IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nakul Pant

20-Dec-2023

Outline

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

Executive Summary

- Summary of methodologies:

❑ Data Collection : Working with SPACEX data generated from REST API.

❑ Web Scrapping : Using python Beautiful Soup.

❑ Data Wrangling : Using panda and Numpy.

❑ Exploratory Data Analysis : With the help of Matplotlib, Seaborne and Folium and Dashboard.

❑ Model Development : With the help of Si-kit.

Outline

•Summary of all results:

- ❑ Data Collection Result : Data gathered in proper format with column names.
- ❑ Data Cleaning Result : Removing all the discrepancies in the data.
- ❑ Exploratory Data Analysis Result : Visualization and dashboard and screenshots.
- ❑Model Development Result : Through cross validation and model score.
- ❑ Cross Validation : Checking accuracy scores and hyper parameter tuning.

Introduction

- Project background and context :

We will predict if the Falcon 9 first stage will land successfully. 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.

Introduction

Problems you want to find answers:

Prediction Accuracy:

How accurately can the model predict the successful landing of the Falcon 9 first stage? What is the model's precision, recall, and overall accuracy?

Cost Analysis:

Once the model predicts the success of the landing, how does this information impact the overall cost of the rocket launch? What is the potential cost savings for launches where the first stage is expected to land successfully?

Bid Strategy:

Given the predicted landing outcome, how can an alternate company strategically bid against SpaceX for a rocket launch contract? What factors should be considered in developing a competitive bid?

Operational Insights:

Are there operational insights that can be derived from the data? For example, are there specific conditions or scenarios where the success rate is higher or lower?

Section 1 : Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data was collected using SPACE X API and scraping SPACEX Wikipedia page.
- Perform data wrangling:
  - Data was cleaned using python libraries like Numpy and Pandas.
- Perform exploratory data analysis (EDA) using visualization and SQL.
- Perform interactive visual analytics using Folium and Plotly Dash.
- Perform predictive analysis using classification models.
  - Building ML pipeline: Preprocessing, Train Test Split, hyper parameter tuning , Data Modelling with Logistic Regression, SVM, Decision Tree Classifier and K nearest neighbor.

Data Collection

Following steps were involved in Data Collection:

- SpaceX API: –

  Data was collected by sending a GET request to SpaceX API .

  The data was decoded as json using .json() and turned into a Data Frame using json normalize().

  All the discrepancies in the data were removed like Null values.

- Web-Scraping: –

  Falcon 9 launch records were also collected by web-scraping from Wikipedia.

  A HTTP GET request was sent to Falcon 9 launch HTML page.

  The data was parsed and stored as a Data Frame using BeautifulSoup.

GIT HUB Link : https://github.com/Nakul-Pant-DS/IBM-Data-Science-Capstone-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Data Collection – SpaceX API

Request to SPACEX API:

The code snippet shows how a content is generated by creating a SPACEX URL object by requests.get and then printing the response.

GIT Hub Link : https://github.com/Nakul-Pant-DS/IBM-Data-Science-Capstone-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

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

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

Check the content of the response

```
[15]: print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ship
s":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/NN6Ph45r_o.pn
g","large":"https://images2.imgbox.com/5b/02/QcxHUb5V_o.png"},"reddit":{"campaig
```

Data Collection – SpaceX API

Request and parse the SpaceX launch data using the GET request

[16]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.clo
```

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

[17]:
```
response.status_code
```

[17]:
```
200
```

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

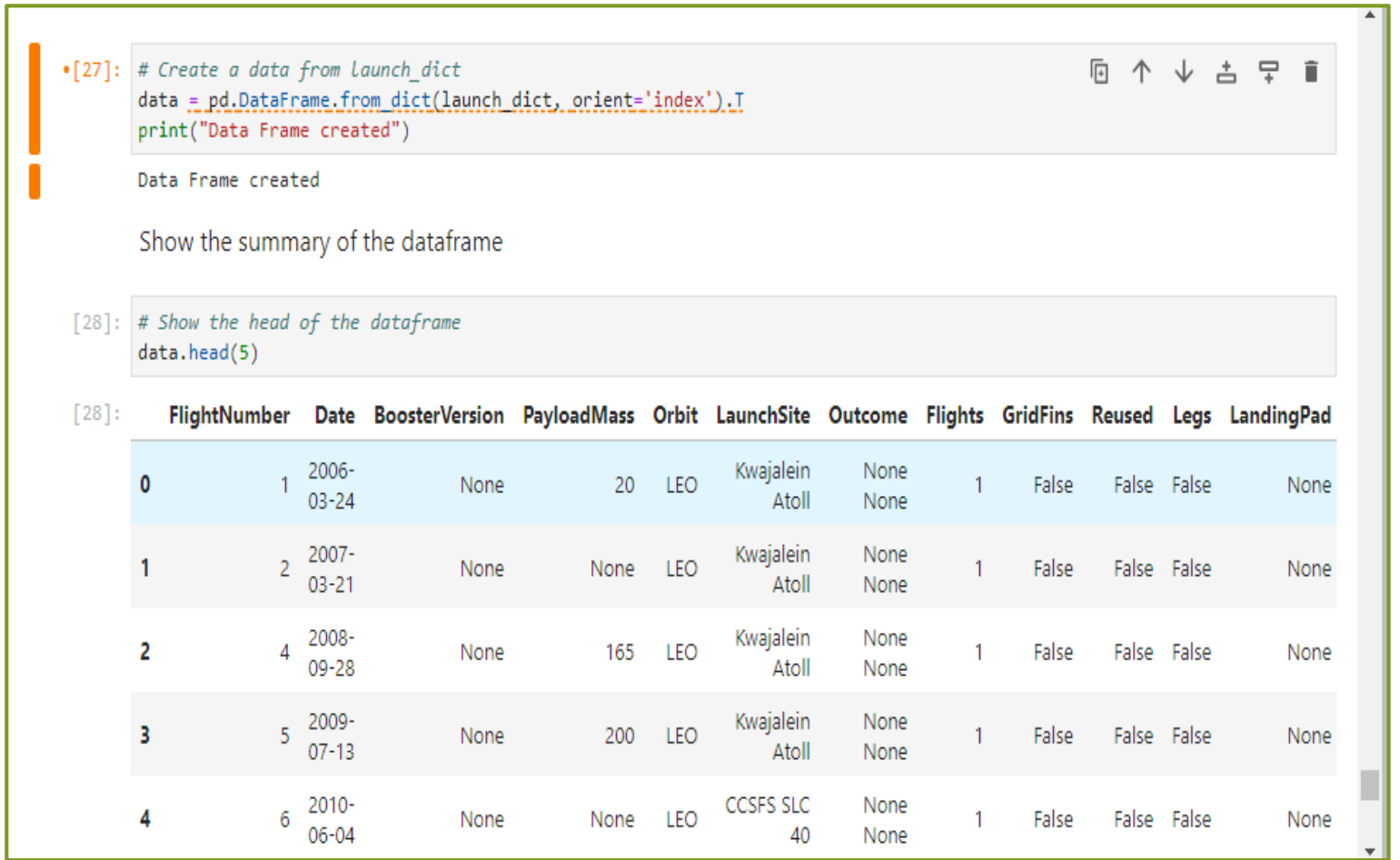
[18]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data= pd.json_normalize(response.json())
print("Normalization Done")
```

```
Normalization Done
```

Data Collection – SpaceX API

After successful retrieval of the data and placing then into a dictionary , we created a panda data frame .

The screenshot shows the sample of data frame generated.



```
[27]: # Create a data from launch_dict
data = pd.DataFrame.from_dict(launch_dict, orient='index').T
print("Data Frame created")
```

```
Data Frame created
```

Show the summary of the dataframe

```
[28]: # Show the head of the dataframe
data.head(5)
```

[28]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad
0	1	2006-03-24	None	20	LEO	Kwajalein Atoll	None None	1	False	False	False	None
1	2	2007-03-21	None	None	LEO	Kwajalein Atoll	None None	1	False	False	False	None
2	4	2008-09-28	None	165	LEO	Kwajalein Atoll	None None	1	False	False	False	None
3	5	2009-07-13	None	200	LEO	Kwajalein Atoll	None None	1	False	False	False	None
4	6	2010-06-04	None	None	LEO	CCSFS SLC 40	None None	1	False	False	False	None

Data Collection - Scraping

- Request the Falcon9 Launch Wiki page from its URL
- Create a BeautifulSoup object from the HTML response.
- **GIT Hub Link** : https://github.com/Nakul-Pant-DS/IBM-Data-Science-Capstone-Project/blob/main/jupyter-labs-webscraping.ipynb

[6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text conte
html_content = response.text
soup = BeautifulSoup(html_content, 'html.parser')
print("Done")
```

Done

Print the page title to verify if the `BeautifulSoup` object was created properly

[17]:
```
# Use soup.title attribute
title_tag = soup.title
title_tag.text
```

Data Collection - Scraping

- Creating a data frame from the scrapped data
- Finally, exporting the data to a csv file .

```
[15]: df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
print("DataFrame Created")
```

```
DataFrame Created
```

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

Data Wrangling

- Some of the attributes that were scrapped were: Flight Number, Date, Booster version, Payload mass Orbit, Launch Site, Outcome, Grid Fins, Legs, Landing pad, Block, Reused count, Serial, Longitude and latitude of launch.
- Removing Null values and replacing them with the MEAN.
- Calculate the number of launches on each site .
- Calculate the number and occurrence of each orbit
- Create a landing outcome label from Outcome column.

GIT Hub Link : https://github.com/Nakul-Pant-DS/IBM-Data-Science-Capstone-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

Data Wrangling


```
[4]: df.isnull().sum()/len(df)*100
```

```
[4]: FlightNumber      0.000000
     Date              0.000000
     BoosterVersion    0.000000
     PayloadMass       0.000000
     Orbit             0.000000
     LaunchSite        0.000000
     Outcome           0.000000
     Flights           0.000000
     GridFins          0.000000
     Reused            0.000000
     Legs              0.000000
     LandingPad       28.888889
     Block             0.000000
     ReusedCount       0.000000
     Serial            0.000000
     Longitude         0.000000
     Latitude          0.000000
     dtype: float64
```

```
: # Apply value_counts() on column LaunchSite
  df['LaunchSite'].value_counts()
```

```
[13]: # landing_class = 0 if bad_outcome
      # landing_class = 1 otherwise
      landing_class=[]
      for i in df['Outcome']:
          if outcome in bad_outcomes:
              landing_class.append(0)
          else:
              landing_class.append(1)
      print("Done")
```

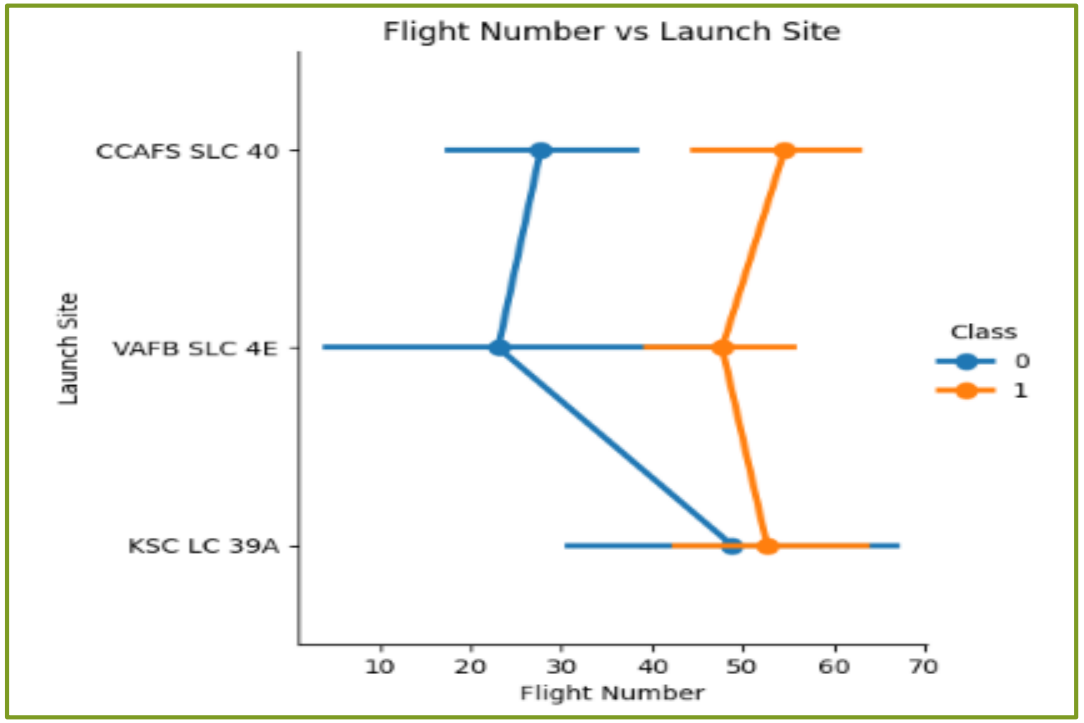
```
Done
```

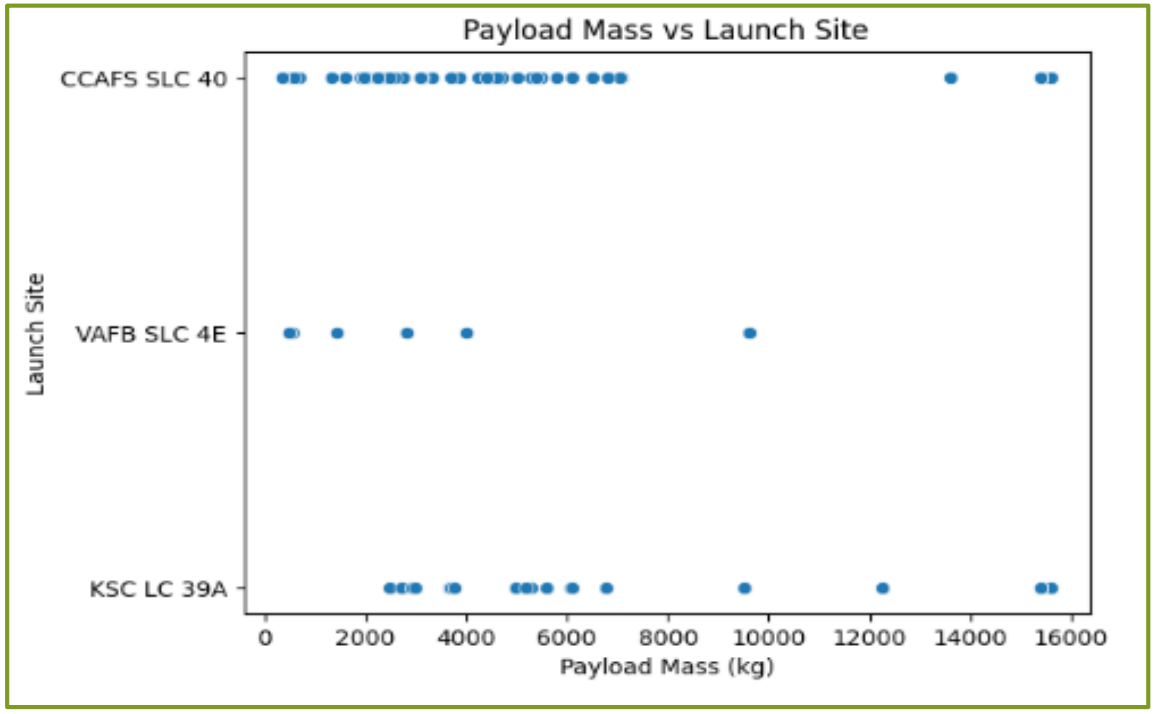
```
[7]: # Apply value_counts on Orbit column
     df['Orbit'].value_counts()
```

```
[7]: GTO      27
     ISS      21
     VLEO     14
     PO        9
     LEO       7
     SSO       5
     MEO       3
     ES-L1     1
     HEO       1
     SO        1
     GEO       1
     Name: Orbit, dtype: int64
```

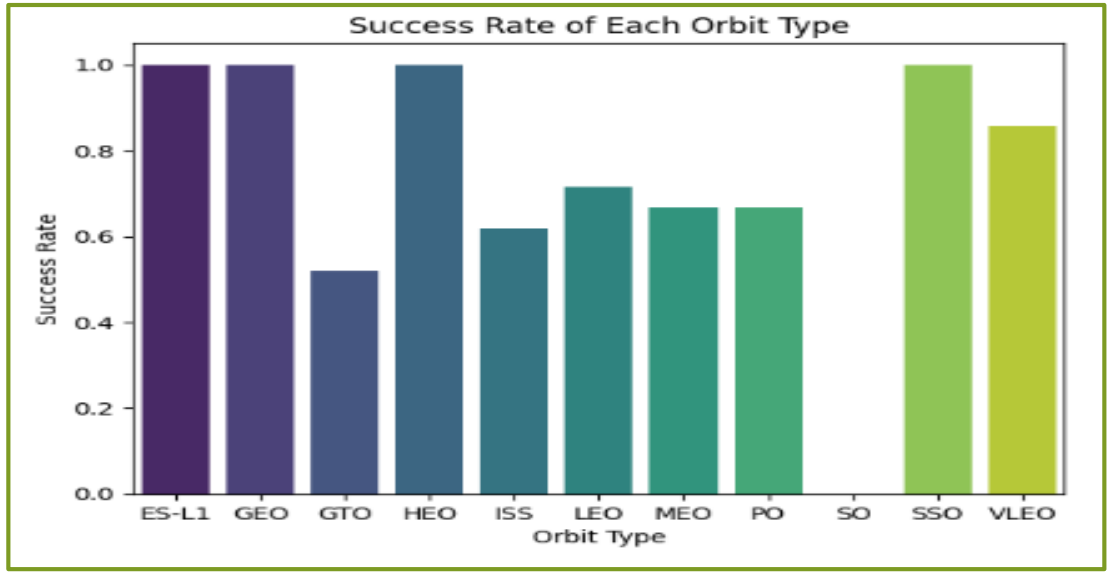
EDA with Data Visualization

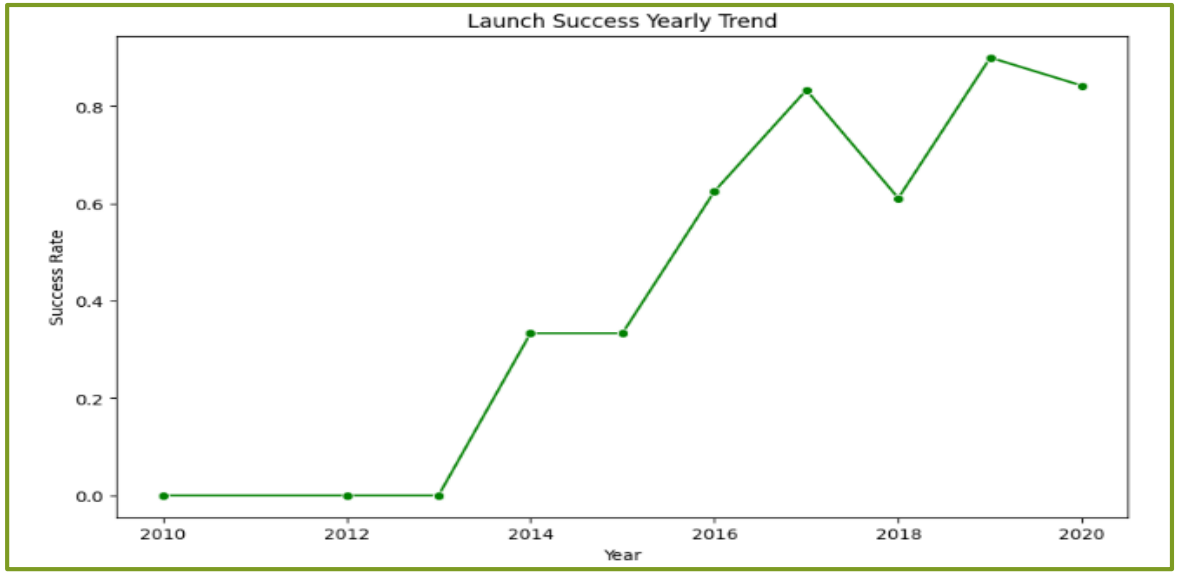
- We visualized the relationship between flight number and launch site, payload and launch site, success rate and orbit, number of flights and orbit, payload mass and orbit, and launch success yearly trend.





EDA with Data Visualization





GitHub Link : https://github.com/Nakul-Pant-DS/IBM-Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

Following SQL queries performed in the data frame :

- 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 (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved.
- List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster versions which have carried the maximum payload mass.
- List the records which will display the month names, failure landing outcomes in drone ship booster versions, launch site for the months in year 2015.
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.
- **GIT Hub Link:**
- **https://github.com/Nakul-Pant-DS/IBM-Data-Science-Capstone-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb**

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 the launch site and its proximities like railways, highways, and cities.

These markers were used to get a better visualization and understanding of the data through geography.

GitHub Link : https://github.com/Nakul-Pant-DS/IBM-Data-Science-Capstone-Project/blob/main/lab_jupyter_launch_site_location.jupyterlite.ipynb

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.

GitHub Link: https://github.com/Nakul-Pant-DS/IBM-Data-Science-Capstone-Project/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- We split the dataset into training and testing set.
- We built different machine learning models and tuned different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, and improved the model using feature engineering and hyperparameter tuning.
- We found the best performing classification model.
- **GIT Hub Link :** https://github.com/Nakul-Pant-DS/IBM-Data-Science-Capstone-Project/blob/main/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Predictive Analysis (Classification)


```
# Extract the 'Class' column and convert it to a NumPy array
Y = data['Class'].to_numpy()
# Ensure 'Y' is a Pandas series
Y = pd.Series(Y)
# Print the result
print(Y)
```

```
from sklearn.preprocessing import StandardScaler
# Standardize the data using StandardScaler
scaler = StandardScaler()
X_standardized = scaler.fit_transform(X)

# Reassign the standardized data to the variable 'X'
X = pd.DataFrame(X_standardized, columns=X.columns)
```

```
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
# Print the best parameters
print("Best Parameters:", logreg_cv.best_params_)

# Use the best model to make predictions on the test set
Y_pred = logreg_cv.predict(X_test)
```

Results

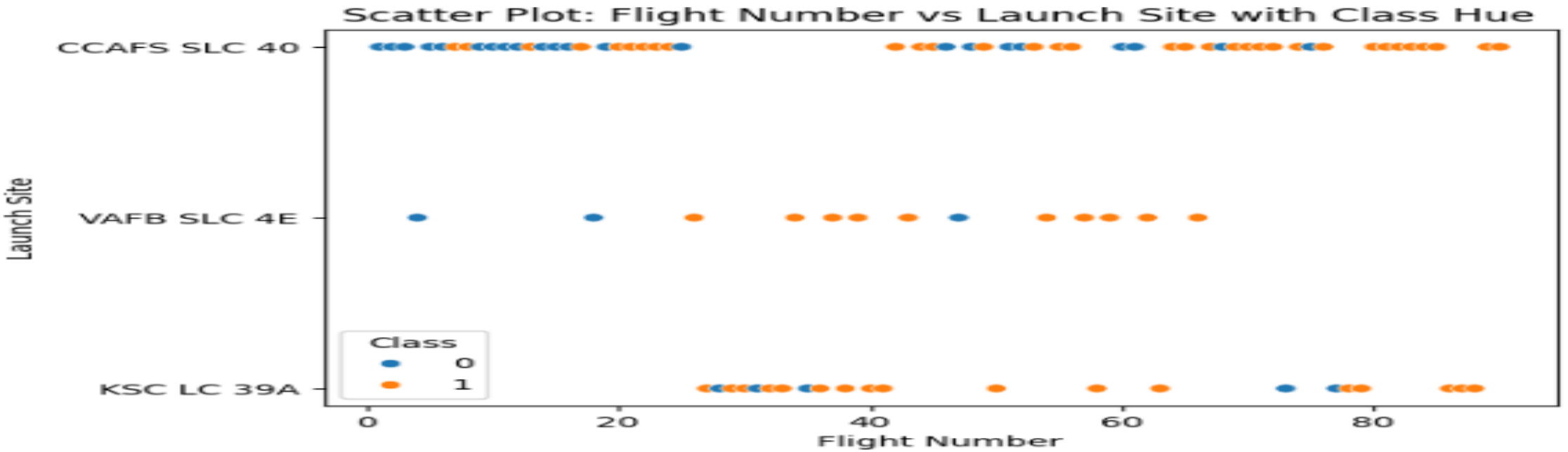
- **Exploratory data analysis results :** The results were shows with screenshots from lab.
- **Interactive analytics demo in screenshots :** This include SQL query results as performed in lab.
- **Predictive analysis results :** This include model score and confusion matrix.

Section 2 : Insights Drawn from EDA

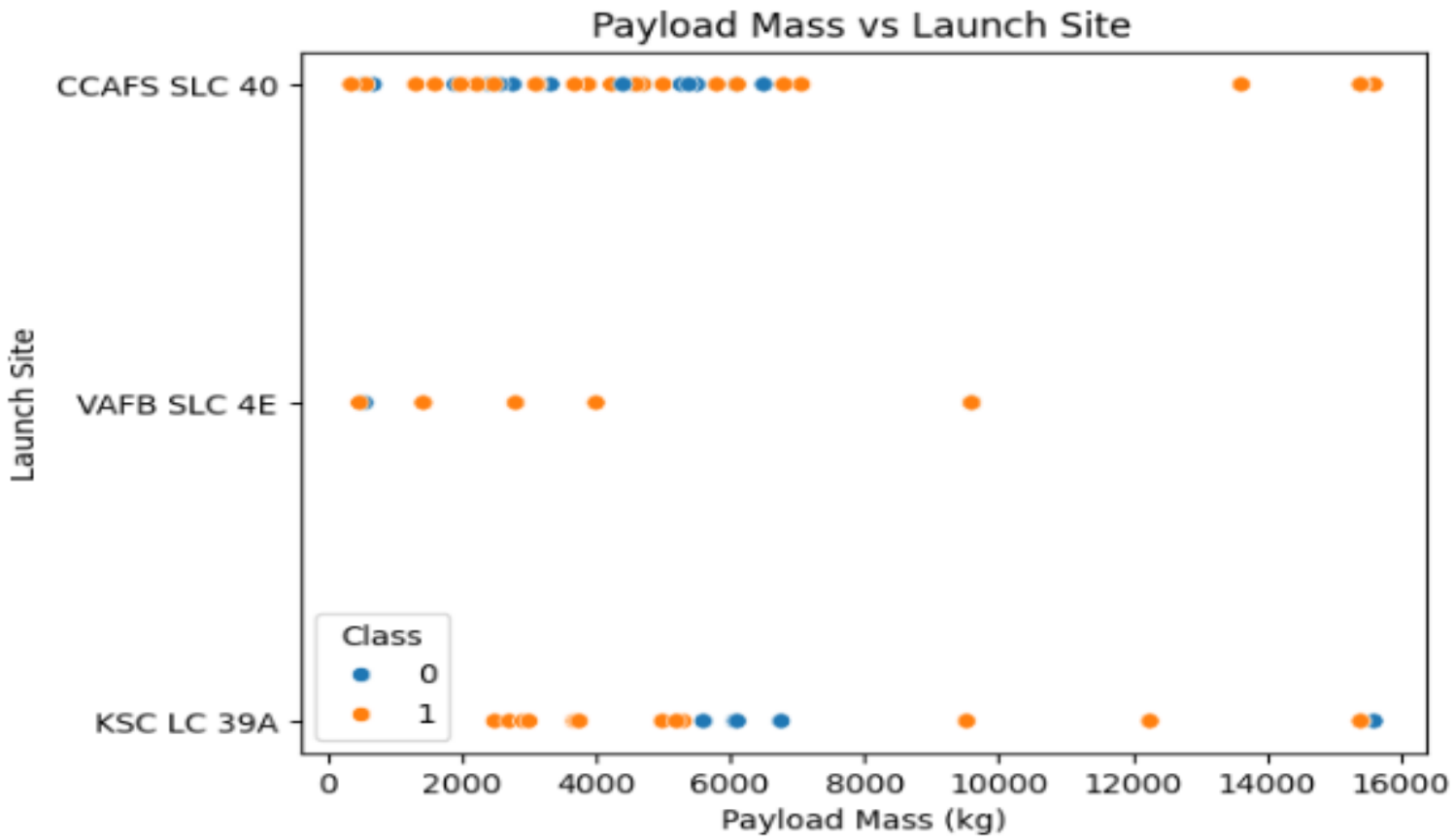
Flight Number vs. Launch Site

- Scatter plot of Flight Number vs. Launch Site

From the scatter plot of Flight Number vs. Launch Site we see that greater the number of flights at a launch site, greater is the success rate at the launch site.

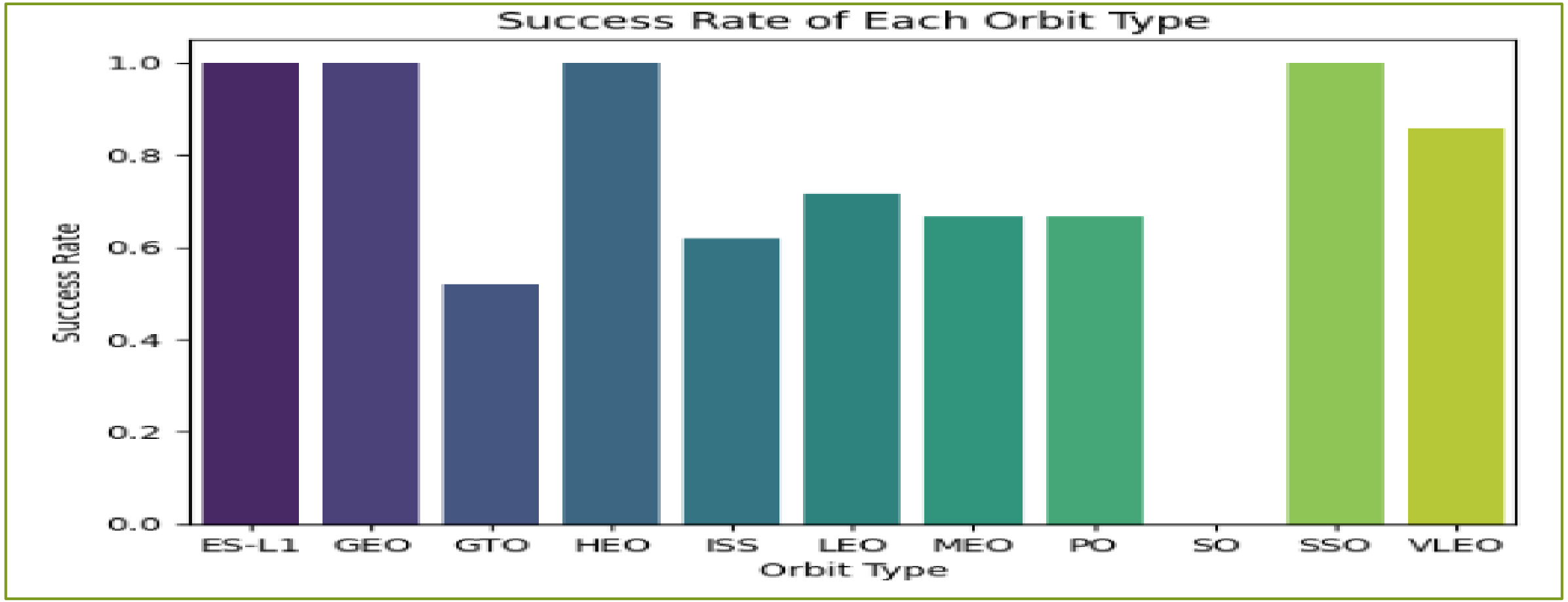


Payload vs. Launch Site

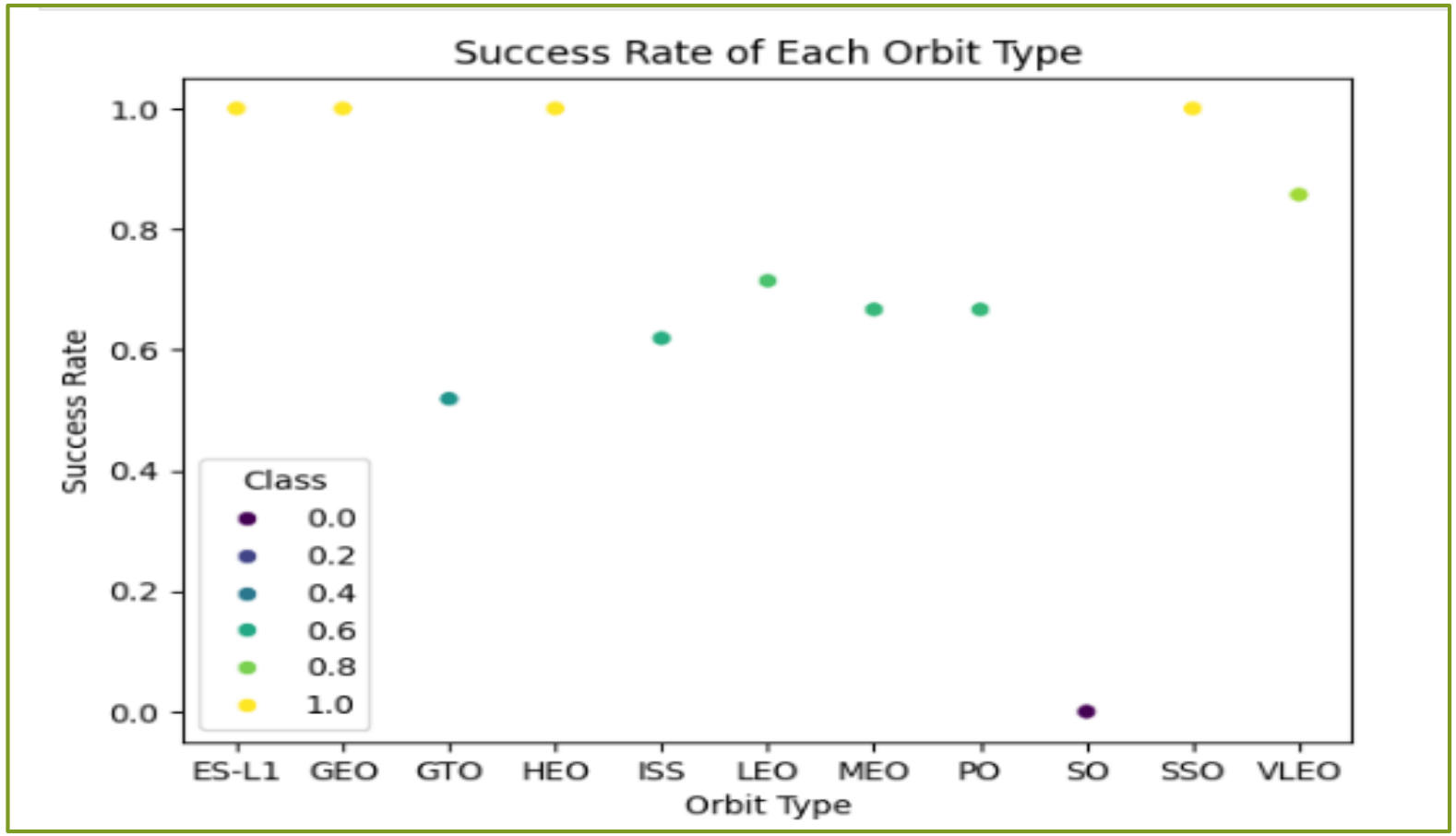


We observed that for Launch Site CCAFS SLC 40 and VAFB SLC 4E there is a linear relationship between payload mass i.e. as the payload mass increases the rate of successful launch also increases but for launch site KSC LC 39A there is no such relation with payload mass.

Success Rate vs. Orbit Type



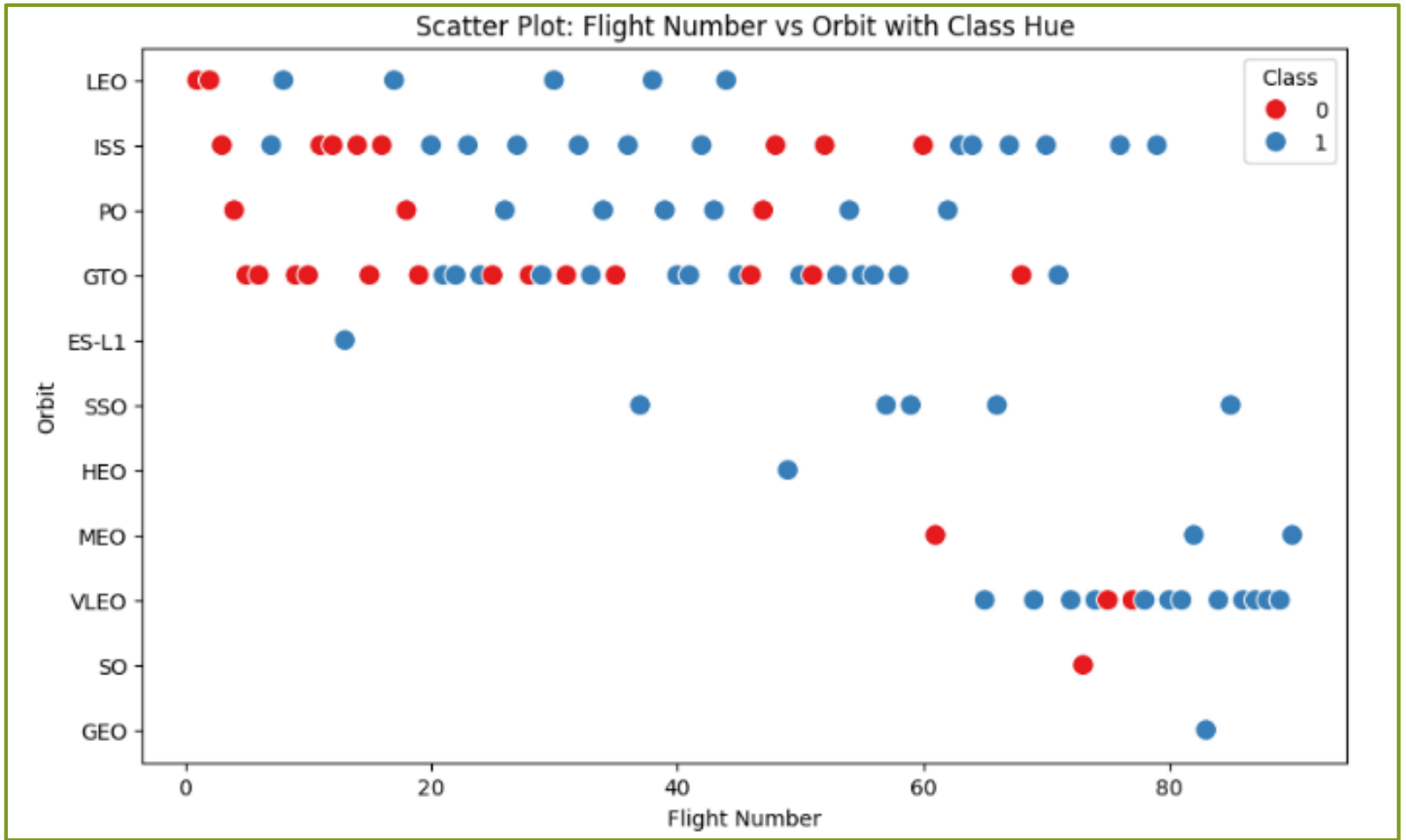
Success Rate vs. Orbit Type



We observe that only 4 orbit type achieved the Success:

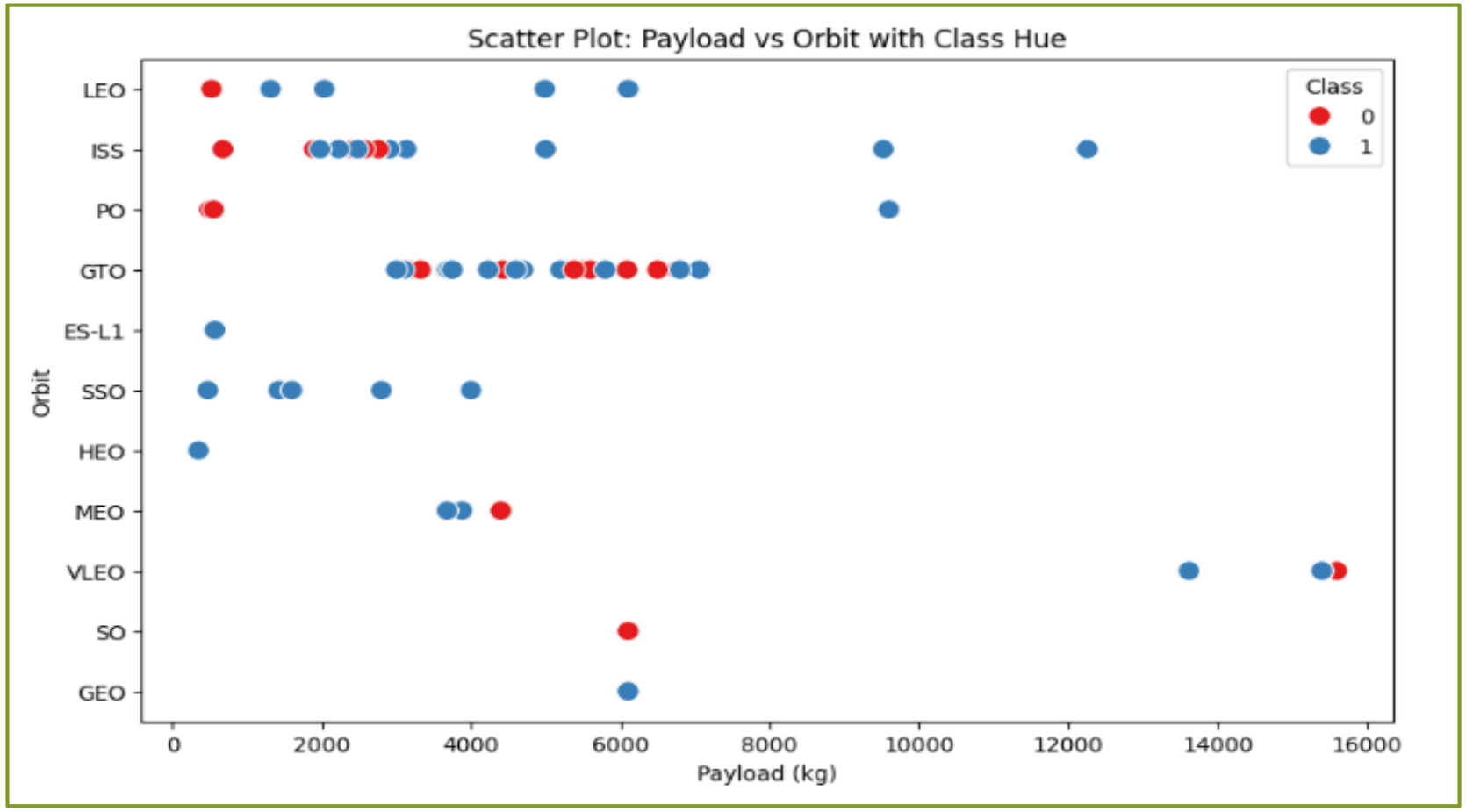
- ES-L1
- GEO
- HEO
- SSO

Flight Number vs. Orbit Type



We can clearly see that in the 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.

Payload vs. Orbit Type



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.

Launch Success Yearly Trend

We can observe that the success rate since 2013 kept increasing till 2020

All Launch Site Names


```
[14]: %sql select Distinct(Launch_Site) from SPACEXTABLE;
 * sqlite:///my_data1.db
Done.
```

[14]:

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

Here we an see that there are 4 unique launch sites :

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

We used the DISTINCT keyword to find all unique launch sites.

Launch Site Names Begin with 'CCA'


```
%sql select * from SPACEXTABLE where launch_Site like 'CCA%' limit 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- The keyword LIKE is used to specify that the launch site name begins with CCA.
- The limit keyword is used to query only 5 records.

Total Payload Mass


```
[32]: %%sql select SUM(PAYLOAD_MASS__KG_) as Total_Payload_Mass from SPACEXTABLE
      where Customer =  'NASA (CRS)';

       * sqlite:///my_data1.db
      Done.

[32]: Total_Payload_Mass
      ----------------
                 45596
```

- The total payload mass for customer NASA (CSR) is 45,596 kg.
- The function SUM is used to obtain the sum of payload mass.
- The WHERE clause is used to specify that the customer should be NASA (CRS).

Average Payload Mass by F9 v1.1


```
[30]: %%sql select AVG(PAYLOAD_MASS__KG_) as Average_Payload_Mass from  SPACEXTABLE
      where  Booster_Version = 'F9 v1.1';

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

[30]:

Average_Payload_Mass
2928.4

- The average payload mass for rockets with booster version F9 v1.1 is 2928.4 kg.
- The function AVG is used to calculate the average of payload mass.
- The WHERE clause is used to specify that the booster version should be F9 v1.1

First Successful Ground Landing Date


```
[34]: %sql select min(Date) from SPACEXTABLE where Landing_Outcome = 'Success (ground pad)';
 * sqlite:///my_data1.db
Done.
[34]: min(Date)
      2015-12-22
```

- The first successful ground landing occurred on 22 December 2015.
- The function MIN is used to find the earliest date.
- The WHERE clause is used to specify that the landing outcome is a successful ground pad landing.

Successful Drone Ship Landing with Payload between 4000 and 6000

[39]:
```
%%sql select Booster_Version,LANDING_OUTCOME, PAYLOAD_MASS__KG_ from  SPACEXTABLE
      where LANDING_OUTCOME='Success (drone ship)'
      and  PAYLOAD_MASS__KG_ between 4000 and 6000;
```

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

[39]:

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

- The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 – F9 FT B1022, F9 FT B1026, F9 FT B1021.2, F9 FT B1031.
- The WHERE clause specifies that the landing was a success drone ship landing.
- The AND clause adds another condition that the payload mass is between 4000 and 600.

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
[8]: %%sql select count(Landing_Outcome) as Total_Number_Of_Successful_Mission_Outcomes from SPACEXTABLE where
       Mission_Outcome LIKE 'Success%';
```

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

[8]:

Total_Number_Of_Successful_Mission_Outcomes
100

```
[10]: %%sql select count(Landing_Outcome) as Total_Number_Of_Failure_Mission_Outcomes from SPACEXTABLE where
       Mission_Outcome LIKE 'Failure%';
```

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

[10]:

Total_Number_Of_Failure_Mission_Outcomes
1

- Total Number of Successful Mission Outcomes –100
- Total Number of Failure Mission Outcomes – 1.
- The COUNT function gives the number of landings.
- The LIKE clause is used to specify that the outcome is a success or a failure.

Boosters Carried Maximum Payload


```
%%sql select Booster_Version, PAYLOAD_MASS__KG_ as MAX_PAYLOAD_MASS from  SPACEXTABLE
      where  PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) from SPACEXTABLE )
       LIMIT 5;
```

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

Booster_Version	MAX_PAYLOAD_MASS
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

- These are the TOP 5 Booster Versions with Max Payload Mass:

1. F9 B5 B1048.4
2. F9 B4 B10494
3. F9 B5 B1056.4
4. F9 B5 B1056.4
5. F9 B5 B1048.5

2015 Launch Records


```
[59]: %%sql  select substr(Date,6,2) as Month_Name , Landing_Outcome, Booster_Version, Launch_Site from SPACEXTABLE
      where Landing_Outcome = 'Failure (drone ship)' and Date LIKE '2015%';
```

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

[59]:

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

- The SUBSTR function selects the year out of the date.
- The WHERE clause specifies that the date should be 2015 and the landing outcome should be a drone ship failure.

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


```
%%sql SELECT Landing_Outcome, COUNT(*) AS Landing_Outcome_Count
  FROM SPACEXTABLE
  WHERE date BETWEEN '2010-06-04' AND '2017-03-20'
  GROUP BY landing_outcome
  ORDER BY Landing_Outcome_Count DESC;
```

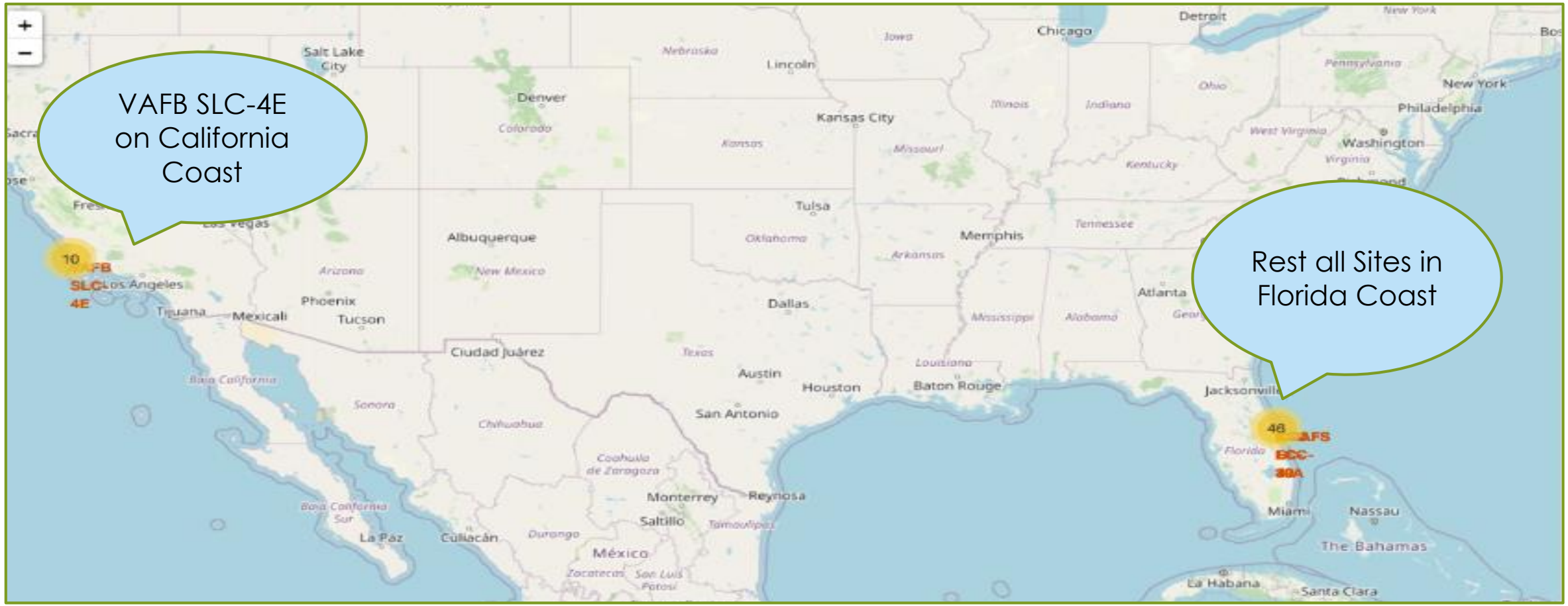
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	Landing_Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

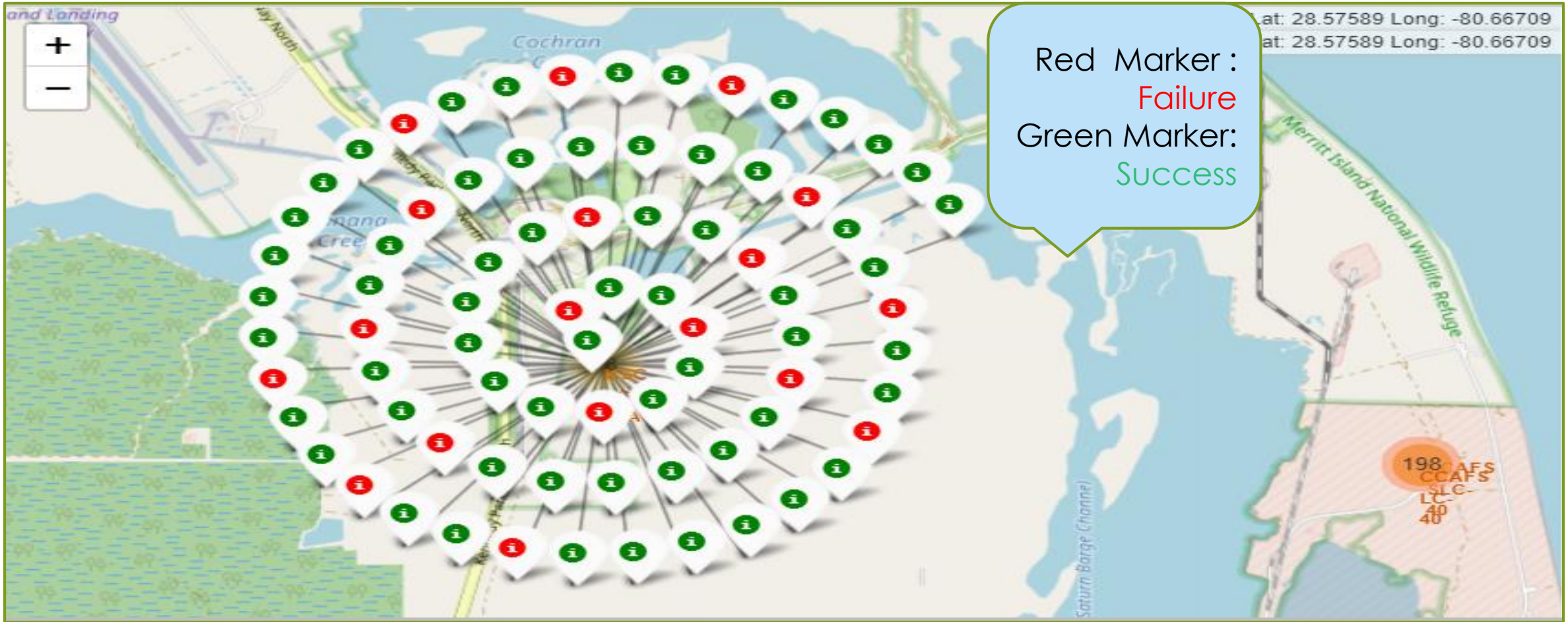
- The result shows the landing outcome with “no attempt” has the maximum count. This was too obvious.
- Where as those with Precluded(drone ship) has the minimum count.
- Otherwise there is a close count between other Landing outcome types.

Section 3 : Proximity Analysis

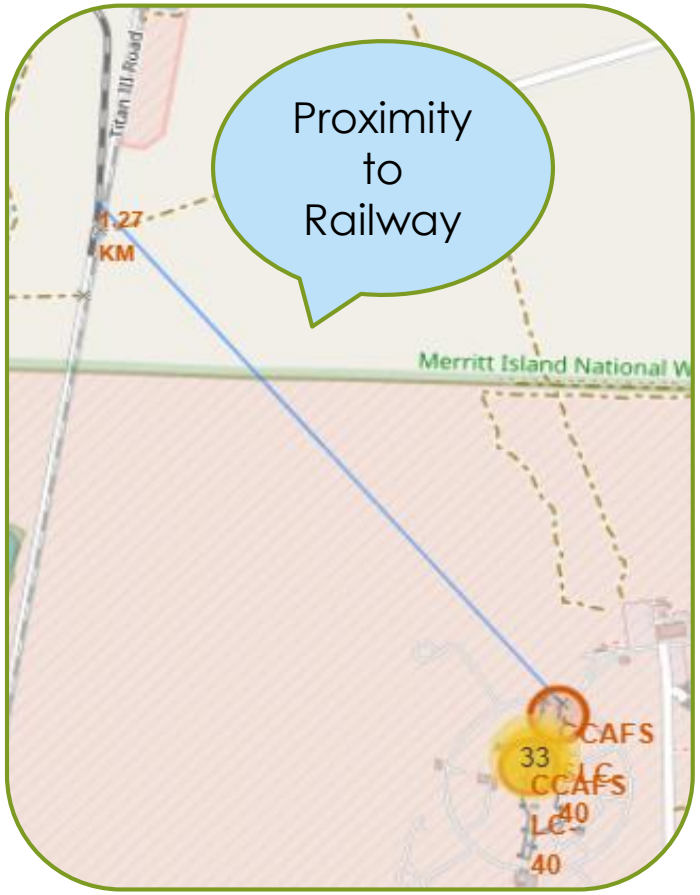
SPACE X : Launch Sites Map

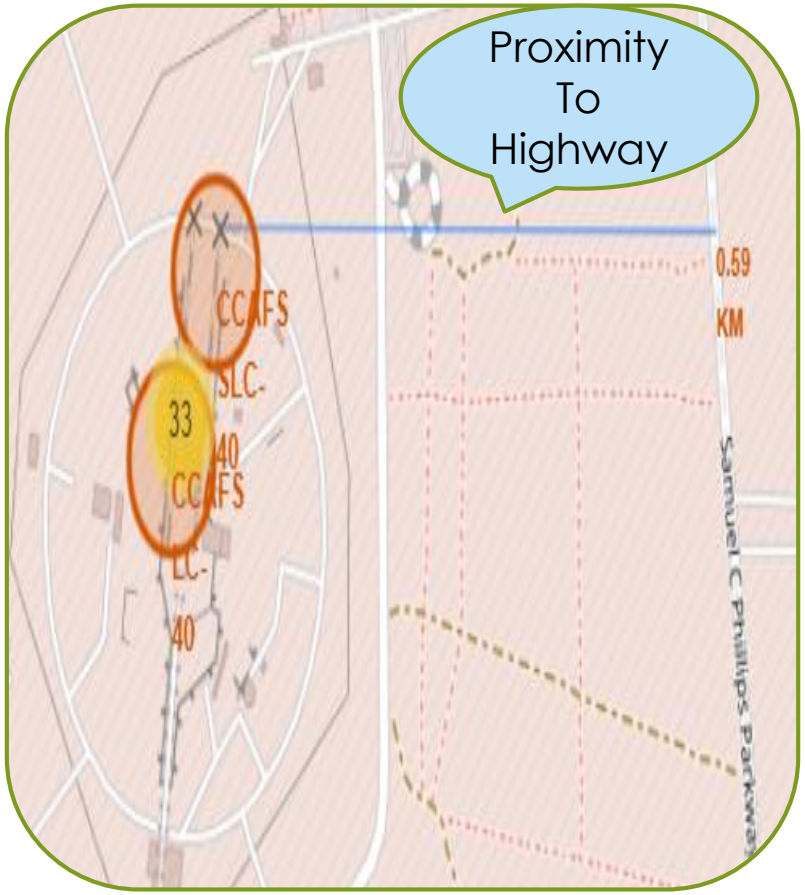


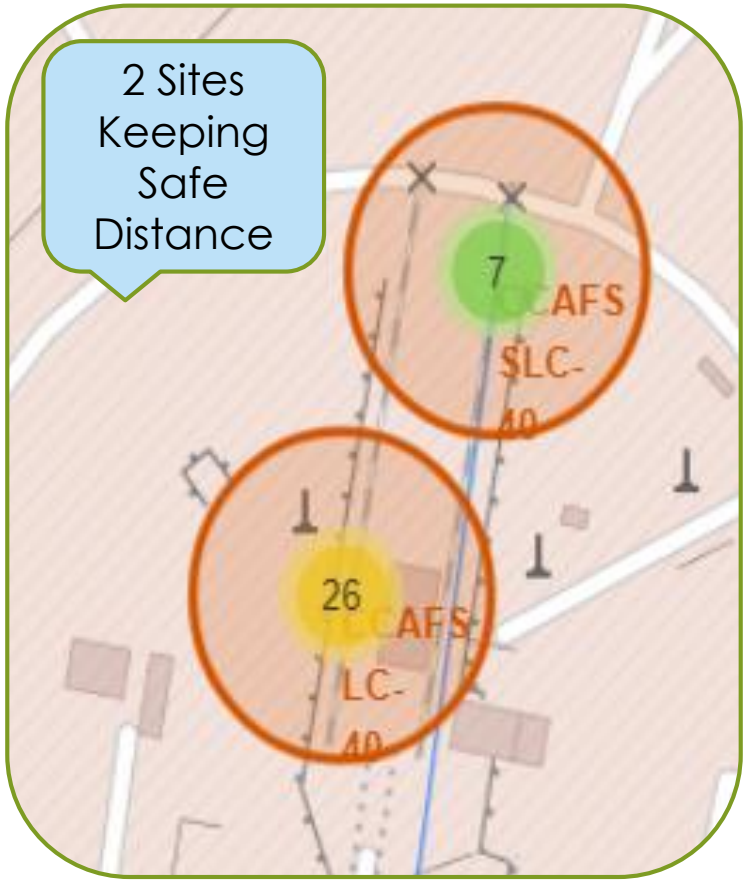
Success/Failures At Specific Launch Site



Infrastructure Near Launch Sites

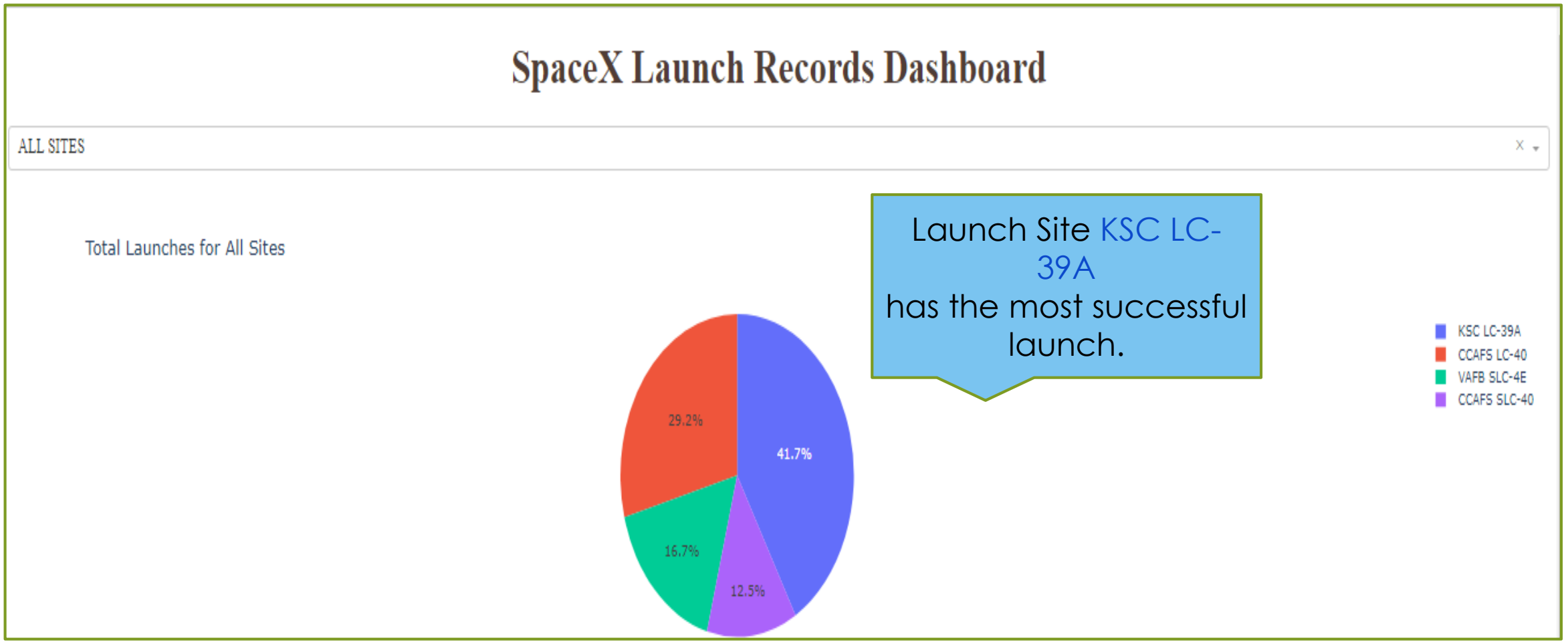






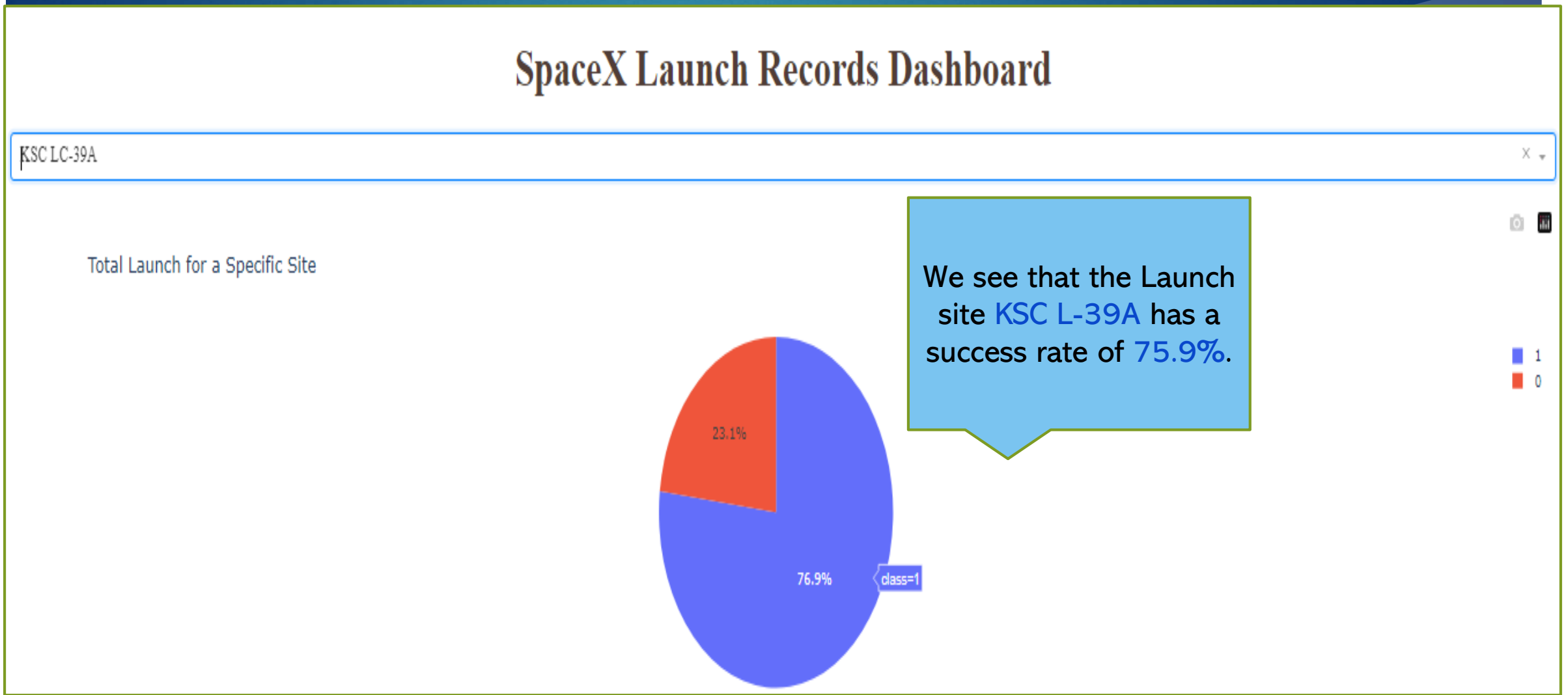
Section 4 : Dashboard

Total Success Launches By All Sites

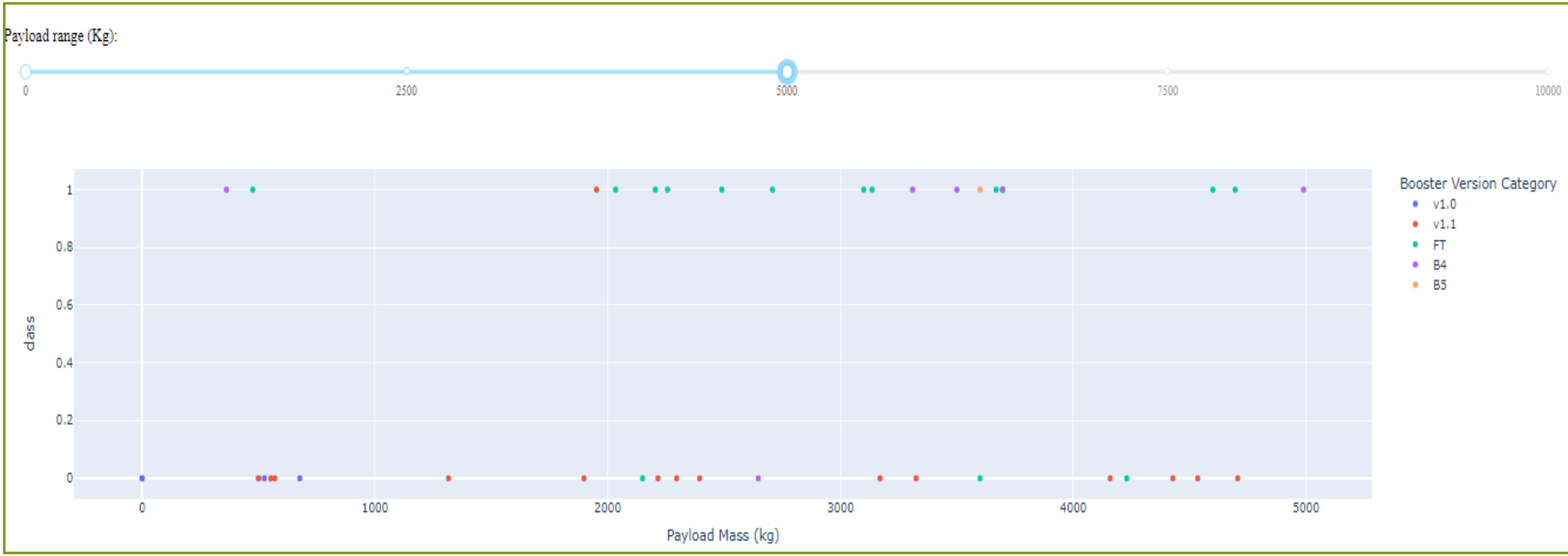


Launch Site KSC LC-39A has the most successful launch.

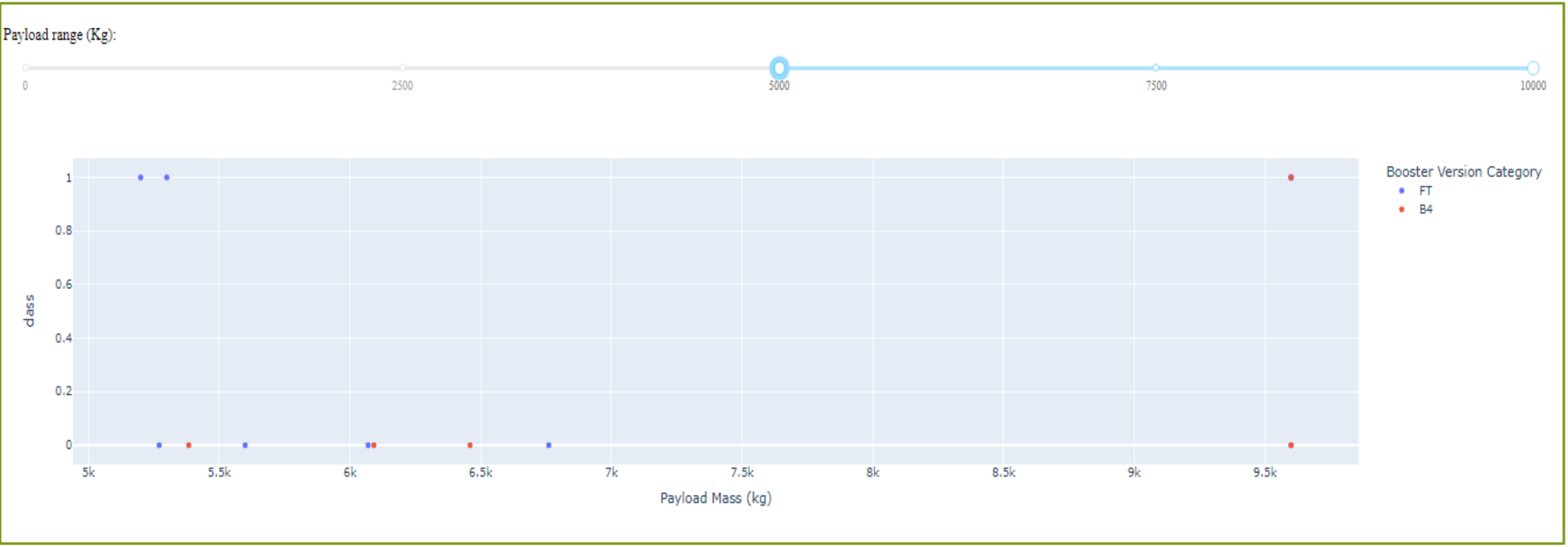
Launch Site With Highest Launch Success Ratio



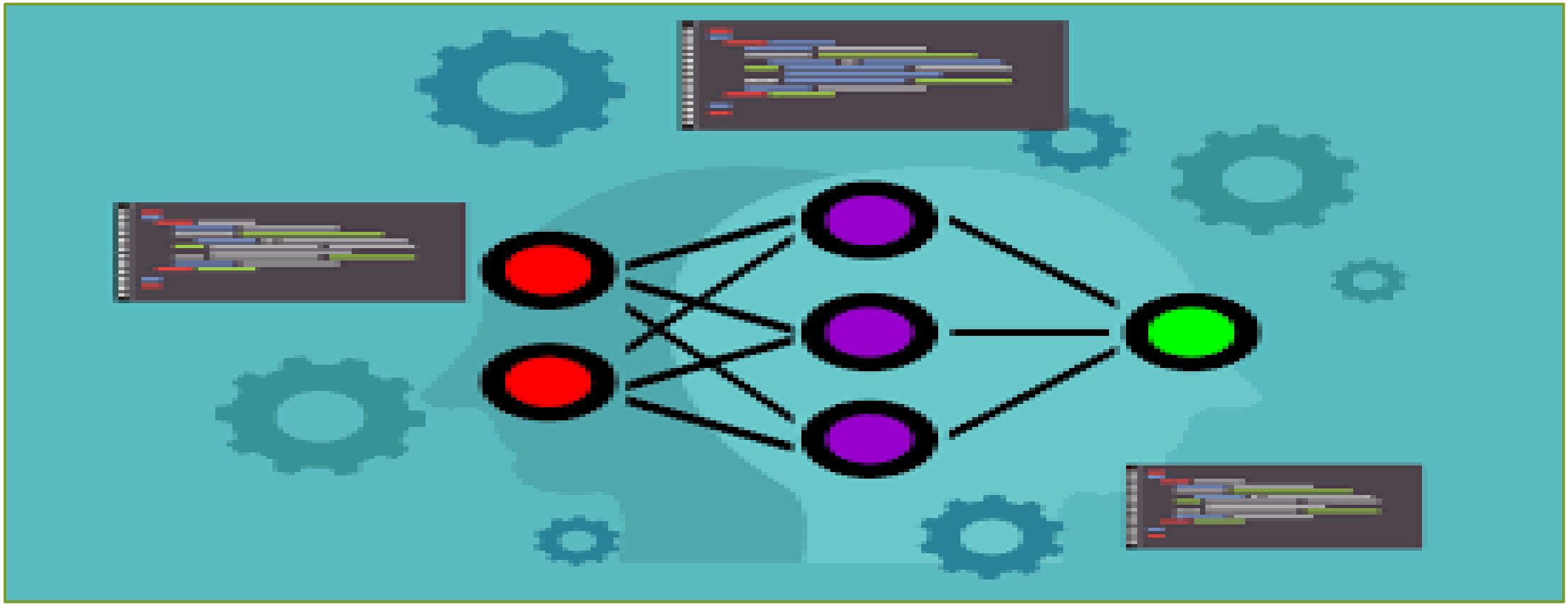
Launch Outcome of Payloads < 5000



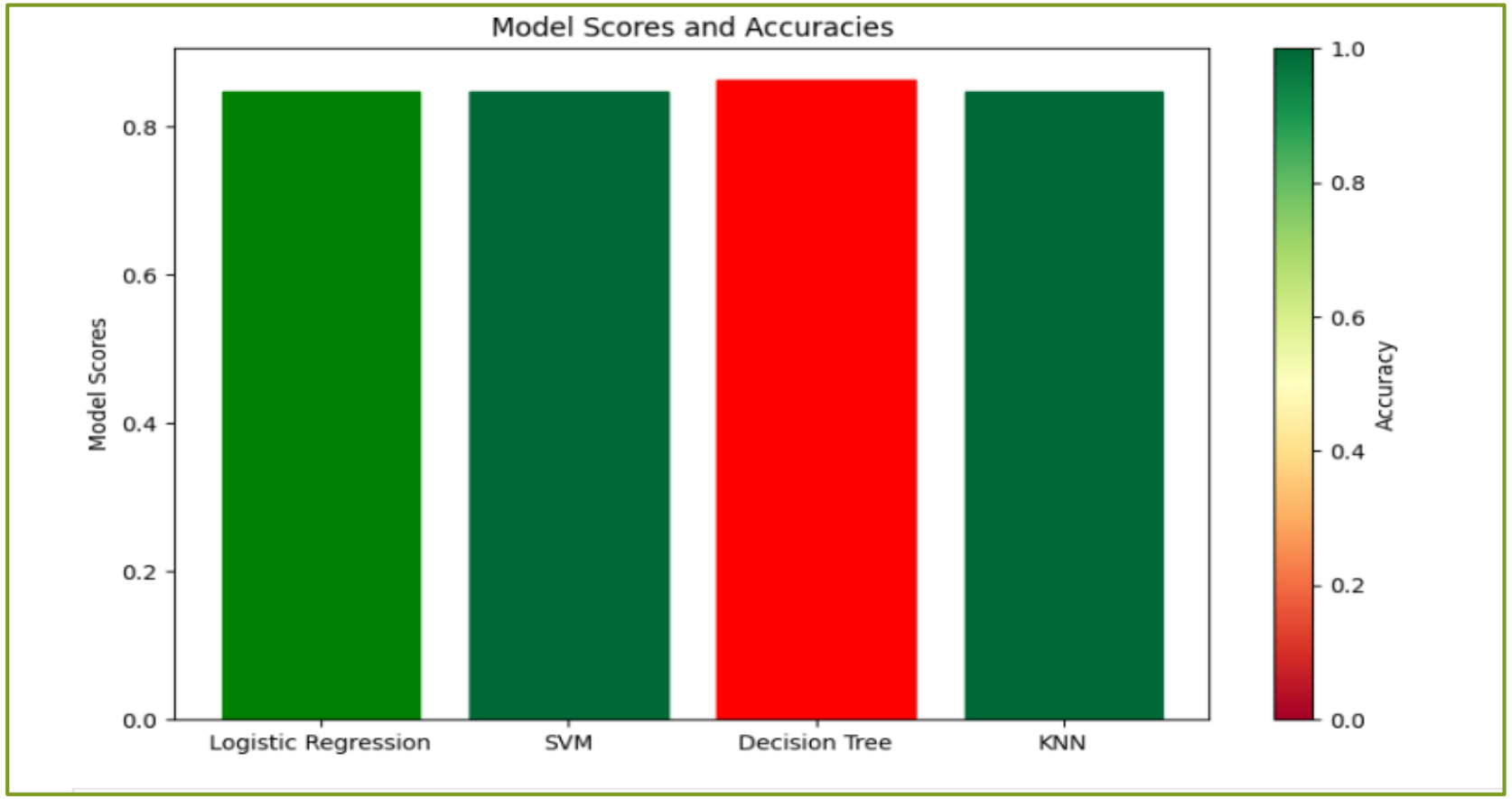
Launch Outcome of PayLoad Between 5000-10,000



Predictive Analysis (Classification)

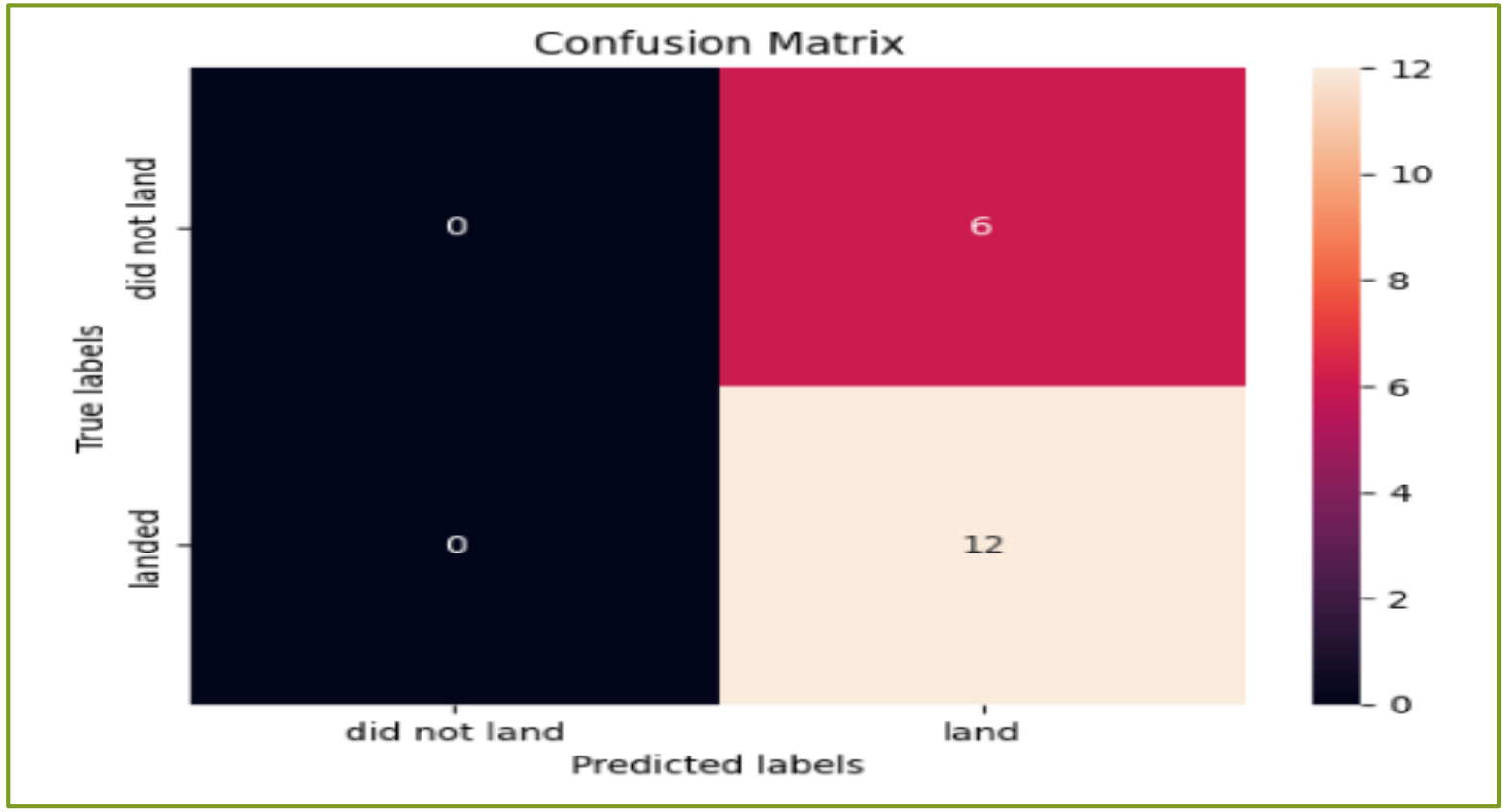


Classification Accuracy



- The bar chart show model with maximum accuracy in red while with low accuracy in green.
- The Decision Tree gives the maximum accuracy.

Confusion Matrix



- The confusion matrix for the Decision Tree Classifier shows that the classifiers has done a good job in predicting12 successful landing while the classifier shows 6 landings that were not supposed to land but it predicted otherwise .

Conclusions

- The greater the number of flights at a launch site, greater is the success rate at the launch site.
- ES-L1, GEO, HEO, and SSO orbits have the highest success rate.
- For heavier payloads, success rate increases for LEO, ISS and Polar orbits.
- Launch success rate increased from 2013 to 2020 with minor fluctuations throughout the period.
- Launch Site KDC LC-39 A has the Greatest Launch Success Rate.
- The Decision Tree Classifier is the best machine learning algorithm to predict SpaceX Falcon 9 rocket’s first stage will land successfully.

Appendix

- **Data Dictionary:** The following features were used to build the model:
  BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad | Block | ReusedCount | Serial | Longitude | Latitude | Class
- **References and Citations:** Getting Started with Data Science: Making Sense of Data with Analytics by Murtaza Haider
- **Supplementary Analyses:**

We observed that the accuracy of all the models are almost similar but still the Decision Tree Algorithm gives a the maximum model score of ~ 87% this is because Decision trees are inherently non-linear models and can capture non-linear relationships in the data more effectively than linear models like logistic regression.

Thank You!!!