IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

V R Mithun Raja
06/06/2025

Outline

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

Executive Summary

- Summary of methodologies

•**Data Collection**
  - •Used SpaceX **REST API** to collect structured launch data (e.g., payload, orbit, landing outcome)
  - •**Web scraping** from Wikipedia using BeautifulSoup to extract missing metadata (e.g., booster version)

•**Data Wrangling**
  - •Cleaned null/missing values and standardized column formats
  - •Merged API and scraped datasets into a single DataFrame
  - •Engineered features such as launch success flag, payload range, and booster type

•**Exploratory Data Analysis (EDA)**
  - •Visualized launch trends and outcomes using matplotlib, seaborn, and SQL
  - •Investigated relationships between launch site, payload, and success rate

•**Interactive Visualization**
  - •Created **Folium maps** to show global launch site markers and proximity insights
  - •Built a **Plotly Dash dashboard** to analyze launch success by payload, site, and booster version

•**Predictive Modeling**
  - •Built classification models: Logistic Regression, SVM, KNN, and Decision Tree
  - •Tuned hyperparameters using **GridSearchCV** with 10-fold cross-validation
  - •Evaluated using accuracy and confusion matrix

- Summary of all results
- Launch success depends on **site**, **booster version**, and **payload mass**.
- Folium maps showed all sites near coastlines with clear success/failure visualization.
- Plotly dashboard revealed that **medium payloads** had higher success rates.
- **Logistic Regression** performed best with highest accuracy (~88%).
- Confusion matrix confirmed reliable classification between success and failure.

Introduction

- Project background and context
  - SpaceX is revolutionizing space travel with reusable rockets to reduce launch costs.
  - Falcon 9 boosters are designed to **land back safely** after launch for reuse.
  - Analyzing past launch data can help **understand patterns and improve future outcomes**.

- Problems you want to find answers
  - What factors affect the **success of booster landings**?
  - Do **launch site**, **payload mass**, or **orbit type** influence landing outcomes?
  - Can we **predict landing success** using historical launch data and machine learning?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Retrieved launch data from SpaceX REST API
- Perform data wrangling
  - Cleaned missing values and standardized formats
  - Merged datasets and created features like launch success flag
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Used visualizations and sql queries to find patterns
- Perform interactive visual analytics using Folium and Plotly Dash
  - Built Folium maps to show launch locations and outcomes and created potly dash dashboard
- Perform predictive analysis using classification models
  - Applied classification models and used GridSearchCV for tuning and accuracy and confustion matrix evaluation.

Data Collection

- Describe how data sets were collected.
  - Primary Source: SpaceX Launch Data from SpaceX REST API
    - Accessed using requests.get() in Python
    - Converted JSON to DataFrame using pandas.json_normalize()
  - Secondary Source: Wikipedia page for Falcon 9 launches
    - Scraped using BeautifulSoup
    - Extracted booster version, landing outcome, and launch site details
  - Merged both datasets using common fields like flight number, launch date, and booster version.

Data Collection – SpaceX API

Get request for rocket launch data using API

Use json_normalize method to convert json result to dataframe

Performed data cleaning and filling the missing value

- From:

https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_
```

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

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

```
response.status_code
```

```
200
```

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

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

# Step 4: Normalize the JSON into a flat table
df = pd.json_normalize(data)
```

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
df.head()
```

Data Collection - Scraping

Request the Falcon9 Launch Wiki page from url

Create a BeautifulSoup from the HTML response

Extract all column/variable names from the HTML header

- From:
- https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/jupyter-labs-webscraping.ipynb

```python
import requests

# Static Wikipedia URL
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

# Make the GET request
response = requests.get(static_url)
```

```python
# Create BeautifulSoup object
soup = BeautifulSoup(response.text, 'html.parser')

# Optional: Preview the page title
print(soup.title.string)
```

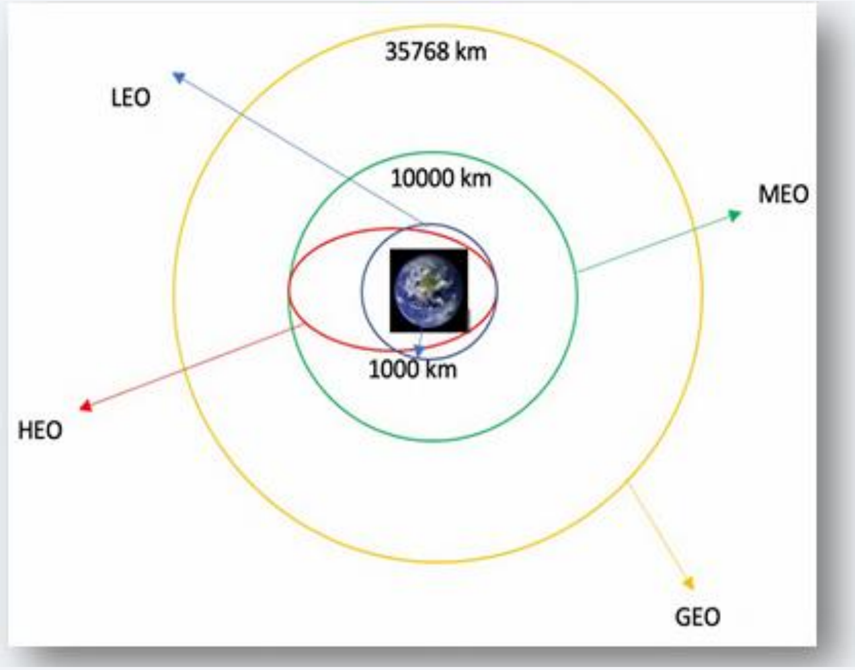
```python
column_names = []

# 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
first_launch_table = html_tables[2]

# Initialize column_names list
column_names = []

# Find all <th> elements
table_headers = first_launch_table.find_all("th")

# Extract and store column names
for th in table_headers:
    name = extract_column_from_header(th)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

Check the extracted column names

```python
print(column_names)
```

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

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

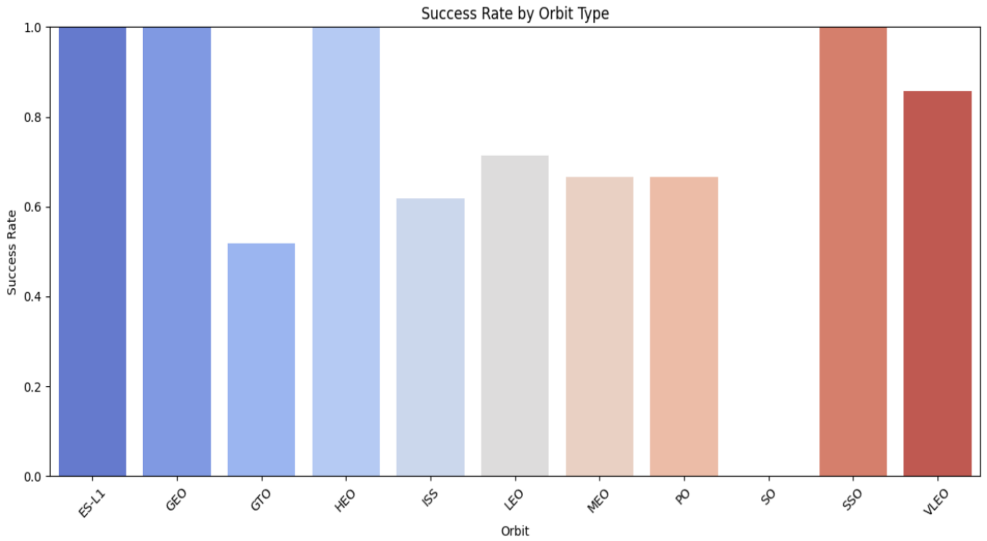
- From:

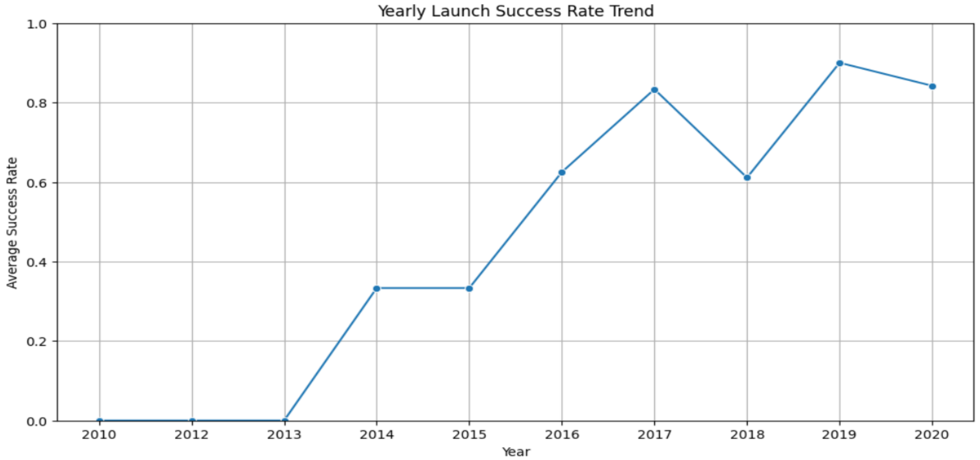
https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb

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.
- Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs. It’s very easy to see which factors affecting the most to the success of the landing outcomes
- From:
- https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/edadataviz.ipynb

EDA with Data Visualization





- Once we get a hint of the relationships using scatter plot. We will then use further visualization tools such as bar graph and line plots graph for further analysis.
- Bar graphs is one of the easiest way to interpret the relationship between the attributes. In this case, we will use the bar graph to determine which orbits have the highest probability of success.
- We then use the line graph to show a trends or pattern of the attribute over time which in this case, is used for see the launch success yearly trend.
- We then use Feature Engineering to be used in success prediction in the future module by created the dummy variables to categorical columns
- From
- https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/edadataviz.ipynb

EDA with SQL

- Using SQL, we had performed many queries to get better understanding of the dataset,
  - Ex:- Displaying the names of the launch sites.
  - \- Displaying 5 records where launch sites begin with the string 'CCA'.
  - \- Displaying the total payload mass carried by booster launched by NASA (CRS).
  - \- Displaying the average payload mass carried by booster version F9 v1.1.
  - \- Listing the date when the first successful landing outcome in ground pad was achieved.
  - \- Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
  - \- Listing the total number of successful and failure mission outcomes.
  - \- Listing the names of the booster_versions which have carried the maximum payload mass.
  - \- Listing the failed landing_outcomes in drone ship, their booster versions, and launch sites names for in year 2015.
  - \- Rank the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order.
- From: https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

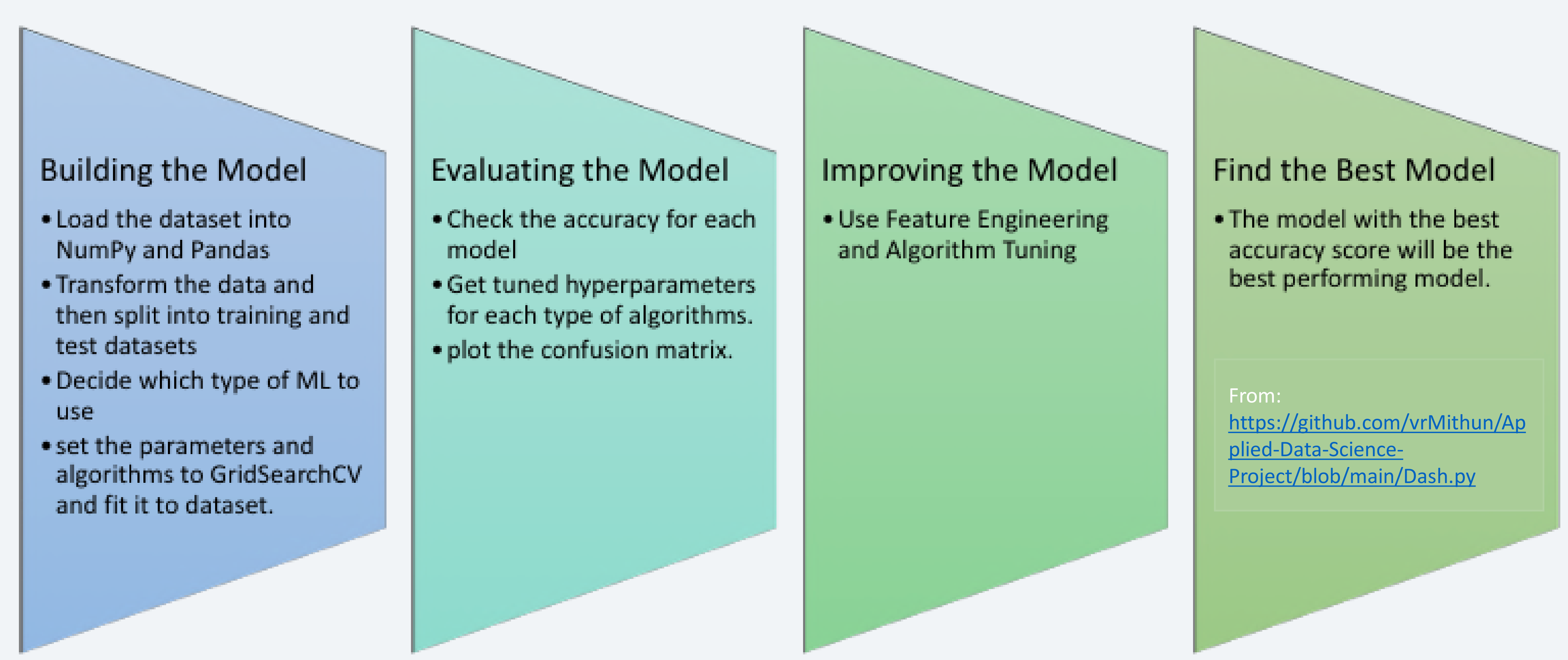
- To visualize the launch data into an interactive map. We took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.
- We then assigned the dataframe launch_outcomes(failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().
- We then used the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:
  - Howclose the launch sites with railways, highways and coastlines?
  - Howclose the launch sites with nearby cities?
- From: https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/lab_jupyter_launch_site_location%20(1).ipynb

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

- From: https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/Dash.py

Predictive Analysis (Classification)



Results

- The results will be categorized to 3 main results which is:
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

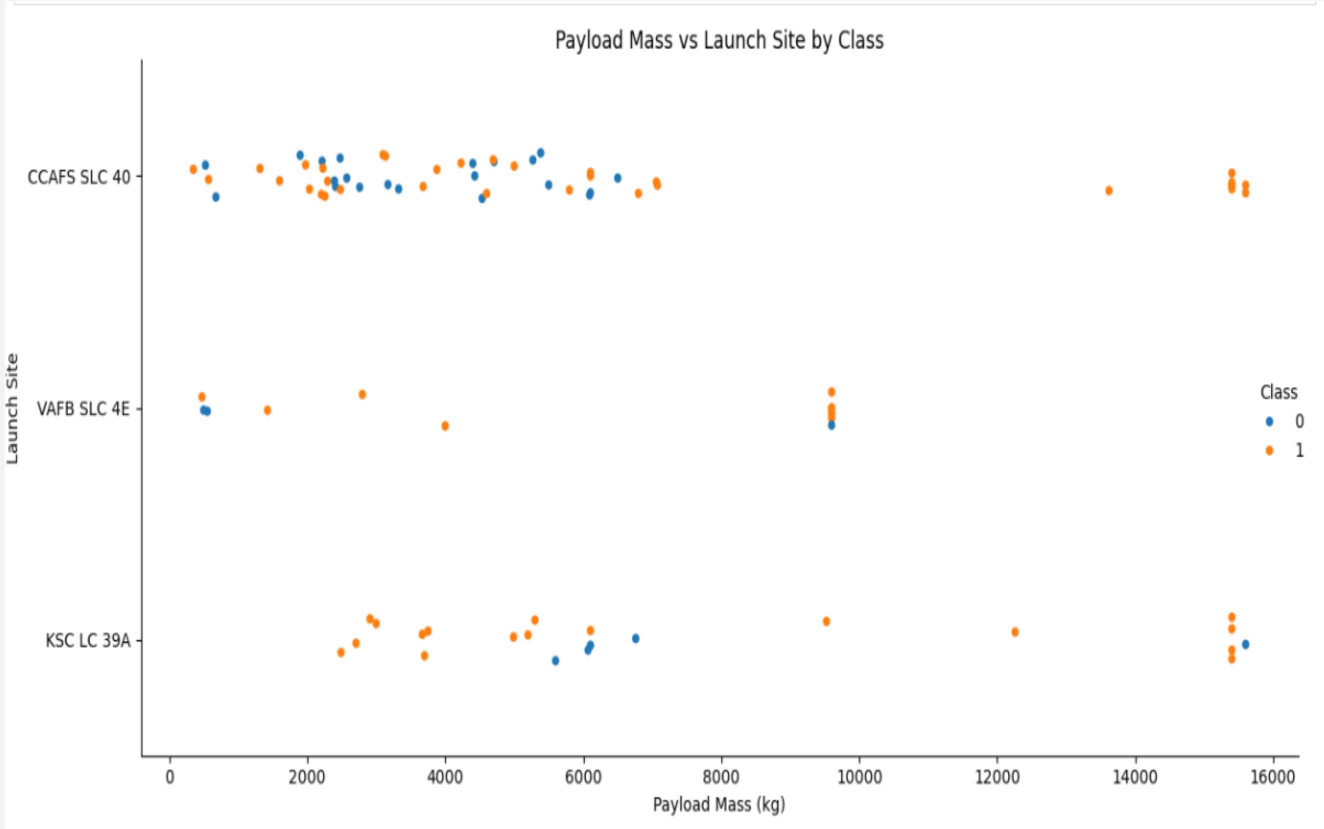
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

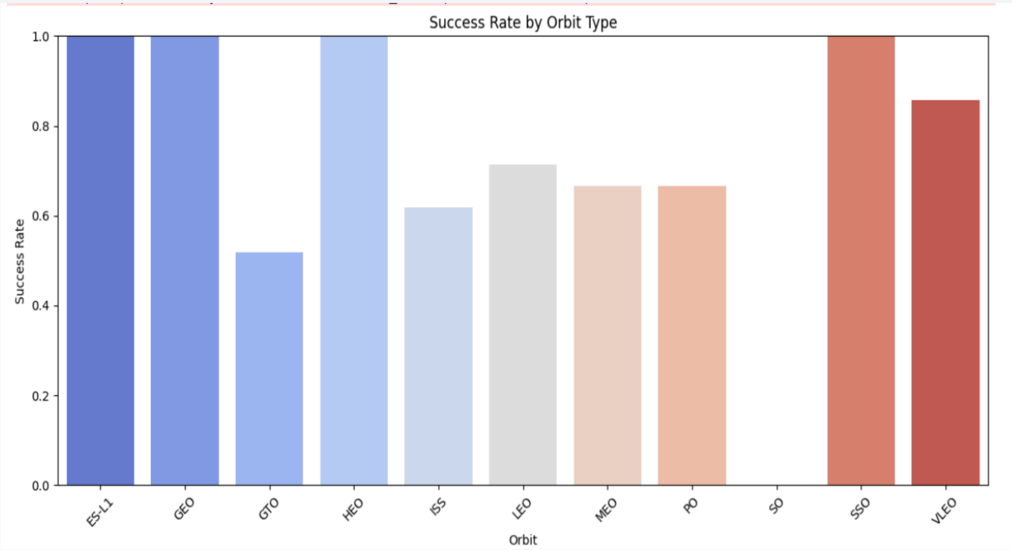
- This scatter plot shows that the larger the flights amount of the launch site, the greater the the success rate will be.
- However, site CCAFS SLC40 shows the least pattern of this.

Payload vs. Launch Site



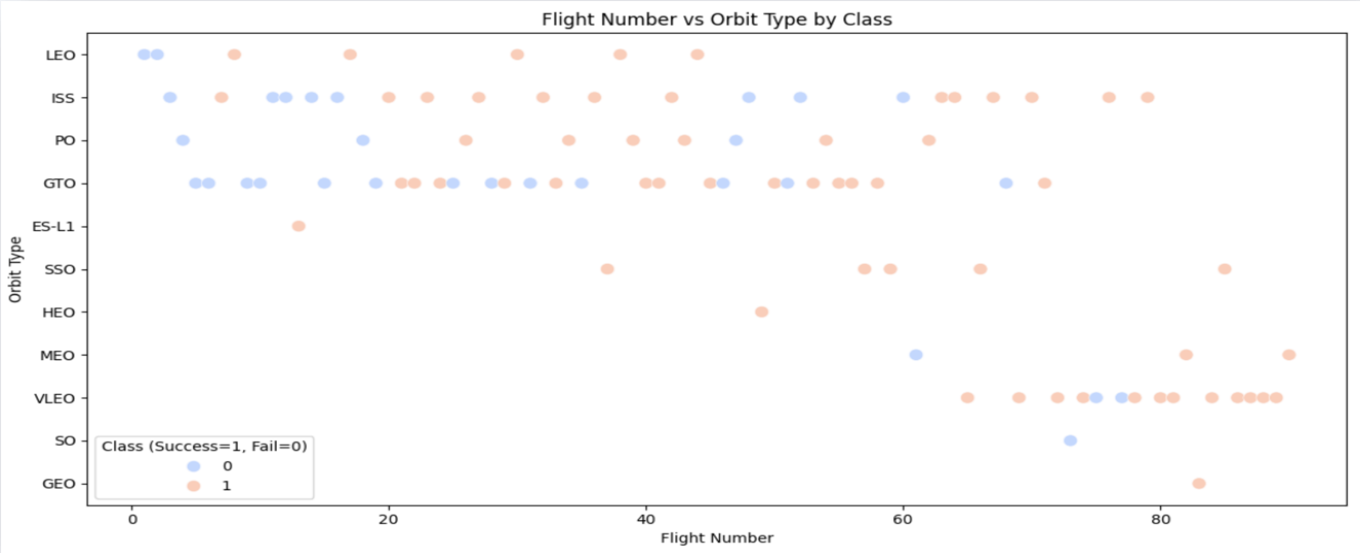
- This scatter plot shows once the pay load mass is greater than 7000kg, the probability of the success rate will be highly increased.
- However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate

Success Rate vs. Orbit Type



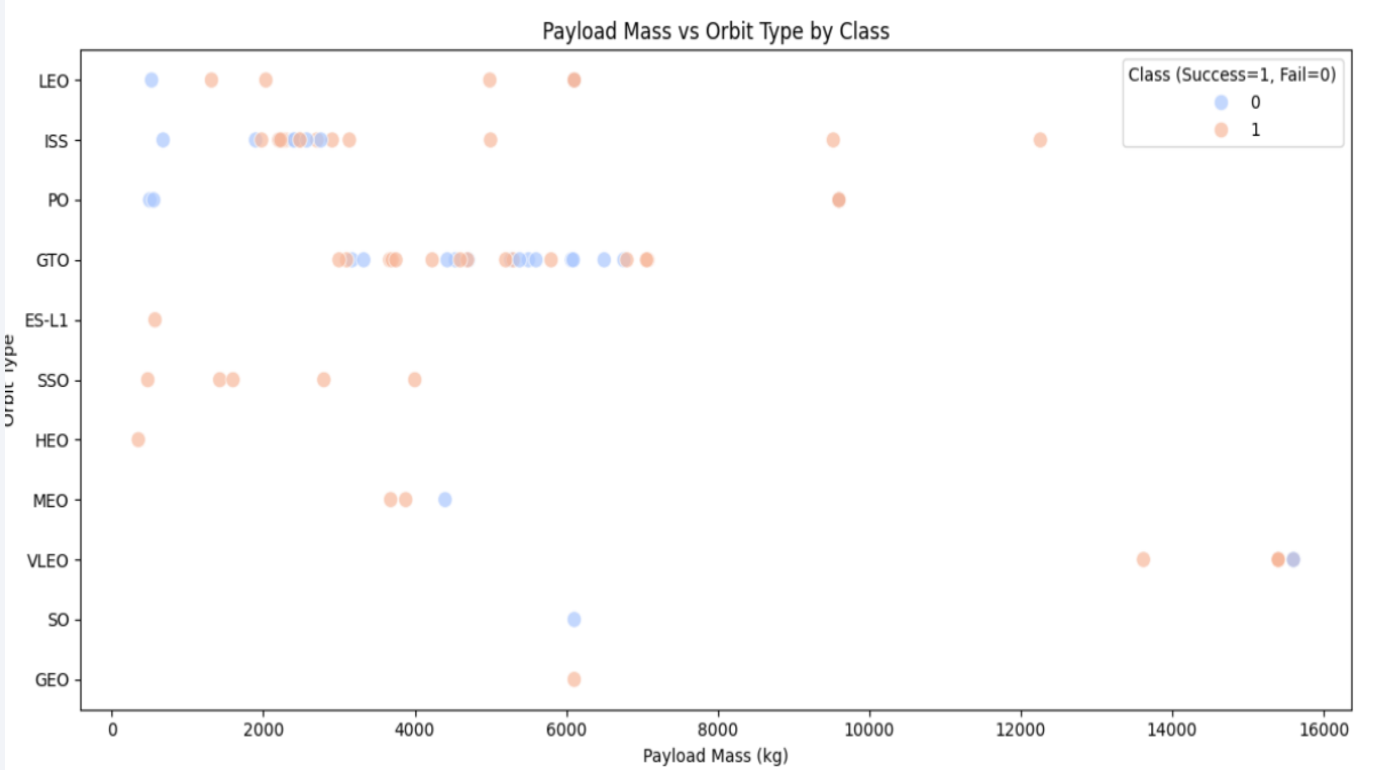
- This figure depicted the possibility of the orbits to influences the landing outcomes as some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.
- However, deeper analysis show that some of this orbits has only 1 occurrence such as GEO, SO, HEO and ES-L1 which mean this data need more dataset to see pattern or trend before we draw any conclusion.

Flight Number vs. Orbit Type



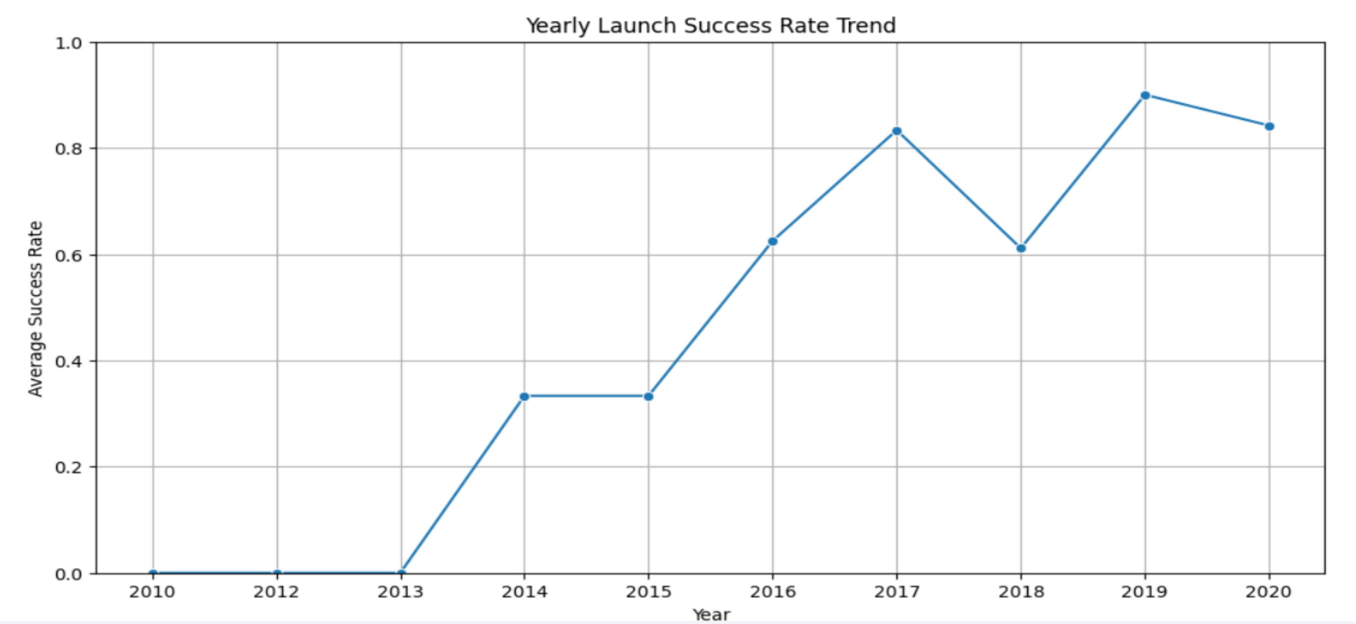
- This scatter plot shows that generally, the larger the flight number on each orbits, the greater the success rate (especially LEO orbit) except for GTO orbit which depicts no relationship between both attributes.
- Orbit that only has 1 occurrence should also be excluded from above statement as it's needed more dataset.

Payload vs. Orbit Type



- Heavier payload has positive impact on LEO, ISS and P0 orbit. However, it has negative impact on MEO and VLEO orbit. GTO orbit seem to depict no relation between the attributes.
- Meanwhile, again, SO, GEO and HEO orbit need more dataset to see any pattern or trend.with explanations

Launch Success Yearly Trend



- This figures clearly depicted and increasing trend from the year 2013 until 2020. If this trend continue for the next year onward.
- The success rate will steadily increase until reaching 1/100% success rate

All Launch Site Names

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

```
%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

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

Launch Site Names Begin with 'CCA'

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

```
%%sql
SELECT *
FROM SPACEXTABLE
WHERE "Launch_Site" LIKE 'CCA%'
LIMIT 5;
```

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

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

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

```
%%sql SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass
FROM SPACEXTABLE
WHERE "Customer" LIKE '%NASA (CRS)%';
```

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

Total_Payload_Mass
48213

- We calculated the total payload carried by boosters from NASA as 48213 using the query below

Average Payload Mass by F9 v1.1

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

```
%%sql SELECT AVG("Payload_Mass__kg_") AS Average_Payload_Mass
FROM SPACEXTABLE
WHERE "Booster_Version" = 'F9 v1.1';
```

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

Average_Payload_Mass
2928.4

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

First Successful Ground Landing Date

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

Hint:Use min function

```
%%sql SELECT MIN("Date") AS First_Successful_Ground_Landing
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (ground pad)';
```

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

First_Successful_Ground_Landing
2015-12-22

- We use the min() function to find the result We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
%%sql SELECT DISTINCT "Booster_Version"
FROM SPACEXTABLE
WHERE "Landing_Outcome" = 'Success (drone ship)'
  AND "Payload_Mass__kg_" > 4000
  AND "Payload_Mass__kg_" < 6000;
```

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

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

List the total number of successful and failure mission outcomes

```
%%sql SELECT "Mission_Outcome", COUNT(*) AS Total_Count
FROM SPACEXTABLE
GROUP BY "Mission_Outcome";
```

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

Mission_Outcome	Total_Count
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

- Used Group By on Mission_Outcome to group the success and failure of the mission
- As a result we used Count(*) to display the total success(100) and failure(1).

Boosters Carried Maximum Payload

List all the booster_versions that have carried the maximum payload mass. Use a subquery.

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

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

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

- We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function

2015 Launch Records

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: SQLLite 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.

```
%%sql SELECT substr("Date", 6, 2) AS Month,
             "Landing_Outcome",
             "Booster_Version",
             "Launch_Site"
FROM SPACEXTABLE
WHERE "Landing_Outcome" LIKE 'Failure (drone ship)'
  AND substr("Date", 0, 5) = '2015';
```

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

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

- We used a 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

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

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.

```
%%sql SELECT "Landing_Outcome", COUNT(*) AS Outcome_Count
FROM SPACEXTABLE
WHERE "Date" BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY "Landing_Outcome"
ORDER BY Outcome_Count DESC;
```

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

Landing_Outcome	Outcome_Count
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 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

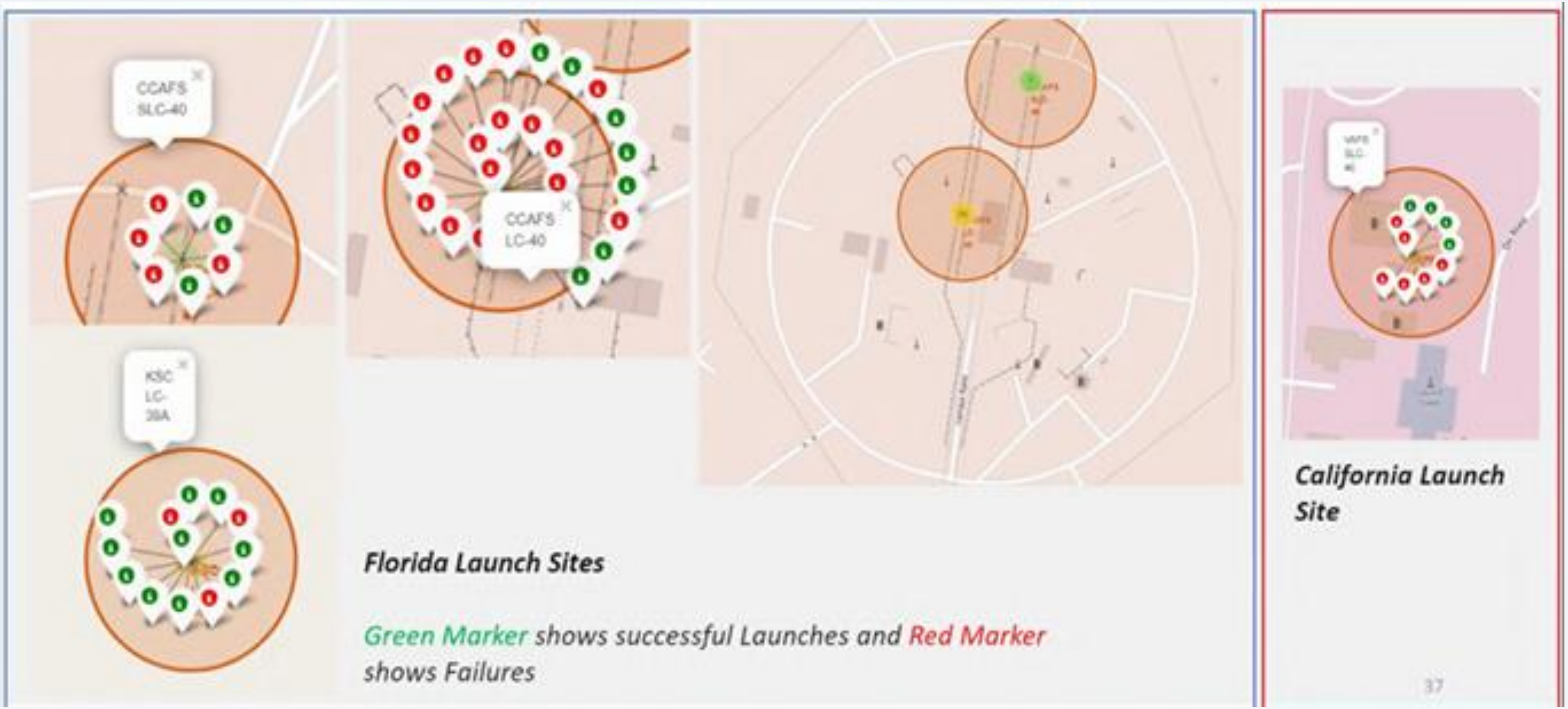
Section 3

Launch Sites Proximities Analysis

<Folium Map Screenshot 1>

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

<Folium Map Screenshot 2>

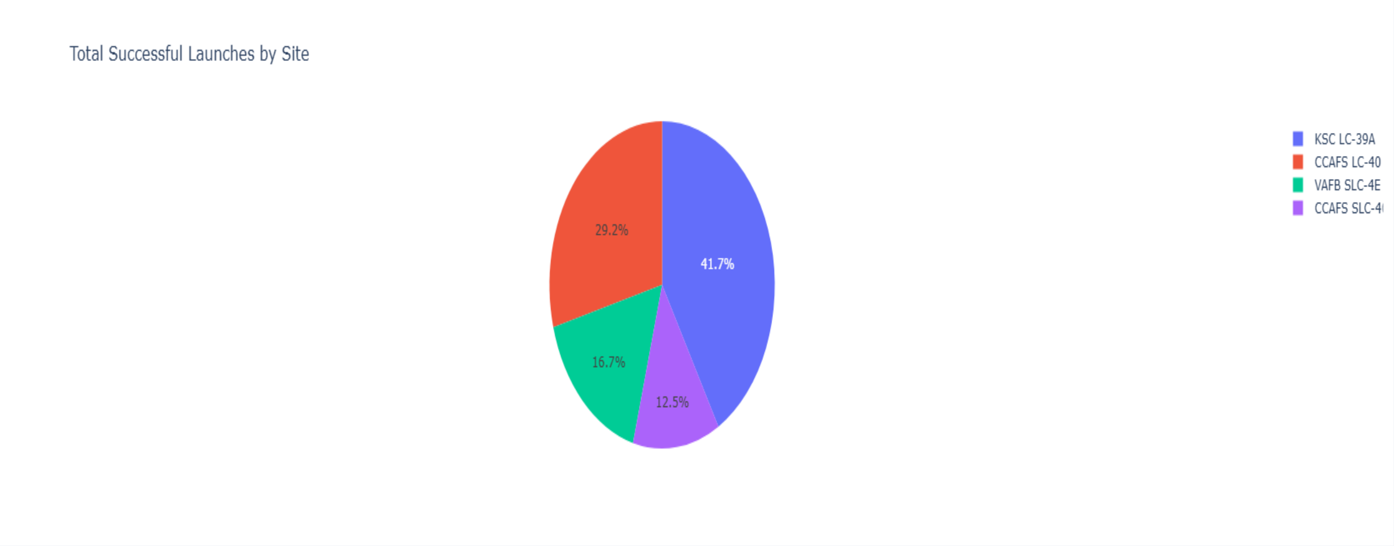


<Folium Map Screenshot 3>

Section 4

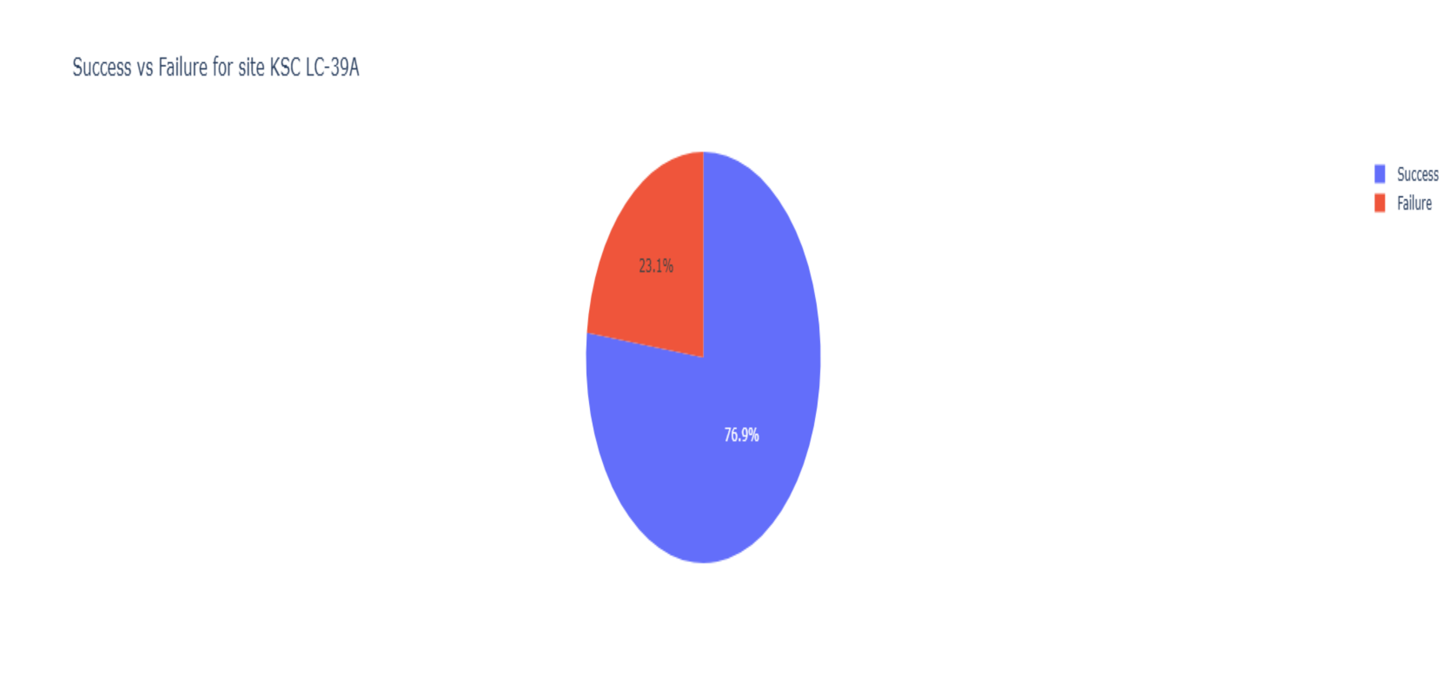
Build a Dashboard with Plotly Dash

The success percentage by each sites.



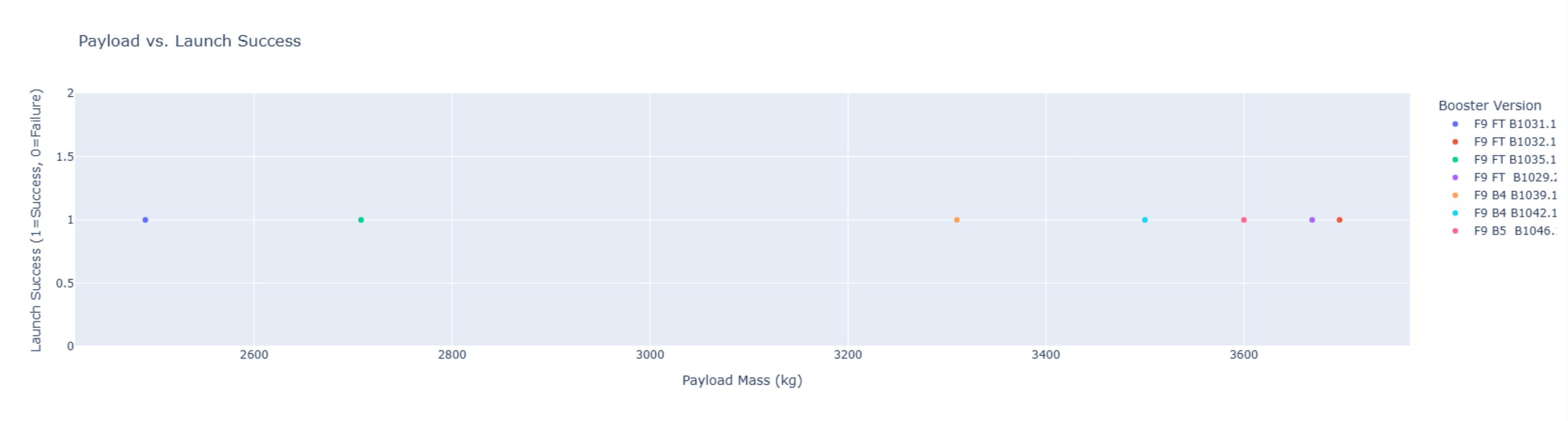
- We can see that KDC LC-39A have the most successful launches out of all.

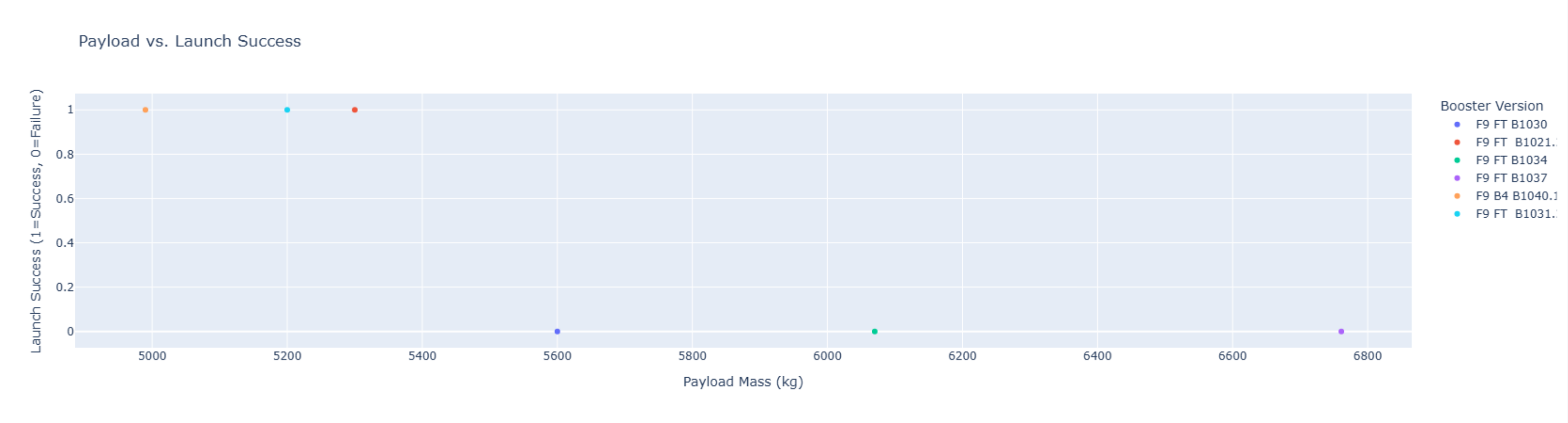
The highest launch-success ratio: KSC LC-39A



- KSC LC-39A has 76.9% success rate
- While 23.1% failure rate.

Payload vs Launch Outcome Scatter Plot





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

Section 5

Predictive Analysis (Classification)

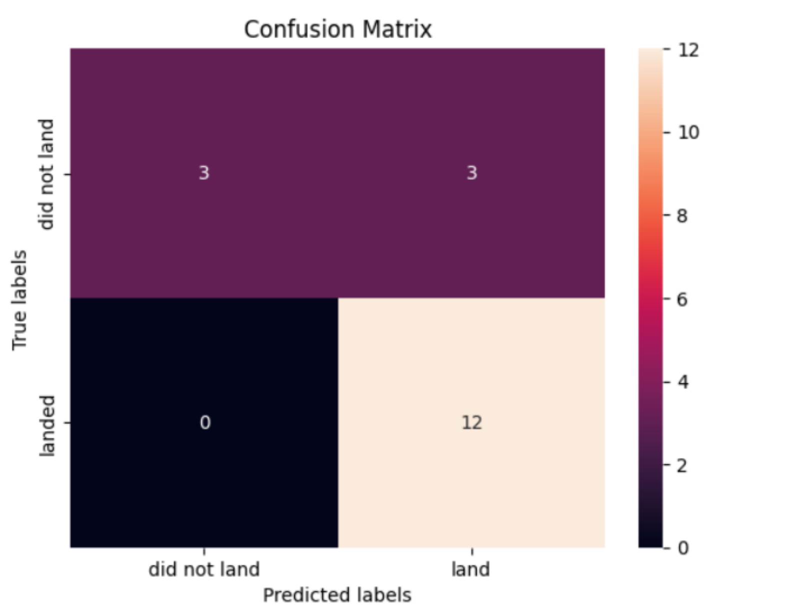
Classification Accuracy

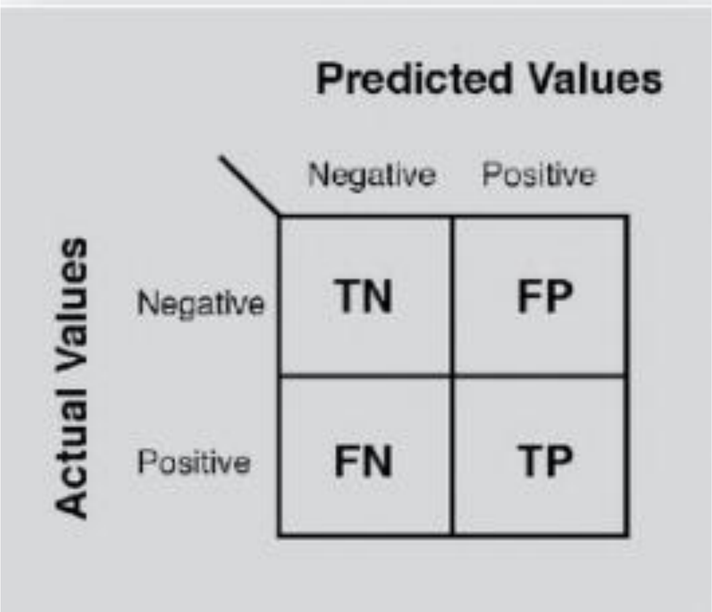
```
acc_logreg = logreg_cv.score(X_test, Y_test)
acc_svm = svm_cv.score(X_test, Y_test)
acc_tree = tree_cv.score(X_test, Y_test)
acc_knn = knn_cv.score(X_test, Y_test)

accuracies = {
    "Logistic Regression": acc_logreg,
    "Support Vector Machine": acc_svm,
    "Decision Tree": acc_tree,
    "K-Nearest Neighbors": acc_knn
}

best_model = max(accuracies, key=accuracies.get)
best_accuracy = accuracies[best_model]

print(f"Best performing model: {best_model} with accuracy {best_accuracy:.2f}")
```

```
Best performing model: Logistic Regression with accuracy 0.83
```

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.

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 Tree Classifier Algorithm is the best Machine Learning approach for this dataset.
- The low weighted payloads (which define as 4000kg and below) performed better than the heavy weighted payloads.
- Starting from the year 2013, the success rate for SpaceX launches is increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.
- KSC LC-39A have the most successful launches of any sites; 76.9%
- SSO orbit have the most success rate; 100% and more than 1 occurrence

Appendix

- Ml analysis code:
- https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb
- Overall Dashboard png:
- https://github.com/vrMithun/Applied-Data-Science-Project/blob/main/Screenshot%202025-06-05%20200132.png

Thank you!