IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Manuel Podestà
12/09/2025

## Outline

## Executive Summary

- Summary of methodologies
  a) Data collection using API
  b) Data collection by Web Scraping
  c) Data Wrangling
  d) Exploratory Data Analysis (EDA) with SQL queries
  e) EDA with Data Visualization
  f) Interactive visual analytics with folium
  g) Machine Learning (ML) prediction
- Summary of all results
  a) EDA results
  b) Interactive analytics images
  c) Predictive analytics result from ML algorithms

## Introduction

SpaceX has upended the space industry by offering Falcon 9 launches for as little as $62 million, whereas competing providers charge upwards of $165 million per flight. This dramatic cost reduction arises from SpaceX's pioneering strategy of recovering the first stage—guiding it back to Earth and reusing it on subsequent missions—which drives prices down even further with each reuse. As a data scientist at a startup aiming to rival SpaceX, my project focuses on building a machine learning pipeline that predicts the future landing outcomes of the first stage. That prediction capability is crucial for determining the optimal bid price when competing against SpaceX for launch contracts.

The primary target of this projects is to find out the factors that influence landing outcome. To solve this problem we need to:

1. Find the relationship between each variables and how is linked to the landing outcome
2. Deduce the best condition to have a better probability of successful landings

Section 1

## Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - Data has been collected from the "SpaceX" data website calling the API  that returned a JSON file which has later  been normalized.
- Perform data wrangling
  - Using user defined function, the dataset has been cleaned and only relevant information was left in the dataset
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Using scikit-learn Python library, various ML algorithms has been built, tuned with different parameters and evaluated by splitting the dataset in Train and Test data

## Data Collection

Data acquisition refers to the act of collecting and quantifying data on specific variables within a predefined framework, which subsequently allows one to respond to pertinent inquiries and assess results. As previously noted, the dataset was obtained via REST API and Web Scraping from Wikipedia.

Regarding the REST API, the process began with a GET request. The response payload was then decoded as JSON and transformed into a pandas dataframe using the `json_normalize()` function. Afterward, the dataset was refined, missing entries were identified, and necessary values were filled in.

For the web scraping portion, BeautifulSoup was employed to retrieve the launch data formatted as an HTML table, which was then parsed and converted into a pandas dataframe for subsequent examination.

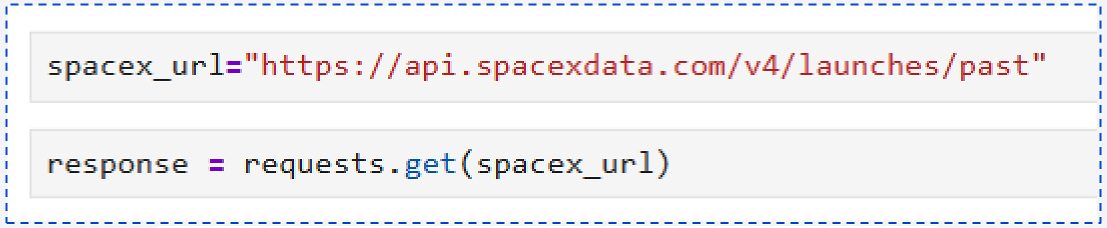
# Data Collection – SpaceX API

Get request for the data about rockets launches using API

Convert into a DatFrame the JSON response using json_normalize method

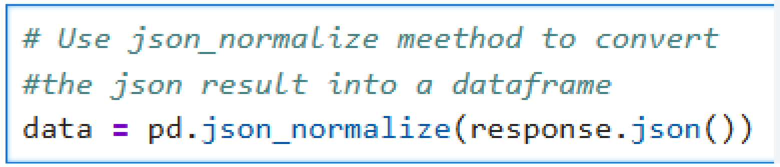
Manage missing values and keep only relevant data perdorming a Data Cleaning procedure

Source: [https://github.com/m97Pod/ds-capstone.git](https://github.com/m97Pod/ds-capstone.git)

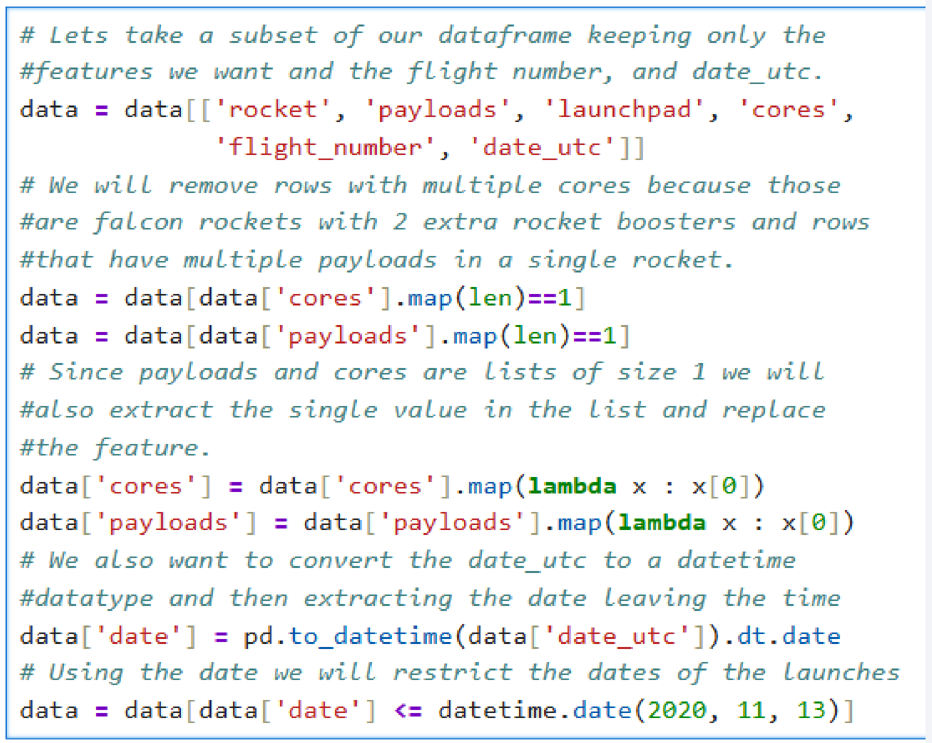


```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

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



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



```
# 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)]
```

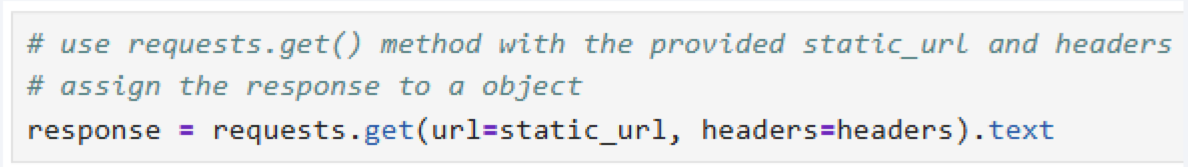
## Data Collection - Scraping

From the Wikipedia URL we can request the Falcon 9 launch data

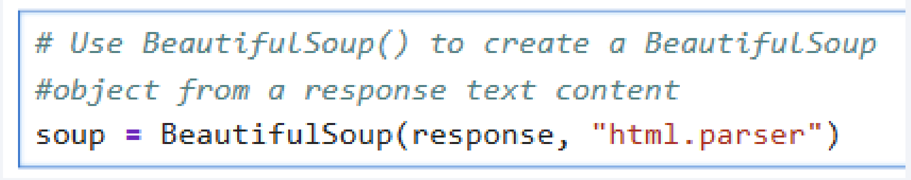
Now we can make a BeautifoulSoup object from the response in HTML format

From the Soup object we can extract all the table columns and variables names. These information are located in the HTML header

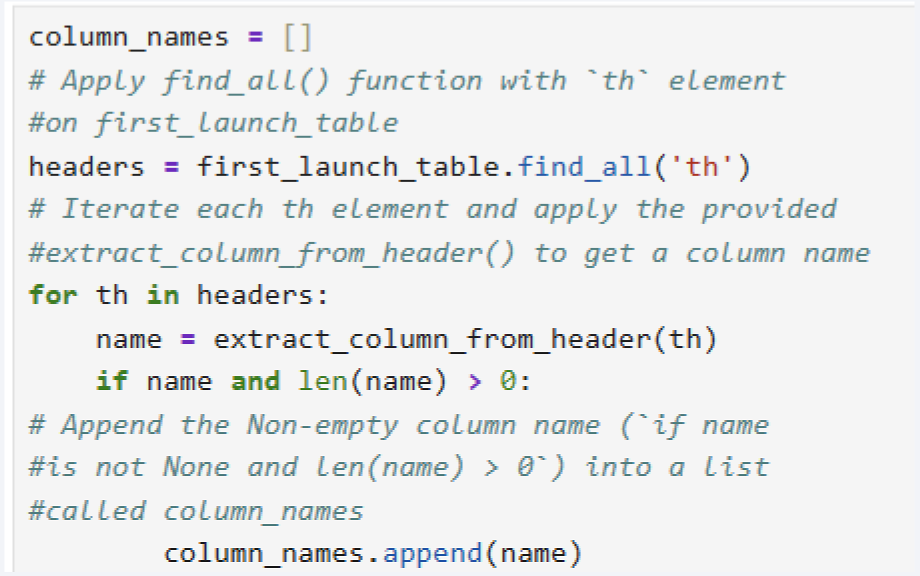
Complete code at: [https://github.com/m97Pod/ds-capstone.git](https://github.com/m97Pod/ds-capstone.git)



```python
# use requests.get() method with the provided static_url and headers
# assign the response to a object
response = requests.get(url=static_url, headers=headers).text
```



```python
# Use BeautifulSoup() to create a BeautifulSoup
#object from a response text content
soup = BeautifulSoup(response, "html.parser")
```



```python
column_names = []
# Apply find_all() function with `th` element
#on first_launch_table
headers = first_launch_table.find_all('th')
# Iterate each th element and apply the provided
#extract_column_from_header() to get a column name
for th in headers:
    name = extract_column_from_header(th)
    if name and len(name) > 0:
# Append the Non-empty column name (`if name
#is not None and len(name) > 0`) into a list
#called column_names
        column_names.append(name)
```

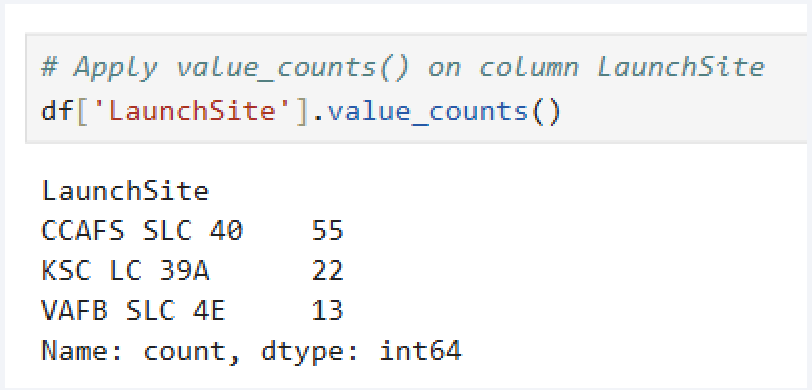
## Data Wrangling

Data Wrangling involves the transformation of disorganized and intricate datasets into a clean, unified format to facilitate easy access and effective Exploratory Data Analysis (EDA).

1. Our initial step is to determine the total number of launches conducted at each site.
2. Secondly, we calculate the number and occurrence of each orbit
3. Following that, we analyze the frequency and distribution of mission outcomes across different orbit types.
4. Next, we derive a landing outcome label based on the existing outcome column, which simplifies subsequent analysis, visualization, and machine learning tasks.
5. Finally, we save the processed results into a CSV file.

## Data Wrangling

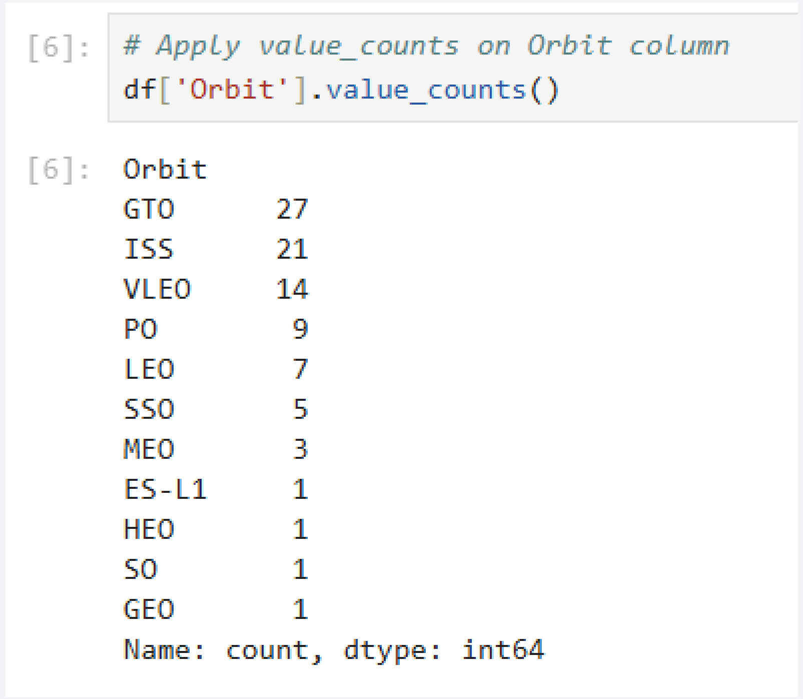
- Our initial step is to determine the total number of launches conducted at each site.



```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()

LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

Complete code at: [https://github.com/m97Pod/ds-capstone.git](https://github.com/m97Pod/ds-capstone.git)

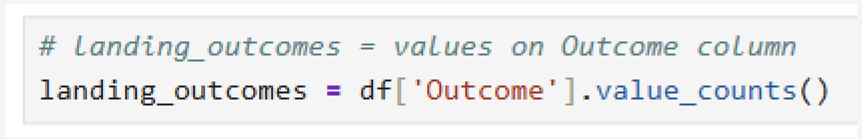
- Secondly, we calculate the number and occurrence of each orbit



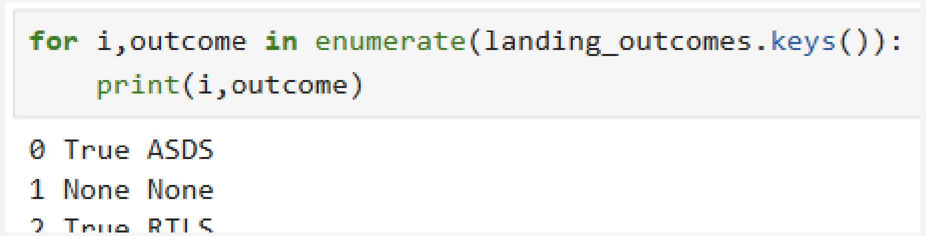
```
[6]: # Apply value_counts on Orbit column
     df['Orbit'].value_counts()

[6]: Orbit
     GTO      27
     ISS      21
     VLEO     14
     PO        9
     LEO       7
     SSO       5
     MEO       3
     ES-L1     1
     HEO       1
     SO        1
     GEO       1
     Name: count, dtype: int64
```

# Data Wrangling

- Following that, we analyze the frequency and distribution of mission outcomes across different orbit types.

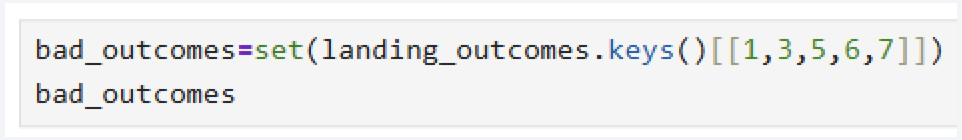


```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
```



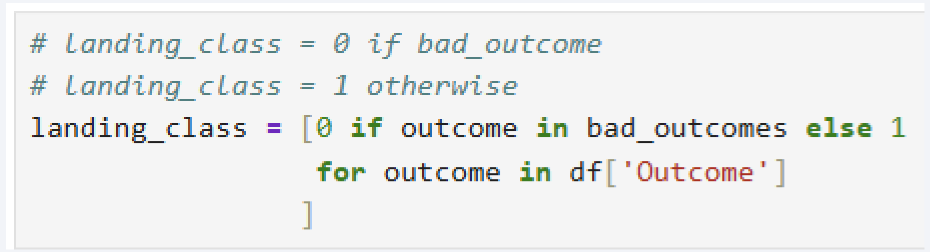
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
```



```
bad_outcomes=set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes
```

- Next, we derive a landing outcome label based on the existing outcome column, which simplifies subsequent analysis, visualization, and machine learning tasks.



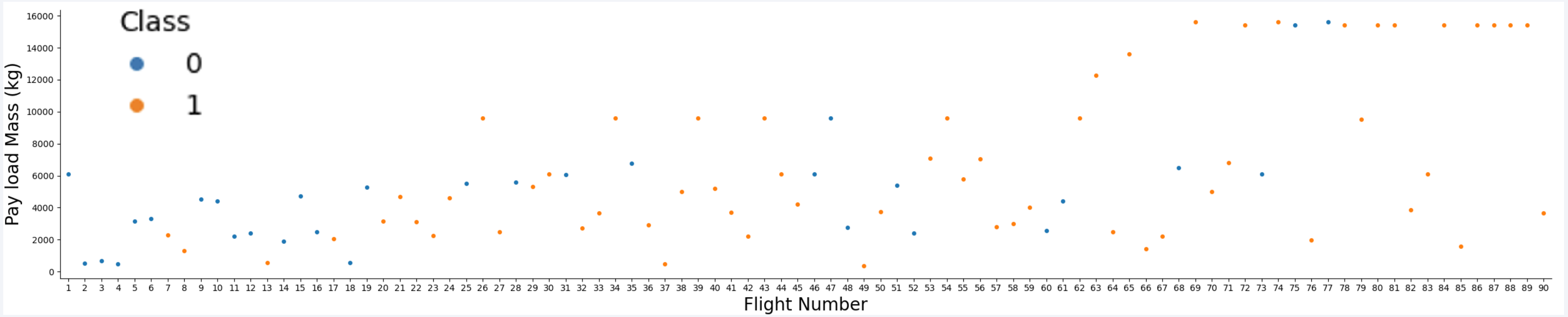
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if outcome in bad_outcomes else 1
                 for outcome in df['Outcome']
                ]
```

And appended the brand new class to the dataframe.

Complete code at: https://github.com/m97Pod/ds-capstone.git

# EDA with Data Visualization

Firstly, we plotted some scatter plot to better visualize some variables dependence. For instance, we plotted the Payload Mass VS Flight number. This graph can tell us how payload mass can influence the outcome of a launch.



*In the graph, class 0 is for a negative outcome and class 1 is for a positive outcome*

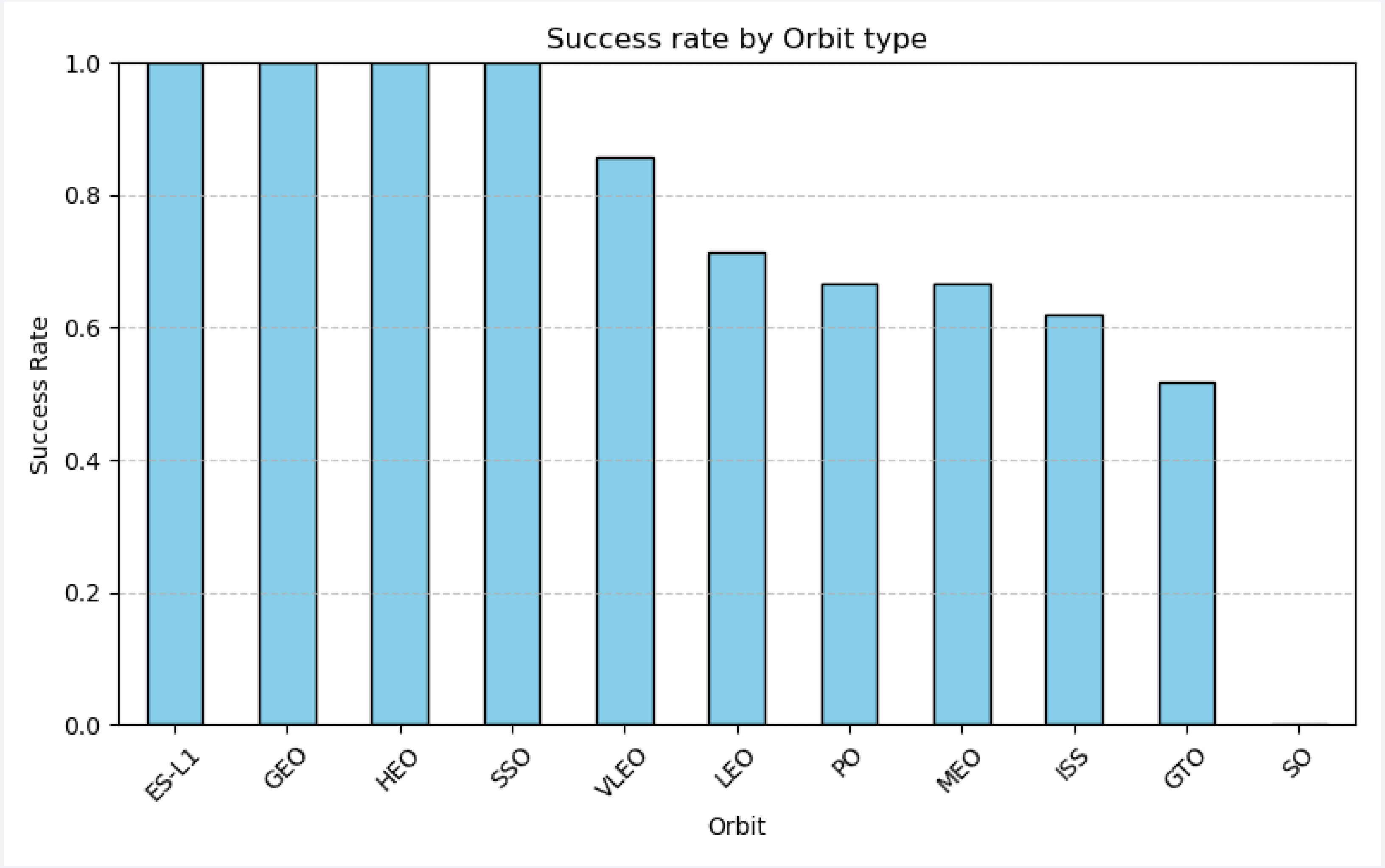
We also made similar plot for Flight Number and Launch Site, Payload Mass and Launch Site.

Complete code at: [https://github.com/m97Pod/ds-capstone.git](https://github.com/m97Pod/ds-capstone.git)

## EDA with Data Visualization

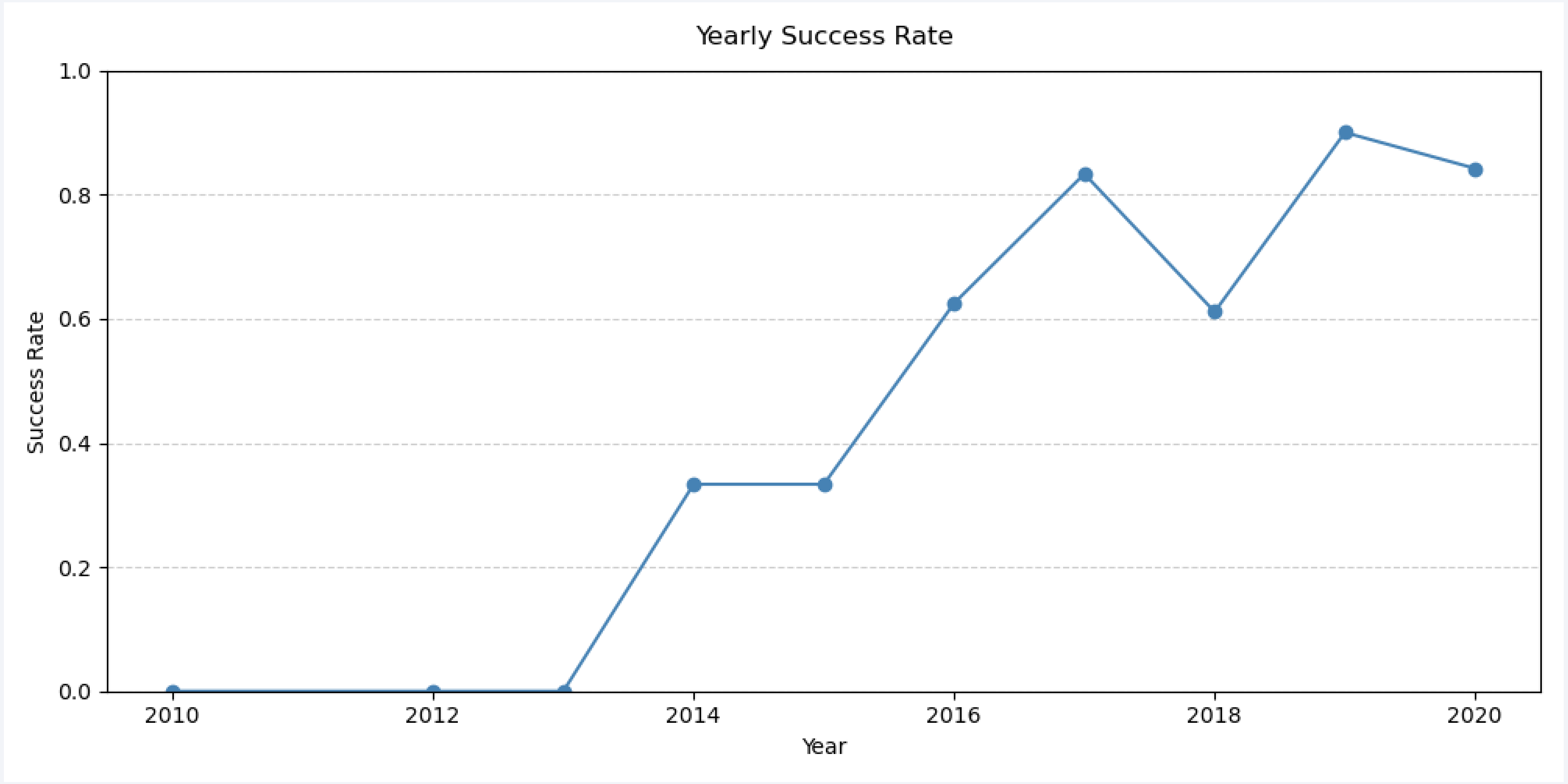
At a later time, we made a barchart to have a percentage success rate for launches by orbit type.

Complete code at: https://github.com/m97Pod/ds-capstone.git



## EDA with Data Visualization

FInally, to have time reference, we plotted the yearly succces rate by year.



Complete code at: https://github.com/m97Pod/ds-capstone.git

# EDA with SQL

Here we have the SQL queries used to explore the data in the provided database:

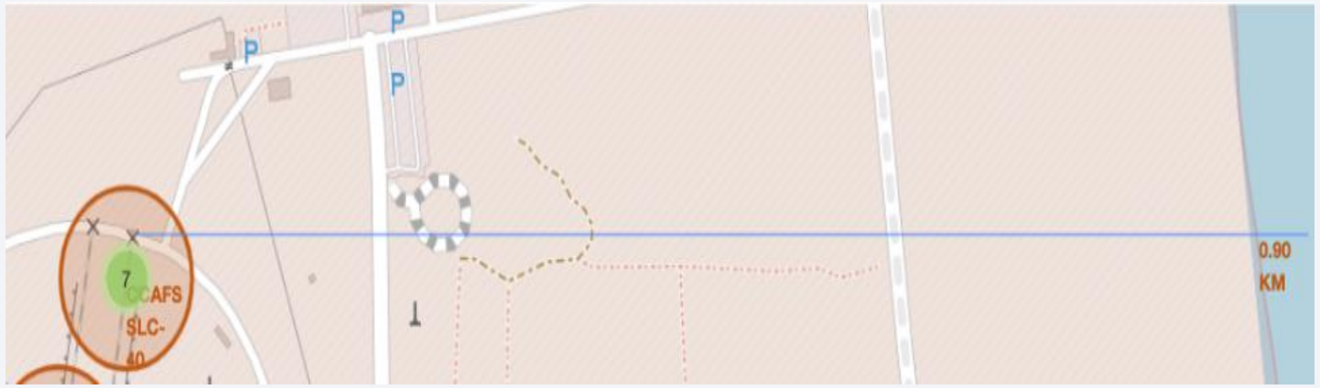
Complete code at: https://github.com/m97Pod/ds-capstone.git

- SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE
- SELECT * FROM SPACEXTABLE WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
- SELECT SUM(PAYLOAD_MASS__KG_) AS total_Nasa_payload_mass FROM SPACEXTABLE WHERE CUSTOMER LIKE 'NASA (CRS)';
- SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_BOOSTER_F9_v1_1 FROM SPACEXTABLE WHERE BOOSTER_VERSION LIKE "%F9 v1.1%"
- SELECT MIN(DATE) AS first_success_ground_pad FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (ground pad)';
- SELECT DISTINCT BOOSTER_VERSION AS booster_name FROM SPACEXTABLE WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
- SELECT Landing_Outcome AS outcome, COUNT(*) AS total_count FROM SPACEXTABLE GROUP BY Landing_Outcome;
- SELECT DISTINCT BOOSTER_VERSION AS booster_name FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE );
- SELECT substr(date, 6, 2) AS month_number, Landing_Outcome AS failure_landing_outcome, BOOSTER_VERSION AS booster_name, LAUNCH_SITE AS launch_site FROM SPACEXTABLE WHERE substr(date, 1, 4) = '2015' AND Landing_Outcome = 'Failure (drone ship)';
- SELECT Landing_Outcome AS outcome, COUNT(*) AS total_count FROM SPACEXTABLE WHERE date BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY Landing_Outcome ORDER BY total_count DESC;

## Build an Interactive Map with Folium

The base map is centered on the NASA Johnson Space Center at Huston, Texas; thislocation is marked with a folium.Marker and a folium.Circle. For better visualization ,weadded the following circles, markers and lines.

- CCAFS LC-40 launch site and its distance from the linecoast
- CCAFS SLC-40 launch site and its distance from the linecoast
- KSC LC-39A launch site and its distance from the linecoast
- VAFB SLC-4E launch site and its distance from the linecoast



Complete code at: https://github.com/m97Pod/ds-capstone.git

## Build a Dashboard with Plotly Dash

We have a drop down menu to choose one specific site or, alternatively, all the sites.
For each selection, there is a pie chart and a scatter plot:

- *Pie chart*: shows the percentage of succesful launches when visualizing all the sites. Alternatively, it shows the pie is splitted in positive and negative outcome.
- *Scatter plot*: shows the outcome by payload mass. There is also a slider to choose a payload mass interval.

This two plots give us a general view about the launches outcomes in relation with which launch site is used and what payload mass is involved. The payload mass slider is useful to see which launch site is used for certain payload masses and it underlines the launch site success outcome rate.
In closing, the dash give us an overall vision about launches outcomes.

Complete code at: https://github.com/m97Pod/ds-capstone.git

## Predictive Analysis (Classification)

Predictive analysis procedure:

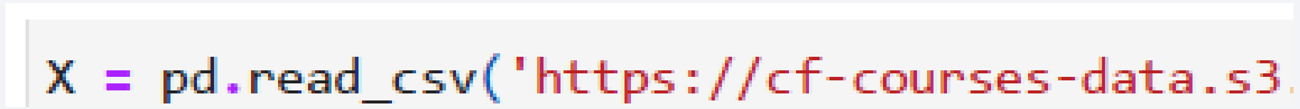
1. Set the variables
2. Split the data inn train and test subset
3. Initialize predeictive models: Logistic Regression, SVM and Decision Tree
4. Train the models
5. Test the models
6. Evaluate confusion matrix and accuracy of all models in order to choose the best performing model

Complete code at: https://github.com/m97Pod/ds-capstone.git

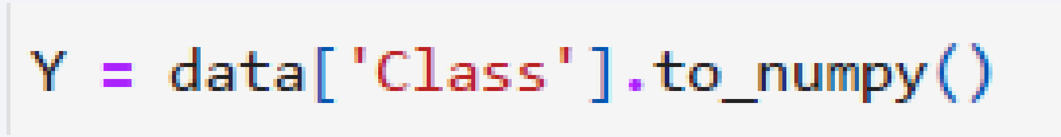
# Predictive Analysis (Classification)

Define the variables:

- Y equal to the «Class» column, which vlaues are 0 for a negative outcome and 1 for a positive outcome
- X equal to all the other classes

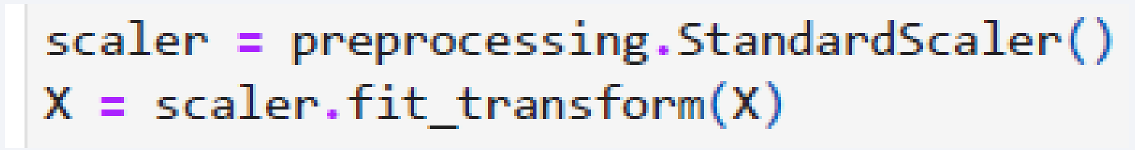


```
X = pd.read_csv('https://cf-courses-data.s3
```



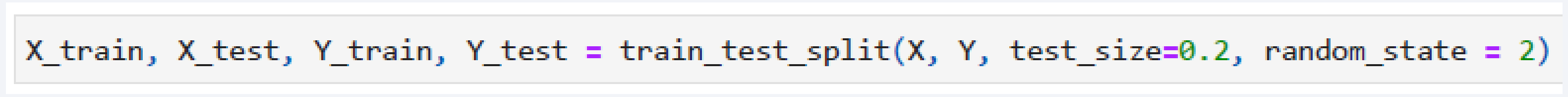
```
Y = data['Class'].to_numpy()
```

Scale the X variable



```
scaler = preprocessing.StandardScaler()
X = scaler.fit_transform(X)
```

Split the data into subset to train and test the models



```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state = 2)
```

Train and test the models → Evaluate the models using confusion matrix and accuracy indicators → Compare results and choose the best performing model

Complete code at: https://github.com/m97Pod/ds-capstone.git

## Results

The results will be categorized into three main groups:

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

# Results

**SQL EDA**

- We found a total of 4 launch sites
- The average payload mass i 2534,7 kg
- The first successful landing outcome was achieved in 2015-12-22
- We have a total of 4 booster used during the analyzed period

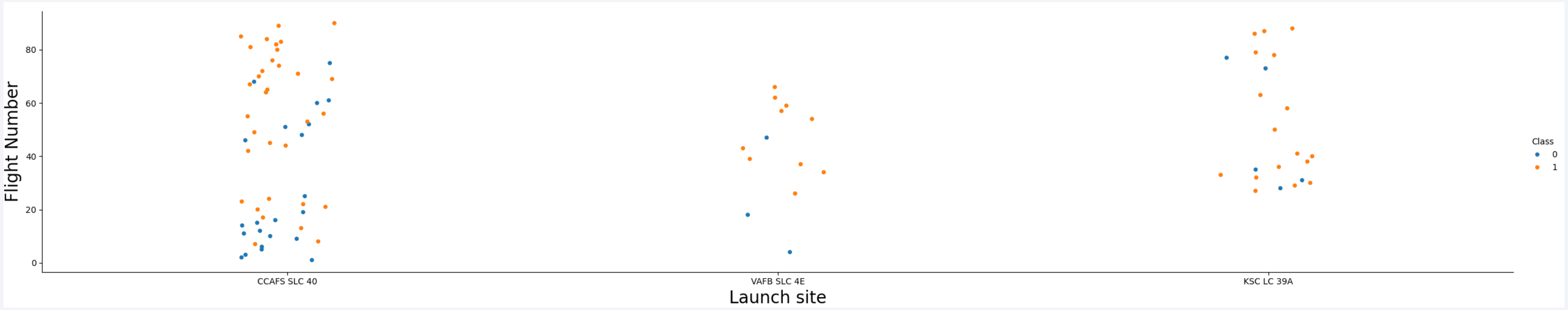
**Visualization EDA**

- We found that the launch sites are chosen relatively to the payload mass:
  - KSC LC-39A  launch site has been used for payload mass above 2000 Kg
  - CCAFS SLC-40 launch site has been used more than the others launch sites
  - VAFB SLC-4E is used lesser than the others
  - Payload mass above 12000 Kg involved only CCAFS SLC-40 and KSC LC-39A sites

Section 2

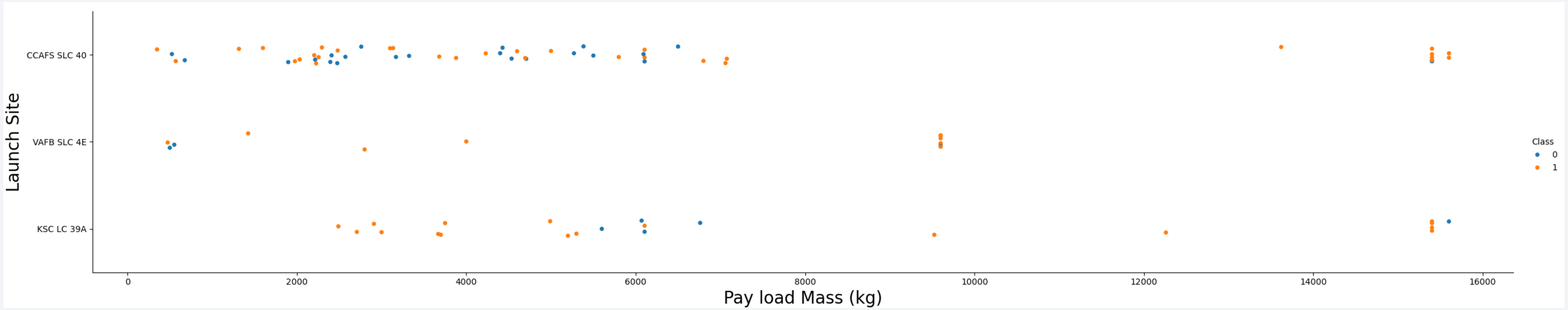
## Insights drawn from EDA

# Flight Number vs. Launch Site



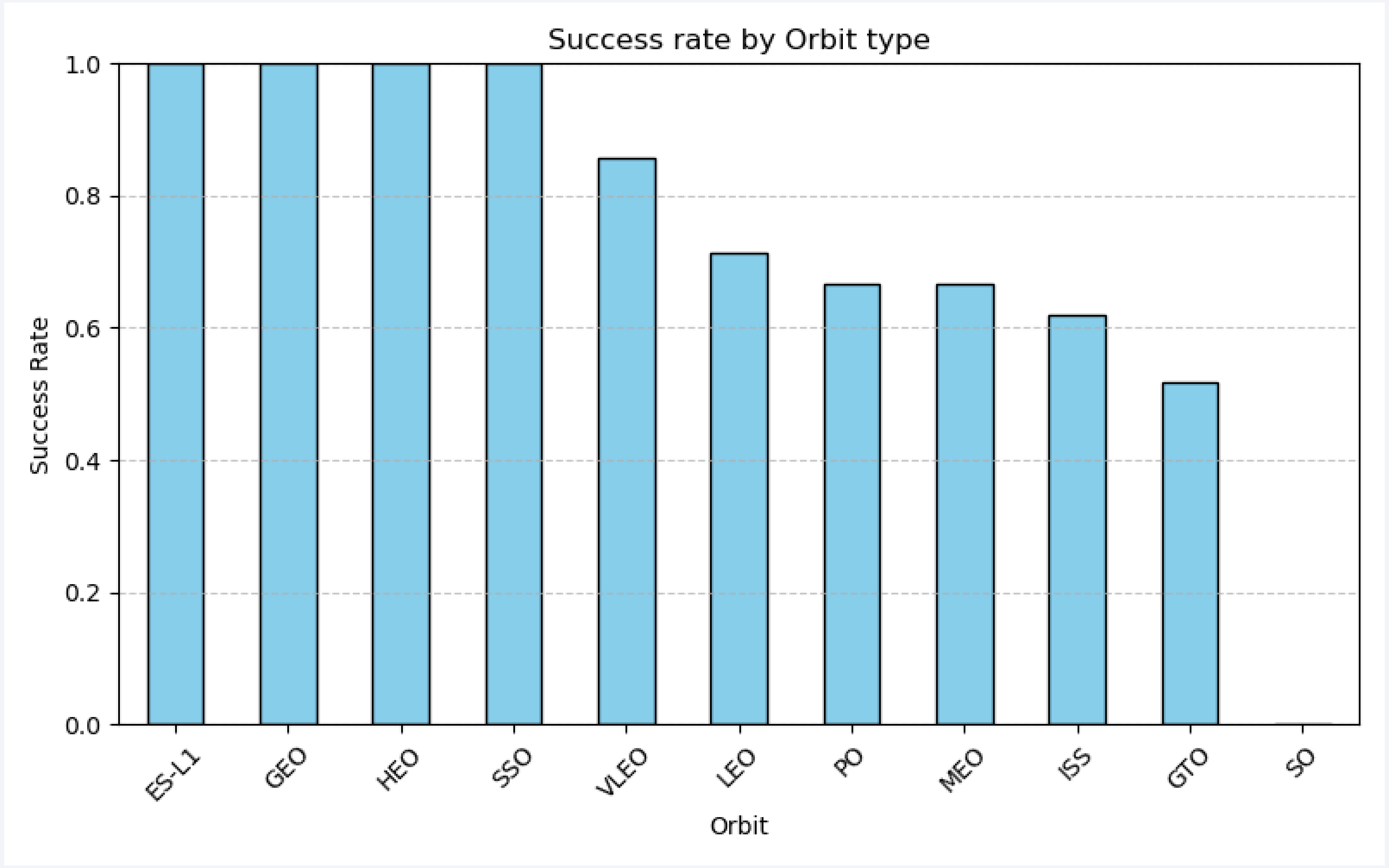
This scatter plot shows that the CCAFS SLC 40 is the most used launch site from the begin to the end of the time period in the dataset. We can also infer that the KSC LC 39A launch site has been built later than the other launch sites. Moreover, this plot tells us that VAFB SLC 4E launch site has the best success rate, but it also has the lower total number of launches.

## Payload vs. Launch Site



This plot shows that the better part of the launches payload mass is less than 8000 kg. Moreover, launches with payload mass larger than 8000 kg has a better success rate. We can also see that KSC LC 39A has never been used for launches where payload mass is less than 2000 kg. On the other hand, all the launches on VAFB SLC 4E have payload mass lower than 10000 kg. Closing, launches with payload mass between 8000 kg and 13000 kg have never been performed at CCAFS SLC 40 site.

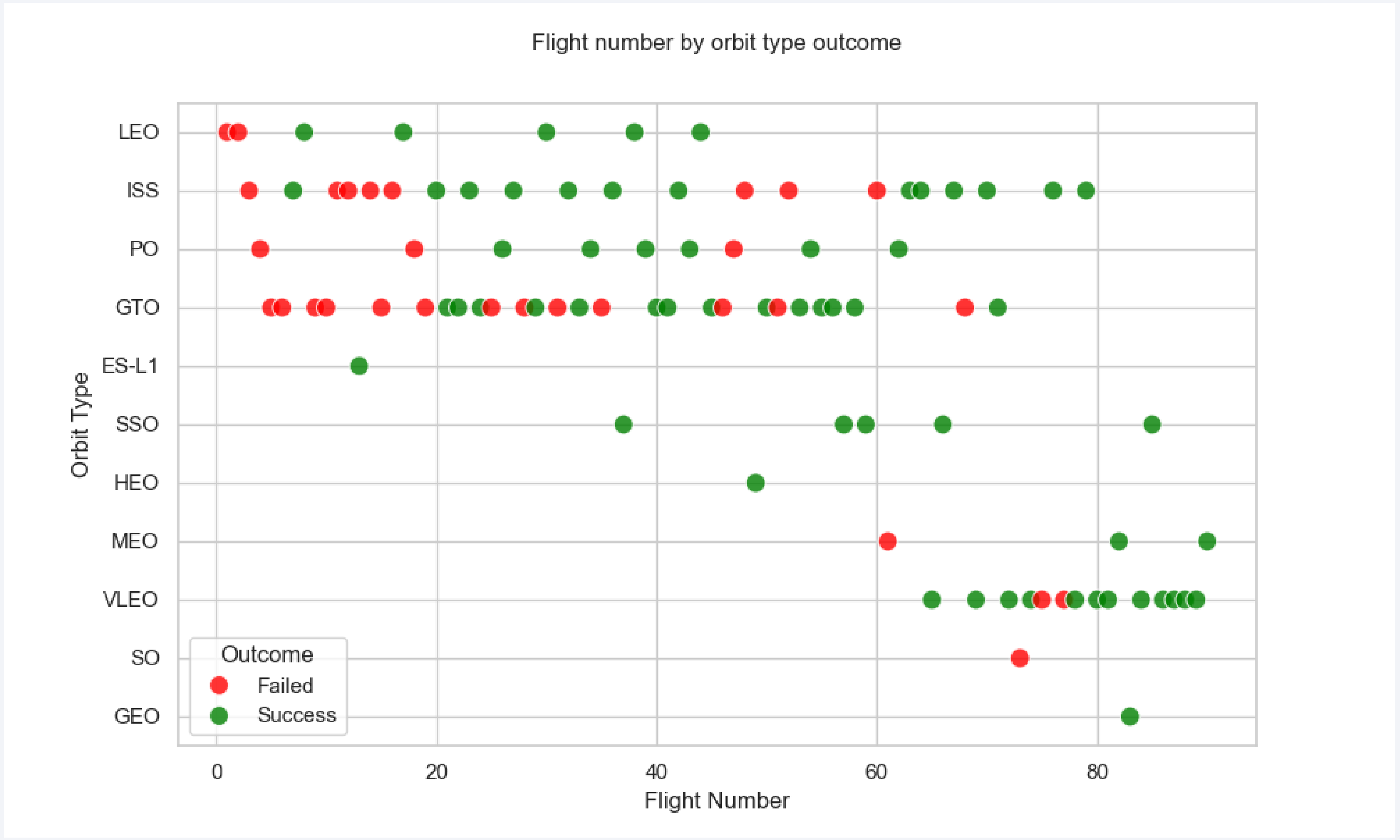
## Success Rate vs. Orbit Type



This graph give us ageneral view aboutsuccess rate based onthe orbit type.

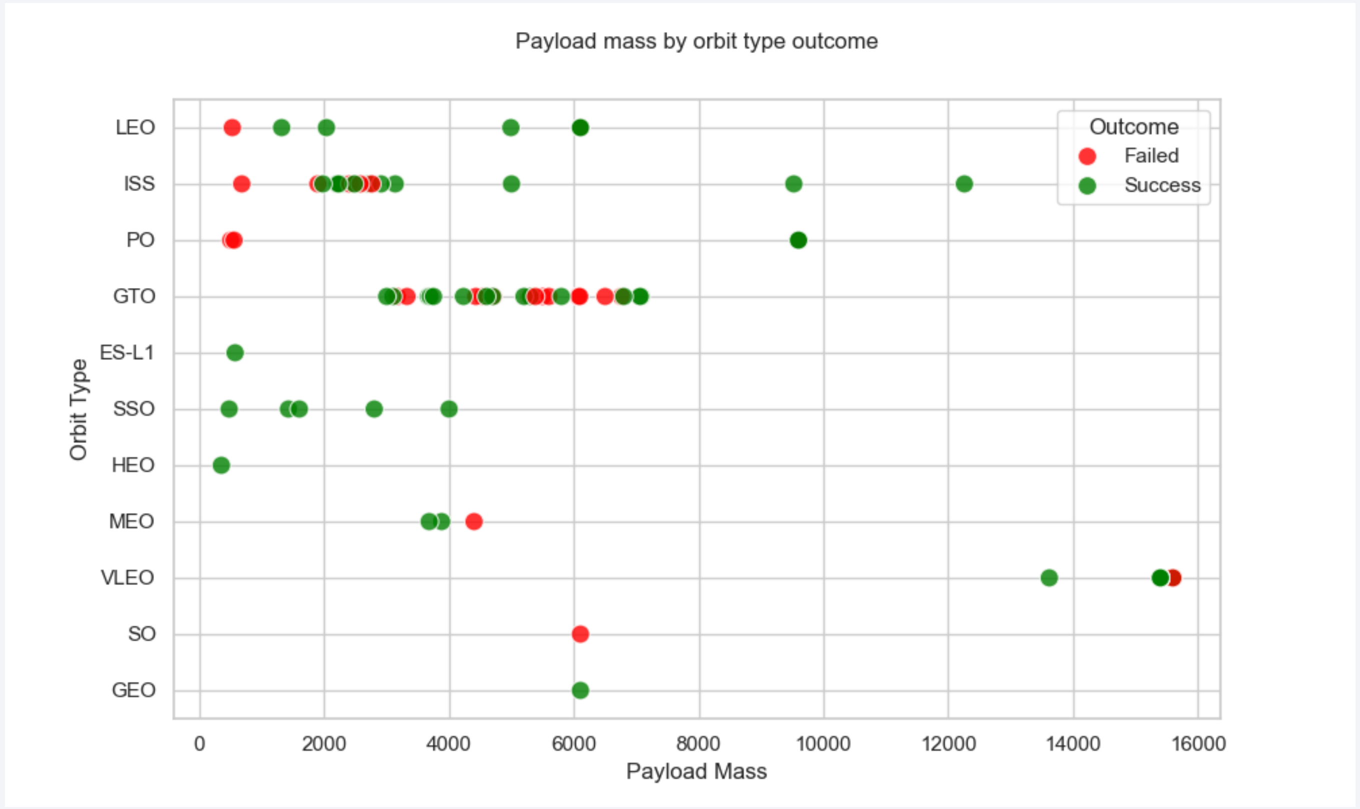
This bar chart let us say that ES-L1, GEO, HEO and SSO orbits have the best success rate (100%). In the following we have VLEO, LEO, PO and MEO, ISS, GTO and SO (0%).

# Flight Number vs. Orbit Type



This scatter plot, like the bar chart, gives us a overall view on success launches by orbit type. Despite of the previous graph, we can see that the SpaceX started using SSO, HEO, MEO, VLEO, SO and GEO orbit type in a second time. The other orbit types has been used up to 80th launch. Moreover we can see that we have only one launch for ES-L1, HEO and GEO orbit type. This last insight, let us say that about the previous graph that the 100% success rate about this three orbits is less significant compared with the others.

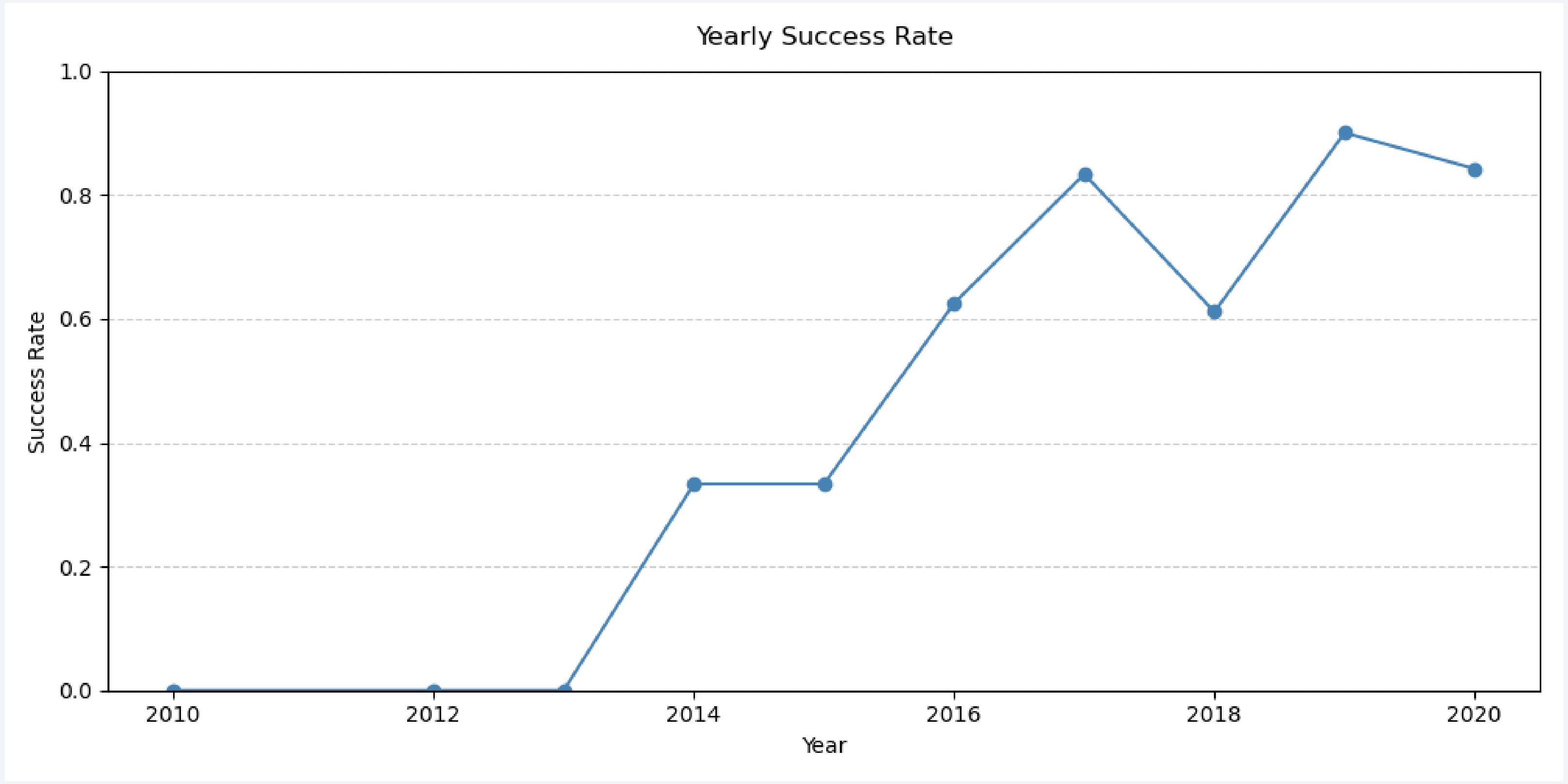
# Payload vs. Orbit Type



Looking at the graph, we can say:

- only payload mass up to 4000 kg have been involved for SSO orbit
- Mass around 4000 kg have been used for MEO orbit
- Mass above 13000 kg have been used for VLEO orbit
- Mass between 2000 kg and 8000 kg has been used for GTO orbit
- The larger mass range is for the ISS orbit, which starts from nearly 500 kg and ends a bit above 12000 kg

## Launch Success Yearly Trend

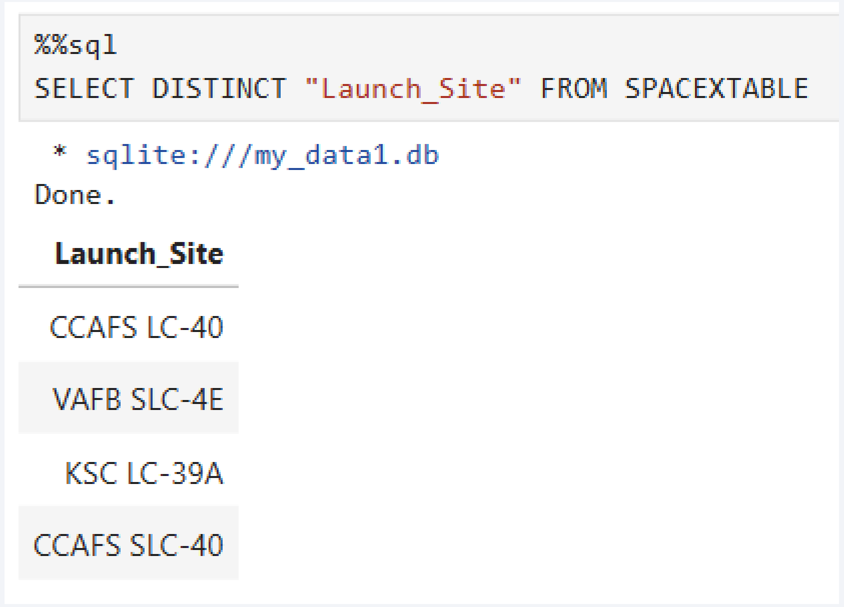


This line chart tells us that there is a clear increase of the success rate from 2013 to 2017. Before this time range, only negative outcomes has been collected. The top success rate has been reached in 2019. Overall, we can say that success rate increases with experience through the years.

## All Launch Site Names

Like all SQL queries, we start with SELECT statement, which begins the query line. In order to extract all the launch site, we use the DISTINCT statement, which pick up from the "SPACEXTABLE" only the unique launch site names. Closing the query result is given in the picture:

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



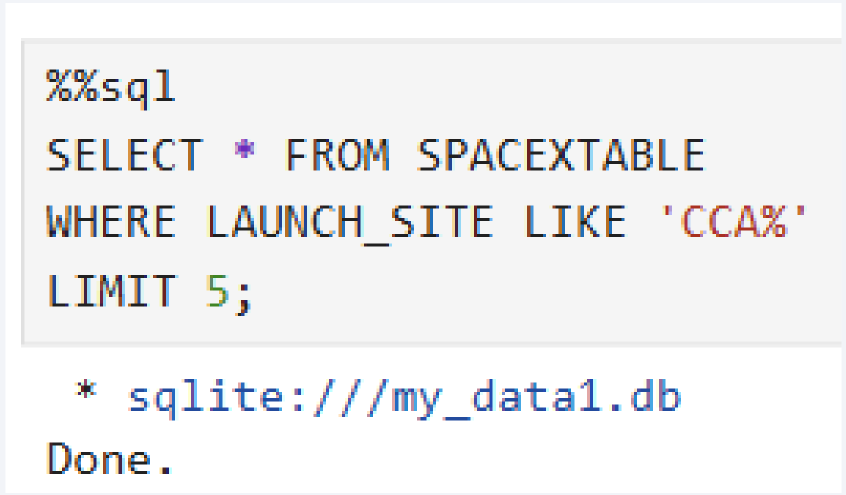
```
%%sql
SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE
```

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

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

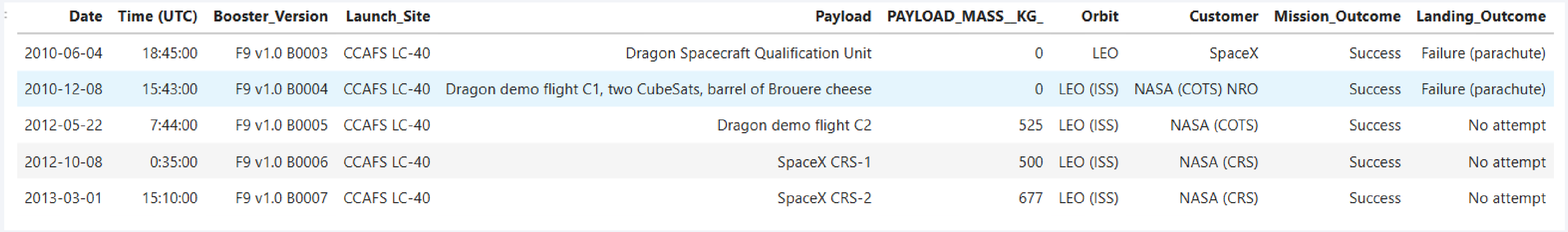
# Launch Site Names Begin with 'CCA'

As always, starting with SELECT statement we select all the attributes (columns) of the table by using a * key. We also select only the records WHERE the "LAUNCH_SITE" is LIKE "CCA%", selecting only the records belonging to the sites which names start with "CCA" string. Closing, we limit the output in 5 records only.



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

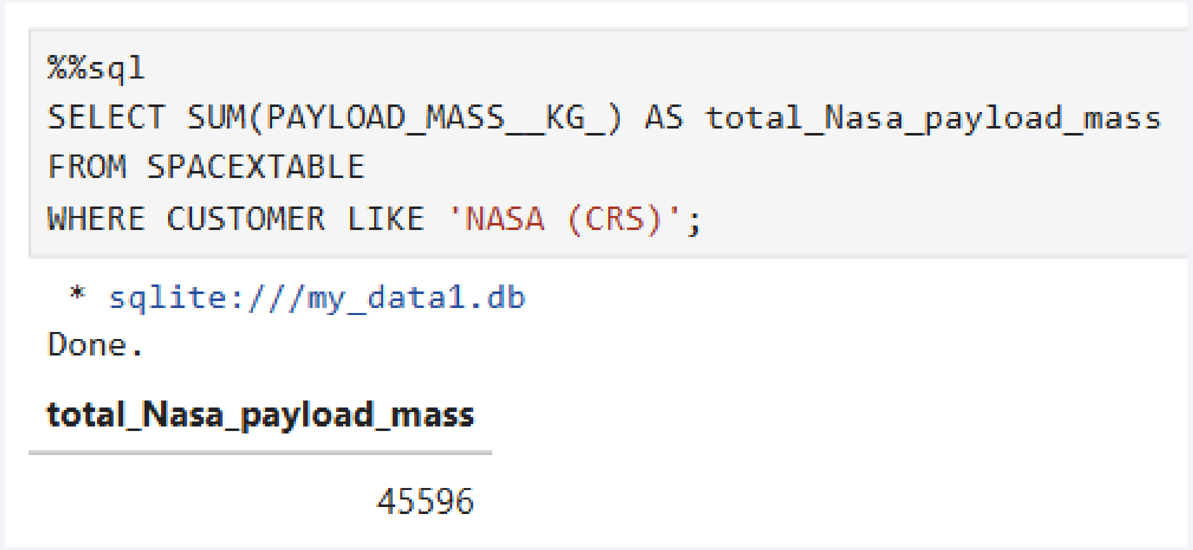
```
 * sqlite:///my_data1.db
Done.
```



| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success | Failure (parachute) |
| 2010-12-08 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of Brouere cheese | 0 | LEO (ISS) | NASA (COTS) NRO | Success | Failure (parachute) |
| 2012-05-22 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

## Total Payload Mass

In this query we used the SUM() function to reveal the total payload mass WHERE the customer is LIKE "NASA (CRS)". The result of the query is a total payload mass of 45596 kg.



```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS total_Nasa_payload_mass
FROM SPACEXTABLE
WHERE CUSTOMER LIKE 'NASA (CRS)';
```

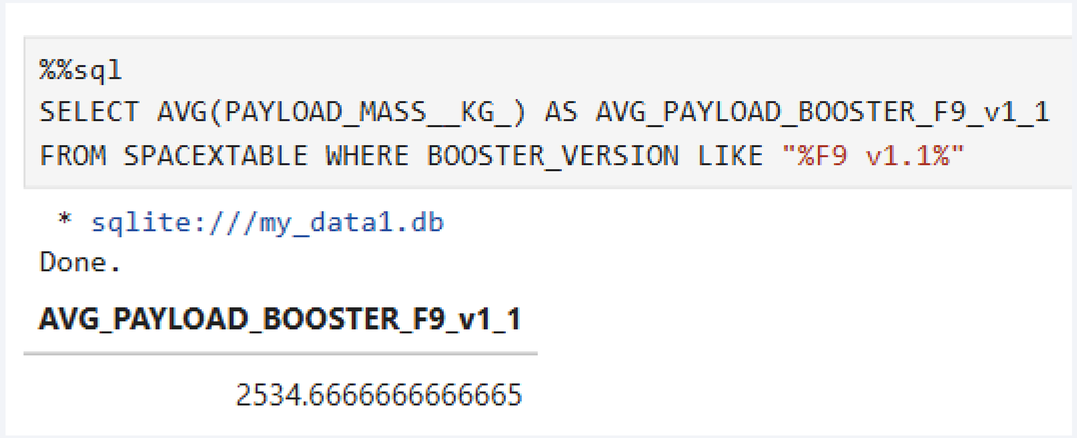
```
 * sqlite:///my_data1.db
Done.
```

| total_Nasa_payload_mass |
|---|
| 45596 |

## Average Payload Mass by F9 v1.1

This query ueses another function to calculate the average payload mass of the F9 booster: AVG().

The result is given in the figure and the number shown is 2534,7 kg



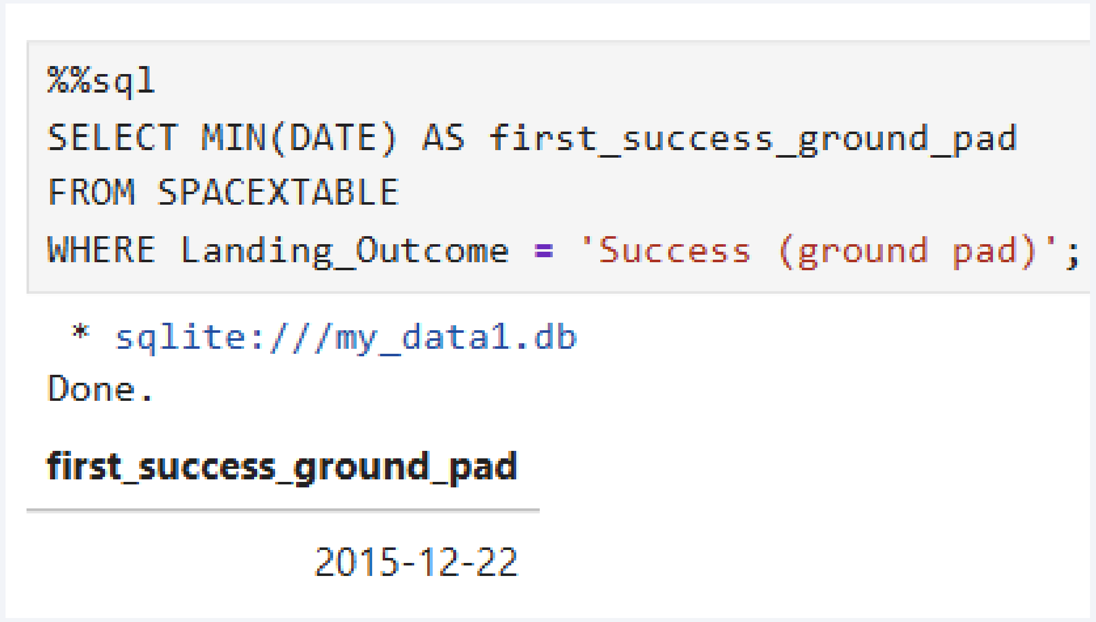
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS AVG_PAYLOAD_BOOSTER_F9_v1_1
FROM SPACEXTABLE WHERE BOOSTER_VERSION LIKE "%F9 v1.1%"
```

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

| AVG_PAYLOAD_BOOSTER_F9_v1_1 |
|---|
| 2534.6666666666665 |

## First Successful Ground Landing Date

This query retrieve the first successful landing outcome on the ground: 2015-12-22. The result of the query is given in the image and its has the form of a new table called "first_success_ground_pad".



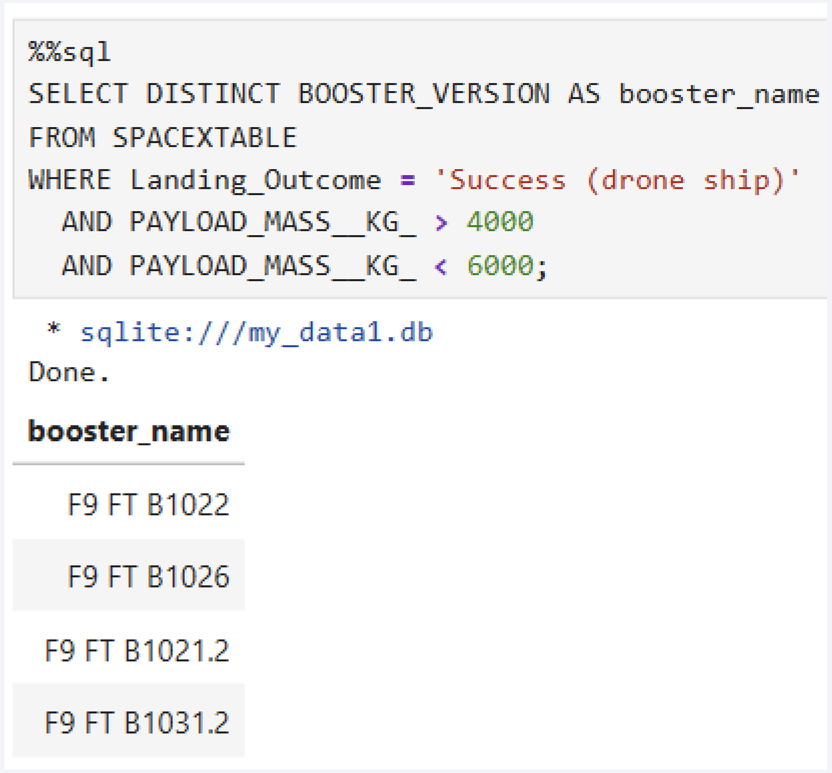
```
%%sql
SELECT MIN(DATE) AS first_success_ground_pad
FROM SPACEXTABLE
WHERE Landing_Outcome = 'Success (ground pad)';
```

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

| first_success_ground_pad |
|---|
| 2015-12-22 |

## Successful Drone Ship Landing with Payload between 4000 and 6000

The query give us the successful landings for payload mass from 4000 kg up to 6000 kg. In the query, a logical AND structure is used. The query result is given in the screenshot as a table named "booster_name".



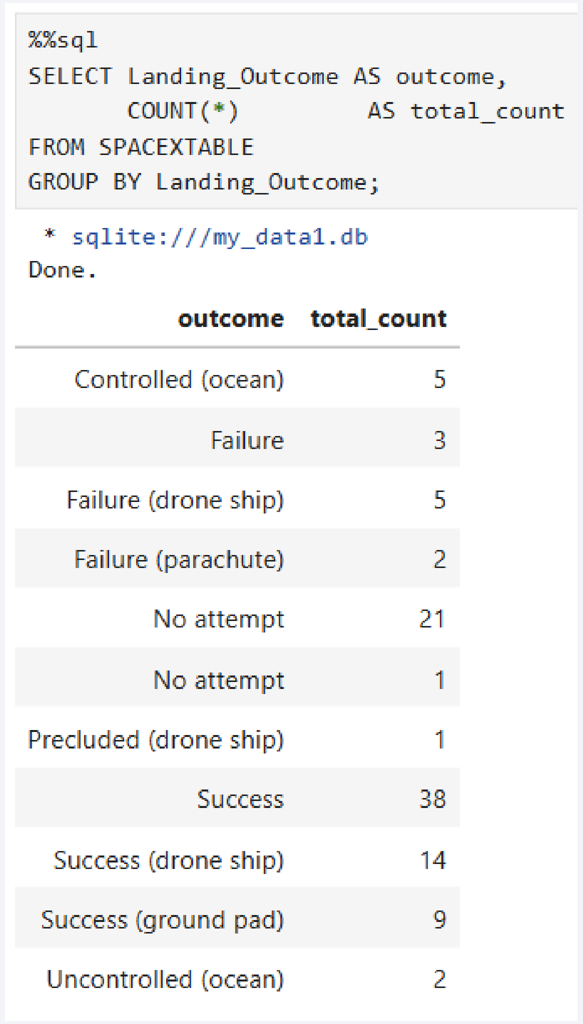
```
%%sql
SELECT DISTINCT BOOSTER_VERSION AS booster_name
FROM SPACEXTABLE
WHERE Landing_Outcome = 'Success (drone ship)'
  AND PAYLOAD_MASS__KG_ > 4000
  AND PAYLOAD_MASS__KG_ < 6000;
```

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

| booster_name |
|---|
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

## Total Number of Successful and Failure Mission Outcomes

This query generates a table as its output. This is possible using two AS statement before the FROM statement. Each AS statement refers to a column in the output table. Moreover, on the query there is a grouping relatively to the landing outcome.



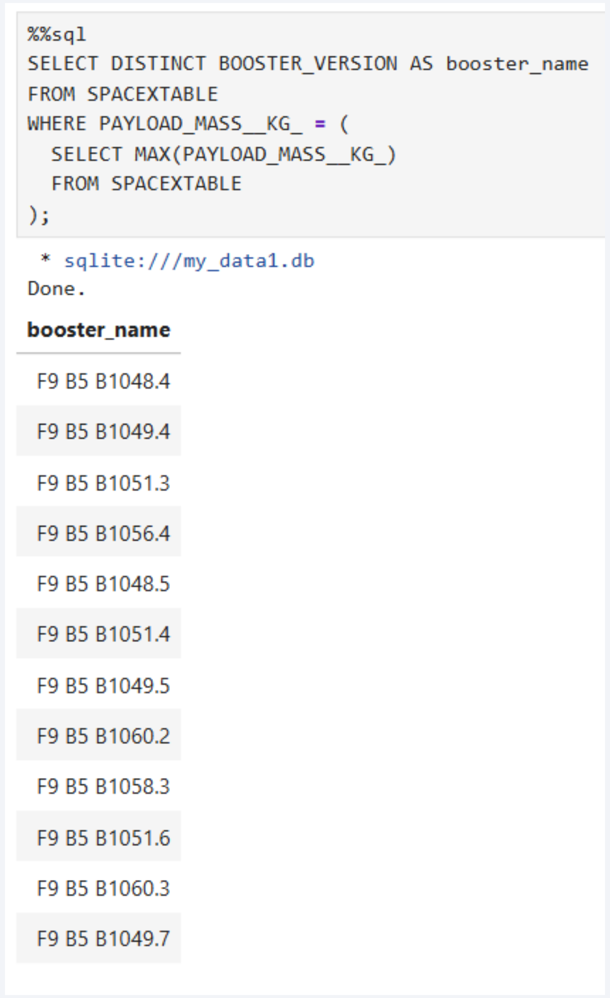
```
%%sql
SELECT Landing_Outcome AS outcome,
       COUNT(*)         AS total_count
FROM SPACEXTABLE
GROUP BY Landing_Outcome;
```

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

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

## Boosters Carried Maximum Payload

This is not a simple query. We are seeing a subquery selection. This approach is necessary when we have to subset the records relatively to a quantity obtained from a mathematical operation, like MAX() function. The result of the query gives us the booster names which have carried the maximum payload mass.



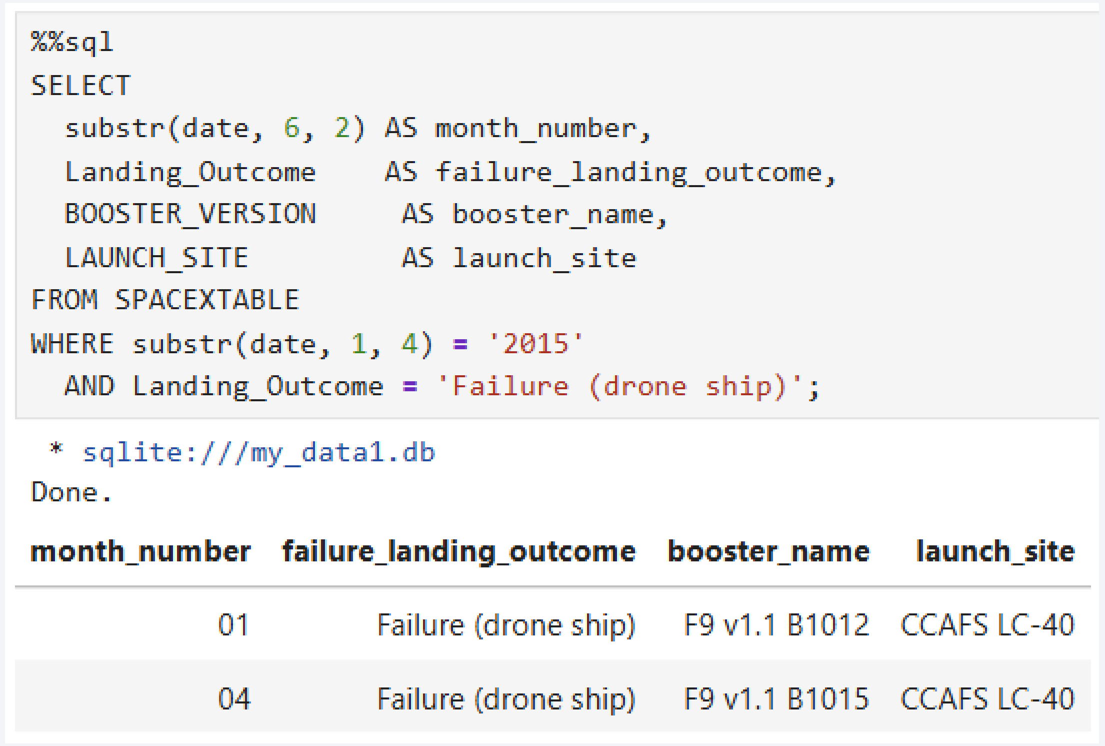
```
%%sql
SELECT DISTINCT BOOSTER_VERSION AS booster_name
FROM SPACEXTABLE
WHERE PAYLOAD_MASS__KG_ = (
  SELECT MAX(PAYLOAD_MASS__KG_)
  FROM SPACEXTABLE
);
```

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

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

# 2015 Launch Records

The query give us as result the subset of negative launches performed in 2015. The first "substr()" statement collects the number of the month as a sub-string of the date string (YYYY-MM-DD format); the second "substr()" statement collects the year from the date string. This two "substr()" statements work as filter in the WHERE clause.



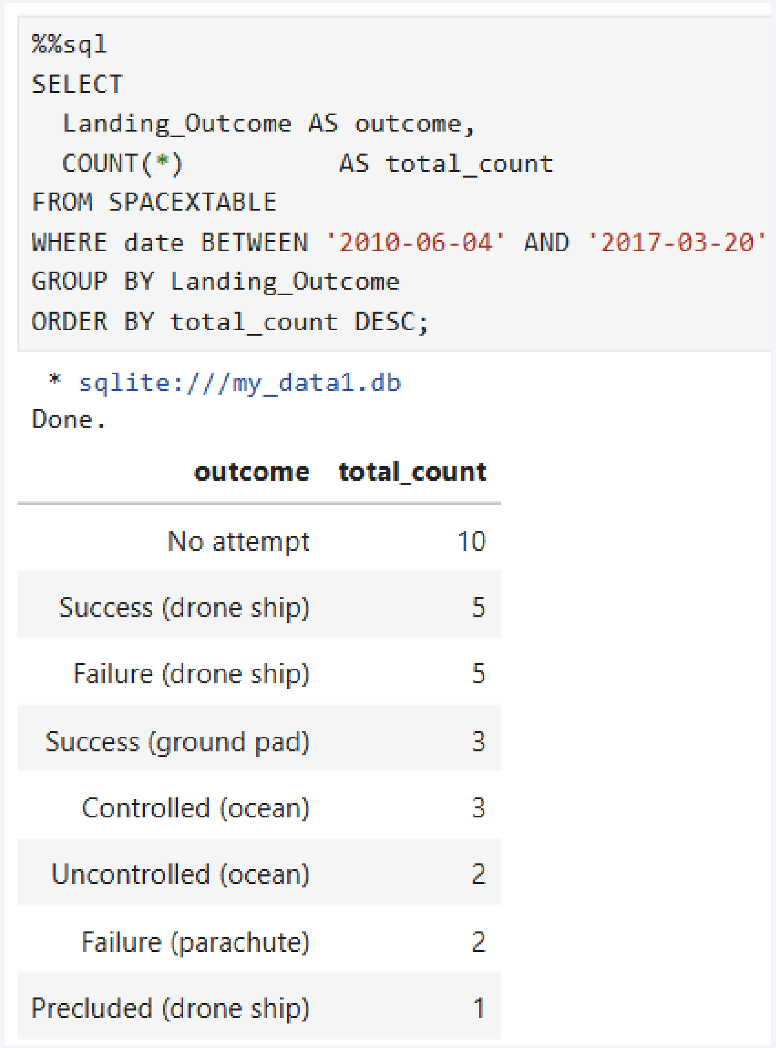
```
%%sql
SELECT
  substr(date, 6, 2) AS month_number,
  Landing_Outcome    AS failure_landing_outcome,
  BOOSTER_VERSION     AS booster_name,
  LAUNCH_SITE         AS launch_site
FROM SPACEXTABLE
WHERE substr(date, 1, 4) = '2015'
  AND Landing_Outcome = 'Failure (drone ship)';
```

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

| month_number | failure_landing_outcome | booster_name | 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 2010-06-04 and 2017-03-20

The query ranks 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 (DESC) order using "ORDER BY" clause. The output is a table with the columns "outcome" and "total_outcome", which tells us that the most common outcome for Landing is "No attempt".



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

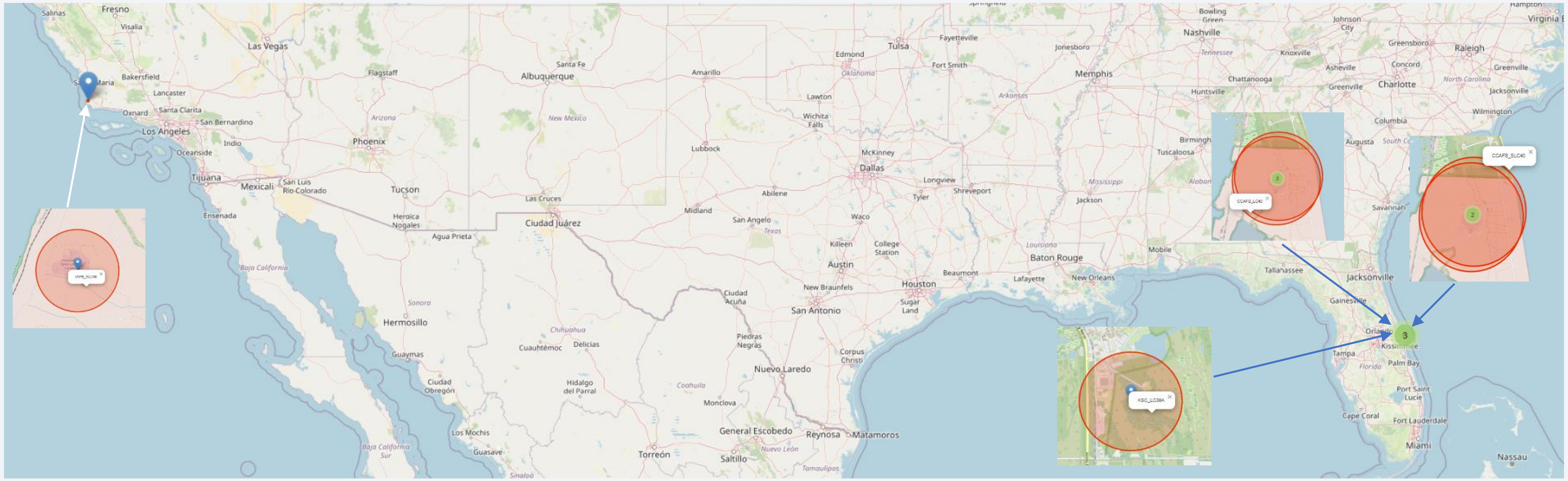
```
 * sqlite:///my_data1.db
Done.
```

| outcome | total_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

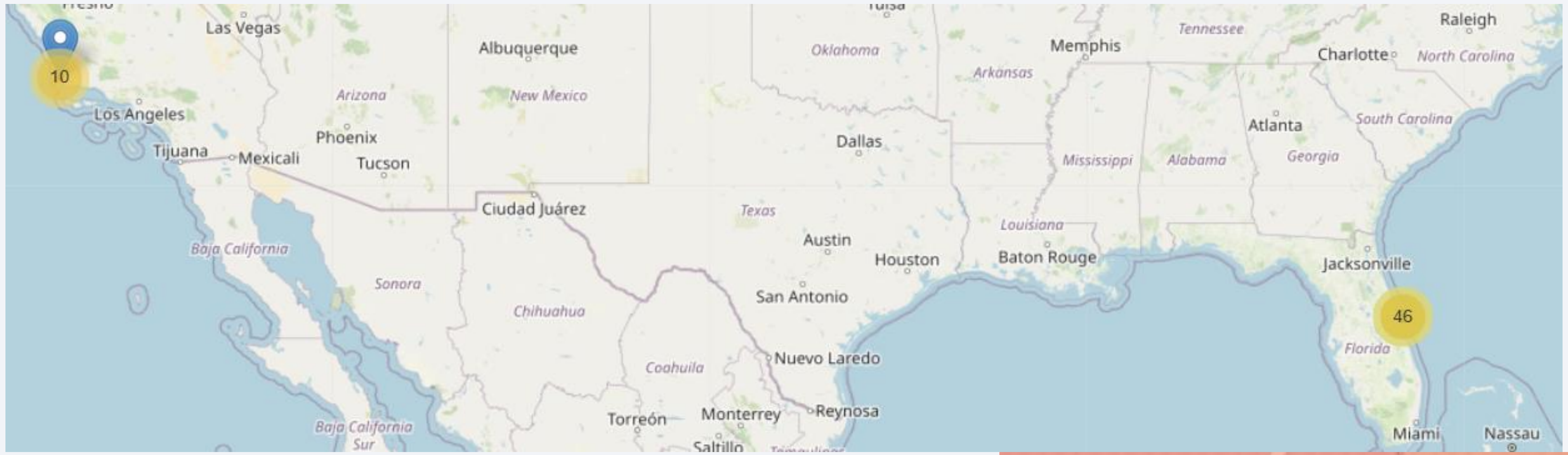
## Launch Sites Proximities Analysis

# Launch sites map



As we can see from the screenshot, launch sites are clearly near the coast. This makes sense because one of the landing outcome is to land exactly on the ocean. For each launch site, we have a pop-up that's show up by clicking in the red circle area.

## Succesful/Failure launches on map

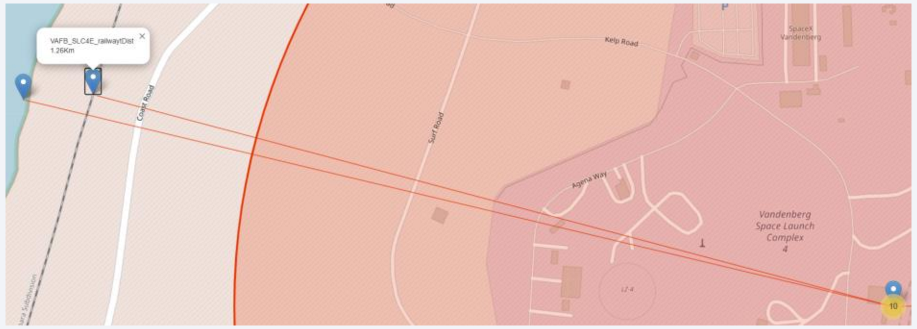


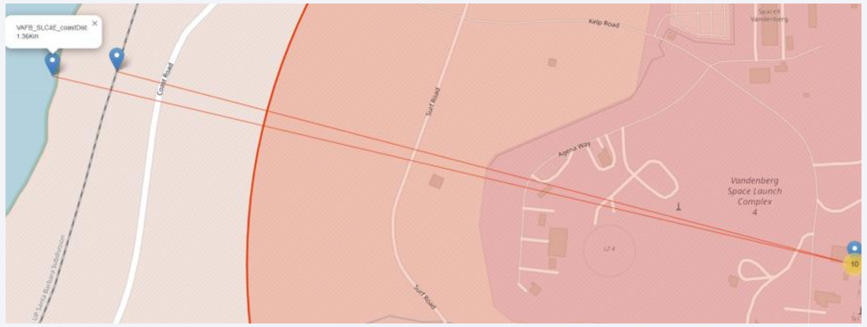
As we can see from the screenshot, each launch site has its launches marked on the map with circles. From the global view, we can see that the majority of the launches has been performed on the eastern coast sites.

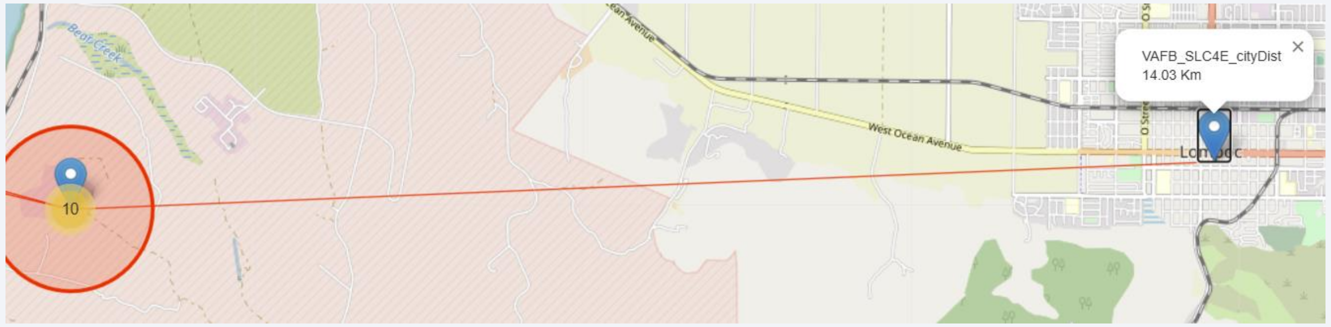
CCAFS_SLC40
7
26

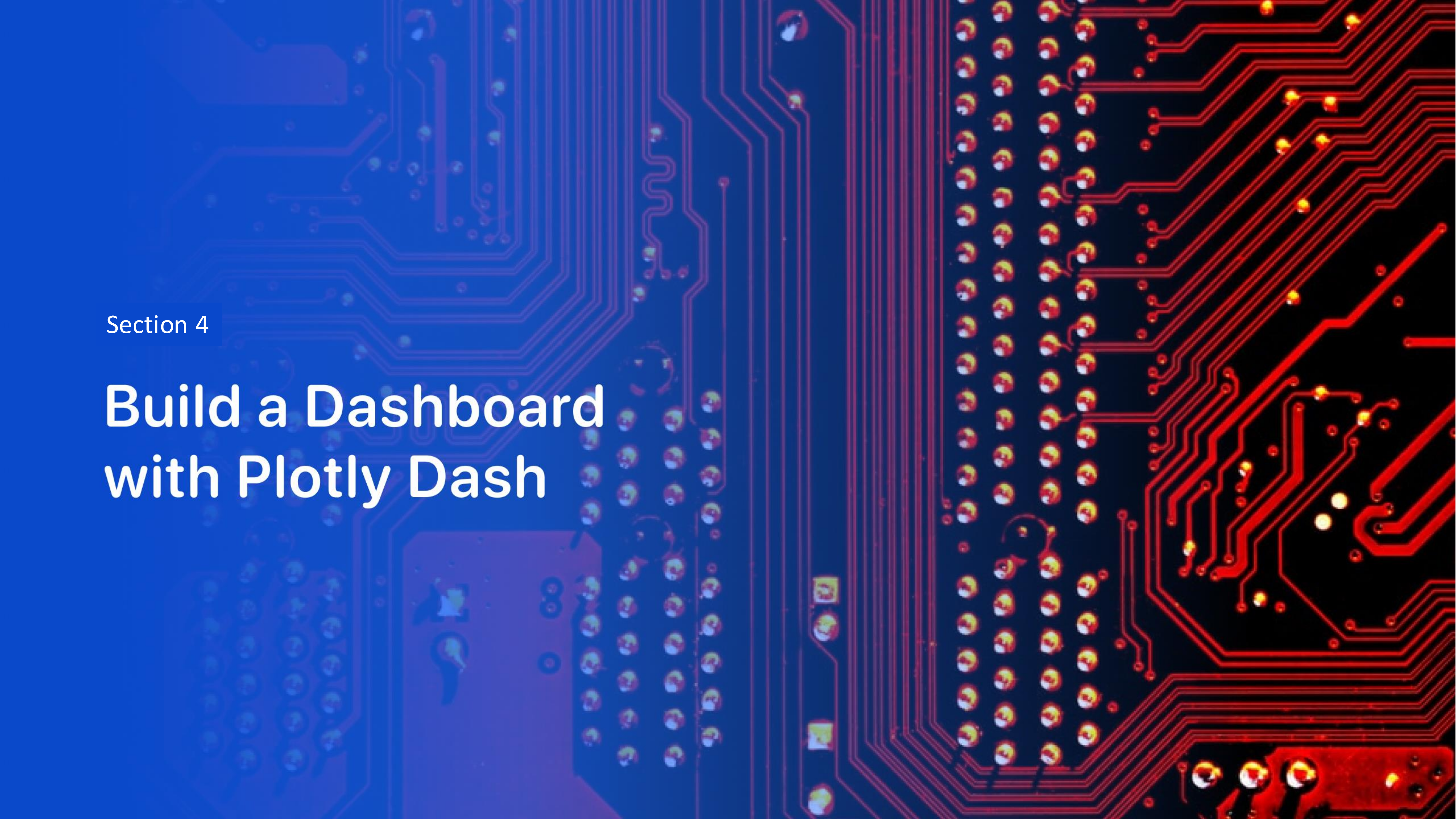
## Relevant points distances

In the screenshot we can see the distances of the VAFB SLC-4E site from the nearest coastline, the nearest railway and the nearest city. As we can see, the nearest city distance from the site is larger than the nearest railway and cost site. It makes sense beacause of citizen are potentially in danger situations if a launch fails.







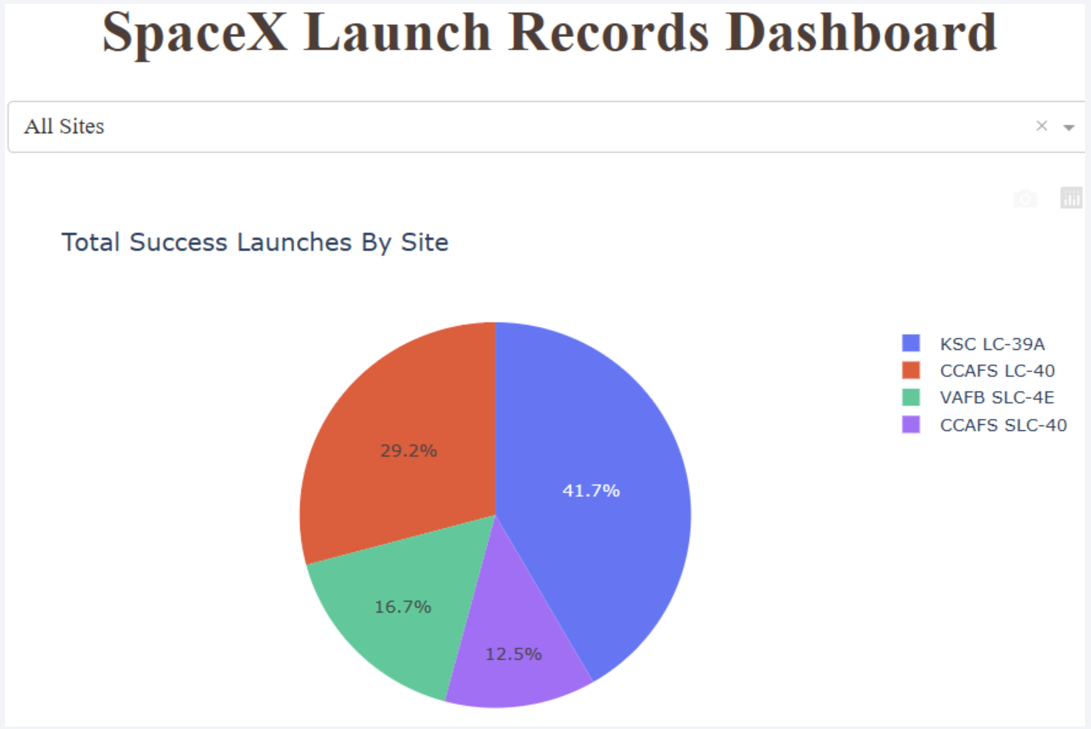


Section 4

## Build a Dashboard with Plotly Dash

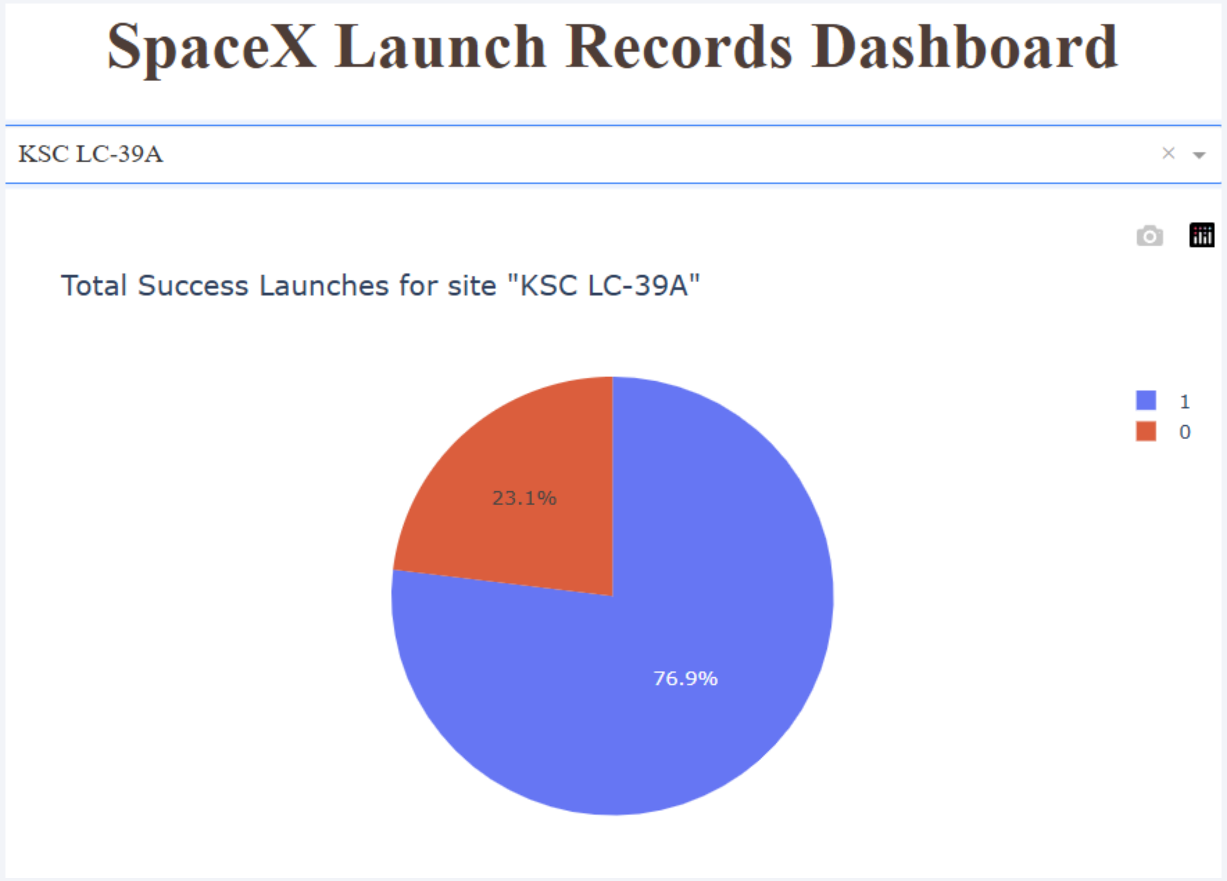
# All sites success rate in a Pie chart

Looking at the screenshot token on the plotly dash app, we can see that the launch site with highest successes, compared with the total number of launches, is KSC LC-39 site, followed by CCAFS LC-40, VAFB SLC-4E and CCAFS SLC-40.



## KSC LC-39A success rate

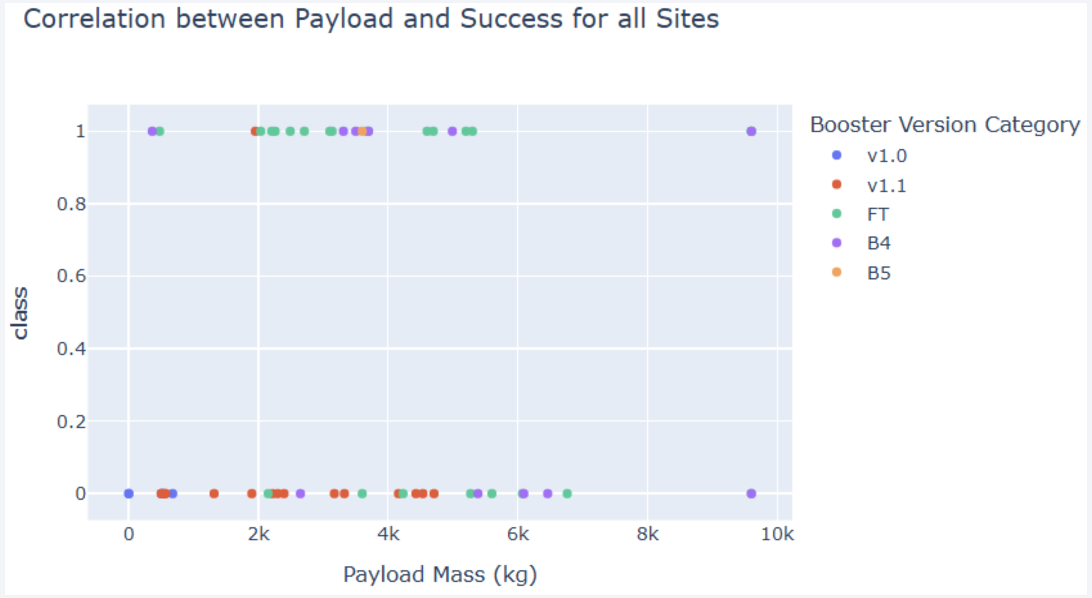
Looking at the Pie chart, we can see a launch success rate of 76.9% for KSC LC-39A site.

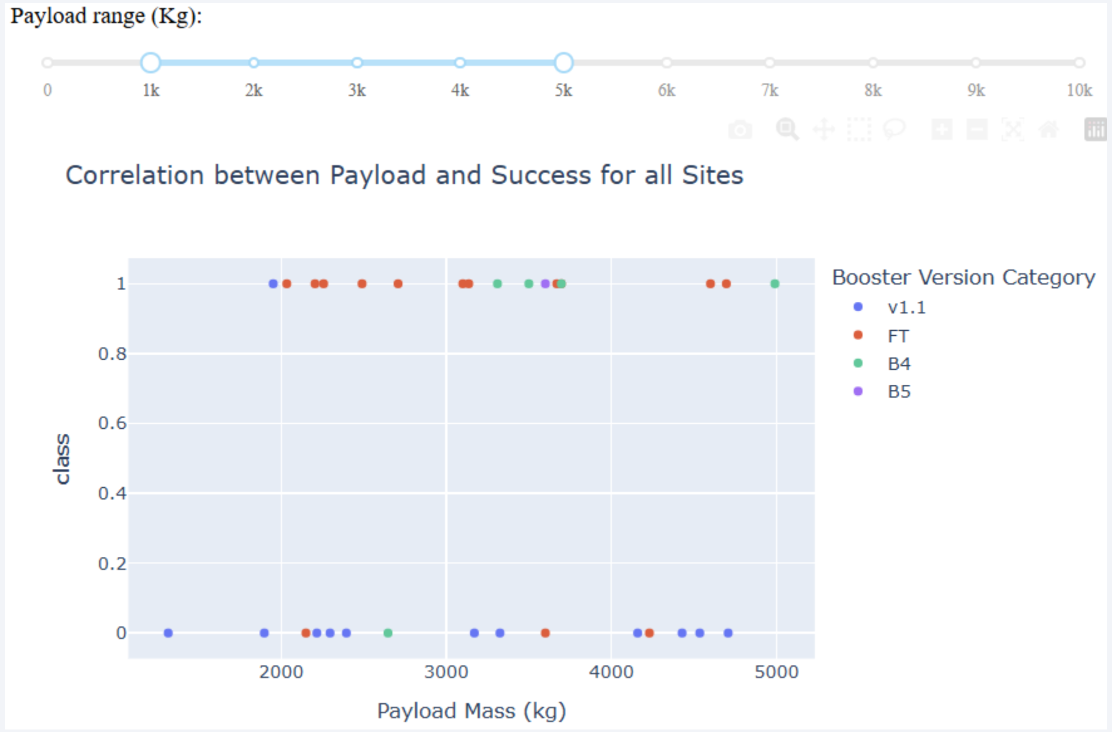

SpaceX Launch Records Dashboard

KSC LC-39A

Total Success Launches for site "KSC LC-39A"

## Correlation between Payload and Success





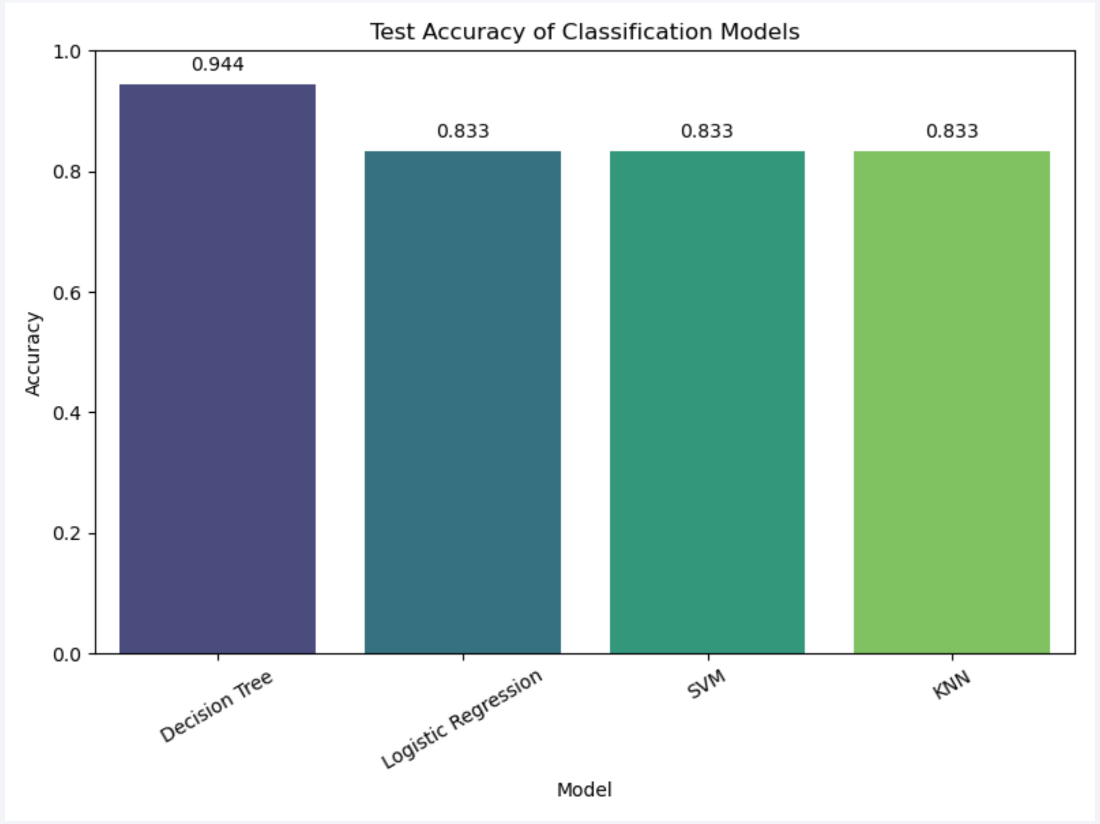
Looking at the screenshots above, we can see that for payload greater than 6000 kg, success rate decreases significantly: we have a total of 6 launches with only 1 positive outcome. Below 6000 kg, we can see an almost equal partition between successful and failure launches.

Section 5

## Predictive Analysis (Classification)

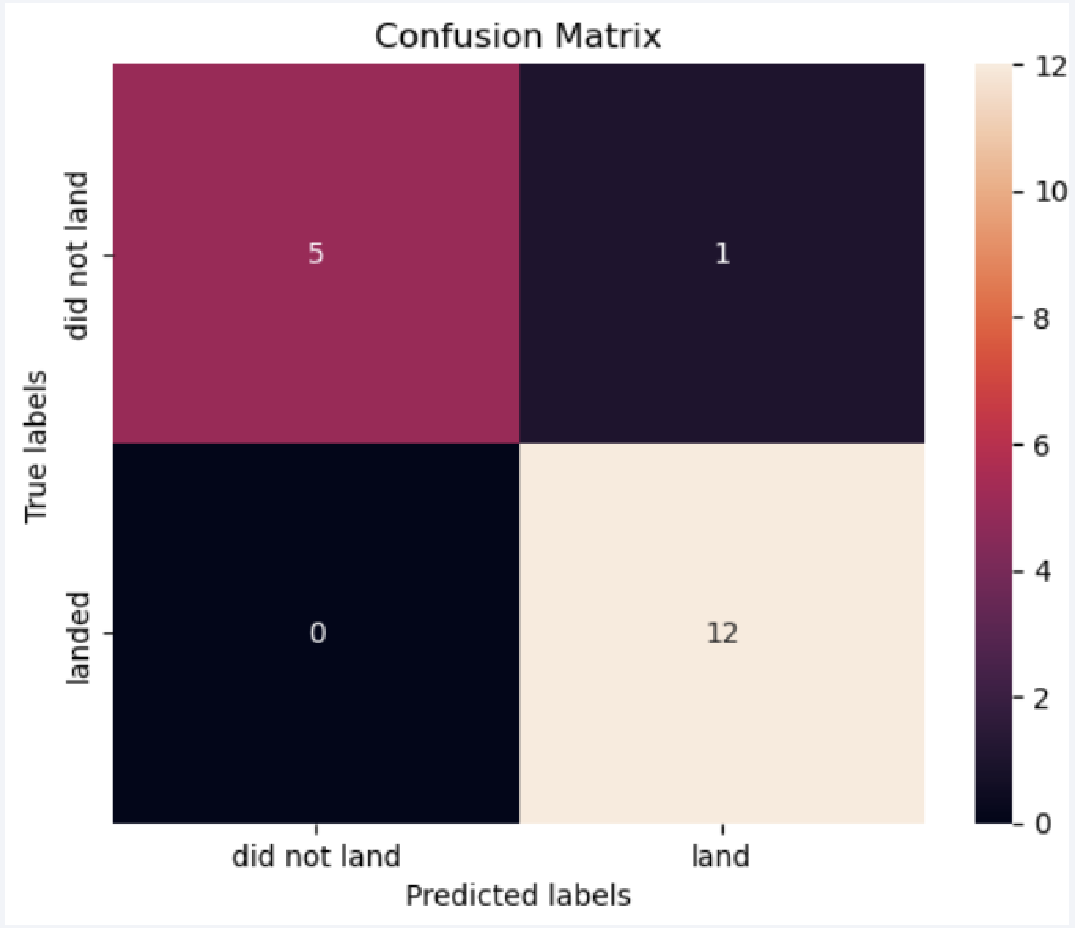
## Classification Accuracy

In the graph, we can see a bar chart showing the accuracy of the deployed models. The accuracy score is the same for Logistic regression, SVM and KNN. The best performing model is clearly the Decision Tree model, with an accuracy score of 0.944 = 94%.



## Confusion Matrix

On the right we have the confusion matrix of the Decision Tree model. This matrix shows that the model performs very good. In fact, we have only one "false positive" classification and all the others classifications equal to the real outcome.



## Conclusions

- Firstly, we can conclude that the decision tree is the ML predictive model that performs the best in launches outcome prediction.
- Logistic Regression, SVM and KNN models perform exactly the same compared each other.
- ML predicting models are a powerful tools for predicting behaviors on historical data.
- Each model performs best in certain cases, depending on the amount, kind and population of data
- The low weighted payloads (4000kg and below) performed better than the heavy weighted payloads.
- Looking at the timeline, in the years between 2013 and 2020, SpaceX success launches rate increases. This tell us that experience can teach.
- KSC LC-39A have the most successful launches of any sites with 76.9%.
- SSO orbit have the most success rate: 100% with more than 1 occurrence.

Thank you!