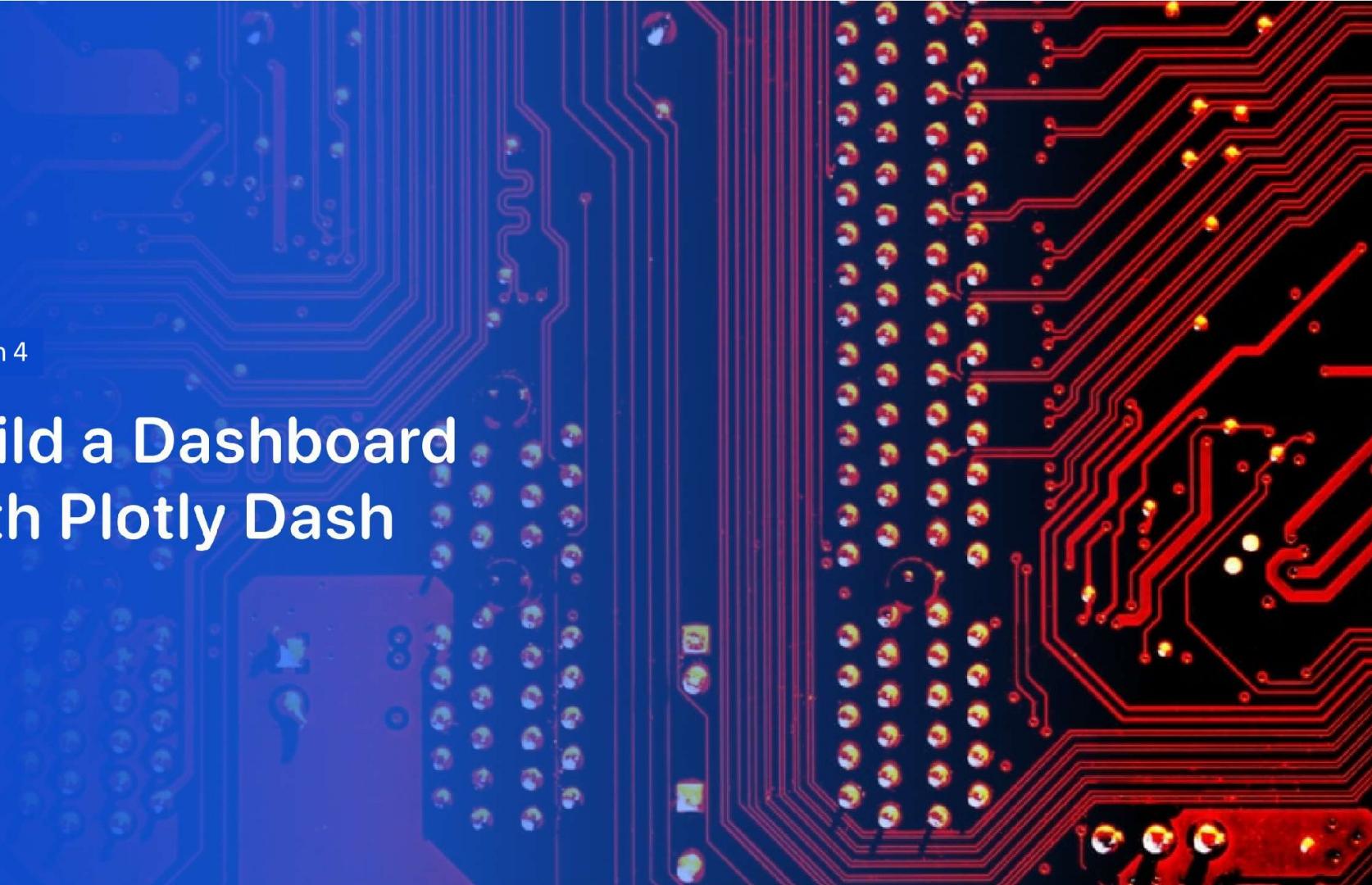
# Launch Site distance to landmarks



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

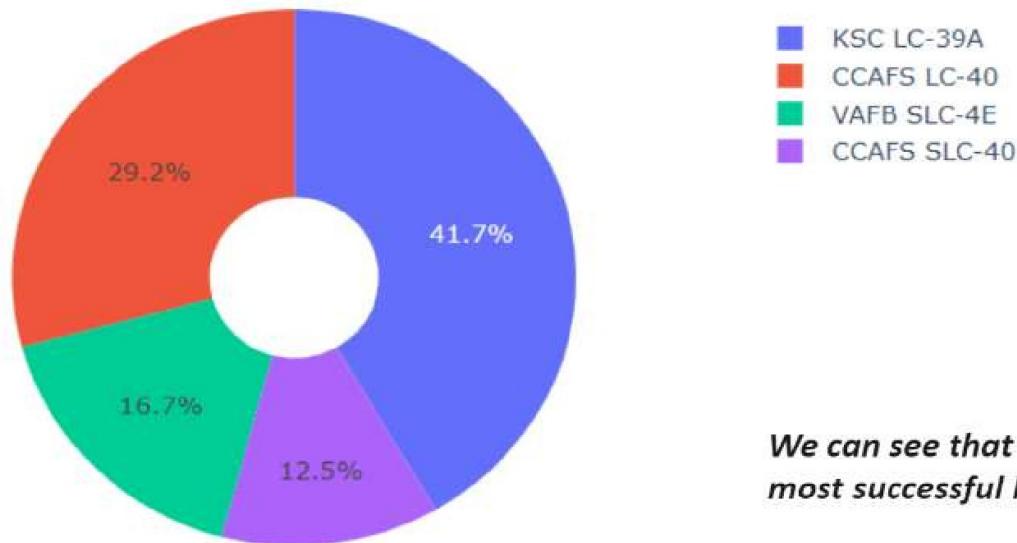
Section 4

# Build a Dashboard with Plotly Dash



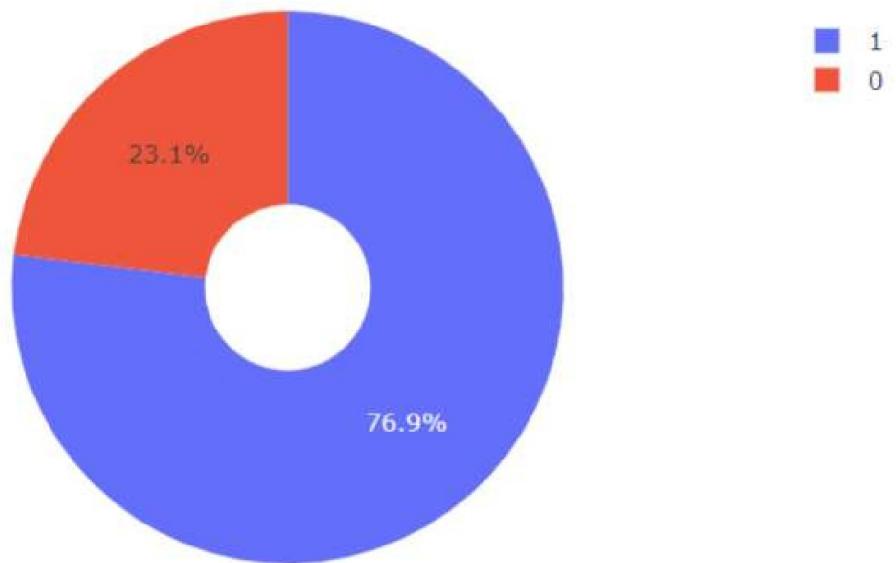
## Pie chart showing the success percentage achieved by each launch site

Total Success Launches By all sites



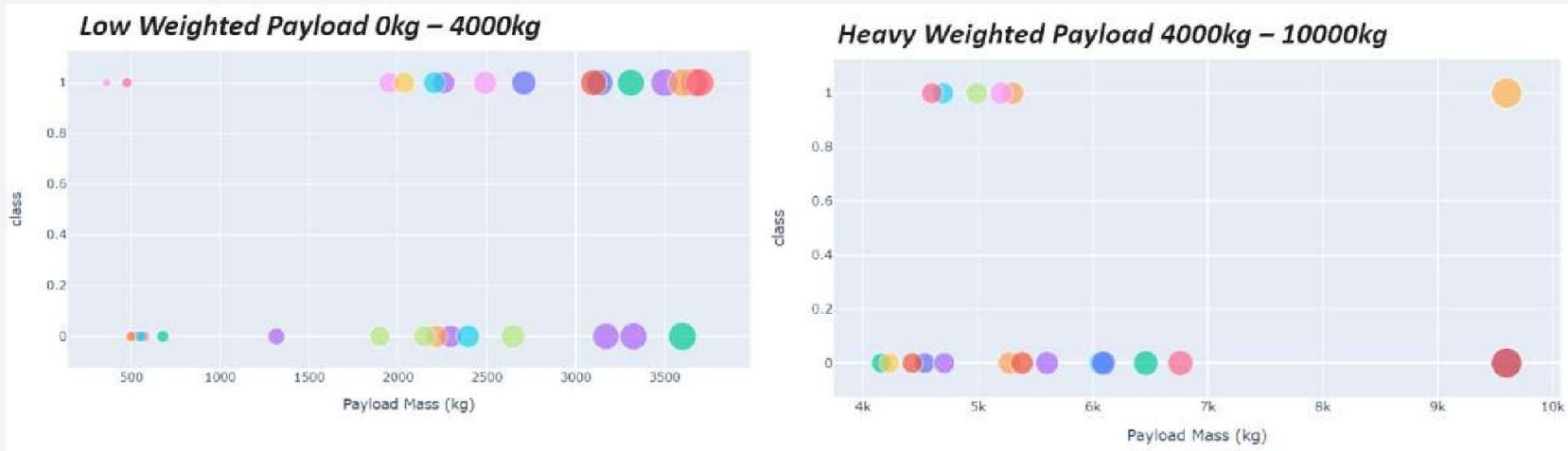
*We can see that KSC LC-39A had the most successful launches from all the sites*

## Pie chart showing the Launch site with the highest launch success ratio



*KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate*

## Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

The background of the slide features a blurred photograph of a tunnel. The image is dominated by shades of blue and white, with streaks of light creating a sense of speed and motion along the curved walls of the tunnel.

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

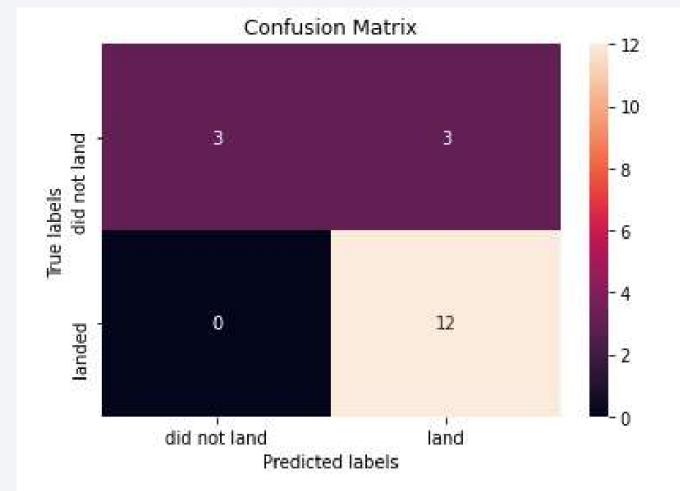
- The decision tree classifier is the model with the highest classification accuracy

```
models = {'KNeighbors':knn_cv.best_score_,  
          'DecisionTree':tree_cv.best_score_,  
          'LogisticRegression':logreg_cv.best_score_,  
          'SupportVector': svm_cv.best_score_}  
  
bestalgorithm = max(models, key=models.get)  
print('Best model is', bestalgorithm, 'with a score of', models[bestalgorithm])  
if bestalgorithm == 'DecisionTree':  
    print('Best params is :', tree_cv.best_params_)  
if bestalgorithm == 'KNeighbors':  
    print('Best params is :', knn_cv.best_params_)  
if bestalgorithm == 'LogisticRegression':  
    print('Best params is :', logreg_cv.best_params_)  
if bestalgorithm == 'SupportVector':  
    print('Best params is :', svm_cv.best_params_)  
  
Best model is DecisionTree with a score of 0.8732142857142856  
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

# Confusion Matrix

---

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



# Conclusions

---

We can conclude that:

- 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!

