

Winning Space Race with Data Science

<Oyewole Jeremiah
Ifeoluwa>
<12/05/2022>



Outline

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

Executive Summary

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

Introduction

- Project background and context

Space X advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

- What factors determine if the rocket will land successfully?
- The interaction amongst various features that determine the success rate of a successful landing.
- What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
 - Describe how data was collected using SPaceX API and Webscraping from Wikipedia
- Perform data wrangling
 - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
 - How to build, tune, evaluate classification models

Data Collection

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

Data Collection – 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://dataplatform.cloud.ibm.com/analytics/notebooks/v2/43a4520d-9359-4e39-82ea-db0b8eb55fa9/view?access_token=67501372f644b6fe85ff2e98defdf4fdeda37efa087d9e466839ea6fbec118f6

The screenshot shows a Jupyter Notebook interface on the dataplatform.cloud.ibm.com platform. The notebook has a title 'Pak for Data' and a subtitle 'thon Basics for Data Science P... / Final Assignment'. The code in cell [5] is for collecting data from the SpaceX API based on cores, while cell [6] is for requesting rocket launch data.

```
In [5]: # Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
        if core['core'] != None:
            response = requests.get("https://api.spacexdata.com/v4/cores/" + core['core']).json()
            Block.append(response['block'])
            ReusedCount.append(response['reuse_count'])
            Serial.append(response['serial'])
        else:
            Block.append(None)
            ReusedCount.append(None)
            Serial.append(None)
        Outcome.append(str(core['landing_success']) + ' ' + str(core['landing_type']))
        Flights.append(core['flight'])
        GridFins.append(core['gridfins'])
        Reused.append(core['reused'])
        Legs.append(core['legs'])
        LandingPad.append(core['landpad'])

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

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

Check the content of the response

In [8]: print(response.content)
```

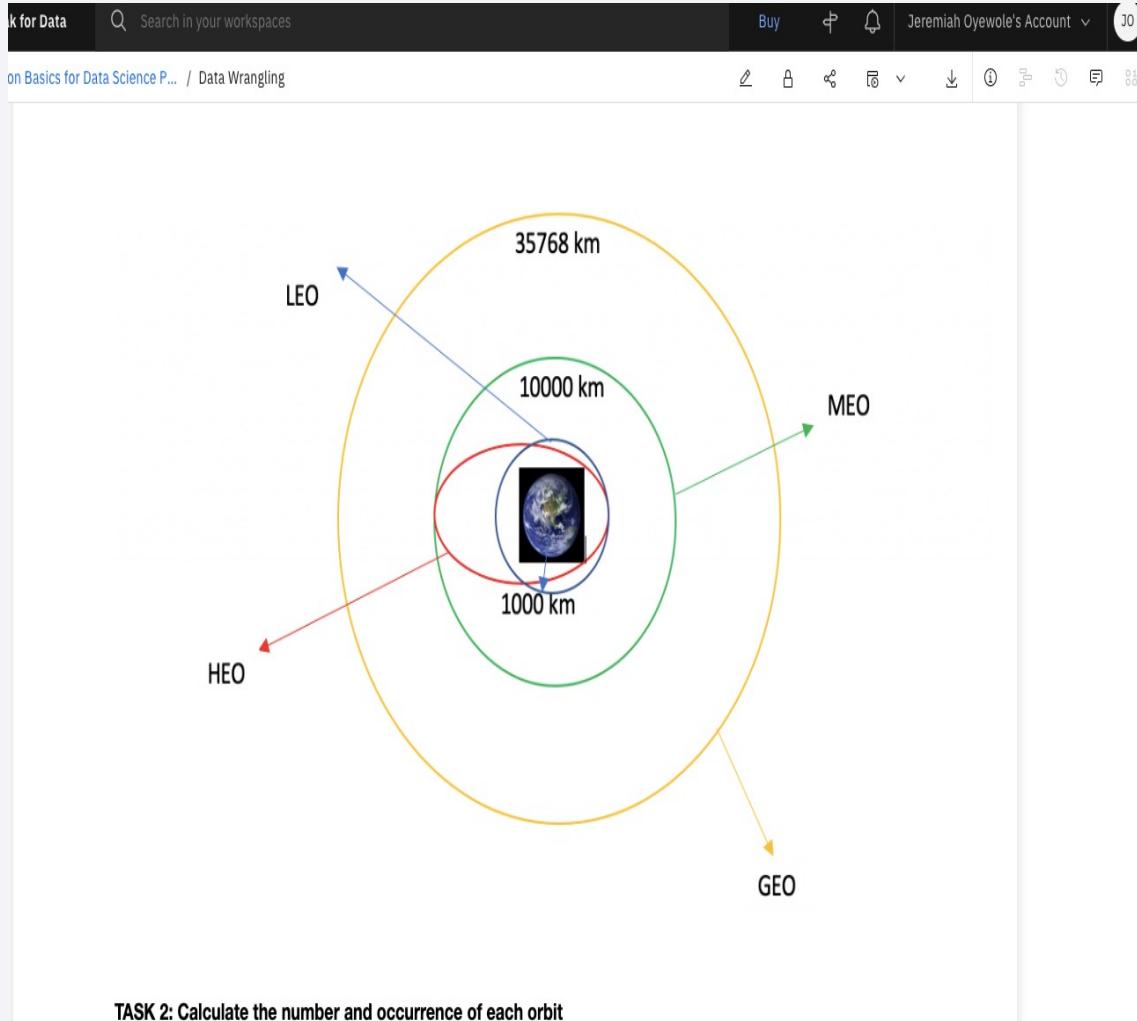
Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/67b4d8ce-aafb-4ca0-b4fd-0e44f745f72b/view?access_token=12652483c66df71d8da1827ca5b84c66305b63779f615f9d0fafdf6b3ecc8f4

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** "Ask for Data" and "Search in your workspaces".
- Header:** "Python Basics for Data Science P..." / "Webscraping Lab".
- Content Area:**
 - A note states: "To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the List of Falcon 9 and Falcon Heavy launches Wikipage updated on 9th June 2021".
 - In [4]: `static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"`
 - A comment: "Next, request the HTML page from the above URL and get a response object".
 - TASK 1: Request the Falcon9 Launch Wiki page from its URL**
 - A note: "First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response."
 - In [5]: `# use requests.get() method with the provided static_url
assign the response to a object
data = requests.get(static_url).text`
 - A comment: "Create a BeautifulSoup object from the HTML response".
 - In [6]: `# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html5lib')`
 - A comment: "Print the page title to verify if the BeautifulSoup object was created properly".
 - In [7]: `# Use soup.title attribute
print(soup.title)`
 - The output: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
 - TASK 2: Extract all column/variable names from the HTML table header**
 - A note: "Next, we want to collect all relevant column names from the HTML table header".
 - A note at the bottom: "Let's try to find all tables on the wiki page first. If you need to refresh your memory about BeautifulSoup, please check the external reference link towards

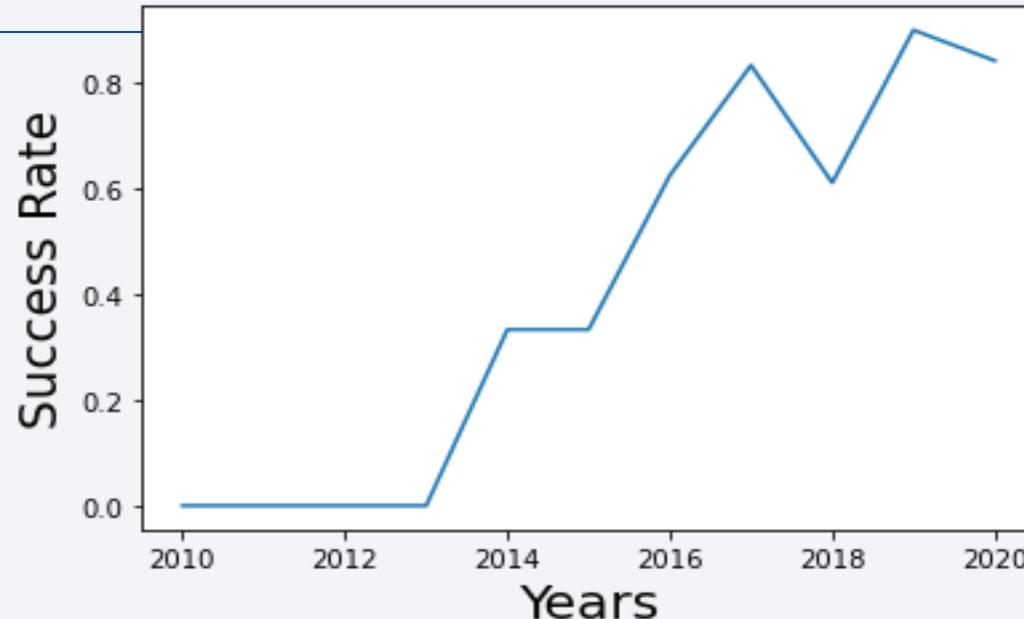
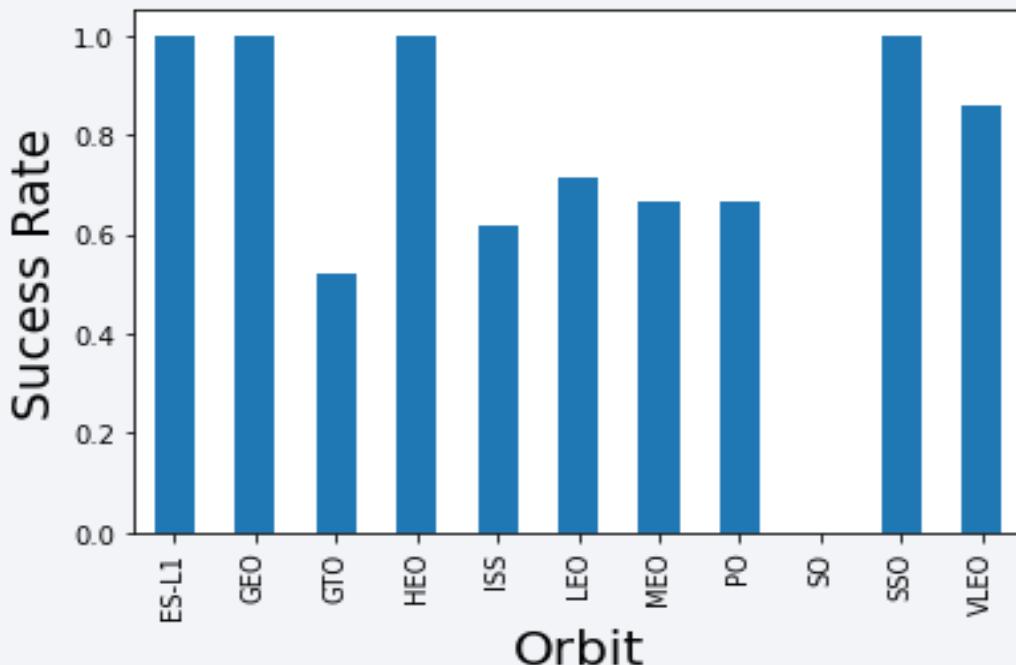
Data Wrangling



- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is
https://dataplatform.cloud.ibm.com/Analytics/notebooks/v2/e4797030-1be8-4d01-920a-2466205433cc/view?access_token=d0014bf6e0373fe2d8d47a36c7a2cc24148016cd2c064fcb6ca776216e167ae9b

EDA with Data Visualization

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



The link to the notebook is:

https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/be59c557-63c4-4655-a952-2c9f01e522fb/view?access_token=45e9c69fb4c92dba8e44a7777be0098f0b0b184cd7844c8d05534a06f513934

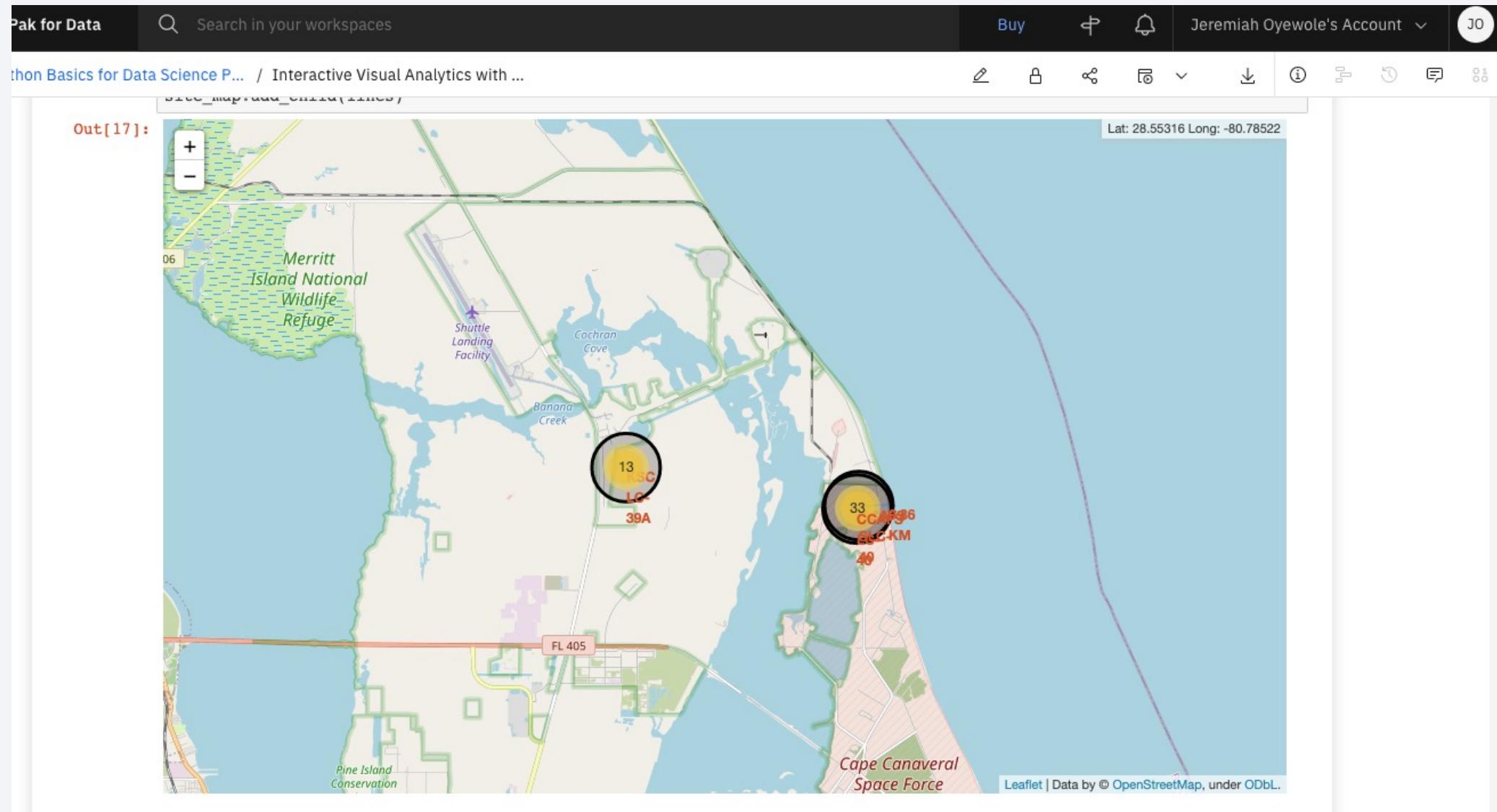
EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
 - The names of unique launch sites in the space mission.
 - The total payload mass carried by boosters launched by NASA (CRS)
 - The average payload mass carried by booster version F9 v1.1
 - The total number of successful and failure mission outcomes
 - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the notebook
is:https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/eca0facf-3d19-4e8e-98b6-c3fb82359f0/view?access_token=900fc45de125756c2892d020a55a103136ff525d261089095863c97b4bc1512c1

Build an Interactive Map with Folium

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

Build an Interactive Map with Folium



Build a Dashboard with Plotly Dash

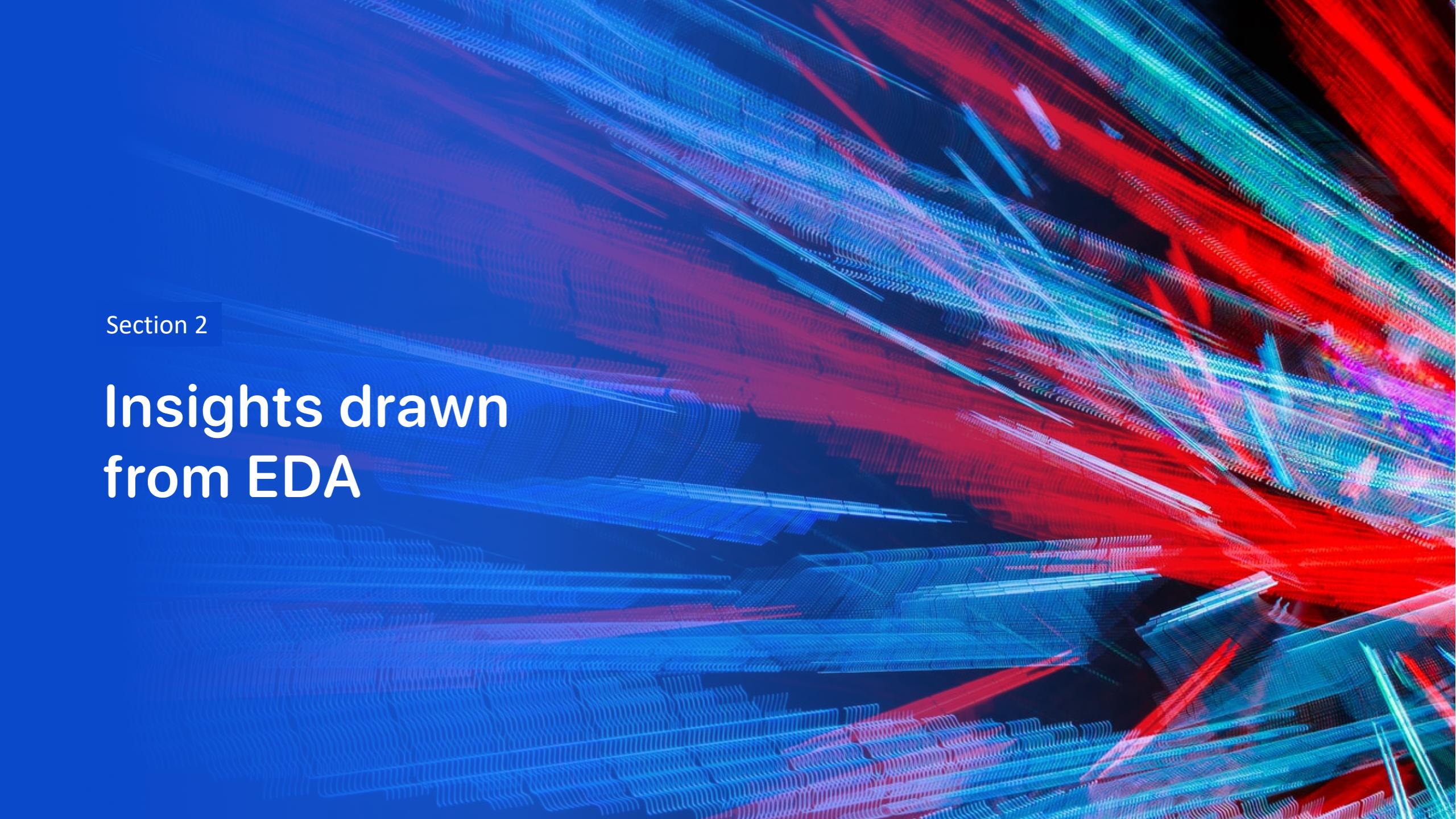
- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is
https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/be59c557-63c4-4655-a952-2c9f01e522fb/view?access_token=45e9c69fbb4c92dba8e44a7777be0098f0b0b184cd7844c8d05534a06f513934

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is
https://dataplatform.cloud.ibm.com/analytics/notebooks/v2/b61628fd-11e9-433a-b0f1-81c1801caf0d/view?access_token=202d1d409ed4965a1c7b2276db4394f18a6791f3eebeb5d788049ce7fef15b19

Results

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

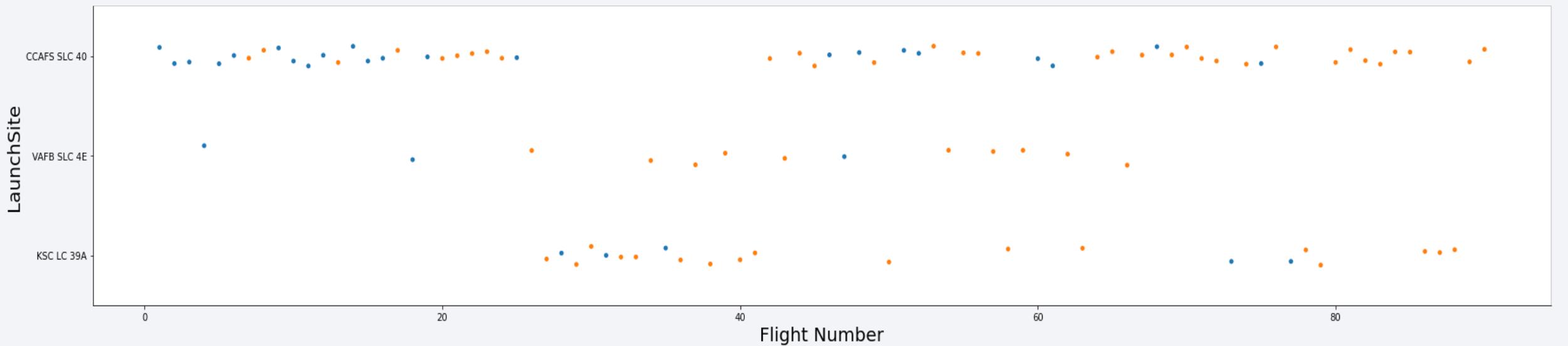
The background of the slide features a dynamic, abstract pattern of glowing particles. The particles are primarily blue and red, creating a sense of motion and depth. They are arranged in several parallel layers that curve upwards from left to right. The intensity of the light varies, with some particles being brighter than others, which adds to the overall visual complexity and depth.

Section 2

Insights drawn from EDA

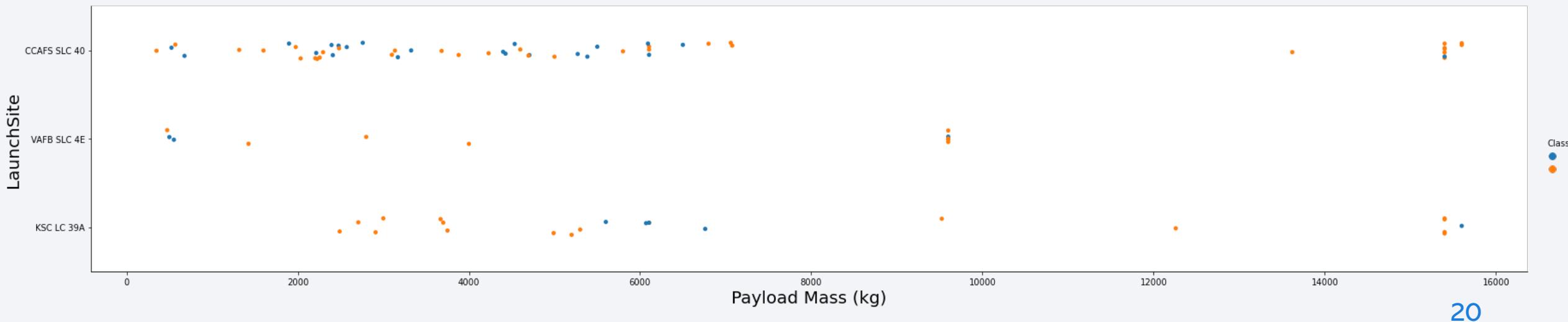
Flight Number vs. Launch Site

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



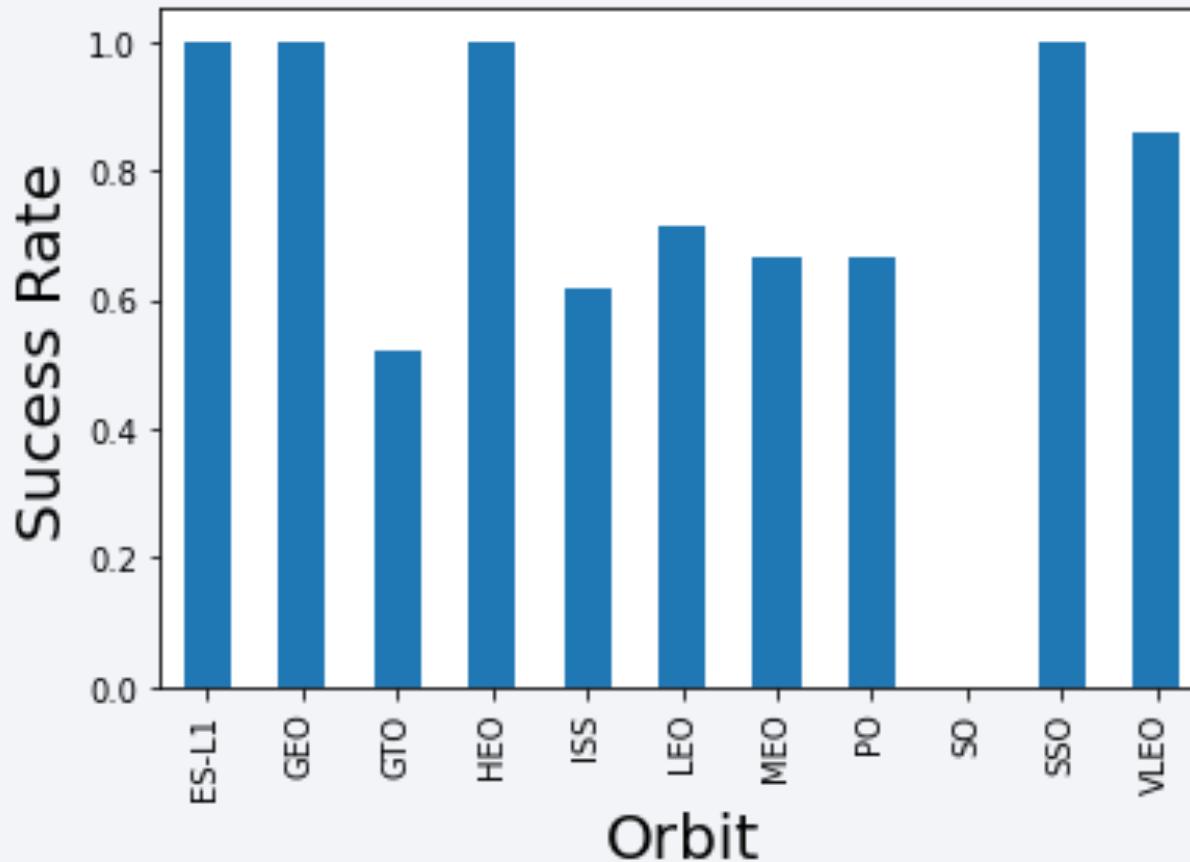
Payload vs. Launch Site

- From the plot, we found that the greater the payload mass for launch site CCAFS SLC 40, the higher the success rate for the rocket.



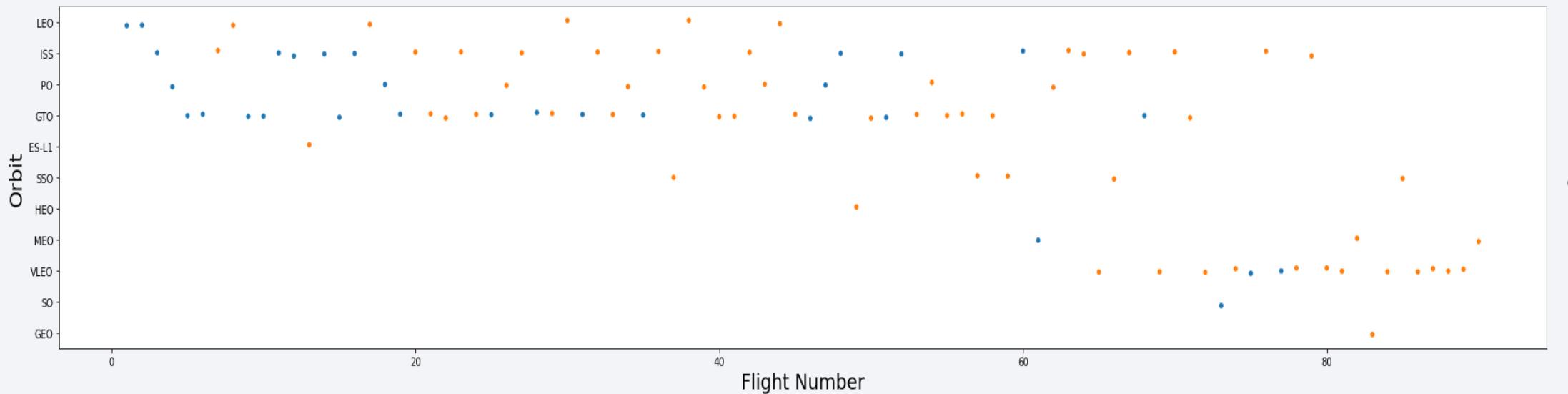
Success Rate vs. Orbit Type

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



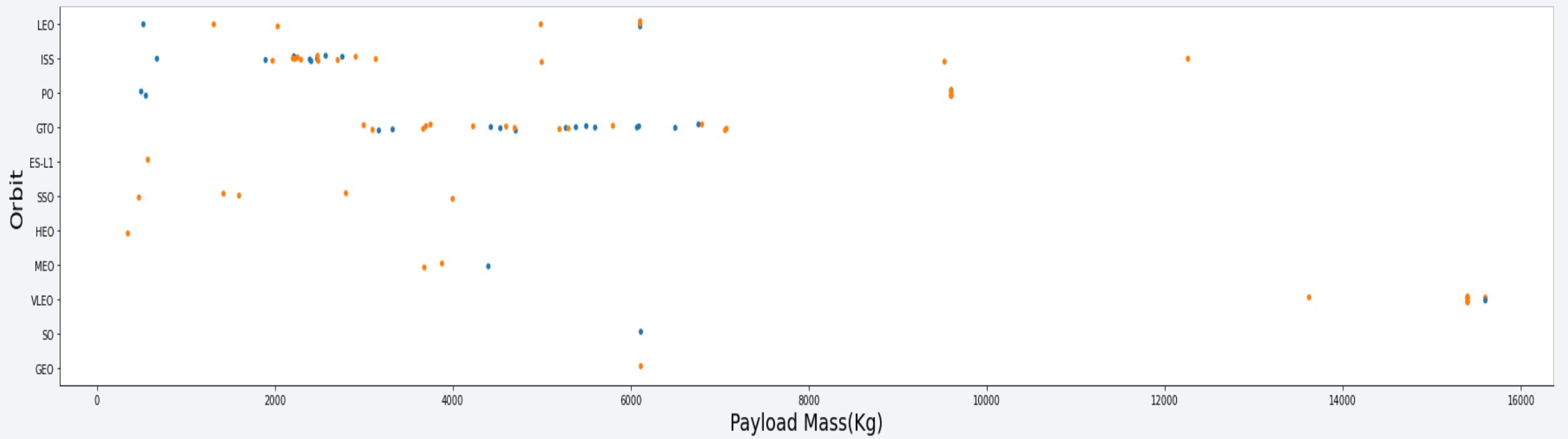
Flight Number vs. Orbit Type

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



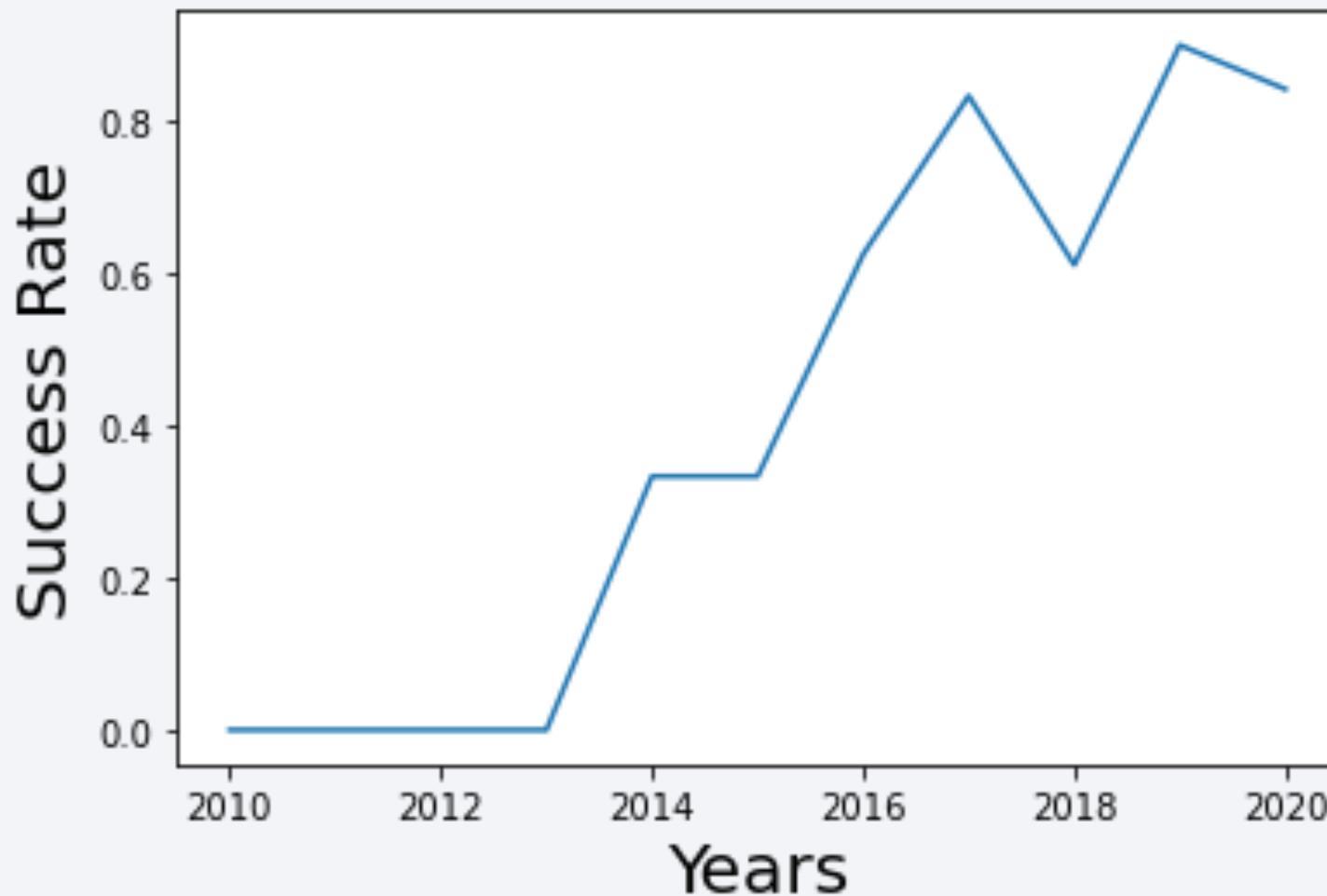
Payload vs. Orbit Type

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



Launch Success Yearly Trend

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



All Launch Site Names

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

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

```
In [10]: task_1 = """
    SELECT DISTINCT LaunchSite
    FROM SpaceX
"""
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

- We used the query below to display 5 records where launch sites 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)

Out[11]:
```

	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

Total Payload Mass

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

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

In [12]:

```
task_3 = """
    SELECT SUM(PayloadMassKG) AS Total_PayloadMass
    FROM SpaceX
    WHERE Customer LIKE 'NASA (CRS)'
    """
create_pandas_df(task_3, database=conn)
```

Out[12]:

total_payloadmass

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

```
Display average payload mass carried by booster version F9 v1.1
In [13]: task_4 = """
            SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
            FROM SpaceX
            WHERE BoosterVersion = 'F9 v1.1'
            """
create_pandas_df(task_4, database=conn)

Out[13]: avg_payloadmass
          0      2928.4
```

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 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

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

```
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
```

Total Number of Successful and Failure Mission Outcomes

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

```
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:
failureoutcome
_____
0      1
```

Boosters Carried Maximum Payload

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

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
In [17]: task_8 = """
    SELECT BoosterVersion, PayloadMassKG
    FROM SpaceX
    WHERE PayloadMassKG = (
        SELECT MAX(PayloadMassKG)
        FROM SpaceX
    )
    ORDER BY BoosterVersion
"""
create_pandas_df(task_8, database=conn)
```

Out[17]:

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

2015 Launch Records

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

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

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

```
Out[18]:   boosterversion  launchsite  landingoutcome
0      F9 v1.1 B1012  CCAFS LC-40  Failure (drone ship)
1      F9 v1.1 B1015  CCAFS LC-40  Failure (drone ship)
```

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

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

```
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

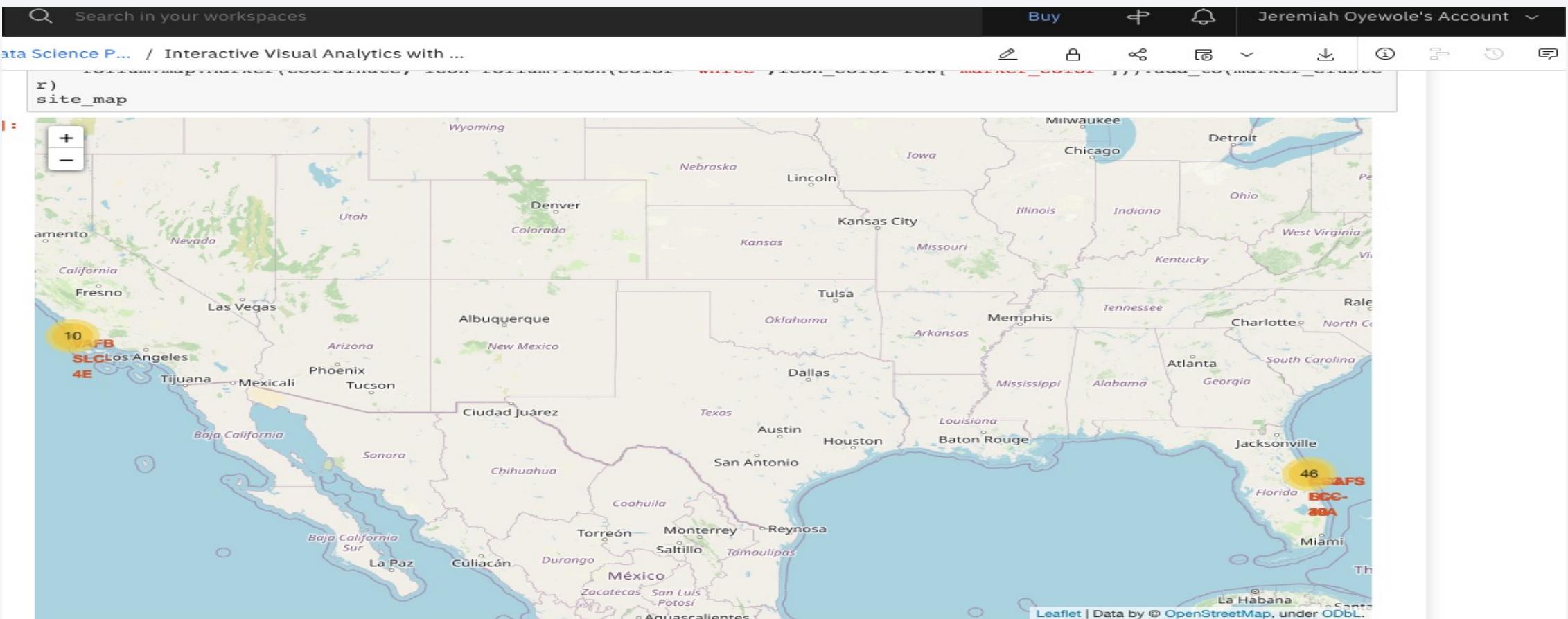
The background of the slide is a photograph taken from space at night. It shows the curvature of the Earth against a dark blue and black void of space. City lights are visible as small white dots and larger clusters of light, primarily concentrated in the lower right quadrant where the United States appears. In the upper right, there are bright green and yellow bands of the Aurora Borealis (Northern Lights) dancing across the sky.

Section 3

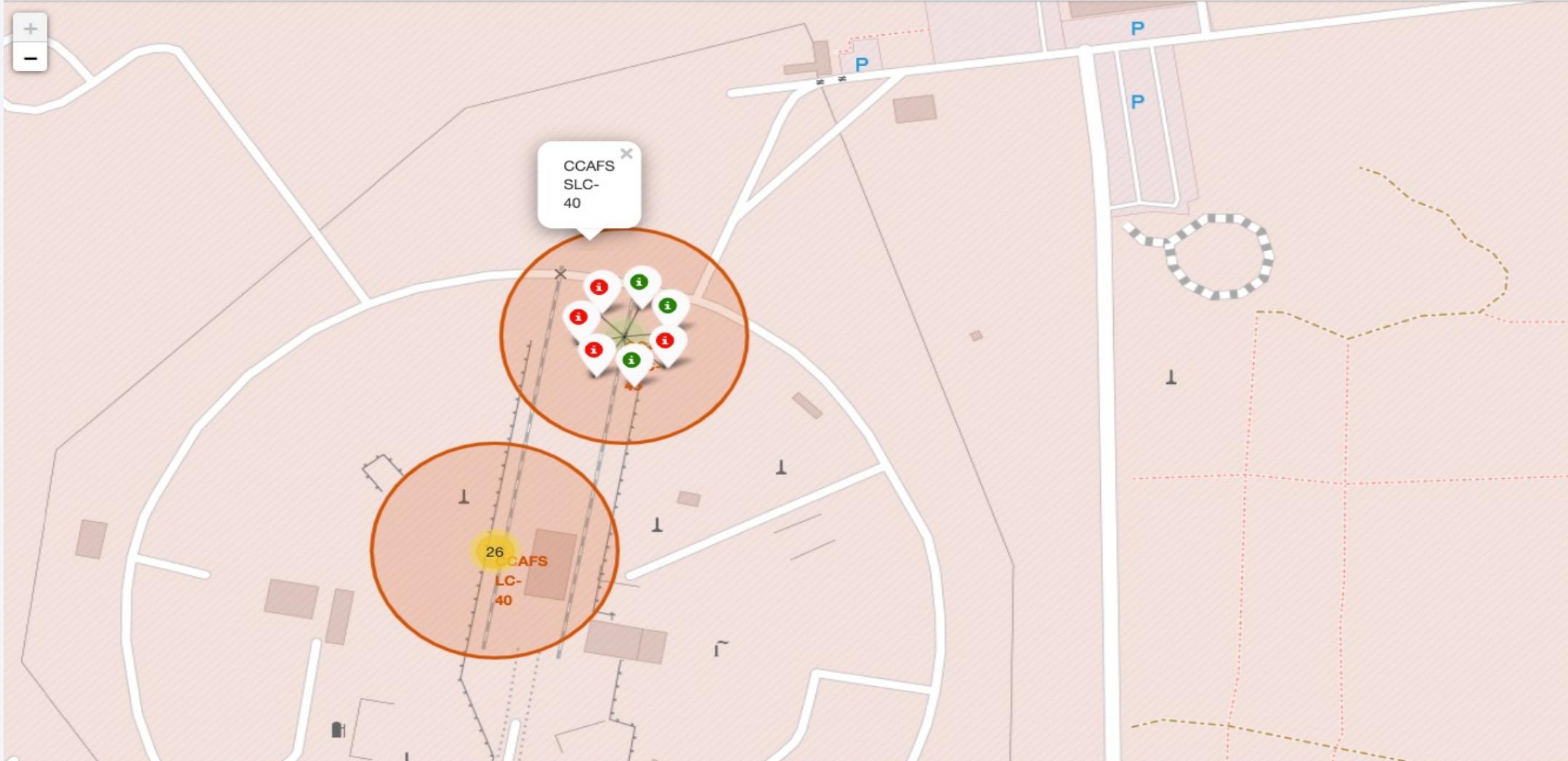
Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

- It is observed that the SpaceX launch site are situated with USA off the coasts of Florida and California



<Folium Map Screenshot 2>



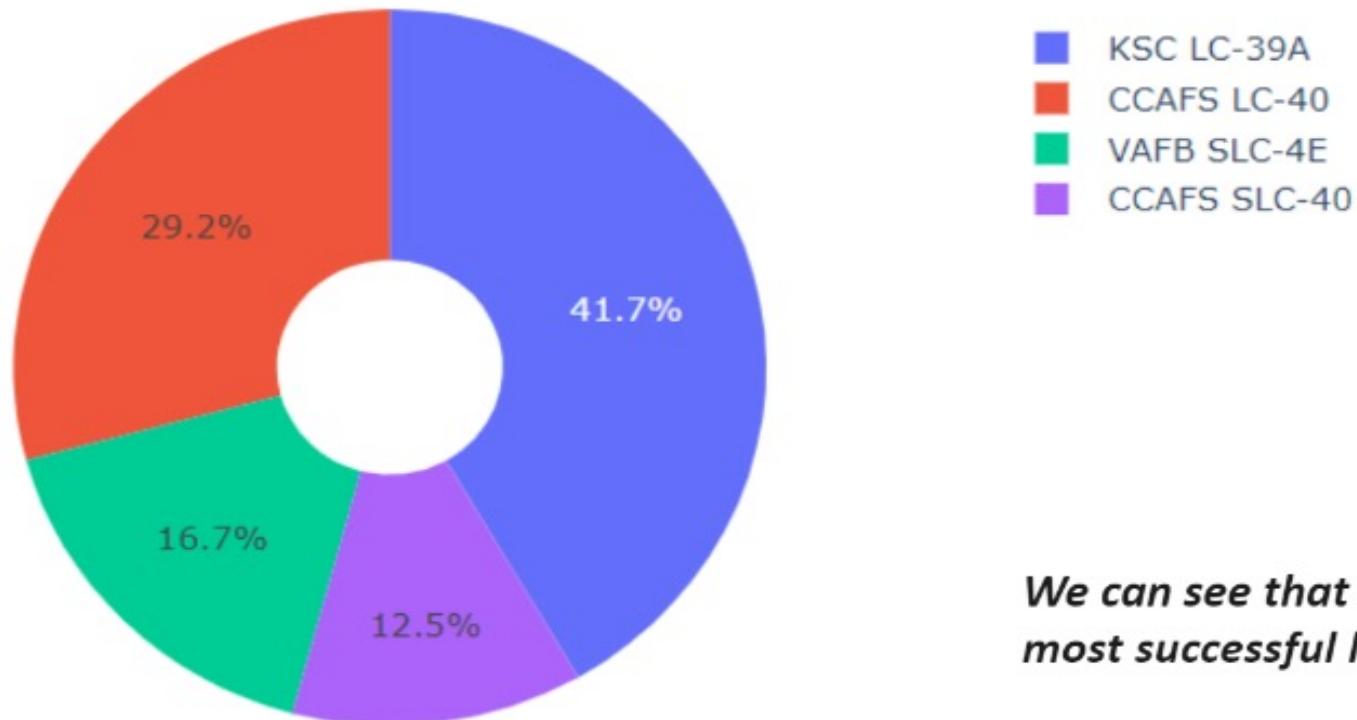
Section 4

Build a Dashboard with Plotly Dash



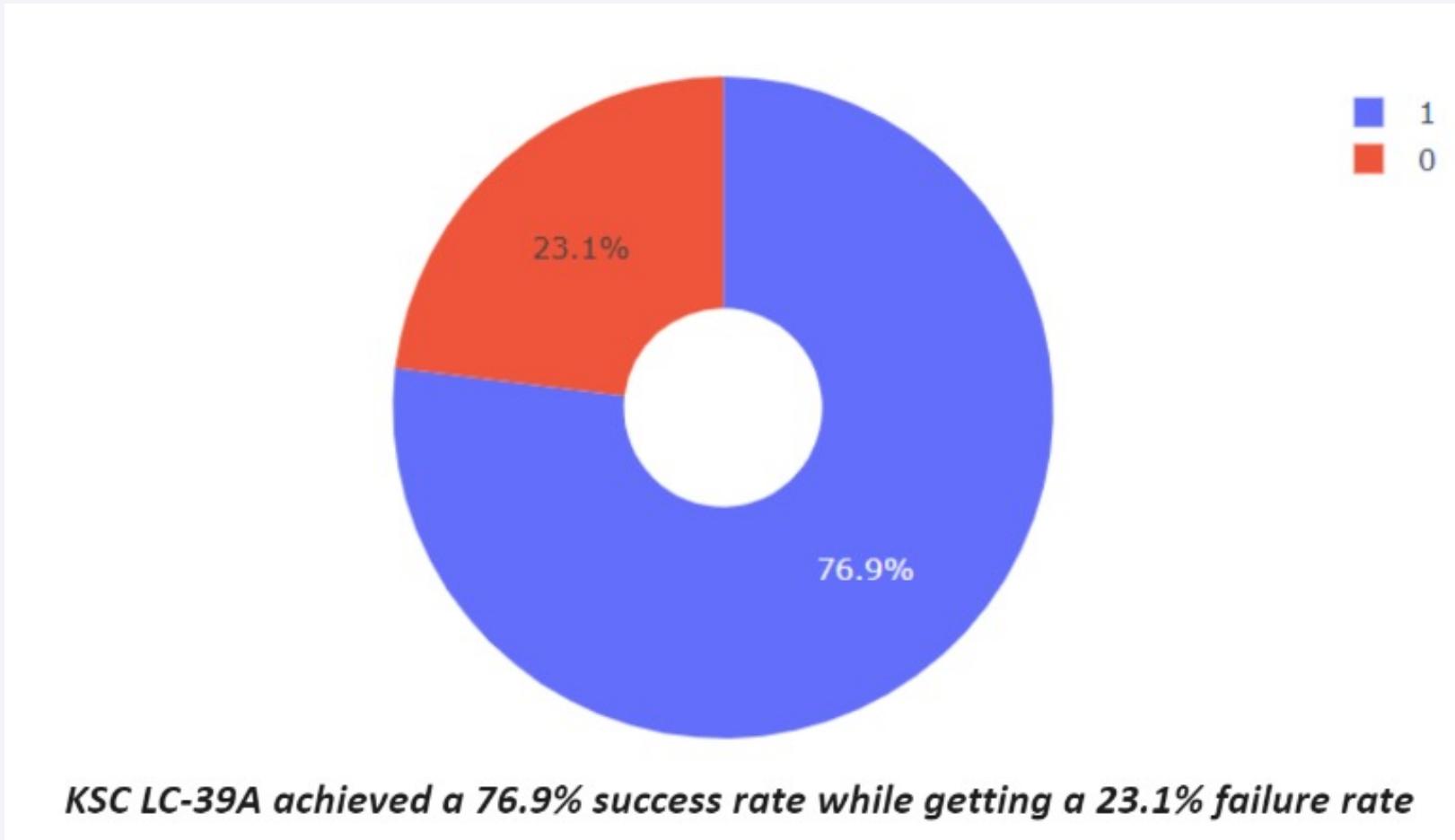
<Dashboard Screenshot 1>

Total Success Launches By all sites

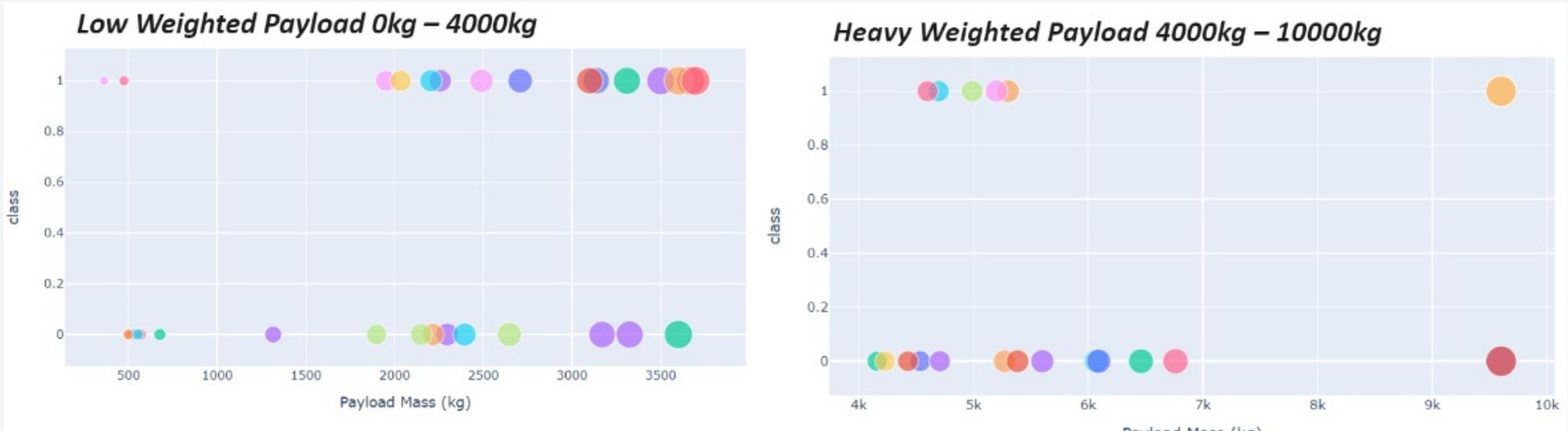


We can see that KSC LC-39A had the most successful launches from all the sites

Pie Chart Showing The Launch Site With The Highest Launch Success Ratio



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 thick, curved lines that transition from a bright yellow at the top right to a deep blue at the bottom left. These lines create a sense of motion and depth, resembling a tunnel or a stylized road. The overall effect is modern and professional.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

The screenshot shows a Jupyter Notebook interface with the following components:

- Header:** Data, Search in your workspaces, Buy, Jeremiah Oyewole's...
- Breadcrumbs:** sics for Data Science P... / Machine Learning Prediction Lab
- Toolbar:** Edit, Lock, Share, Run, Cell, Download, Help
- Section Header:** **TASK 6**
- Text:** Create a support vector machine object then create a GridSearchCV object `svm_cv` with `cv = 10`. Fit the object to find the best parameters from the dictionary `parameters`.
- Code Snippets:**
 - In [15]:

```
parameters = {'kernel':('linear', 'rbf','poly','rbf', 'sigmoid'),
              'C': np.logspace(-3, 3, 5),
              'gamma':np.logspace(-3, 3, 5)}
svm = SVC()
```
 - In [16]:

```
grid_search = GridSearchCV(svm, parameters, cv=10)
svm_cv = grid_search.fit(X_train, Y_train)
```
 - In [17]:

```
print("tuned hpyerparameters :(best parameters) ",svm_cv.best_params_)
print("accuracy :",svm_cv.best_score_)

tuned hpyerparameters :(best parameters)  {'C': 1.0, 'gamma': 0.03162277660168379, 'kernel': 'sigmoid'}
accuracy : 0.8482142857142856
```
- Section Header:** **TASK 7**
- Text:** Calculate the accuracy on the test data using the method `score`:
- Code Snippets:**
 - In [18]:

```
svm_cv.score(X_test, Y_test)
```
 - Out[18]:

```
0.8333333333333334
```
- Text:** We can plot the confusion matrix
- Code Snippet:**

```
In [19]: yhat=svm_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Confusion Matrix

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



Conclusions

We can conclude that:

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

Thank you!

