IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Sergio Paulo
May 11, 2023

# Outline

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

# Executive Summary

- Summary of methodologies
- Summary of all results

# Introduction

SpaceX caused a revolution in the space industry by substantially reducing the cost of launching rockets compared to other companies. Falcon 9 rocket missions cost $62 million; other providers charge $165 million. This was possible because the company manages to reuse the first stage of the rocket. This reuse depends on a successful landing of this first stage after launch.

By predicting the successful landing of the first stage through a classification model, I will estimate the cost of a launch. These results will be very useful for decision-making in companies competing with SpaceX.

I will also explore the factors that determine the success of the first stage landing and understand the complex interactions between several characteristics that influence the success rate. Variables such as payload mass, launch site, and orbits can impact the outcome of the first stage landing. I will evaluate SpaceX's continued improvement by showing whether the rate of successful landings has increased over the years.

To achieve these goals, I will use public information and machine learning models to predict first-stage reuse. This allows estimating the total cost of launches, providing crucial insights for assessing the viability of Space Y, a new company looking to compete with SpaceX. The analysis will also help in identifying the most suitable algorithm for binary classification in this specific case, ensuring accurate predictions.

Section 1

# Methodology

# Methodology

## Executive Summary

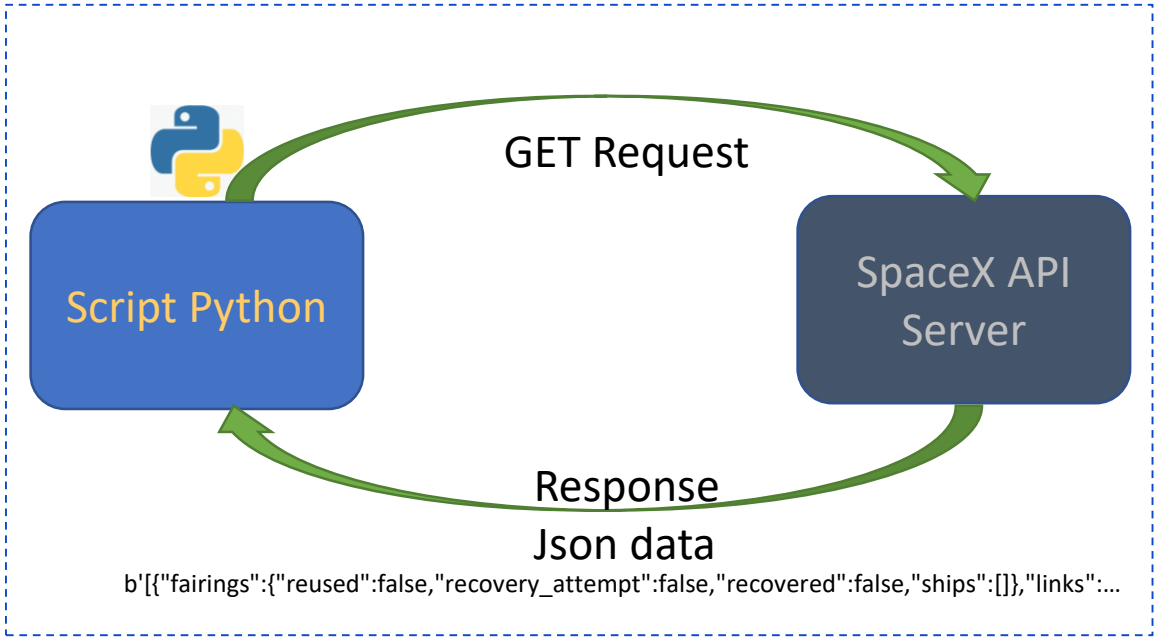
- Data collection methodology:
  - Data was collected from SpaceX API and web scrapping from Wikipedia list of Falcon 9 launches
- Perform data wrangling
  - Data was filtered, missing values was treated and categorical data was one-hot encoded.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Data was split in training and test data and submitted to four classification models. Then was verified the accuracy of each outcome.

# Data Collection

- Data sets were collected from two sources.
  - First I got the data from SpaceX API in json format from https://api.spacexdata.com/v4/launches/past.
  - Then I scraped the Wikipedia page that show a table of launches from https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches

# Data Collection - SpaceX API

- The data, in Json format, was collected from SpaceX API Server
- The data was filtered to contain only Falcon 9 launches.

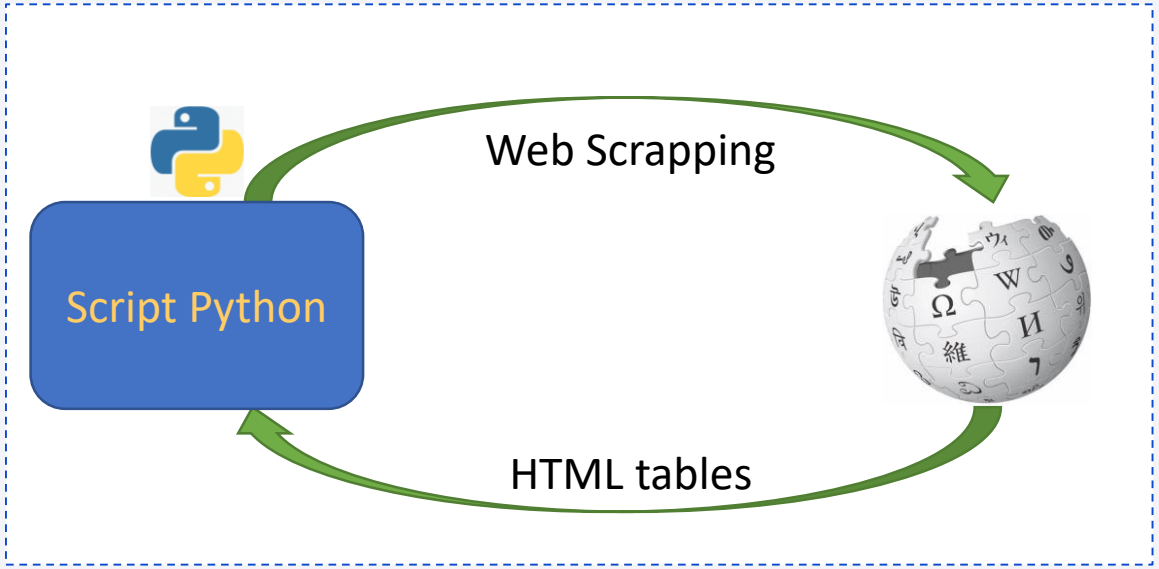



The code can be accessed in:

https://github.com/goldsp/Applied-Data-Science-Capstone-IBM-Coursera/blob/main/0101a-jupyter-labs-spacex-data-collection-api.ipynb.

# Data Collection - Scraping

- First I get the Falcon 9 launches Wikipedia page and extract the tables with the Beautiful Python Library.
- Then I put the data in a dataframe.




The code can be accessed in:
https://github.com/goldsp/Applied-Data-Science-Capstone-IBM-Coursera/blob/main/0102a-jupyter-labs-webscraping.ipynb.

# Data Wrangling

- Some records in dataframe had the feature Payload Mass with null values. The these null values was changed by Payload mean of all rows in dataframe.
- From the feature outcome was definied those that was bad:
  - 'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS‘ and 'None None‘;
- Then was created the column class that received ‘0’ for the rows was a bad outcome. For the other rows, representing good outcomes, the column class takes the value ‘1’. They are: ‘True ASDS’, ‘True RTLS’ and ‘True Ocean’.
  - 0 is a bad outcome, that is, the booster did not land.
  - 1 is a good outcome, that is, the booster did land.

The code can be accessed in

https://github.com/goldsp/Applied-Data-Science-Capstone-IBM-Coursera/blob/main/0101a-jupyter-labs-spacex-data-collection-api.ipynb and

https://github.com/goldsp/Applied-Data-Science-Capstone-IBM-Coursera/blob/main/0103a-IBM-DS0321EN-SkillsNetwork_labs_module_1_L3_labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb.

# EDA with Data Visualization

- To do an exploratory data analysis I used:
  - Scatter plots charts showing failures and success against the tuple of variables:
    - Flight Number and Payload Mass;
    - Flight Number and Launch Site;
    - Pay Load Mass and Launch Site;
    - Flight Number and Orbit;
    - Pay Load Mass and Orbit;
  - A bar chart showing sucess rate by Orbit;
  - A line chart showing the growth of the success rate since 2010.

The code can be accessed in https://github.com/goldsp/Applied-Data-Science-Capstone-IBM-Coursera/blob/main/0202-IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb.

# EDA with SQL

- I performed SQL queries to:
  - List the names of Launch Sites;
  - List records of launches of Cape Canaveral;
  - Calculate the total Pay Load Mass of client NASA;
  - Calculate the avarage Pay Load Mass of launches with rocket Falcon 9 version 1.1;
  - Show the date of the first successful landing outcome in ground pad;
  - List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000;
  - List the total number of successful and failure mission outcomes;
  - List the names of the booster versions which have carried the maximum Pay Load Mass;
  - List the records which display the month names, failure landing outcomes in drone ship ,booster versions, launch_site for the months in year 2015;
  - Count the successful landing outcomes between the date 04-06-2010 and 20-03-2017.
- The code can be accessed in https://github.com/goldsp/Applied-Data-Science-Capstone-IBM-Coursera/blob/main/0201a-jupyter-labs-eda-sql-coursera_sqllite.ipynb.

# Build an Interactive Map with Folium

- A interactive map was created with:
  - A map with initial center location to be NASA Johnson Space Center at Houston to see interesting locations for launch analysis
  - Mark clusters at launch sites to see number of successful launches
  - A distance marker to measure distance between places of interest for analysis
  - Distance indicator line between places of interest

The code can be accessed in https://github.com/goldsp/Applied-Data-Science-Capstone-IBM-Coursera/blob/main/0301a-IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb.

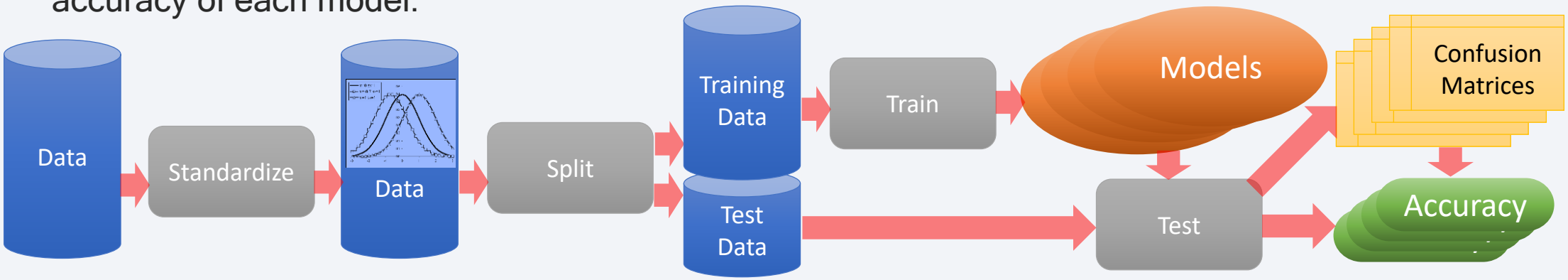
# Build a Dashboard with Plotly Dash

- A dashboard was created with:
  - A pie chart to show the sucess rate of a selected site or the success proportion of all sites.
  - A scatter chart between Pay Load Mass and Class (success or failure) that shows the booster version those have sucess or failure.
  - A dropdown menu to choose wich launch site to analyze or choose all sites to be analized.
  - A range slider to filter a range of Pay Load Mass for analysis.

- The code can be accessed in https://github.com/goldsp/Applied-Data-Science-Capstone-IBM-Coursera/blob/main/0302a-foguete01.py.

# Predictive Analysis (Classification)

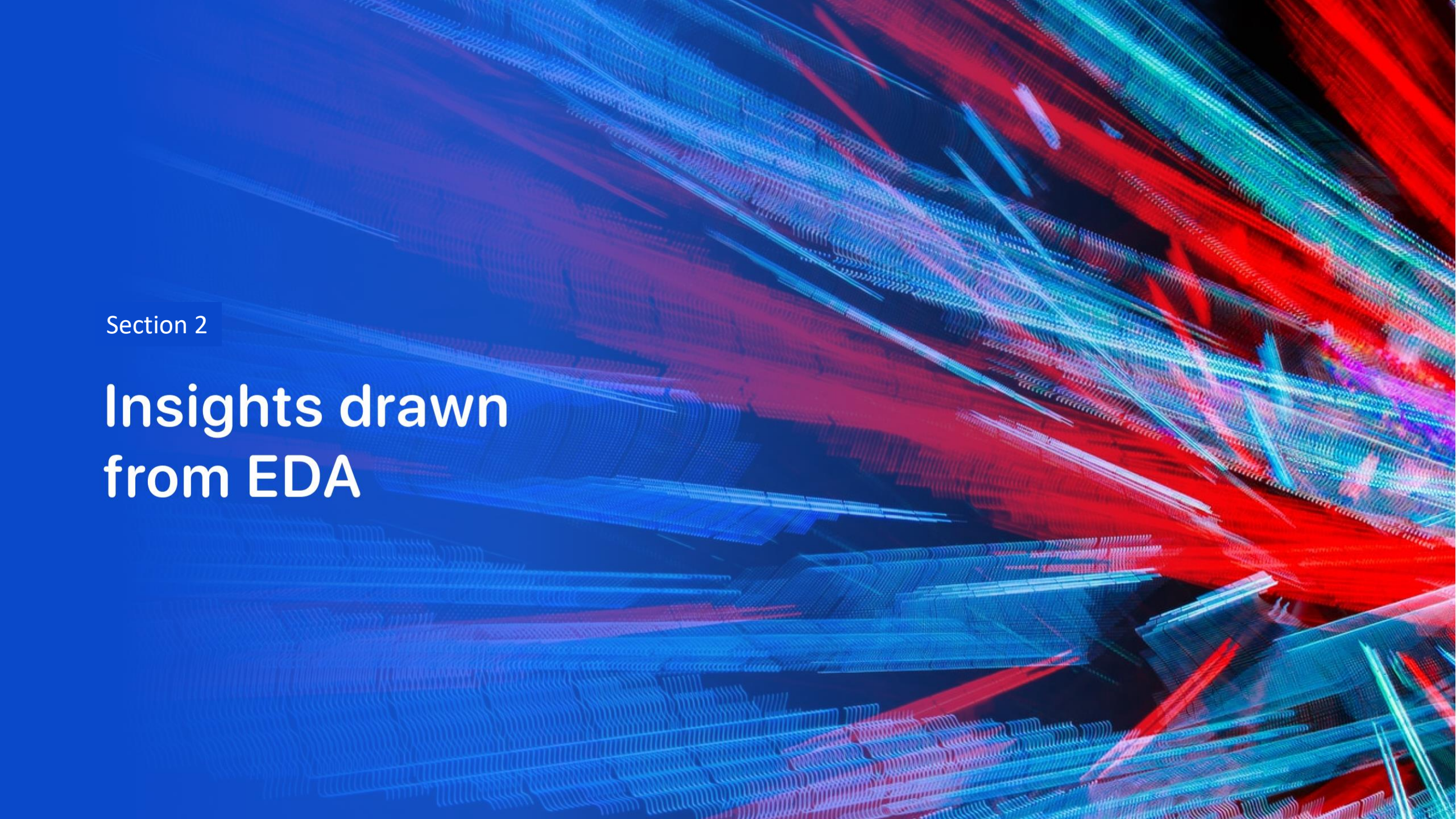
- First I performed a preprocessing step to standardize the data.
- After that I randomly split the data sets into two parts: training data and test data.
- Then I used the training data to train the following models: Logistic Regression, Support Vector Machines (SVM), Decision Tree and K Nearest Neighbour (KNN).
- Finally, I tested the models with the test data and defined a confusion matrix and calculated the accuracy of each model.




The code can be accessed in https://github.com/goldsp/Applied-Data-Science-Capstone-IBM-Coursera/blob/main/0401a-IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb.

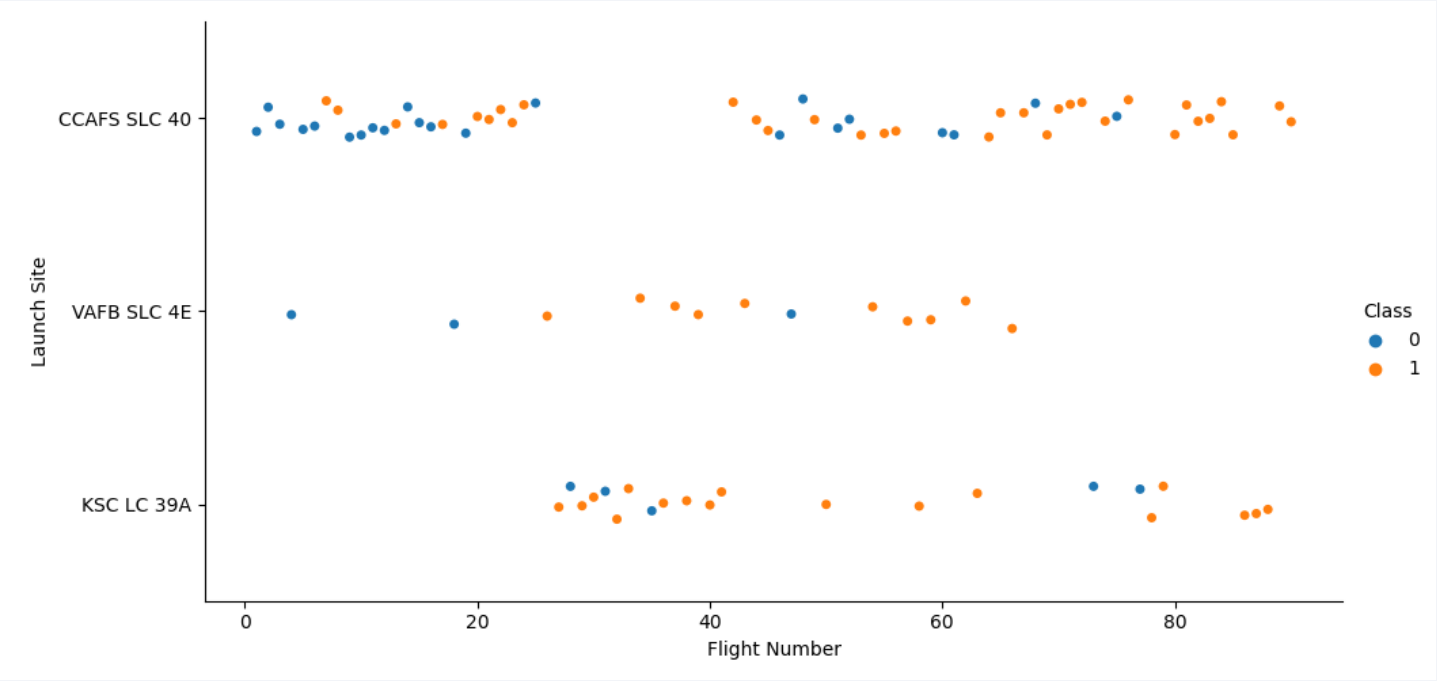
# Results

- In the next sections I will present:
  - Exploratory data analysis results
  - Interactive analytics demo in screenshots
  - Predictive analysis results

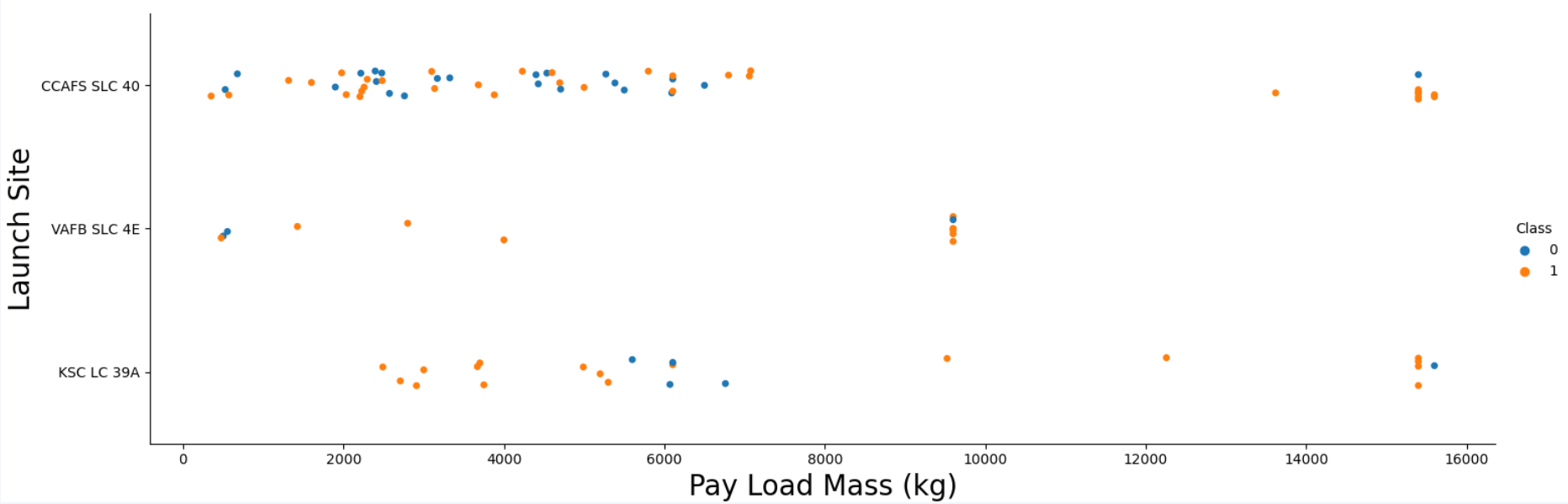

Section 2
Insights drawn from EDA

# Flight Number vs. Launch Site

- CCAFS SLC 40 is the most used site for Falcon 9 launches. VAFB  SLC 4E is the least used.
- The launch success rate tends to increase as the number of launches increases.



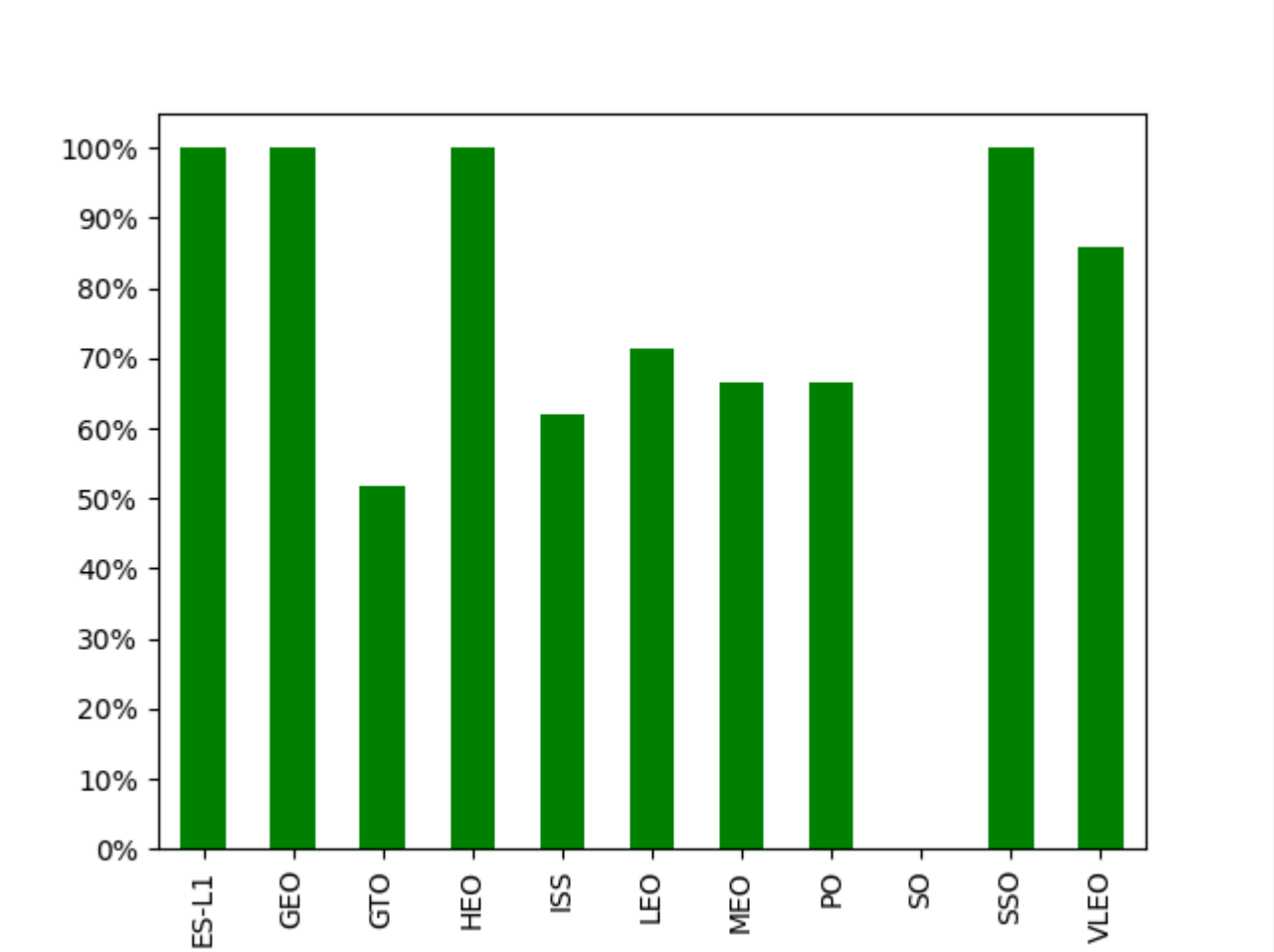
# Payload vs. Launch Site




- VAFB SLC 4E was only used for small or medium Payload mass, smaller than 10 tons. CCAFS SLC 40 and KSC LC 39A were used for small, medium and also Payload Mass up to 16 tons.
- At all three sites the success rate increases for payloads over 7 tons.

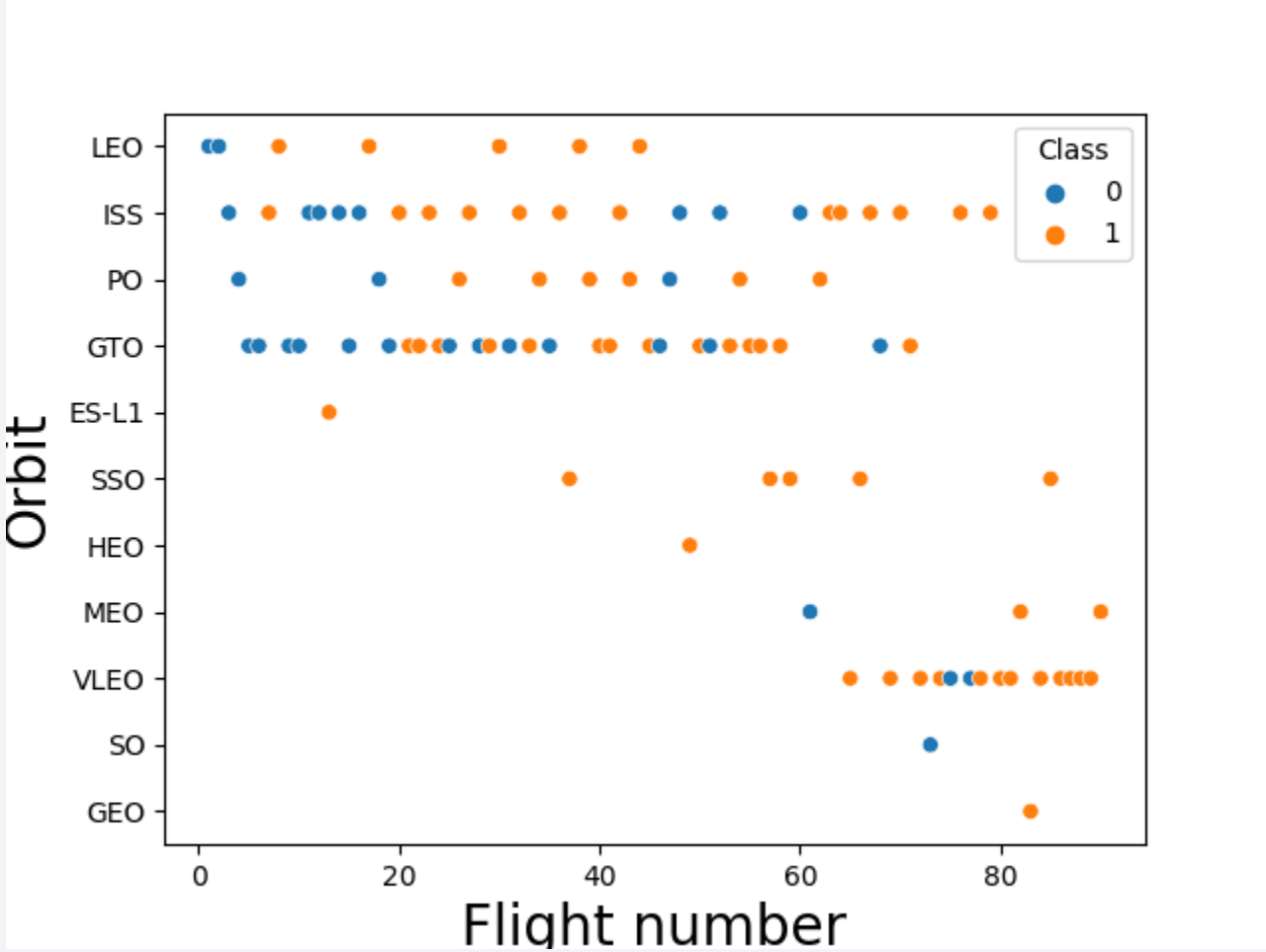
# Success Rate vs. Orbit Type

- All launches to ES-L1, GEO, HEO and SSO orbits were successful.
- Most launches to ISS, LEO, MEO, PO and VLEO orbits were successful.
- Half of launches to GTO orbit were successful.
- No launches to SO orbit were successful.



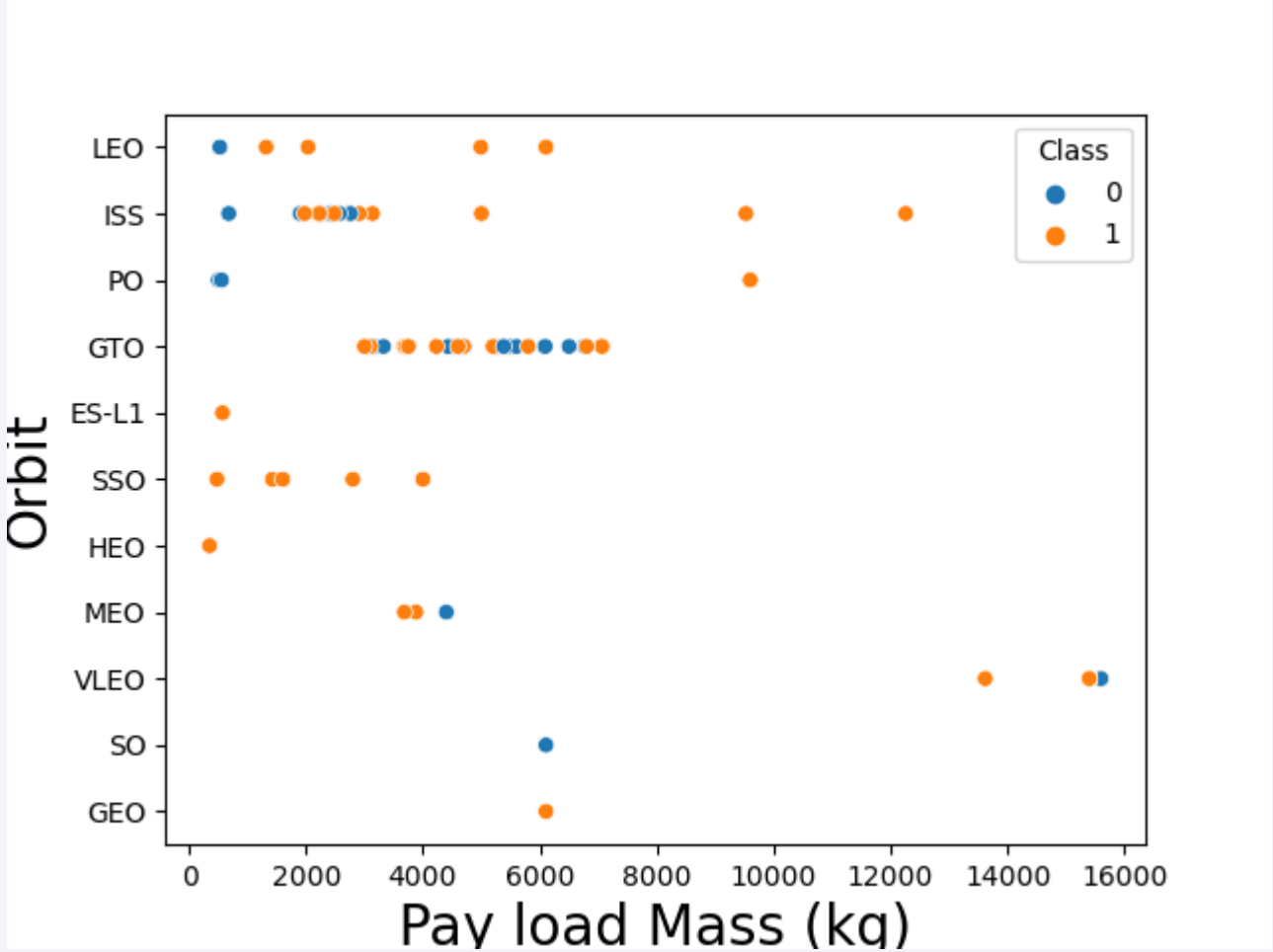
# Flight Number vs. Orbit Type

- For each of the ES-L1, HEO, SO and GEO orbits there was only one launch. The SO didn't have a good landing.
- In the first launches, the LEO, ISS, PO and GTO orbits were widely used. VLEO excels in the last few.



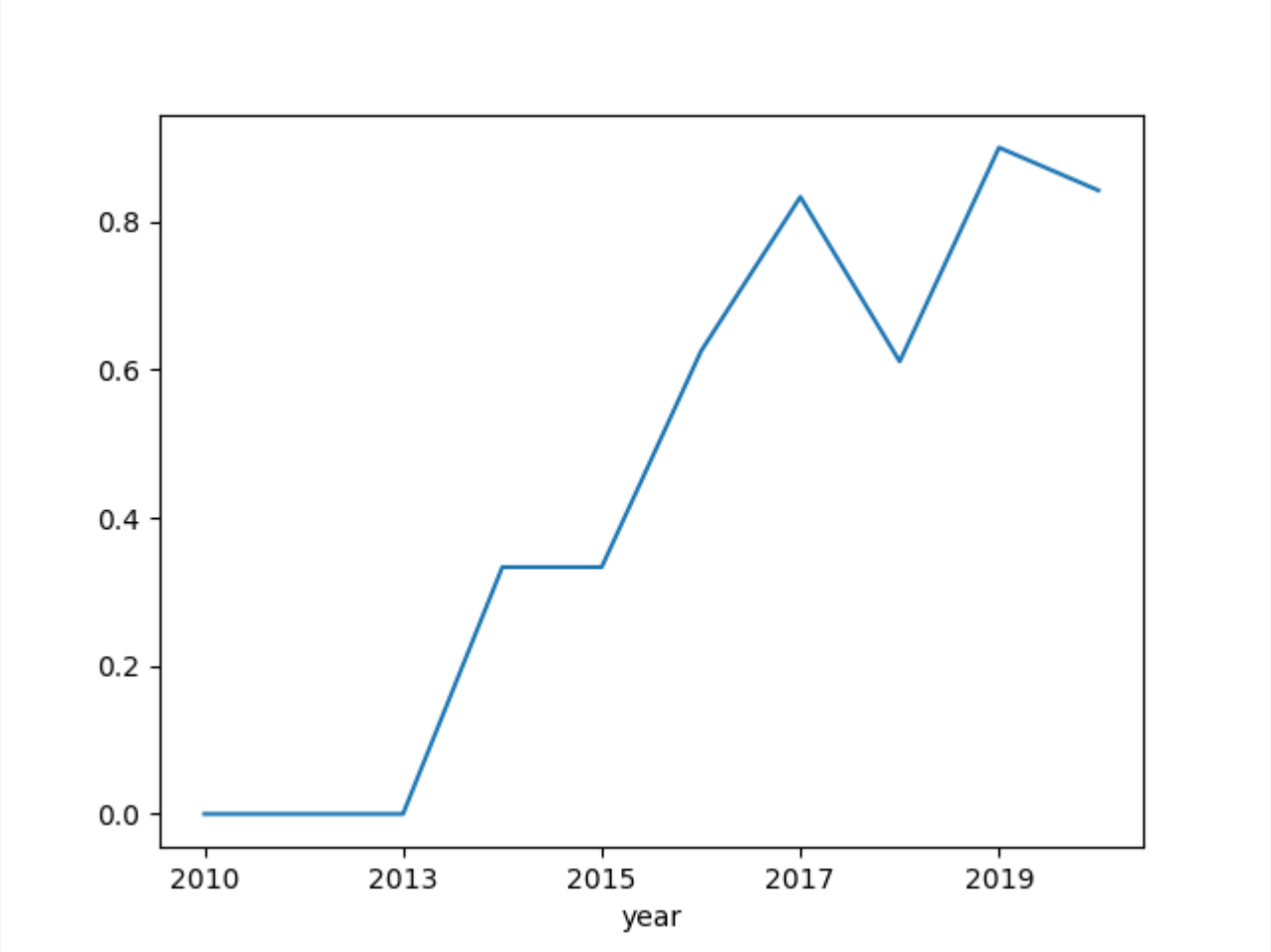
# Payload vs. Orbit Type

- For LEO, ISS and PO orbits, heavier payloads produce successful landings.
- For SSO orbit, light payloads produce successful landings.



# Launch Success Yearly Trend

- The landing success rate is zero until 2013.
- It starts to grow in 2014.
- It is above 80% in 2019 and 2020.



# All Launch Site Names

- The names of the launch sites are:
  - CCAFS LC-40
  - VAFB SLC-4E
  - KSC LC-39A
  - CCAFS SLC-40

```
%sql SELECT distinct launch_site FROM spacextbl;
```

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

| Launch_Site |
| --- |
| CCAFS LC-40 |
| None |
| VAFB SLC-4E |
| KSC LC-39A |
| CCAFS SLC-40 |

I used SELECT command with DISTINCT clause to get all site names without repetition.

# Launch Site Names Begin with 'CCA'

```
%sql SELECT * FROM SPACEXTBL WHERE launch_site LIKE 'CCA%' LIMIT 20;
```

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

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 04-06-2010 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0.0 | LEO | SpaceX | Success | Failure (parachute) |
| 22-05-2012 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525.0 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 08-10-2012 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500.0 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 01-03-2013 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677.0 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 03-12-2013 | 22:41:00 | F9 v1.1 | CCAFS LC-40 | SES-8 | 3170.0 | GTO | SES | Success | No attempt |
| 06-01- | 22:06:00 | F9 v1.1 | CCAFS LC- | Thaicom 6 | 3325.0 | GTO | Thaicom | Success | No attempt |

I used the SELECT command with the LIKE operator to get the launch sites names starting with 'CCA'

# Total Payload Mass

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer LIKE "NASA (CRS)";
 * sqlite:///my_data1.db
Done.
```

| SUM(PAYLOAD_MASS__KG_) |
|---|
| 38856.0 |

I used the SELECT command with the SUM operator on the column I wanted to total. In order to restrict the sum to the 'NASA' client, I used the LIKE operator.

# Average Payload Mass by F9 v1.1

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version LIKE "F9 v1.1";
```

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

| AVG(PAYLOAD_MASS__KG_) |
|---|
| 2928.4 |

I used the SELECT command with the AVG operator on the column I wanted to calculate the average. To restrict the calculation to records from "F9 v1.1" I used the WHERE clause and the LIKE operator.

# First Successful Ground Landing Date

```
%sql SELECT MIN(Date) FROM SPACEXTBL WHERE Landing_Outcome = "Success (ground pad)";
```

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

| MIN(Date) |
| --- |
| 01-05-2017 |

I used the SELECT command with the MIN operator on the column I wanted to find the minimum value. To restrict the search to "Success" records I used the WHERE clause.

# Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql SELECT * FROM SPACEXTBL WHERE Landing_Outcome LIKE "Success (drone ship)" AND PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000;
```

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

| Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| 06-05-2016 | 05:21:00 | F9 FT B1022 | CCAFS LC-40 | JCSAT-14 | 4696.0 | GTO | SKY Perfect JSAT Group | Success | Success (drone ship) |
| 14-08-2016 | 05:26:00 | F9 FT B1026 | CCAFS LC-40 | JCSAT-16 | 4600.0 | GTO | SKY Perfect JSAT Group | Success | Success (drone ship) |
| 30-03-2017 | 22:27:00 | F9 FT B1021.2 | KSC LC-39A | SES-10 | 5300.0 | GTO | SES | Success | Success (drone ship) |
| 11-10-2017 | 22:53:00 | F9 FT B1031.2 | KSC LC-39A | SES-11 / EchoStar 105 | 5200.0 | GTO | SES EchoStar | Success | Success (drone ship) |

I used the SELECT command with the WHERE clause and the LIKE and AND operators to specify the 'Success' records and the payload mass range.

# Total Number of Successful and Failure Mission Outcomes

```
%%sql
SELECT "success" as "Outcome", COUNT(*) AS "Total_Number" FROM SPACEXTBL WHERE Landing_Outcome LIKE 'Succes%'
UNION
SELECT "failure" as "Outcome", COUNT(*) AS "Total_Number" FROM SPACEXTBL WHERE Landing_Outcome NOT LIKE 'Succes%';
```

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

| Outcome | Total_Number |
|---|---|
| failure | 34 |
| success | 32 |

I used the SELECT command with the COUNT operator to count the records and then the WHERE clause and the LIKE operator so that the count was only the successful records. Then I repeated the form of the command changing only the result to failure. After that I used the UNION operator between the commands to present the results in record form.

# Boosters Carried Maximum Payload

```
SELECT DISTINCT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM S
 * sqlite:///my_data1.db
Done.
```

| Booster_Version | PAYLOAD_MASS__KG_ |
|---|---|
| F9 FT B1029.1 | 9600.0 |
| F9 FT B1036.1 | 9600.0 |
| F9 B4 B1041.1 | 9600.0 |
| F9 FT B1036.2 | 9600.0 |
| F9 B4 B1041.2 | 9600.0 |
| F9 B5B1048.1 | 9600.0 |
| F9 B5 B1049.2 | 9600.0 |

I used the SELECT statement with the DISTINCT statement to not repeat boosters, and the WHERE clause to restrict it to the maximum registered payload. To get the maximum payload value I compared, within the mentioned WHERE clause, the payload value with the result of another query. In this other query I used SELECT with the MAX operator to find the maximum payload of all records.

# 2015 Launch Records

```
%%sql
SELECT
    CASE
        WHEN SUBSTR(Date, 4, 2) = '01' THEN 'January'
        WHEN SUBSTR(Date, 4, 2) = '02' THEN 'February'
        WHEN SUBSTR(Date, 4, 2) = '03' THEN 'March'
        WHEN SUBSTR(Date, 4, 2) = '04' THEN 'April'
        WHEN SUBSTR(Date, 4, 2) = '05' THEN 'May'
        WHEN SUBSTR(Date, 4, 2) = '06' THEN 'Jun'
        WHEN SUBSTR(Date, 4, 2) = '07' THEN 'July'
        WHEN SUBSTR(Date, 4, 2) = '08' THEN 'August'
        WHEN SUBSTR(Date, 4, 2) = '09' THEN 'September'
        WHEN SUBSTR(Date, 4, 2) = '10' THEN 'October'
        WHEN SUBSTR(Date, 4, 2) = '11' THEN 'November'
        WHEN SUBSTR(Date, 4, 2) = '12' THEN 'December'
    END AS Month,
    COUNT(CASE WHEN Landing_Outcome = 'Failure (drone ship)' THEN 1 ELSE NULL END) AS Failure_Count,
    Booster_Version, Launch_Site
FROM SPACEXTBL
WHERE
    SUBSTR(Date, 7, 4) = '2015'
GROUP BY
    Month, Booster_Version, Launch_Site
ORDER BY
    CASE
        WHEN Month = 'January' THEN 1
        WHEN Month = 'February' THEN 2
        WHEN Month = 'March' THEN 3
        WHEN Month = 'April' THEN 4
        WHEN Month = 'May' THEN 5
        WHEN Month = 'Jun' THEN 6
        WHEN Month = 'July' THEN 7
        WHEN Month = 'August' THEN 8
        WHEN Month = 'September' THEN 9
        WHEN Month = 'October' THEN 10
        WHEN Month = 'November' THEN 11
        WHEN Month = 'December' THEN 12
    END, Failure_Count;
```

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

| Month | Failure_Count | Booster_Version | Launch_Site |
|---|---|---|---|
| January | 1 | F9 v1.1 B1012 | CCAFS LC-40 |
| February | 0 | F9 v1.1 B1013 | CCAFS LC-40 |
| March | 0 | F9 v1.1 B1014 | CCAFS LC-40 |
| April | 0 | F9 v1.1 B1016 | CCAFS LC-40 |
| April | 1 | F9 v1.1 B1015 | CCAFS LC-40 |
| Jun | 0 | F9 v1.1 B1018 | CCAFS LC-40 |
| December | 0 | F9 FT B1019 | CCAFS LC-40 |

I used SELECT CASE to list the month names according to the month number. I used COUNT with CASE to count the number of failures each month. I used the WHERE clause to restrict to the year 2015. I used GROUP BY to group by month, booster and launch site. Finally, I used the combination of ORDER BY with CASE so that the order was by the month number and not by the month name (otherwise April would be the first in the list).

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

```
%%sql

SELECT Count(*) as Count FROM SPACEXTBL WHERE Landing_Outcome LIKE "Succes%" AND Date between '04-06-2010' AND '20-03-2017';
```

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

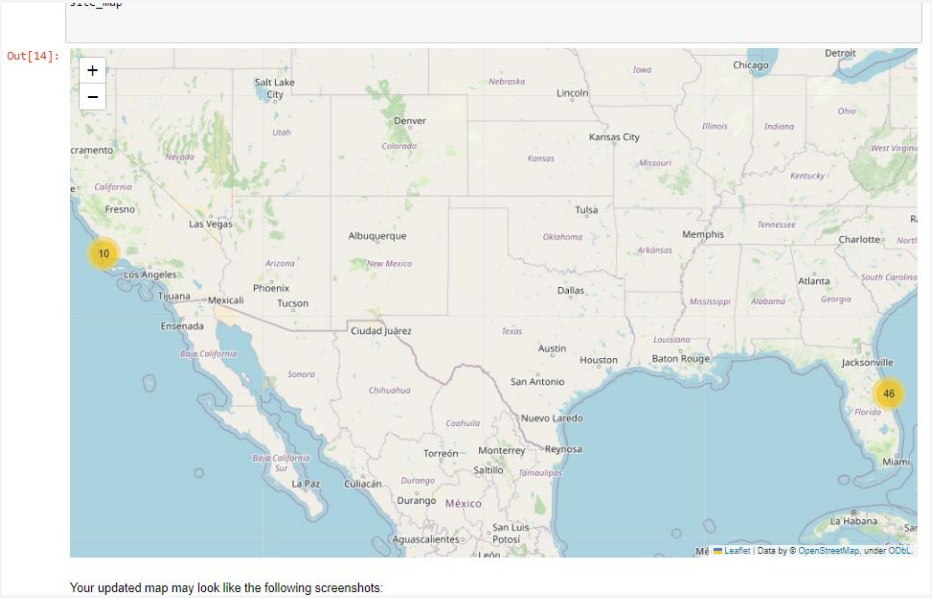
| Count |
|---|
| 20 |

I used SELECT COUNT to count the successful records and I used LIKE combined with AND to define the query range.

Section 3

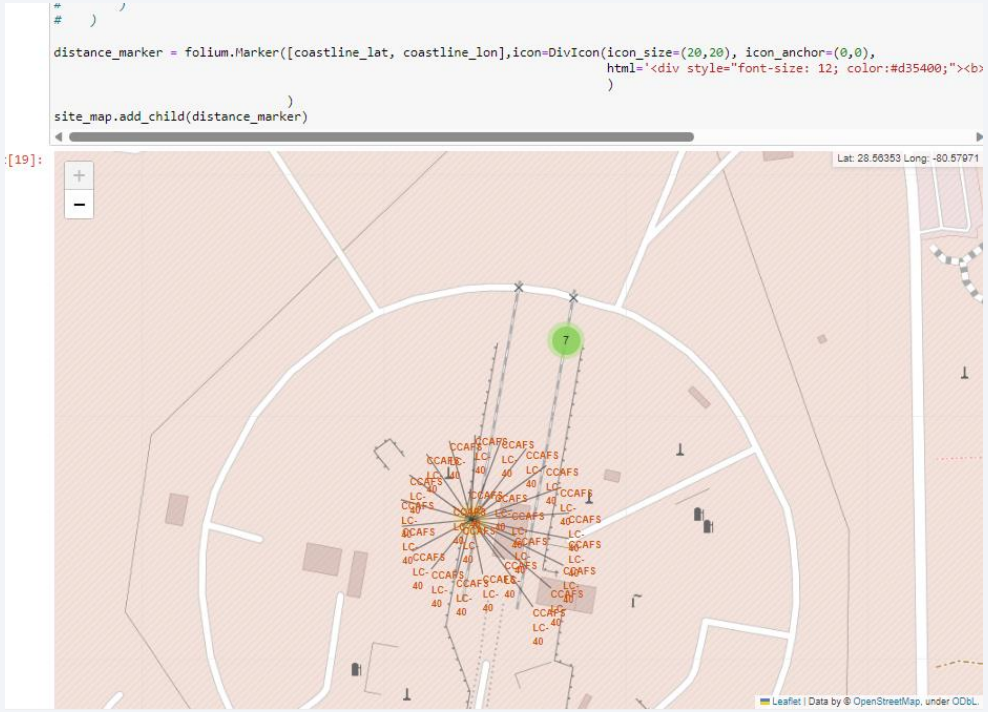
# Launch Sites Proximities Analysis

# Launch Sites Map




Rocket launch sites are located off the coast of Florida and the coast of California, inside United States.

# CCAFS LC-40 Launch Site Map

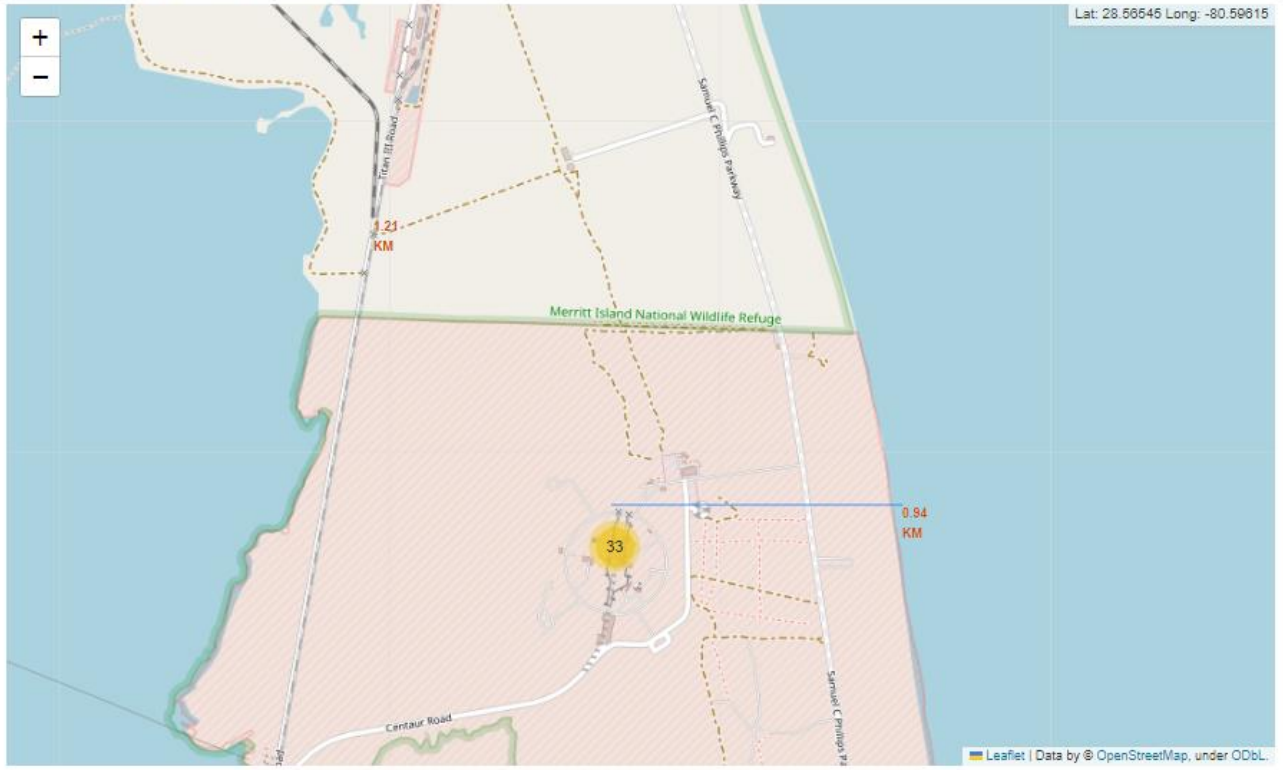


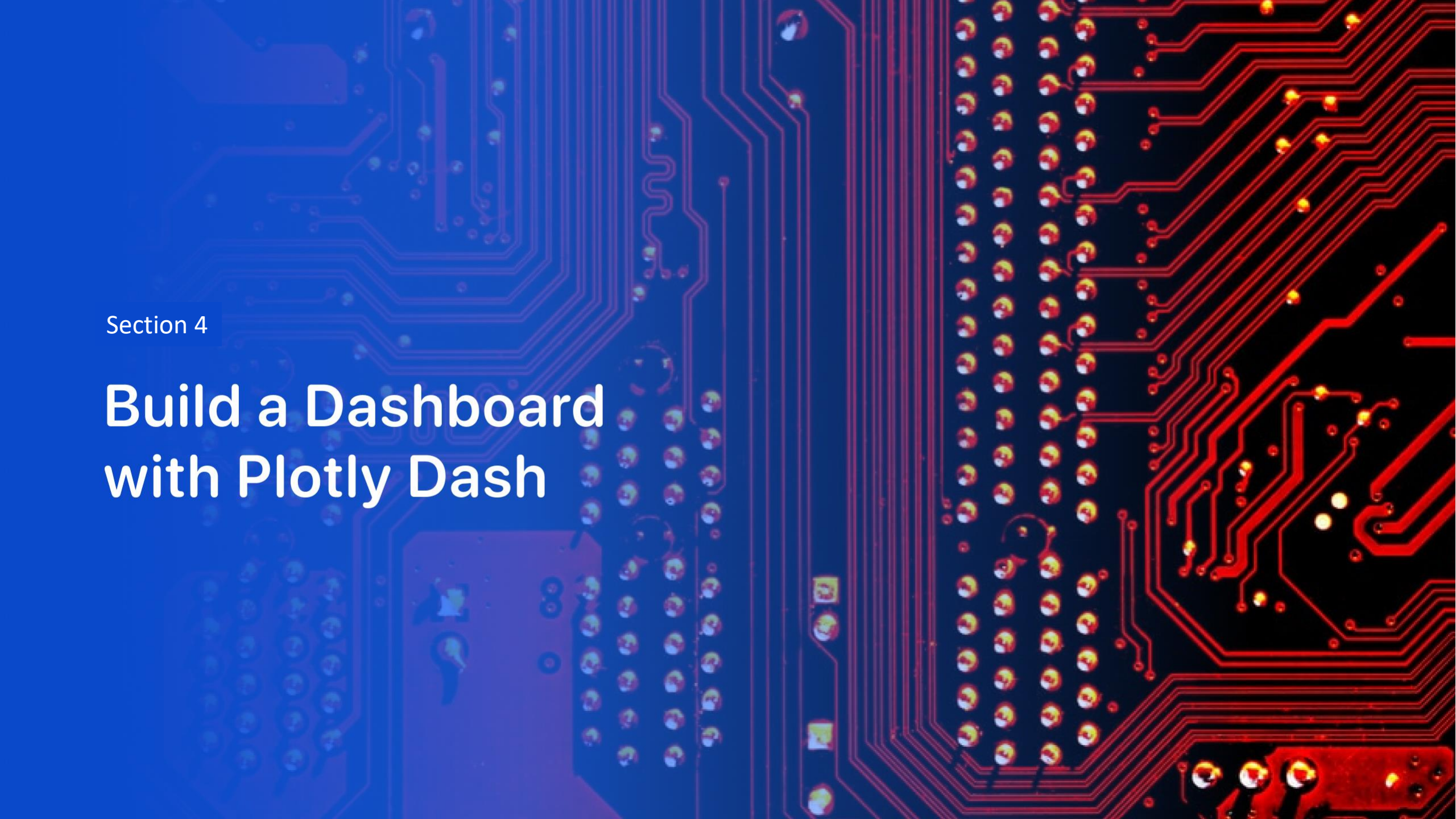

CCAFS LC-40 site launches shown in red

# CCAFS LC-40 distances Map

- Distance from dite launch to coast and to a railroad are sown in red
- A blue line graphically shows the distance from the site to the coast.

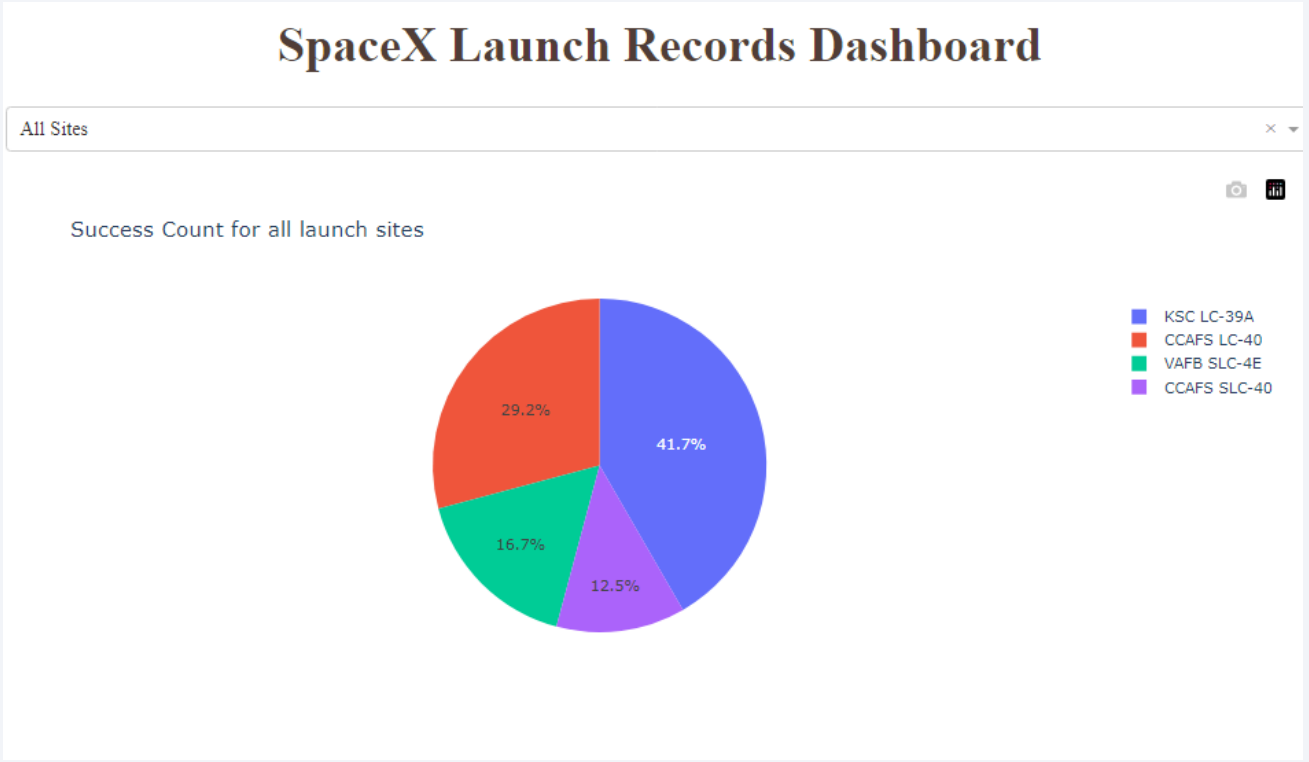
```
distance_marker = folium.Marker(
   [railroad_lat, railroad_lon],
   icon=DivIcon(
       icon_size=(20,20),
       icon_anchor=(0,0),
       html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % "{:10.2f} KM".format(distance_railroad),
       )
   )
site_map.add_child(distance_marker)
```




Section 4
Build a Dashboard with Plotly Dash

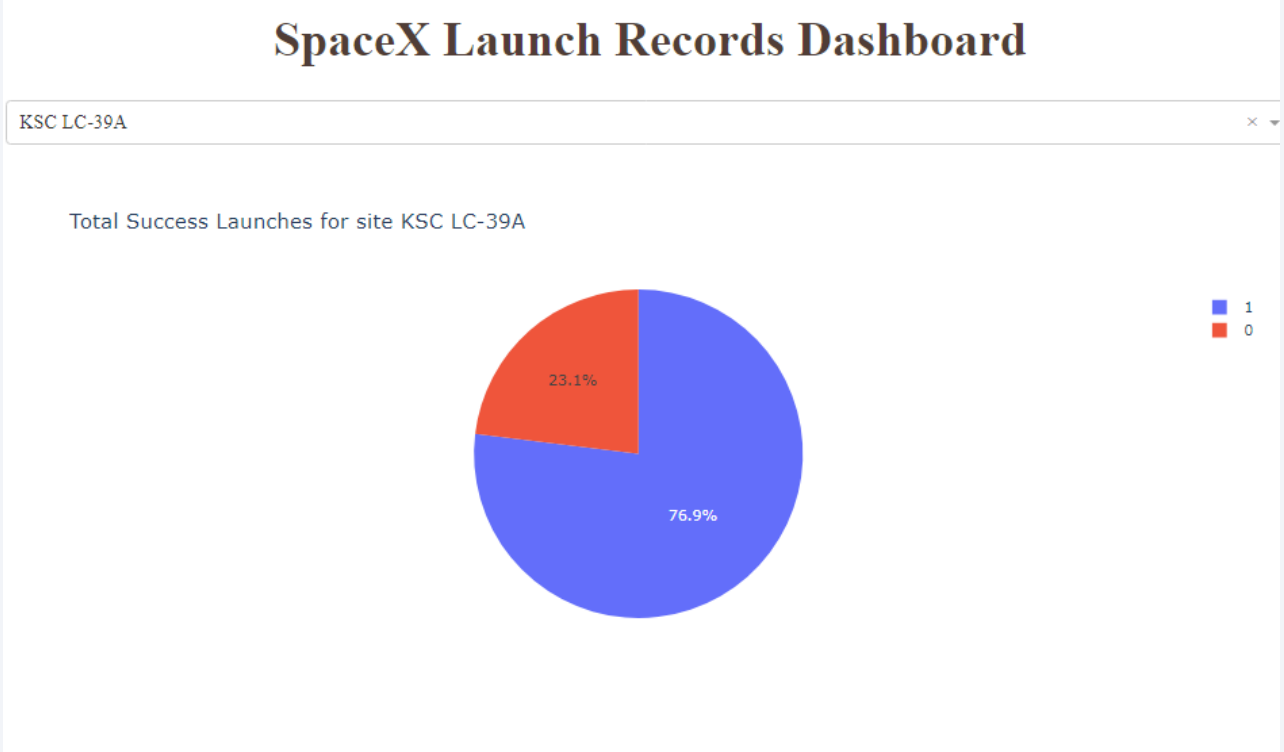
# Pie Chart – Success rate – All sites

- The pie chart shows the success rate for all launches sites.



# <Dashboard Screenshot 2>

- The pie chart shows the screenshot of the piechart KSC LC-39A because it is the most successful in launches and landings



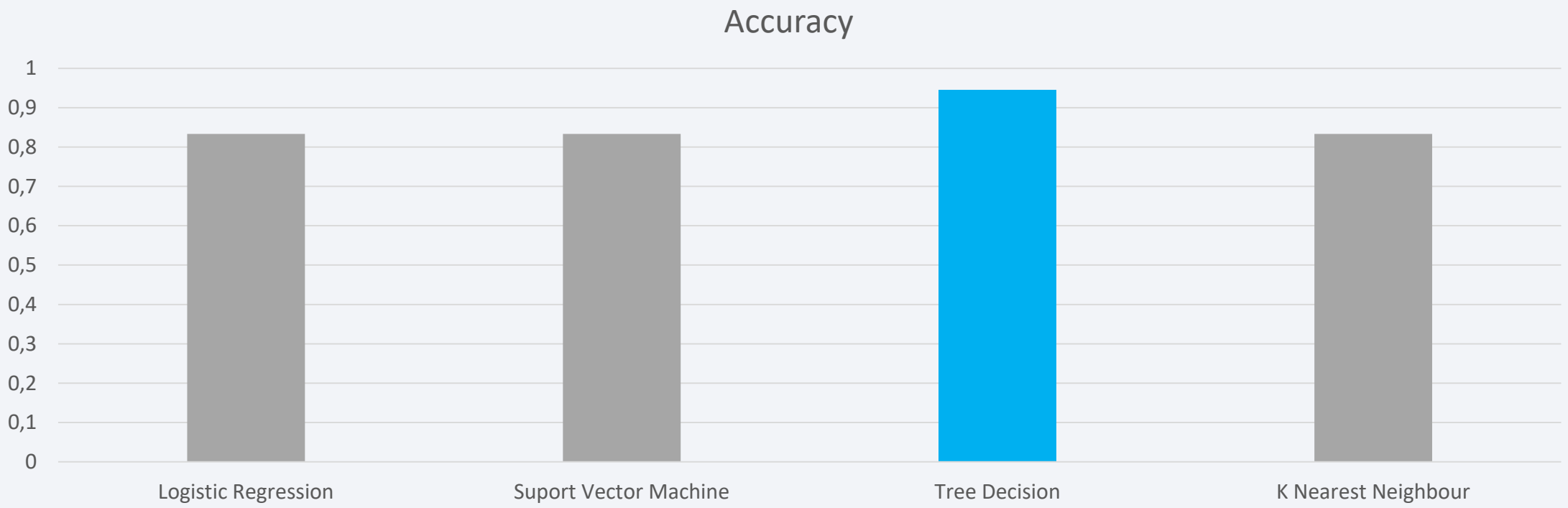
# Scatter chart – Payload vs Launch Outcome


- The charts show success and failures of launches by Pay Load Mass.
- The first graph considers payloads below 5 tons.
- The second graph considers payloads above 5 tons.

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy




Tree decision is the model with best accuracy: 94%.

# Confusion Matrix

- The confusion matrix of the Decision Tree model shows 17 hits in a test set of 18 records, which gives an accuracy of 94.4%.
- There were no false positives. There was 1 false negative.

# Conclusions

- ES-L1, GEO, HEO and SSO orbits favor mission success.
- Over time, the trend is for the success rate to approach 100%.
- The best model found is the Decision Tree.
- Still, because the test set is small, new analyzes in the future may come to different conclusions about the best model.

# Thank you!