IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Emmanuel Obot
12th July, 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection
  - Data wrangling
  - EDA with data visualization
  - EDA with SQL
  - Building an interactive map with Folium
  - Building a Dashboard with Plotly Dash
  - Predictive analysis (Classification)
- Summary of all results
  - EDA results
  - Interactive analytics
  - Predictive analytics

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.
- **Problems you want to find answers**
  - Our task in this project is to determine if the first stage of the SpacesX Falcon 9 Rocket will land successfully.

Section 1

Methodology

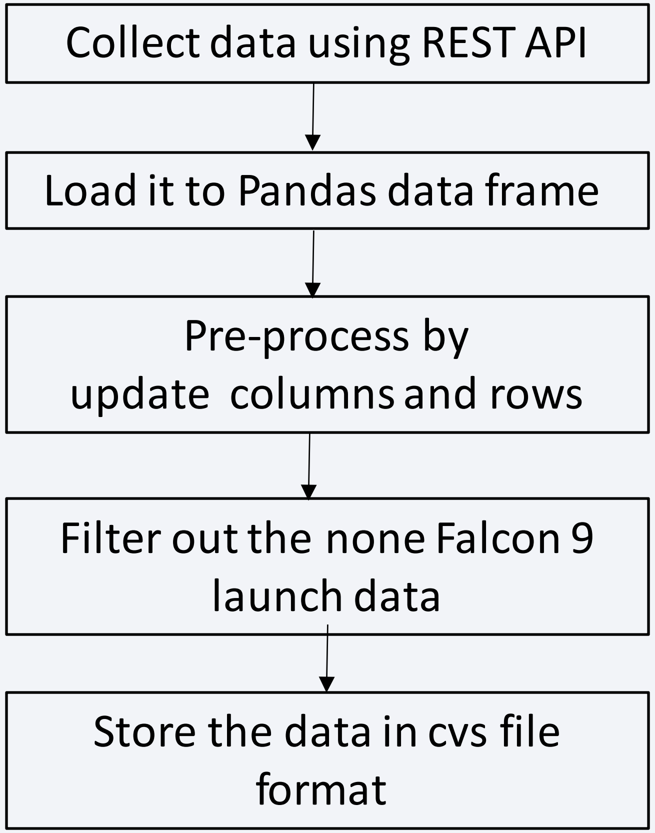
Methodology

Executive Summary

- Data collection methodology:
  - SpaceX Rest API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Using One Hot Encoding to prepare the data field for Machine learning
  - Cleaning the data of null values and remove irrelevant column
  - Convert the outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

Data Collection

- The following datasets were collected:
  - The SpaceX Launch Data was gather from the SpaceX REST API
  - The REST API give us data about launches, including information about the rocket used, payload, launch data, landing data and landing outcome.
  - The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
  - The Falcon9 launch data was web scrapped from wikipedia using the BeautifulSoup module



Data Collection – SpaceX API

1. **Collect Data using API call**

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. **Convert the Data to Pandas Data Frame using JSON**

```
data = pd.json_normalize(response.json())
# Get the head of the dataframe
data.head()
```

	static_fire_date_utc	static_fire_date_unix	net	window	
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f

3. **Pre-process by update columns and rows**

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]

# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket boosters and rows that h
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the single value in the list and replace the featur
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then extracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

4. **Filter out the none Falcon 9 launch data**

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = data[data.BoosterVersion == 'Falcon 9']
data_falcon9
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Ou
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	

5. **Store the data in cvs file format**

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

GitHub URL: https://github.com/emotexplanet/Data_Science_cap/blob/main/data-collection-api.ipynb

Data Collection - Scraping

1. Get HTML response

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

2. Create BeautifulSoup object

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
bSoup = BeautifulSoup(response, 'html.parser')
```

3. Get column names

```
col = bSoup.find_all('th')
for x in range(len(col)):
    try:
        name = extract_column_from_header(col[x])
        if name is not None and len(name) > 0:
            column_names.append(name)
```

4. Create Dictionary for Launch Data

```
launch_dict= dict.fromkeys(column_names)

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

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

5. Covert Dictionary to Data Frame

```
df=pd.DataFrame(launch_dict)
```

6. Store the data in cvs file format

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

GitHub URL: https://github.com/emotexplanet/Data_Science_cap/blob/main/webscraping.ipynb

Data Wrangling

1. Load Data Set

```
df=pd.read_csv(dataset_part_1_csv)
df.head(10)
```

2. Check null values

```
df.isnull().sum()/df.shape[0]*100
```

3. Find number of launches on each site

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

```
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
```

4. Find the number and occurrence of each orbit

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

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
```

5. Find the mean of success outcome

```
df["Class"].mean()
```

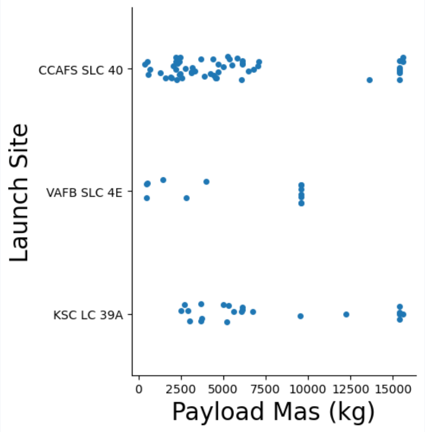
```
0.6666666666666666
```

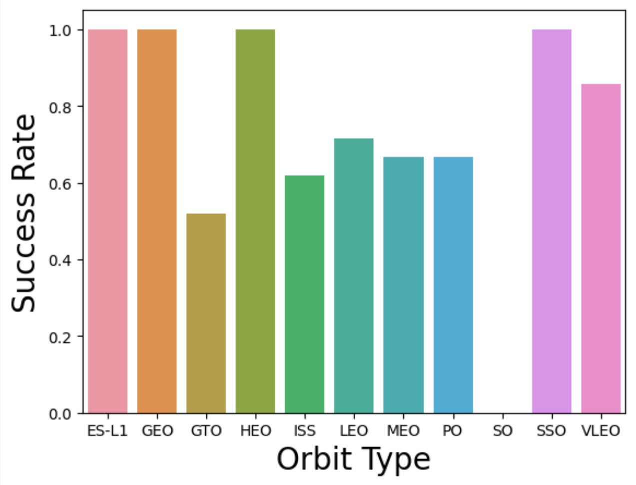
6. Store the data in cvs file format

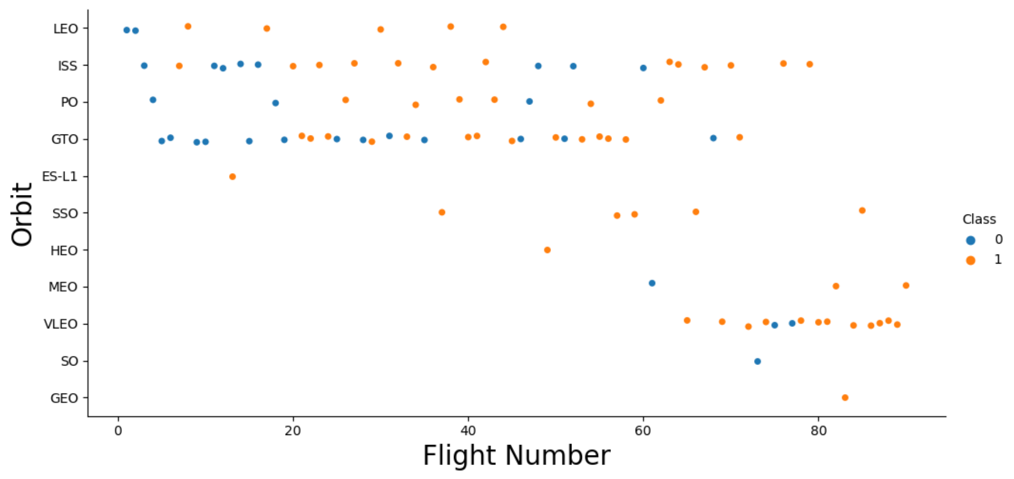
```
df.to_csv("dataset_part_2.csv", index=False)
```

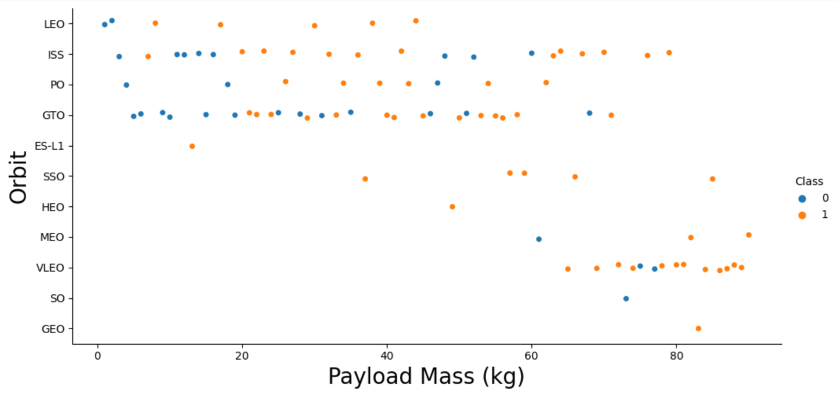
GitHub URL:
https://github.com/emotexplanet/Data_Science_cap/blob/main/data_wrangling.ipynb

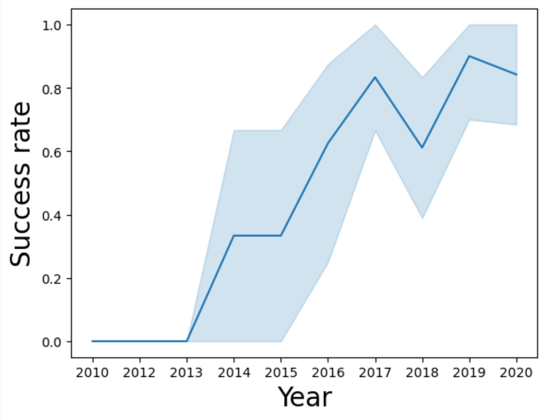
EDA with Data Visualization











GitHub URL: https://github.com/emotexplanet/Data_Science_cap/blob/main/spaceX-eda-dataviz.ipynb

EDA with SQL

The SQl queries answer the following questions:

- 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 succesful landing outcome in ground pad was acheived.
- 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. Use a subquery
- 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.

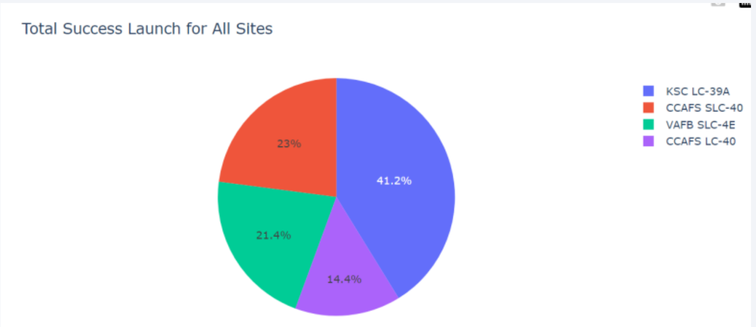
GitHub URL: https://github.com/emotexplanet/Data_Science_cap/blob/main/spaceX_eda-sql-.ipynb

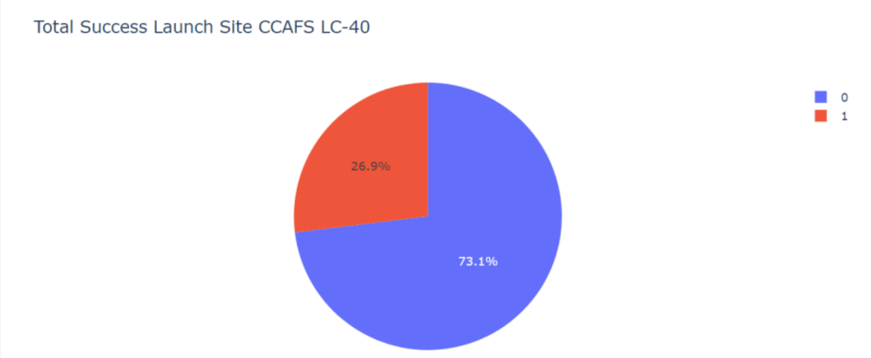
Build an Interactive Map with Folium

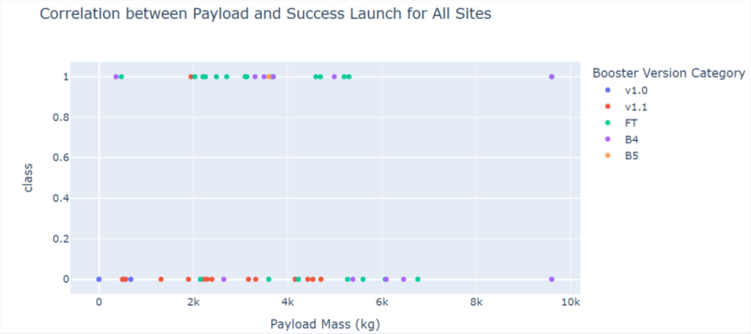
- folium.Marker() was used to marks on the maps.
- folium.Circle() was used to create a circle above markers on the map.
- folium.Icon() was used to create an icon on the map.
- folium.PolyLine() was used to create polynomial line between the points
- folium.plugin.AntPath().was used to create animated line between the points
- markerCluster() was used to simplify the maps which contain several markers with identical coordination.

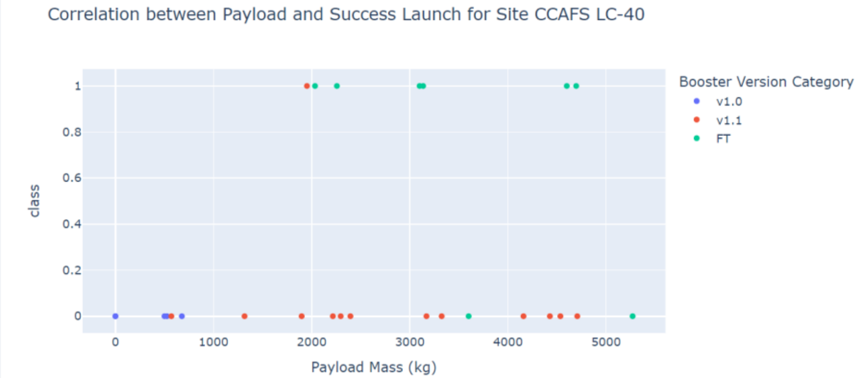
GitHub URL:
https://github.com/emotexplanet/Data_Science_cap/blob/main/launch_site_location.ipynb

Build a Dashboard with Plotly Dash









- Plotly was used in plotting the graphs
- Two types of graphs were plotting: Pie Chart and Scatter Chart
- To interact with Dashboard, a Range slider was used to select the payload mass and Dropdown was used to select the launch sites.

GitHub URL: https://github.com/emotexplanet/Data_Science_cap/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- **Building Model**
  - **Create column “class” in data**
  - **Standardize the data**
  - **Split data into train and test set**
  - Build GridSearchCV model and fit the data.
- **Evaluating the Model**
  - Calculating the accuracies.
  - Calculating the confusion matrixes
  - Plot the result
- **Finding the optimal Model**
  - Find the best hyperparamters for models
  - . Confirm the optimal model

GitHub URL:
https://github.com/emotexplanet/Data_Science_cap/blob/main/SpaceX_Machine_Learning_Prediction.ipynb

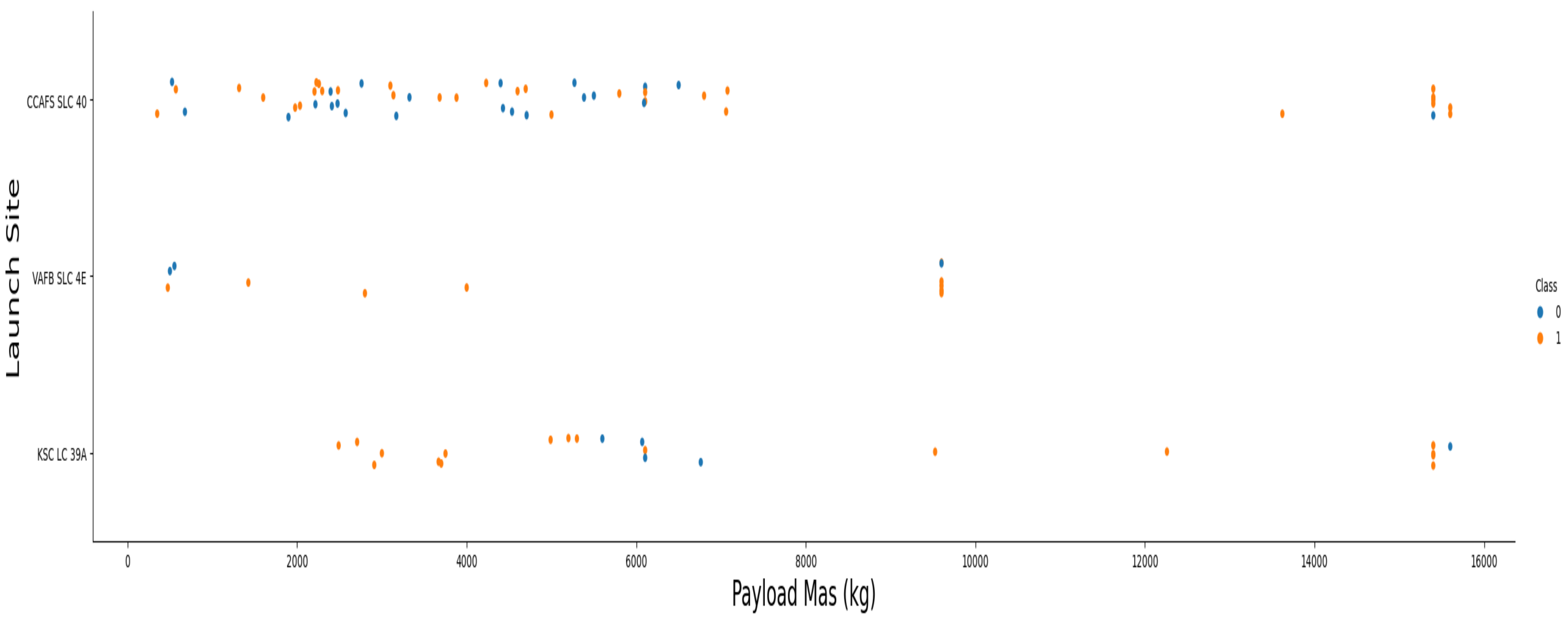
Results

- The success rate for SpaceX launch is directly proportional to the numbers of years they launch.
- KSC LC 39A had most successful launches of all the sites.
- SVM, KNN and Logistic Regression models are the best in predicting accuracy of this SpaceX dataset.

Section 2

Insights drawn from EDA

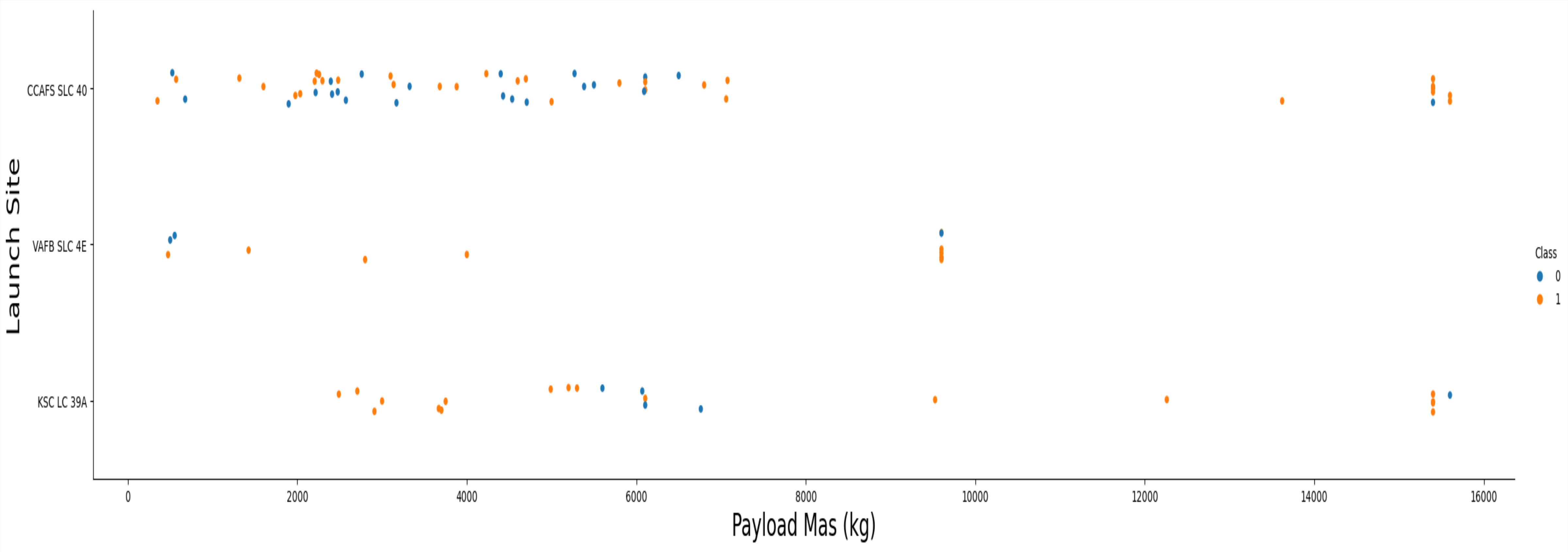
Flight Number vs. Launch Site



- As the flight number increase, the success rate increase in the launch site

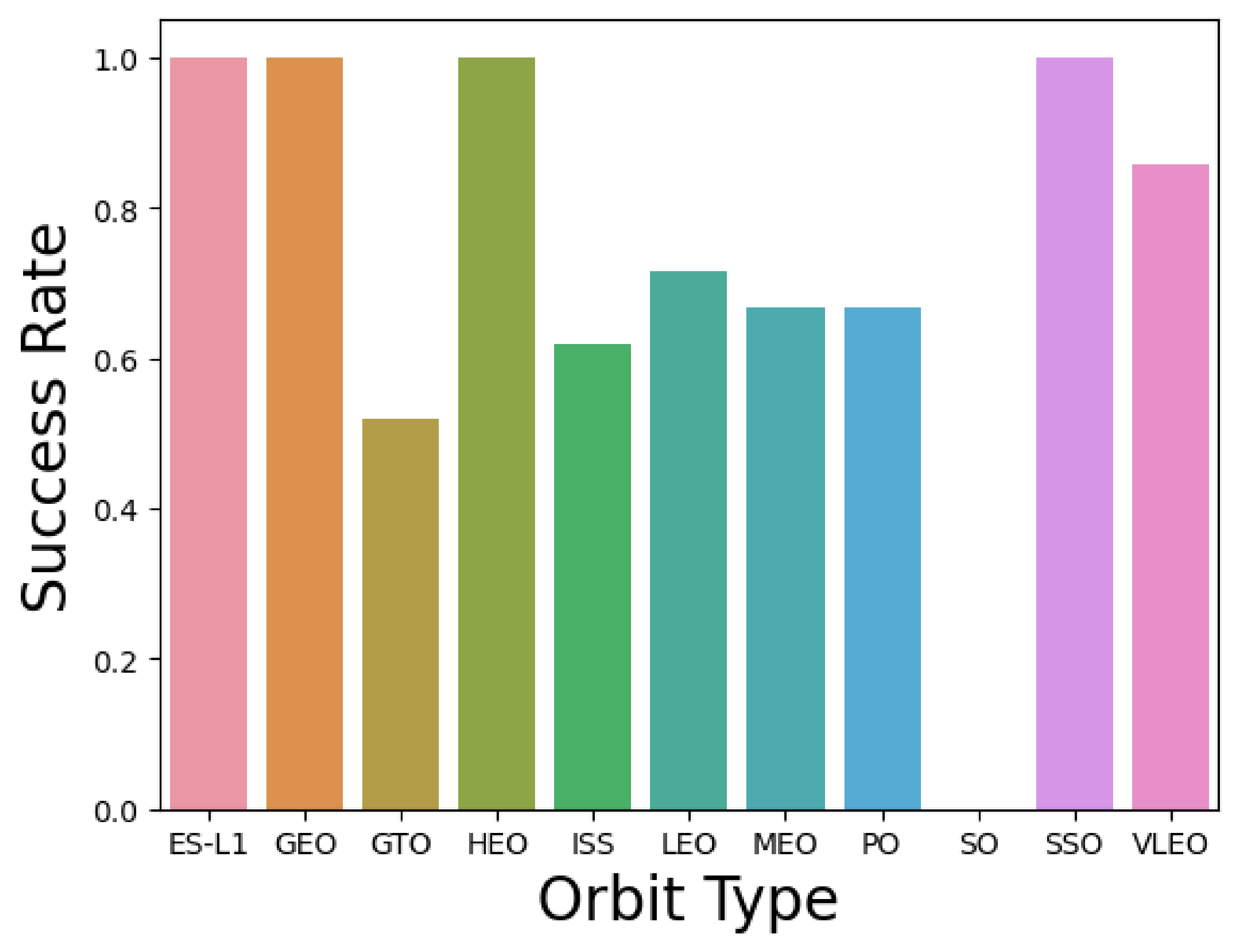
Payload vs. Launch Site

- As the Payload Mass increase, the success rate increase in the launch site.

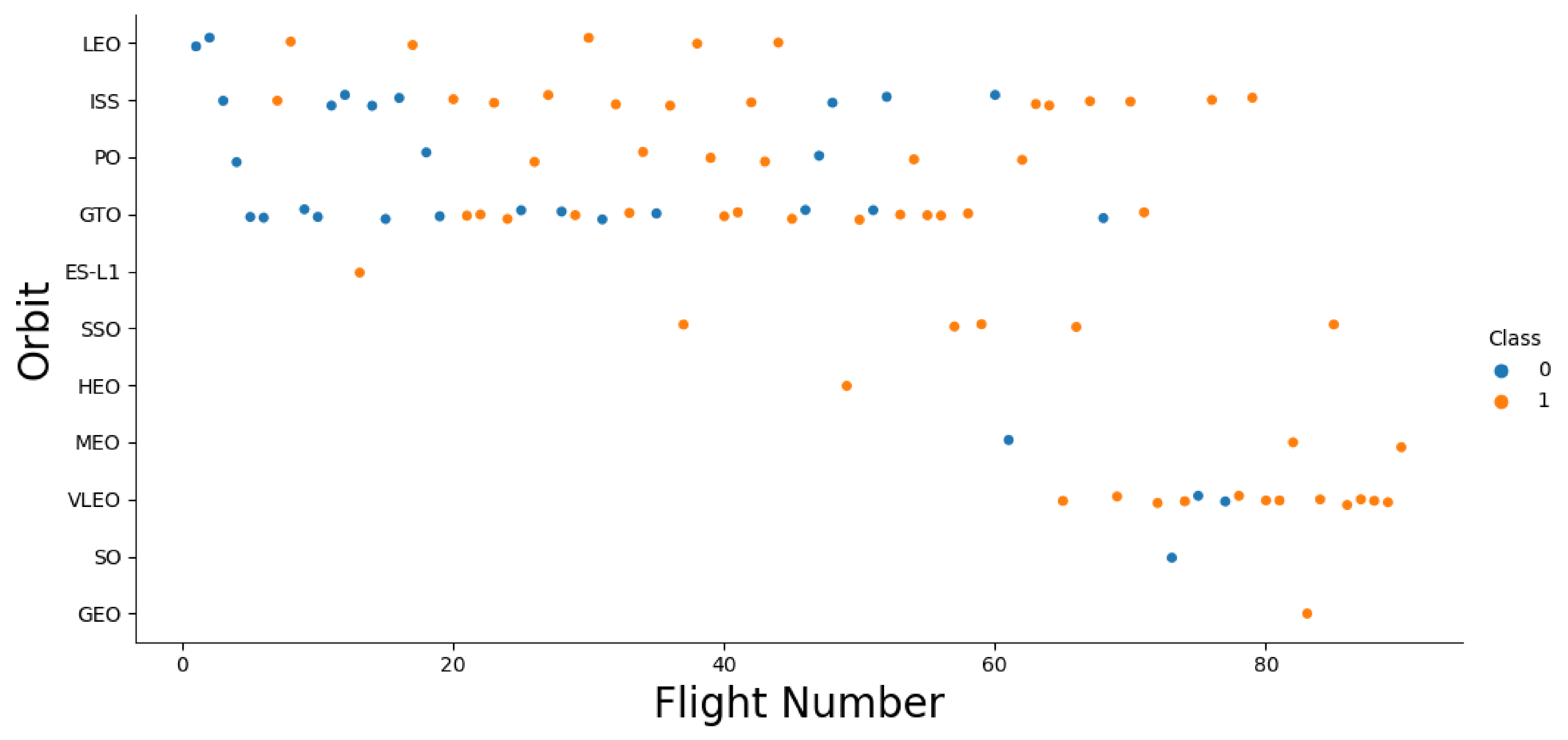


Success Rate vs. Orbit Type

- ES-l1, GEO, HEO and SSO have a success rate of 100% while, SO has a success rate of 0%.



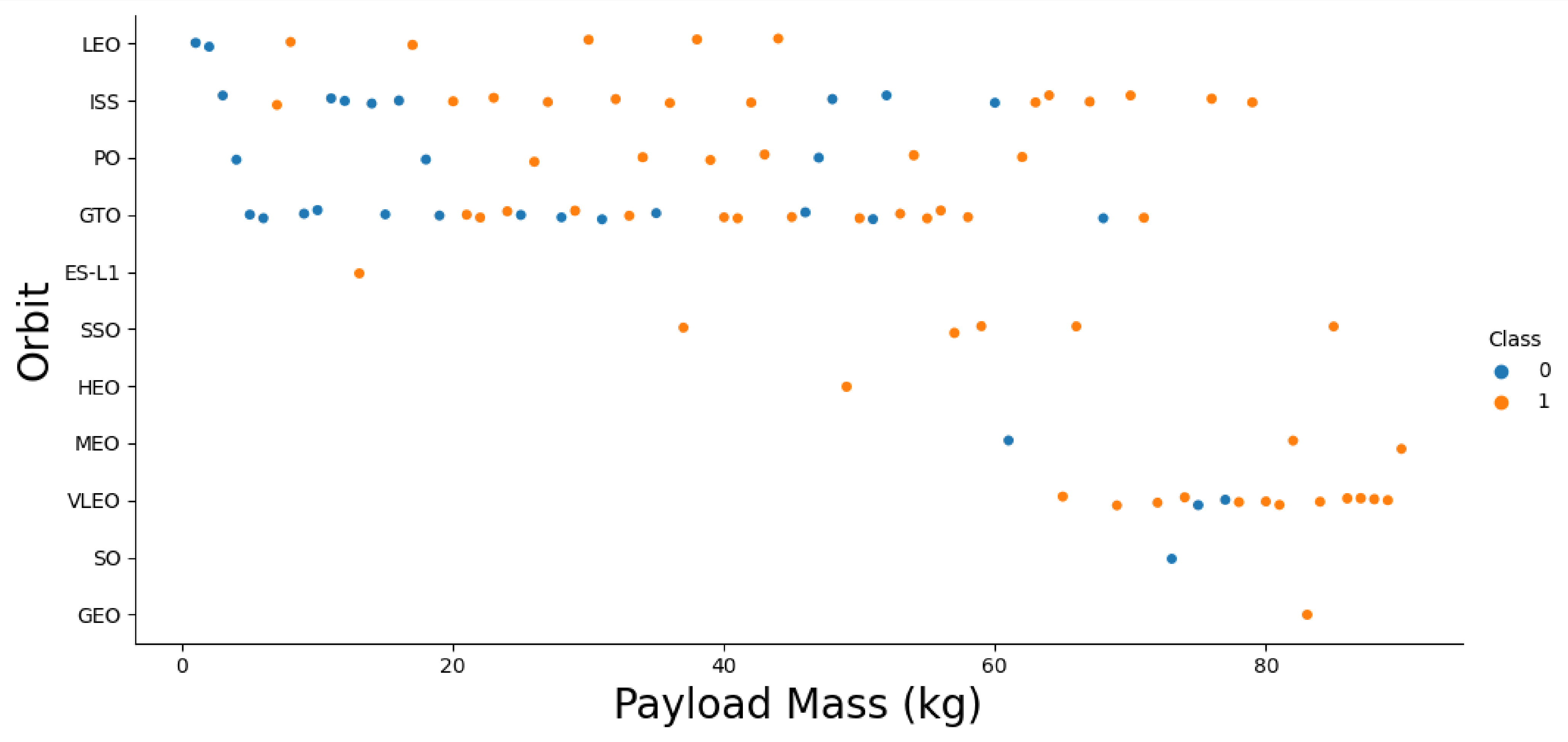
Flight Number vs. Orbit Type



- LEO orbit success increase with the number of flights; and there is no relationship between flight number and GTO orbit

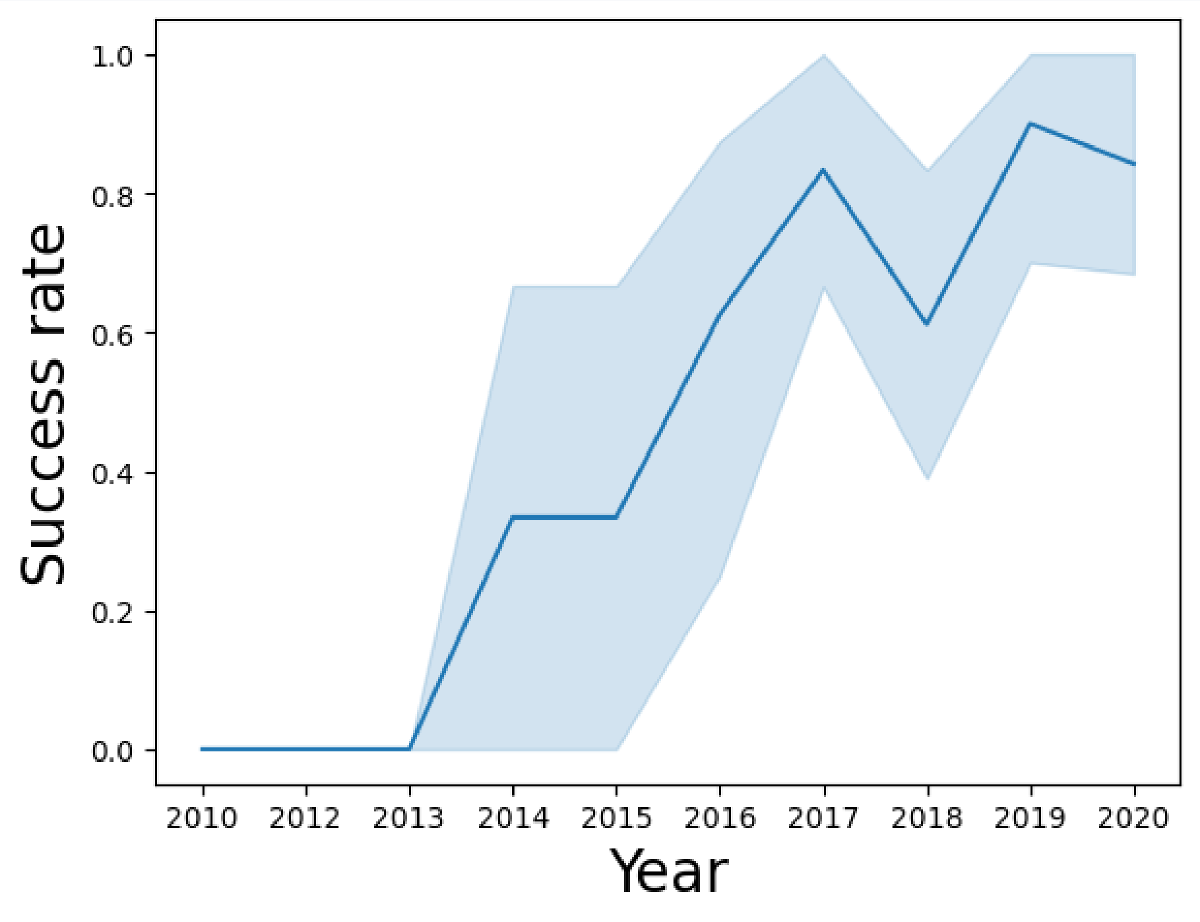
Payload vs. Orbit Type

- Heavy payloads have success landing rate in LEO and ISS and unsuccessful landing rate in GTO



Launch Success Yearly Trend

- There have been increase success rate since 2013. It drop in 2018 and later get stronger till 2020.



All Launch Site Names

```
%sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
```

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

- We used “DISTINCT” to get unique values from database

Launch Site Names Begin with 'CCA'

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE Like 'CCA%' LIMIT 5;
```

- We used “LIMIT” to get only 5 rows

Total Payload Mass

Total Payload Mass by NASA (CRS)
45596.0

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```

- We used the “SUM” to get the total values of Payload Mass

Average Payload Mass by F9 v1.1

Average Payload Mass by Booster Version F9 v1.1
2928.4

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1';
```

- We used “AVG” to get the average values.

First Successful Ground Landing Date

First Succesful Landing Outcome In Ground Pad
01/08/2018

```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome In Ground Pad" FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

- We used “MIN” to get the first successful date.

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

- The payload mass takes between 4000 and 6000 which result in successful drone ship landing.

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEXTBL \
 WHERE MISSION_OUTCOME LIKE "Success%";
```

Successful Mission
100

- We used "COUNT" and "LIKE" "Success%" to get the successful mission .

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Failed Mission" FROM SPACEXTBL \
 WHERE MISSION_OUTCOME LIKE "Failure%";
```

Failed Mission
1

- We used "COUNT" and "LIKE" "Failure%" to get the failed mission .

Boosters Carried Maximum Payload

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

```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

- We used “MAX” to get the maximum payload masses

2015 Launch Records

month	Date	Booster_Version	Launch_Site	Landing_Outcome
10	01/10/2015	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	14/04/2015	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

```
%sql SELECT substr(Date, 4, 2) as month, DATE, BOOSTER_VERSION, LAUNCH_SITE, LANDING_OUTCOME FROM SPACEXTBL \
WHERE LANDING_OUTCOME = 'Failure (drone ship)' and substr(Date, 7, 4) = '2015';
```

- We used month(DATE) and in WHERE we assigned the year “2015” to get the months

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

Landing_Outcome	TOTAL_COUNT
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

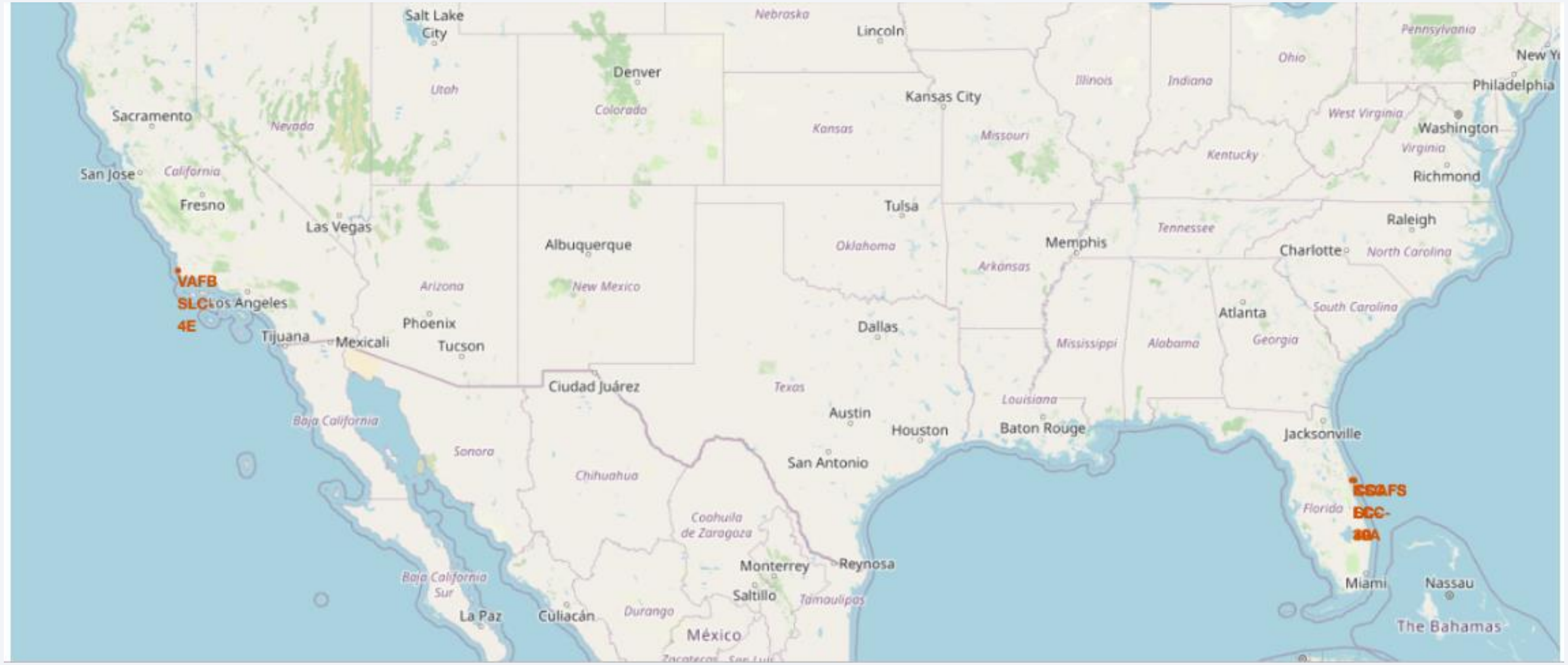
```
%sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) as TOTAL_COUNT FROM SPACEXTBL \
WHERE DATE BETWEEN '04-06-2010' and '20-03-2017' GROUP BY LANDING_OUTCOME \
ORDER BY TOTAL_COUNT DESC;
```

- We used “ORDER” to order the values in descending order and “COUNT” to count the all the available record base on the their categories.

Section 3

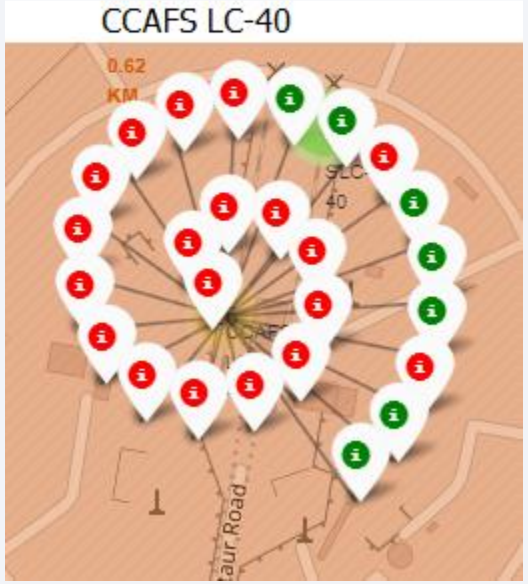
Launch Sites Proximities Analysis

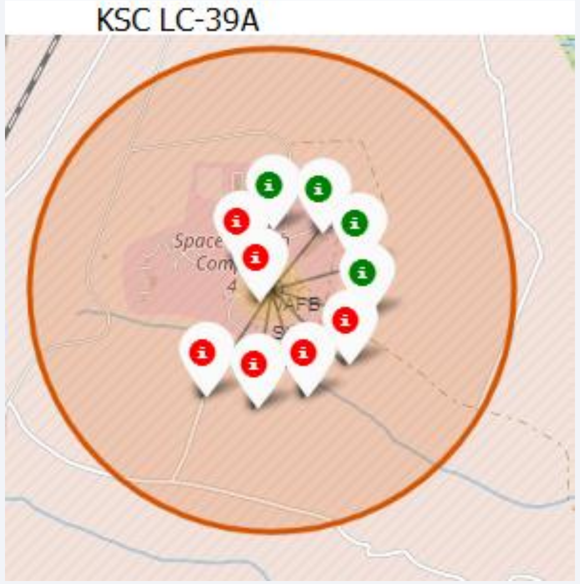
All Launch Sites Location Markers

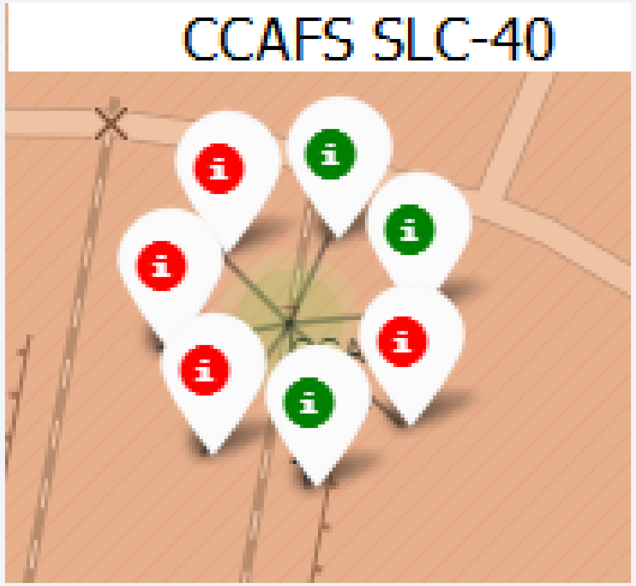


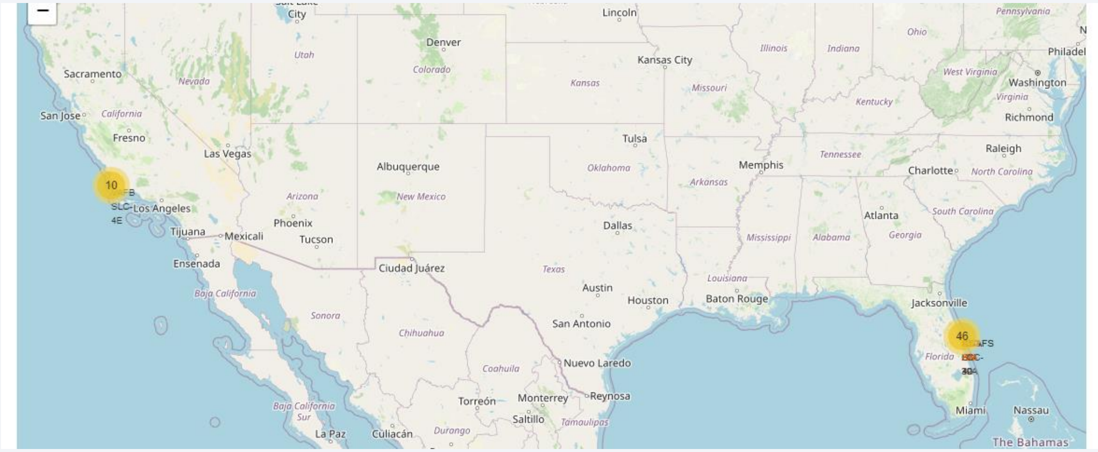
- All the launches are in USA cities

Labelled Launch Outcomes



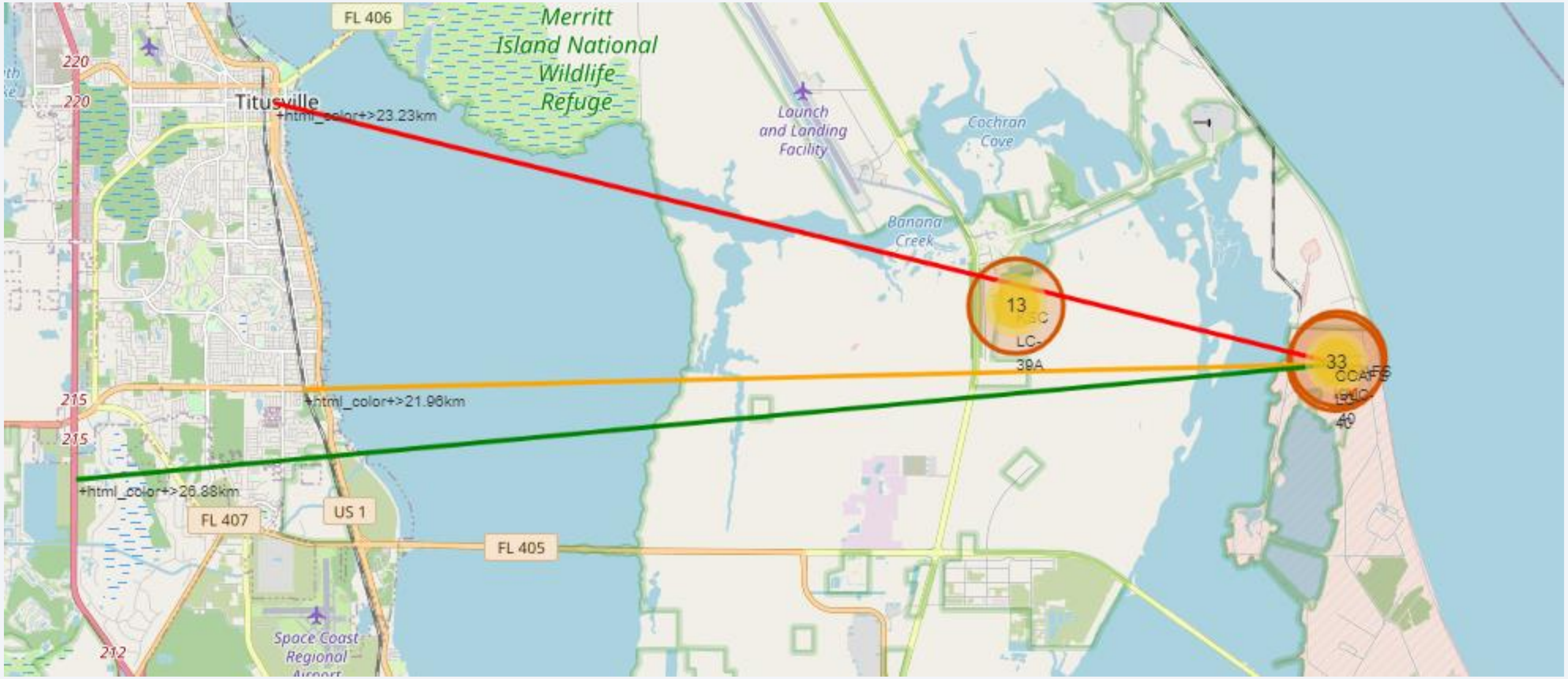






- The Green color indicate success while Red indicate Failure

Launch Sites Proximities

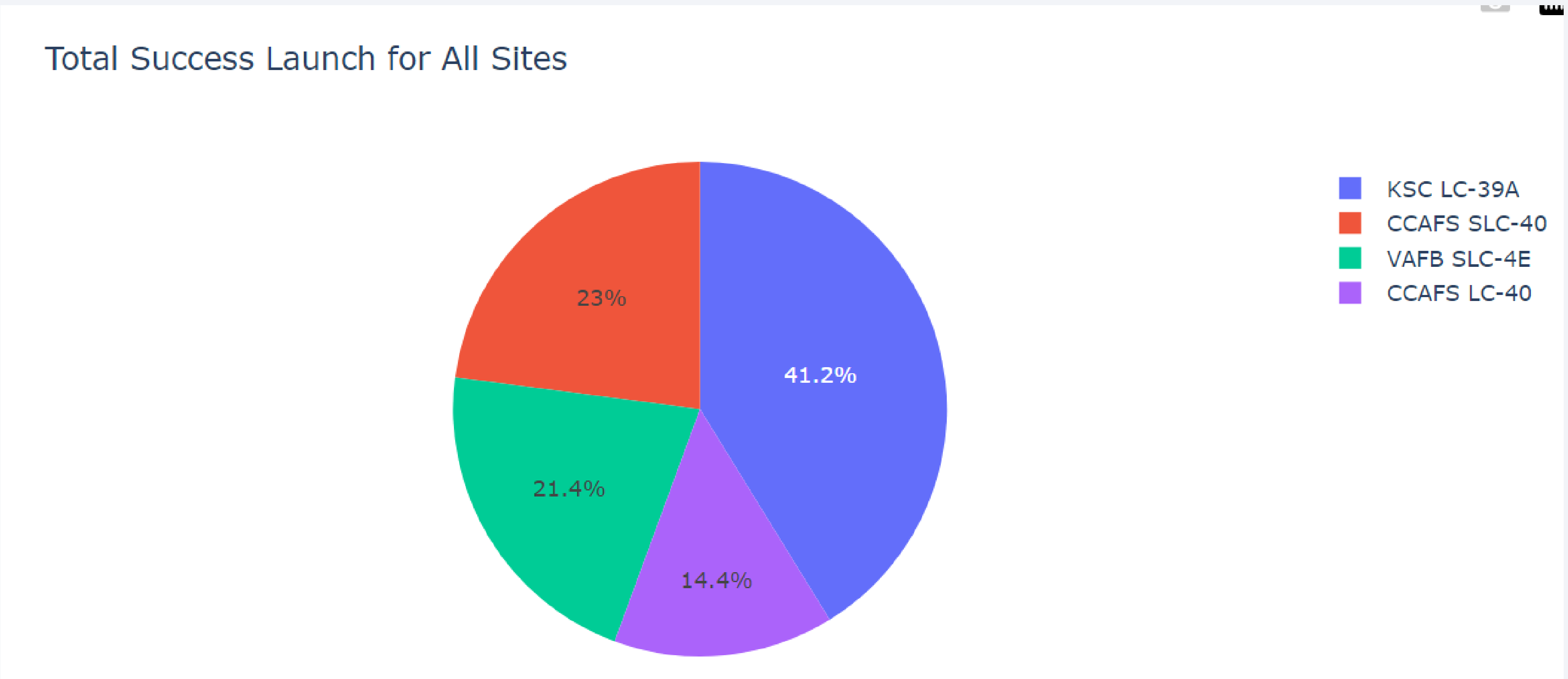


- The launch sites are not far from railway tracks

Section 4

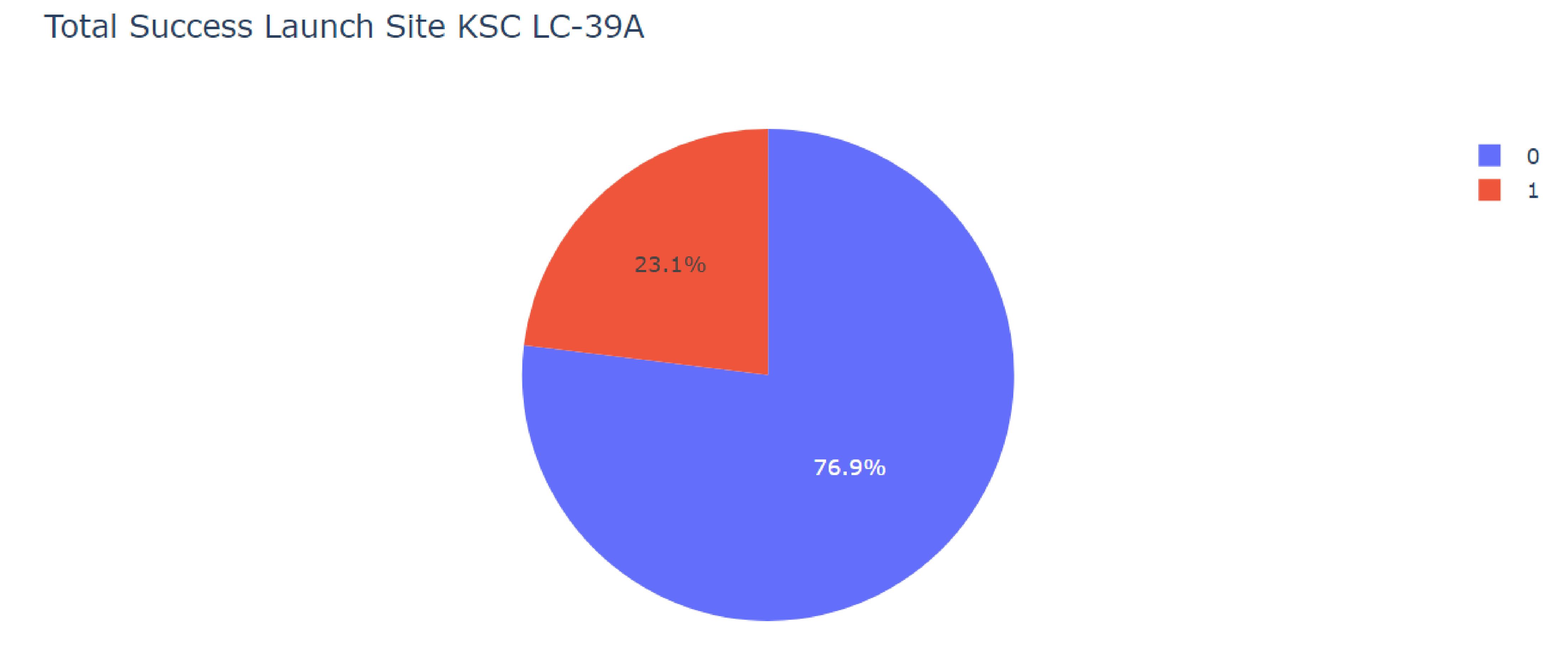
Build a Dashboard with Plotly Dash

Launch Site Success Rate Count



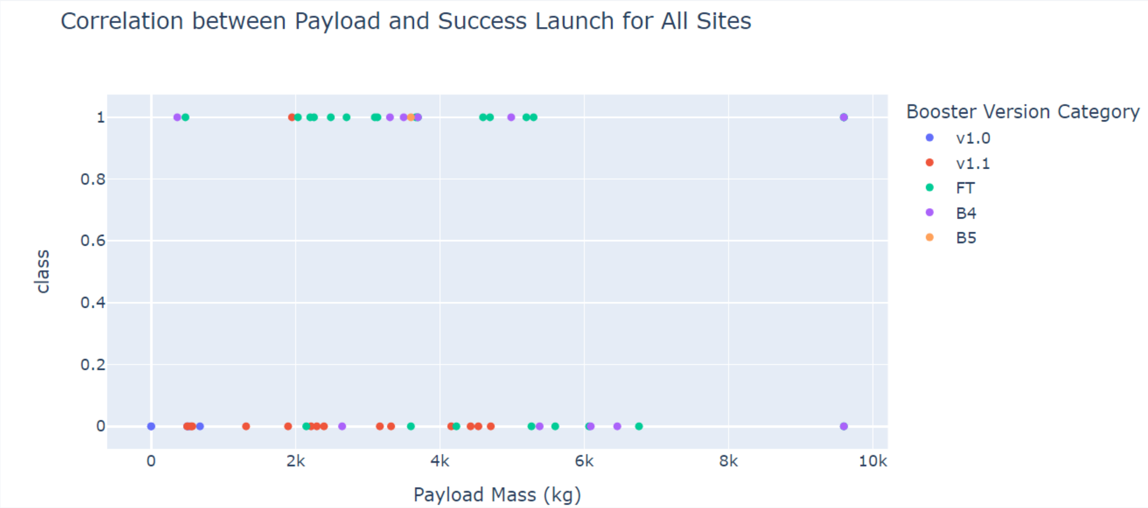
- KSC LC-39A has the highest success rate score 41.7%
- CCAFS LC-40 is next with success rate score of 29.2%
- VAFB-4E has 16.7% and CCAFS SLC-40 is 12.5%.

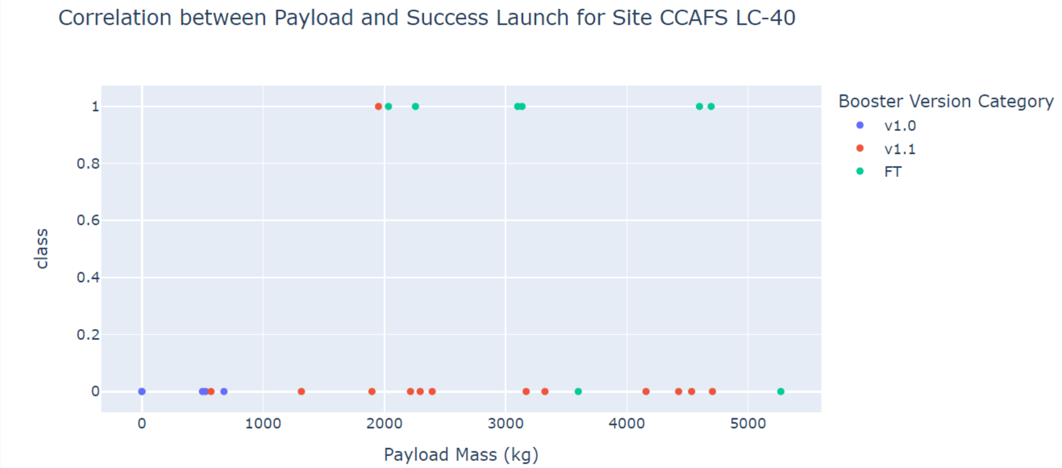
Launch Site



- KSC LC-39A has the highest score of 76.9%.

Payload vs Launch Outcome



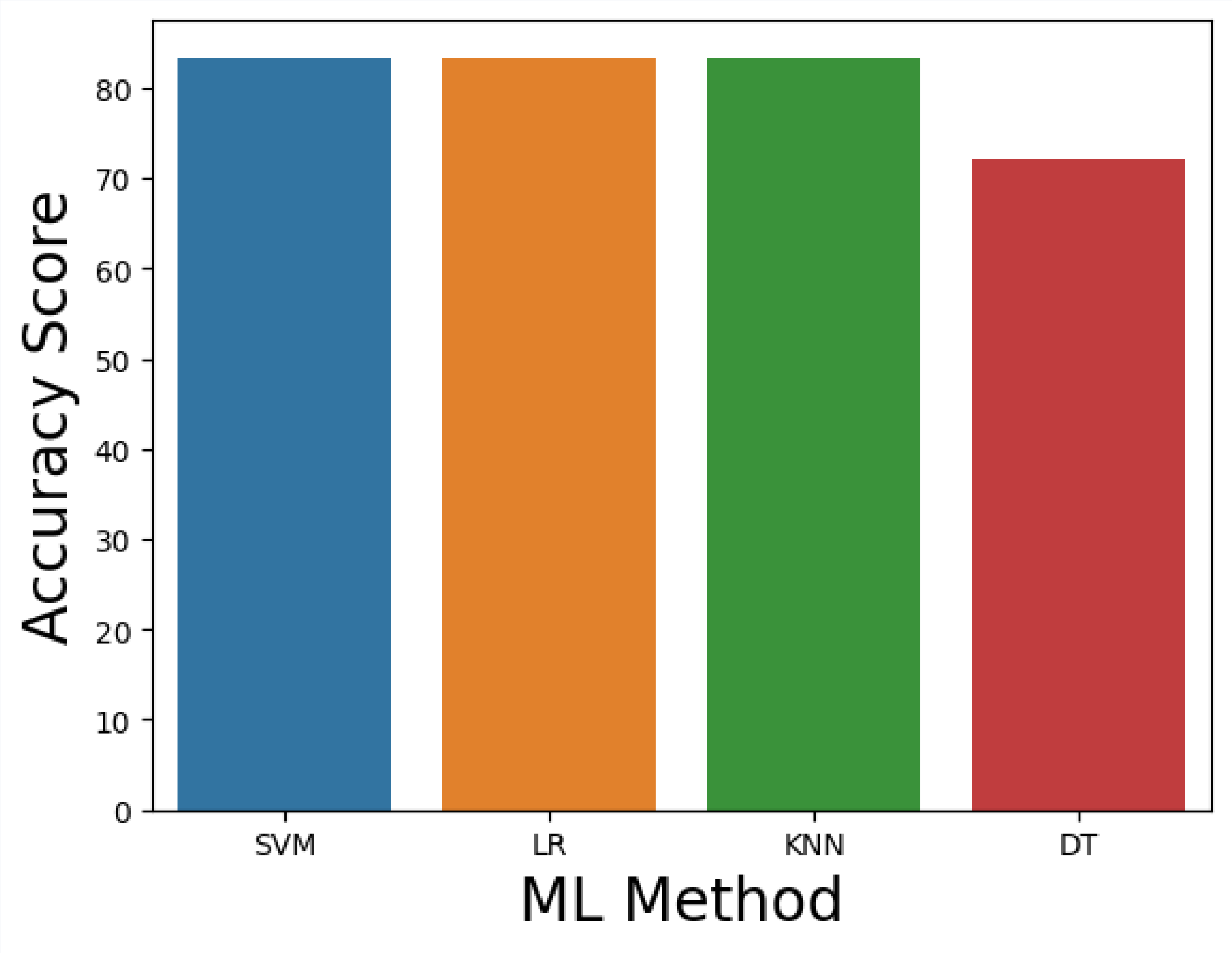


- The Payload of 0kg to 10000kg for all launch sites and for CCAFS LC-40

Section 5

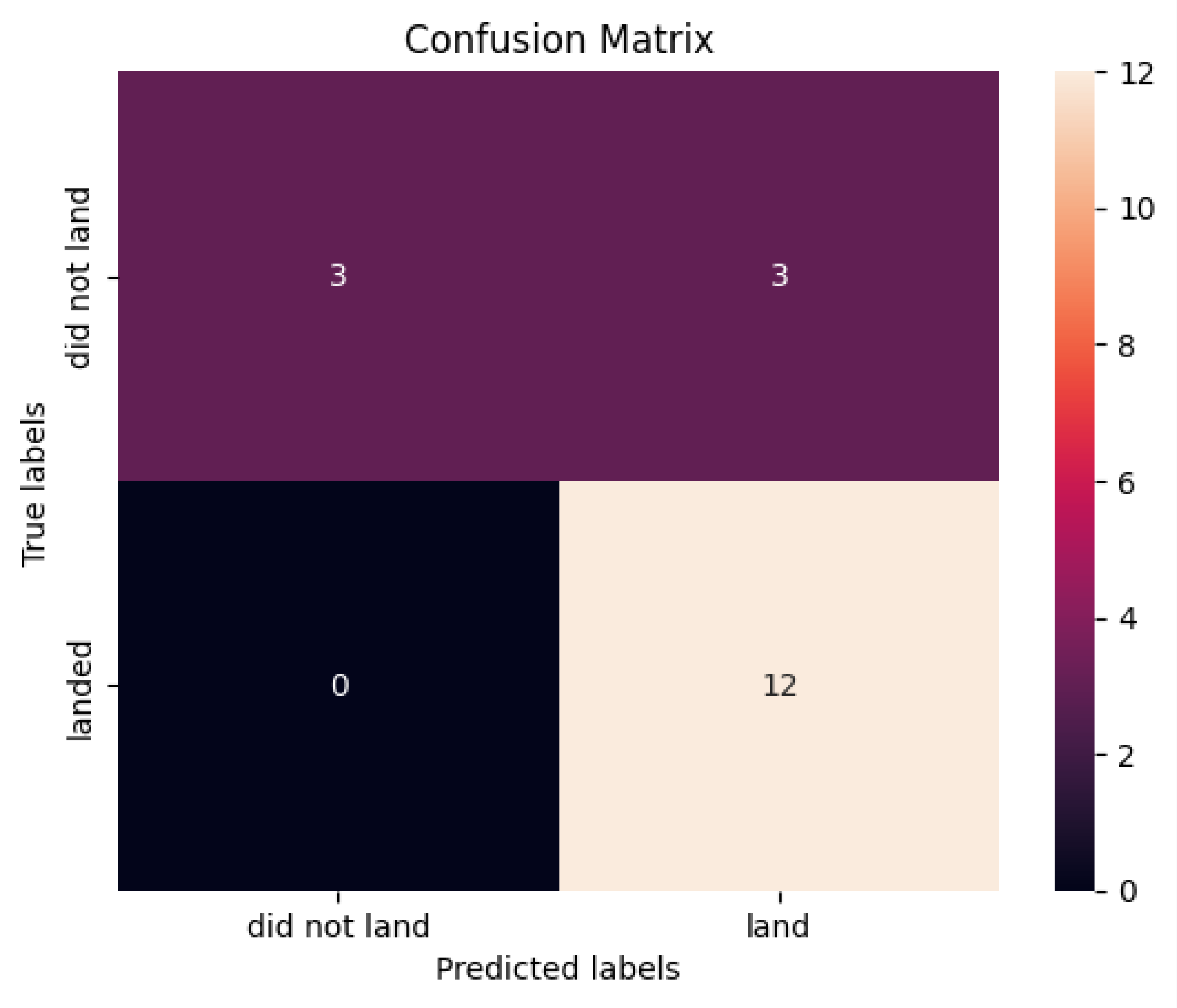
Predictive Analysis (Classification)

Classification Accuracy



- The Decision Tree has the lowest accuracy of 72.2%
- While the other model s have same accuracy of 83.3%.

Confusion Matrix



Conclusions

- The KSC LC-39A has the most successful launches of all the sites.
- The SVM, KNN and Logistic Regression models have highest prediction accuracy of 83.3% for the dataset.
- The Decision Tree accuracy was low accuracy of 72.2%.
- The success rate of SpaceX launches is directly proportional to numbers of years they launch.

Appendix

- All codes for this project can be found on my Github
- https://github.com/emotexplanet/Data_Science_cap.git

Thank you!