Winning Space Race with Data Science

Ignacio Tolosa
19 July 2022

Outline

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

Executive Summary

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

Introduction

- Project background and context

  Space X offers launches with its Falcon 9 rockets at a value of 62 million dollars while its competitors charge 165 million dollars. The big difference is the reuse of the first stage. The greater the certainty that the first stage will land, the greater the certainty in determining the cost of the mission.. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  - Identifying all factors that influence the landing outcome.
  - The relationship between each variables and how it is affecting the outcome.
  - The best condition needed to increase the probability of successful landing..

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX REST API and web scrapping from Wikipedia;
- Perform data wrangling
  - Data was processed using one-hot encoding for categorical features;
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models

Data Collection

The data was collected with some methods:

- Data collection - get request - SpaceX API;
- Decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize();
- Cleaned the data, checked for missing values and fill in missing values when necessary;
- Performed web scraping - Wikipedia - Falcon 9 launch records using with BeautifulSoup;
- Objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis;

Data Collection – SpaceX API

- Request to the SpaceX API to collect data, clean the requested data and data wrangling and formatting;

- Link notebook https://github.com/ignaciotolosa/IBM_Capstone_SpaceX/blob/main/notebook_Data_Collecting_sFSCxb0eG.ipynb

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

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

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

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

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Sk
```

```
In [12]: # Use json_normalize meethod to convert the json result into a dataframe
         data = pd.json_normalize(response.json())
```

```
In [14]: # Lets take a subset of our dataframe keeping only the features we want and the flight number, and
         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 boos
         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 ar
         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
         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)]
```

Data Collection - Scraping

- Web scrapping to webscrap Falcon 9 launch records with BeautifulSoup ;
- Parsed the table and converted it into a pandas df;
- Link notebook https://github.com/ignaciotolosa/IBM_Capstone_SpaceX/blob/main/notebook_Data_Collection_with_Web_Scraping_mPGB-VU2J.ipynb

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        data = requests.get(static_url).text
```

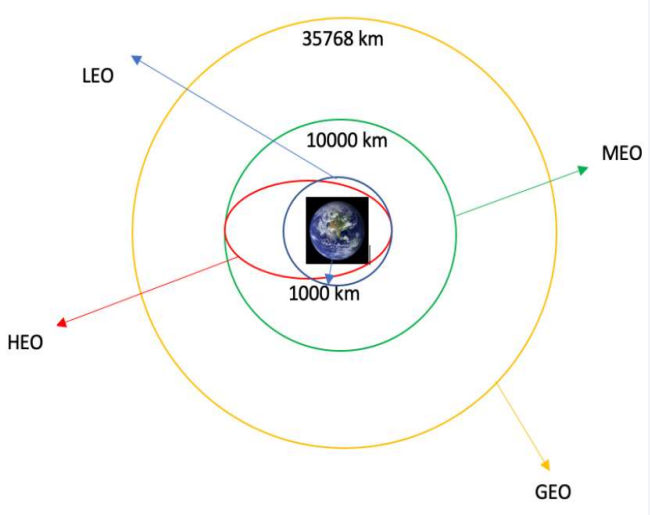
Create a `BeautifulSoup` object from the HTML `response`

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(data, 'html.parser')
```

```
In [13]: extracted_row = 0
         #Extract each table
         for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible"))
            # get table row
             for rows in table.find_all("tr"):
                 #check to see if first table heading is as number corresponding to launch a number
                 if rows.th:
                     if rows.th.string:
                         flight_number=rows.th.string.strip()
                         flag=flight_number.isdigit()
                 else:
```

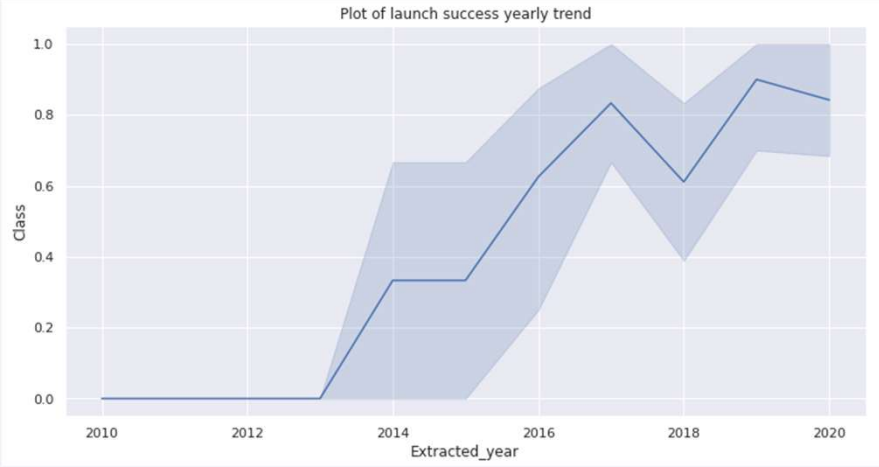
Data Wrangling

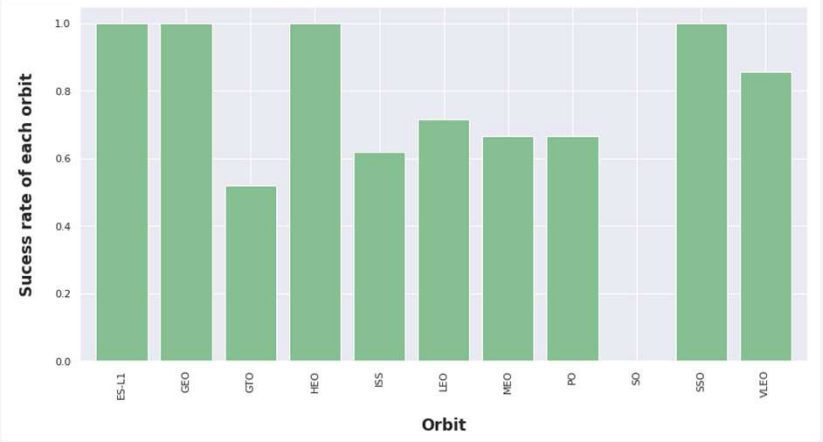
- Exploratory data analysis and determined the training labels;
- Calculated the number of launches at each site, and the number and occurrence of each orbits;
- Created landing outcome label from outcome column and exported the results to csv;
- Link notebook https://github.com/ignaciotolosa/IBM_Capstone_SpaceX/blob/main/notebook_EDA_Data_Wrangling_B5HCHoorN.ipynb



EDA with Data Visualization

- Explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend;
- Link notebook https://github.com/ignaciotolosa/IBM_Capstone_SpaceX/blob/main/notebook_EDA_Data_Visualization_4uqGlxM0j.ipynb





EDA with SQL

- Loaded the SpaceX dataset in a SQL database;
- Used EDA with SQL to get insight from the data using queries to find out for instance:
  - The names of unique launch sites in the space mission;
  - The total payload mass carried by boosters launched by NASA (CRS);
  - The average payload mass carried;
  - The total number of successful and failure mission outcomes;
  - The failed landing outcomes in drone ship, their booster version and launch site names;
- Link notebook https://github.com/ignaciotolosa/IBM_Capstone_SpaceX/blob/main/notebook_EDA_SQL_CnObVhU12.ipynb

Build an Interactive Map with Folium

- Marked all launch sites and added map objects for each site on the folium map;
- Assigned the feature launch outcomes (failure or success) to class 0 and 1;
- Identified which launch sites have relatively high success rate;
- Calculated the distances between a launch site to its proximities;

Link notebook
https://github.com/ignaciotolosa/IBM_Capstone_SpaceX/blob/main/notebook_Interactive_Visual_oLcCfOe8l.ipynb

Build a Dashboard with Plotly Dash

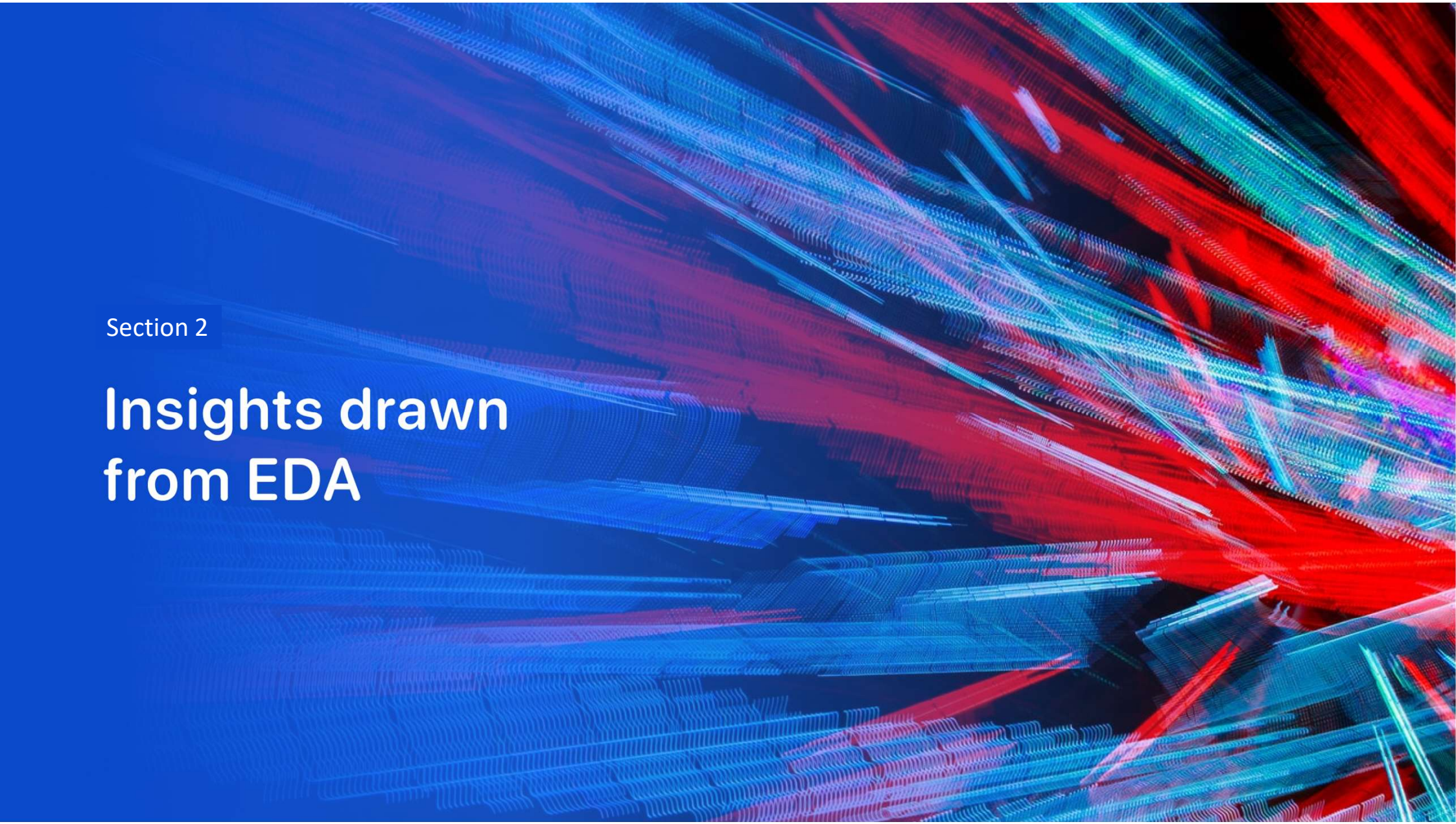
- Built an interactive dashboard with Plotly dash;
- Plotted pie charts showing the total launches by a certain sites;
- Plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version;
- Link notebook https://github.com/ignaciotolosa/IBM_Capstone_SpaceX/blob/main/dash_app.ipynb

Predictive Analysis (Classification)

- Loaded the data using numpy and pandas, transformed the data, split our data into training and testing;
- Built different machine learning models and tune different hyperparameters using GridSearchCV;
- Used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning;
- Found the best performing classification model;
- Link notebook https://github.com/ignaciotolosa/IBM_Capstone_SpaceX/blob/main/notebook_Prediction_-TLkE5KwQ.ipynb

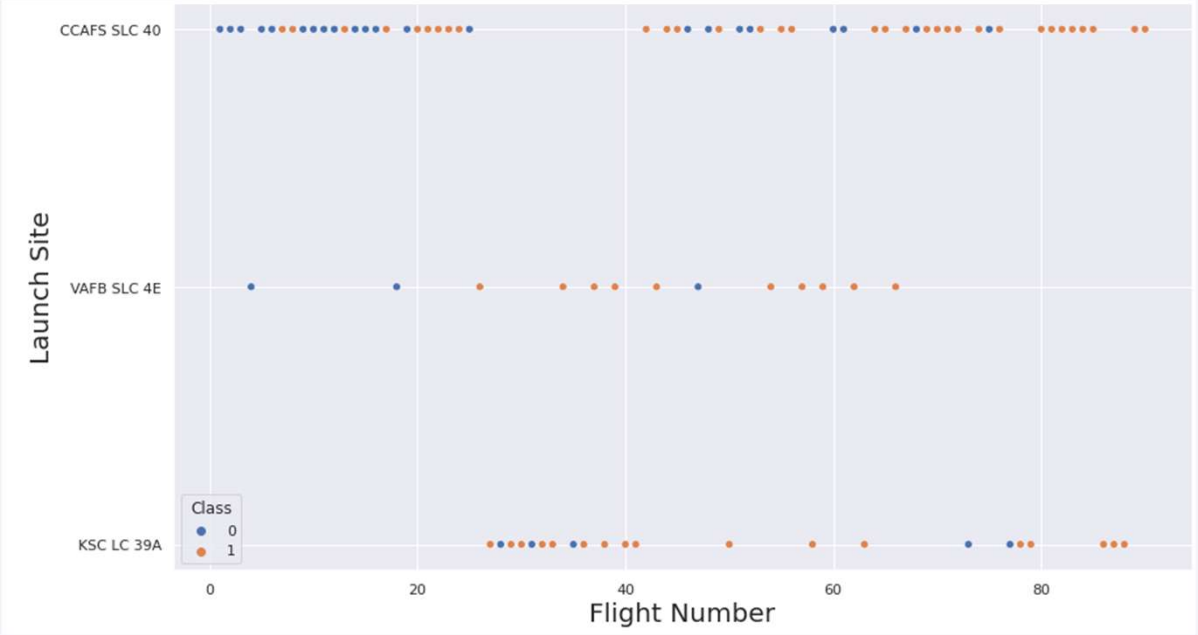
Results

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


Section 2
Insights drawn
from EDA

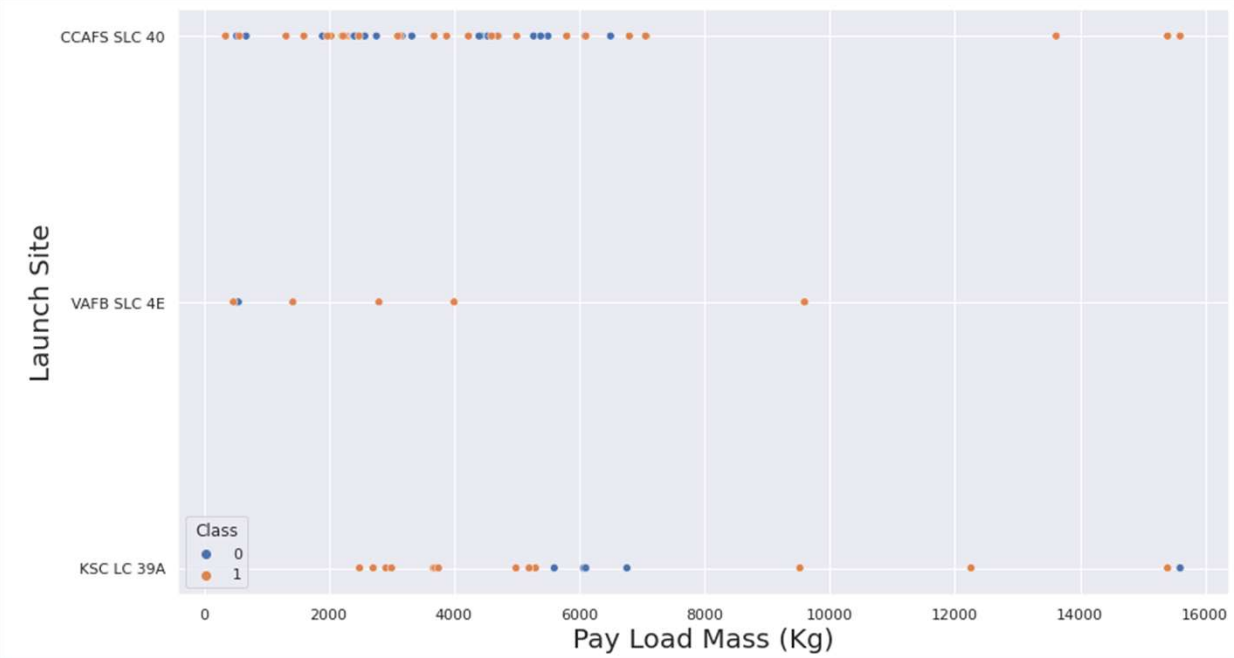
Flight Number vs. Launch Site

- From the plot, we can see that the larger the flight amount at a launch site, the greater the success rate at a launch site;



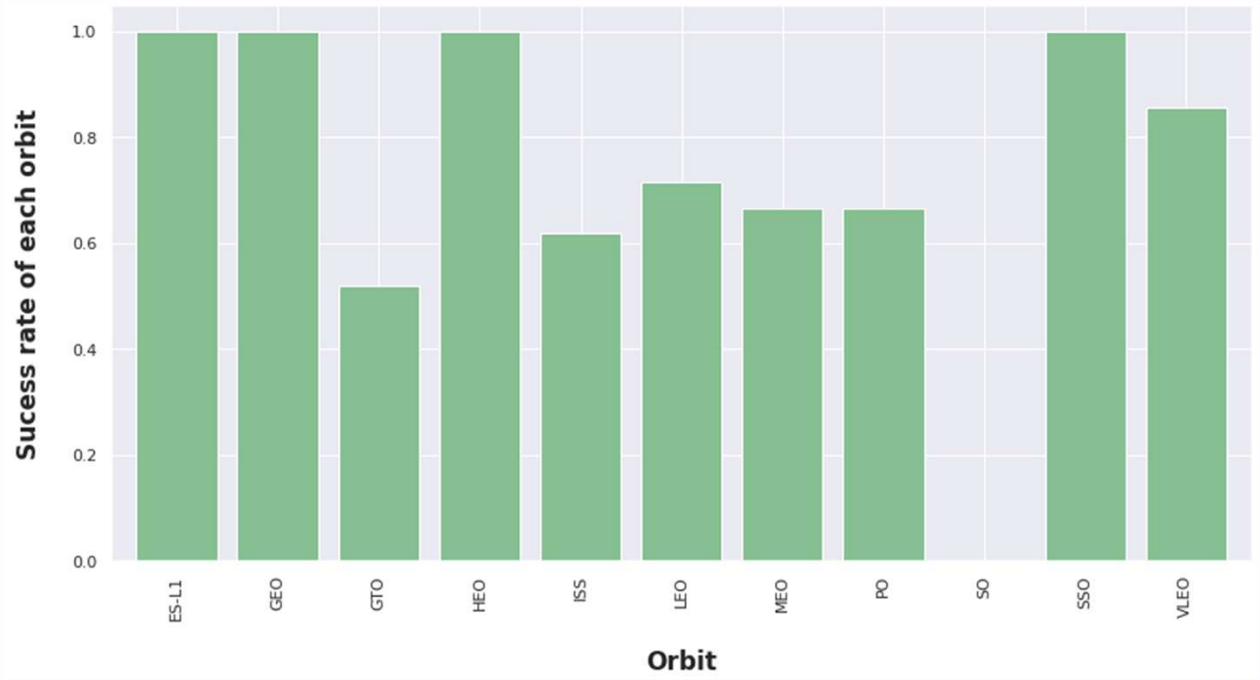
Payload vs. Launch Site

- The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket;



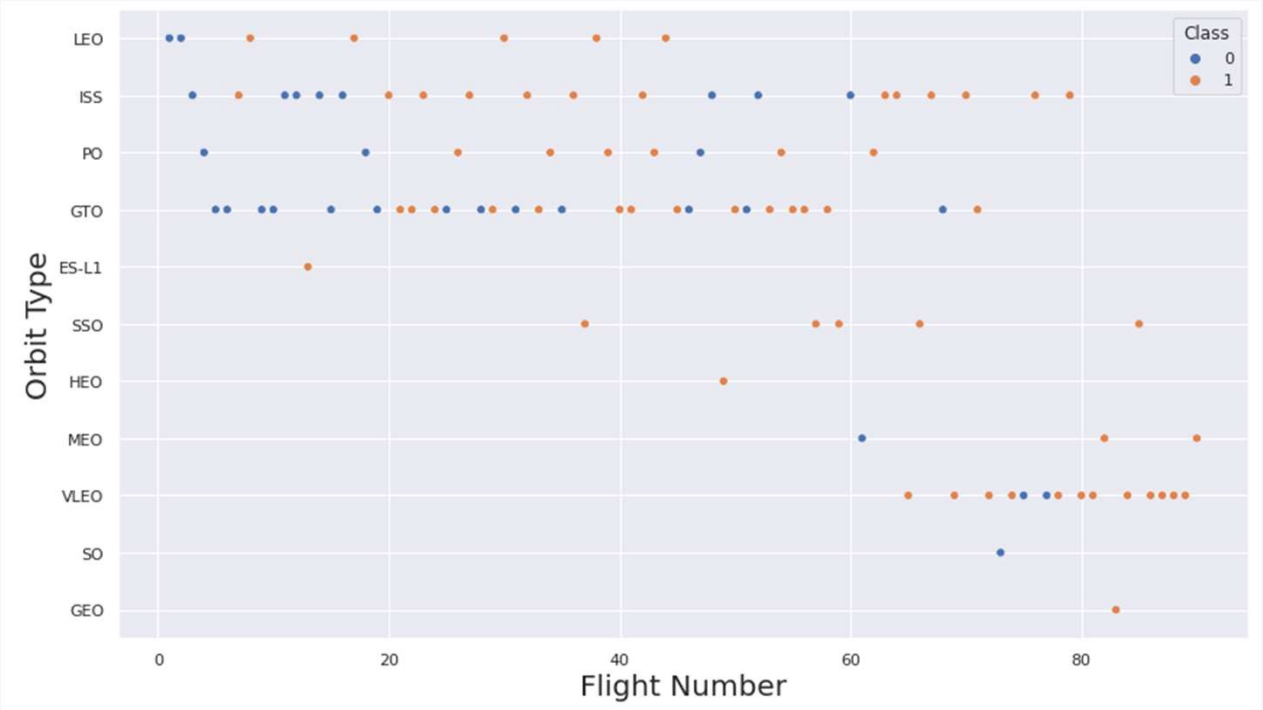
Success Rate vs. Orbit Type

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



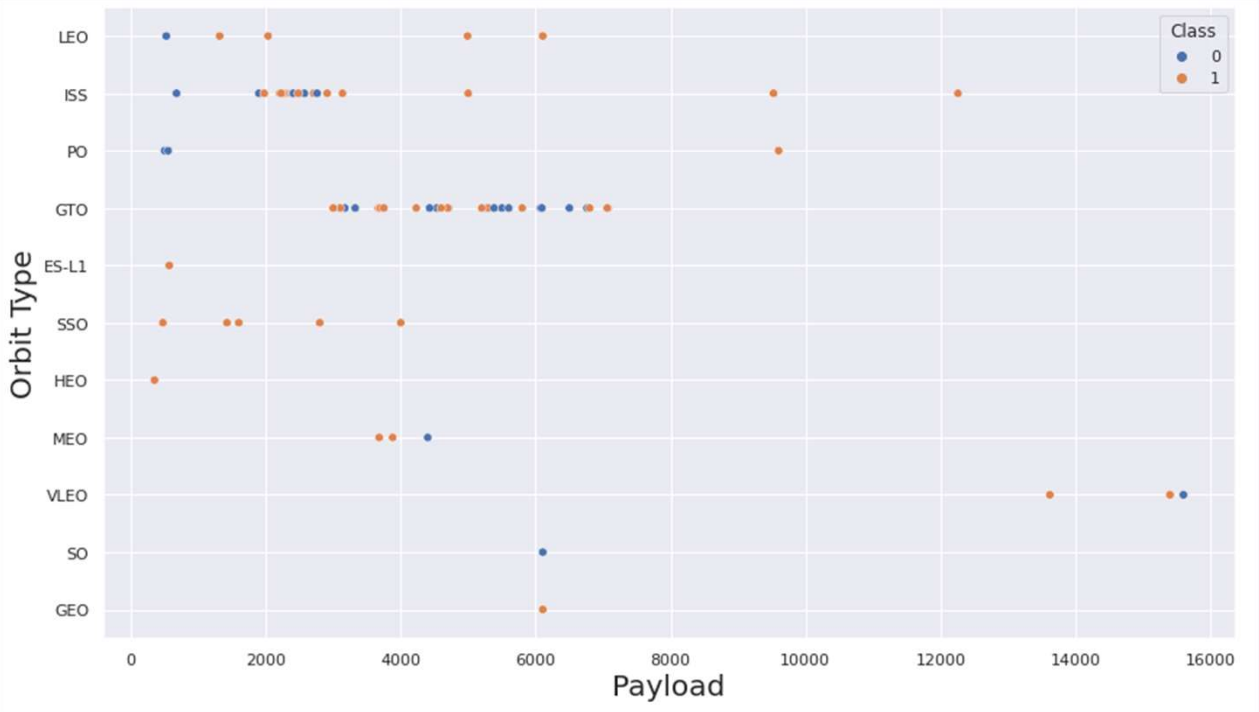
Flight Number vs. Orbit Type

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



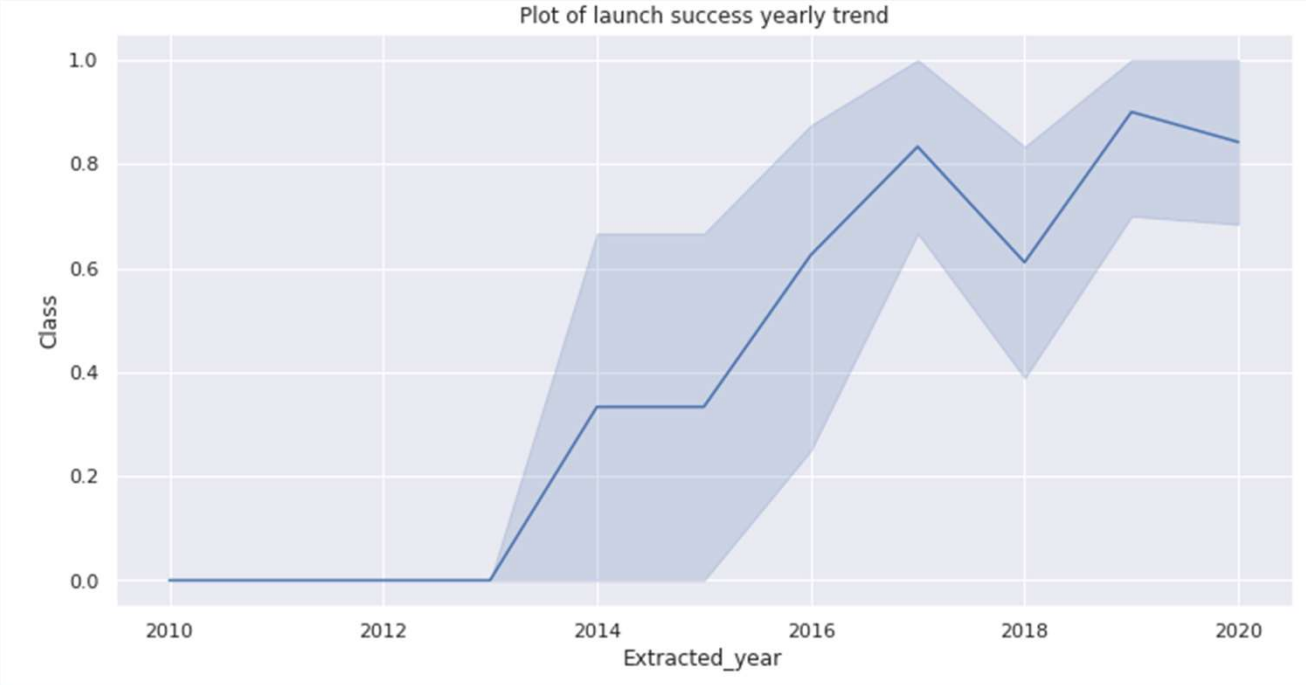
Payload vs. Orbit Type

- With heavy payloads, the successful landing are more for PO, LEO and ISS orbits;



Launch Success Yearly Trend

- Observe that success rate since 2013 kept on increasing untill 2020;



All Launch Site Names

- The key word DISTINCT shows only unique launch sites from the SpaceX data;

Task 1

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

In [11]:
```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEXTBL;
```

```
 * ibm_db_sa://sbc69777:***@a9e6-d59e-4883-8fc0-d6a8cba999f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[11]:

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

Launch Site Names Begin with 'CCA'

- The query displays 5 records where launch sites begin with `CCA`;

Task 2

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

In [16]:
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://sbc69777:***@a9e6-d59e-4883-8fc0-d6a8cba999f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.
cloud:31321/bludb
Done.
```

Out[16]:

DATE	Time (UTC)	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The total payload carried by boosters from NASA as 45596;

Task 3

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

In [22]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE CUSTOMER LIKE 'NASA (CRS)';
```

```
 * ibm_db_sa://sbc69777:***@a9e6-d59e-4883-8fc0-d6a8cba999f7a08f.c1ogj3sd0tgtu01qde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[22]:

1
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 equal 2928.4;

Task 4

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

In [23]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION LIKE 'F9 v1.1';
```

```
 * ibm_db_sa://sbc69777:***@a9e6-d59e-4883-8fc0-d6a8cba999f7a08f.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:31321/bludb
Done.
```

Out[23]:

1
2928

First Successful Ground Landing Date

- The dates of the first successful landing outcome on ground pad was 22/12/2015;

Task 5

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

Hint:Use min function

In [32]:
```
%sql SELECT MIN(Date) AS First_Successful_Landing FROM SPACEXTBL WHERE Landing _Outcome LIKE 'Success (gr
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

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

In [33]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' AND PAYLOAD_MAS
```

Out[15]:

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Used wildcard like % to filter for WHERE MissionOutcome was a success or a failure;

Task 7

List the total number of successful and failure mission outcomes

In [ ]:
```
%sql SELECT COUNT MISSION_OUTCOME AS "succesful mission "FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Succe
```

In [ ]:
```
%sql SELECT COUNT MISSION_OUTCOME AS "failure mission " FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Fail%'
```

Out[16]:

The total number of successful mission outcome is:

	successoutcome
0	100

The total number of failed mission outcome is:

	failureoutcome
0	1

Boosters Carried Maximum Payload

- Determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function;

Task 8

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

In [ ]:
```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SP
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

Out[17]:

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

2015 Launch Records

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

In [ ]:
```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

Out[18]:

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

- Selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 04/06/2010 to 20/03/2010;
- Applied the GROUP BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order;

In [ ]:
```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

Out[19]:

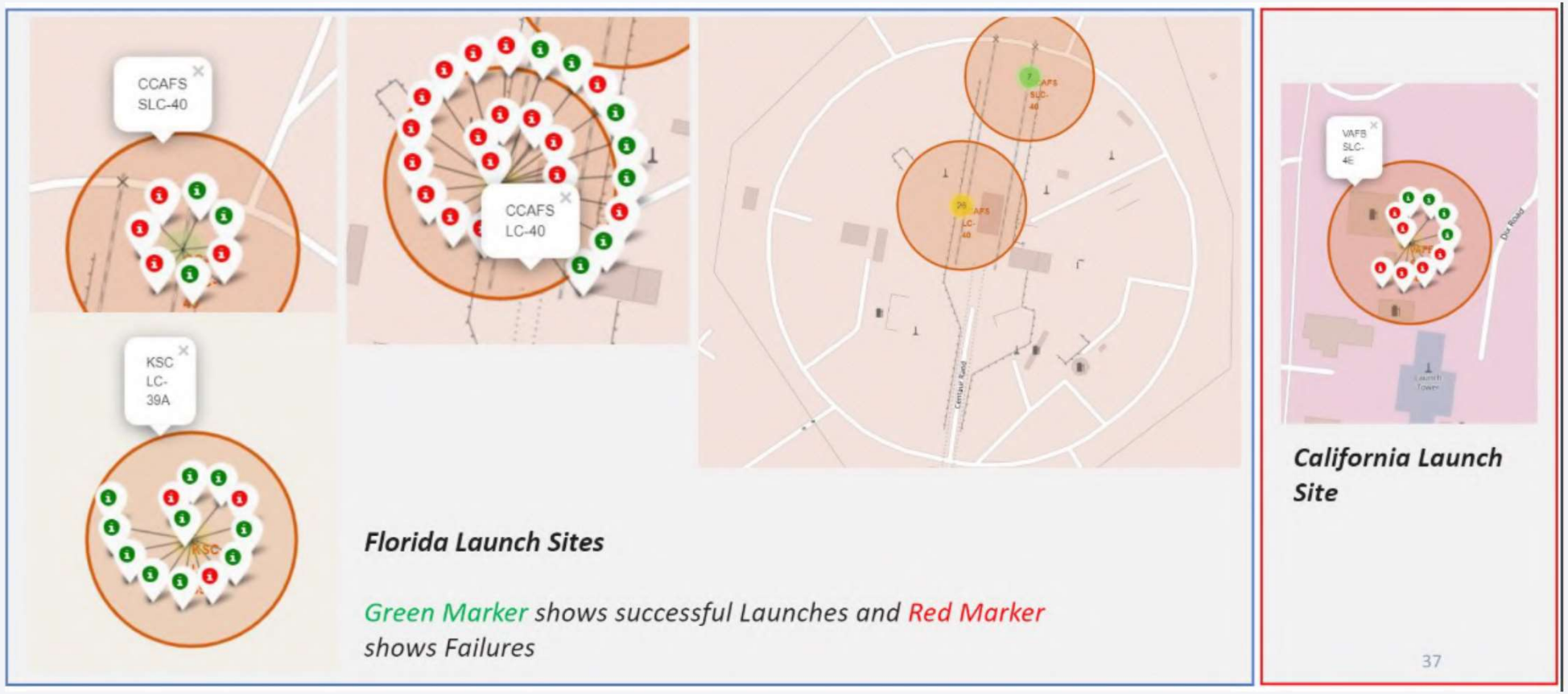
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1


Section 3
Launch Sites
Proximities Analysis

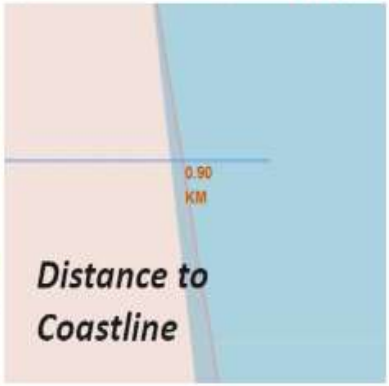
Location of all the Launch Sites

- All the SpaceX launch sites are located inside the United States

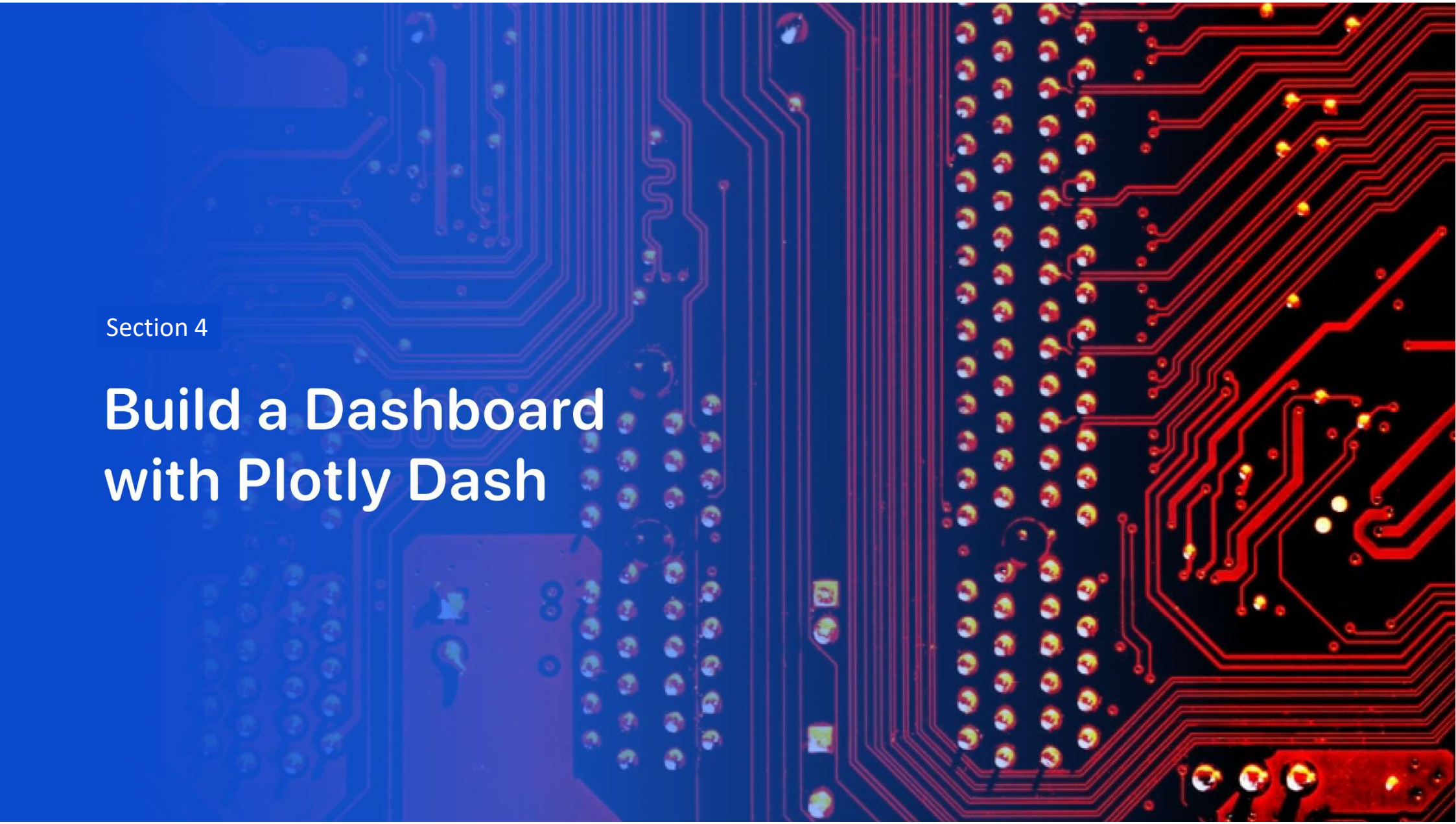
Markers showing launch sites with color labels



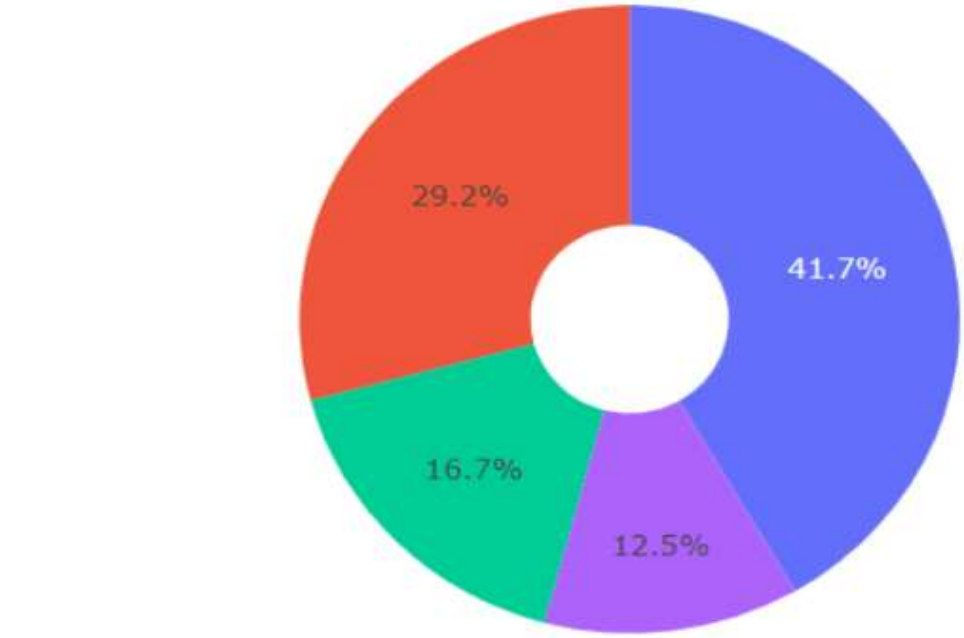
Launch Sites Distance to Landmarks



- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

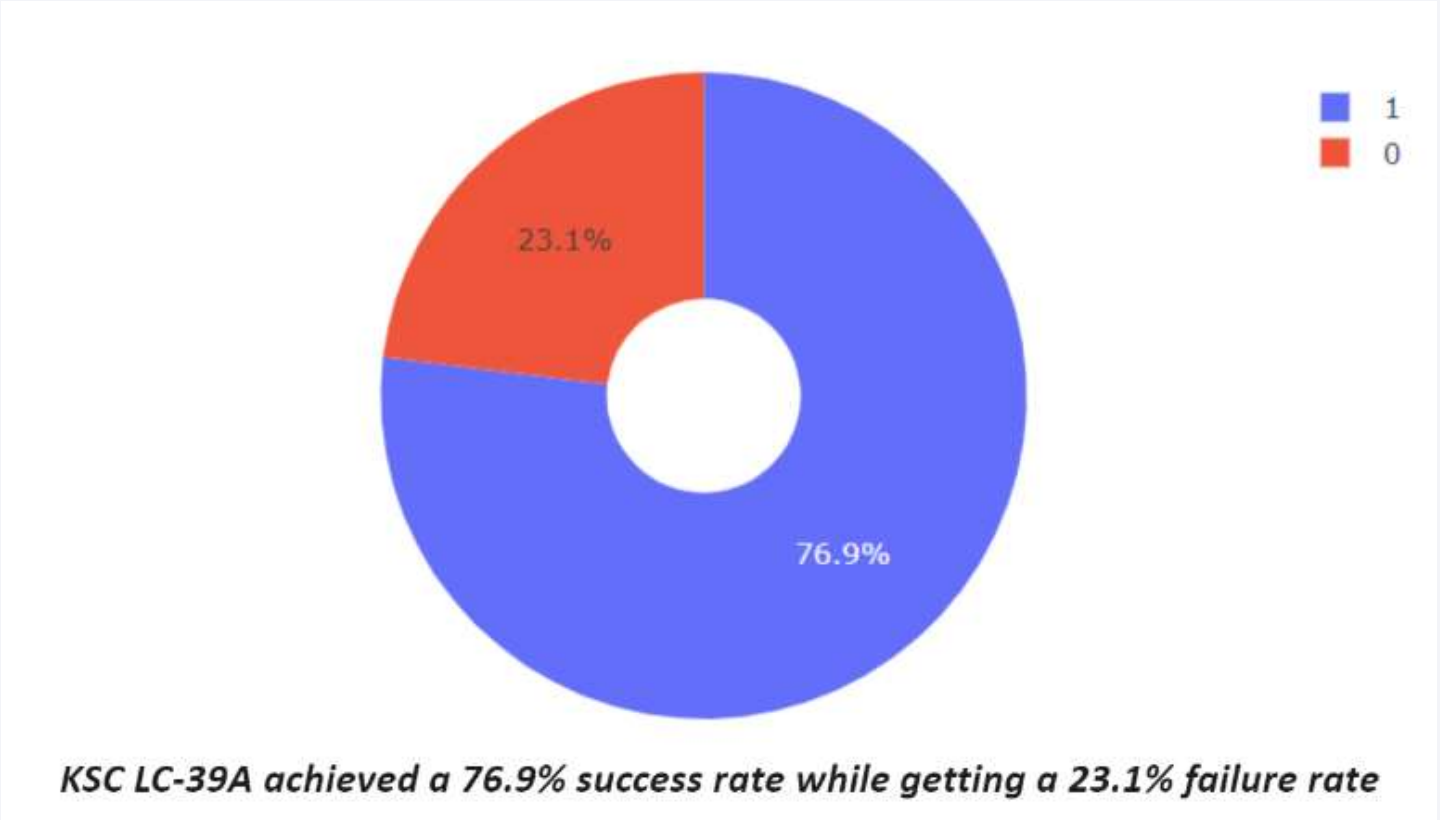

Section 4
Build a Dashboard with Plotly Dash

Success percentage by each sites



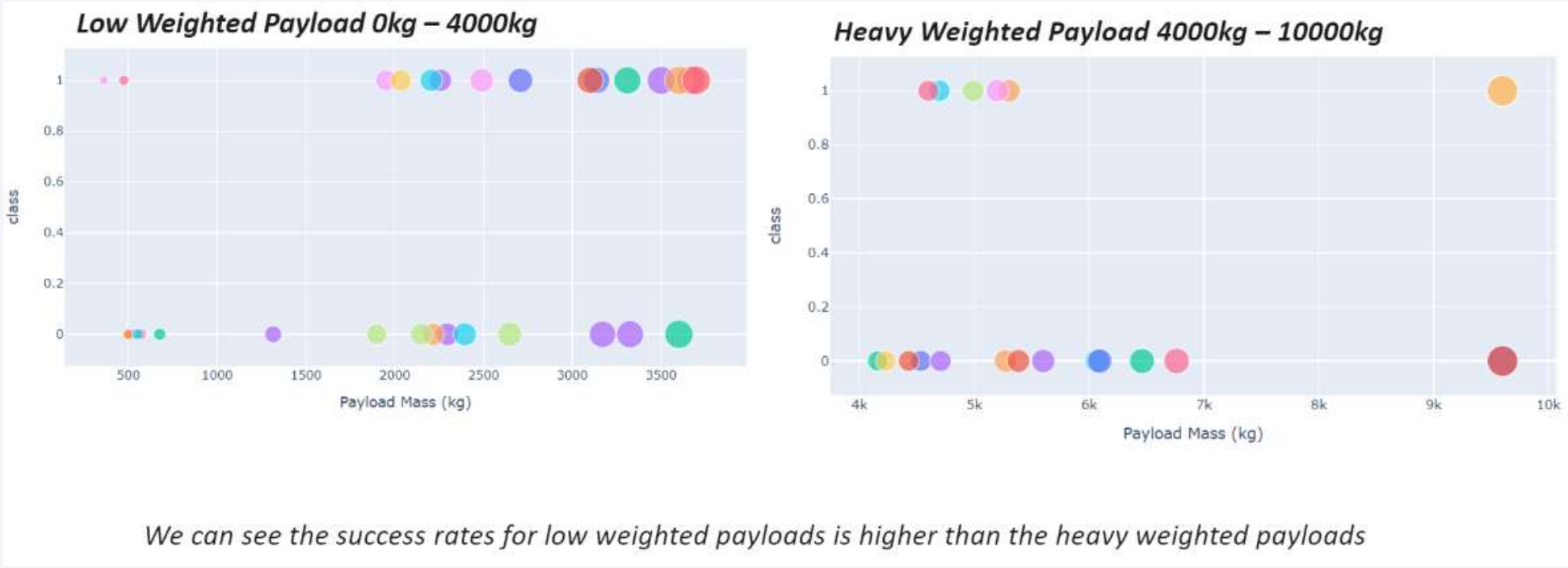
We can see that KSC LC-39A had the most successful launches from all the sites

The highest launch-success ratio: KSC LC-39A



KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs Launch Outcome Scatter Plot



We can see the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

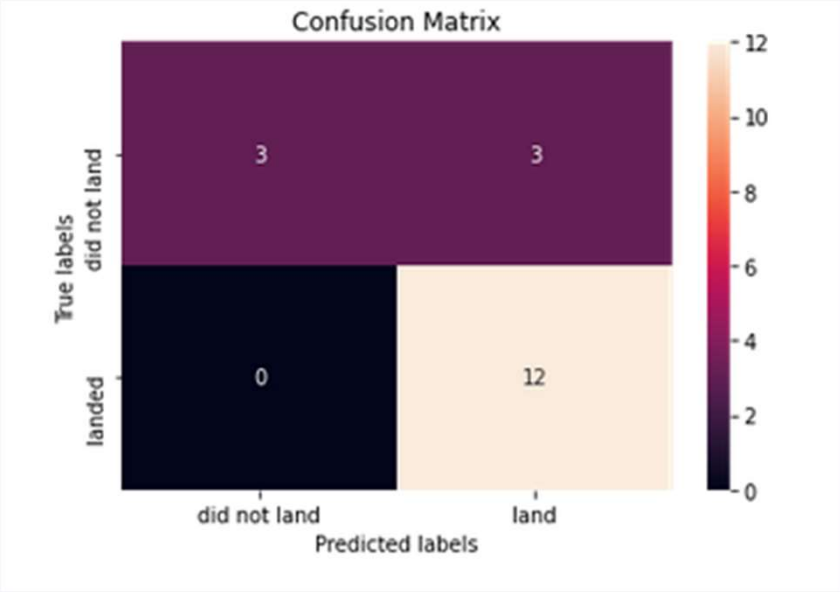
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter':
'random'}
```

Confusion Matrix

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



Conclusions

Follows:

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

Thank you!