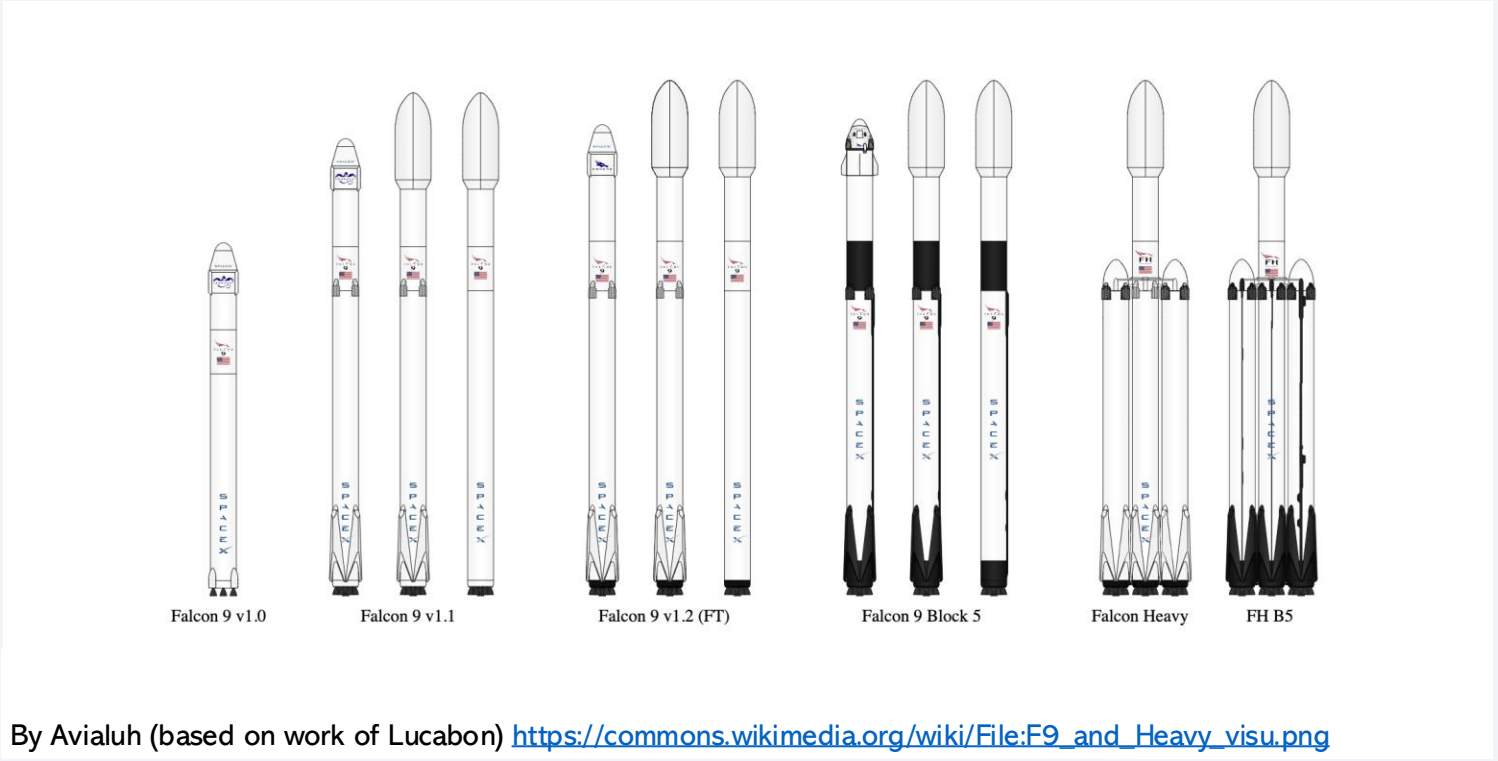
IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Marina Sequinel
July 2024

Outline

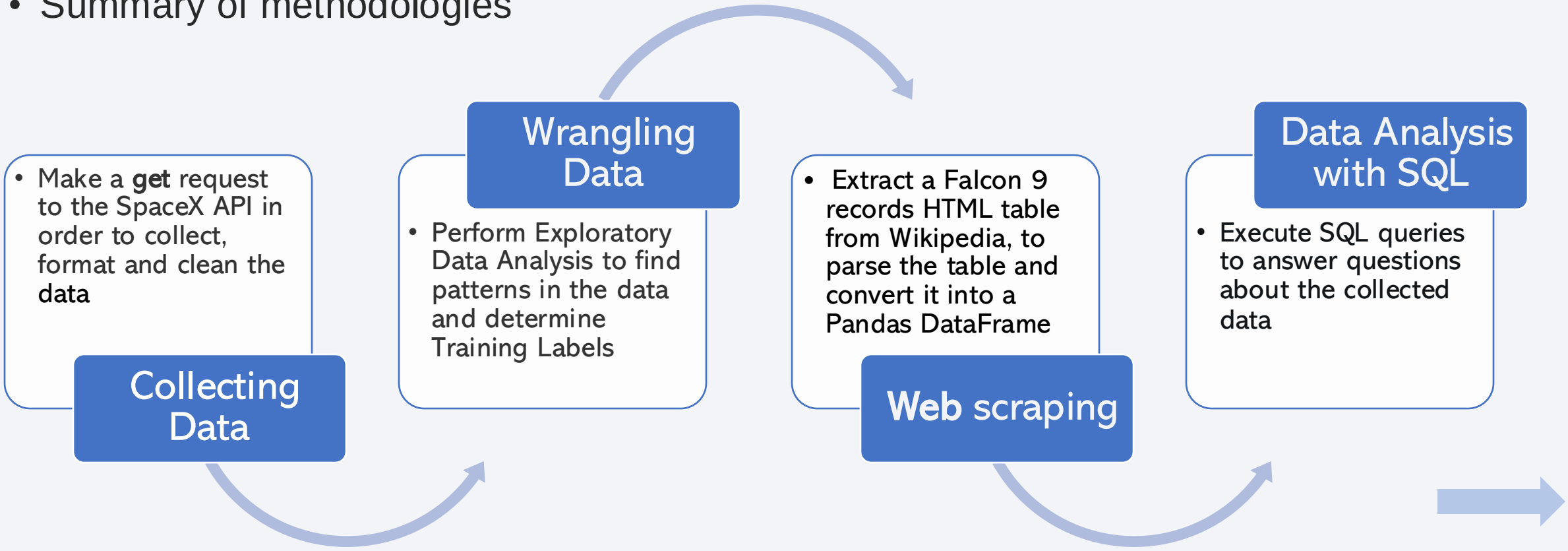
- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix



By Avialuh (based on work of Lucabon) https://commons.wikimedia.org/wiki/File:F9_and_Heavy_visu.png

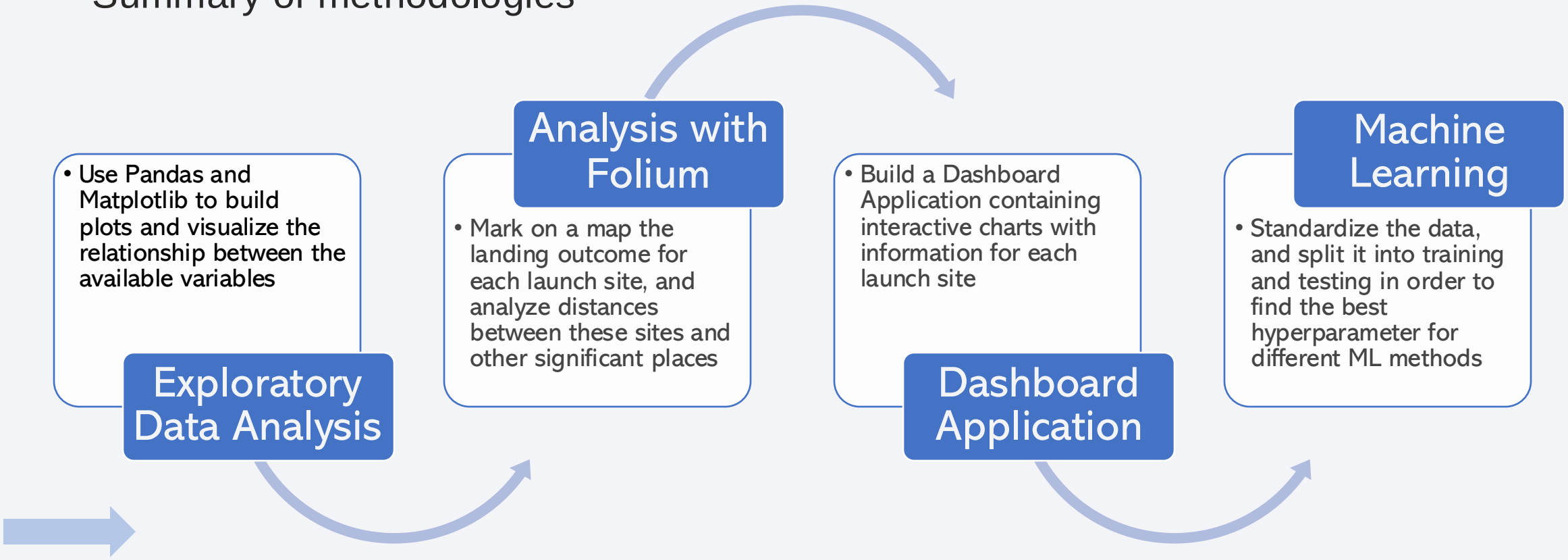
Executive Summary

- Summary of methodologies



Executive Summary

- Summary of methodologies



Executive Summary

- Summary of all results

- Exploratory Data Analysis Results
- Interactive Analysis with Dashboard and Folium Maps
- Predictive Analysis using Machine Learning

Introduction

- In the 21st century space race, one of the companies that stands out the most is **SpaceX,** founded in 2002 by the entrepreneur Elon Musk. Unlike other rocket providers, the **Falcon 9 spacecraft** can recover the bottom section of the rocket stack and reuse it, saving a significant amount of money and resources in its missions. On other occasions, SpaceX sacrifices the **first stage** due to parameters like **payload, orbit and customer.**

- Therefore, our goal is to use the data provided by the company to predict whether SpaceX will reuse the first stage or not, taking into consideration different parameters such as the ones mentioned above. We will also analyze the relationship between variables (launch sites, landing outcome, orbit and so on) and how one can influence the other.

Section 1

Methodology

Methodology

Executive Summary

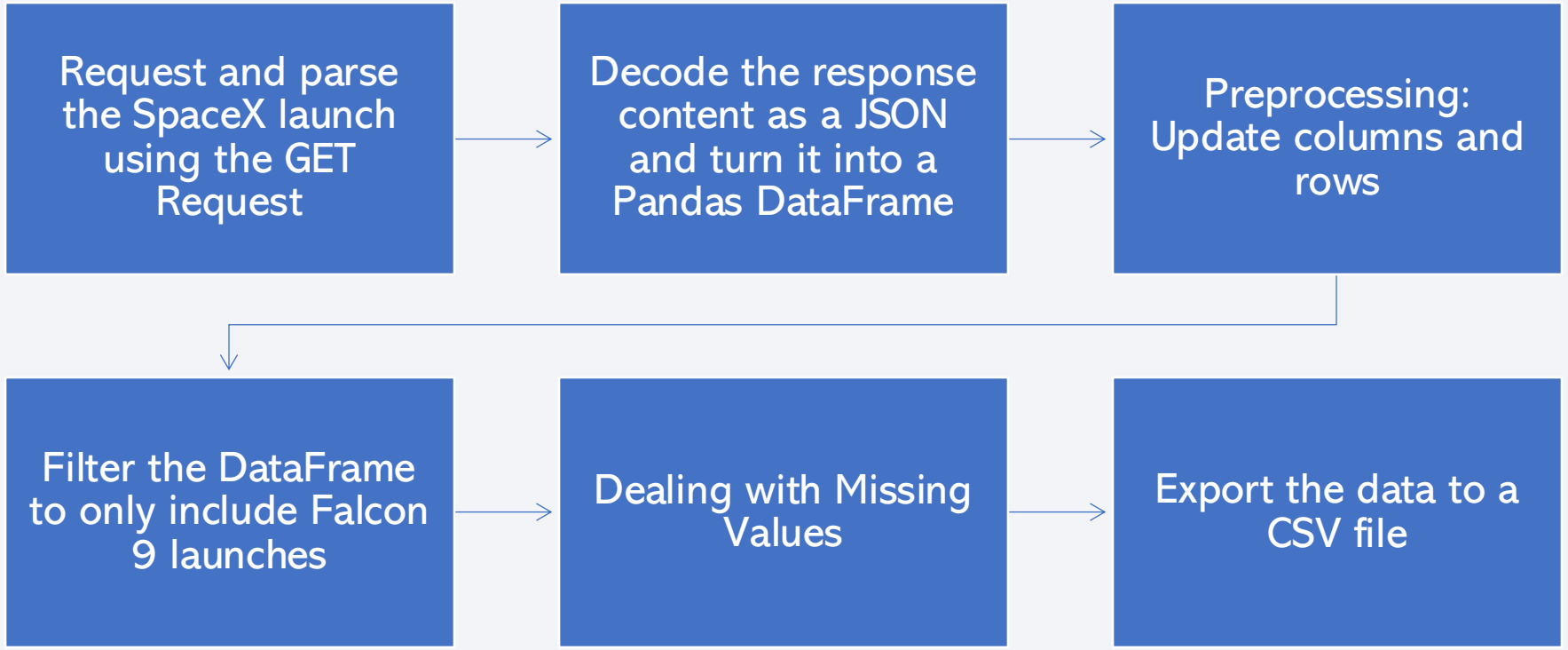
- Data collection methodology
  - A get request to the SpaceX API made it possible to collect and normalize the data we need it
- Perform data wrangling
  - With Pandas and NumPy, we found patterns in the data and determined Training Labels. We also used BeautifulSoup to extract HTML records to clean and format the data
- Perform Exploratory Data Analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - We built, trained and evaluated the best hyperparameters for SVM, Classification Trees and Logistic Regression

Data Collection

github link:
https://github.com/marinasequinel/spacexdata/blob/main/jupyter-labs-spacex-data-collection-api%20(2).ipynb

The requested rocket launch data was collected from SpaceX API with this URL: https://api.spacexdata.com/v4/launches/past.

*Before the extraction, a series of helpers functions were developed to get information from the columns: rocket, launchpad, payload and cores.



Data Collection - SpaceX API

1. Get Request

In [6]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```
response = requests.get(spacex_url)
```

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

2. JSON file, Pandas DataFrame

In [11]:
```
# Use json_normalize meethod to convert the json result into a dataframe
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

In [13]:
```
# 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
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
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)]
```

In [21]:
```
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}
```

Then, we need to create a Pandas data frame from the dictionary launch_dict.

In [22]:
```
# Create a data from launch_dict
df=pd.DataFrame.from_dict(launch_dict)
```

3. Filter and Missing Values

In [25]:
```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

Now that we have removed some values we should reset the FlgihtNumber column

In [26]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

In [33]:
```
# Calculate the mean value of PayloadMass column
mean=data_falcon9["PayloadMass"].mean()
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass']= data_falcon9['PayloadMass'].fillna(mean)

data_falcon9.isnull().sum()
```

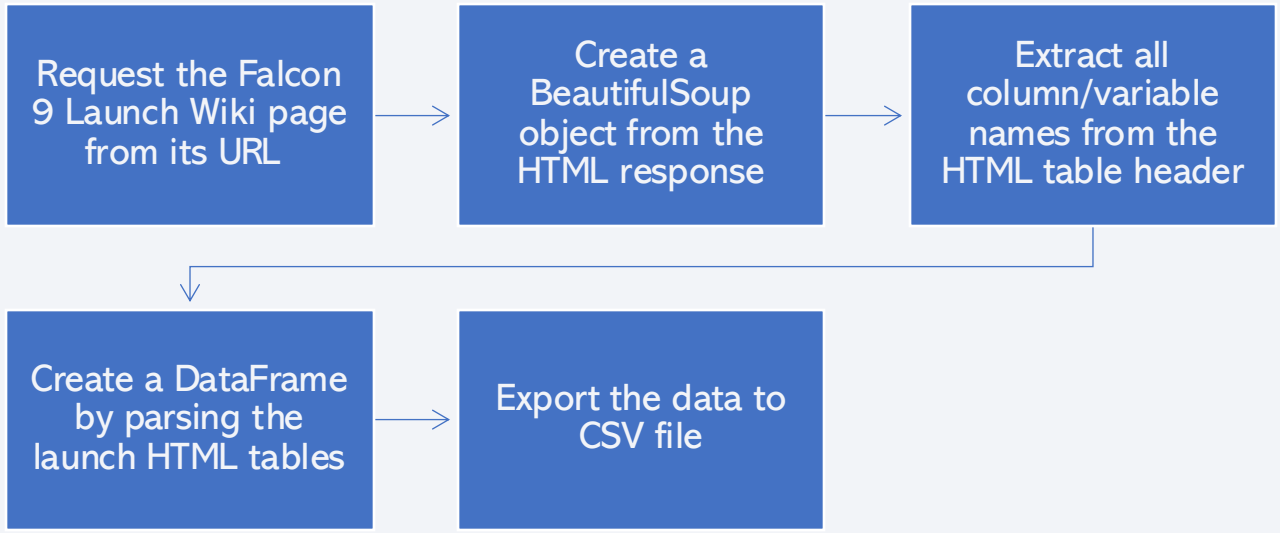
4. Export to CSV file

In [34]:
```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

github: https://github.com/marinasequinel/spacexdata/blob/main/jupyter-labs-spacex-data-collection-api%20(2).ipynb

Data Collection - Scraping

Using BeautifulSoup, we performed web scraping to collect Falcon 9 historical launch records from a Wikipedia page.



Official SpaceX Photos - Spaceflight SSO-A Mission

github: https://github.com/marinasequinel/spacexdata/blob/main/jupyter-labs-webscraping%20(2).ipynb

Data Collection - Scraping

1. Get Request and BeautifulSoup object

In [9]:
```
# use requests.get() method with the provided static_url
# assign the response to a object

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

Create a `BeautifulSoup` object from the HTML `response`

In [10]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
Soup = BeautifulSoup(response, 'html.parser')
```

2. Extract all column/variable names

In [15]:
```
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`
html_tables = Soup.find_all('table')
```

In [17]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name)>0):
        column_names.append(name)
```

3. Create a DataFrame

In [19]:
```
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']=[]
```

In [23]:
```
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:
```

*The complete code is available on the github link

4. Export to CSV file

In [26]:
```
df.to_csv('spacex_web_scraped.csv', index=False)
```

github: https://github.com/marinasequinel/spacexdata/blob/main/jupyter-labs-webscraping%20(2).ipynb

Data Wrangling

- With Pandas and Numpy, we explored the data to find patterns and determine the label for training supervised models

Using value_counts(), we calculate:

- the number of launches on each site
- the number and occurrence of each orbit
- the number and occurrence of the orbits' mission outcome

Create a label from the Outcome column

- the variable represents the outcome of each launch:
- Value 0 = failure
- Value 1 = success

Calculate the success rate and save the data

- Using .mean(), we can find the success rate
- Final step: export the data to CSV file

github: https://github.com/marinasequinel/spacexdata/blob/main/labs-jupyter-spacex-Data%20wrangling%20(1).ipynb

Data Wrangling

github: https://github.com/marinasequinel/spacexdata/blob/main/labs-jupyter-spacex-Data%20wrangling%20(1).ipynb

1. Use value_counts()

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

```
Out[6]:
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

```
In [7]:
# Apply value_counts on Orbit column
df.value_counts('Orbit')
```

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

```
In [9]:
# landing_outcomes = values on Outcome column
landing_outcomes = df.value_counts('Outcome')
landing_outcomes
```

```
Out[9]:
Outcome
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: count, dtype: int64
```

2. Create label

```
In [13]:
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

landing_class = []
for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

This variable will represent the classification variable that represents the outcome of each launch. If the value is zero, the first stage did not land successfully; one means the first stage landed Successfully

```
In [14]:
df['Class']=landing_class
df[['Class']].head(8)
```

```
Out[14]:
```

	Class
0	0
1	0
2	0
3	0
4	0
5	0
6	1
7	1

3. Calculate mean and export the data

```
In [16]:
df["Class"].mean()
```

```
Out[16]:
0.6666666666666666
```

We can now export it to a CSV for the next section,but to make the answers consistent, in the next lab we will provide data in a pre-selected date range.

```
In [17]:
df.to_csv("dataset_part_2.csv", index=False)
```

EDA with Data Visualization

With libraries such as Pandas, Matplotlib and Seaborn, the following plots were developed:

- Scatterplot with the variables Flight Number vs Payload, Flight Number vs Launch Site, Payload vs Launch Site, Flight Number vs Orbit Type, Payload vs Orbit Type.
  - These plots determine the correlation between two variables, so we can easily visualize how one affects the other.
- Bar chart with the relationship between Success Rate and Orbit.
  - These charts compare different categories or groups - here we used the groupby() method.
- Line chart to show the launch success trend by year.
  - Line charts are perfect to display trends over time.

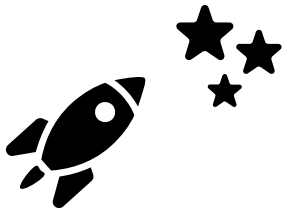
github: https://github.com/marinasequinel/spacexdata/blob/main/jupyter-labs-eda-dataviz%20(1).ipynb

EDA with SQL

github: https://github.com/marinasequinel/spacexdata/blob/main/jupyter-labs-eda-sql-coursera_sqllite%20(1).ipynb

We executed SQL queries to find the following information:

1. Unique launch sites in the space mission
2. 5 records where launch sites begin with 'CCA'
3. The total payload mass carried by boosters launched by NASA (CRS)
4. Average payload mass carried by the booster version F9 v1.1
5. The date of the first successful landing outcome in ground pad
6. The names of the boosters that successfully landed in drone ship and have payload mass between 4000 and 6000kg
7. The total number of successful and failure mission outcomes
8. The names of the booster versions that carried the maximum payload mass
9. The records with the month names, failure landing outcomes in drone ship, booster versions and launch sites in 2015.
10. Rank the count of landing outcomes between 2010-06-04 and 2017-03-20 in descending order



Build an Interactive Map with Folium

To build the maps using Folium, we first found the coordinates (latitude and longitude) for each launch site. The following steps were:

- With **folium.Circle** and **Marker**, we created a circle showing the location and name of the NASA centers, and through iteration we added the same thing for each launch site on the map.
  - These make it easier for us to visualize the places of the launches.

- Then we built a **MarkerCluster** object to group the launches in each site, and created a DataFrame called marker_color to indicate if the landing was successful or not.
  - This is a great way to simplify a map containing many markers having the same coordinate.

- For each launch result, we added a **folium.Marker** to the **marker_cluster**.
  - Now we can easily browse the map and check the landing outcomes for each cluster or individually.

• Later we calculated the distances between a launch site to its proximities (coast, cities, roads and highways) using a math method and adding the mouse position to get the coordinates on the map. Then we created a **folium.Marker** and a **folium.PolyLine** to show those distances.

github: https://github.com/marinasequinel/spacexdata/blob/main/lab_jupyter_launch_site_location%20(1).ipynb

Build a Dashboard with Plotly Dash

We created a Launch Records Dashboard with Plotly Dash following these steps:

First a **Drop-Down Input Component was added**, so that we could select the graphs for each launch site or for all sites

We added a **callback function** to render a pie chart that analyzes the success launch rate based on the selected site dropdown

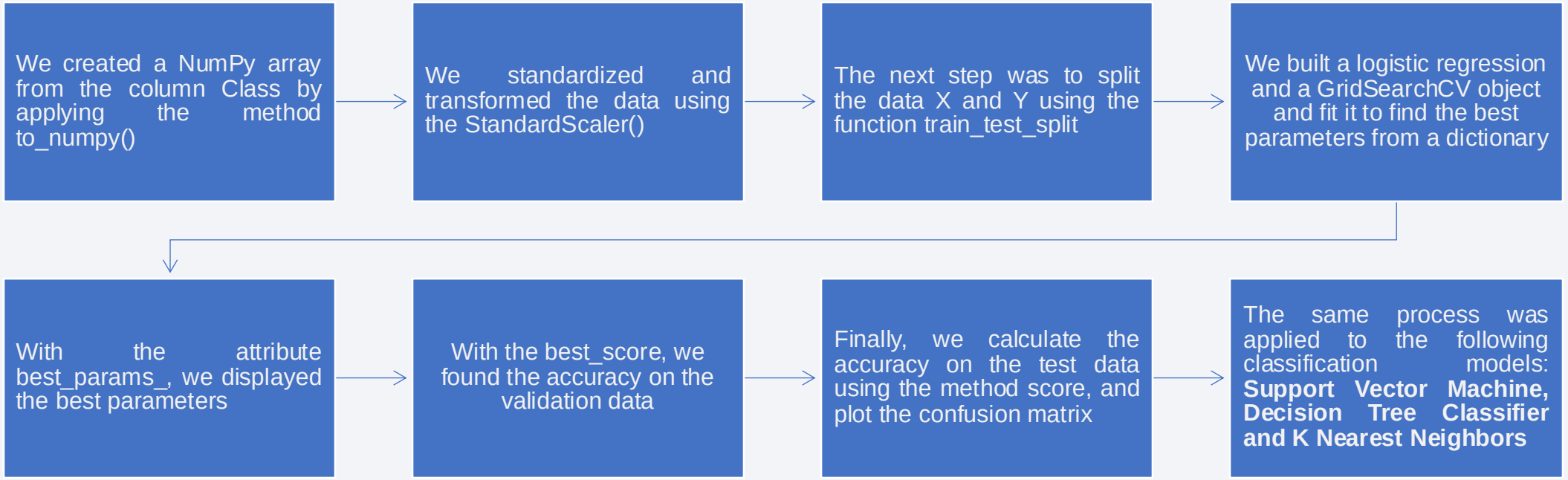
A **range slider** was built to select the payload mass (kg)

We added another **callback function** to create a scatter plot presenting the correlation between the payload mass and the success rate

github: https://github.com/marinasequinel/spacexdata/blob/main/spacex_dash_app.py
graphs: https://github.com/marinasequinel/spacexdata/blob/main/SpaceXDash.png

Predictive Analysis (Classification)

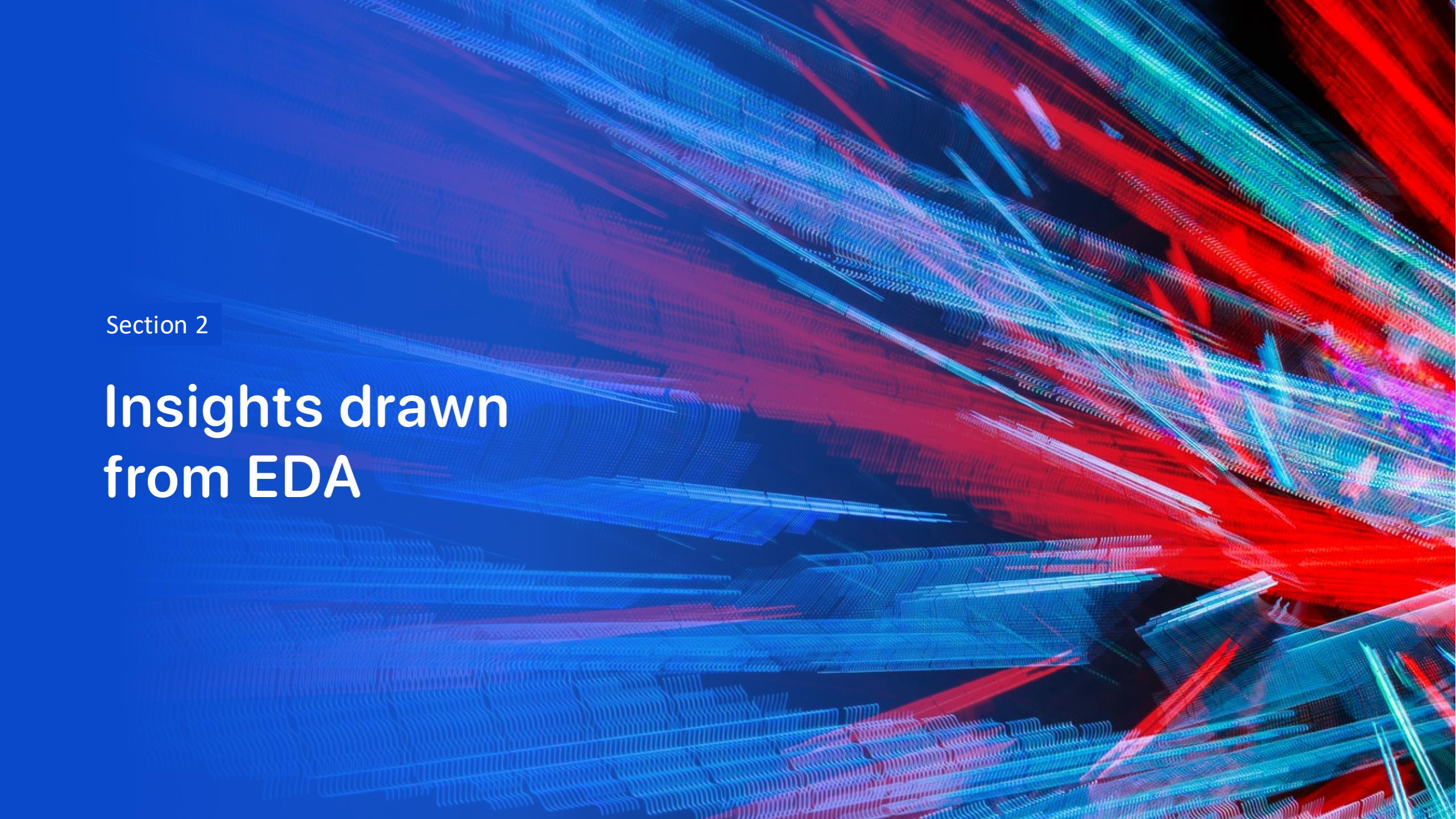
To find the best parameters for different Machine Learning methods, the first step was to load the data frame. Then:



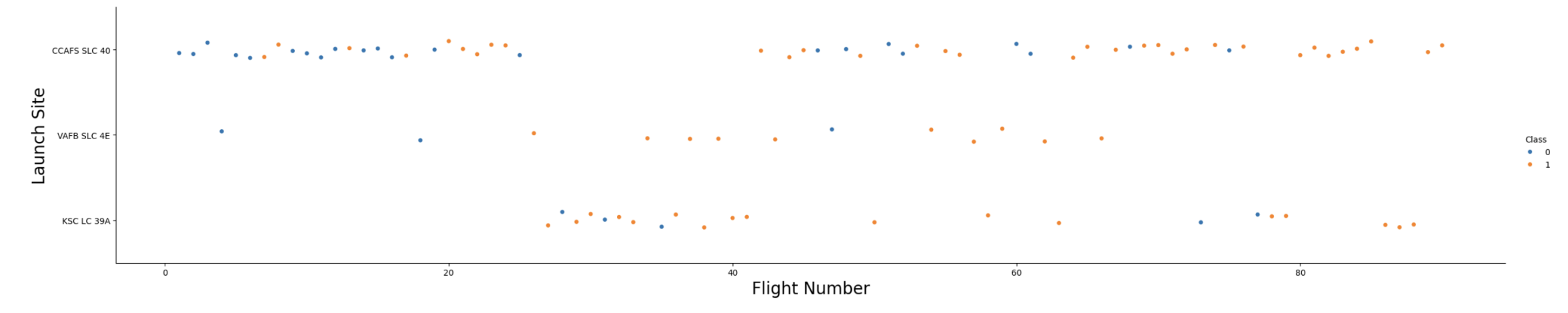
github: https://github.com/marinasequinel/spacexdata/blob/main/SpaceX_Machine%20Learning%20Prediction_Part_5.ipynb

Results

- As the flight number increases, the first stage is more likely to land successfully. Also, the more massive the payload, the less likely the first stage will return.
- The Orbits with the most success rate are ES-L1, GEO, HEO and SSO.
- The success rate kept increasing between 2013 and 2017, with stability in 2014.
- The Decision Tree was the model with the highest accuracy (0.87), while the remaining models had a lower result (0.83).
- All launch sites are in proximity to the Equator Line and to the coast, as they are relatively close to railways and highways, because of recourses' transportation.
- On the other hand, the areas chosen for the tests tend to keep a certain distance from the cities, in order to ensure people's safety.

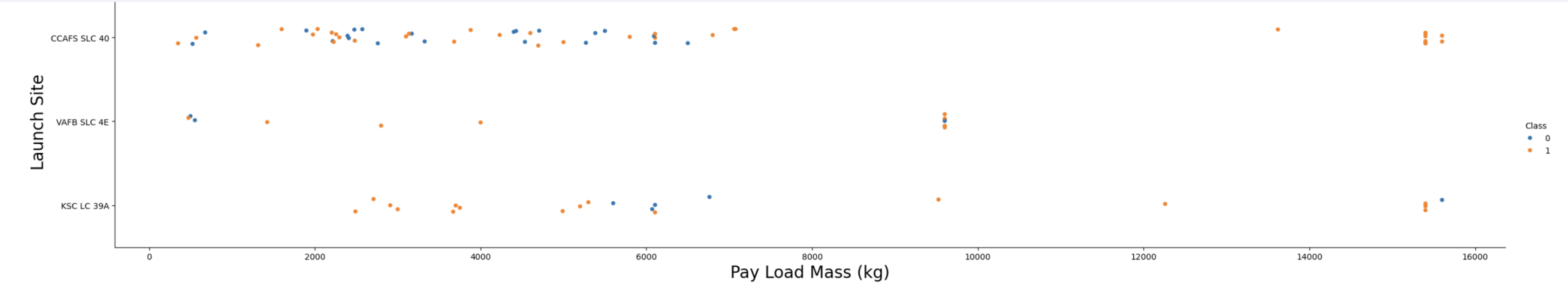

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



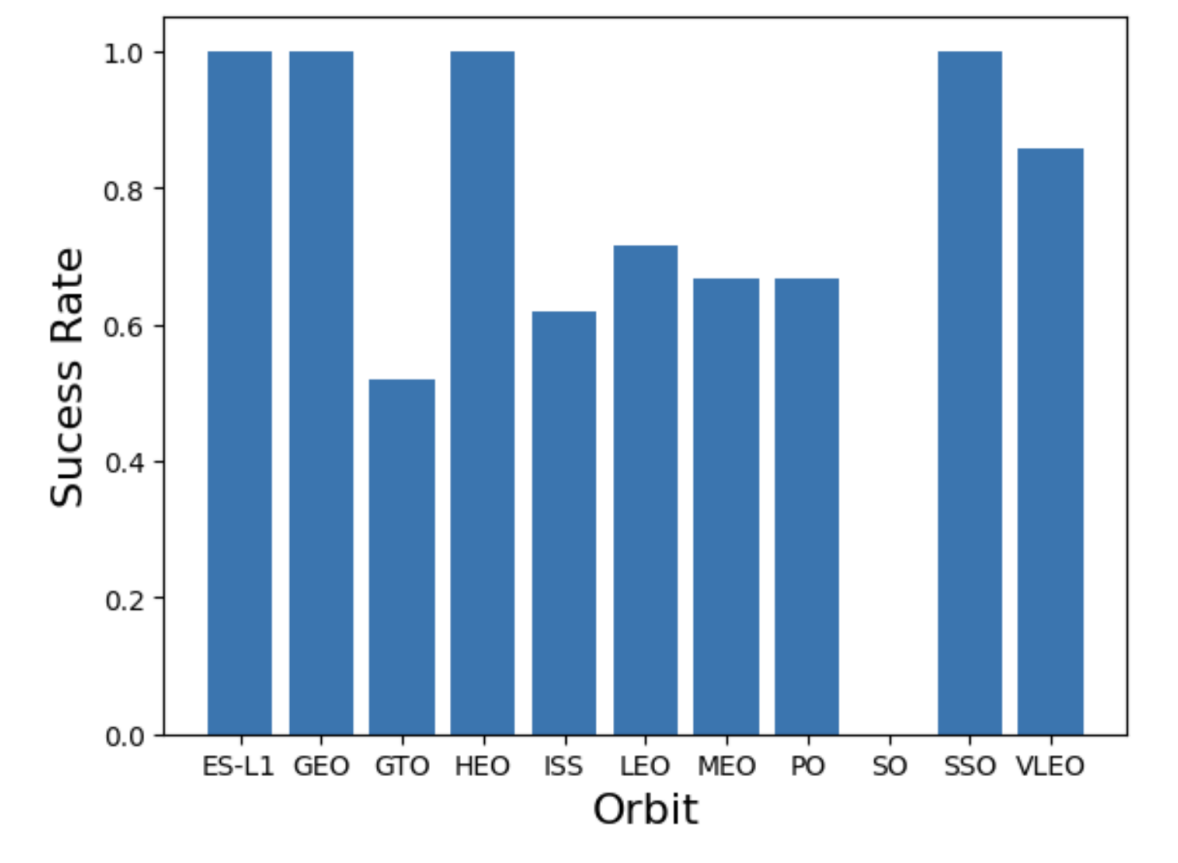
- In general, in every launch site, it is possible to visualize that the bigger the number of flights, the better the launches' success rate.

Payload vs. Launch Site



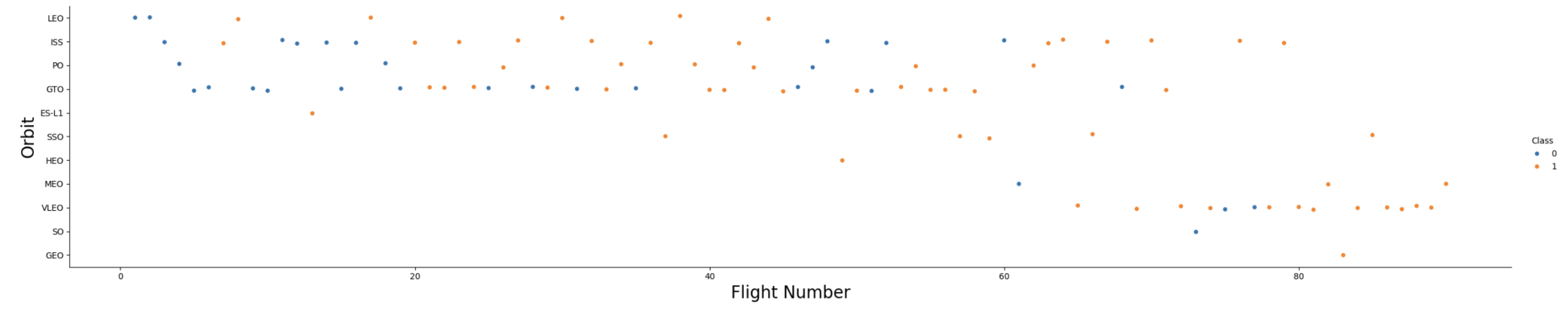
- In the VAFB SLC launch site there are no rockets launched with payload mass greater than 10000 kg.
- The majority of rockets with lower payload mass was launched in the CCAFS SLC site.

Success Rate vs. Orbit Type



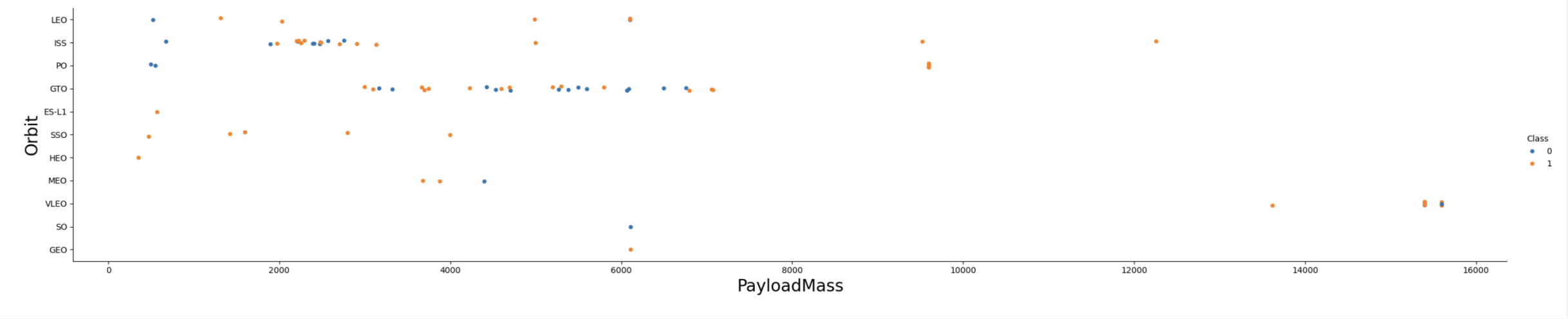
- The Orbits with the most success rate are ES-L1, GEO, HEO and SSO.

Flight Number vs. Orbit Type



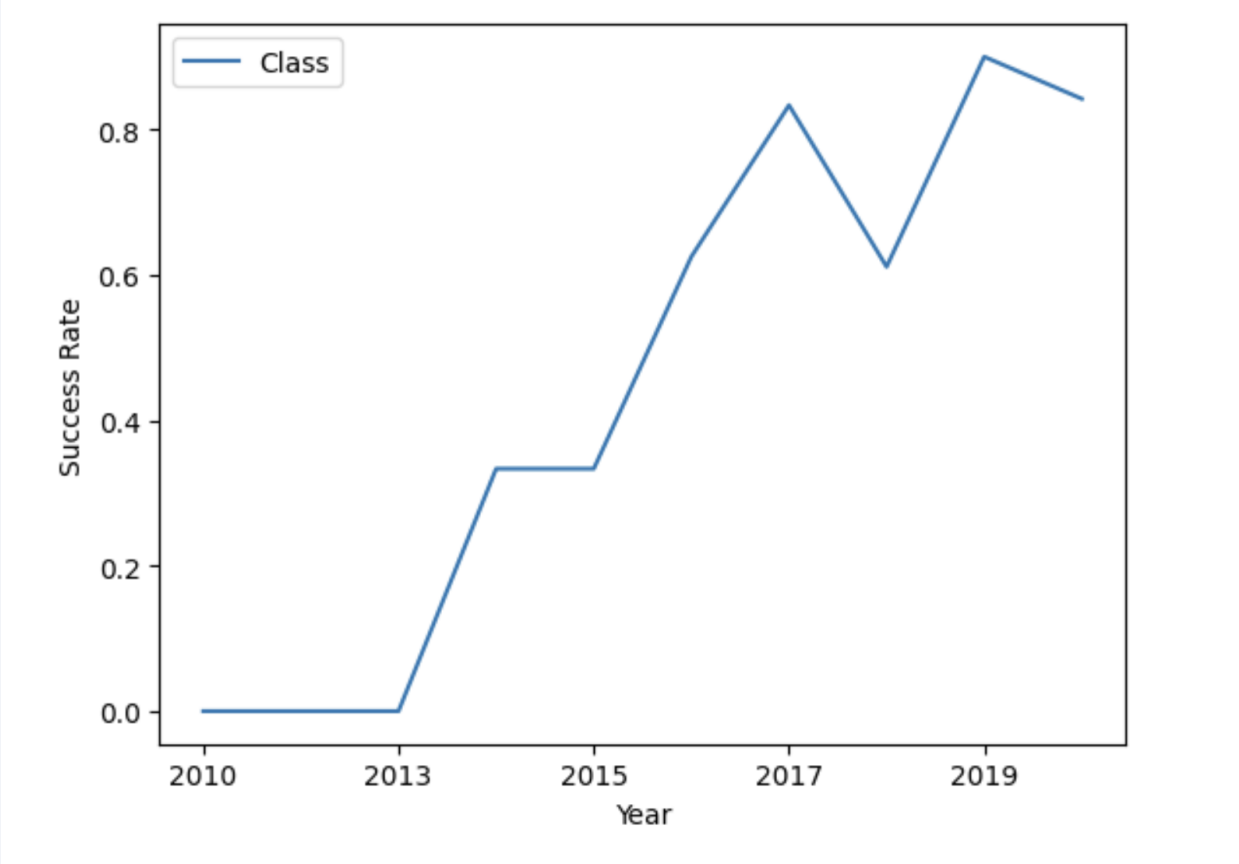
- In the LEO orbit the success rate appears to be related to the number of flights.
- On the other hand, that seems not to be the case in the GTO orbit.

Payload vs. Orbit Type



- When the payloads are heavy, the success rate seems to be higher in the Polar, LEO and ISS Orbits.
- In the GTO, however, it is not possible to distinguish this well as both positive and negative landing are present.

Launch Success Yearly Trend



- The success rate kept increasing between 2013 and 2017, with stability in 2014.

All Launch Site Names

- Display the names of the unique launch sites in the space mission:
  - %sql SELECT DISTINCT "Launch_Site" FROM SPACEXTABLE;

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

Launch Site Names that Begin with 'CCA'

- Display 5 records where launch sites begin with the string 'CCA':
  - %sql SELECT * FROM SPACEXTABLE WHERE "Launch_Site" LIKE "CCA%" LIMIT 5;

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

- Display the total payload mass carried by boosters launched by NASA (CRS):
  - %sql SELECT SUM("PAYLOAD_MASS__KG_") FROM SPACEXTABLE WHERE "Customer" = "NASA (CRS)";

SUM("PAYLOAD_MASS__KG_")
45596

Average Payload Mass by F9 v1.1

- Display average payload mass carried by the booster version F9 v1.1:
  - %sql SELECT AVG("PAYLOAD_MASS__KG_")FROM SPACEXTABLE WHERE "Booster_Version" = "F9 v1.1";

AVG("PAYLOAD_MASS__KG_")
2928.4

First Successful Ground Landing Date

- List the date when the first successful landing outcome in ground pad was achieved:
  - %sql SELECT MIN("Date") FROM SPACEXTABLE WHERE "Landing_Outcome" = "Success (ground pad)";

MIN("Date")
2015-12-22

*The first successful landing outcome in ground pad was achieved on December 22, 2015.

Successful Drone Ship Landing with Payload between 4000 and 6000

- List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000:
  - %sql SELECT "Booster_Version", "PAYLOAD_MASS__KG_" FROM SPACEXTABLE WHERE "Landing_Outcome" = "Success (drone ship)" AND "PAYLOAD_MASS__KG_" BETWEEN 4000 AND 6000;

Booster_Version	PAYLOAD_MASS__KG_
F9 FT B1022	4696
F9 FT B1026	4600
F9 FT B1021.2	5300
F9 FT B1031.2	5200

Total Number of Successful and Failure Mission Outcomes

- List the total number of successful and failure mission outcomes:
  - %sql SELECT Mission_Outcome, COUNT(*) FROM SPACEXTABLE GROUP BY Mission_Outcome;

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

Total Success: 100
Total Failure: 1

Boosters Carried Maximum Payload

- List the names of the booster versions which have carried the maximum payload mass:
  - %sql SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);

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

2015 Launch Records

- List the records with the month names, failure landing outcomes in drone ship, booster versions and launch sites in 2015:
  - %sql SELECT substr(Date, 6,2), Booster_Version, Launch_Site, Landing_Outcome FROM SPACEXTABLE WHERE Landing_Outcome = "Failure (drone ship)" AND substr(Date,0,5)='2015';

substr(Date, 6,2)	Booster_Version	Launch_Site	Landing_Outcome
01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	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 between 2010-06-04 and 2017-03-20 in descending order:
    - %sql SELECT COUNT(Landing_Outcome), Landing_Outcome FROM SPACEXTABLE WHERE Date BETWEEN "2010-06-04" AND "2017-03-20" GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC;

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

Section 3

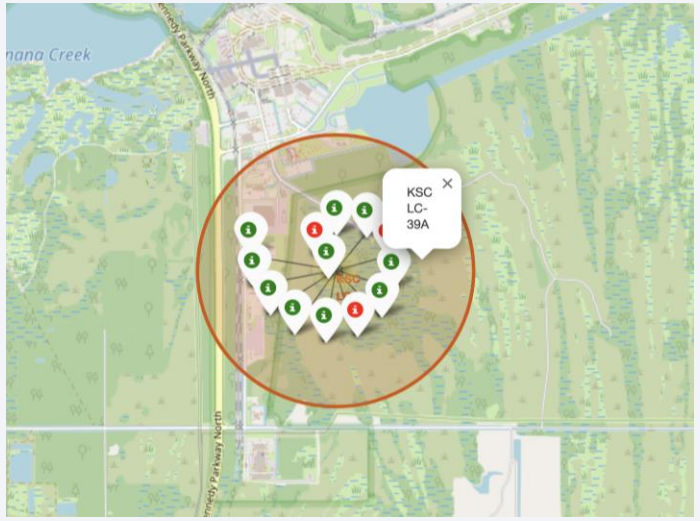
Launch Sites Proximities Analysis

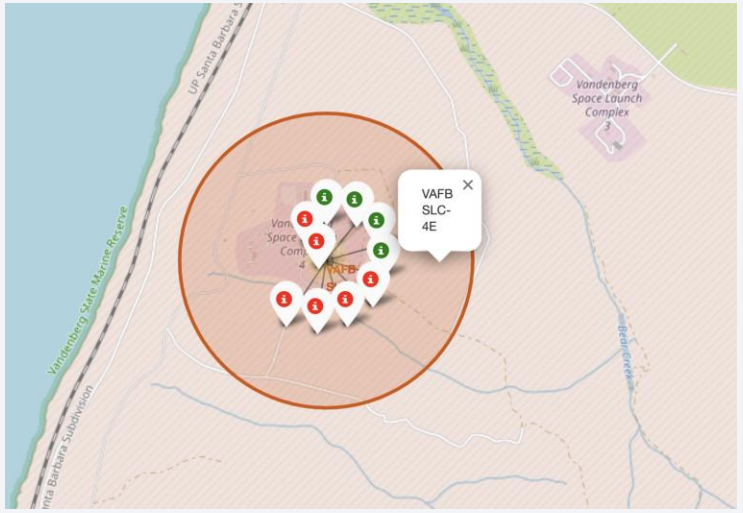
All launch sites on a Folium Map

- First we marked each launch site on the map with its name, according to its coordinates (latitude and longitude).
- Three of them are located in Florida and one in California.

Success and failed launches

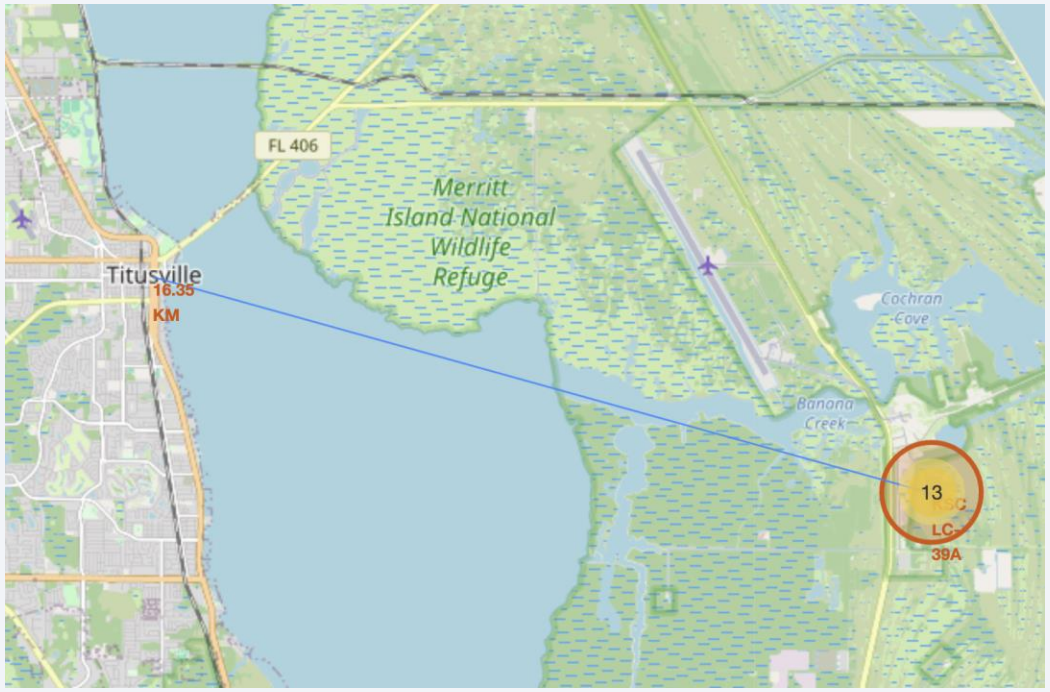
- The following images show each launch site cluster with colored labels: a red marker represents a failed launch and a green marker indicates a successful one.



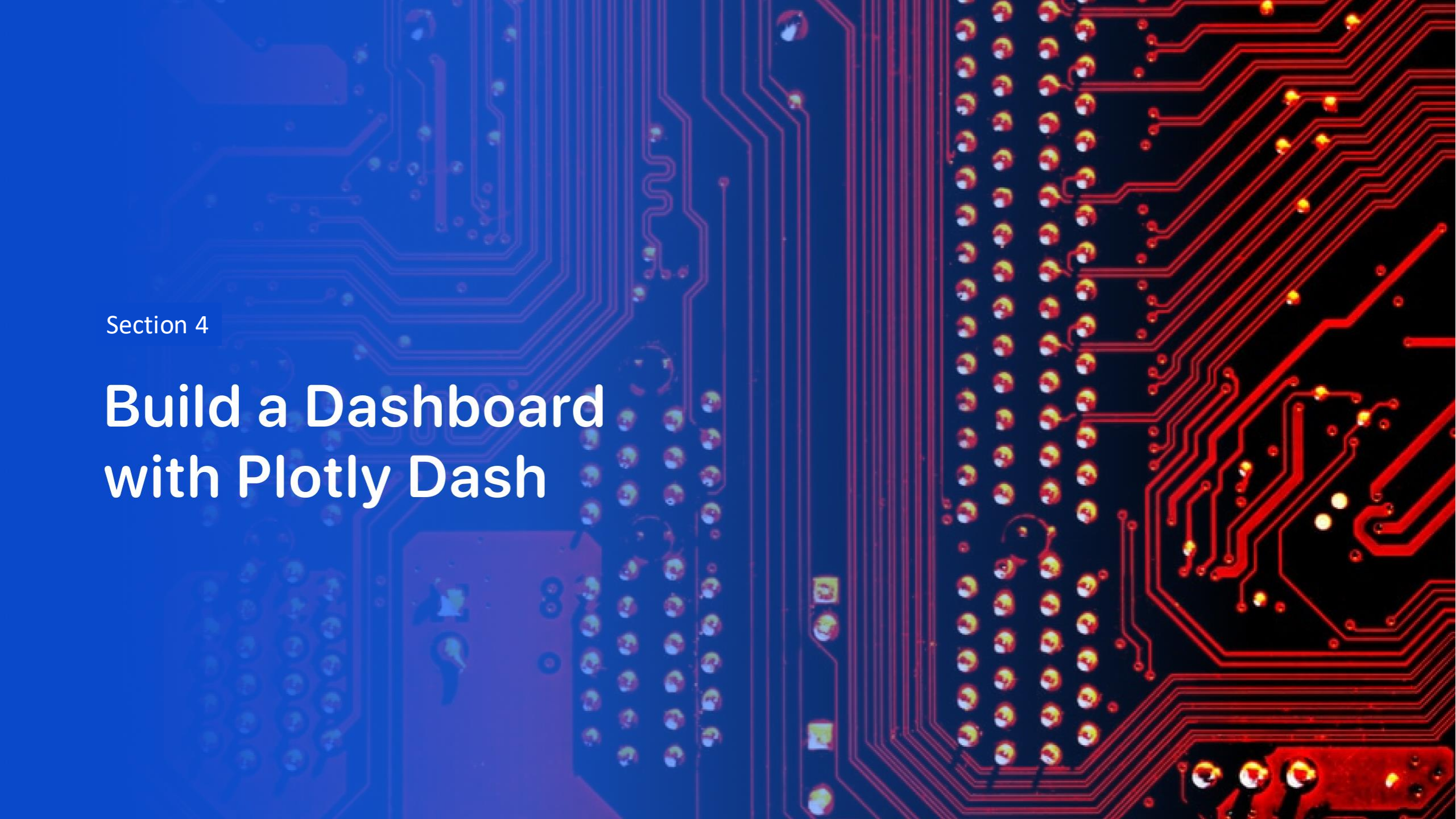


Launch Sites and its Proximities

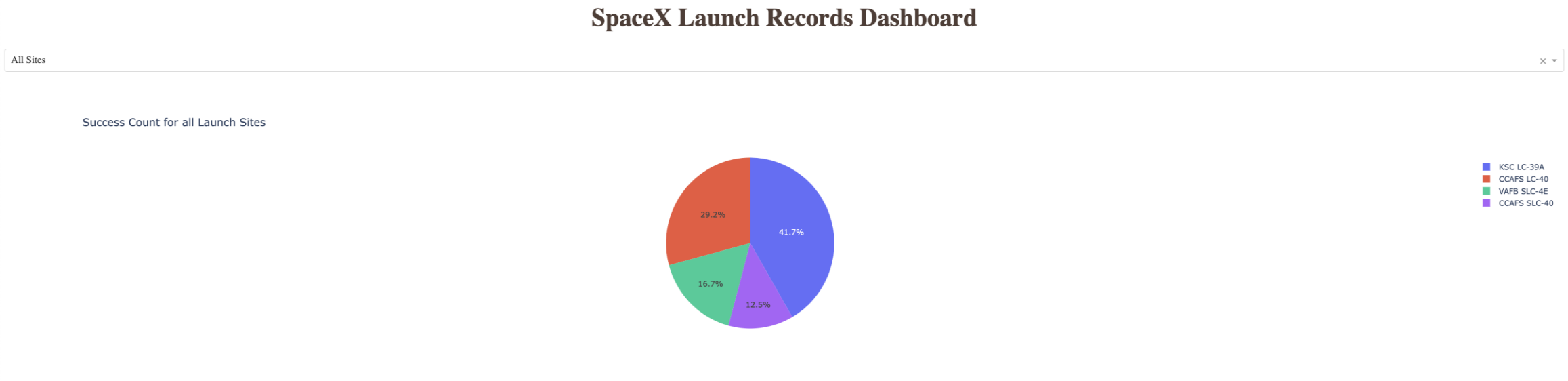
- In the first image, the blue line shows the distance between the launch site CCAFS SLC-40 and the coast line (0.91 km), and the Samuel C Philips Parkway Road (0.67 km).
- In the second image, the blue line indicates the distance between the site KSC LC-39A and the Titusville City (16.35 km).



- All launch sites are in proximity to the coast, as they are relatively close to railways and highways (because of recourses' transportation). On the other hand, the areas chosen for the tests tend to keep a certain distance from the cities, in order to ensure people's safety.

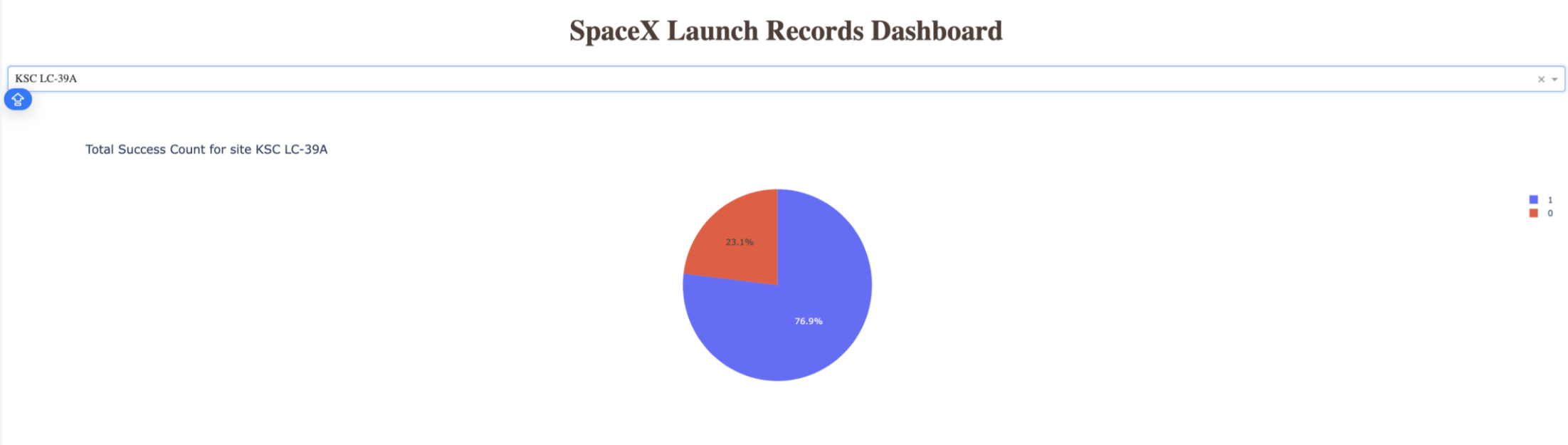

Section 4
Build a Dashboard with Plotly Dash

Pie Chart - Success Count for all Launch Sites



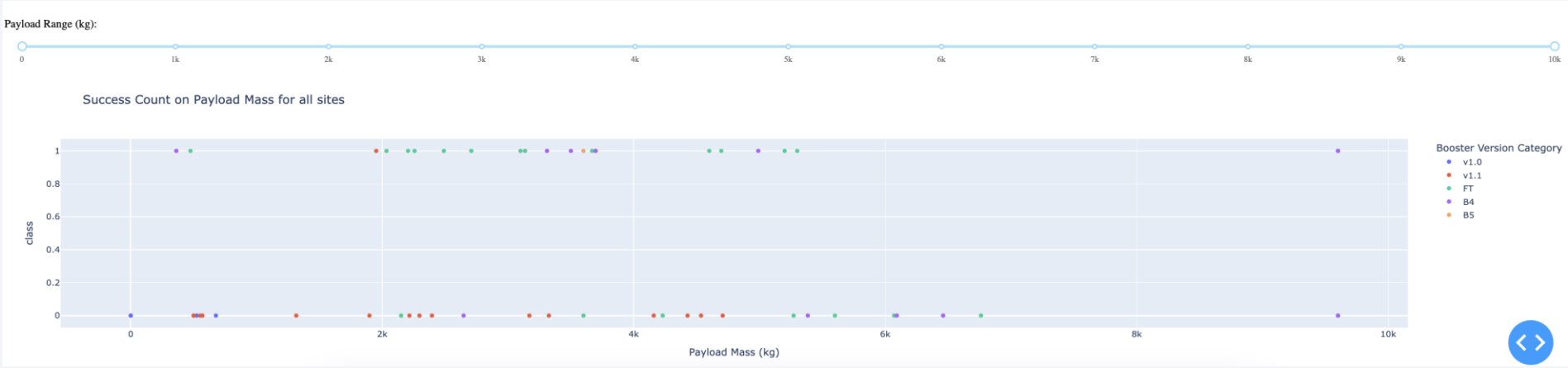
- This is the output when you select the "All Sites" option.
- KSC LC-39A is the launch site with the highest success ratio.

Pie Chart - Launch Site with the highest success ratio



- The pie chart shows the launch success count for the KSC LC-39A site, which has the highest positive landings' ratio (76.9%).

Scatter plot - Success Count on Payload Mass for all sites

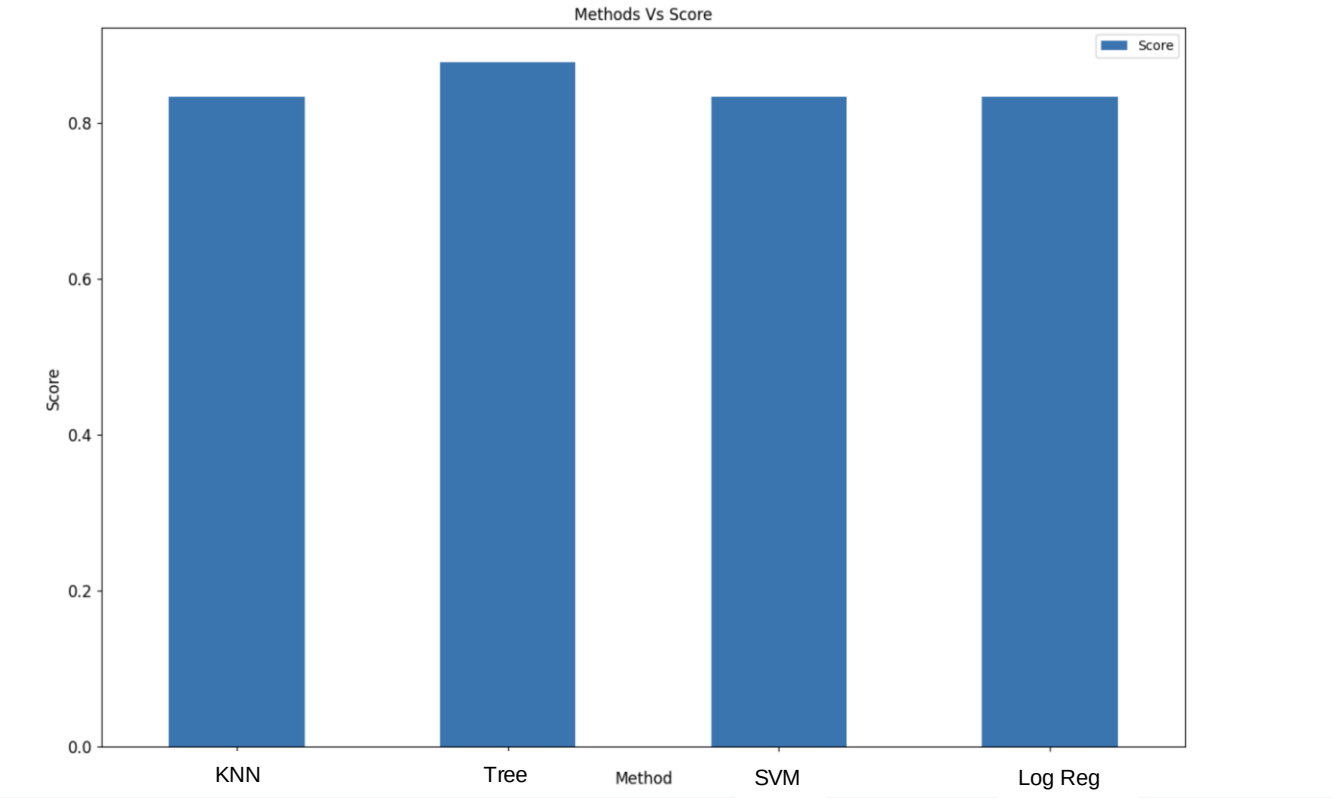


- The payload rage with the highest launch success rate is between 2000-4000kg, while the one with the lowest ratio is between 6000-8000kg.
- The booster versions that have the highest launch success rate are the FT and B4.

Section 5

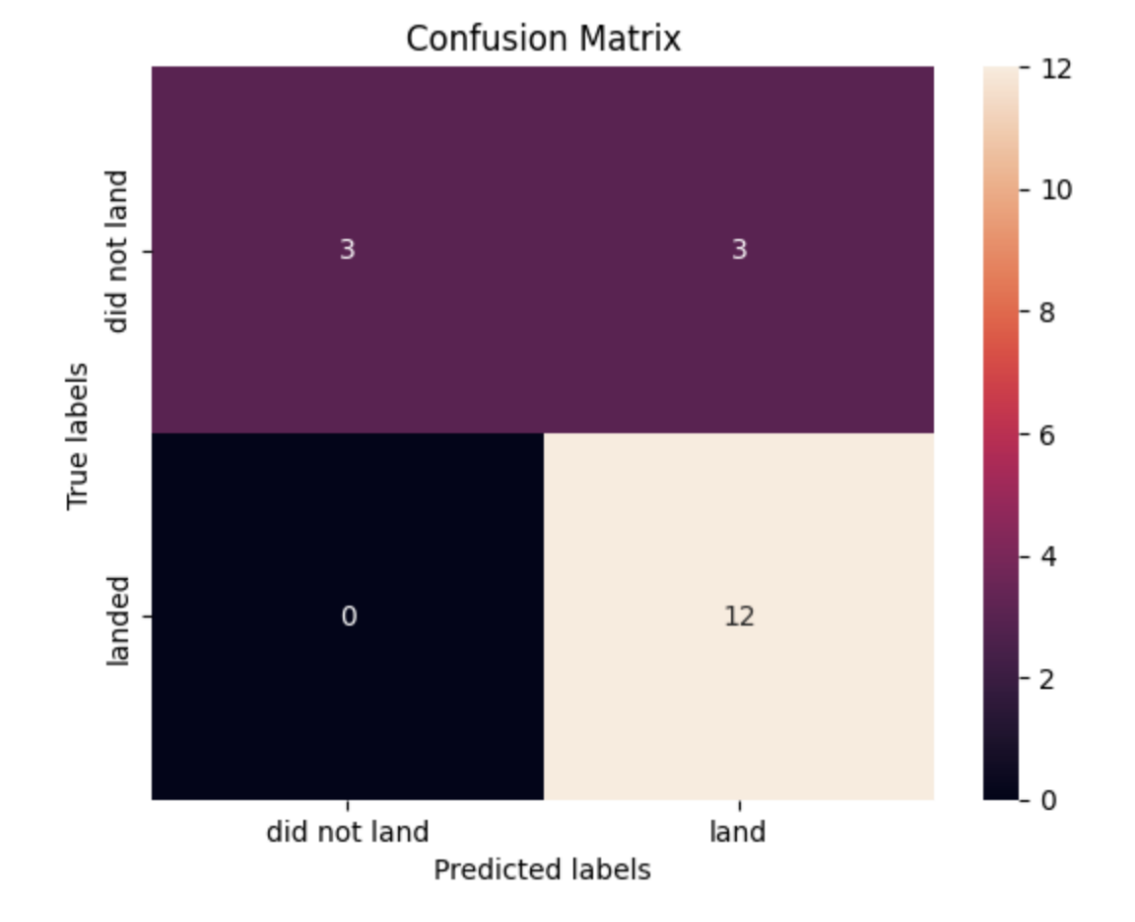
Predictive Analysis (Classification)

Classification Accuracy



- The Decision Tree was the model with the highest accuracy (0.87), while the remaining models had a lower result (0.83).

Confusion Matrix



- The Confusion Matrix for the four methods is basically the same. It has:

TRUE NEGATIVE = 3

FALSE NEGATIVE = 3

TRUE POSITIVE = 12

FALSE NEGATIVE = 0

Conclusions

- As the flight number increases, the first stage is more likely to land successfully. Also, the more massive the payload, the less likely the first stage will return.
- The Orbits with the most success rate are ES-L1, GEO, HEO and SSO.
- The launch site with the highest success ratio is KSC LC-39A.
- The booster versions that have the highest launch success rate are the FT and B4.
- The success rate kept increasing between 2013 and 2017, with stability in 2014.
- The Decision Tree was the model with the highest accuracy (0.87), while the remaining models had a lower result (0.83).
- All launch sites are in proximity to the Equator Line and to the coast, as they are relatively close to railways and highways, because of recourses' transportation.
- On the other hand, the areas chosen for the tests tend to keep a certain distance from the cities, in order to ensure people's safety.

Thank you!