Winning Space Race with Data Science

Mangaljit Singh
March 05, 2022

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

- Data Collection with SpaceX API as well as with Web Scrapping using Beautiful Soup package in Python.
- Data cleaning (*e.g.,* removing N.A.s) and data preprocessing (*e.g.,* rescaling data, One Hot encoding *etc.*)
- Exploratory data analysis using SQL, Folium, as well as with Seaborn and Matplotlib package in Python.
- Useful Feature selection for predictive analysis.
- Predictive analysis using various Machine Learning models.

Introduction

- Project background and context

  SpaceX is an American aerospace company specializing in launch vehicles and various rocket engines. SpaceX has also developed reusable rockets, which means that the first stage of the rocket system can be landed back, and reused for the upcoming missions. The cost of the overall project is reduced because the the ability of the SpaceX to reuse the first stage. However, not all first stage landings are successful, and various launch factor can be decisive in resulting a success or a failure. Therefore, it is important to study these factor, so that we can predict if a given rocket launch mission will result into a successful first stage landing or not.

- Problems you want to find answers

  - To study the various factors that are crucial for the success of first stage landing.
  - To check if we can through predictive modelling predict the success of the first stage landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API as well as through web scrapping.
- Perform data wrangling
  - Removing undefined values in the dataset, performing the data scaling including One Hot encoding.
- Perform exploratory data analysis (EDA) using visualizations
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Employing different Machine Learning models, and optimizing the hyper-parameters.
  - Comparing the accuracy of different developed models.

Data Collection – SpaceX API

- Requests package to extract data from SpaceX API
- Link to the complete notebook is below:

https://github.com/Mangaljit/Capstone_IBM_Data_Science/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

Python

```
response = requests.get(spacex_url)
data = pd.json_normalize(response.json())
data.head()
```

```
response = requests.get(static_json_url)
response.status_code
```

Python

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())
```

Python

Using the dataframe `data` print the first 5 rows

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

Python

You will notice that a lot of the data are IDs. For example the rocket column has no information about the rocket just an identification number.

We will now use the API again to get information about the launches using the IDs given for each launch. Specifically we will be using columns `rocket`, `payloads`, `launchpad`, and `cores`.

```
# 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 have multiple payloads in a single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]
```

Data Collection - Scraping

- We used Beautiful soup package to extract data from the webpage.
- Link to the complete notebook is below:

https://github.com/Mangaljit/Capstone_IBM_Data_Science/blob/main/jupyter-labs-webscraping.ipynb

```python
t0 = time.time()
url = "https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches"
r = requests.get(url)
soup = BeautifulSoup(r.text, 'html.parser')
r.status_code
```

Python

A quick inspection of the wikipedia page linked to the above url indicates that the datframes we want to extract have a class "wikitable plainrowheaders collapsible", which can be easily extracted using the 'find_all' method in BeautifulSoup.

+ Code + Markdown

```python
tables = soup.find_all("table", class_ = "wikitable plainrowheaders collapsible")
```

Python

```python
print("There are a total of", len(tables), "number of tables that we would like to extract.")
```

Python

Below we loop through different elements of the html code to extract the required information. First we loop through tables with index [1, 2 ,3 ,4, 7, 8 ,9], because they follow the same exact pattern in the html code, and hence we can loop through all the tables in a single piece of code.

```python
all_tables = []
for ind in [1, 2 ,3 ,4, 7, 8 ,9]: #the index that corresponds to the table to be extracted
    t_body = tables[ind].find("tbody")
    t_r = t_body.find_all("tr")
    date = []
    Booster_Version = []
    Launch_Site = []
    Payload = []
    Payload_mass = []
    Orbit = []
    customer = []
    Launch_Outcome = []
    Booster_Landing = []
    date = []
    #below we loop through different 'td' tags in the html to extract the different columns
    for index in range(1,len(t_r),2):
        date_iter = t_r[index].find_all("td")[0].text
        date.append(date_iter[:-1])
        Booster_Version_iter = t_r[index].find_all("td")[1].text
        Booster_Version.append(Booster_Version_iter[:-1])
        Launch_Site_iter = t_r[index].find_all("td")[2].text
        Launch_Site.append(Launch_Site_iter[:-1])
        Payload_iter = t_r[index].find_all("td")[3].text
        Payload.append(Payload_iter[:-1])
        Payload_mass_iter = t_r[index].find_all("td")[4].text
        Payload_mass.append(Payload_mass_iter[:-1])
        Orbit_iter = t_r[index].find_all("td")[5].text
        Orbit.append(Orbit_iter[:-1])
        customer_iter = t_r[index].find_all("td")[6].text
```

Data Wrangling

- Checking the type/class of data, and changing the data types as required.
- Counting the N.A. values present in different columns in the data set.
- Removing N.A.s from the dataset.
- Renaming different columns in the dataset with appropriate names.
- Link to the complete notebook is below:

  Capstone_IBM_Data_Science/labs-jupyter-spacex-Data wrangling.ipynb at main · Mangaljit/Capstone_IBM_Data_Science (github.com)

EDA with Data Visualization

- Visualized the relationship between:
  - Flight Number and Launch Site
  - Flight Number and Payload Mass
  - Payload Mass and Launch Site
  - success rate of each orbit type
  - Flight Number and Orbit type
  - Payload Mass and Orbit type
- Visualized launch success yearly trend.

https://github.com/Mangaljit/Capstone_IBM_Data_Science/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Evaluated names of the unique launch sites in the space mission.
- Displayed 5 records where launch sites begin with the string 'CCA’.
- Displayed the total payload mass carried by boosters launched by NASA (CRS).
- Displayed average payload mass carried by booster version F9 v1.1.
- Listed the date when the first successful landing outcome in ground pad was achieved.
- Listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
- Listed the total number of successful and failure mission outcomes.
- Listed the names of the booster versions which have carried the maximum payload mass.
- Listed the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015.
- Ranked 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.

https://github.com/Mangaljit/Capstone_IBM_Data_Science/blob/main/jupyter-labs-eda-sql-coursera.ipynb

Build an Interactive Map with Folium

- Marked all the launch sites on an interactive map.
- Added map objects such as markers, circles, and lines to mark the success or failure of launches for each site.
- Assigned the launch outcomes (failure or success) to class 0 and 1, *i.e.,* 0 for a failure and 1 for a success.
- Color-labeled marker clusters to identify which launch sites have relatively high success rate.

https://github.com/Mangaljit/Capstone_IBM_Data_Science/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- Built an interactive dashboard using Plotly dash.
- Plotted pie charts showing the total launches by a certain sites.
- Generated a scatter plot depicting the relationship between Outcome and Payload Mass (Kg) for the different booster versions.

https://github.com/Mangaljit/Capstone_IBM_Data_Science/blob/main/interactive_dashboard_SpaceX.ipynb

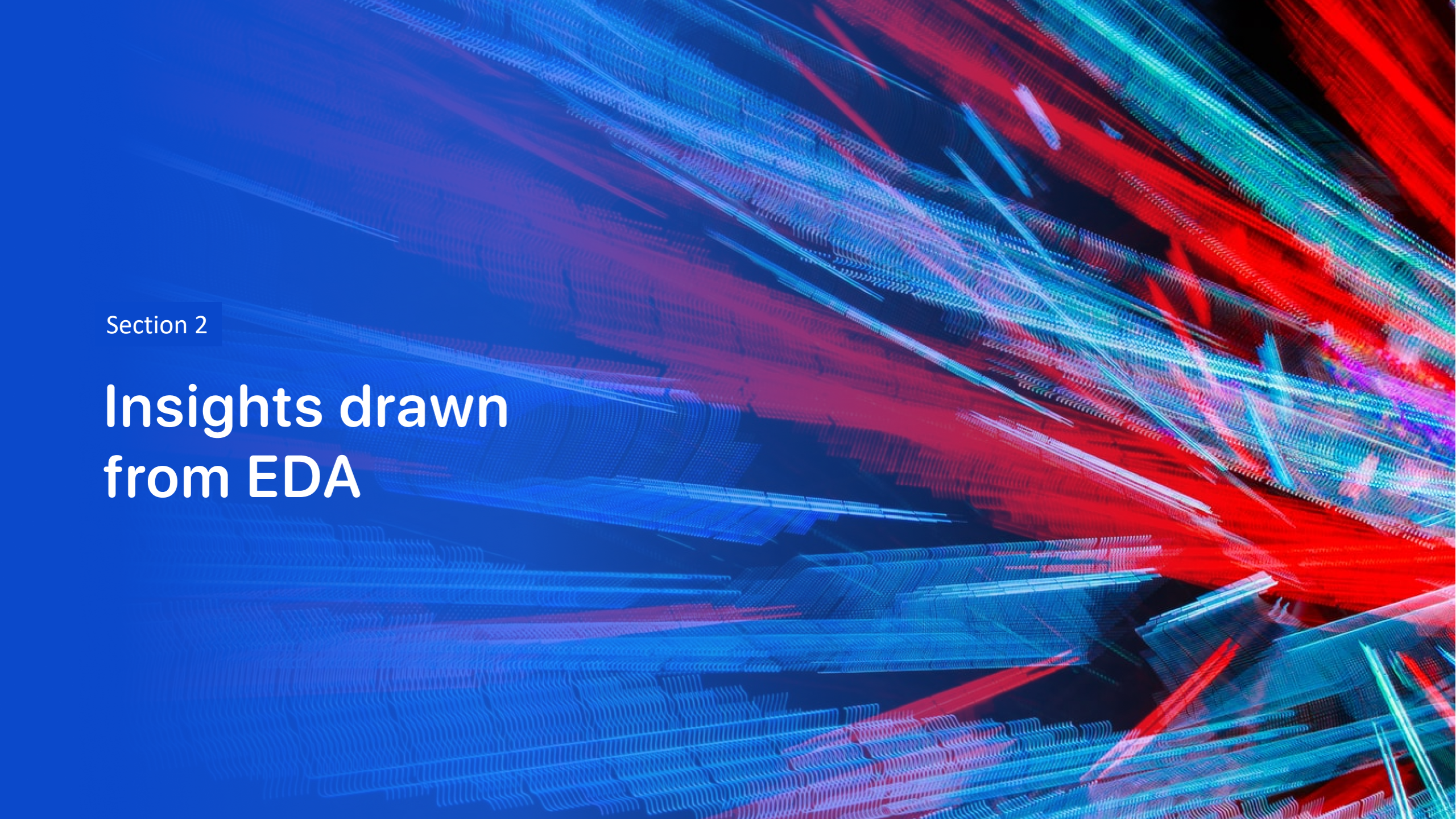
Predictive Analysis (Classification)

- Employed different Machine Learning Models to predict success or failure of a mission.
  - Data scaling using StandardScalar() from sklearn library in Python.
  - Data splitting into testing and training datasets.
  - GridSearchCV with 10-fold cross-validation for hyperparameter tuning to achieve the best model accuracy.
  - Listed the model providing the best accuracy.

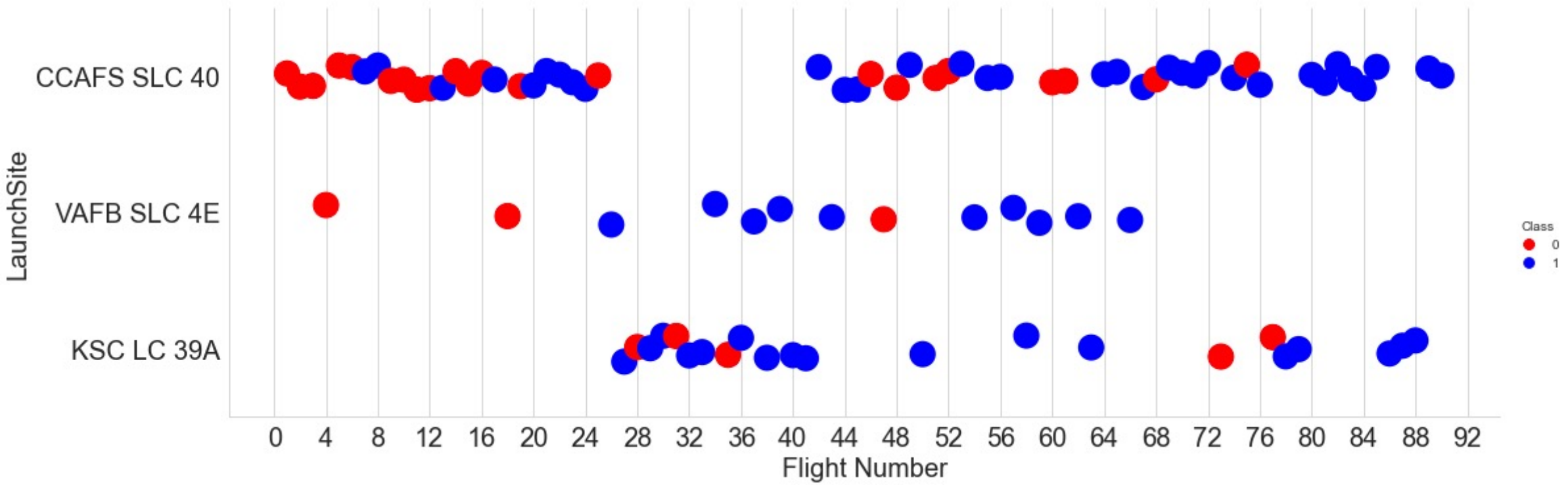
https://github.com/Mangaljit/Capstone_IBM_Data_Science/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

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

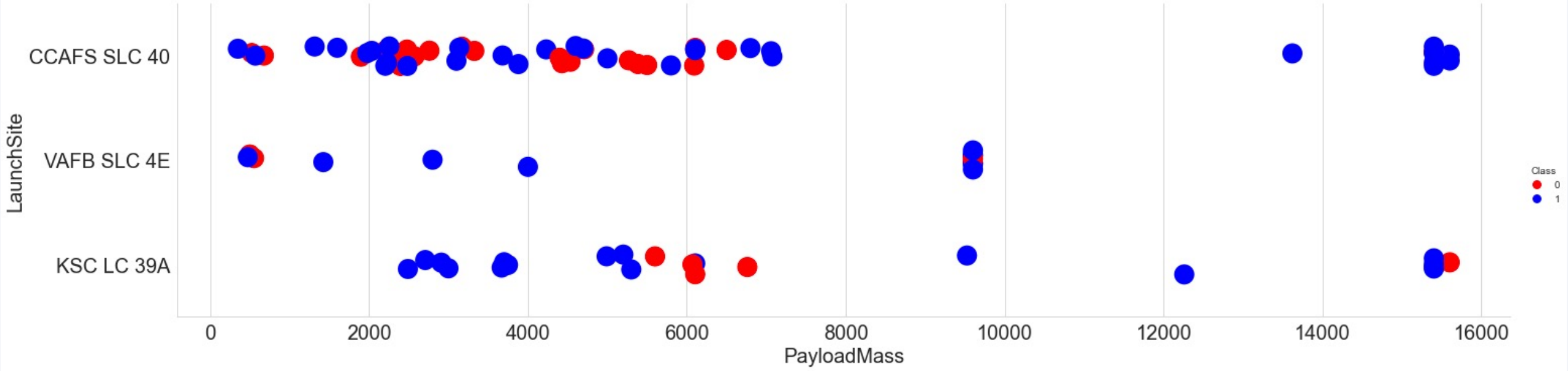

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



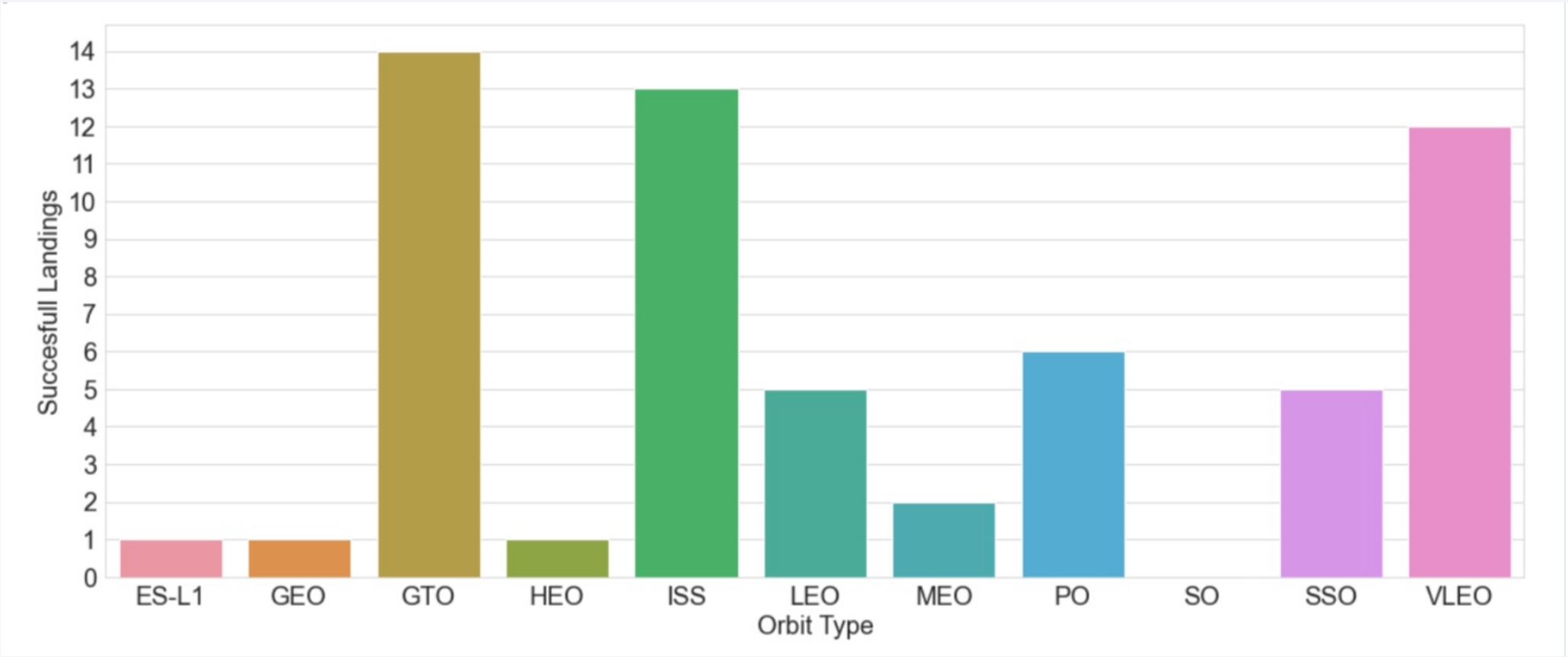
We observe that at each launch site, the initial flights experienced higher rate of failure. However, as the time passed, the later flights had a relatively higher success rates.

Payload vs. Launch Site



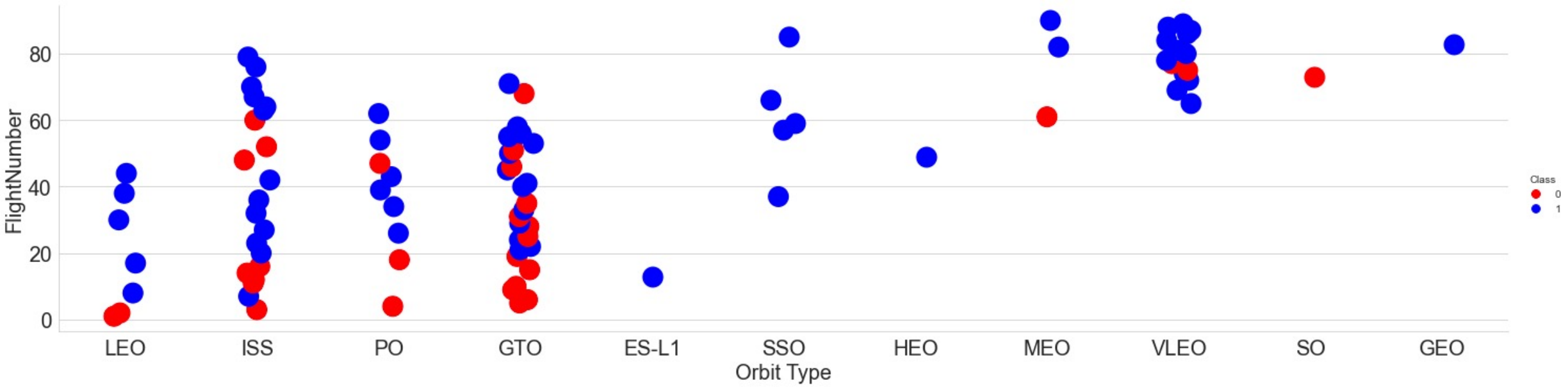
- For the VAFB-SLC launch site, there are no rockets launched for heavy payload mass (greater than 10000).
- For CCAFS SLC launch site, payload mass >6000 Kg experienced a high success rate.

Success Rate vs. Orbit Type



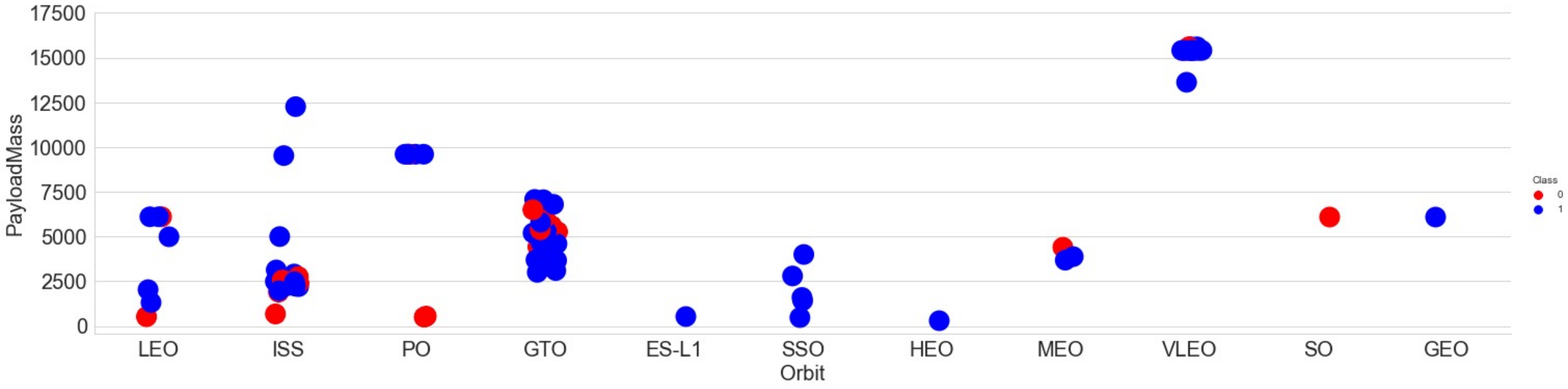
GTO, ISS and VLEO orbits were among the most successful.

Flight Number vs. Orbit Type



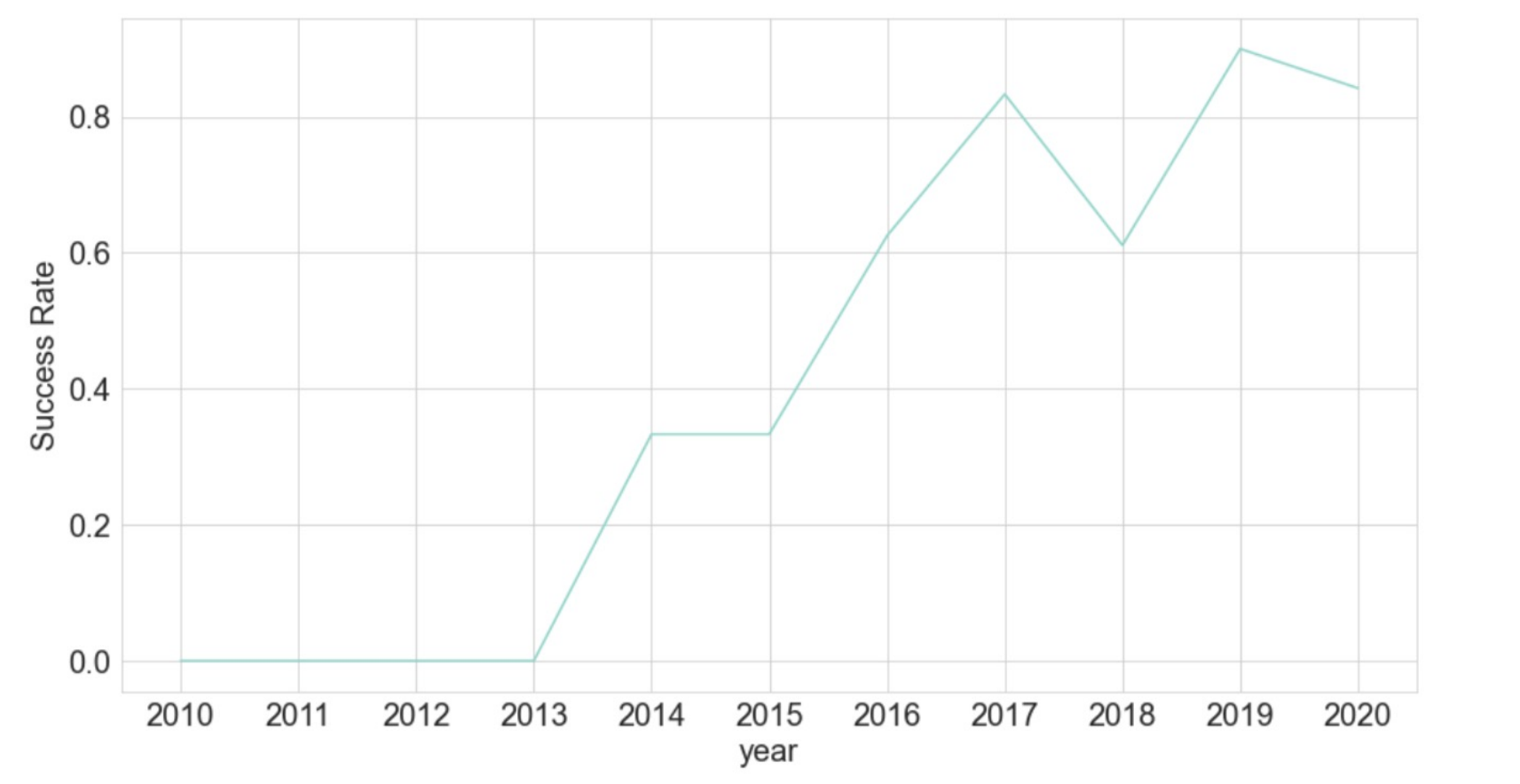
- We see that LEO, ISS, PO and GTO orbits constitute the major percentage of flight launches.
- We also see that in the LEO orbit, the success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.

Payload vs. Orbit Type



- With heavy payloads, the successful landing or positive landing rate are more for Polar, LEO and ISS orbit.
- However for GTO, we cannot distinguish, because both positive and negative landing are present.

Launch Success Yearly Trend



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

All Launch Site Names

Task 1

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

```
task1 = '''SELECT DISTINCT LAUNCHSITE
        FROM SPACEXTABLE'''
pd.read_sql(task1, pconn)
```

Python

```
/Users/mangaljitsingh/opt/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

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

Launch Site Names Begin with 'CCA'

Task 2

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

```
task_2 = '''
        SELECT *
        FROM SPACEXTABLE
        WHERE LAUNCHSITE LIKE 'CCA%'
        LIMIT 5
        '''
pd.read_sql(task_2, pconn)
```

Python

```
/Users/mangaljitsingh/opt/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

	DATE	TIME	BOOSTERVERSION	LAUNCHSITE	PAYLOAD	PAYLOADMASSKG	ORBIT	CUSTOMER	MISSIONOUTCOME	OUTCOMELANDING
0	2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-12-08	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	2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

Task 3

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

```
task_3 = '''
        SELECT SUM(PAYLOADMASSKG) AS Total_Payload_Mass
        FROM SPACEXTABLE
        WHERE CUSTOMER LIKE 'NASA (CRS)'
        '''
pd.read_sql(task_3, pconn)
```

Python

```
/Users/mangaljitsingh/opt/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

	TOTAL_PAYLOAD_MASS
0	45596

Average Payload Mass by F9 v1.1

Task 4

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

```
task_4 = '''
        SELECT AVG(PAYLOADMASSKG) AS Avg_PayloadMass
        FROM SPACEXTABLE
        WHERE BOOSTERVERSION = 'F9 v1.1'
        '''
pd.read_sql(task_4, pconn)
```

Python

```
/Users/mangaljitsingh/opt/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

	AVG_PAYLOADMASS
0	2928

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

```
task_5 = '''
        SELECT MIN(DATE) AS FirstSuccessfull_landing_date
        FROM SPACEXTABLE
        WHERE OUTCOMELANDING LIKE 'Success (ground pad)'
        '''

pd.read_sql(task_5, pconn)
```

Python

```
/Users/mangaljitsingh/opt/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

	FIRSTSUCCESSFULL_LANDING_DATE
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

Task 6

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

```
task_6 = '''
        SELECT  BOOSTERVERSION
        FROM SPACEXTABLE
        WHERE OUTCOMELANDING = 'Success (drone ship)'
            AND PAYLOADMASSKG > 4000
            AND PAYLOADMASSKG < 6000
        '''
pd.read_sql(task_6, pconn)
```

Python

```
/Users/mangaljitsingh/opt/anaconda3/lib/python3.9/site-packages/pandas/io/sql.py:761: UserWarning: pandas only support SQLAlchemy connectable(engine/connection) ordatabase string URI or sqlite3 DBAPI2 connectionother DBAPI2 objects are not tested, please consider using SQLAlchemy
  warnings.warn(
```

	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

Task 7

List the total number of successful and failure mission outcomes

```
import warnings
warnings.filterwarnings("ignore")
task_7_1 = '''
        SELECT COUNT(MISSIONOUTCOME) AS SuccessMission
        FROM SPACEXTABLE
        WHERE MISSIONOUTCOME LIKE 'Success%'
        '''

task_7_2 = '''
        SELECT COUNT(MISSIONOUTCOME) AS FailureMission
        FROM SPACEXTABLE
        WHERE MISSIONOUTCOME LIKE 'Failure%'
        '''
print("Total number of successfull missions are ",pd.read_sql(task_7_1, pconn).iloc[0,0],".")
print("Total number of unsuccessfull missions are ", pd.read_sql(task_7_2, pconn).iloc[0,0],".")
```

Python

```
Total number of successfull missions are  100 .
Total number of unsuccessfull missions are  1 .
```

Boosters Carried Maximum Payload

Task 8

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

```
task_8 = '''
        SELECT BOOSTERVERSION, PAYLOADMASSKG
        FROM SPACEXTABLE
        WHERE PAYLOADMASSKG = (
                              SELECT MAX(PAYLOADMASSKG)
                              FROM SPACEXTABLE
                              )
        ORDER BY BOOSTERVERSION
        '''
pd.read_sql(task_8, pconn)
```

Python

	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

Task 9

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

```python
task_9 = '''
        SELECT BOOSTERVERSION, LAUNCHSITE, OUTCOMELANDING
        FROM SPACEXTABLE
        WHERE OUTCOMELANDING LIKE 'Failure (drone ship)'
            AND DATE BETWEEN '2015-01-01' AND '2015-12-31'
        '''
pd.read_sql(task_9, pconn)
```

Python

	BOOSTERVERSION	LAUNCHSITE	OUTCOMELANDING
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

Task 10

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

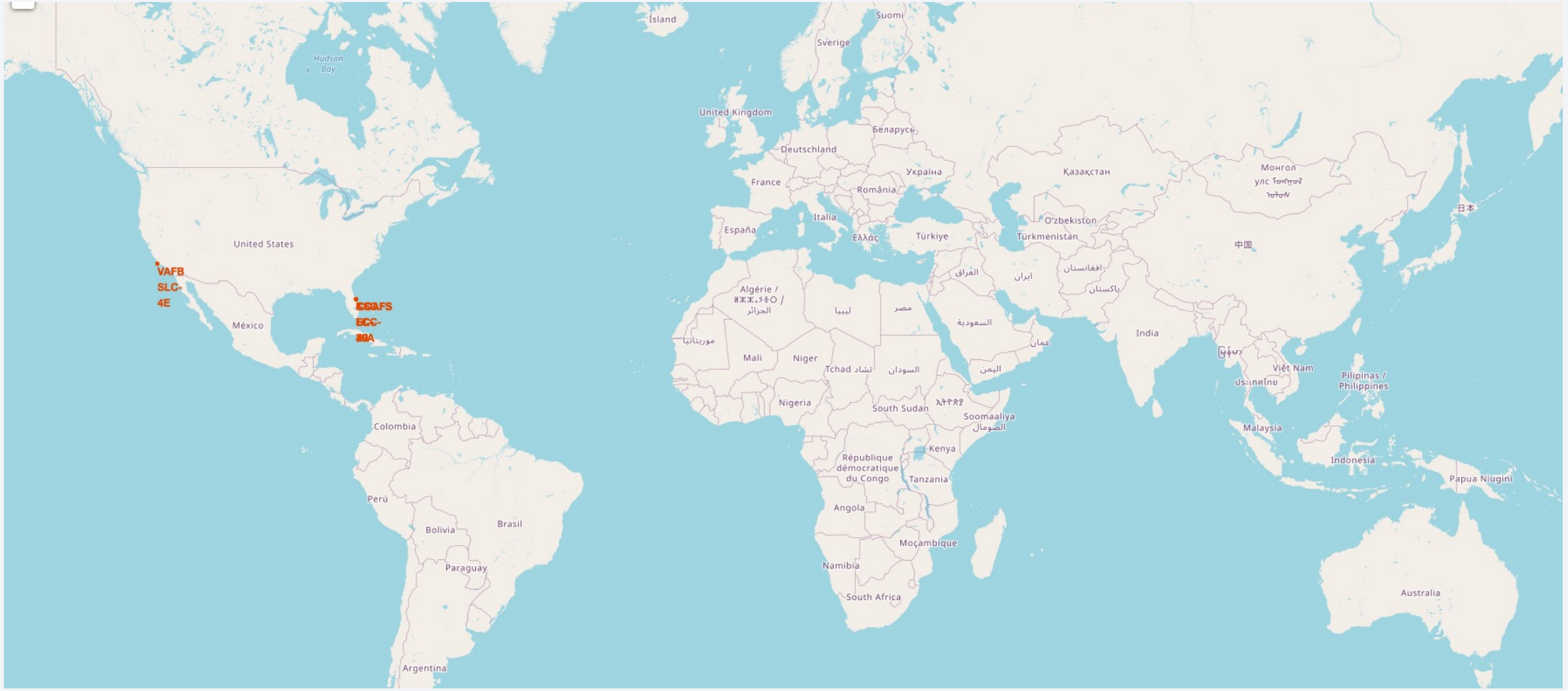
```python
task_10 = '''
        SELECT OUTCOMELANDING, COUNT(OUTCOMELANDING)
        FROM SPACEXTABLE
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY OUTCOMELANDING
        ORDER BY COUNT(OUTCOMELANDING) DESC
        '''
df = pd.read_sql(task_10, pconn)
df
# df.rename(columns = {'2' : 'count', 'OUTCOMELANDING': 'landing type'})
```

Python

	OUTCOMELANDING	2
0	No attempt	10
1	Failure (drone ship)	5
2	Success (drone ship)	5
3	Controlled (ocean)	3
4	Success (ground pad)	3
5	Failure (parachute)	2
6	Uncontrolled (ocean)	2
7	Precluded (drone ship)	1

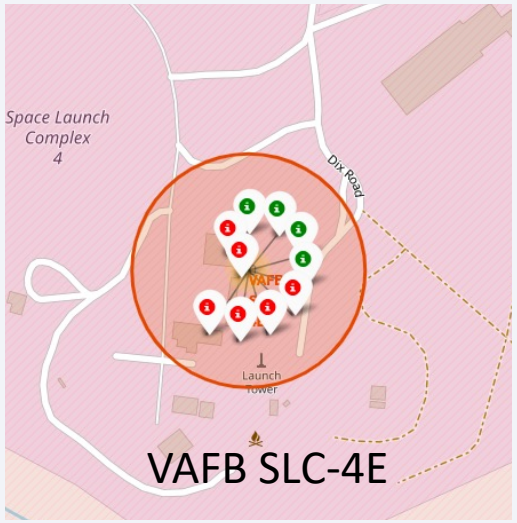

Section 3
Launch Sites
Proximities Analysis

Launch Site Locations

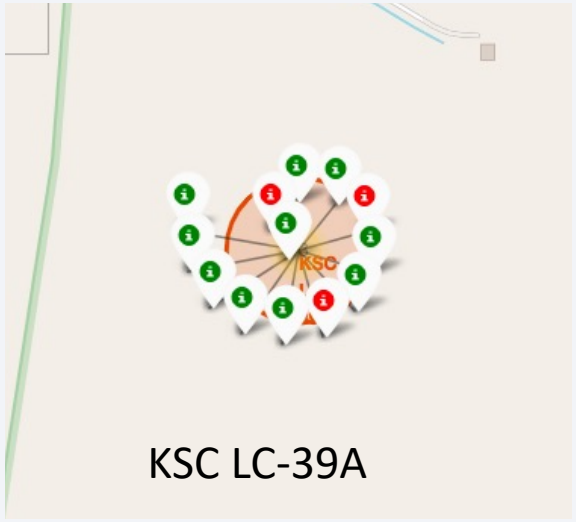


All launch sites are located in the United States of America.

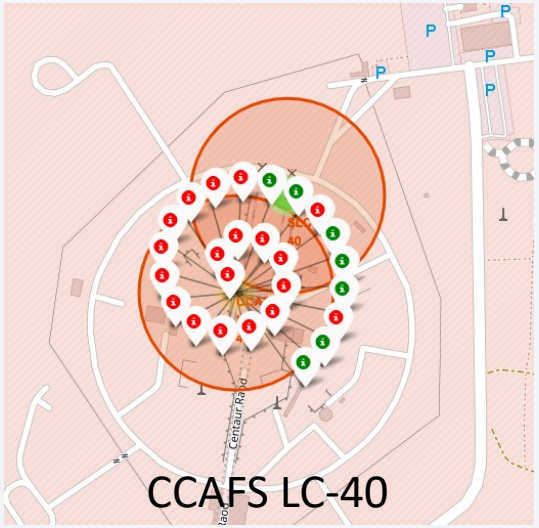
Launch Sites with Colored Markers



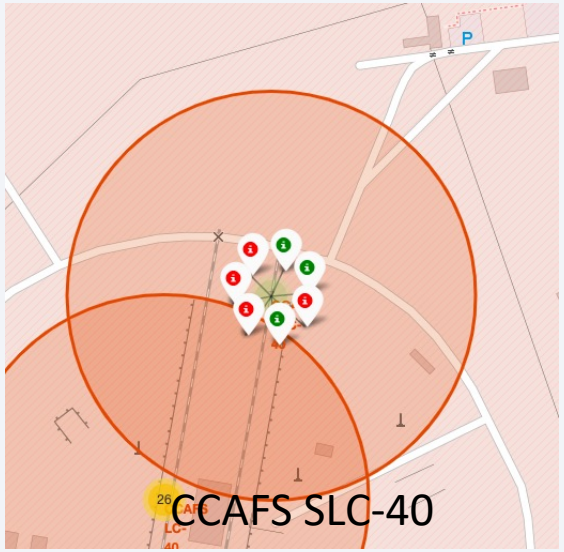
VAFB SLC-4E



KSC LC-39A

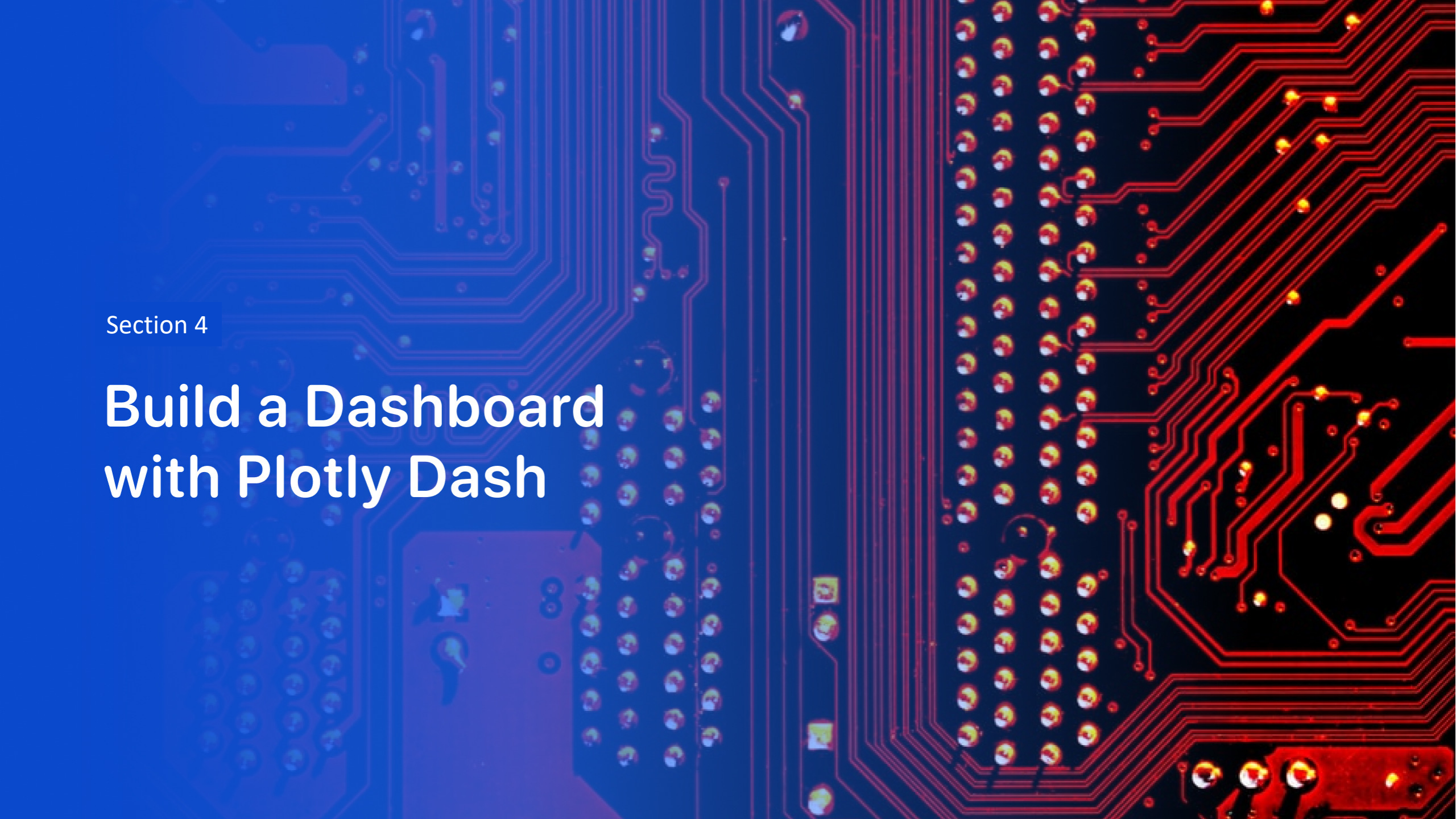


CCAFS LC-40

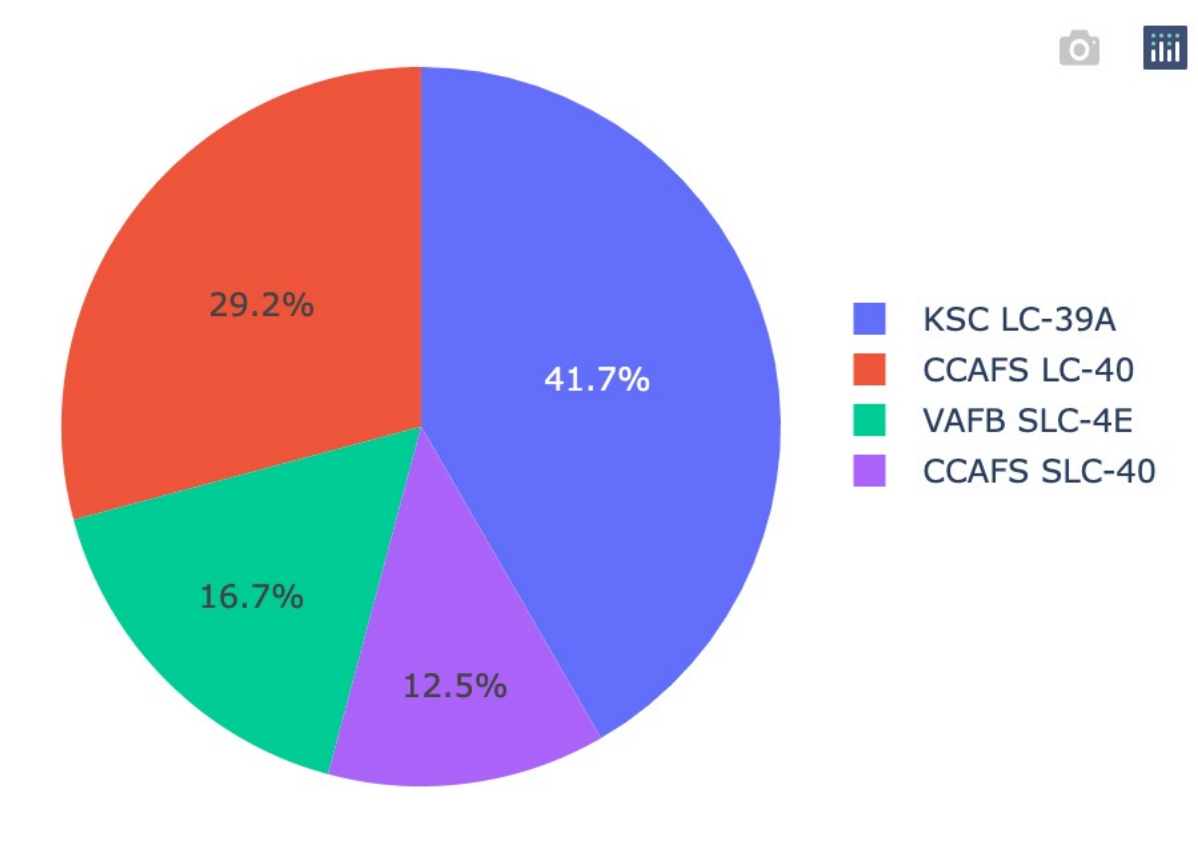


CCAFS SLC-40

Green marker depicts a success, while a red marker indicates a failure

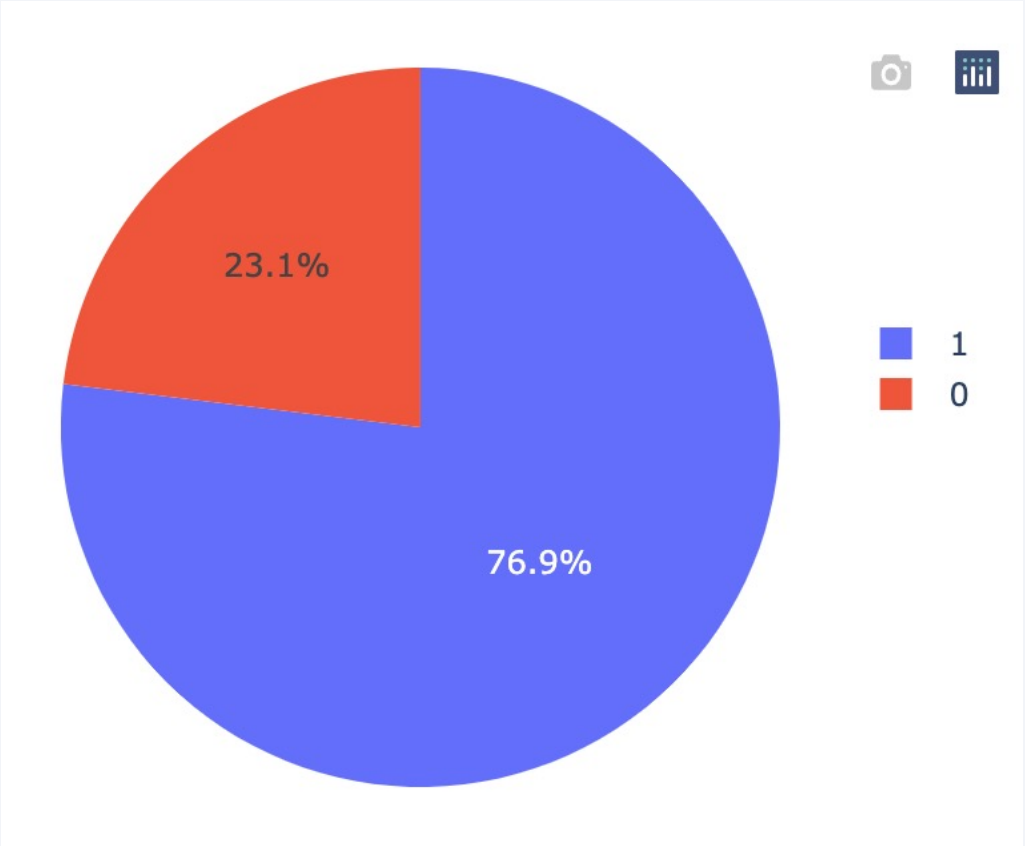

Section 4
Build a Dashboard with Plotly Dash

Success Launch Rate for All Sites



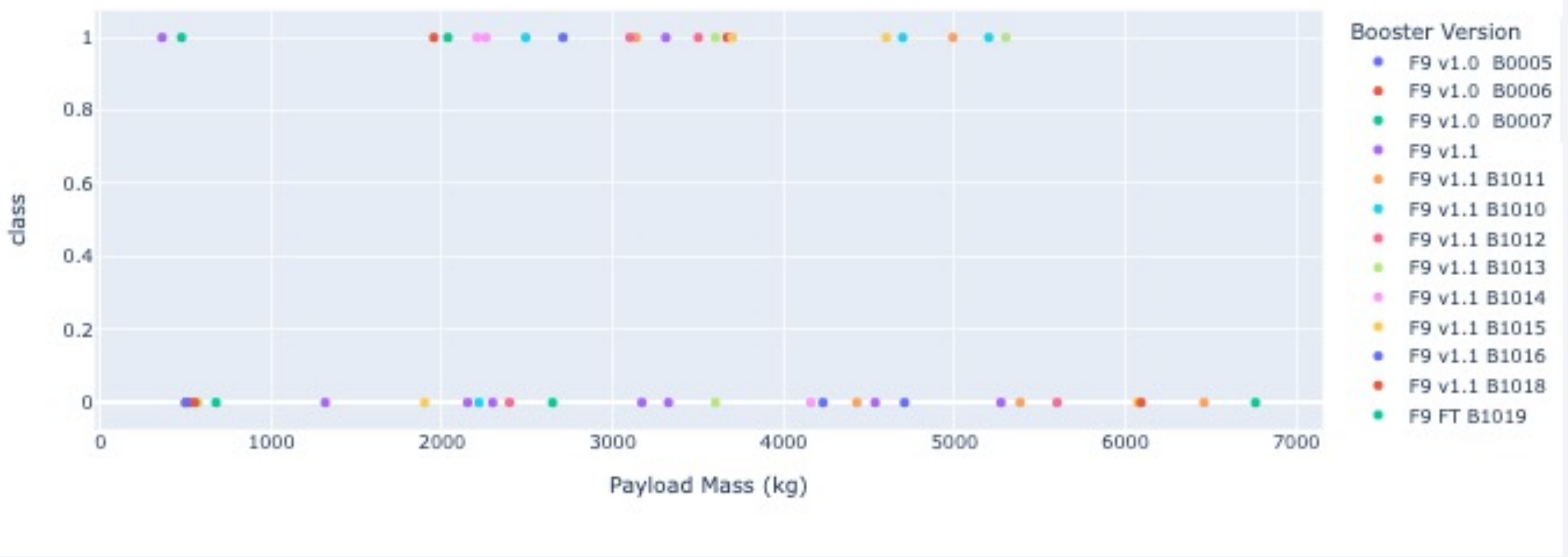
The launch site KSC LC-39A has the highest success rate among all the orbits.

KSC LC-39A Launch Site Success Rate



The launch site KSC LC-39A has the success rate of 76.9 %.

Correlation Between Payload and Success For All Sites



Section 5

Predictive Analysis (Classification)

Classification Accuracy

```
# logreg_cv svm_cv tree_cv, knn_cv
model_accuracies = pd.DataFrame({
    'Model' : ['Logistic regression', 'Support Vector Machine', 'Decesion Tree', 'KNN'],
    'Score' : [logreg_cv.best_score_, svm_cv.best_score_, tree_cv.best_score_, knn_cv.best_score_]
})
model_accuracies
```

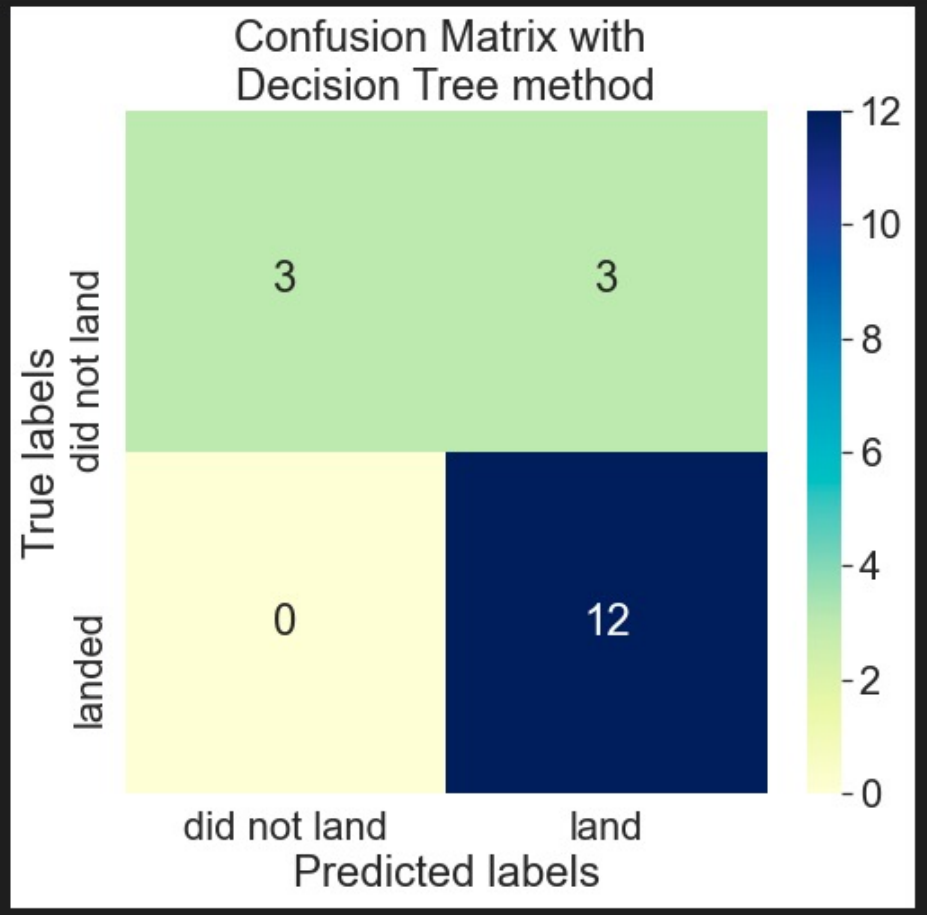
	Model	Score
0	Logistic regression	0.846429
1	Support Vector Machine	0.848214
2	Decesion Tree	0.876786
3	KNN	0.848214

```
print('The decision tree model performs best with the score of ' + str(tree_cv.best_score_)+'.')
```

```
The decision tree model performs best with the score of 0.8767857142857143.
```

Confusion Matrix

```
plot_confusion_matrix(Y_test,yhat, method = "Decision Tree")
```



The decision tree model can distinguish between the different classes with high accuracy. The major problem is the false positives *.i.e.*, unsuccessful landing marked as successful landing by the model.

Conclusions

- To conclude, we performed predictive analysis to predict the success rate of the SpaceX launches.
- Through exploratory data analysis, we studies several features that could influence the success rate of the launch.
- Through predictive modeling using different Machine Learning models, we were able to predict the launch outcome with high accuracy.
- The accuracy was maximum for the Decision Tree Machine Learning Model.

Thank you!