Navigating the Cosmos with Data Science

Kaushal Kshirsagar

Outline

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

Executive Summary

An overview of the techniques

- Gathering data through web scraping
- Organizing data; conducting exploratory data analysis using SQL Exploratory
- Analysing data using data visualization
- Folium for interactive visual analytics
- Prediction using Machine Learning

Synopsis of all outcomes –

- Interactive analytics for exploratory data analysis
- In screenshots The Machine Learning Lab's predictive analytics results

INTRODUCTION

SpaceX has transformed the space industry by offering rocket launches, specifically with the Falcon 9, for as low as $62 million per launch—compared to $165 million or more from other providers. Much of this cost reduction is due to SpaceX's innovative approach of reusing the first stage of the rocket by safely landing it back to be used in future missions. This process allows for further price reductions over time. As a data scientist for a startup competing with SpaceX, your task is to develop a machine learning pipeline to predict the first stage's landing outcome in future launches. This project is vital in setting competitive prices for launches. Key objectives include:

- Identifying all factors that impact landing outcomes
- Understanding the relationships between variables and their effects on landing success
- Determining optimal conditions to maximize the likelihood of successful landings

Methodology

Data Collection Methodology:
Data was gathered using the SpaceX REST API along with web scraping from Wikipedia.

Data Wrangling:
Preprocessing steps were applied to clean and prepare the data.

Feature Engineering:
Categorical features were transformed using one-hot encoding.

Exploratory Data Analysis (EDA):
Data exploration was conducted through visualizations and SQL queries.

Interactive Visual Analytics:
Interactive visualizations were created using tools like Folium and Plotly Dash.

Predictive Analysis with Classification Models:
Classification models were built, tuned, and evaluated for predictive analysis.

Data Collection

Data collection involves gathering and measuring information on specific variables within a defined system, enabling us to answer relevant questions and assess outcomes. In this case, the dataset was compiled through a combination of REST API and web scraping from Wikipedia.

For the REST API method, we started by sending a GET request. The response was then decoded into JSON format, which was converted into a pandas DataFrame using json_normalize(). Afterward, we cleaned the data, checked for any missing values, and filled them as needed.

For the web scraping process, BeautifulSoup was used to retrieve launch records in an HTML table format. The table was parsed and converted into a pandas DataFrame for further analysis.

Data Collection –SpaceX API

A GET request was used to retrieve rocket launch data via an API. The JSON results were then converted into a DataFrame using the json_normalize method. Following this, data cleaning was performed, and missing values were filled in as required.

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

```
response = requests.get(spacex_url)
```

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

```
# Lets take a subset of our dataframe keeping only the features we want a
nd the flight number, and date_utc.
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number',
'date_utc']]

# We will remove rows with multiple cores because those are falcon rocket
s with 2 extra rocket boosters and rows that have multiple payloads in a
single rocket.
data = data[data['cores'].map(len)==1]
data = data[data['payloads'].map(len)==1]

# Since payloads and cores are lists of size 1 we will also extract the s
ingle value in the list and replace the feature.
data['cores'] = data['cores'].map(lambda x : x[0])
data['payloads'] = data['payloads'].map(lambda x : x[0])

# We also want to convert the date_utc to a datetime datatype and then ex
tracting the date leaving the time
data['date'] = pd.to_datetime(data['date_utc']).dt.date

# Using the date we will restrict the dates of the launches
data = data[data['date'] <= datetime.date(2020, 11, 13)]
```

Data Collection –Scraping

Accessed the Falcon9 Launch Wikipedia page using its URL, created a BeautifulSoup object from the HTML response, and extracted all column or variable names from the HTML header

```
# use requests.get() method with the provided static_url
# assign the response to a object
data = requests.get(static_url).text
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response te
xt content
soup = BeautifulSoup(data,'html.parser')
```

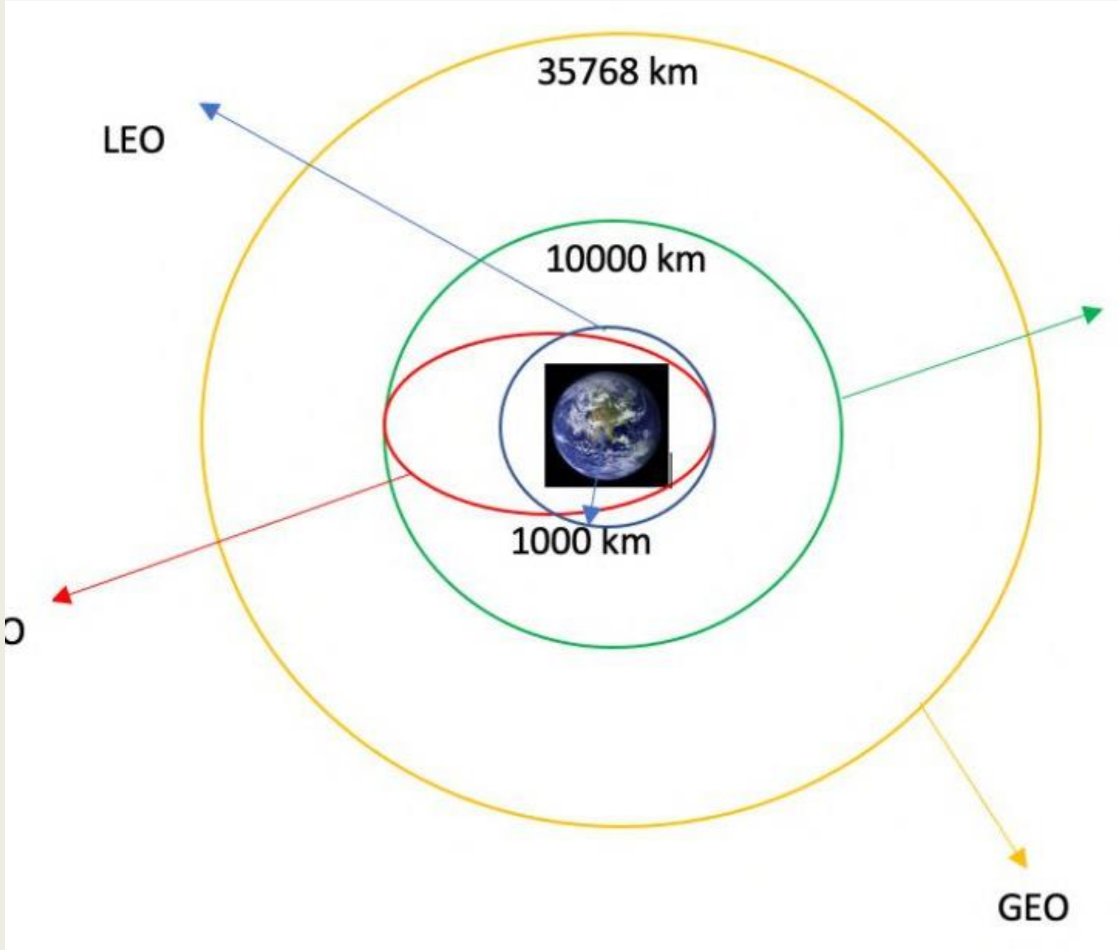
```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plai
nrowheaders collapsible")):
    # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding t
o launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
```

Data Wrangling

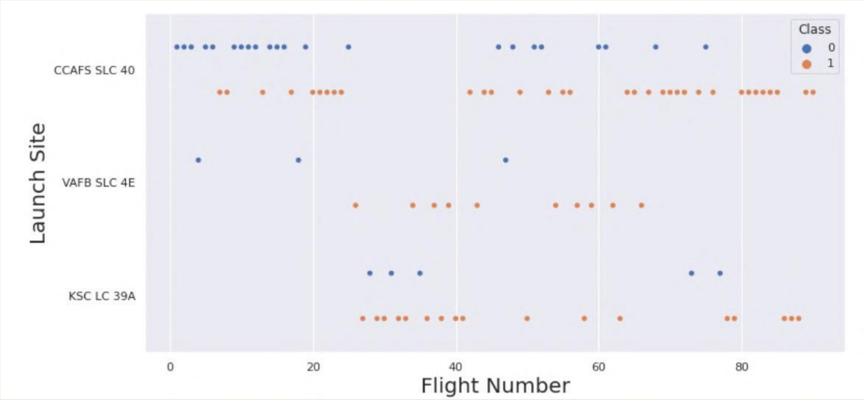
- Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).

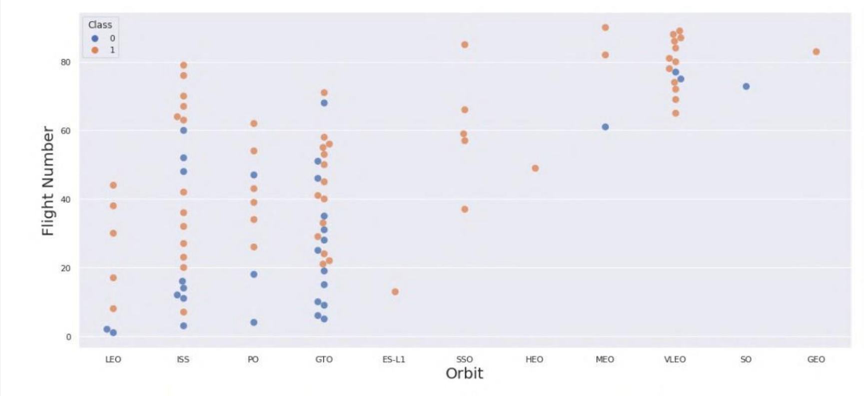
- We will first calculate the number of launches on each site, then calculate the number and occurrence of mission outcome per orbit type.

- We then create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Lastly, we will export the result to a CSV.



EDA with data Visualization

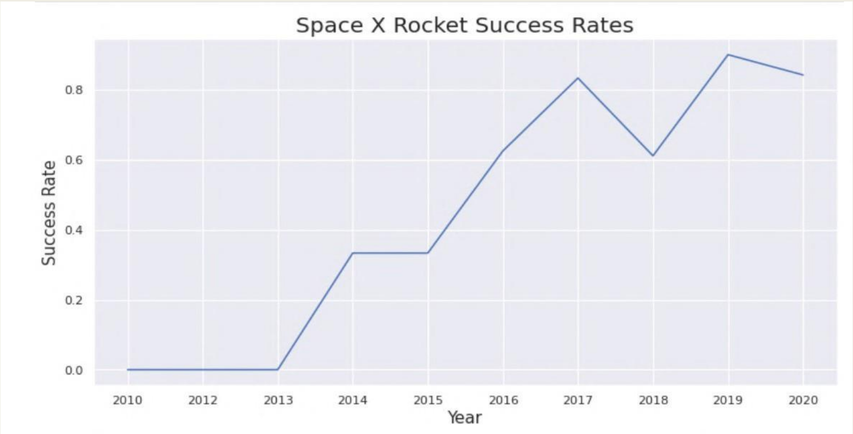


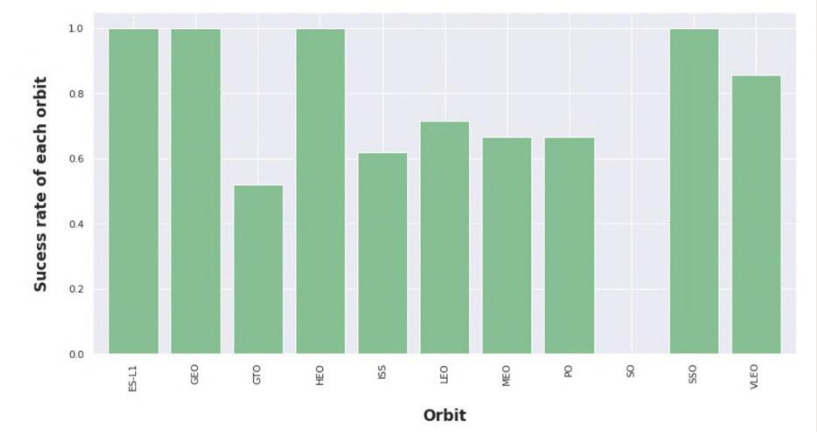


We first started by using scatter graph to find the relationship between the attributes such as between:

- Payload and Flight Number.
- Flight Number and Launch Site.
- Payload and Launch Site.
- Flight Number and Orbit Type.
- Payload and Orbit Type.

EDA with Data Visualization





- After initially examining the relationships using a scatter plot, we’ll proceed with additional visualization tools like bar and line graphs for deeper analysis. Bar graphs provide an easy way to interpret relationships between attributes, and here, we’ll use them to identify which orbits show the highest probability of success. Following this, a line graph will be used to display trends or patterns over time, specifically to observe the yearly trend in launch success. Finally, we’ll apply feature engineering by creating dummy variables for categorical columns, which will support success prediction in future modules.

EDA with SQL

Using SQL, we conducted a variety of queries to gain insights into the dataset, including the following examples:

- Displaying the names of launch sites.
- Showing five records where the launch site names start with "CCA."
- Calculating the total payload mass carried by boosters launched by NASA (CRS).
- Finding the average payload mass carried by the booster version F9 v1.1.
- Listing the date of the first successful landing outcome on a ground pad.
- Identifying booster names that achieved a successful drone ship landing and carried a payload mass between 4,000 and 6,000.
- Counting the total number of successful and failed mission outcomes.
- Listing booster versions that carried the maximum payload mass.
- Displaying failed drone ship landings, along with the booster versions and launch site names, for the year 2015.
- Ranking the count of successful landing outcomes between the dates 2010-06-04 and 2017-03-20, in descending order.

Build an Interactive Map with Folium

To visualize the launch data into an interactive map. We took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.

We then assigned the dataframe launch_outcomes(failure,success) to classes 0 and 1with Red and Green markers on the map in MarkerCluster().

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allowing the user to play around with the data as they need.
- We plotted pie charts showing the total launches by a certain sites.

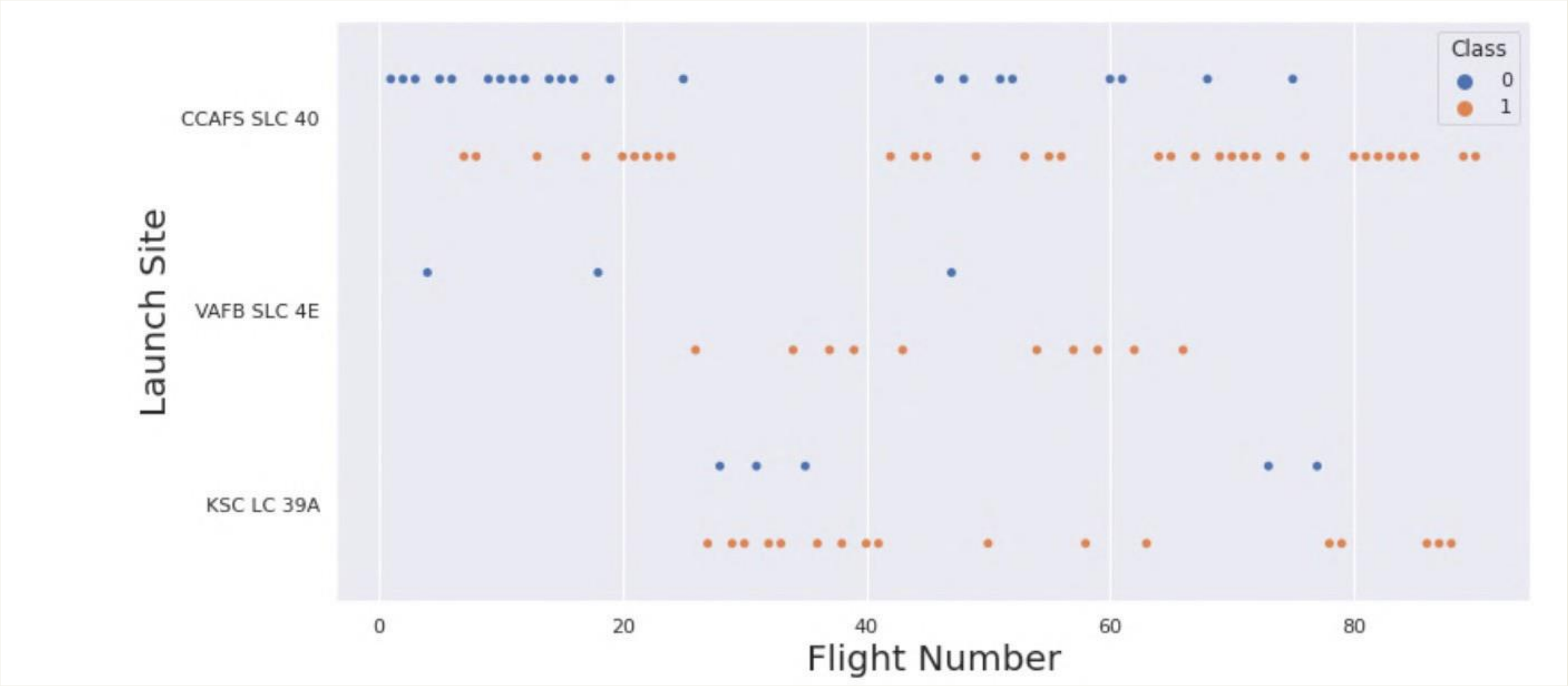
  We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

RESULTS

The results will be categorized to 3 main results which is:

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

Flight Number Vs LaunchSite

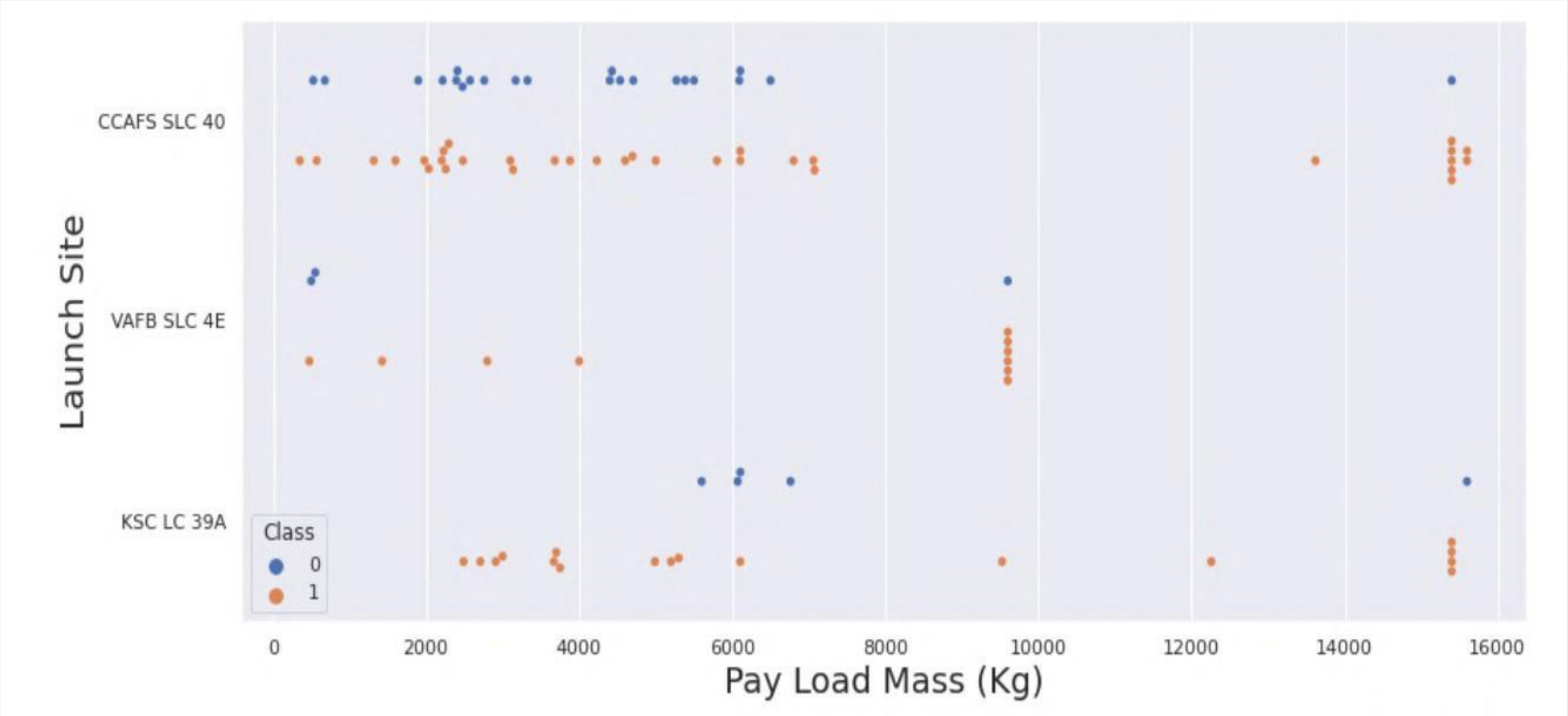


This scatter plot indicates that as the number of flights at a launch site increases, the success rate tends to rise. However, the CCAFS SLC40 site shows the weakest correlation with this pattern.

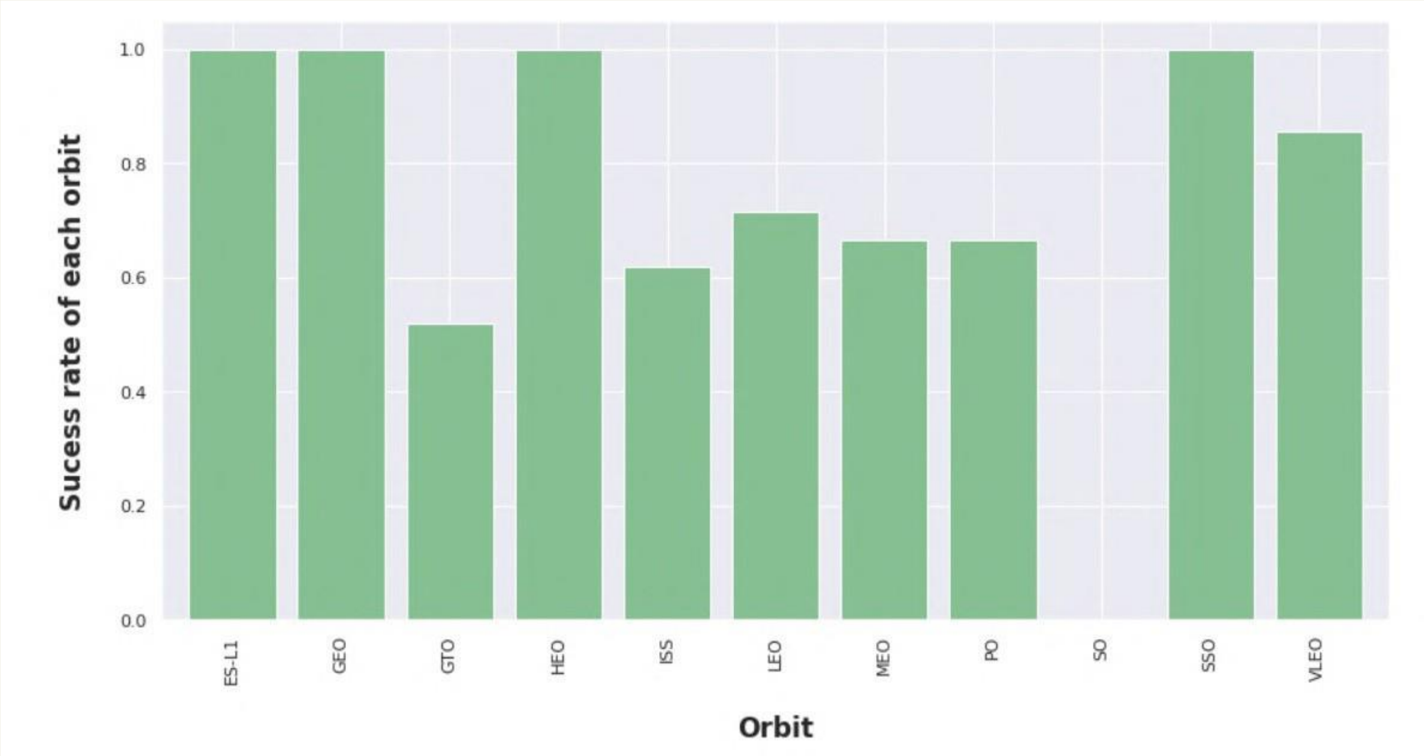
Payload Vs LaunchSite

This scatter plot shows once the pay load mass is greater than 7000kg, the probability of the success rate will be highly increased.

However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.



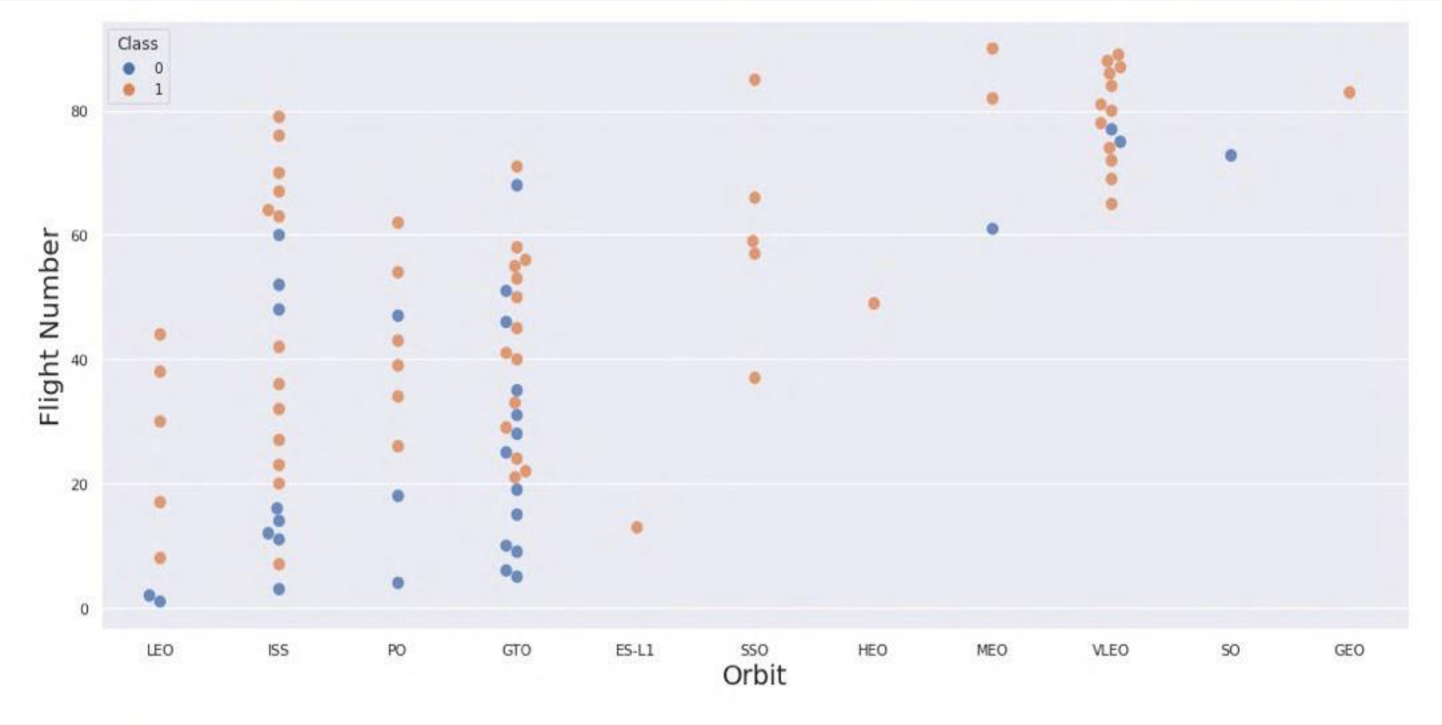
Success rate vs. Orbit Type



This figure illustrates how different orbits can affect landing outcomes, with certain orbits like SSO, HEO, GEO, and ES-L1 showing a 100% success rate, while the SO orbit has a 0% success rate. However, a closer examination reveals that some of these orbits, such as GEO, SO, HEO, and ES-L1, have only one recorded occurrence. This indicates that more data is needed to identify patterns or trends before making any definitive conclusions.

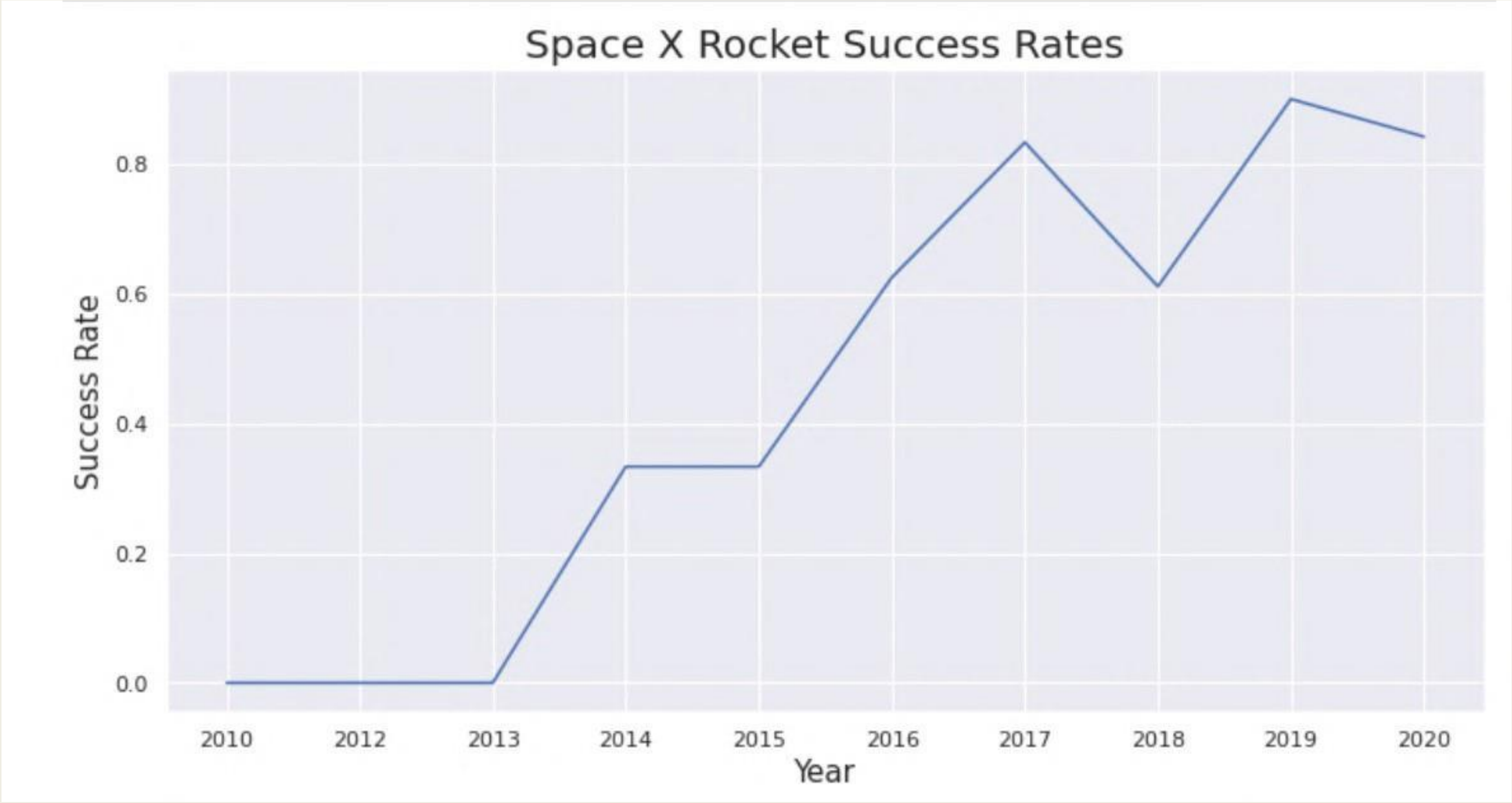
Flight Number vs Orbit Number

The scatter plot indicates that, in general, a higher flight number corresponds to a greater success rate, particularly in LEO orbit. However, this trend does not apply to GTO orbit, where no correlation is observed between the two variables. Additionally, orbits with only a single occurrence should be excluded from this analysis, as they require a larger dataset for more reliable conclusions.



Launch Success Yearly Trend

These figures clearly show an increasing trend from 2013 to 2020. If this trend continues in the coming years, the success rate is expected to steadily rise, potentially reaching a 100% success rate.



All Launch Site Names

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

In [5]:

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[5]:

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

Launch Site Names Begin with 'CCA'

We used the query above to display 5 records where launch sites begin with `CCA`

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

```
In [11]: task_2 = '''
                SELECT *
                FROM SpaceX
                WHERE LaunchSite LIKE 'CCA%'
                LIMIT 5
                '''
         create_pandas_df(task_2, database=conn)
```

Out[11]:

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

We calculated 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)

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Total Payload Mass by NASA (CRS)
45596

Average Payload Mass by F9 v1.1

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

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Average Payload Mass by Booster Version F9 v1.1
2928

First Successful Ground Landing Date

We use the min() function to find the result We observed that the dates of the

first successful landing outcome on ground
pad was 22ndDecember 2015

```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

First Succesful Landing Outcome in Ground Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

We used the WHEREclause to filter for boosters which have successfully landed on drone ship and applied the ANDcondition to determine successful landing with payload mass greater than 4000 but less than 6000

```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.datab
ases.appdomain.cloud:32731/bludb
Done.
```

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

Total Number of Successful and FailureMission Outcomes

We used wildcard like ‘%’ to filter for WHERE MissionOutcomewas a success or a failure.

List the total number of successful and failure mission outcomes

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Success%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Successful Mission
100

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Failure%';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.clou
d:32731/bludb
Done.
```

Failure Mission
1

Boosters Carried MaximumPayload

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEX
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.clou
d:32731/bludb
Done.
```

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

We determined the booster that have carried the maximum payload using a subquery in the WHEREclause and the MAX() function.

2015 Launch Records

We used a combinations of the WHEREclause, LIKE, AND, and BETWEEN conditions to filter for failedlanding outcomes in drone ship, their booster versions, andlaunch site names for year 2015

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.
databases.appdomain.cloud:32731/bludb
Done.
```

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://zpw86771:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

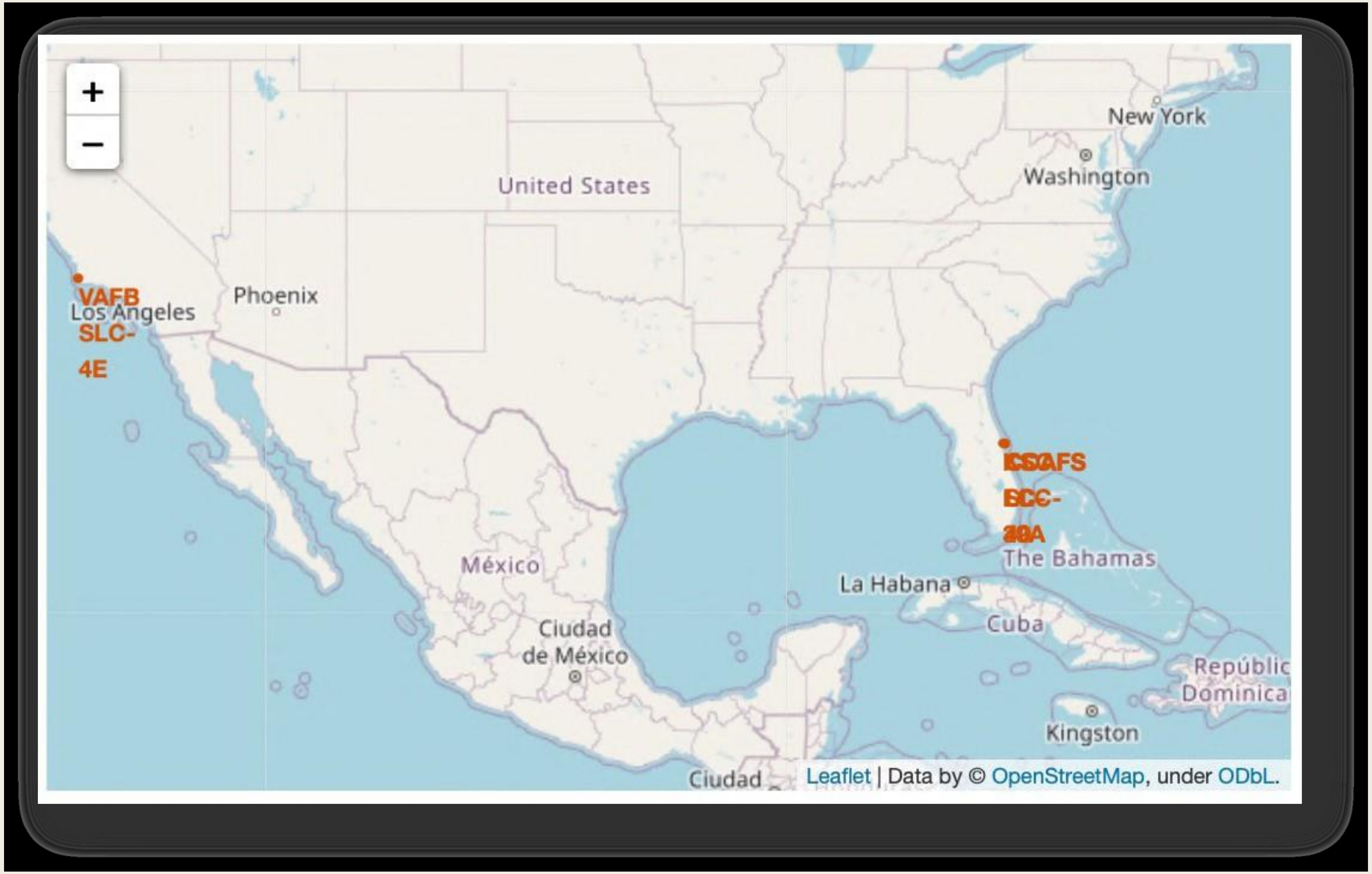
Landing Outcome	Total Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

We selected Landing outcomes and the COUNTof landing outcomes from the data and used the WHEREclause to filter for landing outcomes BETWEEN2010-06-04 to 2010-03-20.

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.

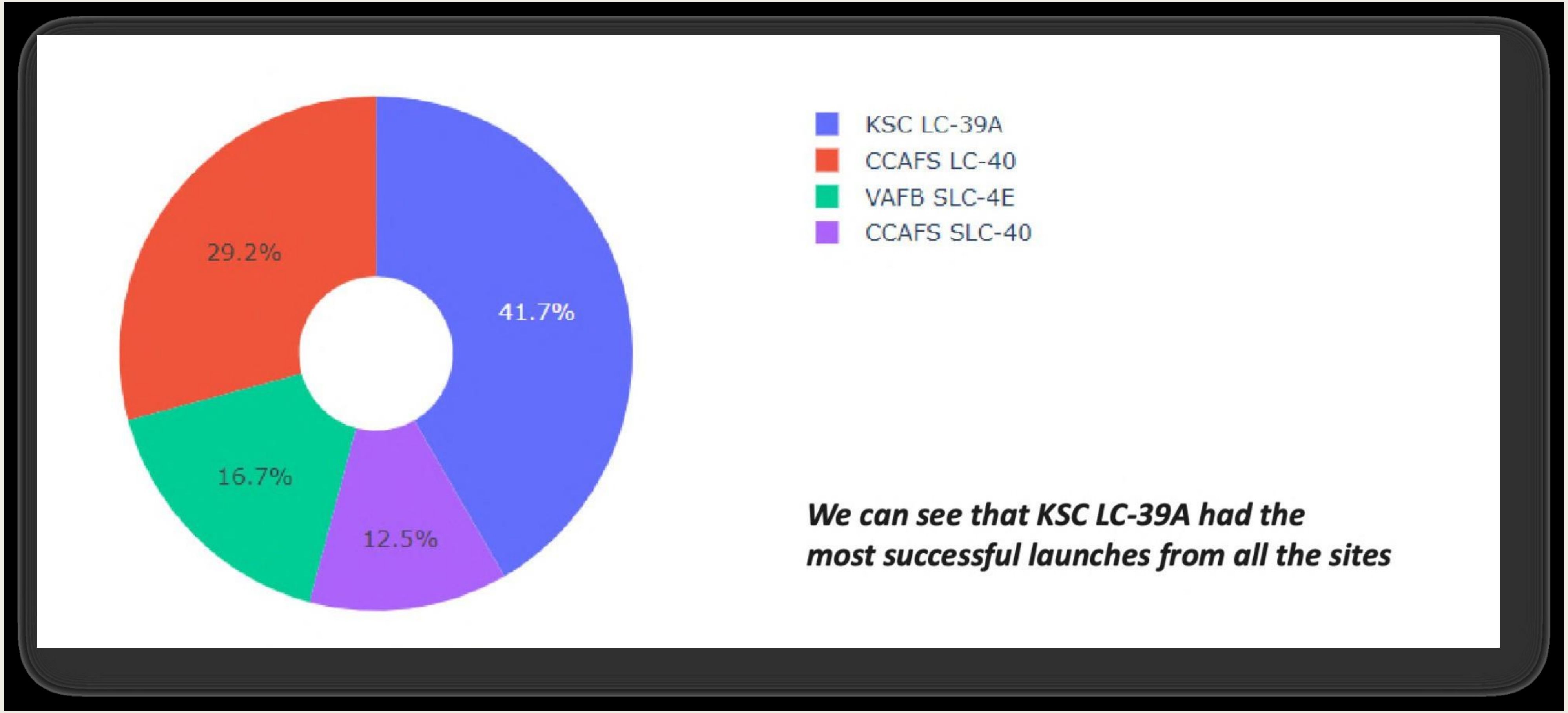
Location of all the Launch Sites



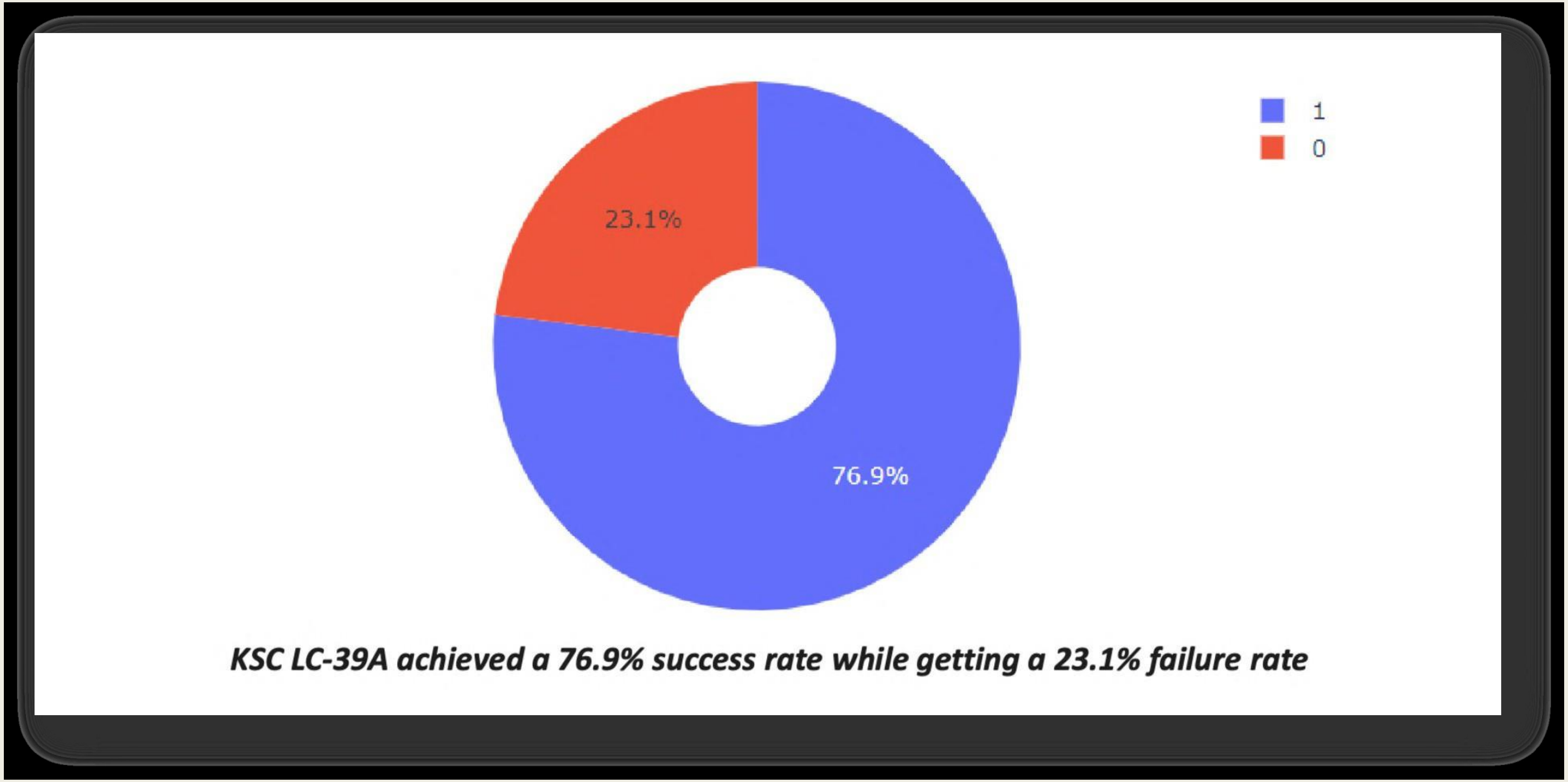
We can see that all the SpaceX launch sites are located inside the United States

Build a Dashboard with Ploty Dash

The success percentage by each sites.

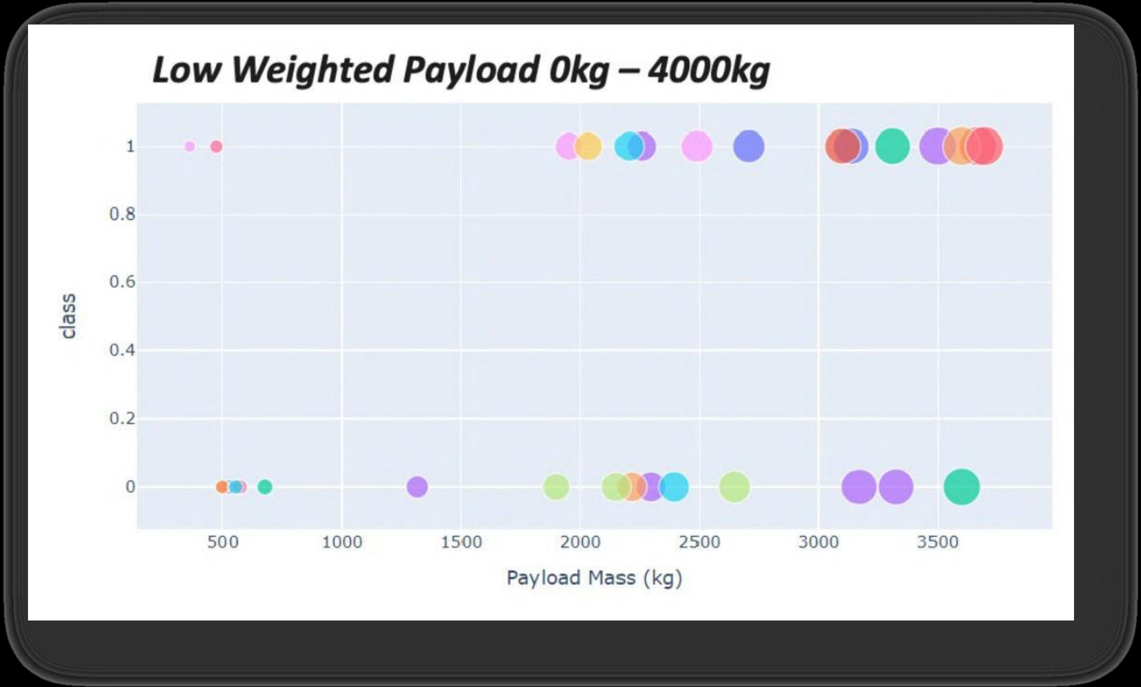


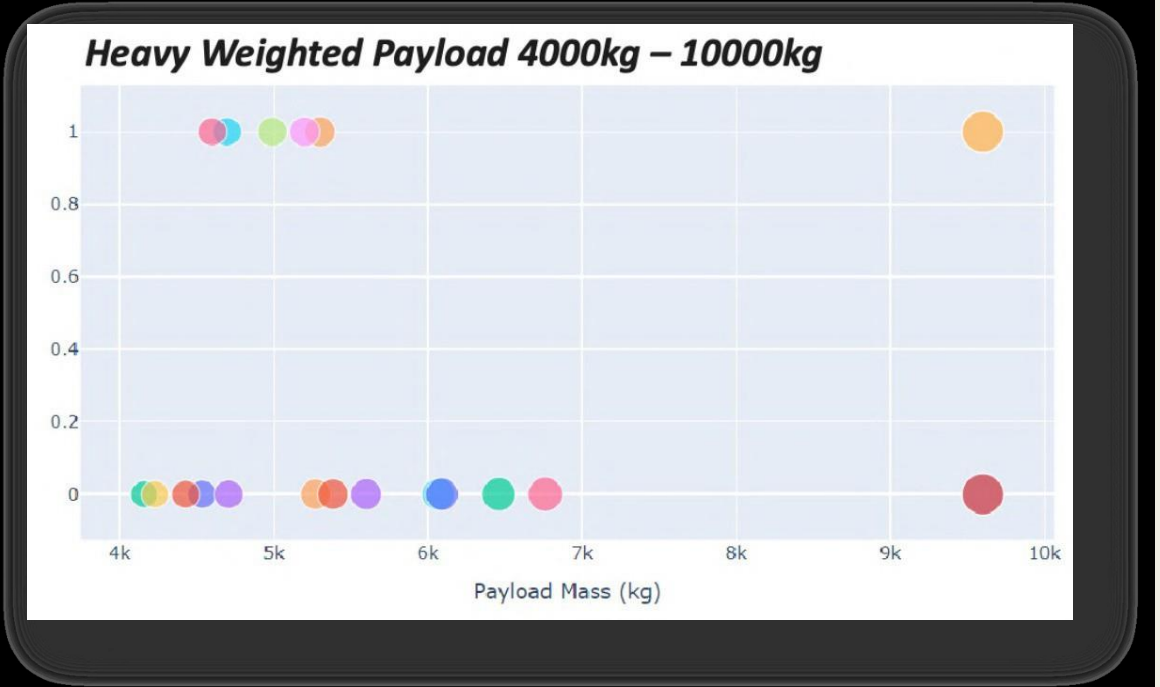
The highest launch-success ratio: KSC LC-39A



Payload vs Launch Outcome Scatter Plot

We can see that all the success rate for low weighted payload is higher than heavy weighted payload





Predictive Analysis (Classification)

Classification Accuracy

As we can see, by using the code as below: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is',bestalgorithm,'with a score of',algorithms[bestalgorithm])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)
```

```
Best Algorithm is Tree with a score of 0.9017857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_sampl
es_split': 10, 'splitter': 'random'}
```

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.