Skills Network

Moreira Ginarte Rodrigo Carlos
October 2023

Outline

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

Executive Summary

- 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)

Introduction

- Project background and context

  SpaceX, a leader in the space industry, strives to make space travel affordable for everyone. Its accomplishments include sending spacecraft to the international space station, launching a satellite constellation that provides internet access and sending manned missions to space. SpaceX can do this because the rocket launches are relatively inexpensive ($62 million per launch) due to its novel reuse of the first stage of its Falcon 9 rocket. Other providers, which are not able to reuse the first stage, cost upwards of $165 million each. By determining if the first stage will land, we can determine the price of the launch. To do this, we can use public data and machine learning models to predict whether SpaceX – or a competing company – can reuse the first stage.

- Problems you want to find answers to

  - What factors determine if the rocket will land successfully?
  - The interaction among features that determine the success of a landing.
  - What operating conditions needs to be in place to ensure a successful landing program.

Methodology

Methodology

Executive Summary

- Data collection methods:
  - The data was collected using SpaceX REST API and web scraping from Wikipedia.
- Perform data wrangling
  - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization libraries and SQL
- Perform interactive maps using Folium and interactive dashboard using PowerBI
- Perform predictive analysis using classification models
  - Logistic regression, Support Vector Machine, Decision Tree Classifier, K Nearest Neighbors

Data Collection

- The data was collected using various methods
  - Data collection was done using get request to the SpaceX REST API.
  - Next, we decoded the response content in JSON format using the .json() function, and then transformed it into a pandas dataframe using .json_normalize().
  - Then, we clean the data, check 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 an HTML table, parse the table, and convert it into a pandas dataframe for future analysis.

Data Collection – SpaceX API

- Make a GET request using the SpaceX API to collect data, clean the requested data, and perform some data wrangling and formatting.
- The notebook's link is: https://github.com/pakyxs/IBM_data_science_project/blob/main/notebooks/01-jupyter-labs-spacex-data-collection-api.ipynb

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

In [12]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

In [13]:
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

In [30]:
```
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 – Web Scraping

- Perform web scrapping to webscrap Falcon 9 launch records using BeautifulSoup.
- Parse the table and converted it into a pandas dataframe.
- The notebook's link is: https://github.com/pakyxs/IBM_data_science_project/blob/main/notebooks/02-jupyter-labs-webscraping.ipynb

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# 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[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.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>
```

3. Extract all column names from the HTML table header

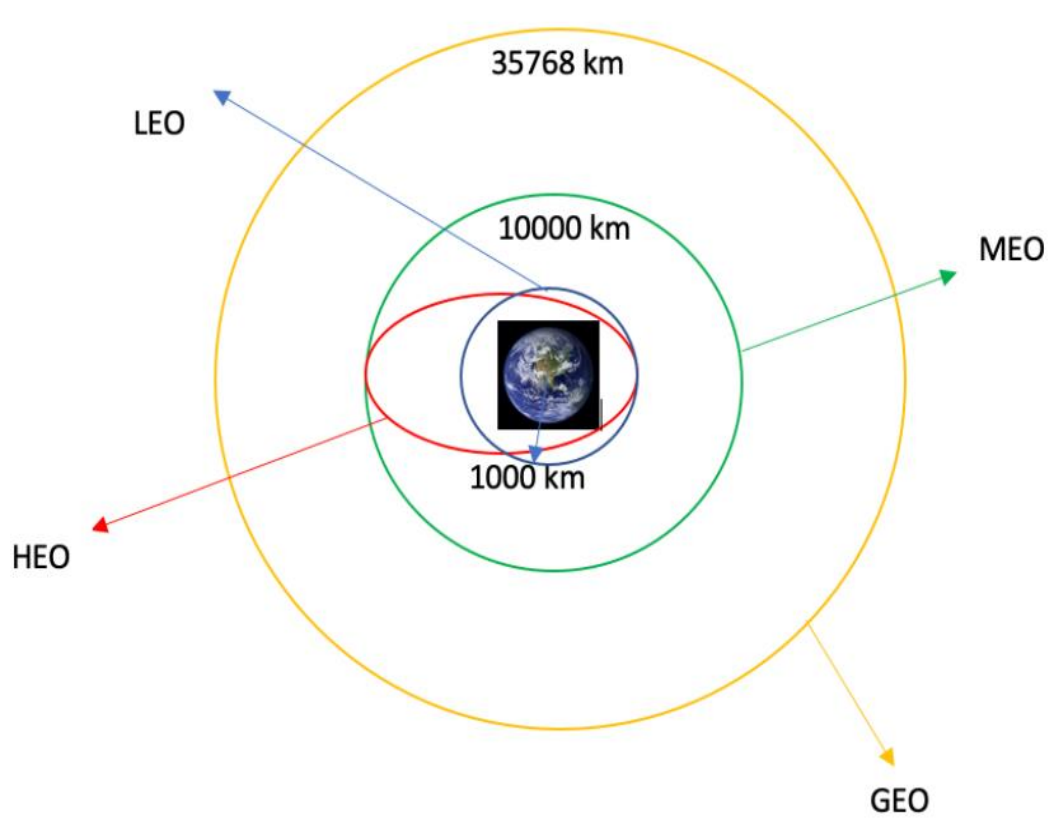
In [10]:
```
column_names = []

# 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
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

Data Wrangling



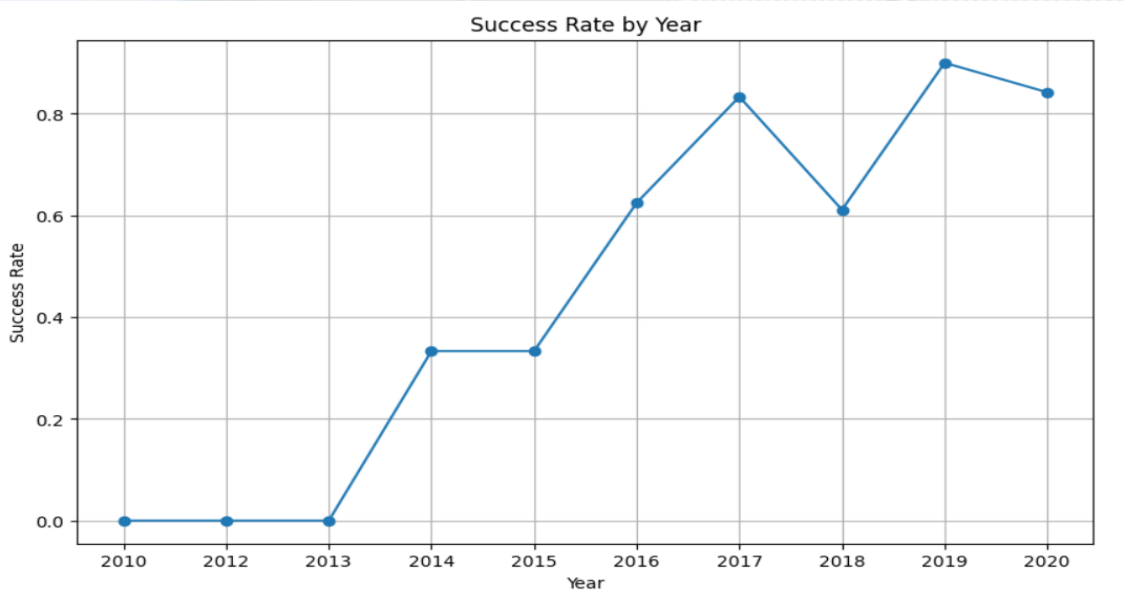
- Perform exploratory data analysis and determined the training labels.
- Calculate the number of launches at each site, and the number and occurrence of each orbits
- Create landing outcome label from outcome column and exported the results to csv.
- The notebook's link is: https://github.com/pakyxs/IBM_data_science_project/blob/main/notebooks/03-labs-jupyter-spacex-Data%20wrangling.ipynb

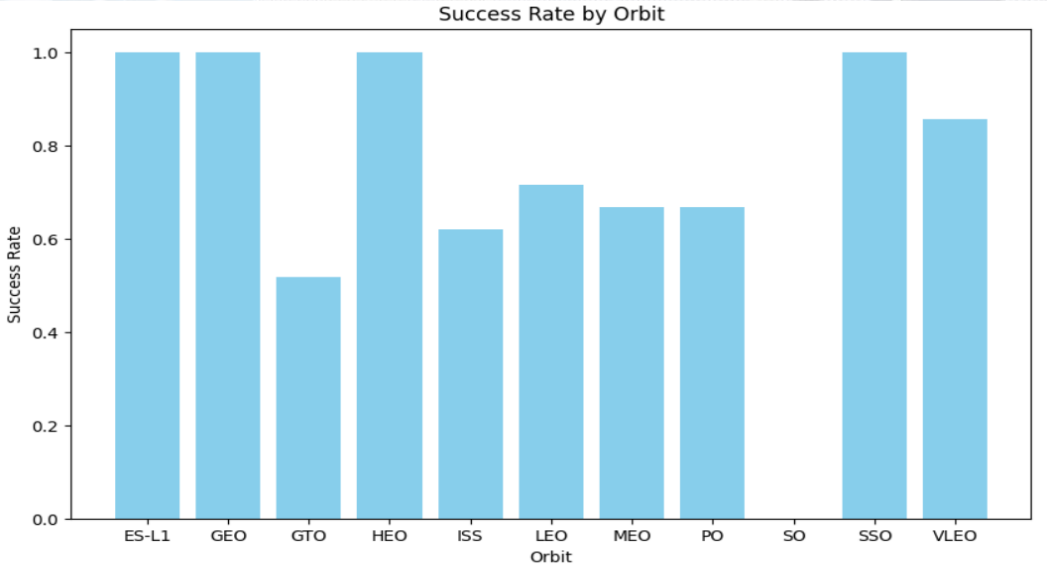
EDA with SQL

- Load SpaceX dataset into a SQLLite database in jupyter notebook.
- Perform EDA with SQL to get insights from the data. Write queries to find out for instance:
  - Names of unique launch sites in the space mission.
  - Total payload mass carried by boosters launched by NASA (CRS).
  - Average payload mass carried by booster version F9 v1.1.
  - Total number of successful and failure mission outcomes.
  - Failed landing outcomes in drone ship, their booster version and launch site names.
- The notebook's link is: https://github.com/pakyxs/IBM_data_science_project/blob/main/notebooks/04-jupyter-labs-eda-sql-coursera_sqllite.ipynb

EDA with Data Visualization

- Explore 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 notebook's link is: https://github.com/pakyxs/IBM_data_science_project/blob/main/notebooks/05-IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

Build an Interactive Map with Folium

- Mark all launch sites, and add map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We 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, we identified which launch sites have relatively high success rate.
- We 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.

Build a Dashboard with PowerBI

- Build an interactive dashboard using PowerBI
- Plot donut charts showing launches and success by launch sites.
- Draw scatter plots showing the evolution of % success rate in time and launch success by payload mass and launch sites.
- Draw histogram showing success rate by orbit.
- The dashboard's link is: https://github.com/pakyxs/IBM_data_science_project/blob/main/dashboard/Dashboard-IBM-SpaceX.pbix

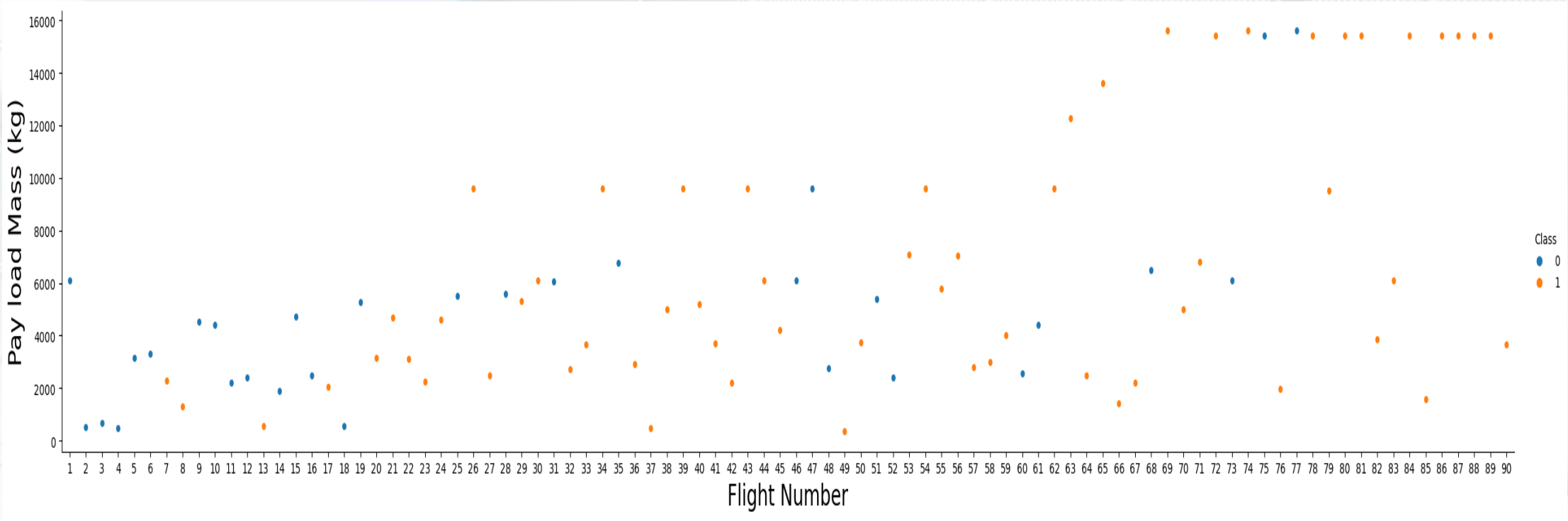
Predictive Analysis (Classification)

- Loadthe data using numpy and pandas, transformed the data, split our data into training and testing.
- Build different machine learning models and tune different hyperparameters using GridSearchCV.
- Use accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- Find the best performing classification model.
- The notebook's link is: https://github.com/pakyxs/IBM_data_science_project/blob/main/notebooks/07-IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Insights

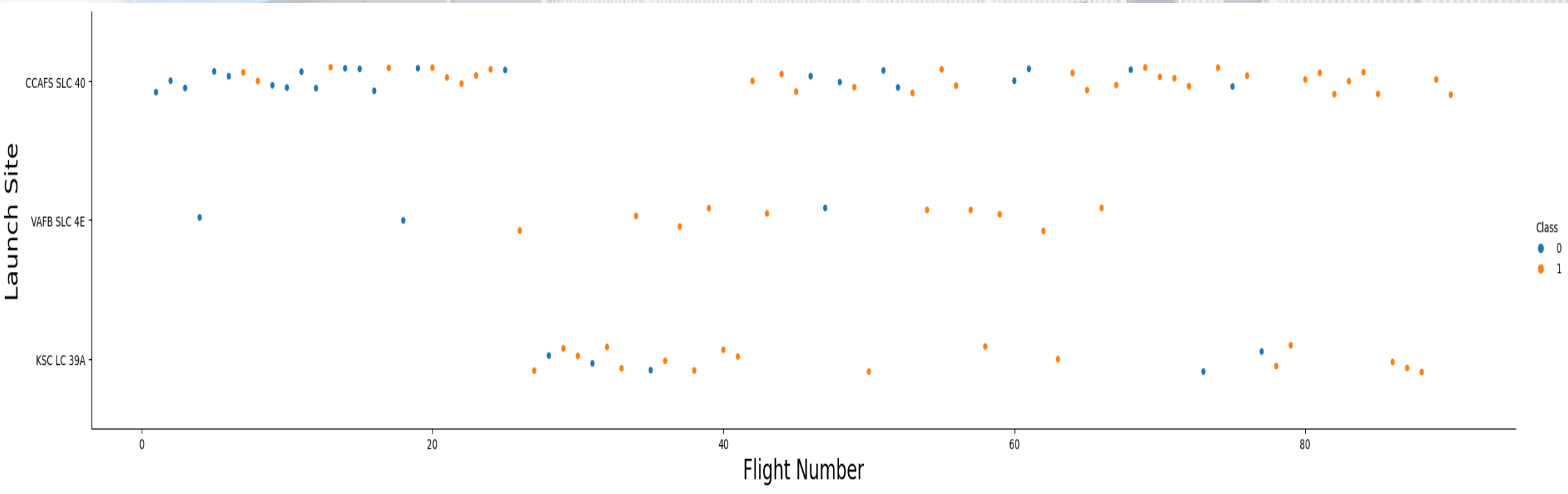
Flight Number vs. Launch Site

- The larger the flight amount at a launch site, the greater the success rate at a launch site.



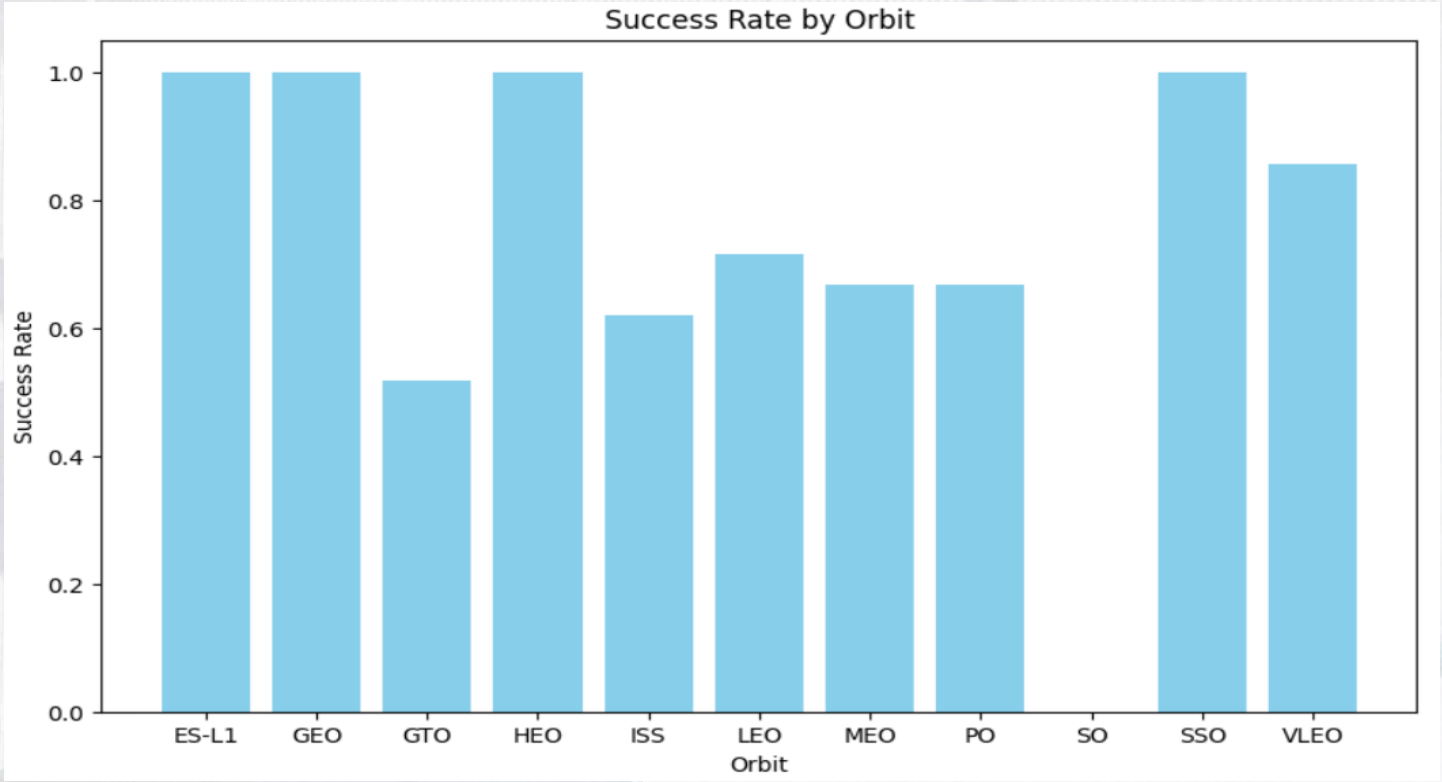
Flight Number vs. Launch Site

- KSC LC-39A has the highest success rate among landing sites.



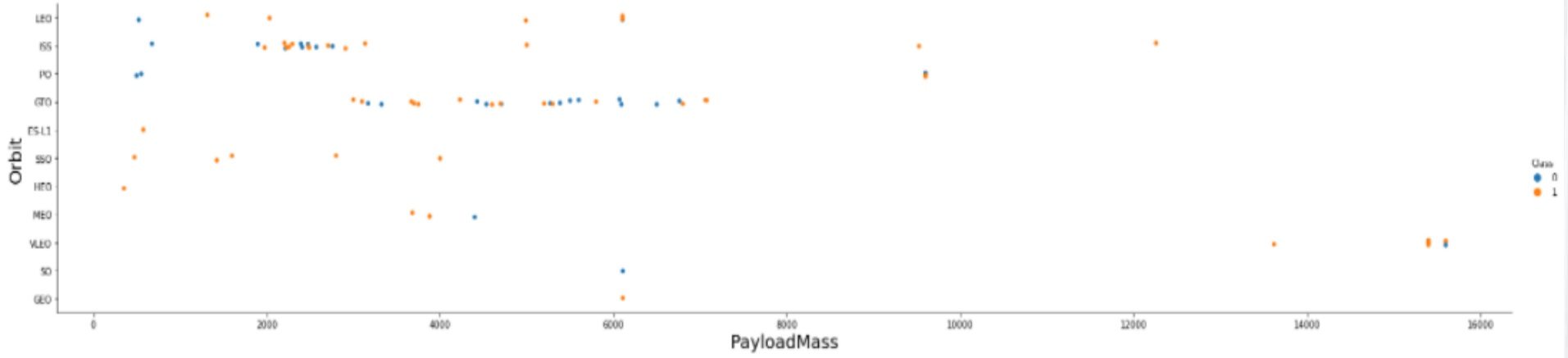
Success Rate by Orbit

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



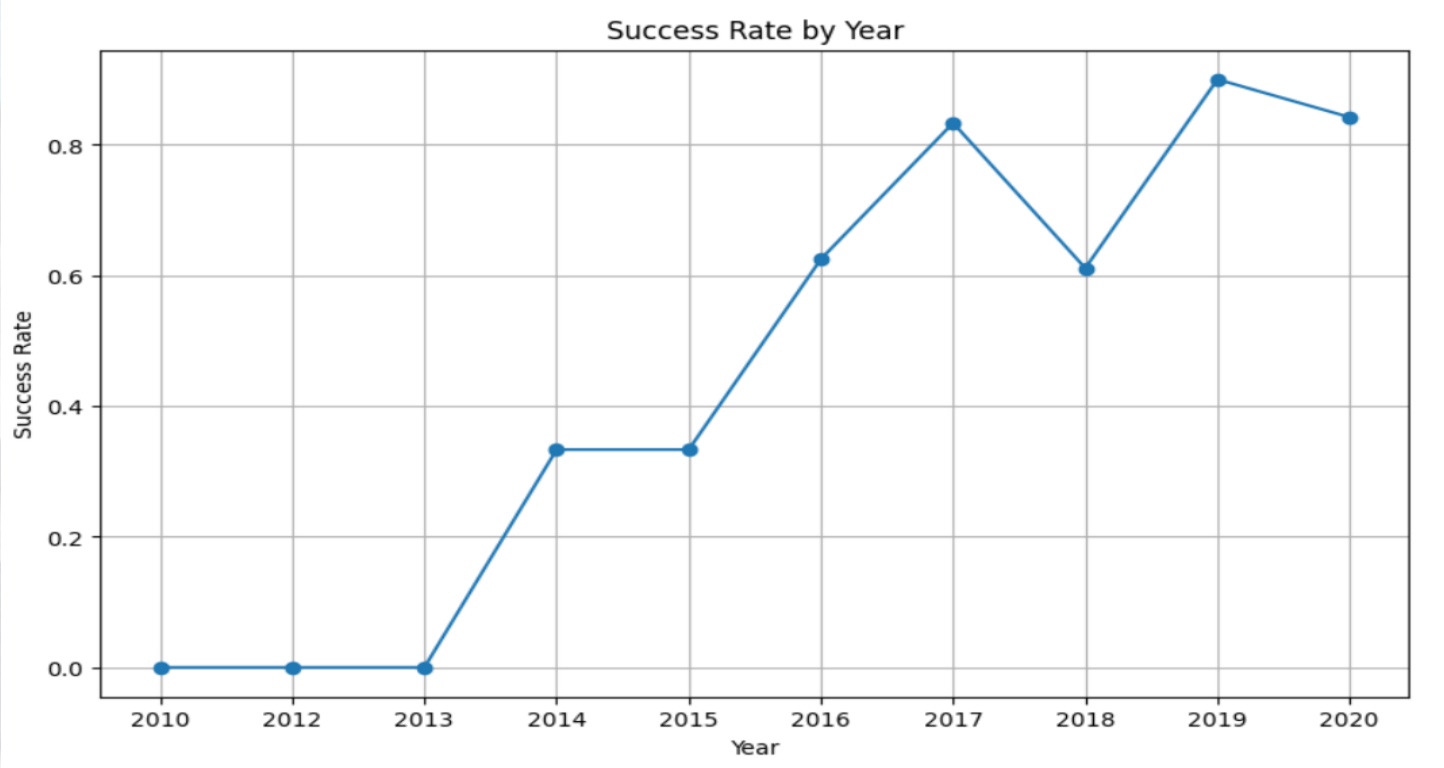
Payload vs. Orbit Type

- For heavy payloads, the best success rate is achieved in PO, LEO, and ISS orbits.



Success rate by year

- Success rate increased from 2013 until 2020.



All Launch Site Names

- Use **DISTINCT** to show unique launch sites from the SpaceX data.

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Total Payload Mass

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

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

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[12]:

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1

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

- Observe that the date of the first successful landing outcome on ground pad was 22nd December 2015

In [14]:

```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [16]:
```
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

Out[16]:

	failureoutcome
0	1

- Use wildcard like '%' to filter for **WHERE** MissionOutcome was a success or a failure.

Boosters Carried Maximum Payload

- Determine 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 [17]:

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

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

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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

Out[19]:

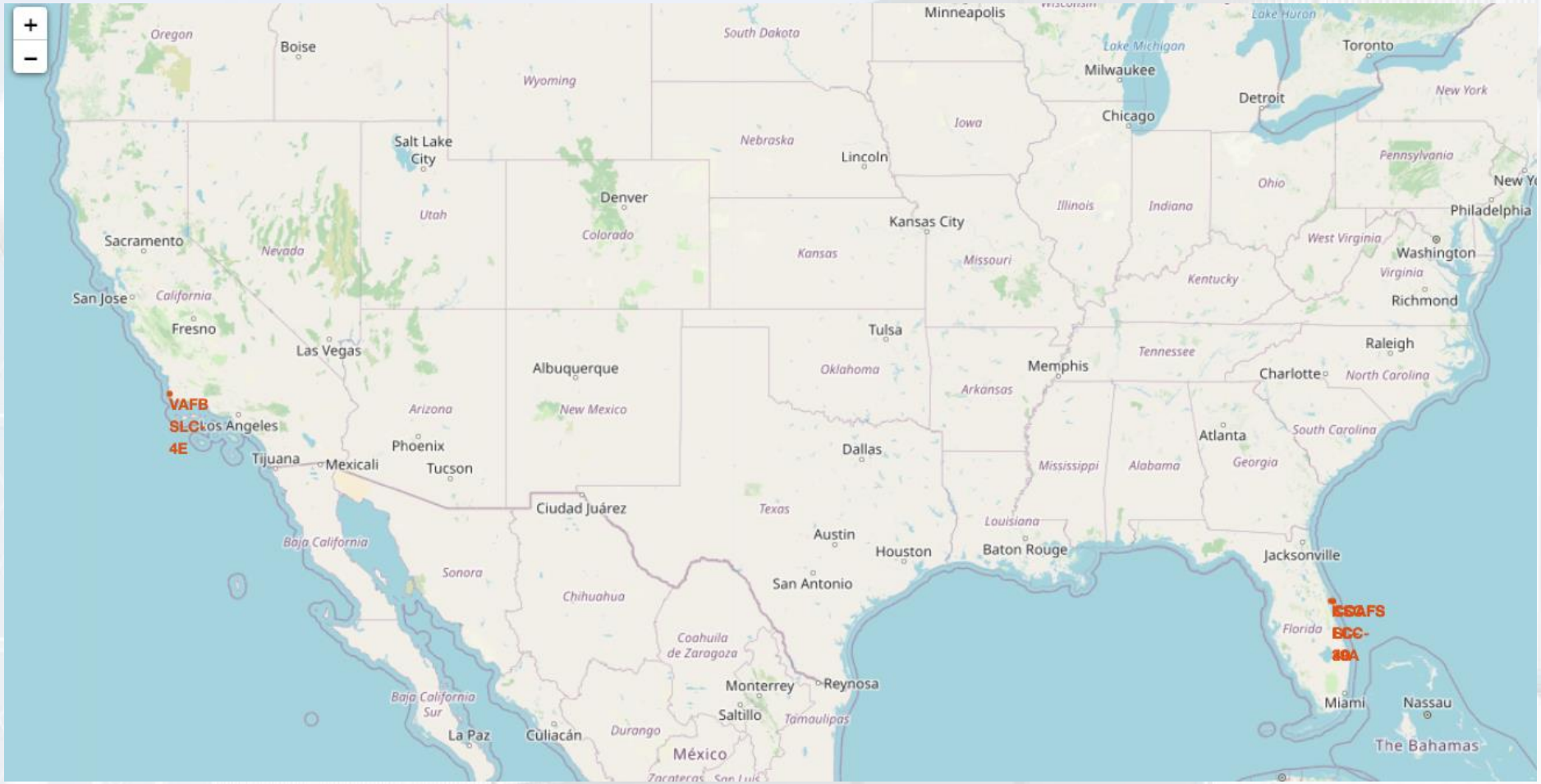
	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

- Select landing outcomes and **COUNT** landing outcomes from the data and use a **WHERE** clause to filter landing outcomes **BETWEEN** 2010-03-20 and 2010-06-04.
- Apply **GROUP BY** clause to group the landing outcomes and **ORDER BY** clause to order the grouped landing outcome in descending order.

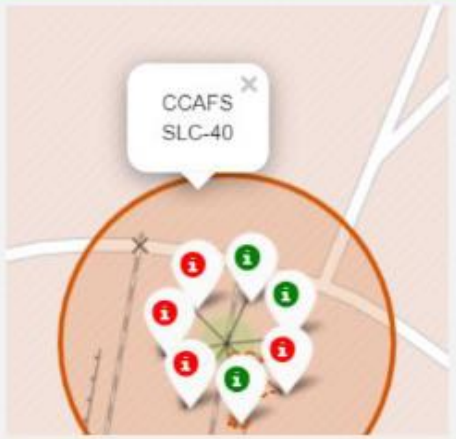
Launch Sites map with Folium

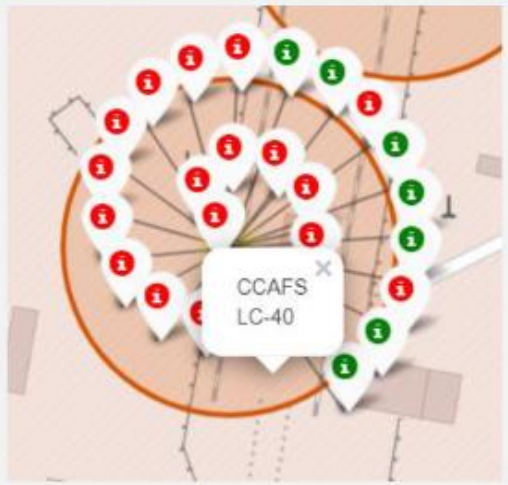
All launch sites global map markers

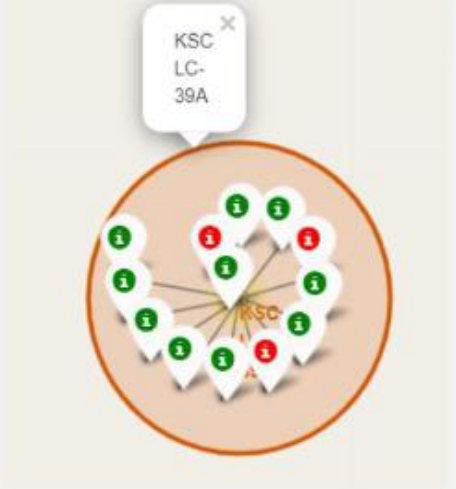
- The launch sites are located on the east and west coasts of the United States of America.



Markers showing launch sites with color labels







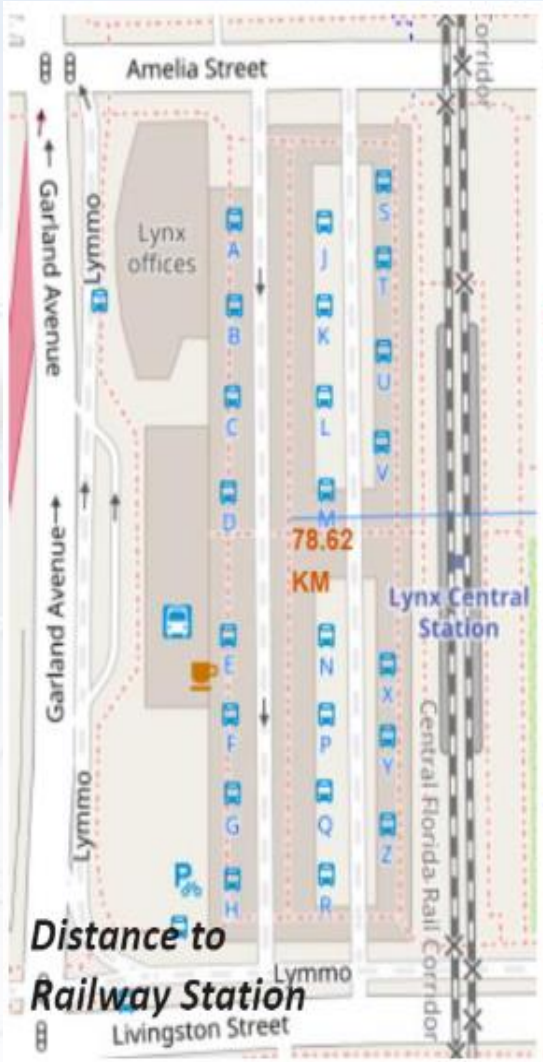
Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

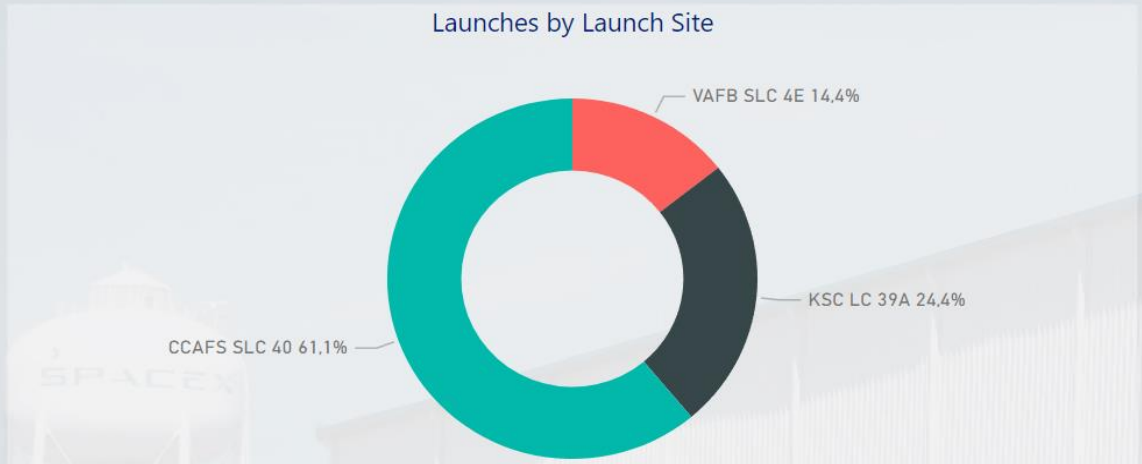
Launch Site distance to landmarks



- 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

Dashboard with PowerBI

Overall analysis

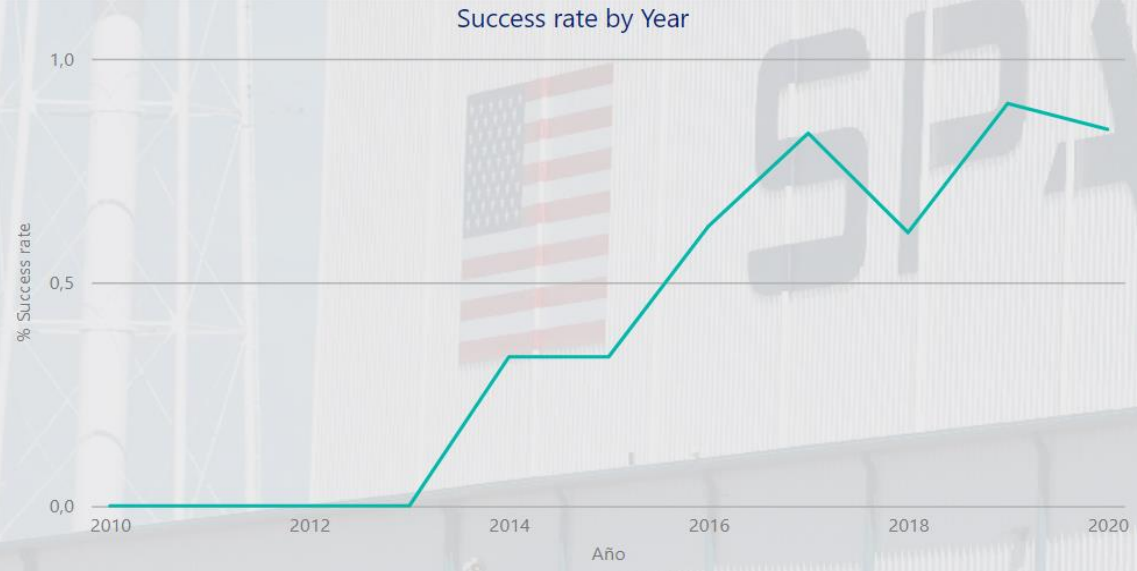


161
Number of flights

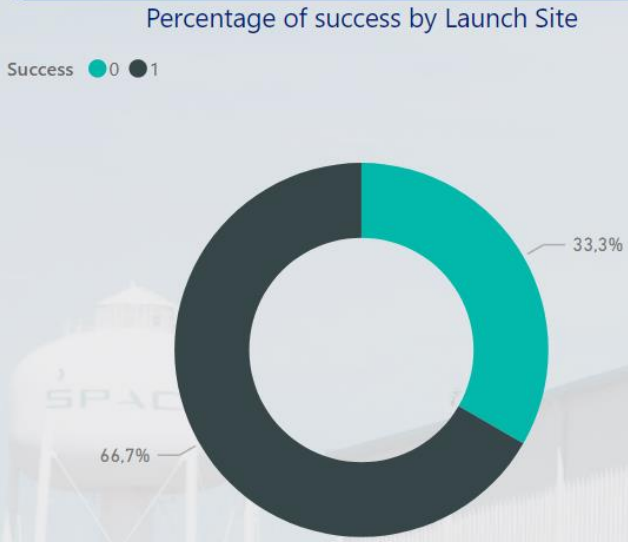
90
Number of reused boosters

Filters

Año, Trimestre
- Select all
- 2010
- 2011
- 2012
- 2013



Launch Site Analysis

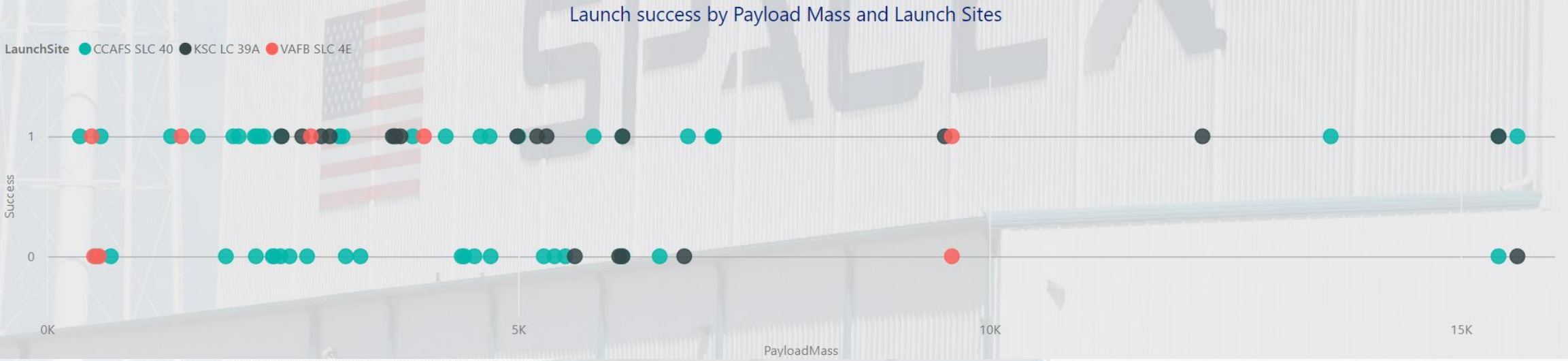


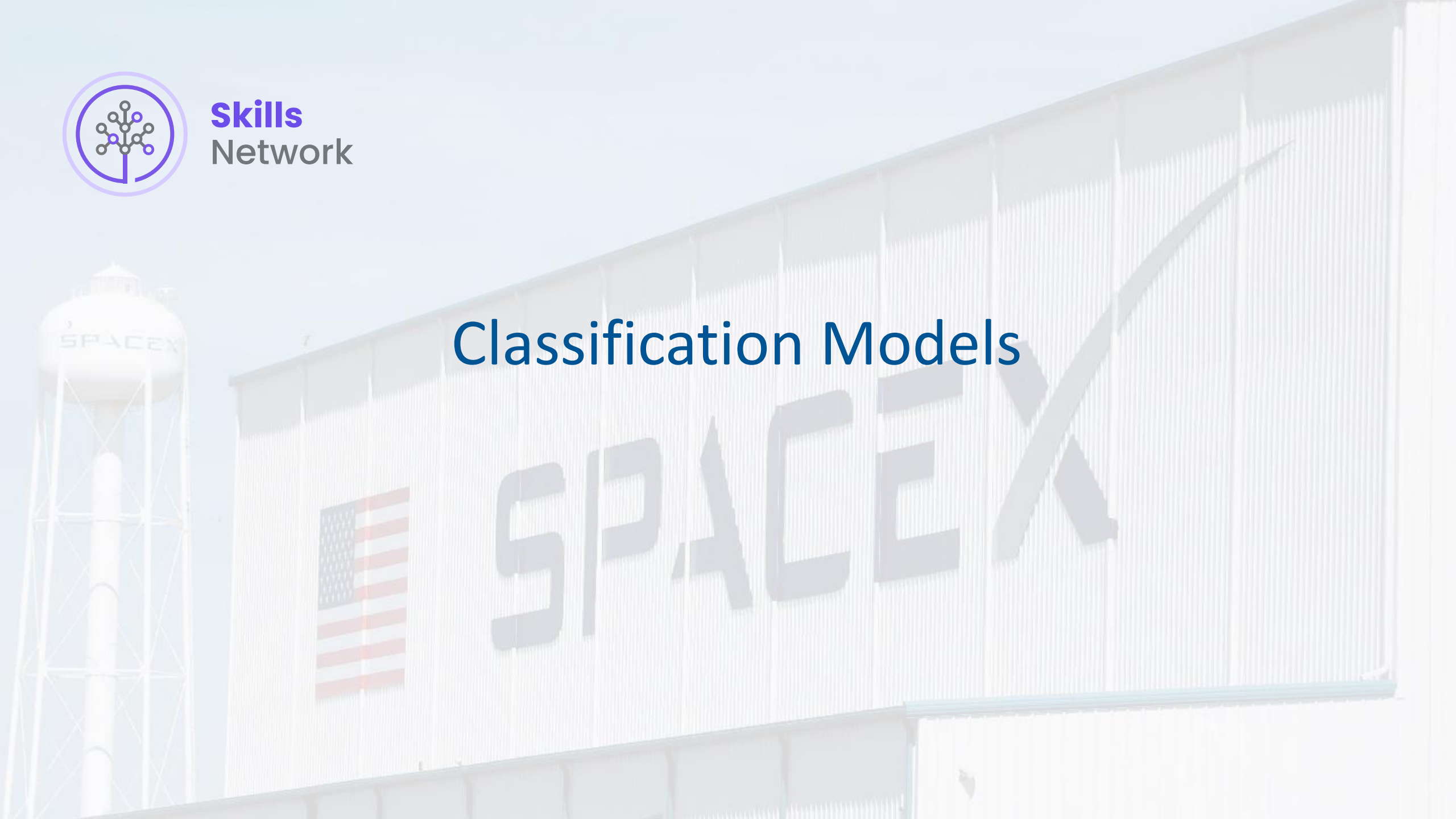
161
Number of flights

90
Number of reused boosters

Filters

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





Classification Models

Classification Accuracy

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

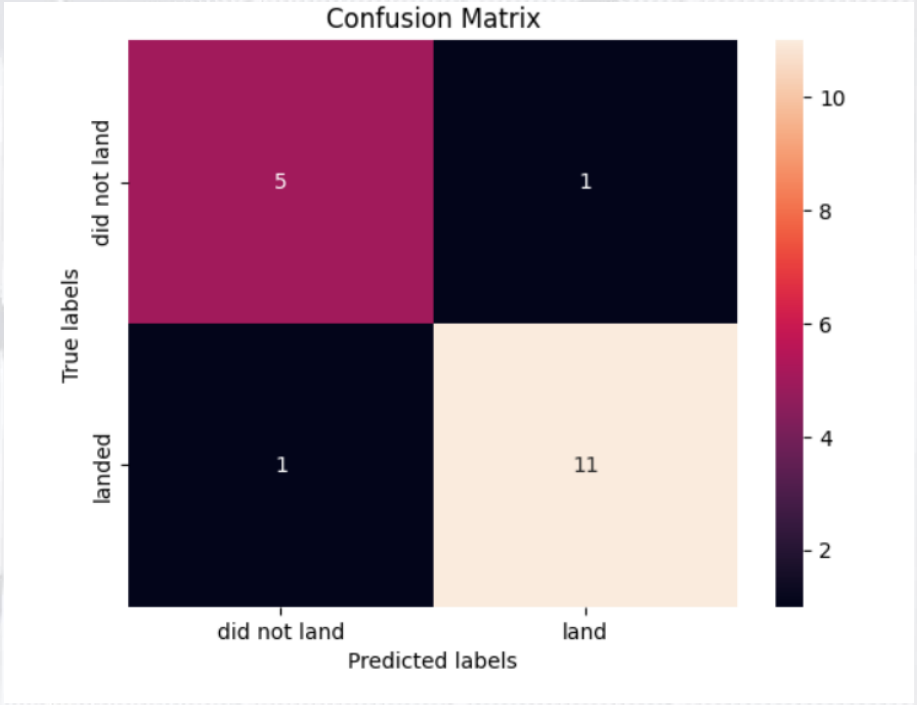
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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.



Conclusions

We can conclude that:

- **Model Performance:** The models performed similarly on the test set with the decision tree model slightly outperforming.
- **Equator:** Most of the launch sites are near the equator for an additional natural boost - due to the rotational speed of earth - which helps save the cost of putting in extra fuel and boosters.
- **Coast:** All the launch sites are close to the coast just in case of failure of the launch, the satellite does not fall on built-up hinterland.
- **Launch Success:** Increases over time thanks to new technology.
- **KSC LC-39A:** Has the highest success rate among launch sites. Has a 100% success rate for launches less than 5,500 kg
- **Orbits:** ES-L1, GEO, HEO, and SSO have a 100% success rate.
- **Payload Mass:** Across all launch sites, the higher the payload mass (kg), the higher the success rate.


Skills
Network
Thank you!
SPACEX