



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Francisco Javier Carrillo
July 23, 2024



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

Space X advertises Falcon 9 rocket launches on its site for \$62 million, significantly less than competitors' offerings which start at \$165 million per launch. This cost disparity is largely due to Space X's ability to reuse the first stage. Therefore, accurately predicting the first stage landing can directly influence launch costs. This predictive capability is crucial for other companies wishing to compete with Space X in rocket launch bids. The objective of this project is to develop a machine learning pipeline for precisely predicting the success of the first stage landing.

- Finding answers to problems:
 - What factors contribute to the successful landing of a rocket?
 - The interplay of different factors determines the success rate of a landing.
 - What operational conditions must be met 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 gathered through diverse methodologies:

Initially, data collection involved making GET requests to the SpaceX API.



Subsequently, the JSON response content was decoded using the `.json()` function and converted into a pandas dataframe using `.json_normalize()`.



Following this, we conducted data cleaning, addressed missing values, and filled them appropriately.



Additionally, we utilized web scraping techniques with BeautifulSoup to extract Falcon 9 launch records from Wikipedia. The goal was to retrieve launch data stored in HTML tables, parse it, and convert it into a pandas dataframe for subsequent analysis.

Data Collection – SpaceX API

- We 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/fran-carrillo/spaceY/blob/main/jupyter-labs-spacex-data-collection-api.ipynb>

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

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

```
In [9]: static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
<div style="background-color: #ccc; height: 15px; width: 100%; position: relative;">
  <div style="position: absolute; left: 5px; top: 5px;">
    <div style="width: 100%; height: 100%; background-color: #ccc;">
```

We should see that the request was successful with the 200 status response code

```
In [10]: response.status_code
```

```
Out[10]: 200
```

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

```
In [13]: # Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
```

```
In [26]: # Hint data['BoosterVersion']!='Falcon 1'
filt = df['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[filt]
data_falcon9.head
```

```
In [27]: data_falcon9.loc[:, 'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
In [3]: # Calculate the mean value of PayloadMass column
plm_mean = data_falcon9['PayloadMass'].mean()
print(plm_mean)
```


Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is <https://github.com/fran-carrillo/spaceY/blob/main/jupyter-labs-webscraping.ipynb>

```
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)
```

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
```

```
beautiful_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
```

```
beautiful_soup.title
```

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

```
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 = beautiful_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

```

Check the extracted column names

```
In [11]: print(column_names)
```

```
[ 'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome', 'Flight No.', 'Da
```

Data Wrangling

- We conducted exploratory data analysis to define the training labels.
- This involved calculating the frequency of launches at each site and the distribution of orbits.
- From the outcome column, we generated landing outcome labels and saved the results to a CSV file.
- The link for this notebook is <https://github.com/fran-carrillo/spaceY/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb>

```
In [7]: # Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

```
Out[7]: CCAFS SLC 40    55
        KSC LC 39A    22
        VAFB SLC 4E    13
        Name: LaunchSite, dtype: int64
```

```
In [8]: # Apply value_counts on Orbit column
df['Orbit'].value_counts()
```

```
Out[8]: GTO      27
        ISS      21
        VLEO     14
        PO       9
        LEO       7
        SSO       5
        MEO       3
        ES-L1     1
        HEO       1
        SO        1
        GEO       1
        Name: Orbit, dtype: int64
```

```
In [9]: landing_outcomes = df.value_counts('Outcome')
        landing_outcomes
```

```
Out[9]: Outcome
        True ASDS    41
        None None    19
        True RTLS    14
        False ASDS    6
        True Ocean    5
        False Ocean   2
        None ASDS     2
        False RTLS    1
        dtype: int64
```

```
In [15]: # Landing_class = 0 if bad_outcome
          # Landing_class = 1 otherwise

          landing_class = []
          for outcome in df['Outcome']:
              if outcome in bad_outcomes:
                  landing_class.append(0)
              else:
                  landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

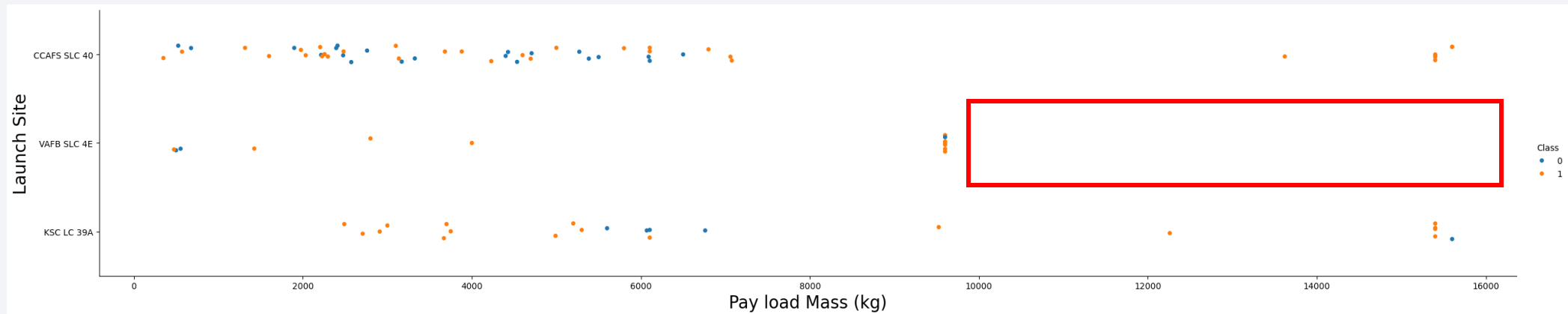
```
In [16]: df['Class'] = landing_class
          df[['Class']].head(8)
```

EDA with Data Visualization (I)

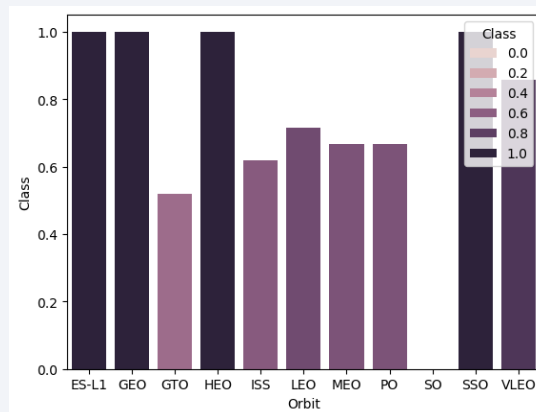
We conducted data exploration by visualizing several relationships:

- The correlation between flight number and launch site.
- The connection between payload and launch site.
- The success rates across different orbit types.
- The relationship between flight number and orbit type.
- The annual trend in launch success rates.

EDA with Data Visualization (II)

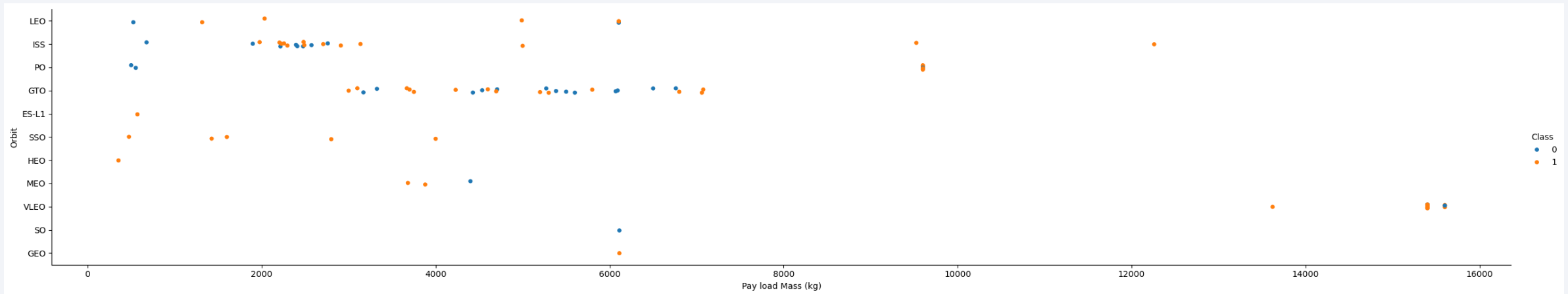


For the VAFB-SLC launchsite there are no rockets launched for heavypayload mass.

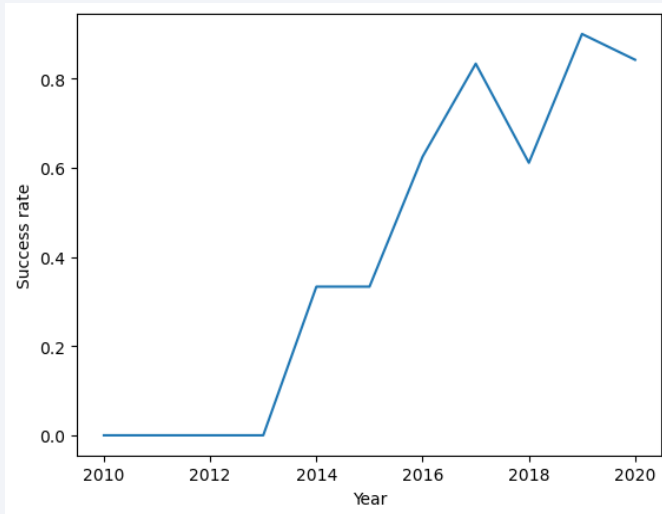


The orbits ES-L1, GEO, HEO, SSO and VLEO have the highest success rates.

EDA with Data Visualization (III)



With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.



The success rate since 2013 kept increasing until 2020.

- The link for this notebook is <https://github.com/fran-carrillo/spaceY/blob/main/edadataviz.ipynb>

EDA with SQL

- We imported the SpaceX dataset directly into a PostgreSQL database within the Jupyter Notebook environment.
- Using SQL for exploratory data analysis, we executed queries to uncover key insights such as:
 - Identifying the unique launch sites involved in the space missions.
 - Calculating the total payload mass carried by boosters launched by NASA (CRS).
 - Determining the average payload mass carried by booster version F9 v1.1.
 - Tabulating the total number of successful and failed mission outcomes.
 - Extracting details on failed landing outcomes on drone ships, including their booster version and launch site names.
- The link for this notebook is https://github.com/fran-carrillo/spaceY/blob/main/jupyter-labs-eda-sql-coursera_sqlite.ipynb

Build an Interactive Map with Folium

We annotated all launch sites and incorporated map elements such as markers, circles, and lines on a Folium map to indicate the success or failure of launches at each site.

- Launch outcomes (failure or success) were categorized into classes: 0 for failure and 1 for success.
- Using color-coded marker clusters, we identified launch sites with notably high success rates.
- We also computed distances from each launch site to nearby features and addressed questions such as:
 - Proximity of launch sites to railways, highways, and coastlines.
 - Whether launch sites maintain a specified distance from urban areas.
- The link to this notebook is https://github.com/fran-carrillo/spaceY/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We developed an interactive dashboard using Plotly Dash.
- In the dashboard, we included pie charts depicting the total launches from specific sites.
- Additionally, we created scatter plots to visualize the relationship between launch outcomes and payload mass (in kilograms) across various booster versions.
- The link to this notebook is [https://github.com/fran-carrillo/spaceY/blob/main/spacex_dash_app%20\(1\).py](https://github.com/fran-carrillo/spaceY/blob/main/spacex_dash_app%20(1).py)

Predictive Analysis (Classification)

- We loaded and transformed the data using numpy and pandas, then divided it into training and testing sets.
- Next, we constructed multiple machine learning models and fine-tuned their hyperparameters using GridSearchCV.
- Our evaluation metric was accuracy, and we enhanced the model through feature engineering and algorithm refinement.
- Ultimately, we identified the best-performing classification model.
- The link to this notebook is https://github.com/fran-carrillo/spaceY/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

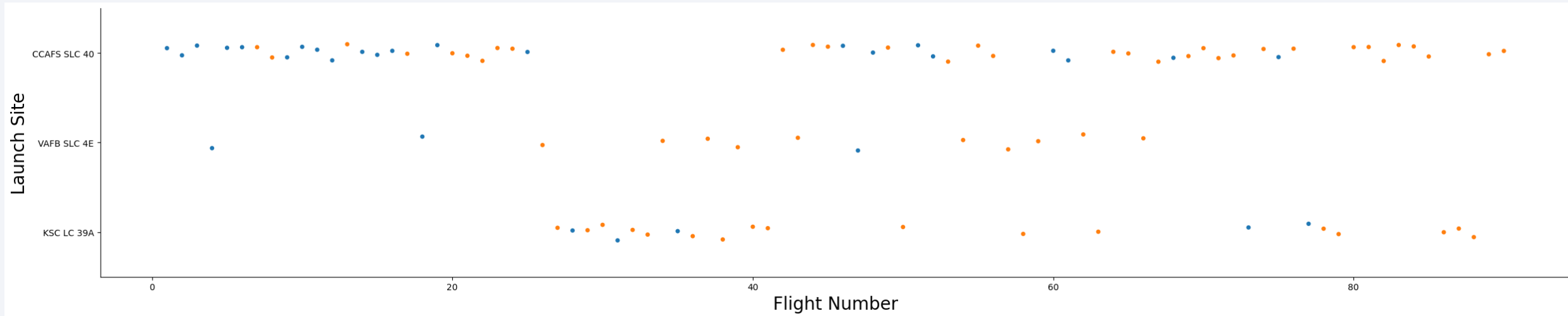
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

The background of the slide is an abstract composition of numerous thin, overlapping lines and streaks in shades of blue and red. These lines are oriented diagonally, creating a sense of motion and depth. The lines vary in opacity and thickness, with some appearing as sharp, bright streaks and others as more diffuse, textured bands. The overall effect is a dynamic, high-tech aesthetic.

Section 2

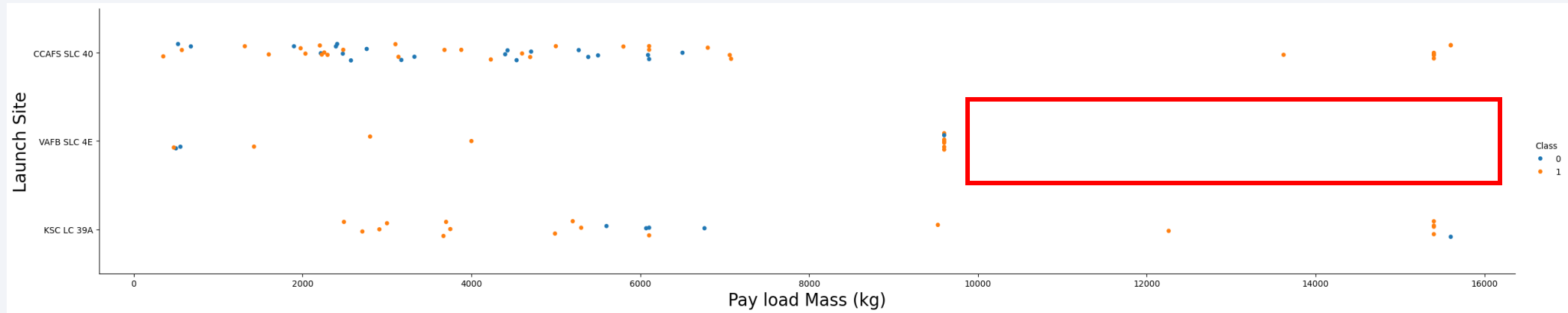
Insights drawn from EDA

Flight Number vs. Launch Site



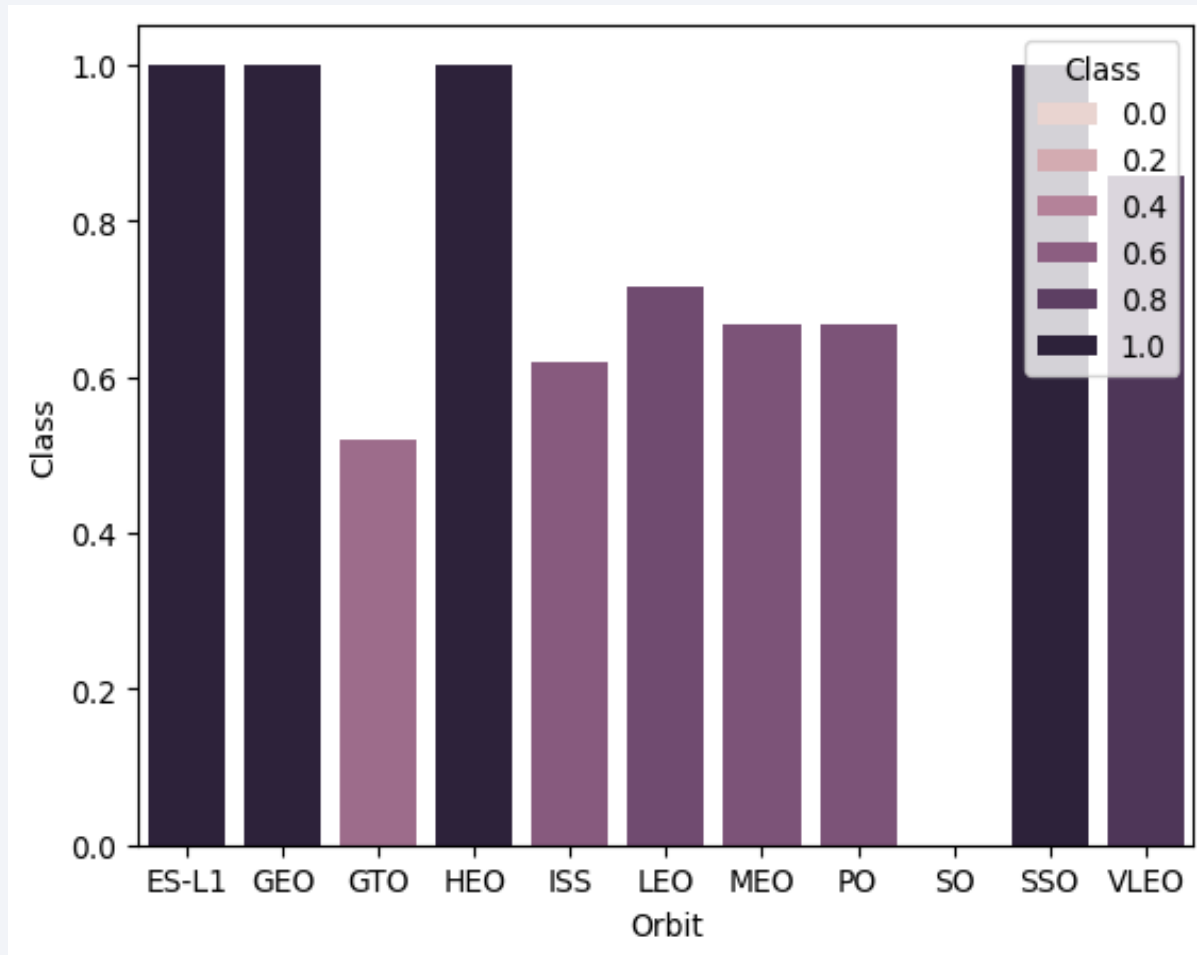
We observed a positive correlation between the number of flights at a launch site and its success rate.

Payload vs. Launch Site



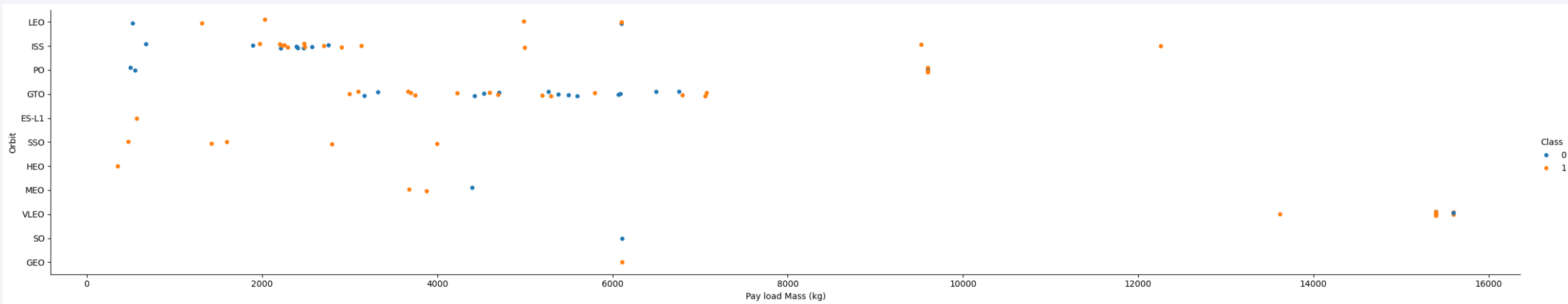
For the VAFB-SLC launchsite there are no rockets launched for heavypayload mass.

Success Rate vs. Orbit Type



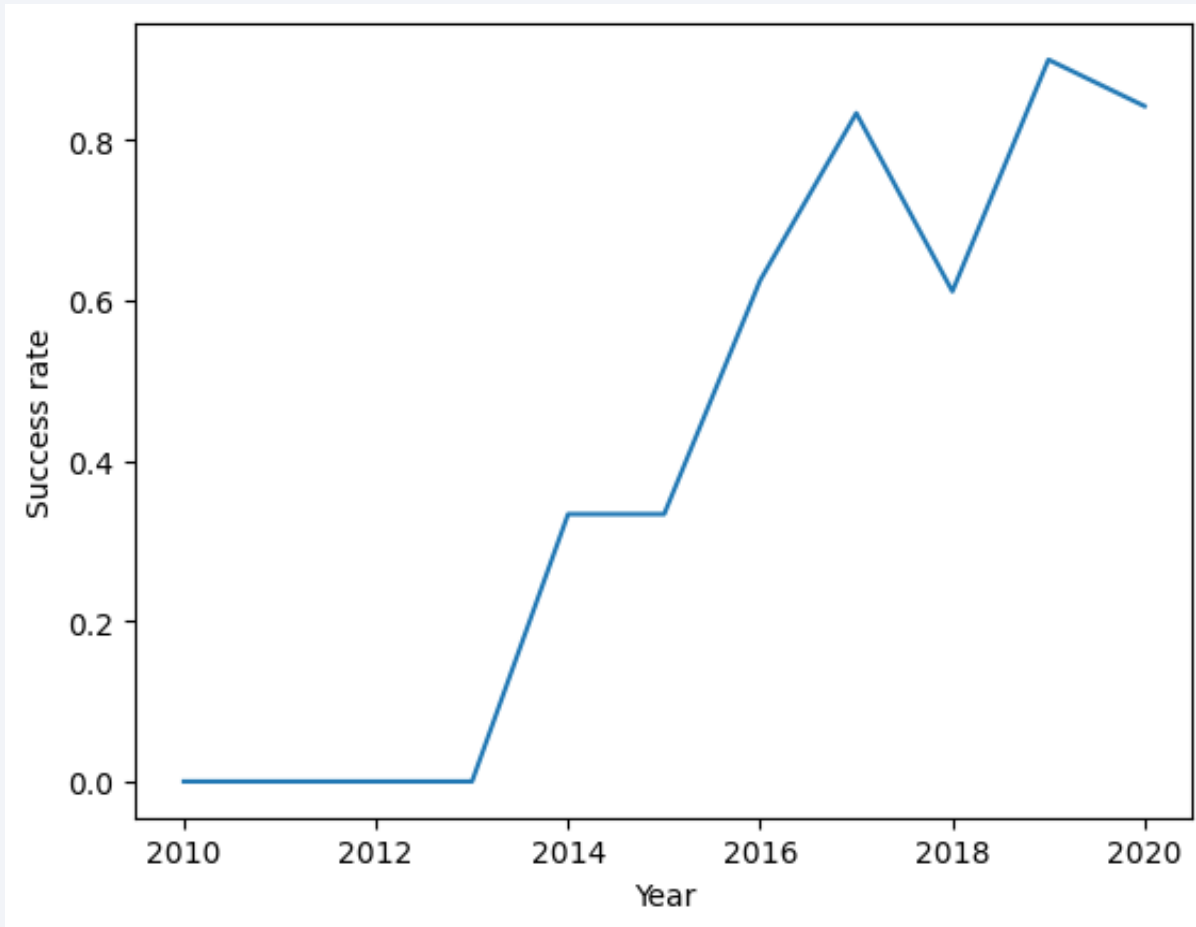
The orbits ES-L1, GEO, HEO, SSO and VLEO have the highest success rates.

Payload vs. Orbit Type



- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend



The success rate since 2013 kept increasing until 2020.

All Launch Site Names

```
In [12]: %sql select DISTINCT Launch_Site from SPACEXTABLE
```

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

```
Out[12]: Launch_Site
```

```
CCAFS LC-40
```

```
VAFB SLC-4E
```

```
KSC LC-39A
```

```
CCAFS SLC-40
```

We utilized the keyword DISTINCT to display only unique launch sites from the SpaceX data.

Launch Site Names Begin with 'CCA'

```
In [23]: %sql SELECT * FROM SPACEXTABLE WHERE Launch_Site LIKE 'CCA%' 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_Outcon |
|------------|------------|-----------------|-------------|---|------------------|-----------|-----------------|-----------------|-------------------|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachut |
| 2010-12-08 | 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 (parachut |
| 2012-05-22 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attem |
| 2012-10-08 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attem |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attem |

We used the query above to retrieve 5 records where launch sites start with CCA.

Total Payload Mass

```
In [16]: %sql SELECT SUM(PAYLOAD_MASS_KG_) FROM SPACEXTABLE WHERE Customer='NASA (CRS)'  
  
* sqlite:///my_data1.db  
Done.  
Out[16]: SUM(PAYLOAD_MASS_KG_)  
         45596
```

We determined that the total payload carried by boosters from NASA was 45,596 units using the query below.

Average Payload Mass by F9 v1.1

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

In [18]: %sql SELECT AVG(PAYLOAD_MASS_KG_) FROM SPACEXTABLE WHERE Booster_Version='F9 v1.1'

* sqlite:///my_data1.db
Done.

Out[18]: AVG(PAYLOAD_MASS_KG_)
          2928.4
```

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4.

First Successful Ground Landing Date

```
In [21]: %sql SELECT MIN(Date) FROM SPACEXTABLE WHERE Landing_Outcome='Success (ground pad)'  
* sqlite:///my_data1.db  
Done.  
Out[21]: MIN(Date)  
2015-12-22
```

We observed that the dates of the first successful landing outcome on ground pad was Dec 22, 2015.

Successful Drone Ship Landing with Payload between 4000 and 6000

In [26]:

```
%sql SELECT * FROM SPACEXTABLE WHERE Landing_Outcome='Success (drone ship)' AND PAYLOAD_MASS_KG_>4000 AND PAYLOAD_MASS_KG_<6000
```

* sqlite:///my_data1.db
Done.

Out[26]:

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS_KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|------------|------------|-----------------|-------------|-----------------------|------------------|-------|------------------------|-----------------|----------------------|
| 2016-05-06 | 5:21:00 | F9 FT B1022 | CCAFS LC-40 | JCSAT-14 | 4696 | GTO | SKY Perfect JSAT Group | Success | Success (drone ship) |
| 2016-08-14 | 5:26:00 | F9 FT B1026 | CCAFS LC-40 | JCSAT-16 | 4600 | GTO | SKY Perfect JSAT Group | Success | Success (drone ship) |
| 2017-03-30 | 22:27:00 | F9 FT B1021.2 | KSC LC-39A | SES-10 | 5300 | GTO | SES | Success | Success (drone ship) |
| 2017-10-11 | 22:53:00 | F9 FT B1031.2 | KSC LC-39A | SES-11 / EchoStar 105 | 5200 | GTO | SES EchoStar | Success | Success (drone ship) |

We used the WHERE clause to filter for boosters that successfully landed on a drone ship. Additionally, we applied an AND condition to identify cases where the landing was successful and the payload mass was between 4000 and 6000 units.

Total Number of Successful and Failure Mission Outcomes

```
In [40]: %sql SELECT COUNT(Mission_Outcome) FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Success%'
* sqlite:///my_data1.db
Done.

Out[40]: COUNT(Mission_Outcome)
          100

In [41]: %sql SELECT COUNT(Mission_Outcome) FROM SPACEXTABLE WHERE Mission_Outcome LIKE 'Failure%'
* sqlite:///my_data1.db
Done.

Out[41]: COUNT(Mission_Outcome)
          1
```

Successful missions: 100

Failure mission outcomes: 1

We used '%' to filter for **WHERE** MissionOutcome was a success or a failure.

Boosters Carried Maximum Payload

```
In [48]: %sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS_KG_ = (SELECT MAX(PAYLOAD_MASS_KG_) FROM SPACEXTABLE)
```

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

```
Done.
```

```
Out[48]: 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
```

We used a subquery in the **WHERE** clause and the **MAX()** function.

2015 Launch Records

```
In [49]: %sql select substr(Date, 6,2), Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr(Date,0,5)='2015'
```

* sqlite:///my_data1.db
Done.

```
Out[49]:
```

| | substr(Date, 6,2) | Landing_Outcome | Booster_Version | Launch_Site |
|--|-------------------|----------------------|-----------------|-------------|
| | 01 | Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| | 04 | Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |

- We used a combination of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes on drone ships, including their booster versions and launch site names, specifically for the year 2015.

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

```
[11]: landing_Outcome, COUNT(Landing_Outcome) AS 'NUMBER OF LANDING OUTCOMES' FROM SPACEXTABLE WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DE
```

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

```
Done.
```

```
[11]:
```

| Landing_Outcome | NUMBER OF LANDING OUTCOMES |
|------------------------|----------------------------|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

| Landing_Outcome | NUMBER OF LANDING OUTCOMES |
|------------------------|----------------------------|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

We selected landing outcomes and the count of landing outcomes from the data, filtering for landing outcomes between June 4, 2010, and March 20, 2010, using the WHERE clause.

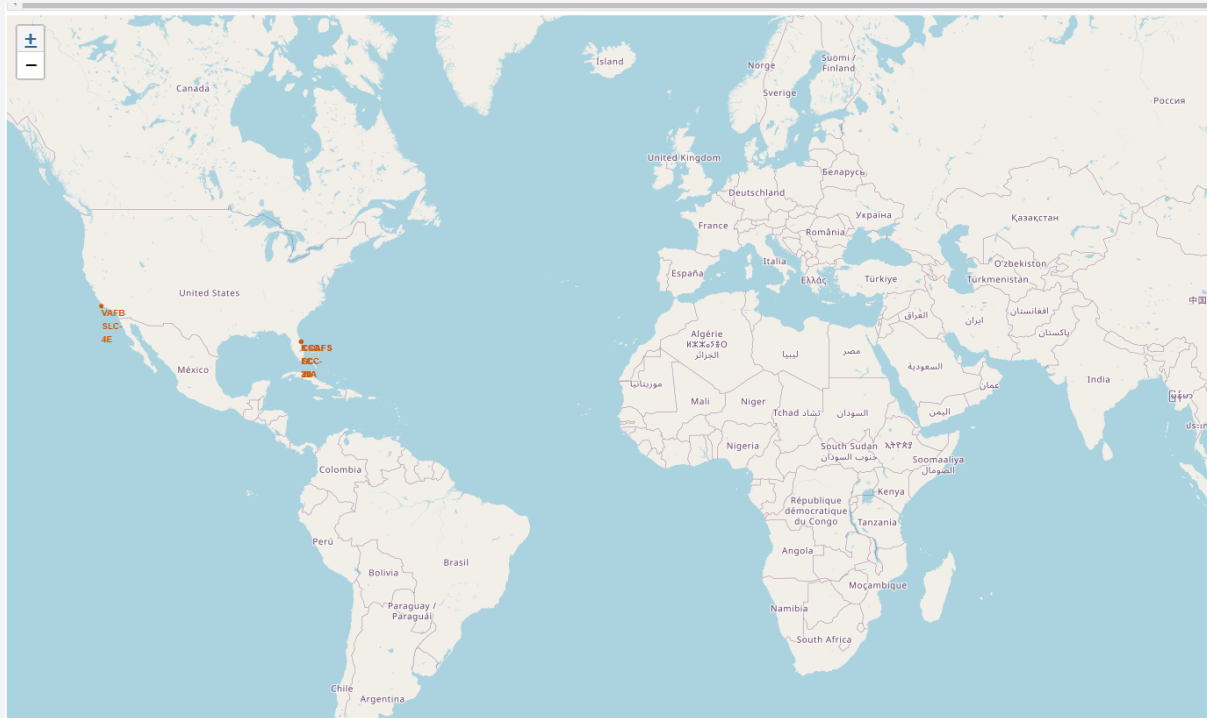
We then used the GROUP BY clause to group the landing outcomes and the ORDER BY clause to sort the grouped landing outcomes in descending order.

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a solid blue background on the left and a satellite photograph of Earth on the right. The Earth's surface is dark, with numerous bright yellow and orange lights representing cities and urban areas. The horizon of the Earth is visible as a curved line separating the dark surface from the deep blue of space.

Section 3

Launch Sites Proximities Analysis

Global map of all launch sites

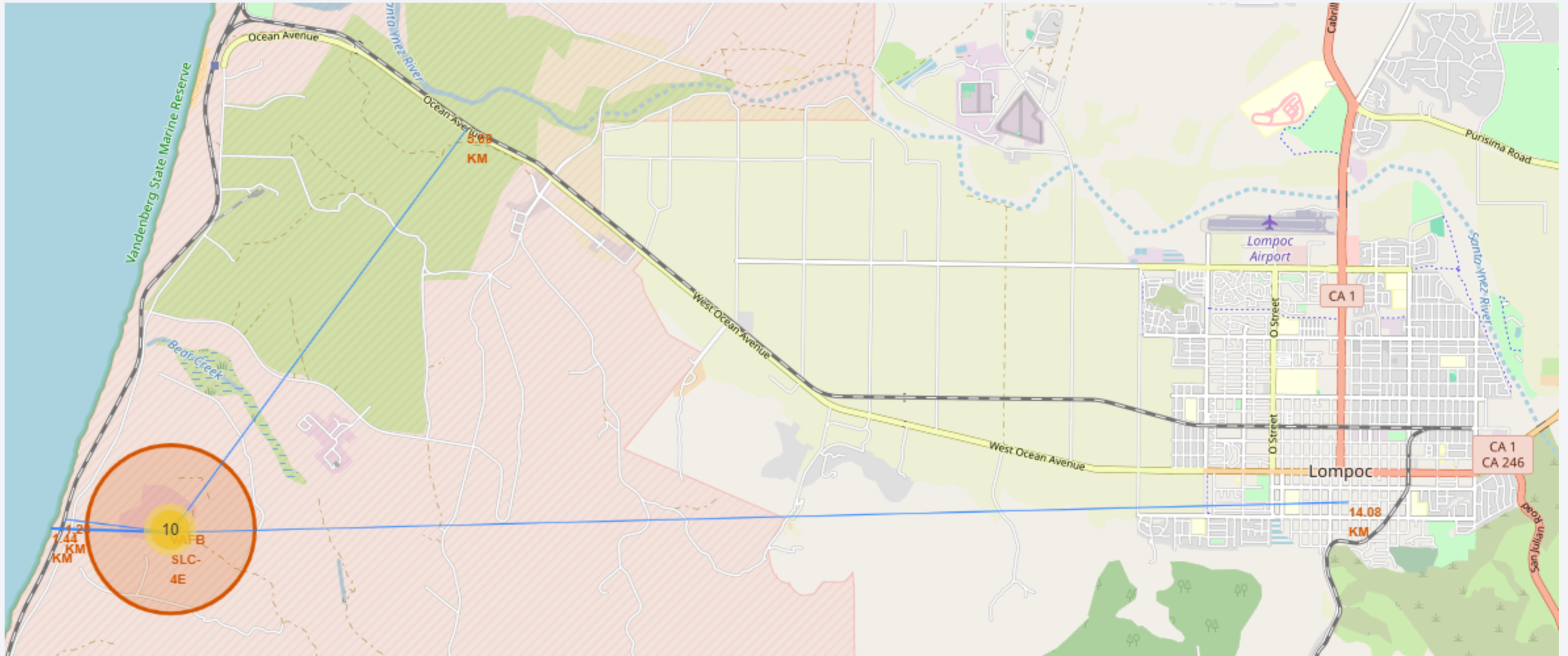


Space X launch sites
are either in the west
coast of the US
(California) or
east coast (Florida)

Successful (green) and failure (red) launches



<Folium Map Screenshot 3>



- All launch sites are far away from railways, highways and cities, and keep close to the coastline

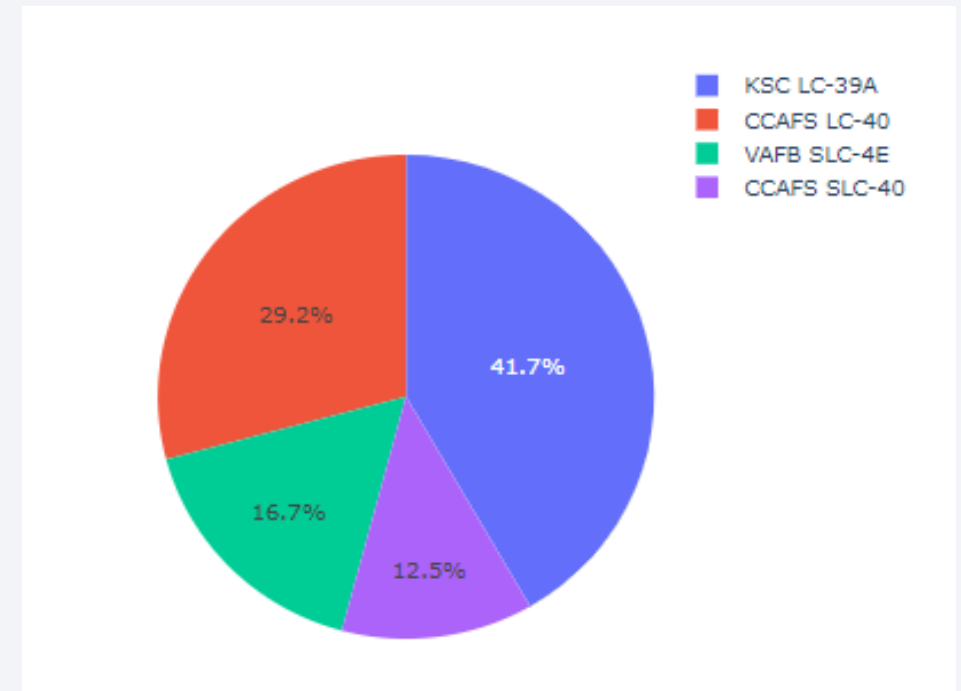


Section 4

Build a Dashboard with Plotly Dash

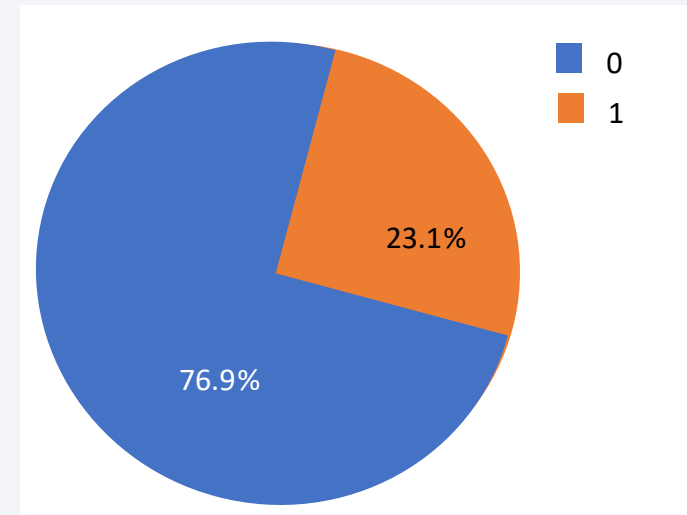
NASA's Kennedy Space Center has the highest Launch success rate

- NASA's **Kennedy Space Center (KSC LC-39A)** in Merritt Island, Florida.
- East coast launch sites at Florida have higher Launch success rate than VAFB at Santa Barbara, CA.



NASA's Kennedy Space Center has the highest Launch success rate

- KSC LC-39A has a 77% success rate and a 23% of failure rate



Scatter plot of Payload vs Launch Outcome for all sites



Higher success rates for low weighted payloads (left circle) than heavy wieghted payloads (right circle)

Section 5

Predictive Analysis (Classification)

Classification Accuracy

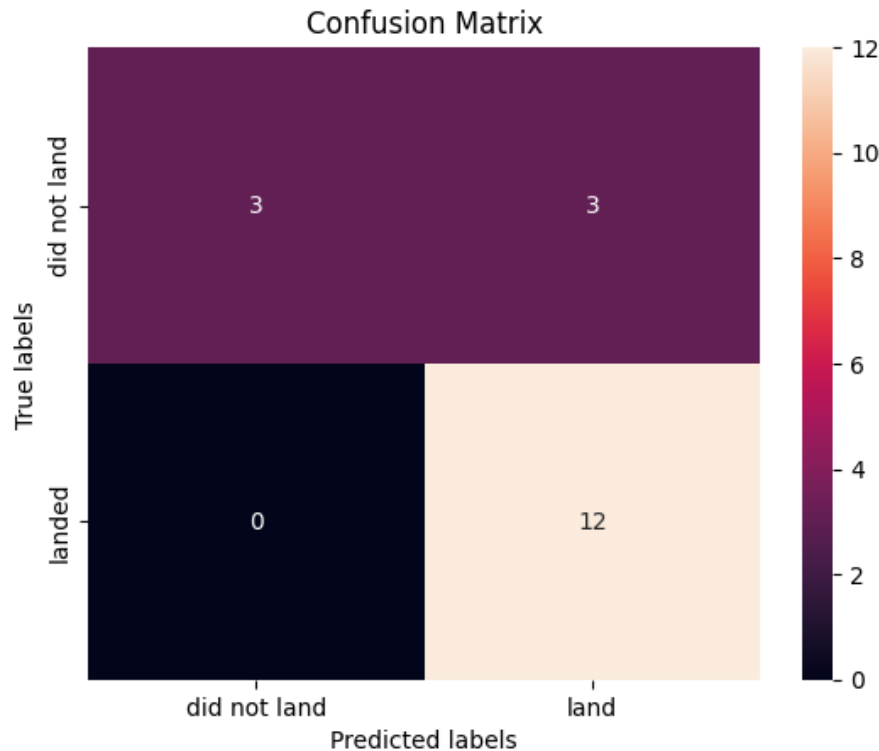
```
In [37]: print('lr accuracy =', '{:.2%}'.format(logreg_cv.best_score_))  
         print('svm accuracy =', '{:.2%}'.format(svm_cv.best_score_))  
         print('tree accuracy =', '{:.2%}'.format(tree_cv.best_score_))  
         print('knn accuracy =', '{:.2%}'.format(knn_cv.best_score_))
```

```
lr accuracy = 84.64%  
svm accuracy = 84.82%  
tree accuracy = 87.14%  
knn accuracy = 84.82%
```

The decision tree classifier is the model that has the highest classification accuracy.

Confusion Matrix

```
In [27]: yhat = tree_cv.predict(X_test)
         plot_confusion_matrix(Y_test,yhat)
```



The confusion matrix of the decision tree classifier reveals its ability to distinguish between different classes. The primary issue lies in false positives, where unsuccessful landings are incorrectly classified as successful.

Conclusions

Based on our analysis, we can conclude the following:

- There is a positive correlation between the number of flights at a launch site and its success rate.
- The launch success rate has shown a steady increase from 2013 to 2020.
- Orbits ES-L1, GEO, HEO, SSO, and VLEO exhibited the highest success rates.
- KSC LC-39A had the highest number of successful launches among all sites.
- The decision tree classifier stands out as the optimal machine learning algorithm for this specific task.

Thank you!

