IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

VISHNU SASIDHARAN
28-09-2024

# Outline

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

# Executive Summary

- Summary of Methodologies

  - SpaceX Data Collection using SpaceX API
  - SpaceX Data Collection with Web Scraping
  - SpaceX Data Wrangling
  - SpaceX Exploratory Data Analysis using SQL
  - SpaceX EDA Data Viz Using Python Pandas and Matplotlib
  - SpaceX Launch Sites Analysis with Folium-Interactive Visual Analytics and Plotly Dash
  - SpaceX Landing Prediction using Machine Learning Models.

- Summary of all results

  - EDA results
  - Interactive Visual Analytics and Dashboards
  - Predictive Analysis using Classification Models.

# Introduction


- Project background and context

SpaceX 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 SpaceX 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 SpaceX for a rocket launch.

- Problems you want to find answers

In this capstone, we will predict if the Falcon 9 first stage will land successfully using data from Falcon 9 rocket launches advertised on its website.

Section 1

# Methodology

# Methodology

## Executive Summary

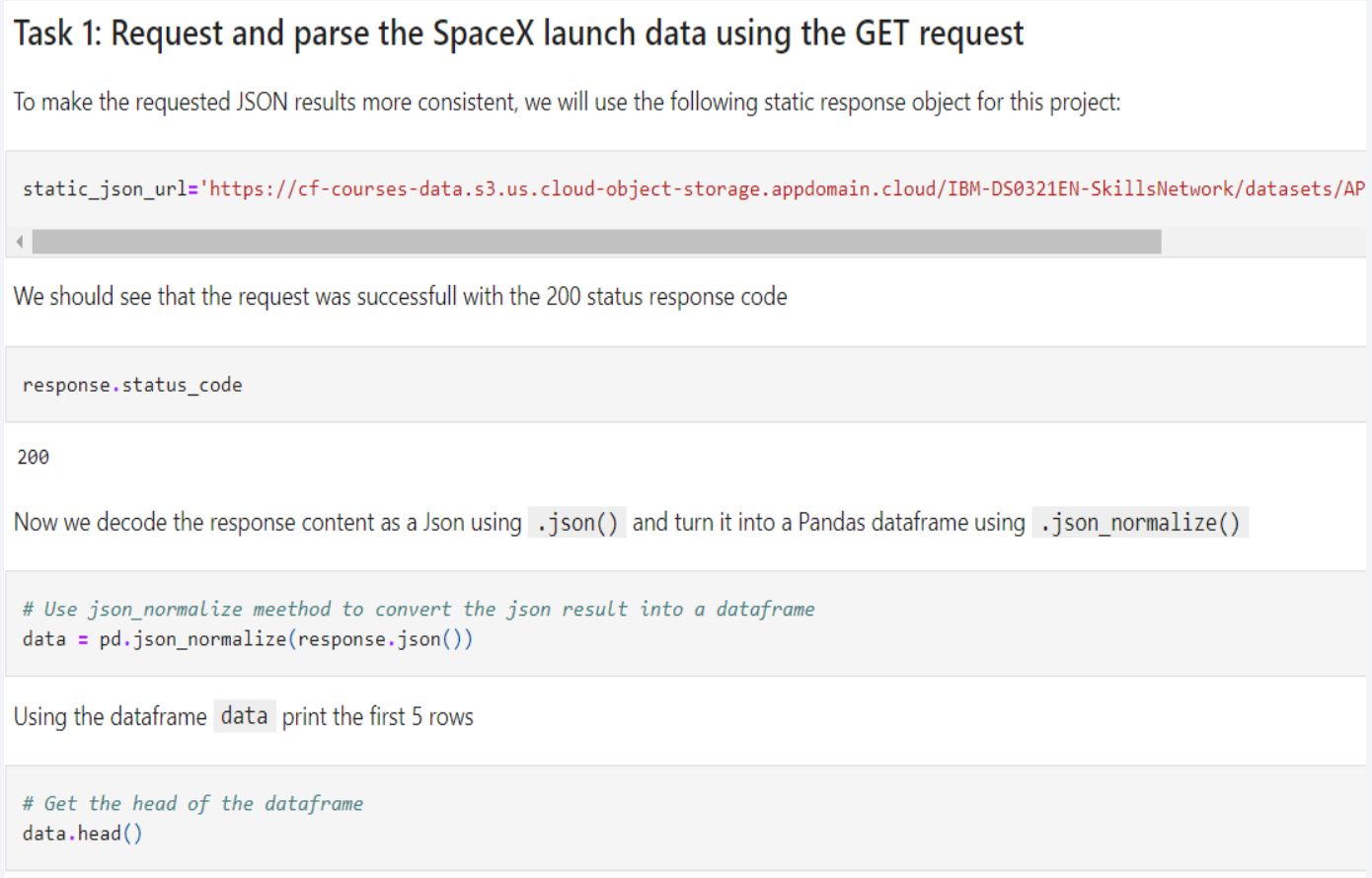
- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

# Data Collection

- Description of  how the data sets were collected SpaceX Falcon9 project
  - Data was first collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API. This was done by first defining a series helper functions that would help in the use of the API to extract information using identification numbers in the launch data and then requesting rocket launch data from the SpaceX API url.
  - Finally, to make the requested JSON results more consistent, the SpaceX launch data was requested and parsed using the GET request and then decoded the response content as a Json result which was then converted into a Pandas data frame.
  - Also performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches of the launch records are stored in a HTML. Using 'BeautifulSoup' and request Libraries, I extract the Falcon 9 launch HTML table records from the Wikipedia page, Parsed the table and converted it into a Pandas data frame

# Data Collection – SpaceX API

- Data collected using SpaceX API (a RESTful API) by making a get request to the SpaceX API then requested and parsed the SpaceX launch data using the GET request and decoded the response content as a Json result which was then converted into a Pandas data frame
- Here is the GitHub URL of the completed SpaceX API calls notebook:
- ([https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb](https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb))



Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/AP
```

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

# Data Collection - Scraping

- Performed web scraping to collect Falcon9 historical launch records from a Wikipedia using BeautifulSoup and request, to extract the Falcon9 launch records from HTML table of the Wikipedia page, then created a data frame by parsing the launch HTML.
- GitHub URL: (https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/jupyter-labs-webscraping.ipynb)

## TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

## TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup` reference link towards the end of this lab

```
# 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')
```

# Data Wrangling

- After obtaining and creating a Pandas DF from the collected data, data was filtered using the **BoosterVersion** column to only keep the Falcon9 launches, then dealt with the missing data values in the **LandingPad** and **PayloadMass** columns. For the PayloadMass , missing data values were replaced using mean value of column.
- Also performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.
- GitHub URL: ([https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb](https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb))

**TASK 4: Create a landing outcome label from Outcome column**

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcor` it's one. Then assign it to the variable `landing_class`:

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for i in df['Outcome']:
    if i not in bad_outcomes:
        landing_class.append(1)
    else:
        landing_class.append(0)
```

```
#df['Class'].head(8)
```

```
#landing_class
```

This variable will represent the classification variable that represents the outcome of each launc land successfully; one means the first stage landed Successfully

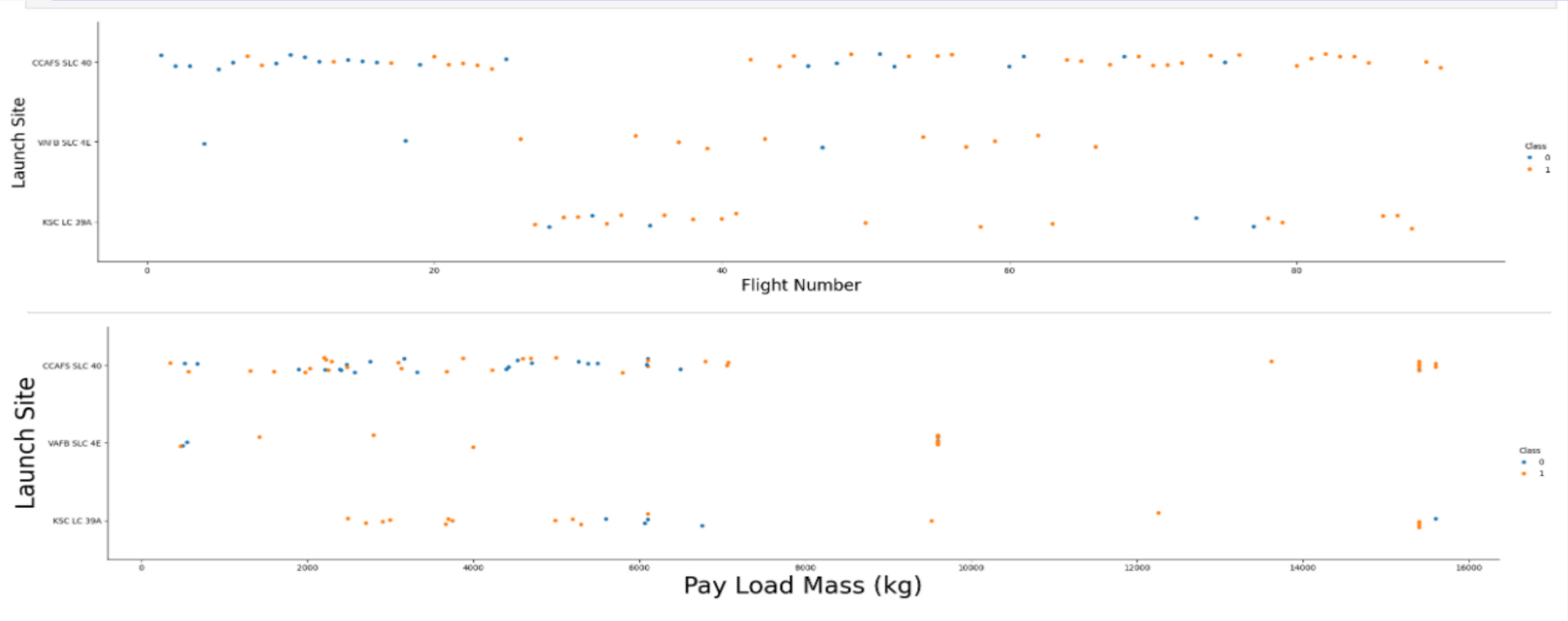
```
df['Class']=landing_class
df[['Class']].head(8)
```

| | Class |
|---|---|
| 0 | 0 |
| 1 | 0 |

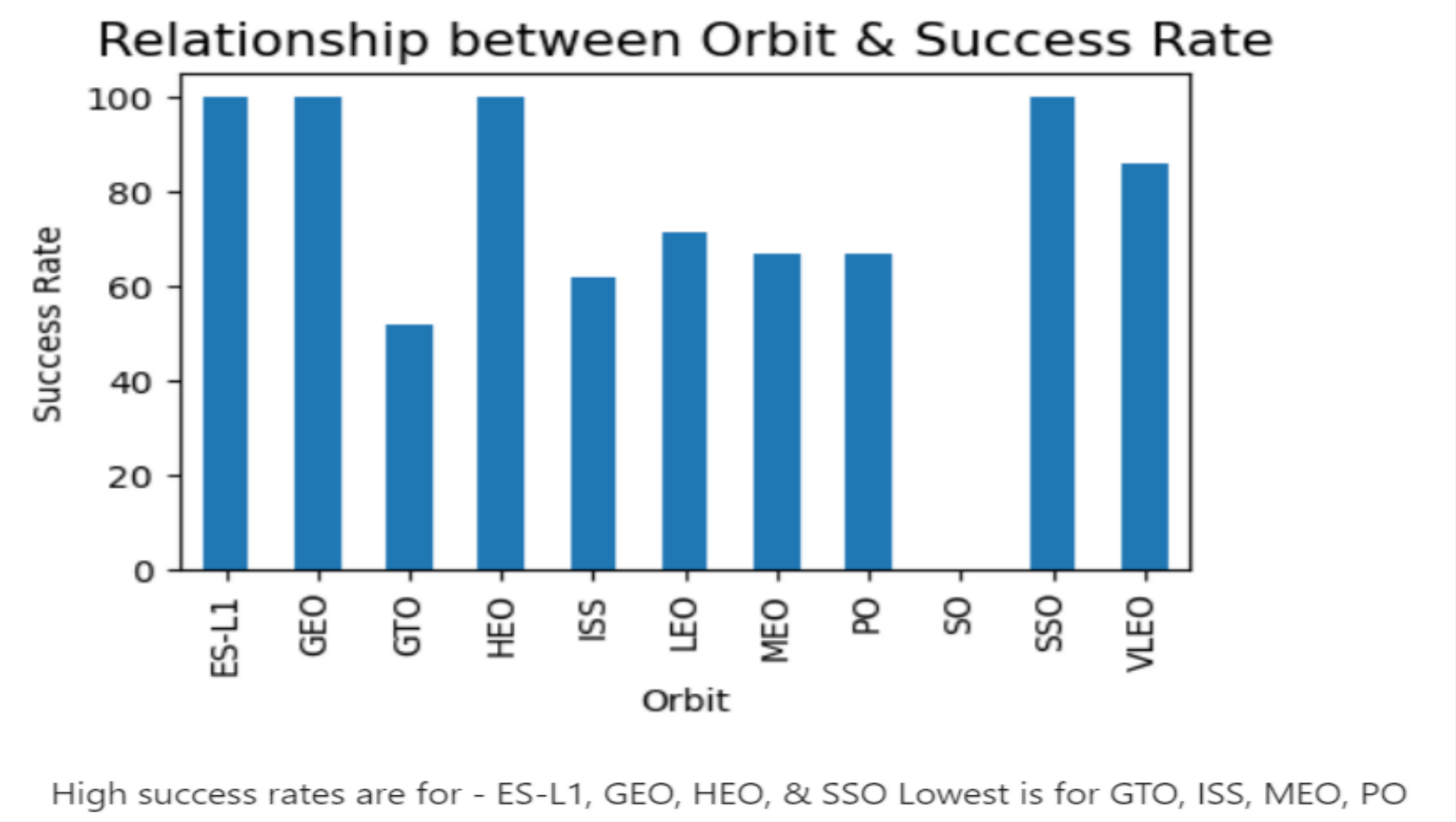
# EDA with Data Visualization

- Performed data Analysis and Feature Engineering using Pandas and Matplotlib.i.e.
  - Exploratory Data Analysis - making the dataset well more reliable for Prediction.
  - Feature Engineering – selected the best features that can predict the outcome accurately.
- Used scatter plots to Visualize the relationship between Flight Number and Launch Site, Payload and Launch Site, FlightNumber and Orbit type, Payload and Orbit type.
- Used Bar chart to Visualize the relationship between success rate of each orbit type.
- Line plot to Visualize the launch success yearly trend.
- GitHub URL: (https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/edadataviz.ipynb)

# EDA with Data Visualization (Plots)

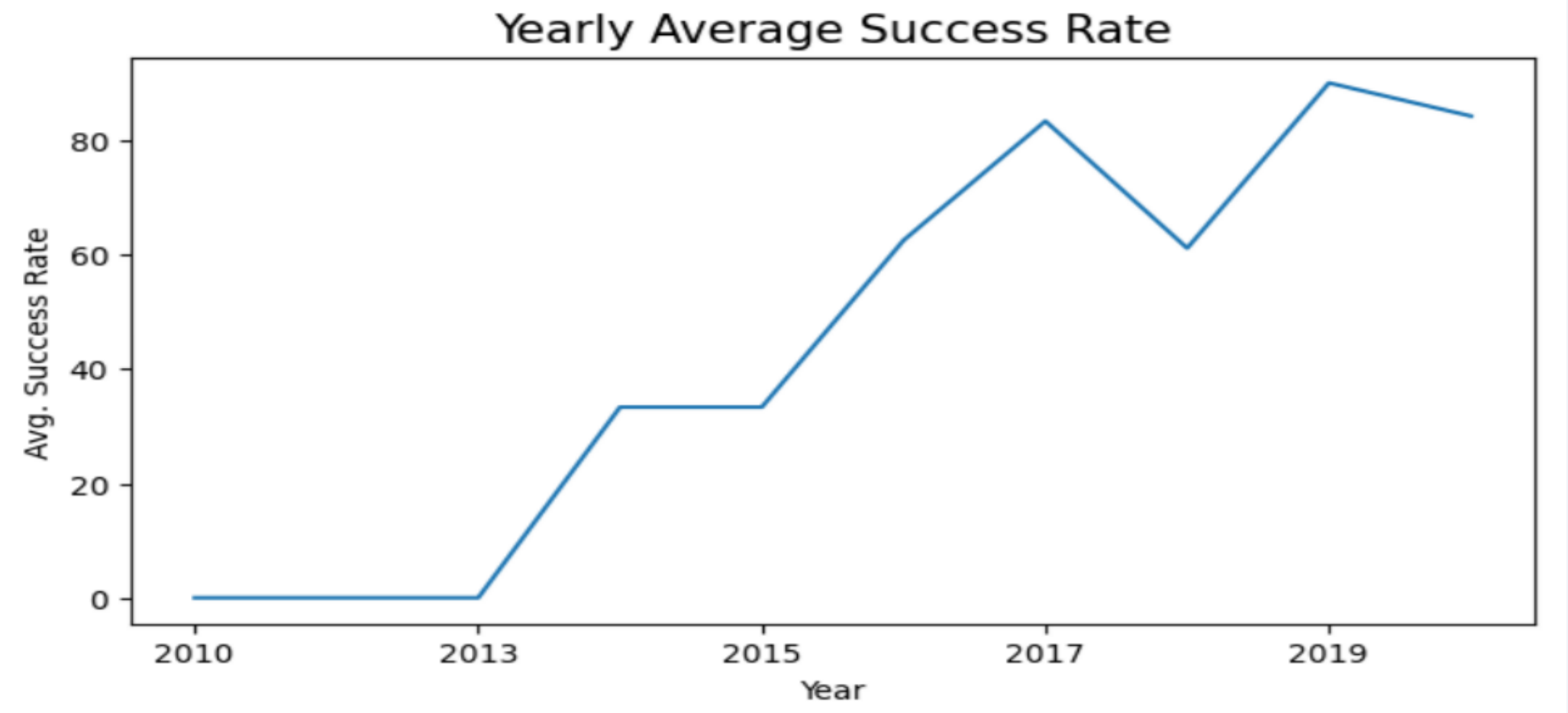


# EDA with Data Visualization (Plots)




High success rates are for - ES-L1, GEO, HEO, & SSO Lowest is for GTO, ISS, MEO, PO

# EDA with Data Visualization (Plots)



# EDA with SQL

- Queries performed using SQL were the following:
  - Display the names of the unique launch sites in the space mission.

```
%sql select Distinct(LAUNCH_SITE) from SPACEXTABLE;
```

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

```
%sql Select Launch_Site from SPACEXTABLE where Launch_Site like 'CCA%' Limit 5;
```

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

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL where Customer = "NASA (CRS)";
```

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

```
%sql select * from SPACEXTABLE limit 100;
```

# EDA with SQL

- List the date when the first successful landing outcome in ground pad was achieved.

```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome = "Success (ground pad)";
```

- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.

```
%sql select Booster_Version from SPACEXTABLE where Landing_Outcome = "Success (drone ship)" and PAYLOAD_MASS__KG_ between 4000 and 6000;
```

- List the total number of successful and failure mission outcomes.

```
%sql select Mission_Outcome, count(Mission_Outcome) as "Count" from SPACEXTABLE Group By Mission_Outcome;
```

- GitHub URL: (https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb)

# Build an Interactive Map with Folium

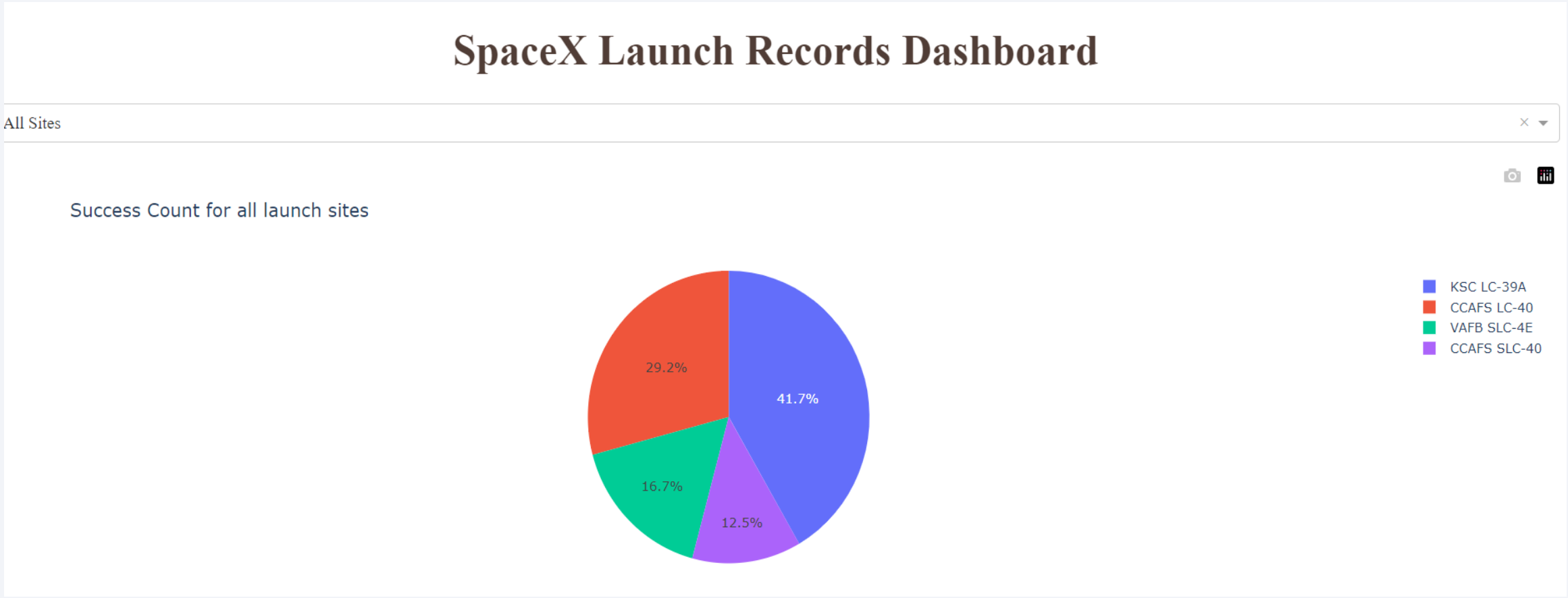
- Created folium map to marked all the launch sites, and created map objects such as markers, circles, lines to mark the success or failure of launches for each launch site.
- Created a launch set outcomes (failure=0 or success=1).
- GitHub URL: (https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/lab_jupyter_launch_site_location.ipynb).

# Build a Dashboard with Plotly Dash

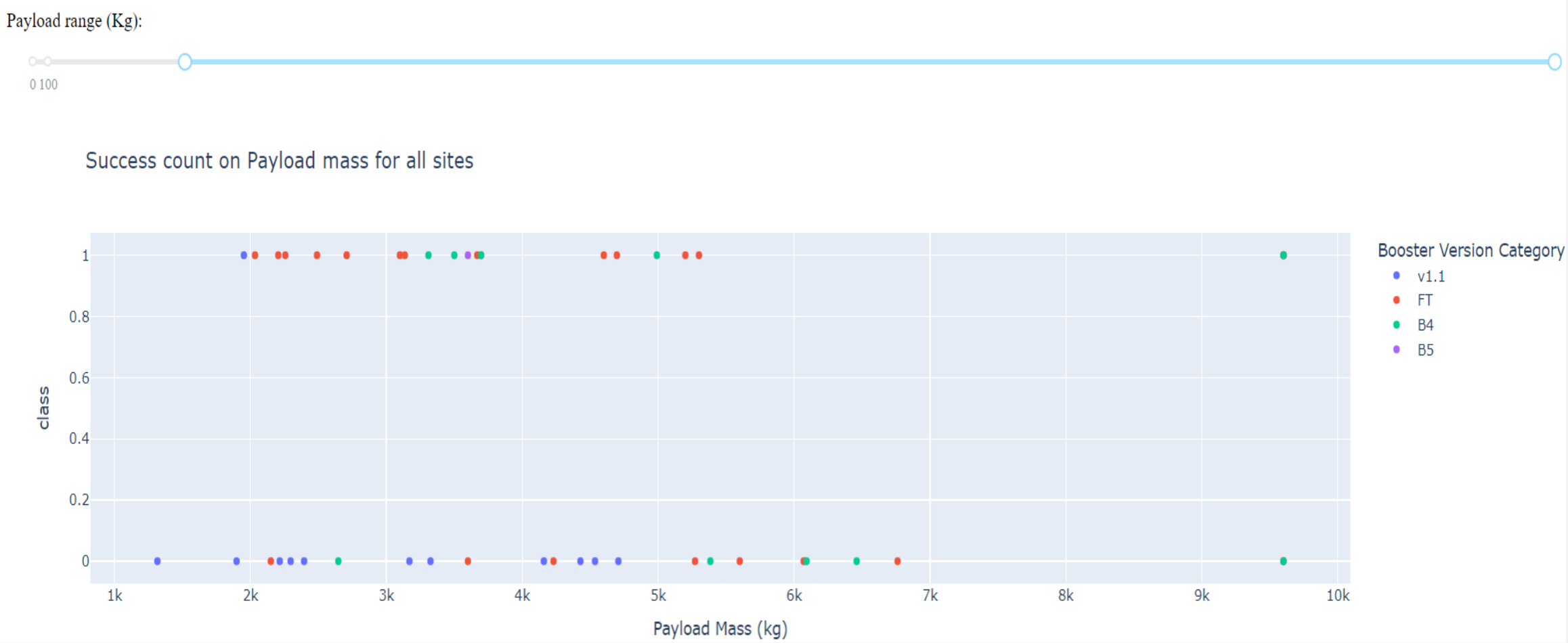
- Built an interactive dashboard application with Plotly dash by:
  - Adding a Launch Site drop-down list as Input Component.
  - Adding a callback function to render success pie chart based on sites selected on drop-down list.
  - Adding a Range slider to select the PayLoad.
  - Adding a callback function to render the Success vs PayLoad scatter plot.

- GitHub URL: (https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/spacex_dash_app.py)

# SpaceX Dash App



# SpaceX Dash App



# Predictive Analysis (Classification)

- Summary of how I built, evaluated, improved, and found the best performing classification model
- After loading the data as a Pandas Dataframe, I set out to perform exploratory Data Analysis and determine Training Labels by;
  - Creating a NumPy array from the column Class in data, by applying the method to_numpy() then assigned it to the variable Y as the outcome variable.
  - Then standardized the feature dataset (x) by transforming it using preprocessing.StandardScaler() function from Sklearn.
  - After which the data was split into training and testing sets using the function train_test_split from sklearn.model_selection with the test_size parameter set to 0.2 and random_state to 2.

# Predictive Analysis (Classification)

- In order to find the best ML model/ method that would performs best using the test data between SVM, Classification Trees, k nearest neighbors and Logistic Regression;
  - First created an object for each of the algorithms then created a GridSearchCV object and assigned them a set of parameters for each model.
  - For each of the models under evaluation, the GridsearchCV object was created with cv=10, then fit the training data into the GridSearch object for each to Find best Hyperparameter.
  - After fitting the training set, we output GridSearchCV object for each of the models, then displayed the best parameters using the data attribute best_params_ and the accuracy on the validation data using the data attribute best_score_.
  - Finally using the method score to calculate the accuracy on the test data for each model and plotted a confussion matrix for each using the test and predicted outcomes.

# Predictive Analysis (Classification)

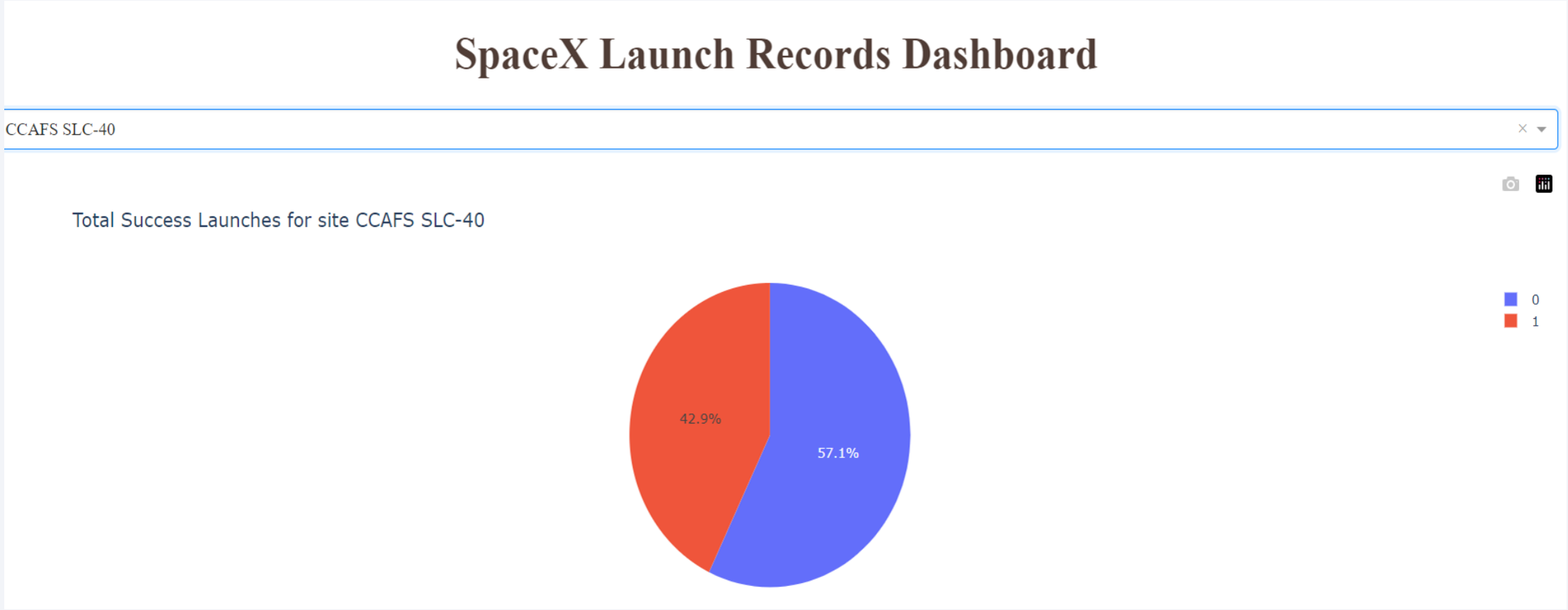
- The table below shows the test data accuracy score for each of the methods comparing them to show which performed best using the test data between SVM, Classification Trees, k nearest neighbors and Logistic Regression;
- All the four models happens to have same Accuracy score of 83.33%.
- GitHub URL: (https://github.com/rooneyy-98/SpaceX-Landing-Prediction-IBM-Project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5%20(1).ipynb).

# Results

- Exploratory data analysis results
  - CCAFS SLC-40 Launch Site has the highest Individual (Success vs Fail) Success Rate.
  - FT BoosterVersion has highest Success count for PayLoad between 2K to 10K (Kg).
- Predictive analysis results
  - All the models shows same accuracy score (83.33%) and confusion matrix. Any of the model with their best parameters can be selected for the prediction.

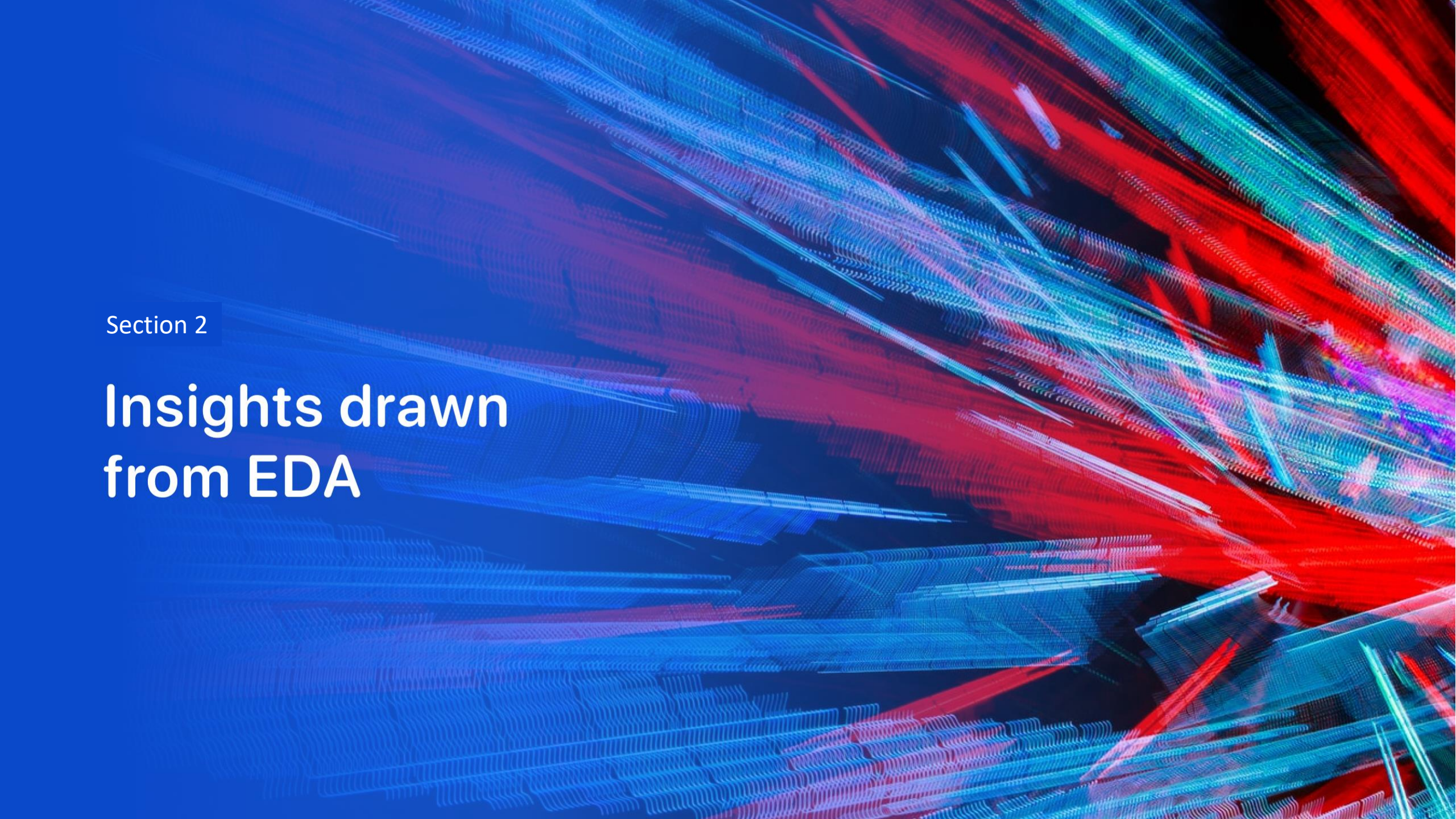
# Results

- Interactive analytics demo in screenshots

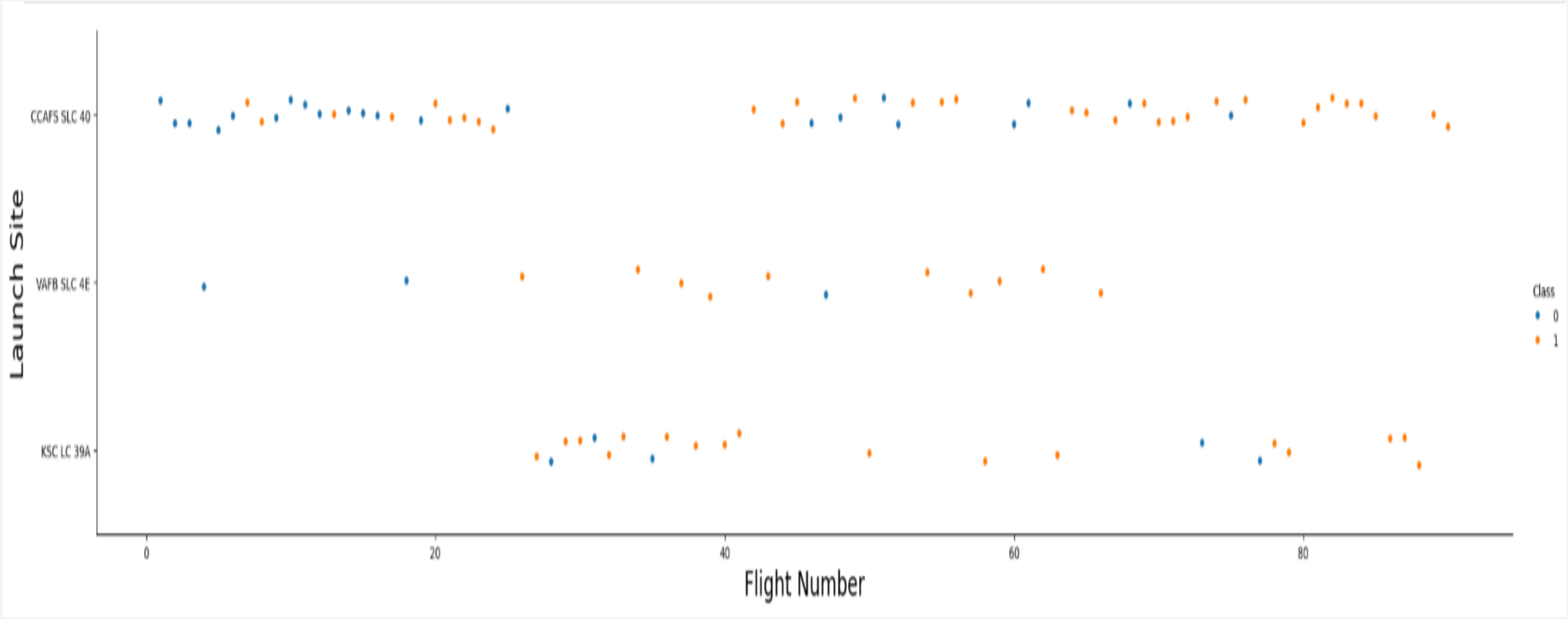


# Results

- Interactive analytics demo in screenshots


Section 2
Insights drawn from EDA

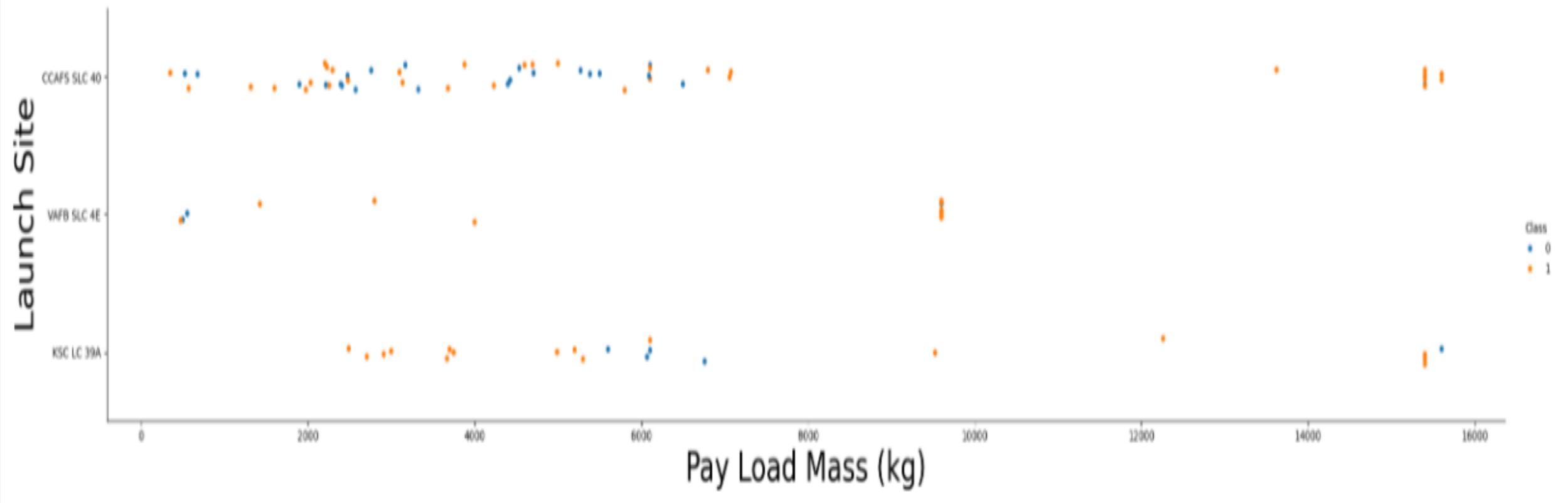
# Flight Number vs. Launch Site



# Flight Number vs. Launch Site

- Launch Site CCAFS SLC 40 has more success rate compared to the rest and flight number above 40 also has a significant success rate.

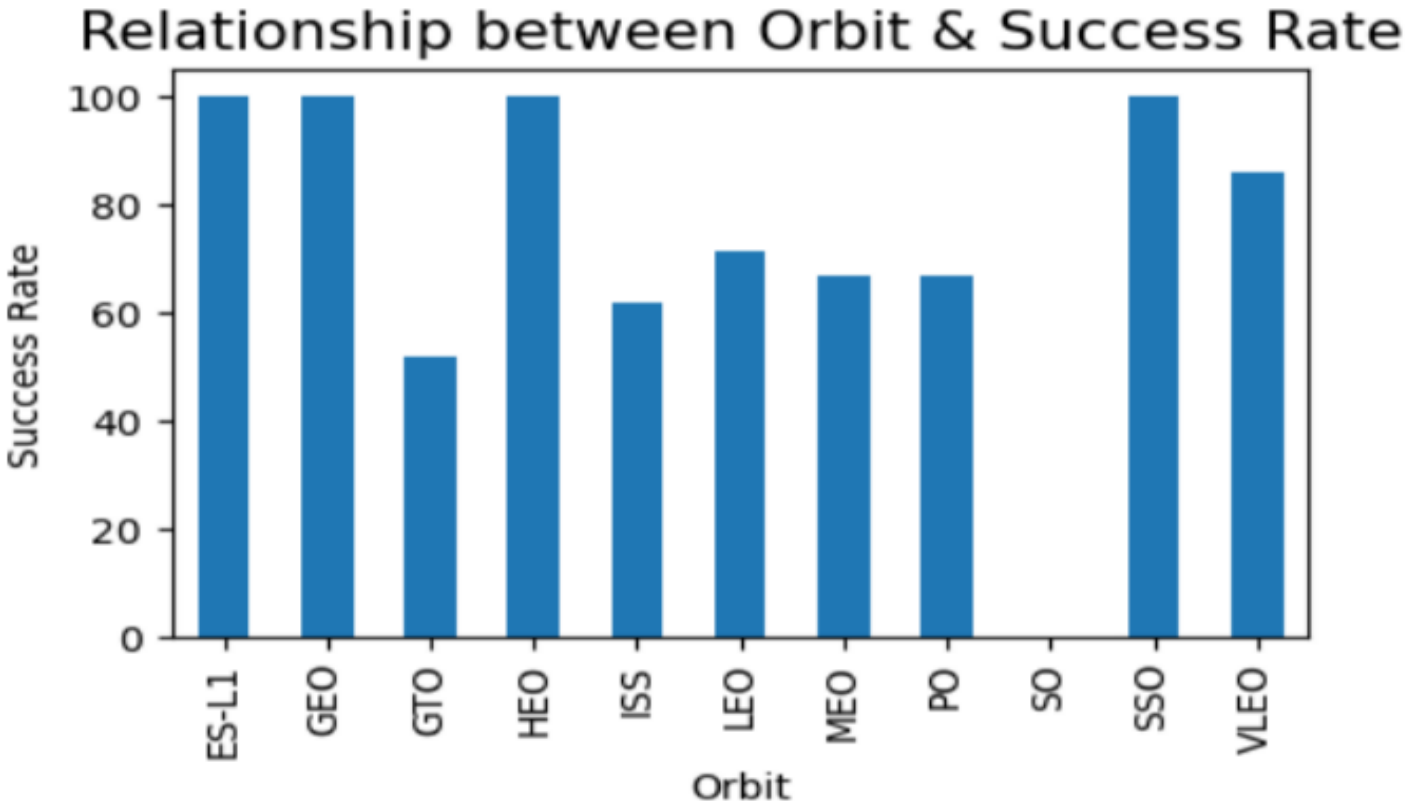
# Payload vs. Launch Site



# Payload vs. Launch Site

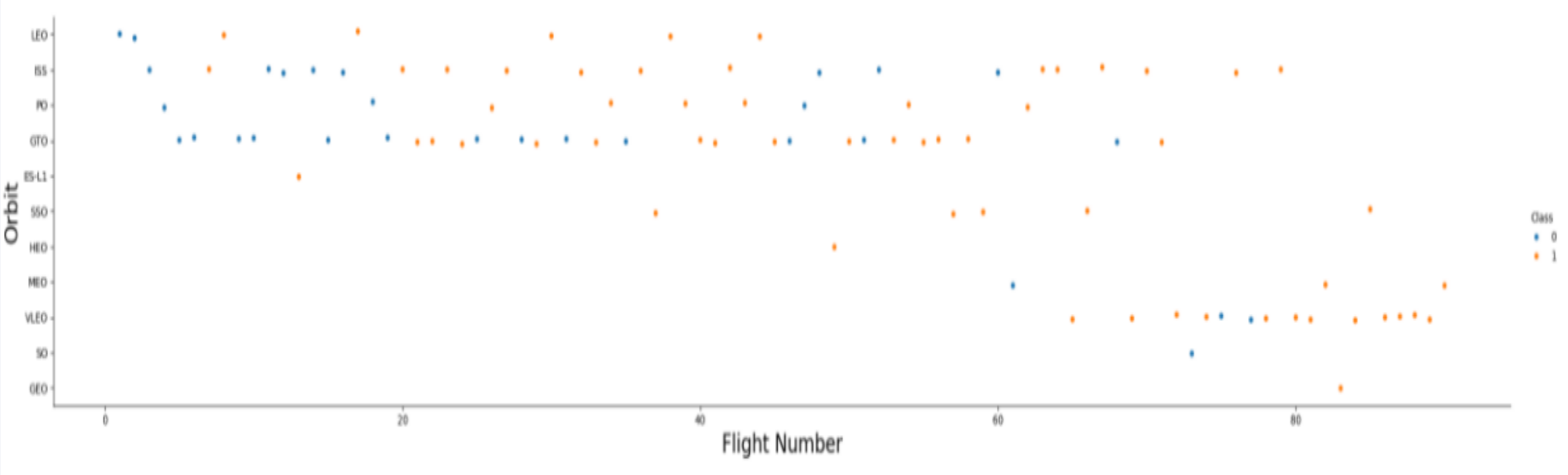
- if you observe Payload Mass Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

# Success Rate vs. Orbit Type



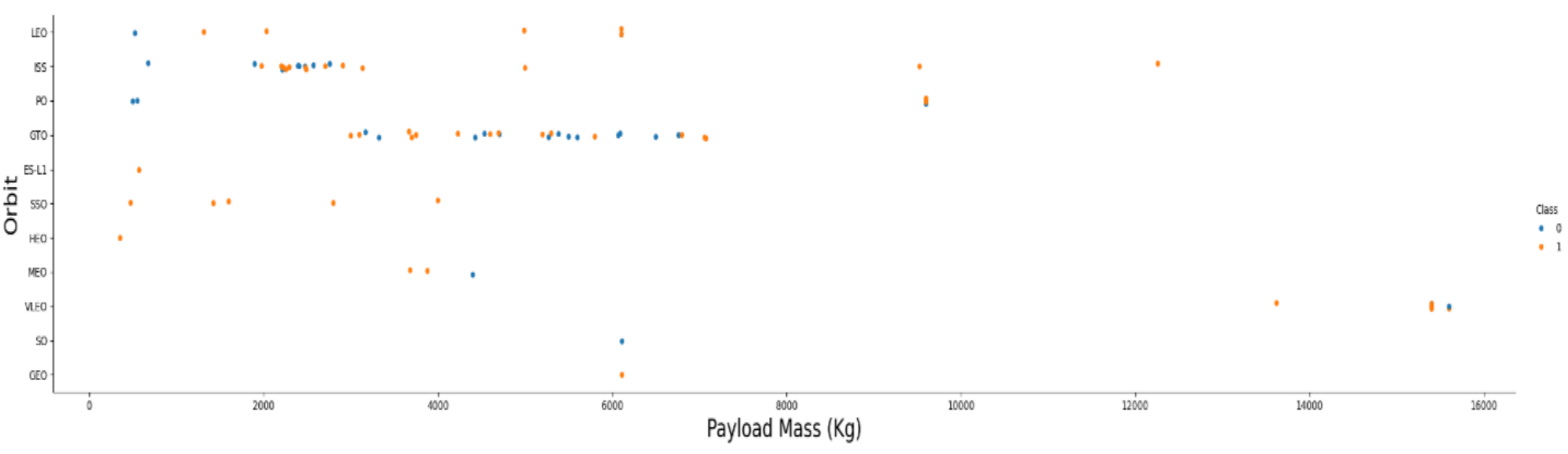

High success rates are for - ES-L1, GEO, HEO, & SSO Lowest is for GTO, ISS, MEO, PO

# Flight Number vs. Orbit Type




You can observe that in the LEO orbit, success seems to be related to the number of flights. Conversely, in the GTO orbit, there appears to be no relationship between flight number and success.

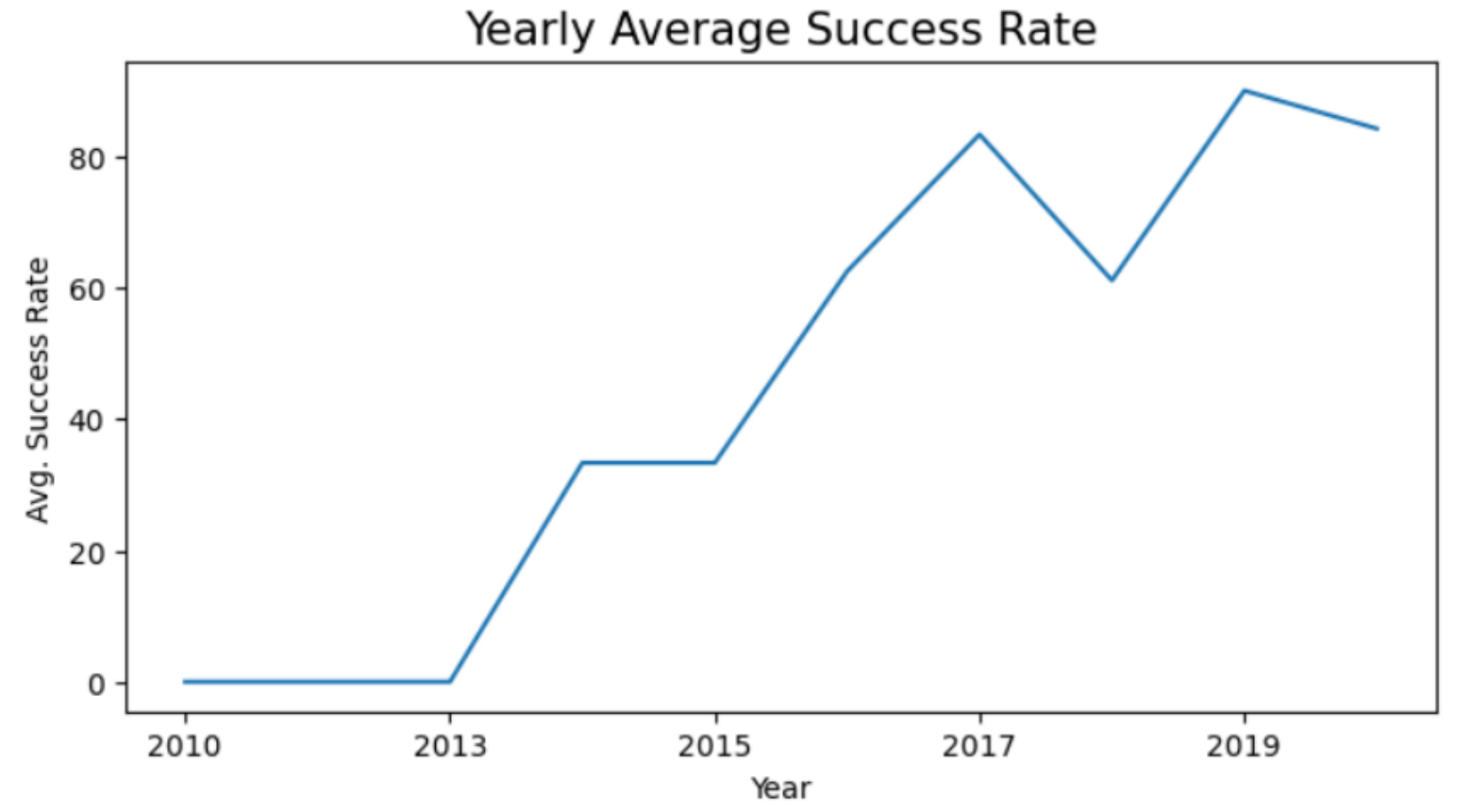
# Payload vs. Orbit Type




With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS. However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

# Launch Success Yearly Trend

The success rate since 2013 kept increasing till 2020



# All Launch Site Names

| Launch_Site |
| --- |
| CCAFS LC-40 |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |

```
%sql select Distinct(LAUNCH_SITE) from SPACEXTABLE;
```

- The query selects all distinct / unique names (using the 'DISTINCT()' function) of Launch Sites from the table SPACEXTABLE.

# Launch Site Names Begin with 'CCA'

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

```
%sql Select Launch_Site from SPACEXTABLE where Launch_Site like 'CCA%' Limit 5;
```

- The query return all the Launch Sites from the SPACEXTABLE, with a condition where launch site name contains ‘CCA’ in it.
- Then we limit the results to 5 results.

# Total Payload Mass

```
%sql select sum(PAYLOAD_MASS__KG_) as payloadmass from SPACEXTBL where Customer = "NASA (CRS)";
```

| payloadmass |
| --- |
| 45596 |

- The query calculates the sum of PayloadMass using the SUM() function with a condition given Customer = ‘NASA (CRS)’.

# Average Payload Mass by F9 v1.1

```
%sql select avg(PAYLOAD_MASS__KG_) as payloadmass_avg from SPACEXTABLE where Booster_Version = "F9 v1.1" ;
```

| payloadmass_avg |
| --- |
| 2928.4 |

- The query returns average PayloadMass using the AVG() function and a condition given Booster Version = ‘F9 v1.1’.

# First Successful Ground Landing Date

```
%sql select min(Date) from SPACEXTABLE where Landing_Outcome = "Success (ground pad)";
```

| min(Date) |
| --- |
| 2015-12-22 |

- The query returns the first successful ground landing date by using the MIN() function on the date column and the condition where landing outcome column = “Success (ground pad)”.

# Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select Booster_Version from SPACEXTABLE where Landing_Outcome = "Success (drone ship)" and PAYLOAD_MASS__KG_ between 4000 and 6000;
```

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

- The query executes two conditions; 1- where landing outcome = “Success (drone ship)” and 2- where PayloadMass between 4000 & 6000.
- There are 4 such Booster Versions as shown in the above screenshot.

# Total Number of Successful and Failure Mission Outcomes

```
%sql select Mission_Outcome, count(Mission_Outcome) as "Count" from SPACEXTABLE Group By Mission_Outcome;
```

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

- There are total of 100 Success cases among which, one is Success count belongs to the payload status which is unclear. Total Failure counts is 1.
- The query uses the COUNT() function to count & GROUP BY function to get this result.

# Boosters Carried Maximum Payload

```
%sql select Booster_Version as "Booster Version" from SPACEXTABLE where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_)
                                                                                              from SPACEXTABLE);
```

| Booster Version |
|---|
| F9 B5 B1048.4 |
| F9 B5 B1049.4 |
| F9 B5 B1051.3 |
| F9 B5 B1056.4 |
| F9 B5 B1048.5 |
| F9 B5 B1051.4 |
| F9 B5 B1049.5 |
| F9 B5 B1060.2 |
| F9 B5 B1058.3 |
| F9 B5 B1051.6 |
| F9 B5 B1060.3 |
| F9 B5 B1049.7 |

- There are 12 Booster Versions which have carried maximum payload mass.
- The query uses MAX() function on the Payload Mass column to obtain the maximum payload value.

# 2015 Launch Records

```
%sql select substr(Date, 6,2) as Month, Mission_Outcome, Booster_Version, Landing_Outcome, Launch_Site from SPACEXTABLE
where substr(Date, 0,5)='2015' and Landing_Outcome = "Failure (drone ship)";
```

| Month | Mission_Outcome | Booster_Version | Landing_Outcome | Launch_Site |
|---|---|---|---|---|
| 01 | Success | F9 v1.1 B1012 | Failure (drone ship) | CCAFS LC-40 |
| 04 | Success | F9 v1.1 B1015 | Failure (drone ship) | CCAFS LC-40 |

- There are two records, 1st record on Jan 2014 & 2nd record on April 2015.
- The query returns the records based on the substr() function to split the months & year from the date & checks the condition for 'Failure (drone ship)'.

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

```
%sql SELECT Landing_Outcome, count(Landing_Outcome) as "Count" FROM SPACEXTBL Group By Landing_Outcome Having DATE BETWEEN
'2010-06-04' AND '2017-03-20' ORDER BY count(Landing_Outcome) DESC;
```

| Landing_Outcome | Count |
|---|---|
| No attempt | 21 |
| Success (drone ship) | 14 |
| Success (ground pad) | 9 |
| Failure (drone ship) | 5 |
| Controlled (ocean) | 5 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

- The highest count is for ‘No attempt’ & the least count is for ‘Precluded (drone ship)’.

Section 3

# Launch Sites Proximities Analysis

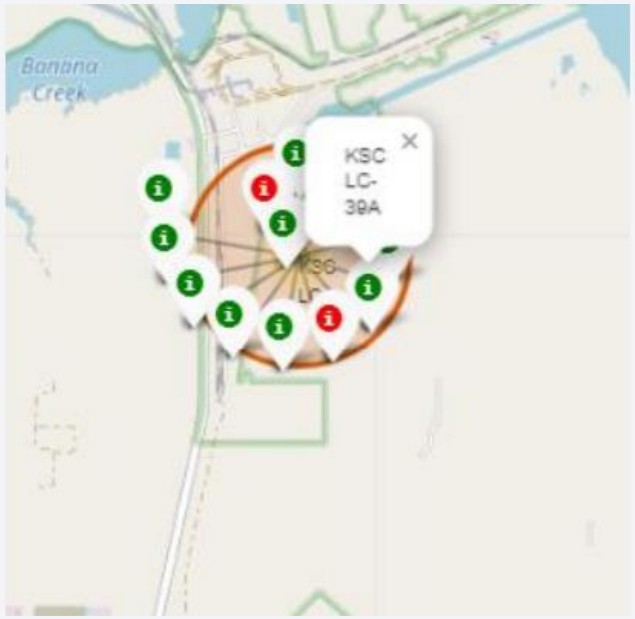
# Launch Sites on Global Map


- All launch sites are in proximity to the Equator, (located southwards of the US map). Also, all the launch sites are in very close proximity to the coast.

# Launch Outcomes with Color Marker on Map

## Florida:



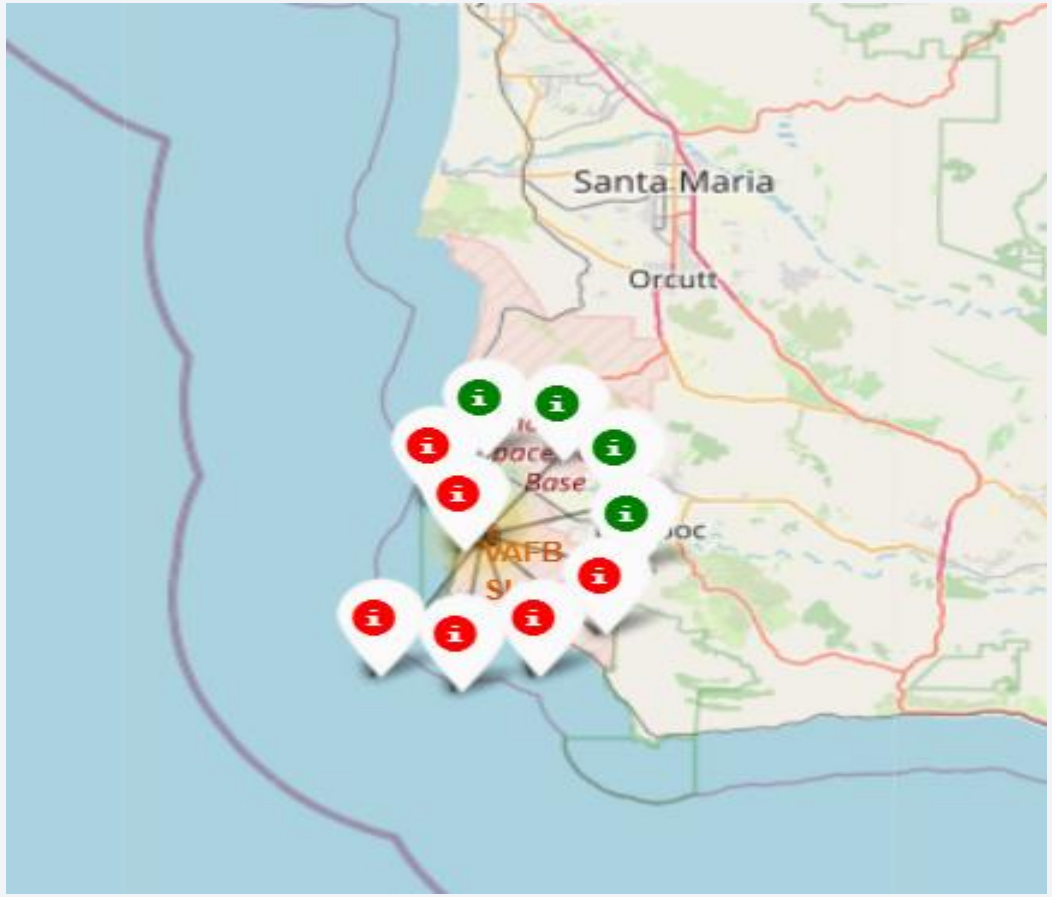



- In the Eastern coast (Florida) Launch site KSC LC-39A has relatively high success rates compared to CCAFS SLC-40 & CCAFS LC-40.

# Launch Outcomes with Color Marker on Map

## West Coast / California:

- In the West Coast (California) Launch site VAFB SLC-4E has relatively lower success rates 4/10 compared to KSC LC39A launch site in the Eastern Coast of Florida.



# Distance between Launch Site and Coastline

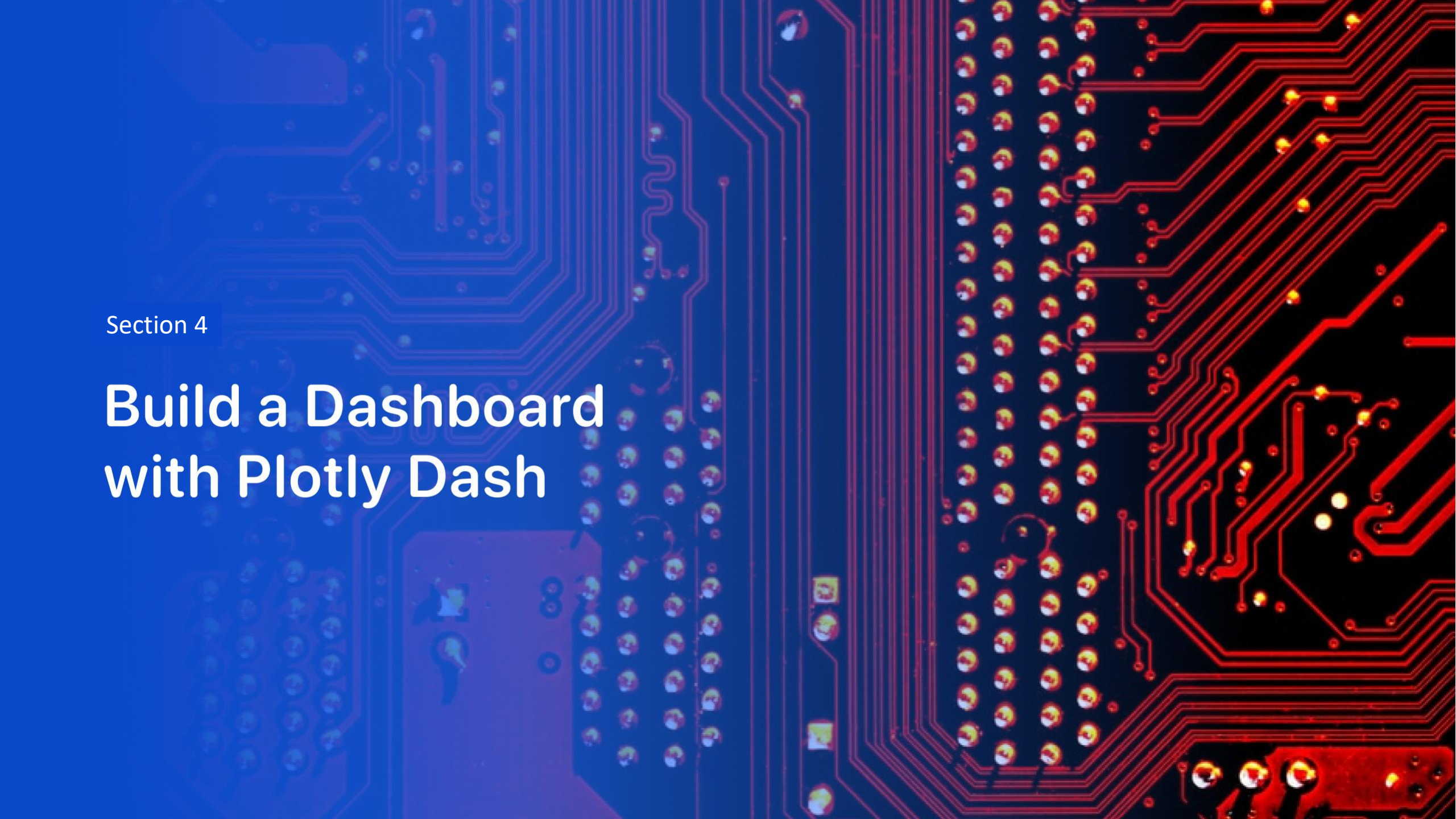

- The Launch site CCAFS SLC-40 proximity to coastline is 0.86km

# Launch Site and its Proximities

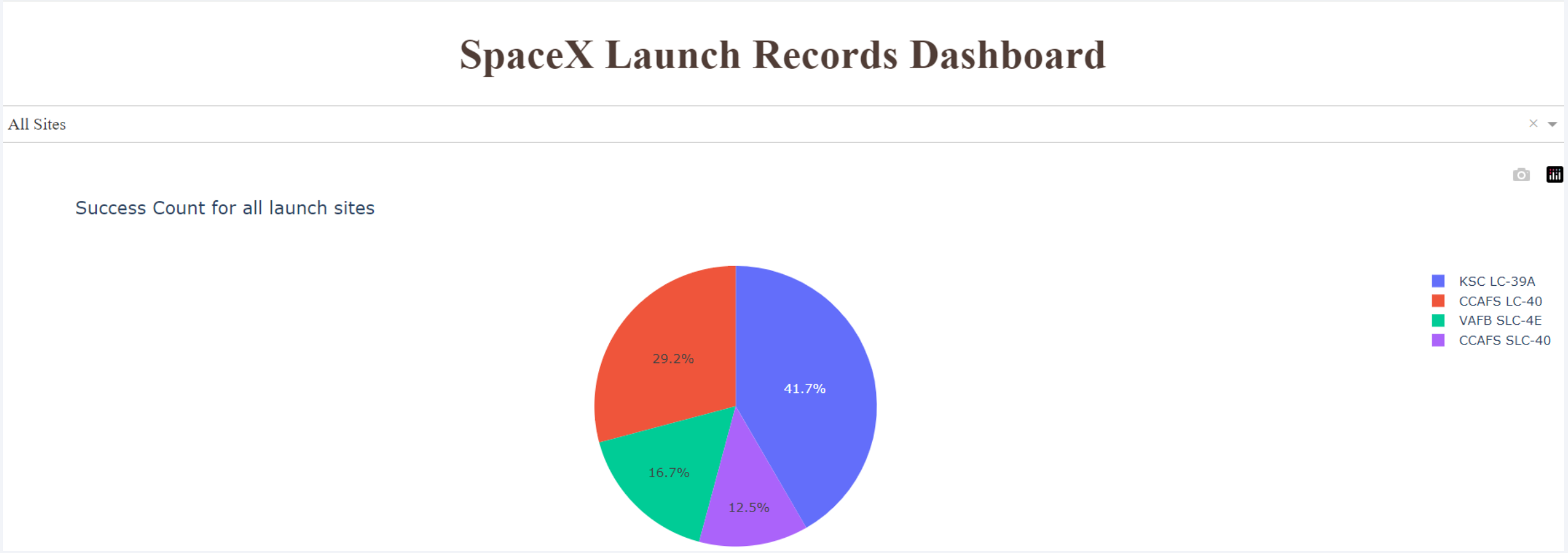

- The Launch site KSC LC 39A has close proximity to Railway, which is 0.55km.

# Launch Site and its Proximities - Findings

- It can be observed that resourceful facilities like railways & coastline are always in closer proximity to the Launch site, ensuring the movement of materials is smooth.
- On the contrary, the cities are the farthest from the Launch sites.
- Highways passes somewhat distant from the Launch sites.

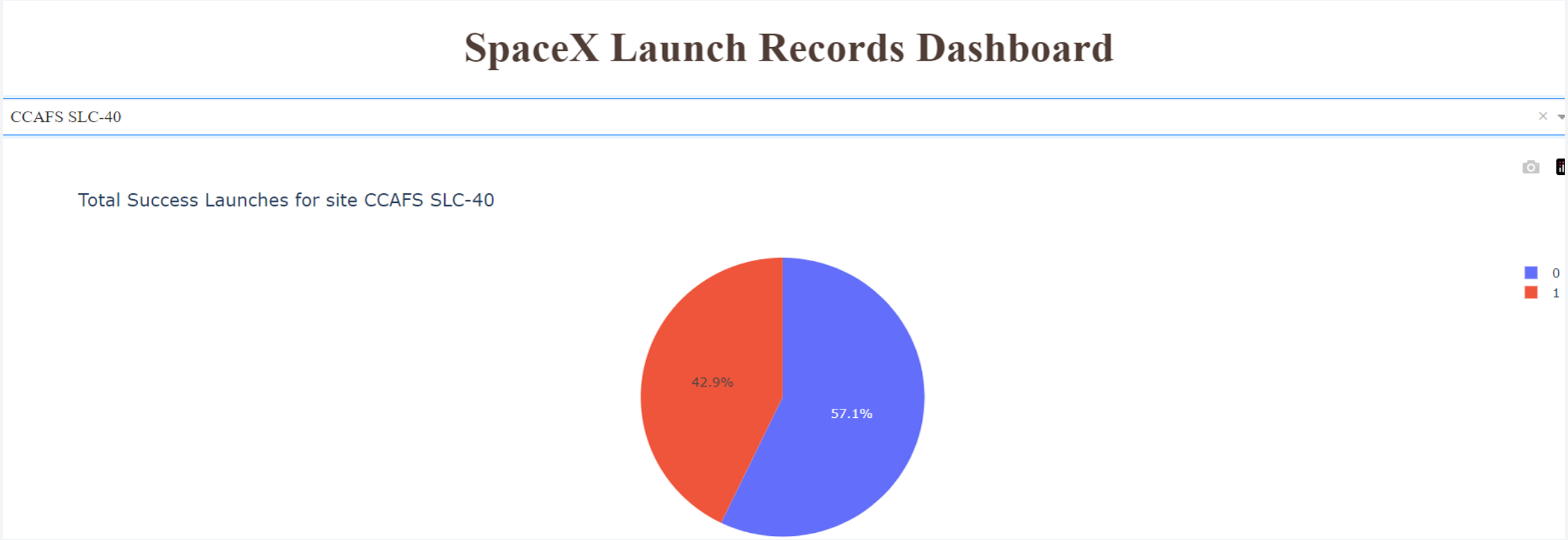

Section 4
Build a Dashboard with Plotly Dash

# Pie Chart – Success Count for All Sites



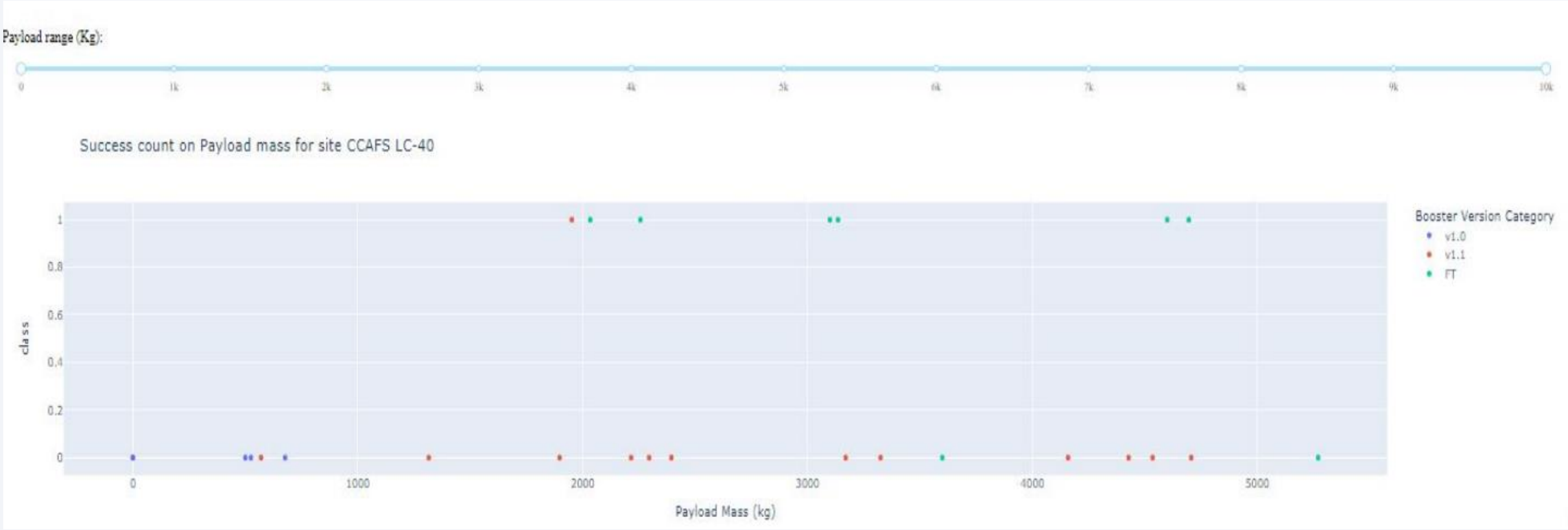

- Launch site KSC LC-39A has the highest launch success rate at 42% followed by CCAFS LC-40 at 29%, VAFB SLC-4E at 17% and lastly launch site CCAFS SLC-40 with a success rate of 13%

# Pie Chart – Total Success Launches (Highest Ratio)




- Launch site CCAFS SLC-40 has the highest success ratio of 42.9% success against 57.1% failed launches

# Payload vs Launch Outcome – Scatter Plot for all sites




- For Launch site CCAFS LC-40 the booster version FT has the largest success rate from a payload mass of >2000kg

Section 5

# Predictive Analysis (Classification)

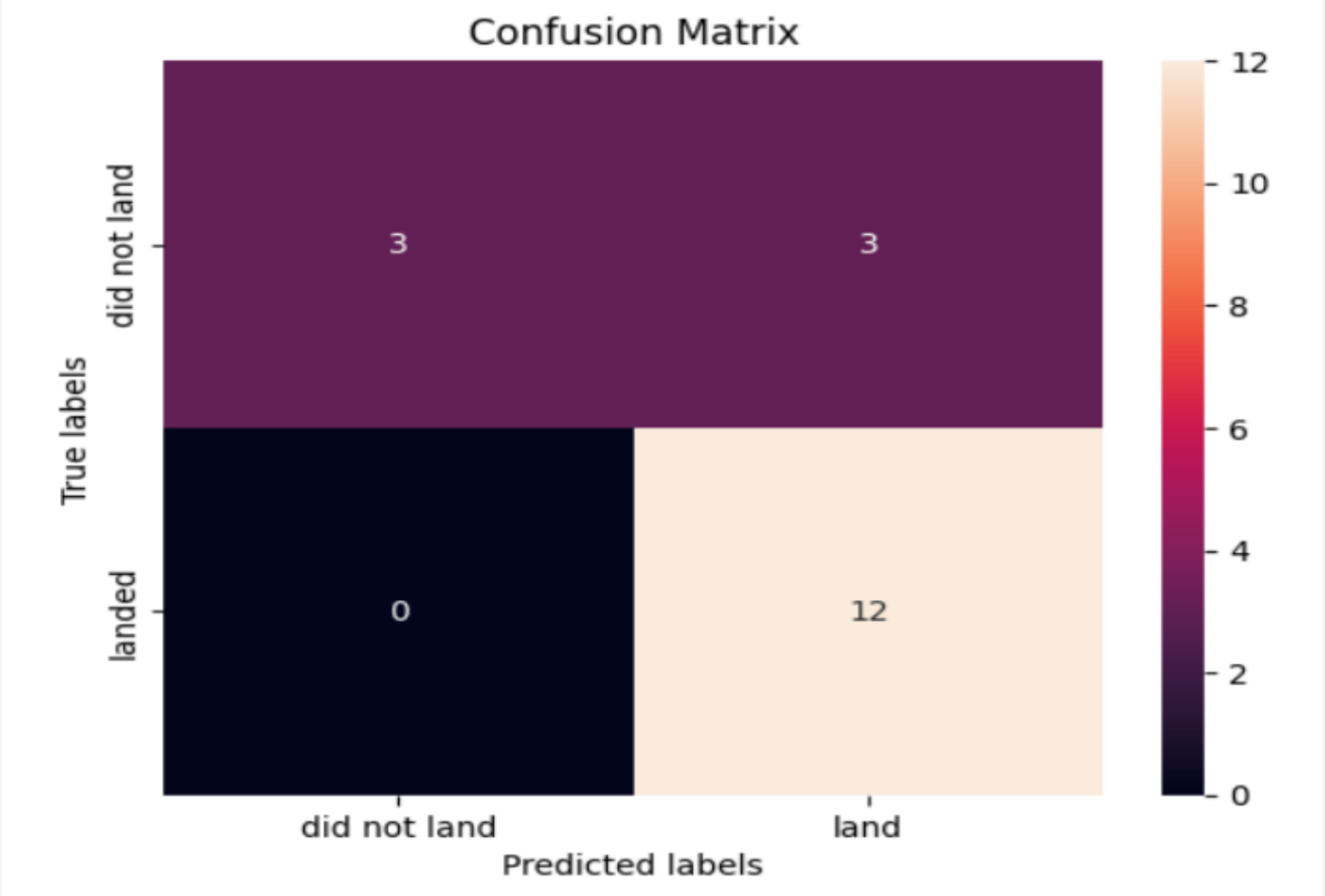
# Classification Accuracy

| Method | Test Data Accuracy |
|---|---|
| Logistic_Reg | 0.833333 |
| SVM | 0.833333 |
| Decision Tree | 0.833333 |
| KNN | 0.833333 |

- All the models performs equally on test data, they all have the same accuracy of 83.33%.

# Confusion Matrix

- All four models have the same confusion matrix.
- A significant drawback comes in the False Positives.



# Conclusions

- Different launch sites have different success rates. CCAFS LC-40, has a success rate of 60 %, while KSC LC-39A and VAFB SLC 4E has a success rate of 77%.
- We can deduce that, as the flight number increases in each of the 3 launch sites, so does the success rate. For instance, the success rate for the VAFB SLC 4E launch site is 100% after the Flight number 50. Both KSC LC 39A and CCAFS SLC 40 have a 100% success rates after 80th flight
- If you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launch-site there are no rockets launched for heavy payload mass(greater than 10000).
- Orbits ES-L1, GEO, HEO & SSO have the highest success rates at 100%, with SO orbit having the lowest success rate at ~50%. Orbit SO has 0% success rate. • LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

# Conclusions

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS. However, for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.
- Ans finally the success rate since 2013 kept increasing till 2020.

# Appendix

```
parameters = {'criterion': ['gini', 'entropy'],
      'splitter': ['best', 'random'],
      'max_depth': [2*n for n in range(1,10)],
      'max_features': ['auto', 'sqrt'],
      'min_samples_leaf': [1, 2, 4],
      'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train,Y_train)
```

```
/lib/python3.12/site-packages/sklearn/model_selection/_validation.py:547: FitFailedWarning:
3240 fits failed out of a total of 6480.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
3240 fits failed with the following error:
Traceback (most recent call last):
  File "/lib/python3.12/site-packages/sklearn/model_selection/_validation.py", line 895, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/lib/python3.12/site-packages/sklearn/base.py", line 1467, in wrapper
    estimator._validate_params()
  File "/lib/python3.12/site-packages/sklearn/base.py", line 666, in  validate params
```

- An error often occurred while training the tree model. Stack overflow did suggested a solution of deleting the rows with empty cells. This often erro caused difference performance.

# Thank you!