

Winning Space Race with Data Science

Lucas I. Lucas
17th April 2024



Outline

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

Executive Summary

Summary of Methodologies

The research attempts to identify the factors for a successful rocket landing. To make this determination, the following methodologies were used:

- **Collect** data using SpaceX REST API and web scraping techniques
- **Wrangle** data to create success/fail outcome variable
- **Explore** data with data visualization techniques, considering the following factors: payload, launch site, flight number and yearly trend
- **Analyze** the data with SQL, calculating the following statistics: total payload, payload range for successful launches, and total # of successful and failed outcomes
- **Explore** launch site success rates and proximity to geographical markers
- **Visualize** the launch sites with the most success and successful payload ranges
- **Build Models** to predict landing outcomes using logistic regression, support vector machine (SVM), decision tree and K-nearest neighbor (KNN)

Summary of All Results

Exploratory Data Analysis:

- Launch success has improved over time
- KSC LC-39A has the highest success rate among landing sites
- Orbit ES-L1, GEO, HEO, and SSO have a 100% success rate

Visualization/Analytics:

- Most launch sites are near the equator, and all are close to the coast

Predictive Analytics:

- All models performed similarly on the test set. The decision tree model slightly outperformed

Introduction

Background

SpaceX, a leader in the space industry, strives to make space travel affordable for everyone. Its accomplishments include sending spacecraft to the international space station, launching a satellite constellation that provides internet access and sending manned missions to space. SpaceX can do this because the rocket launches are relatively inexpensive (\$62 million per launch) due to its novel reuse of the first stage of its Falcon 9 rocket. Other providers, which are not able to reuse the first stage, cost upwards of \$165 million each. By determining if the first stage will land, we can determine the price of the launch. To do this, we can use public data and machine learning models to predict whether SpaceX – or a competing company – can reuse the first stage.

Explore

- How payload mass, launch site, number of flights, and orbits affect first-stage landing success
- Rate of successful landings over time
- Best predictive model for successful landing (binary classification)

Section 1

Methodology

Methodology

Steps

- **Collect** data using SpaceX REST API and web scraping techniques
- **Wrangle** data – by filtering the data, handling missing values and applying one hot encoding – to prepare the data for analysis and modeling
- **Explore** data via EDA with SQL and data visualization techniques
- **Visualize** the data using Folium and Plotly Dash
- **Build Models** to predict landing outcomes using classification models. Tune and evaluate models to find best model and parameters

Data Collection

Steps

- **Request data** from SpaceX API (rocket launch data)
- **Decode response** using `.json()` and convert to a dataframe using `.json_normalize()`
- **Request information** about the launches from SpaceX API using custom functions
- **Create dictionary** from the data
- **Create dataframe** from the dictionary
- **Filter dataframe** to contain only Falcon 9 launches
- **Replace missing values** of Payload Mass with calculated `.mean()`
- **Export data** to csv file

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The link to the notebook is:

https://github.com/Cyberage173-Creative/Applied-Data-Science-Capstone/blob/main/1_Jupyter-labs-spacex-Data-Collection-api.ipynb

1. Get request for rocket launch data using API

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

2. Use json_normalize method to convert json result to dataframe

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

```
In [13]: # apply json_normalize  
data = pd.json_normalize(static_json_df)
```

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

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

Data Collection - Scraping

Steps

- **Request data** (Falcon 9 launch data) from Wikipedia
- **Create BeautifulSoup object** from HTML response
- **Extract column names** from HTML table header
- **Collect data** from parsing HTML tables
- **Create dictionary** from the data
- **Create dataframe** from the dictionary
- **Export data** to csv file
- The link to the notebook is:
https://github.com/Cyberage173-Creative/Applied-Data-Science-Capstone/blob/main/2_Jupyter-labs-Webscraping.ipynb

```
1. Apply HTTP Get method to request the Falcon 9 rocket launch page
In [4]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

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

Out[5]: 200

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

Print the page title to verify if the BeautifulSoup object was created properly
In [7]: # Use soup.title attribute
soup.title

Out[7]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>

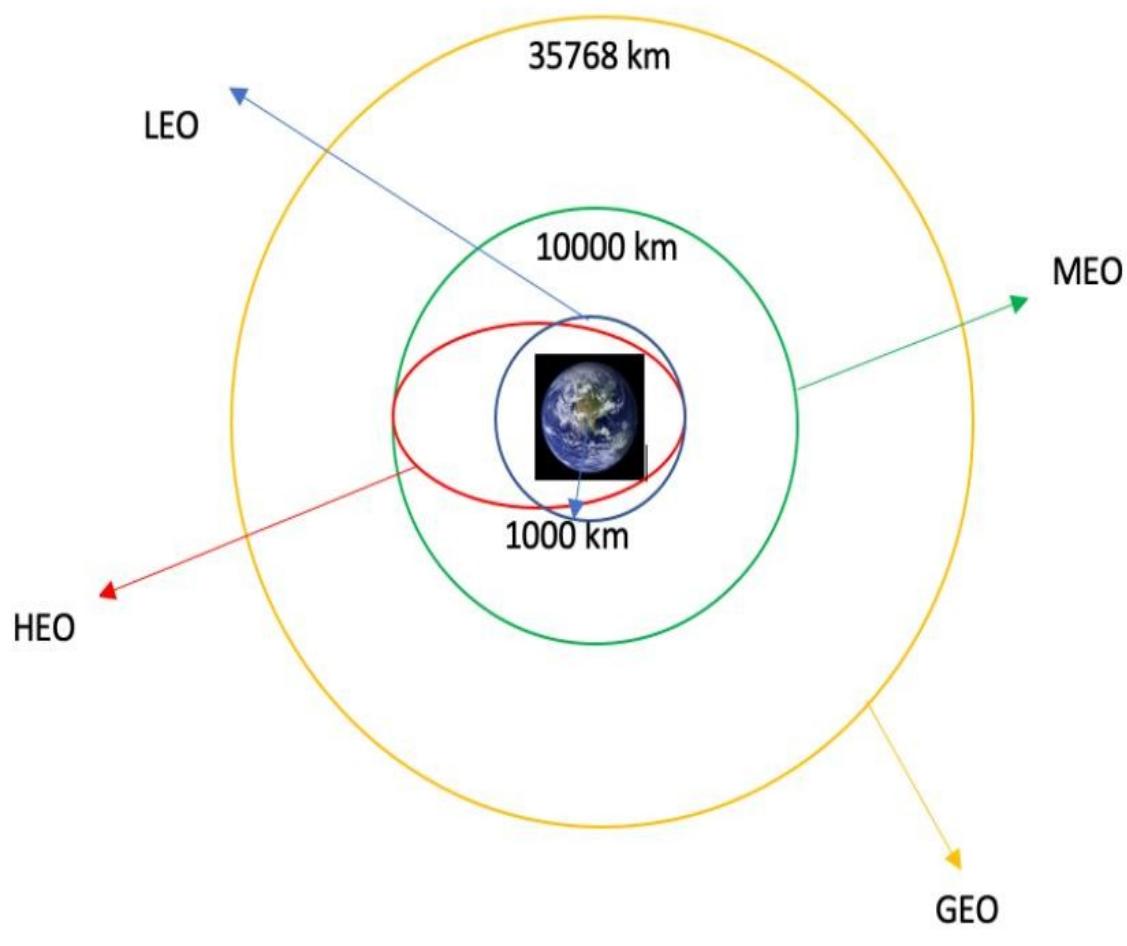
3. Extract all column names from the HTML table header
In [10]: column_names = []

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

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv
```

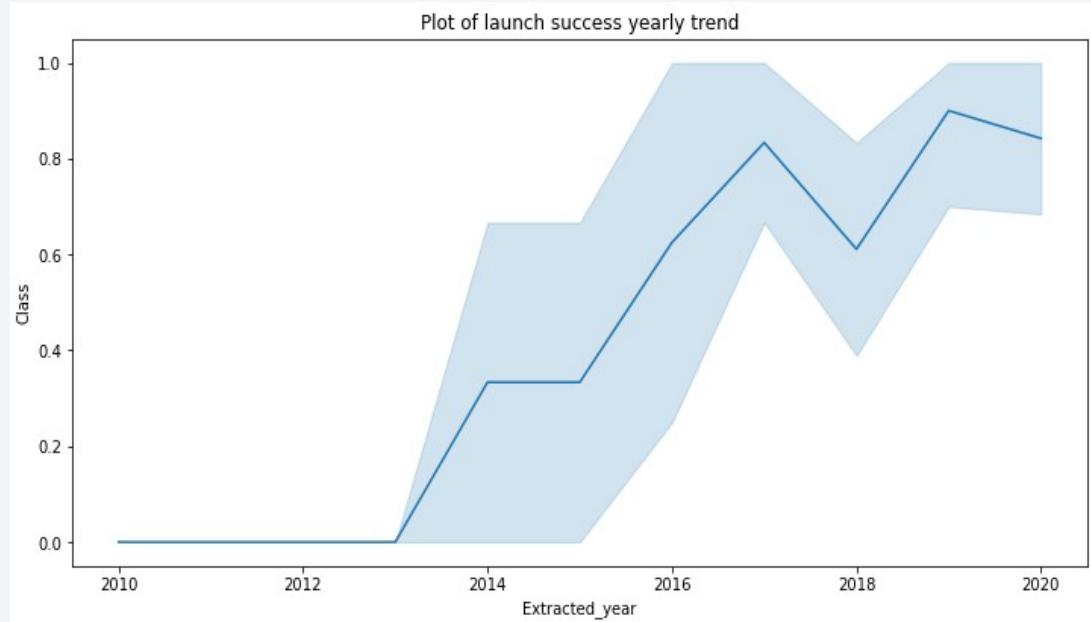
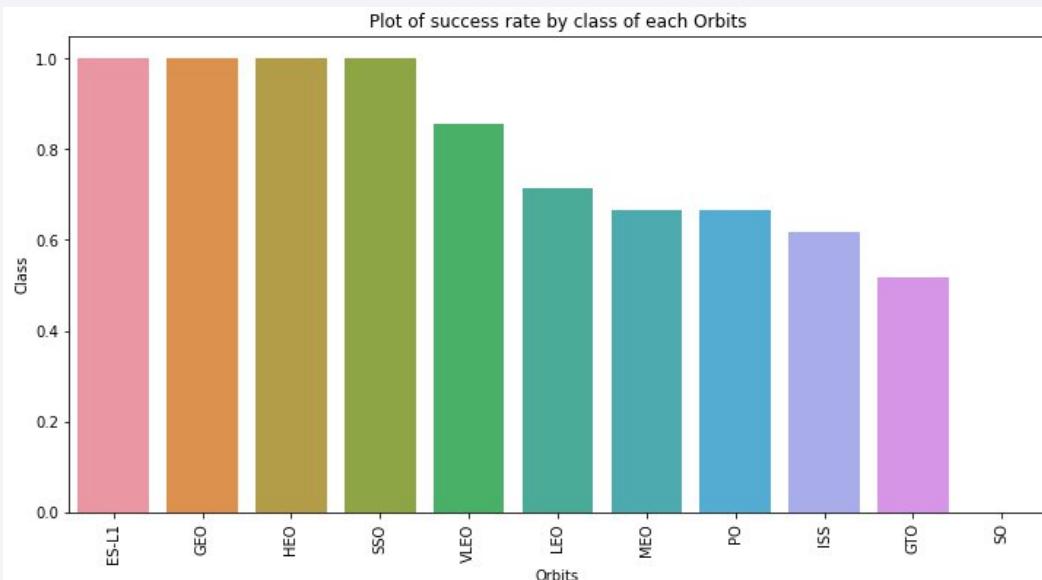
Data Wrangling



- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site, and the number and occurrence of each orbits
- Created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is:
https://github.com/Cyberage173-Creative/Applied-Data-Science-Capstone/blob/main/3_Labs-Jupyter-spacex-DataWrangling.ipynb

EDA with Data Visualization

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



- The link to the notebook is:
https://github.com/Cyberage173-Creative/Applied-Data-Science-Capstone/blob/main/5_EDA_Data_Visualization.ipynb

Build an Interactive Map with Folium

- 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.
- 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.
- 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

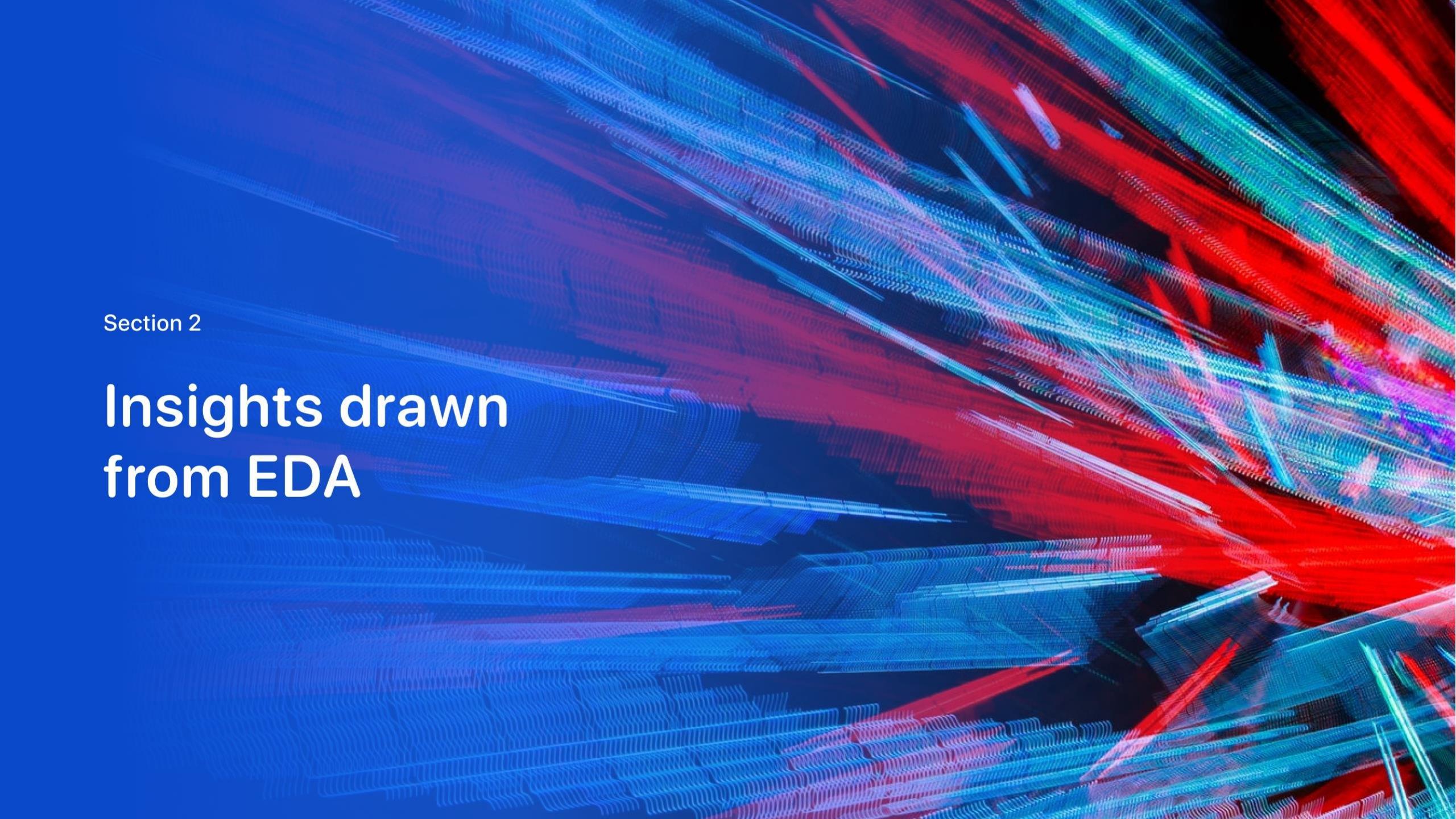
- Built an interactive dashboard with Plotly dash
- Plotted pie charts showing the total launches by a certain sites
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is: https://github.com/Cyberage173-Creative/Applied-Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- Built different machine learning models and tune different hyperparameters using GridSearchCV.
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Found the best performing classification model.
- The link to the notebook is <https://github.com/chuksoo/IBM-Data-Science-Capstone-SpaceX/blob/main/Machine%20Learning%20Prediction.ipynb>

Results

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

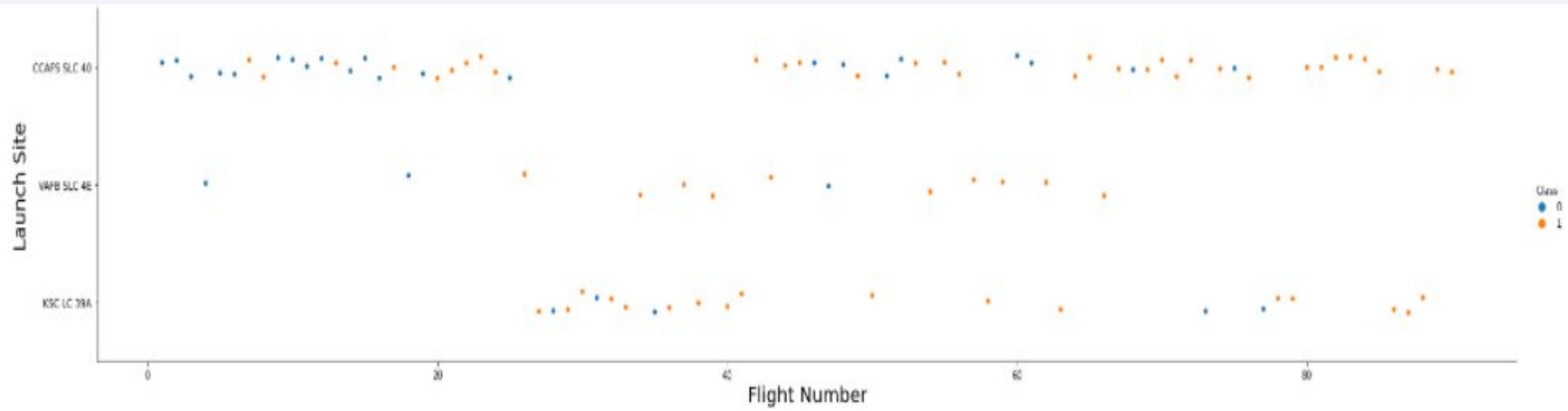
The background of the slide features a complex, abstract pattern of glowing lines. These lines are primarily blue and red, creating a sense of depth and motion. They appear to be composed of numerous small, individual points of light, giving them a granular or digital appearance. The lines curve and twist in various directions, some converging towards the center of the frame while others recede into the distance. The overall effect is one of a dynamic, futuristic, or high-tech environment.

Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

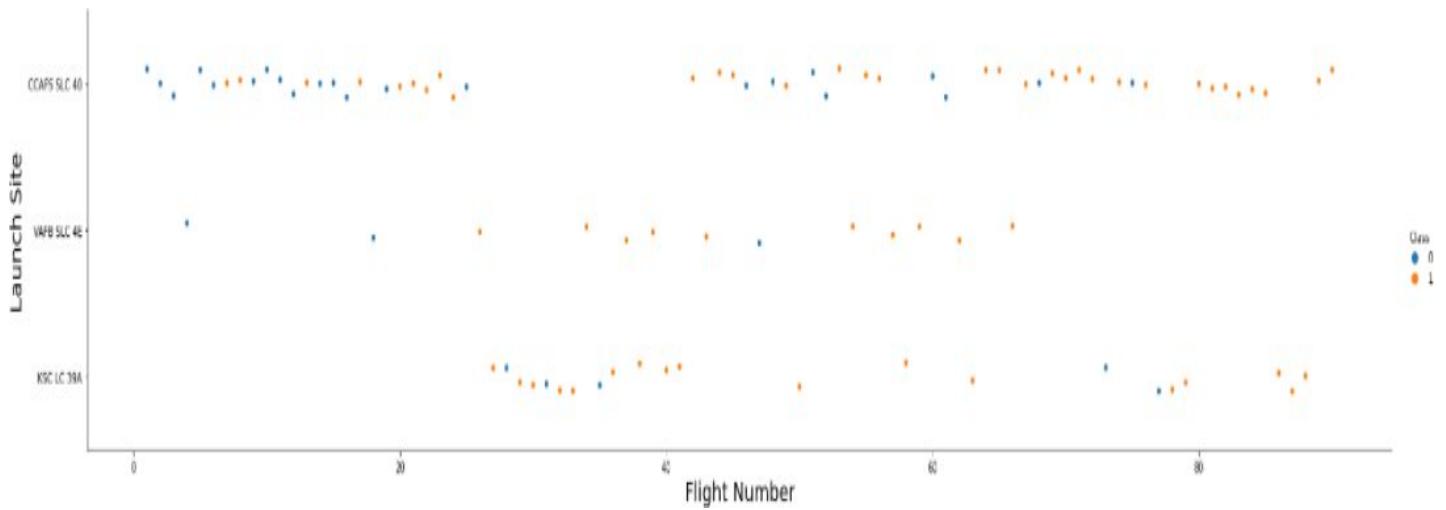
- From the plot, found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



Payload vs. Launch Site

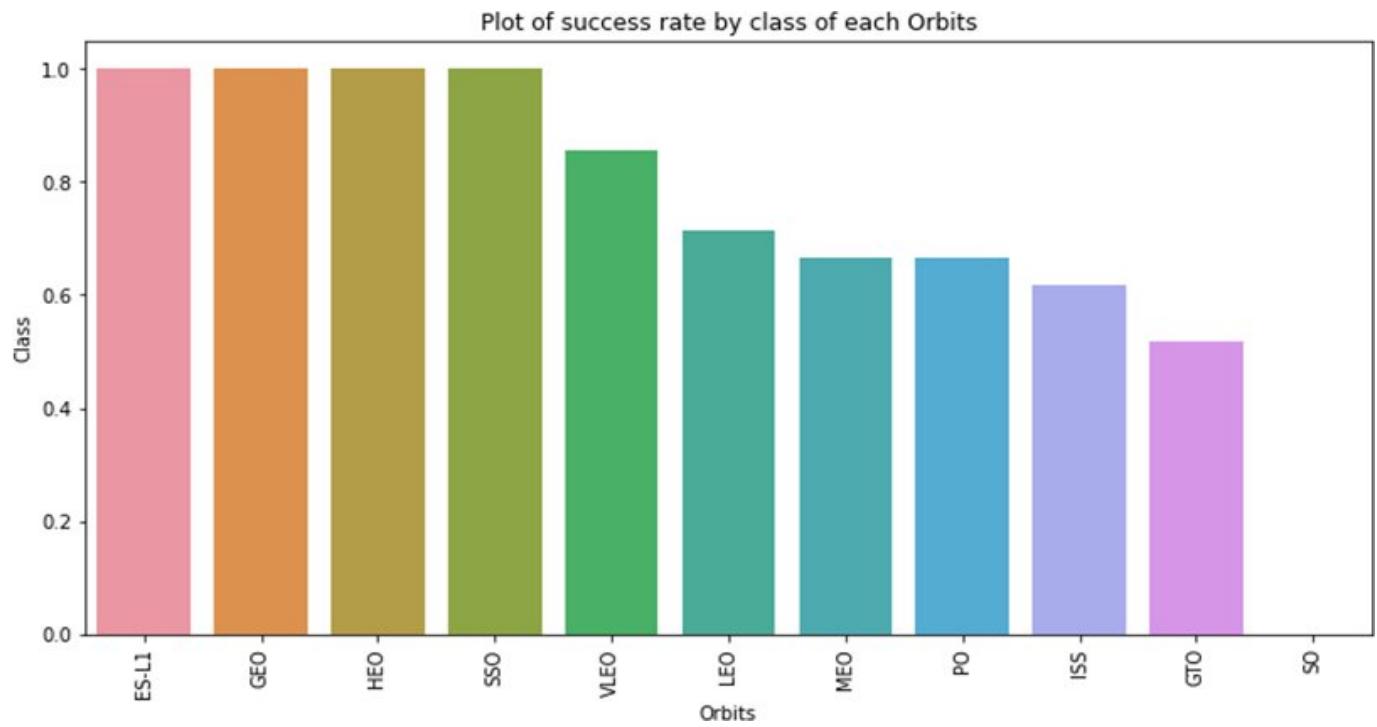


The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.



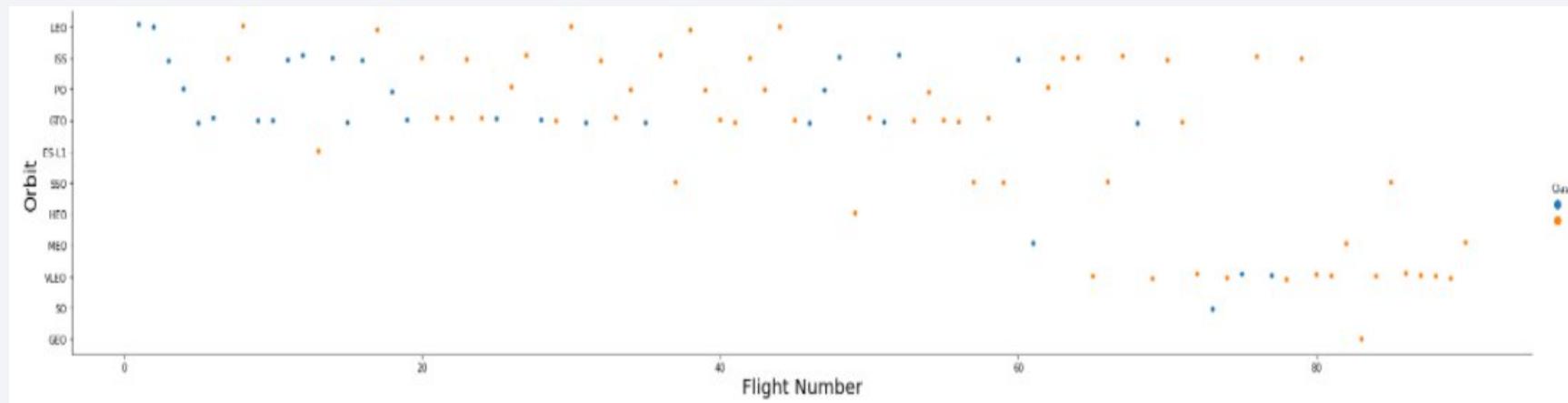
Success Rate vs. Orbit Type

- From the plot, it was evident that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



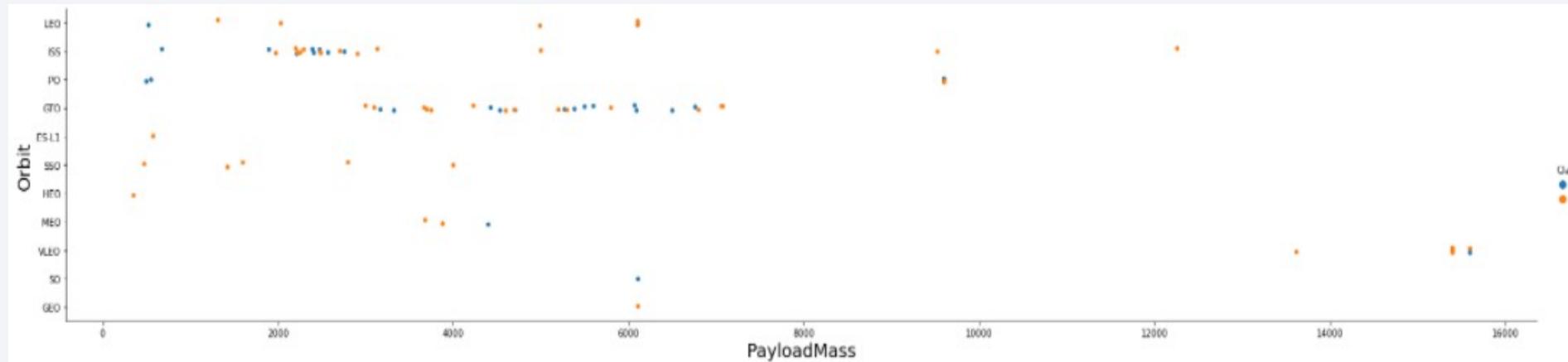
Flight Number vs. Orbit Type

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



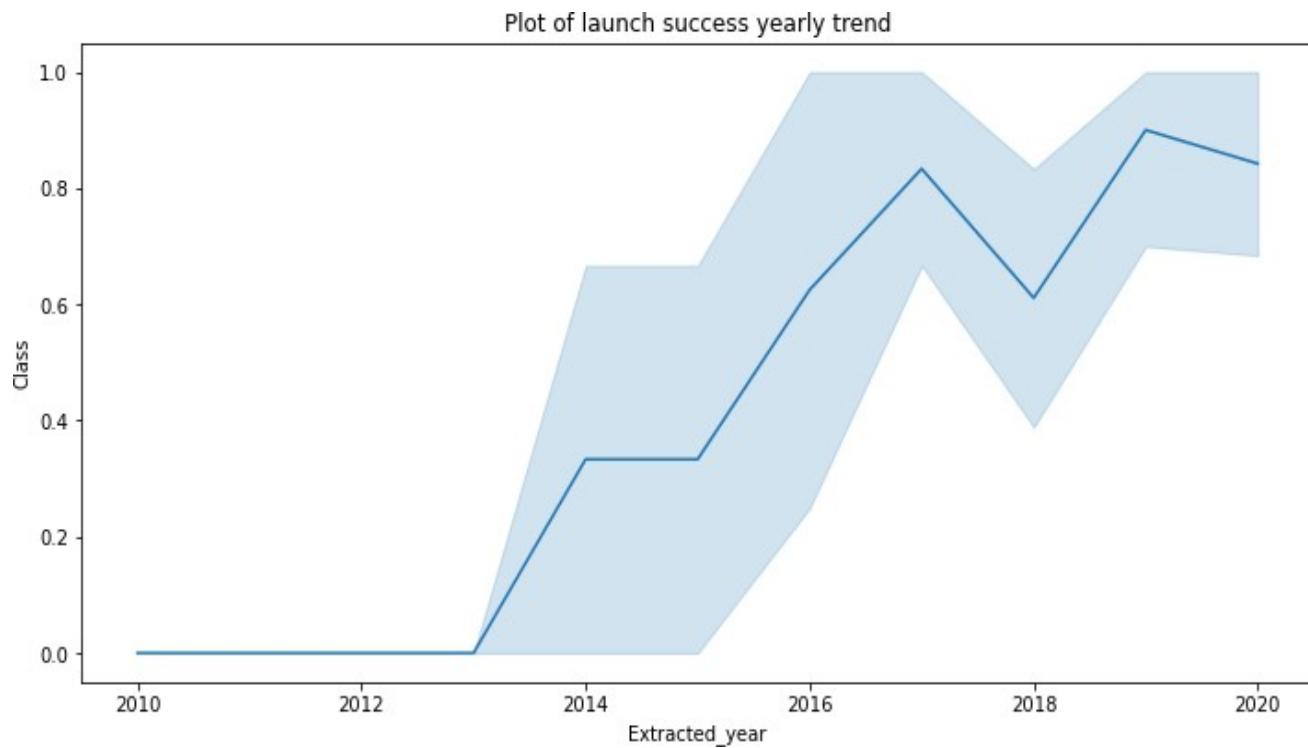
Payload vs. Orbit Type

- It was observed that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From the plot, it was observed that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- 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

- Used the query above to display 5 records where launch sites begin with 'CCA'

Total Payload Mass

- 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

- 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

- Observed that the dates of the first successful landing outcome on ground pad was 22nd 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

- 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

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

Boosters Carried Maximum Payload

- 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

- 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

- 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.
- 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's horizon against a dark blue sky. City lights are visible as numerous small white and yellow dots, primarily concentrated in the lower right quadrant where the United States appears. In the upper right, there are bright green and yellow bands of light, likely the Aurora Borealis or Australis. The overall atmosphere is dark and mysterious.

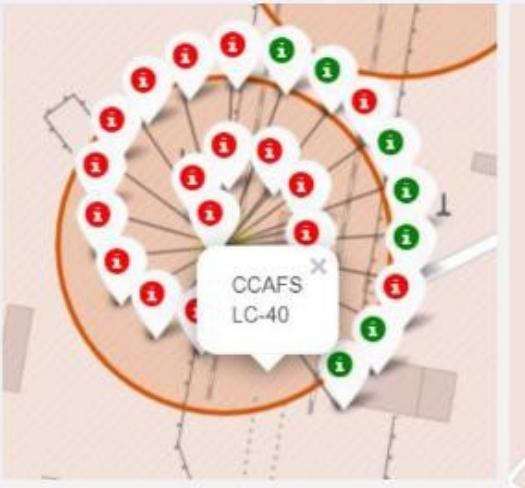
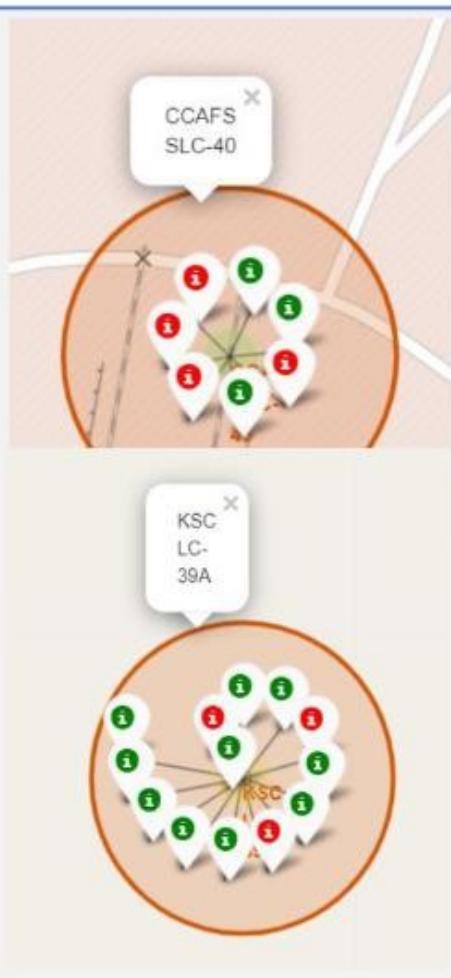
Section 4

Launch Sites Proximities Analysis

All launch sites global map markers



Markers showing launch sites with color labels



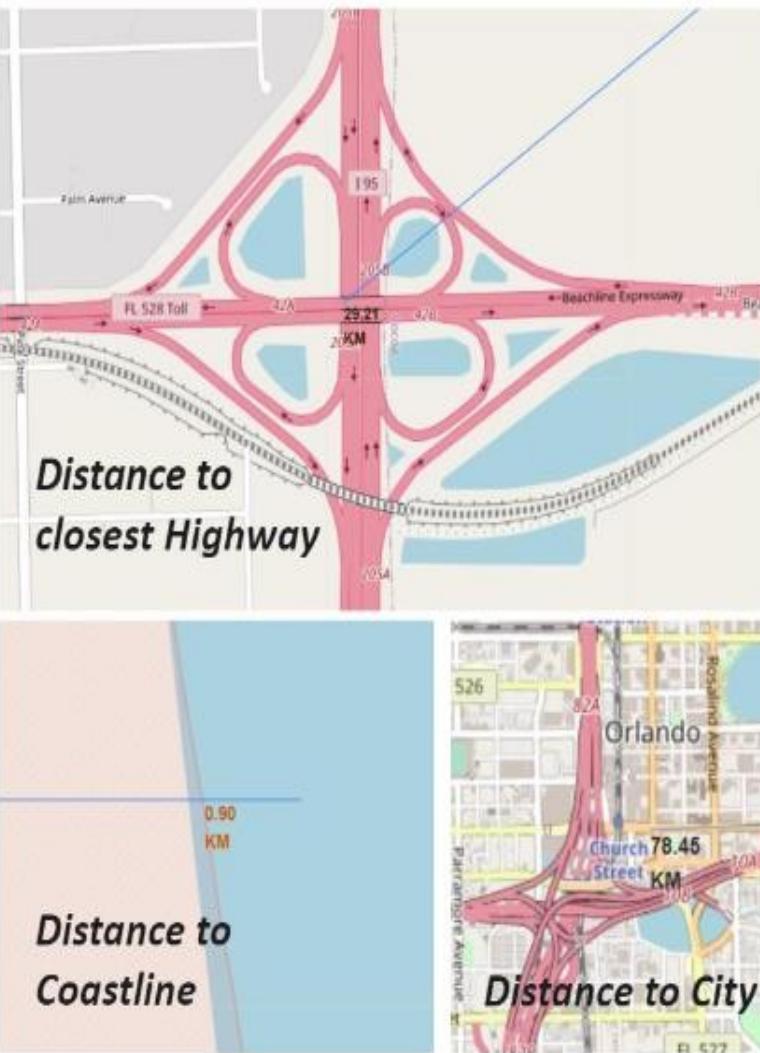
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures



California Launch Site

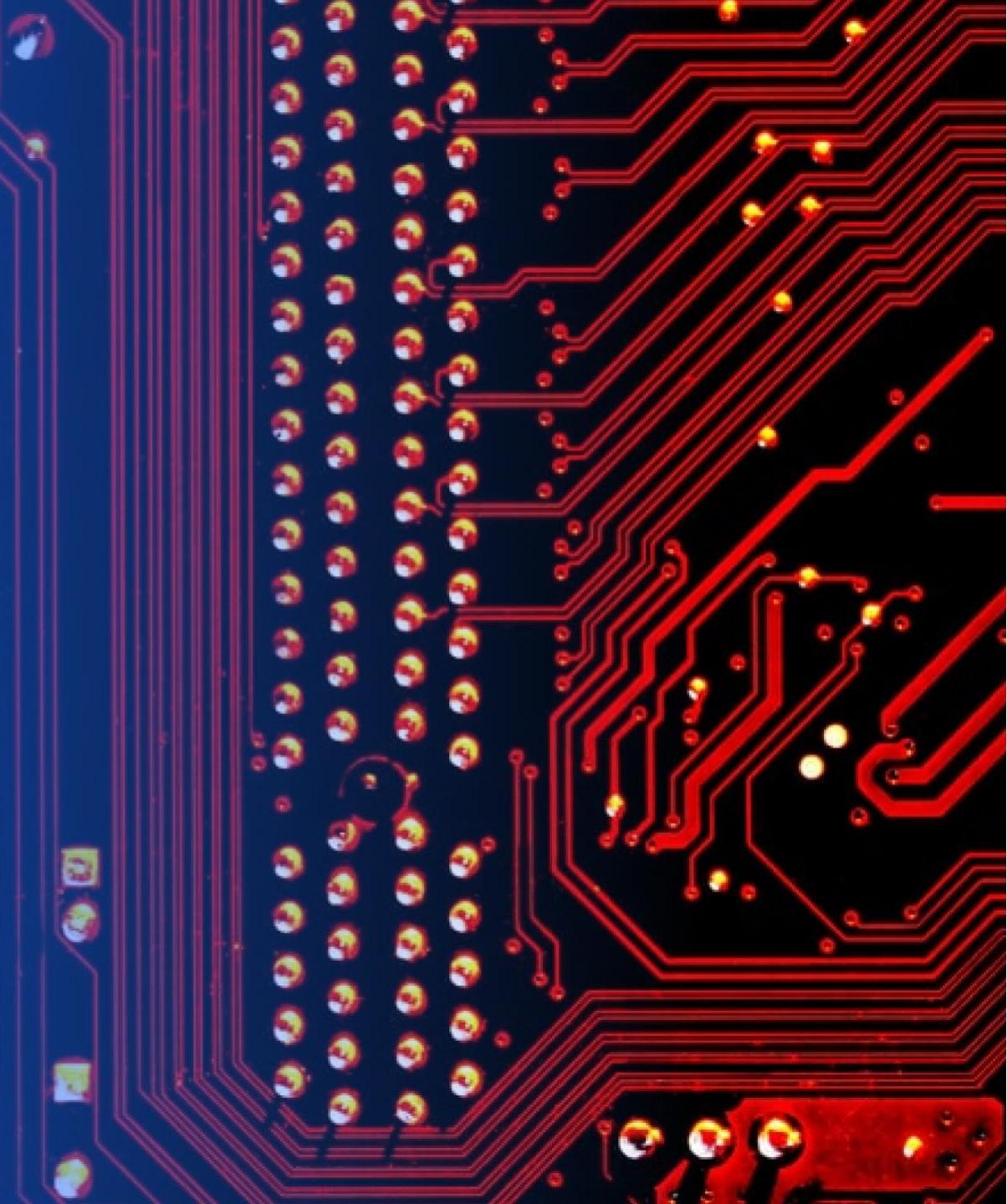
Launch Site distance to landmarks



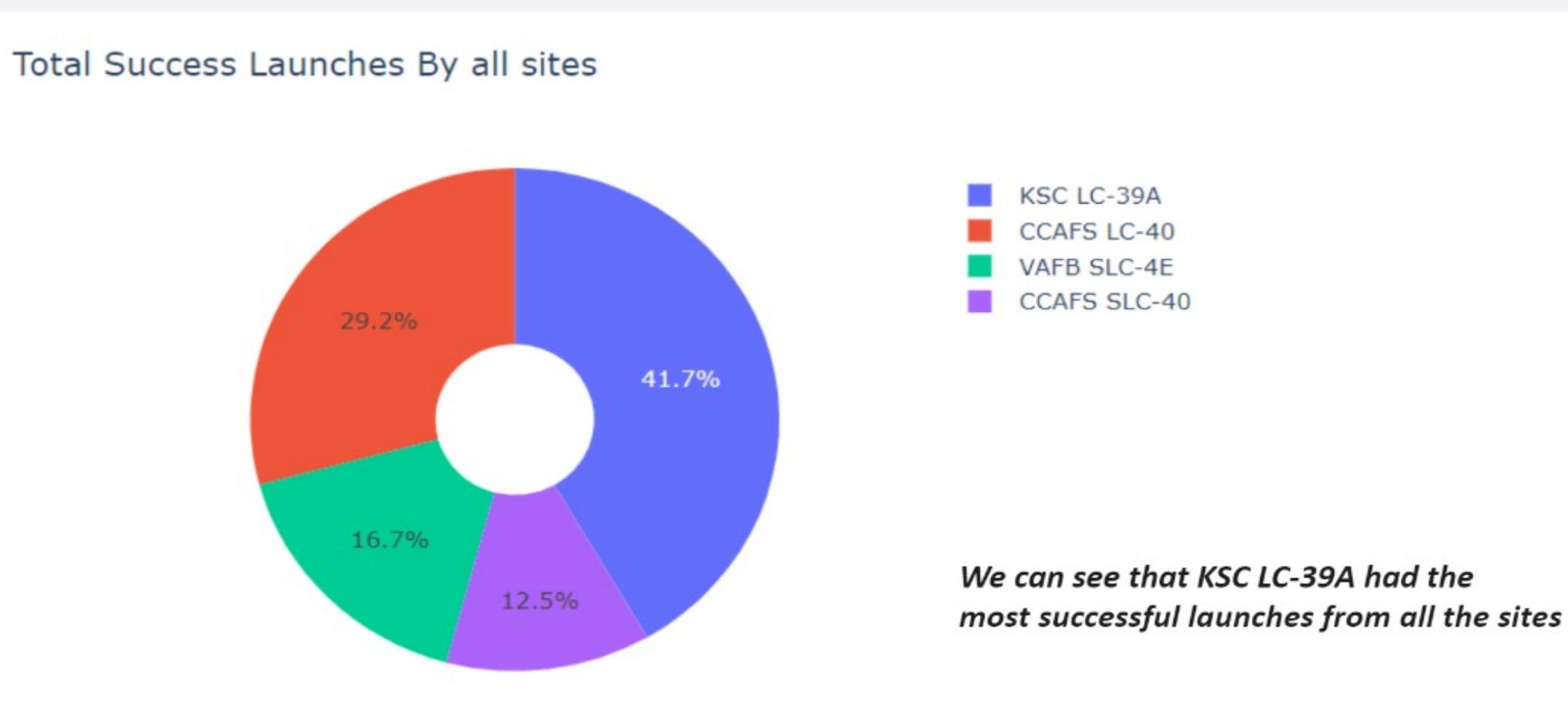
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 5

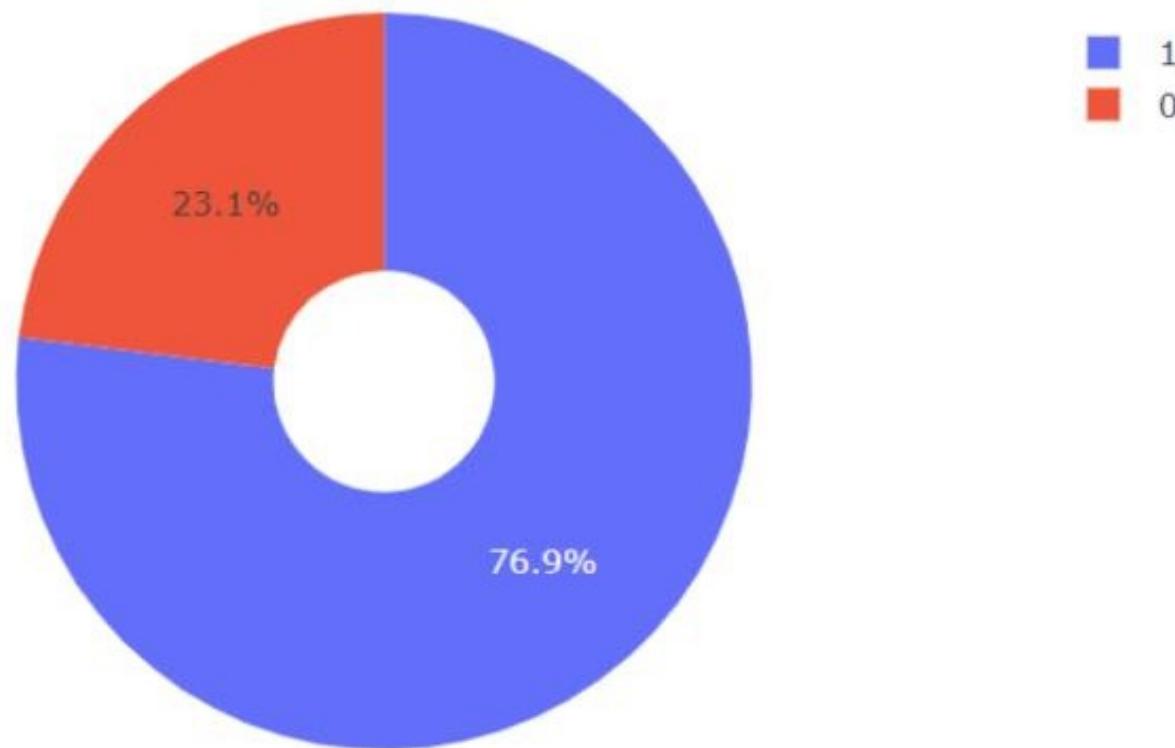
Build a Dashboard with Plotly Dash



Pie chart showing the success percentage achieved by each launch site

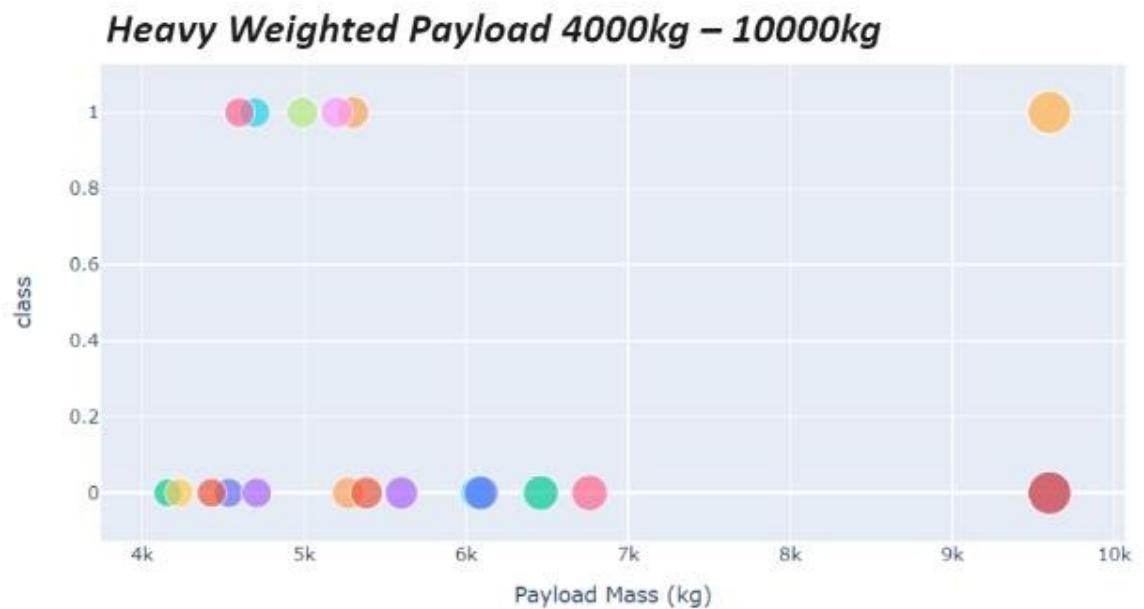
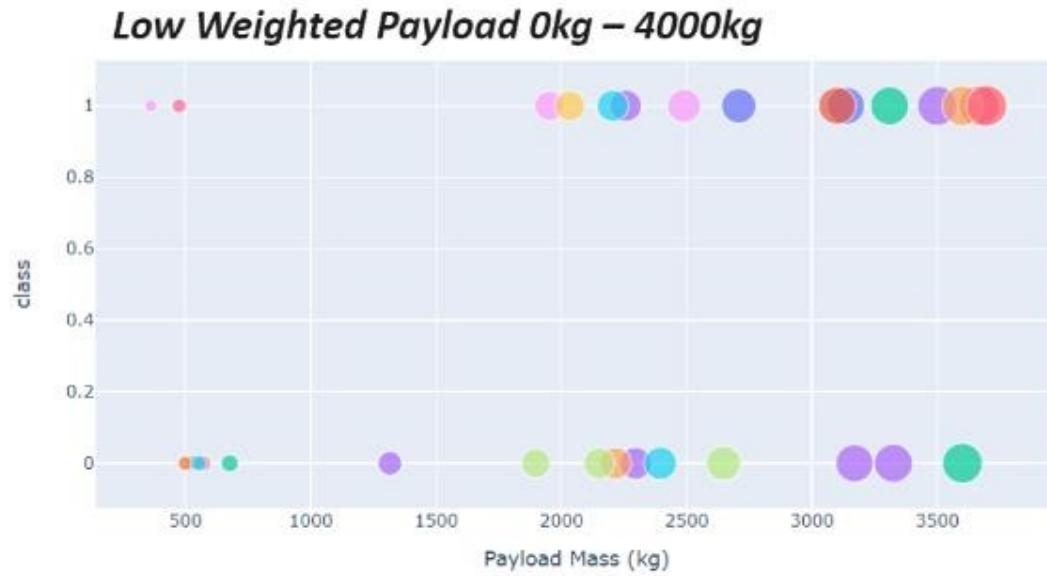


Pie chart showing the Launch site with the highest launch success ratio



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 dynamic, abstract design. It consists of several curved, overlapping bands of color. A prominent band in the center-left is a bright blue, while another band on the right is a warm yellow. These colors transition into lighter shades of blue and yellow towards the edges. The overall effect is one of motion and depth, resembling a tunnel or a stylized landscape.

Section 6

Predictive Analysis (Classification)

Classification Accuracy

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

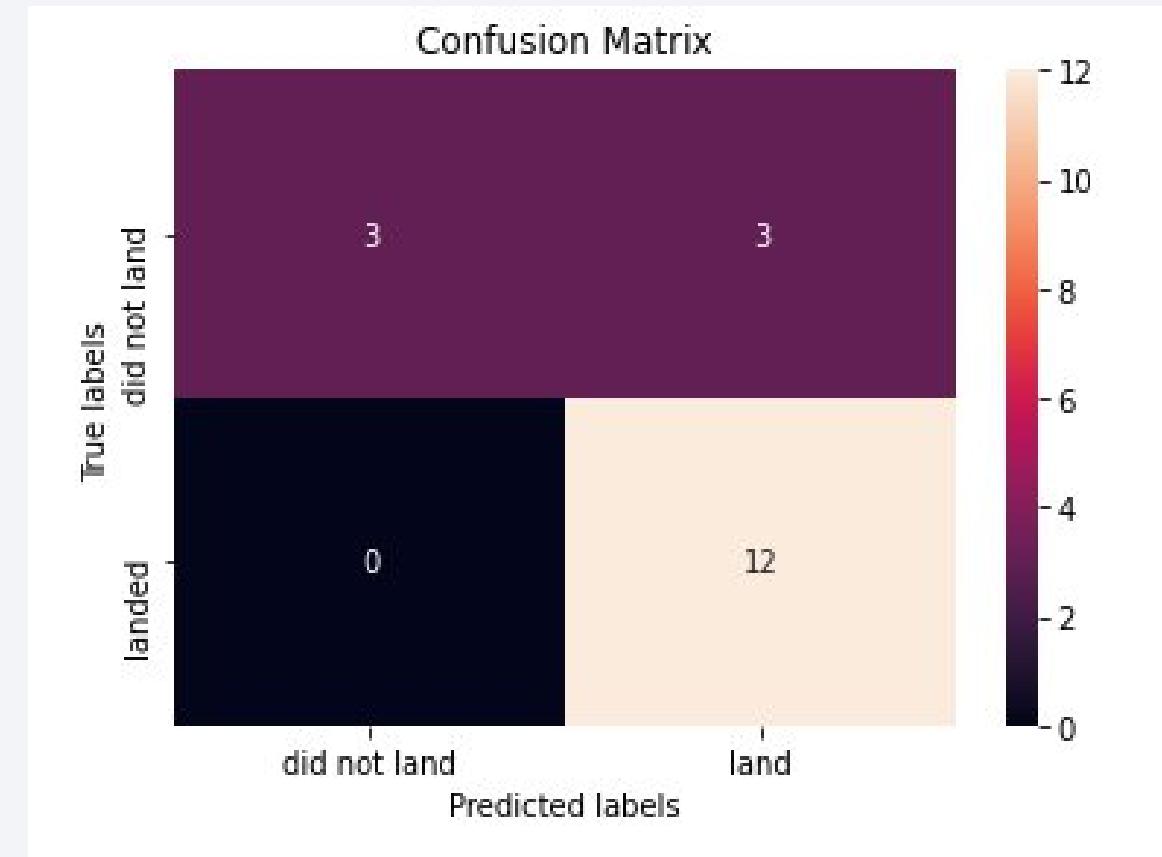
```
models = {'KNeighbors':knn_cv.best_score_,  
          'DecisionTree':tree_cv.best_score_,  
          'LogisticRegression':logreg_cv.best_score_,  
          'SupportVector': svm_cv.best_score_}  
  
bestalgorithm = max(models, key=models.get)  
print('Best model is', bestalgorithm, 'with a score 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.
- KSCLC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!

