



Winning Space Race with Data Science

Sathyanandan Prabakaran
09/06/2025



Outline

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

Executive Summary

- Summary of methodologies
 - SpaceX Data Collection using SpaceX API
 - Data Collection with Web Scraping
 - Data Wrangling
 - Exploratory Data Analysis using SQL
 - EDA DataViz Using Python Pandas and Matplotlib
 - Launch Sites Analysis with Folium-Interactive Visual Analytics and Plotly Dash
 - Machine Learning Landing Prediction

Executive Summary

- Summary of all results
 - EDA results
 - Interactive Visual Analytics and Dashboards
 - Predictive Analysis(Classification)

Introduction



- Project background and context
 - **Project Background and Context**
SpaceX advertises Falcon 9 rocket launches on its website at a cost of **62 million dollars** per launch. In comparison, other providers charge upwards of **165 million dollars**. A major reason for this cost advantage is SpaceX's ability to **reuse the first stage** of the rocket.
 - By predicting whether the **first stage of the Falcon 9** will land successfully, we can estimate the **actual cost of a launch**. This insight can be valuable for **competitors or clients** seeking to bid against SpaceX for launch contracts.
- Problems you want to find answers
 - In this capstone project, we aim to **predict the success of the Falcon 9 first-stage landing** using data from Falcon 9 rocket launches, as advertised on SpaceX's website. The results can help assess the potential cost and reliability of a launch.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
 - Describe how data was collected
- Perform data wrangling
 - Describe how data was processed
- 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

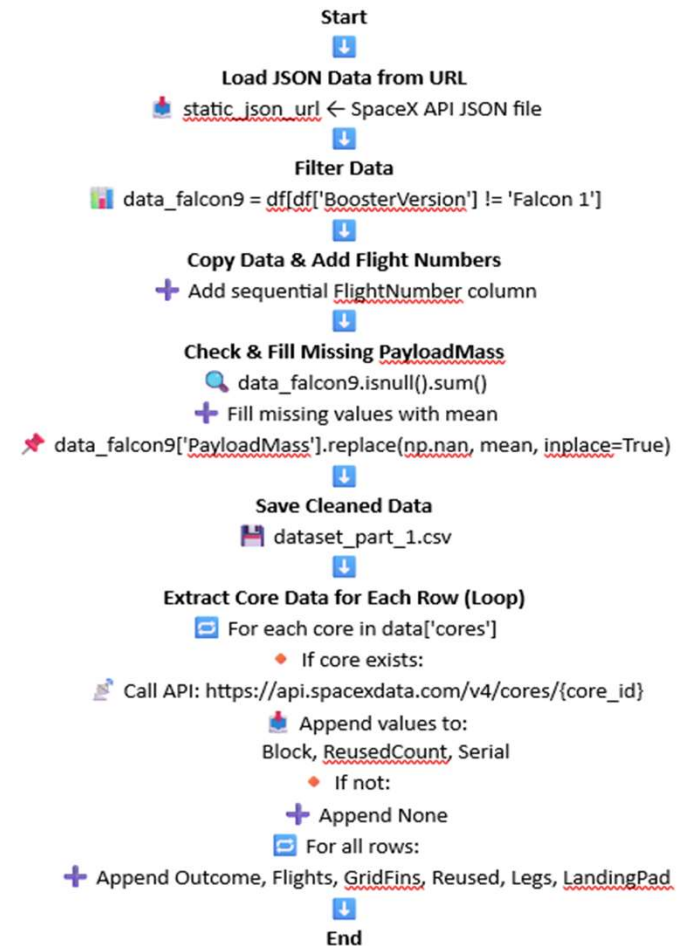
Data Collection

- The data used in this project was collected from two primary sources:

➤ The SpaceX API


➤ Wikipedia

Flowchart for SpaceX Core Data Extraction and Preprocessing



Data Collection – SpaceX API

1. Using the SpaceX API

- The Falcon 9 launch data was collected using the **SpaceX RESTful API** by making a **GET request** to the API endpoint.
- A set of **helper functions** was defined to efficiently access and extract detailed launch information using **identification numbers** embedded in the launch data.
- The requested launch data was **parsed and decoded** from the JSON response format.
- The response content was then **converted into a structured format** using the **Pandas** library, resulting in a **DataFrame** suitable for analysis.
- This process allowed for consistent extraction of key data points such as rocket type, payload mass, launch site, and outcome of first-stage landings.
-  Here is the GitHub URL of the completed SpaceX API calls notebook:
<https://github.com/sathyanandanp/CAPSTONE/blob/main/jupyter-labs-spacex-data-collection-api.ipynb>

been reused, and the serial of the core.

```
[5]: # Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
        if core['core'] != None:
            response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
            Block.append(response['block'])
            ReusedCount.append(response['reuse_count'])
            Serial.append(response['serial'])
        else:
            Block.append(None)
            ReusedCount.append(None)
            Serial.append(None)
    Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
    Flights.append(core['flight'])
    GridFins.append(core['gridfins'])
    Reused.append(core['reused'])
    Legs.append(core['legs'])
    LandingPad.append(core['landpad'])
```

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

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

```
[7]: response = requests.get(spacex_url)
```


Check the content of the response

```
[8]: print(response.content)

b'{"fairings":{"reused":false,"recovery_attempt":false,"recovered":false,"ships":[]},"links":{"patch":{"small":"https://images2.imgbox.com/94/f2/IN6Ph45r_o.png","large":"https://images2.imgbox.com/5b/02/QcxHub5V_o.png"},"reddit":{"campaign":null,"launch":null,"media":null,"recovery":null},"flickr":{"small":[],"original":[]},"presskit":null,"webcast":"https://www.youtube.com/watch?v=0a_00n_Y88","youtube_id":"0a_00n_Y88","article":"https://www.sp
```

Data Collection – Web Scraping

2. Web Scraping from Wikipedia

- To complement the API data, **web scraping** was performed on the Wikipedia page titled: **"List of Falcon 9 and Falcon Heavy launches"**, which contains **historical launch records** formatted in HTML tables.
- Using the **BeautifulSoup** and **Requests** libraries in Python, the HTML table containing Falcon 9 launch records was extracted from the Wikipedia page.
- The table was then **parsed, cleaned, and transformed** into Pandas DataFrame for consistency with the API data format
- This enabled integration of historical launch records with the API data, ensuring a more **comprehensive dataset** for analysis and modeling.
-  **Here is the GitHub URL of the completed web scraping notebook:**
<https://github.com/sathyanandanp/CAPSTONE/blob/main/jupyter-labs-webscraping.ipynb>

```
[ ]: # use requests.get() method with the provided static url
response = requests.get("https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922")

# assign the response to a object
html_content = response.content
```

Create a BeautifulSoup object from the HTML response

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

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

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about BeautifulSoup, please check the external reference link towards the end of this lab

```
[15]: # Use the find_all function in the BeautifulSoup object, with element type 'table'
html_tables = soup.find_all('table')
# Assign the result to a list called 'html_tables'
html_tables
```

```
[15]: [(<table class="col-begin" role="presentation">
  <tbody><tr>
    <td class="col-break">
      <div class="my-heading my-heading3"><h3 id="Rocket_configurations">Rocket configurations</h3></div>
      <div class="reset_navigate" ctitle="Navigation: 100% main content: 100% max width: 100%")</div>
```

Data Wrangling

After collecting and converting the SpaceX launch data into a **Pandas DataFrame**, several **data cleaning and preprocessing** steps were performed to prepare the dataset for analysis and modeling:

1. Filtering the Dataset

- The data was **filtered using the BoosterVersion column** to include only **Falcon 9** launches.
- This ensured that only relevant launch data was retained for the analysis.

Continued...

TASK 3: Calculate the number and occurrence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes`. Then assign it to a variable `landing_outcomes`.

```
[24]: # landing_outcomes = values on Outcome column
      landing_outcomes = df['Outcome'].value_counts()
      landing_outcomes
```

```
[24]: Outcome
      True ASDS      41
      None None      19
      True RTLS      14
      False ASDS       6
      True Ocean       5
      False Ocean       2
      None ASDS         2
      False RTLS        1
      Name: count, dtype: int64
```

`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad. `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship. `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

```
[25]: for i,outcome in enumerate(landing_outcomes.keys()):
      print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```


We create a set of outcomes where the second stage did not land successfully:

Data Wrangling

2. Handling Missing Values

- Missing values in critical columns such as **LandingPad** and **PayloadMass (kg)** were addressed:
 - For the PayloadMass (kg) column, **missing values were replaced with the column's mean** to maintain consistency and avoid loss of data.
 - Entries with missing LandingPad data were reviewed for potential exclusion or further data retrieval.

3. Exploratory Data Analysis (EDA)

- Initial **EDA** was conducted to uncover patterns in the data, such as:
 - Relationships between launch features (e.g., payload mass, orbit type, launch site) and landing success.
 - Distribution of values and potential outliers.
- ☐ This analysis also helped identify the **label (target variable)** for training supervised machine learning models — specifically, whether the **first stage landing was successful**.
-  Here is the GitHub URL of the completed data wrangling notebook:
[<https://github.com/sathyanandanp/CAPSTONE/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb>]

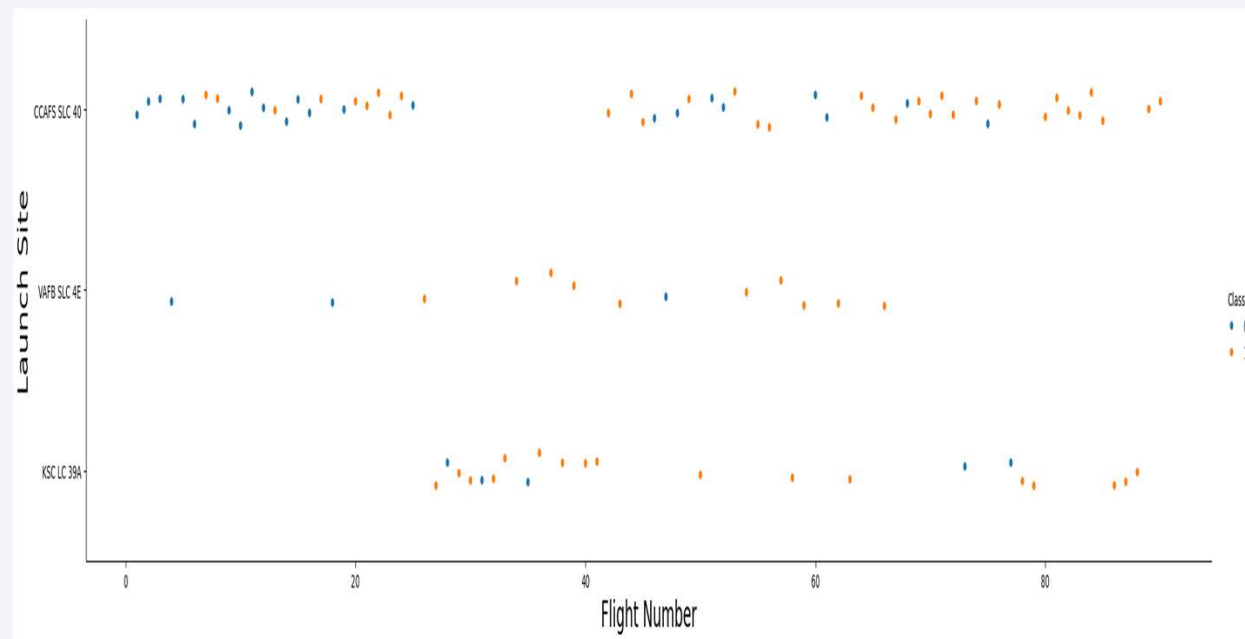
EDA with Data Visualization

Data Analysis and Feature Engineering

- Comprehensive **data analysis** and **feature engineering** were performed using **Pandas** and **Matplotlib** to better understand the data and prepare it for model development. This process involved the following steps:

1. Exploratory Data Analysis (EDA)

- Conducted in-depth EDA to uncover patterns, correlations, and distributions within the dataset.
- Analyzed key relationships between features relevant to Falcon 9 launch success.



EDA with Data Visualization

Feature Engineering

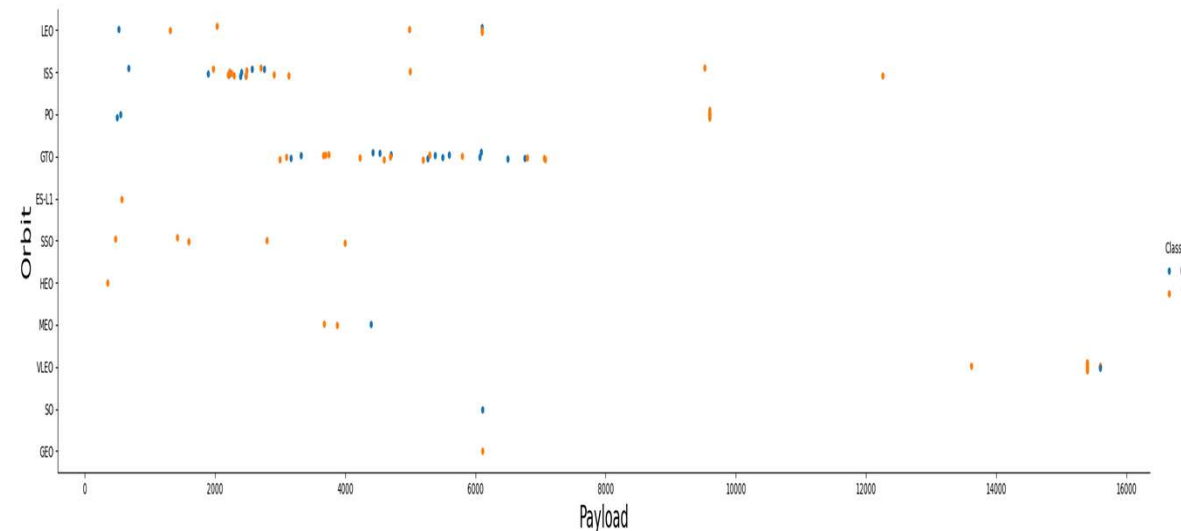
- Created and refined features to be used as inputs for machine learning models.
- Derived meaningful relationships between numerical and categorical variables to enhance predictive performance.

3. Data Visualization

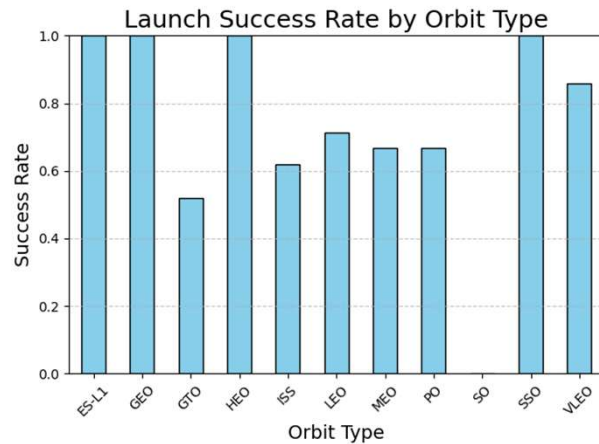
Utilized several types of plots to better interpret the data:

- **Scatter Plots:**
 - **Flight Number vs. Launch Site**
 - **Payload vs. Launch Site**
 - **Flight Number vs. Orbit Type**
 - **Payload vs. Orbit Type**

Continued...



EDA with Data Visualization



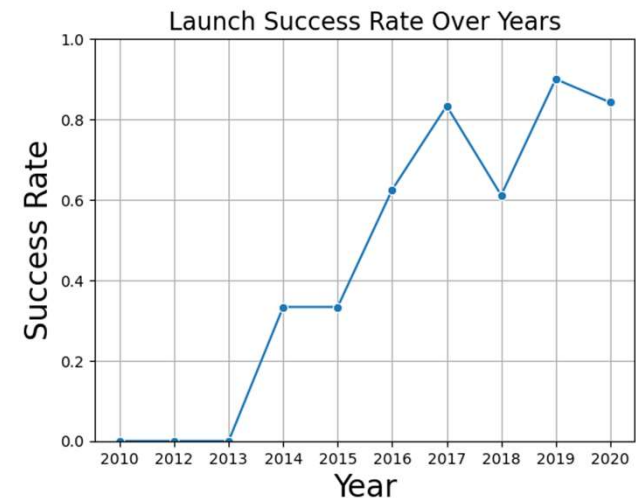
- **Bar Chart:**

- Visualized the **success rate for each orbit type** to identify patterns and high-performing categories.

- **Line Plot:**

- Showed the **yearly trend of launch success**, highlighting improvements or declines over time.

-  Here is the **GitHub URL of the completed EDA and Data Visualization notebook:**
[<https://github.com/sathyanandanp/CAPSTONE/blob/main/edadataviz.ipynb>]



EDA with SQL

Exploratory Data Analysis Using SQL

- SQL queries were used to explore and extract meaningful insights from the SPACEXTBL dataset. The analysis focused on identifying patterns related to launch sites, payload mass, booster versions, and mission outcomes.

Key SQL Queries and Findings:

1. Unique Launch Sites:

- Queried the table to display all **distinct launch sites** used in the missions.
- Helped identify the geographical diversity of SpaceX's launch operations.

```
SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```

2. Launch Sites Beginning with 'CCA':

- Displayed the **first 5 records** where the launch site name starts with 'CCA'.

```
SELECT LAUNCH_SITE FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

3. Total Payload by NASA (CRS):

- Calculated the **total payload mass** launched by NASA missions labeled 'NASA (CRS)'.

```
SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

4. Average Payload Mass for F9 v1.1 Boosters:

- Determined the **average payload mass** carried by Falcon 9 v1.1 boosters.

```
SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE 'F9 v1.0%';
```

EDA with SQL

5. First Successful Ground Pad Landing:

- Identified the **earliest date** when a booster successfully landed on a ground pad.

```
SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';
```

6. Successful Drone Ship Landings (Payload 4000–6000 kg):

- Listed boosters that successfully landed on a drone ship and carried **payloads between 4000 and 6000 kg**.

```
SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND  
PAYLOAD_MASS__KG_ < 6000;
```

7. Mission Outcomes Summary:

- Counted the total number of **successful and failed mission outcomes**.

```
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
```

8. Booster with Maximum Payload:

- Identified which booster carried the **maximum payload**.

```
SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM  
SPACEXTBL);
```

EDA with SQL

9. Failures on Drone Ship in 2015:

- Retrieved failure records specifically for **drone ship landings in 2015**, including launch site and booster details.

```
SELECT Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE Landing_Outcome = 'Failure (drone ship)' AND  
strftime('%Y', Date) = '2015';
```

10. Monthly Breakdown of 2015 Failures:

- Analyzed drone ship failures **month-by-month for the year 2015**.

```
SELECT strftime('%m', Date) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE Landing_Outcome  
= 'Failure (drone ship)' AND strftime('%Y', Date) = '2015';
```


11. Landing Outcome Frequency (2010–2017):

- Grouped and sorted landing outcomes by frequency within a defined date range.

```
SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-  
04' AND '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY TOTAL_NUMBER DESC;
```


-  Here is the GitHub URL of the completed EDA using SQL notebook:
[\[https://github.com/sathyanandanp/CAPSTONE/blob/main/jupyter-labs-eda-sql-coursera_sqlite.ipynb\]](https://github.com/sathyanandanp/CAPSTONE/blob/main/jupyter-labs-eda-sql-coursera_sqlite.ipynb)

Build an Interactive Map with Folium

- **Objective:** Visualized all SpaceX launch sites using **Folium**, marking launch outcomes with interactive map elements.
- **Implementation Highlights:**
 - Created **Folium map objects** including markers, circles, and polylines to visually represent **launch site locations** and the **success/failure** outcomes of each launch.
 - Used color coding and interactive popups for intuitive understanding of **launch outcomes** (0 = Failure, 1 = Success).
 - Implemented data-driven mapping logic using `folium.Marker`, `folium.Circle`, and `folium.PolyLine`.
- **Outcome:** Enhanced understanding of spatial distribution and performance of launch sites using geospatial data visualization.
-  **GitHub Repository:**
https://github.com/sathyanandanp/CAPSTONE/blob/main/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

SpaceX Launch Dashboard with Plotly Dash

- **Objective:** Built an **interactive dashboard** application using **Plotly Dash** to visualize SpaceX launch data and analyze performance metrics dynamically.
- **Key Features:**
 - **Launch Site Dropdown Selector:** Allows users to select a specific launch site or view data across all sites.
 - **Dynamic Success Pie Chart:** Utilizes Dash callback functions to display **mission success rate** based on the selected site.
 - **Payload Range Slider:** Enables filtering of launches by payload mass (kg), offering greater insight into launch characteristics.
 - **Interactive Scatter Plot:** Displays a **Success vs. Payload scatter chart**, dynamically updated using Dash callbacks for selected launch site and payload range.
- **Technologies Used:** Python, Plotly Dash, Pandas, Bootstrap Components, Callback Functions
-  **GitHub Repository:**
https://github.com/sathyanandanp/CAPSTONE/blob/main/Updated_SpaceX_Dashboard.ipynb

Predictive Analysis (Classification)

- **Objective:** Built, evaluated, and compared multiple classification models to identify the best-performing method on a given dataset.

Workflow Summary

- **Exploratory Data Analysis (EDA):**
 - Loaded data into a **Pandas DataFrame**.
 - Extracted target labels by converting the Class column to a **NumPy array** using `.to_numpy()`, assigned to variable Y.
- **Preprocessing:**
 - Standardized feature dataset X using `StandardScaler()` from `sklearn.preprocessing`.
 - Split data into **training and test sets** using `train_test_split()` with `test_size=0.2` and `random_state=2`.

Predictive Analysis (Classification)

- **Model Training and Optimization**

- Evaluated **four classification models**:

- **Support Vector Machine (SVM)**
 - **Decision Tree Classifier**
 - **k-Nearest Neighbors (KNN)**
 - **Logistic Regression**

- **Grid Search with Cross-Validation (cv=10):**

- Created a **GridSearchCV** object for each model with a hyperparameter grid.
 - Trained each model on the training set to identify the **best hyperparameters**.
 - Extracted:
 - `best_params_` → optimal parameters
 - `best_score_` → validation accuracy

Predictive Analysis (Classification)

- **Model Evaluation**

- Calculated **test accuracy** using `.score()` on each trained model.
- Plotted a **confusion matrix** to visualize prediction outcomes for each model.
- **Performance Comparison Table** compiled to compare test accuracy across all four models and determine the best performer.



GitHub Repository:

[https://github.com/sathyanandanp/CAPSTONE/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5%20\(1\).ipynb](https://github.com/sathyanandanp/CAPSTONE/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5%20(1).ipynb)

Results

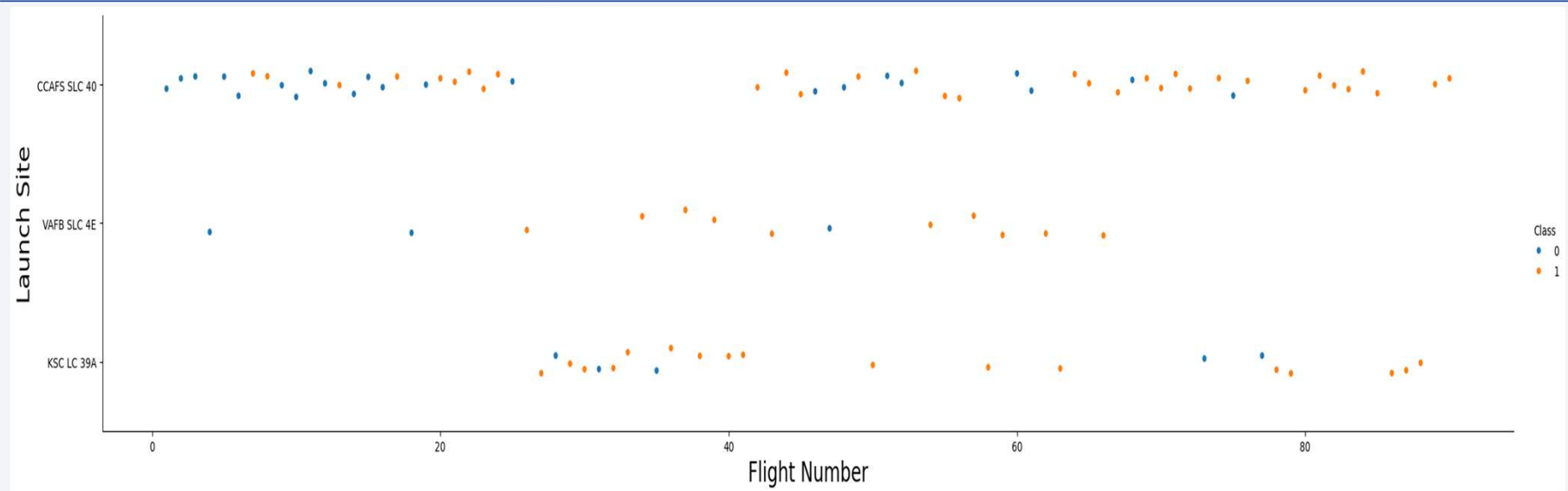
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results



Section 2

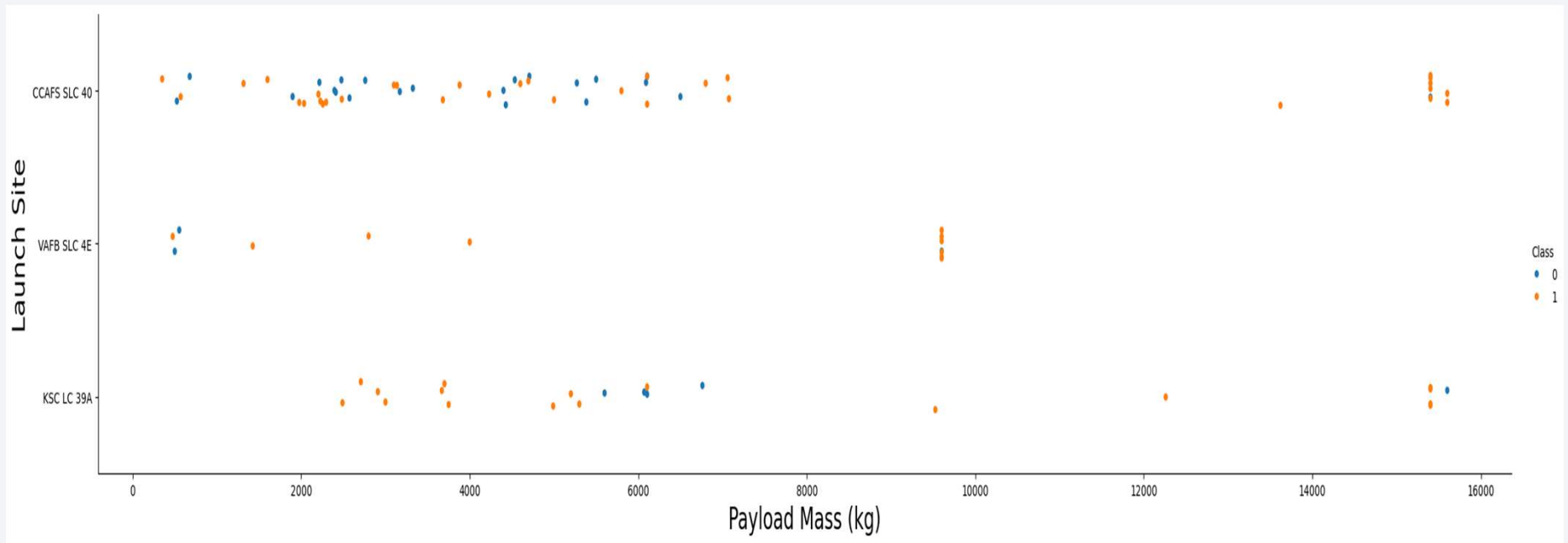
Insights drawn from EDA

Flight Number vs. Launch Site



- As the flight no increases, the success rate is getting increased
- Both KSC LC 39A and CCAFS SLC 40 After Flight No 80, the success rate is 100%

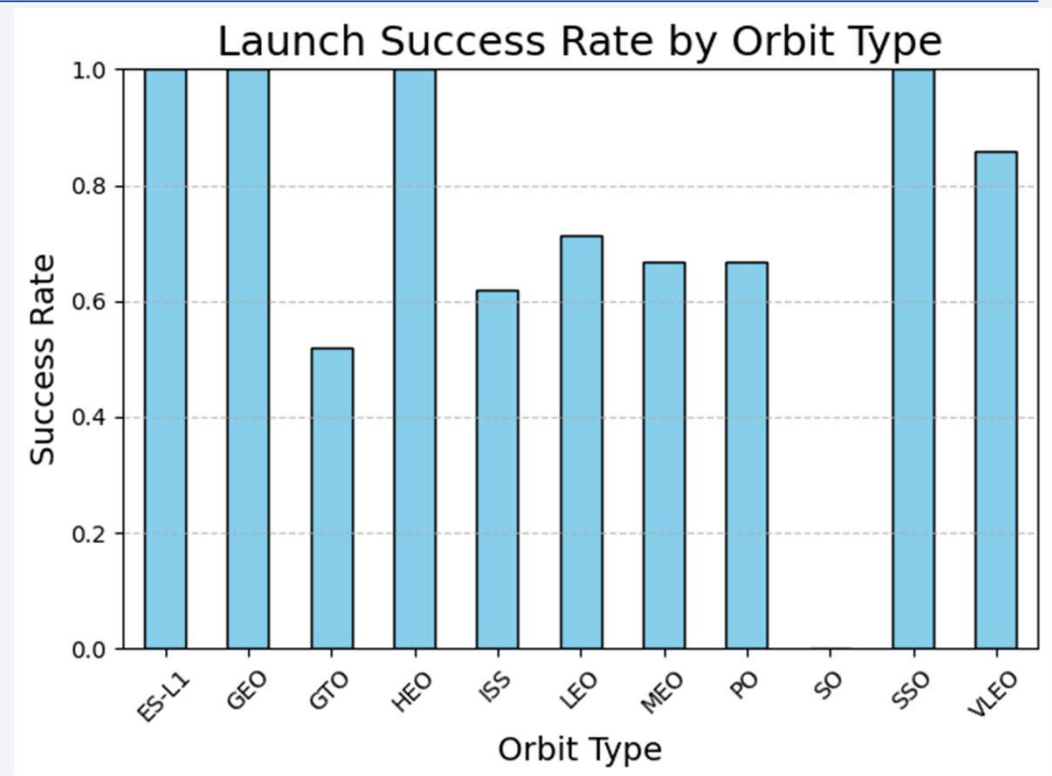
Payload vs. Launch Site



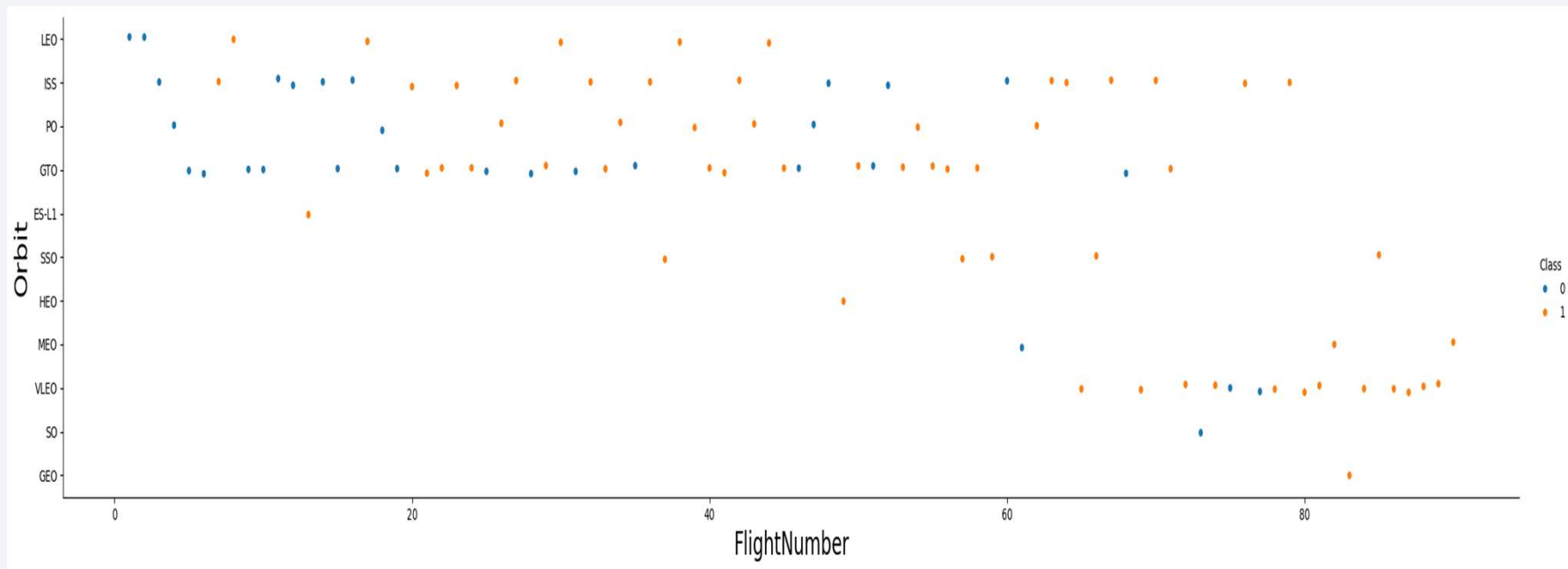
- For the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).
- For Payload above 8000, number of failures are lesser

Success Rate vs. Orbit Type

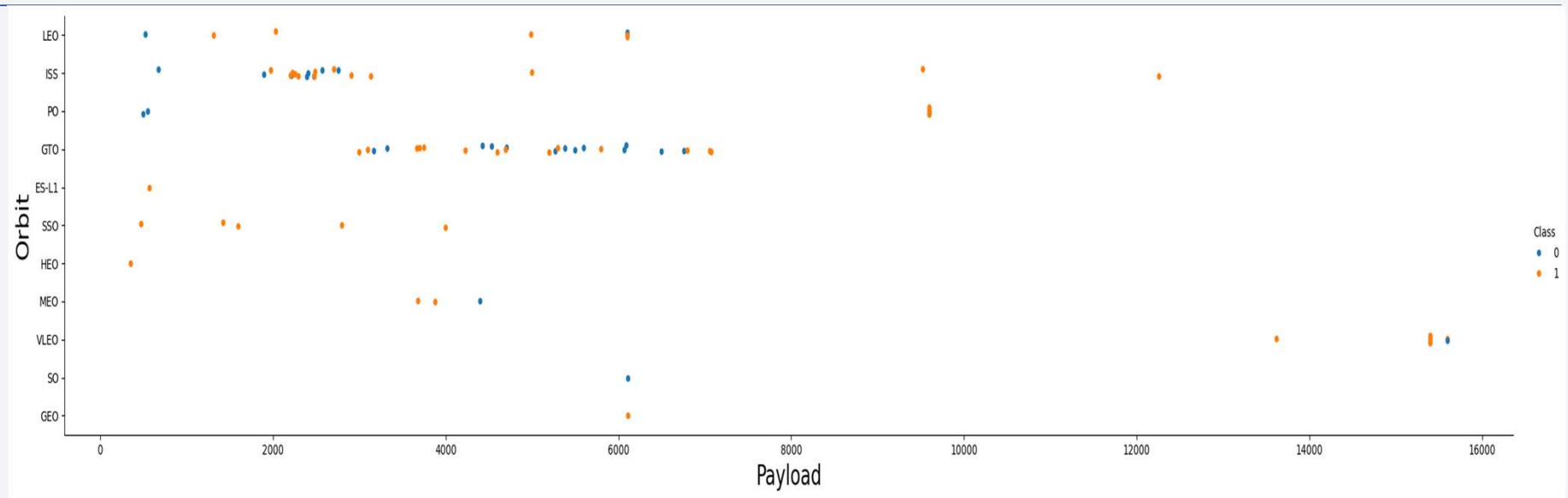
- ES-L1, GEO, HEO, SSO Orbits given 100% success rate
- GTO gave 50%
- Lowest is SO given 0%



Flight Number vs. Orbit Type



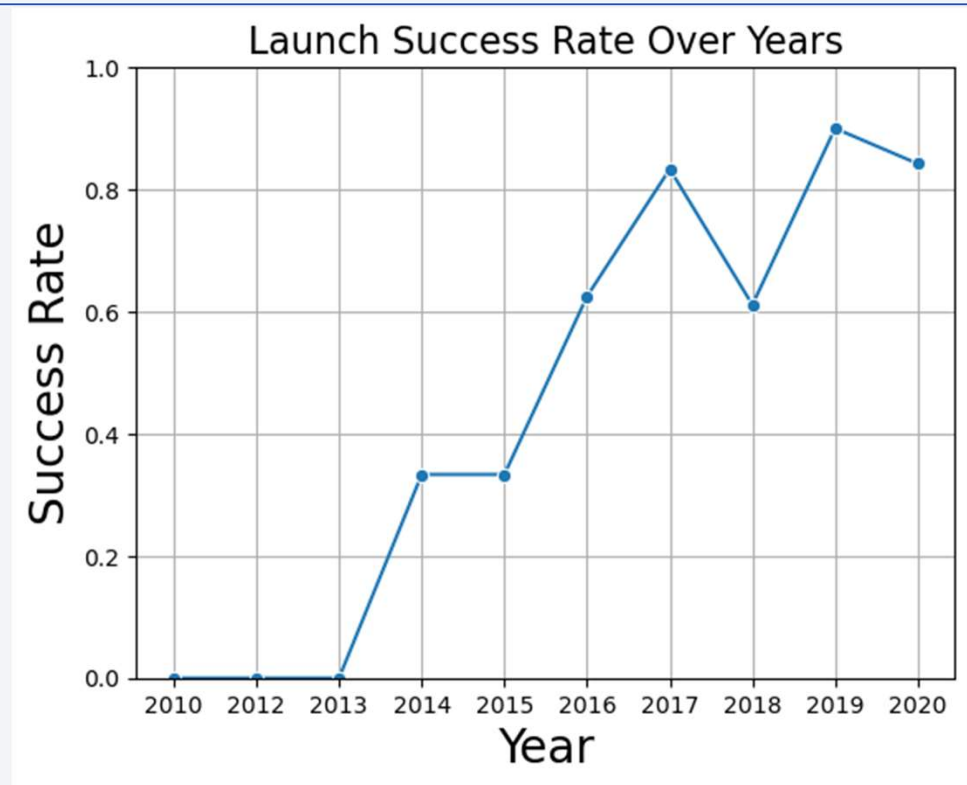
Payload vs. Orbit Type



- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However, for GTO, it's difficult to distinguish between successful and unsuccessful landings as both outcomes are present.

Launch Success Yearly Trend

- The success rate since 2013 kept on improving till 2020



All Launch Site Names

```
[11]: %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
* sqlite:///my_data1.db
Done.
[11]: Launch_Site
      CCAFS LC-40
      VAFB SLC-4E
      KSC LC-39A
      CCAFS SLC-40
```

- **CCAFS LC-40** – Cape Canaveral Air Force Station, Launch Complex 40 📍 **Cape Canaveral, Florida, USA**
- **VAFB SLC-4E** – Vandenberg Air Force Base, Space Launch Complex 4E 📍 **Lompoc, California, USA**
- **KSC LC-39A** – Kennedy Space Center, Launch Complex 39A 📍 **Merritt Island, Florida, USA**
- **CCAFS SLC-40** – Cape Canaveral Space Launch Complex 40 📍 **Cape Canaveral, Florida, USA**
- ♦ *Note: LC = Launch Complex, SLC = Space Launch Complex.* ♦ CCAFS and KSC are adjacent facilities on the Florida Space Coast, used by NASA and SpaceX.

Launch Site Names Begin with 'CCA'

- **%sql**
 - This is a magic command used in Jupyter Notebooks to execute SQL queries via a Python-SQL interface (like ipython-sql).
- **SELECT LAUNCH_SITE**
 - You are retrieving the LAUNCH_SITE column from the database table SPACEXTBL.
- **WHERE (LAUNCH_SITE) LIKE 'CCA%'**
 - This filters the rows to return only those where the LAUNCH_SITE starts with 'CCA'.
 - The % is a wildcard that matches any sequence of characters after 'CCA'.
 - So it matches strings like:
 - CCAFS LC-40
 - CCAFS SLC-40
- **LIMIT 5**
 - This restricts the output to the **first 5 matching records** only.

```
[12]: %sql SELECT LAUNCH_SITE from SPACEXTBL where (LAUNCH_SITE) LIKE 'CCA%' LIMIT 5;
```

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

```
Done.
```

```
[12]: Launch_Site
```

```
CCAFS LC-40
```

```
CCAFS LC-40
```

```
CCAFS LC-40
```

```
CCAFS LC-40
```

```
CCAFS LC-40
```

Total Payload Mass

Task 3

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

```
[13]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';  
* sqlite:///my_data1.db  
Done.  
[13]: SUM(PAYLOAD_MASS__KG_)  
45596
```

SELECT SUM(PAYLOAD_MASS__KG_)

- This part calculates the **total (sum) of the payload mass** (in kilograms) across all relevant rows.
- PAYLOAD_MASS__KG_ is the name of the column storing payload mass data. (Note: the underscores suggest this name may have been auto-generated or imported from a CSV.)
- **WHERE Customer = 'NASA (CRS)'**
 - Filters the records to include **only those launches where the customer is NASA (CRS)**.
 - CRS stands for **Commercial Resupply Services**, a NASA program to deliver cargo to the ISS via commercial providers (like SpaceX).

Average Payload Mass by F9 v1.1

TASK 4

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

```
%sql SELECT AVG(PAYLOAD_MASS_KG_) as "Payload Mass Kgs", Customer, Booster_Version FROM 'SPACEXTBL' WHERE Booster_Version LIKE 'F9 v1.1%';
```

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

Payload Mass Kgs	Customer	Booster_Version
2534.6666666666665	MDA	F9 v1.1 B1003

SELECT AVG(PAYLOAD_MASS_KG_)

- SELECT: Used to retrieve data from a database.
- AVG(PAYLOAD_MASS_KG_): Calculates the **average** (mean) of the column PAYLOAD_MASS_KG_, which stores the payload mass in kilograms.
- WHERE Booster_Version LIKE 'F9 v1.1%'
- Filters the rows to only include those where the **Booster_Version** column starts with **F9 v1.1**.
- The % is a **wildcard** in SQL, meaning "zero or more characters".
 - So it includes **F9 v1.1**, **F9 v1.1A**, **F9 v1.1B**, etc.

First Successful Ground Landing Date

```
[29]: %sql SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';
```

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

```
Done.
```

```
[29]: MIN(Date)
```

```
2015-12-22
```

- MIN(Date) is a **SQL aggregate function** that finds the **earliest (minimum) date** from the Date column
- WHERE filters only rows where the **Landing_Outcome** is exactly **Success (ground pad)** will be included.

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

```
[28]: %sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS_KG_ > 4000 AND PAYLOAD_MASS_KG_ < 6000;
* sqlite:///my_data1.db
Done.
[28]:
```

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

The names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

F9 FT B1022

F9 FT B1026

F9 FT B1021.2

F9 FT B1031.2

- **SELECT BOOSTER_VERSION**
 - This tells SQL to **retrieve the values in the BOOSTER_VERSION column.**
 - It will only return this column in the output.
- **WHERE clause**
 - This filters the rows based on **three conditions**:
 - Landing_Outcome = 'Success (drone ship)'
 - ◆ Only include launches where the booster successfully landed on a **drone ship**.
 - PAYLOAD_MASS_KG_ > 4000
 - ◆ Payload mass must be **greater than 4000 kg**.
 - PAYLOAD_MASS_KG_ < 6000
 - ◆ Payload mass must be **less than 6000 kg**.
- So only launches with payload mass **between 4000 and 6000 kg** and **drone ship landings** are selected.

Total Number of Successful and Failure Mission Outcomes

```
0]: %sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL GROUP BY MISSION_OUTCOME;
* sqlite:///my_data1.db
Done.
0]:
```

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

COUNT(MISSION_OUTCOME) AS TOTAL_NUMBER

- This counts how many times **each mission outcome** appears in the table.
- AS TOTAL_NUMBER gives the count a clear **label or alias**.

GROUP BY MISSION_OUTCOME

- This groups all rows by their MISSION_OUTCOME value **before counting**.
- So all rows with "Success" are grouped together, "Failure" together, and so on.

Boosters Carried Maximum Payload

```
%sql SELECT "Booster_Version",Payload, "PAYLOAD_MASS_KG_" FROM SPACEXTBL WHERE "PAYLOAD_MASS_KG_" = (SELECT MAX("PAYLOAD_MASS_KG_") FROM SPACEXTBL)
```

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

Booster_Version	Payload	PAYLOAD_MASS_KG_
F9 B5 B1048.4	Starlink 1 v1.0, SpaceX CRS-19	15600
F9 B5 B1049.4	Starlink 2 v1.0, Crew Dragon in-flight abort test	15600
F9 B5 B1051.3	Starlink 3 v1.0, Starlink 4 v1.0	15600
F9 B5 B1056.4	Starlink 4 v1.0, SpaceX CRS-20	15600
F9 B5 B1048.5	Starlink 5 v1.0, Starlink 6 v1.0	15600
F9 B5 B1051.4	Starlink 6 v1.0, Crew Dragon Demo-2	15600
F9 B5 B1049.5	Starlink 7 v1.0, Starlink 8 v1.0	15600
F9 B5 B1060.2	Starlink 11 v1.0, Starlink 12 v1.0	15600
F9 B5 B1058.3	Starlink 12 v1.0, Starlink 13 v1.0	15600
F9 B5 B1051.6	Starlink 13 v1.0, Starlink 14 v1.0	15600
F9 B5 B1060.3	Starlink 14 v1.0, GPS III-04	15600
F9 B5 B1049.7	Starlink 15 v1.0, SpaceX CRS-21	15600

- Select Booster Version, WHERE PAYLOAD_MASS_KG_ = (SELECT MAX(...))
Filters only the row(s) where payload mass equals the **maximum found in the table**

2015 Launch Records

```
8]: %sql SELECT Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL WHERE Landing_Outcome = 'Failure (drone ship)' AND strftime('%Y', Date) = '2015';
```

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

```
Done.
```

```
8]:
```

Landing_Outcome	Booster_Version	Launch_Site
-----------------	-----------------	-------------

Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
----------------------	---------------	-------------

Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
----------------------	---------------	-------------

- Code to Get the **month**, **landing outcome**, **booster version**, and **launch site** for all launches in **2015** where the **booster landing failed on a drone ship**

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

```
1]: %sql SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY TOTAL_NUMBER DE
* sqlite:///my_data1.db
Done.
```

```
1]:
```

Landing_Outcome	TOTAL_NUMBER
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

SELECT LANDING_OUTCOME, COUNT(LANDING_OUTCOME) AS TOTAL_NUMBER

- This selects two things:
 - The **type of landing outcome** (e.g., *Success (drone ship)*, *Failure (ocean)*, etc.).
 - The **number of times** each landing outcome occurred, using COUNT() function.
 - It renames the count column as TOTAL_NUMBER.

WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'

- This filters the data to only include launches that happened **between June 4, 2010, and March 20, 2017** (both dates inclusive).
- **GROUP BY LANDING_OUTCOME** This groups the filtered data **based on unique landing outcomes**, so the count can be calculated **per outcome**.
- **ORDER BY TOTAL_NUMBER DESC** Finally, it sorts the result by the count in **descending order**, so the most frequent outcome appears at the top.

A satellite view of Earth from space, showing the curvature of the planet and city lights at night. The image is a composite of a solid blue rectangle on the left and a satellite photograph of Earth on the right. The Earth is shown from a high altitude, with the horizon line curving across the frame. The night side of the Earth is visible, with numerous bright yellow and orange lights from cities and towns scattered across the dark landmasses. The atmosphere is visible as a thin blue layer along the horizon.

Section 3

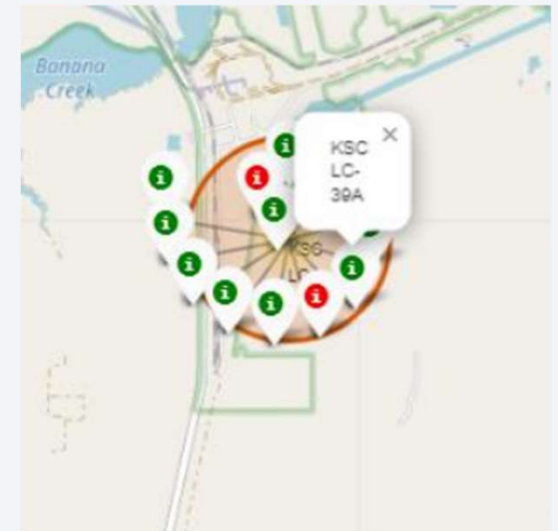
Launch Sites Proximities Analysis

All Launch location Markers in global map



- All the locations are near coast
- Aso nearer to the equator at around 3000Kms

Launch Outcome with color markers for launch locations



- In the Eastern coast (Florida) Launch site **KSC LC-39A** – (Kennedy Space Center, Launch Complex) has relatively high success rates compared to CCAFS SLC-40 & CCAFS LC-40.

Proximities of Launch locations



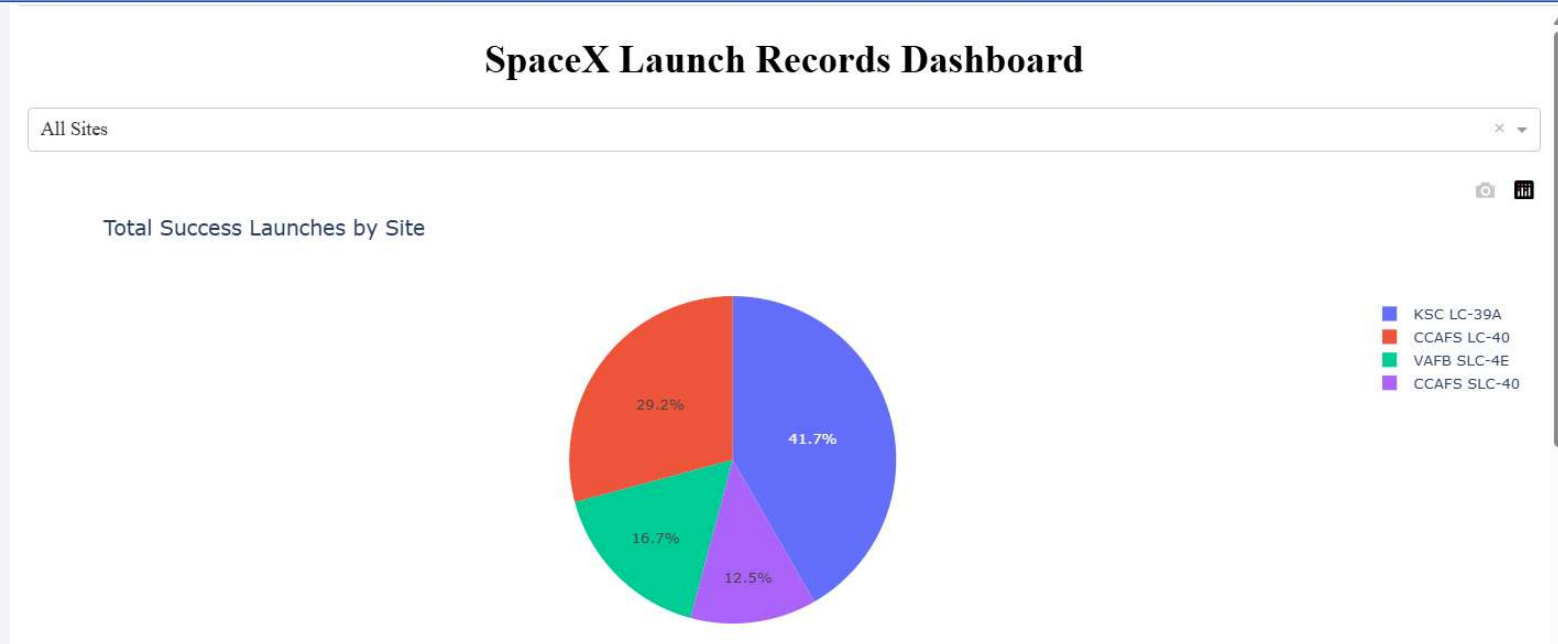
- Distance from Coast line – 0.86 KM
- Distance from Highway - 0.58 KM



Section 4

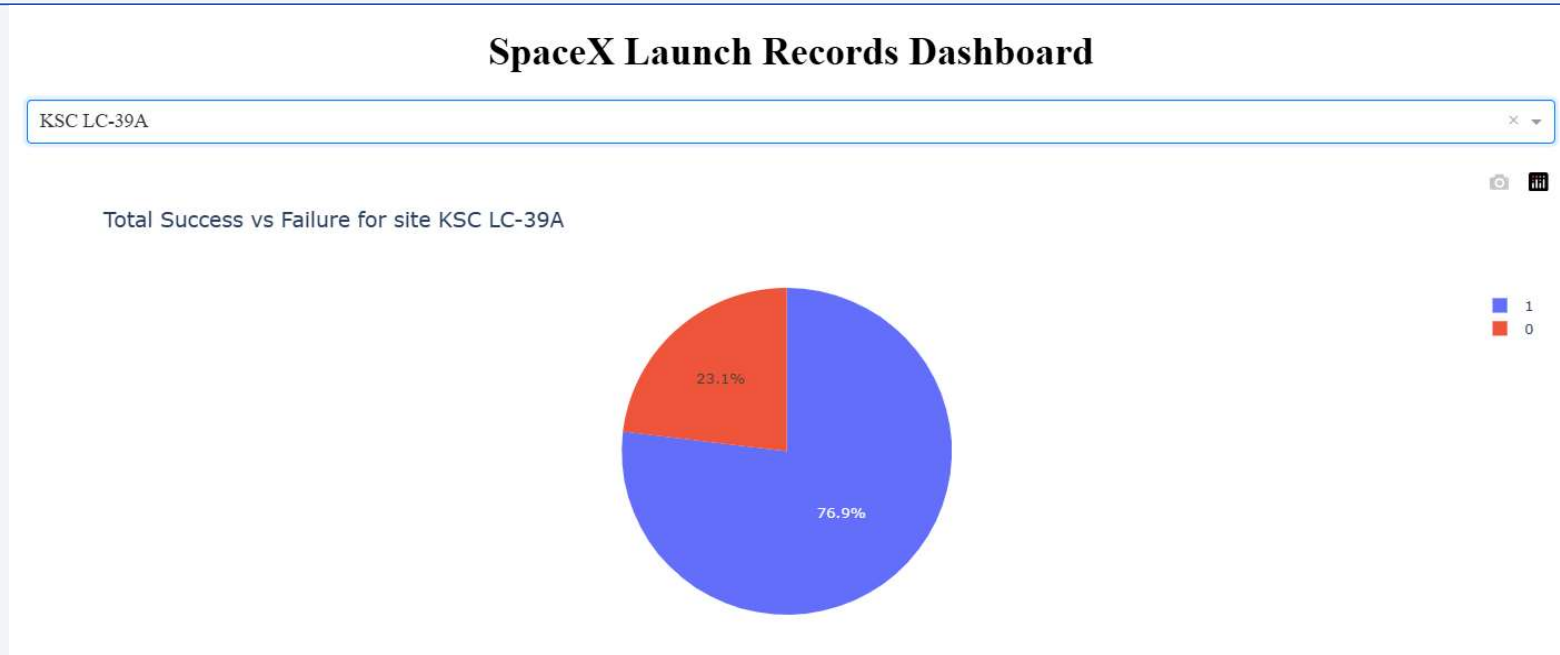
Build a Dashboard with Plotly Dash

Interactive Pie-Chart on Launch success rate



- Launch site KSC LC-39A has the highest launch success rate at 42% followed by CCAFS LC-40 at 29%, VAFB SLC-4E at 17% and lastly launch site CCAFS SLC-40 with a success rate of 13%

Interactive Pie-Chart on Launch success rate



- KSC LC-39A given highest success rate of 77%
- Followed by CCAFS – LC-40 having success rate of 73%

Interactive Scatterplot with Slider



- In Launch site CCAFS LC-40 the booster version FT has the largest success rate from a payload mass of >4000kg

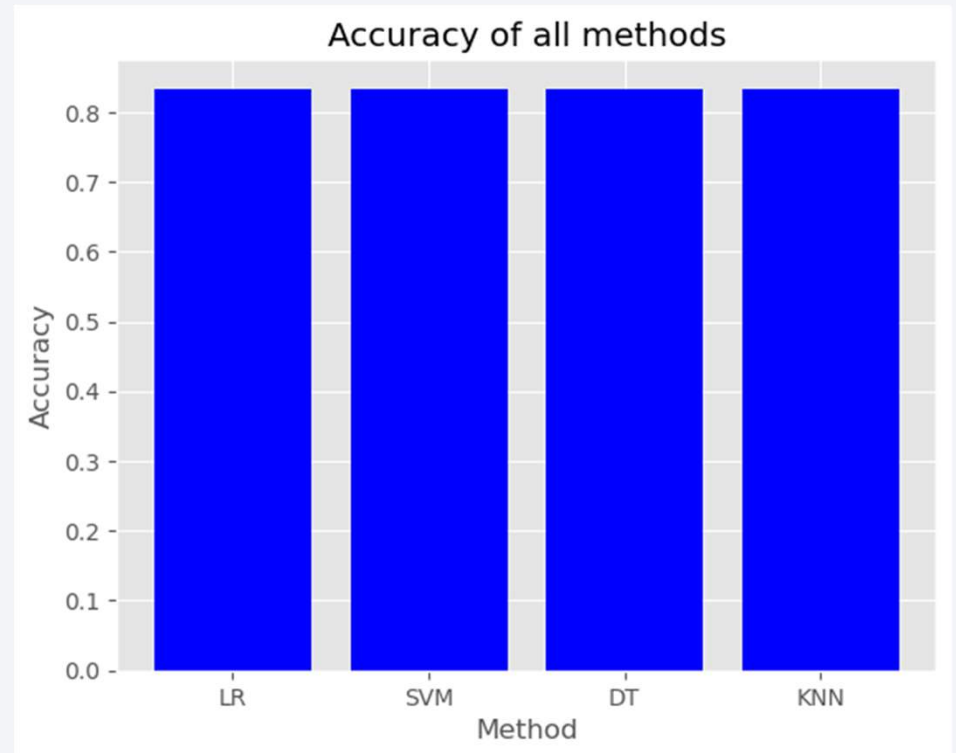


Section 5

Predictive Analysis (Classification)

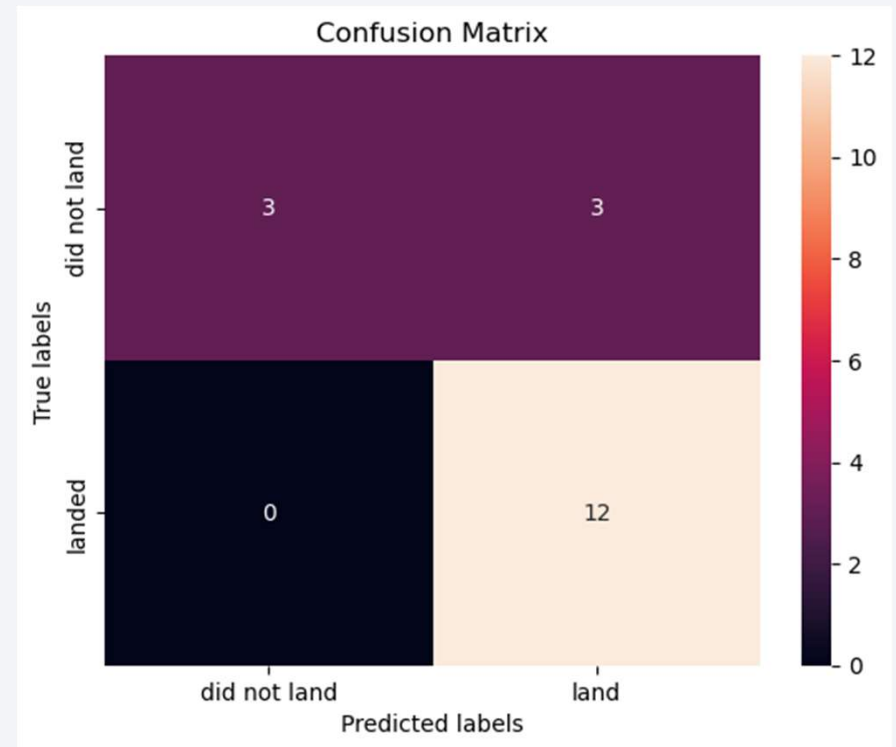
Classification Accuracy

- Logistic Regression, SVM, Decision Tree, KNN all models accuracy are 0.83



Confusion Matrix

- Similar to accuracy, confusion matrix also similar
- The problem is false positives
- Requires further research for prediction



Conclusions

- **Success Rate Trends:**
Launch success rate has **steadily increased since 2023**, indicating improved reliability and technological advancements.
- **Payload Impact:**
Missions with **higher payloads** tend to have **higher landing success rates**, especially those **above 8000 kg**.
- **Launch Sites Performance:**
 - After **Flight No. 80**, both **KSC LC-39A** and **CCAFS SLC-40** have achieved a **100% success rate**.
 - Indicates consistent reliability post mid-phase launches.
- **Orbital Success:**
Missions targeting **ES-L1, GEO, HEO, and SSO** orbits have achieved a **100% success rate**, suggesting favorable conditions or mature mission profiles.
- **Predictive Model Accuracy:**
 - The current model shows an **accuracy of 83%**.
 - However, **false negatives** exist in the **confusion matrix**, signaling the need for:
 - Better model tuning
 - Incorporating **new features or attributes**
 - Exploring **alternative algorithms** or **ensemble approaches**

Appendix

- Github links:
 - [https://github.com/sathyanandanp/CAPSTONE/blob/main/SpaceX Machine%20Learning%20Prediction Part 5%20\(1\).ipynb](https://github.com/sathyanandanp/CAPSTONE/blob/main/SpaceX%20Machine%20Learning%20Prediction%20Part%205%20(1).ipynb).
 - [https://github.com/sathyanandanp/CAPSTONE/blob/main/Updated SpaceX Dashboard.ipynb](https://github.com/sathyanandanp/CAPSTONE/blob/main/Updated%20SpaceX%20Dashboard.ipynb)
 - [https://github.com/sathyanandanp/CAPSTONE/blob/main/edadataviz.ipynb](https://github.com/sathyanandanp/CAPSTONE/blob/main/eda-dataviz.ipynb)
 - [https://github.com/sathyanandanp/CAPSTONE/blob/main/jupyter-labs-eda-sql-coursera sqlite.ipynb](https://github.com/sathyanandanp/CAPSTONE/blob/main/jupyter-labs-eda-sql-coursera-sqlite.ipynb)
 - <https://github.com/sathyanandanp/CAPSTONE/blob/main/jupyter-labs-spacex-data-collection-api.ipynb>
 - <https://github.com/sathyanandanp/CAPSTONE/blob/main/jupyter-labs-webscraping.ipynb>
 - https://github.com/sathyanandanp/CAPSTONE/blob/main/lab_jupyter_launch_site_location.ipynb
 - <https://github.com/sathyanandanp/CAPSTONE/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb>
- Data from:
 - [https://api.spacexdata.com/v4/rockets/"+str\(x\)\).json\(\)](https://api.spacexdata.com/v4/rockets/)
 - [https://en.wikipedia.org/wiki/List of Falcon 9 and Falcon Heavy launches](https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches)

Thank you!

