IBM Developer SKILLS NETWORK

IBM DATA SCIENCE CAPSTONE PROJECT

Sri Harshitha Anugu

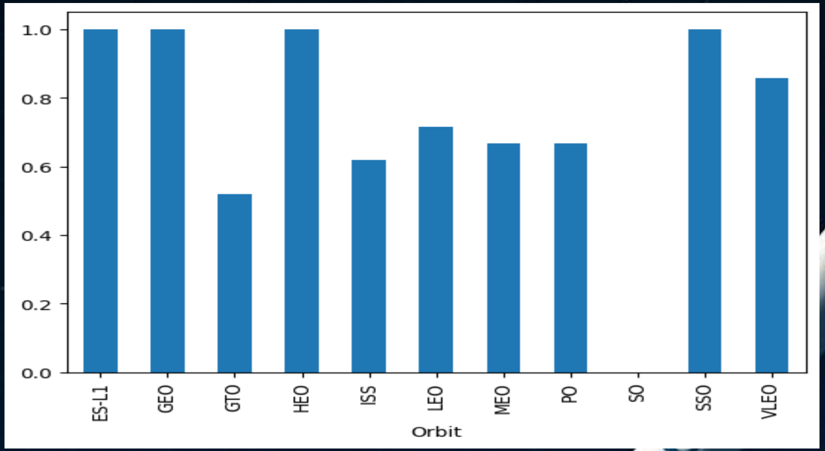
Outline

Executive Summary

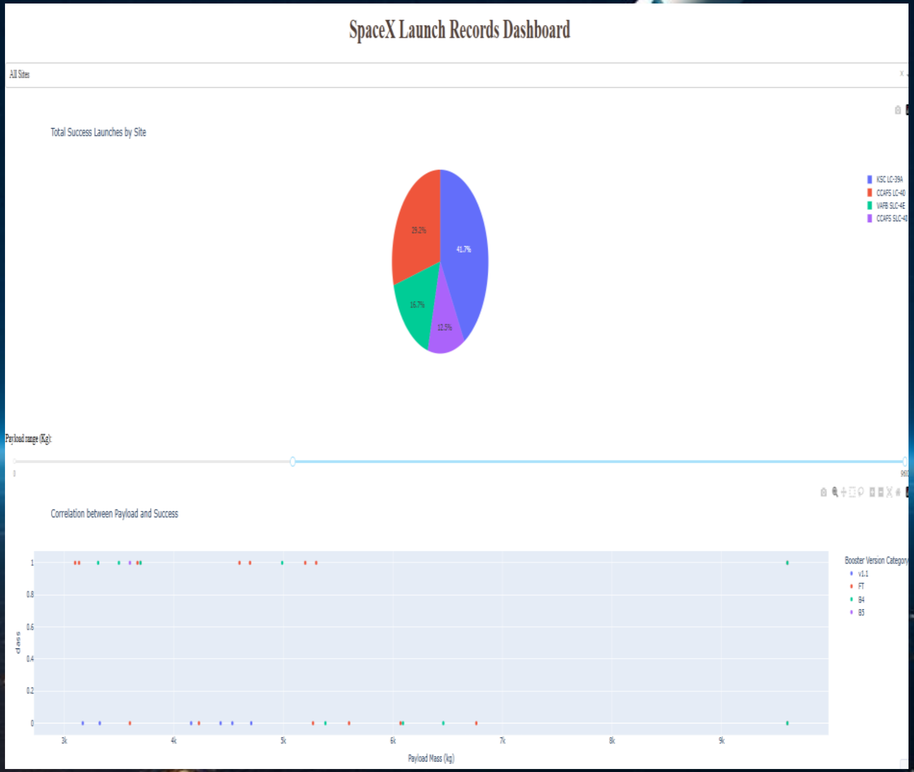
- Used SpaceX REST API and web scraping for data collection
- Processed and wrangled Falcon 9 launch data
- Performed EDA with visualizations & SQL
- Built Folium maps & Plotly Dash dashboards
- Developed classification models to predict landing outcomes
- Key finding: Falcon 9 first stage landing success rate improves over time

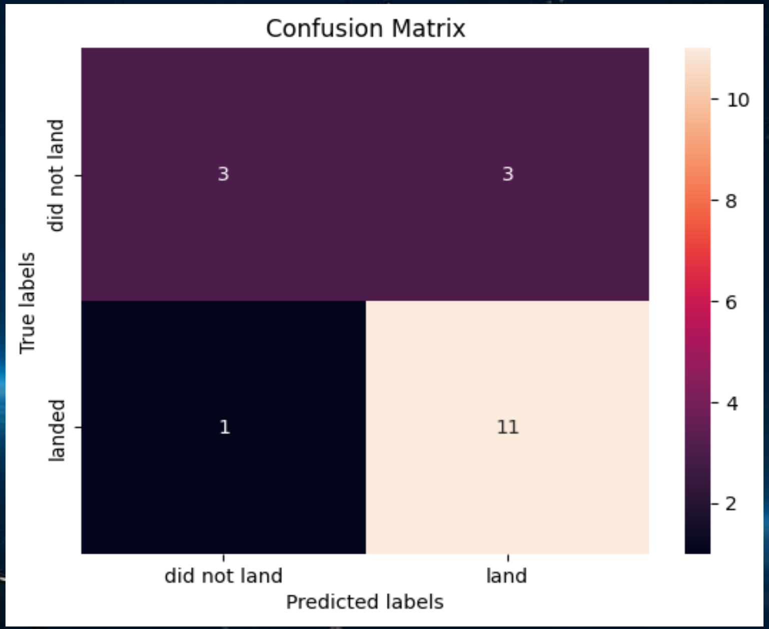
Summary of results

1. Exploratory Data Analysis(EDA) results
2. Geospatial analytics
3. Interactive dashboard
4. Predictive analysis



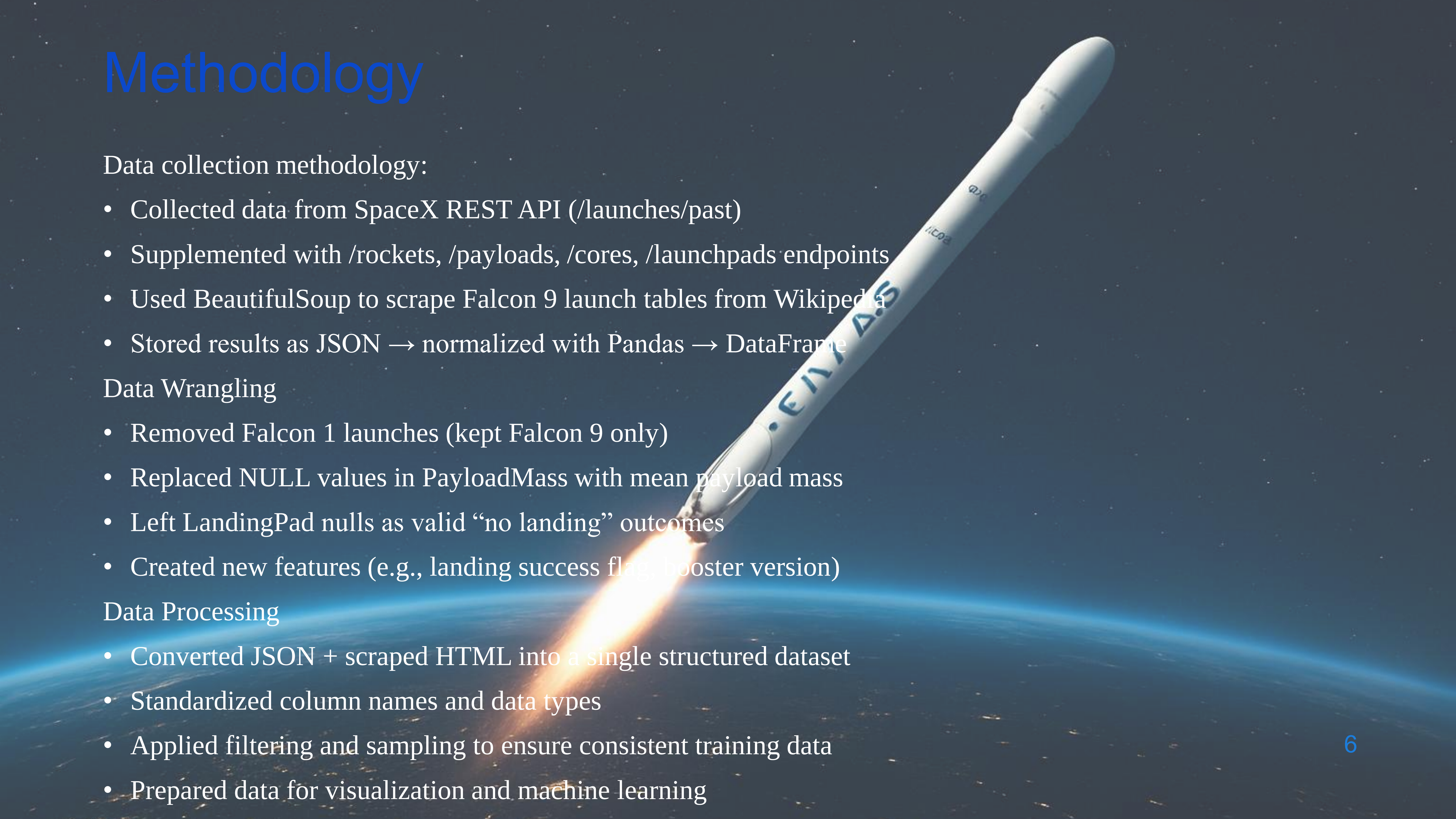
CCAFS SLC-40
26
CCAFS LC-40





Introduction

Project background: Predict SpaceX Falcon 9 landing success
Problem: Can we predict if a first stage landing attempt will succeed?
Dataset: SpaceX API + web scraped Falcon 9 Wiki tables
Goal: Support cost reduction insights for rocket reusability



Methodology

Data collection methodology:

- Collected data from SpaceX REST API (/launches/past)
- Supplemented with /rockets, /payloads, /cores, /launchpads endpoints
- Used BeautifulSoup to scrape Falcon 9 launch tables from Wikipedia
- Stored results as JSON → normalized with Pandas → DataFrame

Data Wrangling

- Removed Falcon 1 launches (kept Falcon 9 only)
- Replaced NULL values in PayloadMass with mean payload mass
- Left LandingPad nulls as valid “no landing” outcomes
- Created new features (e.g., landing success flag, booster version)

Data Processing

- Converted JSON + scraped HTML into a single structured dataset
- Standardized column names and data types
- Applied filtering and sampling to ensure consistent training data
- Prepared data for visualization and machine learning

Data Collection Process

Key Phrases:
•**Identify Sources** → Reliable datasets, APIs, Wikipedia, Kaggle
•**Collect Data** → CSV, JSON, SQL queries, Web scraping
•**Store Data** → Local storage, Cloud, Database
•**Validate Data** → Quality check, consistency, accuracy
•**Clean & Preprocess** → Handle missing values, normalize, remove duplicates

Flowchart (ToptoBottom):
[Identify Sources]
↓
[Collect Data]
↓
[Clean & Preprocess]
↓
[Store Data]
↓
[Validate Data]

Data Collection

API endpoint: https://api.spacexdata.com/v4/launches/past

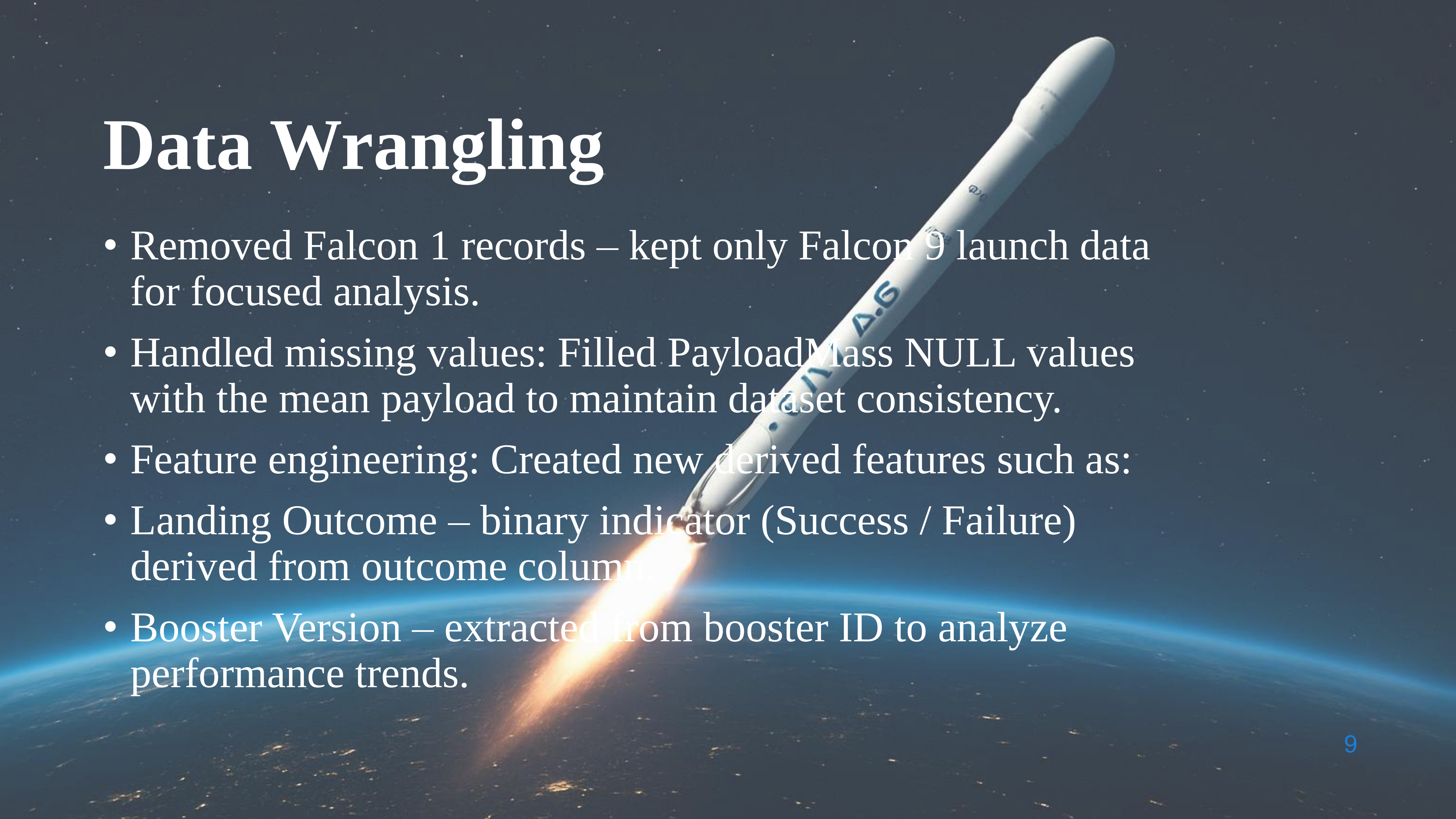
JSON response normalized into DataFrame

Supplementary API calls for rockets, payloads, cores, and launchpads

Web scraping from Wikipedia Falcon 9 launch history tables

GitHub URL:

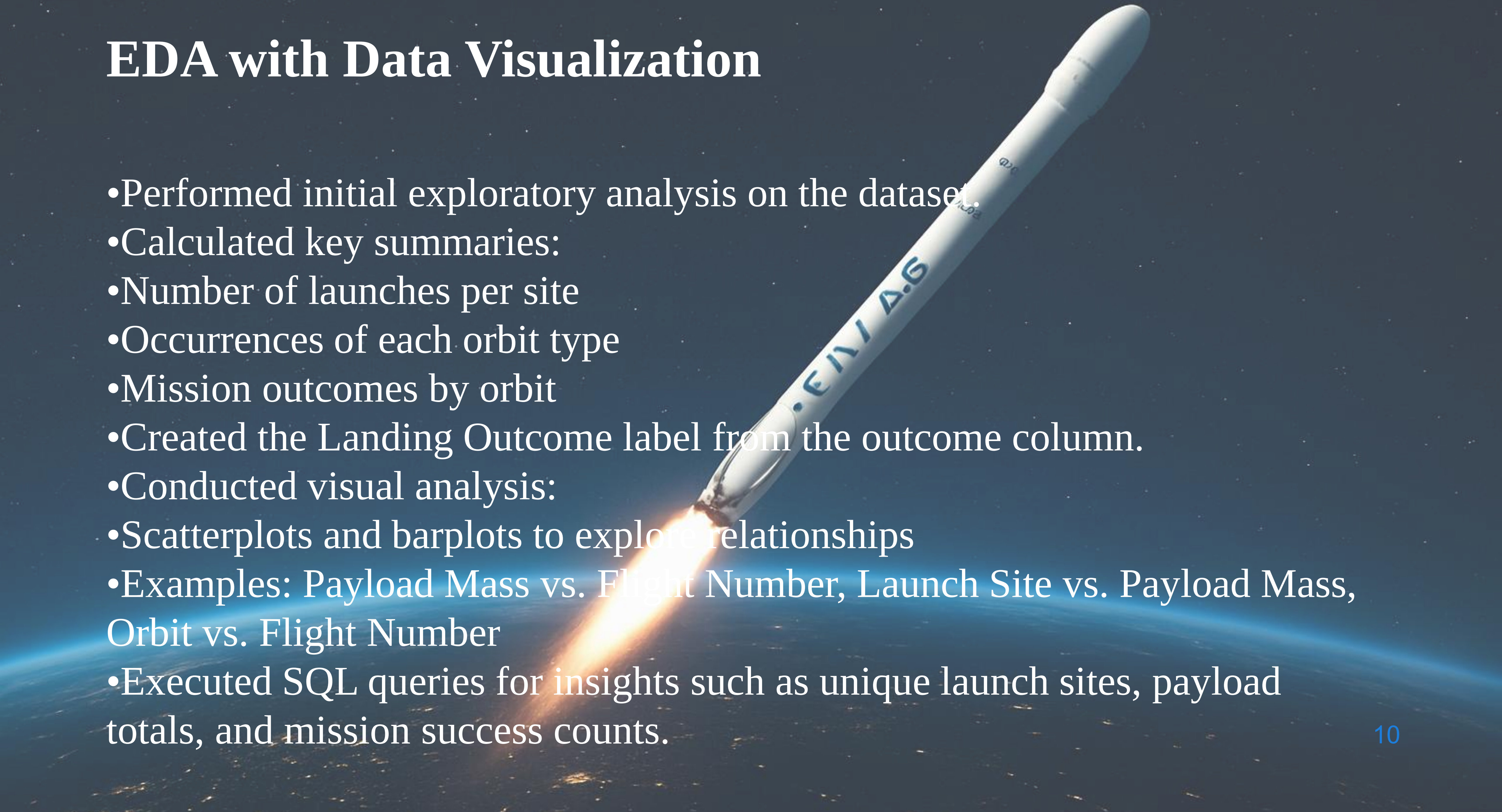
https://github.com/harshithaanugu5/Coursera_capstone_DataScience/blob/main/Data_Collection_API.ipynb



Data Wrangling

- Removed Falcon 1 records – kept only Falcon 9 launch data for focused analysis.
- Handled missing values: Filled PayloadMass NULL values with the mean payload to maintain dataset consistency.
- Feature engineering: Created new derived features such as:
- Landing Outcome – binary indicator (Success / Failure) derived from outcome column.
- Booster Version – extracted from booster ID to analyze performance trends.

EDA with Data Visualization



•Performed initial exploratory analysis on the dataset.
•Calculated key summaries:
•Number of launches per site
•Occurrences of each orbit type
•Mission outcomes by orbit
•Created the Landing Outcome label from the outcome column.
•Conducted visual analysis:
•Scatterplots and barplots to explore relationships
•Examples: Payload Mass vs. Flight Number, Launch Site vs. Payload Mass, Orbit vs. Flight Number
•Executed SQL queries for insights such as unique launch sites, payload totals, and mission success counts.

EDA with SQL

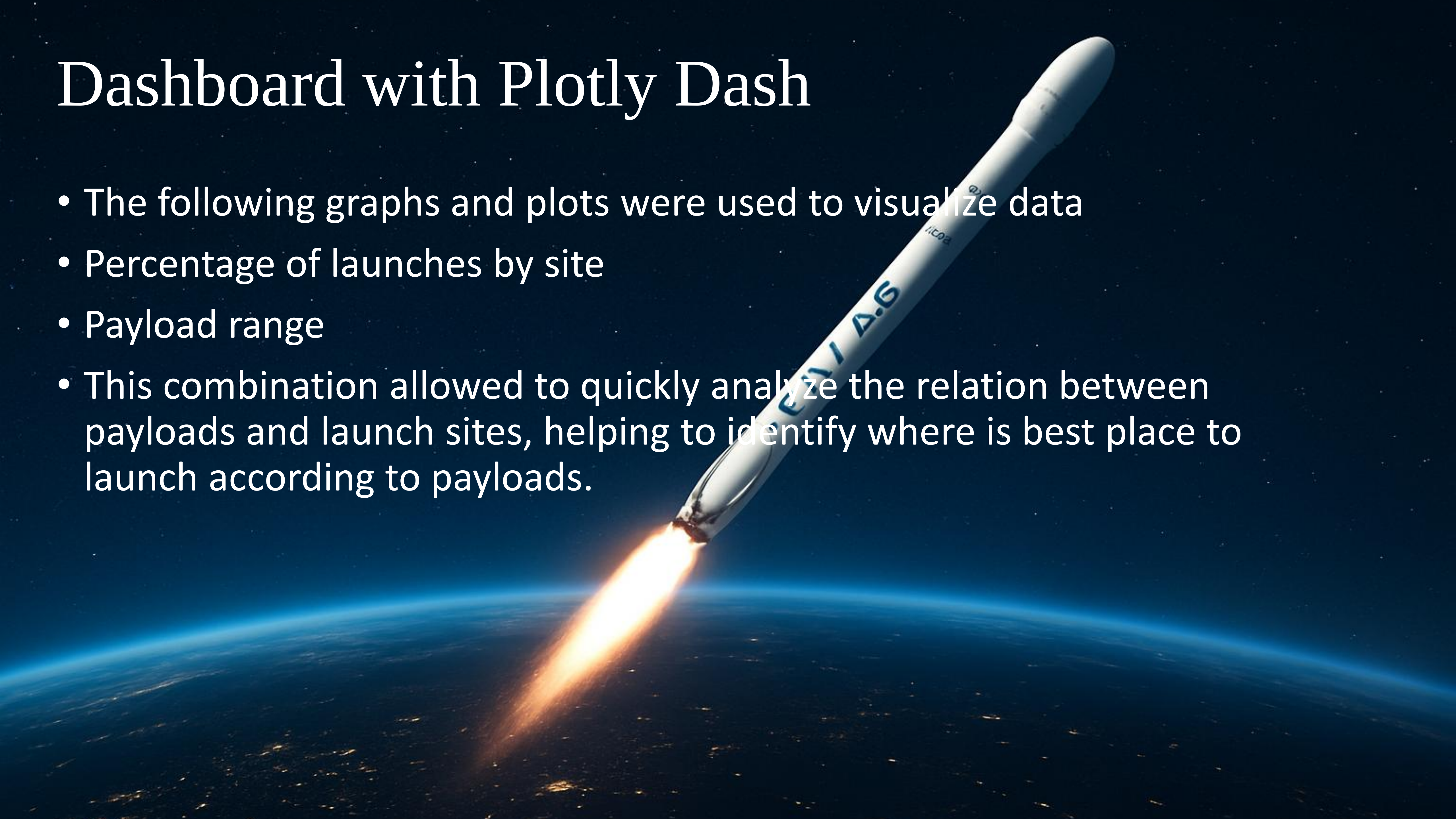
- Queried launch sites, payload stats, mission outcomes
- Examples:
- Total payload mass for NASA launches
- Average payload for booster version F9 v1.1
- Successful drone ship landings with payload 4000–6000
- Failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015; and
- Rank 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
- GitHub URL: https://github.com/harshithaanugu5/Coursera_capstone_DataScience/blob/main/EDA_SQL_Coursera.ipynb

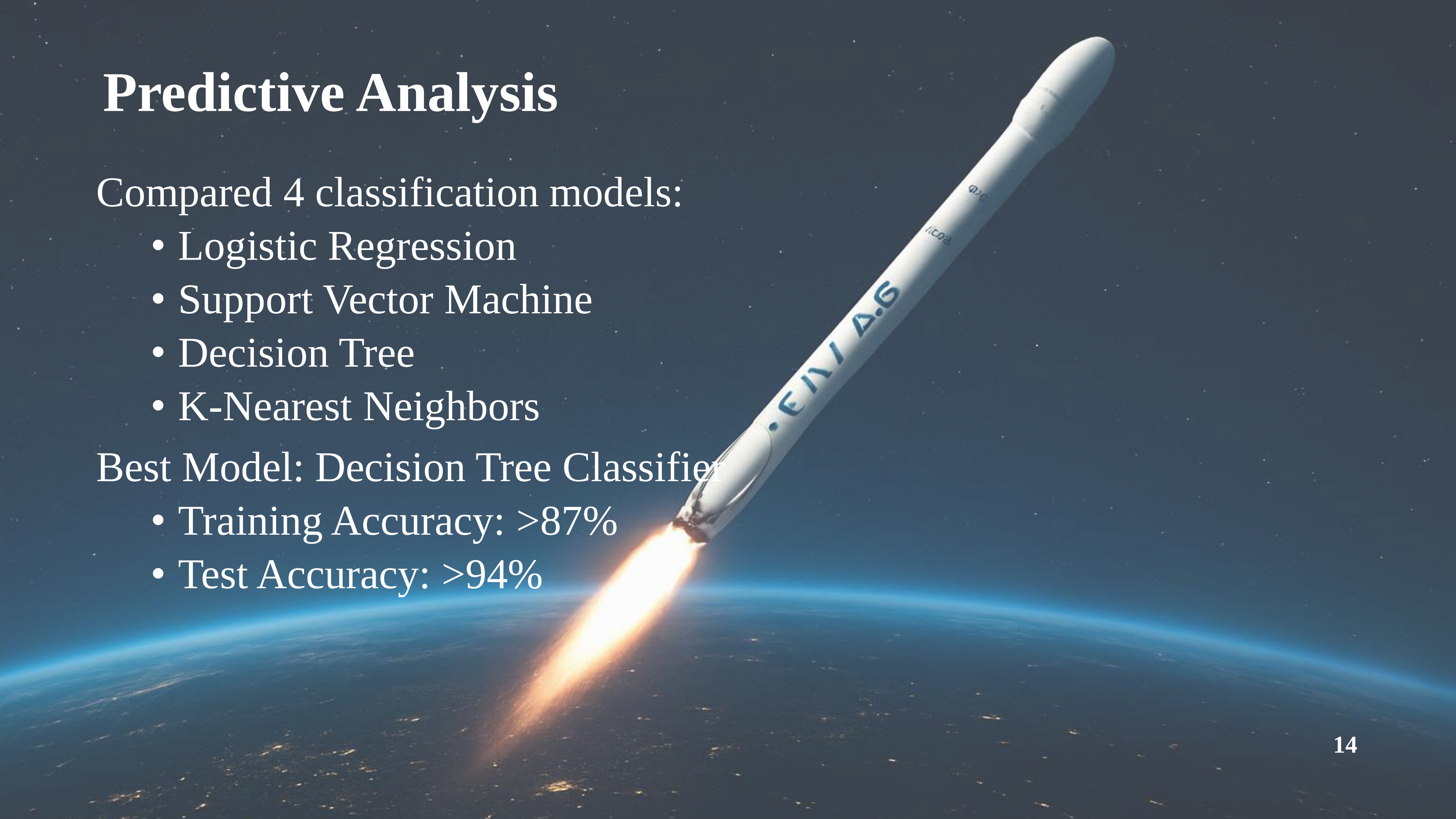
Interactive Map with Folium

- Mapped launch site locations worldwide
- Added markers for launch outcomes
- Circles indicate highlighted areas around specific coordinates, like NASA Johnson Space Center;
- Marker clusters indicates groups of events in each coordinate, like launches in a launch site; and • Lines are used to indicate distances between two coordinates
- Visualized proximities: railways, highways, coastlines
- GitHub URL: https://github.com/harshithaanugu5/Coursera_capstone_DataScience/blob/main/Interactive_Visual_analytics_with_folium.ipynb

Dashboard with Plotly Dash

- The following graphs and plots were used to visualize data
- Percentage of launches by site
- Payload range
- This combination allowed to quickly analyze the relation between payloads and launch sites, helping to identify where is best place to launch according to payloads.





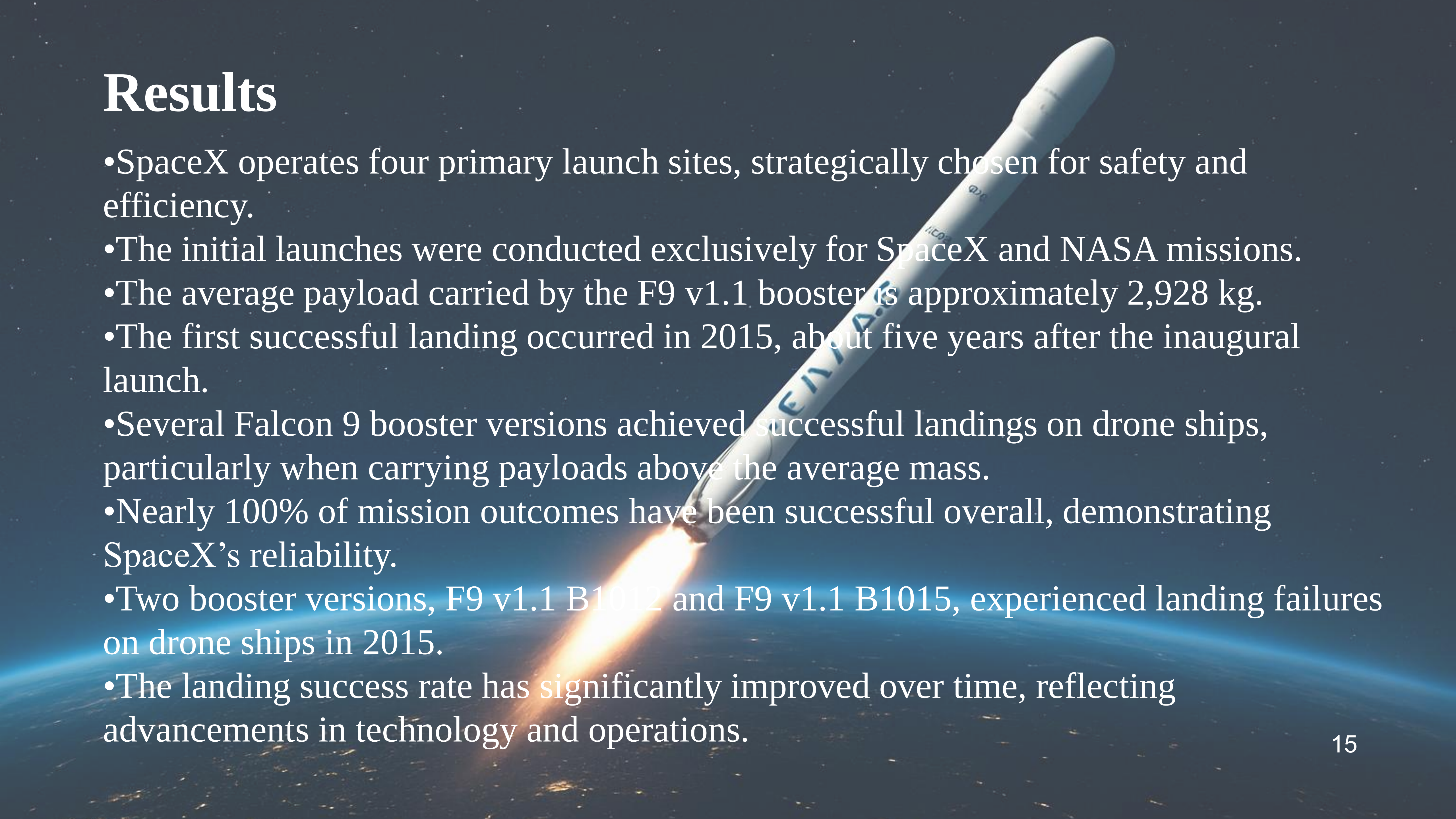
Predictive Analysis

Compared 4 classification models:

- Logistic Regression
- Support Vector Machine
- Decision Tree
- K-Nearest Neighbors

Best Model: Decision Tree Classifier

- Training Accuracy: >87%
- Test Accuracy: >94%



Results

•SpaceX operates four primary launch sites, strategically chosen for safety and efficiency.
•The initial launches were conducted exclusively for SpaceX and NASA missions.
•The average payload carried by the F9 v1.1 booster is approximately 2,928 kg.
•The first successful landing occurred in 2015, about five years after the inaugural launch.
•Several Falcon 9 booster versions achieved successful landings on drone ships, particularly when carrying payloads above the average mass.
•Nearly 100% of mission outcomes have been successful overall, demonstrating SpaceX’s reliability.
•Two booster versions, F9 v1.1 B1012 and F9 v1.1 B1015, experienced landing failures on drone ships in 2015.
•The landing success rate has significantly improved over time, reflecting advancements in technology and operations.

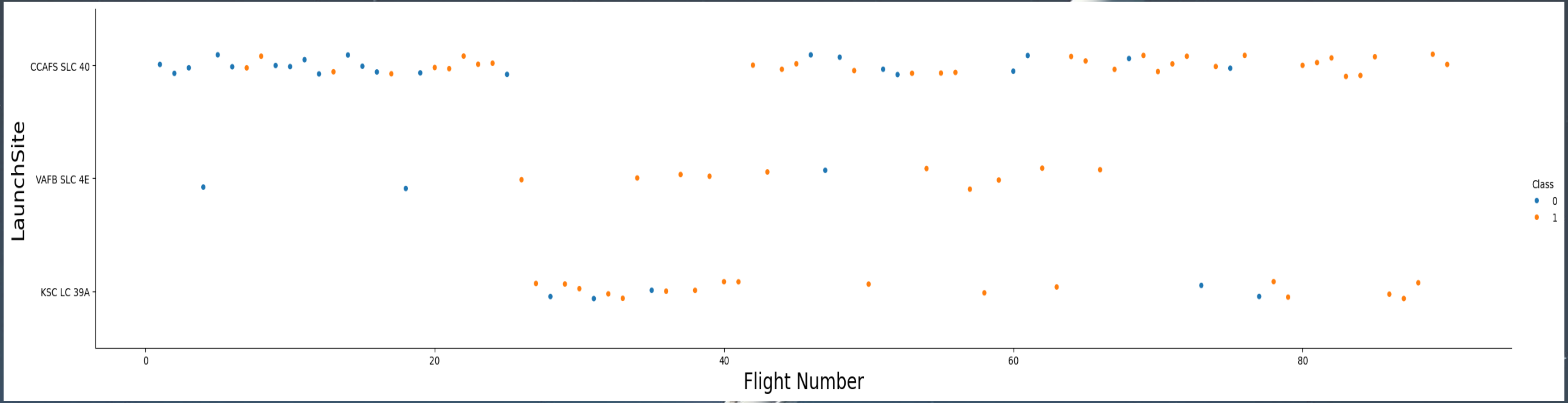
Results

•Interactive analytics revealed that launch sites are typically near coastal regions for safety and are supported by robust logistical infrastructure.
•Predictive modeling identified the Decision Tree Classifier as the most accurate model for predicting landing success, achieving over 87% training accuracy and over 94% test accuracy.

Section 2

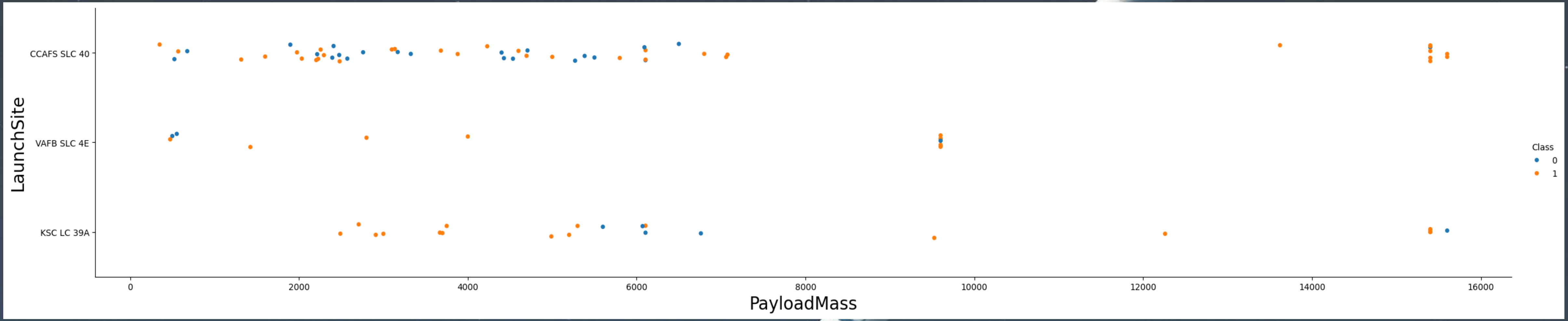
Insights drawn from EDA

Flight Number vs. Launch Site



•The plot shows that the **best-performing launch site** is **CCAFS SLC-40**, where the majority of recent launches have been successful.
•**VAFB SLC-4E** is the **second most successful site**, followed by **KSC LC-39A** in third place.
•The overall **success rate has improved significantly over time**, demonstrating continuous advancements in SpaceX's launch technology, operational efficiency, and landing precision

Payload vs. Launch Site



•Payloads above **9,000 kg** show a very high success rate.

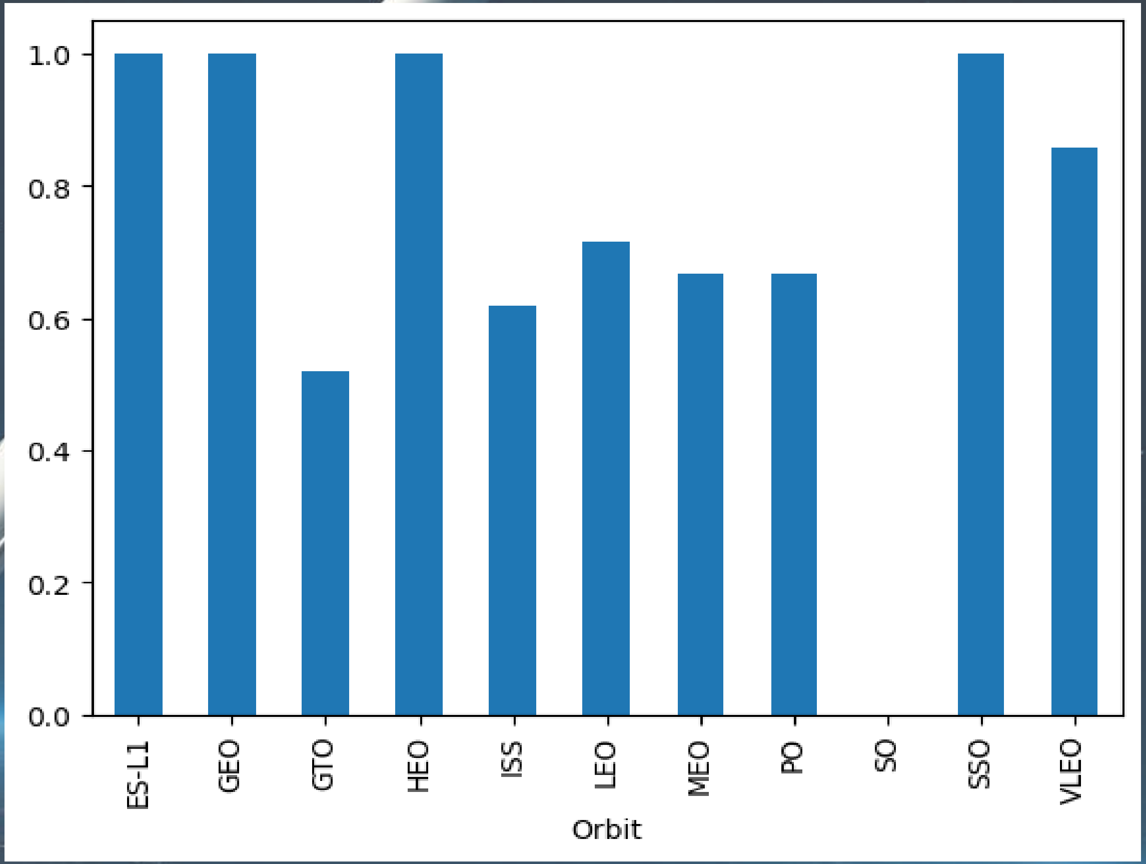
•Payloads over **12,000 kg** are launched only from **CCAFS SLC-40** and **KSC LC-39A**, indicating these sites handle heavier missions.

•**CCAFS SLC-40** is the most frequently used and successful site across different payload ranges.

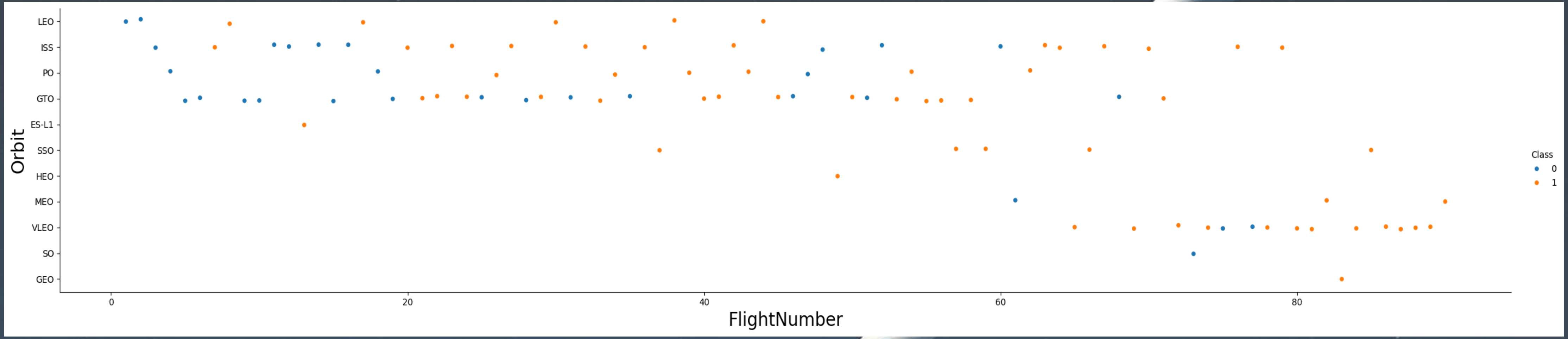
•Launch site choice significantly influences mission success, especially for heavier payloads.

Success Rate vs. Orbit Type

- The bar chart of Success Rate vs. Orbit Type shows that the following orbits have the highest (100%) success rate:
  - ES-L1 (Earth-Sun First Lagrangian Point)
  - GEO (Geostationary Orbit)
  - HEO (High Earth Orbit)
  - SSO (Sun-synchronous Orbit)
- The orbit with the lowest (0%) success rate is:
  - SO (Heliocentric Orbit)

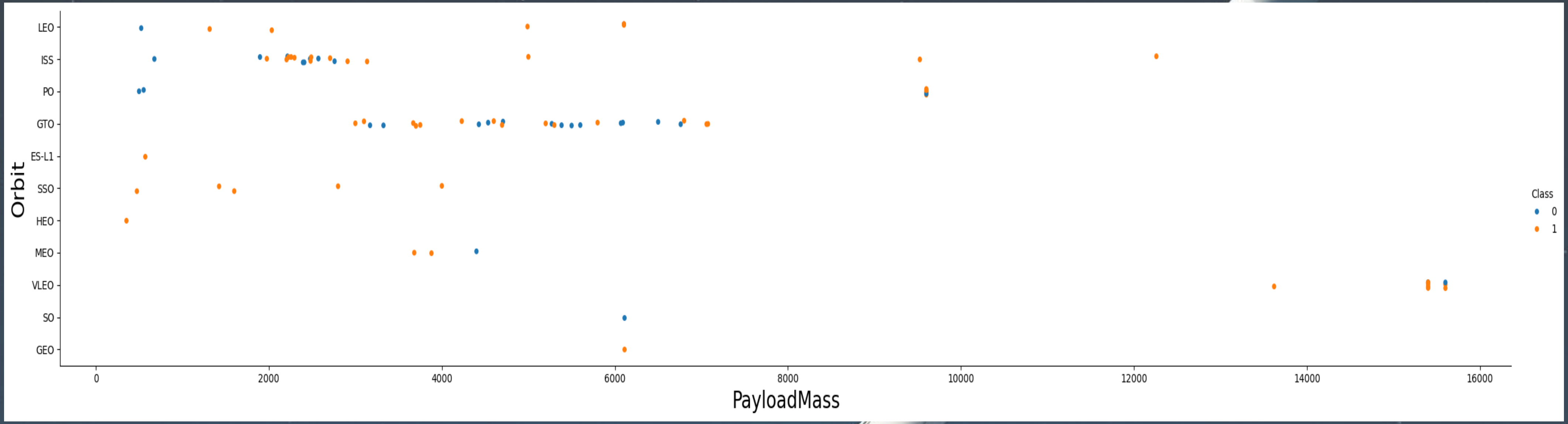


Flight Number vs. Orbit Type



- The **100% success rate** for **GEO**, **HEO**, and **ES-L1** orbits is due to only **one launch** into each of these orbits.
- The **SSO orbit** achieved a 100% success rate with **five successful launches**, which is more significant.
- There is **no clear relationship** between flight number and success rate for **GTO** missions.
- Overall, **success rates increase as flight numbers grow**, with **LEO** showing the most improvement — unsuccessful landings occurred only in the **early missions** with lower flight numbers.

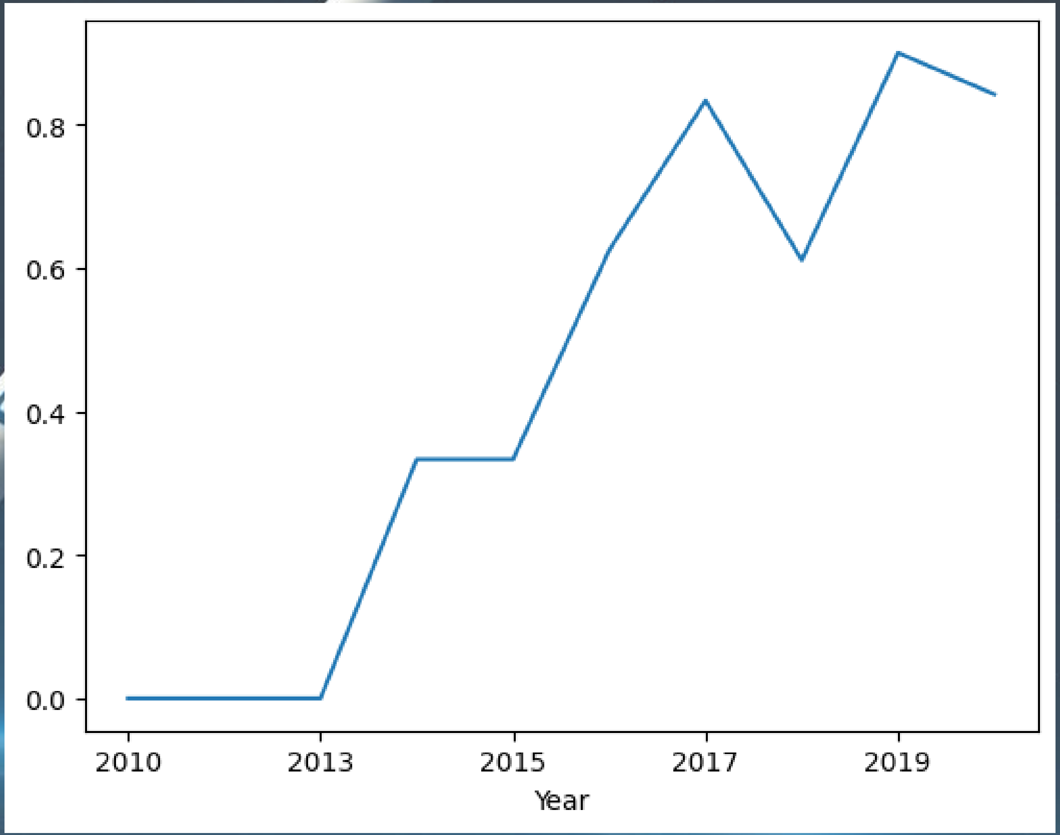
Payload vs. Orbit Type



•**PO, ISS, and LEO orbits** show higher success rates with heavier payloads, although PO has limited data points.
•**VLEO launches** are also associated with heavier payloads, which aligns with expectations.
•For **GTO**, there is no clear relationship between payload mass and success rate.
•The **ISS orbit** supports the widest range of payloads while maintaining a strong success rate.
•Very few launches have been recorded for **SO** and **GEO**, making it difficult to draw firm conclusions.

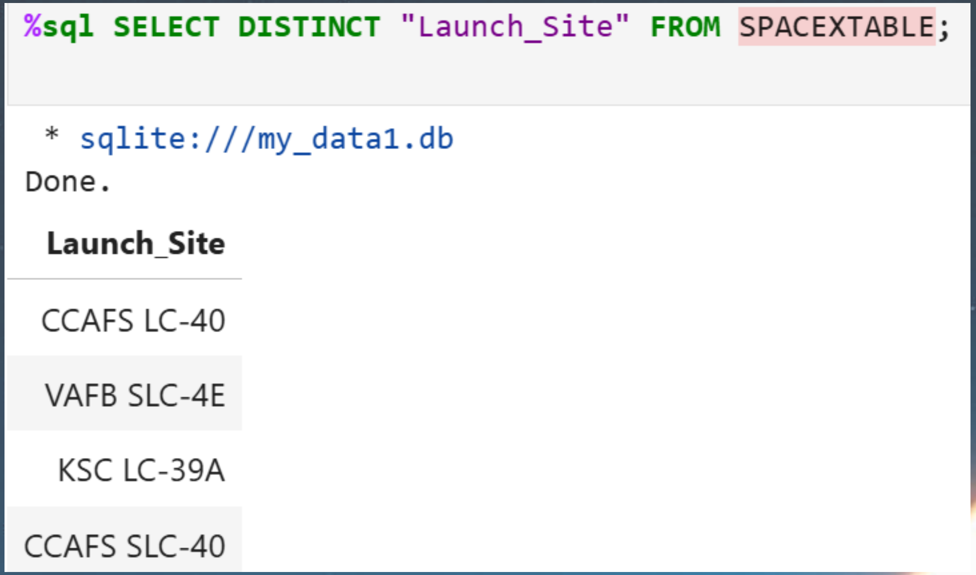
Launch Success Yearly Trend

- The line chart of yearly average success rate shows that:
  - Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
  - After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
  - After 2016, there was always a greater than 50% chance of success.



All Launch Site Names

- Find the names of the unique launch sites.

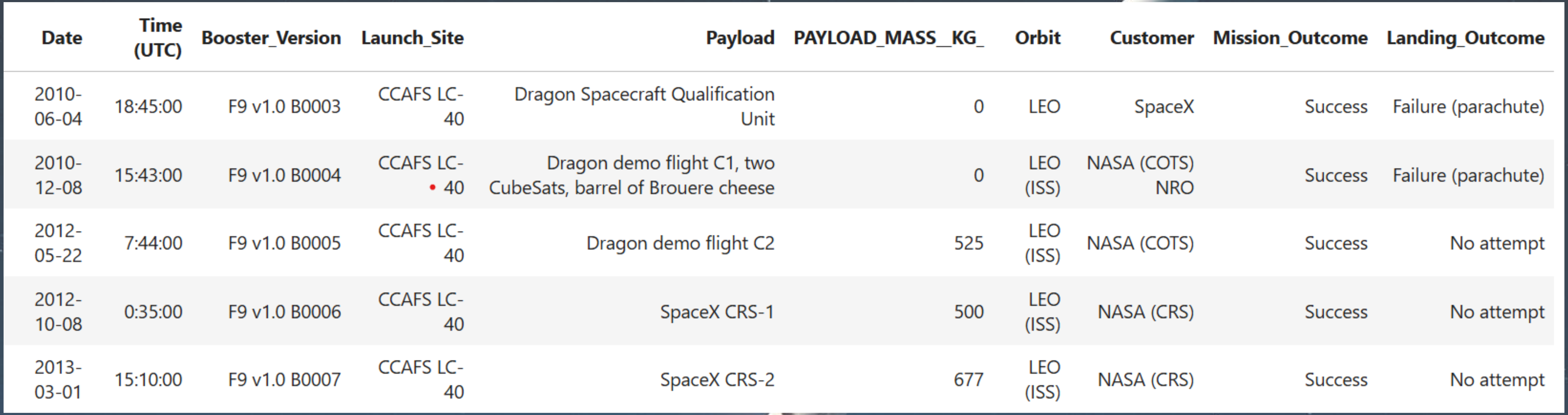


```
%sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;

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

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

The word DISTINCT returns only unique values from the Launch_site column of the SPACEXTBL table.

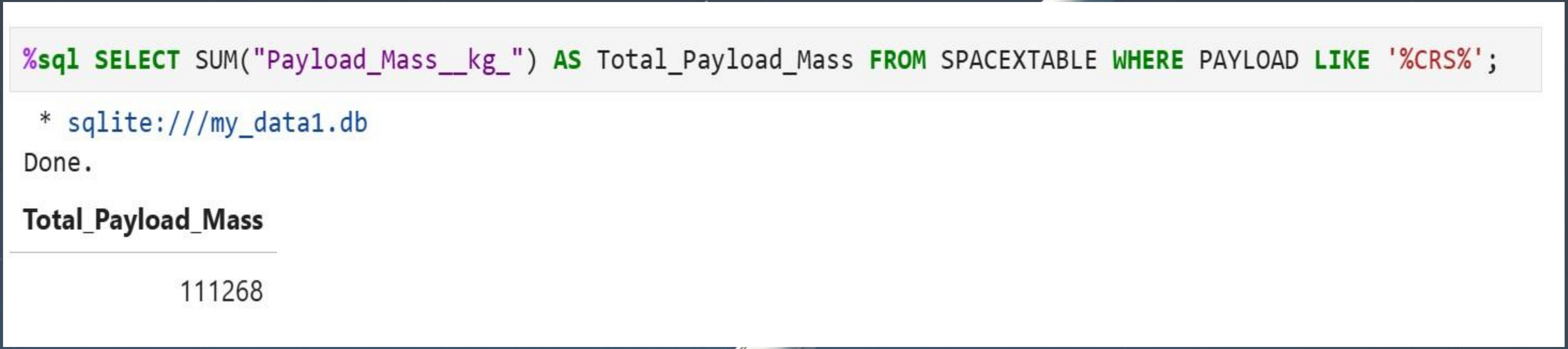
Launch Site Names Begin with 'CCA'



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

- The query retrieves the first **five launches** where the launch site name begins with **"CCA"**, showing early missions from **Cape Canaveral (CCAFS LC-40)** along with their payloads, orbits, mission outcomes, and landing results.

Total Payload Mass

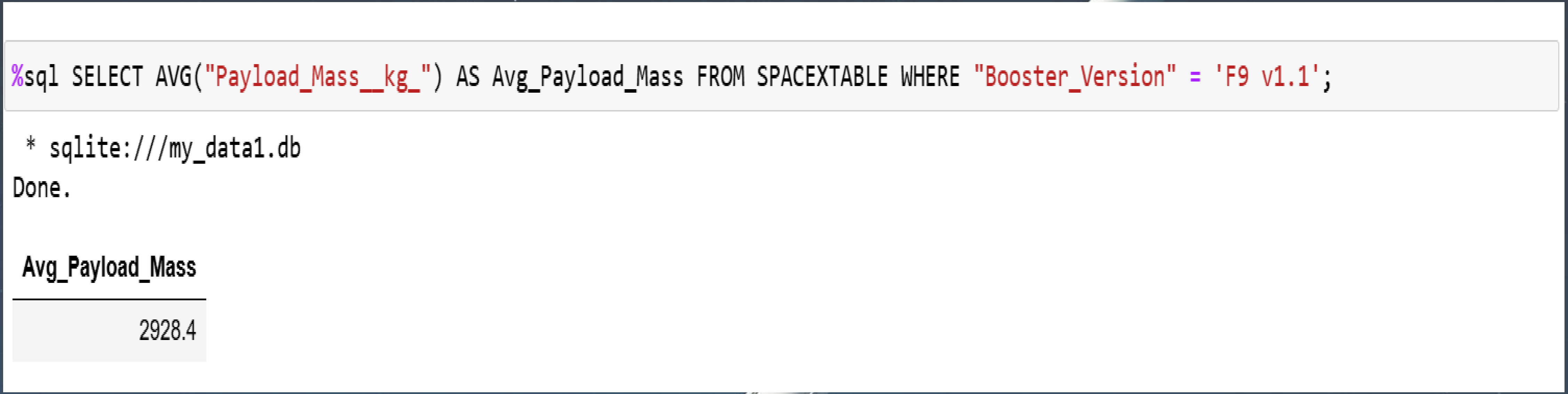


```
%sql SELECT SUM("Payload_Mass__kg_") AS Total_Payload_Mass FROM SPACEXTABLE WHERE PAYLOAD LIKE '%CRS%';

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

Total_Payload_Mass
111268

- The query calculates the **total payload mass** carried by boosters for NASA’s CRS missions by selecting all payloads containing **"CRS"**. It returns the overall payload launched for these missions.

Average Payload Mass by F9 v1.1

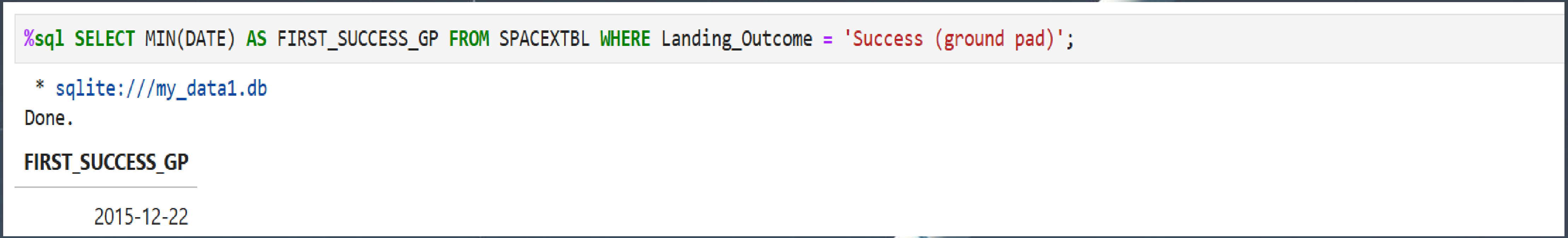


```
%sql SELECT AVG("Payload_Mass__kg_") AS Avg_Payload_Mass FROM SPACEXTABLE WHERE "Booster_Version" = 'F9 v1.1';

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

Avg_Payload_Mass
2928.4

- The query calculates the **average payload mass** for launches using the **F9 v1.1 booster version**, showing the typical payload capacity for this booster type. The result is approximately **2,928 kg**.

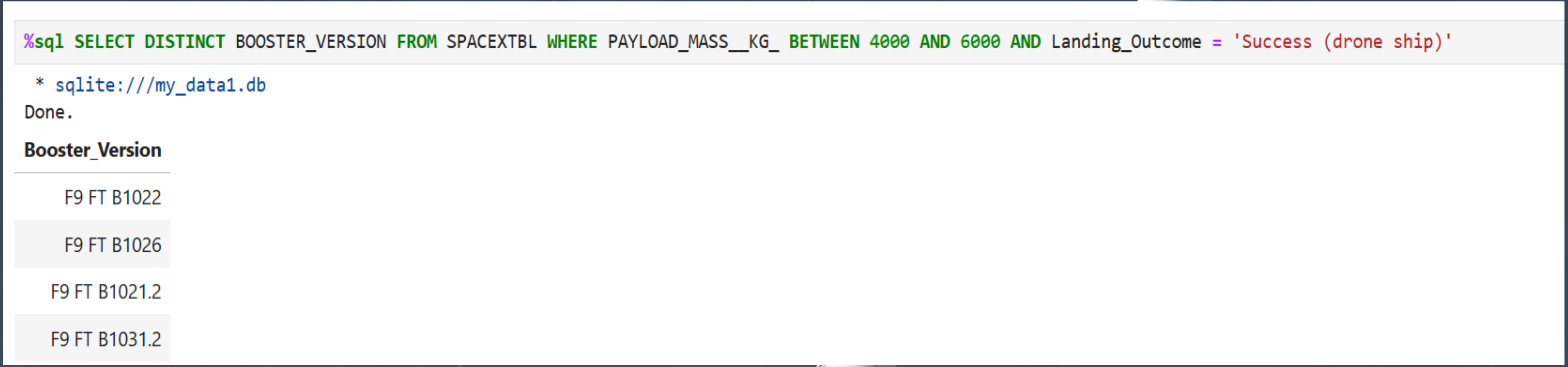
First Successful Ground Landing Date



```
%sql SELECT MIN(DATE) AS FIRST_SUCCESS_GP FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';

 * sqlite:///my_data1.db
Done.

FIRST_SUCCESS_GP
      2015-12-22
```

- The query finds the **earliest date** of a successful landing on a **ground pad**. The result shows that the **first successful ground landing** occurred on **December 22, 2015**.

Successful Drone Ship Landing with Payload between 4000 and 6000



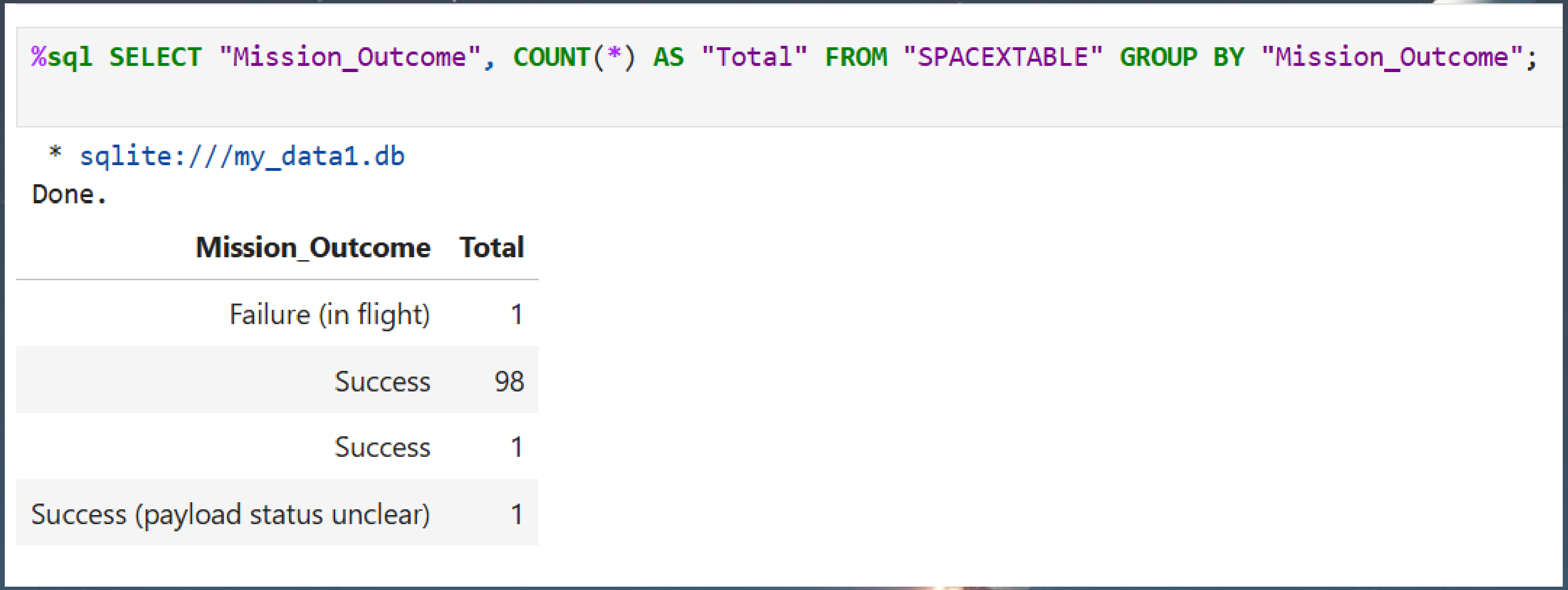
```
%sql SELECT DISTINCT BOOSTER_VERSION FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000 AND Landing_Outcome = 'Success (drone ship)'
```

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

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

- The WHERE keyword is used to filter the results to include only those that satisfy both conditions in the brackets (as the AND keyword is also used). The BETWEEN keyword allows for 4000 < x < 6000 values to be selected.
- Selecting distinct booster versions according to the filters above, these 4 are the result.

Total Number of Successful and Failure Mission Outcomes



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

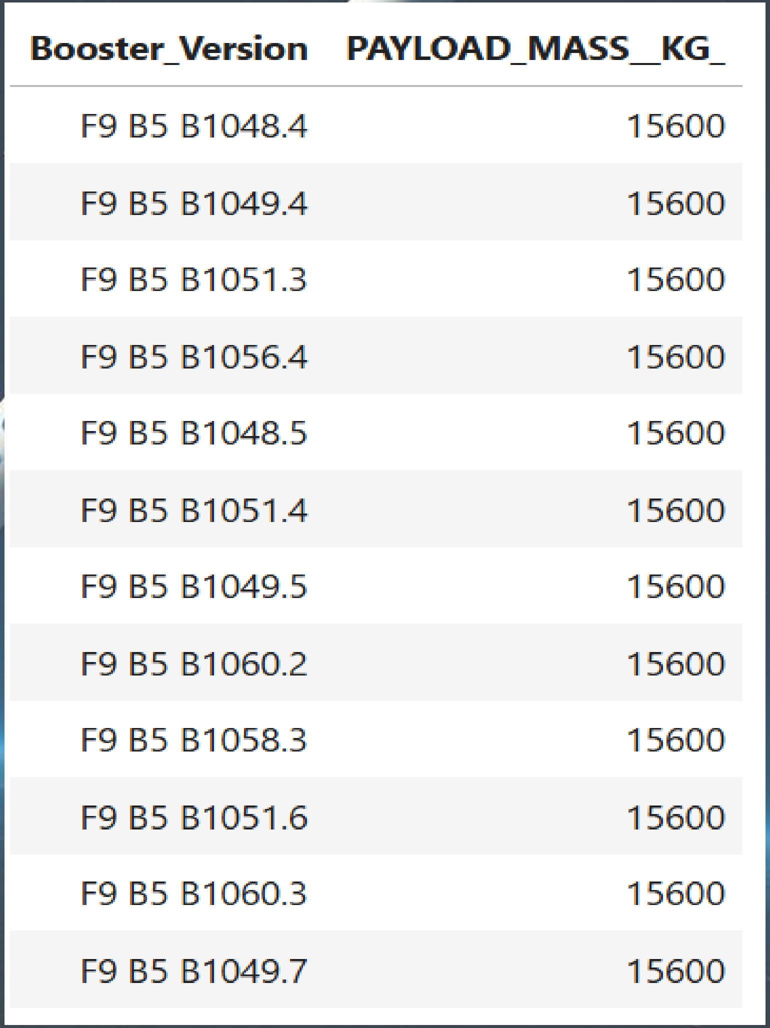
```
 * sqlite:///my_data1.db
Done.
```

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

This query selects each unique **mission outcome** from the SPACEXTABLE and uses **COUNT(*)** to calculate the **total number of occurrences** for each outcome by grouping the results with **GROUP BY**.

Boosters Carried Maximum Payload

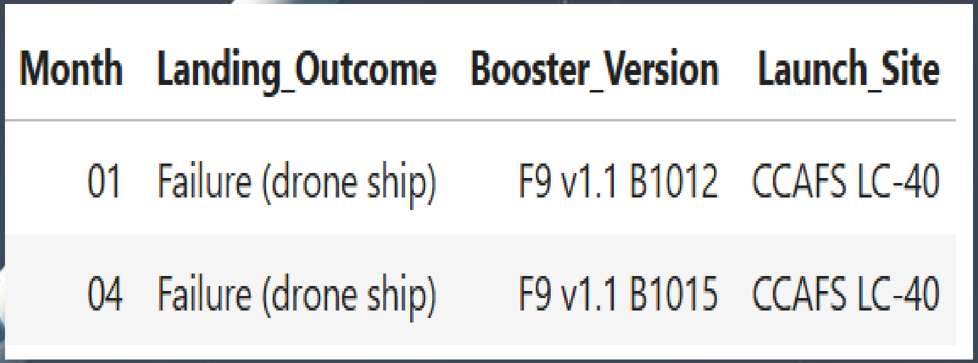
- This query retrieves the **names of booster versions** that have carried the **maximum payload mass** recorded in the dataset.
- The result includes boosters such as **F9 B5 B1048.4**, **F9 B5 B1049.5**, and others, showing which boosters handled the heaviest missions.



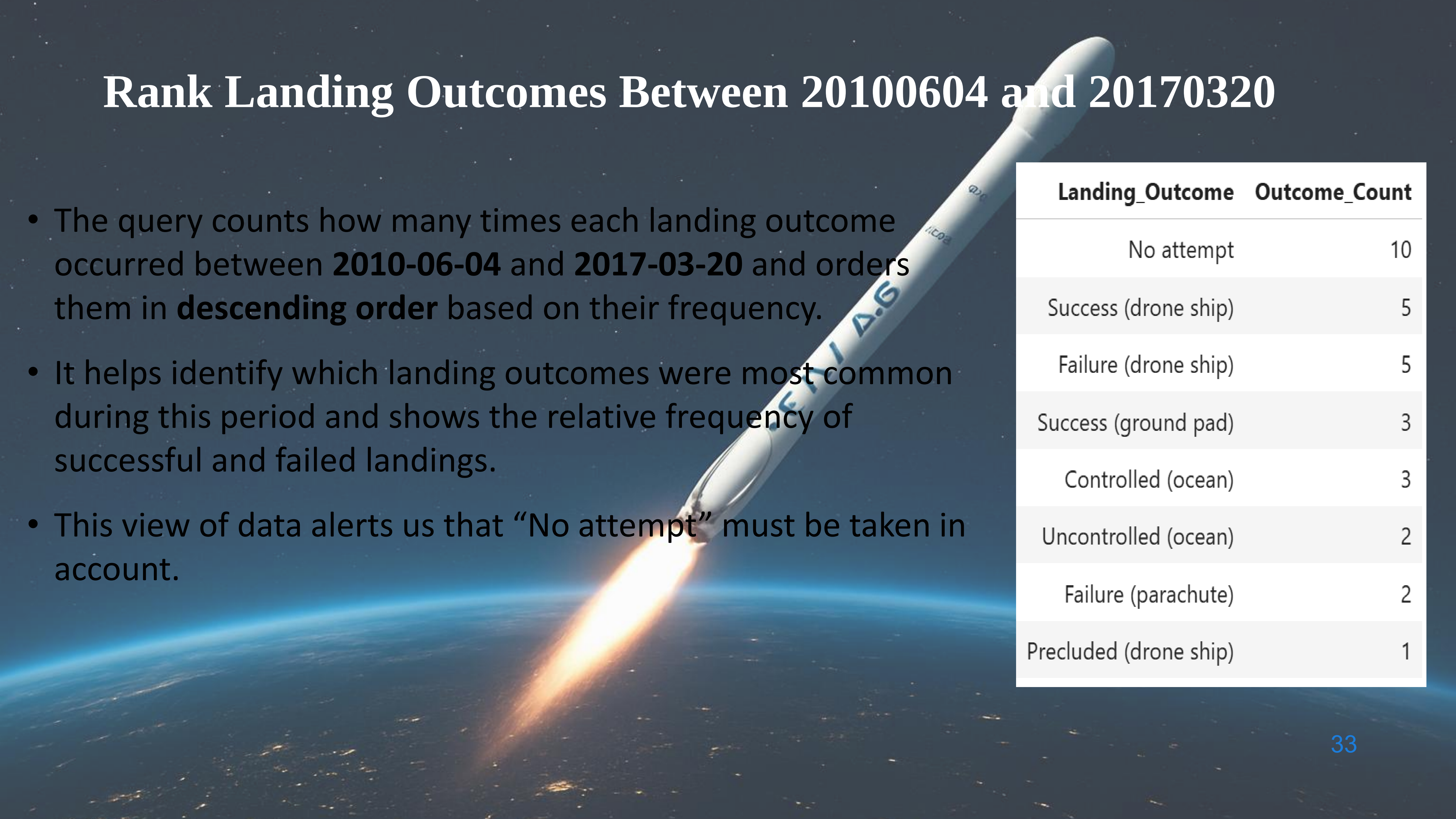
Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

2015 Launch Records

- The query retrieves all launches in 2015 where the landing outcome was "Failure (drone ship)", showing the month, booster version, and launch site.
- Result:
  - Two failed landings occurred in 2015 — one in January (01) and another in April (04).
  - Both used the F9 v1.1 booster and took place at the CCAFS LC-40 launch site.

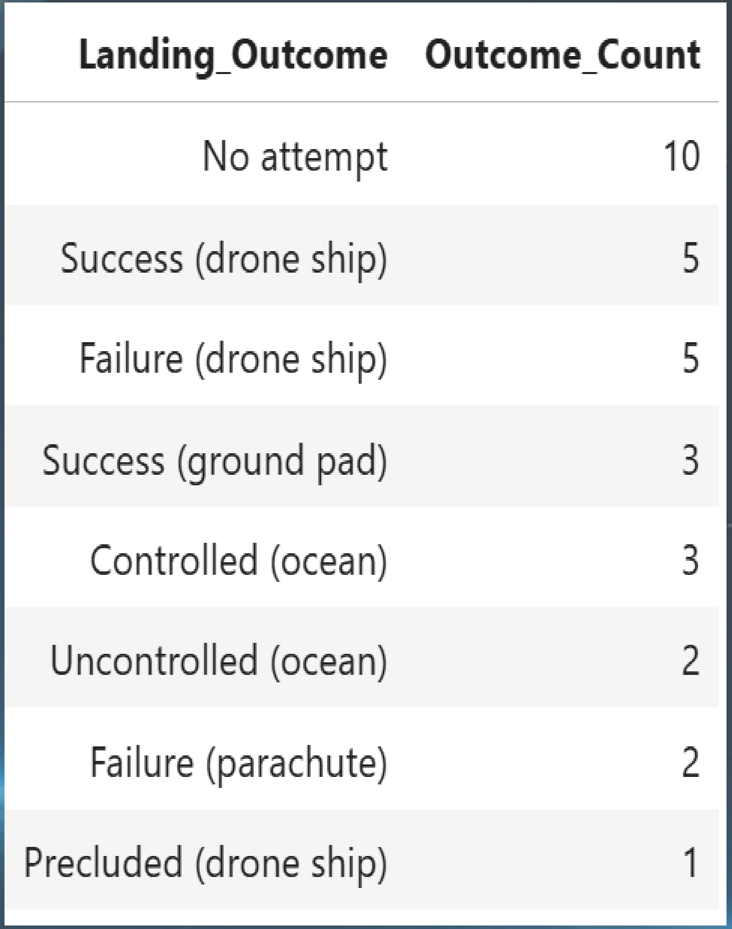


Month	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40



Rank Landing Outcomes Between 20100604 and 20170320

- The query counts how many times each landing outcome occurred between **2010-06-04** and **2017-03-20** and orders them in **descending order** based on their frequency.
- It helps identify which landing outcomes were most common during this period and shows the relative frequency of successful and failed landings.
- This view of data alerts us that “No attempt” must be taken in account.

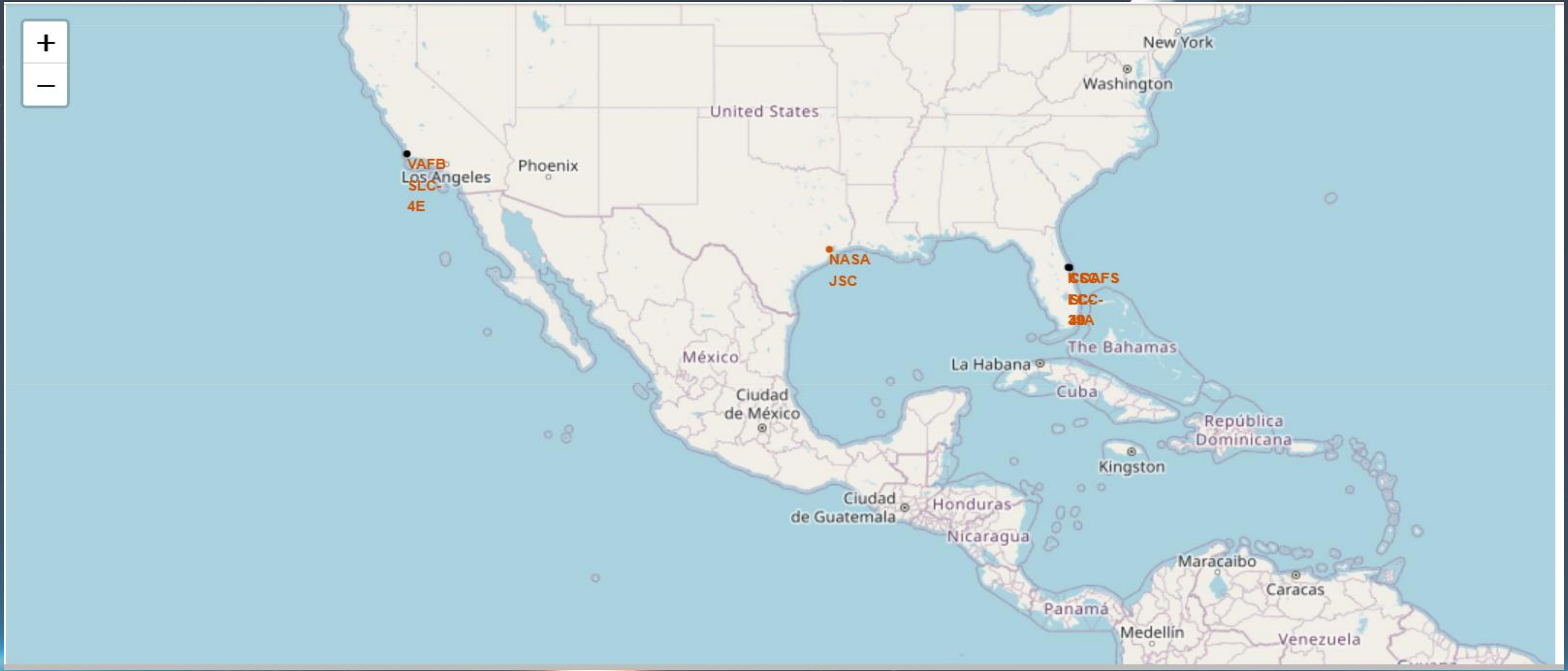


Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

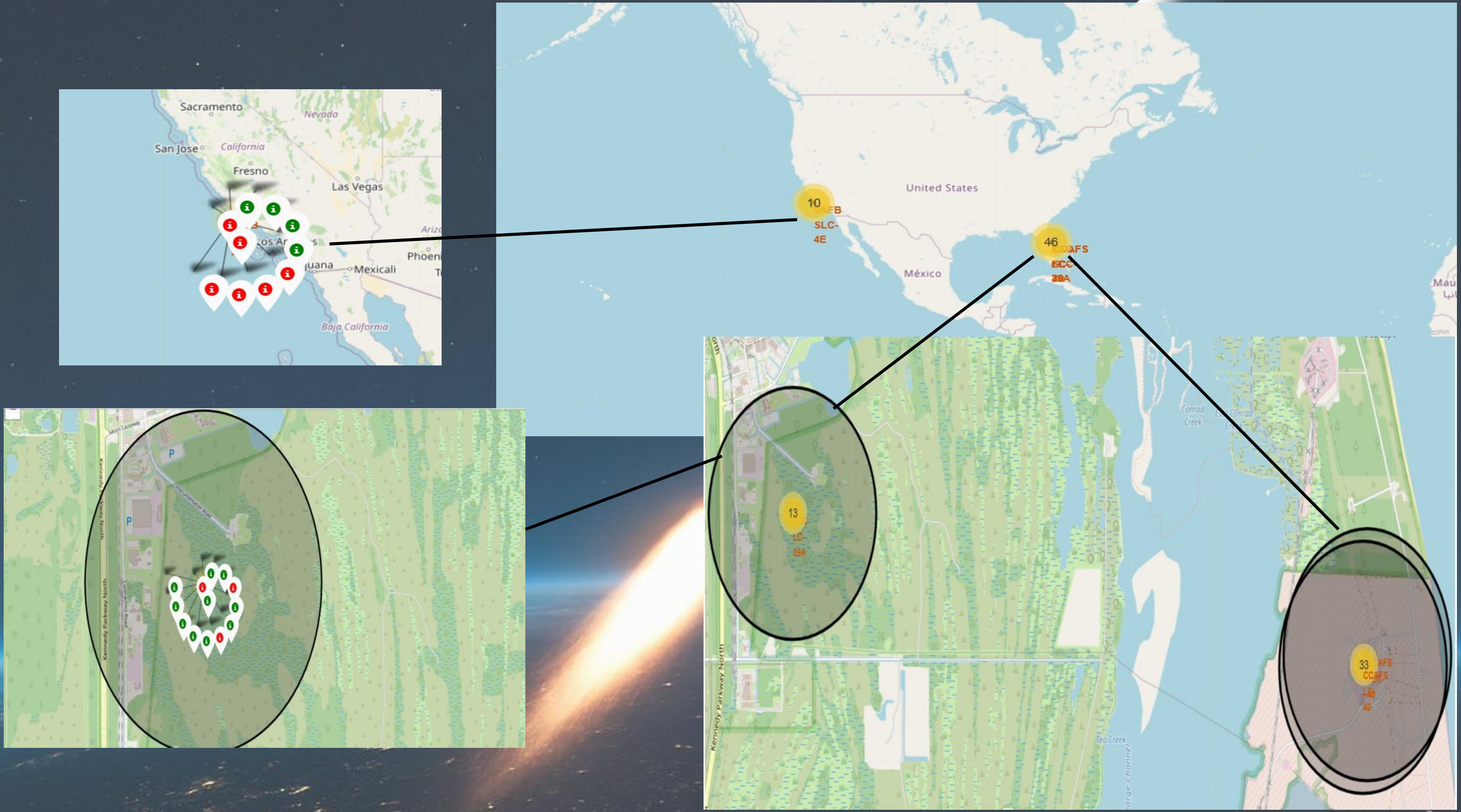
Launch Sites Proximities Analysis

All launch sites



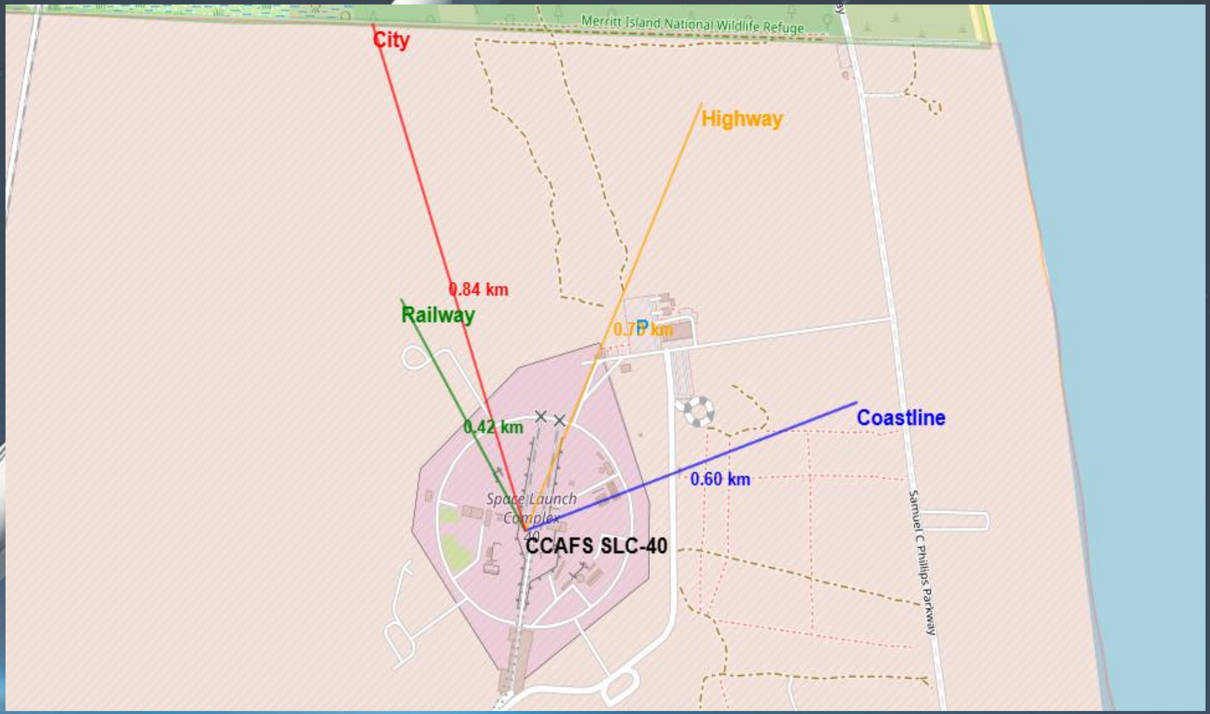
Launch sites are near sea, probably by safety, but not too far from roads and railroads

Launch Outcomes by Site



Proximity Analysis of CCAFS SLC-40 Launch Site and Nearby Features

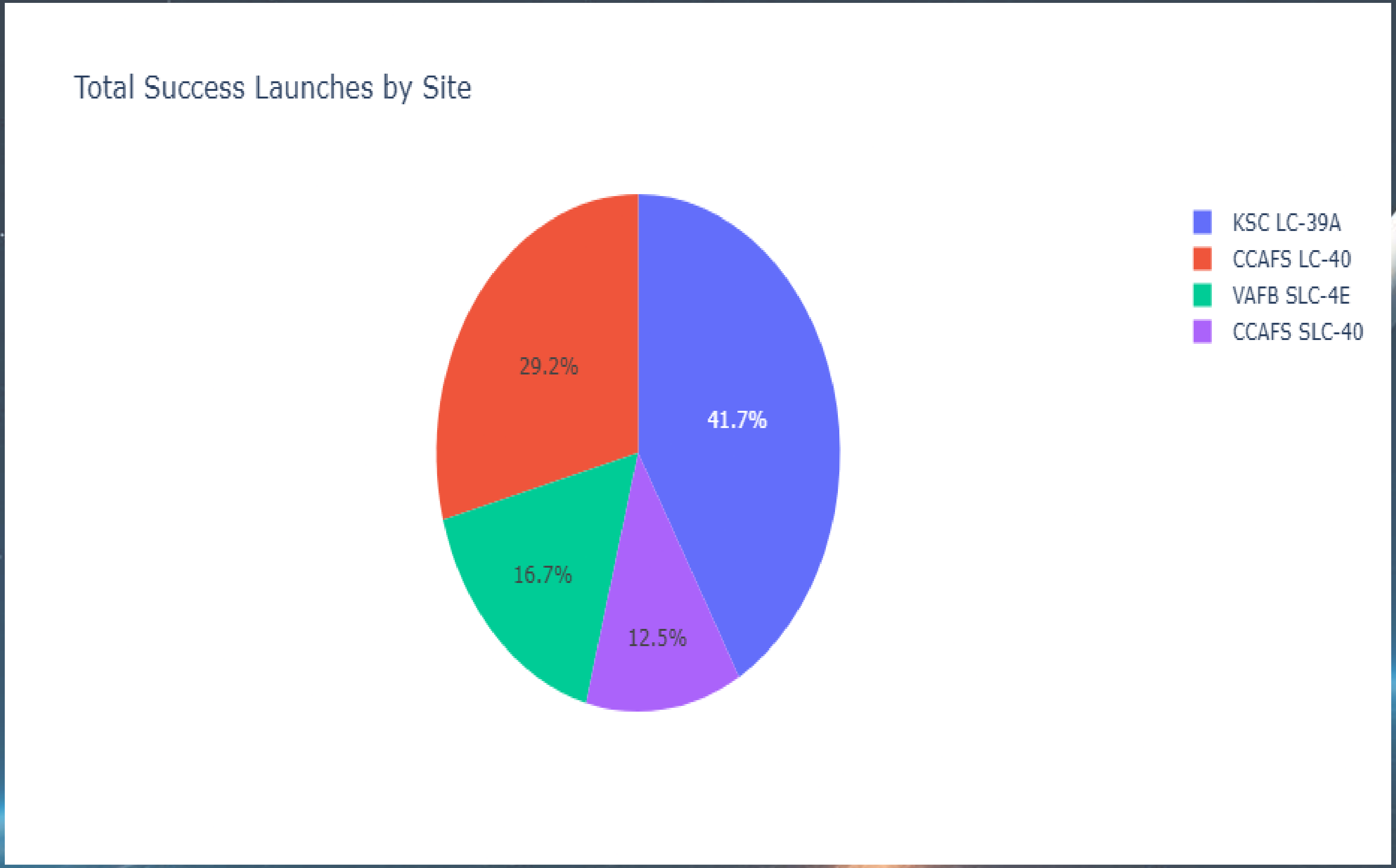
- The map shows CCAFS SLC-40 launch site and its proximity to key features like coastline, railway, highway, and city.
- Lines connect the launch site to each feature, showing their spatial relationship.
- Labels display feature names and distances directly on the map.
- This visualization helps analyze accessibility, logistics, and safety around the launch site



Section 4

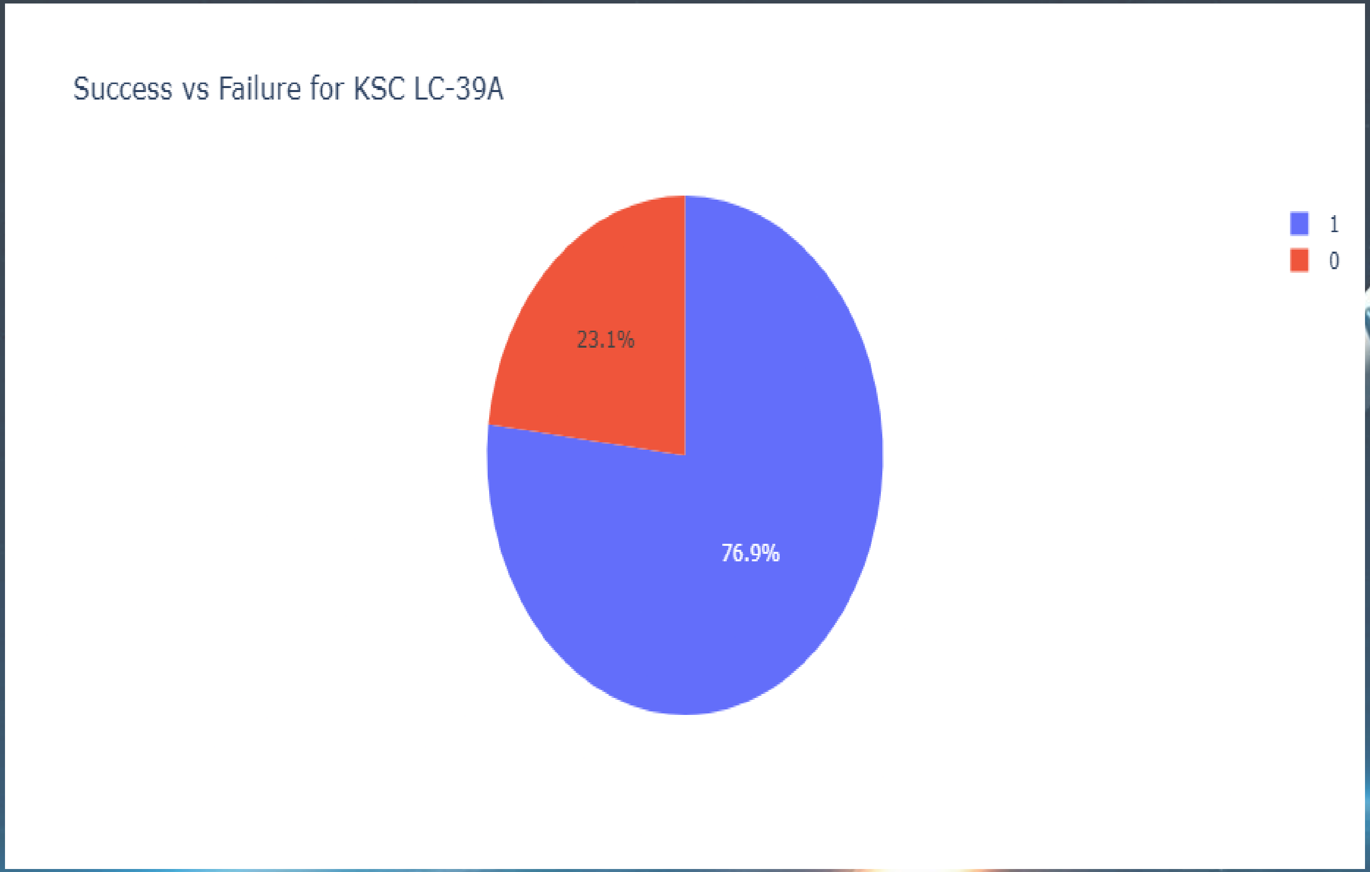
Build a Dashboard with Plotly Dash

Successful Launches by Site



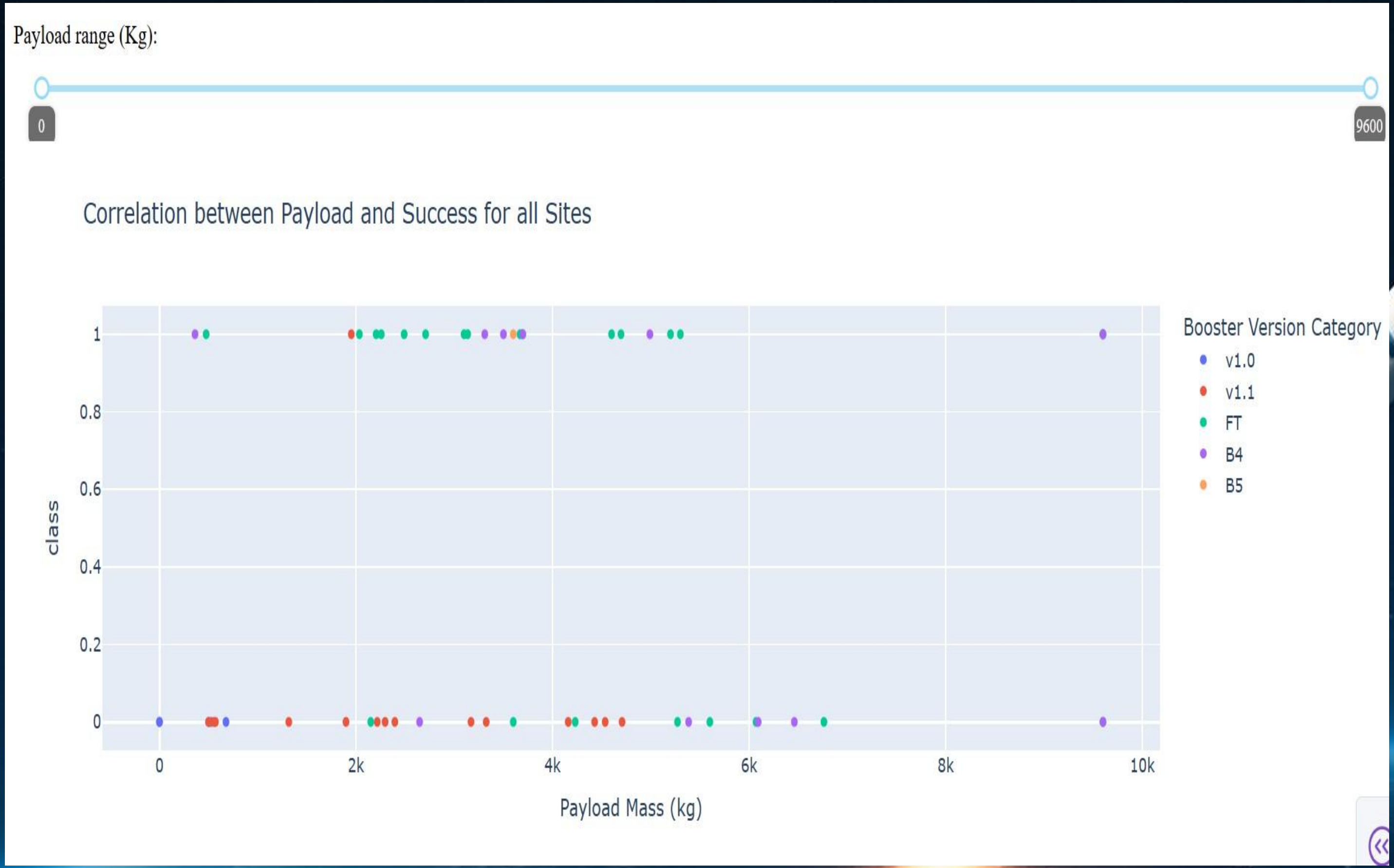
- Pie chart shows the total successful launches across all SpaceX launch sites, with slice size representing success count per site. The largest slice indicates the site with the highest number of successful launches, highlighting operational efficiency

Lauch site with highest launches



- The Launch site KSC LC-39A is having the highest launch success ratio with success rate of 76.9% .
- The pie chart shows **success (class = 1) vs failure (class = 0)** for that launch site, with slice size representing counts.
- A dominant success slice highlights the site's **high launch reliability and operational efficiency**.

Payload vs Launch Outcome across all sites



•Each point represents a launch, plotted by **Payload Mass (kg)** on the x-axis vs. **Launch Outcome (0 = failure, 1 = success)** on the y-axis.
•Points are **colored by Booster Version Category**, showing which booster types are associated with higher success rates.
•Observations may include: **specific payload ranges** or **booster versions** achieving the **highest success rates**, while extreme payloads may show occasional failures.

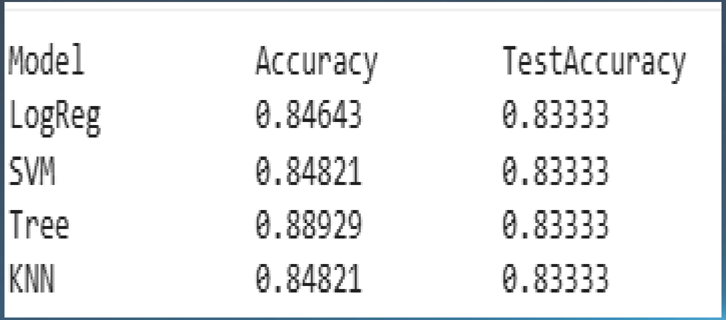
Section 6

Predictive Analysis- Classification

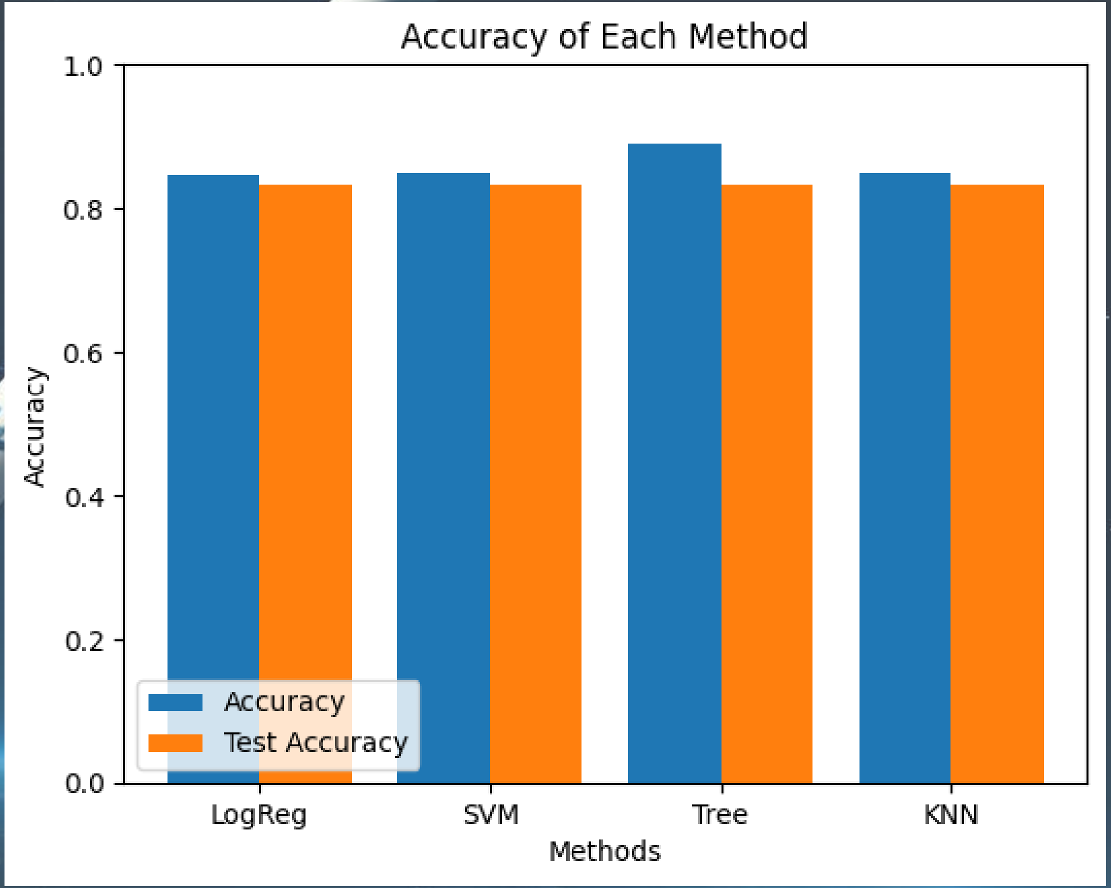
Classification Accuracy

Among the four models, **Decision Tree** achieved the **highest training accuracy (86.25%)**

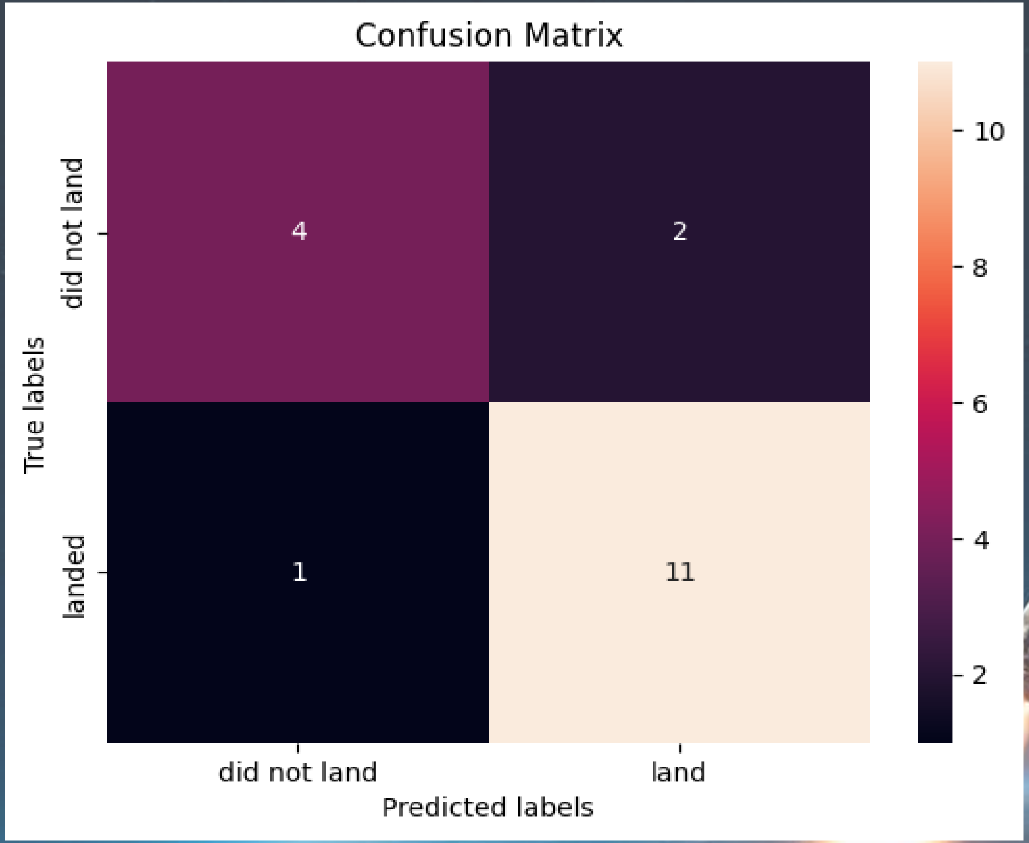
All models have the **same test accuracy (83.33%)**, indicating similar performance on unseen data.



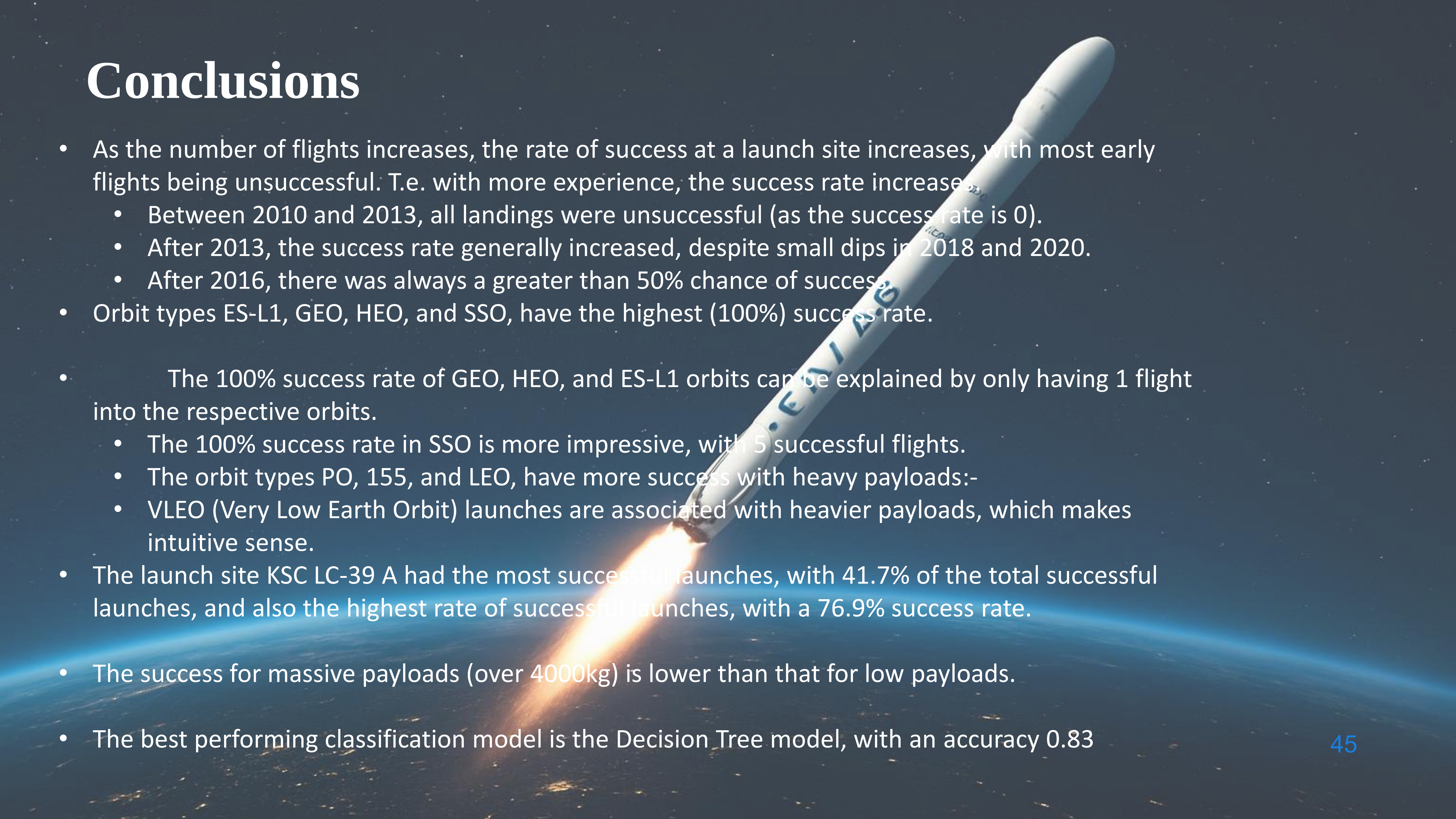
```
Model       Accuracy    TestAccuracy
LogReg      0.84643     0.83333
SVM         0.84821     0.83333
Tree        0.88929     0.83333
KNN         0.84821     0.83333
```



Confusion Matrix



Confusion matrix of Decision Tree Classifier proves its accuracy by showing the big numbers of true positive and true negative compared to the false ones.



Conclusions

- As the number of flights increases, the rate of success at a launch site increases, with most early flights being unsuccessful. T.e. with more experience, the success rate increases.
  - Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
  - After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
  - After 2016, there was always a greater than 50% chance of success.
- Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.

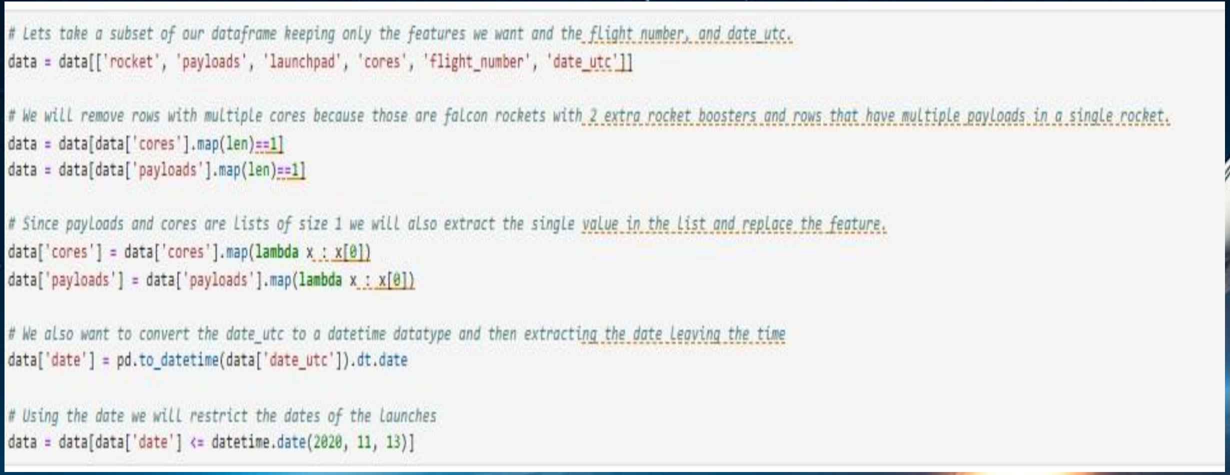
- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
  - The 100% success rate in SSO is more impressive, with 5 successful flights.
  - The orbit types PO, 155, and LEO, have more success with heavy payloads:-
  - VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.
- The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches, and also the highest rate of successful launches, with a 76.9% success rate.

- The success for massive payloads (over 4000kg) is lower than that for low payloads.

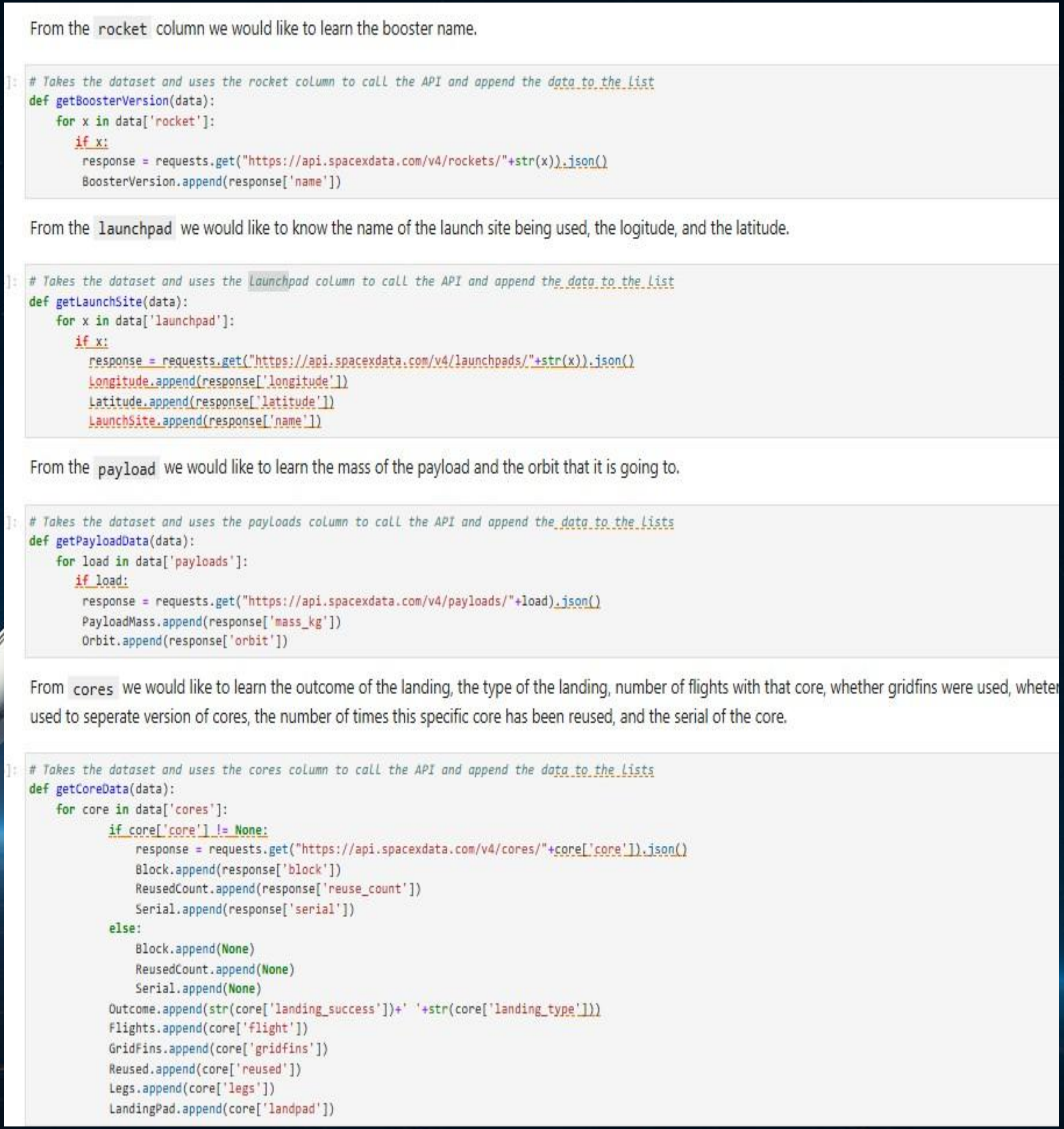
- The best performing classification model is the Decision Tree model, with an accuracy 0.83

Data Collection –Space X Rest API

- Custom functions to retrieve the required information
- Custome logic to clean the data



```
# Lets take a subset of our dataframe keeping only the features we want and the flight number, and date_utc.
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 boosters and rows that have multiple payloads in a single rocket.
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 list and replace the feature.
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 leaving the time
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)]
```



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

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

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

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

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

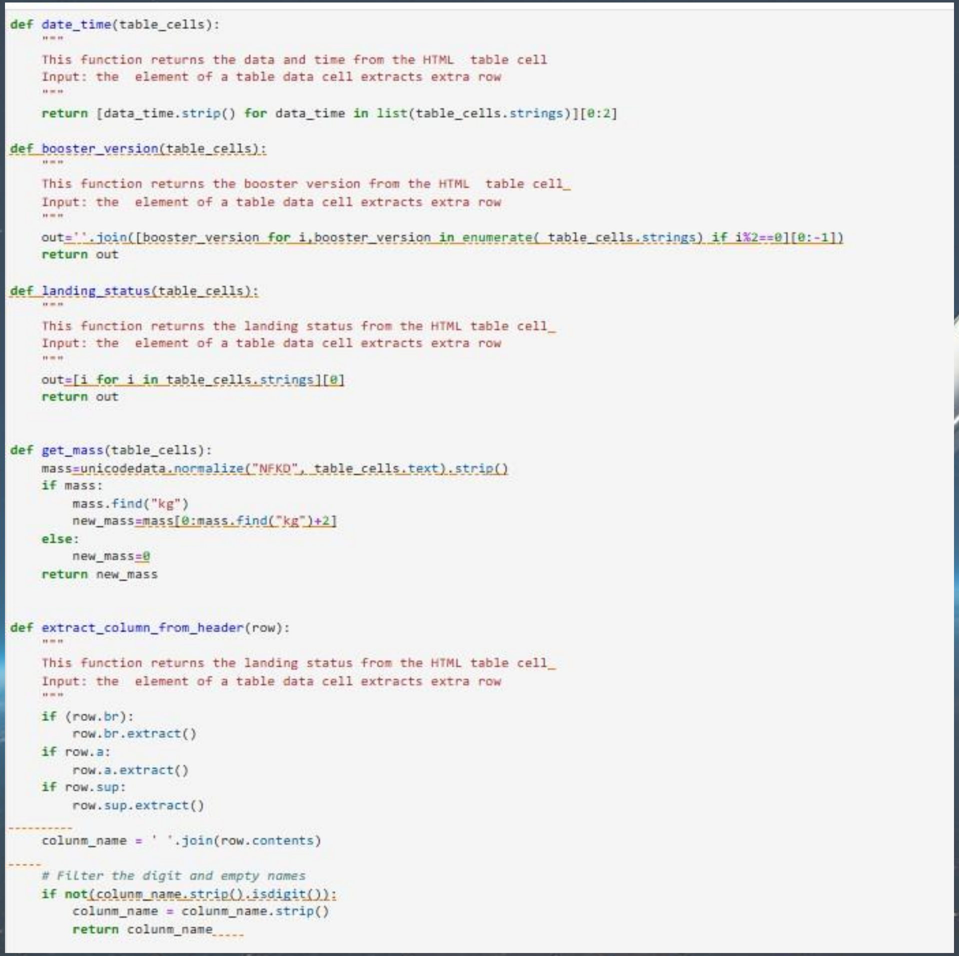
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])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

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

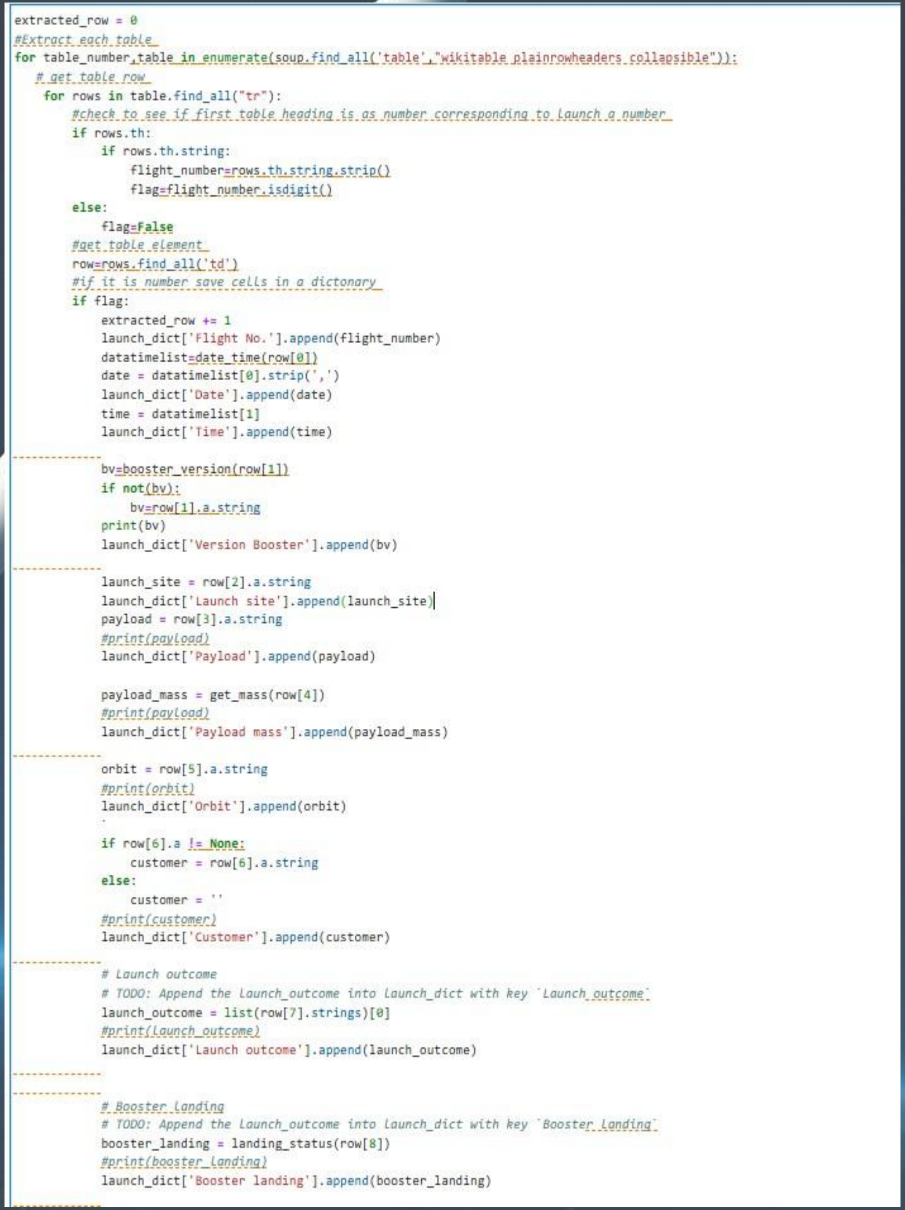
```
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                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'])
```

Data Collection –Web Scraping

- Custom functions for web scraping
- Custome logic to fill up the launch_dict values from the lauch tables.



```python
def date_time(table_cells):
    """
    This function returns the data and time from the HTML  table cell
    Input: the  element of a table data cell extracts extra row
    """
    return [data_time.strip() for data_time in list(table_cells.strings)][0:2]

def booster_version(table_cells):
    """
    This function returns the booster version from the HTML  table cell 
    Input: the  element of a table data cell extracts extra row
    """
    out=''.join([booster_version for i,booster_version in enumerate( table_cells.strings) if i%2==0][0:-1])
    return out

def landing_status(table_cells):
    """
    This function returns the landing status from the HTML table cell 
    Input: the  element of a table data cell extracts extra row
    """
    out=[i for i in table_cells.strings][0]
    return out


def get_mass(table_cells):
    mass=unicodedata.normalize("NFKD", table_cells.text).strip()
    if mass:
        mass.find("kg")
        new_mass=mass[0:mass.find("kg")+2]
    else:
        new_mass=0
    return new_mass


def extract_column_from_header(row):
    """
    This function returns the landing status from the HTML table cell 
    Input: the  element of a table data cell extracts extra row
    """
    if (row.br):
        row.br.extract()
    if row.a:
        row.a.extract()
    if row.sup:
        row.sup.extract()

    colunm_name = ' '.join(row.contents)

    # Filter the digit and empty names
    if not(colunm_name.strip().isdigit()):
        colunm_name = colunm_name.strip()
        return colunm_name
```



```python
extracted_row = 0
#Extract each table 
for table_number,table in enumerate(soup.find_all('table',"wikitable plainrowheaders collapsible")):
   # get table row 
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding to launch a number 
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element 
        row=rows.find_all('td')
        #if it is number save cells in a dictonary 
        if flag:
            extracted_row += 1
            launch_dict['Flight No.'].append(flight_number)
            datatimelist=date_time(row[0])
            date = datatimelist[0].strip(',')
            launch_dict['Date'].append(date)
            time = datatimelist[1]
            launch_dict['Time'].append(time)

            bv=booster_version(row[1])
            if not(bv):
                bv=row[1].a.string
            print(bv)
            launch_dict['Version Booster'].append(bv)

            launch_site = row[2].a.string
            launch_dict['Launch site'].append(launch_site)
            payload = row[3].a.string
            #print(payload)
            launch_dict['Payload'].append(payload)

            payload_mass = get_mass(row[4])
            #print(payload)
            launch_dict['Payload mass'].append(payload_mass)

            orbit = row[5].a.string
            #print(orbit)
            launch_dict['Orbit'].append(orbit)

            if row[6].a != None:
                customer = row[6].a.string
            else:
                customer = ''
            #print(customer)
            launch_dict['Customer'].append(customer)

            # Launch outcome
            # TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
            launch_outcome = list(row[7].strings)[0]
            #print(launch_outcome)
            launch_dict['Launch outcome'].append(launch_outcome)


            # Booster landing
            # TODO: Append the launch_outcome into launch_dict with key `Booster landing`
            booster_landing = landing_status(row[8])
            #print(booster_landing)
            launch_dict['Booster landing'].append(booster_landing)
```