IBM Developer
SKILLS NETWORK

# Winning Space Race with Data Science

Veeranon Thuvasin
25/09/2024

# Outline

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

# Executive Summary

- Summary of methodologies

  - Collecting the data by using web scraping and SpaceX API.
  - Do Exploratory Data Analysis (EDA), including data cleaning, data wrangling, data visualization and interactive visual analytics to gain better understanding and insight values of the data.
  - Using various Machine Learning methods to predict the outcome.

- Summary of all results

  - By Using EDA method. It allowed to identify which features are important to predict success of landing the first stage of the rockets .
  - KNN and LR are the models that predict the most accuracy to an unseen data (test set, and real word data)

# Introduction

- The objective is to evaluate the viability of the new company Space Y to compete with Space X.

- Problems you want to find answers
  - What are the main characteristics of a successful or failed landing ?
  - What are the data all about?
  - What features are important to predict the outcome?
  - What model that actually performs the best in predicting the outcome to an unseen data or real world data?

Section 1

# Methodology

# Methodology

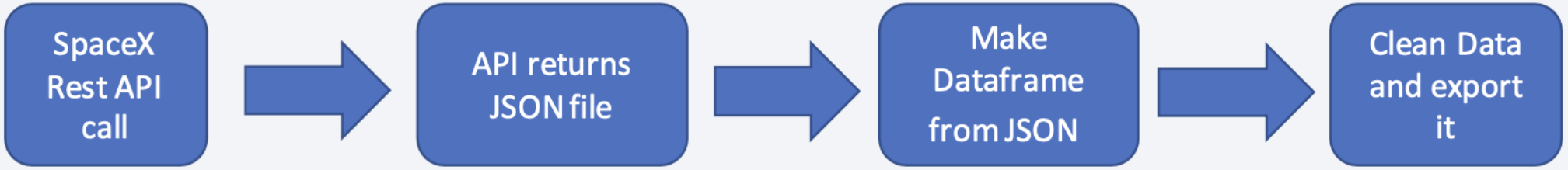
## Executive Summary

- Data collection methodology:
  - SpaceX REST API
  - Web Scrapping from Wikipedia
- Perform data wrangling
  - Dropping unnecessary columns
  - Using One Hot Encoding for classification models
- 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 Collection - API

- Data sets were collected from Space X API (https://api.spacexdata.com/v4/rockets/) and from Wikipedia (https://en.wikipedia.org/wiki/List_of_Falcon/_9/_and_Falcon_Heavy_launches) by using web scarping.

- The Space X REST API URL is api.spacexdata.com/v4/

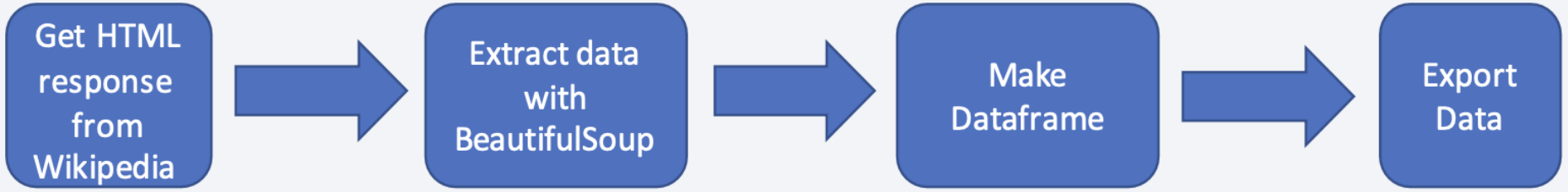



https://github.com/Simmmyirmm/IBM-DS-Certificate-Project/blob/main/01-Data-Collection-With-API.ipynb

# Data Collection - Wikipedia

- Data sets were collected from Space X API (https://api.spacexdata.com/v4/rockets/) and from Wikipedia (https://en.wikipedia.org/wiki/List_of_Falcon/_9/_and_Falcon_Heavy_launches) by using web scarping.

- URL is https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922




https://github.com/Simmmyirmm/IBM-DS-Certificate-Project/blob/main/01-Data-Collection-Web-Scraping.ipynb

# Data Collection – SpaceX API

- Request data from SpaceX API (rocket launch data)
- Decode response using .json() and convert to a dataframe using .json_normalize()
- Request information about the launches from SpaceX API using custom functions
- Create dictionary from the data
- Create dataframe from the dictionary
- Filter dataframe to contain only Falcon 9 launches
- Replace missing values of Payload Mass with calculated .mean()
- Export data to csv file

https://github.com/Simmmyirmm/IBM-DS-Certificate-Project/blob/main/01-Data-Collection-With-API.ipynb

# Data Collection – SpaceX API

## Getting Response from API

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

response = requests.get(spacex_url)
```

## Convert Response to JSON File

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

## Request additional data

```
# Call getLaunchSite
getLaunchSite(data)

# Call getPayloadData
getPayloadData(data)

# Call getCoreData
getCoreData(data)
```

## Create Dictionary

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

## Create Dataframe

```
# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
df.head(2)
```

## Filter Dataframe

```
data_falcon9 = df.loc[df['BoosterVersion'] == 'Falcon 9']
```

## Replace NaN

```
# Calculate the mean value of PayloadMass column
data_falcon9['PayloadMass'] = data_falcon9['PayloadMass'].fillna(data_falcon9['PayloadMass'].mean())
```

## Export to CSV

```
data_falcon9.to_csv('dataset_part_1.csv',index=False)
```

# Data Collection - Scraping

- Request data (Falcon 9 launch data) from Wikipedia
- Create BeautifulSoup object from HTML response
- Extract column names from HTML table header
- Collect data from parsing HTML tables
- Create dictionary from the data
- Create dataframe from the dictionary
- Export data to csv file

https://github.com/Simmmyirmm/IBM-DS-Certificate-Project/blob/main/01-Data-Collection-With-API.ipynb

# Data Collection – Scraping

## Getting Response from API

```
response = requests.get(static_url).text
```

## Convert Response to JSON File

```
soup = BeautifulSoup(response,'html.parser')
```

## Extract column names

```
html_tables = soup.select('tr')
first_launch_table = html_tables[2]
```

```
th_list = first_launch_table.select('th')
column_names = []
for i in th_list:
    name = extract_column_from_header(i)
    if name != '':
        column_names.append(name)
```

## Create Dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

## Add each data to each keys

```
import numpy as np

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()
```

## Create Dataframe

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
df = df.iloc[:106]
```

## Export to CSV

```
data_falcon9.to_csv('dataset_part_1.csv',index=False)
```

# Data Wrangling

- Perform EDA and determine data labels
- Calculate:
  - Number of launches for each site
  - Number and occurrence of orbit
  - Number and occurrence of mission outcome per orbit type
- Create binary landing outcome column (dependent variable)
- Export data to csv file

# Data Wrangling

## Landing Outcome (Landing Outcome column) -> Before converted

- True Ocean: mission outcome had a successful landing to a specific region of the ocean
- False Ocean: represented an unsuccessful landing to a specific region of ocean
- True RTLS: meant the mission had a successful landing on a ground pad
- False RTLS: represented an unsuccessful landing on a ground pad
- True ASDS: meant the mission outcome had a successful landing on a drone ship
- False ASDS: represented an unsuccessful landing on drone ship

## Landing Outcome (Landing Outcome column) -> After converted

- Outcomes converted into 1 for a successful landing and 0 for an unsuccessful landing

# Data Wrangling

Calculate Number launches of each site

```
df['LaunchSite'].value_counts()
```

Calculate Number and Occurrence of Obit

```
df['Orbit'].value_counts()
```

Number and occurrence of mission outcome per orbit type

```
landing_outcomes = df['Outcome'].value_counts()
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

Create binary landing outcome column

Export to CSV

```
df.to_csv("dataset_part_2.csv", index=False)
```

https://github.com/Simmmyirmm/IBM-DS-Certificate-Project/blob/main/02-Data-Wrangling.ipynb

# EDA with Data Visualization

- Charts
  - Flight Number vs. Payload • Flight Number vs. Launch Site
  - Payload Mass (kg) vs. Launch Site
  - Payload Mass (kg) vs. Orbit type EDA with Visualization Analysis
  - View relationship by using scatter plots.
- Analysis
  - View relationship by using scatter plots. The variables could be useful for machine learning if a relationship exists.
  - Show comparisons among discrete categories with bar charts. Bar charts show the relationships among the categories and a measured value.

https://github.com/Simmmyirmm/IBM-DS-Certificate-Project/blob/main/04-EDA-Data-Visualization.ipynb

# EDA with Pandas

https://github.com/Simmmyirmm/IBM-DS-Certificate-Project/blob/main/03-EDA-With-Pandas.ipynb

Using Pandas to gather and understand data from dataset:

- Displaying the names of the unique launch sites in the space mission.
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS).
- Display average payload mass carried by booster version F9 v1.1.
- List the date when the first successful landing outcome in ground pad was achieved.
- 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 payload mass.
- List the records which will display the month names, failure landing outcomes in drone ship, booster versions, launch site for the months in year 2015.
- Rank the count of successful landing outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

# Build an Interactive Map with Folium

- Markers, circles, lines and marker clusters were used with Folium Map
  - Markers indicate points like launch sites;
  - 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.
  - Lines are used to indicate distances between two coordinates.

https://github.com/Simmmyirmm/IBM-DS-Certificate-Project/blob/main/05-Interactive-EDA-Visualization-Folium.ipynb

# Build a Dashboard with Plotly Dash

- Dropdown List with Launch Sites
  - Allowing user to select all launch sites or a certain launch site
- Pie Chart Showing Successful Launches
  - Allowing user to see successful and unsuccessful launches as a percentage of the total launches
- Scatter Plot Showing Payload Mass vs. Success Rate by Booster Version
  - Allowing user to see the correlation between Payload and Launch Success
- Slider of Payload Mass Range
  - Allowing user to select payload mass range

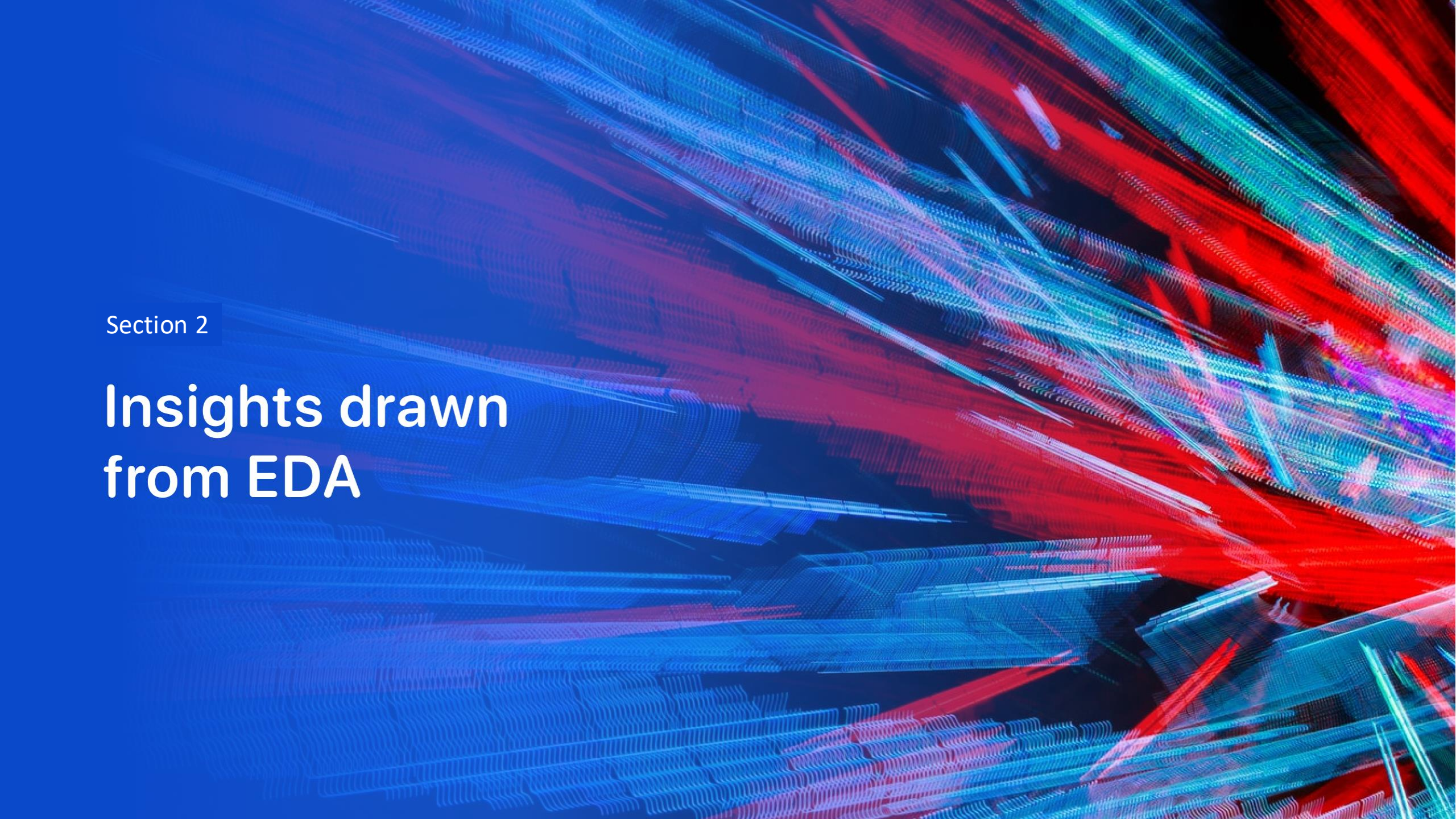
# Predictive Analysis (Classification)

- Data preparation
  - Load dataset
  - Normalize data
  - Split data into training and test sets.
- Model preparation
  - Selection of machine learning algorithms
  - Set parameters for each algorithm to GridSearchCV
  - Training GridSearchModel models with training dataset
- Model evaluation
  - Get best hyperparameters for each type of model
  - Compute accuracy for each model with test dataset
  - Plot Confusion Matrix
- Model comparison
  - Comparison of models according to their accuracy
  - The model with the best accuracy will be chosen (see Notebook for result)

https://github.com/Simmmyirmm/IBM-DS-Certificate-Project/blob/main/06-Machine-Learning-First-Stage-Prediction.ipynb

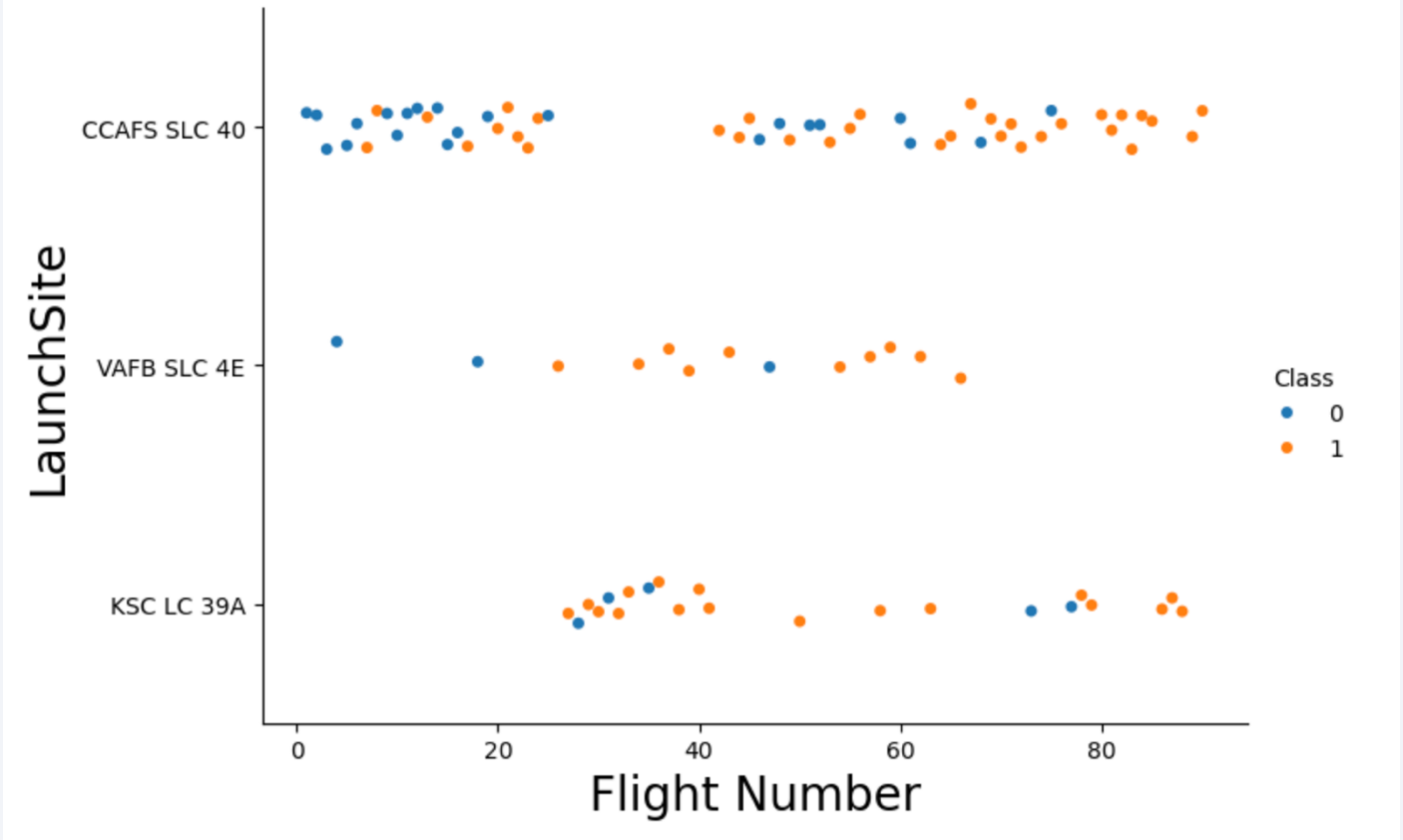
# Results

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


Section 2
Insights drawn from EDA

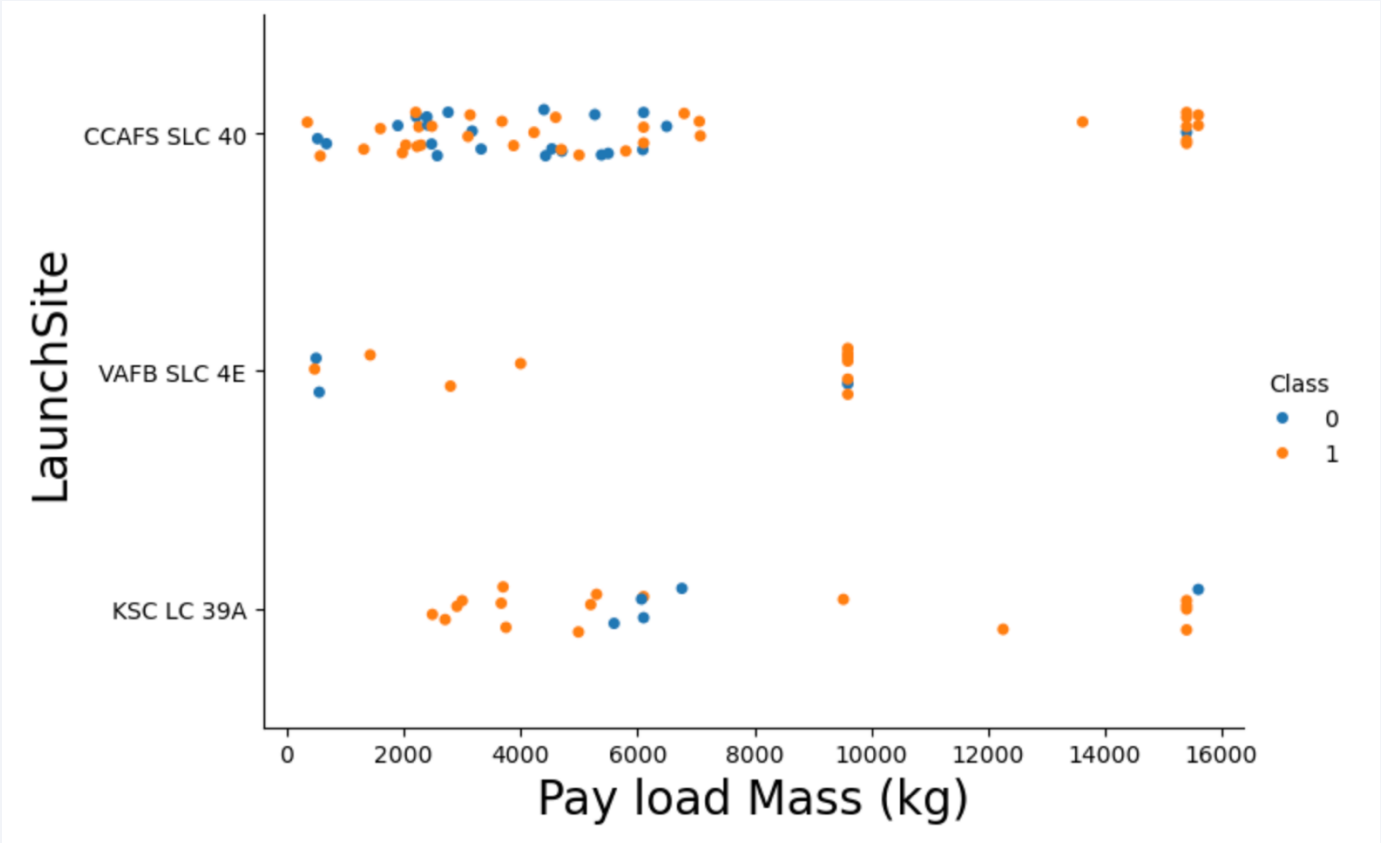
# Flight Number vs. Launch Site

- **Exploratory Data Analysis**
  - Earlier flights had a lower success rate (blue = fail)
  - Later flights had a higher success rate (orange = success)
  - Around half of launches were from CCAFS SLC 40 launch site
  - VAFB SLC 4E and KSC LC 39A have high success rates



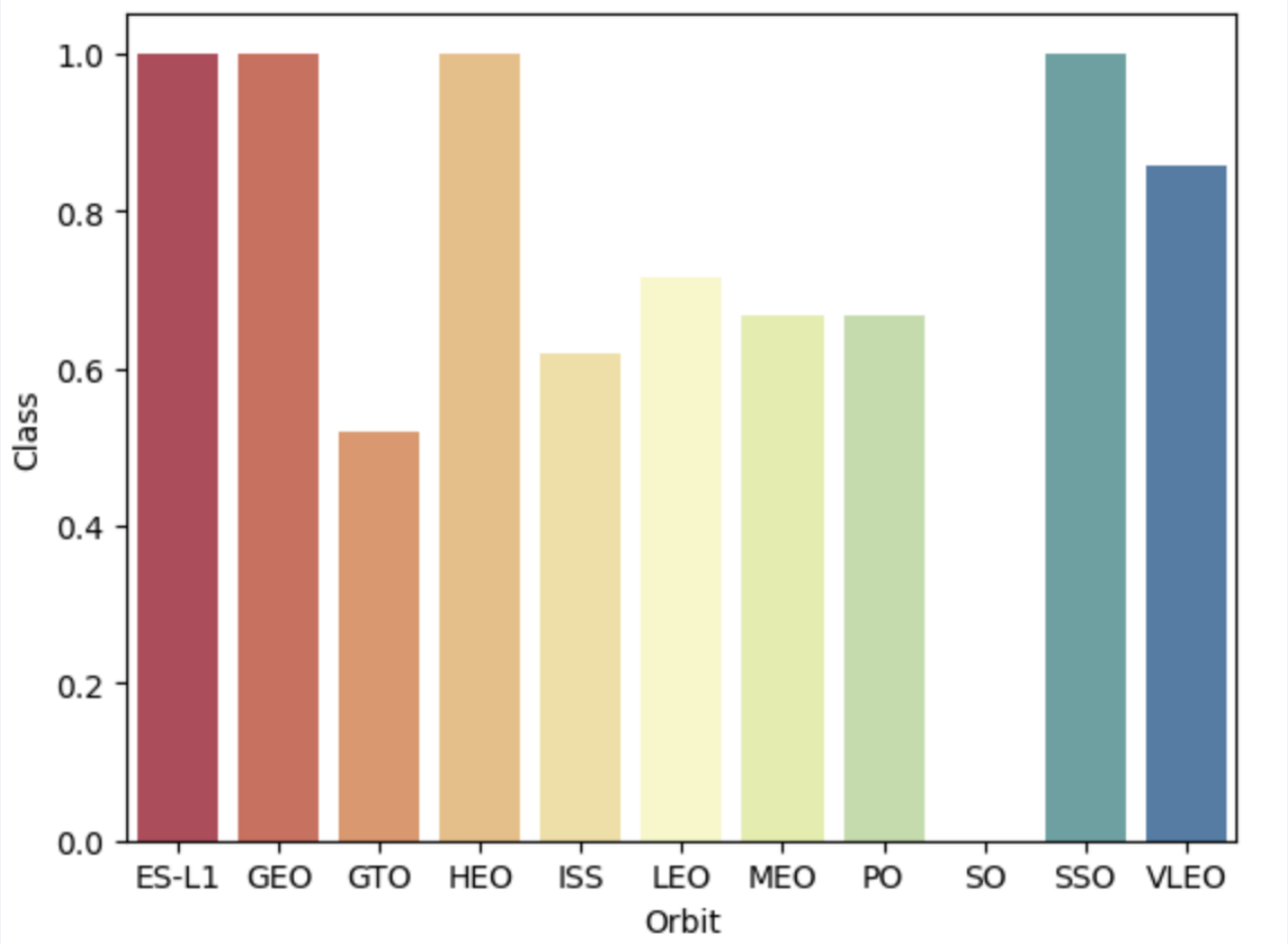
# Payload vs. Launch Site

- **Exploratory Data Analysis**
  - KSC LC 39A has a 100% success rate of launching rockets that weights less than 5,500 kg
  - Rockets that weight more than 7,000 kg are likely to launch successfully
  - The higher the payload mass (kg), the higher the success rate
  - Interestingly, VAFB SLC 4E never launches rockets that weight more than 10,000 kg



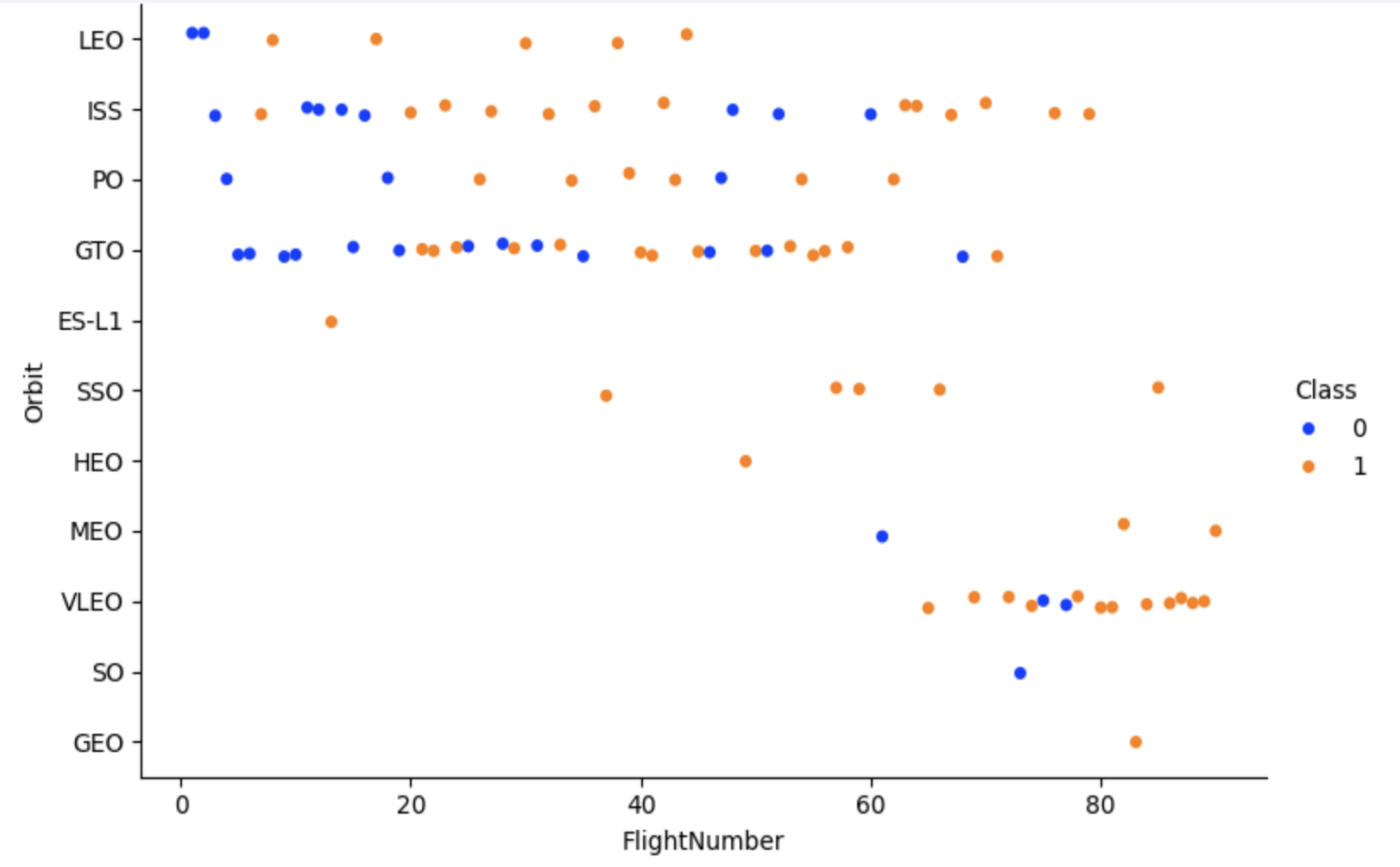
# Success Rate vs. Orbit Type

- **Exploratory Data Analysis**
  - ES-L1, GEO, HEO, and SSO have a 100% success rate
  - GTO, ISS, LEO, MEO, PO, and VLEO have a success rate between 50 – 80%
  - SO has a 0% success rate



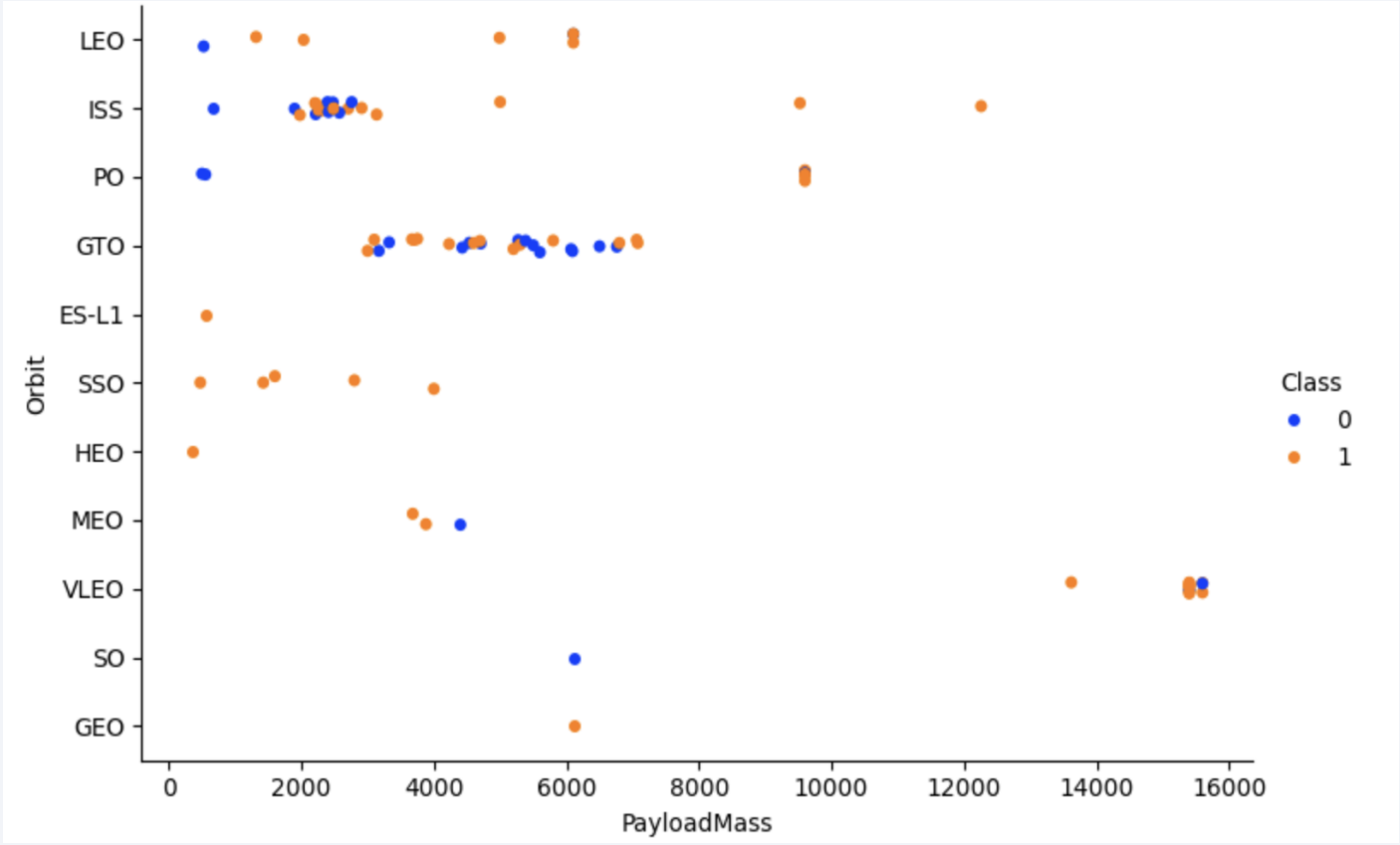
# Flight Number vs. Orbit Type

- **Exploratory Data Analysis**
  - Flights that have been launched lately have a higher chance of success than previously
  - Flights that have been launched to SSO, VLEO, and LEO have high success rate
  - There is insufficient evidence of data to conclude the success rate for GEO, ES-L1, and HEO



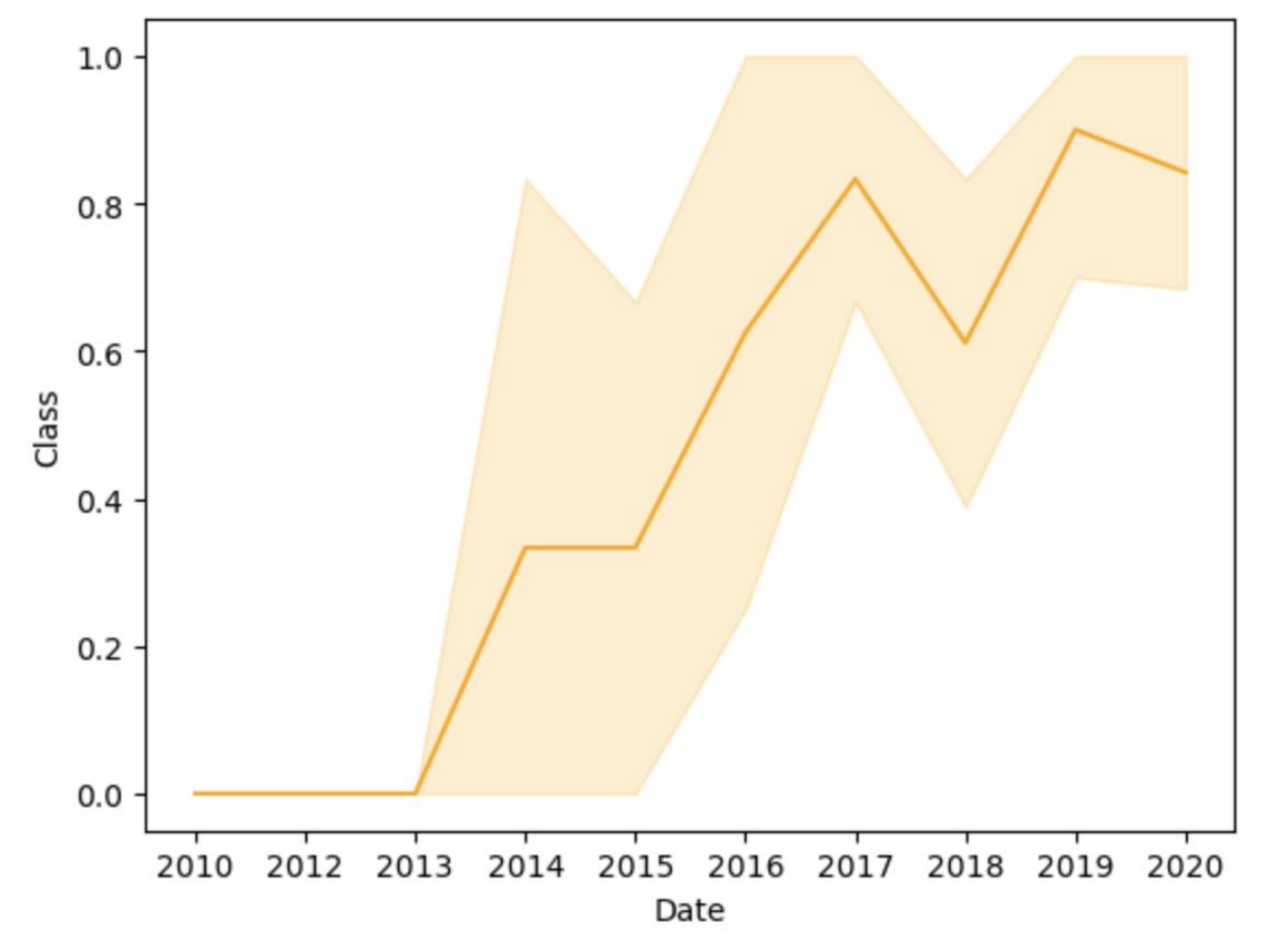
# Payload Mass vs. Orbit Type

- **Exploratory Data Analysis**
  - SSO, and LEO are the most suitable orbit for rockets that have weight less than 5,000 kg
  - PO, VLEO, and ISS are suitable for rockets that have weight more than 9,000 kg
  - The riskiest orbit to send rockets to is GTO



# Launch Success Yearly Trend

- **Exploratory Data Analysis**
  - The success rate has drastically improved from 2013-2017 and 2018-2019
  - The success rate decreased from 2017-2018 and from 2019-2020
  - Overall, the success rate has improved since 2013



# All Launch Site Names

- According to data, there are four launch sites:

```
df['Launch_Site'].value_counts()
```

```
Launch_Site
CCAFS SLC-40    34
CCAFS LC-40     26
KSC LC-39A      25
VAFB SLC-4E     16
```

# Launch Site Names Begin with 'CCA'

- Displaying 5 records below

```
df.loc[df['Launch_Site'].str.startswith('CCA')].head()
```

| | Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2010-06-04 | 18:45:00 | F9 v1.0 B0003 | CCAFS LC-40 | Dragon Spacecraft Qualification Unit | 0 | LEO | SpaceX | Success |
| **1** | 2010-12-08 | 15:43:00 | F9 v1.0 B0004 | CCAFS LC-40 | Dragon demo flight C1, two CubeSats, barrel of... | 0 | LEO (ISS) | NASA (COTS) NRO | Success |
| **2** | 2012-05-22 | 7:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success |
| **3** | 2012-10-08 | 0:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success |
| **4** | 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success |

# Total Payload Mass from NASA

- 45,596 kg (total) are carried by rockets that were launched by NASA(CRS)

```
group_df = df.groupby('Customer')['PAYLOAD_MASS__KG_'].sum()
group_df.loc[['NASA (CRS)']]
```

```
Customer
NASA (CRS)    45596
Name: PAYLOAD_MASS__KG_, dtype: int64
```

# Average Payload Mass by F9 v1.1

- 2,535 kg (average) are carried by booster version F9 v1.1

```
df.loc[df['Booster_Version'].str.contains('F9 v1.1')][['PAYLOAD_MASS__KG_']].mean()
```

```
PAYLOAD_MASS__KG_    2534.666667
dtype: float64
```

# First Successful Ground Landing Date

- The first successful landing outcome that was landed on the ground pad.

```
df.loc[df['Landing_Outcome'] == 'Success (ground pad)'].sort_values(by=['Date']).head(1)
# df.head(20)
```

| | Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome |
|---|---|---|---|---|---|---|---|---|---|
| **19** | 2015-12-22 | 1:29:00 | F9 FT B1019 | CCAFS LC-40 | OG2 Mission 2 11 Orbcomm-OG2 satellites | 2034 | LEO | Orbcomm | Success |

# Successful Drone Ship Landing with Payload between 4000 and 6000

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

```
df.loc[(df['Landing_Outcome'] == 'Success (drone ship)') & (df['PAYLOAD_MASS__KG_'] > 4000) & (df['PAYLOAD_MASS__KG_']
```

| | Date | Time (UTC) | Booster_Version | Launch_Site | Payload | PAYLOAD_MASS__KG_ | Orbit | Customer | Mission_Outcome | Landing_Outc |
|---|---|---|---|---|---|---|---|---|---|---|
| **23** | 2016-05-06 | 5:21:00 | F9 FT B1022 | CCAFS LC-40 | JCSAT-14 | 4696 | GTO | SKY Perfect JSAT Group | Success | Success (d |
| **27** | 2016-08-14 | 5:26:00 | F9 FT B1026 | CCAFS LC-40 | JCSAT-16 | 4600 | GTO | SKY Perfect JSAT Group | Success | Success (d |
| **31** | 2017-03-30 | 22:27:00 | F9 FT B1021.2 | KSC LC-39A | SES-10 | 5300 | GTO | SES | Success | Success (d |
| **42** | 2017-10-11 | 22:53:00 | F9 FT B1031.2 | KSC LC-39A | SES-11 / EchoStar 105 | 5200 | GTO | SES EchoStar | Success | Success (d |

# Total Number of Successful and Failure Mission Outcomes

- 99 Success
- 1 Success (payload status unclear)
- 1 Failure in Flight

```
df['Mission_Outcome'].value_counts()
```

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

# Boosters Carried Maximum Payload

- List of the names of the booster which have carried the maximum payload mass

```
df.loc[df['PAYLOAD_MASS__KG_'] == df['PAYLOAD_MASS__KG_'].max()][['Booster_Version']]
```

| | Booster_Version |
|---|---|
| **74** | F9 B5 B1048.4 |
| **77** | F9 B5 B1049.4 |
| **79** | F9 B5 B1051.3 |
| **80** | F9 B5 B1056.4 |
| **82** | F9 B5 B1048.5 |
| **83** | F9 B5 B1051.4 |
| **85** | F9 B5 B1049.5 |
| **92** | F9 B5 B1060.2 |
| **93** | F9 B5 B1058.3 |
| **94** | F9 B5 B1051.6 |
| **95** | F9 B5 B1060.3 |
| **99** | F9 B5 B1049.7 |

# 2015 Launch Records

- List of the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015

```
df.loc[(df['Landing_Outcome'] == 'Failure (drone ship)') & (df['Date'] >= '2015-01-01') & (df['Date'] <= '2015-12-31')]
```

| | Date | Booster_Version | Launch_Site | Landing_Outcome |
|---|---|---|---|---|
| **13** | 2015-01-10 | F9 v1.1 B1012 | CCAFS LC-40 | Failure (drone ship) |
| **16** | 2015-04-14 | F9 v1.1 B1015 | CCAFS LC-40 | Failure (drone ship) |

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

- Rank 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 order

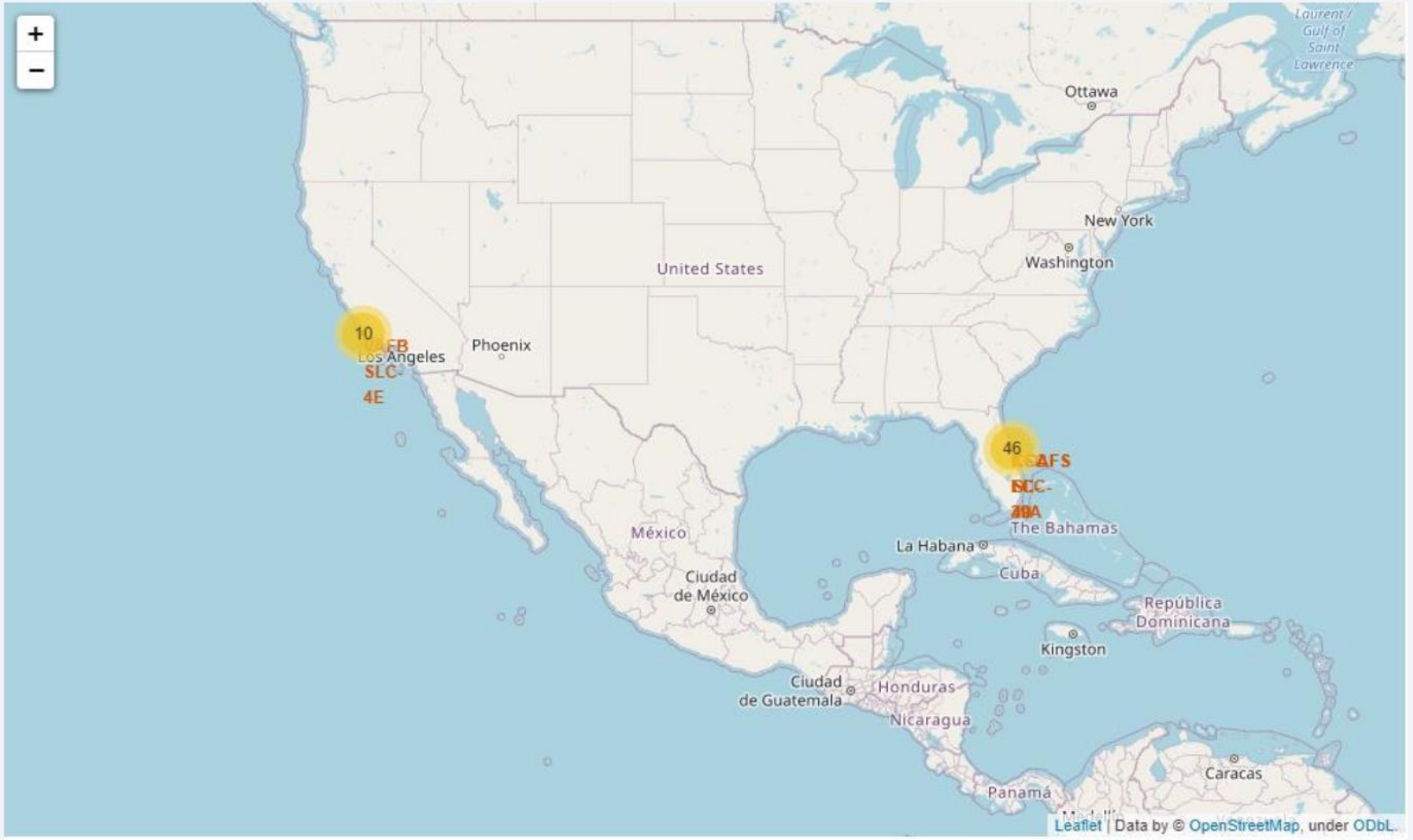
```
rank_grouped_df = df.loc[(df['Date'] >= '2010-06-04') & (df['Date'] <= '2017-03-20')].groupby('Landing_Outcome')['Landi
rank_grouped_df = rank_grouped_df.sort_values(ascending=False)
rank_grouped_df
```

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

Section 3

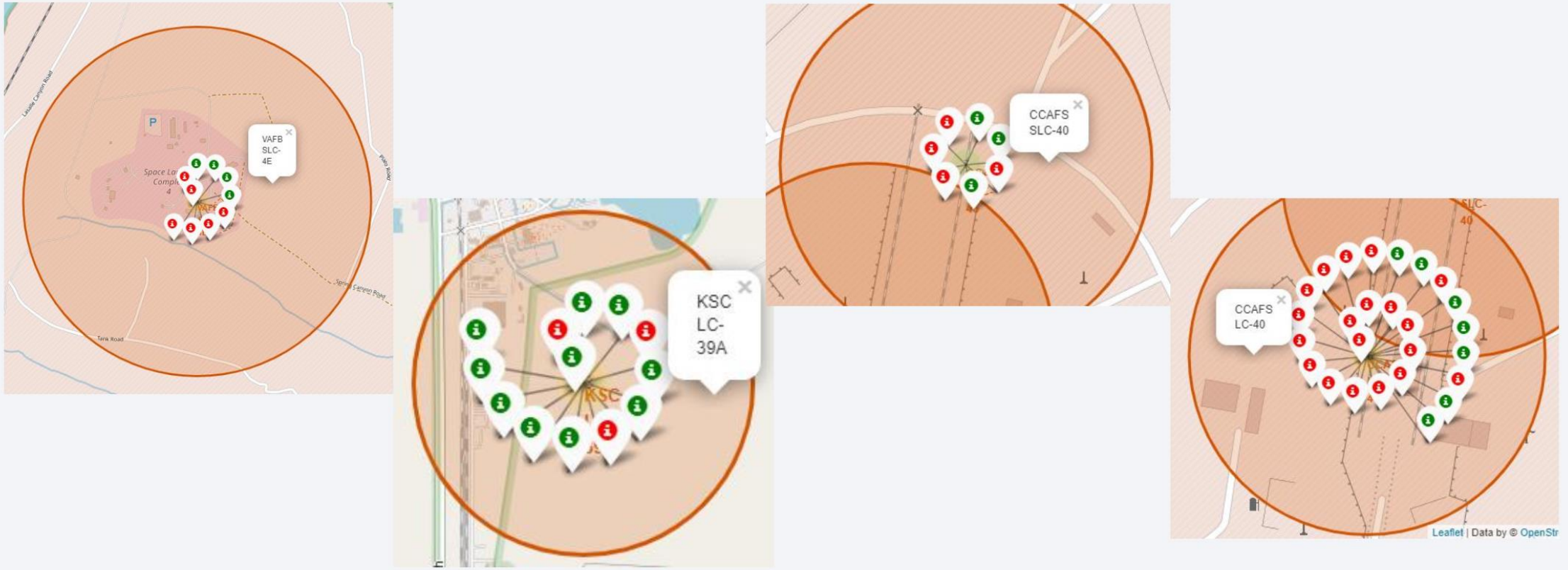
# Launch Sites Proximities Analysis

# Folium map – Launch sites



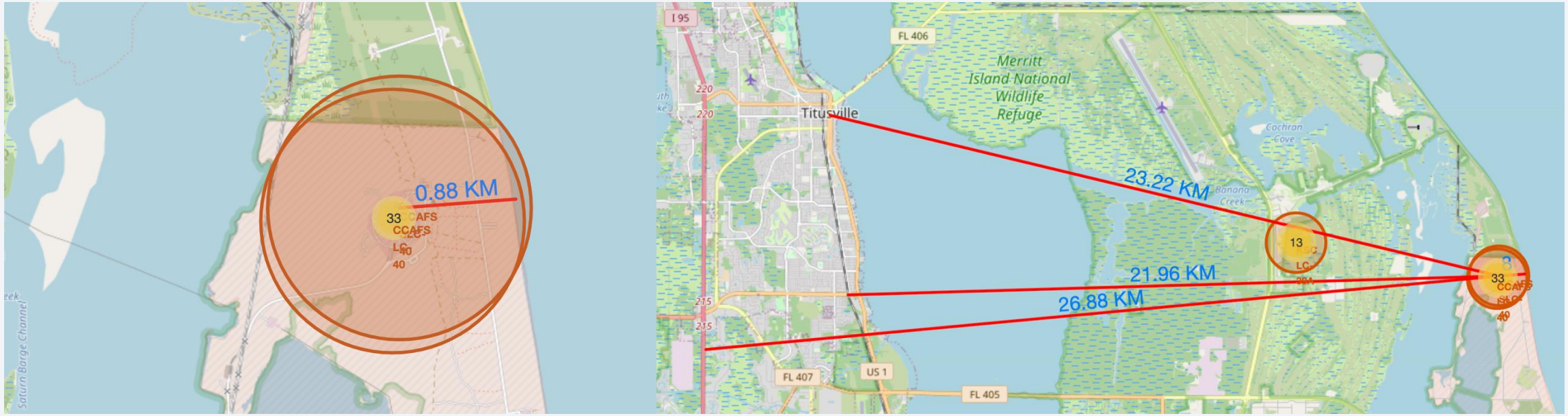

- Explain the important elements and findings on the screenshot

# Folium map – Color markers showing the launch outcomes

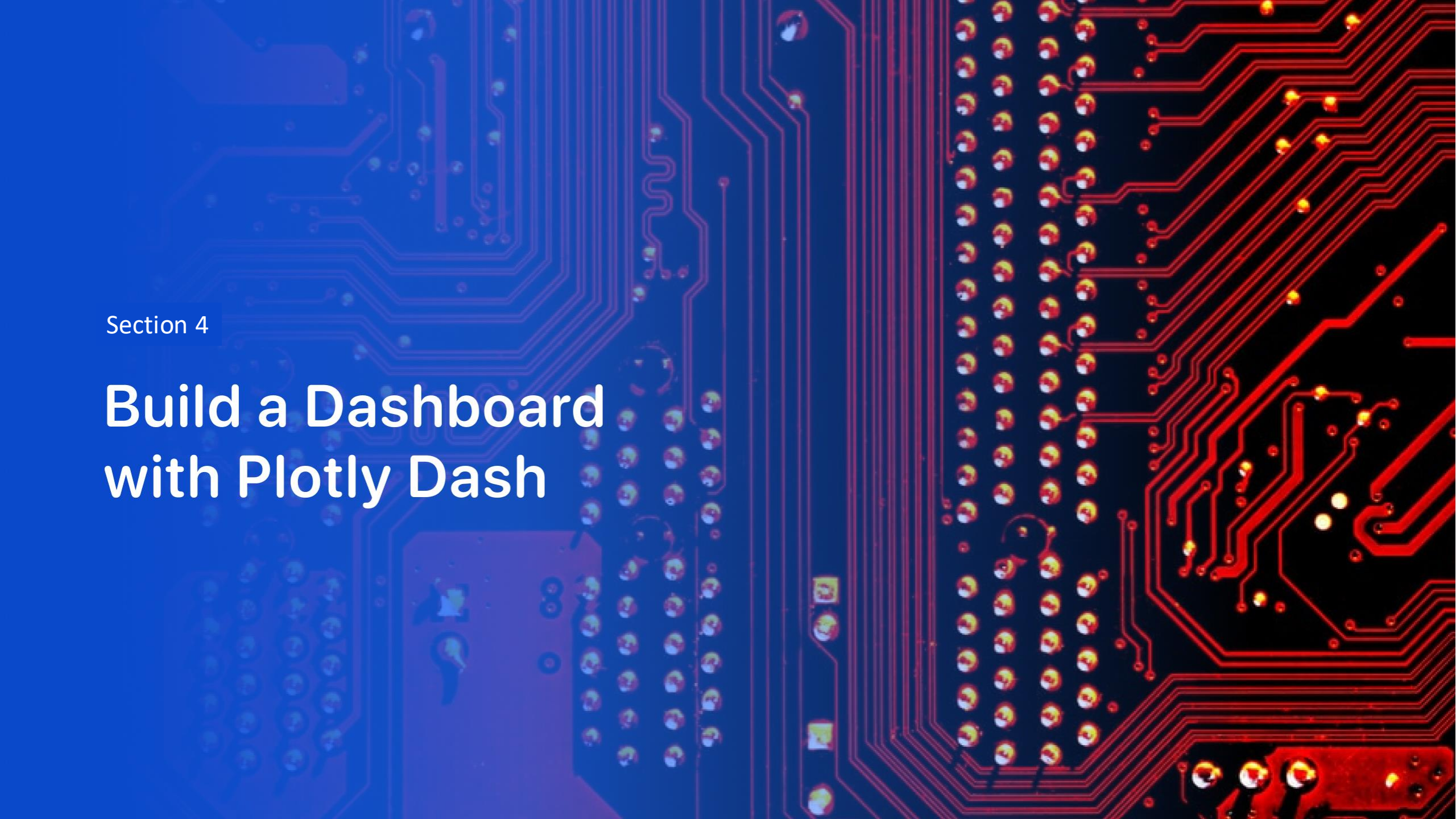



- KSC LC-39A has the most successful rate of launching the rockets, and CCAFS LC-40 has the least successful.

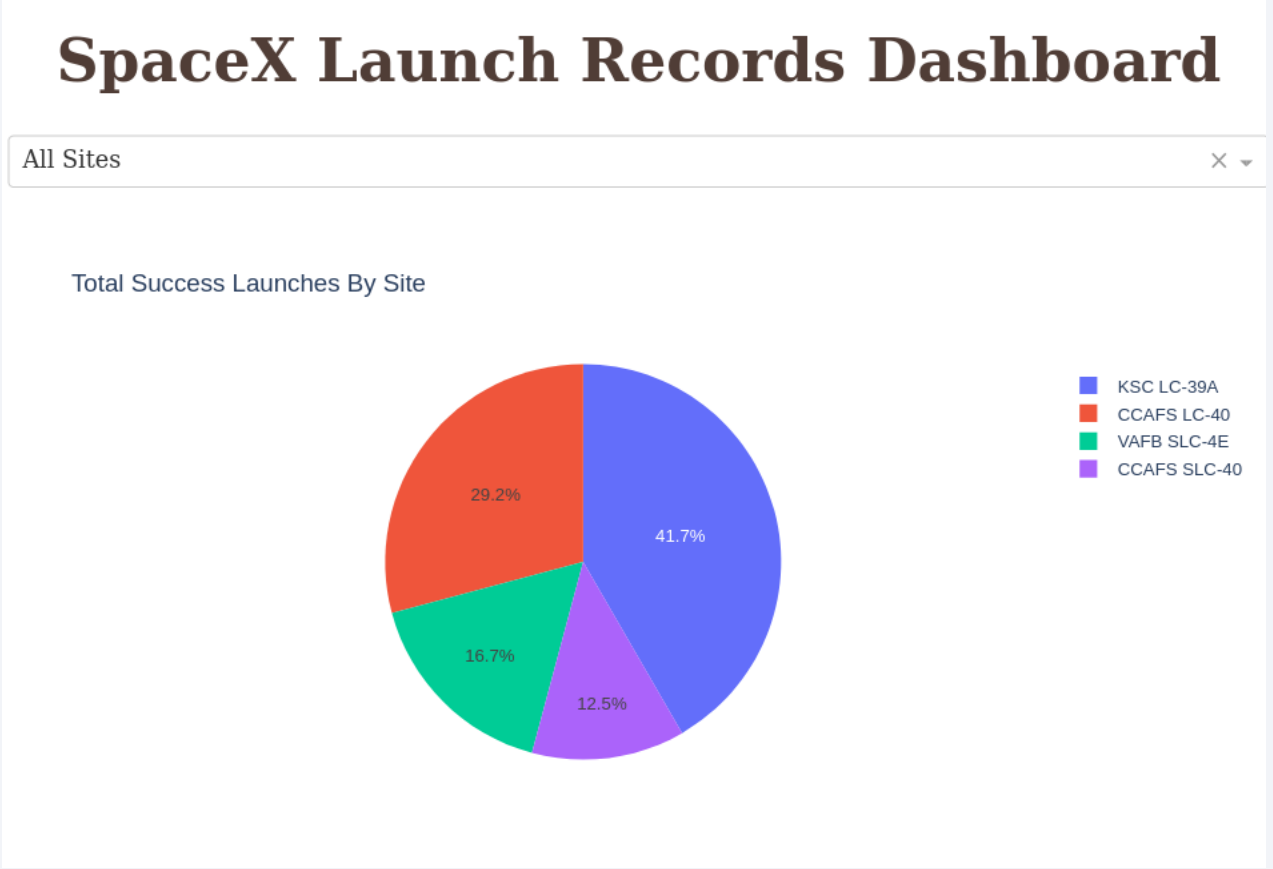
# Folium Map - Distance to Proximities (CCAFS SLC-40)




- 0.88 km from nearest coastline
- 23.23 km from nearest city
- 21.96 km from nearest railway
- 26.88 km from nearest highway

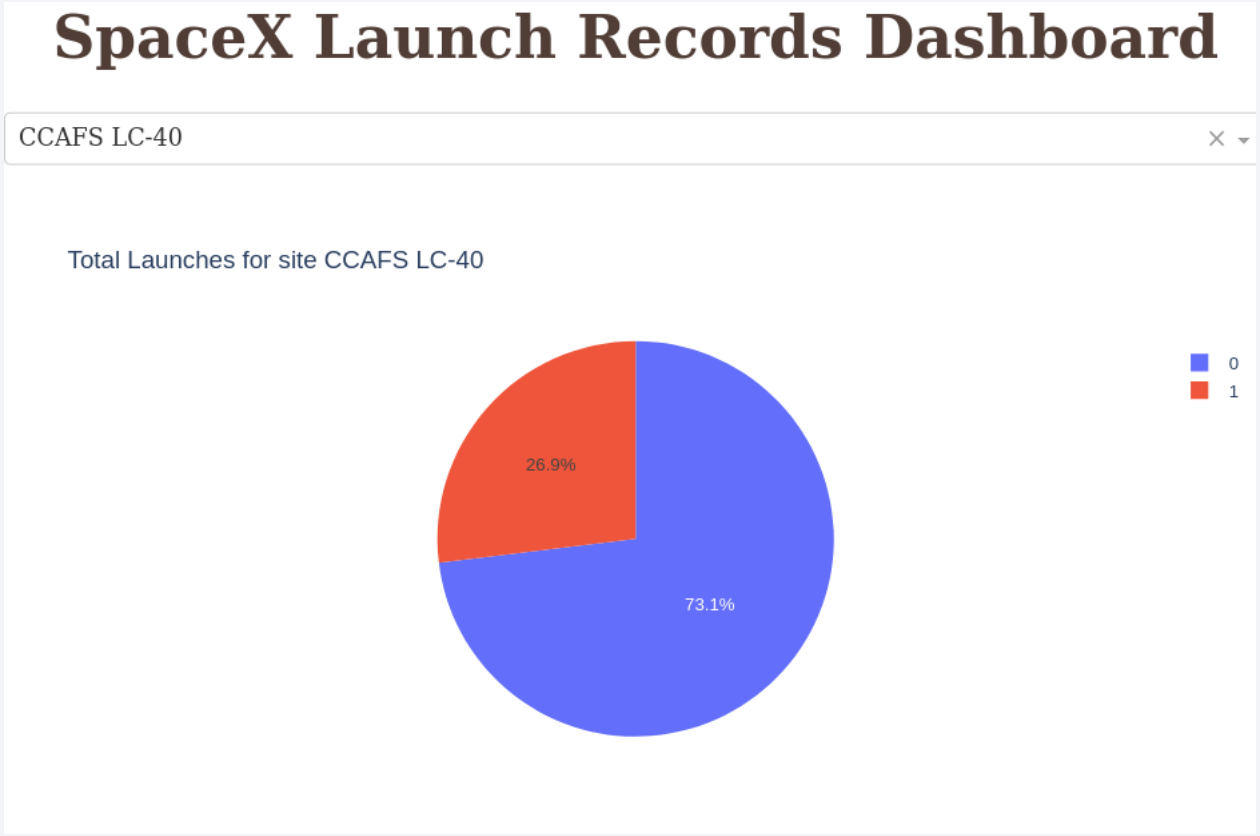

Section 4
Build a Dashboard with Plotly Dash

# Dashboard – Total success by Site



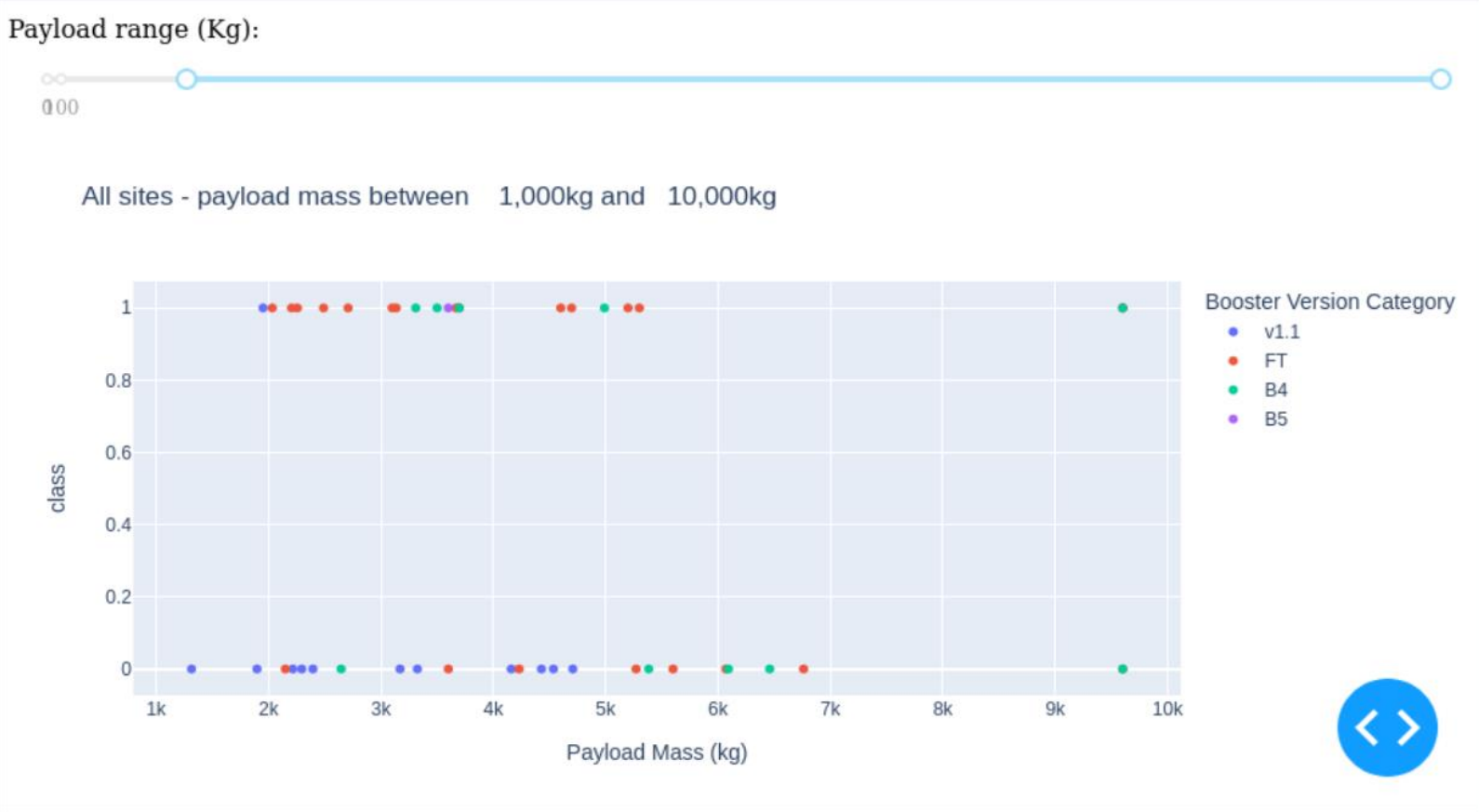

- KSC LC-39A has the most successful launches, and CCAFS LC-40 has successful launches nearly double of VAFB SLC-4E and CCAFS SLC-40.

# Dashboard – Total success launches for Site CCAFS LC-40




- KSC LC-39A has achieved a 73.1% success rate while getting a 26.9% failure rate.

# Dashboard – Payload mass vs Outcome

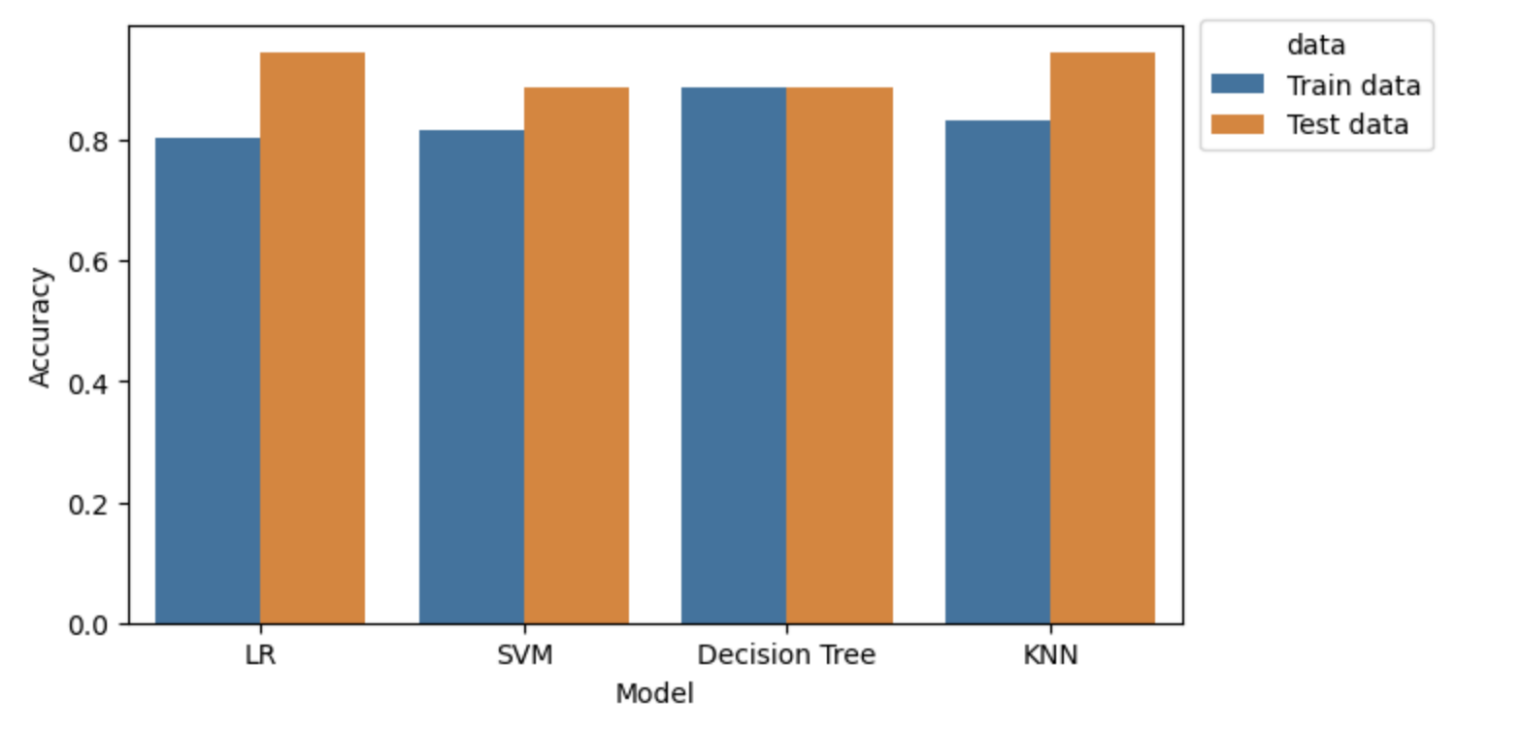



- Payloads that have mass lower than 6,000kg and booster version is FT have the highest success rate chances. On the other hand, payloads that have booster version 1.1 have the lowest success rate.

Section 5

# Predictive Analysis (Classification)

# Classification Accuracy



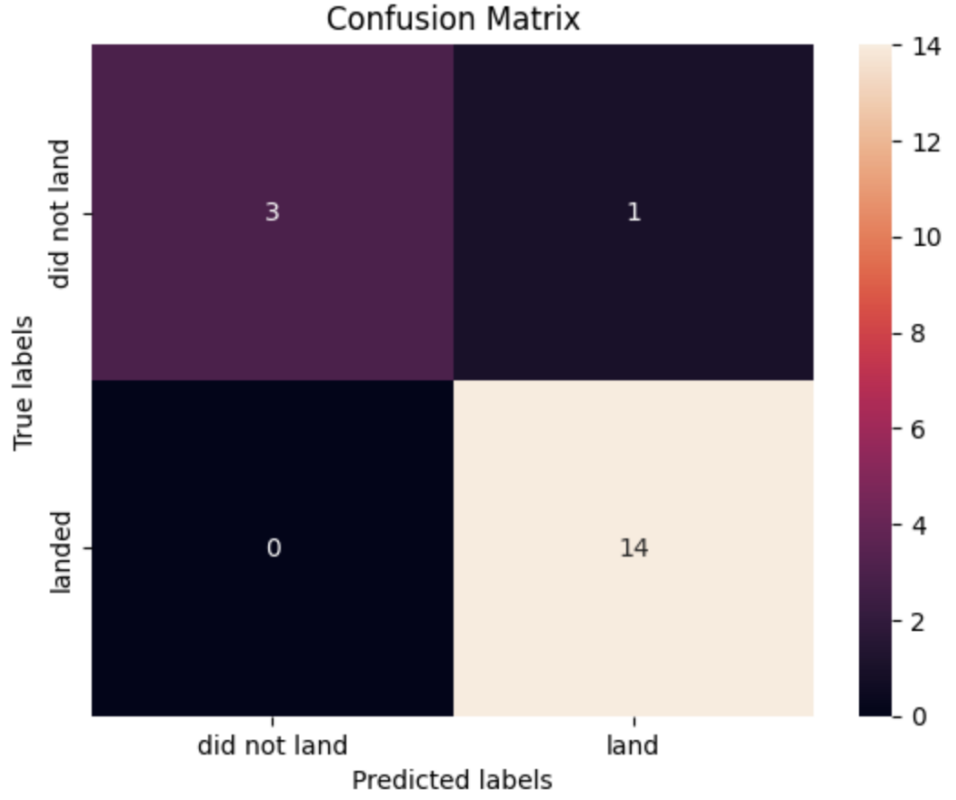

- The model that was performed the best on training data is Decision Tree
- The model that was performed the best on testing data is KNN
- The testing dataset closely resembles to real-world data (newly unseen data)
- Therefore, KNN is the most suitable model to choose in this case.

# Confusion Matrix

- A confusion matrix summarises the performance of a classification algorithm
- Confusion Matrix Outputs:
  - 14 True positive
  - 3 True negative
  - 1 False positive
  - 0 False Negative

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.75 | 0.86 | 4 |
| 1 | 0.93 | 1.00 | 0.97 | 14 |
| accuracy | | | 0.94 | 18 |
| macro avg | 0.97 | 0.88 | 0.91 | 18 |
| weighted avg | 0.95 | 0.94 | 0.94 | 18 |

```
{'algorithm': 'auto', 'n_neighbors': 7, 'p': 1}
```



# Conclusions

- The success of a mission can be explained by several factors such as the launch site, the orbit and especially the payload mass.
- While most mission outcomes are successful, landing success rates appear to improve over time due to advancements in processes and rocket technology.
- The orbits with the best success rates are SSO, VLEO, and LEO.
- KSC LC-39A: Has the highest success rate among launch sites. 100% success rate for launches less than 5,500 kg.
- The payload mass can be an important factor for mission success, depending on the orbit. In general, lighter payloads tend to perform better than heavier ones.
- KNN performed the best on the testing data, which closely mirrors real-world (unseen) data. As a result, KNN is the most suitable model to choose in this scenario.

# Thank you!