IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Juan Pellejero Solans
23 / 09 / 2022

Outline

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

Executive Summary

- Summary of methodologies
  - ✓ Data was collected through API and Web scraping
  - ✓ Data was then pre-processed accordingly (Data Wrangling)
  - ✓ Exploratory data analysis was implemented through SQL and Visualization methods
  - ✓ A visual analysis was implemented using Folium
  - ✓ A final prediction was obtained used Machine Learning Prediction
- Summary of all results
  - ✓ Results from Exploratory Data Analysis
  - ✓ Interactive graphic results
  - ✓ A result from the predictive analysis

Introduction

- Project background and context

  SpaceX is currently using Falcon 9 rocket launchers, advertised on its website at a cost of 62M$. (Being M$ million dollars). The sector competitor providers, use rocket launchers with cost upward of 165M$ each. The main reason of the savings are the reusability of the first stage that SpaceX offers. Being so, determining if the first stage returns safely is a main concern for expenses concerns. The goal of this project is determining whether a first stage lands safely for subsequent use.

- Problems you want to find answers

  - ❑ ¿What factors determine if the first stage lands safely?
  - ❑ ¿How do these factors interact with each other to determine the success rate?
  - ❑ ¿What other conditions must be met to guarantee a successful landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
    - Data was collected using SpaceX API and web scrapping from Wikipedia tables using python.
- Perform data wrangling
    - Additional columns were added, and categorical features were replaced with dummy variables using one-hot encoding
- 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

- Data was obtained using `get request` to the SpaceX API
- Data was then “translated” from its home format to JSON using `.json()`, and then exported to a pandas dataframe with `.json_normalize()`
- Data was pre-processed, eliminating NaN values and replacing missing values with mean values
- Web scraping was performed to get Falcon 9 launch records out of Wikipedia record tables. This was done using BeautifulSoup
- Every table obtained was then imported to its own pandas dataframe, for the purpose of ease of use.

Data Collection – SpaceX API

- Using the get request to the SpaceX API lets us collect data.

- [Link to the notebook](https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/jupyter-labs-spacex-data-collection-api.ipynb)
  (https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/jupyter-labs-spacex-data-collection-api.ipynb)

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

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

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

```
[26]: # Calculate the mean value of PayloadMass column
# Replace the np.nan values with its mean value
PayloadMass = pd.DataFrame(data_falcon9['PayloadMass'].values.tolist()).mean(1)
print(PayloadMass)
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

- BeautifulSoup was used to webscrap Falcon 9 launch data from Wikipedia
- Data was then parsed and converted to pandas dataframe and csv
- [Link to the notebook](https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/jupyter-labs-webscraping.ipynb) (https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/jupyter-labs-webscraping.ipynb)

In [3]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

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 [4]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[4]:
```
200
```

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

In [5]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

In [9]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

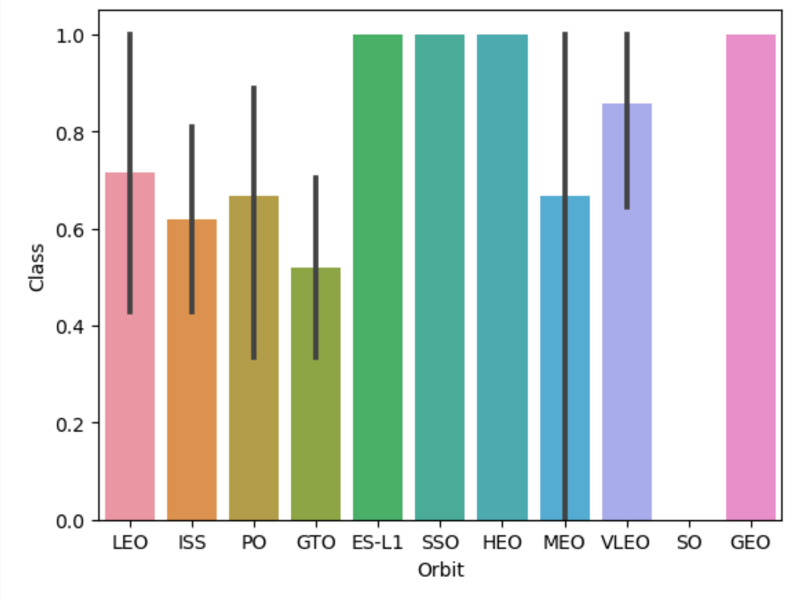
Data Wrangling

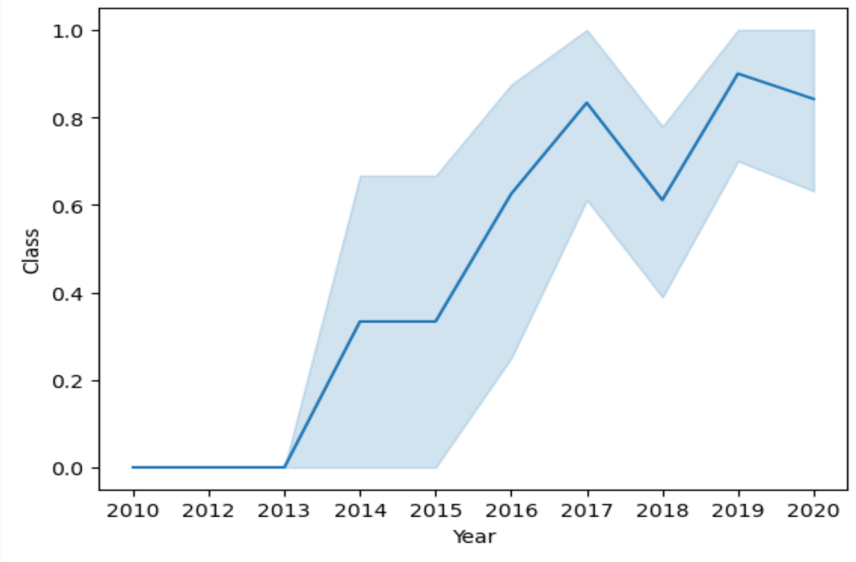
- A landing outcome label was created out of the `Outcome` column into a similar dataframe, then it was exported to .csv
- [Link to the notebook](https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb)(https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb)

EDA with Data Visualization

- An initial data analysis can be shown through visual relationships.
- Two plots are presented stating a relationship between the success rate (`Class`) value and parameters such as orbit or year.

[Link to the notebook](https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/jupyter-labs-eda-dataviz%20(1).ipynb) (https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/jupyter-labs-eda-dataviz%20(1).ipynb)





EDA with SQL

- A SpaceX dataset named `SPACEXTBL` was imported into a SQL database. SQL lite was used in the Jupyter notebook for a EDA to get information from the data.
- Queries done:
  - Unique launch sites
  - Total Payload Mass by NASA
  - Average Payload Mass
  - Successful vs Failure in recorded history
  - Table with failed drone landings, version and landing site.

Link to notebook (https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/jupyter-labs-eda-sql-edx_sqllite.ipynb)

Interactive Map with Folium

- An interactive map can be found in the notebook below. It has information of:
  - Failure/Success in each location (Color graded)
  - Distances to coastline and transit routes
  - Actual locations of landing sites and NASA headquarters.

Link to notebook (https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/lab_jupyter_launch_site_location.ipynb)

Predictive Analysis (Classification)

- Numpy and Pandas libraries were used to do a train-test split
- Diverse machine learning models were implemented to obtain its hyperparameters using `GridSearchCV`
- Acuraccy was the metric used for the model deployment. They were compared using confusion matrixes.

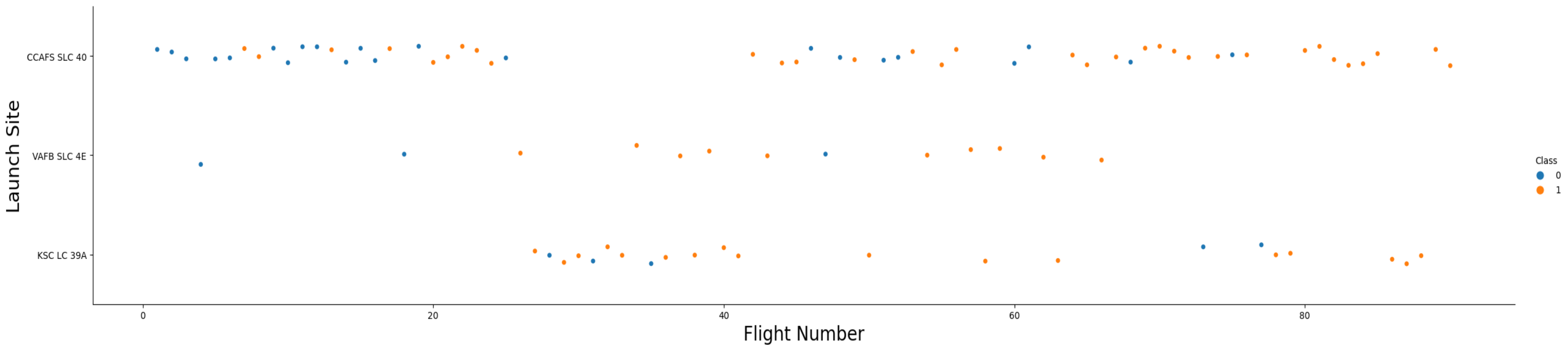
[Link to the notebook](https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb) (https://github.com/JPelleSol/-Data-Science-and-Machine-Learning-Capstone-Project-about-SpaceX-IBM-/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb)

Section 2

Insights drawn from EDA

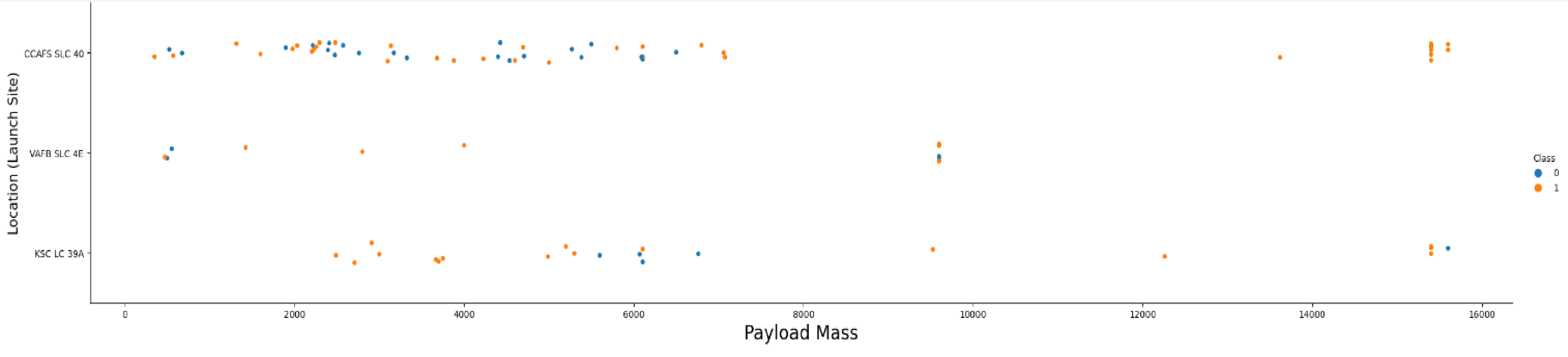
Flight Number vs. Launch Site

- A relation between number of flights per launching site and success rate can be noted.



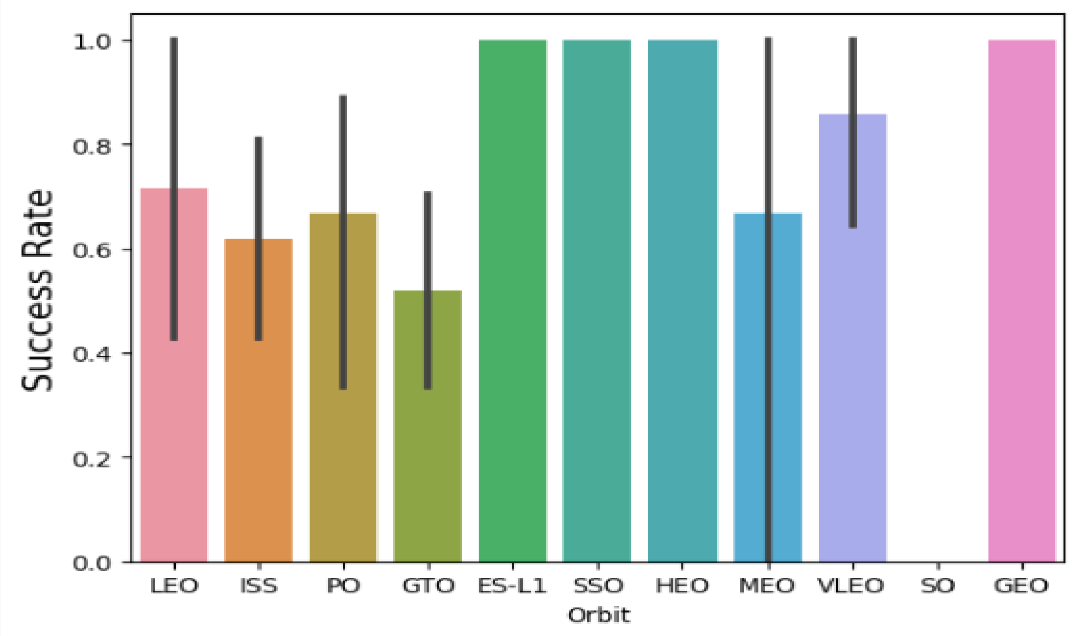
Payload vs. Launch Site

- A relation between payload in each launching site and success rate can be noted

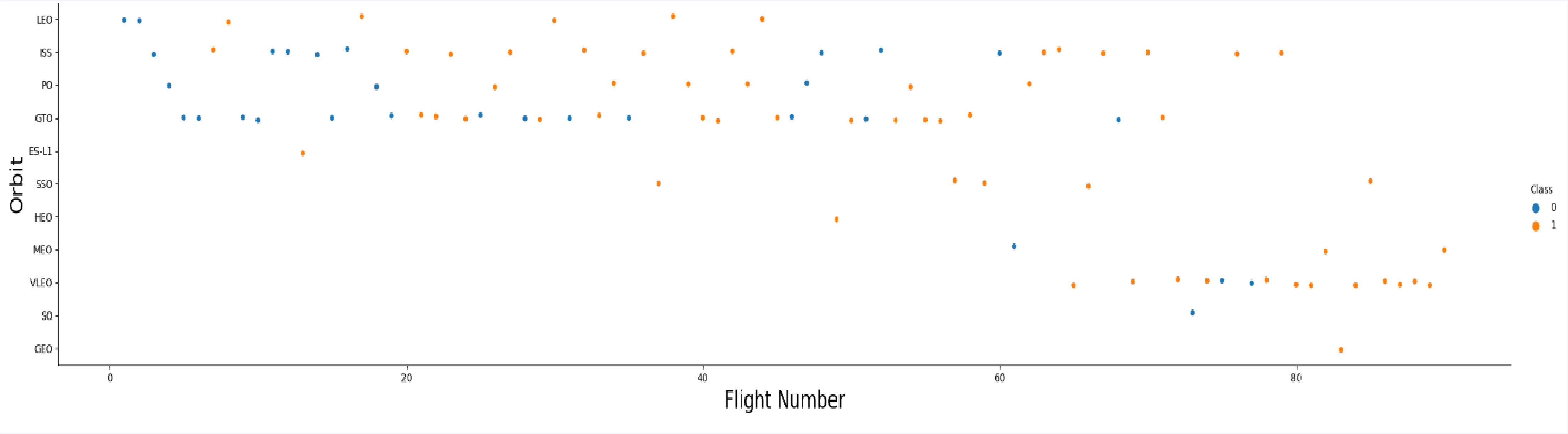


Success Rate vs. Orbit Type

It is observed that ES-L1, HEO, SSO, VLEO and GEO had a higher success rate. One possible cause is the longevity of the test used for the others (such as GTO). This is best seen in the next slide.

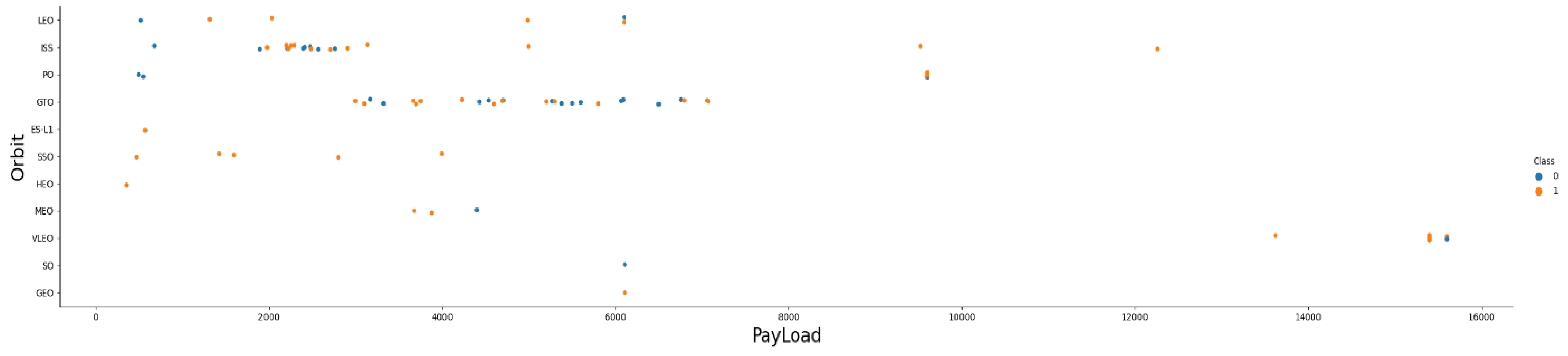


Flight Number vs. Orbit Type



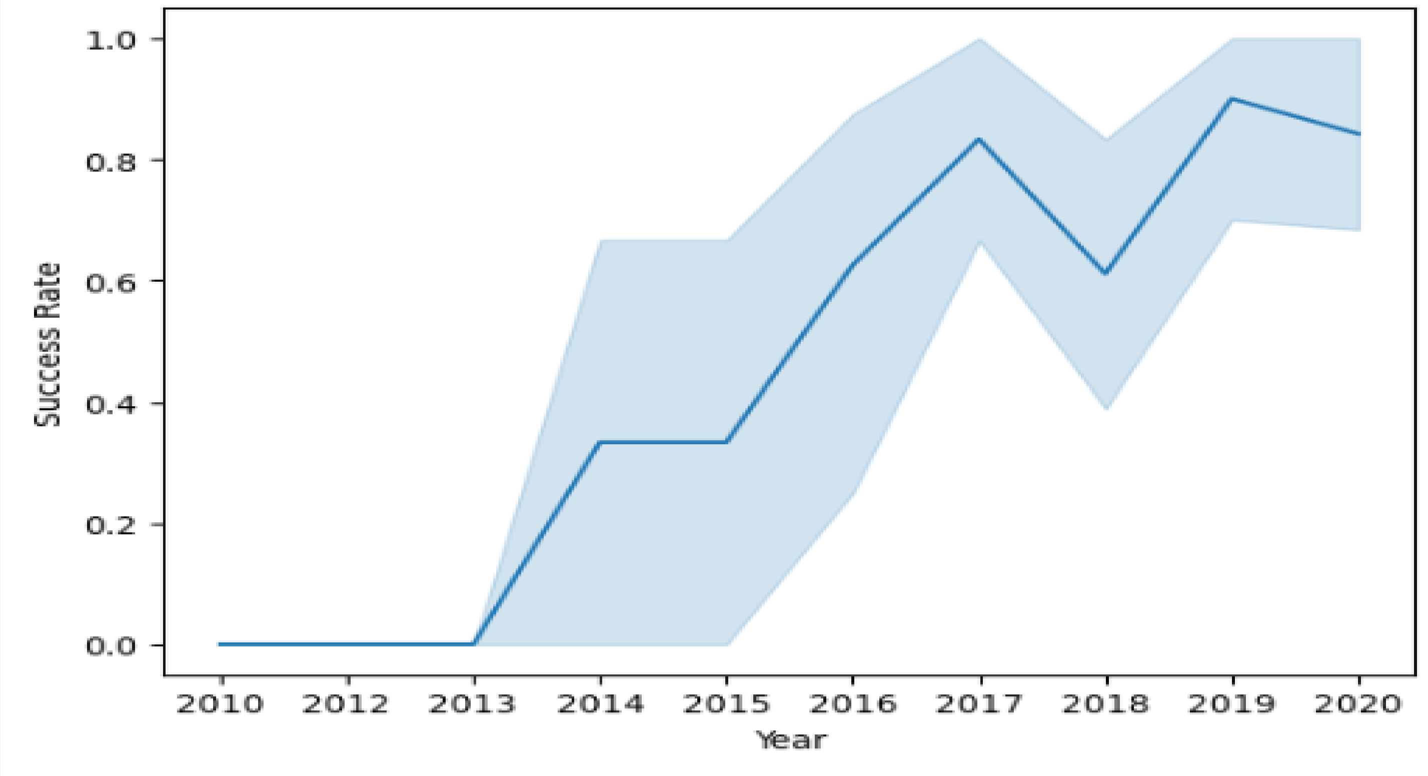
Payload vs. Orbit Type

- Most heavy payloads are for PO, VLEO and ISS orbits. Matching with better success rate for such orbits.



Launch Success Yearly Trend

- A consistent growth in success rate through the years can be stated through the plot



All Launch Site Names

- A simple SQL input is enough to get all of SpaceX unique launch sites. In this case, we use DISTINCT

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

In [22]:
```
%sql    SELECT DISTINCT Launch_Site FROM SPACEXTBL
```

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

Out[22]:

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

Launch Site Names Begin with 'KSC'

- WHERE, a command from SQL allows for searches like this one

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

In [23]:
```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'KSC%' LIMIT 5
```

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

Out[23]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
19-02-2017	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
16-03-2017	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
30-03-2017	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
01-05-2017	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
15-05-2017	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass

- In this case, a SQL query is used to get the total payload mass in Kg.

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

In [25]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass FROM SPACEXTBL WHERE Customer LIKE 'NASA (CRS)'
```

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

Out[25]:

Total_PayloadMass
45596

Average Payload Mass by F9 v1.1

- Another query is used to get the average F9v1.1 payload mass in Kg.

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

In [26]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_PayloadMass FROM SPACEXTBL WHERE Booster_Version= 'F9 v1.1'
```

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

Out[26]:

Avg_PayloadMass
2928.4

First Successful Ground Landing Date

- We can find historical data, such as the first successful landing date, using SQL

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

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

Out[32]:

FirstSuccesful_LandingDate
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

- Complex requests as this one are most simple using SQL. Another query is used for this.

List the names of the boosters which have success in ground pad 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_MASS__KG_ >4000 AND PAYLOAD_MASS__KG_<6000
```

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

Out[33]:

Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

- Two separate queries were run for this question. The results are shown correspondingly in comment form below each one.

```
%sql SELECT COUNT (Mission_Outcome) AS SuccessOutcome FROM SPACEXTBL WHERE Mission_Outcome LIKE 'Success%'
# Output: SuccessOutcome 100
%sql SELECT COUNT (Mission_Outcome) AS FailureOutcome FROM SPACEXTBL WHERE Mission_Outcome LIKE 'Failure%'
# Output: FailureOutcome 1
```

Boosters Carried Maximum Payload

- We can make tables with SQL. In this case, a top payload mass table is done using SELECT.

In [42]:
```
%sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_=(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL) ORDER BY Booster_
```

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

Out[42]:

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

2017 Launch Records

- A SQL table can be as concrete as desired

```
%sql SELECT Booster_Version, Launch_Site, [Landing _Outcome] FROM SPACEXTBL WHERE [Landing _Outcome] LIKE 'Failure%' AND Date BETWEEN '2017-01-01' AND
```

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

- A COUNT can be used like this. Both landing outcome and count are selected with SELECT, specifying LandingOutcome for COUNT. Order by is then used for cleanness.

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

```
%sql SELECT [Landing _Outcome], COUNT([Landing _Outcome]) FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY [Landing _Outcome]
```

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

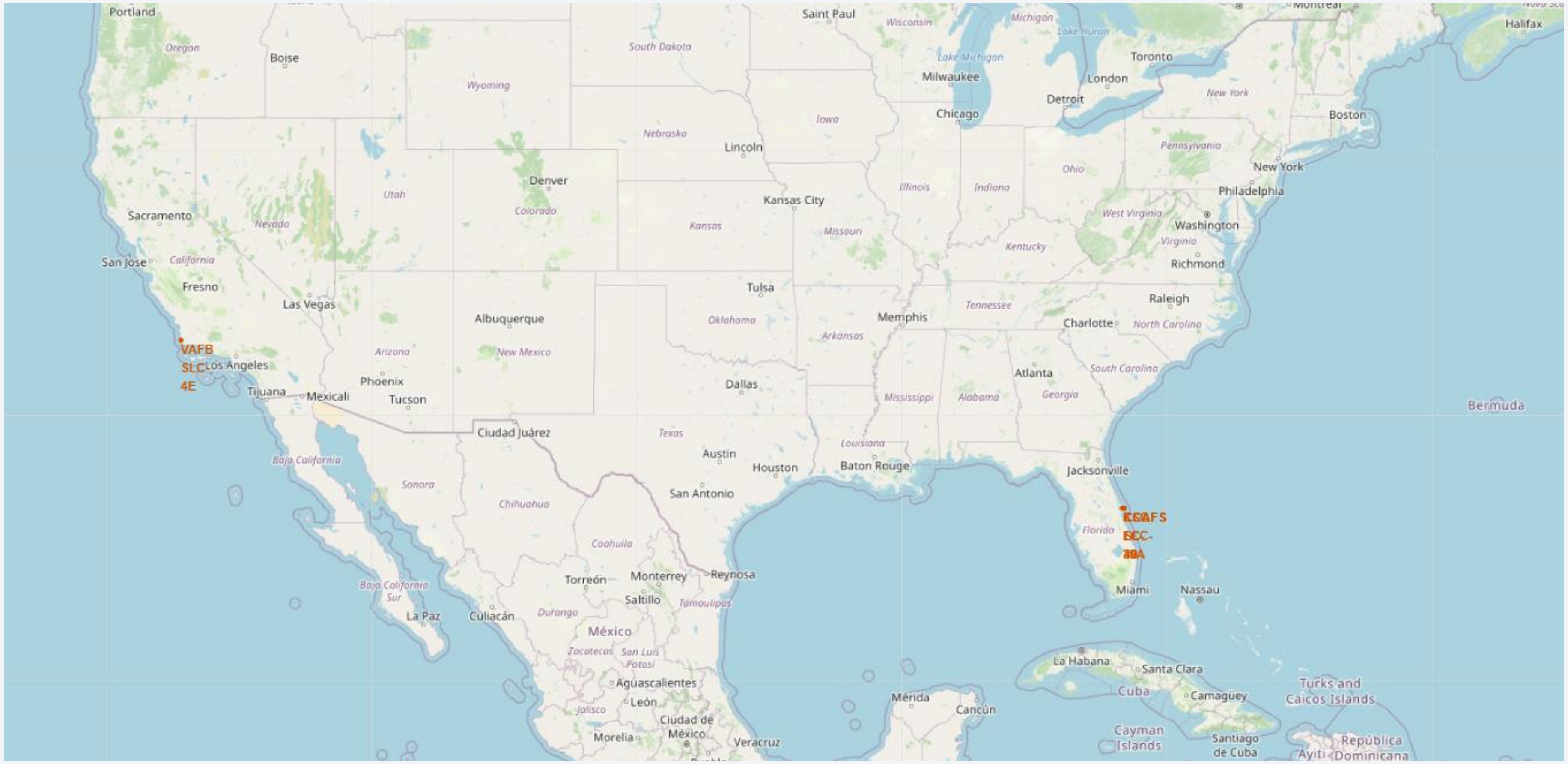
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

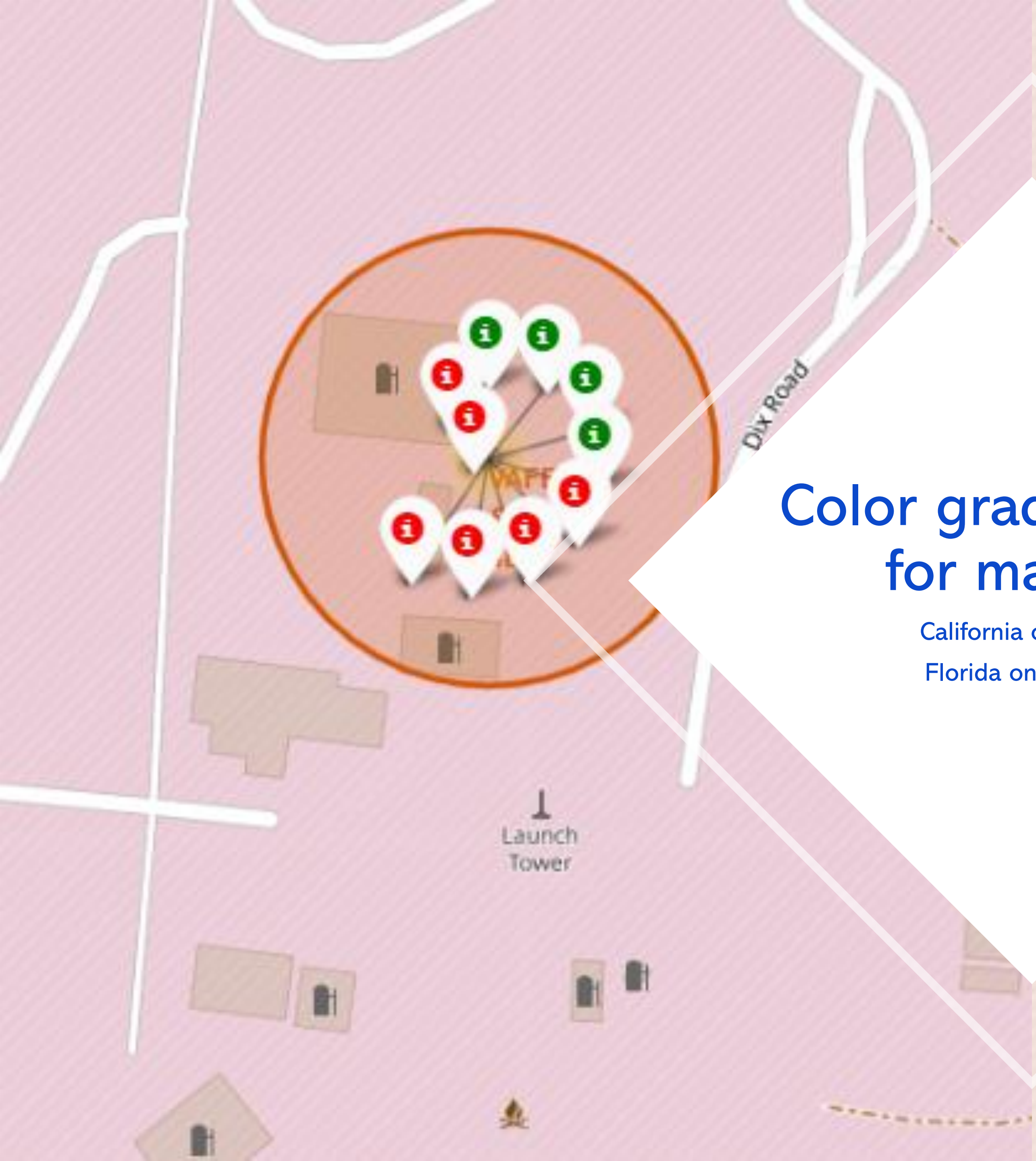
Section 3

Launch Sites Proximities Analysis

Launch sites: Map markers.

- Launch locations are shown in the map

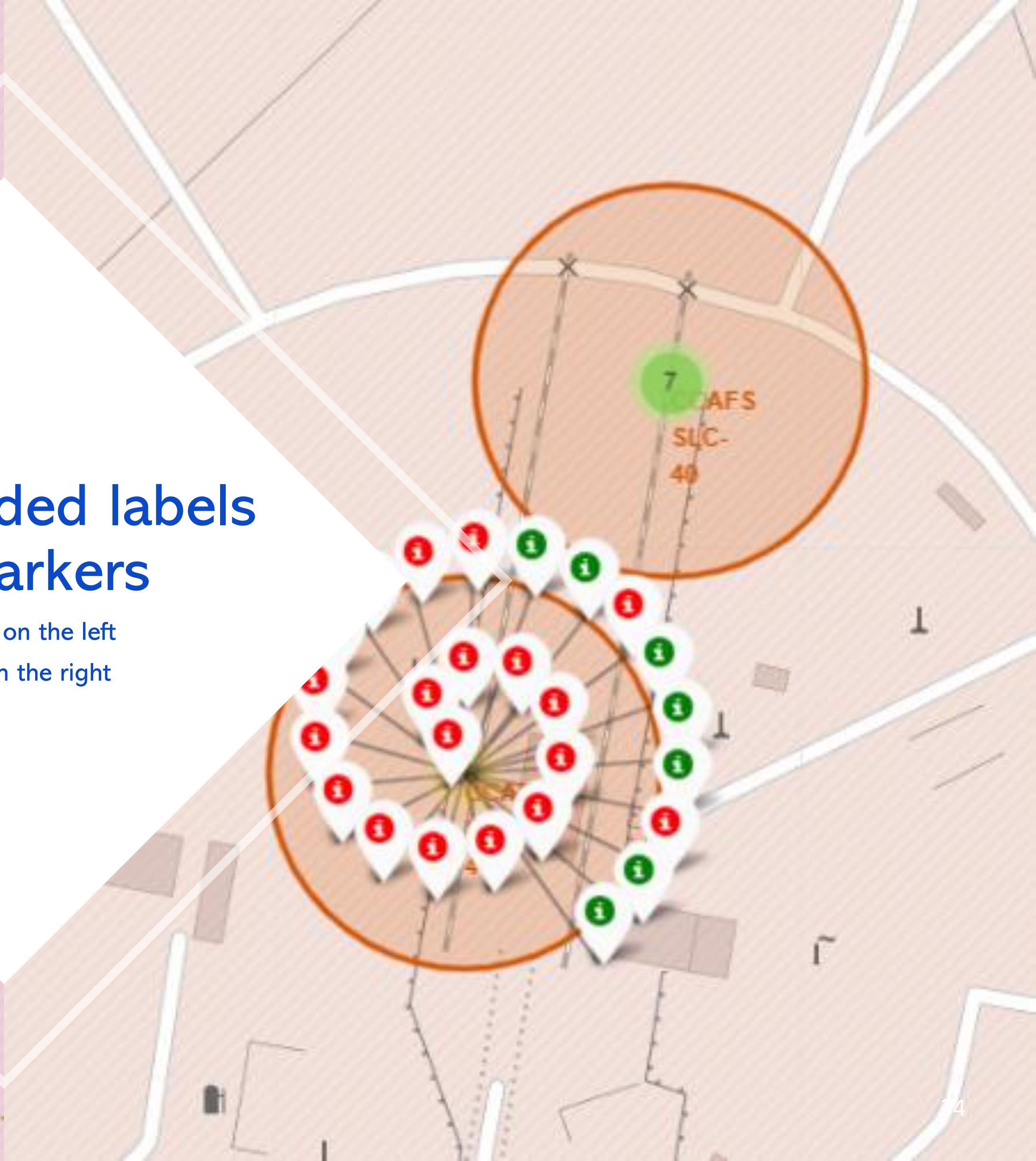




Color graded labels for markers

California on the left

Florida on the right



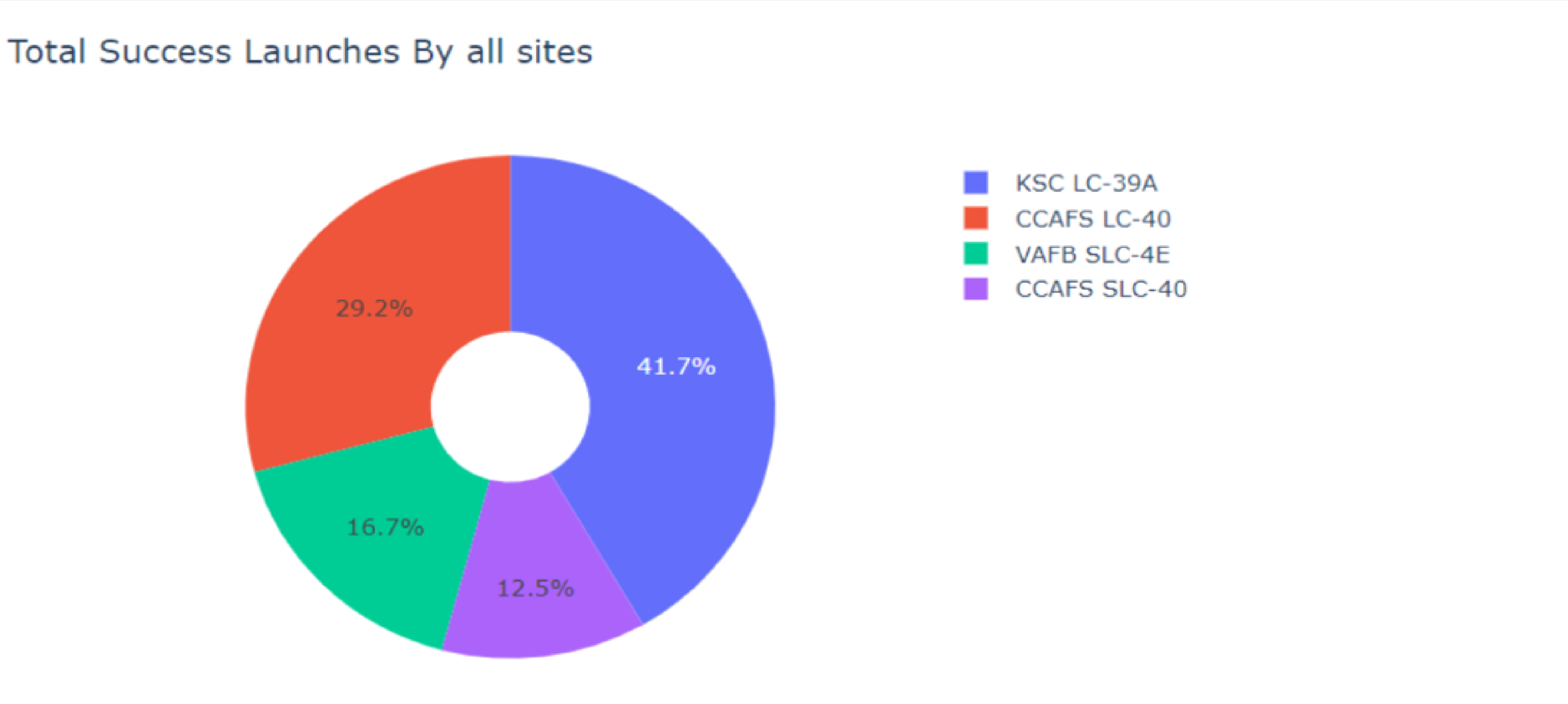
Distances to coastline, highways..

We are able to get distances from the launching site to key locations with folium. This is important for noise and shockwave concerns. We can check that each location is safe for the oceans, transit ways and cities nearby.

Section 4

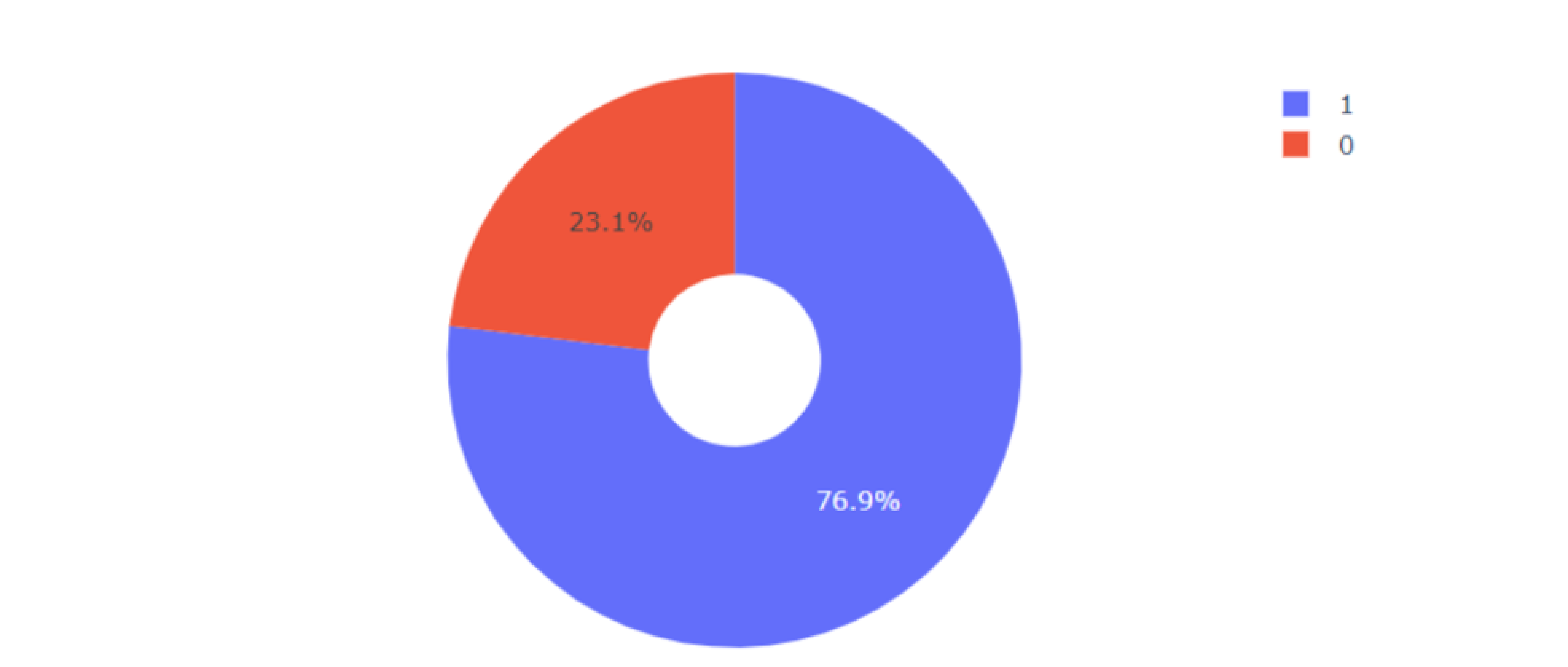
Build a Dashboard with Plotly Dash

Pie chart: Success percentage by launch site



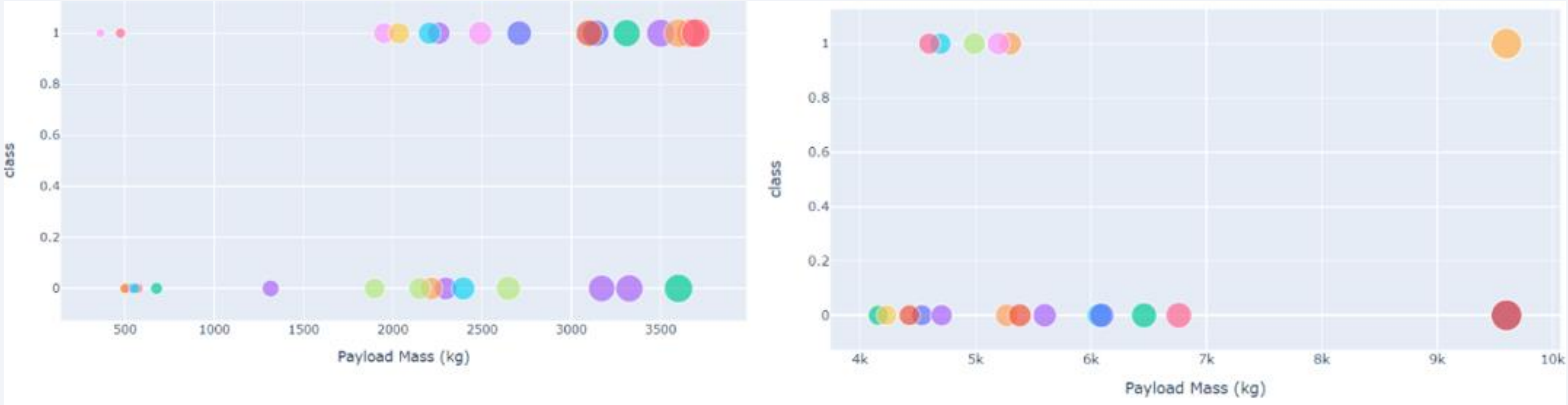
Pie Chart: Launching site with the most success ratio

- Site KSC LC-39A has 76.9% success ratio. In the graph. Result 1 (blue) shows a success (`Class` = 1)



Scatter Plot: Payload and Launch Outcome for all sites (Interactive Graph)

- Two graphs are presented, the first, with the slider set to 0-4000 Kg. The second, with the slider set to 4000-10000 Kg.
- We find success rate (`Class`, on the graph) is higher for low payloads.

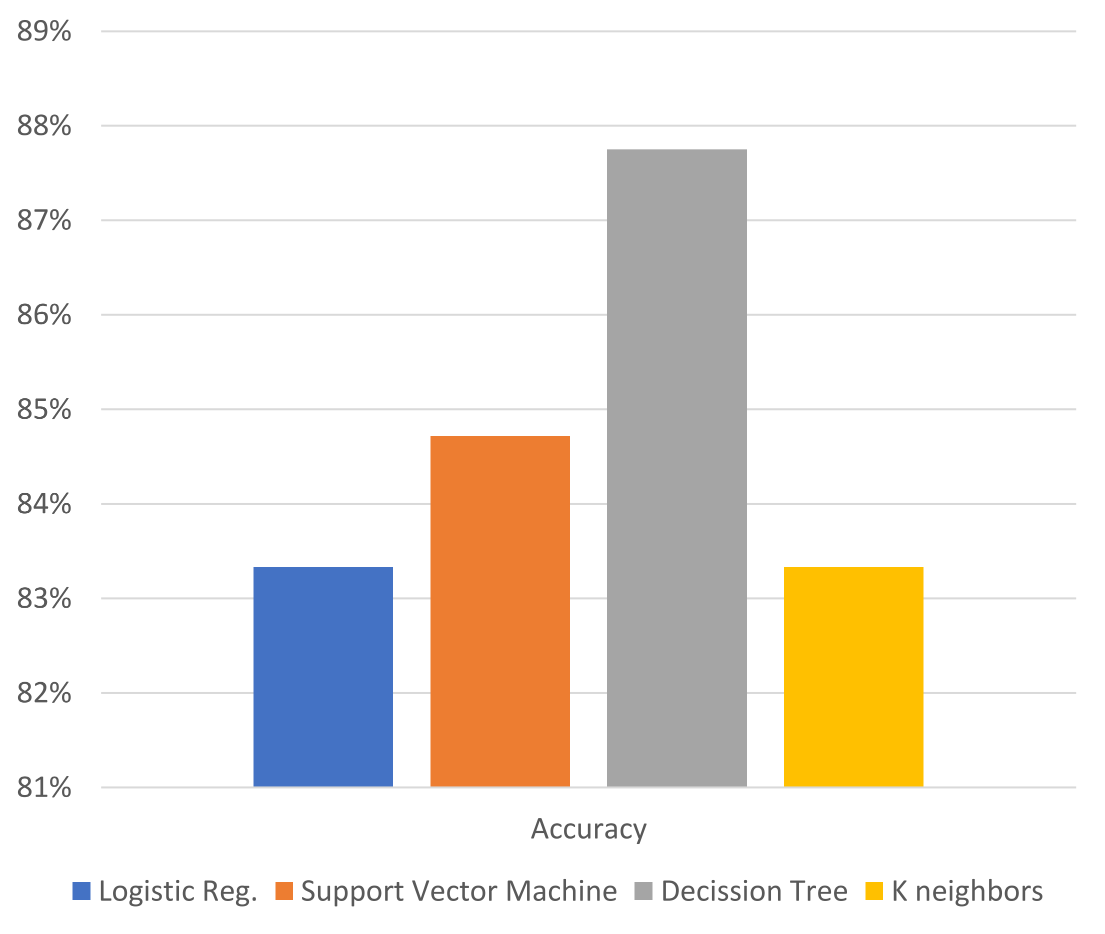


Section 5

Predictive Analysis (Classification)

Classification Accuracy

We find Decission Tree to be the best classification algorithm. However, any of those has similar accuracy.



Find the method performs best:

In [40]:

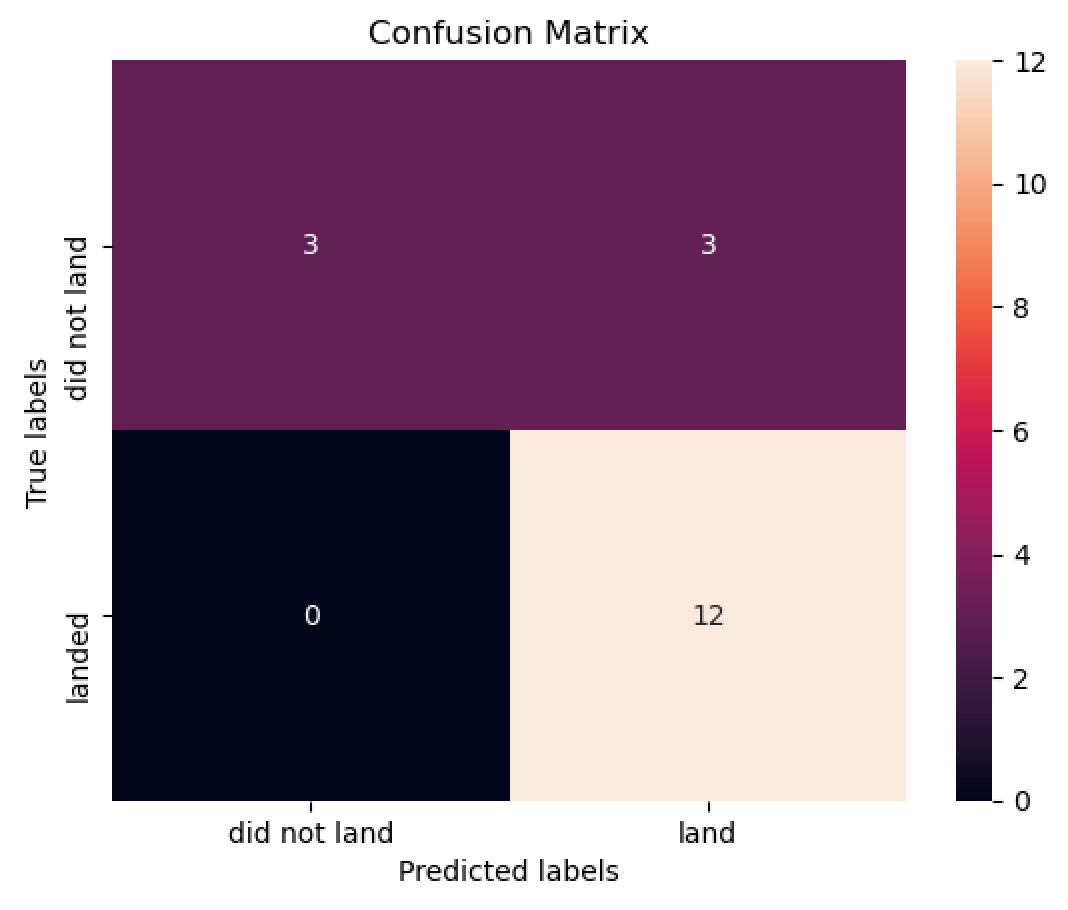
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.875
Best params is : {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}
```

Confusion Matrix

- The confusion matrix shows that the classifier throws a high number of false-positives. This is, unsuccessful landings noted as successful by the Decision Tree Classifier.



Conclusions

It can be concluded that:

- Success Rate can be related to payload mass and flight amount at a launch site. The greater the flight amount, the greater the success, and the lesser the payload mass, the greater the success.
- Orbits ES-L1, HEO, SSO and VLEO showed the most success rate. GEO can be noted, but with only 1 launch successful out of 1 total, it's not conclusive enough.
- Success rates have been improving through the years
- KSC LC-39A is the best location to launch according to success rate.
- From the models tested, the model that best predicts the outcomes of the launches, is a decision tree classifier. Even so, we would have to be weary with false positives from this classifier.

Thank you!