IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sauvik Chaudhury
September 9, 2022

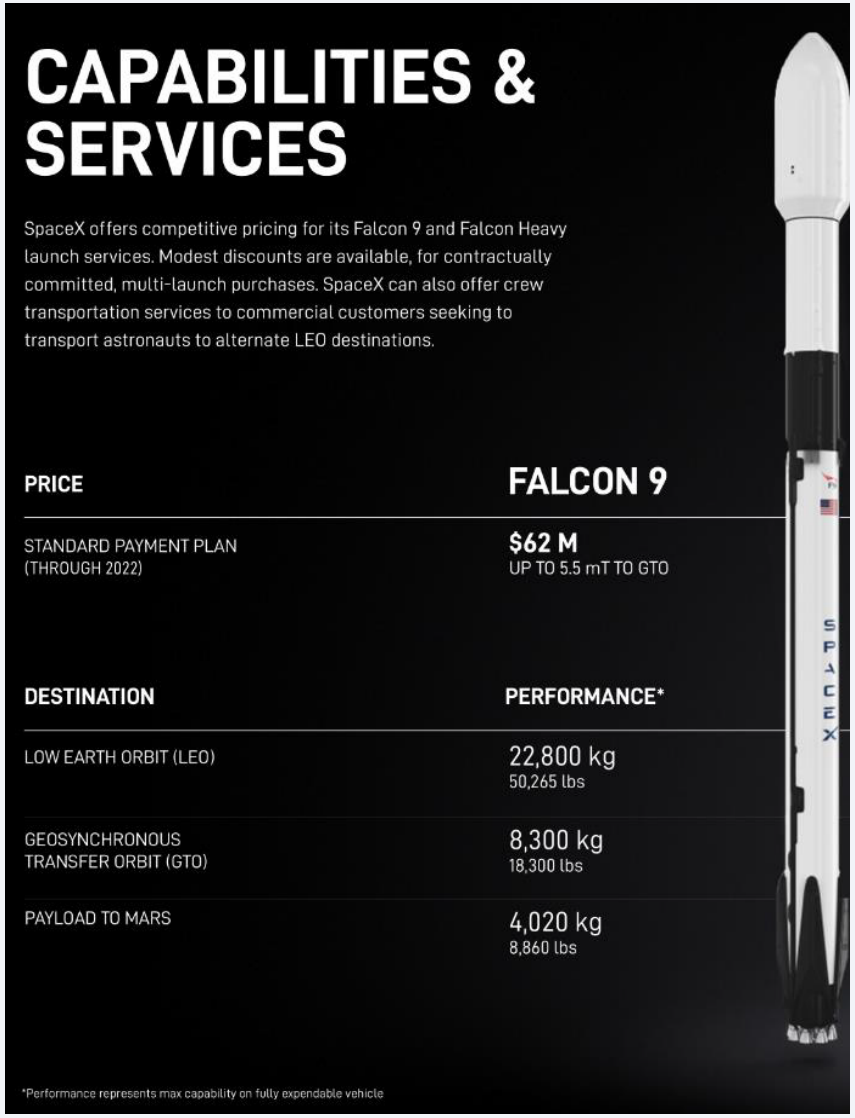
Outline

- 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
  - SpaceX advertises launch services starting at $62 million for missions that allow some fuel to be reserved for landing the 1st stage rocket booster, so that it can be reused
  - SpaceX public statements indicate a 1st stage Falcon 9 booster to cost upwards of $15 million to build without including R&D cost recoupment or profit margin
  - Therefore, if we can determine if the first stage will land, we can determine the cost of a launch
  - This information can be used by SpaceY to bid against SpaceX for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully
- Problems trying to find answers to
  - Factors determining if the rocket will land successfully?
  - Interaction amongst various features that determine the success rate of a successful landing
  - Operating conditions which needs to be in place to ensure a successful landing program

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia
- Perform data wrangling
  - One-hot encoding was applied to 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

Objective of this step is to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data has been collected using the below steps -

- Collected the data using get request to the 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 where necessary
- Performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup

Data Collection – SpaceX API

- Used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting

- The link to the notebook is https://github.com/sauvik258/Applied_Data_Science_Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

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

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

[10]: response.status_code

[10]: 200
```

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

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

```
[27]: # Calculate the mean value of PayloadMass column
payloadmassavg = data_falcon9['PayloadMass'].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].replace(np.nan, payloadmassavg, inplace=True)
```

Data Collection - Scraping

- Applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- Parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/sauvik258/Applied_Data_Science_Capstone/blob/main/jupyter-labs-webscraping.ipynb

[4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=102768692
```

[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`

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

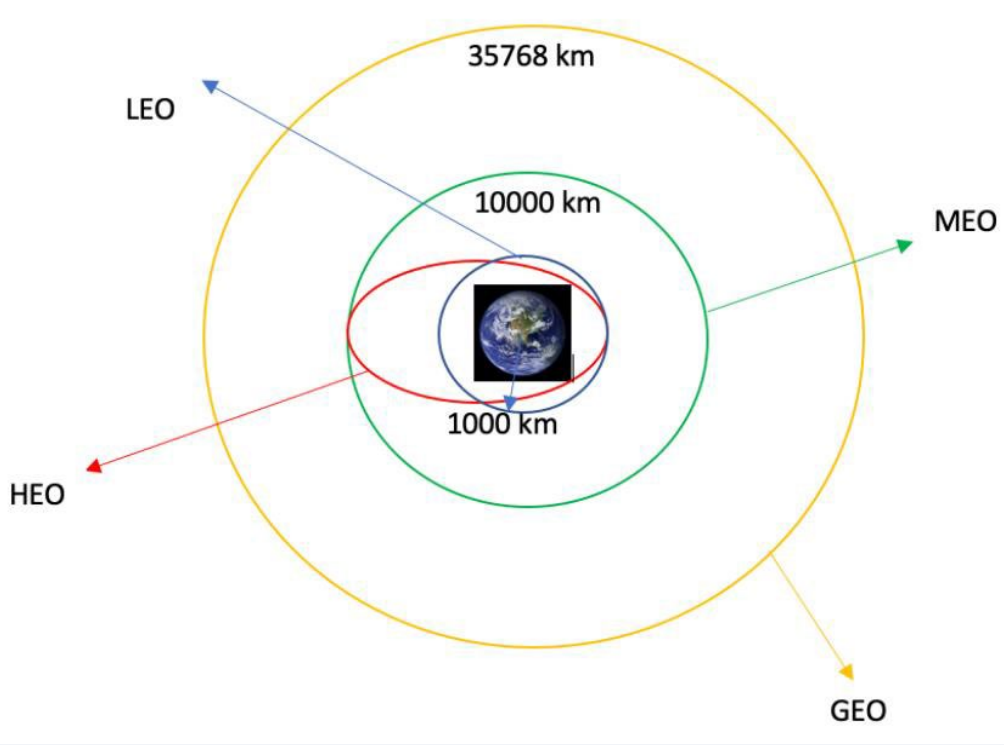
Print the page title to verify if the `BeautifulSoup` object was created properly

[7]:
```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

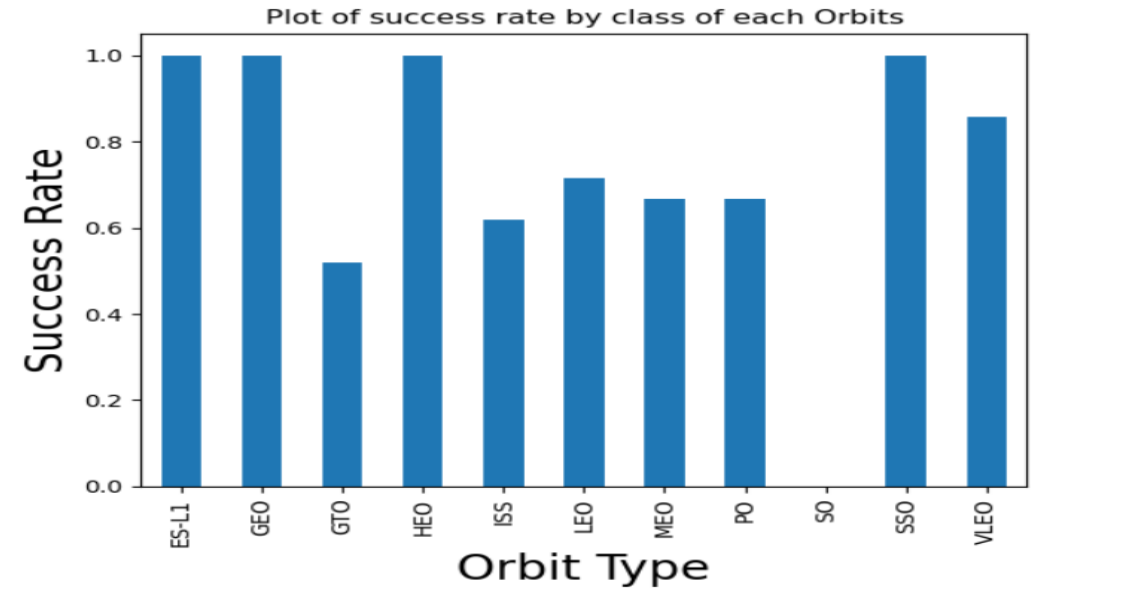
Data Wrangling

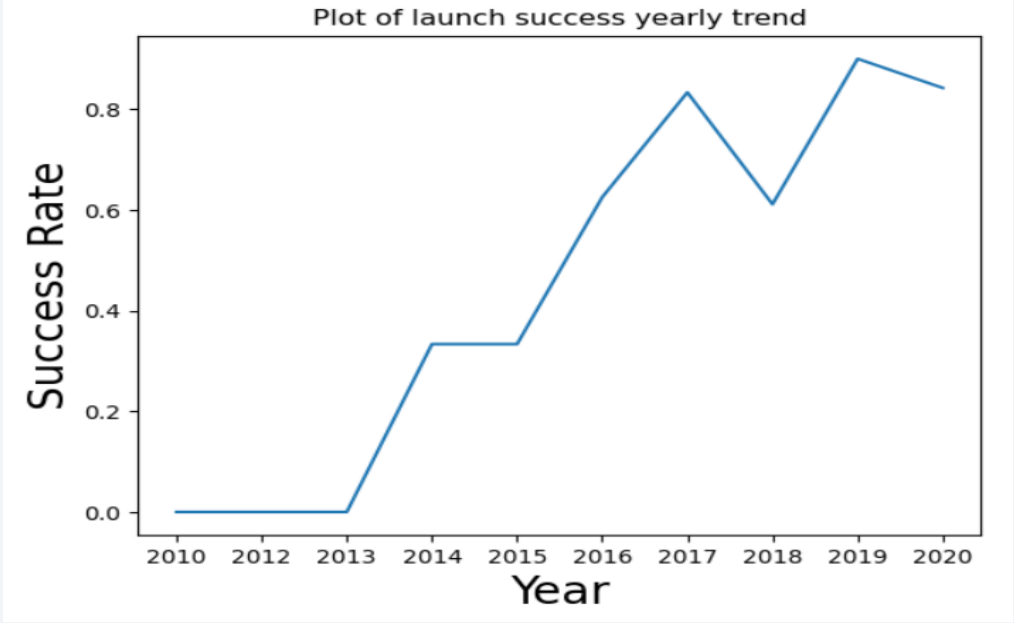
- Performed exploratory data analysis and determined the training labels
- Calculated the number of launches at each site, and the number and occurrence of each orbits
- Created landing outcome label from outcome column and exported the results to csv
- The link to the notebook is https://github.com/sauvik258/Applied_Data_Science_Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.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





- The link to the notebook is

https://github.com/sauvik258/Applied_Data_Science_Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb

EDA with SQL

- Loaded the SpaceX dataset into a DB2 database
- Applied EDA with SQL to get insight from the data
- Executed 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 by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names
- The link to the notebook is https://github.com/sauvik258/Applied_Data_Science_Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- 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.
- Assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Identified which launch sites have relatively high success rate, using the color-labeled marker clusters
- Calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines
  - Do launch sites keep certain distance away from cities
- The link to the notebook is https://github.com/sauvik258/Applied_Data_Science_Capstone/blob/main/lab_jupyter_launch_site_location.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
- The link of the notebook is https://github.com/sauvik258/Applied_Data_Science_Capstone/blob/main/spacex_dash_app.py

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
- The link of the notebook is https://github.com/sauvik258/Applied_Data_Science_Capstone/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

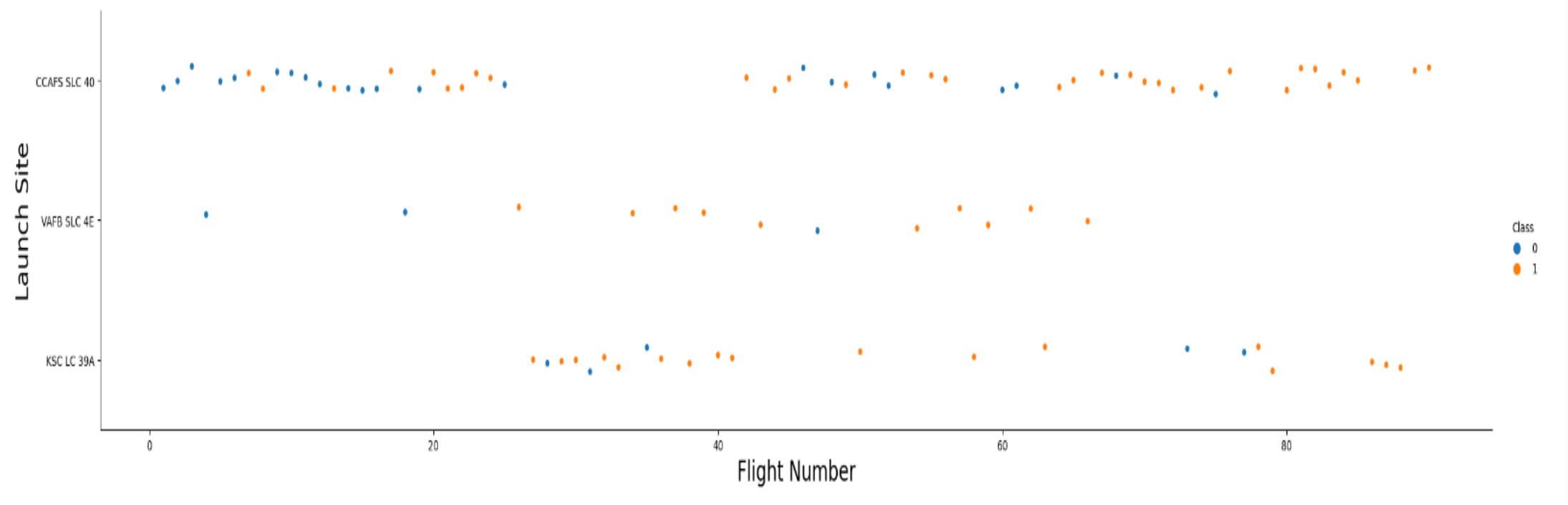
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

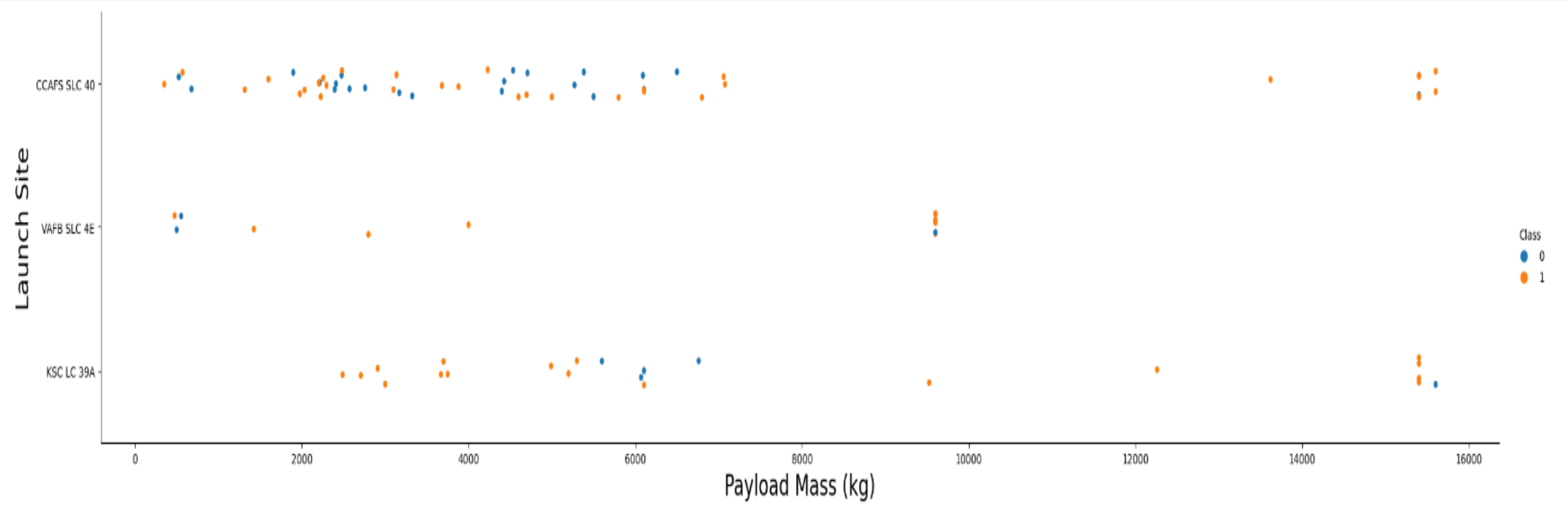
Flight Number vs. Launch Site

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



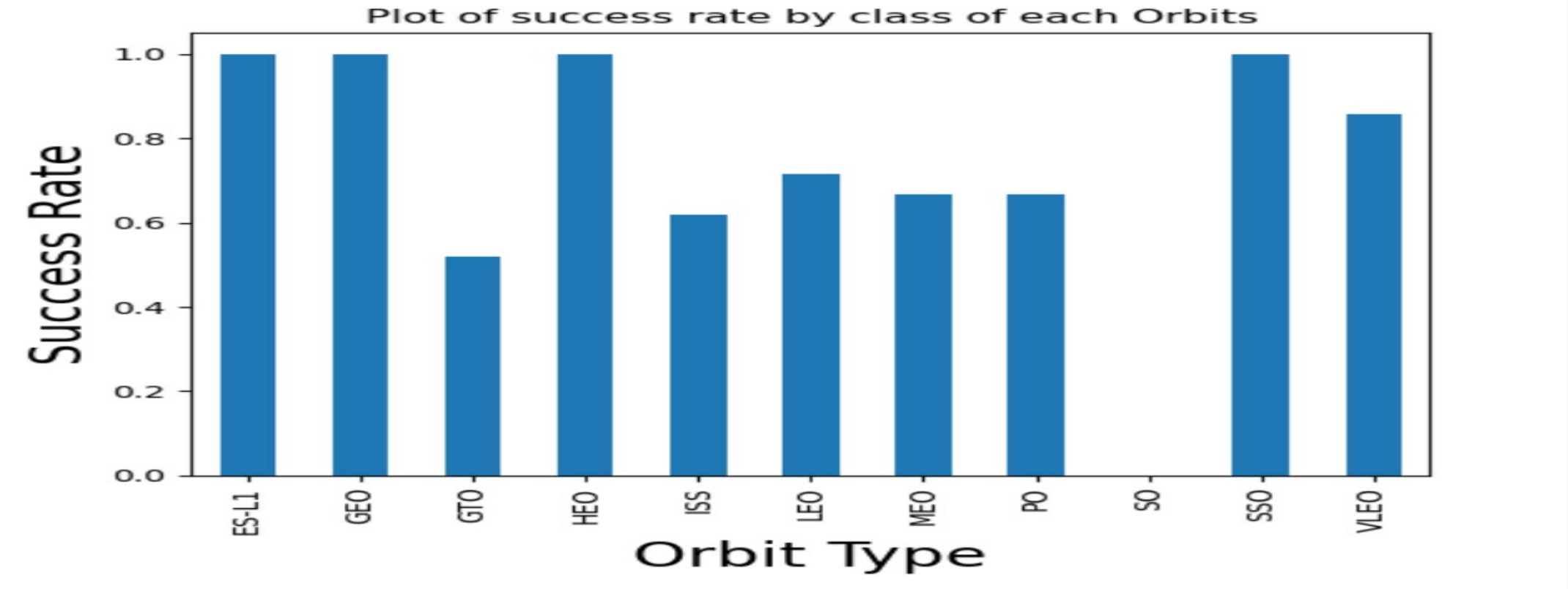
Payload vs. Launch Site

- From the plot, we found that greater the payload mass for launch site CCAFS SLC 40 the higher the success rate of the rocket



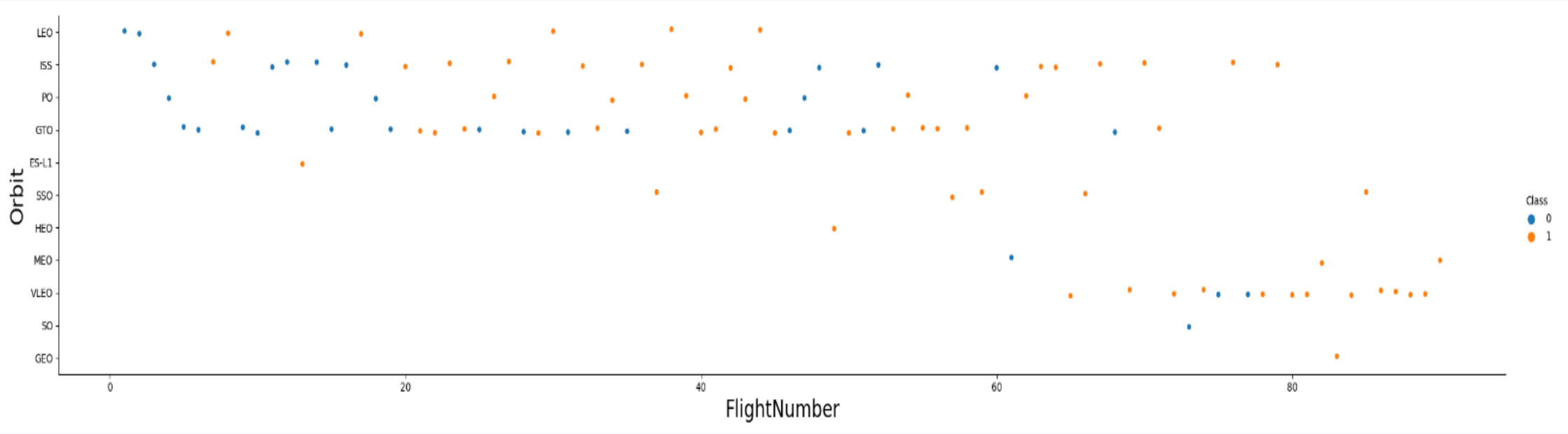
Success Rate vs. Orbit Type

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



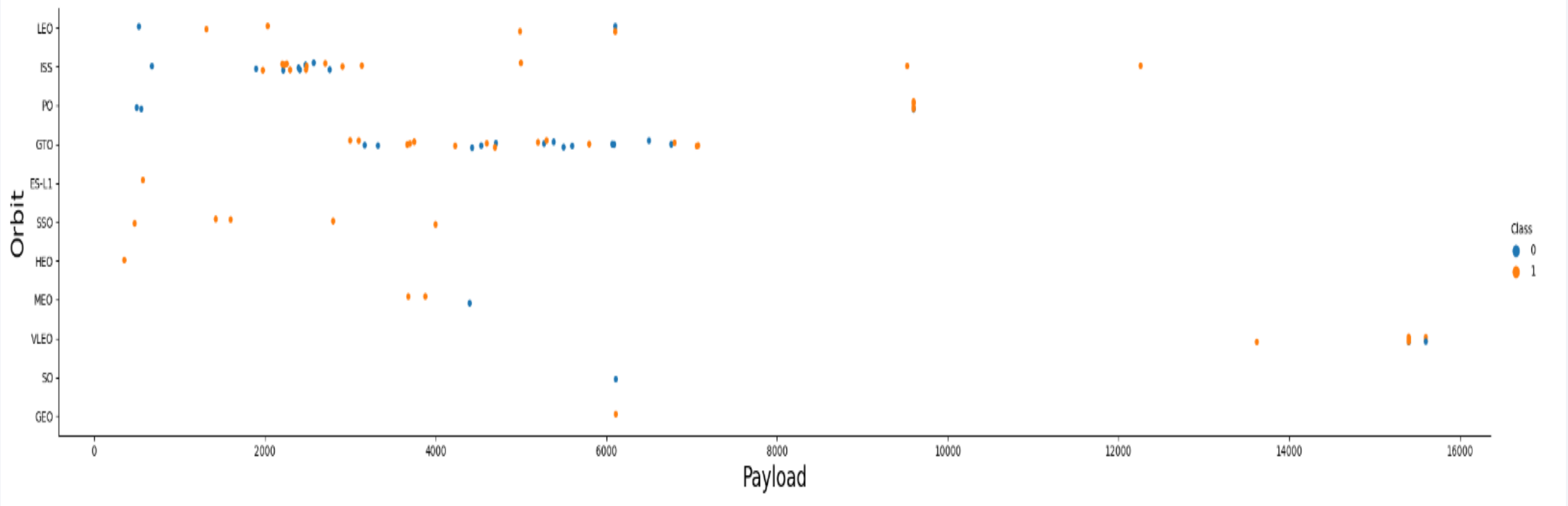
Flight Number vs. Orbit Type

- From the plot, we observe that in 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



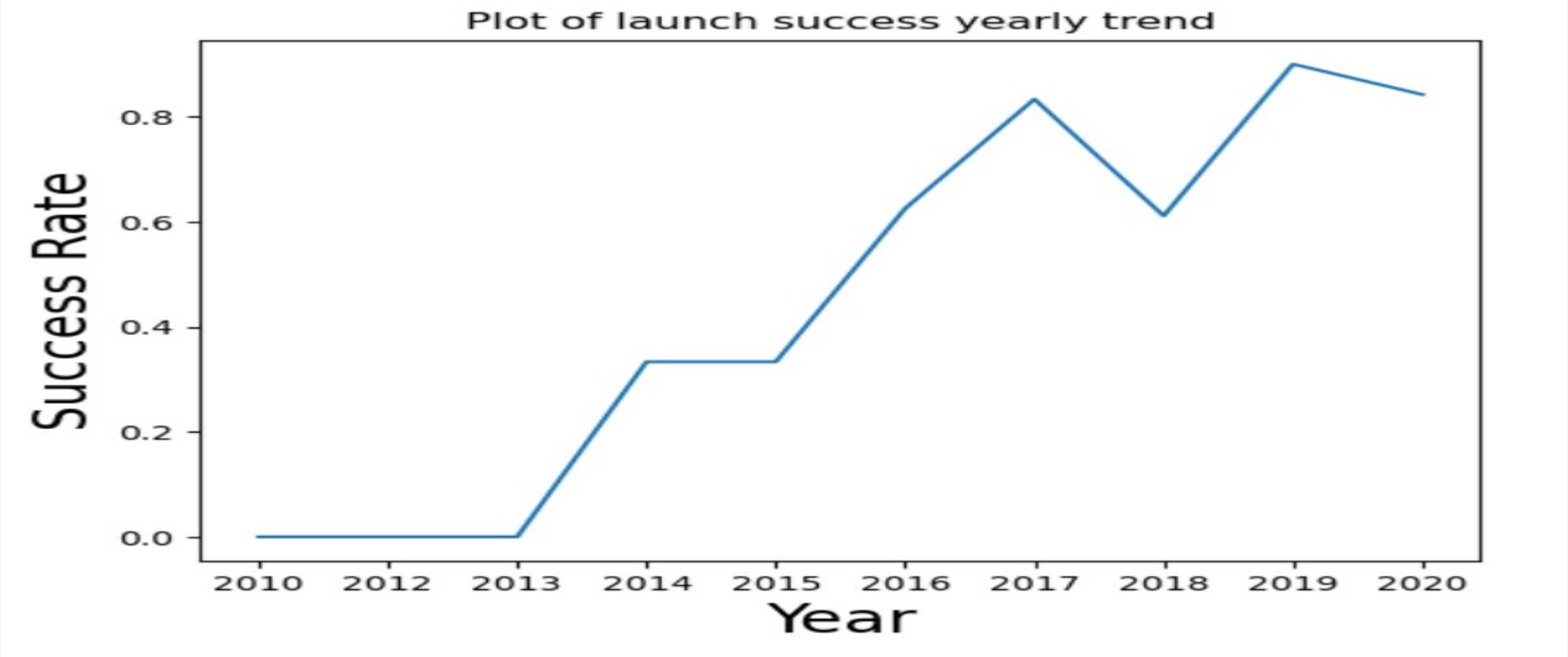
Payload vs. Orbit Type

- From the plot, we observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits



Launch Success Yearly Trend

- From the plot, we observe that success rate since 2013 kept on increasing till 2020



All Launch Site Names

- Used the key word **DISTINCT** to show only unique launch sites from the SpaceX data.

```
%sql SELECT DISTINCT LAUNCH_SITE  FROM SPACEXTBL
```

```
 * ibm_db_sa://fhl12841:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/bludb
   sqlite:///my_data1.db
Done.
```

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

Launch Site Names Begin with 'CCA'

- Used the **LIKE** keyword to find launch sites begin with `CCA` and used **LIMIT** keyword to display the top 5 rows

```
%sql SELECT * FROM SPACEXTBL 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
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

- Calculated the total payload carried by boosters from NASA as **45596** using the query below

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE Customer = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Calculated the average payload mass carried by booster version F9 v1.1 as **2928.4**

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

In [13]:
```
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

Out[13]:

	avg_payloadmass
0	2928.4

First Successful Ground Landing Date

- Observed that the dates of the first successful landing outcome on ground pad was **22nd December 2015**

```
%%sql
SELECT MIN(Date)
FROM SPACEXTBL
WHERE Landing__Outcome = 'Success (ground pad)';
```

```
 * ibm_db_sa://fhl12841:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/bludb
   sqlite:///my_data1.db
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- Used the **WHERE** clause to filter for 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
- The name of the Booster Versions are listed below

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

```
 * ibm_db_sa://fhl12841:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/bludb
   sqlite:///my_data1.db
Done.
```

booster_version
F9 FT B1021.1
F9 FT B1023.1
F9 FT B1029.2
F9 FT B1038.1
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

- Used wildcard **LIKE** '%' to filter for WHERE MissionOutcome was a success or a failure
- Observed that 100 missions are successful whereas 1 had failed

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://fhl12841:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/bludb
   sqlite:///my_data1.db
Done.
```

mission_outcome	total_number
Failure (in flight)	1
Success	99
Success (payload status unclear)	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

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

```
 * ibm_db_sa://fhl12841:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/bludb
   sqlite:///my_data1.db
Done.
```

booster_version
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

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

```
%%sql
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE
FROM SPACEXTBL
WHERE Landing__Outcome = 'Failure (drone ship)'
    AND YEAR(DATE) = 2015;
```

```
 * ibm_db_sa://fhl12841:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/bludb
   sqlite:///my_data1.db
Done.
```

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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** 2010-06-04 to 2010-03-20
- Applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order

```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY TOTAL_NUMBER DESC
```

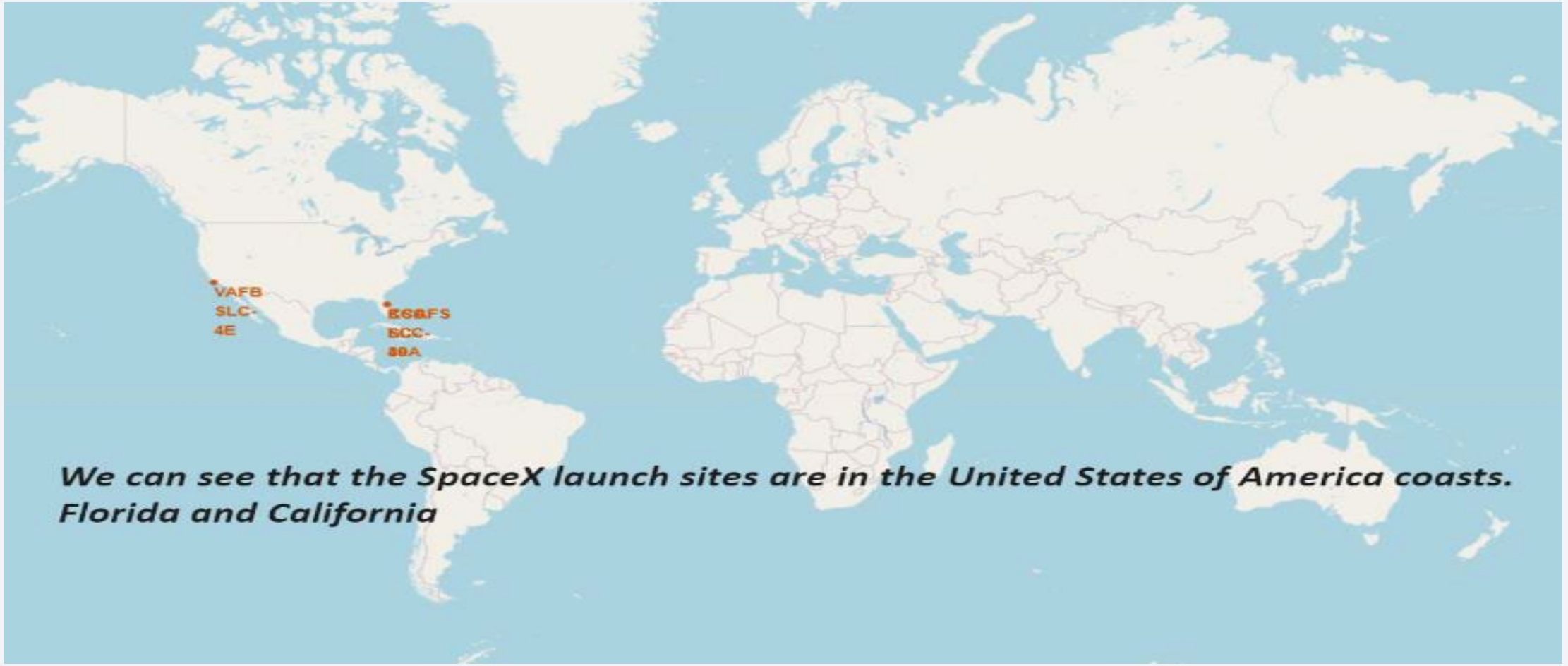
```
 * ibm_db_sa://fhl12841:***@3883e7e4-18f5-4afe-be8c-fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31498/bludb
   sqlite:///my_data1.db
Done.
```

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

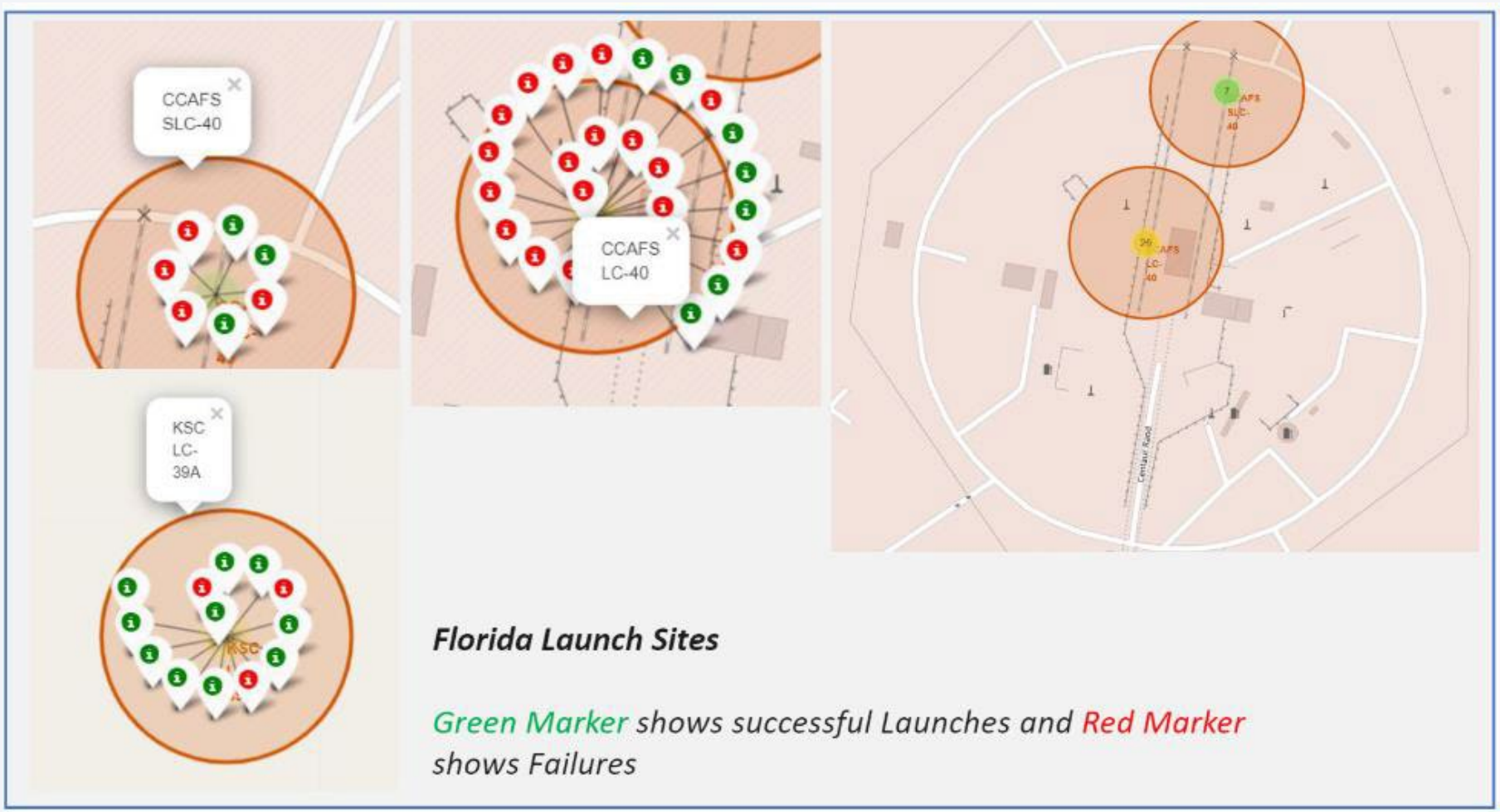
Section 3

Launch Sites Proximities Analysis

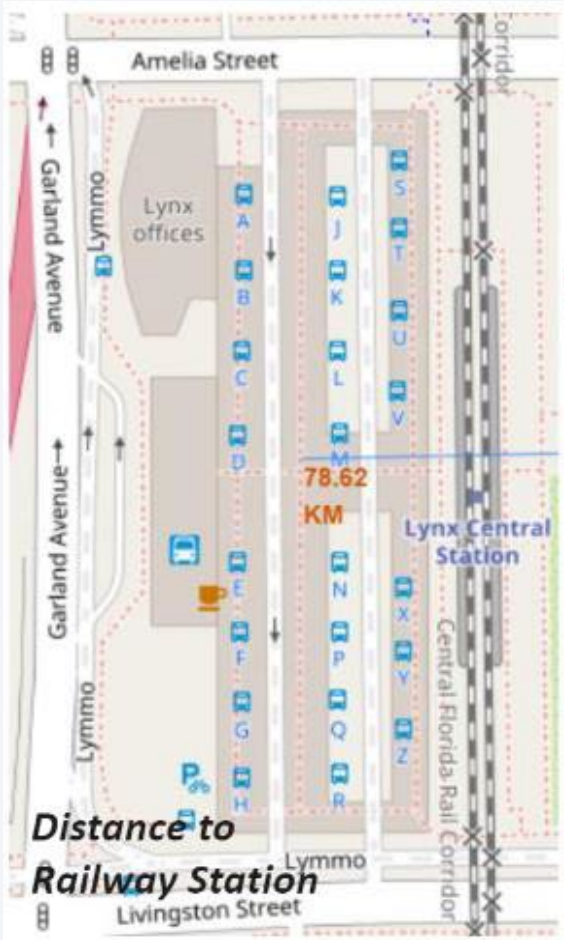
All Launch Sites Global Map Marker

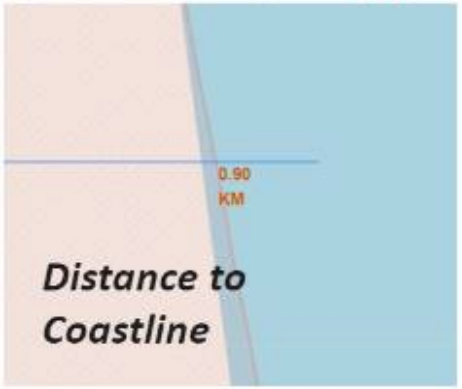


Markers Showing Launch Sites With Color Labels

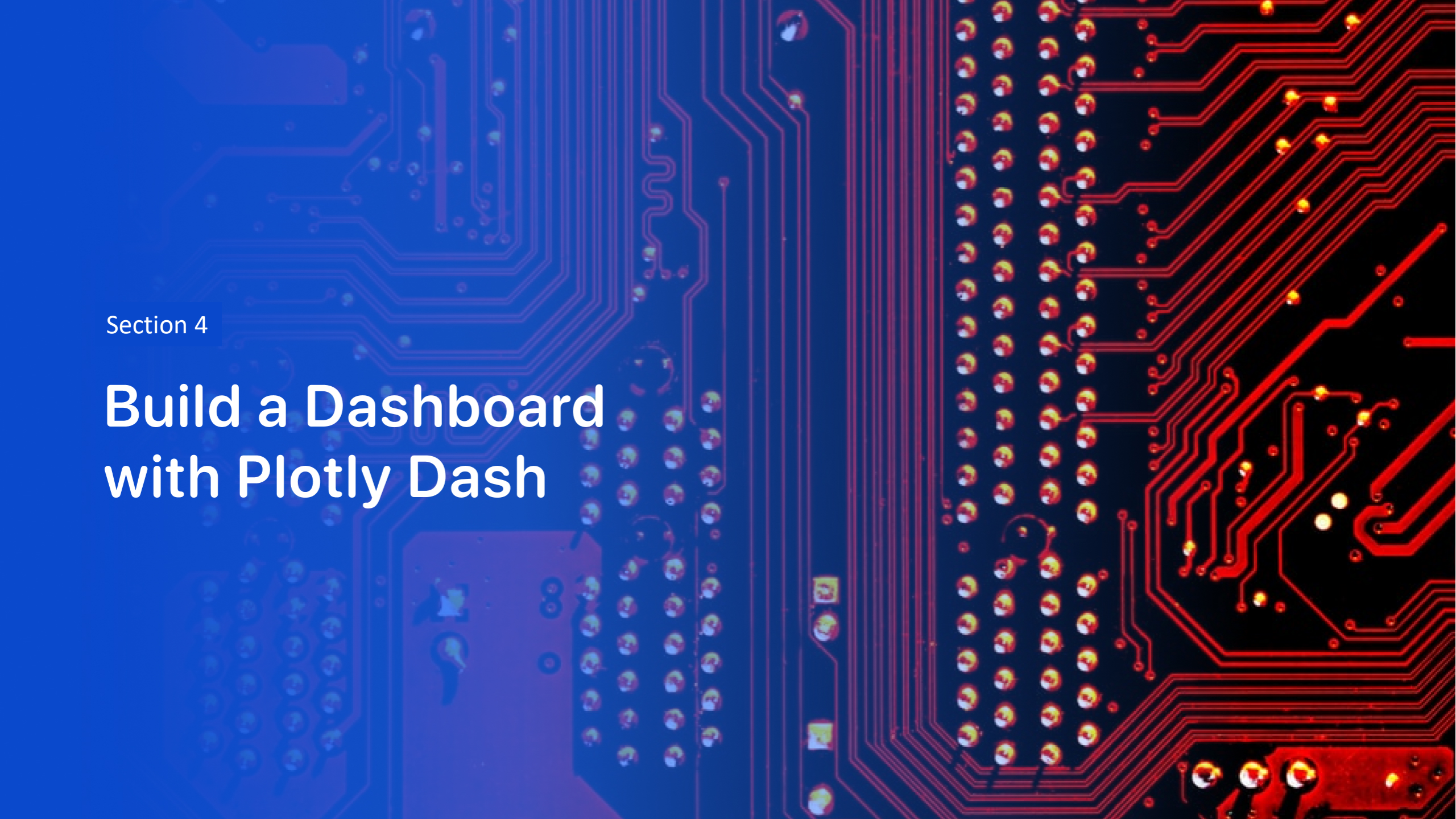


Launch Site Distance to Landmark



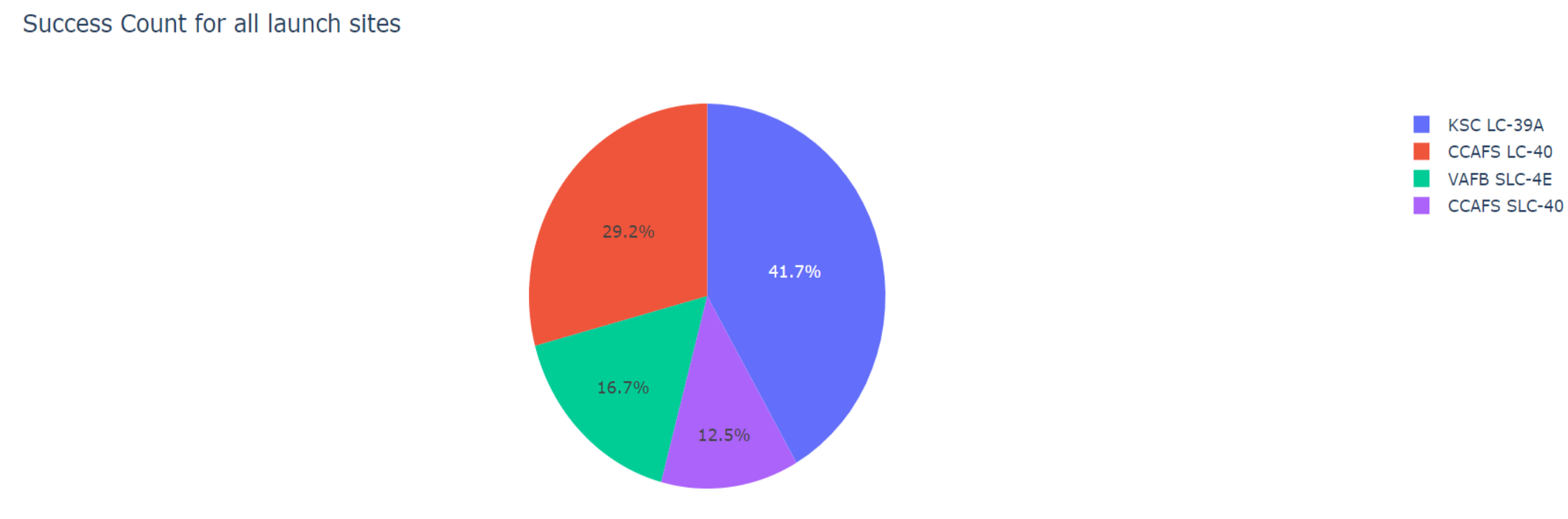


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes


Section 4
Build a Dashboard with Plotly Dash

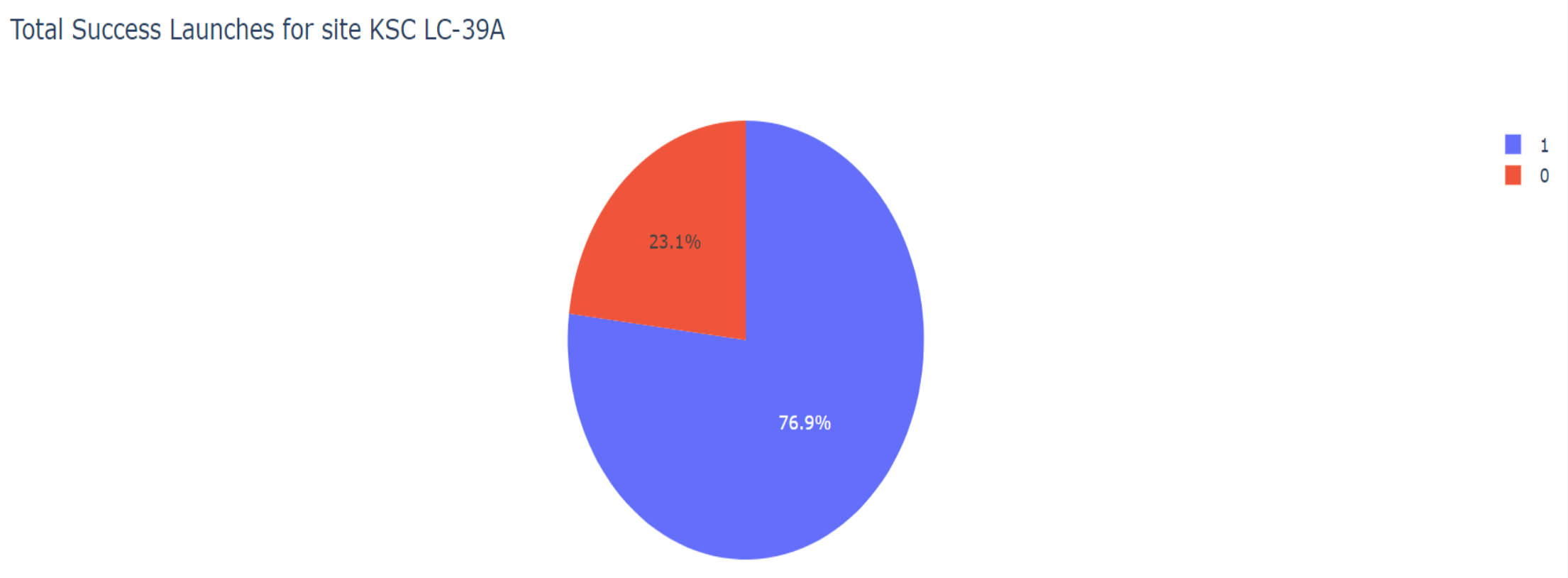
Pie Chart – Success Percentage Achieved by Launch Sites

- Observed that KSC LC-39A had the most successful launches from all sites



Pie Chart – Highest Launch Success Ratio

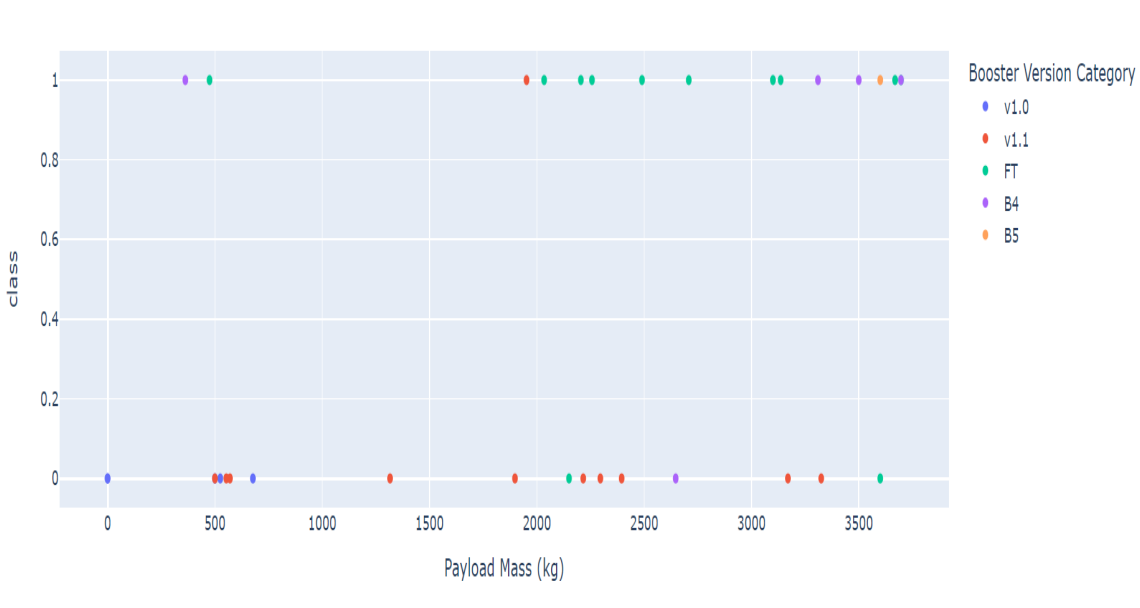
- KSC LC-39A achieved 76.9% success rate and 23.1% failure rate



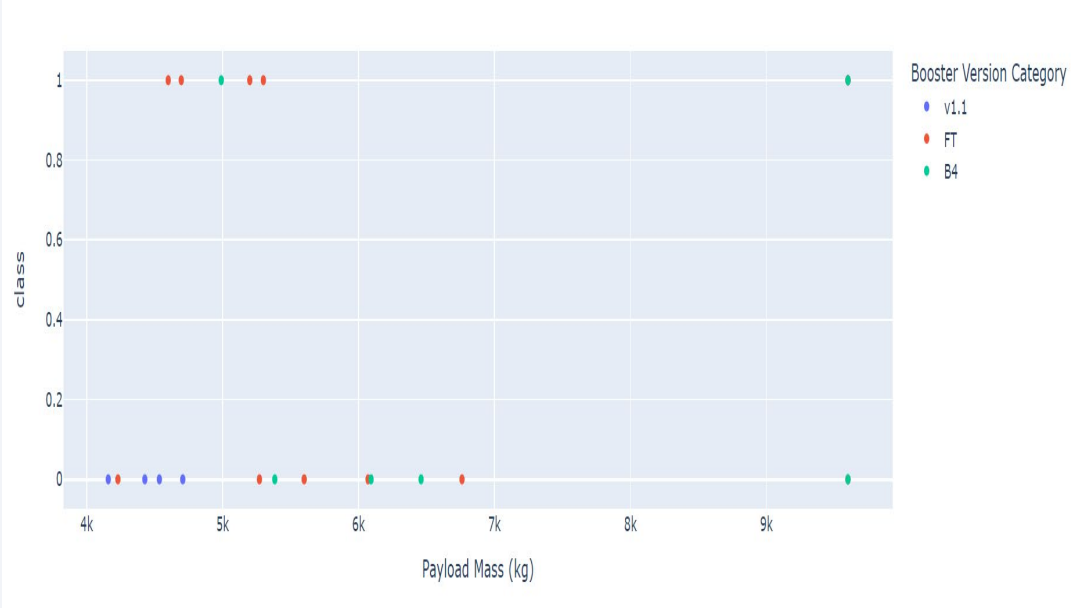
Scatter Plot – Payload Vs Launch Outcome (All Sites)

- Observed that success rate of low weighted payload is higher than heavy weighted payloads

Low Weighted payload 0KG – 4000KG



High Weighted payload 4000KG – 10000KG



Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

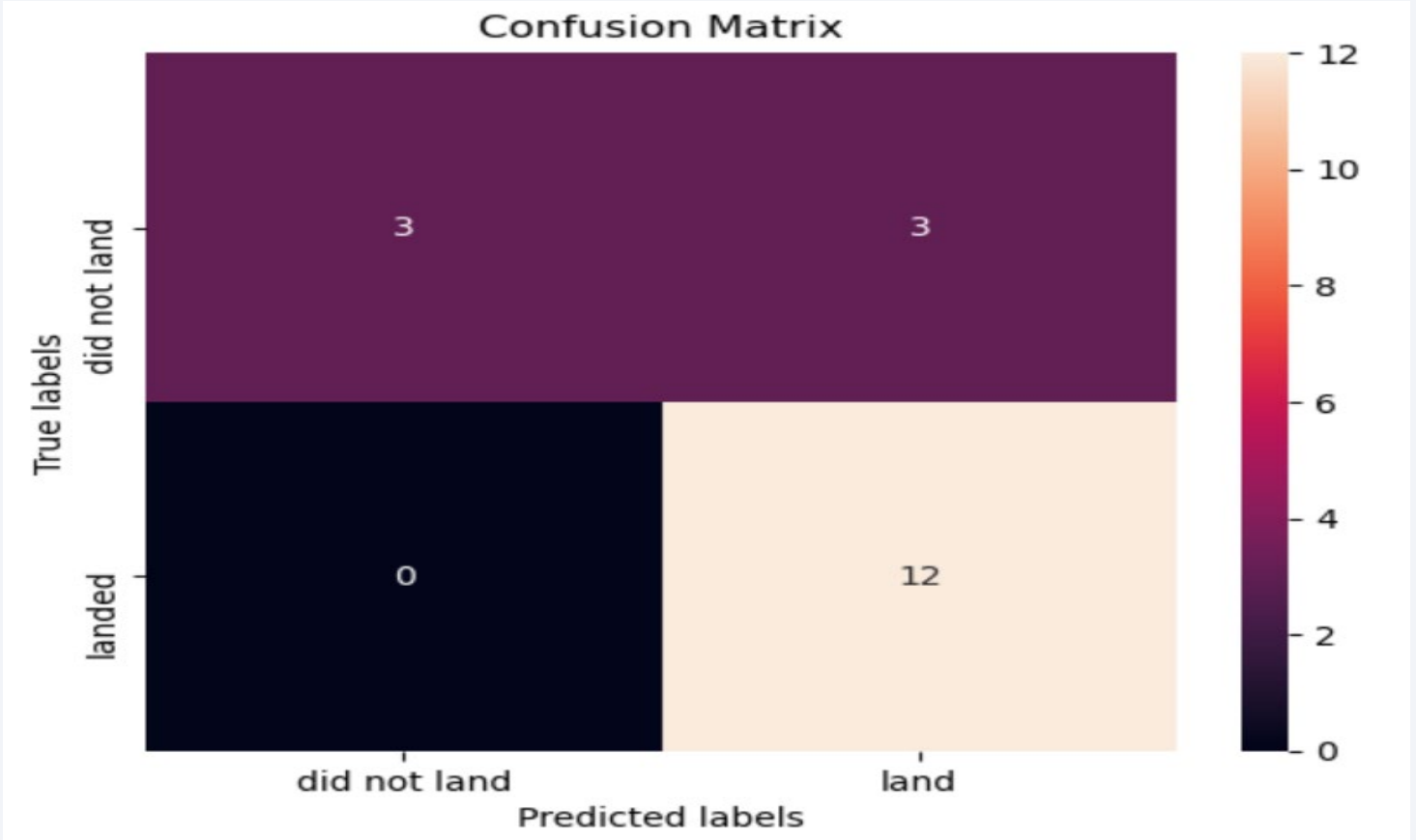
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.8888888888888888
Best params is : {'criterion': 'entropy', 'max_depth': 12, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 2, 'splitter': 'rando
m'}
```

Confusion Matrix

- The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes
- The major problem is false positives as evidenced by the models incorrectly predicting the 1st stage booster to land in 3 out of 18 samples in the test set



Conclusions

- 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
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate
- KSC LC-39A had the most successful launches of any sites
- The Decision tree classifier is the best machine learning algorithm for this task

Appendix

- Notebooks to to recreate dataset, analysis, and models can be found in repository: https://github.com/sauvik258/Applied_Data_Science_Capstone
- Acknowledgements
  - Thank You to Joseph Santarcangelo and other faculties at IBM for creating the course and materials
  - Thank You to Coursera for brining this course online

Thank you!