IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

María Victoria Castro Domenech
14th November 2024

# Outline

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

# Executive Summary



- Summary of methodologies
- Summary of all results

# Introduction



- Project background and context

Nowadays, aerospace companies are trying to make space travel affordable for everyone. Among them, perhaps the most successful is SpaceX. SpaceX was able to reduce the costs of rocket launches to around a third part of the expenses of its competitors. Much of the savings is because SpaceX can reuse the first stage.

Stage two, or the second stage, helps bring the payload to orbit, but most of the work is done by the first stage. This stage does most of the work and is much larger and expensive than the second stage.

Therefore, the main goal of this project is to determine if the first stage will successfully land and to determine, as well, the cost of a launch in those cases.

- Problems you want to find answers

Our new company, Space Y, rulled by Allon Musk wants to determine the price of each launch.

For doing so, information about Space X is gathered and some dashboards are created to observe if SpaceX will reuse the first stage. Diverse machine learning models have been trained using public information to predict if SpaceX will reuse the first stage.

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - SpaceX launch data that was gathered from the SpaceX REST API. To get past launch data, the used endpoint was: api.spacexdata.com/v4/launches/past.
  - The response was a list of JSON objects, each object represents a launch. This structured json data was converted into a flat table by means of the json_normalize function.
  - Another source for obtaining Falcon 9 Launch data was Wiki pages. Python BeautifulSoup package was used to web scrape some HTML tables that contain valuable Falcon 9 launch records. Then, the data from those tables was converted into Pandas data frames for further visualization and analysis
- Perform data wrangling
  - The API was used again targeting another endpoint to gather specific data for each ID number. The data was stored in lists and used to build the dataset.

# Methodology



- Additionally, the launch data was filtered to exclude data from the Falcon 1 given that only Falcon 9's data was needed.
- Finally, the null values in the "Pay Load Mass" feature were imputed to the mean value that column.
- A one hot encoding was performed for the LandingPad feature.

- Perform exploratory data analysis (EDA) using visualization and SQL
  - The features that were present in the data set and their possible values were explored.
  - The correlation between each feature and the outcome was analyzed.
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Several machine learning models were build and evaluated (Linear and Logistic Regression, Decision Tree, KNN and SVM,).
  - Preprocessing, normalizing the data, and Train_test_split, were performed to prepare data before fitting the models.
  - Grid Search was applied to find the hyperparameters that allow a given algorithm to perform best.

# Data Collection



- Collecting Data Using REST API:

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

```
response =requests.get(url)

response.json()
```

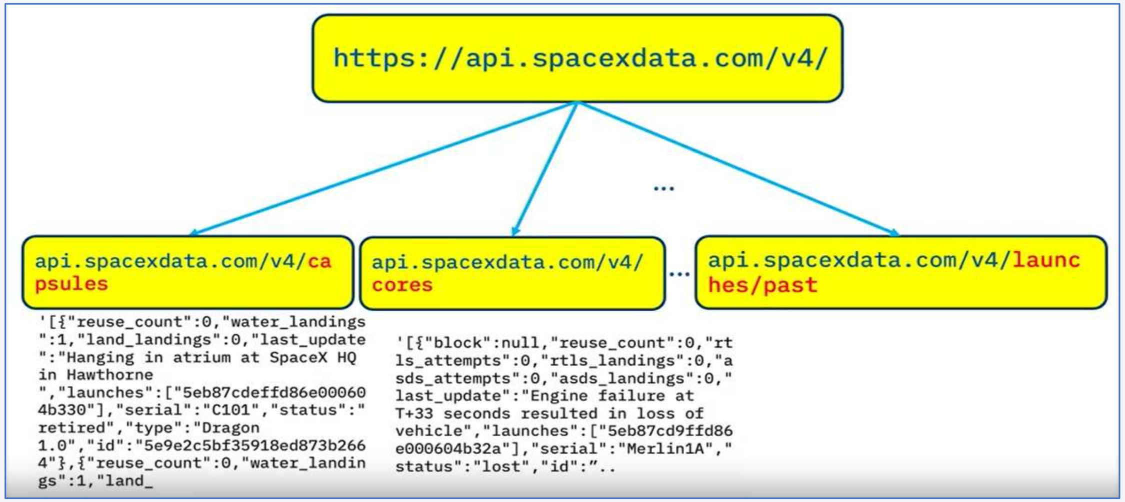
```
data = pd.json_normalize(response.json())
```

| | static_fire_date_utc | static_fire_date_unix | tbd | net | window | rocket | success | details | crew | ships | capsules | payloads | launchpad | auto_update | failures | flight_number | name | date_u |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2006-03-17T00:00:00.000Z | 1.142554e+09 | False | False | 0.0 | [illegible] | False | Engine failure at 33 seconds and loss of vehicle | [] | [] | [] | [illegible] | [illegible] | True | [{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}] | 1 | FalconSat | 2006-0 24T22:30:00.00 |
| 1 | None | NaN | False | False | 0.0 | [illegible] | False | Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage | [] | [] | [] | [illegible] | [illegible] | True | [{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}] | 2 | DemoSat | 2007-0 21T01:10:00.00 |

# Data Collection – SpaceX API



- The goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
- The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
- There are different end points, for example: /capsules and /cores
- The endpoint used was: api.spacexdata.com/v4/launches/past.
- Then a get request is performed using the requests library to obtain the launch data.
- This result can be viewed by calling the .json() method.
- The response will be a list of JSON objects.




https://github.com/viquiriglos/IBM-CapstoneProject/jupyter-labs-spacex-data-collection-api.ipynb

# Data Collection - Scraping

Set the URL and create a response object:

```
static_url =
https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922
response = requests.get(static_url)
```

Parsing data with BeautifulSoup:

```
soup = BeautifulSoup(response.content)
html_tables=[]
html_tables = soup.find_all('table')
```


| | FlightNumber | Date | BoosterVersion | PayloadMass | Orbit | LaunchSite | Outcome | Flights | GridFins | Reused | Legs | LandingPad | Block | ReusedCount | Serial | Longitude | Latitude |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2006-03-24 | Falcon 1 | 20.0 | LEO | Kwajalein Atoll | None None | 1 | False | False | False | None | NaN | 0 | Merlin1A | 167.743129 | 9.047721 |
| 1 | 2 | 2007-03-21 | Falcon 1 | NaN | LEO | Kwajalein Atoll | None None | 1 | False | False | False | None | NaN | 0 | Merlin2A | 167.743129 | 9.047721 |
| 2 | 4 | 2008-09-28 | Falcon 1 | 165.0 | LEO | Kwajalein Atoll | None None | 1 | False | False | False | None | NaN | 0 | Merlin2C | 167.743129 | 9.047721 |

https://github.com/viquiriglos/IBM-CapstoneProject/jupyter-labs-webscraping.ipynb

# Data Wrangling



- Some of the columns, like rocket, there is an identification number, not actual data. Thus it is needed to use the API again targeting another endpoint to gather specific data for each ID number. Some functions were created and used the following: Booster, Launchpad, payload, and core. The data was stored in lists and used to create the dataset.
- Data for the Falcon 1 booster was filtered and excluded as it was not relevant.
- NULL values inside the PayloadMass were imputed with the mean of that column.
- NULL values within LandingPad column remained, as they represented when a landing pad is not used. They were encoded for classification purposes using one hot encoding later on.

https://github.com/viquiriglos/IBM-CapstoneProject/labs-jupyter-spacex-Data wrangling.ipynb

# EDA with Data Visualization



- Charts Summary:

1. FlightNumber vs PayloadMass (scatter plot): this chart was made to observe if these two variables have an effect on the Launch Outcome. A scatter plot was chosen given that the FlightNumber (x-axis) is not a continuous variable.
2. FlightNumber vs PayloadMass (scatter plot), to observe the relationship between these two variables and how they might affect the Launch Outcome.
3. PayloadMass vs LaunchSite (scatter plot), to visualize the relationship between these two variables and the Launch Outcome.
4. Orbit vs SuccessRate (bar plot) to study the effect of the chosen orbit in the Launch Outcome.
5. FlightNumber vs Orbit (scatter plot), to observe the relationship between these two variables and how they affect the Launch Outcome.
6. PayloadMass vs Orbit (scatter plot), to observe the relationship between these two variables and how they affect the Launch Outcome.
7. Date vs SuccessRate (line plot), to visualize the launch success yearly trend.

https://github.com/viquiriglos/IBM-CapstoneProject/edadataviz.ipynb

# EDA with SQL



SQL queries summary:

- Obtain the names of the unique launch sites in the space mission,
- Display 5 records where launch sites begin with the string 'CCA'
- Calculate the total payload carried by boosters from NASA
- Calculate the average payload mass carried by booster version F9 v1.1
- List the names of boosters successfully landed on drone ship, having payload mass between 4000 and 6000
- Calculate the total number of successful and failure mission outcomes
- List the names of the booster which have carried the maximum payload mass
- List the failed landing_outcomes in drone ship, booster versions, and launch site names occurred in 2015
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.



https://github.com/viquiriglos/IBM-CapstoneProject/jupyter-labs-eda-sql-coursera_sqllite(1).ipynb

# Build an Interactive Map with Folium



Summary of displayed:

- General overview of the launch sites locations in a global map.
- A more detailed view of the sites located in the US eastern coast.
- Distance measurement from a launch site to the coast.

https://github.com/viquiriglos/IBM-CapstoneProject/lab_jupyter_launch_site_location.ipynb

# Build a Dashboard with Plotly Dash

Summary of the plots and elements included in the dashboard:

- dropdown list to interact with the pie charts. When "All Sites" is selected, the pie chart should present the proportion of successful launches for each site. In case that a specific site is selected, the chart will display the proportion between successful and failed launches in that site.

- Range slider to interact with the scatter point chart. The scatter plot presents the successful and failed launches as a function of the payload mass. Additionally, the launches are coloured differently for each booster version. The range slider allows to set the maximum and minimum values for the payload.

https://github.com/viquiriglos/IBM-CapstoneProject/Dash-Lab-IBM-C10-Mod3_Final.py

# Predictive Analysis (Classification)



- Four models were built to predict if the launch outcomes would be successful or not.
- As it was a binary classification problem the chosen models were: Logistic Regression, SVM, Decision Tree and KNN.
- A grid search was performed for each model to find the optimum hyperparameters and the confusion matrix was displayed.
- The accuracy for the test set was evaluated in each case.



https://github.com/viquiriglos/IBM-CapstoneProject/SpaceX_Machine Learning Prediction_Part_5.ipynb

# Results

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

Section 2

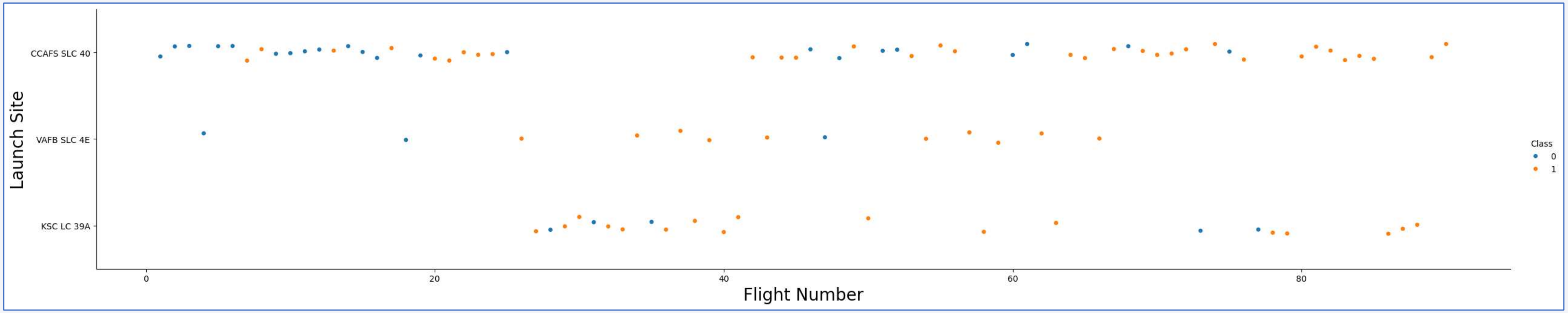
# Insights drawn from EDA

# Flight Number vs. Launch Site



Use the function `catplot` to plot `FlightNumber` vs `LaunchSite`, set the parameter `x` parameter to `FlightNumber`,set the `y` to `Launch Site` and set the parameter `hue` to `'class'`

```
sns.catplot(y="LaunchSite", x="FlightNumber", hue="Class", data=df, aspect = 5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```

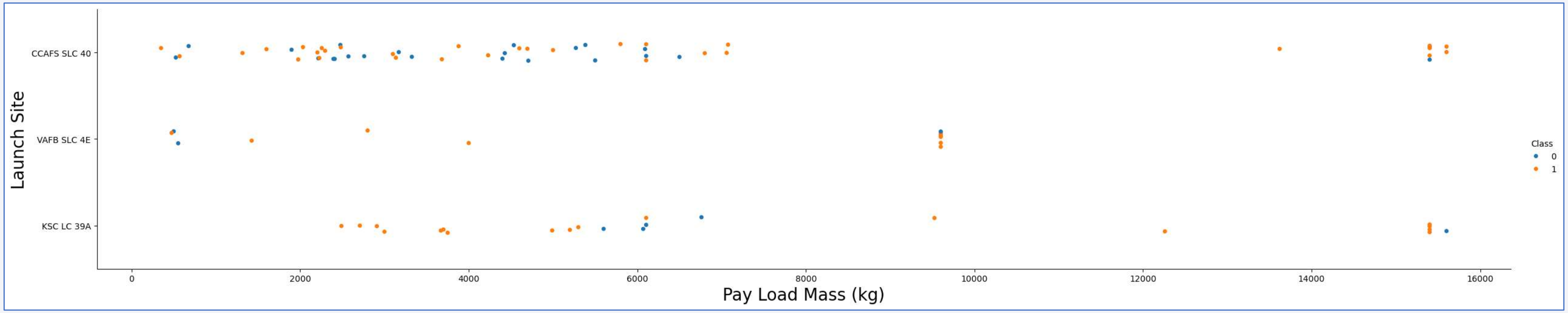


# Payload vs. Launch Site



We also want to observe if there is any relationship between launch sites and their payload mass.

```
# Plot a scatter point chart with x axis to be Pay Load Mass (kg) and y axis to be the lau
sns.catplot(y="LaunchSite", x="PayloadMass", hue="Class", data=df, aspect = 5)
plt.xlabel("Pay Load Mass (kg)",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



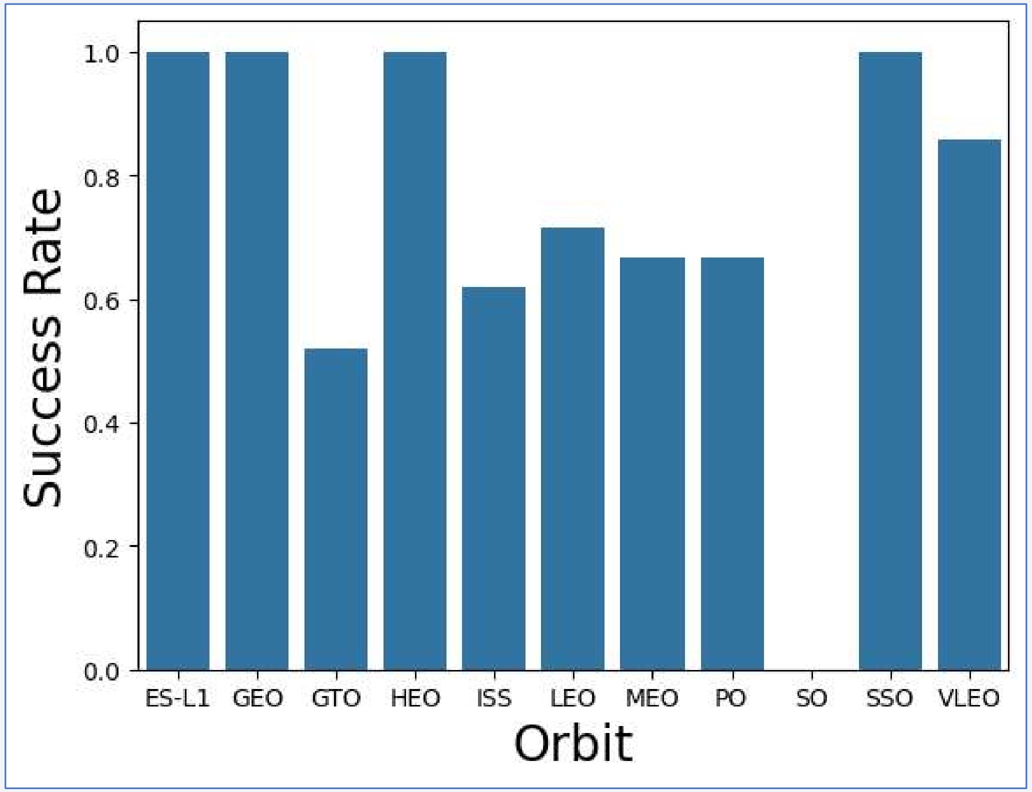
# Success Rate vs. Orbit Type



Next, we want to visually check if there are any relationship between success rate and orbit type.

```
df2=df.groupby("Orbit").mean("Class")

sns.barplot(y="Class", x="Orbit", data=df2)
plt.xlabel("Orbit",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```

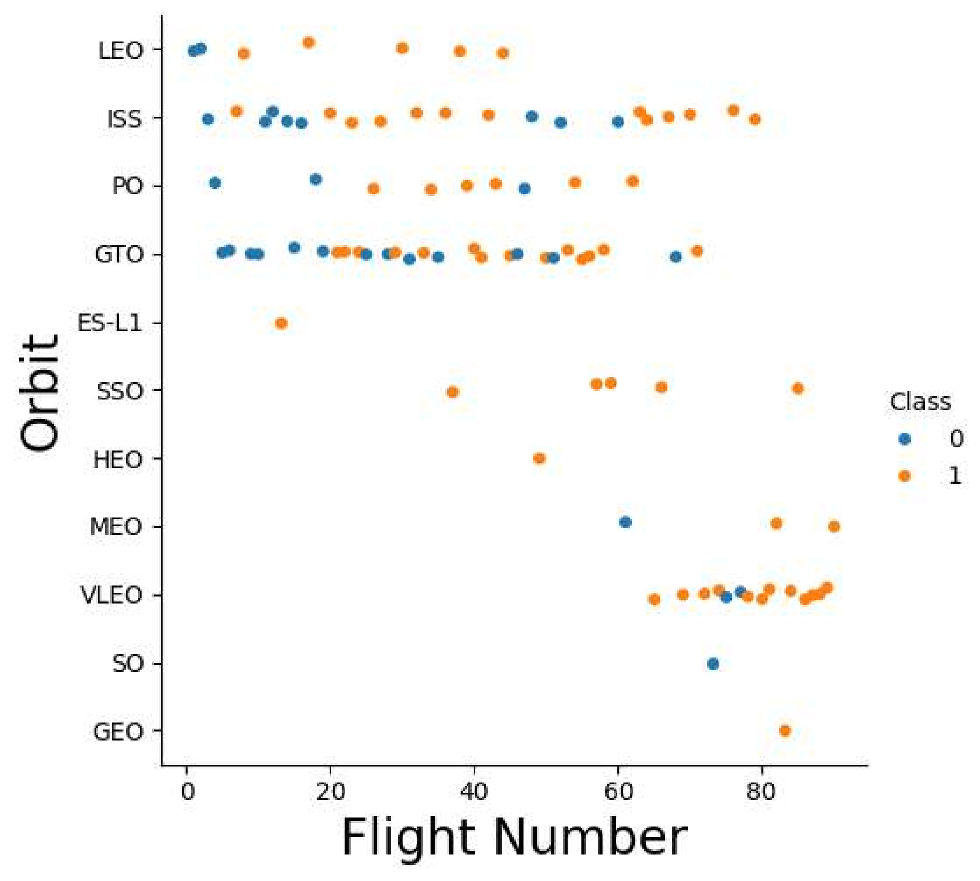


# Flight Number vs. Orbit Type



For each orbit, we want to see if there is any relationship between FlightNumber and Orbit type.

```
sns.catplot(y="Orbit", x="FlightNumber", hue="Class", data=df)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```

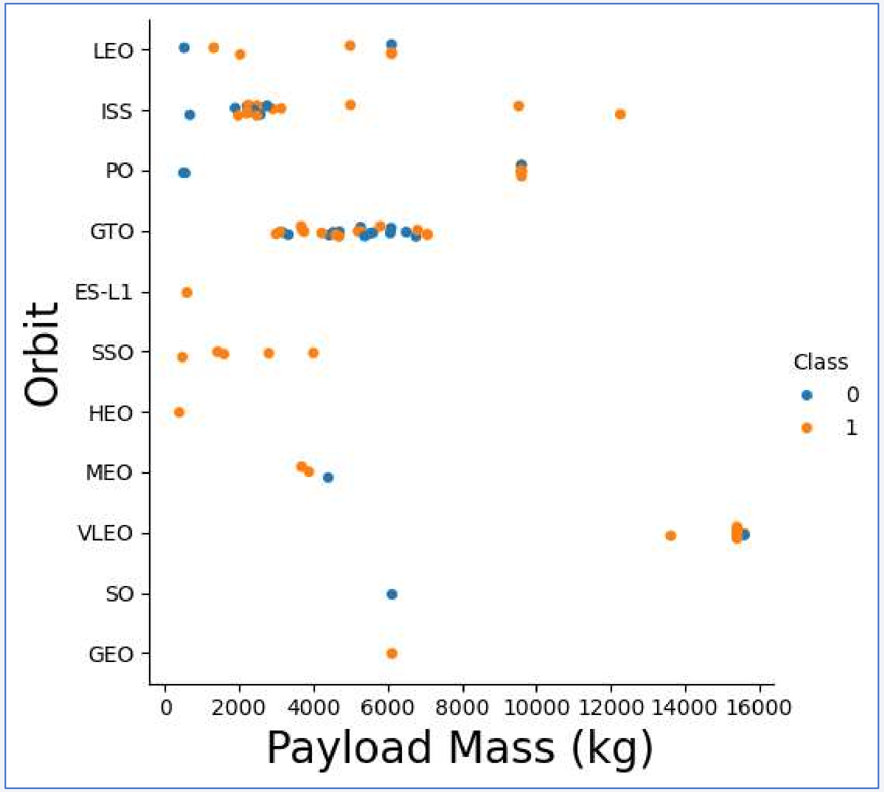


# Payload vs. Orbit Type



Similarly, we can plot the Payload Mass vs. Orbit scatter point charts to reveal the relationship between Payload Mass and Orbit type

```
sns.catplot(y="Orbit", x="PayloadMass", hue="Class", data=df)
plt.xlabel("Payload Mass (kg)",fontsize=20)
plt.ylabel("Orbit",fontsize=20)
plt.show()
```



# Launch Success Yearly Trend



Helper function:

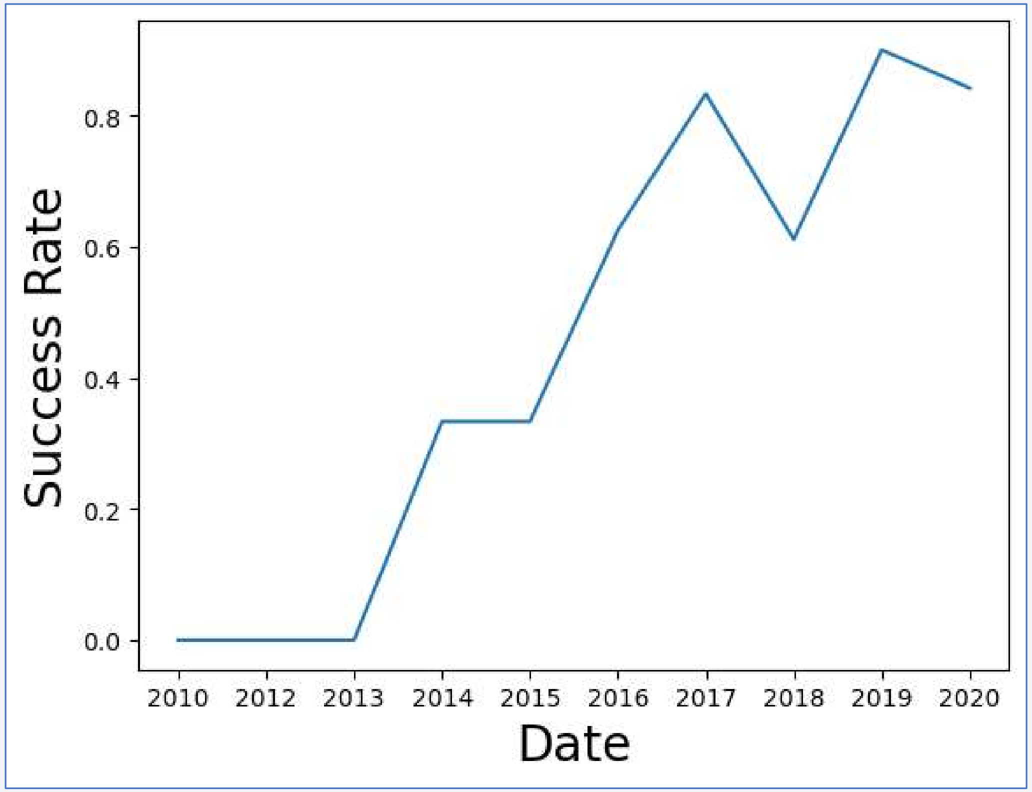
```
year=[]
def Extract_year():
    for i in df["Date"]:
        year.append(i.split("-")[0])
    return year
Extract_year()
df['Date'] = year
df.head()
```

Grouping:

```
df3=df.groupby("Date").mean("Class")
df3
```

Plotting:

```
sns.lineplot(y="Class", x="Date", data=df3)
plt.xlabel("Date",fontsize=20)
plt.ylabel("Success Rate",fontsize=20)
plt.show()
```



# All Launch Site Names



- To obtain every different site name present in the "Launch Site" Column from the SPACEXTABLE table we use the word DISTINCT placed right before the column name.

```
%sql select DISTINCT "Launch_Site" from SPACEXTABLE
```

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

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

# Launch Site Names Begin with 'CCA'



To find 5 records where launch sites begin with `CCA` we need to use the wildcard % to search all the rows starting with CCA in the "Launch Site" column. Then we limit the search to just 5 records.

```
%sql select * from SPACEXTABLE where "Launch_Site" LIKE 'CCA%' LIMIT 5;
```

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

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 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 (parachute) |
| 2012-05-22 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

# Total Payload Mass



To calculate the total payload carried by boosters from NASA we select all the rows in "PAYLOAD_MASS_KG_" column where the "Customer" column value is "NASA (CRS)".

```
%sql select "PAYLOAD_MASS__KG_" from SPACEXTABLE where "Customer"='NASA (CRS)';
```

| PAYLOAD_MASS__KG_ |
|---:|
| 500 |
| 677 |
| 2296 |
| 2216 |
| 2395 |
| 1898 |
| 1952 |
| 3136 |
| 2257 |
| 2490 |
| 2708 |
| 3310 |
| 2205 |
| 2647 |
| 2697 |
| 2500 |
| 2495 |
| 2268 |
| 1977 |
| 2972 |

# Average Payload Mass by F9 v1.1



To calculate the average payload mass carried by booster version F9 v1.1, we need to group all the rows corresponding to each booster version and then average the payload mass in each group

```
%sql select "Booster_Version", AVG("PAYLOAD_MASS__KG_") from SPACEXTABLE GROUP BY ("Booster_Version");
```

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

| Booster_Version | AVG(PAYLOAD_MASS__KG_) |
|---|---|
| F9 v1.0 B0005 | 525.0 |
| F9 v1.0 B0006 | 500.0 |
| F9 v1.0 B0007 | 677.0 |
| F9 v1.1 | 2928.4 |
| F9 v1.1 B1003 | 500.0 |
| F9 v1.1 B1010 | 2216.0 |
| F9 v1.1 B1011 | 4428.0 |

# First Successful Ground Landing Date



To find the dates of the first successful landing outcome on ground pad we searched for all the rows where the value in the “Landing_Outcome” column is equal to “Success (ground pad)” and then we ordered them by date, in ascending order (default). The first record is what we are looking for.

```
%sql select * from SPACEXTABLE where "Landing_Outcome"='Success (ground pad)' ORDER BY "Date";
```

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

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2015-12-22 | 1:29:00 | F9 FT B1019 | CCAFS LC-40 | OG2 Mission 2 11 Orbcomm-OG2 satellites | 2034 | LEO | Orbcomm | Success | Success (ground pad) |
| 2016-07-18 | 4:45:00 | F9 FT B1025.1 | CCAFS LC-40 | SpaceX CRS-9 | 2257 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 2017-02-19 | 14:39:00 | F9 FT B1031.1 | KSC LC-39A | SpaceX CRS-10 | 2490 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 2017-05-01 | 11:15:00 | F9 FT B1032.1 | KSC LC-39A | NROL-76 | 5300 | LEO | NRO | Success | Success (ground pad) |
| 2017-06-03 | 21:07:00 | F9 FT B1035.1 | KSC LC-39A | SpaceX CRS-11 | 2708 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 2017-08-14 | 16:31:00 | F9 B4 B1039.1 | KSC LC-39A | SpaceX CRS-12 | 3310 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 2017-09-07 | 14:00:00 | F9 B4 B1040.1 | KSC LC-39A | Boeing X-37B OTV-5 | 4990 | LEO | U.S. Air Force | Success | Success (ground pad) |
| 2017-12-15 | 15:36:00 | F9 FT B1035.2 | CCAFS SLC-40 | SpaceX CRS-13 | 2205 | LEO (ISS) | NASA (CRS) | Success | Success (ground pad) |
| 2018-01-08 | 1:00:00 | F9 B4 B1043.1 | CCAFS SLC-40 | Zuma | 5000 | LEO | Northrop Grumman | Success (payload status unclear) | Success (ground pad) |

# Successful Drone Ship Landing with Payload between 4000 and 6000



In order to list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 we've searched for all the records in the "Landing_Outcome" column having the value "Success (drone ship)", and adding the condition that the value in the"PAYLOAD_MASS_KG_" column was between 4001 and 5999, so we make sure to exclude the values at the beginning and at the end of the range.

```
%sql select * from SPACEXTABLE where "Landing_Outcome"='Success (drone ship)' AND "PAYLOAD_MASS__KG_" BETWEEN 4001 AND 5999;
```

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

| 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) |

# Total Number of Successful and Failure Mission Outcomes



To calculate the total number of successful and failure mission outcomes it was performed a query that group the records by their "Landing_Outcome" value and count the number of records in each group, using the COUNT function.

```
%sql SELECT "Landing_Outcome", COUNT("Landing_Outcome") FROM SPACEXTABLE GROUP BY("Landing_Outcome");
```

| Landing_Outcome | COUNT(Landing_Outcome) |
|---|---|
| Controlled (ocean) | 5 |
| Failure | 3 |
| Failure (drone ship) | 5 |
| Failure (parachute) | 2 |
| No attempt | 21 |
| No attempt | 1 |
| Precluded (drone ship) | 1 |
| Success | 38 |
| Success (drone ship) | 14 |
| Success (ground pad) | 9 |
| Uncontrolled (ocean) | 2 |

# Boosters Carried Maximum Payload



```
%sql SELECT "Booster_Version", PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);
```

To list the names of the booster which have carried the maximum payload mass we've performed a subquery to search the maximum value in the "PAYLOAD_MASS_KG_" column and then we've used this value as part of a WHERE clause to find the records having their values in the "PAYLOAD_MASS_KG_" column equal to the value obtained from the subquery.

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

# 2015 Launch Records

To list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015 we've performed a query setting that the value in the "Landing_Outcome" column must be equal to "Failure (drone ship)" and using the % wildcard to state that the "Date" column should start with the year 2015, despite the values of month and day.

```
%sql SELECT * FROM SPACEXTABLE WHERE "Landing_Outcome"= 'Failure (drone ship)' AND "Date" LIKE '2015%';

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

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2015-01-10 | 9:47:00 | F9 v1.1 B1012 | CCAFS LC-40 | SpaceX CRS-5 | 2395 | LEO (ISS) | NASA (CRS) | Success | Failure (drone ship) |
| 2015-04-14 | 20:10:00 | F9 v1.1 B1015 | CCAFS LC-40 | SpaceX CRS-6 | 1898 | LEO (ISS) | NASA (CRS) | Success | Failure (drone ship) |

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



```
%sql select "Landing_Outcome", COUNT("Landing_Outcome") from SPACEXTABLE WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY "Landing_Outcome" ORDER BY COUNT("Landing_Outcome") DESC;
```

To obtain the ranking of the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order, a query was performed grouping by the values inside the “Landing_Outcome” column and countin the values in each group. Additionally, the values inside the “Date” have been set according to the required period of time. Finally, the list has been ordered in descending order by the number of landings in each of the groups obtained from the landing outcomes.

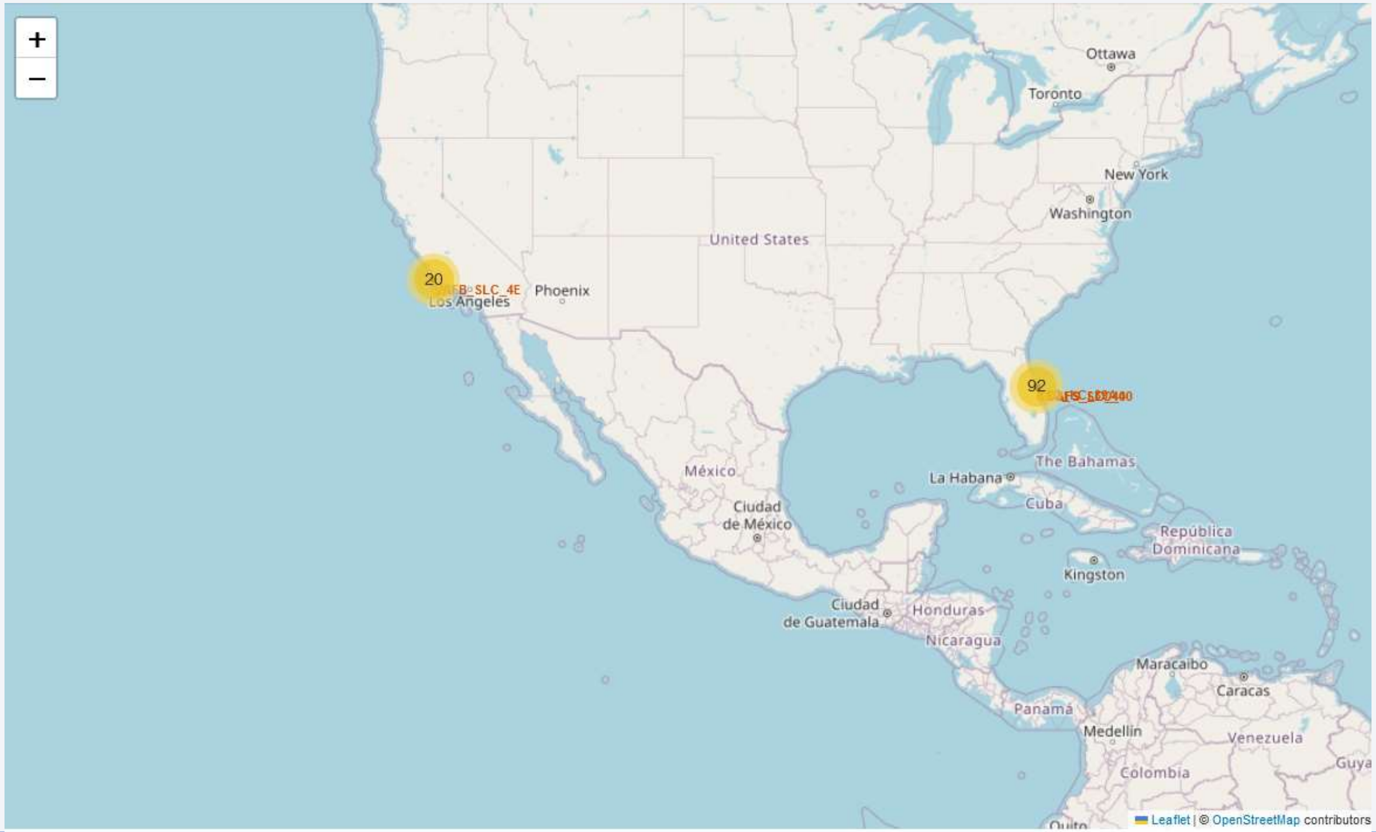
| Landing_Outcome | COUNT(Landing_Outcome) |
|---|---|
| No attempt | 10 |
| Success (drone ship) | 5 |
| Failure (drone ship) | 5 |
| Success (ground pad) | 3 |
| Controlled (ocean) | 3 |
| Uncontrolled (ocean) | 2 |
| Failure (parachute) | 2 |
| Precluded (drone ship) | 1 |

Section 3

# Launch Sites Proximities Analysis

# Global Map of the Launch Sites






From this global chart it can be observed that most of the launches have been carried out in the launch sited located in US eastern coast.

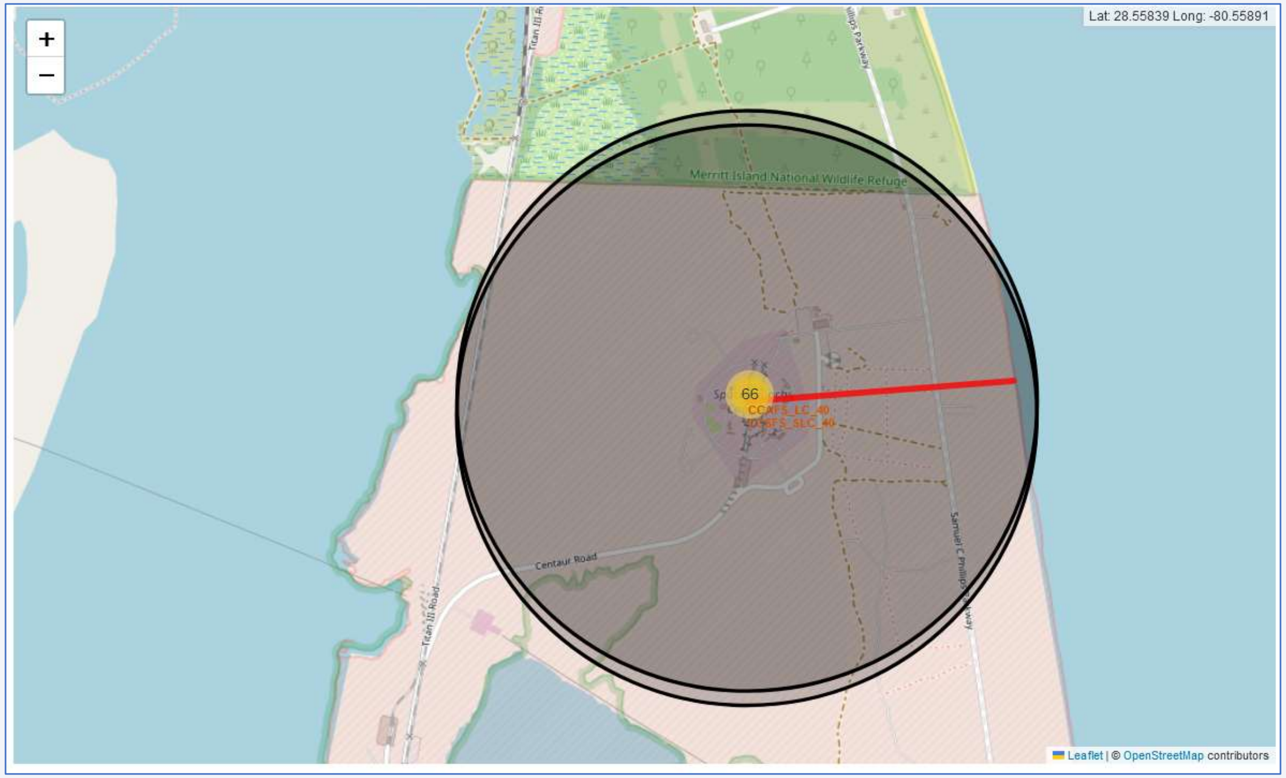
# Detail of East Coast Lauch Sites



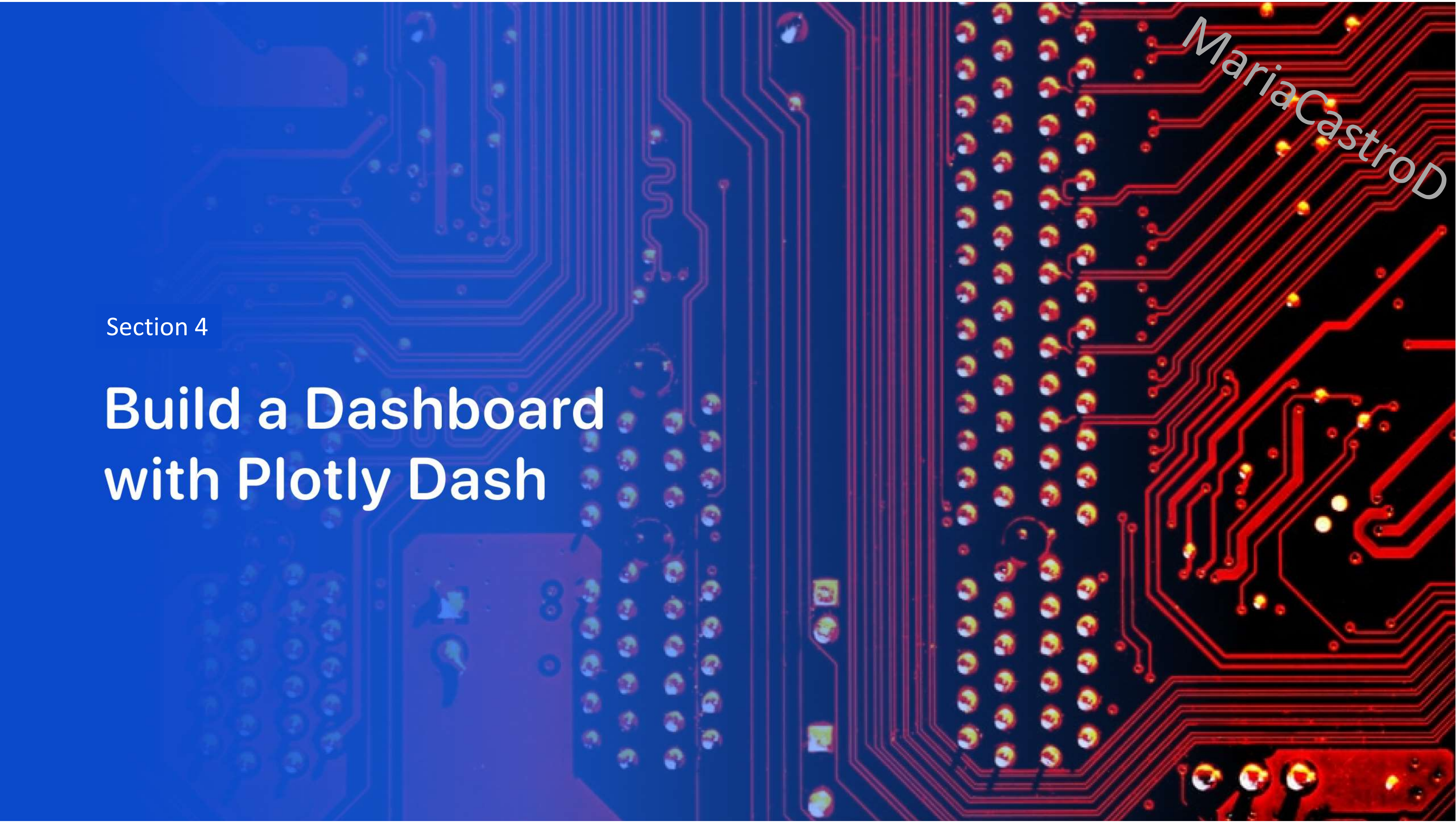

Zooming in, markers can explain how lauches are distributed for each launch site

# Distance to the Coast






The distance from the chosen Launch Site to any feature in the map can be highlighted and calculated, as well. In this case, it is displayed the line that connect one of the launch sites to the coast.

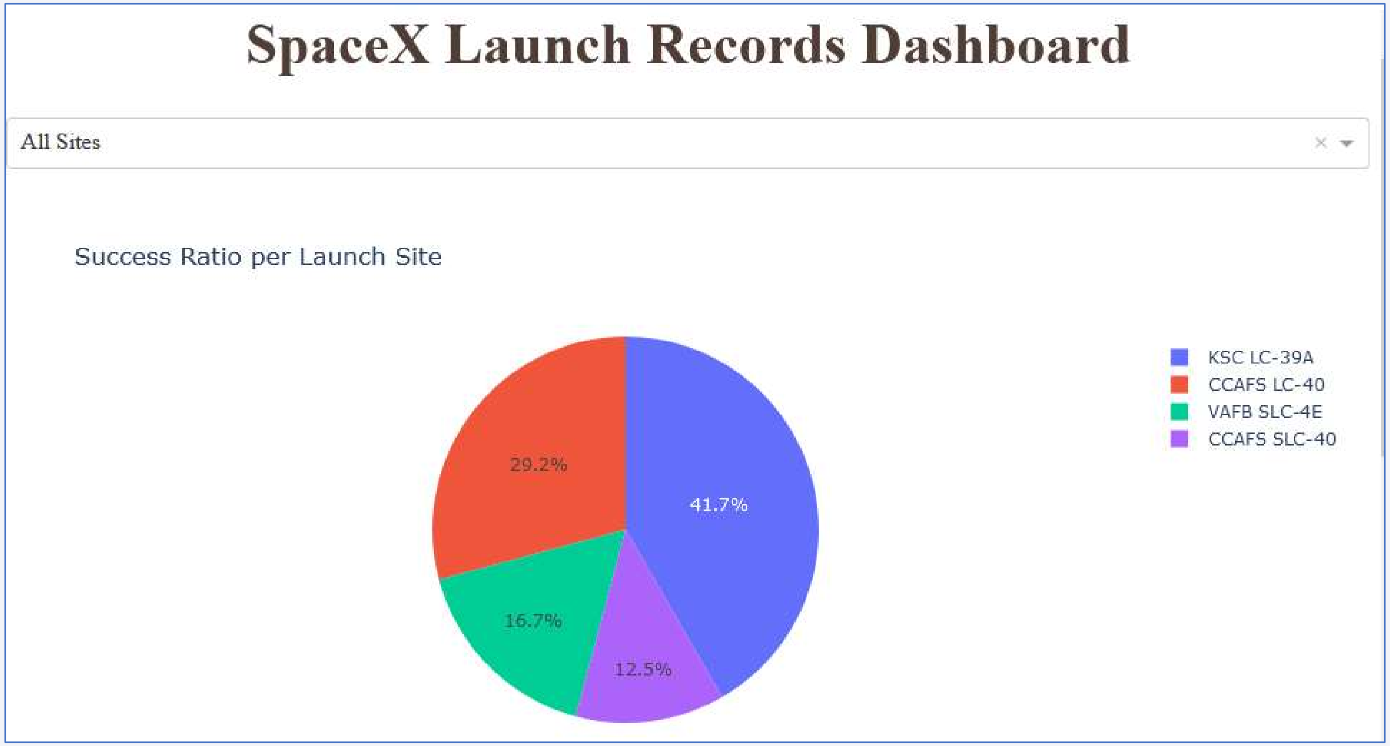


Section 4

# Build a Dashboard with Plotly Dash

# Launch success count for each site

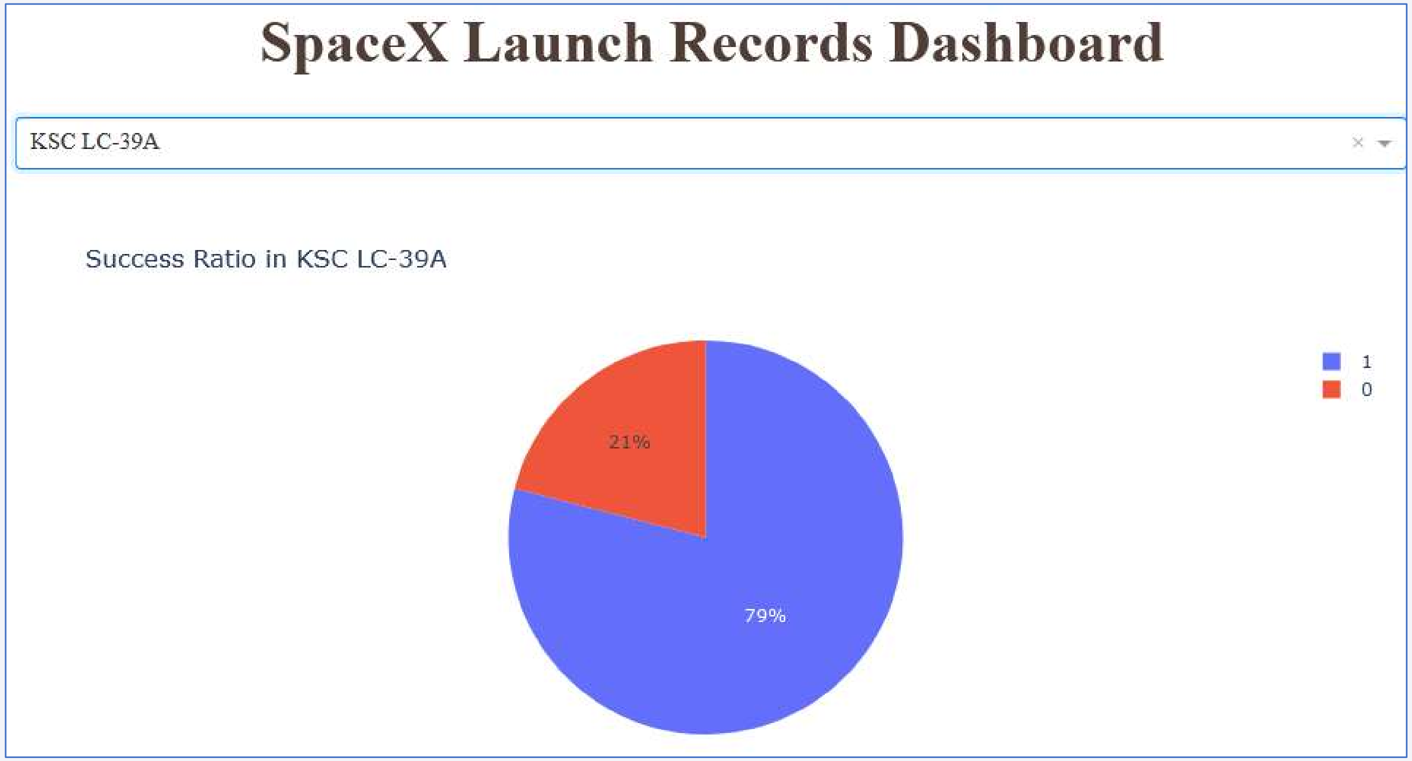





It is observed that most of the successful launches occurred the site named "KSC LC-39A" and that the less successful site was "CCAFS SLC-40.

# Success ratio of KSC LC-39A

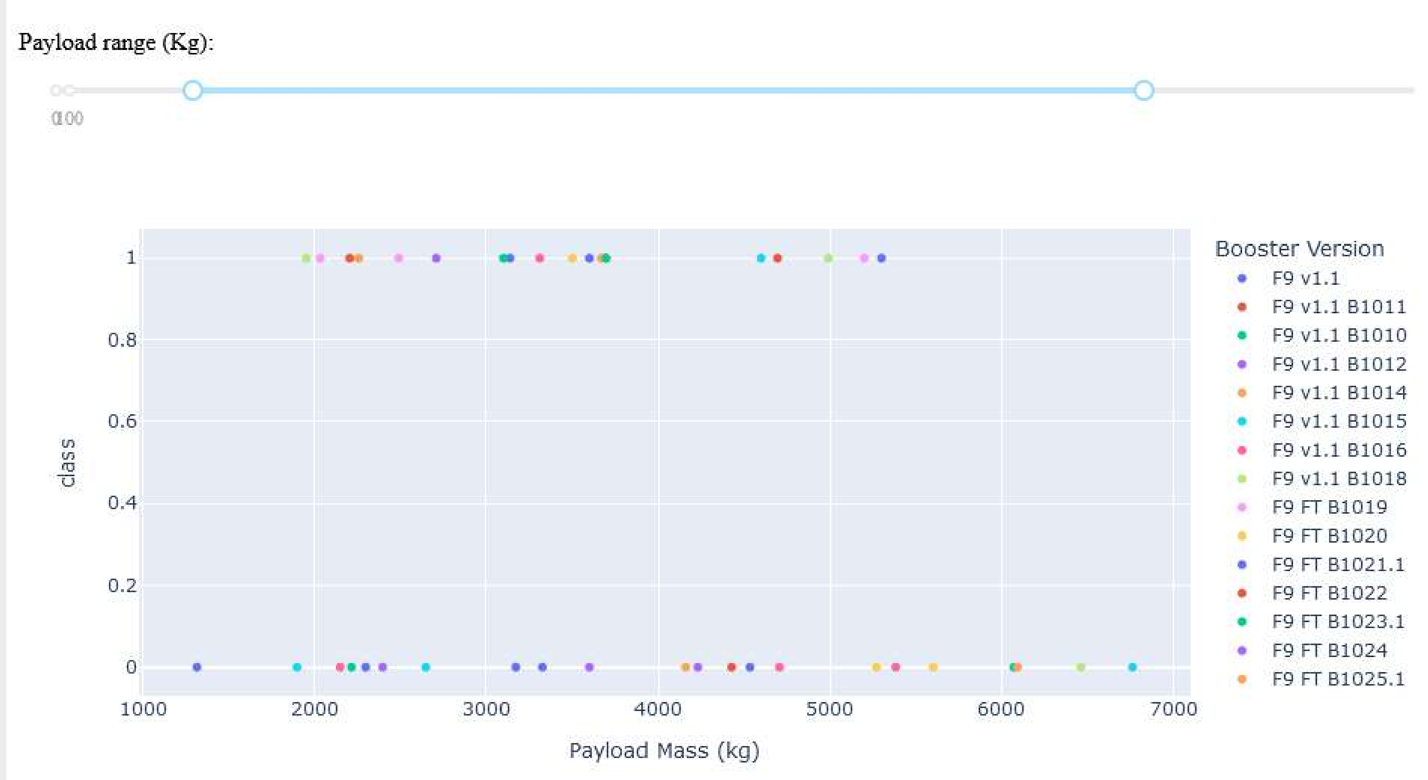





Given KSC LC-39A was the site having most successful launches counts, it can be observed in this pie chart the proportion between successful and failed launches.

# Payload vs. Launch Outcome





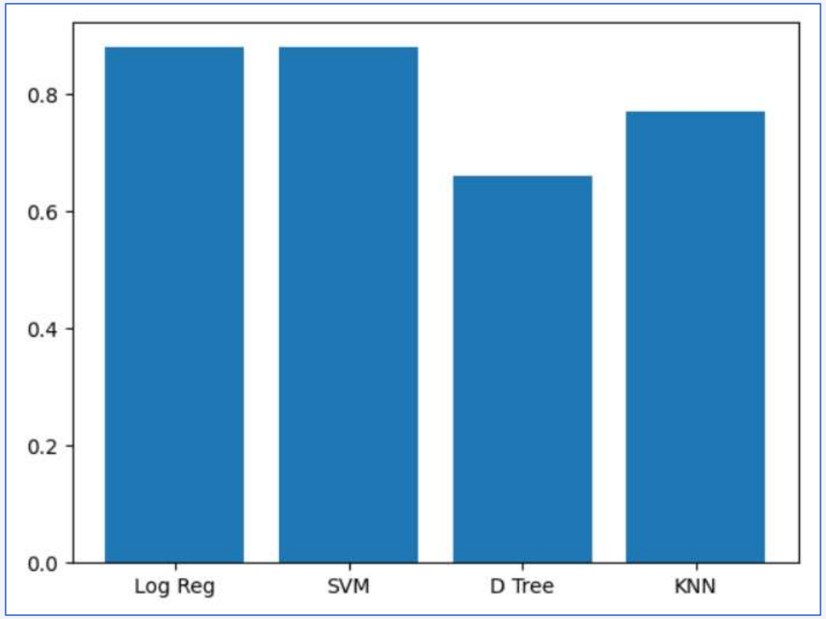

Payload vs. Launch Outcome scatter plot for all sites and for each Booster Version with different payload selected in The range slider can be used to set the limits of the range of interest in the Payload mass. It is observed that the launches have a higher tendency to fail for Payloads above 5500 kg.

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

Comparison of the accuracy obtained for the test set in every model:

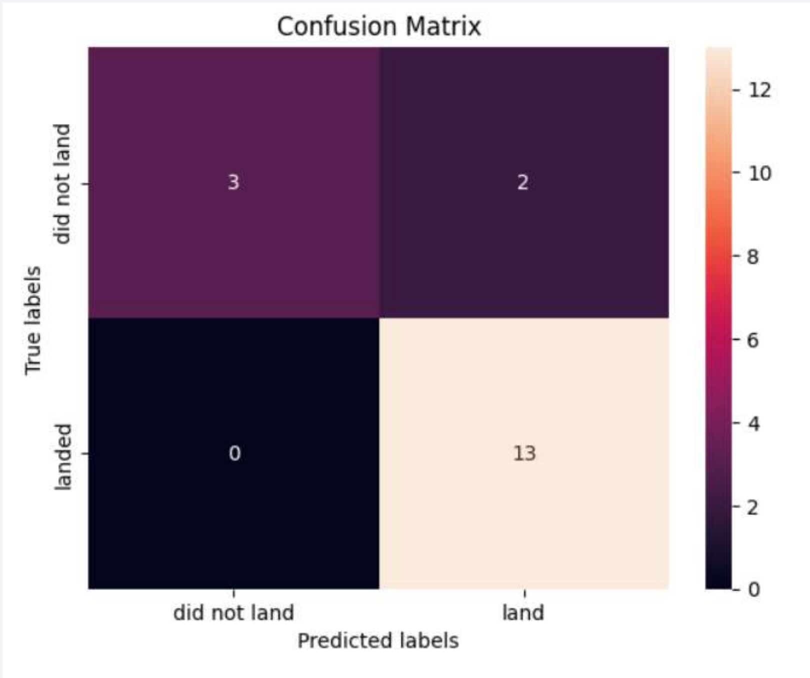



Logistic Regression and SVM were the models which performed best.

# Confusion Matrix



The best performing model was an SVM. Its confusion matrix tells us that there are 13 True Positives, 3 True Negatives and 2 False Negatives. There are no False Positives.



# Conclusions



Most significant insights obtained from EDA and SQL queries:

- The launch outcomes tend to succeed as the Flight Number increase and the time advances.
- The orbits GEO, ES-L1, HEO and SSO presented the higher success ratio.
- For lower Payloads, SSO was the orbit that performed best.
- The first successful landing outcome on ground pad occurred in 2015.

Folium and Dashboard insights:

- KSC LC-39A was the site having most successful launches counts.
- Most of the launches were carried out in the US east coast.
- launches have a higher tendency to fail for Payloads above 5500 kg

# Appendix



- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project


Thank you!