IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Marcos Silveira
11/20/2022

# Outline

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

# Executive Summary

- Summary of methodologies
  - Data collection via SpaceX API and Web Scraping;
  - Exploratory Data Analysis (EDA) and Data Visualization with Matplotlib and Seaborn;
  - EDA with SQL;
  - Interactive Folium map;
  - Predictive Analysis with Sklearn;
- Summary of all results
  - Interactive map and static plots;
  - Predictive results with confusion matrix;

# Introduction

- Project background and context:
  - SpaceX is a multibillion dollar company that does research in space exploration. The first step of their project is to be able to reuse rockets, considering their 62 million dollars price tag. Considering this, there is a massive interest in predicting the result of a launch before performing it.
- Problems you want to find answers:
  - What are the main characteristics of a successful launch?
  - How does each data relate to each other?
  - What conditions guarantee a successful launch?

Section 1

# Methodology

# Methodology

## Executive Summary

- Data collection methodology:
  - SpaceX API, Web Scraping.
- Perform data wrangling:
  - Filtered Data;
  - Transform categorical variables into continuous;
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium
- Perform predictive analysis using classification models
  - Models used: Logistic Regression, KNNeighbours, SVM and Decision Tree.

# Data Collection

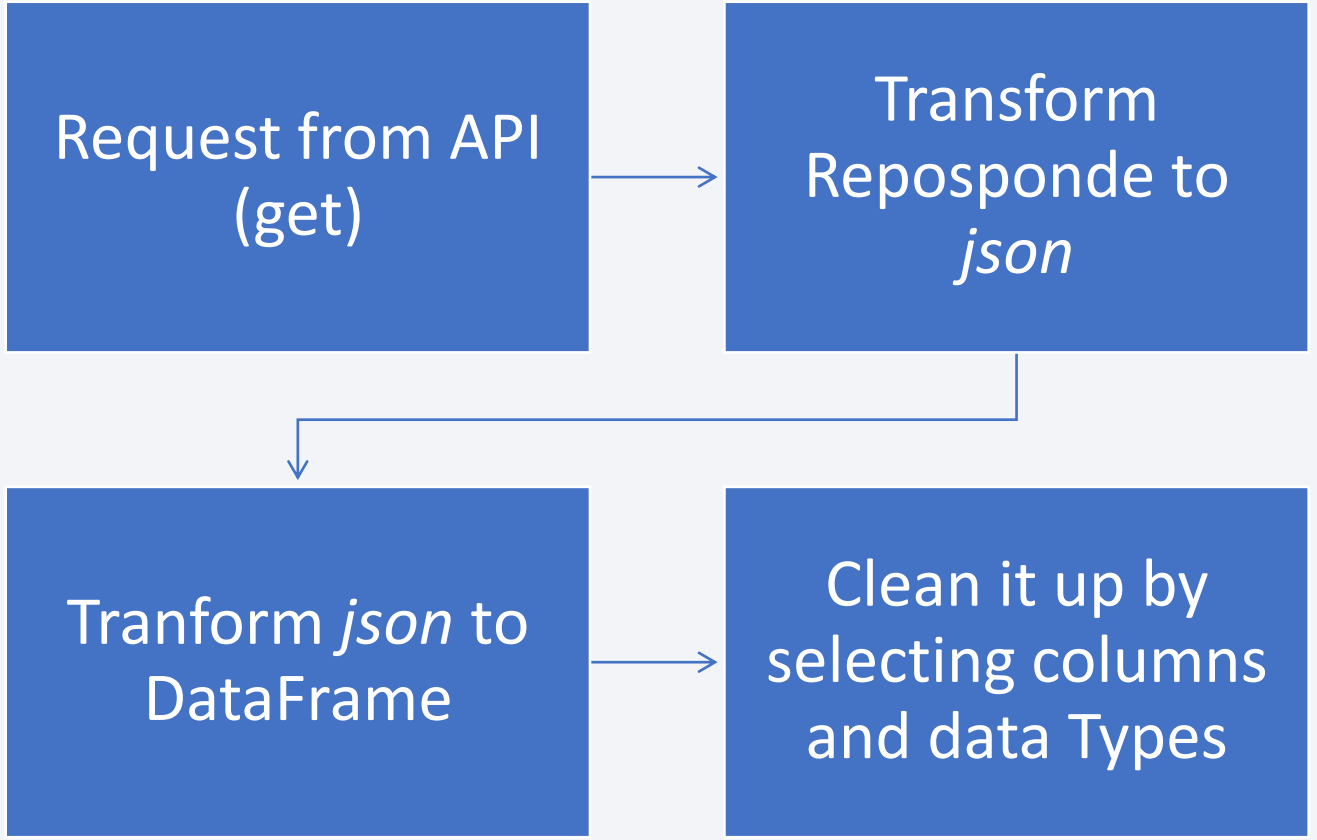
Data was collected using 2 (two) methods:

- Using a static *json* response. Utilizing the SpaceX Rest API call in order to receive the data in a *json* format.

- Using *webscraping* with the Wikipedia entry of SpaceX. Utilizing the *html* response of the website and exploring said data with BeatifulSoup.

# Data Collection – SpaceX API

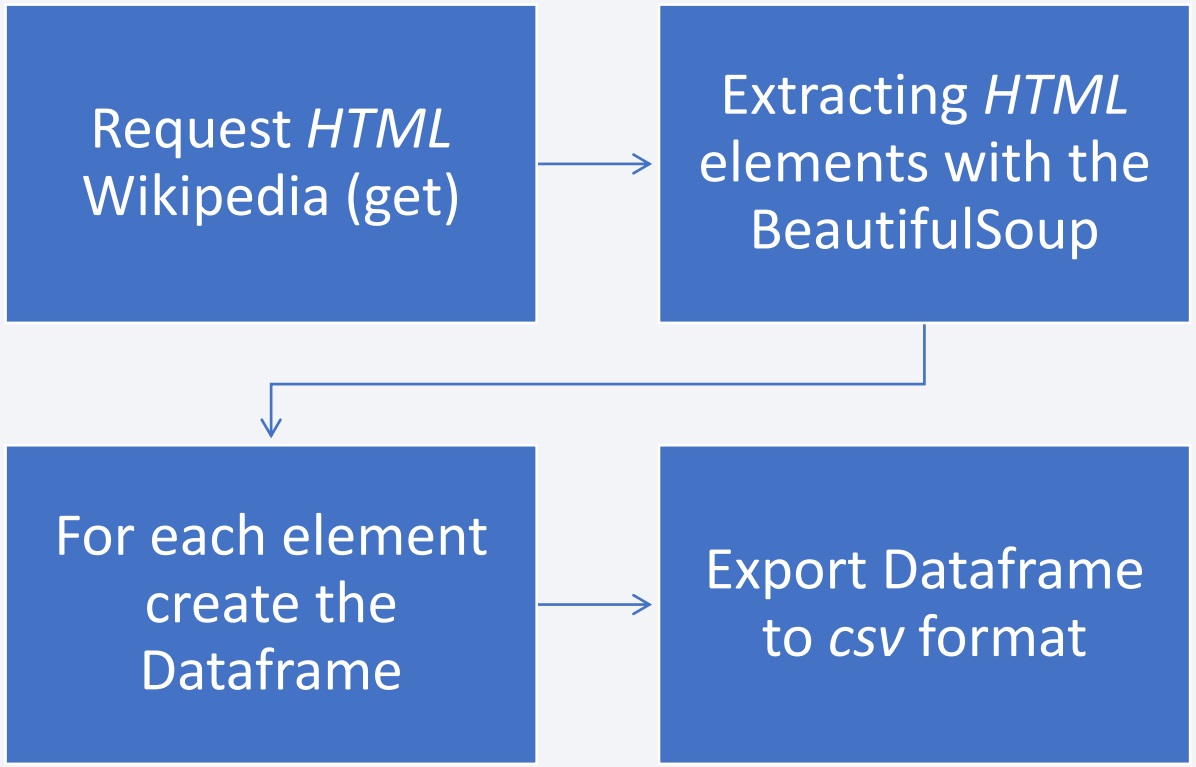
- Utilizing *Requests* library and the *Dataframe* library we where able to extract and clean the Data sent by the SpaceX API




GitHub Repository

# Data Collection - Scraping

- Web Scraping was done with *Request, Dataframe* and *BeautifulSoup* libraries. By first extracting the *html* from the Wikipedia repositor for SpaceX, we Where able to clean the text and extract Only the table.




GitHub Repository

# Data Wrangling

- The SpaceX data was processed by first adding a new parameter to each landing data, the ‘Class’. Which is 1 if the landing was successful and 0 if the landing was not unsuccessful. In order to do this was looked on the ‘Outcome’ variable, which states True if it was successful and False if it was unsuccessful.
- In order to do this step we followed these steps:

Extract Outcomes

By checking every type of Outcome we categorized which were successful and which where not.

Change Values

For each value in the column ‘Outcome’ we changed its categorical value to its contínuos conterpart.

Creating Column and Saving

For each new continuous value we added its number to a new column ‘Class’ and saved the result in a *csv*.

GitHub Repository

# EDA with Data Visualization

## The graphs presented in this report where:

- Scatter plots:
  - Flight Number X Payload Mass
  - Flight Number X Launch Site
  - Payload Mass X Launch Site
  - Flight Number X Orbit Type
  - Payload Mass X Orbit Type
- Bar Graph:
  - Success rate for each Orbit Type.
- Line Plot:
  - Launch success by year.

GitHub Repository

# EDA with SQL

- SQL Queries performed in this report where:
  - Unique launch sites;
  - 5 records with launch sites beginning with CCA;
  - Total mass caried by booster launched by NASA (CRS);
  - Average payload carried by Boosters Version F9 v1.1;
  - Date of first successful landing in Ground Pad;
  - Names of boosters witch had successful landing on drone ships with mass between 4000 and 6000;
  - Total number of successful and failure missions outcome;
  - Names of boosters which have carried maximum mass;
  - Information of launches in 2015;
  - Count of successful landings between 04-06-2010 and 20-03-2017;

GitHub Repository

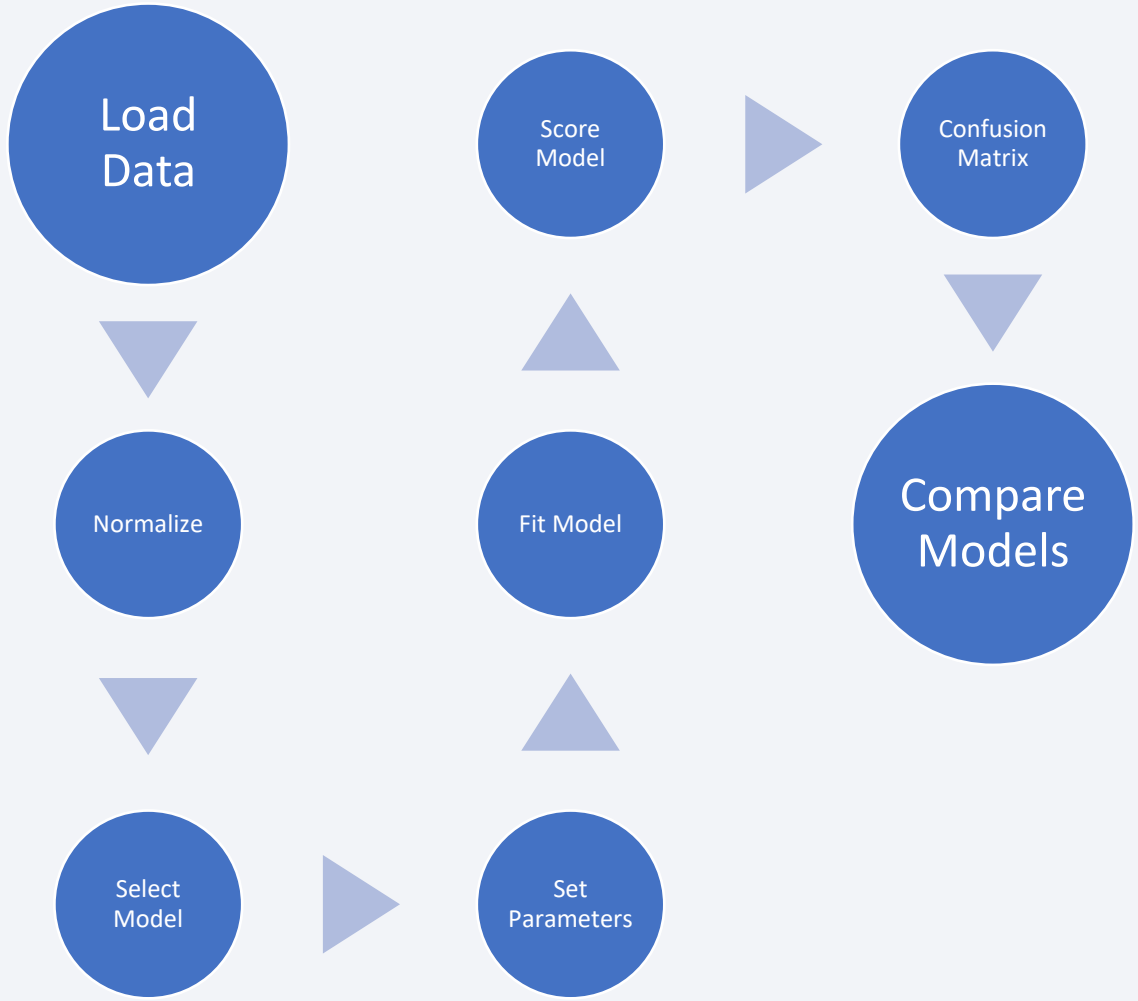
# Build an Interactive Map with Folium

- To create a interactive map with Folium, different objects where created:
  - Markers;
  - Circles;
  - Polyline.
- These where created to mark each launch site and landing sites for each launch, as well as distance between them.

GitHub Repository

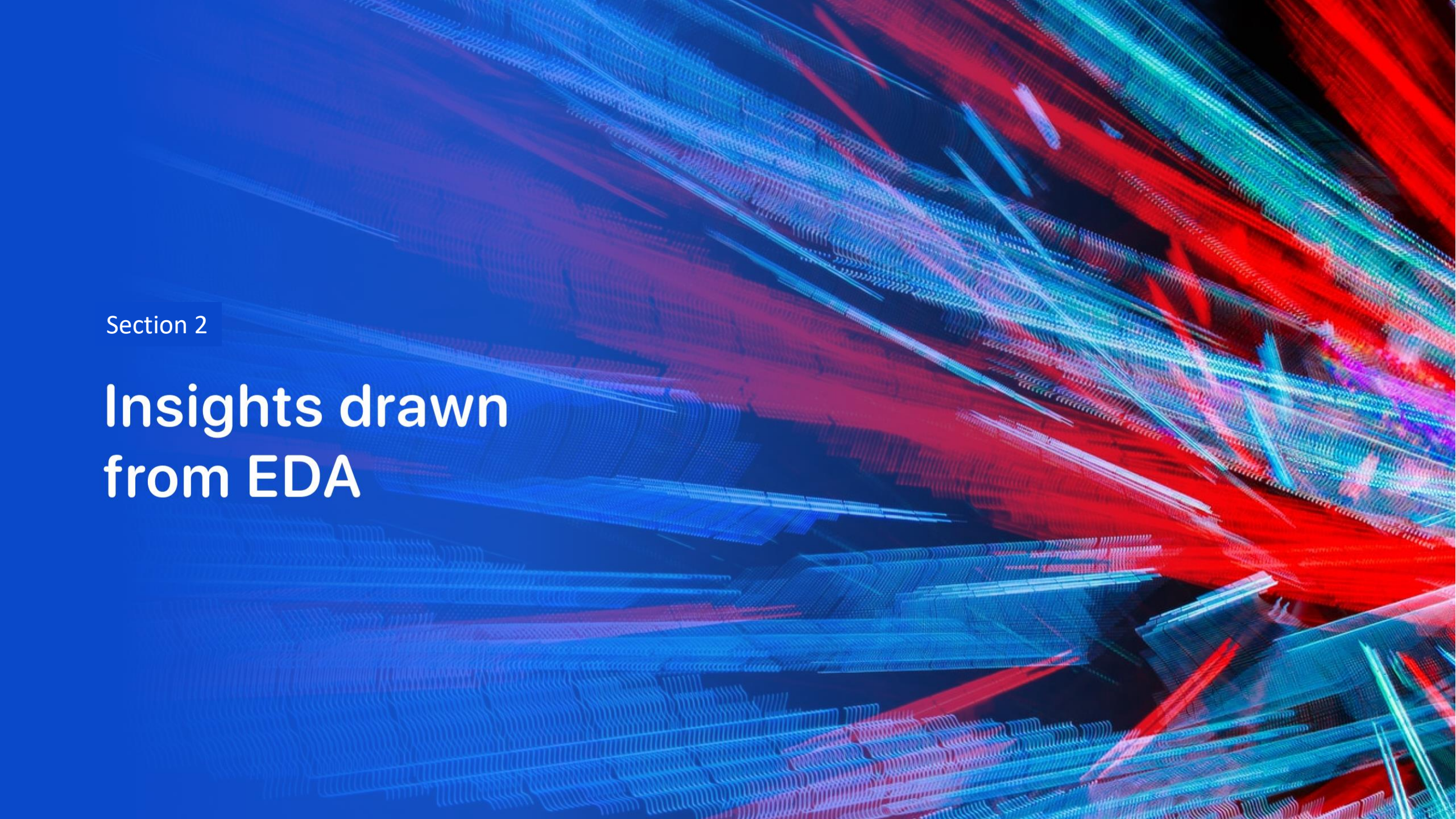
# Predictive Analysis (Classification)

- Data Preparation:
  - Load csv to DataFrame;
  - Normalize;
  - Split data into Training and Test data;
- Model Preparation:
  - Select models from Sklearn:
    - Logistic Regression;
    - SVC;
    - Decision Tree;
    - KNNeighbors;
  - Set parameters;
  - Fit model in GridSearchCV;
- Model Evaluation:
  - Score model based on the best result;
  - Create confusion matrix on its prediction;

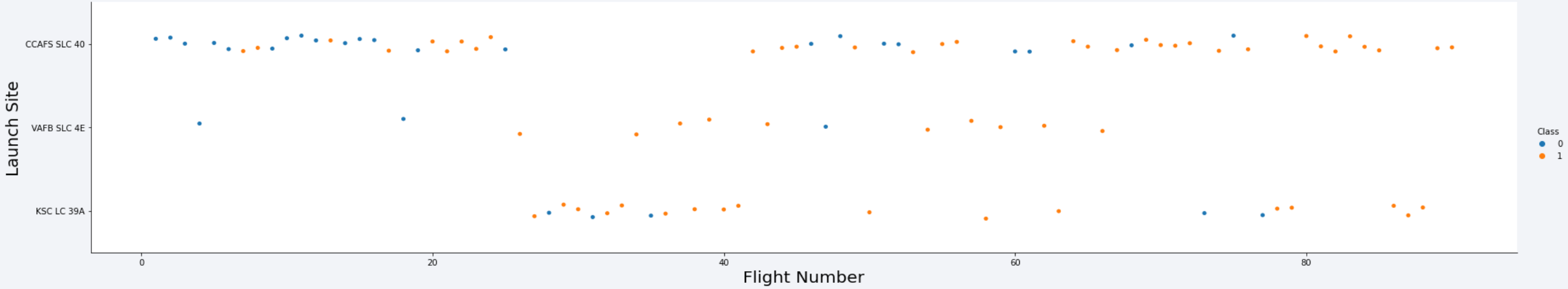


# Results

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

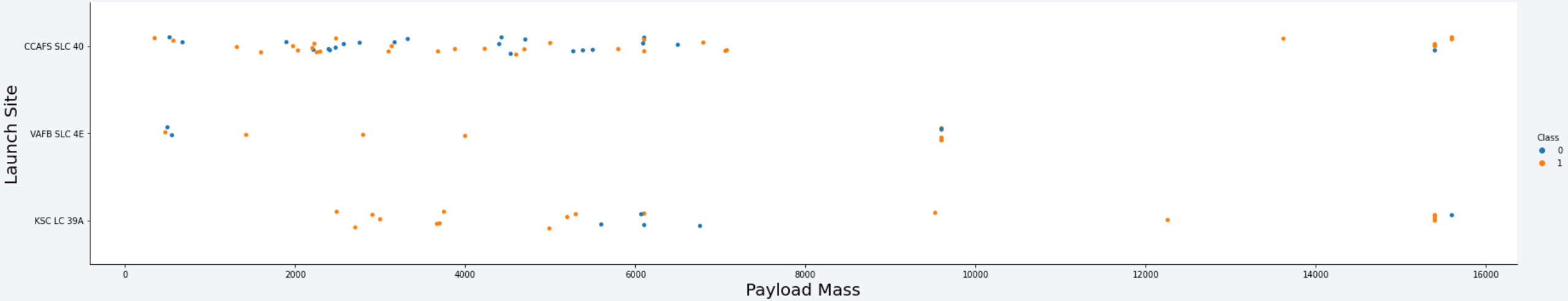

Section 2
Insights drawn from EDA

# Flight Number vs. Launch Site




- Correlation between flight number and its outcome, with every early flight where shown to be unsuccessful, showing that with experience more good results where shown.

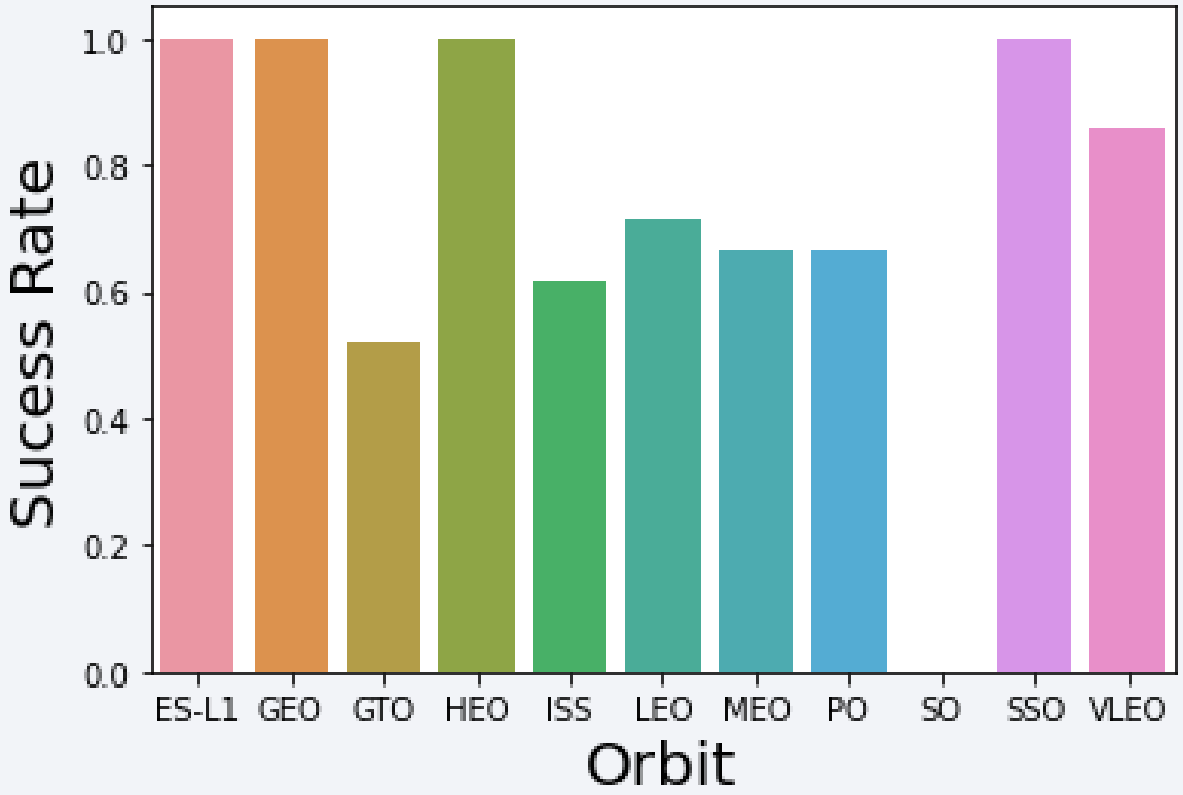
# Payload vs. Launch Site



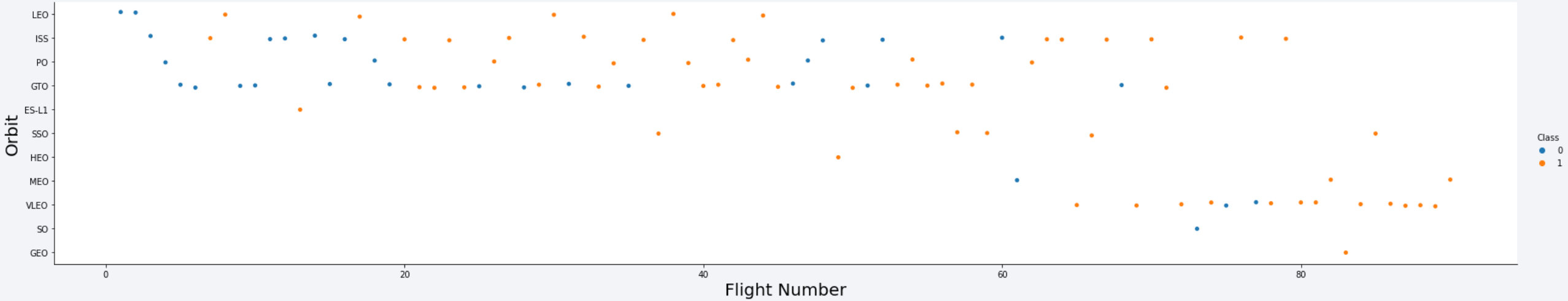

- VAFB SLC 4E has no heavy launches if compared to the other launch sites.
- No other correlations where found.

# Success Rate vs. Orbit Type

- ES-L1, GEO, HEO and SSO have the heighest success rates.
- SO showed no successful launches.

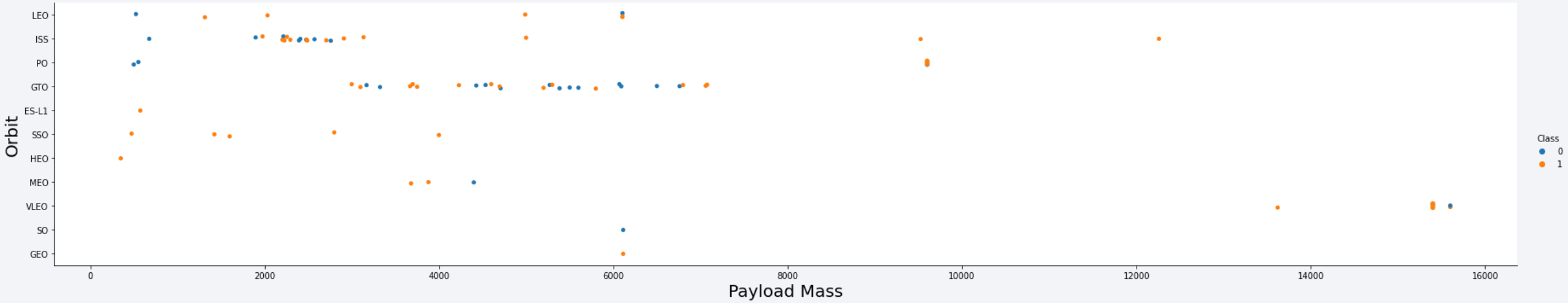


# Flight Number vs. Orbit Type




- We can see a correlation between flight number and its success rates for each orbit type.

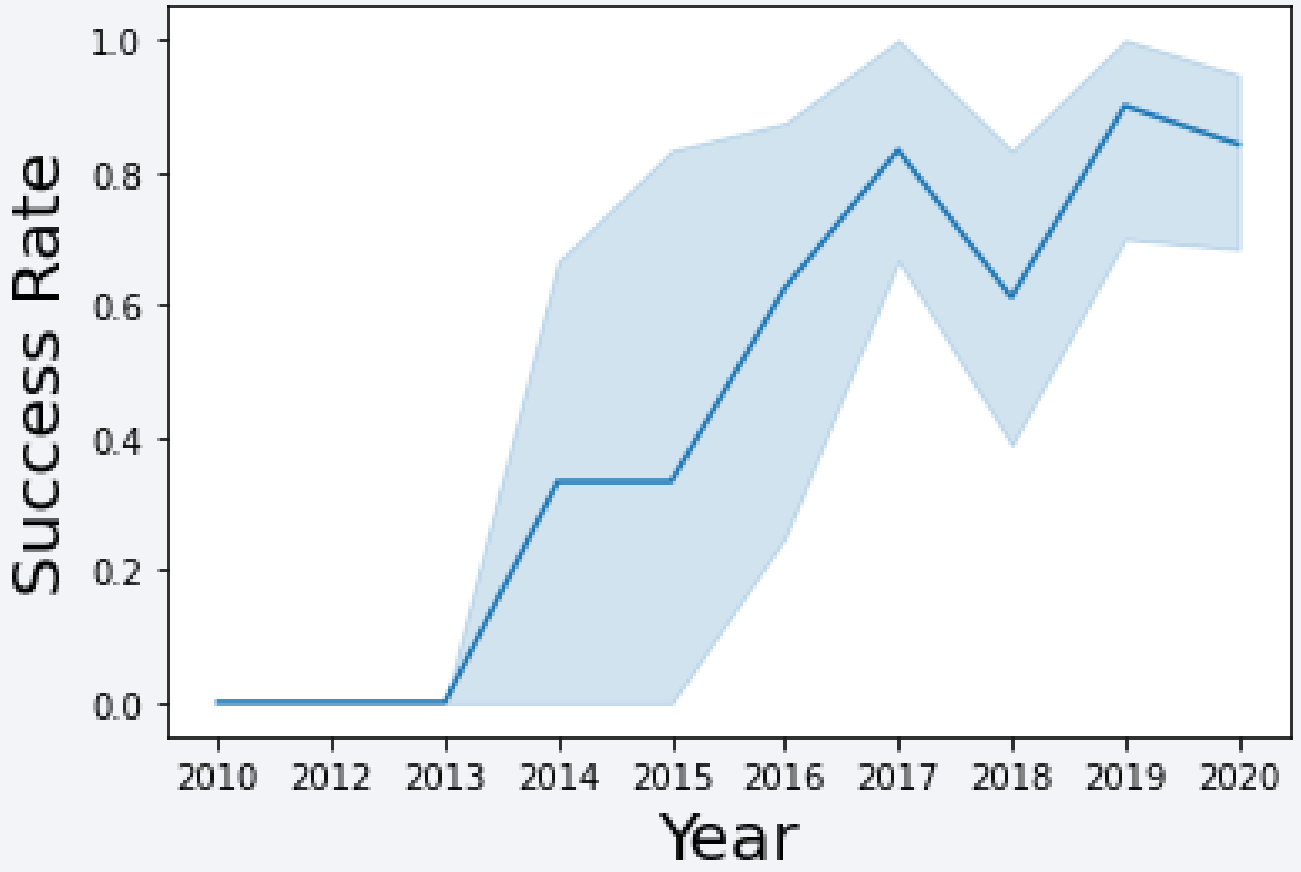
# Payload vs. Orbit Type




- We can see that VLEO is more dedicated to heavier payloads and GTO has a variety of masses.
- There seems to be no correlation between paylaod mass and success rate.

# Launch Success Yearly Trend

- As the years go by we can see a trend upwards to the launches success rate.



# All Launch Site Names

- SQL Query:

```
1  %sql select distinct Launch_Site from SPACEXTBL
```

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

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

- The distinct key brings only unique values from column Launch_Site from the table SPACEXTBL.

# Launch Site Names Begin with 'CCA'

- SQL Query:

```
1  %sql select * from SPACEXTBL where Launch_Site like 'CCA%' limit 5
```

Python

```
 * 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 | LEO | SpaceX | Success | Failure (parachute) |
| 08-12-2010 | 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) |
| 22-05-2012 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 08-10-2012 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 01-03-2013 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

- Utilizing the SQL wildcard '%' to bring only values that start with 'CCA'.

# Total Payload Mass

- SQL Query:

```
1  %sql select sum(PAYLOAD_MASS__KG_) as SUM from SPACEXTBL where Customer = 'NASA (CRS)'
```

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

| SUM |
| --- |
| 45596 |

- The sum SQL function sums every value from the column PAYLOAD_MASS__KG_, and the where clause states which rows to sum from.

# Average Payload Mass by F9 v1.1

- SQL Query:

```
1  %sql select avg(PAYLOAD_MASS__KG_) as AVG from SPACEXTBL where Booster_Version like 'F9 v1.1'
```

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

| AVG |
| --- |
| 2928.4 |

- The avg SQL function averages every value from the column PAYLOAD_MASS__KG_, and the where clause states which rows to sum from.

# First Successful Ground Landing Date

- SQL Query:

```
1  %sql SELECT min(Date) from SPACEXTBL where [Landing_Outcome] = 'Success (ground pad)'
```

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

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

- The min SQL function brings the minimal value of the column Date.

# Successful Drone Ship Landing with Payload between 4000 and 6000

- SQL Query:

```
1  %sql select Booster_Version from SPACEXTBL where [Landing_Outcome] = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
```

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

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

- The and statement makes sure that all values in the where clause are true.

# Total Number of Successful and Failure Mission Outcomes

- SQL Query:

```
1  %sql select Mission_Outcome, count(date) as count from SPACEXTBL group by Mission_Outcome
```

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

| Mission_Outcome | count |
|---|---|
| Failure (in flight) | 1 |
| Success | 98 |
| Success | 1 |
| Success (payload status unclear) | 1 |

- The count SQL function counts every occurrence and the group by statement groups every row by a column, in this case the Mission_Outcome.

# Boosters Carried Maximum Payload

- SQL Query:

```
1  %sql select Booster_Version from SPACEXTBL where PAYLOAD_MASS__KG_ in (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

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

| Booster_Version |
|---|
| 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 |

- The in statement defines a condition that the value must be within a collection of values, the collection in this case is the maximum payload mass in the subquery.

# 2015 Launch Records

- SQL Query:

```
%%sql select
    substr(Date, 4, 2) as Month,
    [Landing _Outcome],
    Booster_Version,
    Launch_Site
from SPACEXTBL
where
    substr(Date, 7, 4) = '2015'
    and [Landing _Outcome] like '%drone ship%'
```

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

| Month | Landing _Outcome | Booster_Version | Launch_Site |
|---|---|---|---|
| 01 | Failure (drone ship) | F9 v1.1 B1012 | CCAFS LC-40 |
| 04 | Failure (drone ship) | F9 v1.1 B1015 | CCAFS LC-40 |
| 06 | Precluded (drone ship) | F9 v1.1 B1018 | CCAFS LC-40 |

- The substr SQL function breaks down a string based on a starting character and a final character.

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

- SQL Query:

```
%%sql
    select [Landing _Outcome],
    count(Date) as count
from SPACEXTBL
where
    Date between '04-06-2010' and '20-03-2017'
group by [Landing _Outcome]
order by count desc
```

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

| Landing _Outcome | count |
|---|---|
| Success | 20 |
| No attempt | 10 |
| Success (drone ship) | 8 |
| Success (ground pad) | 6 |
| Failure (drone ship) | 4 |
| Failure | 3 |
| Controlled (ocean) | 3 |
| Failure (parachute) | 2 |
| No attempt | 1 |

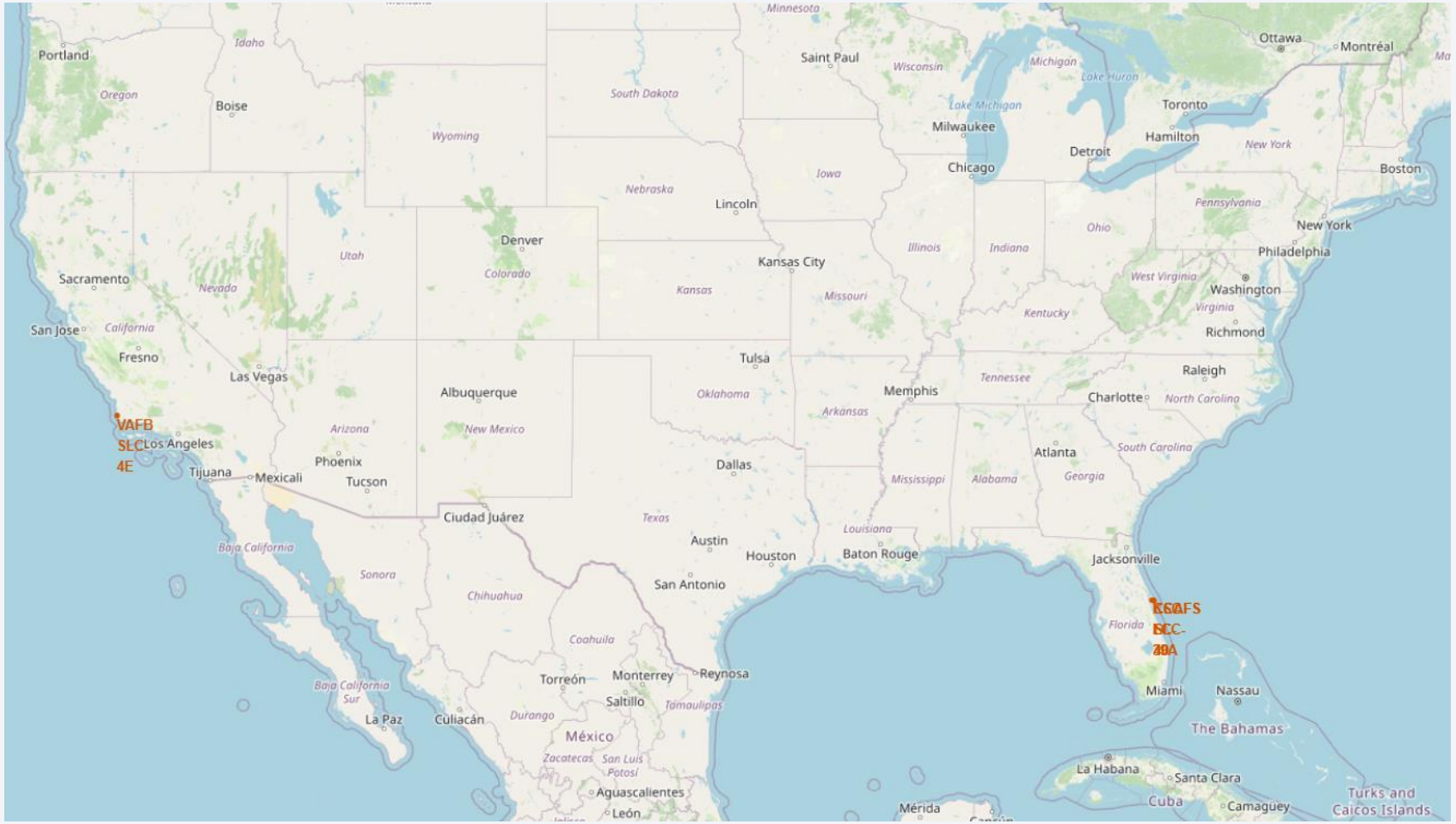
- The unique statement in this query is the order by, which organizes the result based on a column, in this case the statement ‘desc’ explicitly orders in descending order.

Section 3

# Launch Sites Proximities Analysis

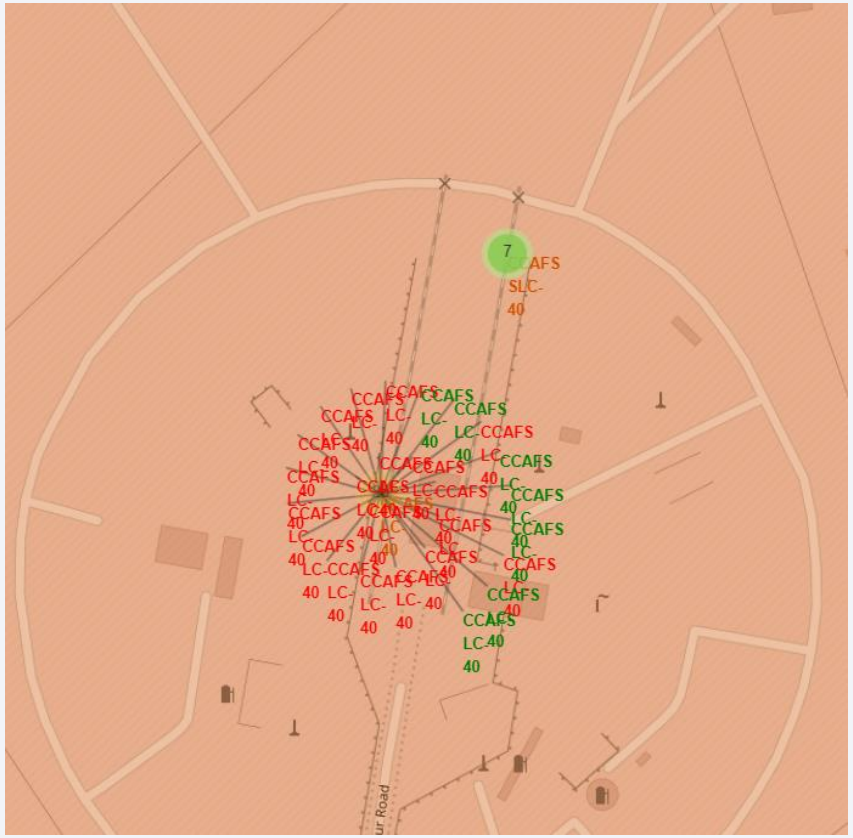
# SpaceX Launch Sites

- All launch sites where in the coast of the USA, making sure all sites where far from any major city in case of failure.
- Most of them where in Florida.



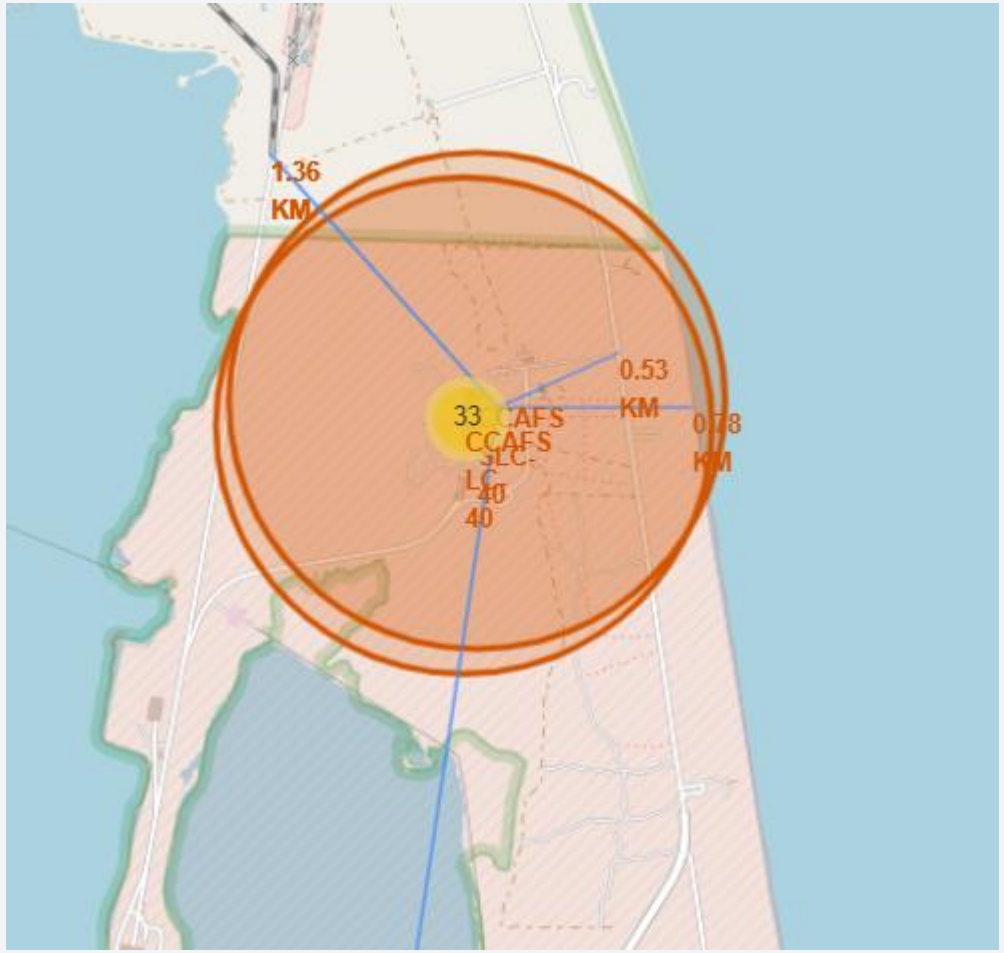
# Outcome Markers

- There seems to have been no correlation between success rate and launch site.



# Launch Site Proximity

- The launch site selected was shown to have proximity to Highway, Railway and cast line, with all within less than 1 mile of the launch site, yet it is quite far from the nearest city.



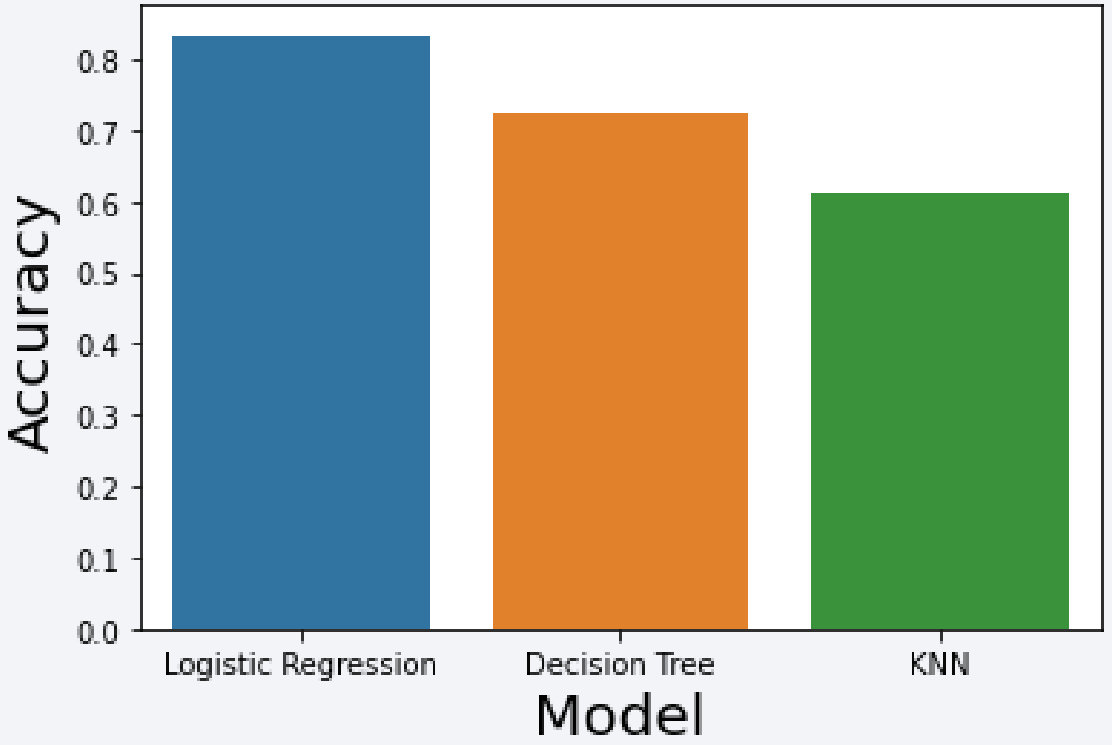
Section 5

# Predictive Analysis (Classification)

# Classification Accuracy

- With the code provided, SVM model where unable to run, it never stopped running even after 30min of waiting, so results where based on all the other models.

- The Logistic Regression model showed to be the most accurate.



# Confusion Matrix

- Confusion Matrix for the Logistic Regression:
  - Predicted label X Real Label;
  - No false negatives where predicted;
  - 3 false positives where predicted;

# Conclusions

- Many factors we had may affect the outcome:
  - Booster type, payload mass, launch site, orbit.
- Many factors we do not have may also affect the outcome like:
  - Weather conditions, software, etc;
- Some factors may appear to have strong effect on the outcome, but they may bee a result of the number of cases, like the Orbit type.
- The higher the number of launches on a site and/or rocket has a high impact on its success rate.
- The logistic regression model showed to be very effective on this dataset.

# Thank you!