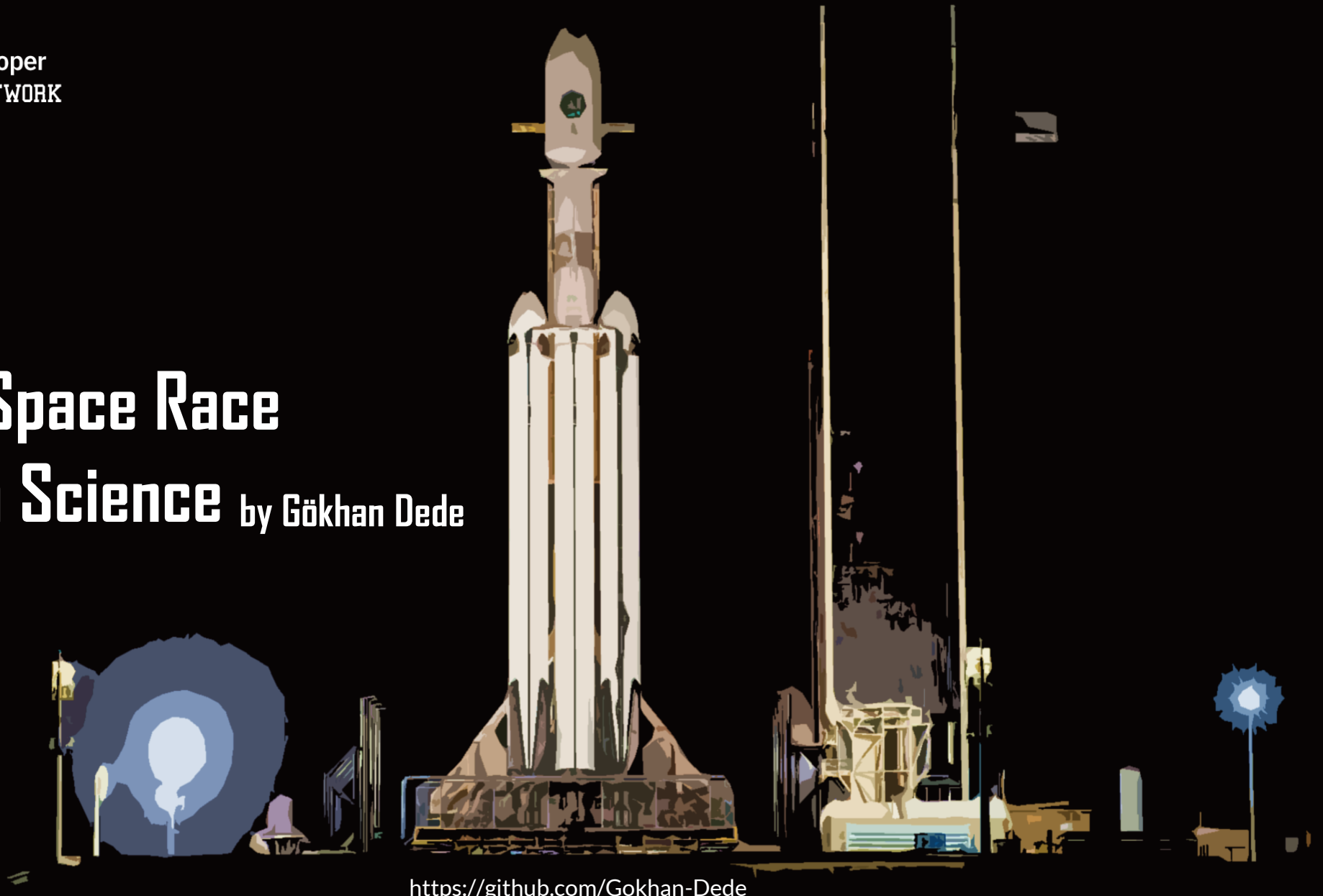
IBM Developer
SKILLS NETWORK

05.03.2023

Winning Space Race with Data Science

by Gökhan Dede



https://github.com/Gokhan-Dede

Outline

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

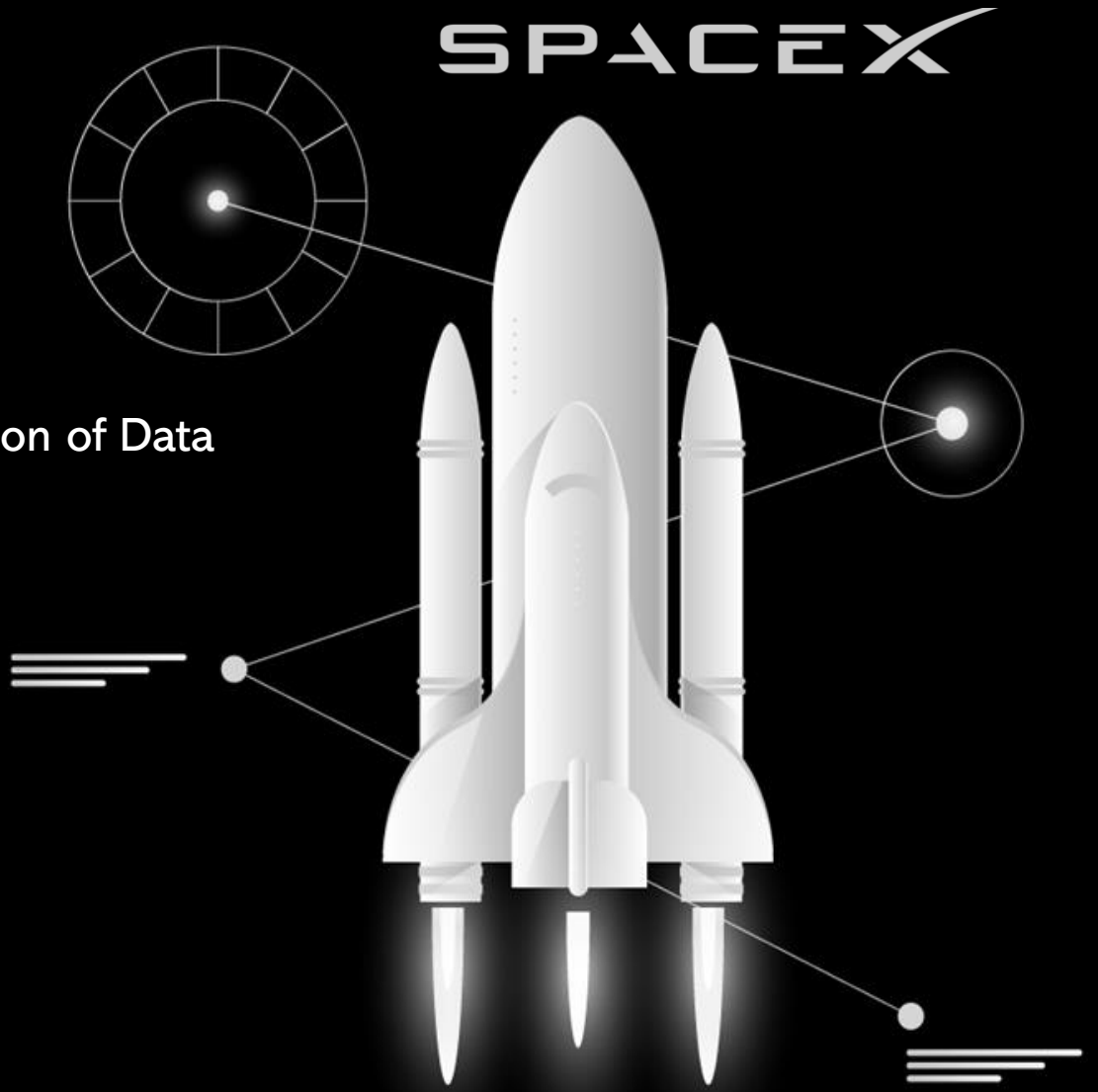
Executive Summary

Summary of techniques

- Collection of data via API and web scraping
- Exploratory Data Analysis (EDA) with Visualization of Data
- EDA with SQL
- Interactive Map using Folium
- Dashboard utilizing Plotly Dash
- Predictive Analysis

A summary of all results

- Outcomes of exploratory data analysis
- Dashboard with Interactive Maps
- Predictive Results



Introduction

As a revolutionary company, SpaceX has shaken up the space market by delivering rocket launches, in particular Falcon 9, for as little as $62 million, while rival companies charge as much as $165 million. Most of these cost savings have come from SpaceX's brilliant plan to land rockets again after each launch and use them for other missions. If you keep doing this, you can get the price even lower.

«For this project, my role as a data scientist for a startup that aims to compete with SpaceX is to develop a machine learning pipeline that can predict the success of a prospective first-stage landing. With the results of this experiment, competitors can determine their launch bids against SpaceX.»

The problems included:

- determining all of the variables that could affect the success of the landing;
- how each independent variable correlates to the outcome;
- the optimal conditions for optimizing the likelihood of a successful landing.

Section 1

Methodology

Methodology

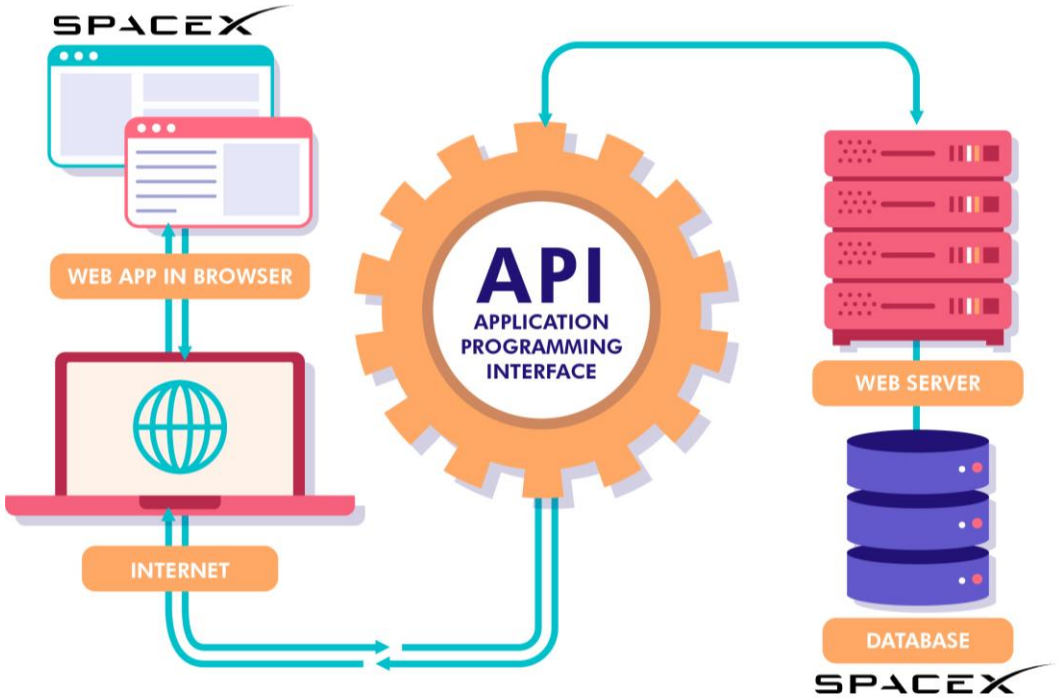
Executive Summary

- Data collection methodology:
  - Two different data sources were used to construct Space X dataframes:
    - SpaceX API: https://api.spacexdata.com/v4/
    - Webscraping from the internet (https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches)
- Perform data wrangling
  - After collecting and analyzing the characteristics of the collected data, a landing outcome label was derived to further enrich the dataset.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Data acquired up until this point was normalized, separated into training and test data sets, and evaluated by four different classification models, with the accuracy of each model assessed utilizing various variable combinations.

Data Collection

Firstly, the datasets were compiled by using the SpaceX REST API, so that we were able to interact with launch data from SpaceX.

- The API will give us information about launches, such as the type of rocket used, the payload that was sent, the parameters of the launch, the parameters of the landing, and the outcome of the landing.
- We intend to employ this data to determine the likelihood that a SpaceX rocket will land successfully.
- To access the SpaceX REST API, enter https://api.spacexdata.com/v4/ into your browser's address bar.



Secondly, webscraping Wikipedia with BeautifulSoup is another method used for obtaining Falcon 9 launch data.

Data Collection

SpaceX API

[GitHub URL to Notebook]

Request API and parse the SpaceX launch data

↓

Filter data to only include Falcon 9 launches

↓

Deal with Missing Values

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

In [10]:
```
response.status_code
```

Out[10]: 200

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

In [16]:
```
# Call getBoosterVersion
getBoosterVersion(data)
```

In [18]:
```
# Call getLaunchSite
getLaunchSite(data)
```

In [19]:
```
# Call getPayloadData
getPayloadData(data)
```

In [20]:
```
# Call getCoreData
getCoreData(data)
```

[21]:
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

In [24]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=launch_data[launch_data["BoosterVersion"]!="Falcon 1"]
data_falcon9
```

[25]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

In [27]:
```
# Calculate the mean value of PayloadMass column
mean_PayloadMass=data_falcon9["PayloadMass"].mean()

# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"] = data_falcon9["PayloadMass"].replace(np.nan, mean_PayloadMass)

data_falcon9.isnull().sum()
```

In [28]:
```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

1. The following URL will send a request for rocket launch data to SpaceX's API.
2. In order to maintain consistency, we were used the following static response object to return JSON results.
3. The JSON data in the response object converted into a Pandas dataframe using .json_normalize().
4. To retrieve features from the dataframe, we employed custom functions.
5. The columns of a dataframe were constructed by assigning extracted lists to dictionaries.
6. With a filtering operation on the BoosterVersion column, we were able to extract only the launches involving a Falcon 9 and save them in a separate dataframe we called data_falcon9.
7. We reseted the FlightNumber column after removing some values.
8. We calculated the mean for the PayloadMass using the .mean(). Then, we used it in the .replace() function to replace np.nan values in the data.
9. Dataframe was saved as a comma-separated values (CSV) file.

Data Collection

Scraping

[GitHub URL to Notebook]

Extract data
using BeautifulSoup

Get HTML Response
From Wikipedia

Create a data frame by
parsing the HTML tables

Normalize data into flat data file
such as .csv

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
response=requests.get(static_url).text
```

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

In [8]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables=soup.find_all("table")
```

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 = first_launch_table.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
```

In [12]:
```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

In [13]:
```
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
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
```

In [14]:
```
df=pd.DataFrame(launch_dict)
```

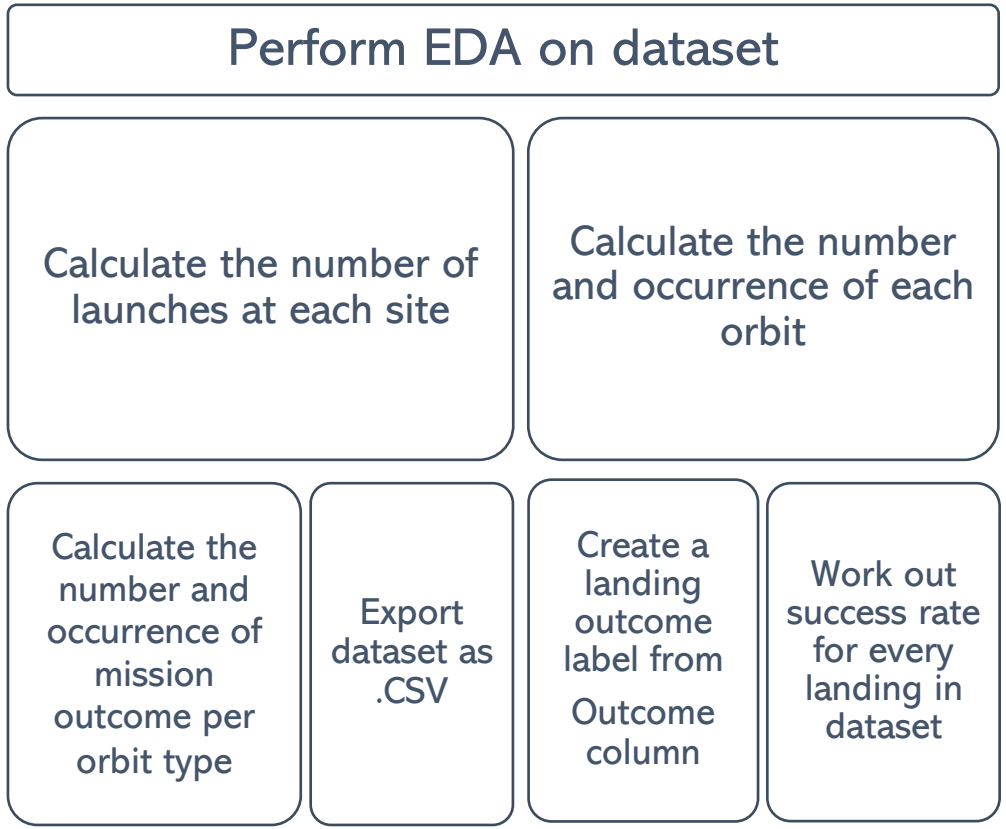
In [15]:
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

1. An object is created with HTML page from the URL.
2. We used requests.get() method to have response from the HTML object.
3. We created a BeautifulSoup object from the HTML response.
4. We used the find_all function in the BeautifulSoup object, with element type `table` to assign the result to a list called `html_tables`.
5. We iterated through the <th> elements and applied the extract_column_from_header() to extract column name one by one.
6. We created an empty dictionary with keys from the extracted column names. Later, this dictionary was converted into a Pandas dataframe
7. We appended data to keys in the dictionary.
8. The dictionary was converted into a dataframe.
9. Dataframe saved as a CSV file.

Data Wrangling

GitHub URL to Notebook

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



In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example,

- True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean.
- True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad.
- True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.

In this lab we will mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

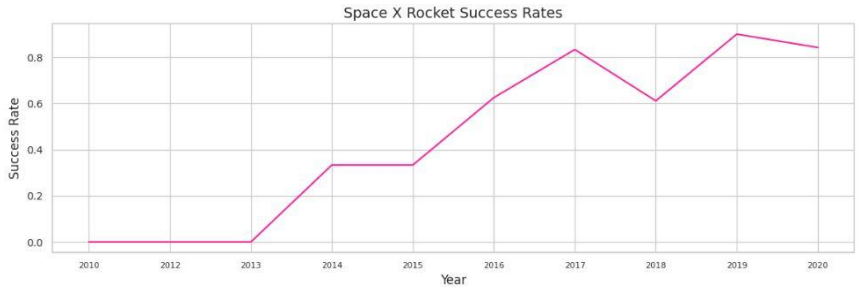
EDA with Data Visualization

GitHub URL to Notebook

Line Graph being drawn:

- Success Rate VS. Year

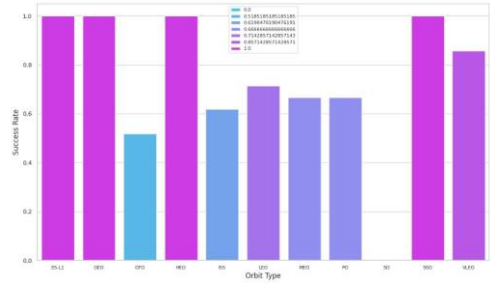
Line graphs are helpful because they clearly display variables and trends, and they may be used to extrapolate future data.



Bar Graph being drawn:

- Mean VS. Orbit

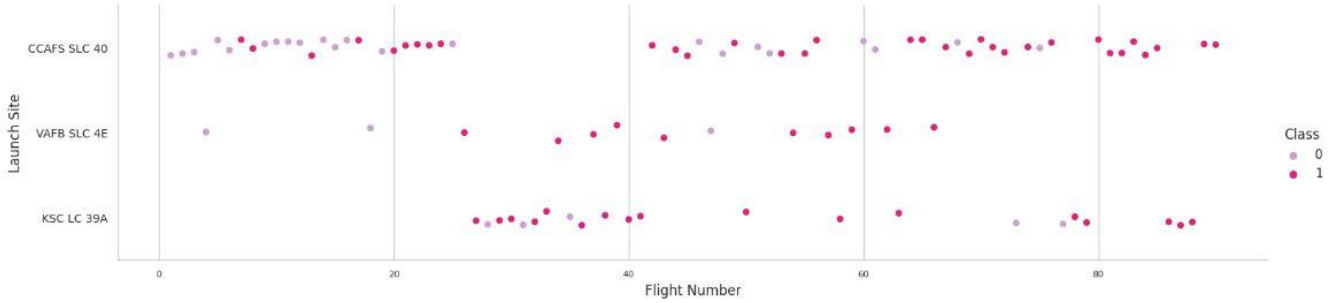
A bar diagram makes it easy to compare data across groupings. On one axis, the graph shows categories and on the other, discrete values. Showing the two axes' relationship is the purpose. Bar charts can display large data changes over time.

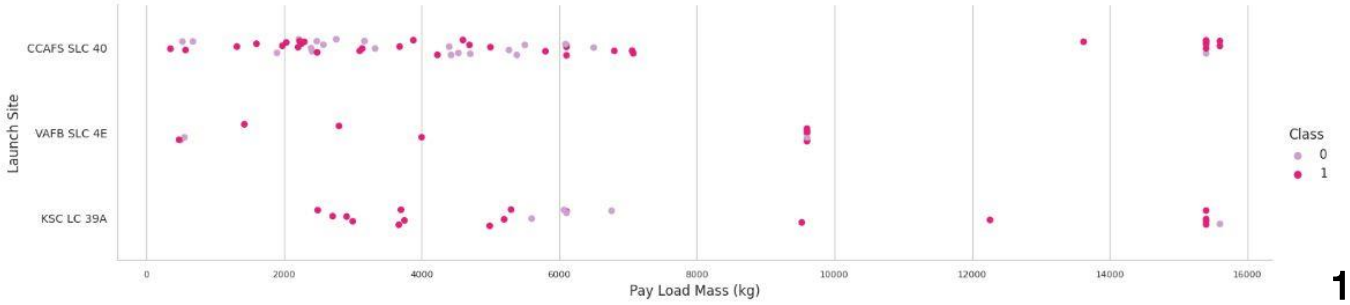


Scatter Graphs being drawn:

- Flight Number VS. Payload Mass
- Flight Number VS. Launch Site
- Payload VS. Launch Site
- Orbit VS. Flight Number
- Payload VS. Orbit Type
- Orbit VS. Payload Mass

Scatter plots reveal the extent to which one variable influences another. The connection between two metrics is known as a correlation. Data sets are typically quite large in scatter plots.





EDA with SQL

GitHub URL to Notebook

We executed SQL queries to extract details from the dataset. The following tasks are asked concerning the data to obtain the results from the dataset.

- Display the names of the unique launch sites in the space mission;
- Display 5 records where launch sites begin with the string "CCA";
- Display the total payload mass carried by boosters launched by NASA;
- Display average payload mass carried by booster version F9 v1.1;
- List the date when the first succesful landing outcome in ground pad was acheived;
- List the names of the boosters that have had success in drone ships and have a payload mass greater than 4000 but less than 6000;
- List the total number of successful and failed mission outcomes;
- List the names of the booster_versions which have carried the maximum payload mass;
- List the records that will display the month names, failure landing_outcomes in drone ships, ,booster versions, launch_site for the months in 2015;
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

Build an Interactive Map with Folium [GitHub URL to Notebook]

Folium

The goal is to create an interactive map based on the launch data. We marked each launch site on the map by placing a marker in the shape of a circle at its latitude and longitude coordinates and labeling it with the location's name.

The launch_sites(failure, success) dataframe was then classified as class 0 and class 1, with red and green markers on the map, using MarkerCluster().

Then, we utilized Haversine's formula to figure out how far away the launch sites were from a variety of landmarks, such as:

- How close are the launch sites to railways, highways, and coastlines?
- How close are the launch sites to nearby cities?

Build a Dashboard with Plotly Dash

GitHub URL to Python File

With Plotly dash, we created an interactive dashboard that lets the user explore the data in whatever way they like.

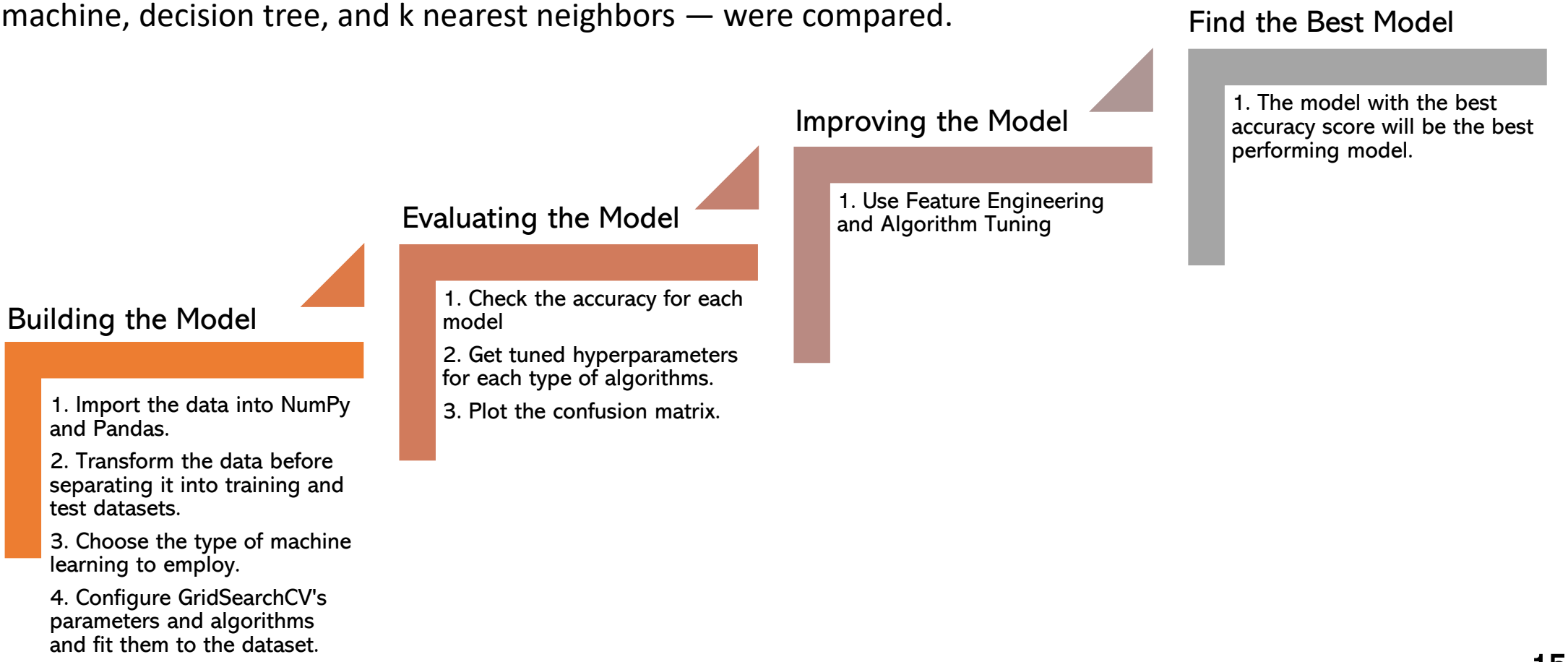
A pie chart is a graphical representation of data that shows the percentage breakdown of different categories. Pie Chart in our dashboard is showing the total launches by a certain site/all sites, and display relative proportions of multiple classes of data.

A scatter graph demonstrates the correlation between two independent factors, and it is the greatest way to demonstrate a nonlinear pattern.Result vs Payload Mass (Kg) relationship exhibited using a Scatter Graph for different Booster Versions.

Predictive Analysis (Classification)

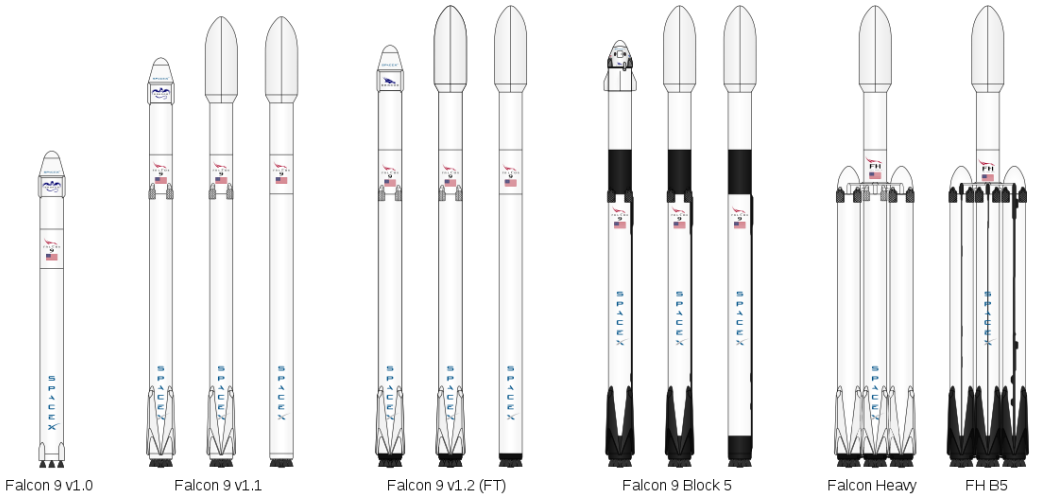
GitHub URL to Notebook

Four classification methods — logistic regression, support vector machine, decision tree, and k nearest neighbors — were compared.



Results

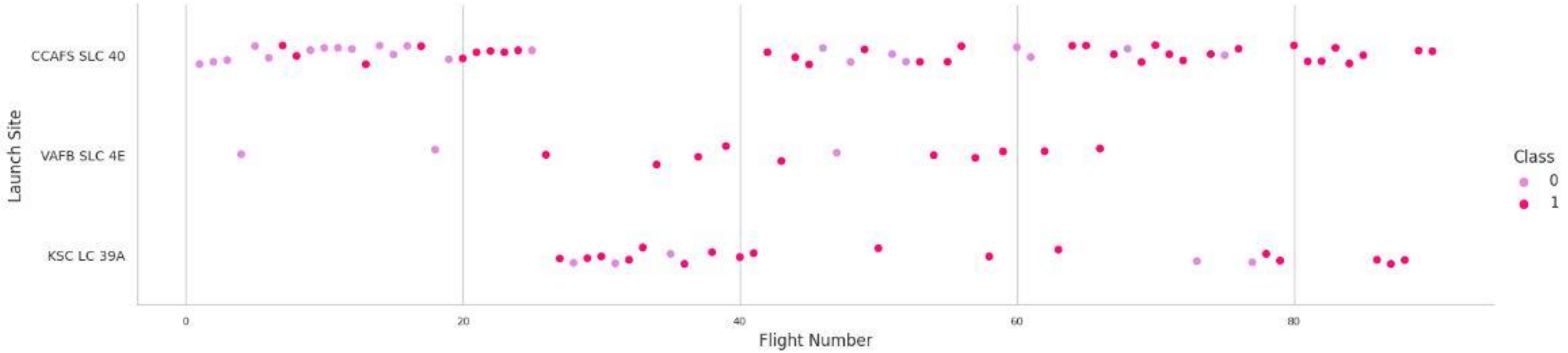
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

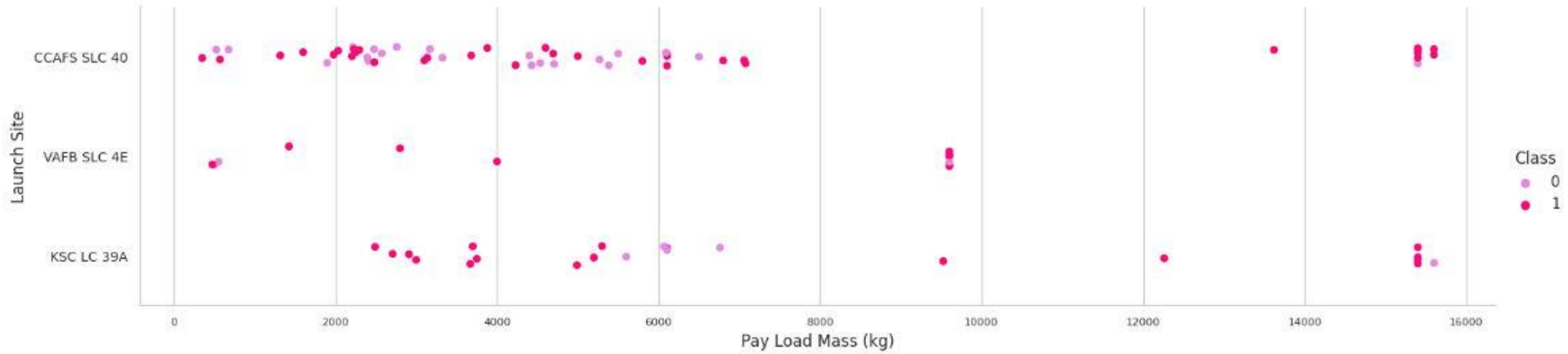
Insights drawn from EDA

Flight Number vs. Launch Site



- The most recent successes have all occurred at CCAF5 SLC 40, making it the optimal launch point over the time.
- The second successful launch location is VAFB SLC 4E, followed by KSC LC 39A.
- Moreover, it is feasible to see a rise in success rate over time.

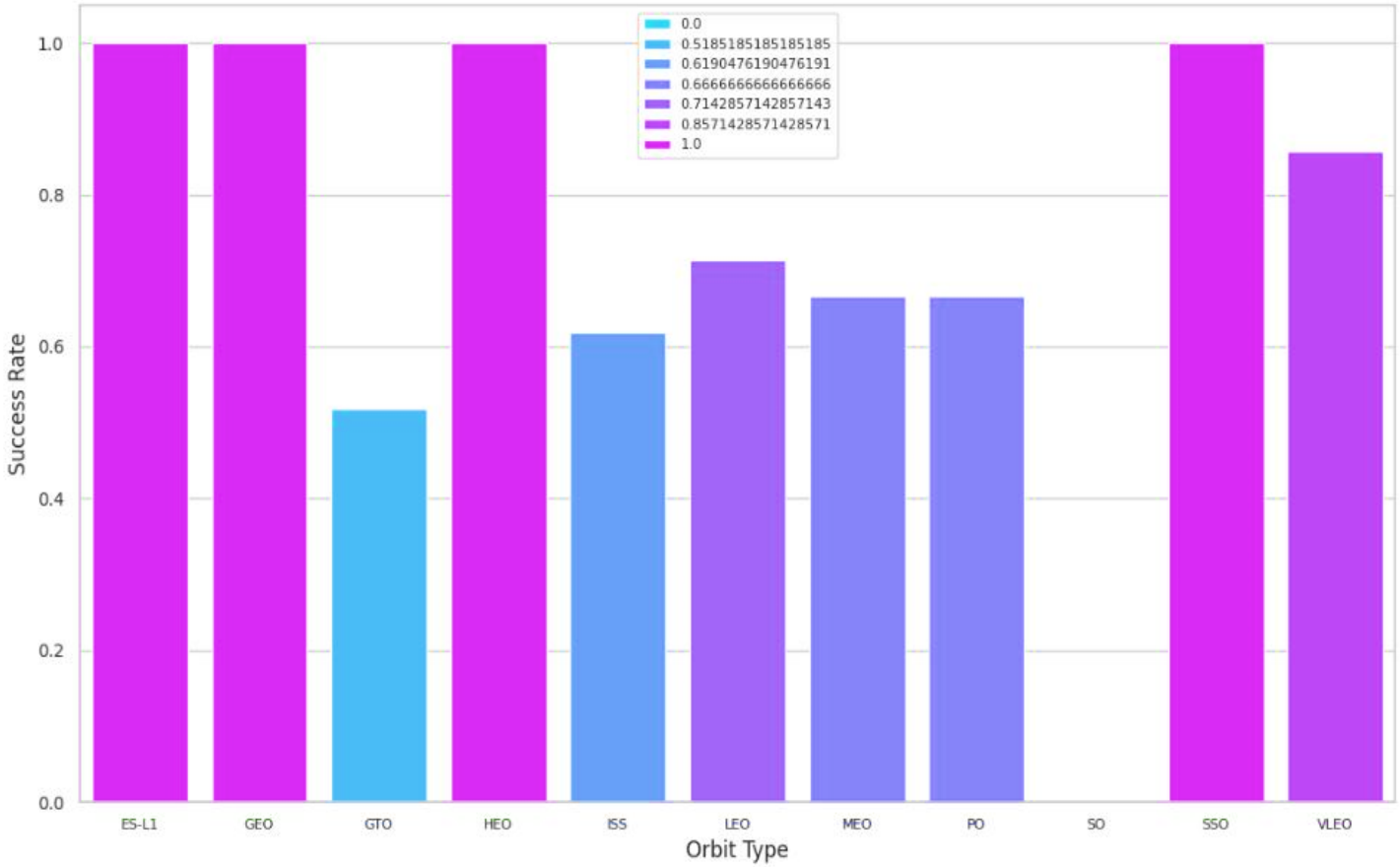
Payload vs. Launch Site



- The success rate for payloads exceeding 9,000kg is outstanding; payloads over 12,000kg are possible only at the CCAFS SLC 40 and KSC LC 39A launch sites.
- This scatter plot demonstrates that after the payload mass exceeds 7,000 kg, the success rate probability will climb dramatically. Yet, there is no conclusive evidence that the launch site's success percentage is dependent on the payload's mass.

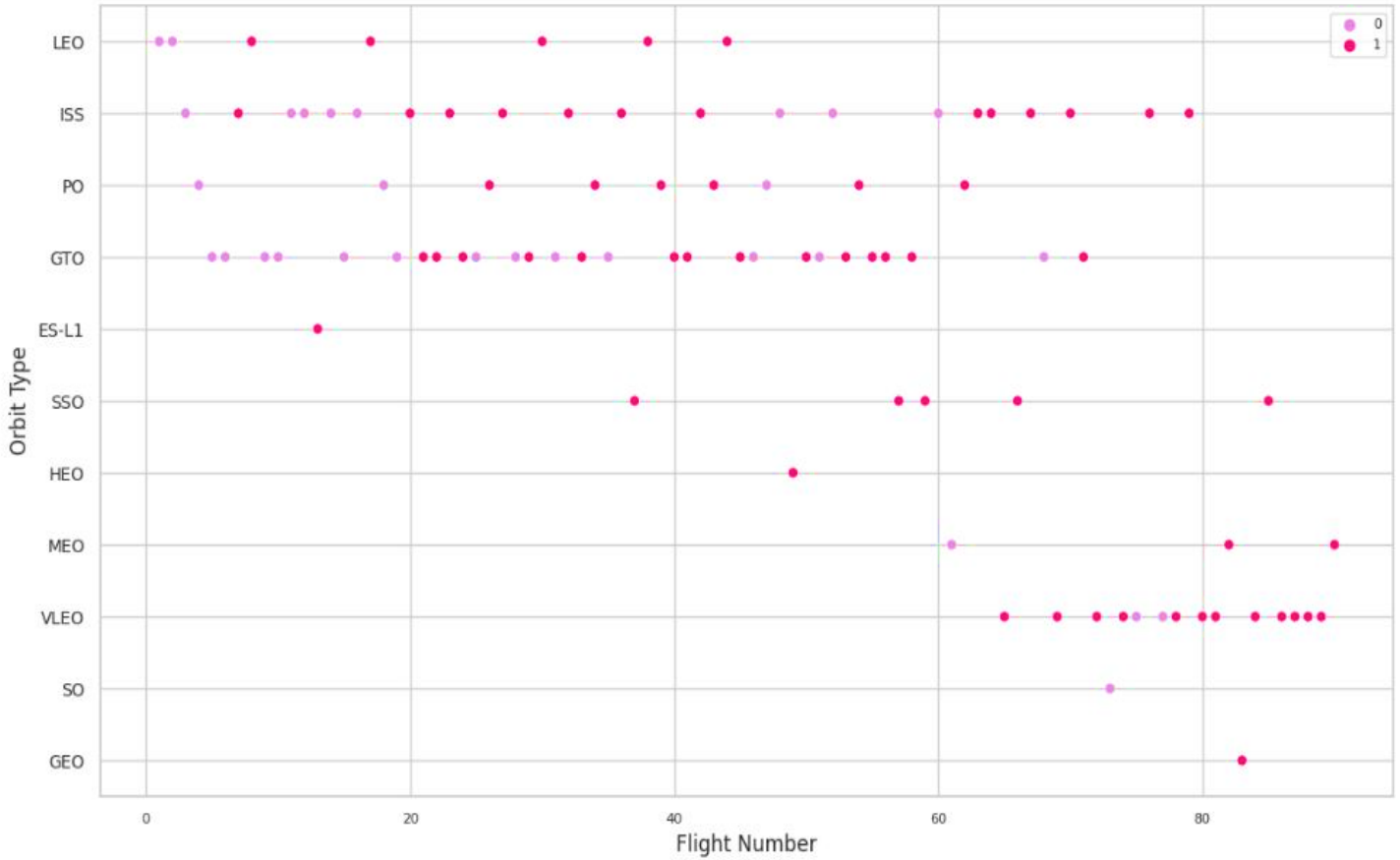
Success Rate vs. Orbit Type

- SSO, HEO, GEO, and ES-L1 orbits all resulted in a success rate of one hundred percent, whereas SO orbit resulted in a failure rate of one hundred percent.
- The bar chart depicted the potential for orbits to influence landing results. However, a deeper examination finds that certain orbits, such as GEO, SO, HEO, and ES-L1, have only one occurrence, indicating that this data need additional datasets to identify patterns or trends before we can draw conclusions.



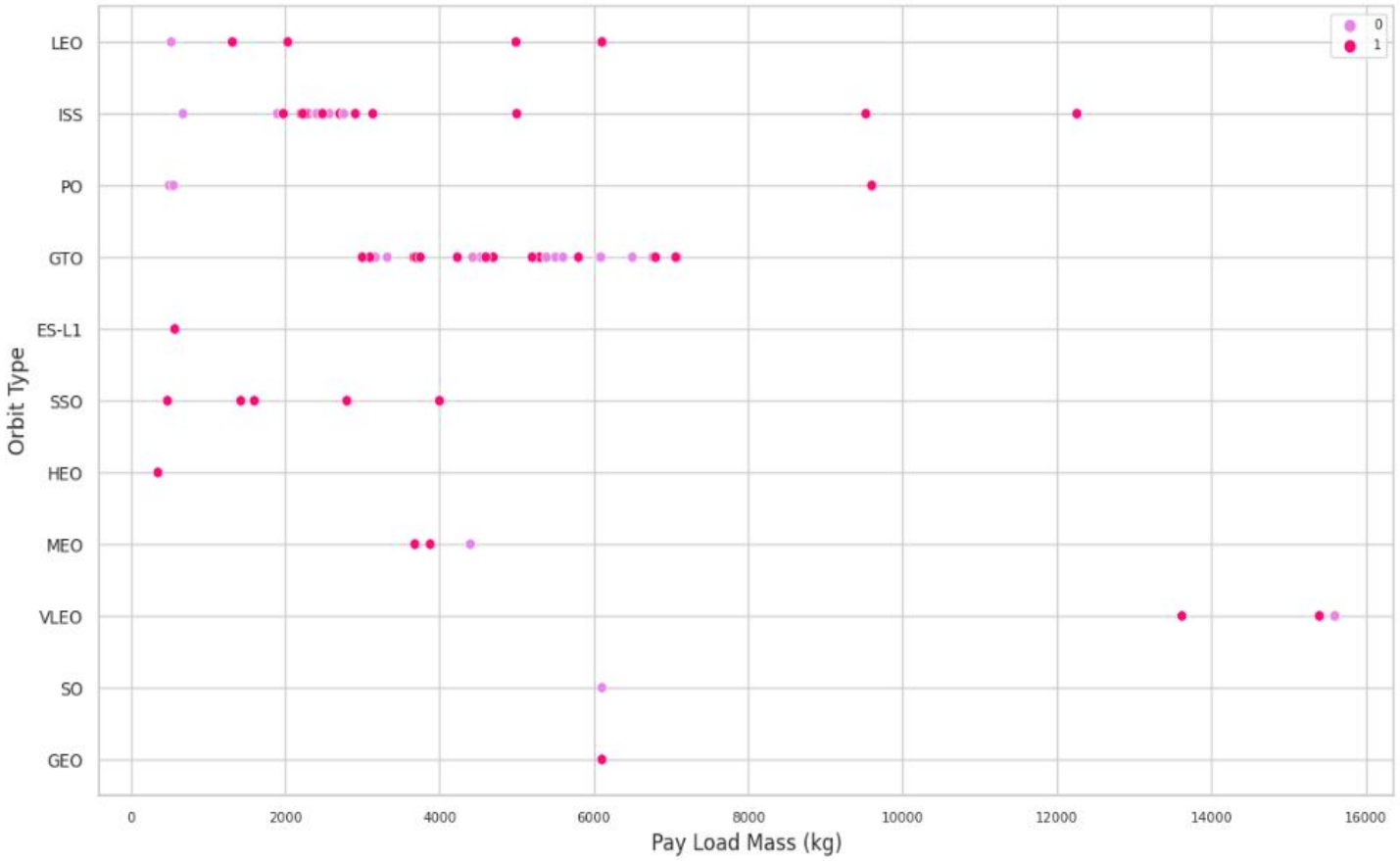
Flight Number vs. Orbit Type

- The scatter plot reveals that, with the exception of the GTO orbit, the more the flight number on each orbit, the higher the success rate (particularly on the LEO orbit).
- Orbits with a single occurrence should likewise be excluded from the preceding statement, as it requires further information.
- Apparently, success rates for all orbits have increased over time; the VLEO orbit represents a new economic possibility because its frequency has just increased.



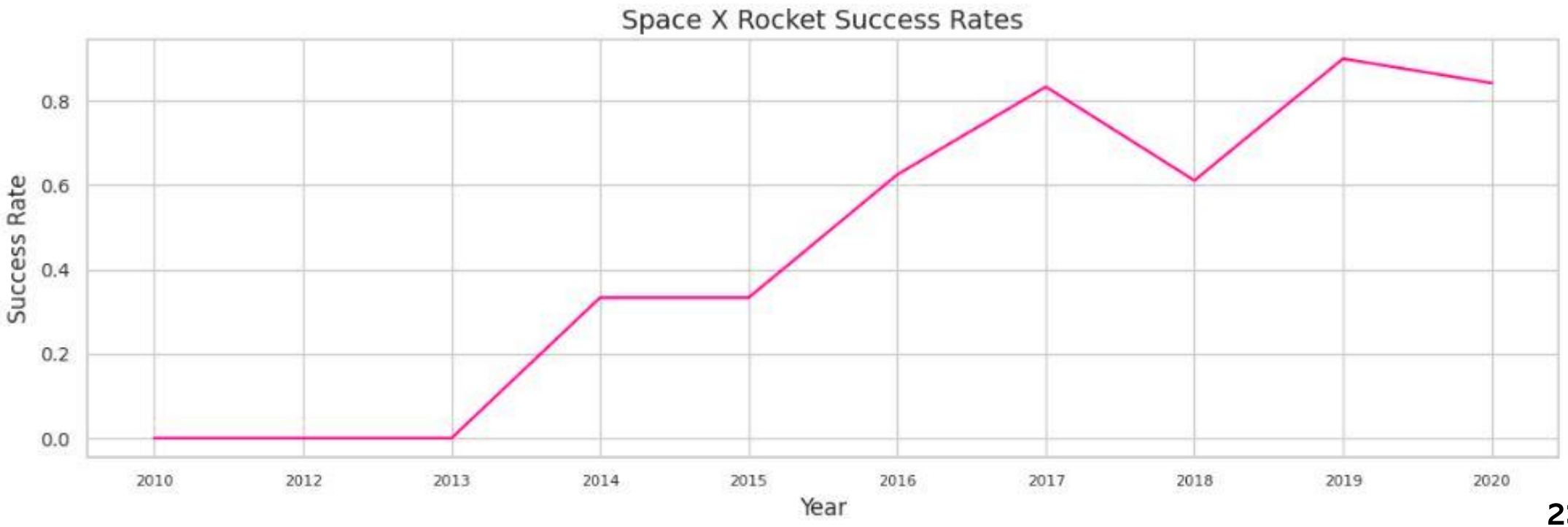
Payload vs. Orbit Type

- LEO, ISS, and PO orbits all benefit from bigger payloads. The impact is negative for MEO and VLEO orbits. There appears to be no correlation between the GTO orbit's depicted properties. Moreover, additional information is necessary for SO, GEO, and HEO orbits to discover trends or patterns.
- It is essential to remember that heavier payloads may have a negative impact on GTO orbits but a positive impact on ISS and LEO orbits.



Launch Success Yearly Trend

- From 2013 through 2020, the line chart clearly illustrates a growing tendency. If this pattern continues over the following year and beyond, the success rate will eventually rise to 100%.
- The success rate began to rise in 2013 and continued to rise until 2020. It appears that the first three years were a time of technological adjustment and development.



All Launch Site Names

- We ran a SQL query on the SPACEXTBL dataset with the keyword DISTINCT to retrieve the unique names of the launch sites.

In [8]:
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

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

Out[8]:

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

- Based on the query results, there are four launch locations. They are derived by filtering the dataset for unique instances of "launch_ values."

Launch Site Names Begin with ‘CCA’

- The following query was performed to retrieve the results for the top 5 launch sites that start with the characters 'CCA’.

In [9]:
```
%sql SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

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

Out[9]:

Launch_Site
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40
CCAFS LC-40

- The LIMIT 5 clause specifies that just the first five records from SPACEXTBL will be returned, and the LIKE keyword serves as a wild card by matching any string, so the string "CCA%" implies that the "LAUNCH_SITE" name must begin with CCA.

Total Payload Mass

- With the query below, we were able to estimate that NASA boosters have carried a total of 45596 kilos of payload.

In [10]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "TOTAL PAYLOAD MASS BY NASA(CRS)" FROM SPACEXTBL WHERE CUSTOMER = "NASA (CRS)";
```

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

Out[10]:

TOTAL PAYLOAD MASS BY NASA(CRS)
45596

- The SUM function is used to aggregate the values in the column labeled PAYLOAD MASS KG . With the WHERE clause, we can restrict our data collection and obtain exact results for the Customer column with a NASA (CRS) value.

Average Payload Mass by F9 v1.1

- With the query below, we determined that version F9 v1.1 of the booster had an average payload mass of 2,928.4 kilograms.

```
In [11]: %sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
 * sqlite:///my_data1.db
Done.
Out[11]: AVG(PAYLOAD_MASS__KG_)
         2928.4
```

- The average value in the field PAYLOAD MASS KG is calculated by the function AVG. With the WHERE clause, we can restrict the data set to just include rows about Booster version F9 v1.1 for analysis.

First Successfull Ground Landing Date

- The date of the ground pad landing's successful outcome was obtained using the query below. To verify this, we made use of the MIN() function. On May 1, 2017, we observed the first successful landing outcome on a ground pad.

In [12]:
```
%sql SELECT MIN(DATE) AS "First Successful Landing" FROM SPACEXTBL WHERE "Landing _Outcome"='Success (ground pad)';
```

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

Out[12]:

First Successful Landing
01-05-2017

- The minimum date in the column Date was calculated using the MIN() method. To narrow down the data set, we use the WHERE clause to look for the value "Success (ground pad)" in Landing _Outcome Success.

Successful Drone Ship Landing with Payload between 4000 and 6000

- With the use of the query below, we were able to pull out Boosters that have successfully landed on drone ship and had a payload mass of more than 4000 but less than 6000.

In [13]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE "Landing _Outcome"='Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

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

Out[13]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

- We selected the column labeled BOOSTER_VERSION from the table. To narrow down the data collection, we utilize the WHERE clause to specify that Landing_Outcome must be "Success" (drone ship). Further filter conditions are listed in the AND clause. Both the Payload MASS KG 4000 and 6000 parameters are required.

Total Number

of Successful and Failure Mission Outcomes

- We utilized the WHERE clause in nested queries to narrow the data, the LIKE '%' statement to filter "Success" and "Failure" statements in the MISSION OUTCOME column, and subqueries to aggregate the results. The following summary was derived from the query by counting records for each subquery.

In [14]:
```
%%sql SELECT (SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Success%') AS SUCCESS,
(SELECT COUNT("MISSION_OUTCOME") FROM SPACEXTBL WHERE "MISSION_OUTCOME" LIKE '%Failure%') AS FAILURE
```

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

Out[14]:

SUCCESS	FAILURE
100	1

Boosters Carried Maximum Payload

- By utilizing a WHERE clause and the MAX() function in a subquery, we were able to identify the booster that contained the most mass. Before collecting the maximum payloads from the table using the WHERE clause, we used the DISTINCT clause in the main query to retrieve only the unique booster versions from the BOOSTER VERSION column.

In [15]:
```
%sql SELECT DISTINCT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

Out[15]:

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

2015 Launch Records

- We selected the BOOSTER_VERSION and LAUNCH_SITE columns and filtered the DATE column using substr(Date, 4, 2) to retrieve just months. Then, we utilized the WHERE clause to get "Failure (drone ship)" values from the LANDING OUTCOME column and the AND statement in the WHERE clause to choose the year 2015 by using substr(Date,7,4)='2015'.

In [16]:
```
%%sql SELECT substr("DATE", 4, 2) AS MONTH, "BOOSTER_VERSION", "LAUNCH_SITE" FROM SPACEXTBL
WHERE "LANDING _OUTCOME" = 'Failure (drone ship)' and substr("DATE",7,4) = '2015'
```

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

Out[16]:

MONTH	Booster_Version	Launch_Site
01	F9 v1.1 B1012	CCAFS LC-40
04	F9 v1.1 B1015	CCAFS LC-40

Ranking Landing Outcomes

Between 2010-06-04 and 2017-03-20

- 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 and 2010-03-20. We applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcomes in descending order.

[17]:
```
%%sql SELECT "Landing _Outcome" as "Landing Outcome", COUNT("Landing _Outcome") AS "Total Count" FROM SPACEXTBL WHERE DATE BETWEEN '04-06-2010' AND '20-03-2017'
GROUP BY  "Landing _Outcome"
ORDER BY COUNT("Landing _Outcome") DESC ;
```

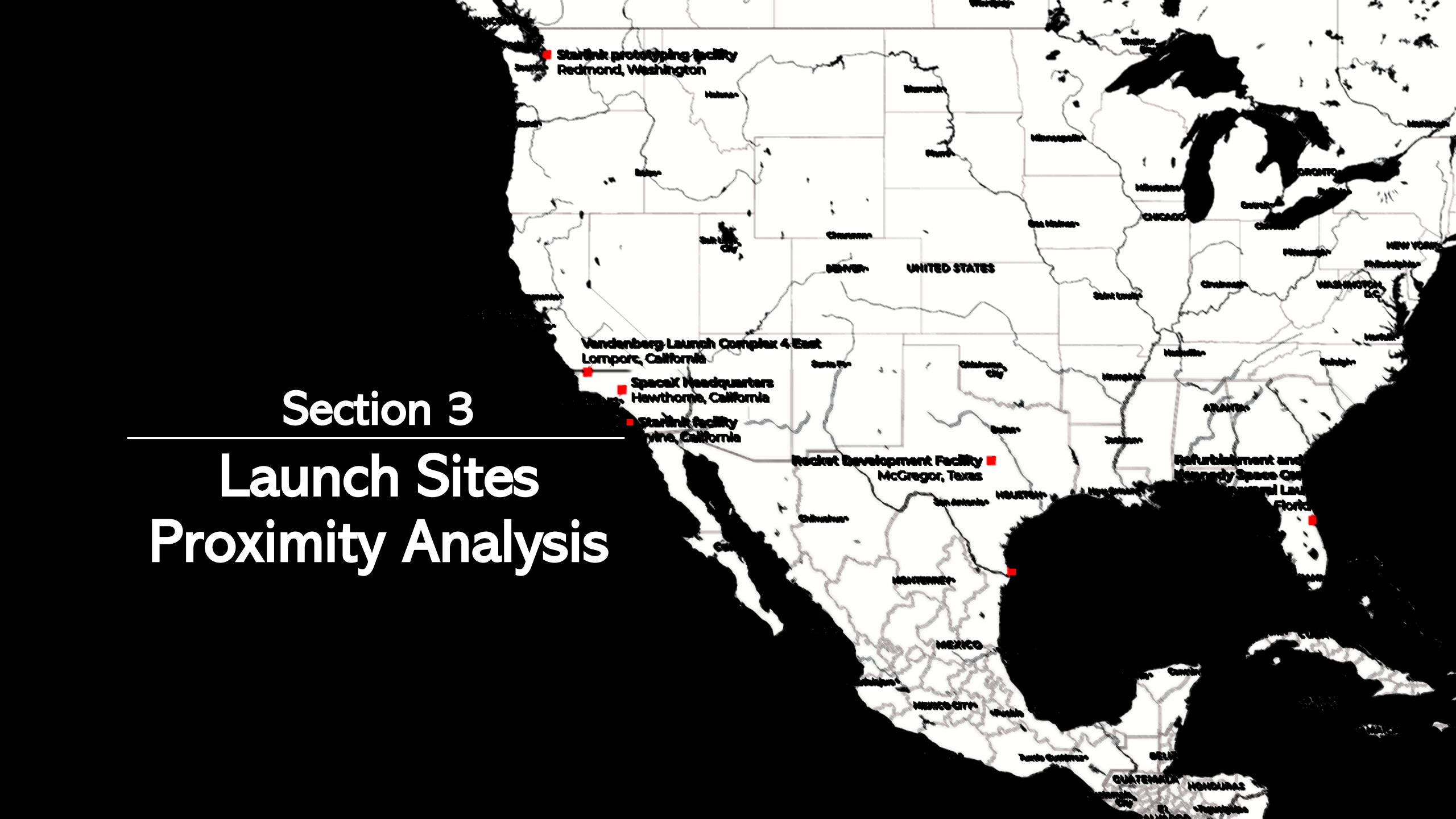
```
 * sqlite:///my_data1.db
Done.
```

[17]:

Landing Outcome	Total Count
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

Section 3

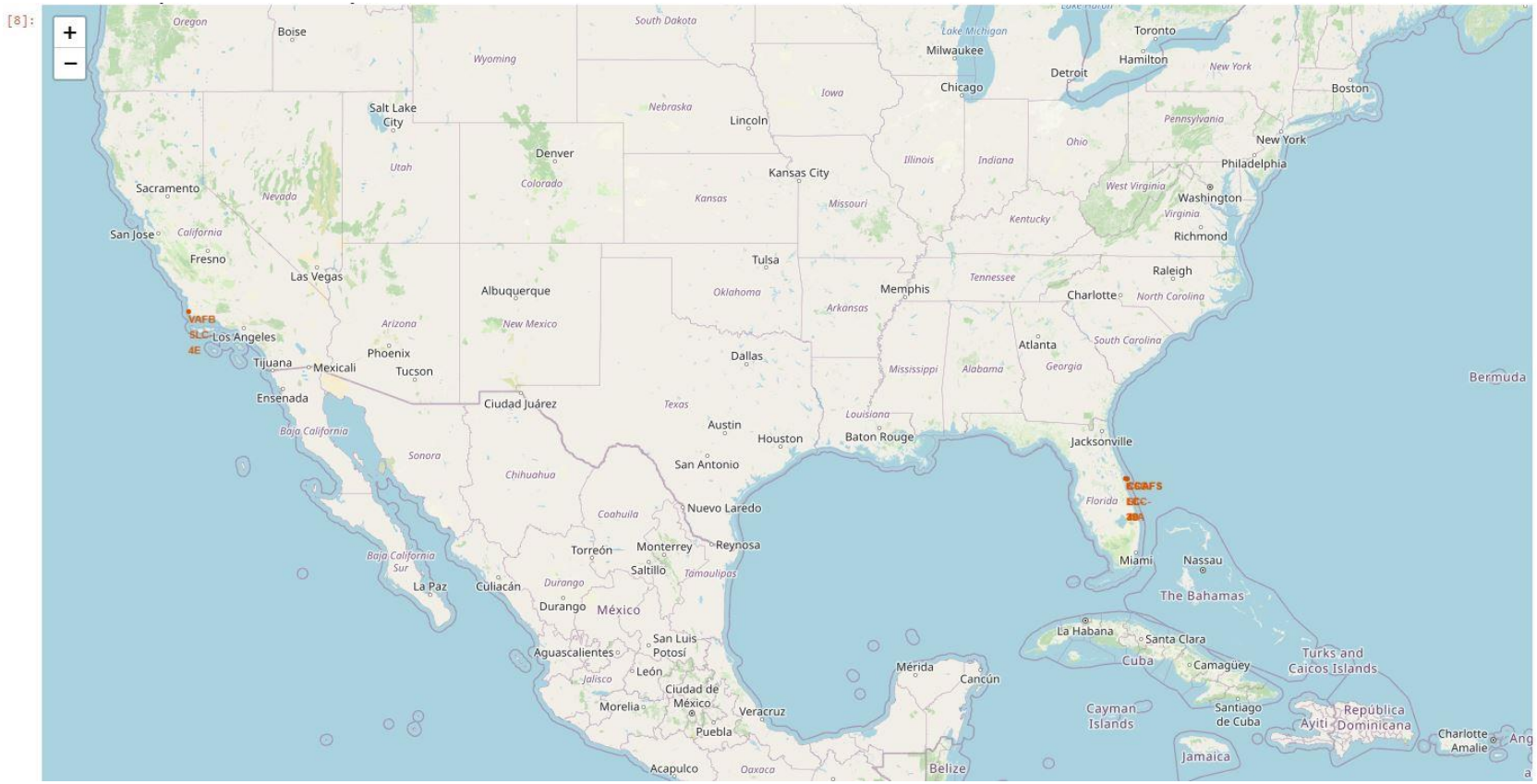
Launch Sites Proximity Analysis



The Map with Marked Launch Sites

Folium

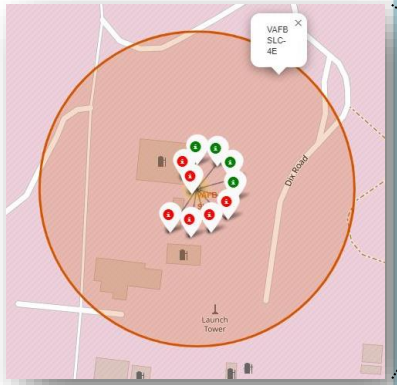
- As seen on the map, each and every one of SpaceX's launch locations are located entirely within the United States.

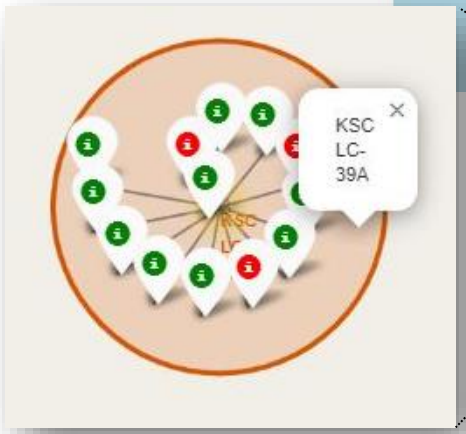


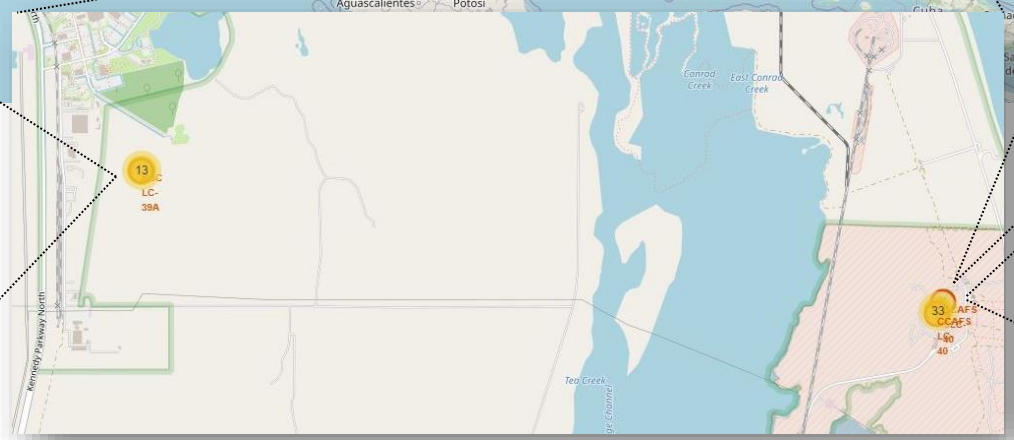
The Map with Marked Launch Sites

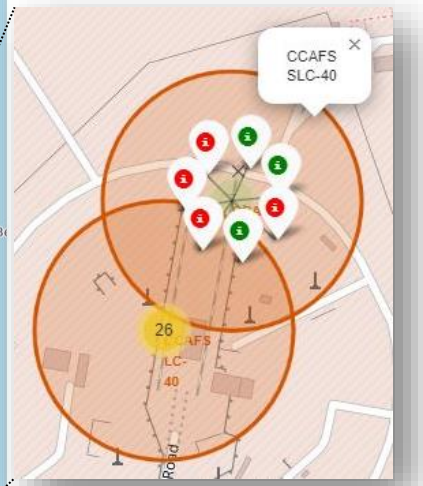
Folium

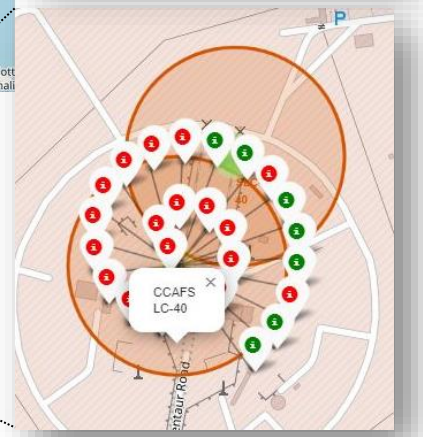
California Launch Sites







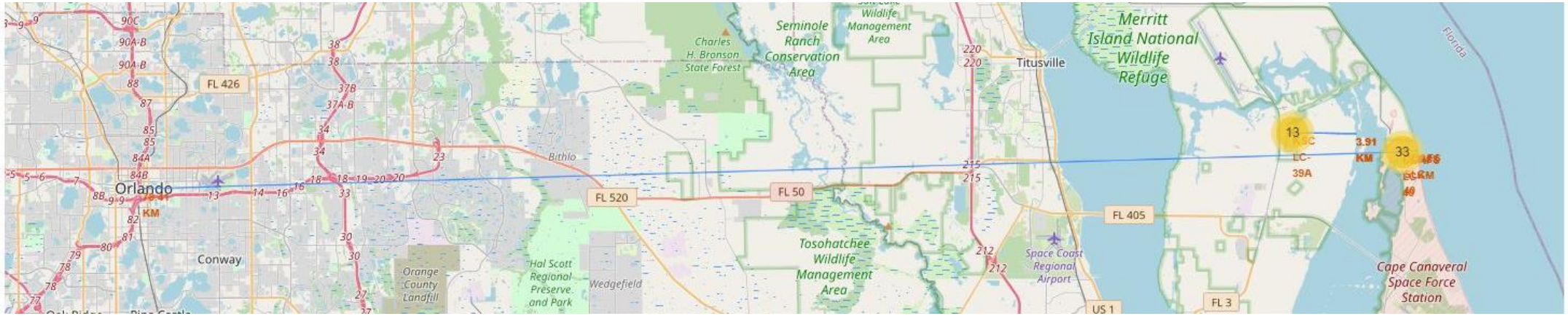




Relative Proximity of Launch Sites

Folium

- We calculated the distance between launch sites and landmarks using the Haversine formula in order to identify trends in their locational characteristics. Distances between launch locations CCAFS LC-40 and the city center and KSC LC-39A and the coast are depicted in the image below.

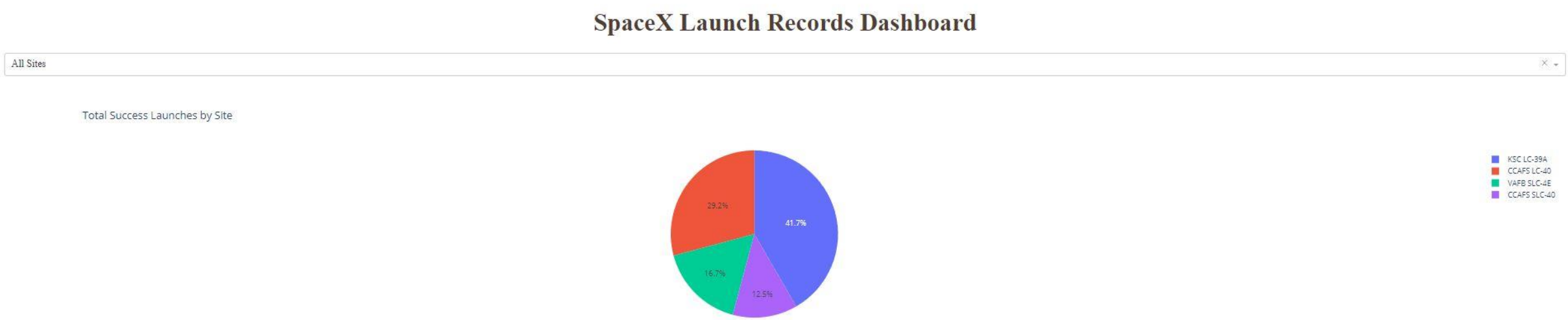


Section 4

Build a Dashboard with Plotly Dash

The Highest Success Rates by Site

Plotly | Dash

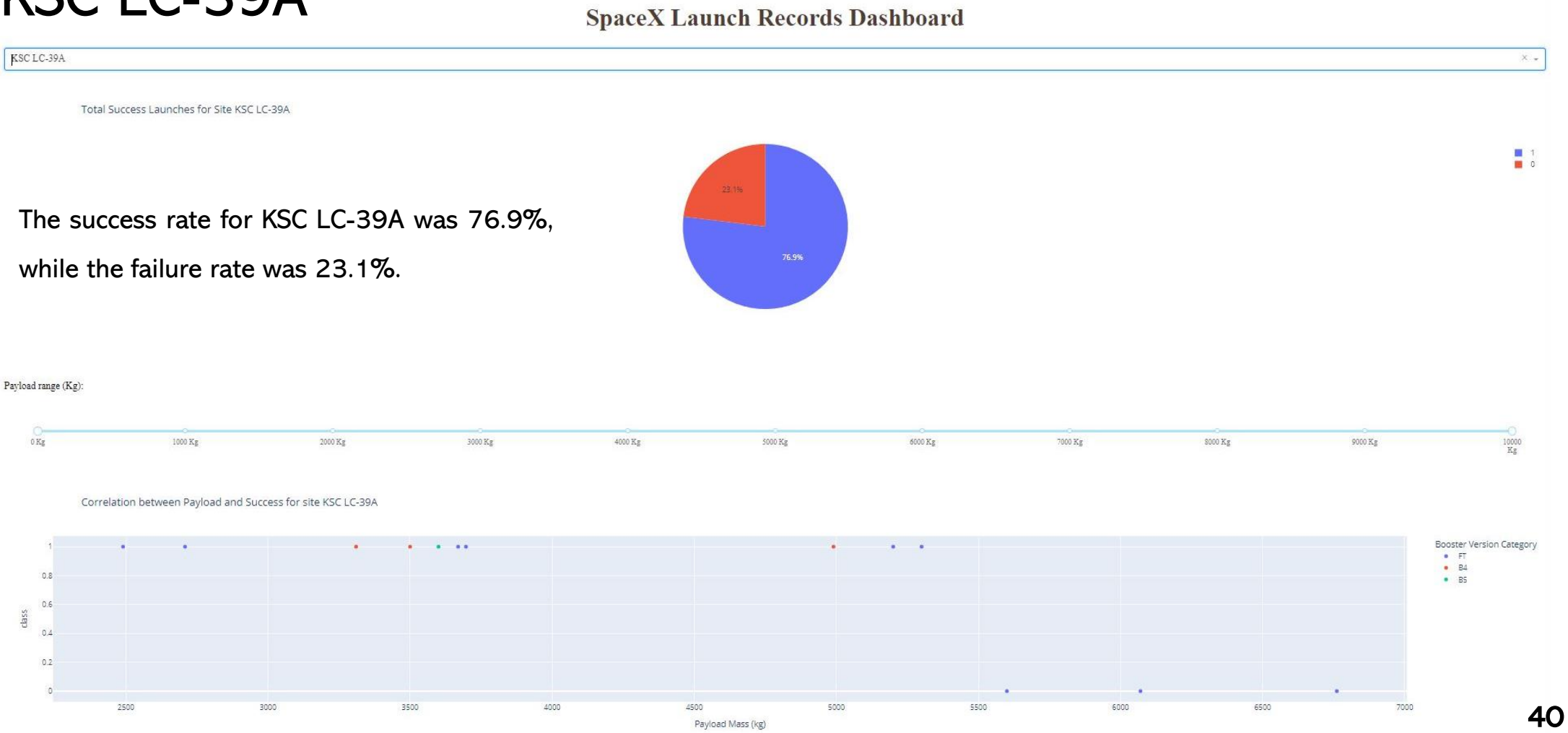


- With a 41.7% success rate, KSC LC-39A had the highest rate of successful launches among all locations. CCAFS LC-40 is the second launch site, with a success rate of 29.2%.

- With a 16.7% success rate, VAFB SLC-4E appears to be the third-highest rate location. CCAFS SLC-40 is the fourth launch location and has a 12.5% success rate.

The highest launch-success ratio: KSC LC-39A

Plotly | Dash



The success rate for KSC LC-39A was 76.9%, while the failure rate was 23.1%.

Scatter Plot

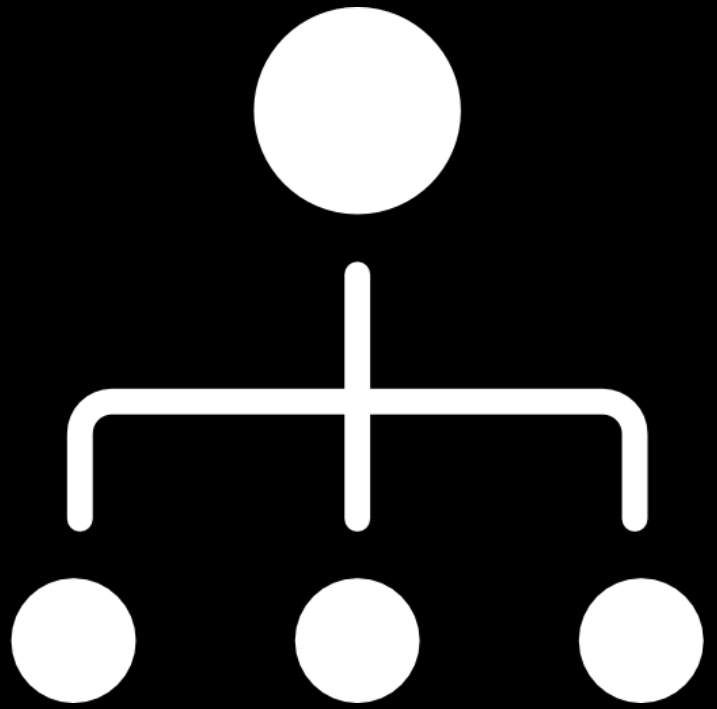
Plotly | Dash

Launch Outcome vs. Payload

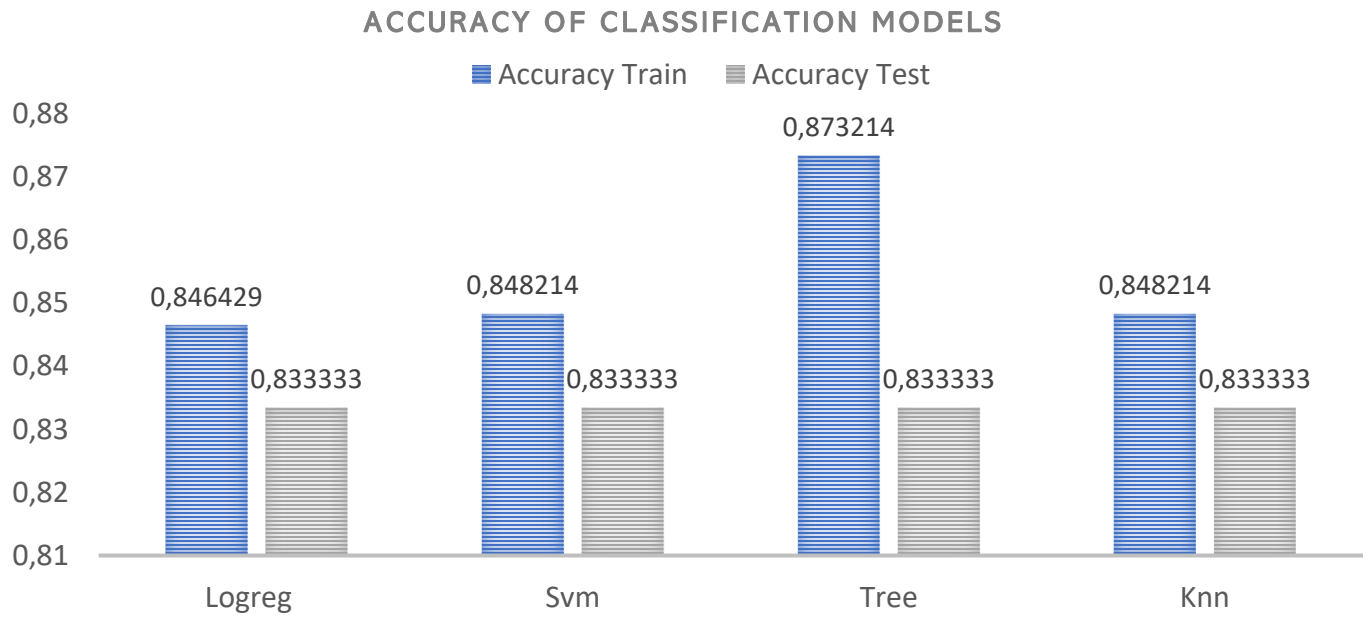
- The majority of launches with payloads under 2,000 kg were unsuccessful.
- The majority of launches between 2,000 and 4,000 kg were successful.
- The most successful launches had weights between 2,000 and 4,000 kg.
- The failure rate for launches with payloads between 4,000 and 6,000 kg was greater than the success rate.
- Except for one, all launches with payloads higher than 6,000 kg were unsuccessful.

Section 5

Predictive Analysis (Classification)



Classification Accuracy



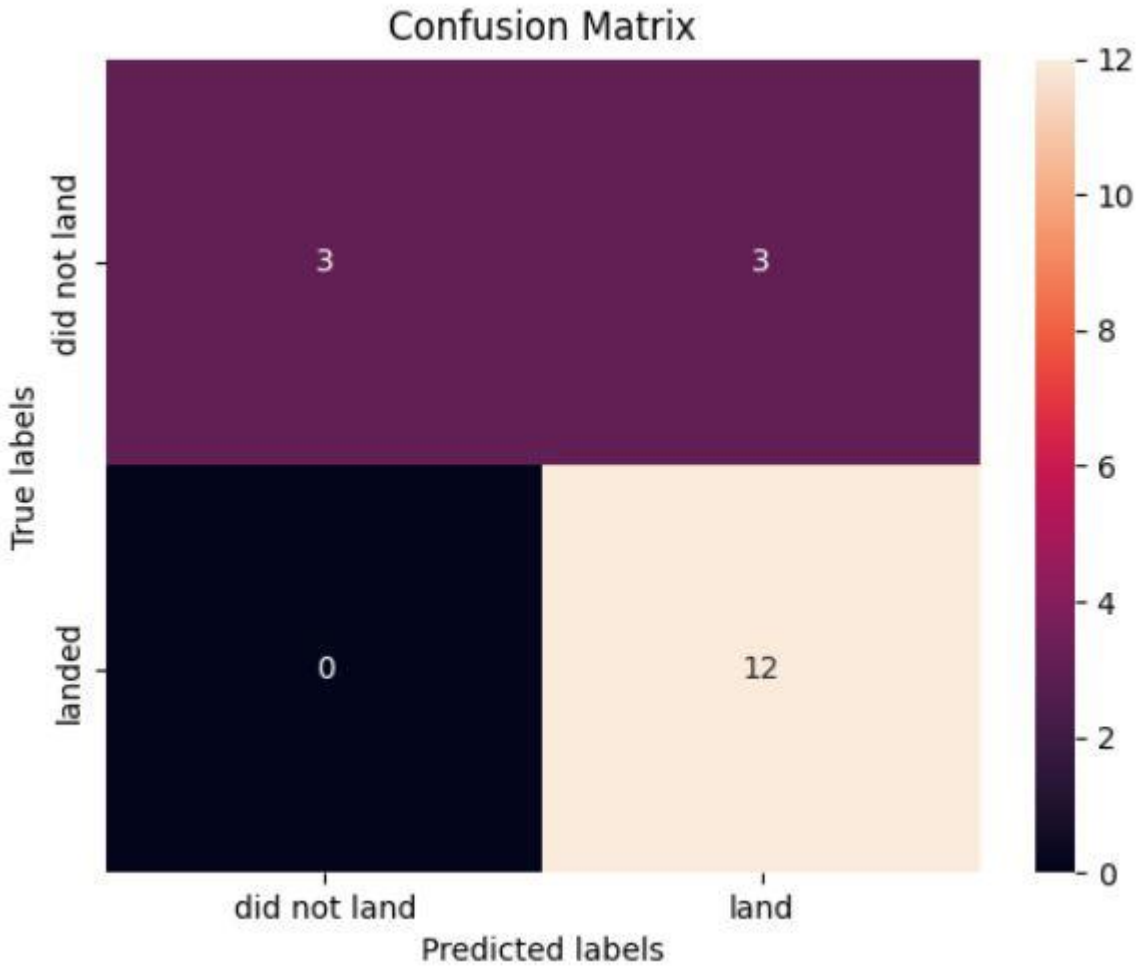
As shown by the bar chart, the accuracy of all classification algorithms for the test dataset is same; however, the classification tree algorithm has the highest accuracy for the training dataset. To test this outcome, the above code demonstrates that the classification tree algorithm is the most accurate model.

In [48]:

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
best_algorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',best_algorithm,'with a score of',algorithms[best_algorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.8732142857142857
Best Params is : {'criterion': 'gini', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 4, 'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix



The confusion matrix for the decision tree classifier demonstrates that the classifier can differentiate between classes. The primary issue is false positives. The classifier considers an unsuccessful landing as a successful landing.

Predicted Values	Actual Values: Positive (1)	Actual Values: Negative (0)
Positive (1)	TP	FP
Negative (0)	FN	TN

Conclusions

- For this dataset, our analysis leads us to the following conclusion:
- The Tree Classifier Algorithm is the most effective machine learning technique.
- The performance of lighter payloads (those weighing 4,000 kg or less) was superior to that of heavier payloads.
- Launching more than 6,000 kilos carries a considerable risk of failure.
- Starting in 2013, the success rate for SpaceX launches increased, directly proportional to the number of years until 2020, when it will eventually perfect the launches in the future.
- Among launch sites at KSC, LC-39A has the highest success rate (76.9%).
- SSO orbits have the highest success rate (100% or more than 1 occurrence). GEO, HEO, SSO, and ES L1 are the most often used orbital trajectories.

THANK YOU!