# Space Race Spacex with Data Science

Savitha Pillala

10/23/2023

# OUTLINE

- Executive Summary
- Introduction
- Methodology
- Results
  - Visualization – Charts
  - Dashboard
- Discussion
  - Findings & Implications
- Conclusion
- Appendix

# EXECUTIVE SUMMARY

- SpaceX has earned worldwide acclaim for its remarkable series of historic achievements. In a feat unparalleled by any other private company, SpaceX successfully brought back a spacecraft from low-earth orbit, a momentous feat that they first accomplished in December 2010. On its website, SpaceX promotes Falcon 9 rocket launches at a price of $62 million, which is significantly lower compared to other providers whose costs exceed $165 million per launch. The considerable savings mainly stem from the fact that SpaceX has the ability to reuse the first stage.

- This data analysis research will support in answering question like:
  - Determining the landing of the first stage, we can calculate what is the **launch cost?**
  - Will any alternative company intending to compete with SpaceX in bidding for a rocket launch?
  - How each payload carried during a SpaceX mission into outer space provides meaningful insights?

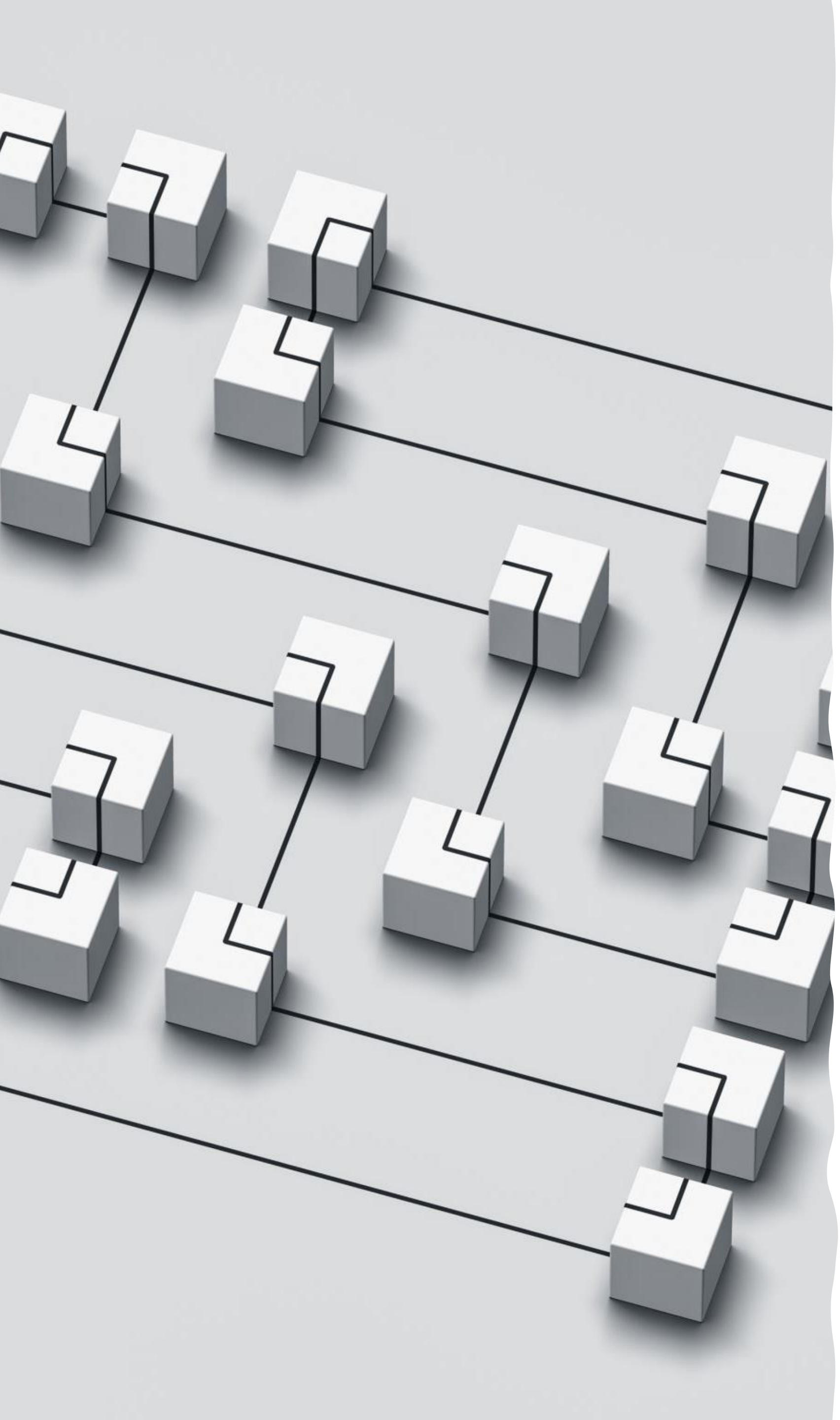
# INTRODUCTION

- The main goal of the project is to utilize data science methodology to define and formulate a Spacex business edge using statistical data analysis, data visualization and machine learning technique to load a dataset, clean it, and find out interesting insights.
- Problems you want to find answers
- What factors determine if the rocket will land successfully?
- The interaction amongst various features that determine the success rate of a successful landing.
- What operating conditions needs to be in place to ensure a successful landing program.

# Methodology

# METHODOLOGY

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with Data Visualization
  - Exploratory Data Analysis with SQL
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive analytics in screenshots
  - Predictive Analytics result

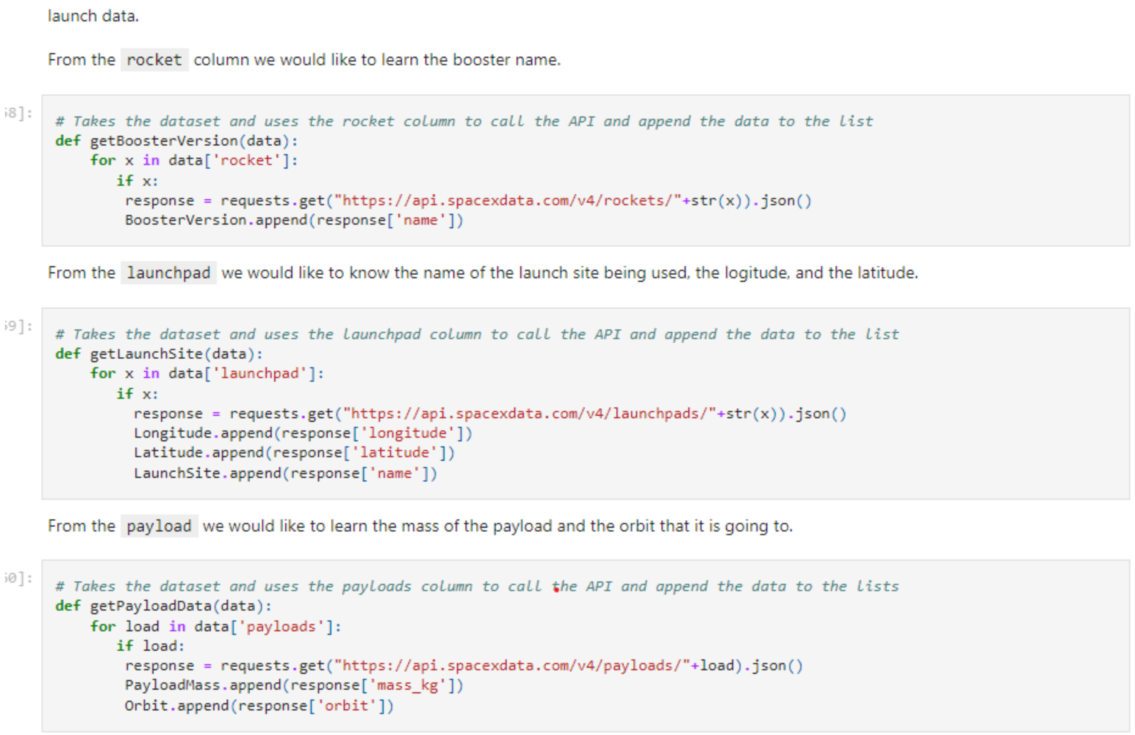


# Data Collection

- The data was collected using various methods
- Data collection was done using get request to the SpaceX API.
- Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
- We then cleaned the data, checked 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 HTML table, parse the table and convert it to a pandas dataframe for future analysis.

# Data Collection through 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/Enlightment2000/IBM-Data-Science-Capstone-SpaceX_Final-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb](https://github.com/Enlightment2000/IBM-Data-Science-Capstone-SpaceX_Final-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb)



launch data.

From the `rocket` column we would like to learn the booster name.

```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

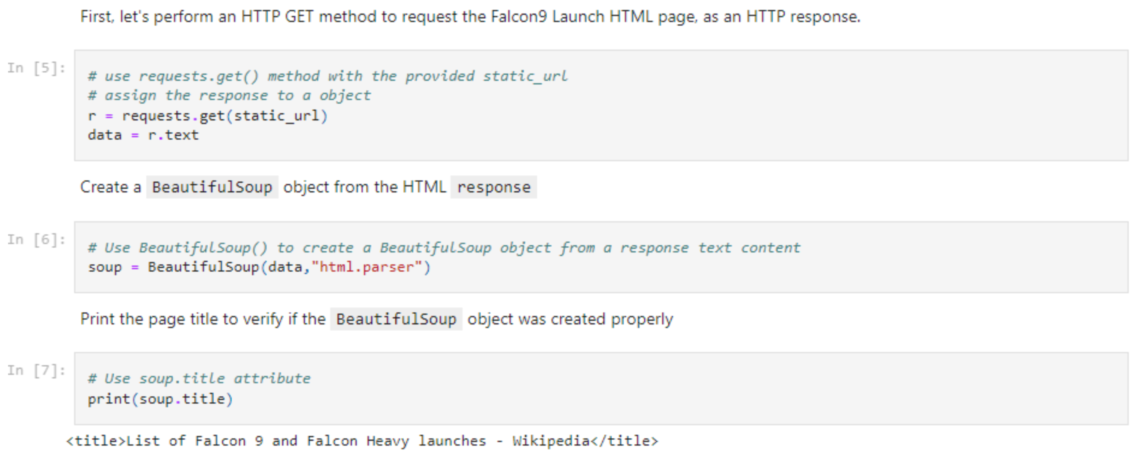
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

# Data Collection with Web 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/Enlightment2000/IBM-Data-Science-Capstone-SpaceX_Final-Project/blob/main/jupyter-labs-webscraping.ipynb



First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        r = requests.get(static_url)
        data = r.text
```

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

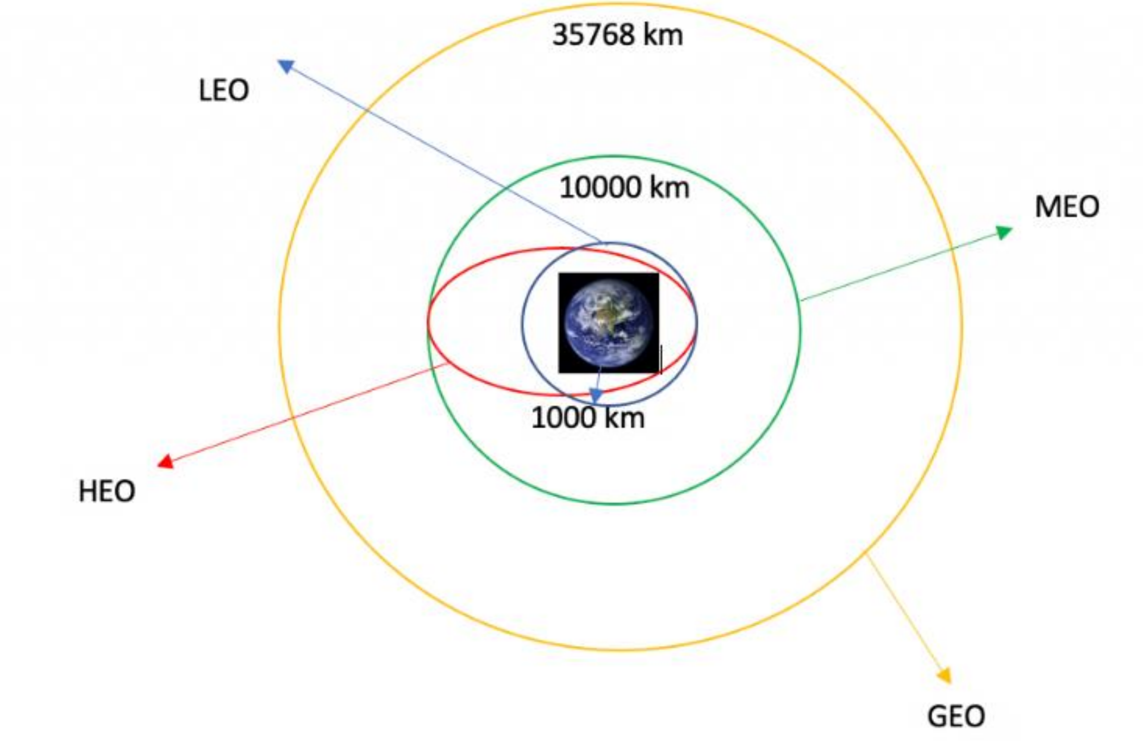
```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(data,"html.parser")
```

Print the page title to verify if the `BeautifulSoup` object was created properly

```
In [7]: # Use soup.title attribute
        print(soup.title)

<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

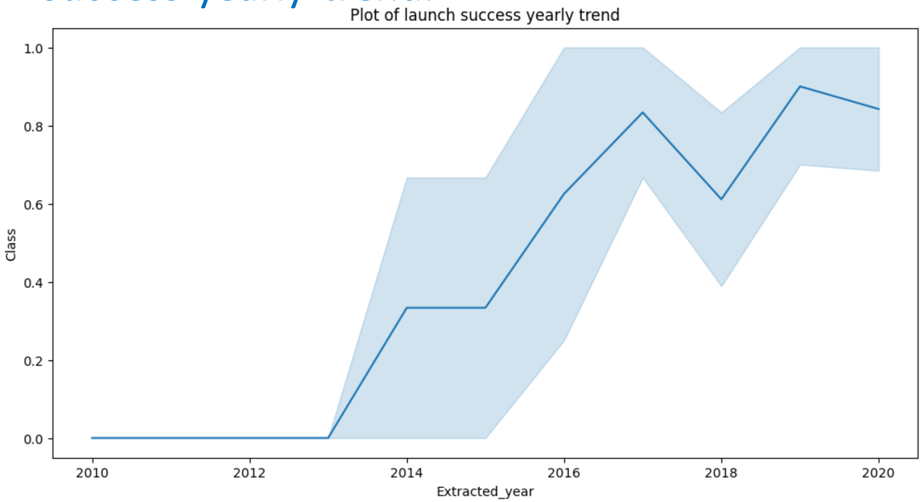
# Data Wrangling

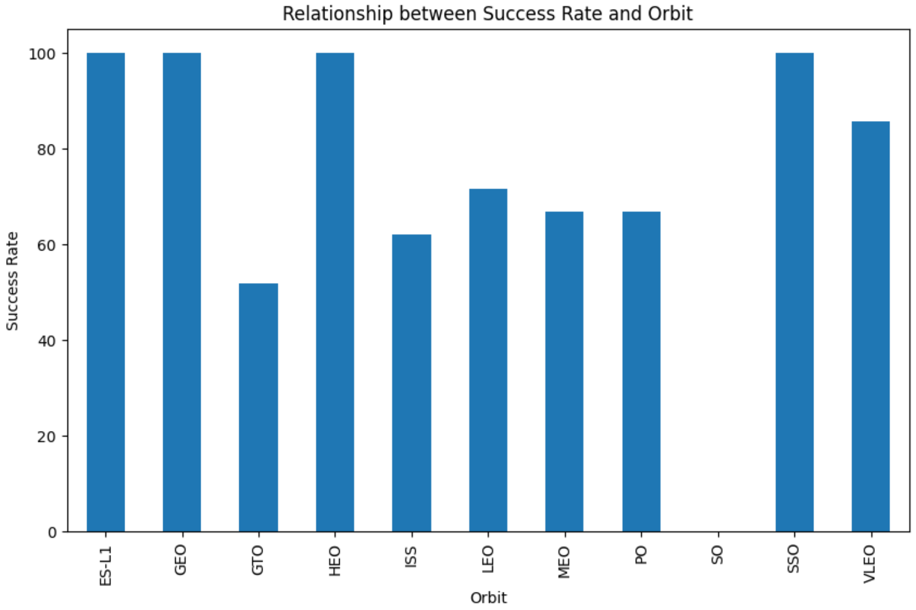
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is:
- https://github.com/Enlightment2000/IBM-Data-Science-Capstone-SpaceX_Final-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



# Exploratory Data Analysis with data visualization

We explored 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 link to the notebook is:

https://github.com/Enlightment2000/IBM-Data-Science-Capstone-SpaceX_Final-Project/blob/main/labs-EDA-Data%20Visualization.ipynb.jupyterlite.ipynb

# Exploratory Data Analysis with SQL

- We loaded the SpaceX dataset into a PostgreSQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.

- The link to the notebook is: https://github.com/Enlightment2000/IBM-Data-Science-Capstone-SpaceX_Final-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

# Interactive Visual Analytics with Folium

- We marked all launch sites, and added 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.
- The link to the notebook is : https://github.com/Enlightment2000/IBM-Data-Science-Capstone-SpaceX_Final-Project/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

# Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.
- The link to the notebook is: [https://github.com/Enlightment2000/IBM-Data-Science-Capstone-SpaceX_Final-Project/blob/main/spacex_dash_app.py](https://github.com/Enlightment2000/IBM-Data-Science-Capstone-SpaceX_Final-Project/blob/main/spacex_dash_app.py)

# Machine Learning Prediction

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model.
- The link to the notebook is

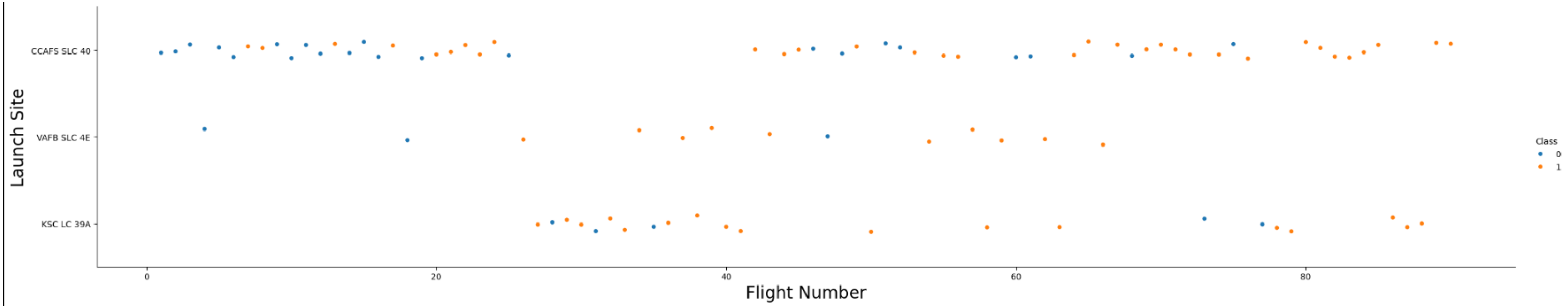
# RESULTS

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

# Insight Drawn from EDA

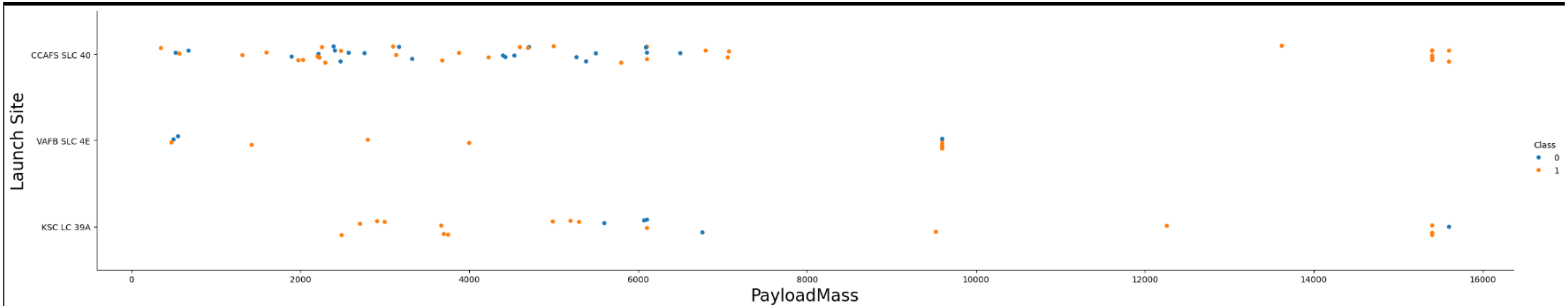
# Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

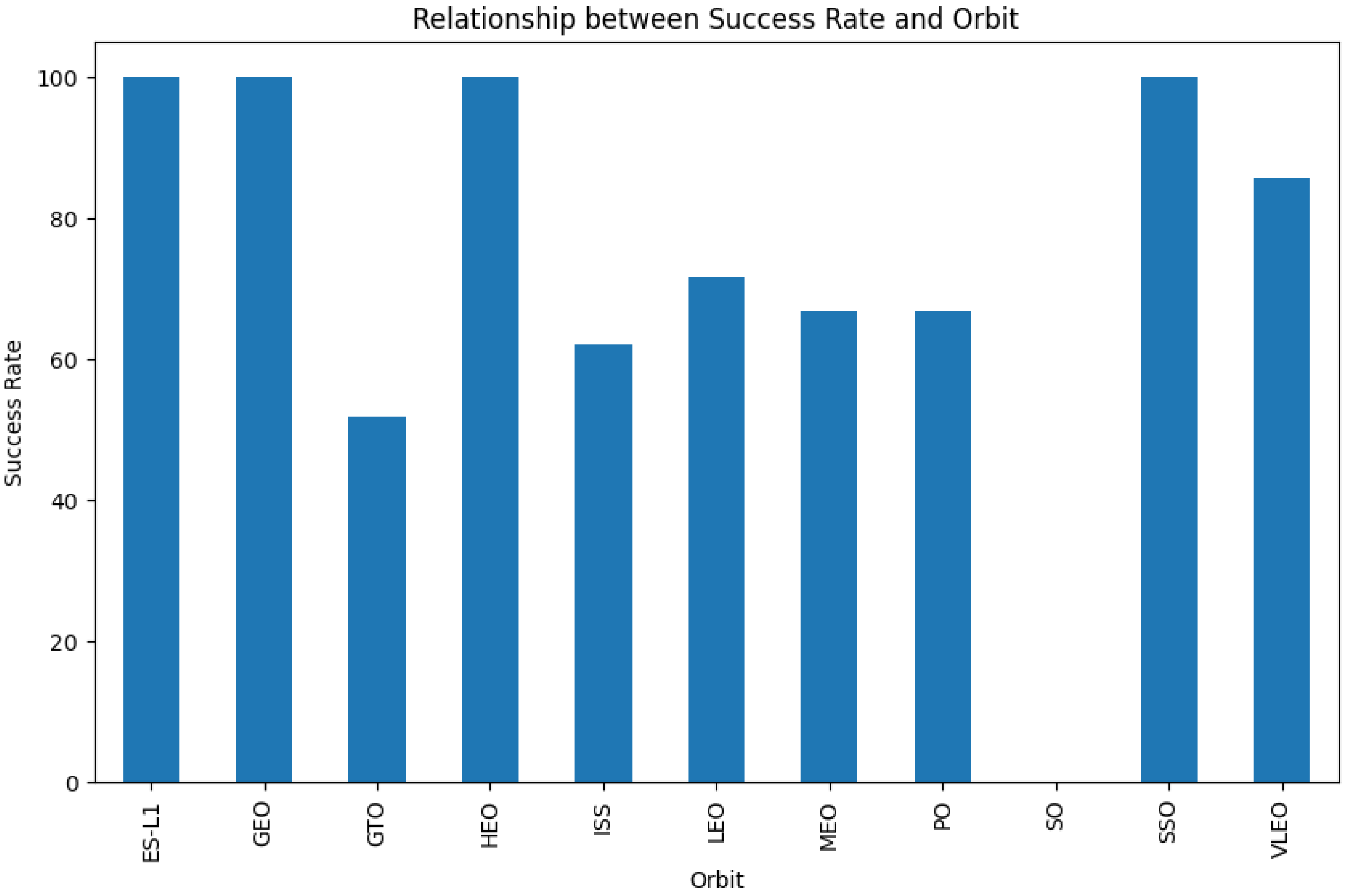


# Payload vs. Launch Site

- From the scatter point chart you will find for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



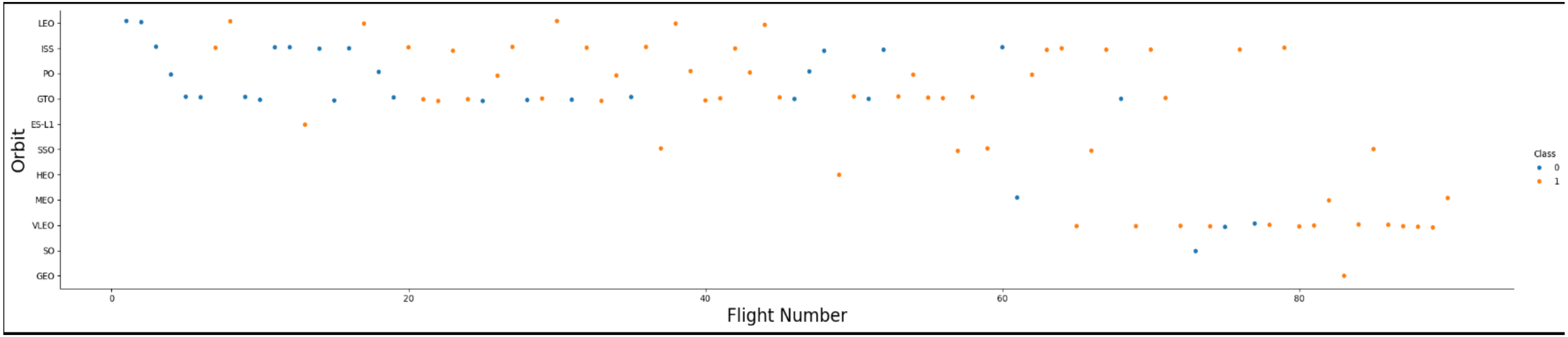
# Success Rate vs. Orbit Type



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

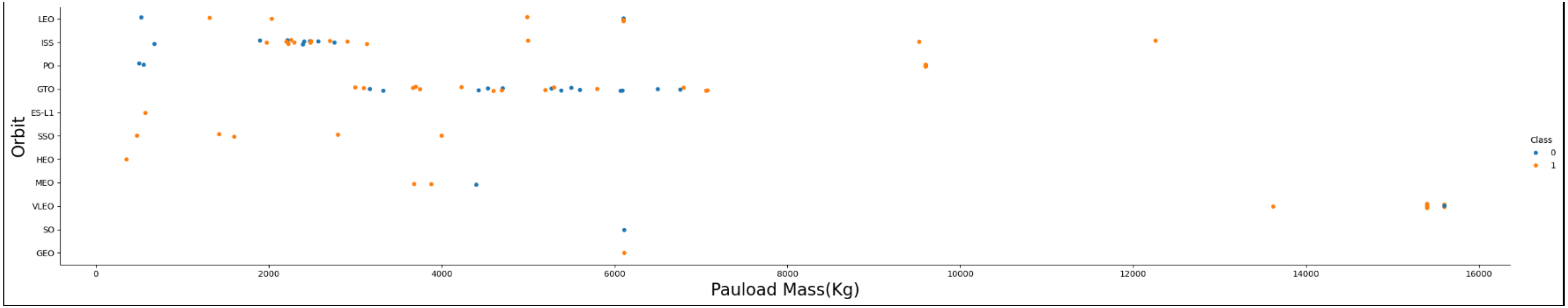
# Flight Number vs. Orbit Type

From the plot we 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.

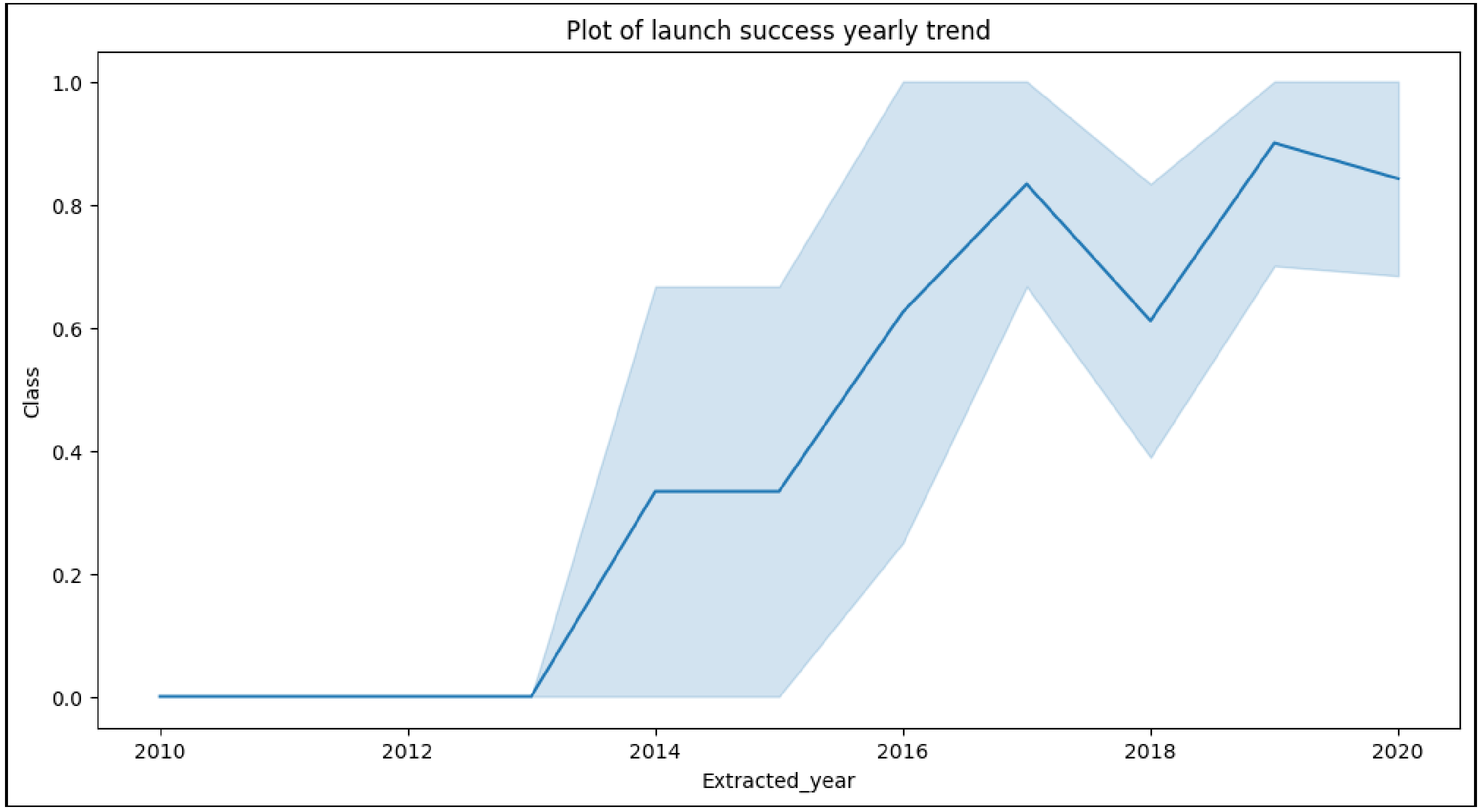


# Payload and Orbit type

From the plot , with heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS. However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



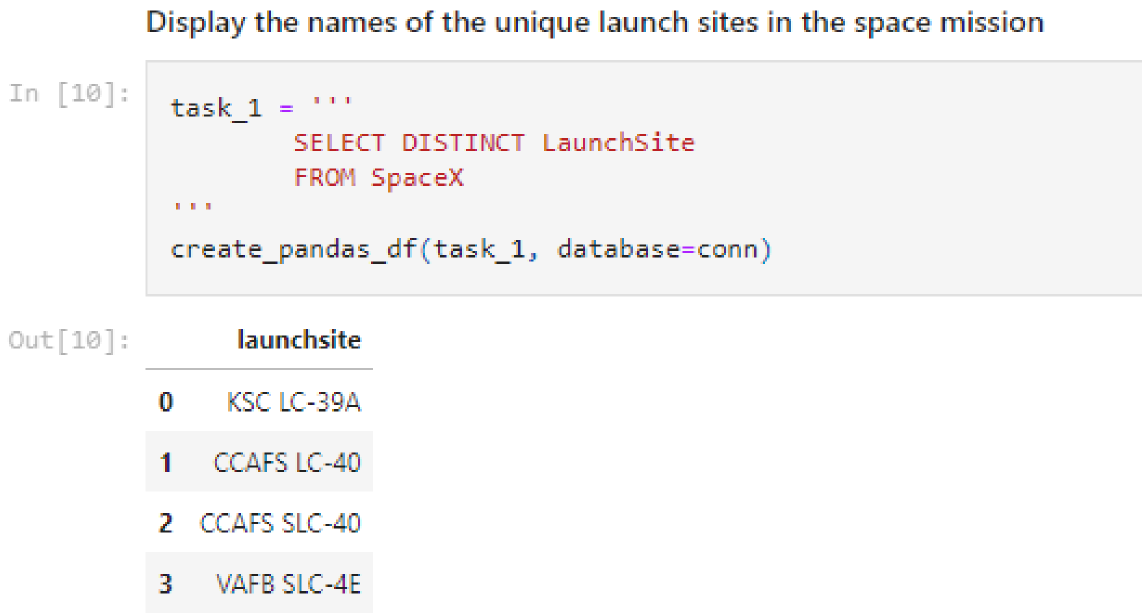
# Launch Success Yearly Trend



From the plot, we can observe that success rate since 2013 kept on increasing till 2020.

# All Launch Site Names

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



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

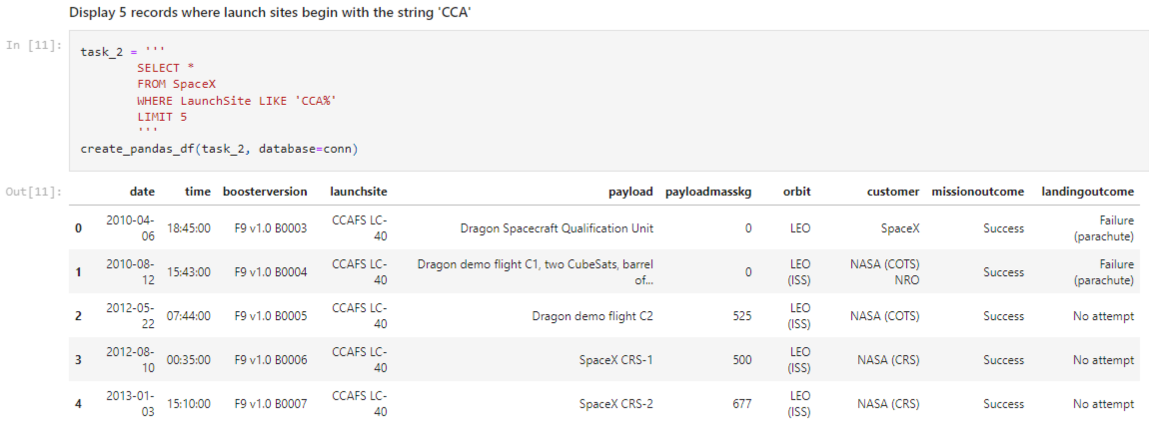
In [10]:

```
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 |

# Launch Site Names 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)
```

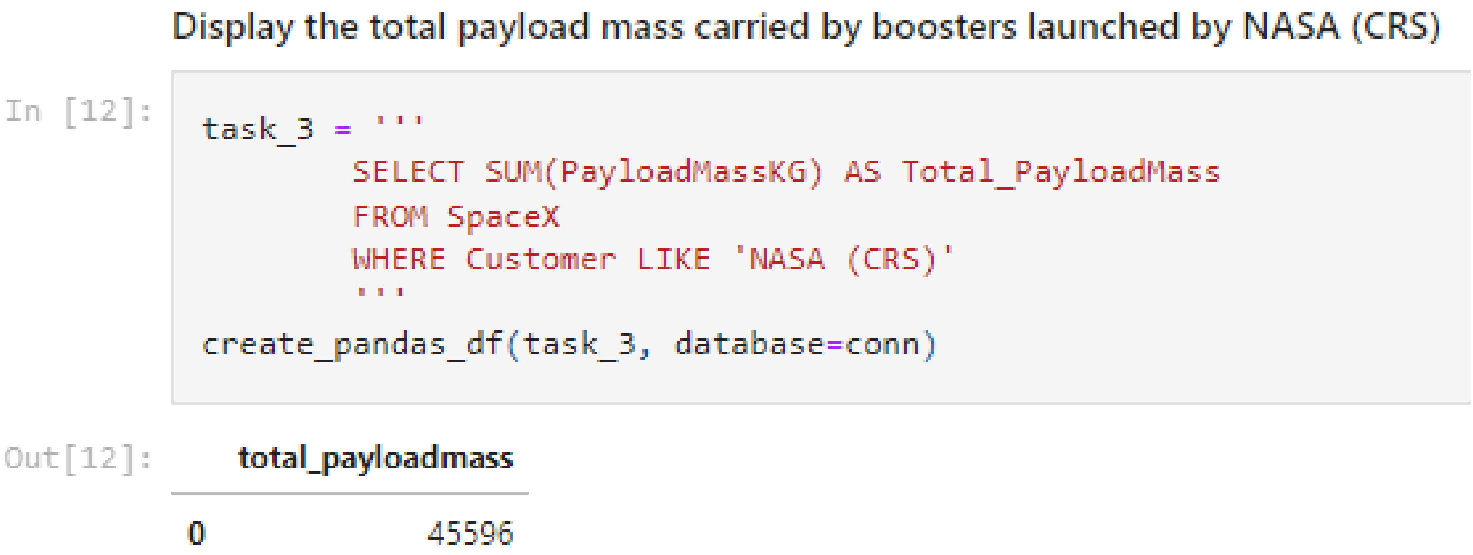
Out[11]:

| | 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 |

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

# Total Payload Mass

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



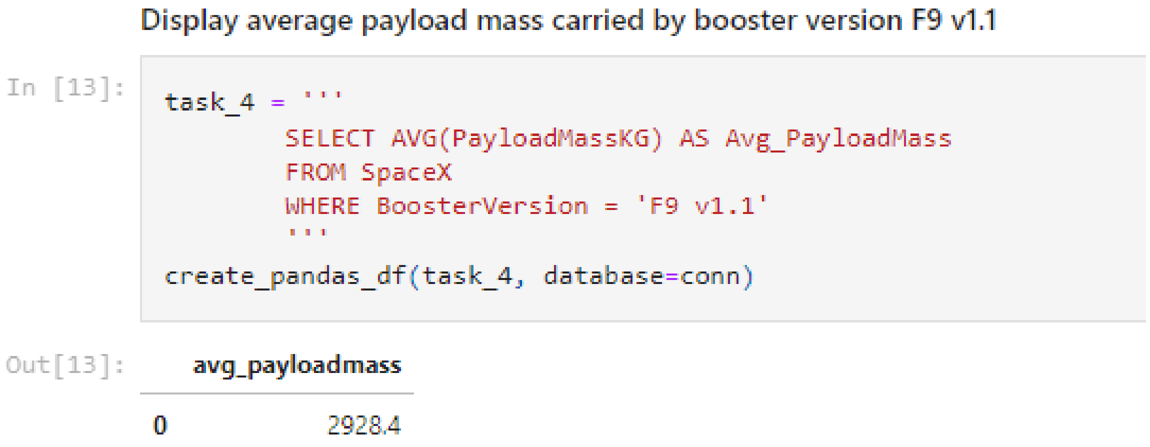
Display the total payload mass carried by boosters launched by NASA (CRS)

```
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

- 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

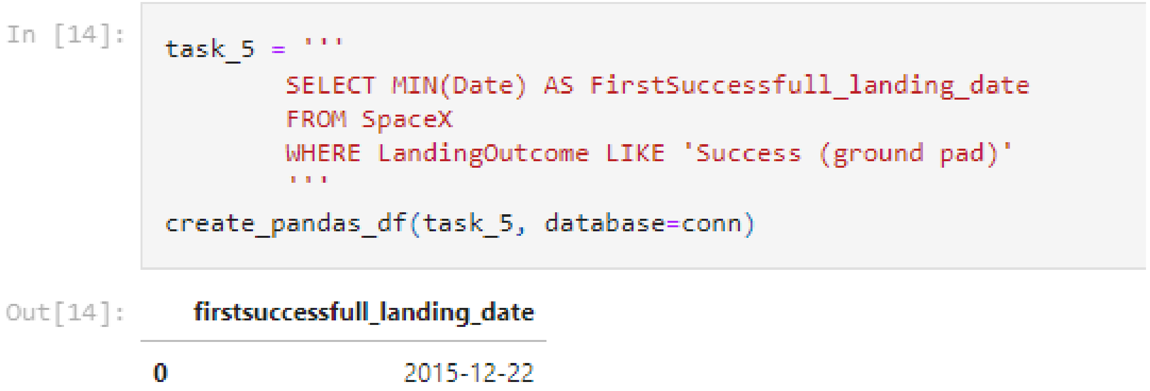
```
In [13]: task_4 = '''
                 SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
                 FROM SpaceX
                 WHERE BoosterVersion = 'F9 v1.1'
                 '''
         create_pandas_df(task_4, database=conn)
```

Out[13]:

|   | avg_payloadmass |
|---|---|
| 0 | 2928.4 |

# First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

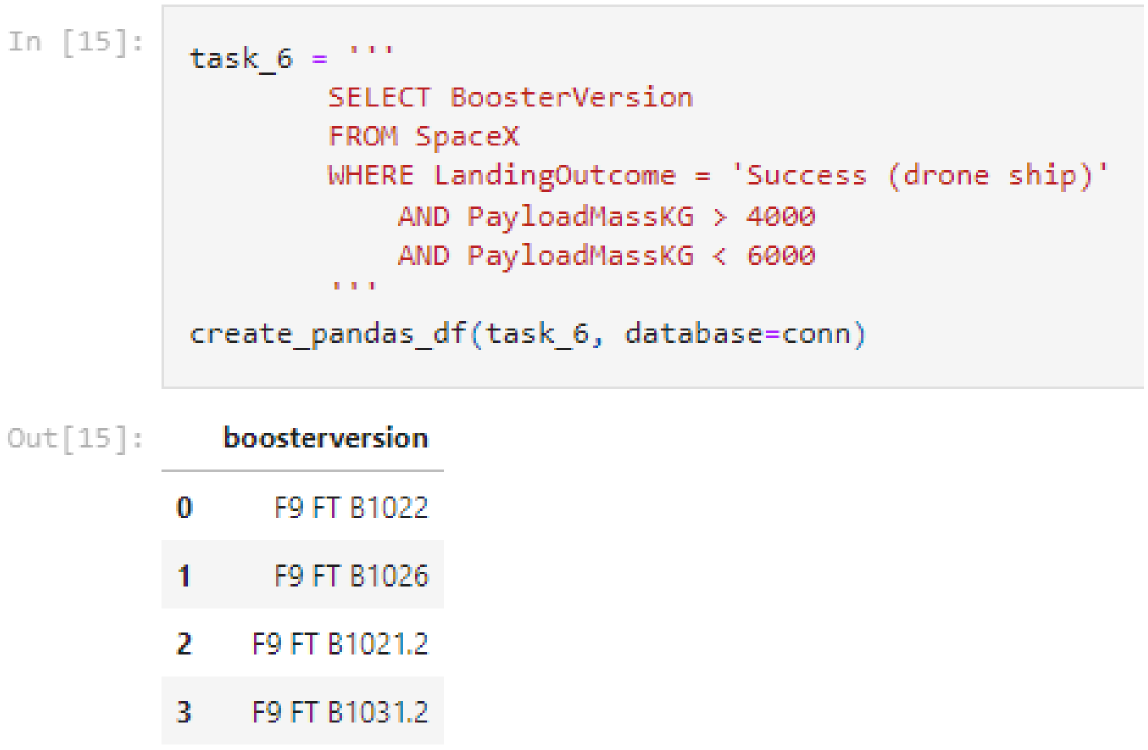


```
In [14]: task_5 = '''
                 SELECT MIN(Date) AS FirstSuccessfull_landing_date
                 FROM SpaceX
                 WHERE LandingOutcome LIKE 'Success (ground pad)'
                 '''
         create_pandas_df(task_5, database=conn)
```

```
Out[14]:    firstsuccessfull_landing_date
         0                     2015-12-22
```

# Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000



In [15]:

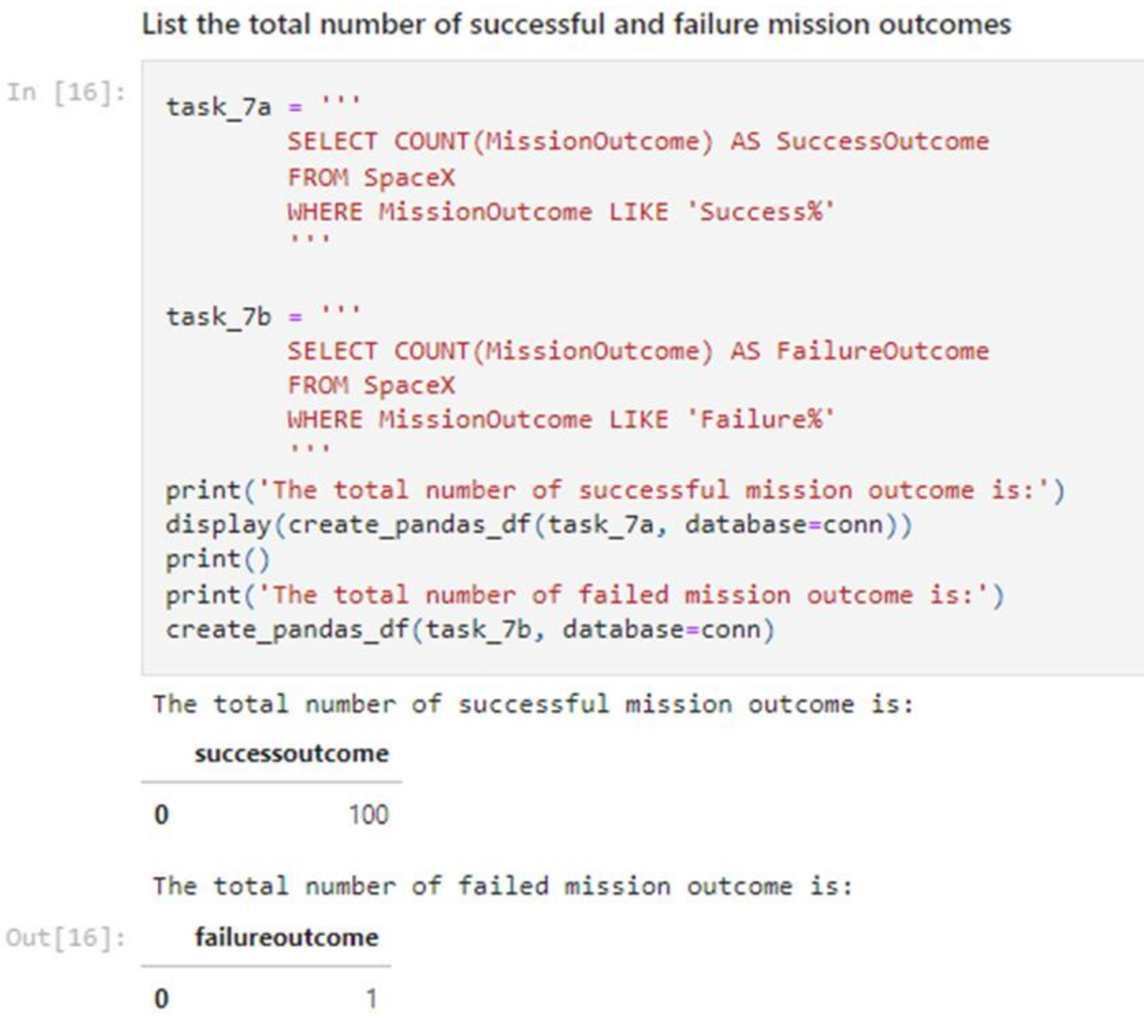
```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

Out[15]:

|   | boosterversion |
|---|---|
| 0 | F9 FT B1022 |
| 1 | F9 FT B1026 |
| 2 | F9 FT B1021.2 |
| 3 | F9 FT B1031.2 |

# Total Number of Successful and Failure Mission Outcomes

- We used wildcard like ‘%’ to filter for **WHERE** MissionOutcome was a success or a failure.



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 |

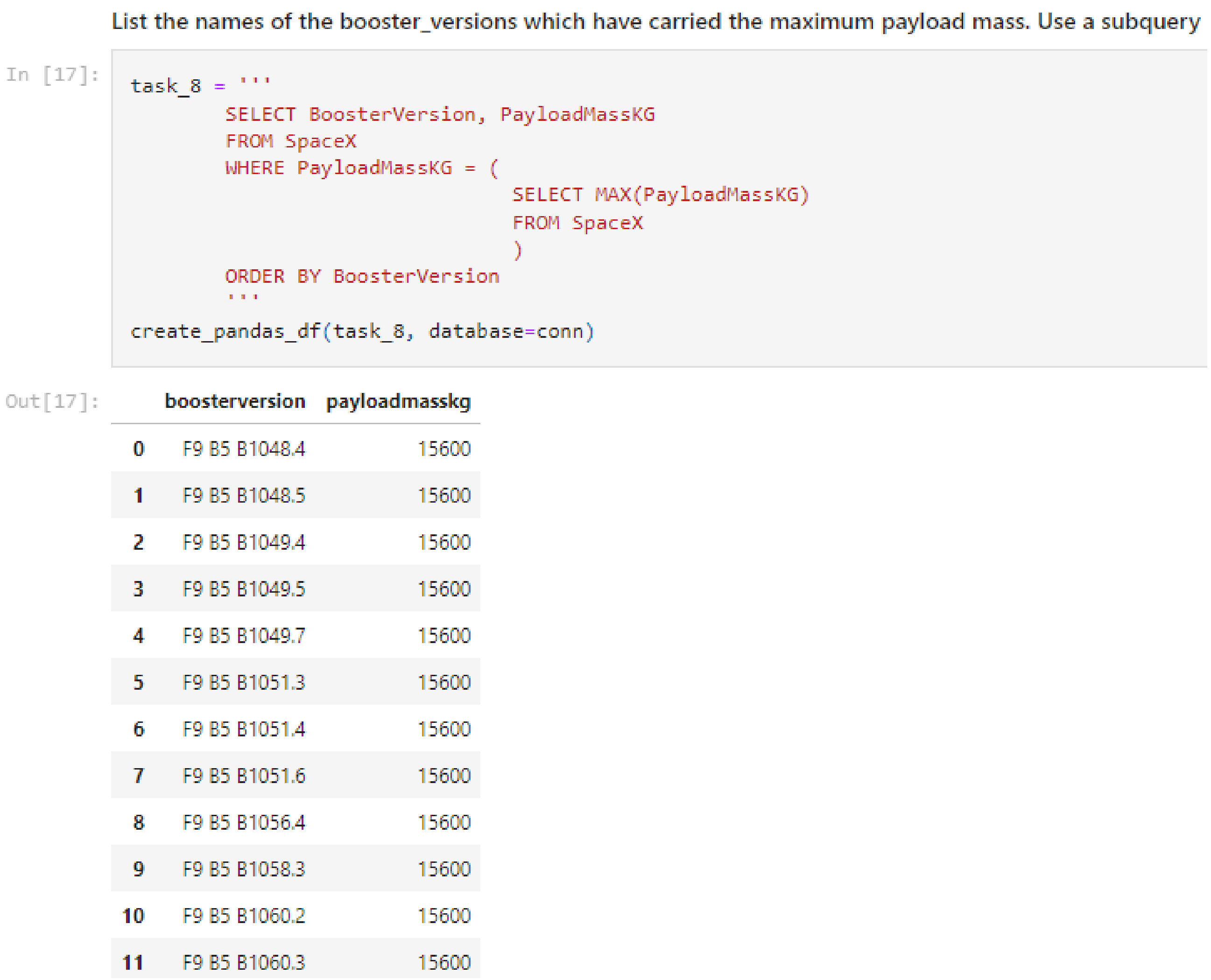
```
The total number of failed mission outcome is:
```

Out[16]:

| | failureoutcome |
|---|---|
| **0** | 1 |

# Boosters Carried Maximum Payload

- We determined 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]:

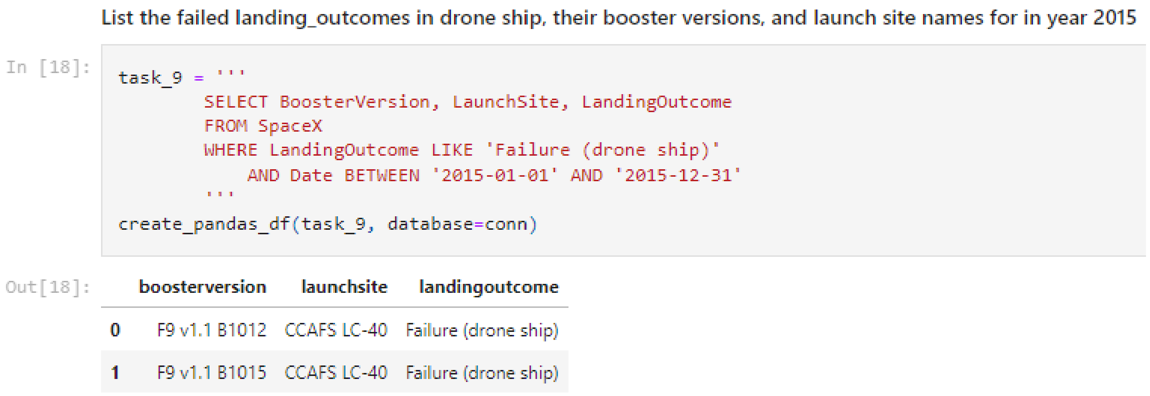
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8,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

# 2015 Launch Records

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015



List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015

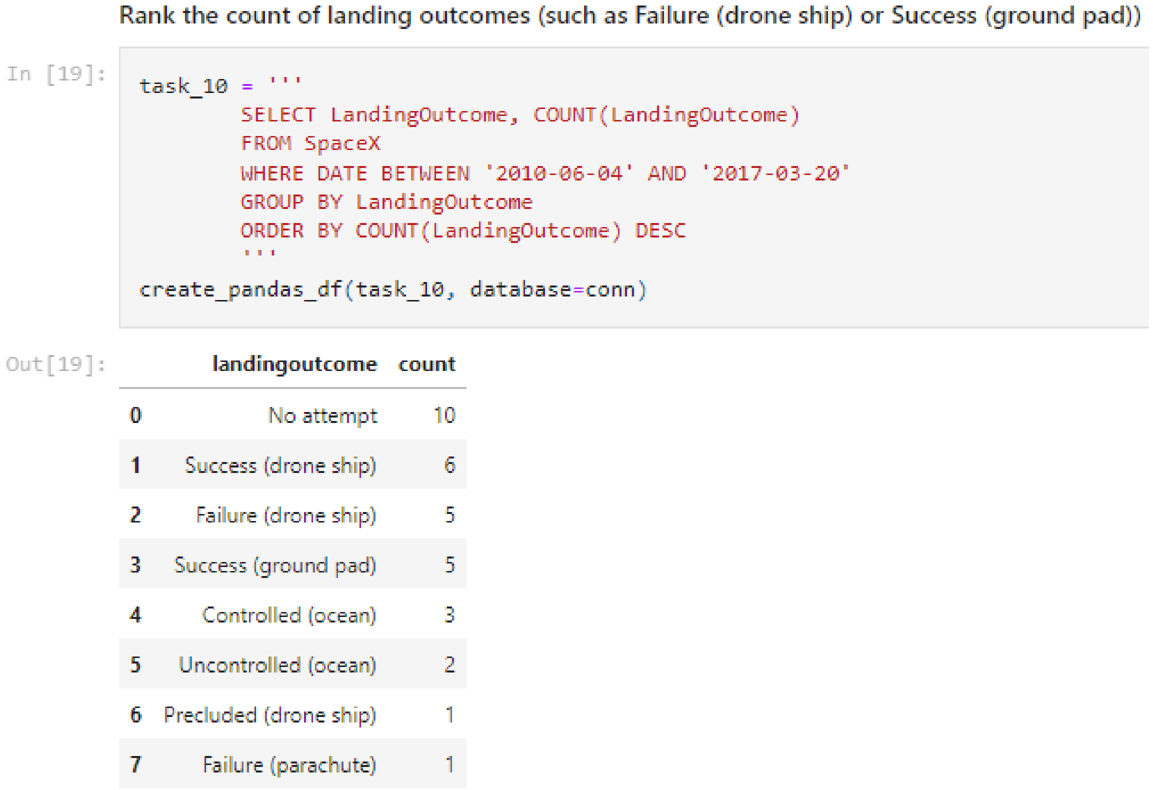
```
In [18]:  task_9 = '''
                  SELECT BoosterVersion, LaunchSite, LandingOutcome
                  FROM SpaceX
                  WHERE LandingOutcome LIKE 'Failure (drone ship)'
                      AND Date BETWEEN '2015-01-01' AND '2015-12-31'
                  '''
          create_pandas_df(task_9, database=conn)
```

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

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 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.



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

In [19]:

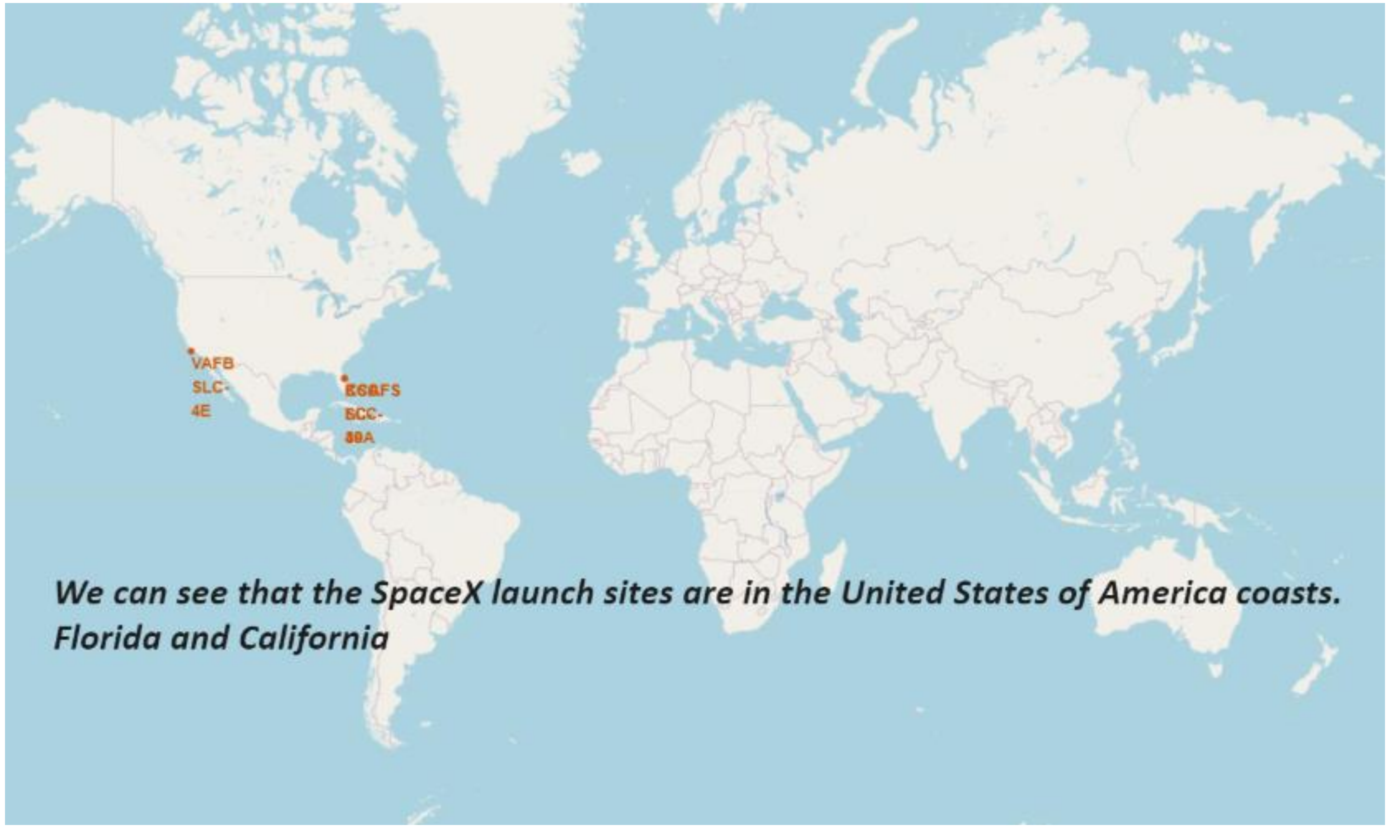
```
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

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 |

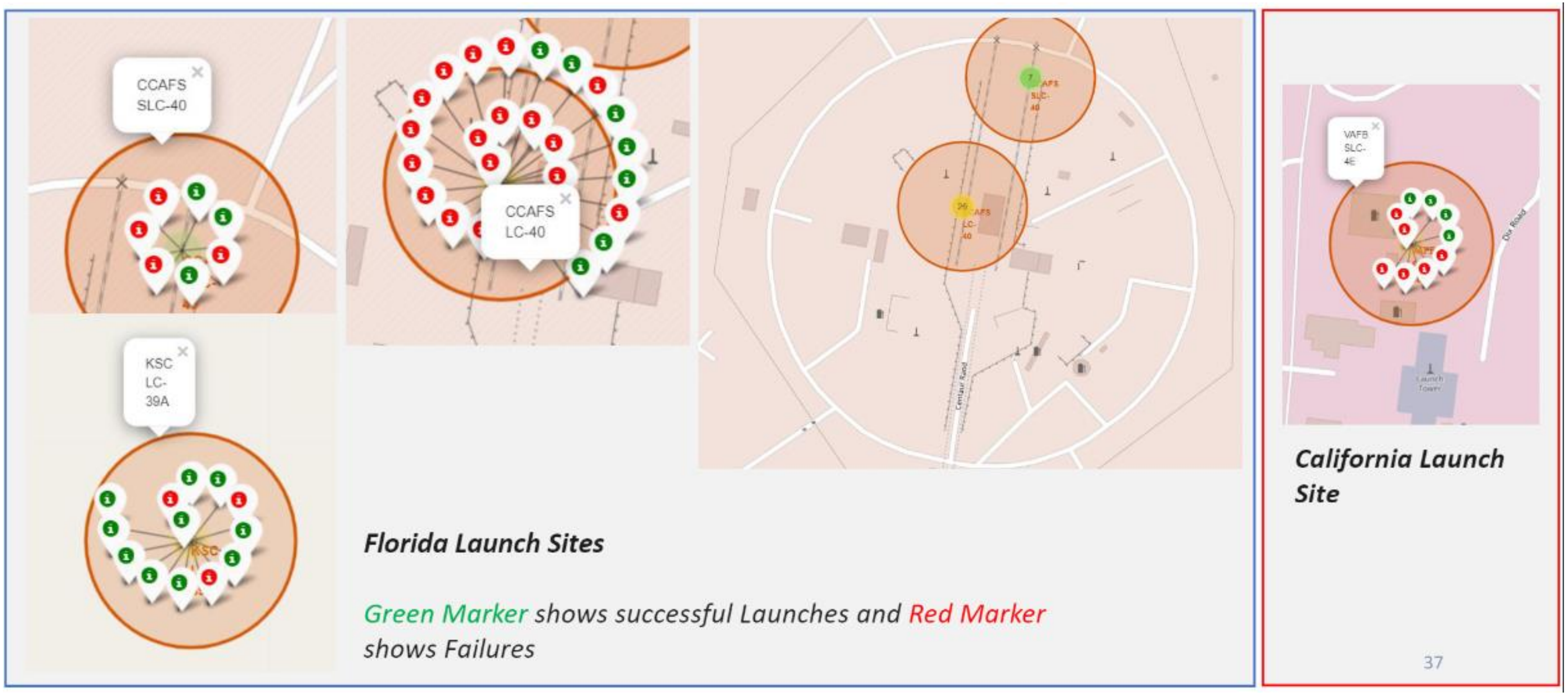
# Launch Sites Proximities Analysis

# All launch sites global map markers



*We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California*

# Markers showing launch sites with color labels

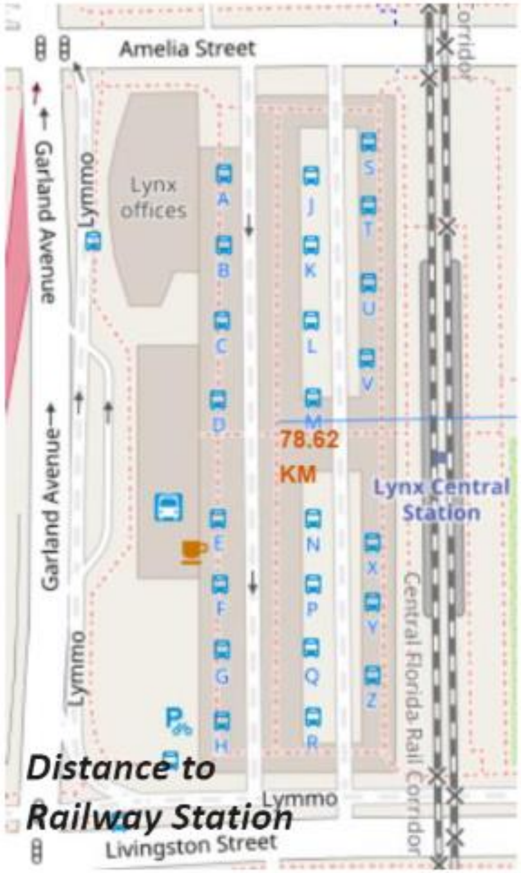


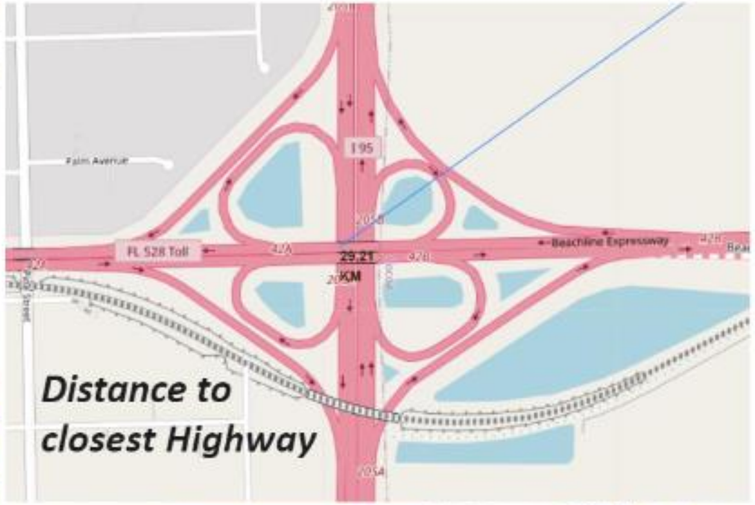
***Florida Launch Sites***

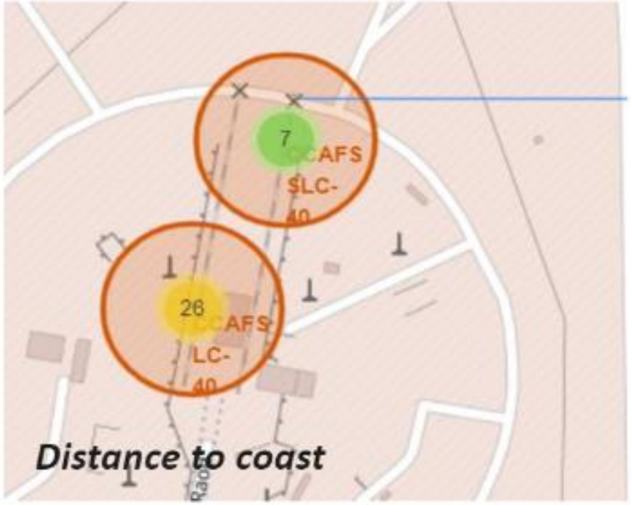
*Green Marker shows successful Launches and Red Marker shows Failures*

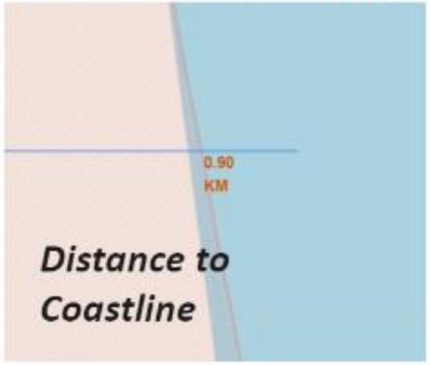
***California Launch Site***

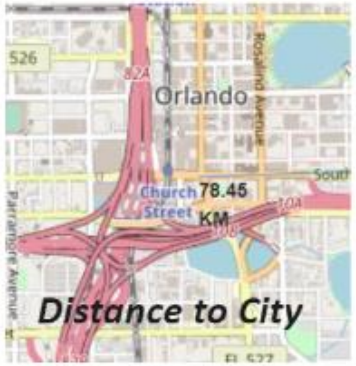
# Launch Site distance to landmarks







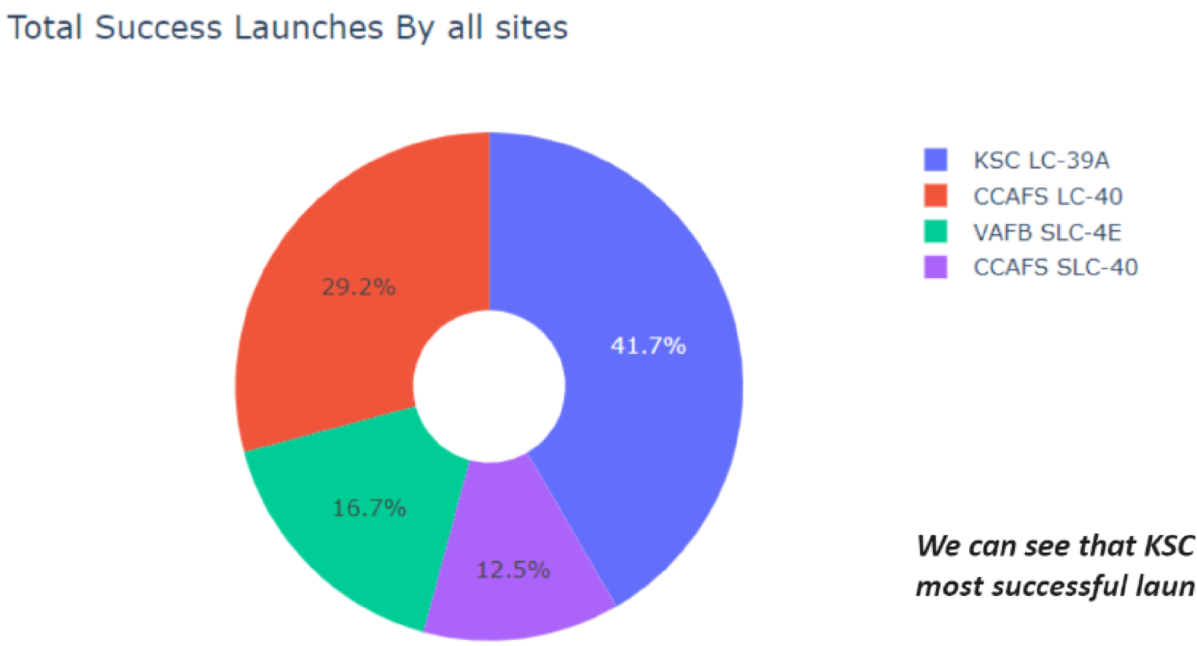




- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

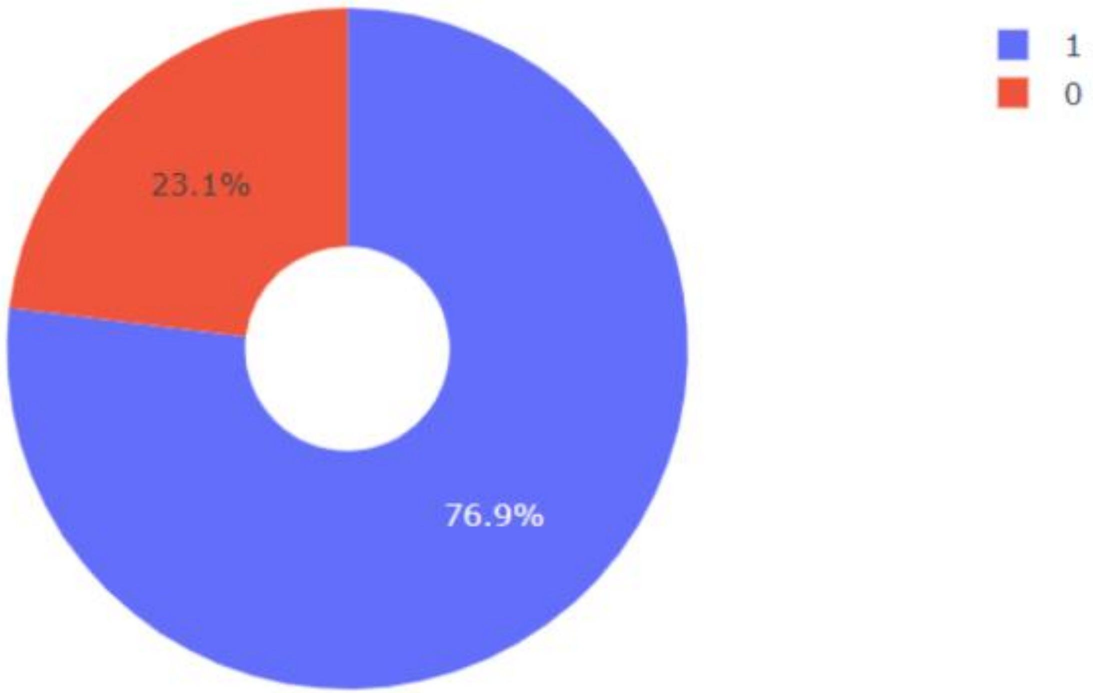
# Build a dashboard with Plotly Dash

# Pie chart showing the success percentage achieved by each launch site



*We can see that KSC LC-39A had the most successful launches from all the sites*

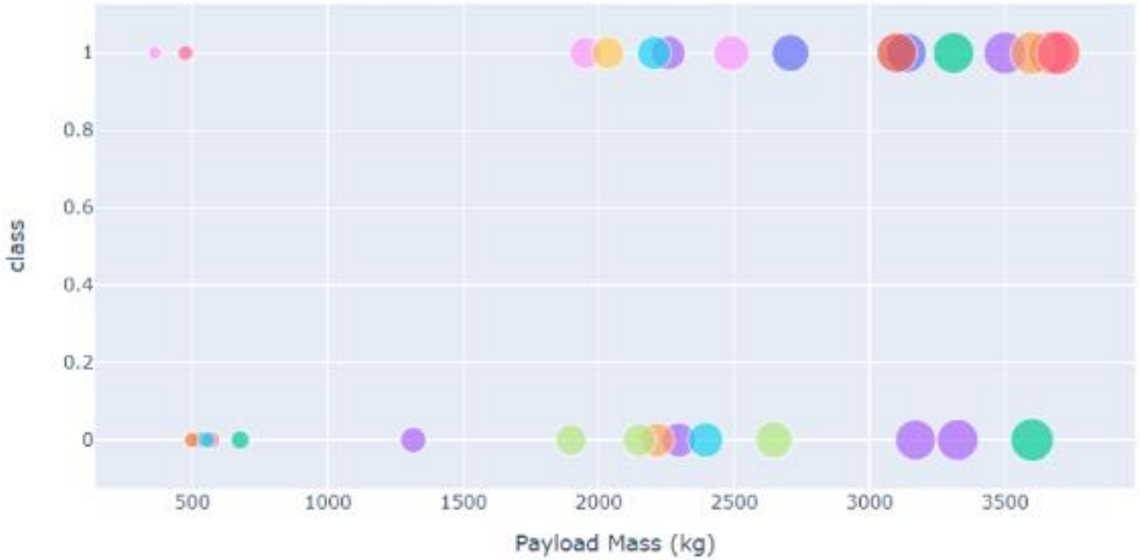
# Pie chart showing the Launch site with the highest launch success ratio



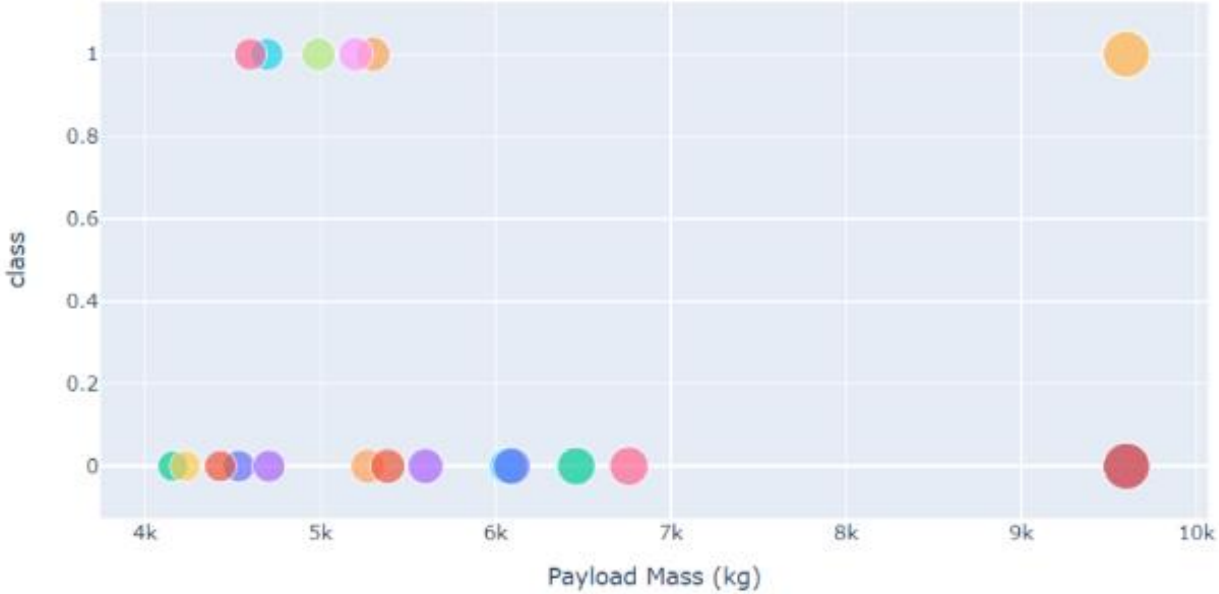
***KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate***

# Scatter plot of Payload vs Launch Outcome for all sites, with different payload selected in the range slider

## *Low Weighted Payload 0kg – 4000kg*



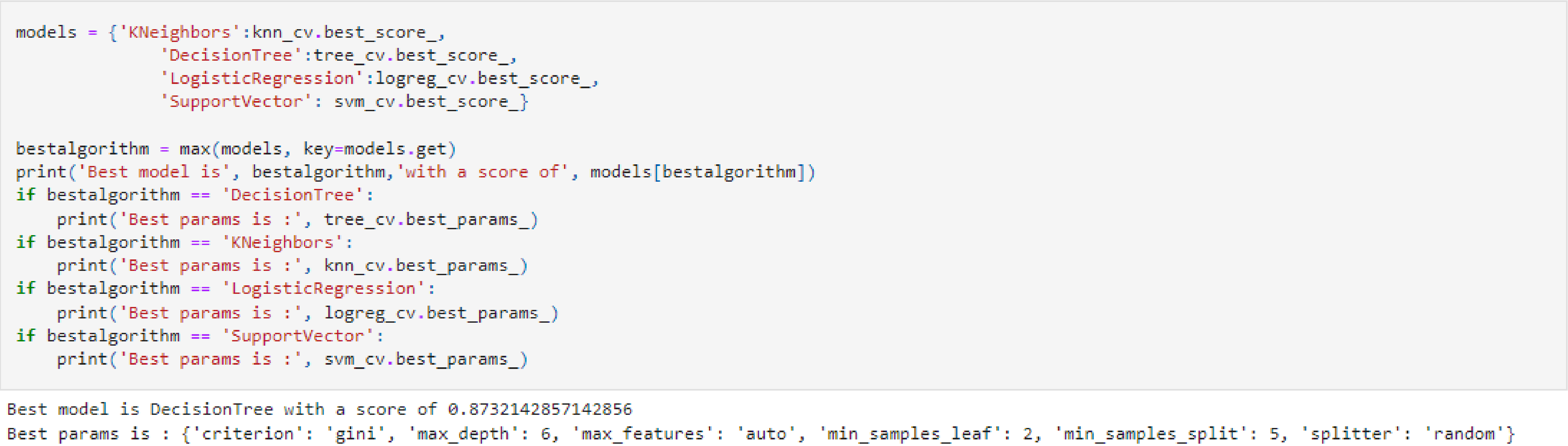
## *Heavy Weighted Payload 4000kg – 10000kg*



*We can see the success rates for low weighted payloads is higher than the heavy weighted payloads*

# Predictive Analysis (Classification)

# Classification Accuracy



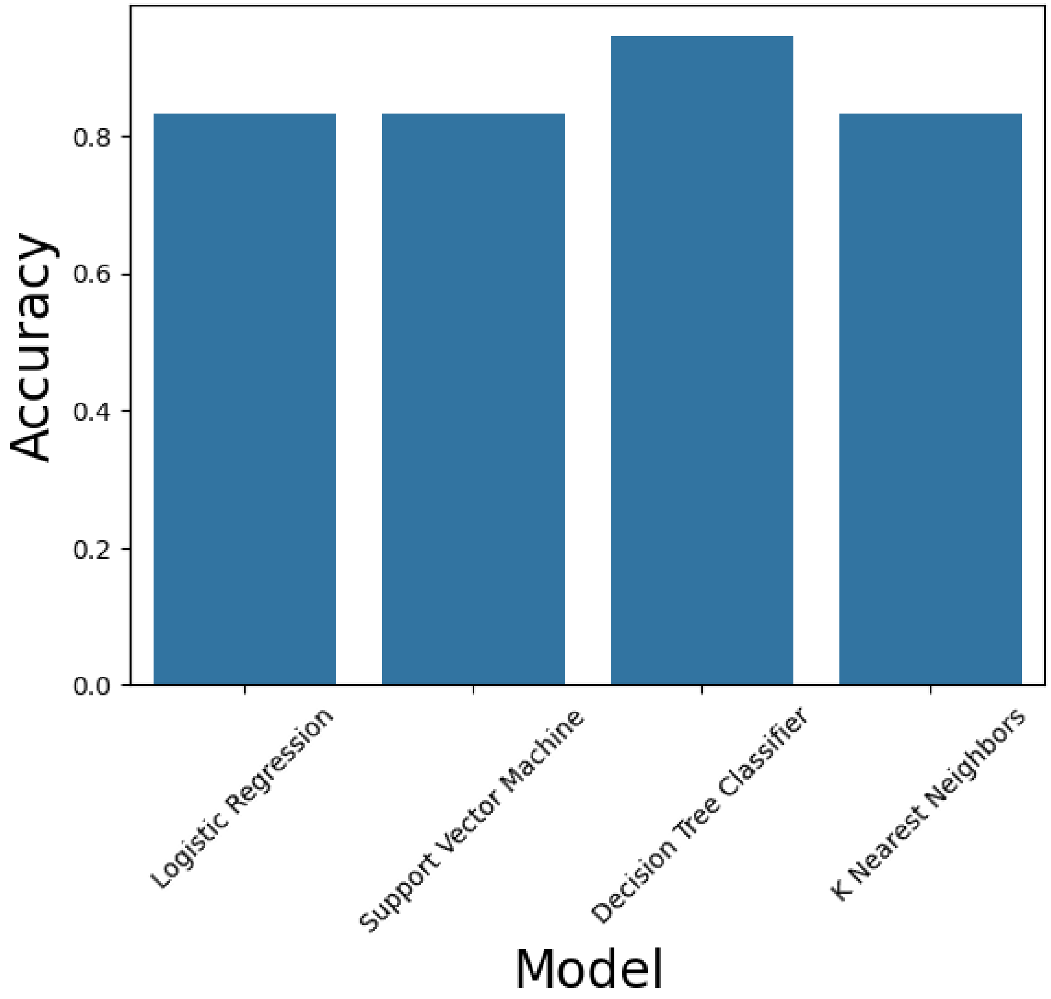
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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'}
```

# Model Accuracy

Predict analysis using classification models in Scikit-learn library

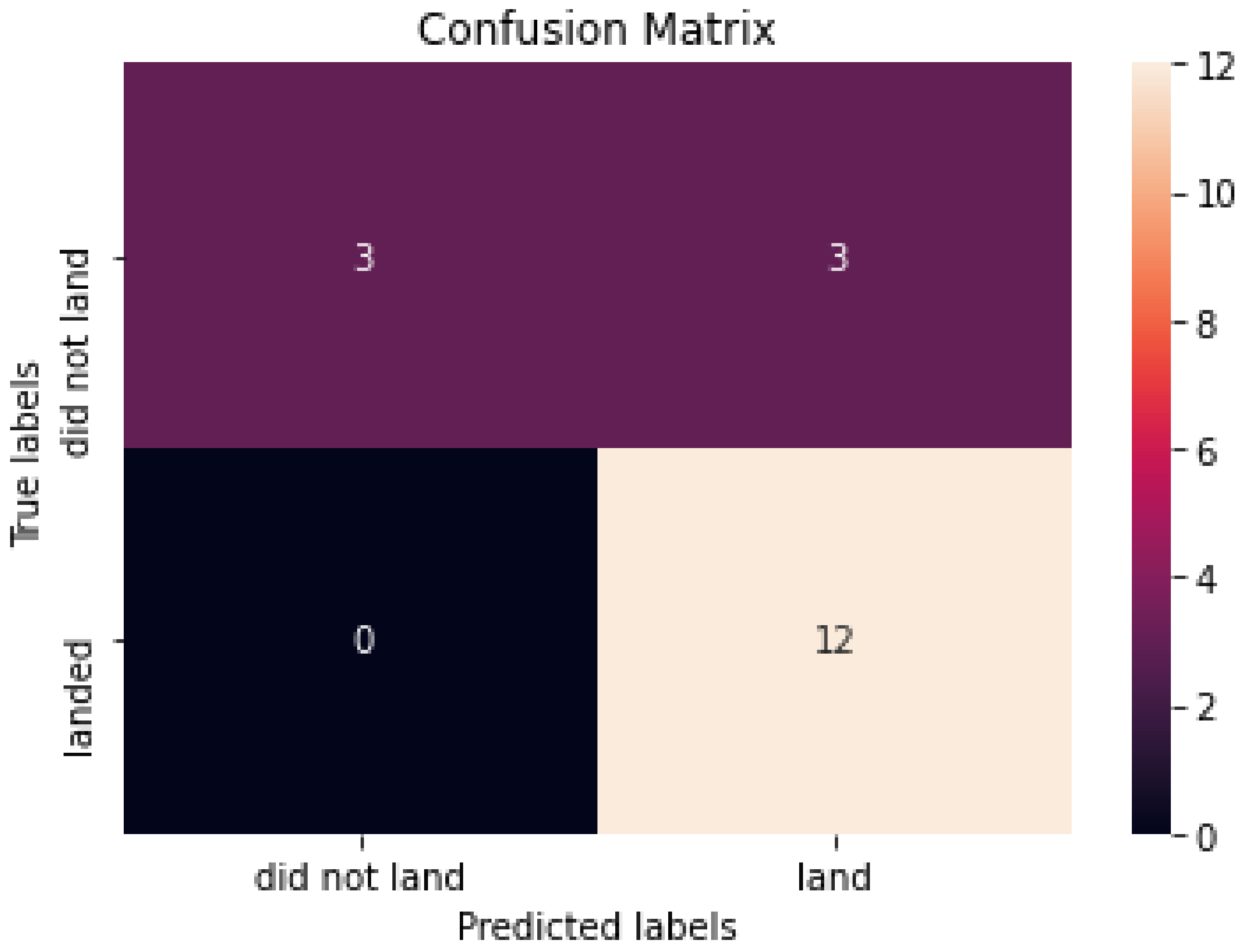
- Preprocessed data
  - One hot encoding for category data
  - Normalized data using standard scaler
  - Split train and test data
  - Trained and evaluated model
- Trained classification models in Scikit-learn library
  - Tuned hyperparameters using GridSearchCV objects
  - Calculated each model performance
  - Visualized each model performance usig confusion matrix
  - Compared each model performance using table and chart



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

# 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.



# Conclusions

We can conclude that:

- The larger the flight amount at a launch site, the greater the success rate at a launch site.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank You!