IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Linyu Liu
09/22/2024

Outline

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

Summary of methodologies

- Collect data using SpaceX REST API and web scraping techniques
- Wrangle data to create success/fail outcome variable
- Explore data with data visualization techniques, considering the following factors: payload, launch site, flight number and yearly trend
- Analyze the data with SQL, calculating the following statistics: total payload, payload range for successful launches, and total # of successful and failed outcomes
- Explore launch site success rates and proximity to geographical markers
- Visualize the launch sites with the most success and successful payload ranges
- Build Models to predict landing outcomes using logistic regression, support vector machine (SVM), decision tree and K - nearest neighbor (KNN)

Summary of all results

- KSC LC-39A has the highest success rate among landing sites
- Orbits ES-L1, GEO, HEO, and SSO have a 100% success rate

Background

SpaceX, a pioneer in the space industry, aims to make space travel accessible and affordable for everyone. Its achievements include sending spacecraft to the International Space Station, deploying a satellite constellation to provide global internet coverage, and conducting manned space missions. The company's ability to achieve this is largely due to the relatively low cost of its rocket launches—approximately $62 million per launch—made possible by the innovative reuse of the first stage of its Falcon 9 rocket. In contrast, other providers, who do not reuse the first stage, incur launch costs exceeding $165 million each. By predicting whether the first stage can be successfully recovered, we can estimate the cost efficiency of a launch. This can be achieved by using public data and machine learning models to forecast whether SpaceX or a competitor will be able to reuse the first stage.

Goal of Study

The impact of payload mass, launch site, number of flights, and orbits on the success of first-stage landings.

The success rate of first-stage landings over time.

The best predictive model for determining successful landings (binary classification).

Section 1

Methodology

Methodology

Data collection methodology:

- SpaceX REST API and web scraping technique
- Perform data wrangling

- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models

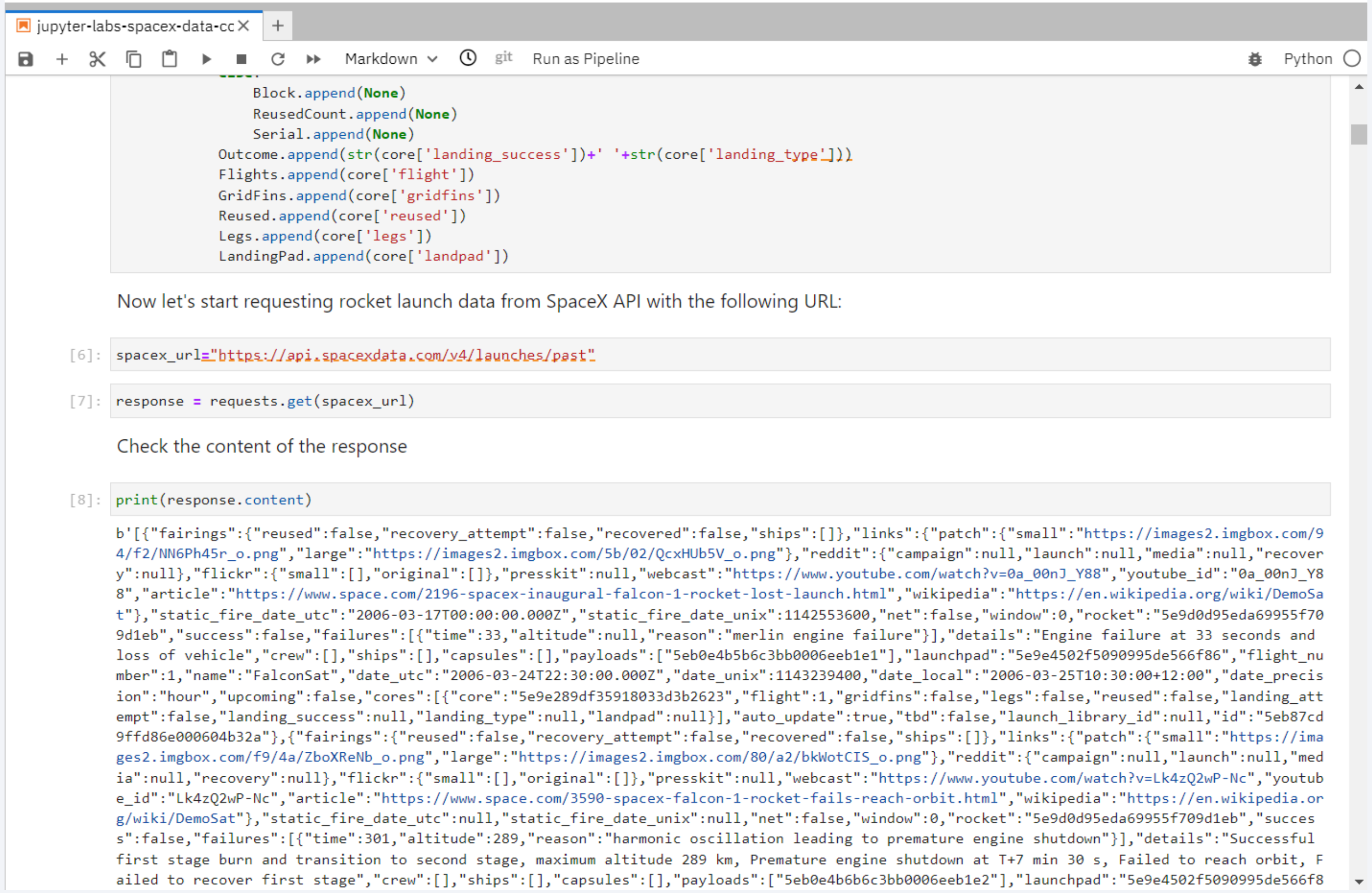
Data Collection

Steps

- Request data from SpaceX API (rocket launch data)
- Decode response using .json() and convert to a dataframe using .json_normalize()
- Request information about the launches from SpaceX API using custom functions
- Create dictionary from the data
- Create dataframe from the dictionary
- Filter dataframe to contain only Falcon 9 launches
- Replace missing values of Payload Mass with calculated .mean()
- Export data to csv file

Data Collection – SpaceX API

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



Data Collection - Scraping

- https://github.com/CharAznableUC/Falcon/blob/main/Web_Scraping.ipynb

- Request the Falcon9 Launch Wiki page from its URL
- Extract all column/variable names from the HTML table header
- Create a data frame by parsing the launch HTML tables

Data Wrangling

- https://github.com/CharAznableUC/Falcon/blob/main/Wrangling.ipynb

- Perform EDA Analysis
- Create Binary
- Export csv

- Calculate number of launches for each site (GTO, HEO, LEO,MEO)
- Calculate landing outcome

EDA with Data Visualization

- https://github.com/CharAznableUC/Falcon/blob/main/EDA.ipynb

Used Plots:

Flight number vs payload

Payload mass vs launch site

Payload mass vs orbit type

- Why

We see that as the flight number increases, the first stage is more likely to land successfully. The payload mass is also important; it seems the more massive the payload, the less likely the first stage will return.

EDA with SQL

- https://github.com/CharAznableUC/Falcon/blob/main/SQL.ipynb

SQL performed:

Flight number vs payload

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- List the date when the first successful landing outcome in ground pad was achieved.

Build an Interactive Map with Folium

- https://github.com/CharAznableUC/Falcon/blob/main/Folium.ipynb

Why use Folium:

- Mark all launch sites on a map
- Color coded the launch outcomes
- Added color lines to show distance between launch site and its proximity to the city, railway and coastline.

Build a Dashboard with Plotly Dash

- https://github.com/CharAznableUC/Falcon/blob/main/Ploty%20Dash.py

Why use Plotly Dash:

- Create a dash application with simple UI to show launch sites, successful launches, payload range and correlation between payload and success for all Sites

Predictive Analysis (Classification)

1. Create NumPy array from the Class column
2. Standardize the data with StandardScaler. Fit and transform the data.
3. Split the data using train_test_split
4. Create a GridSearchCV object with cv=10 for parameter optimization
5. Apply GridSearchCV on different algorithms: logistic regression (LogisticRegression()), support vector machine (SVC()), decision tree (DecisionTreeClassifier()), K-Nearest Neighbor (KNeighborsClassifier())
6. Calculate accuracy on the test data using .score() for all models
7. Assess the confusion matrix for all models
8. Identify the best model using F1_Score and Accuracy

Results

1. Exploratory Data Analysis

2. Launch success has improved over time
3. KSC LC-39A has the highest success rate among landing sites
4. Orbits ES-L1, GEO, HEO and SSO have a 100% success rate

5. Visual Analytics
6. Most launch sites are near the equator, and all are close to the coast
7. Launch sites are far enough away from anything a failed launch can damage
8. Predictive Analytics
9. Decision Tree model is the best predictive model for the dataset

Section 2

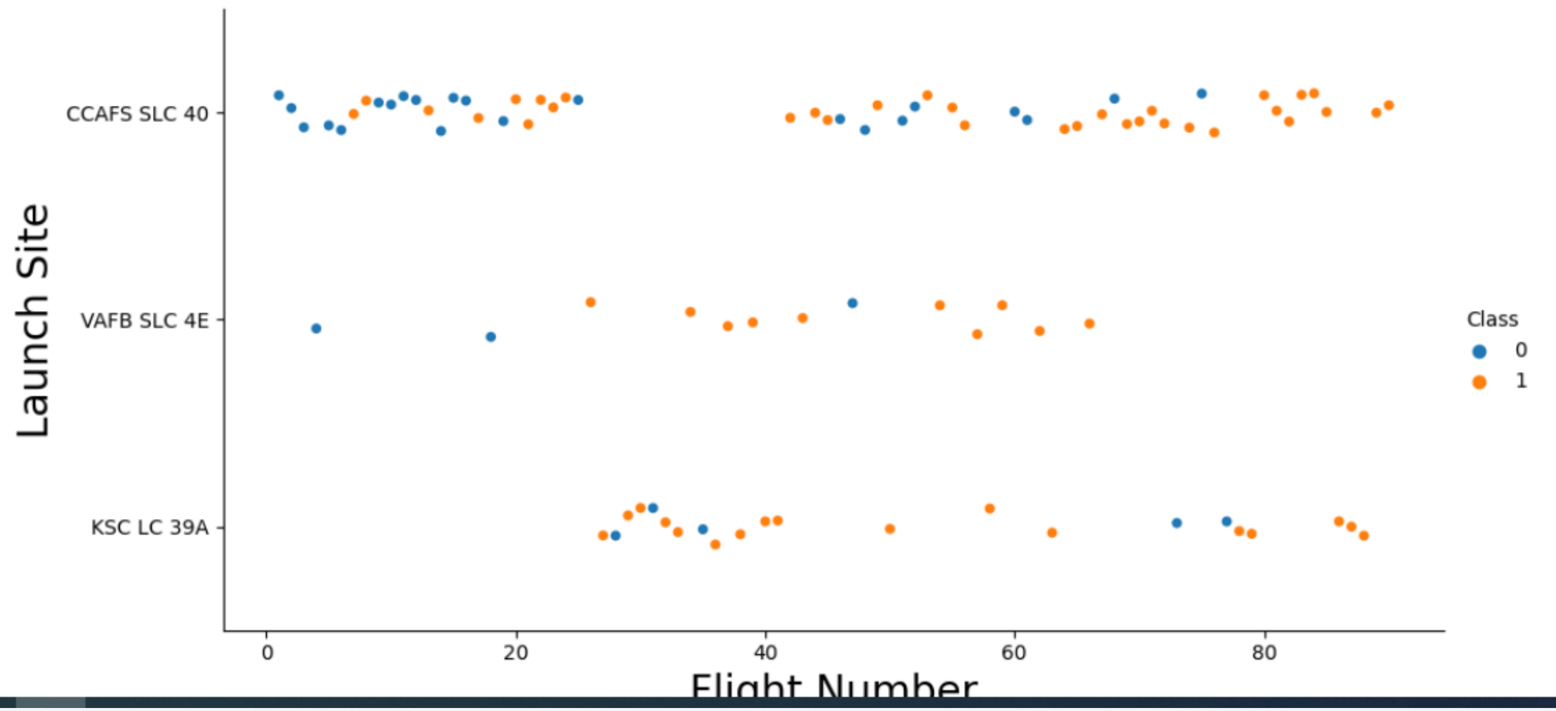
Insights drawn from EDA

Flight Number vs. Launch Site

Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

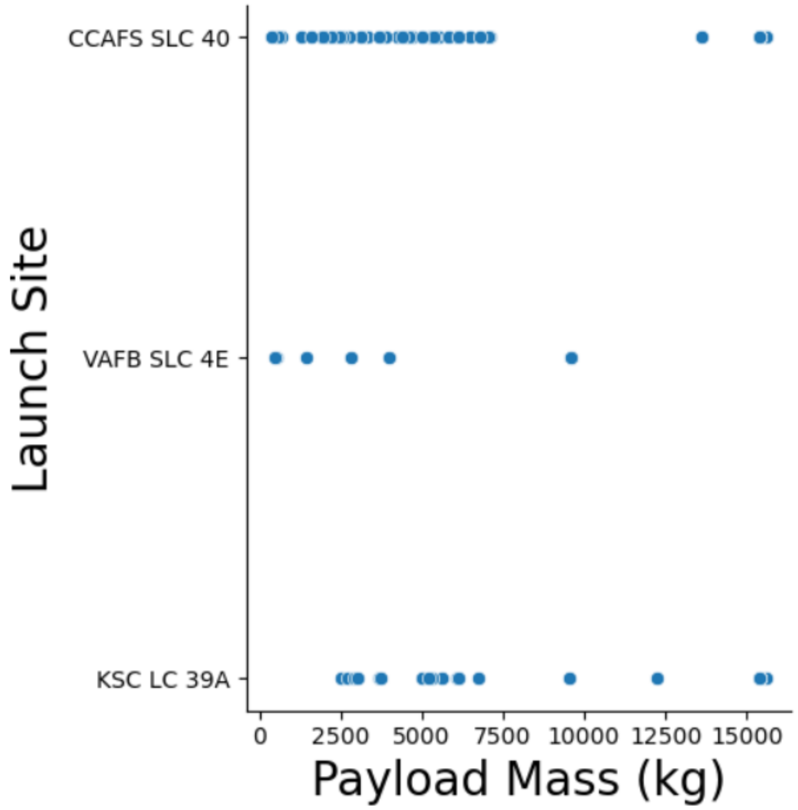
In [31]:

```
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class va
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df,aspect=2)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Payload vs. Launch Site

```
In [12]: ### TASK 2: Visualize the relationship between Payload and Launch Site
         sns.relplot(y="LaunchSite", x="PayloadMass", data=df)
         plt.xlabel("Payload Mass (kg)",fontsize=20)
         plt.ylabel("Launch Site",fontsize=20)
         plt.show()
```



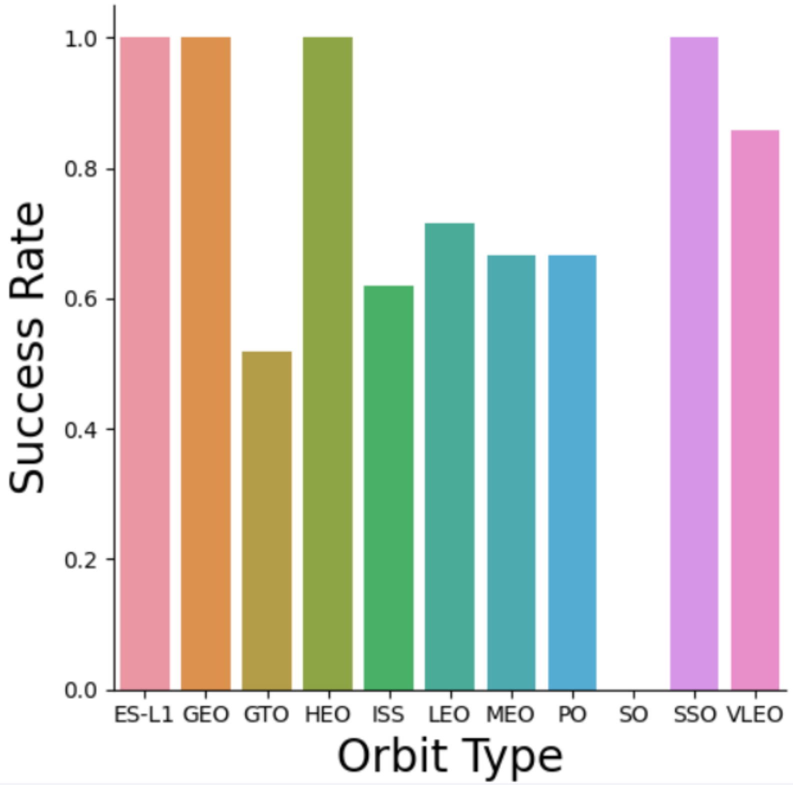
Success Rate vs. Orbit Type

Next, we want to visually check if there are any relationship between success rate and orbit type.

Let's create a `bar chart` for the sucess rate of each orbit

In [33]:

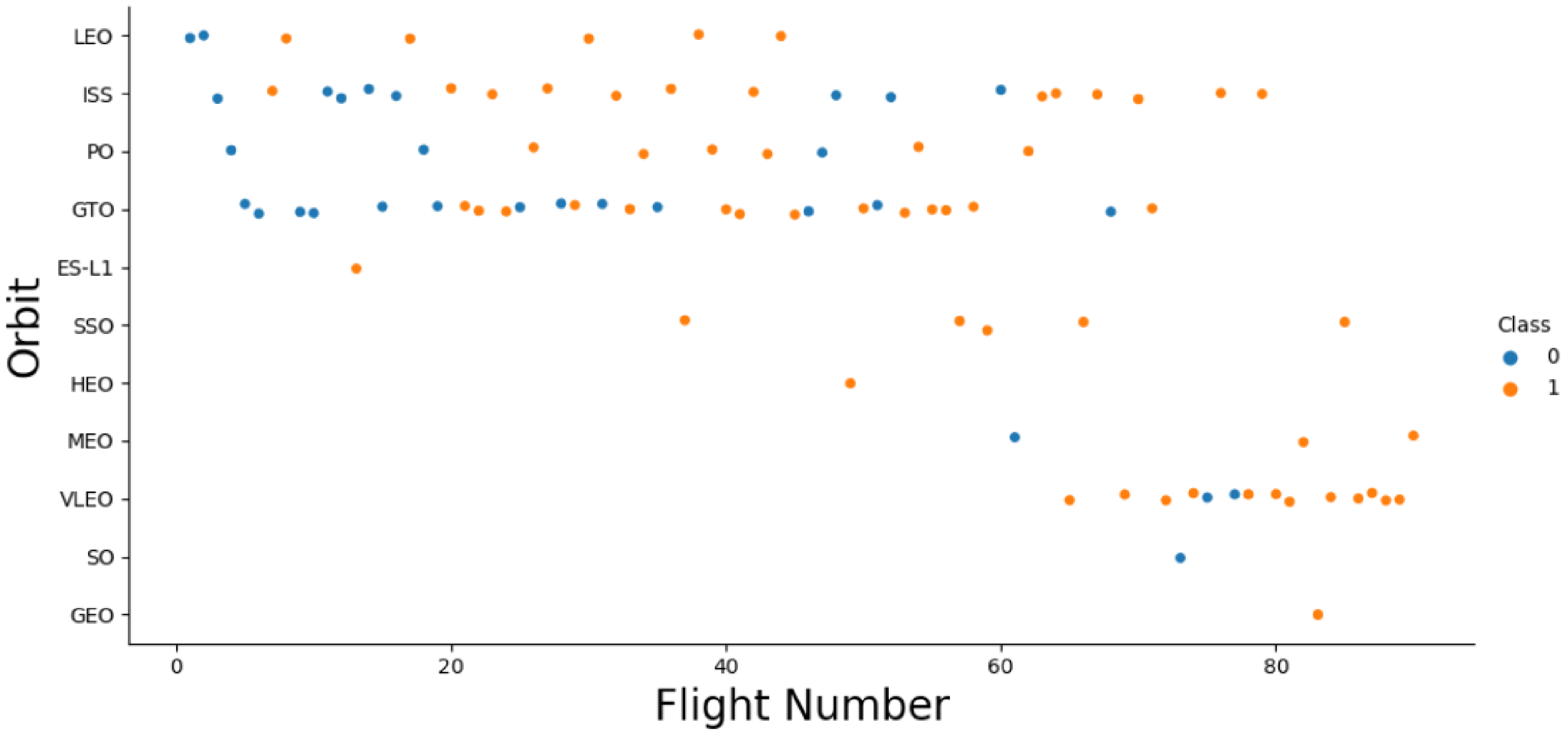
```
# HINT use groupby method on Orbit column and get the mean of Class column
sns.catplot(x= 'Orbit', y = 'Class', data = df.groupby('Orbit')['Class'].mean().reset_index(), kind = 'bar')
plt.xlabel('Orbit Type',fontsize=20)
plt.ylabel('Success Rate',fontsize=20)
plt.show()
```



Flight Number vs. Orbit Type

In [34]:

```
# Plot a scatter point chart with x axis to be FlightNumber and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df,aspect=2)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



You should 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

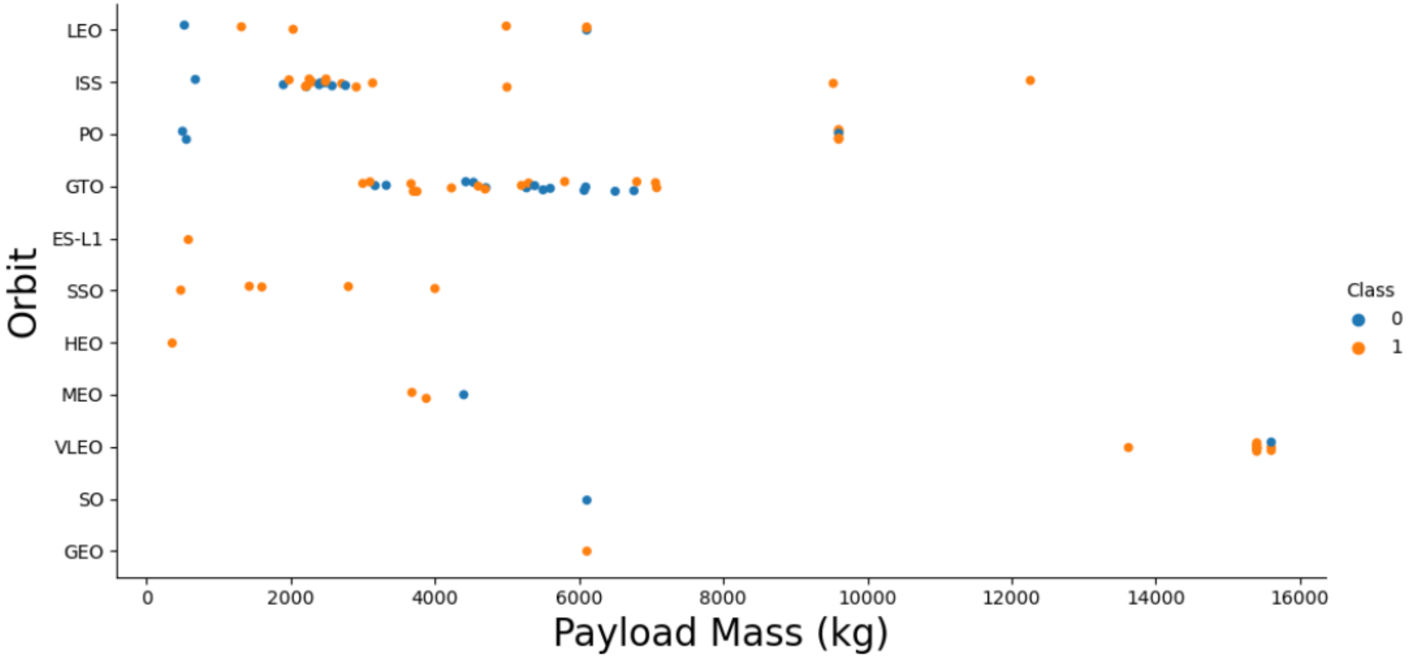
In [18]:

```
### TASK  5: Visualize the relationship between Payload and Orbit type
```

Similarly, we can plot the Payload vs. Orbit scatter point charts to reveal the relationship between Payload and Orbit type

In [35]:

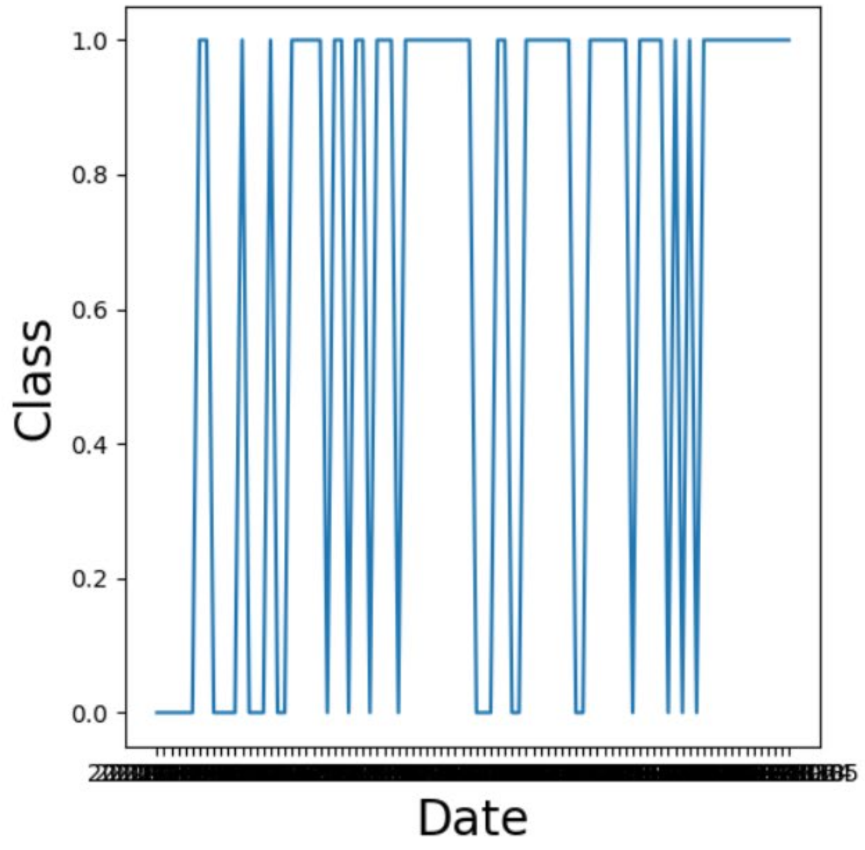
```
# Plot a scatter point chart with x axis to be Payload and y axis to be the Orbit, and hue to be the class value
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df,aspect=2)
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



Launch Success Yearly Trend

In [20]:

```
### TASK  6: Visualize the launch success yearly trend
sns.lineplot(y="Class", x="Date", data=df)
plt.xlabel("Date",fontsize=20)
plt.ylabel("Class",fontsize=20)
plt.show()
```



All Launch Site Names

In [24]:
```python
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingP
features.head()
```

Out[24]:

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0007
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

In [25]:
```python
### TASK  7: Create dummy variables to categorical columns
```

Launch Site Names Begin with 'CCA'

Task 2

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

In [9]:
```
%sql SELECT * \
    FROM SPACEXTBL \
    WHERE LAUNCH_SITE LIKE'CCA%' LIMIT 5;
```

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

Out[9]:

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

Task 3

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

In [10]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE CUSTOMER = 'NASA (CRS)';
```

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

Out[10]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

Task 4

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

In [11]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

Out[11]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

Task 5

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

Hint:Use min function

In [12]:

```
%sql SELECT MIN(DATE) \
FROM SPACEXTBL \
WHERE LANDING__OUTCOME = 'Success (ground pad)'
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: SELECT MIN(DATE)  FROM SPACEXTBL  WHERE LANDING__OUTCOME = 'Success (ground pad)']
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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

In [13]:

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

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: LANDING__OUTCOME
[SQL: SELECT PAYLOAD  FROM SPACEXTBL  WHERE LANDING__OUTCOME = 'Success (drone ship)'  AND PAYLOAD_MASS__KG_ BETWEEN 4000 A
ND 6000;]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

Total Number of Successful and Failure Mission Outcomes

Task 7

List the total number of successful and failure mission outcomes

In [14]:

```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

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

Out[14]:

Mission_Outcome	total_number
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

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

In [15]:

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

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

Out[15]:

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

2015 Launch Records

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [38]:

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

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

Out[38]:

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

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

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

In [37]:

```
%sql SELECT [Landing _Outcome], count(*) as count_outcomes \
FROM SPACEXTBL \
WHERE DATE between '04-06-2010' and '20-03-2017' group by [Landing _Outcome] order by count_outcomes DESC;
```

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

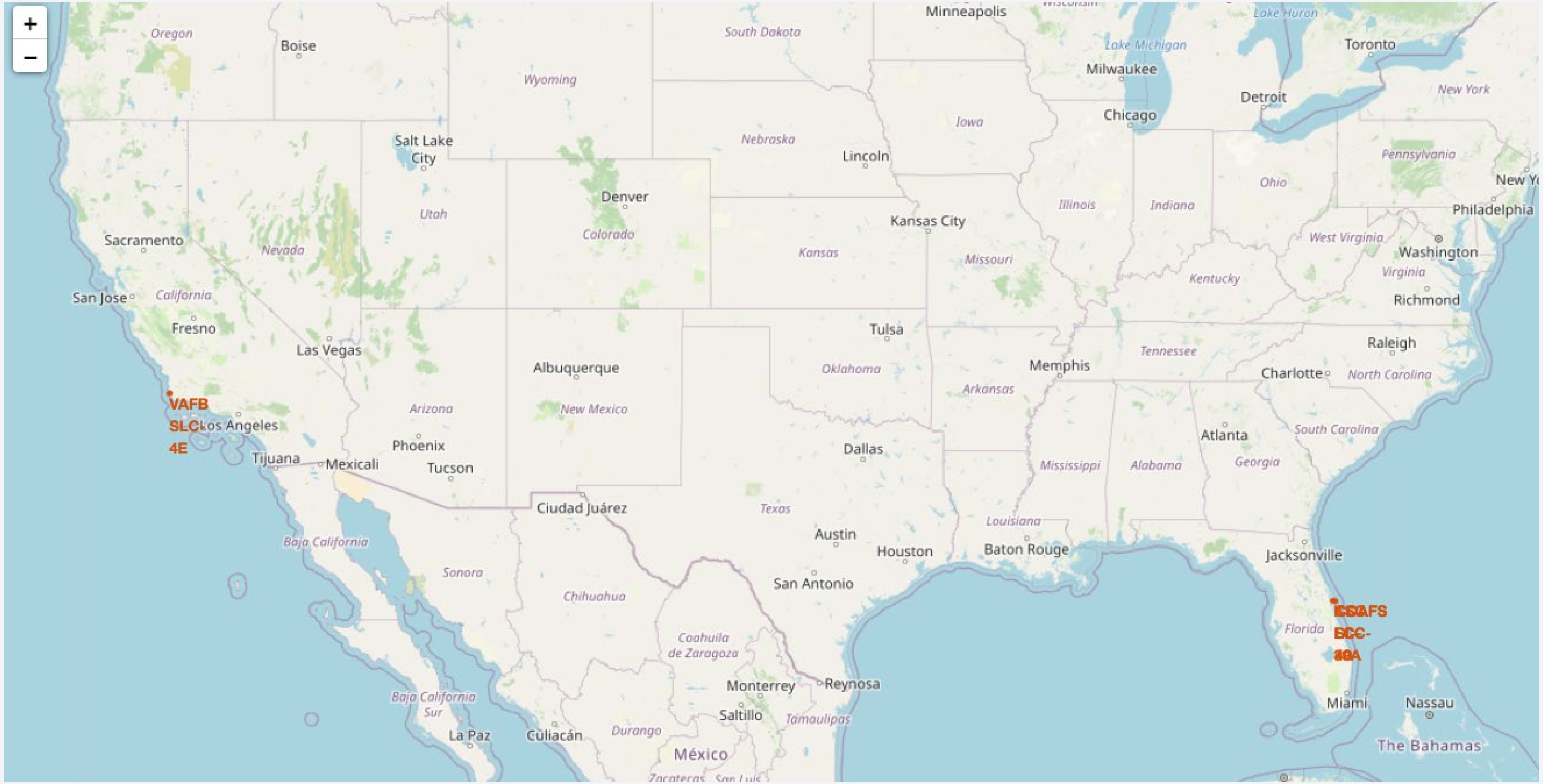
Out[37]:

Landing _Outcome	count_outcomes
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
Failure (parachute)	2
No attempt	1

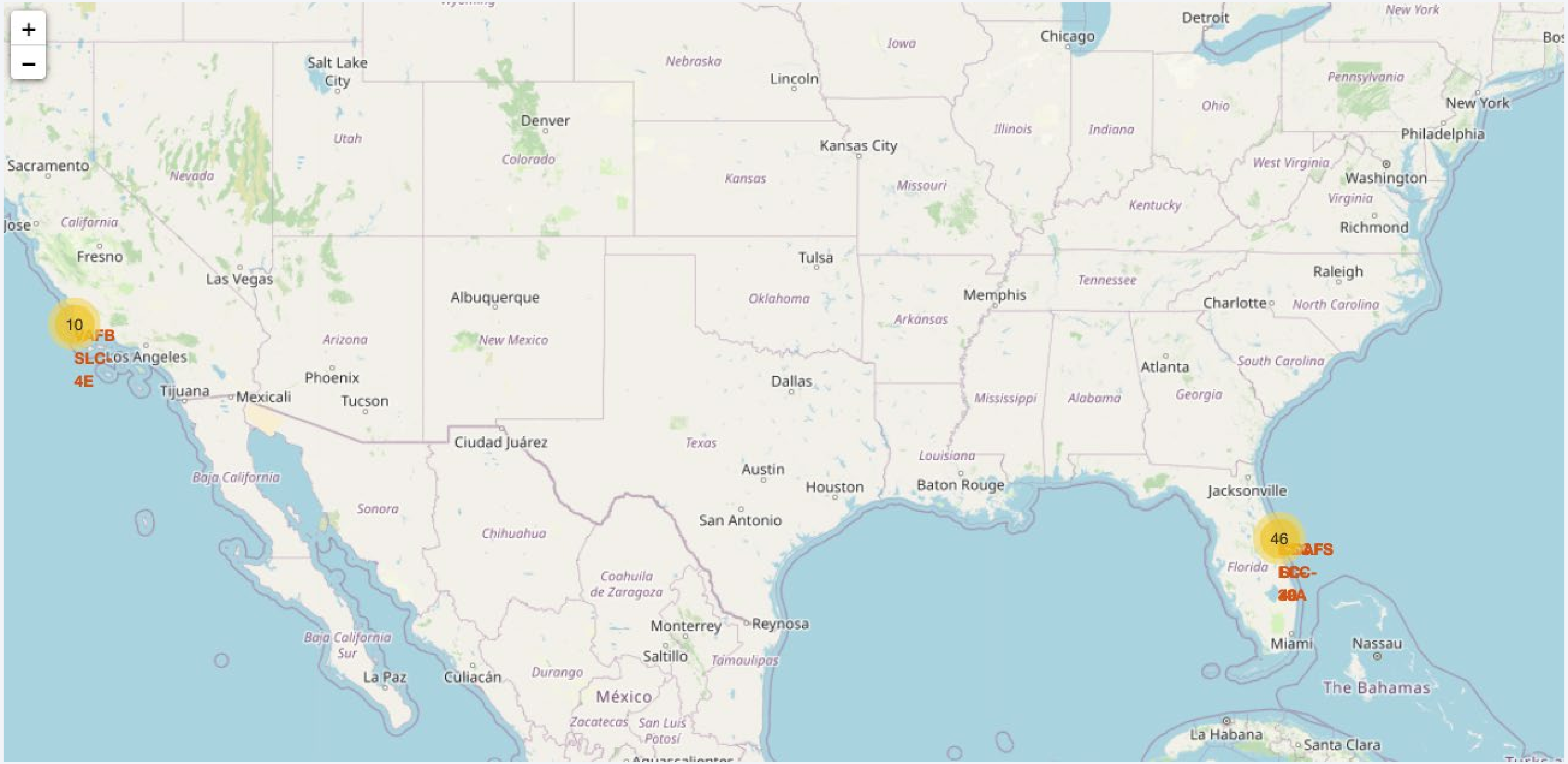
Section 3

Launch Sites Proximities Analysis

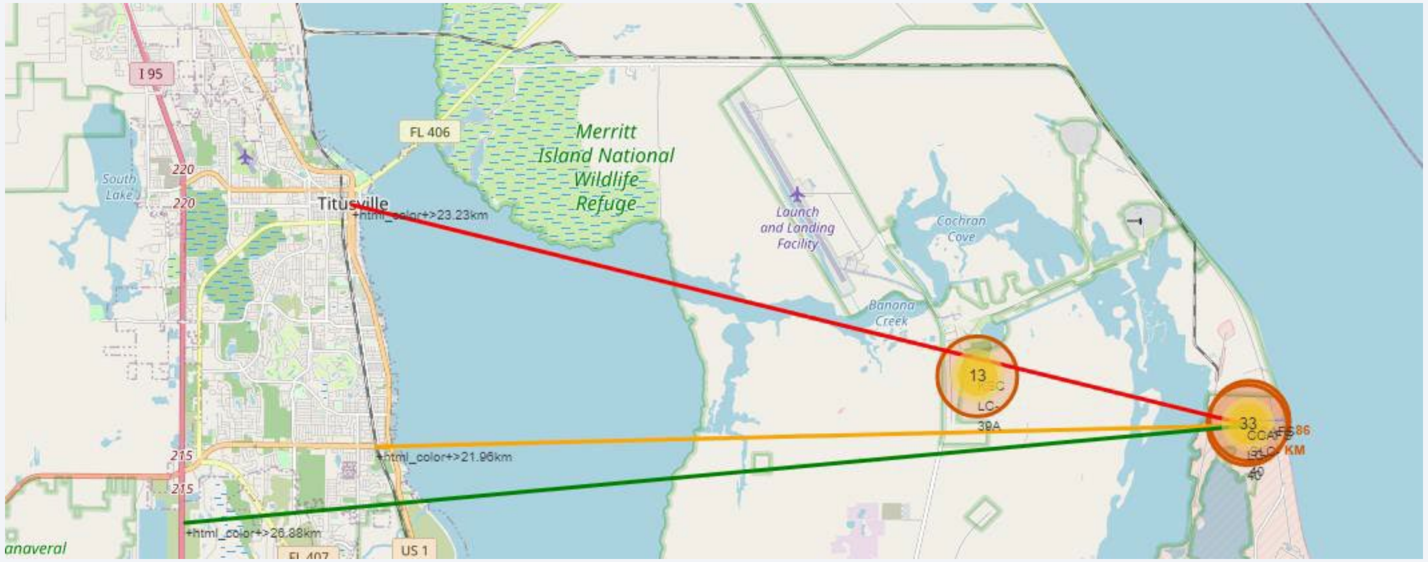
All Launch Site Locations



Launch Outcome



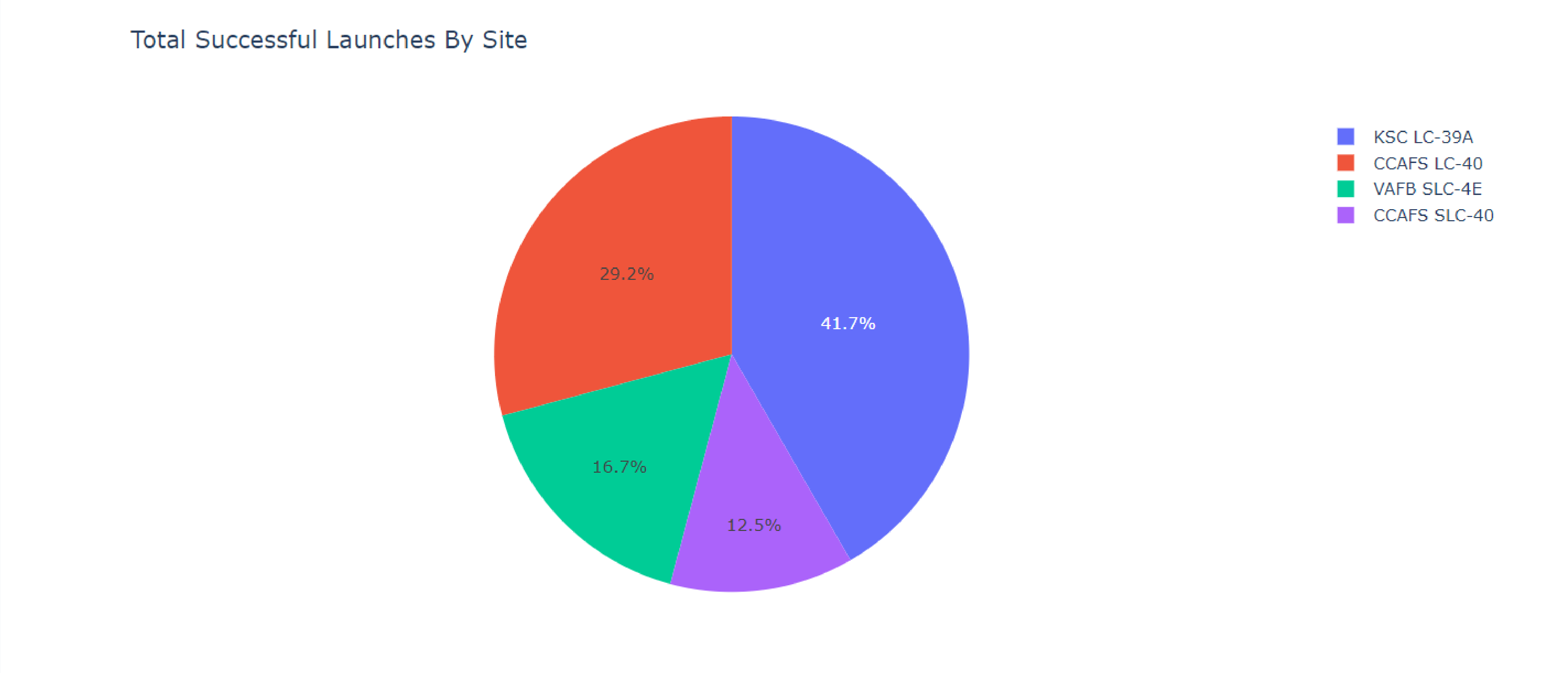
<Folium Map Screenshot 3>



Section 4

Build a Dashboard with Plotly Dash

Count of Successful Launch for All Sites



Correlation between payload and success for all site

<Dashboard Screenshot 3>

- Replace <Dashboard screenshot 3> title with an appropriate title

- Show screenshots of Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

- Explain the important elements and findings on the screenshot, such as which payload range or booster version have the largest success rate, etc.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

In [31]:

```
from sklearn.metrics import jaccard_score, f1_score

# Examining the scores from Test sets
jaccard_scores = [
                  jaccard_score(Y_test, logreg_yhat, average='binary'),
                  jaccard_score(Y_test, svm_yhat, average='binary'),
                  jaccard_score(Y_test, tree_yhat, average='binary'),
                  jaccard_score(Y_test, knn_yhat, average='binary'),
                 ]

f1_scores = [
             f1_score(Y_test, logreg_yhat, average='binary'),
             f1_score(Y_test, svm_yhat, average='binary'),
             f1_score(Y_test, tree_yhat, average='binary'),
             f1_score(Y_test, knn_yhat, average='binary'),
            ]

accuracy = [logreg_score, svm_cv_score, tree_cv_score, knn_cv_score]

scores_test = pd.DataFrame(np.array([jaccard_scores, f1_scores, accuracy]), index=['Jaccard_Score', 'F1_Score', 'Accuracy
scores_test
```

Out[31]:

	LogReg	SVM	Tree	KNN
Jaccard_Score	0.800000	0.800000	0.800000	0.800000
F1_Score	0.888889	0.888889	0.888889	0.888889
Accuracy	0.833333	0.833333	0.833333	0.833333

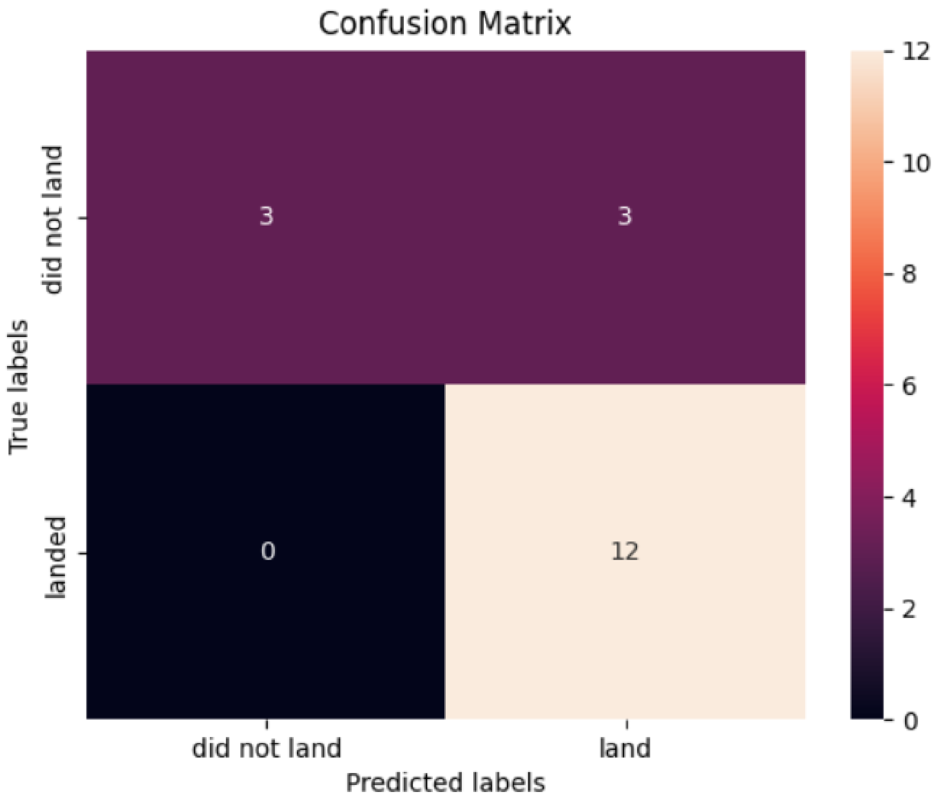
All the models performed at about the same level and had the same scores and accuracy. This is likely due to the small dataset.

The Decision Tree model slightly outperformed the rest when looking at .best_score_

.best_score_ is the average of all cv folds for a single combination of parameters

Confusion Matrix

In [16]:
```
logreg_yhat=logreg_cv.predict(X_test)
plot_confusion_matrix(Y_test,logreg_yhat)
```



Examining the confusion matrix, we see that logistic regression can distinguish between the different classes. We see that the major problem is false positives.

Conclusions

Go Elon Musk!

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!