



IBM Developer  
SKILLS NETWORK

# Winning Space Race with Data Science

Pauline Likoso  
March 20th 2025



# 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

---

- Project background and context

SpaceX offers Falcon 9 rocket launches for \$62 million, while other providers charge over \$165 million. The main reason for SpaceX's lower cost is that it can reuse the first stage of the rocket. If we can predict whether the first stage will land successfully, we can estimate launch costs. This information could help other companies compete with SpaceX for rocket launch contracts. The goal of this project is to build a machine learning model to predict first-stage landings.

- 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.



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 collected using various methods
  - Data collection was done using get request to the SpaceX API.
  - We then 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 – 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/PaulineLikoso/IBMDS/blob/main/capstone\\_folder/jupyter-labs-spacex-data-collection-api%20\(1\).ipynb](https://github.com/PaulineLikoso/IBMDS/blob/main/capstone_folder/jupyter-labs-spacex-data-collection-api%20(1).ipynb)

Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [7]:

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

In [8]:

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

In [22]:

```
response=response.get(static_json_url)
```

In [23]:

```
response.status_code
```

Out[23]:

```
200
```

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

In [24]:

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

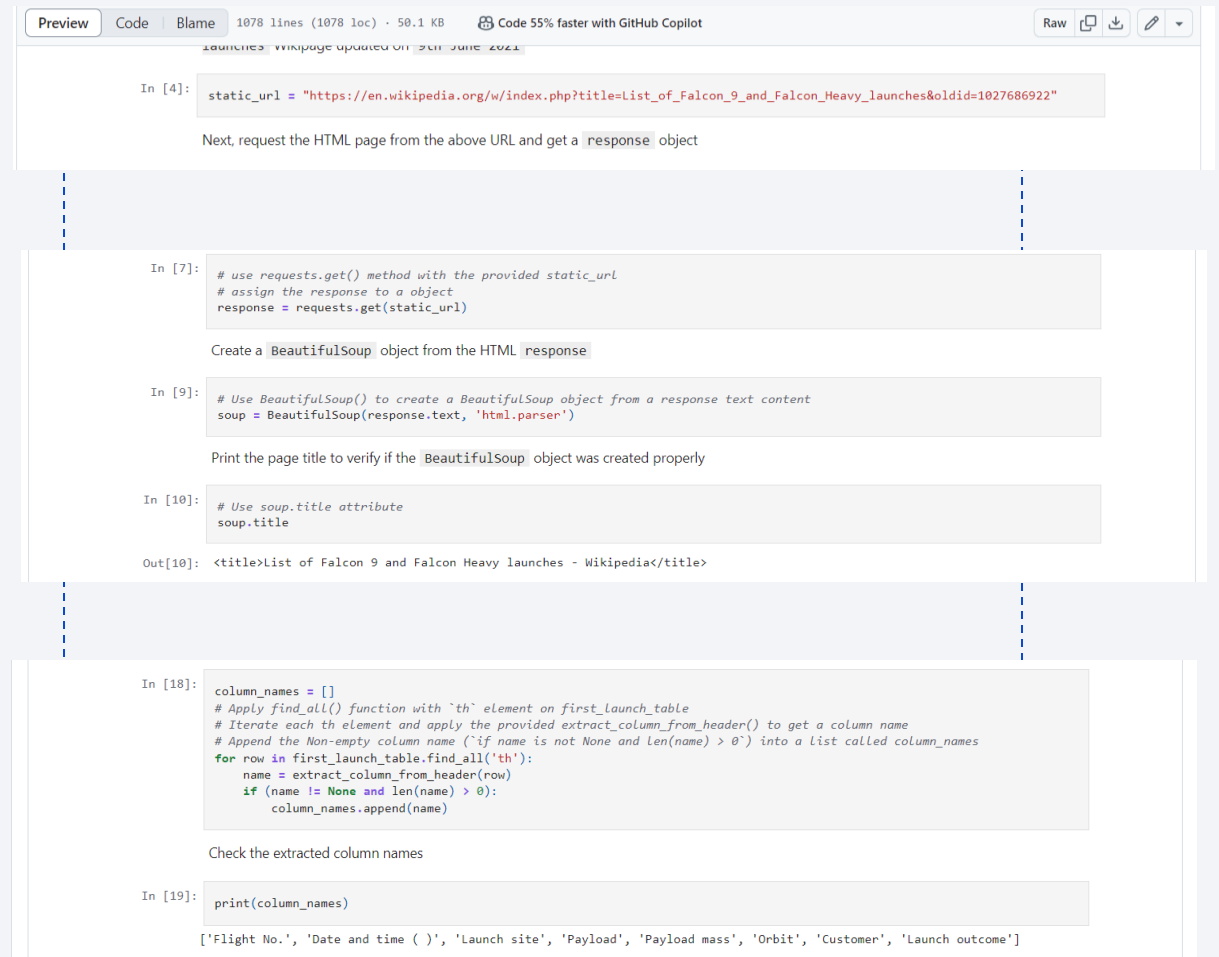
In [25]:

```
# Lets take a subset of our dataframe keeping only the features we want and the flight number  
data = data[['rocket', 'payloads', 'launchpad', 'cores', 'flight_number', 'date_utc']]  
  
# We will remove rows with multiple cores because those are falcon rockets with 2 extra rocket cores  
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 single value in the lists  
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 extracting the date only  
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.head()
```



# 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/PaulineLikoso/BMDS/blob/main/capstone\\_folder/jupyter-labs-webscraping.ipynb](https://github.com/PaulineLikoso/BMDS/blob/main/capstone_folder/jupyter-labs-webscraping.ipynb)



```
Preview Code Blame 1078 lines (1078 loc) · 50.1 KB Code 55% faster with GitHub Copilot Raw Copy Download Edit
launches webpage updated on 30th June 2021

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

Next, request the HTML page from the above URL and get a response object

In [7]: # use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)

Create a BeautifulSoup object from the HTML response

In [9]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')

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

In [10]: # Use soup.title attribute
soup.title

Out[10]: <title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>

In [18]: 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
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)

Check the extracted column names

In [19]: print(column_names)

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

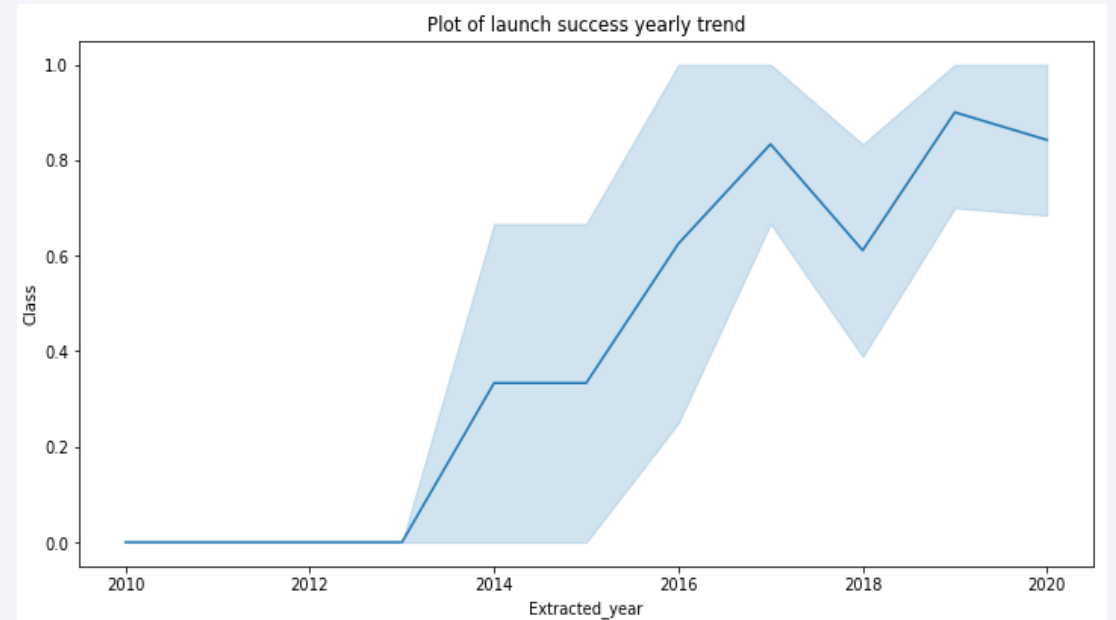
# Data Wrangling



- 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/PaulineLikoso/IBMDS/blob/main/capstone folder/labs-jupyter-spacex-Data%20wrangling.ipynb](https://github.com/PaulineLikoso/IBMDS/blob/main/capstone%20folder/labs-jupyter-spacex-Data%20wrangling.ipynb)

# EDA 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/PaulineLikoso/IBMD\\_S/blob/main/capstone folder/edadataviz.ipynb](https://github.com/PaulineLikoso/IBMD_S/blob/main/capstone%20folder/edadataviz.ipynb)

# EDA 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/PaulineLikoso/IBMDS/blob/main/capstone\\_folder/jupyter-labs-eda-sql-coursera\\_sqllite.ipynb](https://github.com/PaulineLikoso/IBMDS/blob/main/capstone_folder/jupyter-labs-eda-sql-coursera_sqllite.ipynb)

# Build an Interactive Map 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.



# 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 unavailable.

# Predictive Analysis (Classification)

---

- 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 [https://github.com/PaulineLikoso/IBMDS/blob/main/capstone folder/SpaceX Machine%20Learning%20Prediction Part 5.ipynb](https://github.com/PaulineLikoso/IBMDS/blob/main/capstone%20folder/SpaceX%20Machine%20Learning%20Prediction%20Part%205.ipynb)



The background of the slide is an abstract composition. It features a solid blue area on the left side, which transitions into a dynamic pattern of diagonal streaks in shades of blue, red, and teal on the right. These streaks are layered over a faint, grid-like pattern, creating a sense of depth and movement.

Section 2

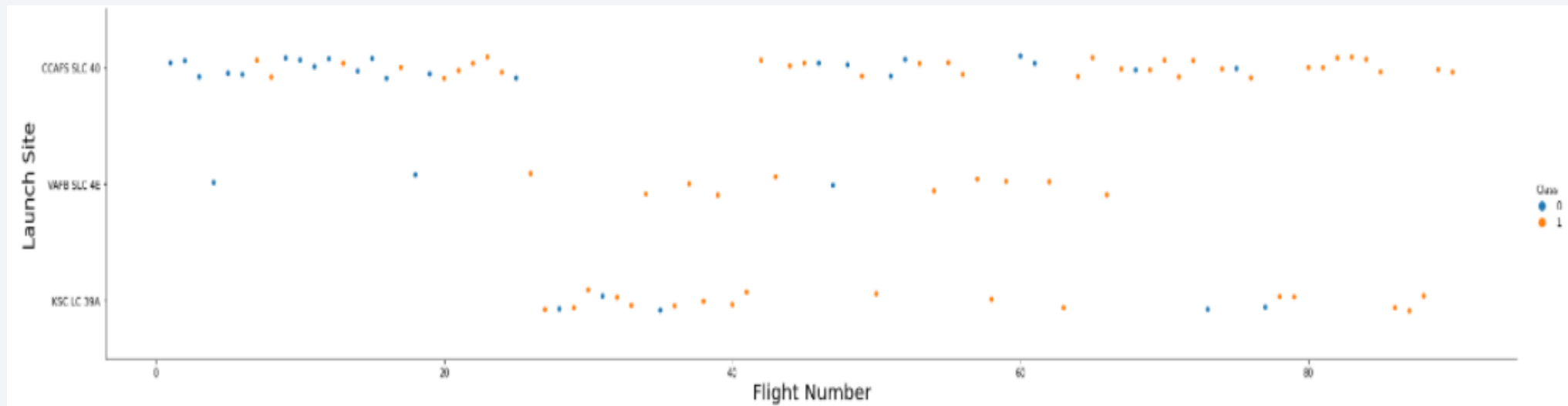
# Insights drawn from EDA



# Flight Number vs. Launch Site

---

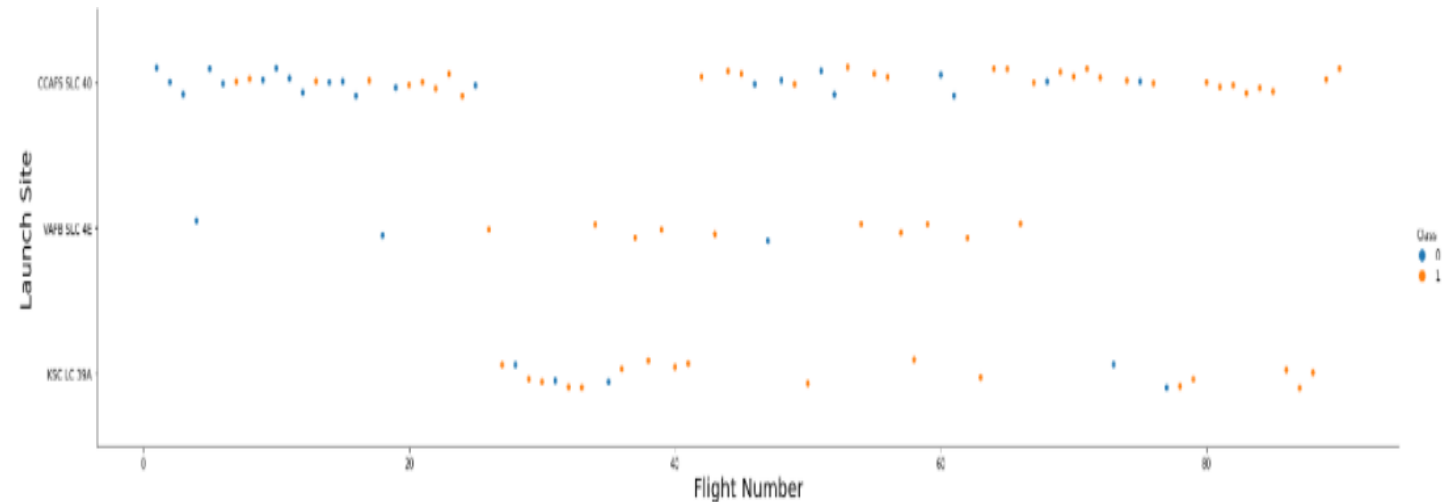
- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.



## Payload vs. Launch Site



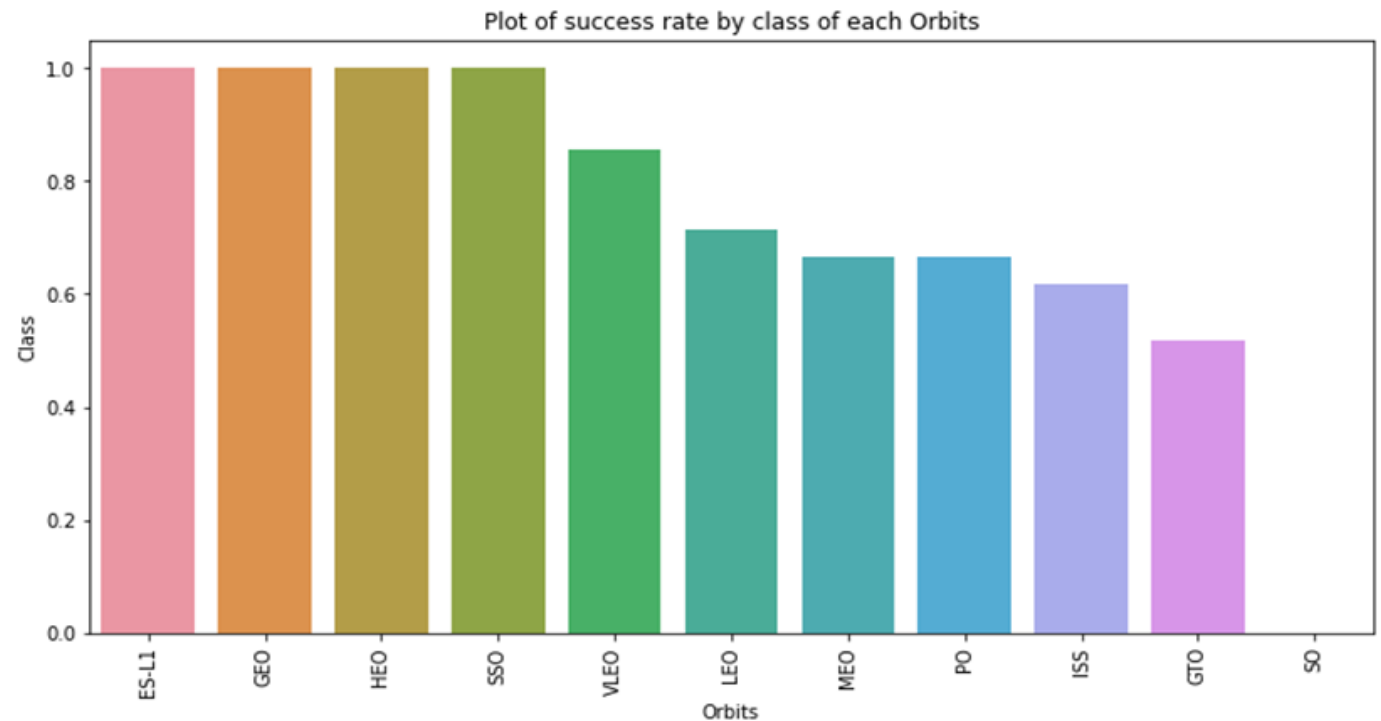
The greater the payload mass for launch site CCAFS SLC 40 the higher the success rate for the rocket.





# Success Rate vs. Orbit Type

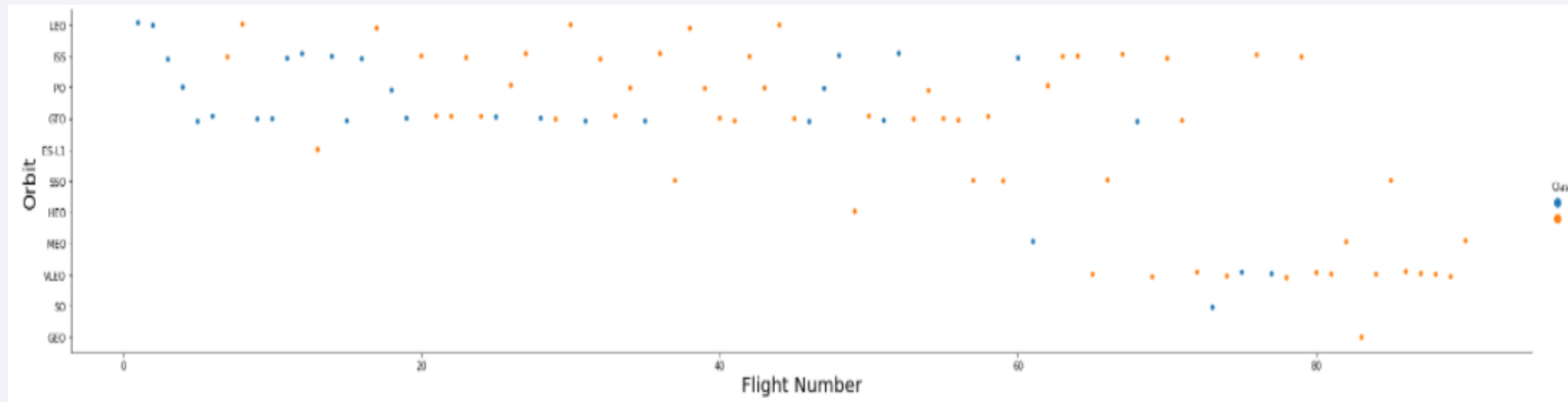
- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



# Flight Number vs. Orbit Type

---

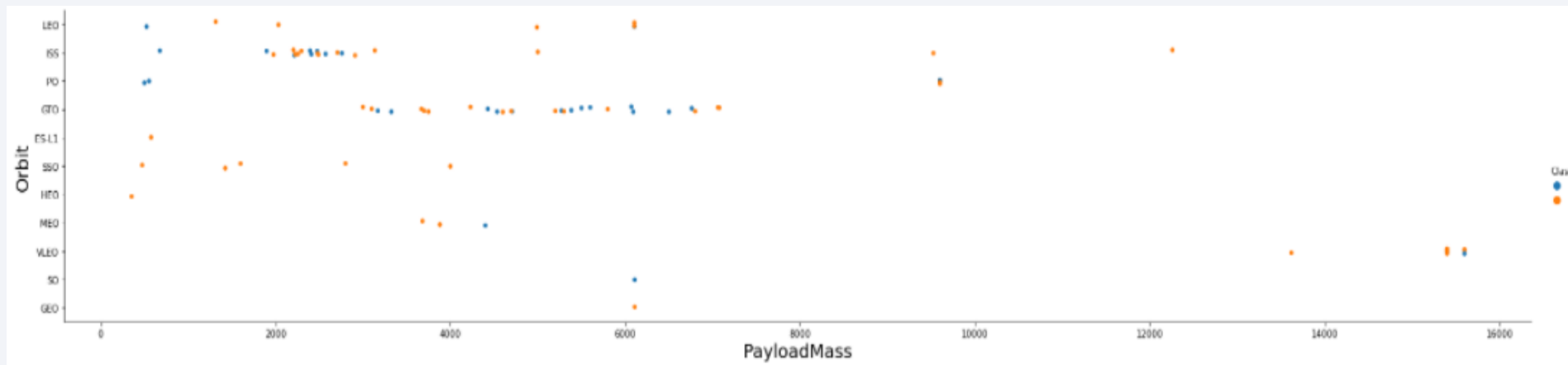
- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



# Payload vs. Orbit Type

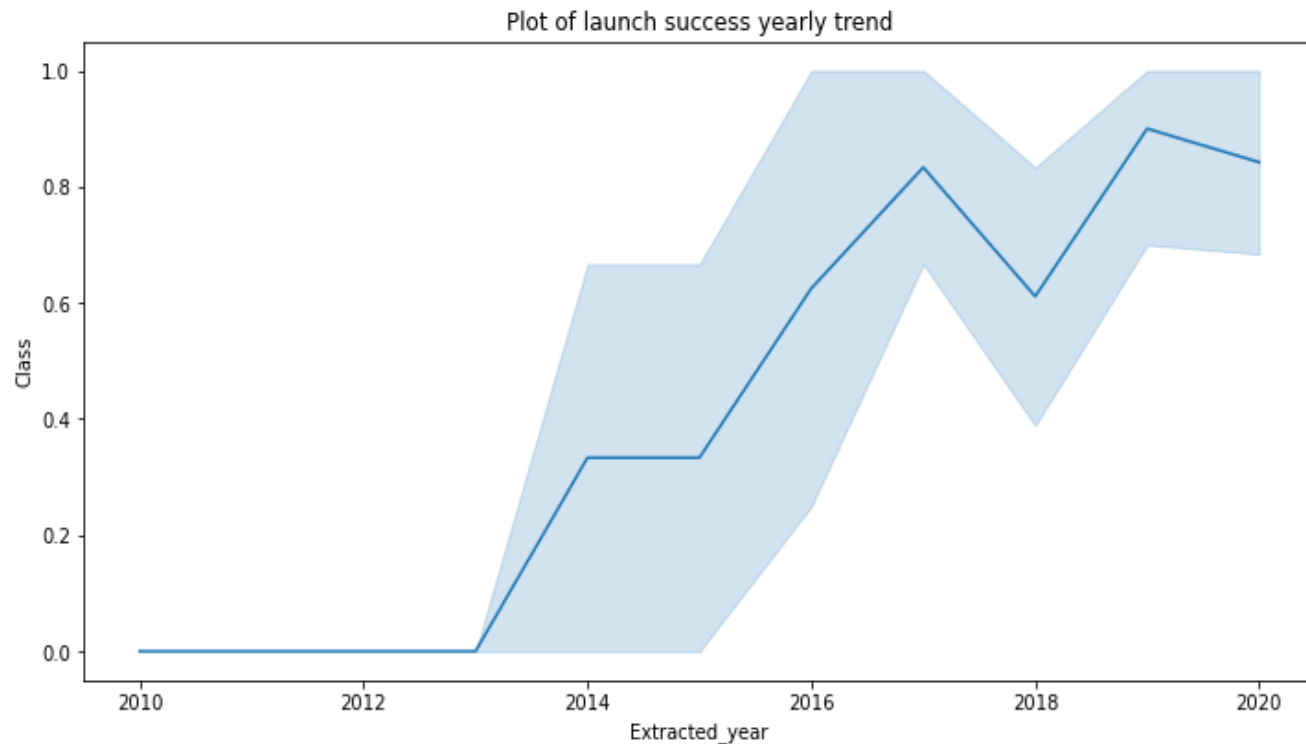
---

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



# 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.

## Task 1

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

```
In [21]: %sql select distinct "Launch_Site" from SPACEXTABLE;
```

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

```
Out[21]:
```

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



# Launch Site Names Begin with 'CCA'

**Task 2**

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

- 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

## Task 3

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

```
In [32]: %sql select sum("PAYLOAD_MASS__KG_") as "TOTAL_PLM_LAUNCHED_NASA(CRS)" from SPACEXTABLE where Customer = 'NASA (CRS)'
```

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

```
Done.
```

```
Out[32]: TOTAL_PLM_LAUNCHED_NASA(CRS)
```

```
45596
```

# Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2534.67

## Task 4

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

```
In [31]: %sql select avg("PAYLOAD_MASS_KG_") as "AVG_PLM_CARRIED_BY_F9V1.1" from SPACEXTABLE where "Booster_Version" like "F9 v1.1%"

* sqlite:///my_data1.db
Done.

Out[31]: AVG_PLM_CARRIED_BY_F9V1.1
          2534.6666666666665
```

# First Successful Ground Landing Date

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

## Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

*Hint: Use min function*

```
In [35]: %sql select Date from SPACEXTABLE where "Landing_Outcome"='Success (ground pad)' order by Date limit 1
```

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

```
Out[35]:
```

Date
------

2015-12-22
------------

# Successful Drone Ship Landing with Payload between 4000 and 6000

## Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

```
In [37]: %sql select "Booster_Version" from SPACEXTABLE where "Landing_Outcome"='Success (drone ship)' and "PAYLOAD_MASS__KG_" between 4000 and 6000
```

\* sqlite:///my\_data1.db  
Done.

```
Out[37]: Booster_Version  
F9 FT B1022  
F9 FT B1026  
F9 FT B1021.2  
F9 FT B1031.2
```

- 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



# Total Number of Successful and Failure Mission Outcomes

## Task 7

List the total number of successful and failure mission outcomes

In [42]:

```
%%sql
SELECT
  CASE
    WHEN "Mission_Outcome" LIKE 'Success%' THEN 'Success'
    WHEN "Mission_Outcome" LIKE 'Failure%' THEN 'Failure'
  END AS Outcome,
  COUNT(*) AS Total_Count
FROM SPACEXTABLE
WHERE "Mission_Outcome" LIKE 'Success%' OR "Mission_Outcome" LIKE 'Failure%'
GROUP BY Outcome;
```

\* sqlite:///my\_data1.db  
Done.

Out[42]:

Outcome	Total_Count
Failure	1
Success	100

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

# 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.

```
In [45]: %%sql
SELECT "Booster_Version"
FROM SPACEXTABLE
WHERE "PAYLOAD_MASS_KG_" = (
    SELECT MAX("PAYLOAD_MASS_KG_")
    FROM SPACEXTABLE
)
ORDER BY "Booster_Version";
```

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

```
Out[45]: Booster_Version
```

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

# 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

## Task 9

List the records which will display the month names, failure landing\_outcomes in drone ship ,booster versions, launch\_site for the months in year 2015.

**Note:** SQLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

```
In [46]: %%sql
SELECT "Booster_Version", "Launch_Site", "Landing_Outcome"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE 'Failure (drone ship)'
AND Date BETWEEN '2015-01-01' AND '2015-12-31'
```

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

```
Out[46]:
```

Booster_Version	Launch_Site	Landing_Outcome
F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

## Task 10

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.

```
In [47]: %%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
```

\* sqlite:///my\_data1.db  
Done.

```
Out[47]:
```

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

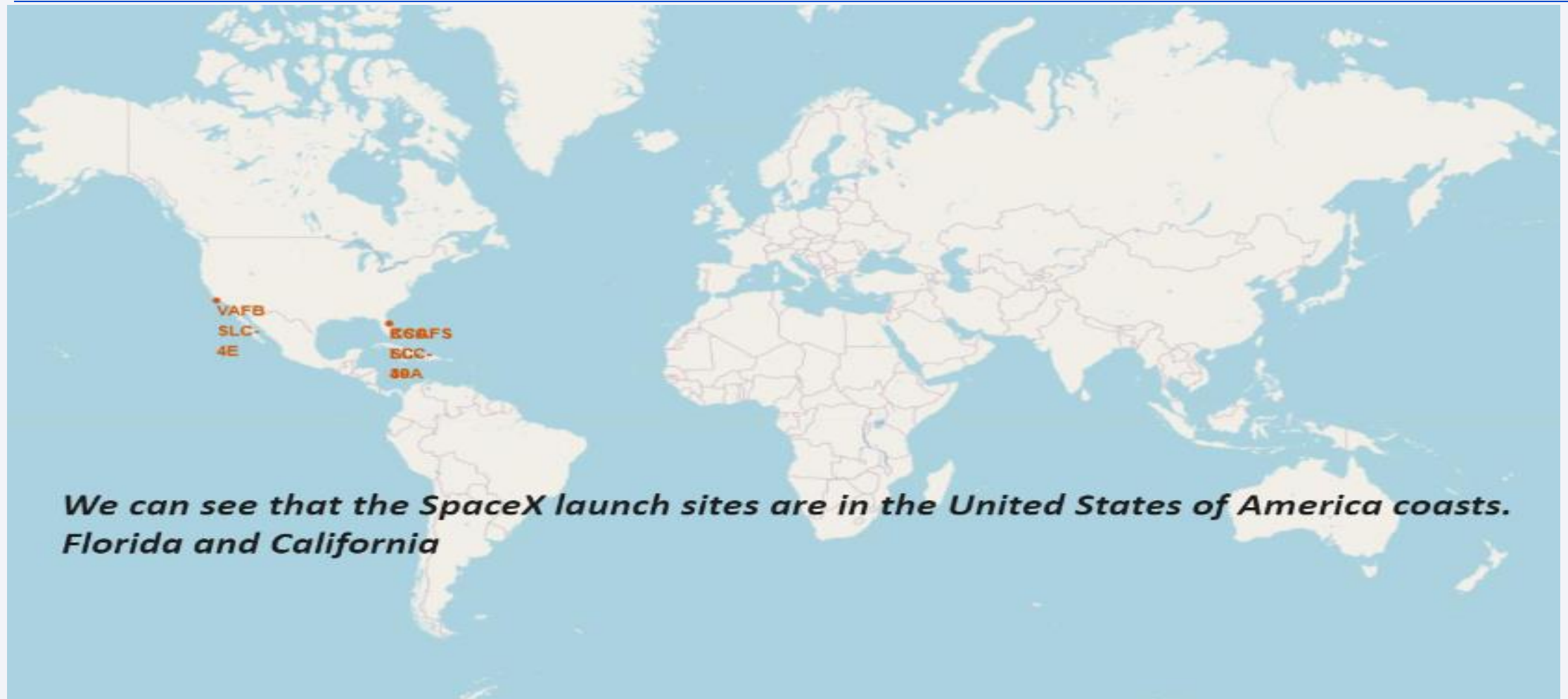
- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2017-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Section 4

# Launch Sites Proximities Analysis

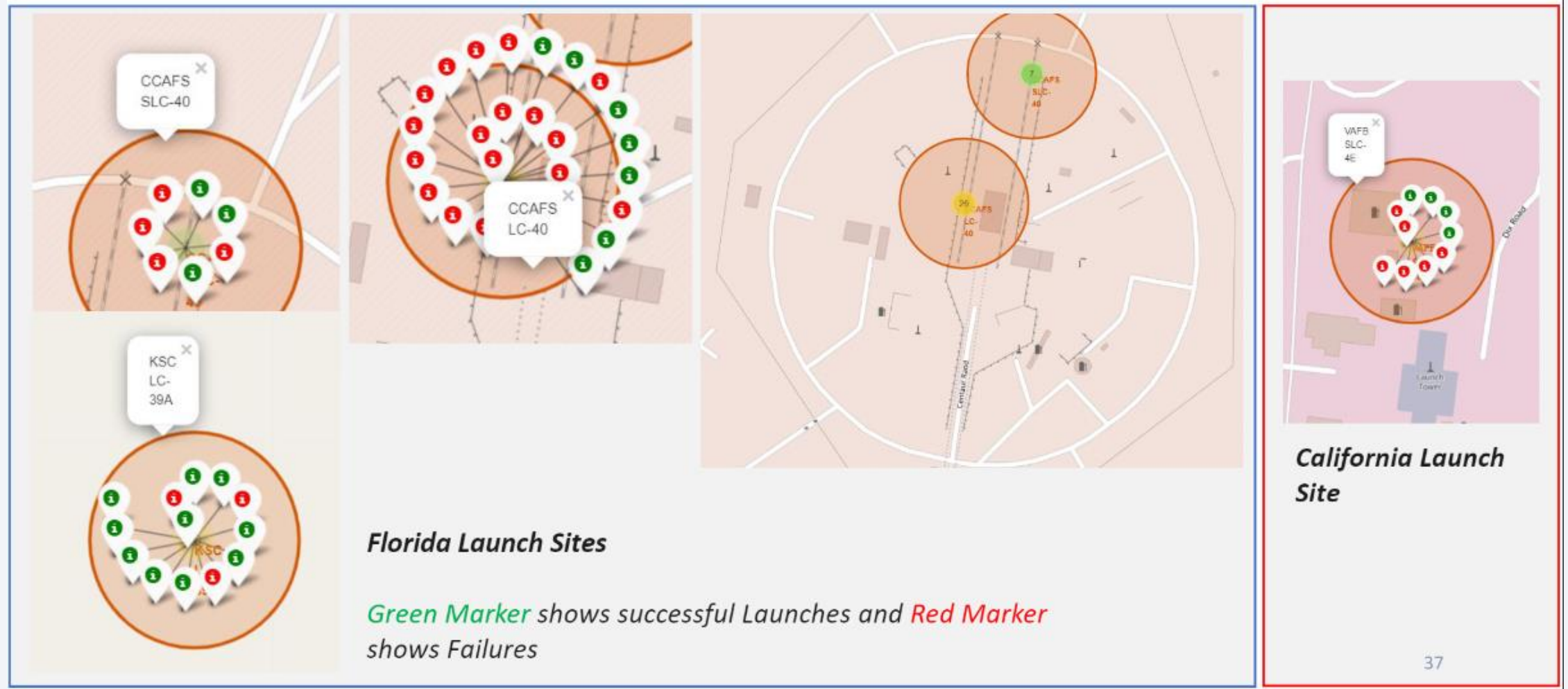


# All launch sites global map markers



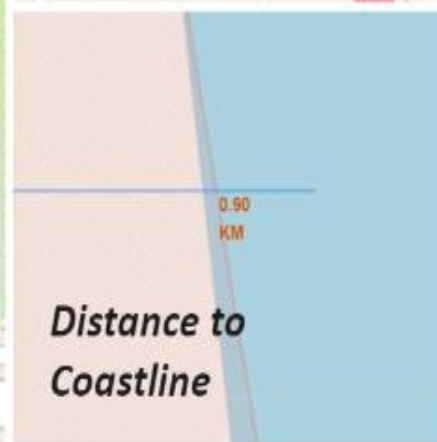


# Markers showing launch sites with color labels





# 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





Section 5

# Build a Dashboard with Plotly Dash

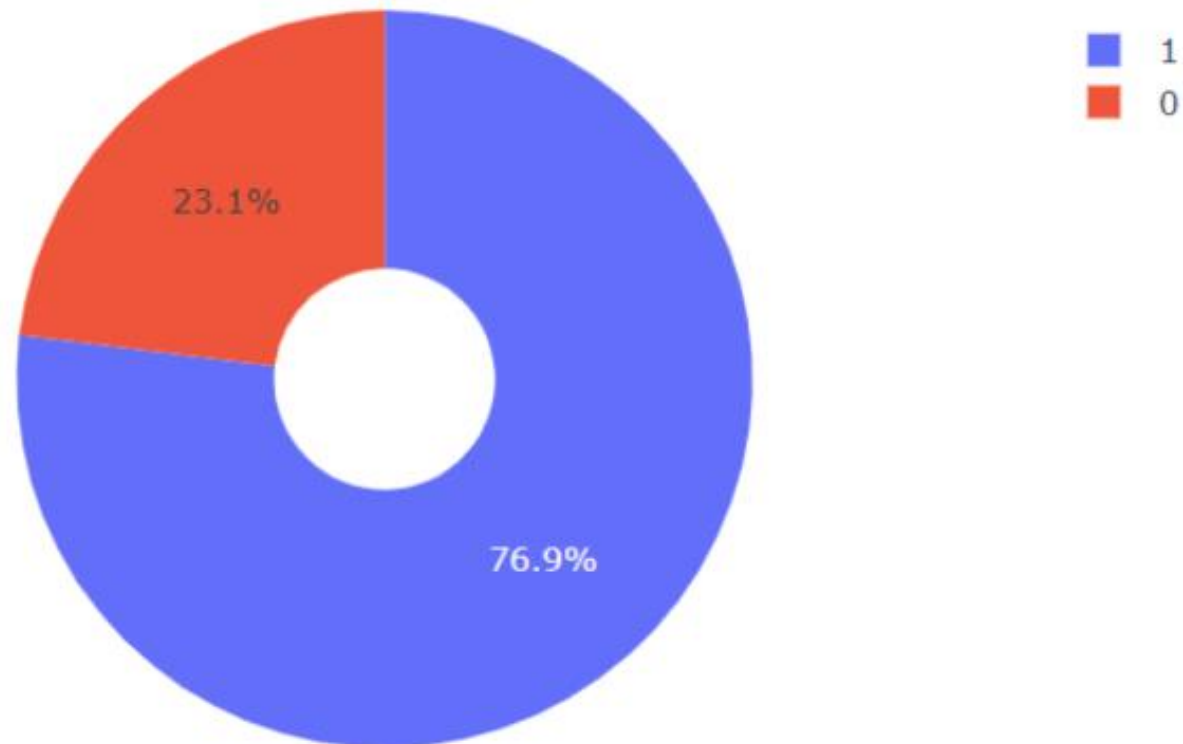
## Pie chart showing the success percentage achieved by each launch site

Total Success Launches By all sites



***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



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





Section 6

# Predictive Analysis (Classification)

# Classification Accuracy

- The SVM classifier is the model with the highest classification accuracy

## TASK 12

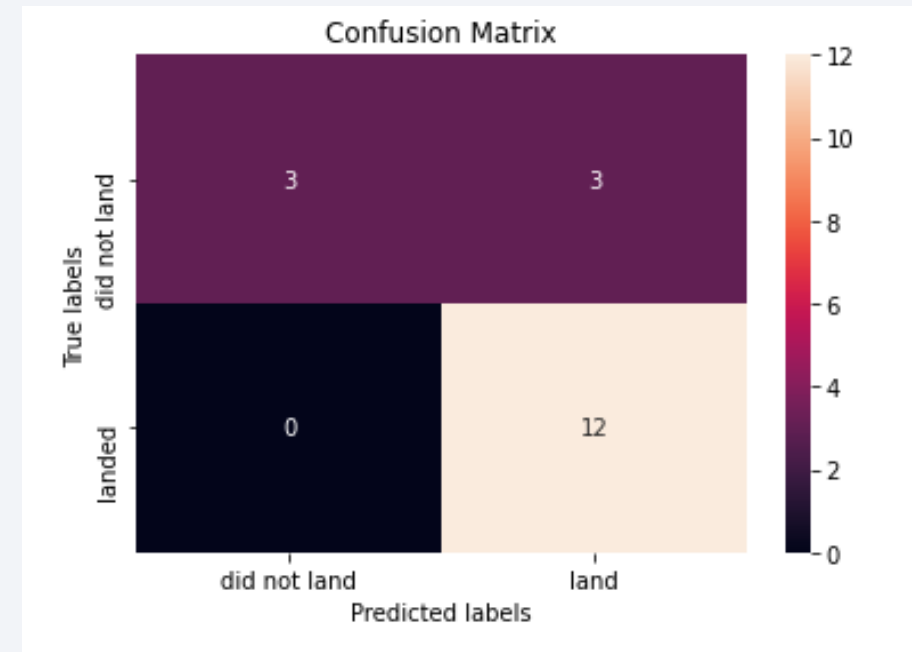
Find the method performs best:

```
In [62]: best_model = max((svm_score, "SVM"), (tree_score, "Decision Tree"), (knn_score, "KNN"))  
         print("Best performing model:", best_model[1])
```

Best performing model: SVM

# Confusion Matrix

- The confusion matrix for the SVM 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:

- Launch sites with more flights tend to have higher success rates.
- Success rates improved steadily from 2013 to 2020.
- The most successful orbits were ES-L1, GEO, HEO, SSO, and VLEO.
- KSC LC-39A had the highest number of successful launches.
- The SVM classifier was the best machine learning model for this task.

Thank you!

