

# Winning Space Race with Data Science

<Name>  
<Date>



# Outline

---

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

# Executive Summary

---

- Summary of methodologies
  - Data Collection
  - Web Scraping
  - Data Wrangling
  - SQL EDA
  - Data Visualization EDA
  - Folium – Interactive Geospatial Visualizations
  - Dashboard Application built with Plotly Dash
  - Predictive Analysis
- Summary of all results
  - EDA results
  - Interactive Analytics – Dashboard
  - Predictive Analysis – Machine Learning Models

# Introduction

---

- Background:
  - Companies are attempting to make space travel affordable for everyone. SpaceX can provide similar offerings at a much lower cost than competitors. Where other companies provide offerings at around \$165 million, SpaceX advertises a cost of \$65 million for the Falcon 9 rocket. Mainly due to the Falcon 9 rocket having a reusable 1<sup>st</sup> stage.
- Problems you want to find answers
  - For the cost merit to become reality, the launch must land successfully. The analysis is focused on analyzing history of successful launches and predicting the success rates of future launches using Machine Learning models.

Section 1

# Methodology

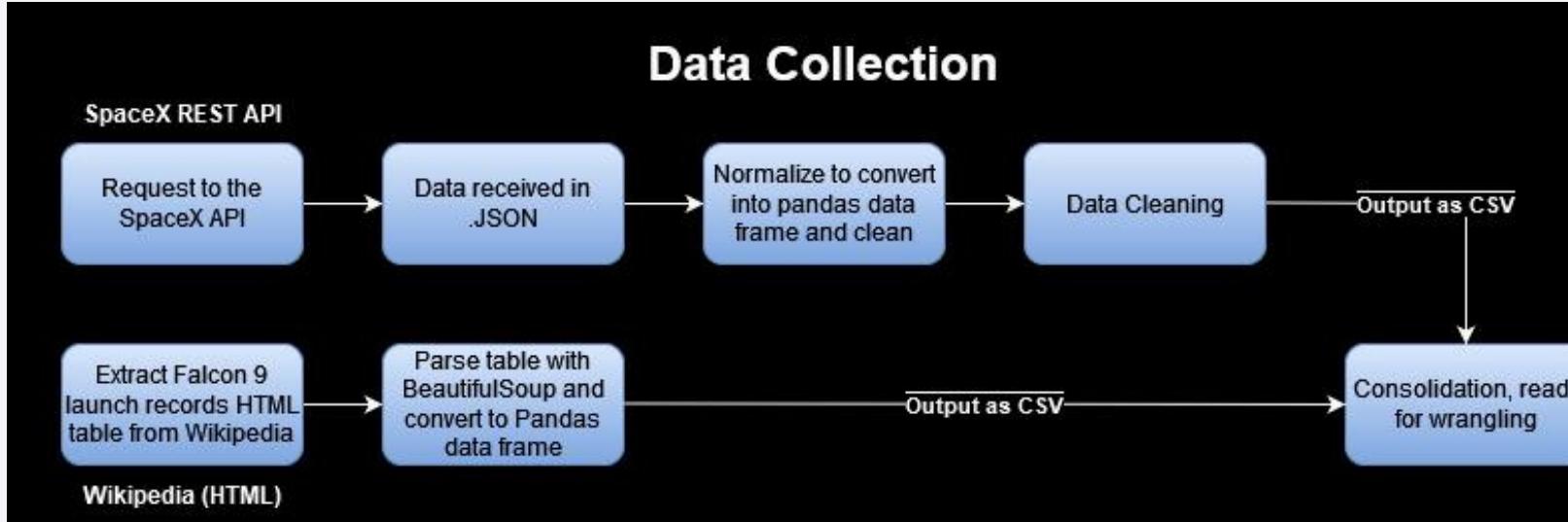
# Methodology

---

## Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - Wikipedia Web-Scraping
- Perform data wrangling
  - One hot encoding to prepare for Machine Learning Methods
  - Data was “cleaned” by removing null values and removing unnecessary fields
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Models Built and evaluated:
    - Logistic Regression
    - K-Nearest Neighbor
    - Support Vector Classification
    - Decision Tree Classifier

# Data Collection



- Collected from 2 sources:
  - SpaceX REST API
  - Wikipedia web scraping
- Made compatible, and then consolidated

# Data Collection – SpaceX API

1. Calling the SpaceX REST API
2. Retrieve JSON and convert to pd df
3. Extract relevant info
4. Use custom function to convert to usable format
5. Populate dictionary from JSON and create df
6. Replace nan with mean of PayloadMass column

[https://github.com/MLiebhart/IBM\\_Python\\_for\\_Data\\_Science/blob/1e32bfa8537bc7e5b2dfd5d5c6fc95129ff0e50d/Part1\\_jupyter-labs-spacex-data-collection-api.ipynb](https://github.com/MLiebhart/IBM_Python_for_Data_Science/blob/1e32bfa8537bc7e5b2dfd5d5c6fc95129ff0e50d/Part1_jupyter-labs-spacex-data-collection-api.ipynb)

```
1. Calling the SpaceX REST API
spacex_url='https://api.spacexdata.com/v4/launches/past'
✓ 0s

response = requests.get(spacex_url)
✓ 0.4s

Check the content of the response
print(response.content)
✓ 0.0s

2. Using static source, retrieve JSON and convert to pandas data frame
static_json_url= https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json
✓ 0s

response.status_code
✓ 0.0s
200

# Use json_normalize method to convert the json result into a dataframe
data = response.json()
data = pd.json_normalize(data)
data
✓ 0.0s

3. Extract relevant information
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]
# We will remove rows with multiple cores because those are Falcon rockets with 1 extra rocket boosters and rows that have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
✓ 0.0s

4. Use custom function to convert to usable format
# Call getBoosterVersion
getBoosterVersion(data)
✓ 43.4s

5. Fill the data frame using a created dictionary
launch_dict = {'FlightNumber': list(data['flight_number']),
               'Date': list(data['date']),
               'BoosterVersion': list(data['cores']),
               'Payloads': list(data['payloads']),
               'Orbit': list(data['orbit']),
               'LaunchSite': list(data['launchpad']),
               'Outcome': list(data['date']),
               'Flights': list(data['flights']),
               'GridFins': list(data['gridfins']),
               'Reused': list(data['reused']),
               'Legs': list(data['legs']),
               'LandingPad': list(data['landing_pads']),
               'Block': list(data['block']),
               'NumberOfReusedCount': list(data['number_of_reused']),
               'Serial': list(data['serial']),
               'Longitude': list(data['longitude']),
               'Latitude': list(data['latitude'])}
✓ 0.0s

6. Replace nan values with the mean of PayloadMass
# Calculate the mean value of PayloadMass column
avg_payload = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].replace(np.nan, avg_payload)
data_falcon9.isnull().sum()
✓ 0.0s

# Create a data frame launch_dict
data2 = pd.DataFrame.from_dict(launch_dict)
data2
✓ 0.0s
```

# Data Collection – Webscraping

1. Request launch Wiki and create BeautifulSoup object
2. Locate target table on page
3. Extract column names
4. From column names, parse data into launch\_dict
5. Construct df from launch\_dict

The diagram illustrates the five steps of webscraping using a Jupyter notebook interface. Each step is associated with a specific code snippet and a numbered arrow indicating its flow.

- 1. Request SpaceX launch Wiki and create BeautifulSoup Object**  
Code: 

```
# use requests.get() method with the provided static_url  
# assign the response to a object  
response = requests.get(static_url)
```

 (Time: 7.7s)  

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

 (Time: 1.1s)
- 2. Locate and verify target table**  
Code: 

```
# Let's print the third table and check its content  
first_launch_table = html_tables[2]  
print(first_launch_table)
```

 (Time: 0.0s)
- 3. Extract Column Names**  
Code: 

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

 (Time: 0.0s)
- 4. Use column names and parse data into dictionary**  
Code: 

```
extracted_row = 0  
#Extract each table  
for table_number,table in enumerate(soup.find_all('table','wikitable plainrowheaders collapsible')):  
    # get table row  
    for rows in table.find_all("tr"):  
        #check to see if first table heading is as number corresponding to launch a number  
        if rows.th:  
            if rows.th.string:  
                flight_number=rows.th.string.strip()  
                flag=flight_number.isdigit()  
            else:  
                flag=False  
            get table element  
            rows.find_all('td')  
            #if it is number save cells in a dictionary  
            if flag:  
                extracted_row += 1  
                # flight Number value  
                # TODO: Append the flight_number into launch_dict with key 'Flight No.'  
                launch_dict['Flight No.'].append(flight_number)  
                print(flight_number)  
                datetimeList=datetime.strptime(rows[0])
```
- 5. Construct df from launch\_dict**  
Code: 

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })  
df.head()
```

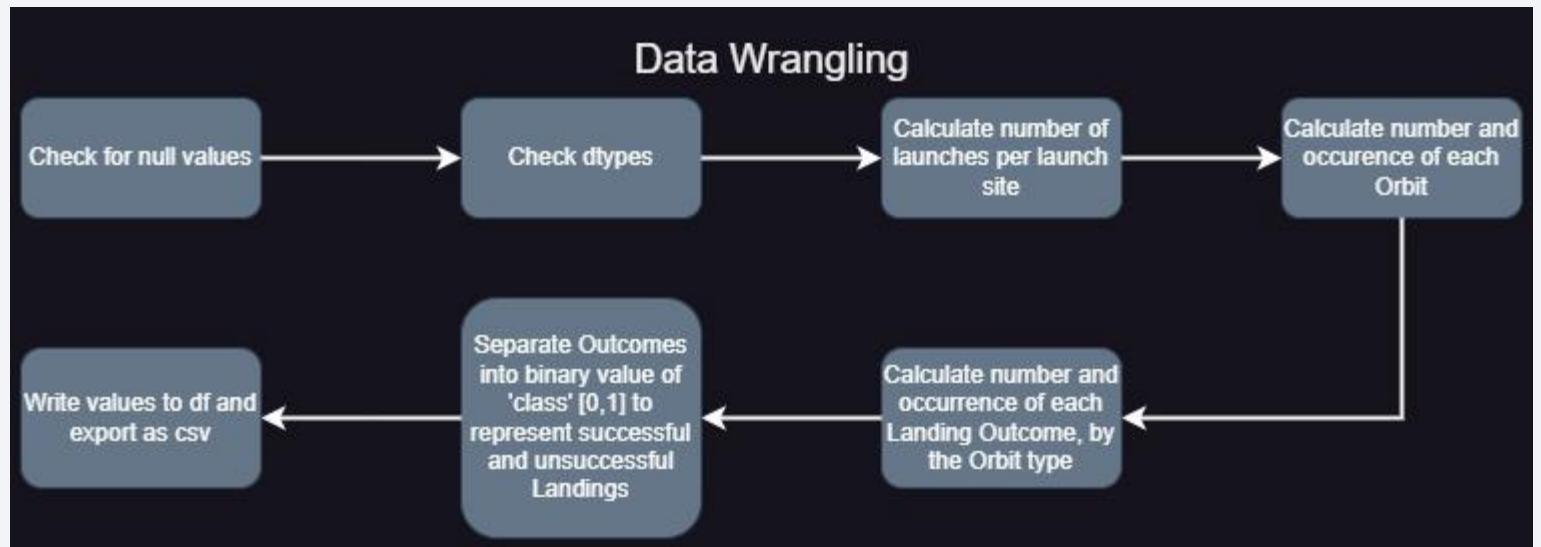
 (Time: 0.0s)

[https://github.com/MLiebhart/IBM\\_Python\\_for\\_Data\\_Science/blob/cb432e8e208b14bd797541d0f278b6703ac6286/Part2\\_jupyter-labs-webscraping\\_SpaceX.ipynb](https://github.com/MLiebhart/IBM_Python_for_Data_Science/blob/cb432e8e208b14bd797541d0f278b6703ac6286/Part2_jupyter-labs-webscraping_SpaceX.ipynb)

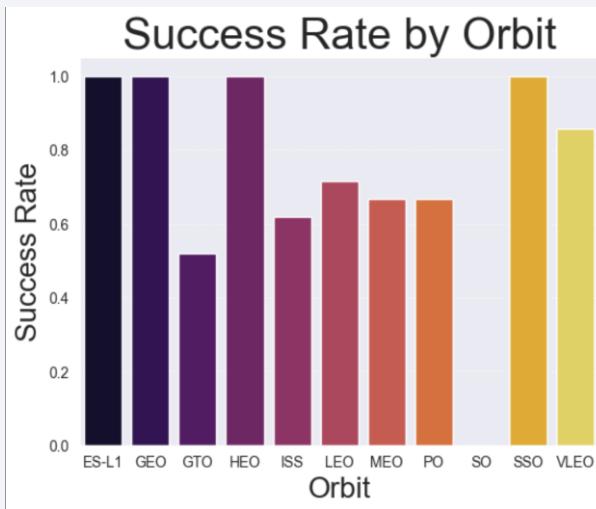
# Data Wrangling

---

1. Check for null values
2. Check dtypes
3. Calculate number of launches per site
4. Calculate Orbit
5. Calculate Outcomes
6. Classify and separate outcomes into binary
7. Write values to df and export as csv

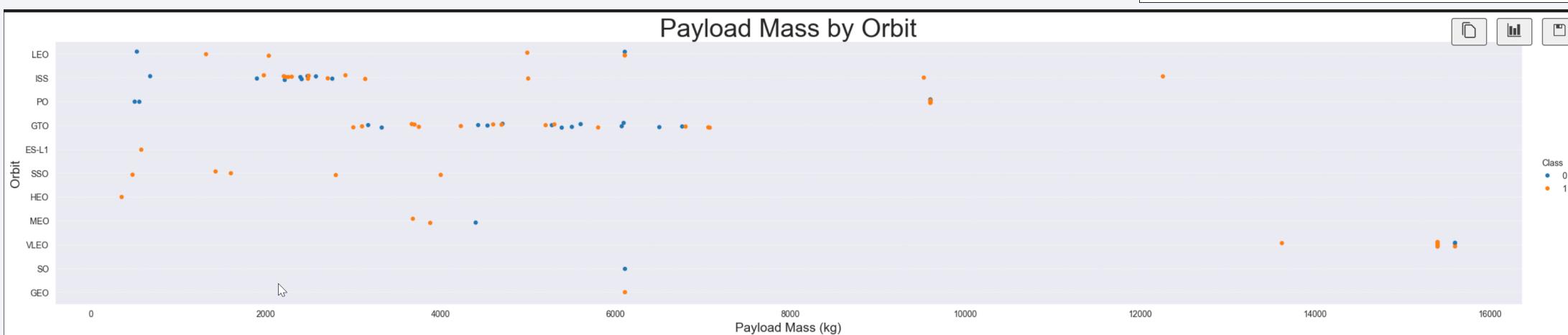
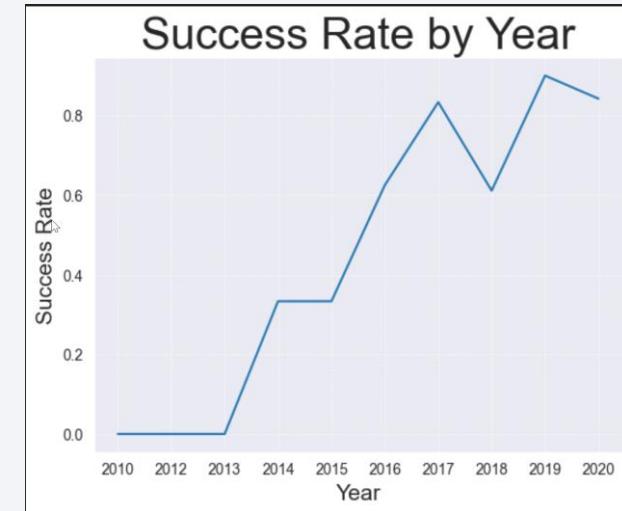


# EDA with Data Visualization



These are a few of the graphs made for EDA.

Please visit the link at the bottom for the full Jupyter Notebook



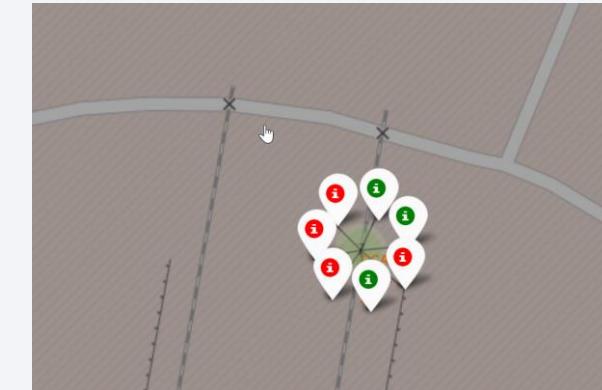
# EDA with SQL

---

- SQL Query using SQLite, a light-weight Relational database management system (RDBMS) in Python.
  - Create table where date value is not null for analysis
  - Display unique values for column (launch site)
  - Display records with specific launch site (CCA)
  - Display total payload mass launched by specific customer (NASA)
  - Display average payload mass carried by specific booster version (F9 v1.1)
  - Display the date of the first successful launch
  - Display distinct combinations of 2 categorical fields (landing outcome, mission outcome)
  - Listing booster version names with success on drone ship and payload mass between 4000 and 6000 kg
  - Total counts of outcomes by type
  - Display names of booster versions that have carried the maximum payload using a subquery
  - List records which display the month, failure landing outcomes in “drone ship”, booster versions, and launch site by month for the year 2015
  - Rank counts of landing outcomes for date range (2010-06-04 – 2017-03-20)

# Build an Interactive Map with Folium

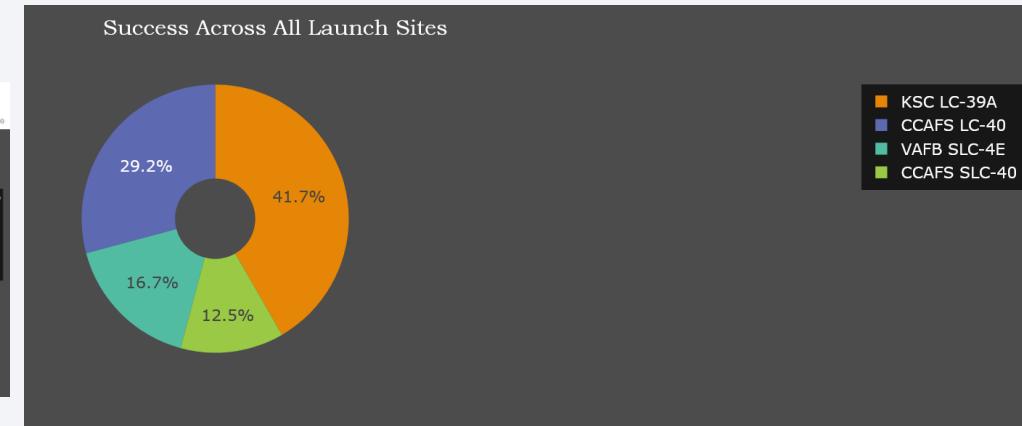
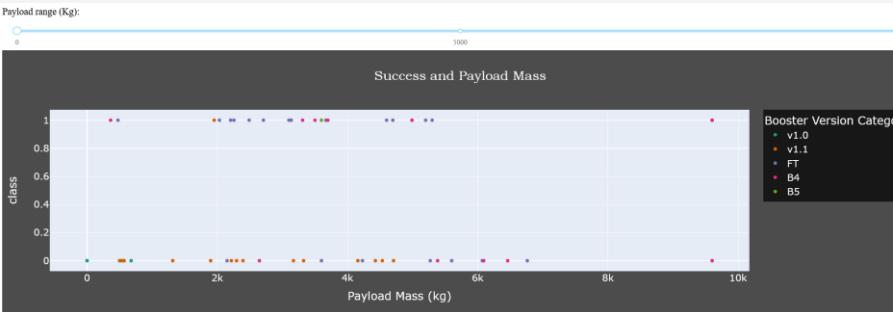
- Map markers added at launch sites
- Features added for interactive viewing:
  - Marker clusters
  - Circle markers
  - Icons
  - Identified and calculated distances to objects of interest:
    - Nearest City, Railroad track, Highway, and Disney World



13

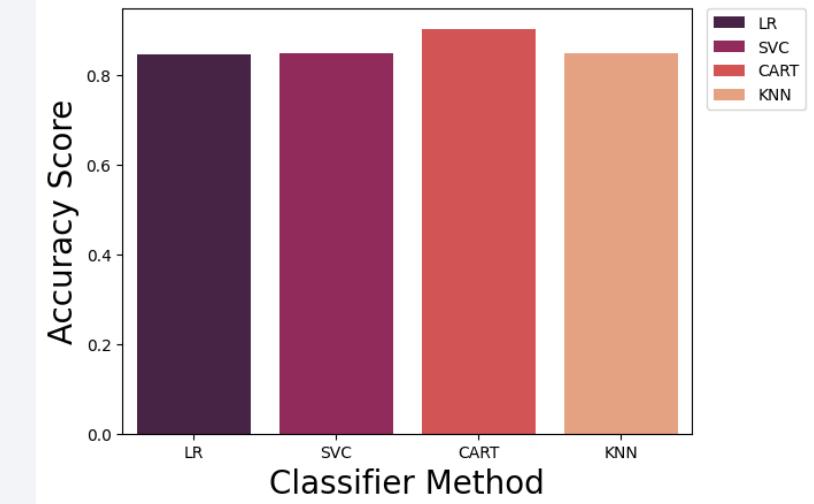
# Build a Dashboard with Plotly Dash

- Dash application was made with pie charts representing the success rates by launch site – which is filterable.
- Scatter plot representing success rates for payloads and a slider to filter out and select certain payload ranges.
- These plots were made to highlight and visualize factors that may contribute to success rate.



# Predictive Analysis (Classification)

- Used 4 common supervised machine learning models for classification:
  - Logistic Regression (LR)
  - Support Vector Machine (SVM)
  - K-Nearest Neighbor (KNN)
  - Decision Tree Classification (CART)
- Data was split into 80/20 (train/test) and the same training data was used to train each model.
- Using a grid search with cross-validation, each model selected the best predictors out of a dictionary of provided parameters.
- The model with the best accuracy on the training set was the Decision Tree Classifier (CART)



# Results

---

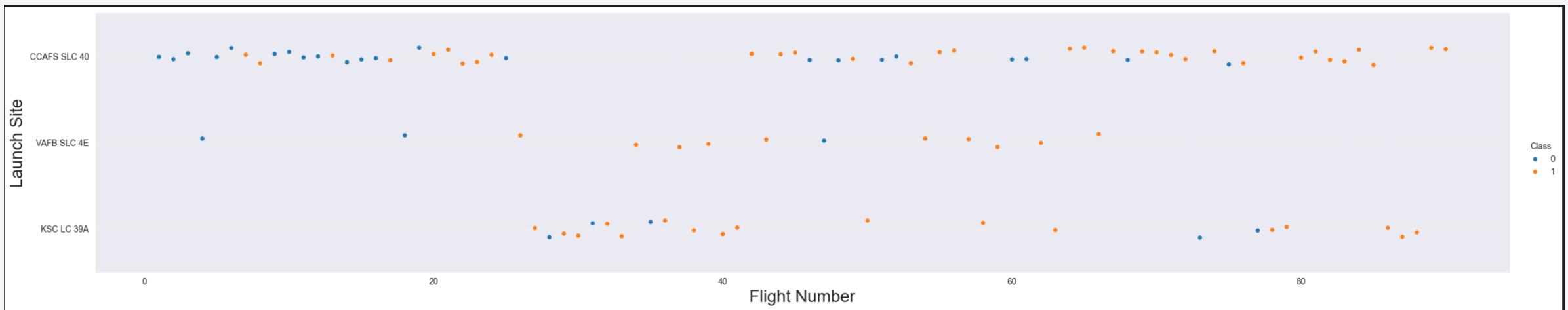
- Results of the following to be shared next in greater detail:
  - Exploratory data analysis results:
  - Interactive analytics demo in screenshots
  - Predictive analysis results

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

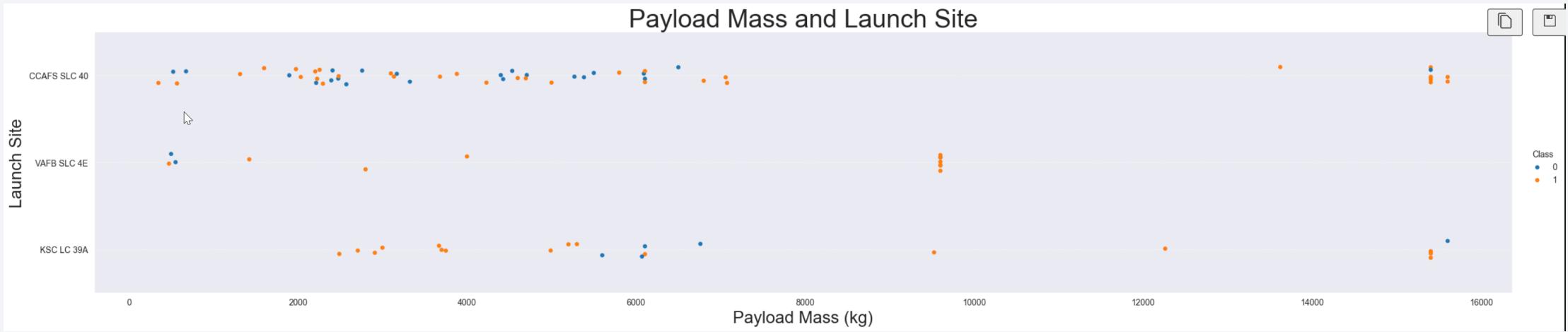
## Insights drawn from EDA

# Flight Number vs. Launch Site



- CCAFS is the most common launch site over time.
- Considering rates of success (0=Failure 1=Success), CCAFS may sit on the low end as the data also suggests probability of success may have increased over time
  - Likely due to “lessons learned” and process/technology improvements

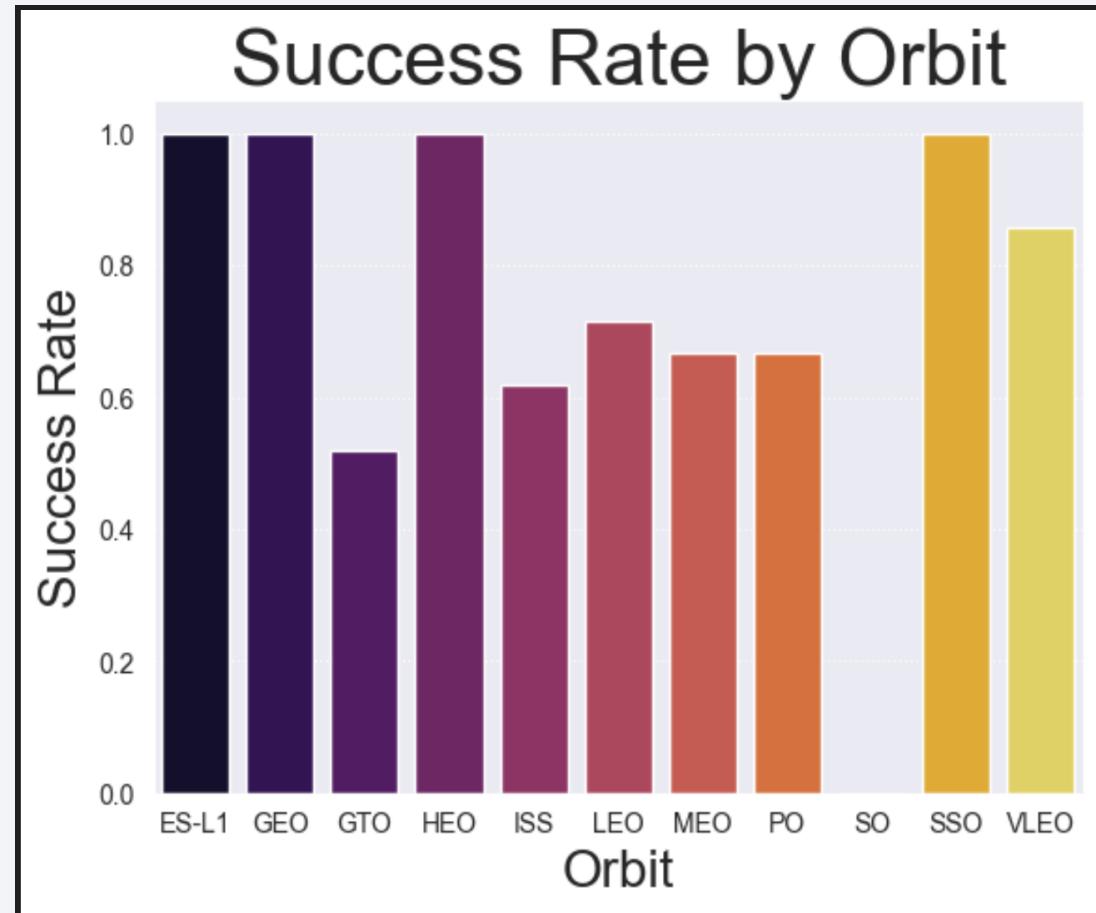
# Payload vs. Launch Site



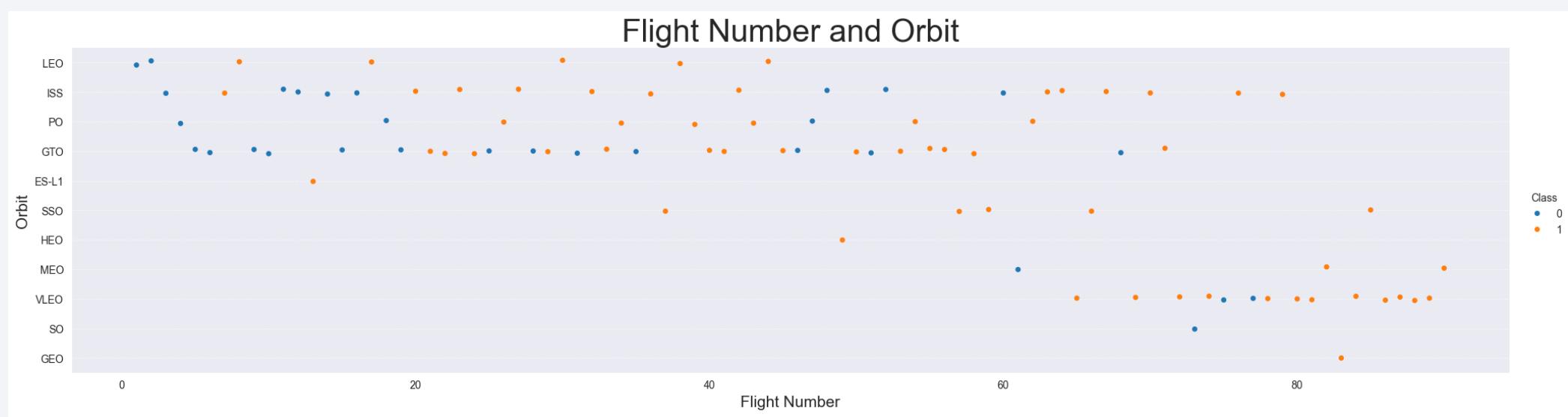
- CCAFS is the most common launch site over time.
- VAFB is favored for payloads between 8,000 to 12,000 kg. Which also have a 100% success rate in this range across all sites.

# Success Rate vs. Orbit Type

- Orbits ES-L1, GEO, HEO, and SSO are most successful.
- GTO is the least successful

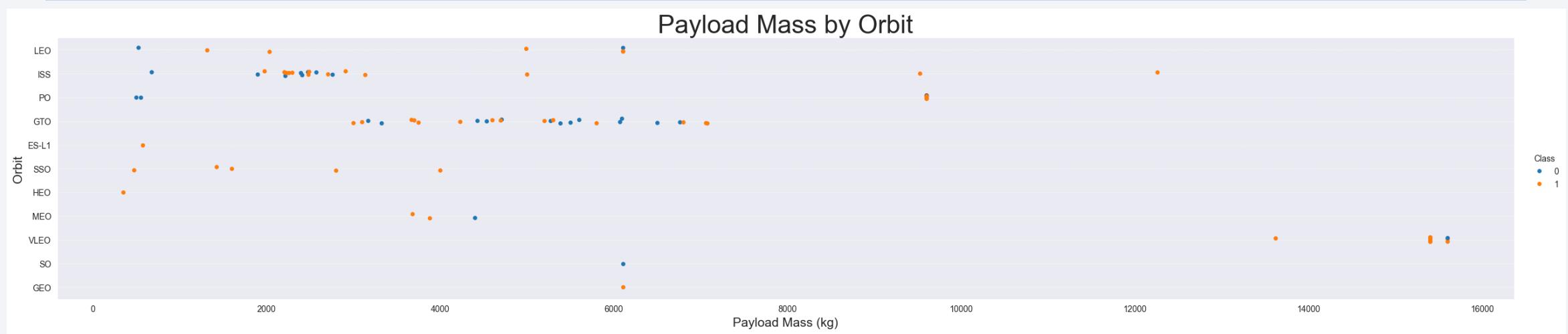


# Flight Number vs. Orbit Type



- Orbit types with highest success rates are underrepresented by quantity.
- Over time, SpaceX seems to have transitioned focus towards VLEO orbit.

# Payload vs. Orbit Type

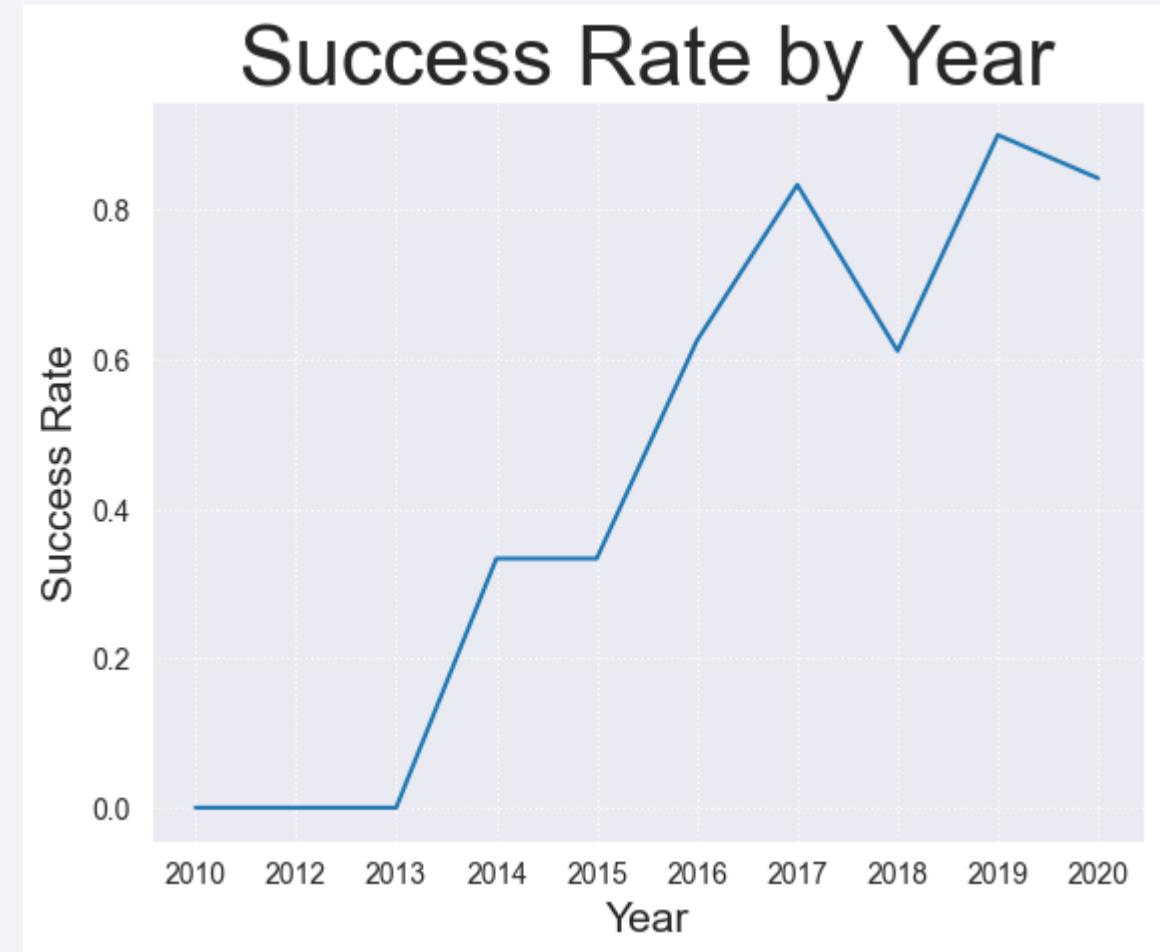


- There seems to be a relationship between payload mass and some Orbit levels.
  - VLEO favors heavier payloads.
  - ISS has tight grouping 2k-4k, and a few outliers.
  - GTO 3k-8k.

# Launch Success Yearly Trend

---

- Success rates have climbed significantly since 2013.
- In the year 2020, SpaceX achieved just over 80% success rate.



# All Launch Site Names

---

- Launch sites from data:
  - CCAFS LC-40
  - CCAFS SLC-40
  - VAFB SLC-4E
  - KSC LC-39A
- Query is run in SQL syntax through SQLite extension for Python.
  - The query returns unique values within the column launch\_site

```
%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 |

# Launch Site Names Begin with 'CCA'

| %sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5 |            |                 |             |   |                 |           |                 |                 |                     |  |
|--|------------|-----------------|-------------|---|-----------------|-----------|-----------------|-----------------|---------------------|--|
| * <a href="sqlite:///my_data1.db">sqlite:///my_data1.db</a>        |            |                 |             |   |                 |           |                 |                 |                     |  |
| 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          |  |

- SQL query displays (limit 5) from all records in data table where the value within Launch\_Site contains CCA as the first 3 characters

# Total Payload Mass

---

- This represents payloads carried by SpaceX where NASA (CRS) was the customer.
- This number represents the total payload mass delivered by SpaceX to the ISS as a part of NASA's Commercial Resupply Services (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

---

- Query displays average payload mass carried by booster version Falcon 9 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
```

# First Successful Ground Landing Date

---

- After reviewing all Landing Outcomes, I queried the table to show the date of first success.

```
# -- %sql select distinct Landing_Outcome from SPACEXTBL  
%sql select min(Date) from SPACEXTBL where Landing_Outcome like 'Success%'
```

```
* sqlite:///my\_data1.db
```

```
Done.
```

```
* sqlite:///my\_data1.db
```

```
Done.
```

**min(Date)**

2015-12-22

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

---

- Booster versions with successful drone ship landings with payload mass between 4000 and 6000 kg.

```
%sql select Booster_Version, Landing_Outcome, PAYLOAD_MASS__KG_ from SPACEXTBL where Landing_Outcome is 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 AND 6000
```

```
* sqlite:///my\_data1.db
```

```
Done.
```

| Booster_Version | Landing_Outcome      | PAYLOAD_MASS__KG_ |
|-----------------|----------------------|-------------------|
| F9 FT B1022     | Success (drone ship) | 4696              |
| F9 FT B1026     | Success (drone ship) | 4600              |
| F9 FT B1021.2   | Success (drone ship) | 5300              |
| F9 FT B1031.2   | Success (drone ship) | 5200              |

# Total Number of Successful and Failure Mission Outcomes

---

- SpaceX has had a total of 1 Failure and 101 Successful Mission Outcomes.

```
%sql select Mission_Outcome, count(*) as status from SPACEXTBL group by Mission_Outcome  
  
* sqlite:///my\_data1.db  
Done.  
  


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


```

# Boosters Carried Maximum Payload

- Falcon 9 B5 is the only Booster version to have carried the maximum Payload.
- It has done this 12 times!

```
%sql select substr(Booster_Version, 1, 5), count(*) as num_trips from SPACEXTBL where PAYLOAD_MASS_KG_ = (select max(PAYLOAD_MASS_KG_) from SPACEXTBL)
```

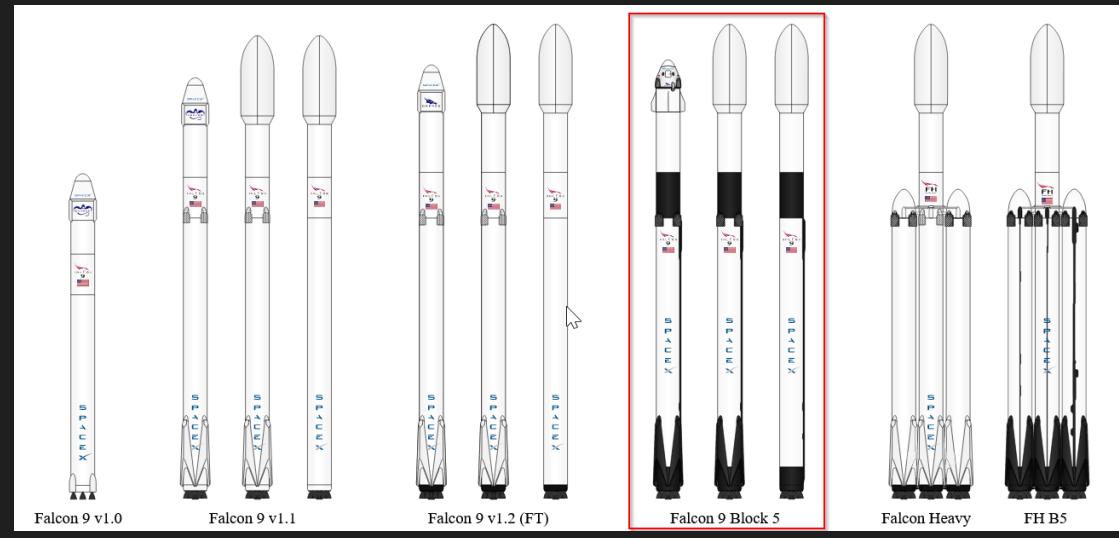
| substr(Booster_Version, 1, 5) | num_trips |
|-------------------------------|-----------|
| F9 B5                         | 12        |

```
%sql select distinct Booster_Version, PAYLOAD_MASS_KG_ from SPACEXTBL where PAYLOAD_MASS_KG_ = (select max(PAYLOAD_MASS_KG_) from SPACEXTBL) order by Booster_Version  
#%sql select distinct substr(Booster_Version, 1, 5) from SPACEXTBL where PAYLOAD_MASS_KG_ = (select max(PAYLOAD_MASS_KG_) from SPACEXTBL) order by Booster_Version
```

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

Done.

| Booster_Version | PAYLOAD_MASS_KG_ |
|-----------------|------------------|
| F9 B5 B1048.4   | 15600            |
| F9 B5 B1048.5   | 15600            |
| F9 B5 B1049.4   | 15600            |
| F9 B5 B1049.5   | 15600            |
| F9 B5 B1049.7   | 15600            |
| F9 B5 B1051.3   | 15600            |
| F9 B5 B1051.4   | 15600            |
| F9 B5 B1051.6   | 15600            |
| F9 B5 B1056.4   | 15600            |
| F9 B5 B1058.3   | 15600            |
| F9 B5 B1060.2   | 15600            |
| F9 B5 B1060.3   | 15600            |



# 2015 Launch Records

---

- 3 attempts were made for drone ship landings in the months of January, April, and June during 2015.
- Out of these 3 attempts there were no successful landings on drone ship.

```
%sql select substr(Date,6,2) as month, substr(Date,0,5) as year, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTBL where substr(Date,0,5) = '2015' and Landing_Outcome like '%drone ship%'
```

Python

```
* sqlite:///my\_data1.db
```

Done.

| month | year | Landing_Outcome        | Booster_Version | Launch_Site |
|-------|------|------------------------|-----------------|-------------|
| 01    | 2015 | Failure (drone ship)   | F9 v1.1 B1012   | CCAFS LC-40 |
| 04    | 2015 | Failure (drone ship)   | F9 v1.1 B1015   | CCAFS LC-40 |
| 06    | 2015 | Precluded (drone ship) | F9 v1.1 B1018   | CCAFS LC-40 |

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

---

- Between 2010-06-04 and 2017-03-20 Landing Outcomes ranked by count.
- There are most attempts on drone ship
- No outcomes other than success on ground pad for date range.

```
%sql select Landing_Outcome, count(Landing_Outcome) from SPACEXTBL where Date between '2010-06-04' and '2017-03-20' group by Landing_Outcome order by count(Landing_Outcome) desc
```

```
* 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                      |

The background of the slide is a photograph taken from space at night. It shows the curvature of the Earth's horizon against a dark blue sky. Numerous glowing yellow and white points represent city lights, concentrated in coastal and urban areas. In the upper right quadrant, there are bright green and yellow bands of light, likely the Aurora Borealis or Australis. The overall atmosphere is dark and mysterious.

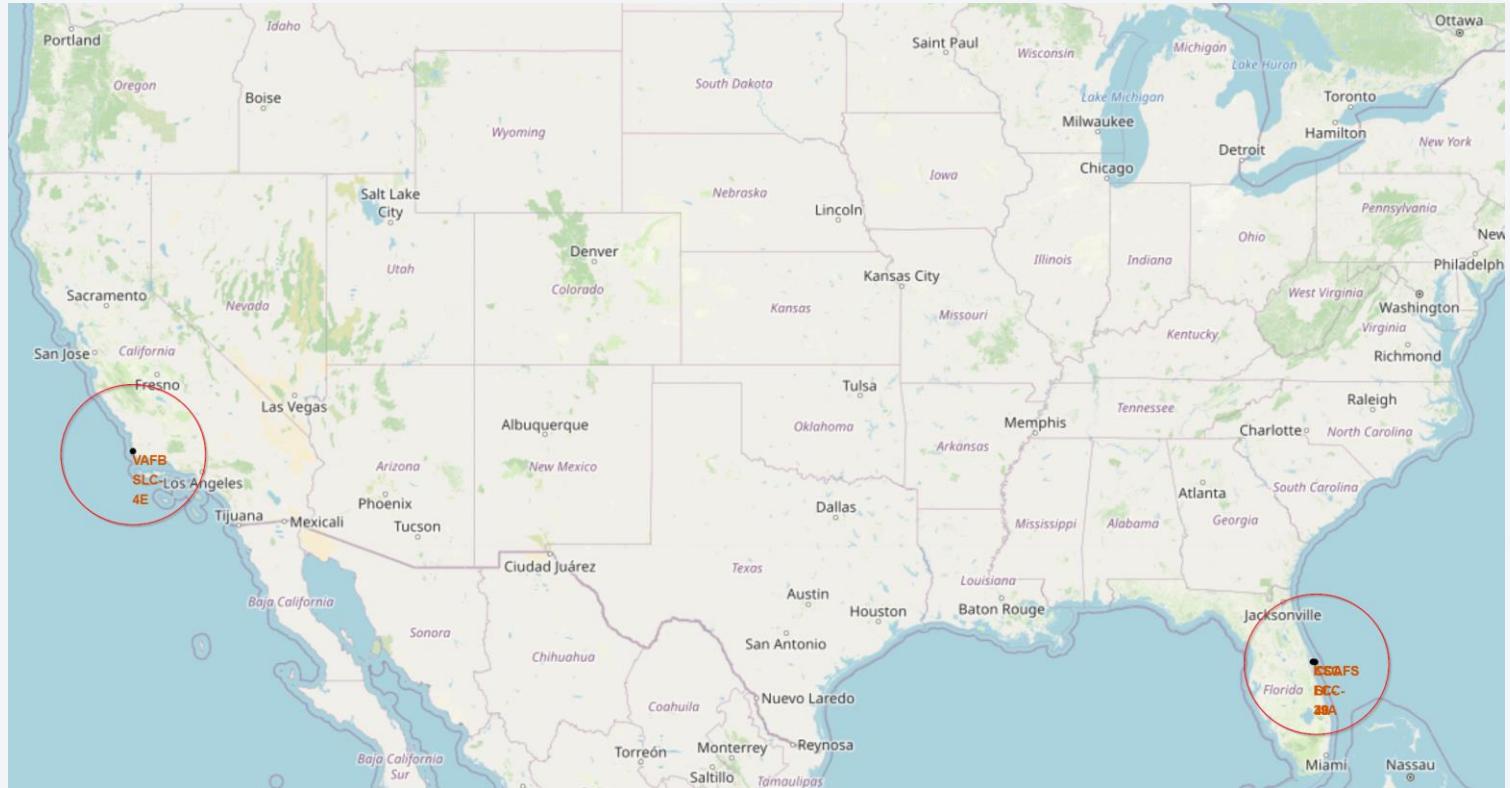
Section 3

# Launch Sites Proximities Analysis

# Launch Sites

---

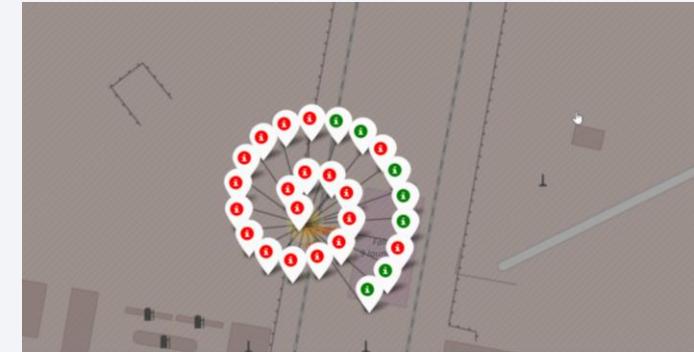
- Launch Locations are at both the East and West coast in very close proximity to the ocean.



# Folium Map Features (Circle markers, clusters, etc.)

---

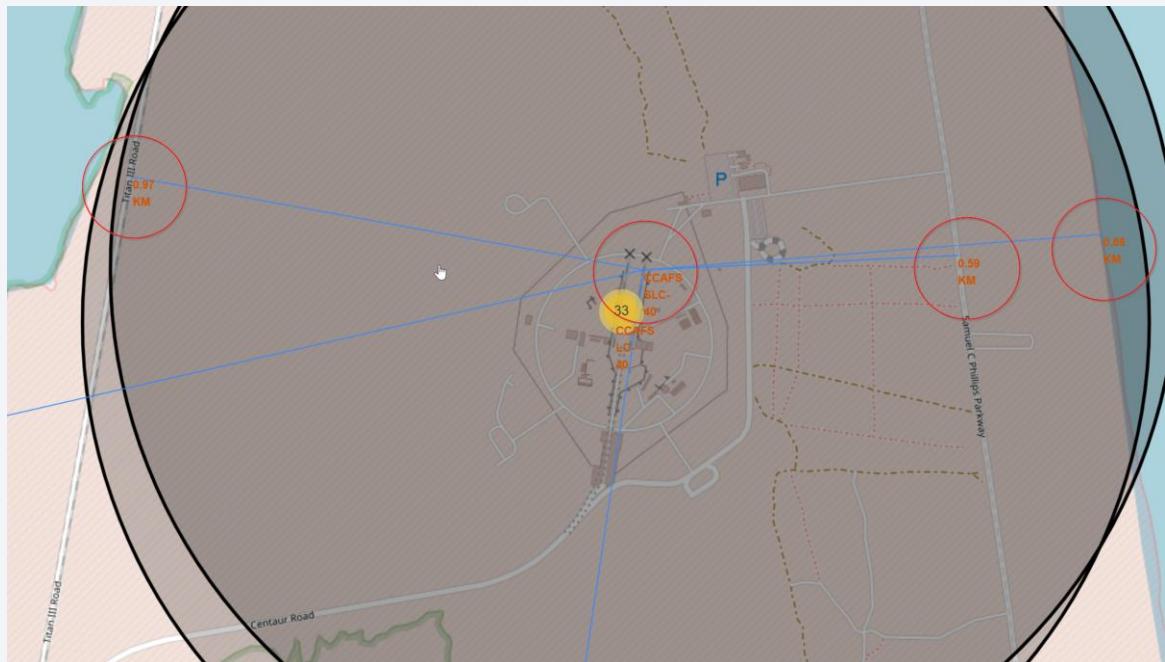
- The majority of launches have occurred in Florida.
- Interactive maps were generated and launch locations were clustered by [lat, long].
- Features were expanded to show green and red popup icons to indicate a success (Green) or failure (Red).
- Site CCAFS has two listed launch sites that are nearly on top of each other.



# Proximity to points of interest

- Due to the amount of launches I chose site CCAFS to analyze proximity of Objects of interest.
  - Drew a polyline to these points and calculated the length of the line taking account the curvature of the earth.
  - This site in particular is close to railways and highways – this is likely for ease of supply transport.
  - Launch site is far from the nearest city – likely for safety and to reduce noise pollution.

Distances of nearest landmarks to Launch Site CCAFS SLC-40:  
highway: 0.5887264310853515 km  
city: 18.497697056288423 km  
railroad: 0.9739935974925776 km  
Disney World: 99.19703464318563 km





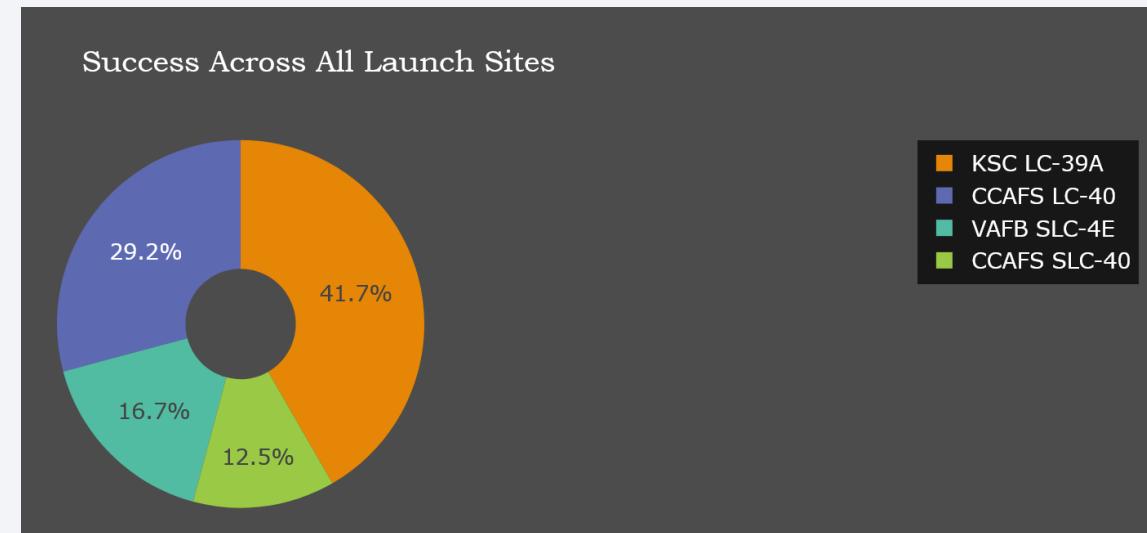
Section 4

# Build a Dashboard with Plotly Dash

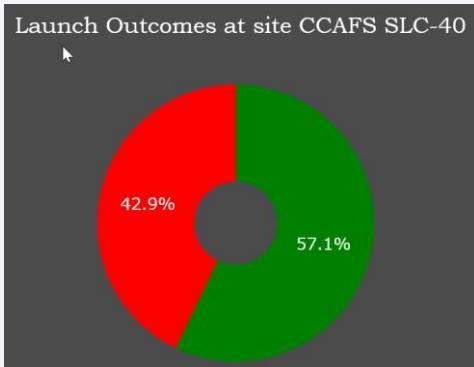
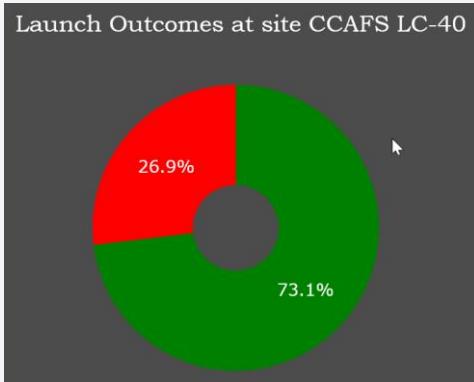
# Dashboard Application – All sites, rate of Success

---

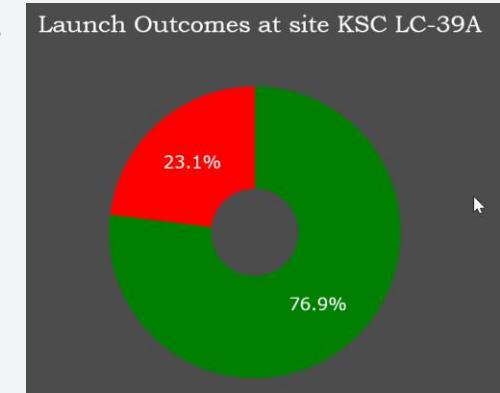
- From the data KSC LC-39A has the highest share of successful launches.
- Using the Folium Maps it was identified that sites CCAFS LC-40 and CCAFS SLC-40 were at a nearly identical location.
  - More information is needed to identify whether these two location are materially different, or why they are listed differently. When combined (41.7%) they are the same as KSC LC-39A. More on the next slide.



# Sites with highest success rates

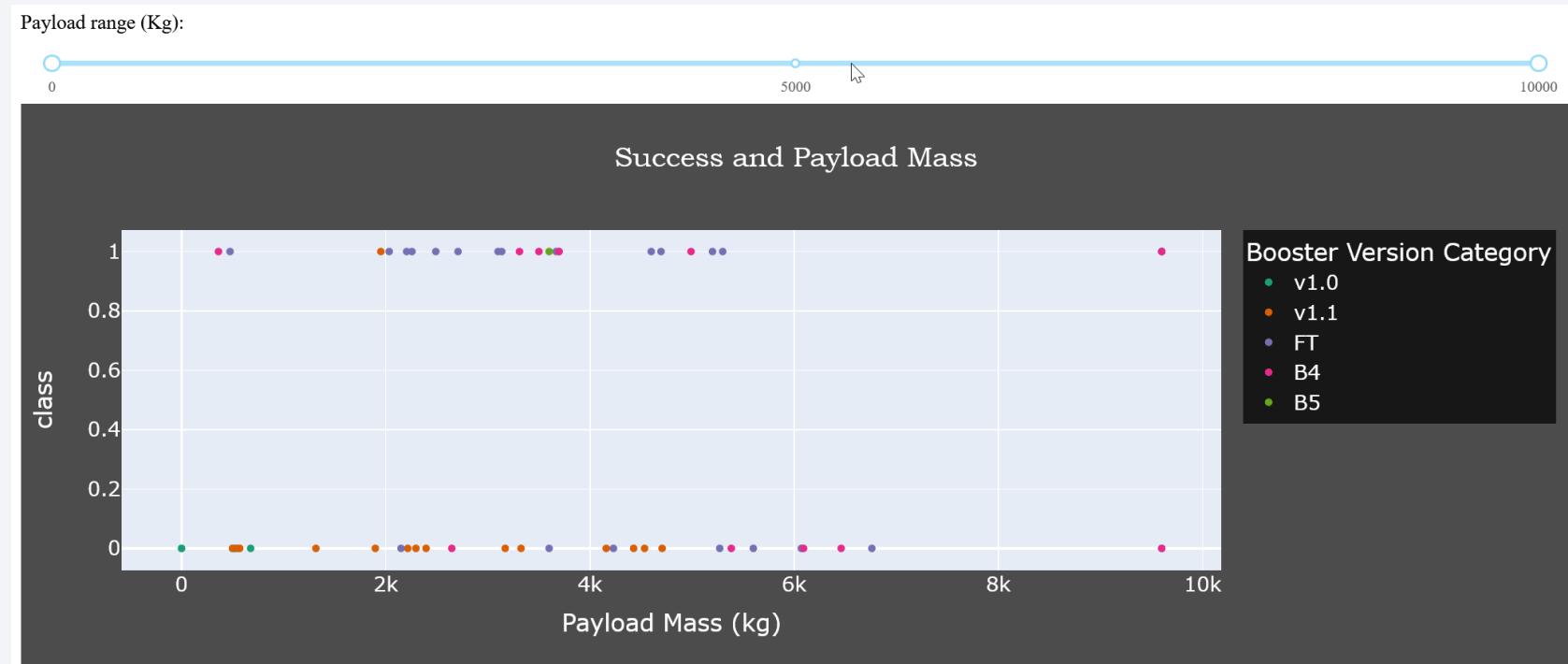


- On right: KSC LC-39A shows the highest success rate by single location. It also has the largest share of success across all sites.
- On left: CCAFS locations when combined show equal share (41.7%) of all successes. As individual locations they each show lower than KSC. Naturally, if combined they would show lower than KSC as a single site. CCAFS SLC-40 is the lowest success rate among those in the study.



# Payload and launch outcome

- Dash Application was given a payload range slider to slice the graph for preference.
- There does not seem to be a clear relationship between Payload mass and success.



The background of the slide features a dynamic, abstract design. It consists of several curved, overlapping bands of color. A prominent band on the left is a bright blue, while another on the right is a warm yellow. These colors transition into lighter, more diffused tones towards the edges of the frame. The overall effect is one of motion and depth.

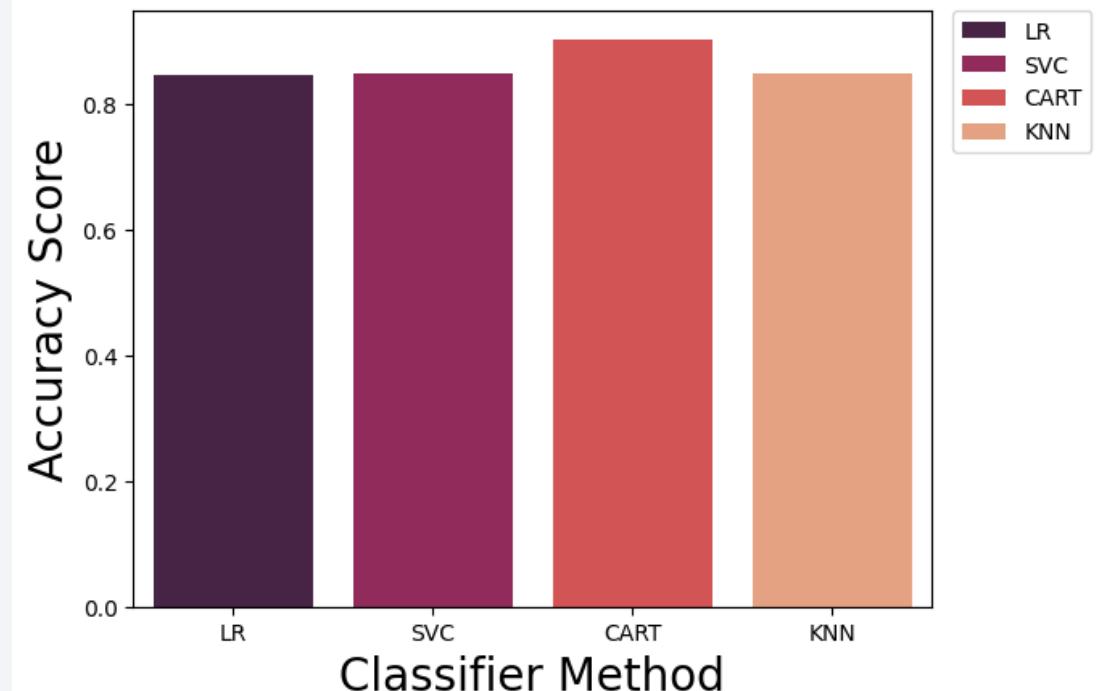
Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

- Decision Tree Model (CART) had the highest classification accuracy.
  - It should be noted that I adjusted some of the parameters to improve the model.
  - The provided parameters would continue to fail so I adjusted the ‘max\_features’ to include  $0 < \text{floats} < 1$  as described in sklearn documentation.
  - As shown below my hyperparameters tuned by cross-validation then selected a value outside the parameters provided by the IBM course.

```
parameters = {'criterion': ['gini', 'entropy'],
              'splitter': ['best', 'random'],
              'max_depth': [2*n for n in range(1,10)],
              'max_features': [0.1, .25, .5, .75, 1.0],
              'min_samples_leaf': [1, 2, 4],
              'min_samples_split': [2, 5, 10]}
tree = DecisionTreeClassifier()
```



```
Using the LR method, the accuracy is 0.8464285714285713
Using the SVC method, the accuracy is 0.8482142857142856
Using the CART method, the accuracy is 0.9035714285714287
Using the KNN method, the accuracy is 0.8482142857142858
```

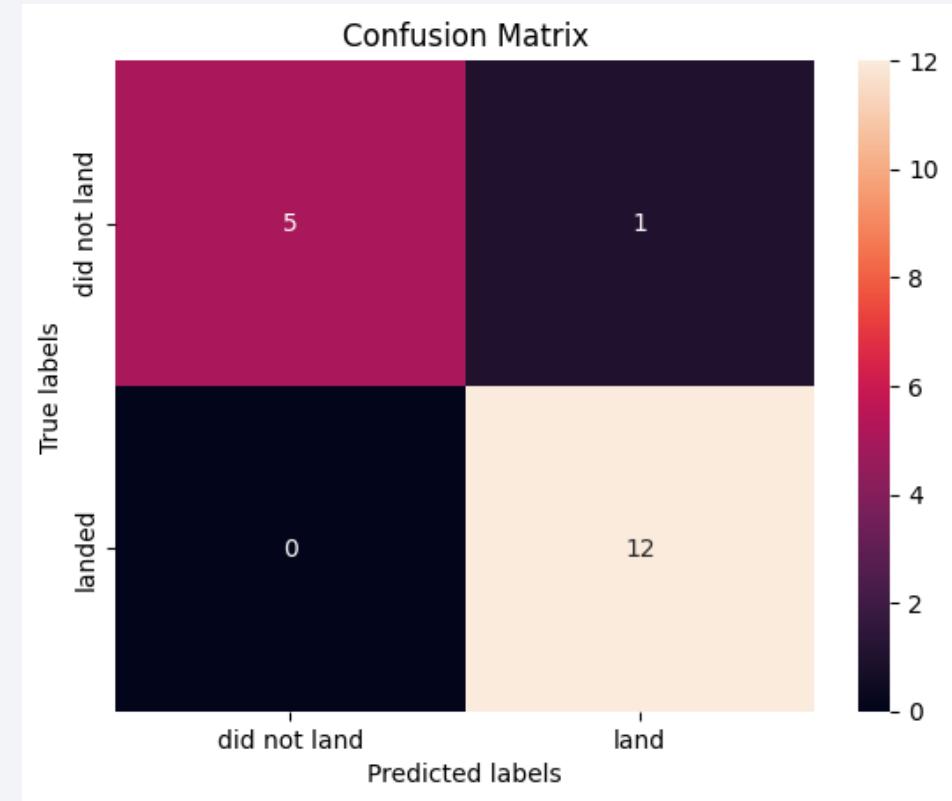
Tuned Hyperparameters:

```
tuned hyperparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 4, 'max_features': 0.25, 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.9035714285714287
```

# Confusion Matrix

---

- The model accuracy was very good when evaluated against the test set with only one false positive.
- The model correctly predicted 12 successful landings and 5 non-landings.



# Conclusions

---

- SpaceX shows a rapidly increasing success rate over time.
- The data reflects an increase in success over time and additionally, using a CART model we are able to reliably predict successes.
- SpaceX will continue to be a reliable provider of comparably low-cost commercial Space travel if they can continue to reuse components where competitors cannot.
- The data might suggest a change in mission due to the newly found prevalence of VLEO orbits.
  - Due to this being new, the model may need adjusted, retrained, or remade periodically to maintain efficacy.

# Appendix

---

Thank you!

