IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Saumalya Ghosh
23/2/2022

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 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 Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X 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 you want to find answers

  - What factors determine if the rocket will land successfully?
  - The interaction amongst various features that determine the success rate of a successful landing.
  - What operating conditions 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

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The 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

- 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/SaumalyaGhosh/IBM-DataScience-SpaceX-Capstone/blob/main/Data%20Collection%20API.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)
```

Check the content of the response

```
In [8]: print(response.content)
```


b'[{"fairings":{"reused":false,"recovery_attempt":false,"recovered":fal
2.imgbox.com/3c/0e/T8iJcSN3_o.png","large":"https://images2.imgbox.com/
h":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[
tch?v=0a_00nJ_Y88","youtube_id":"0a_00nJ_Y88","article":"https://www.sp
h.html","wikipedia":"https://en.wikipedia.org/wiki/DemoSat"},"static_fi
e_unix":1142553600,"net":false,"window":0,"rocket":"5e9d0d95eda69955f70
e":null,"reason":"merlin engine failure"}],"details":"Engine failure at
apsules":[],"payloads":["5eb0e4b5b6c3bb0006eeb1e1"],"launchpad":"5e9e45
t","date_utc":"2006-03-24T22:30:00.000Z","date_unix":1143239400,"date_l
r","upcoming":false,"cores":[{"core":"5e9e289df35918033d3b2623","flight
g_attempt":false,"landing_success":null,"landing_type":null,"landpad":n
d":null,"id":"5eb87cd9ffd86e000604b32a"},{"fairings":{"reused":false,"r
nks":{"patch":{"small":"https://images2.imgbox.com/4f/e3/I0lkuJ2e_o.png
ng"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":nu
l,"webcast":"https://www.youtube.com/watch?v=Lk4zQ2wP-Nc","youtube_id":

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/SaumalyaGhosh/IBM-DataScience-SpaceX-Capstone/blob/main/Complete%20the%20Data%20Collection%20with%20Web%20Scraping%20lab.ipynb

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_
```

Next, request the HTML page from the above URL and get a `response` object

TASK 1: Request the Falcon9 Launch Wiki page from its URL

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

Out[5]:
```
200
```

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

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

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

In [7]:
```
# Use soup.title attribute
soup.title
```

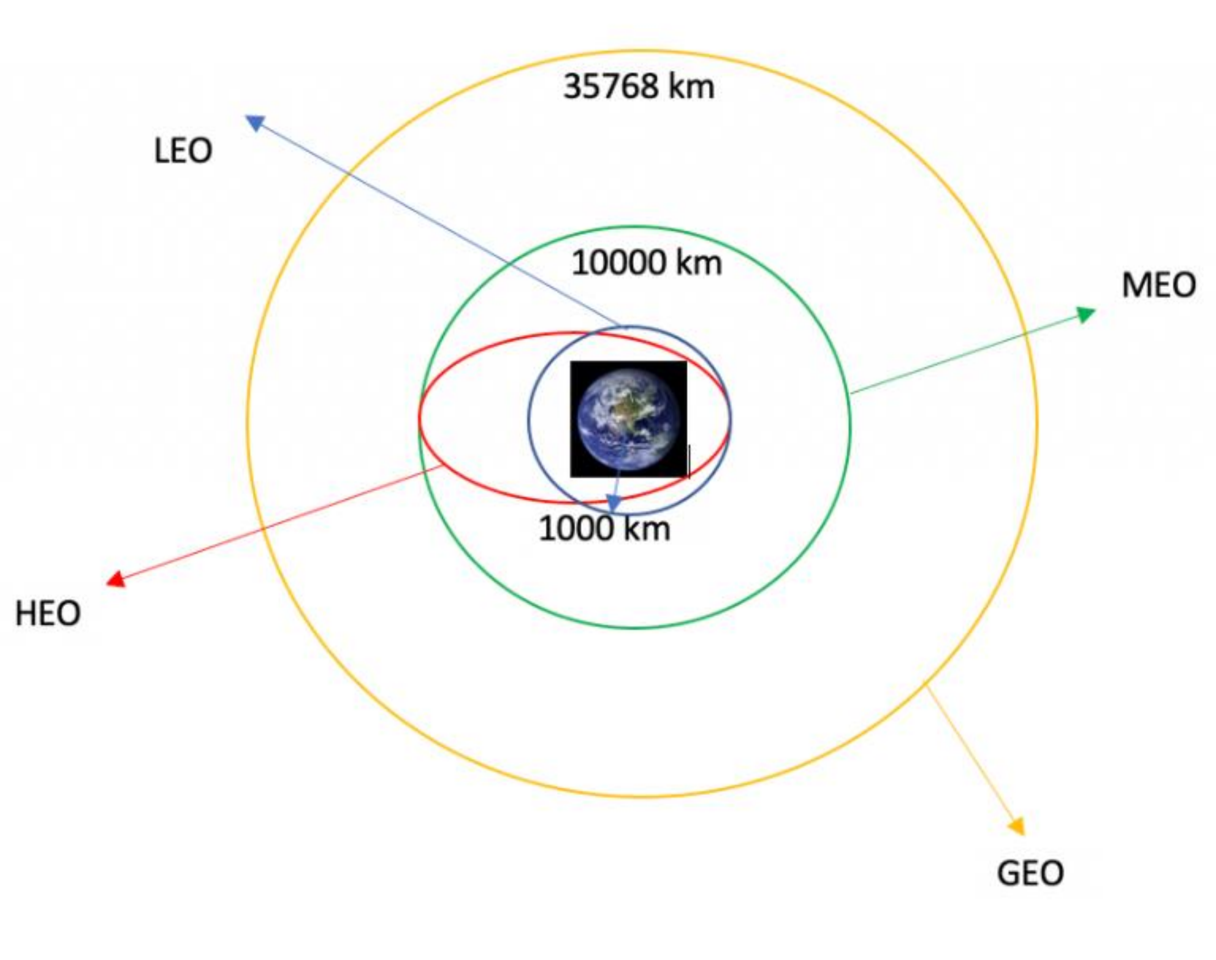
Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

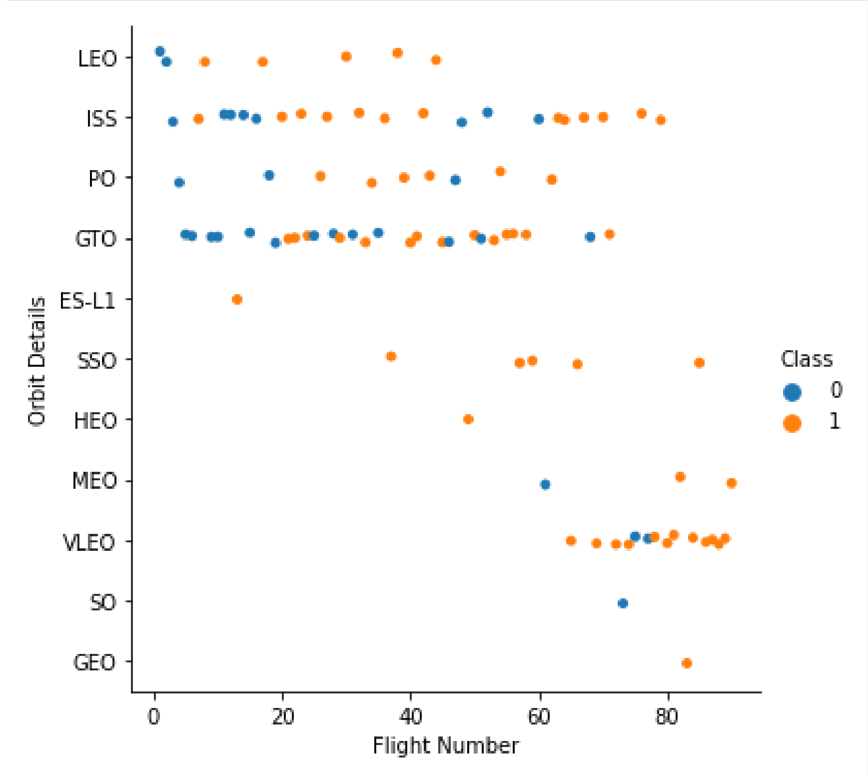
Data Wrangling

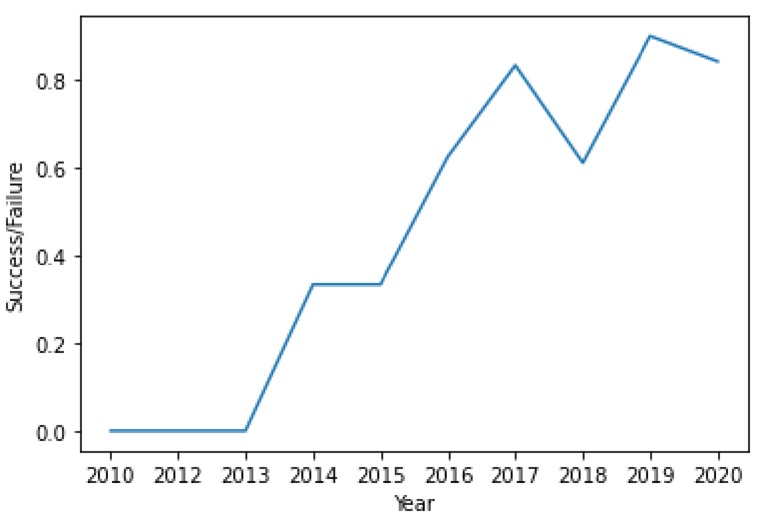
- Performed exploratory data analysis and determined the training labels.
- Calculated the number of launches at each site, and the number and occurrence of each orbits
- Created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is https://github.com/SaumalyaGhosh/IBM-DataScience-SpaceX-Capstone/blob/main/EDA%20lab_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/SaumalyaGhosh/IBM-DataScience-SpaceX-Capstone/blob/main/EDA%20with%20Data%20Visualization.ipynb

EDA with SQL

- Loaded sample SpaceX dataset into a PostgreSQL database.
- Applied EDA with SQL to get insight from the data. Wrote queries to find out for instance: The names of unique launch sites in the space mission.
- The link to the notebook is https://github.com/SaumalyaGhosh/IBM-DataScience-SpaceX-Capstone/blob/main/EDA%20with%20SQL%20lab.ipynb

Task 1

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

In [25]:
```
%sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
```

```
 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[25]:

launch_site
CCAFS LC-40

Task 2

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

In [26]:
```
%%sql
SELECT *
FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 5;
```

```
 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[26]:

booster_version	date_col	time_utc	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing_outcome
F9 v1.0 B0003	2010-04-06	18:45:00	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
F9 v1.0 B0003	2010-04-06	18:45:00	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)

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.
- Using the color-labeled marker clusters, identified which launch sites have relatively high success rate.
- Calculated the distances between a launch site to its proximities. Answered some question for instance:
  - Are launch sites near railways, highways and coastlines.
  - Do launch sites keep certain distance away from cities.
- Git URL: https://github.com/SaumalyaGhosh/IBM-DataScience-SpaceX-Capstone/blob/main/Interactive%20Visual%20Analytics%20with%20Folium%20lab.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 to the notebook is https://github.com/SaumalyaGhosh/IBM-DataScience-SpaceX-Capstone/blob/main/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 to the notebook is https://github.com/SaumalyaGhosh/IBM-DataScience-SpaceX-Capstone/blob/main/Machine%20Learning%20Prediction.ipynb

Predictive Analysis (Classification)

TASK 1

Create a NumPy array from the column `Class` in `data`, by applying the method `to_numpy()` t series (only one bracket df['name of column']).

In [5]:
```
Y = data['Class'].to_numpy()
Y
```

Out[5]:
```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```

TASK 2

Standardize the data in X then reassign it to the variable X using the transform provided below

In [7]:
```
# students get this
X= preprocessing.StandardScaler().fit(X).transform(X)
```

In [8]:
```
X[0:5]
```

Out[8]:
```
array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01,
        -1.57589457e+00, -9.73440458e-01, -1.05999788e-01,
        -1.05999788e-01, -6.54653671e-01, -1.05999788e-01,
        -5.51677284e-01,  3.44342023e+00, -1.85695338e-01,
        -3.33333333e-01, -1.05999788e-01, -2.42535625e-01,
        -4.29197538e-01,  7.97724035e-01, -5.68796459e-01,
        -4.10890702e-01, -4.10890702e-01, -1.50755672e-01,
        -7.97724035e-01, -1.50755672e-01, -3.92232270e-01,
         9.43398113e+00, -1.05999788e-01, -1.05999788e-01,
        -1.05999788e-01, -1.05999788e-01, -1.05999788e-01,
```

TASK 3

Use the function train_test_split to split the data X and Y into training and test data. Set the parameter test_size and test data should be assigned to the following labels.

`X_train, X_test, Y_train, Y_test`

In [9]:
```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
print ('Train set:', X_train.shape,  Y_train.shape)
print ('Test set:', X_test.shape,  Y_test.shape)
```

```
Train set: (72, 83) (72,)
Test set: (18, 83) (18,)
```

we can see we only have 18 test samples.

In [10]:
```
Y_test.shape
```

Out[10]:
```
(18,)
```

TASK 4

Create a logistic regression object then create a GridSearchCV object `logreg_cv` with cv = 10. Fit the object to `parameters`.

In [11]:
```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
lr=LogisticRegression()
grid_search = GridSearchCV(lr, parameters, cv=10)
logreg_cv = grid_search.fit(X_train, Y_train)
```

Results

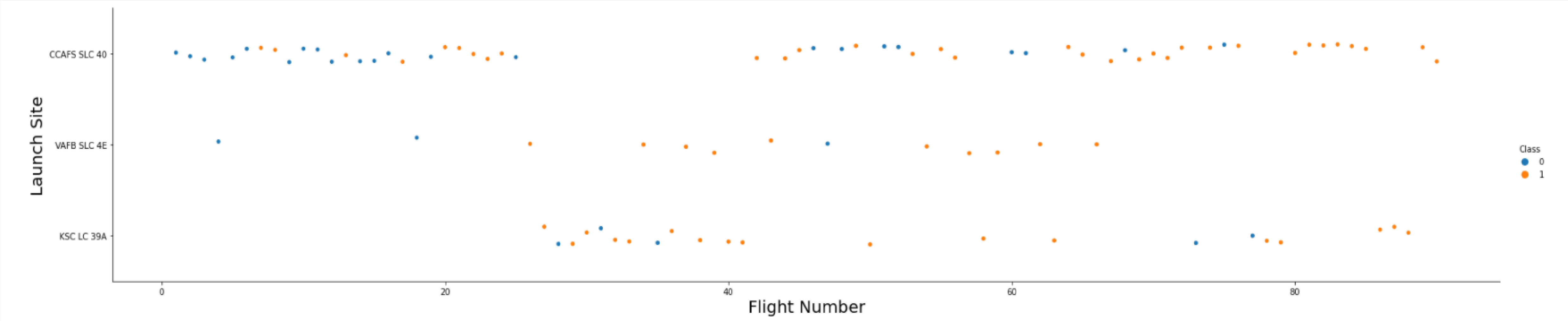
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

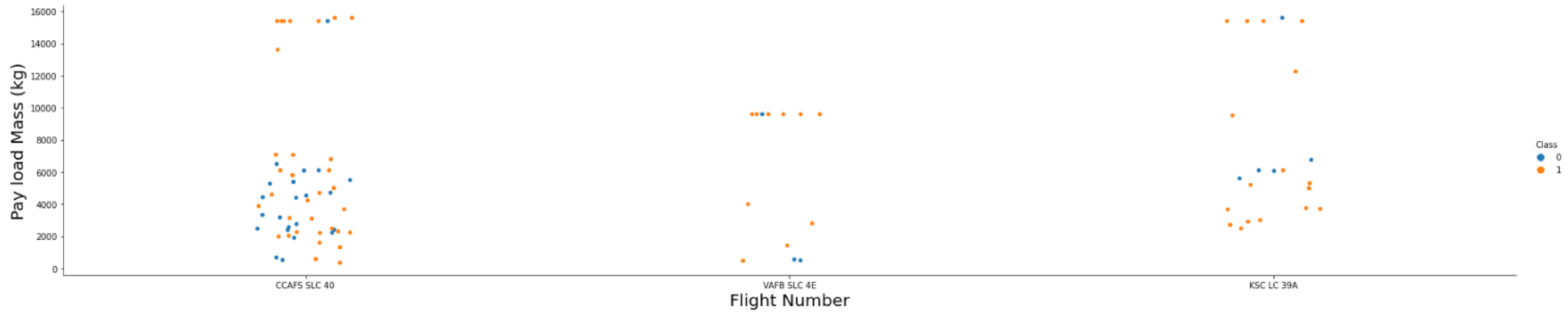
Flight Number vs. Launch Site

- 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

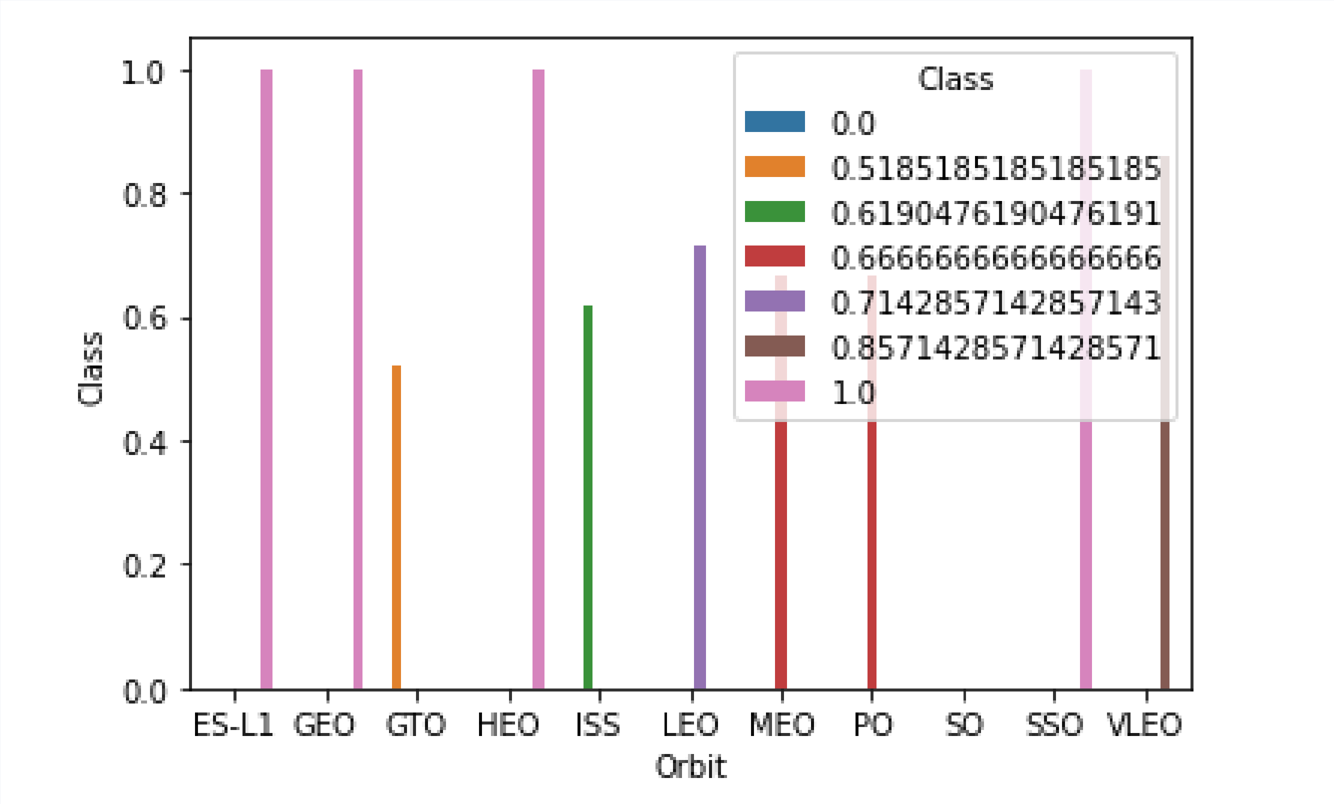
- If we observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000)



Now if you observe Payload Vs. Launch Site scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).

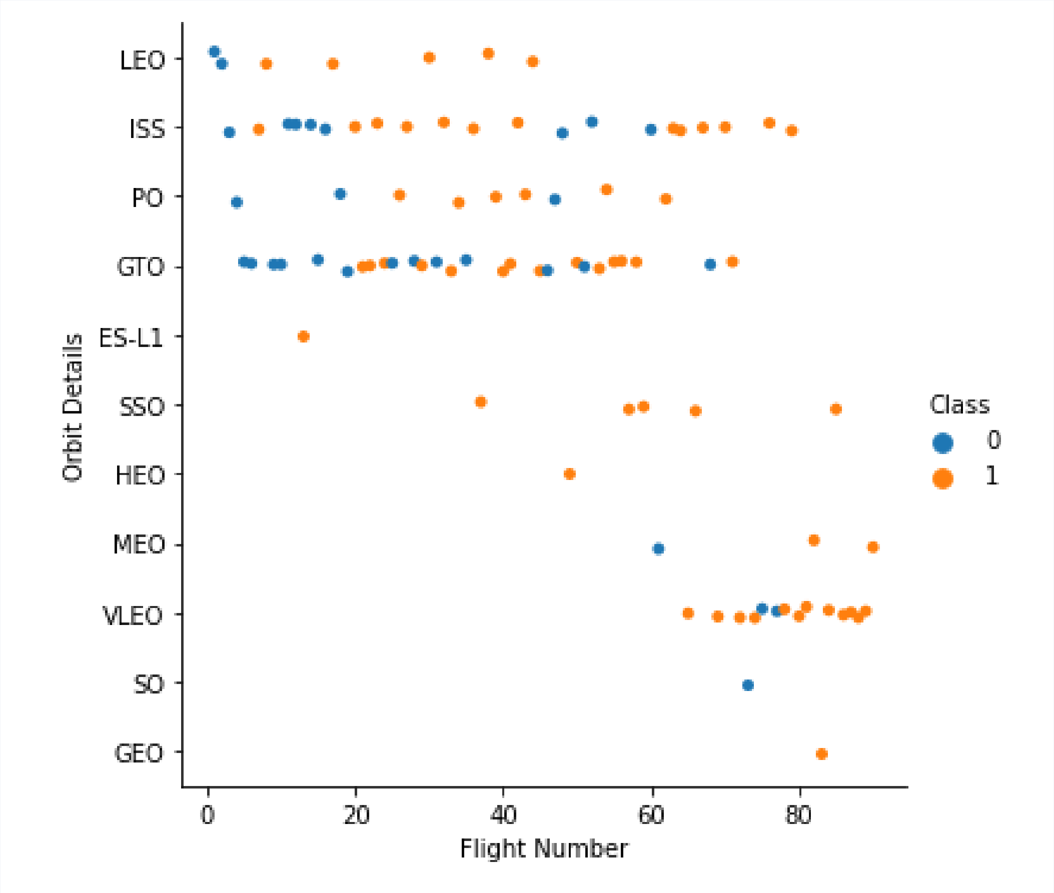
Success Rate vs. Orbit Type

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



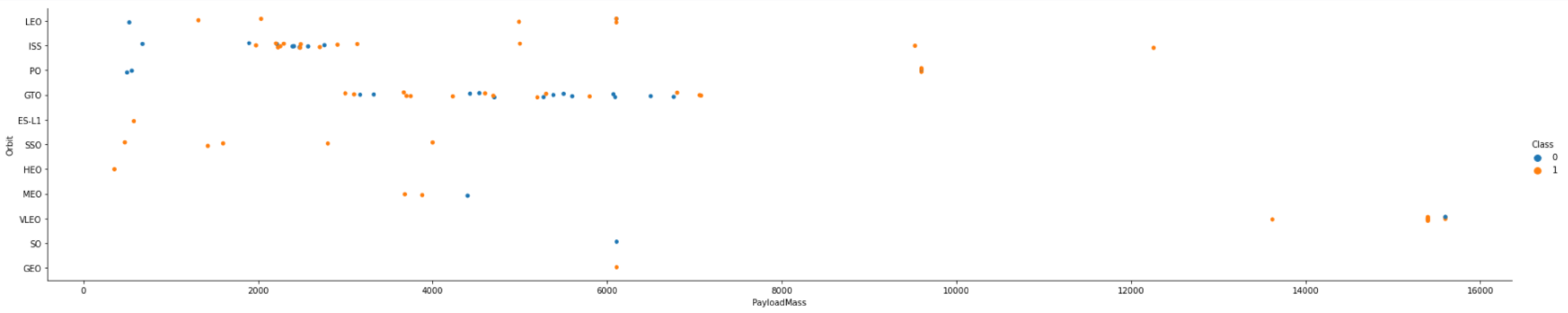
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. 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.



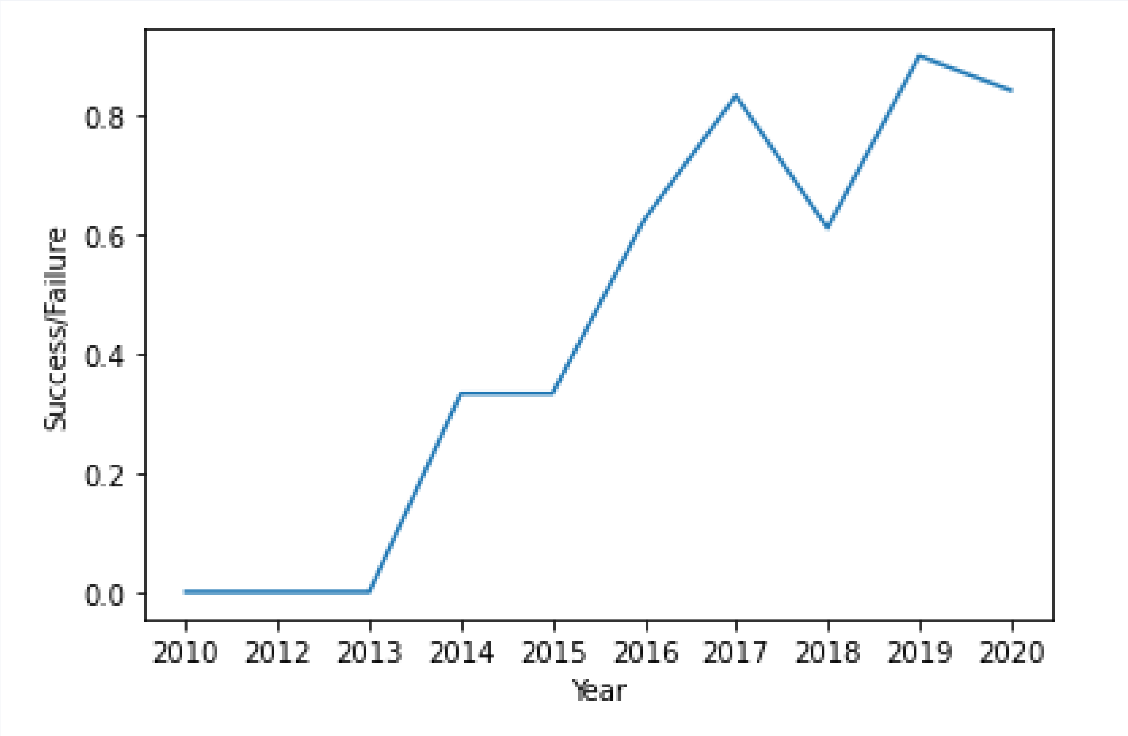
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

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



All Launch Site Names

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

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

In [25]:
```
%sql SELECT DISTINCT(LAUNCH_SITE) FROM SPACEXTBL;
```

```
 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[25]:

launch_site
CCAFS LC-40

Launch Site Names Begin with 'CCA'

- Used the query above to display 2 records where launch sites begin with \`CCA\`

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

In [26]:
```
%%sql
SELECT *
FROM SPACEXTBL
WHERE LAUNCH_SITE LIKE 'CCA%'
LIMIT 2;
```

```
 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Out[26]:

booster_version	date_col	time_utc	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing_outcome
F9 v1.0 B0003	2010-04-06	18:45:00	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
F9 v1.0 B0003	2010-04-06	18:45:00	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)

Total Payload Mass

- Can calculate the total payload carried by boosters from NASA with the below query.

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

```
In [27]: %%sql
SELECT SUM(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE CUSTOMER = 'NASA (CRS)';
```

```
 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

Average Payload Mass by F9 v1.1

- Can calculate the average payload mass carried by booster version F9 v1.1 using the below query.

Task 4

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

In [28]:
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE BOOSTER_VERSION LIKE 'F9 v1.1%';
```

```
 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

First Successful Ground Landing Date

- The date of the first successful landing outcome on ground pad can be fetched using the below query.

```
In [29]: %%sql
SELECT MIN(DATE)
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Success (ground pad)';

 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
```

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

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 [30]:

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

```
 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
```

Total Number of Successful and Failure Mission Outcomes

- Used wildcard like '%' to filter for **WHERE** `LANDING__OUTCOME` was a success or a failure.

```
In [31]: %%sql
SELECT COUNT(LANDING__OUTCOME) AS SUCCESSFUL_MISSIONS
FROM SPACEXTBL
WHERE LANDING__OUTCOME LIKE 'Success%';
```

```
In [32]: %%sql
SELECT COUNT(LANDING__OUTCOME) AS FAILURE_MISSIONS
FROM SPACEXTBL
WHERE LANDING__OUTCOME LIKE 'Failure%';
```

Boosters Carried Maximum Payload

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

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

In [33]:
```
%%sql
SELECT DISTINCT(BOOSTER_VERSION), PAYLOAD_MASS__KG_
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

```
 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
Done.
```

2015 Launch Records

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

Task 9

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

In [34]:
```
%%sql
SELECT LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE, YEAR(DATE) AS DATE_YEAR
FROM SPACEXTBL
WHERE LANDING__OUTCOME = 'Failure (drone ship)' AND YEAR(DATE) = '2015'
```

* ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB

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

- We 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 2017-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

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

In [20]:

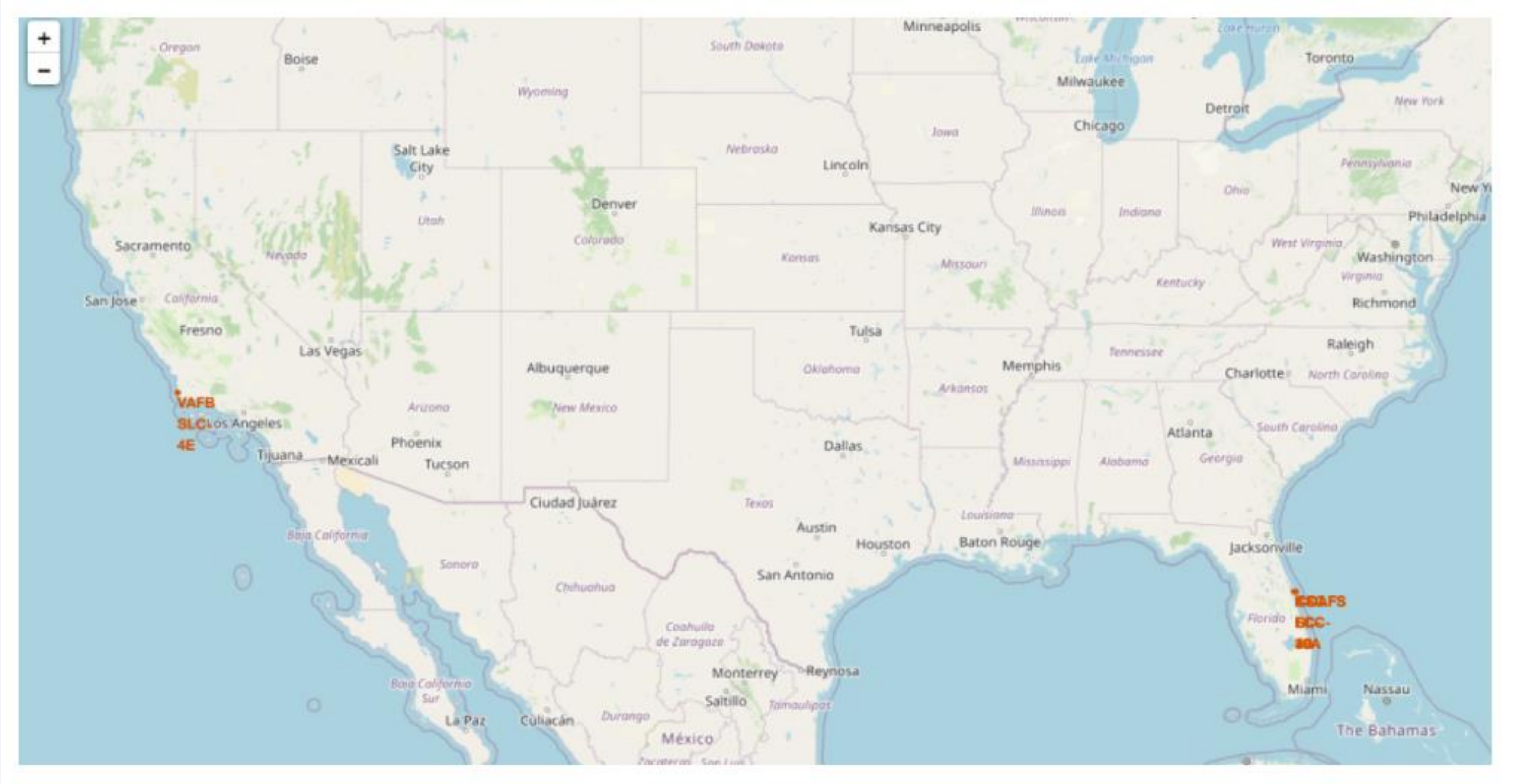
```
%%sql
SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS COUNT
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04'  AND '2017-03-20'
GROUP BY LANDING__OUTCOME
ORDER BY COUNT DESC
```

```
 * ibm_db_sa://lnj43861:***@764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:32536/BLUDB
```

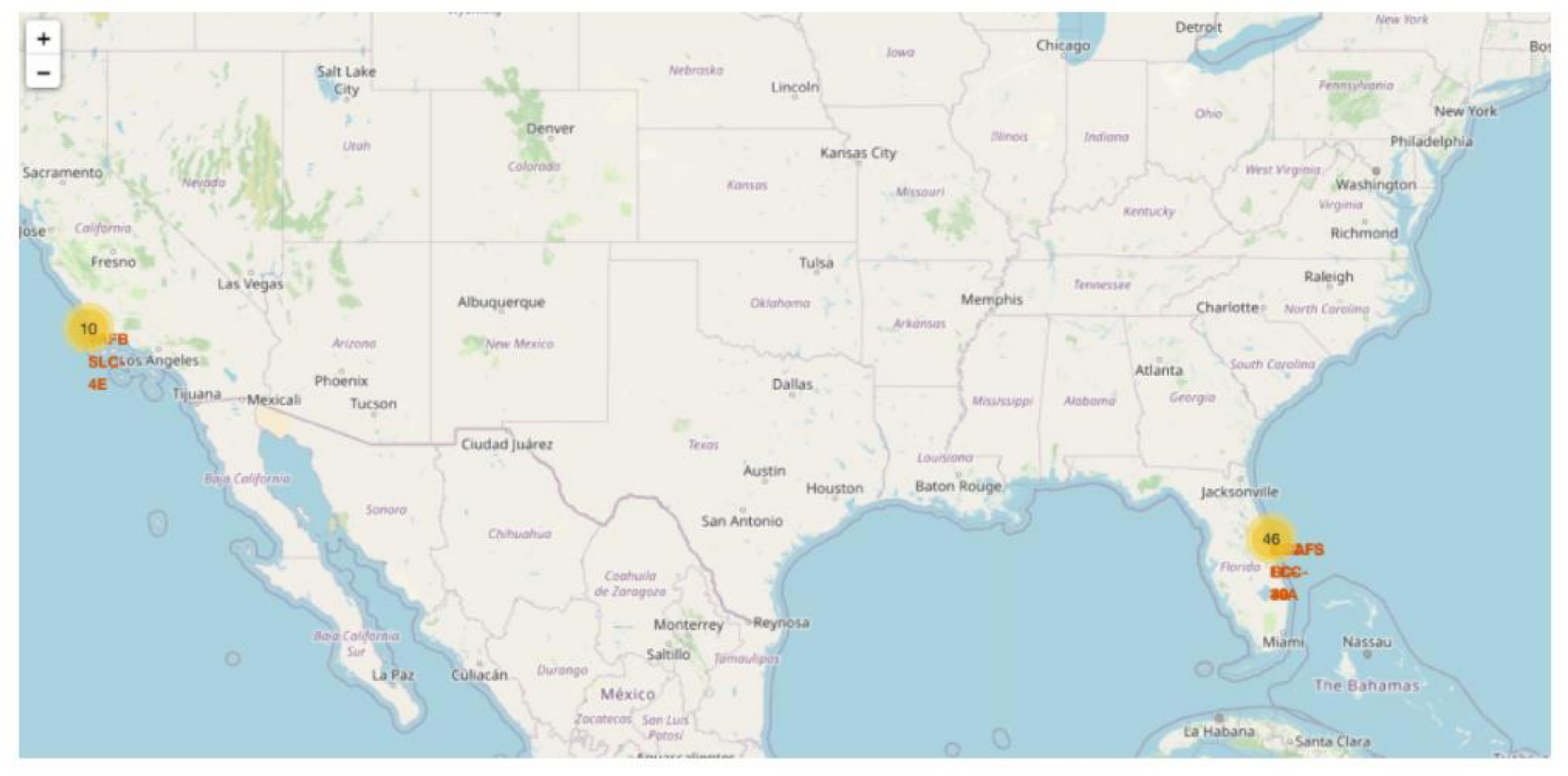
Section 3

Launch Sites Proximities Analysis

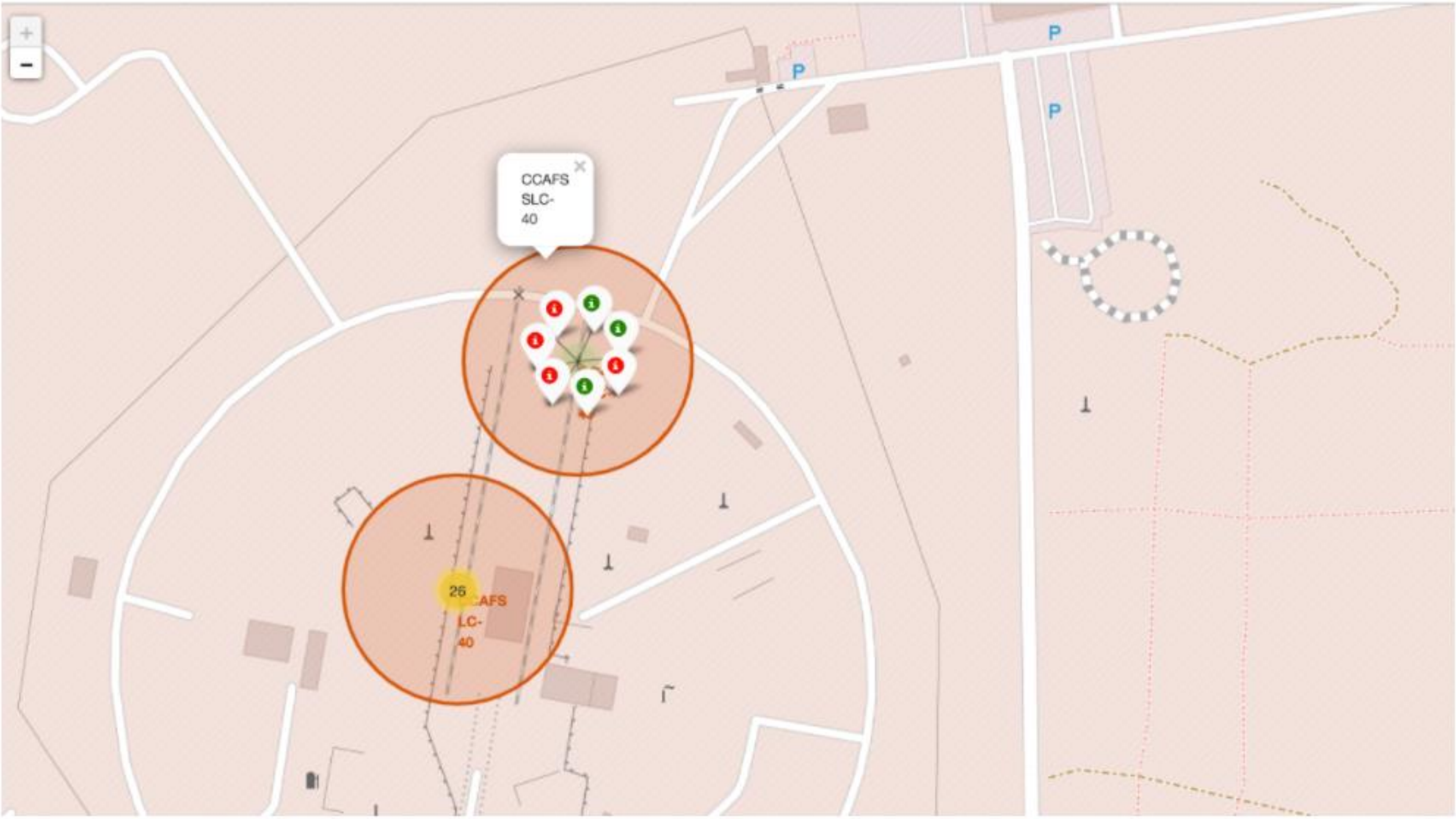
All launch sites on a map



Success/failed launches for each site on the map



Success/failed launches for each site on the map

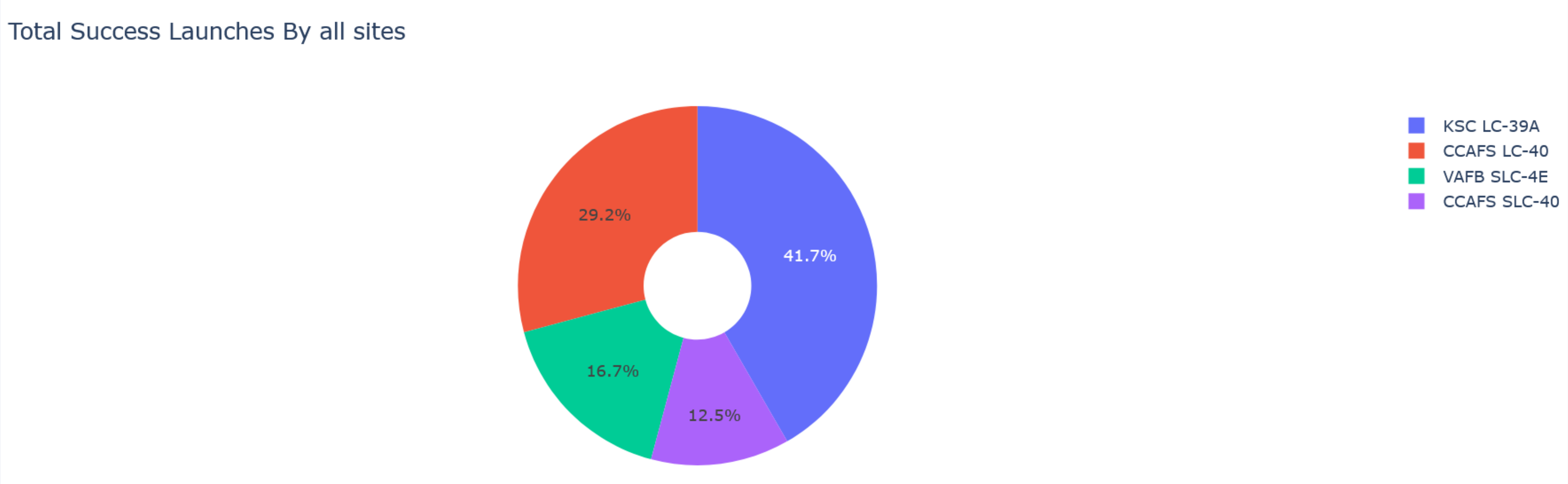


Distances between a launch site to its proximities

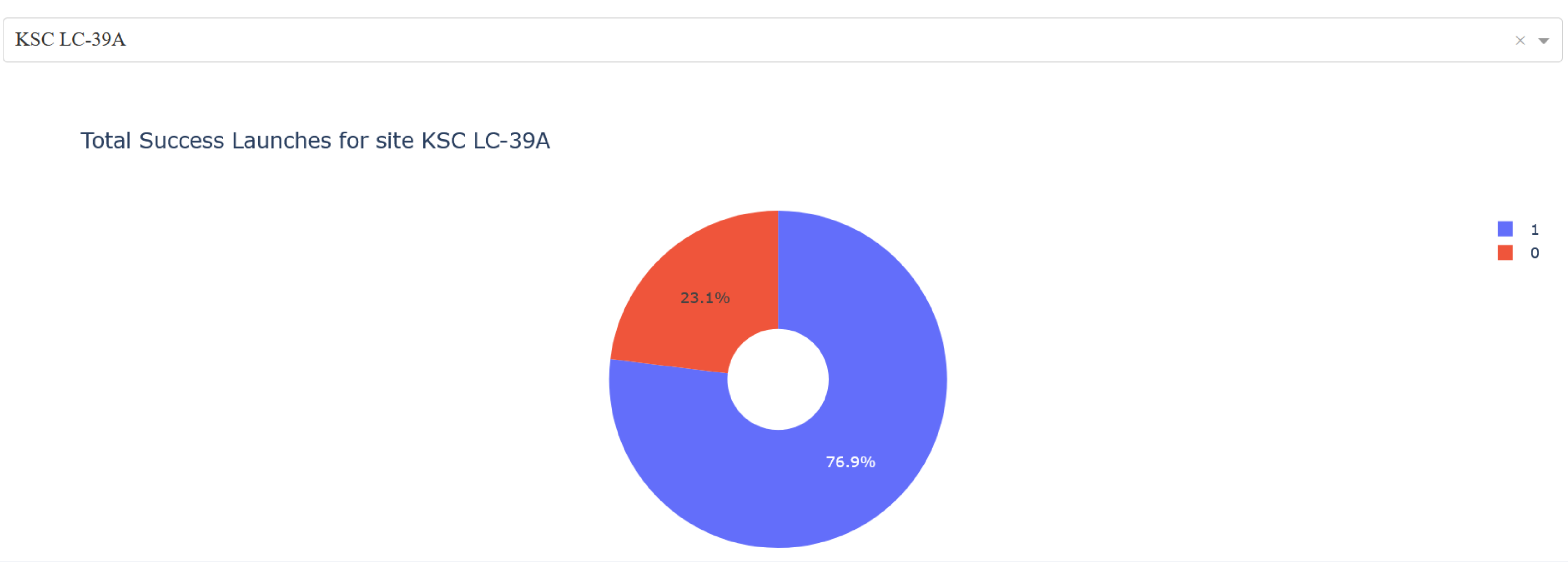
Section 4

Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



Pie chart showing the Launch site with the highest launch success ratio



Scatter plot of Payload vs Launch Outcome for all sites, with payload selected in the range slider

Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

TASK 12

Find the method performs best:

In [30]:
```
print('Accuracy for Logistics Regression method:', logreg_cv.best_score_)
print( 'Accuracy for Support Vector Machine method:', svm_cv.best_score_)
print('Accuracy for Decision tree method:', tree_cv.best_score_)
print('Accuracy for K nearsdt neighbors method:', knn_cv.best_score_)
```

```
Accuracy for Logistics Regression method: 0.8464285714285713
Accuracy for Support Vector Machine method: 0.8482142857142856
Accuracy for Decision tree method: 0.8892857142857142
Accuracy for K nearsdt neighbors method: 0.8482142857142858
```

In [31]:
```
print('Best Params:', tree_cv.best_params_)
```

```
Best Params: {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'best'}
```

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.

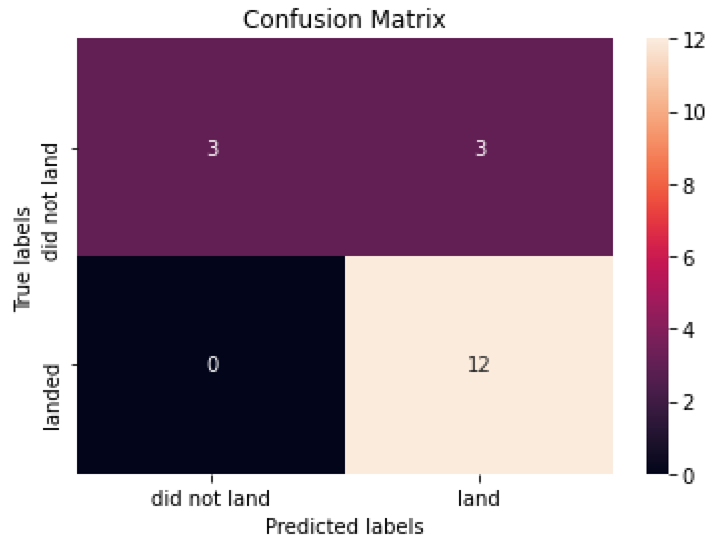
Calculate the accuracy of tree_cv on the test data using the method `score` :

In [27]:
```
tree_cv.score(X_test, Y_test)
```

Out[27]: 0.8333333333333334

We can plot the confusion matrix

In [28]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



Conclusions

It can be concluded that:

- 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

- Git Repo Link: https://github.com/SaumalyaGhosh/IBM-DataScience-SpaceX-Capstone

Thank you!