IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Francisco Lopes
October 24th 2023

Outline

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

Executive Summary

- The project began with data collection through web scraping and APIs, followed by essential data wrangling to clean and structure the dataset. Exploratory data analysis (EDA) allowed me to understand data patterns, and then we created informative dashboards for easy data exploration. Finally, a machine learning model was developed using different methods to predict the success of the first stage landing. This holistic approach, blending data acquisition, EDA, and predictive modeling, enabled us to define a method or more that could successfully predict a landing.
- All the methods used (Logistic Regression, Support Vector Machines, Decision Tree and KNN) presented similar results for out train-test dataset.

Introduction

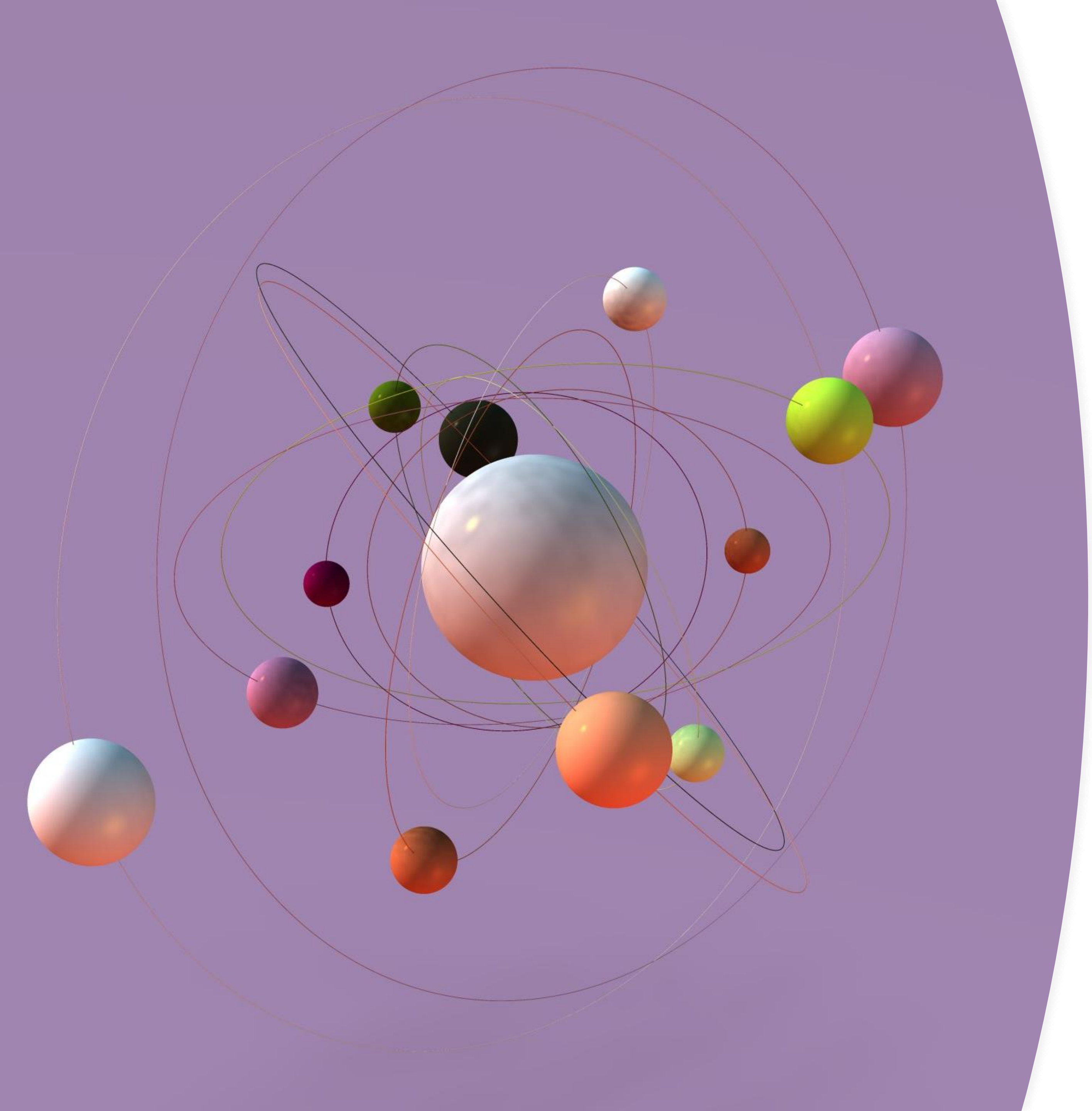
- This project focused on the exploration of SpaceX's Falcon 9 data more specifically, Falcon 9 missions and their information. SpaceX's Falcon 9 has been a transformative force in space exploration. So, our goal is to leverage a data-driven analysis to tackle some questions that could shape the future of space travel.
- The focus of this presentation is to provide context and background on what was used to study this dataset. As well as showing the methods that were used to build a machine learning pipeline to predict if the first stage will land, given the data from SpaceX's Falcon 9.

Section 1

Methodology

Methodology

- Executive Summary
- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - How to build, tune, evaluate classification models



Data Collection

- SpaceX provides an API that enables you to gather information regarding rockets, launches, and launchpads. This data is collected in JSON format using the .json() function and subsequently converted into a Pandas dataframe using .json_normalize().
- The data was then cleaned, and missing data was taken care of by filling it with the mean values of each column.
- Similarly, pertinent details about launches were accessible on Wikipedia, and this information was extracted using Beautiful Soup.

Data Collection – SpaceX API

- With a GET request to the SpaceX API the data was collected. The Requested data was cleaned and then we performed data wrangling and formatting.
- GitHub URL of the completed SpaceX API calls notebook: https://github.com/Aegiel/ds_capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

From the `rocket` column we would like to learn the booster name.

In [2]:
```python
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

In [3]:
```python
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [4]:
```python
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

From `cores` we would like to learn the outcome of the landing, the type of the landing, number of flights with that core, whether gridfins were used, wheter the core is reused, wheter legs were used, the landing pad used, the block of the core which is a number used to seperate version of cores, the number of times this specific core has been reused, and the serial of the core.

In [5]:
```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Now let's start requesting rocket launch data from SpaceX API with the following URL:

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

Data Collection - Scraping

- BeautifulSoup was the selected web scrapping used to webscrap Falcon 9's launch records. The table was then parsed and converted into a Pandas Dataframe.
- GitHub URL of the completed web scraping notebook: https://github.com/Aegiel/ds_capstone/blob/main/jupyter-labs-webscraping.ipynb

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

In [5]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

In [6]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

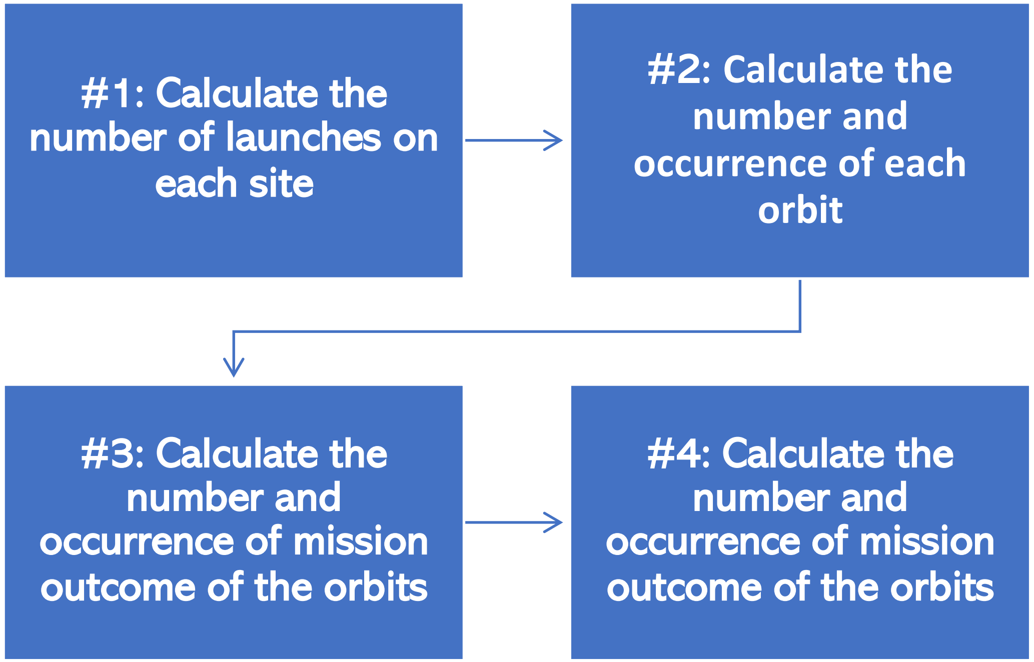
Print the page title to verify if the `BeautifulSoup` object was created properly

In [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

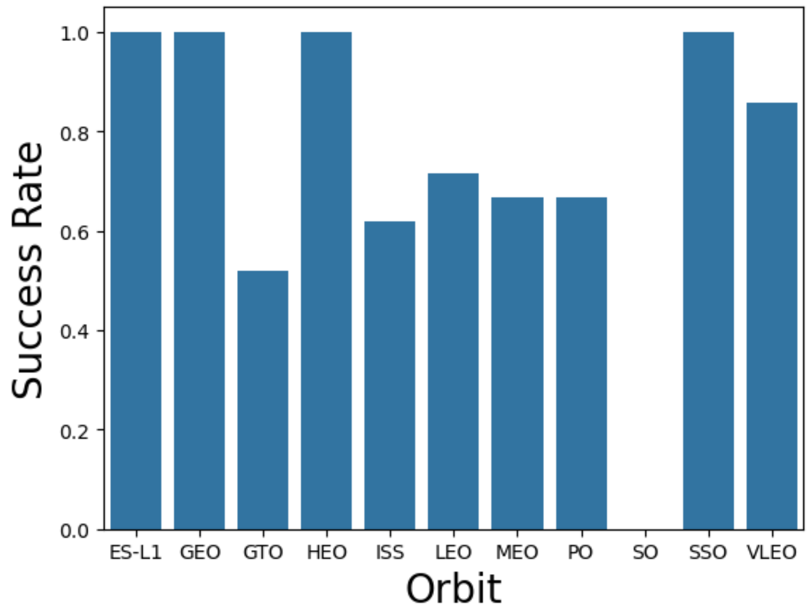
Data Wrangling

- We started by performing Exploratory data analysis (EDA) as our first step, and then we determined the training labels.
- With a better understanding of the data, we calculated the number of launches at each site, as well as the number and occurrences of each orbit.
- Finally, the Landing Outcome label was created using the outcome column.
- Add the GitHub URL of your completed data wrangling related notebooks: https://github.com/Aegiel/ds_capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb



EDA with Data Visualization

- We explored the data by creating several plots. This allowed us to study several variables and better understand their relationship. The variables were:
  - Flight Number and Launch Site;
  - Payload Mass and Launch Site;
  - Success Rate of each Orbit Type;
  - Flight Number and Orbit Type;
  - Payload Mass and Orbit Type;
  - And finally, the Launch Success yearly trend.
- Add the GitHub URL of your completed EDA with data visualization notebook: https://github.com/Aegiel/ds_capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb



Orbits ES-L1, GEO, HEO and SSO have the higher Success Rate.

EDA with SQL

- The SpaceX dataset was loaded into PostgreSQL in a Jupyter Notebook.
- We applied EDA with SQL to get information and insights from the data. The queries that were used are the following:
  - The names of each unique launch site in the space mission;
  - The total payload mass carried by boosters launched by NASA (CRS);
  - The average payload mass carried by booster's version F9 v1.1;
  - The total number of successful and failed mission outcomes;
  - The failed landing outcomes in drone ship, their booster version and respective launch site names.

- Add the GitHub URL of your completed EDA with SQL notebook: https://github.com/Aegiel/ds_capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- We annotated all the launch sites on the map and incorporated various map elements like markers, circles, and lines to visually denote the success or failure of launches at each site using the Folium map library.
- We categorized the launch outcomes into two classes: 0 represented failures, while 1 indicated successes.
- By utilizing color-coded marker clusters, we pinpointed which launch sites exhibited a comparatively higher success rate than the others.
- We computed the distances between a launch site and its surrounding features
- These objects were all added with the goal of answering questions like:
  - How close are launch sites to railways, highways, and coastlines?
  - Do launch sites maintain a certain distance from populated areas?
- Add the GitHub URL of your completed interactive map with Folium map: https://github.com/Aegiel/ds_capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We plotted a pie chart to visualize the launch success counts for all or each Launch Site.
- We plotted a Scatter Plot to see how the Payload may be correlated with the mission outcomes for a selected Launch Site.
- We added these plots and interactions to a Dashboard with the goal of using it to analyze SpaceX launch data.
- Add the GitHub URL of your completed Plotly Dash lab: https://github.com/Aegiel/ds_capstone/blob/main/spacex_dash_app.py

Build a Dashboard with Plotly Dash

- We imported the dataset and divided into training and testing subsets;
- Next, we used various machine learning methods and fine-tuned their hyperparameters utilizing GridSearchCV. The methods were:
  - Logistic Regression
  - SVM
  - Classification Trees
  - KNN
- Our main evaluation metric was accuracy, and we enhanced the model through algorithm optimization and by choosing the selected hyperparameters. Ultimately, we identified the most effective classification model.
- Add the GitHub URL of your completed predictive analysis lab: https://github.com/Aegiel/ds_capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

EXPLORATORY DATA ANALYSIS RESULTS

INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS

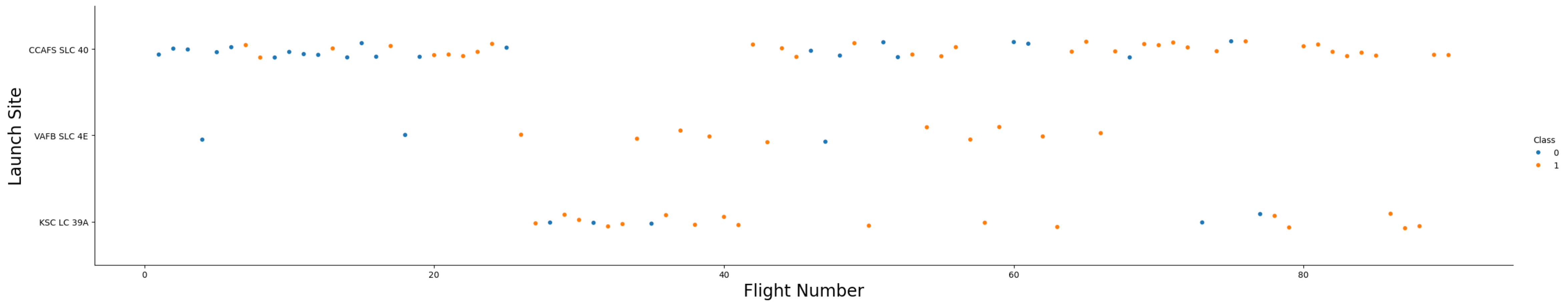
PREDICTIVE ANALYSIS RESULTS

Section 2

Insights drawn from EDA

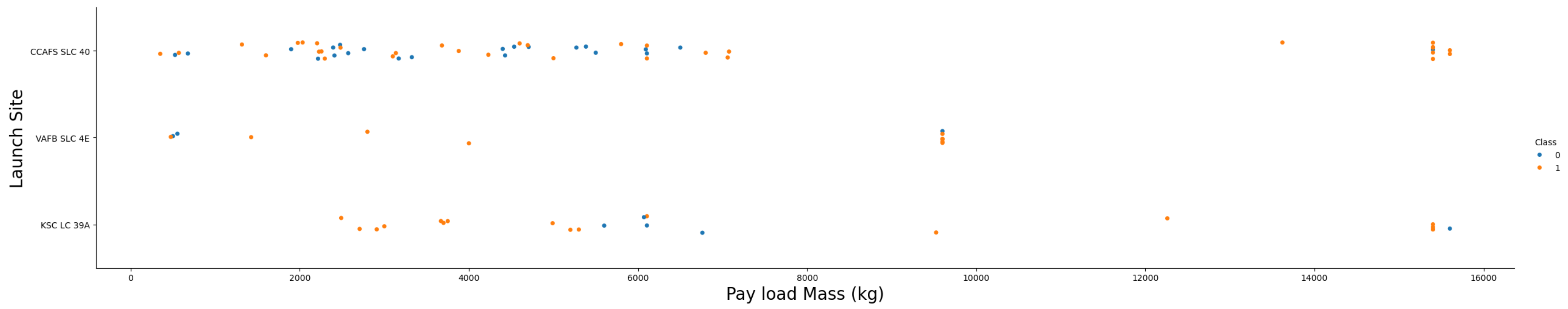
Flight Number vs. Launch Site

- With this scatter plot we can see that the larger the Flight Number, the greater the success rate at any Launch Site.



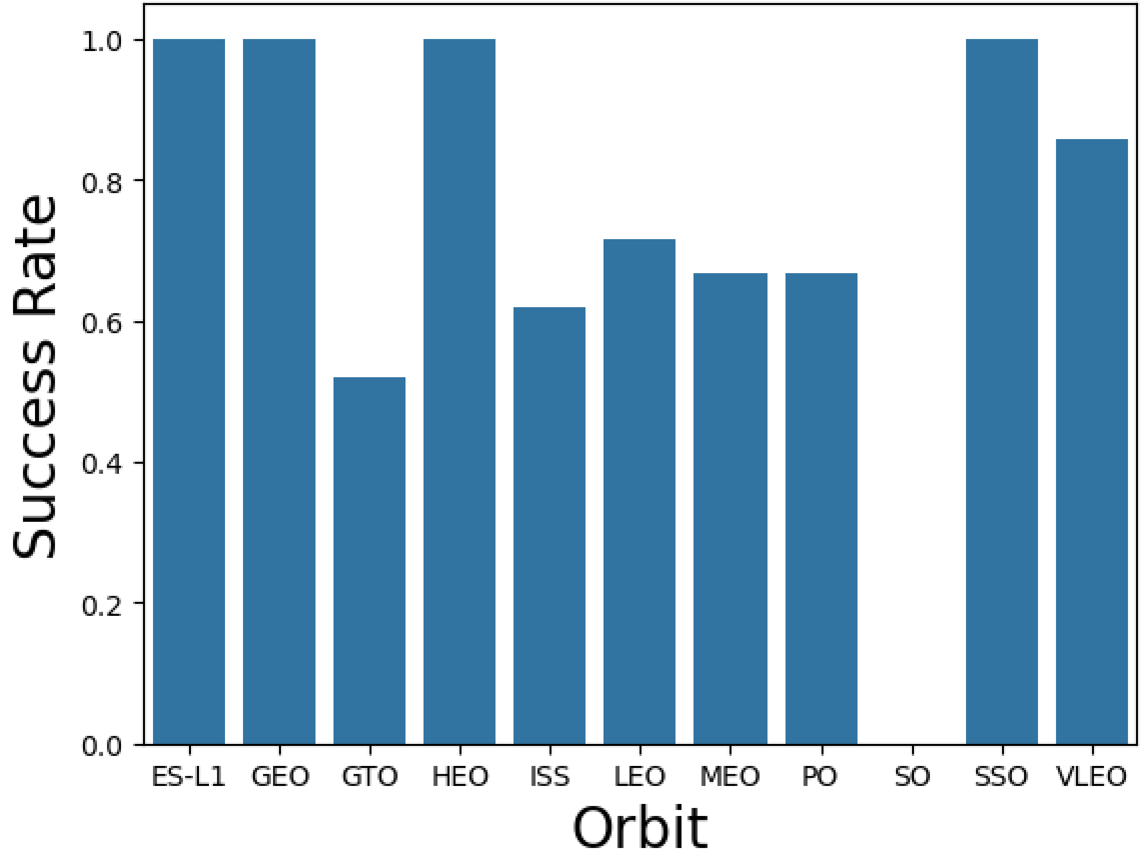
Payload vs. Launch Site

- With this scatter plot we can see that for the VAFB-SLC Launch Site there are no rockets launched with a heavy Payload Mass (greater than 10000)



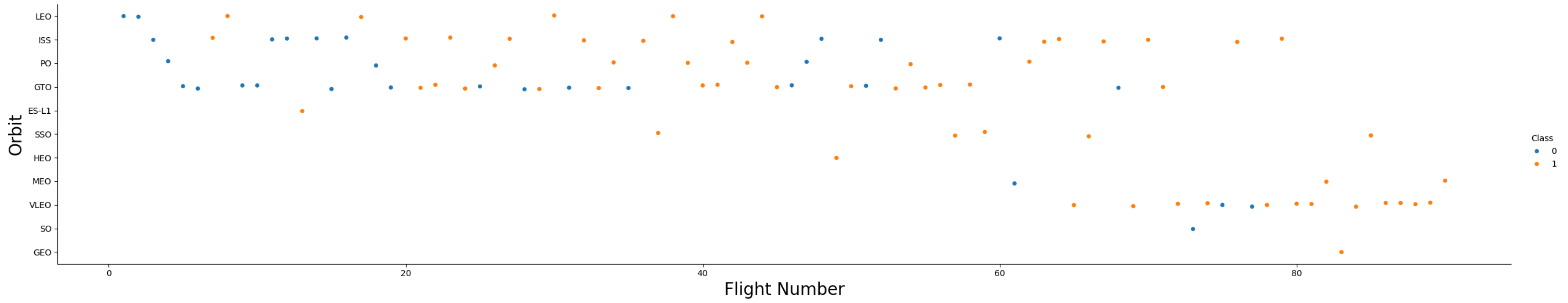
Success Rate vs. Orbit Type

- With this bar chart we can see that Orbits ES-L1, GEO, HEO and SSO have the higher Success Rate.



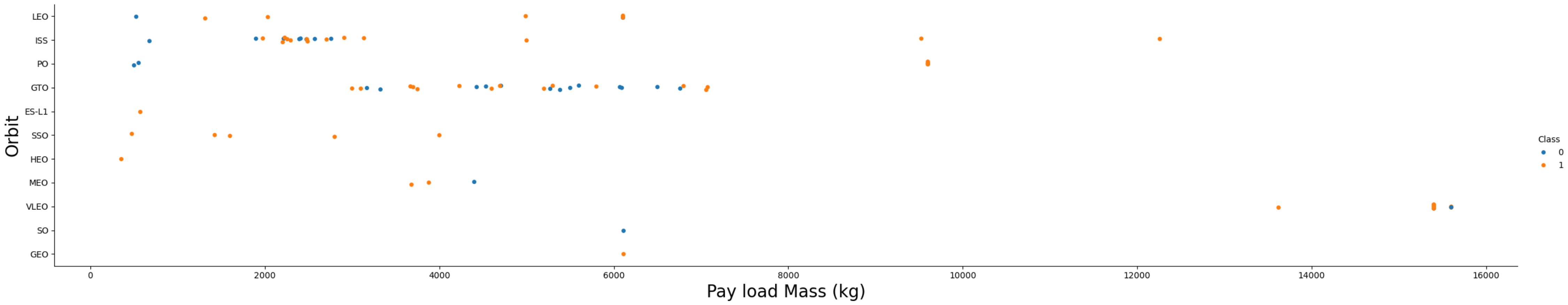
Flight Number vs. Orbit Type

- With this scatter plot we can see that the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



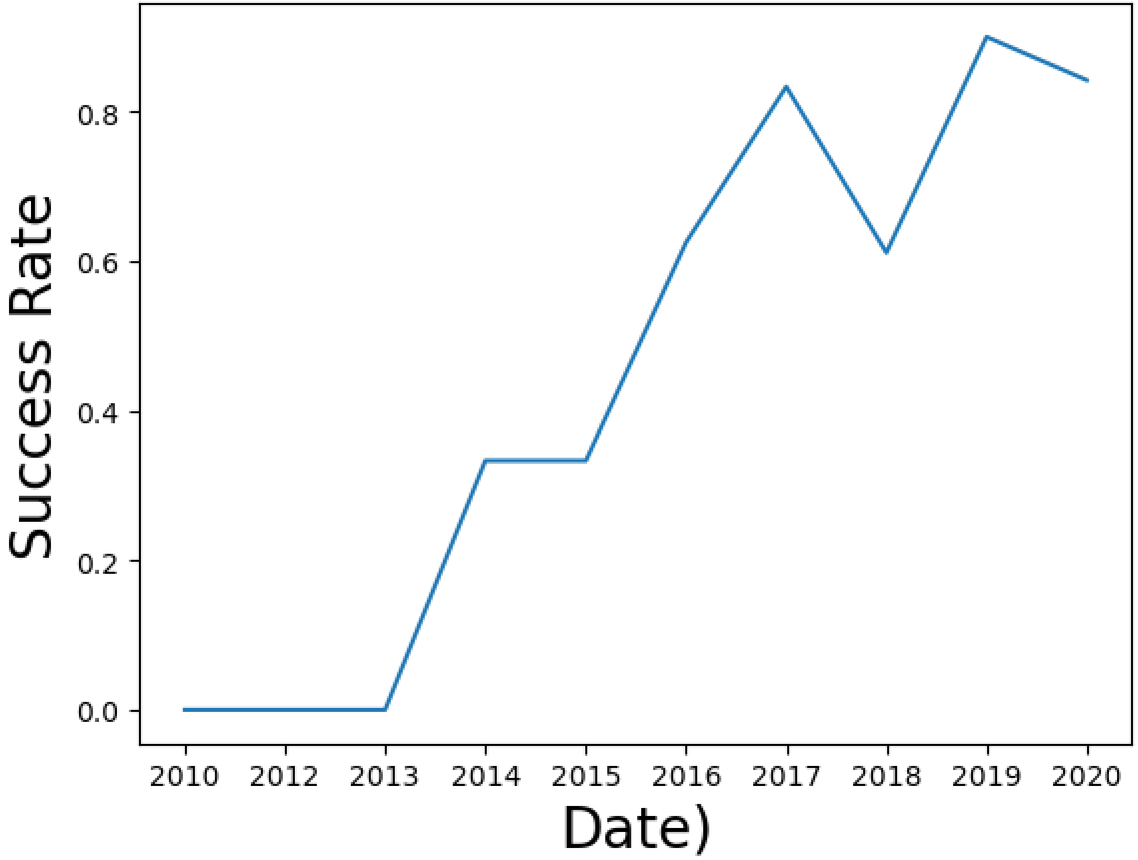
Payload vs. Orbit Type

- With this scatter plot we can see that with heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS.
- However, for GTO we cannot distinguish this well since both positive landing rate and negative landing (unsuccessful mission) are both seen.



Launch Success Yearly Trend

- With this line chart we can see that that the success rate since 2013 kept increasing till 2020



All Launch Site Names

- The names are shown in the table below;
- These were found using the DISTINCT function which shows only unique launch sites names.

Task 1

Display the names of the unique launch sites in the space mission

In [13]:
```
%sql SELECT DISTINCT("Launch_Site") FROM SPACEXTABLE
```

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

Out[13]:

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

Launch Site Names Begin with 'CCA'

- The 5 records where launch sites begin with `CCA` are shown below;
- Used the key word LIKE to catch only the names that start with the string 'CCA' followed by % meaning that it can have any other characters after.

Task 2

Display 5 records where launch sites begin with the string 'CCA'

In [21]:

```
%sql SELECT * FROM SPACEXTABLE WHERE (Launch_Site LIKE 'CCA%')  LIMIT 5
```

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

Out[21]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-08-12	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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- The total payload carried by boosters from NASA is calculated and shown below;
- By using the function SUM we sum all the values present in the Payload Mass column, where the name starts with "NADA (CRS)".

Task 3

Display the total payload mass carried by boosters launched by NASA (CRS)

In [23]:
```
%sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE (Customer LIKE 'NASA (CRS)')
```

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

Out[23]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- The average payload mass carried by booster version F9 v1.1 is calculated and show below;
- We use the AVG function to average all the values present in the Payload Mass column, where the booster version starts with “F9 v1.1”.

Task 4

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

In [24]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTABLE WHERE (Booster_Version LIKE 'F9 v1.1')
```

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

Out[24]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- The date of the first successful landing outcome on ground pad is shown below;
- By using the min function we can retrieve the lowest/first "Date" when a Successful landing was achieved in ground pad.

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

In [27]:

```
%sql SELECT MIN("DATE") FROM SPACEXTABLE WHERE (Landing_Outcome LIKE 'Success (ground pad)')
```

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

Out[27]:

MIN("DATE")
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- The list of the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 is shown below;
- We used the WHERE clause to filter for boosters which have successfully landed on drone ship and applied the AND condition to determine successful landing with payload mass greater than 4000 but less than 6000 (which means BETWEEN, so we used that key word)

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [31]:
```
%sql SELECT Booster_Version FROM SPACEXTABLE WHERE (PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000) AND (Landing_Outcome LIKE 'Succ
```

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

Out[31]:

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

Total Number of Successful and Failure Mission Outcomes

- The total number of successful and failure mission outcomes is seen below, summing all the different types of Successes we have 61 and of Failures we have 10;
- We use the COUNT function to retrieve the count of all the rows where the column Landing Outcome starts with Success or Failure.

Task 7

List the total number of successful and failure mission outcomes

In [ ]:
```
## CASE WHEN Landing_Outcome LIKE 'Success%' THEN 'Success' WHEN Landing_Outcome LIKE 'Failure%' THEN 'Failure' ELSE 'Other
```

In [37]:
```
%sql SELECT Landing_Outcome, COUNT(*) AS Count FROM SPACEXTABLE WHERE Landing_Outcome LIKE 'Success%' OR Landing_Outcome LIK
```

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

Out[37]:

Landing_Outcome	Count
Failure	3
Failure (drone ship)	5
Failure (parachute)	2
Success	38
Success (drone ship)	14
Success (ground pad)	9

Boosters Carried Maximum Payload

- The list of the names of the booster which have carried the maximum payload mass is shown below;
- We used a subquery with the MAX function to retrieve the maximum payload in dataset and then we selected the booster versions that had carried this weight.

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
In [40]: %sql SELECT Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTABLE);

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

Out[40]:

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

2015 Launch Records

- The list of the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015, is shown below;
- As suggested, we used substr to get the month and then selected the rows where the landing outcome failed in a drone ship.

Task 9

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.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [45]:
```
%sql SELECT substr("DATE", 6, 2) AS Month, Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTABLE WHERE substr("DATI
```

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

Out[45]:

Month	Landing_Outcome	Booster_Version	Launch_Site
10	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

- The rank of 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 is shown below;
- We GROUPBY the Landing Outcome and COUNT how many of them were between the requested dates.

Task 10

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.

In [47]:
```
%sql SELECT Landing_Outcome, COUNT(*) AS Outcome FROM SPACEXTABLE WHERE Date >= '2010-06-04' AND Date <= '2017-03-20' GROUP
```

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

Out[47]:

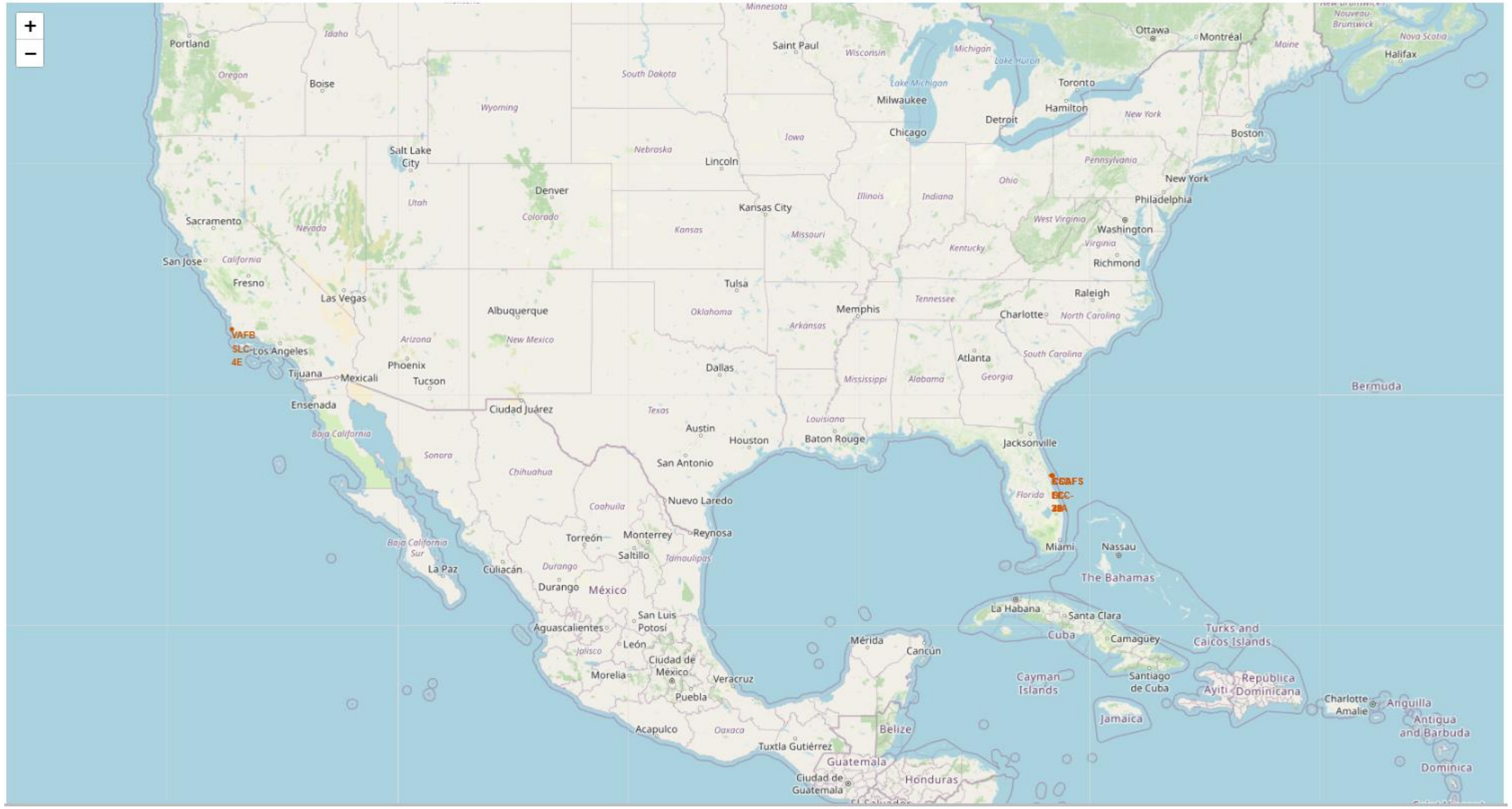
Landing_Outcome	Outcome
No attempt	10
Success (ground pad)	5
Success (drone ship)	5
Failure (drone ship)	5
Controlled (ocean)	3
Uncontrolled (ocean)	2
Precluded (drone ship)	1
Failure (parachute)	1

Section 3

Launch Sites Proximities Analysis

Launch Site Locations

- From this plot we can see that the Launch Sites :
  - Aren’t in proximity to the Equator Line, main VAFB SLC-4E;
  - Are near to the coast;
  - Seem to only be located in the US region.

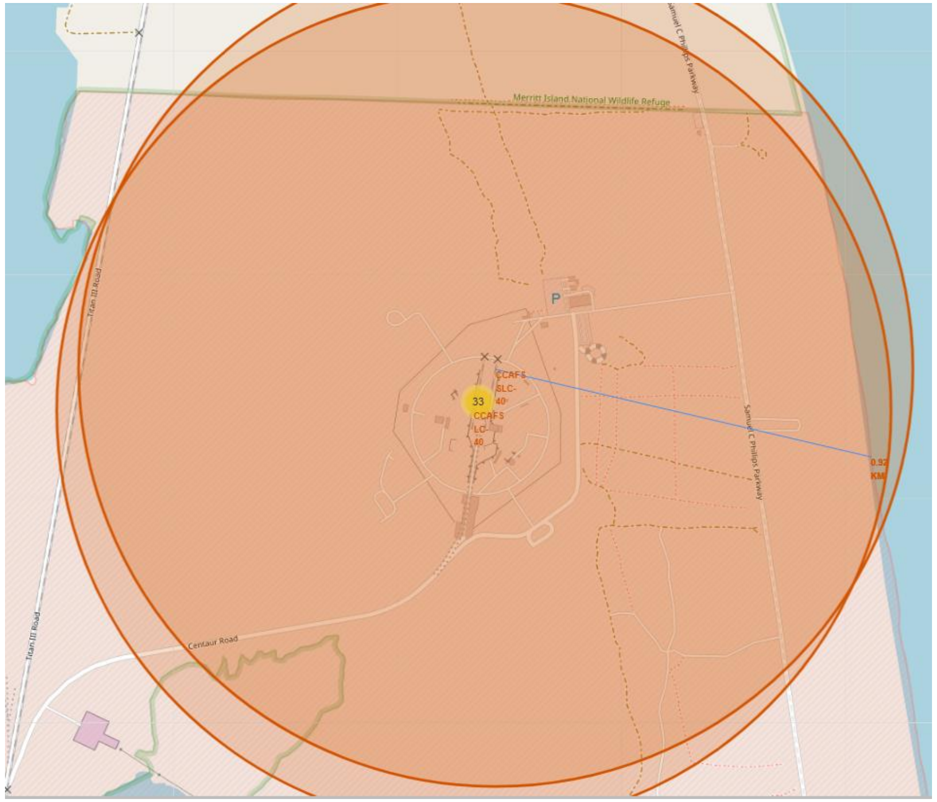


Successful/failed launches for each site on the map

- On the left we can see the California Launch Sites and on the right we can see the Florida ones.
- The red marker represents a failed launch, while the green one represents a successful one.

Launch Site Distance to Coastline

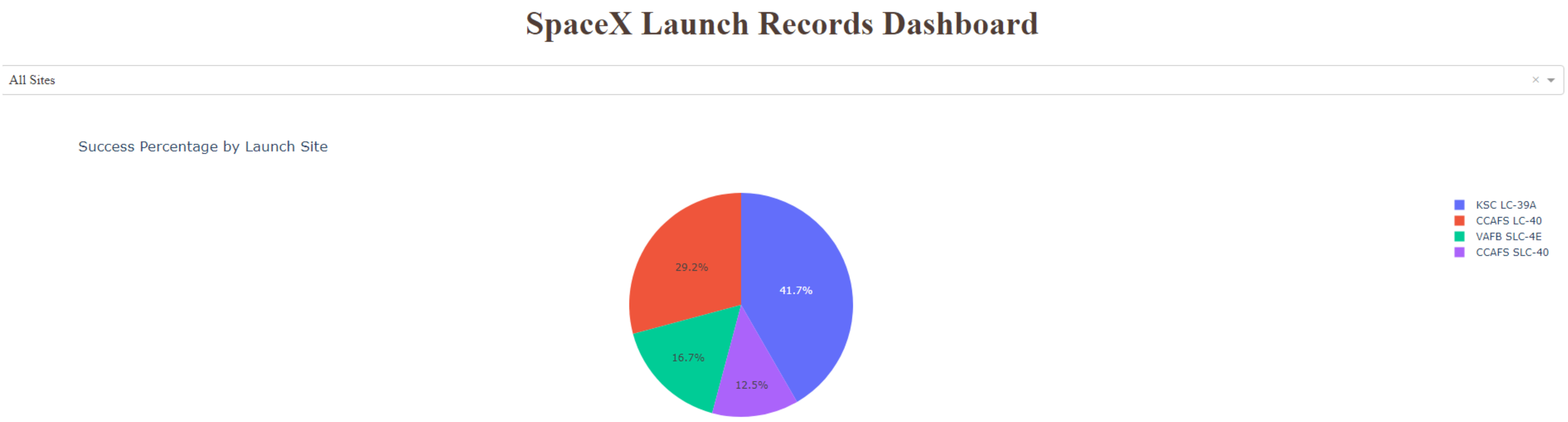
- With this type of plots we can see the distance from a Launch Site to any landmark. In this case we are showing the distance to the Coastline.
- Launch Sites try to be as isolated from cities and as close to the coast as possible. They all have close proximity to railways and highways, most likely to receive the needed materials to work.



Section 4

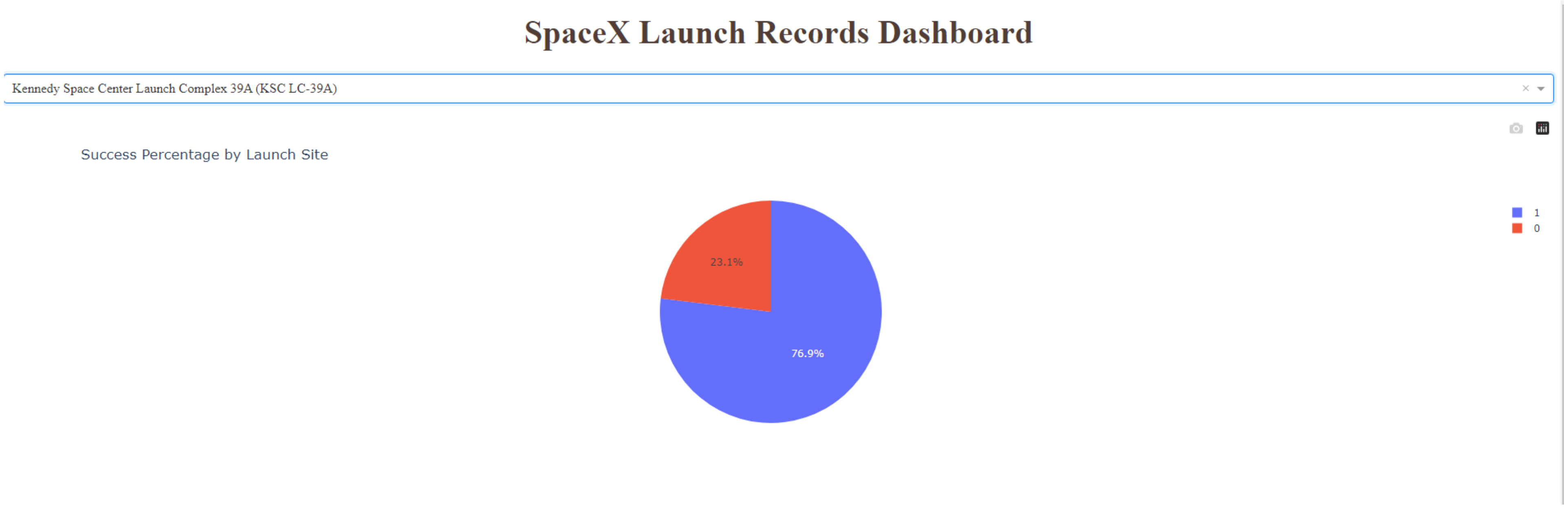
Build a Dashboard with Plotly Dash

Pie Chart of launch success count for all sites



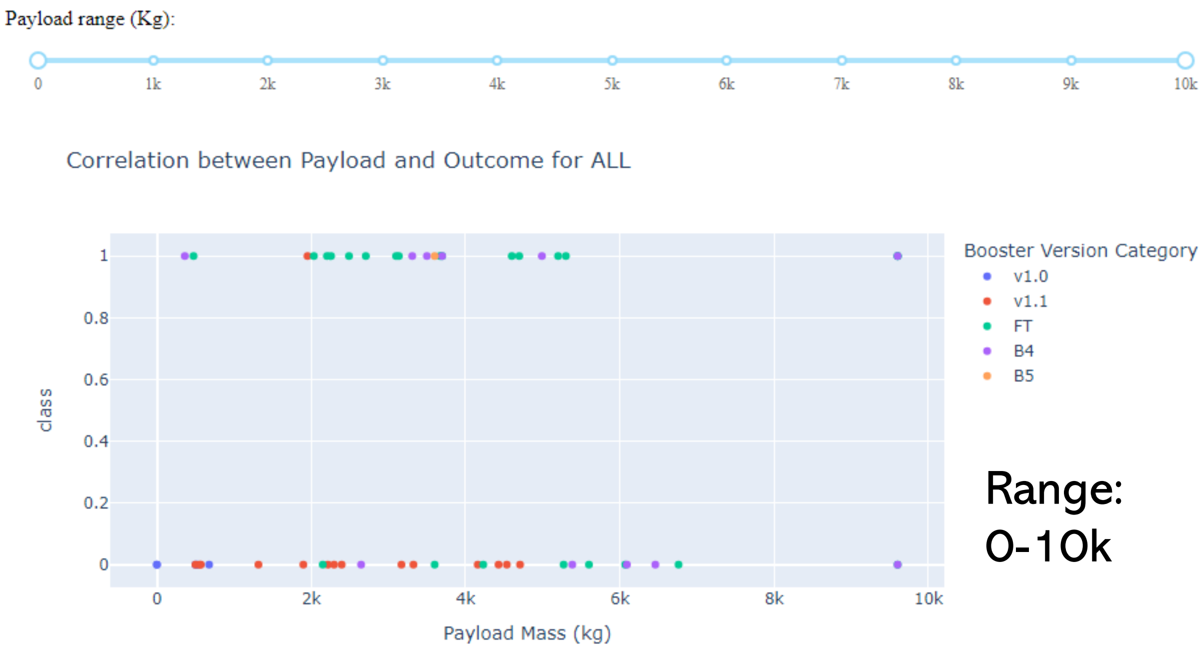
- From this chart we can see that:
  - KSC LC-39A has the highest number of launch successes
  - CCAFS SLC-40 has the lowest number of launch successes

Piechart of the launch site with highest launch success ratio

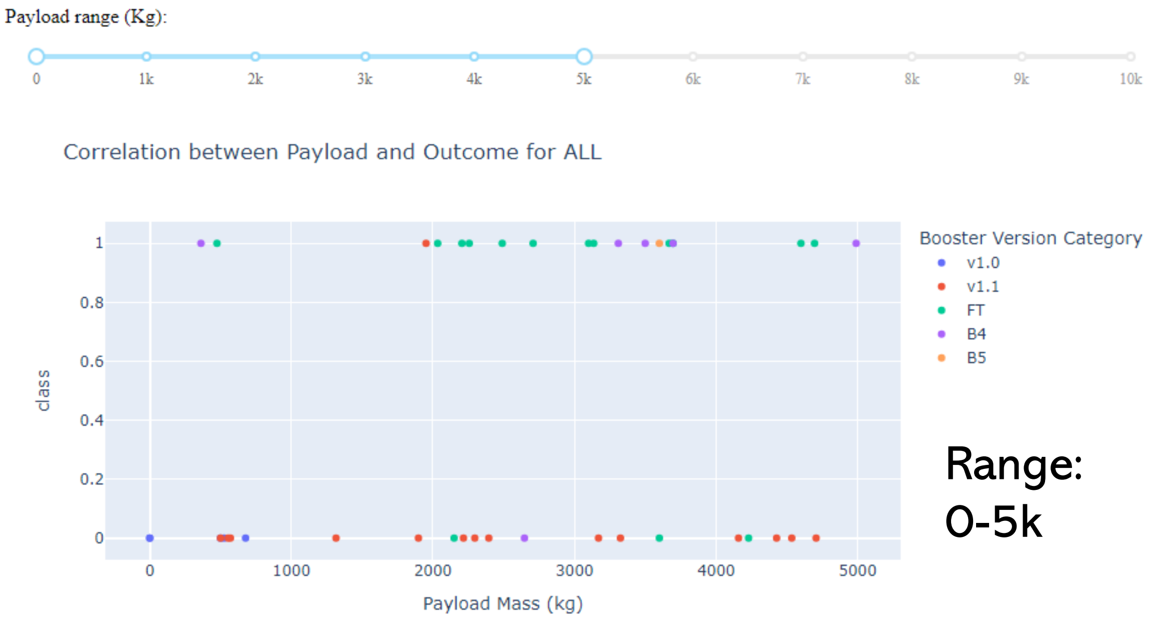


- From this plot we can see that the Launch Site KSC LC-39A has a success rate of 76.9% and a failure rate of 23.1%.

Payload vs. Launch Outcome scatter plot for all sites



Range:
0-10k



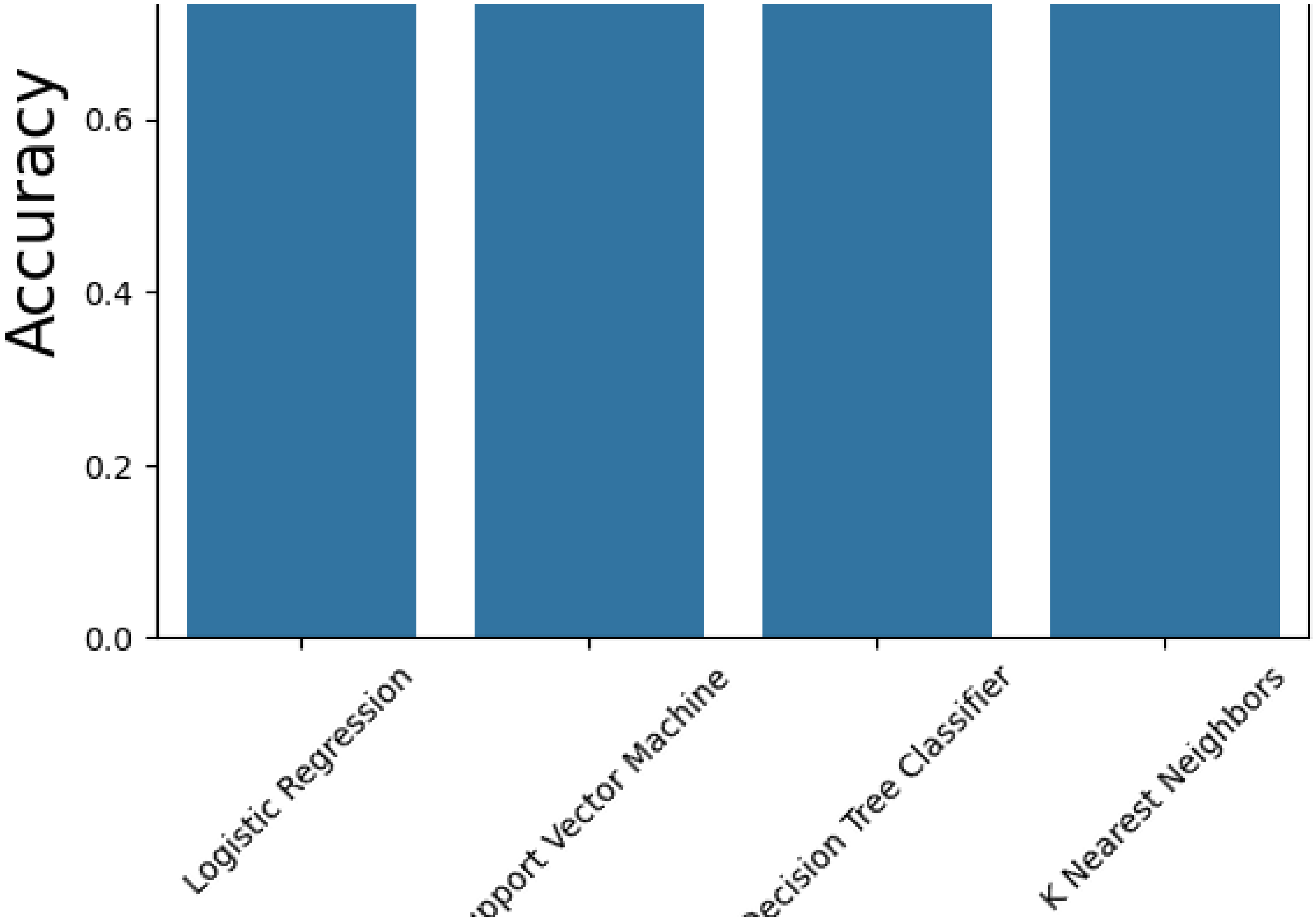
Range:
0-5k

- We can see that the lower payloads have a higher success rate
- And that the Booster Version FT seems to be the one with the highest number of successes

Section 5

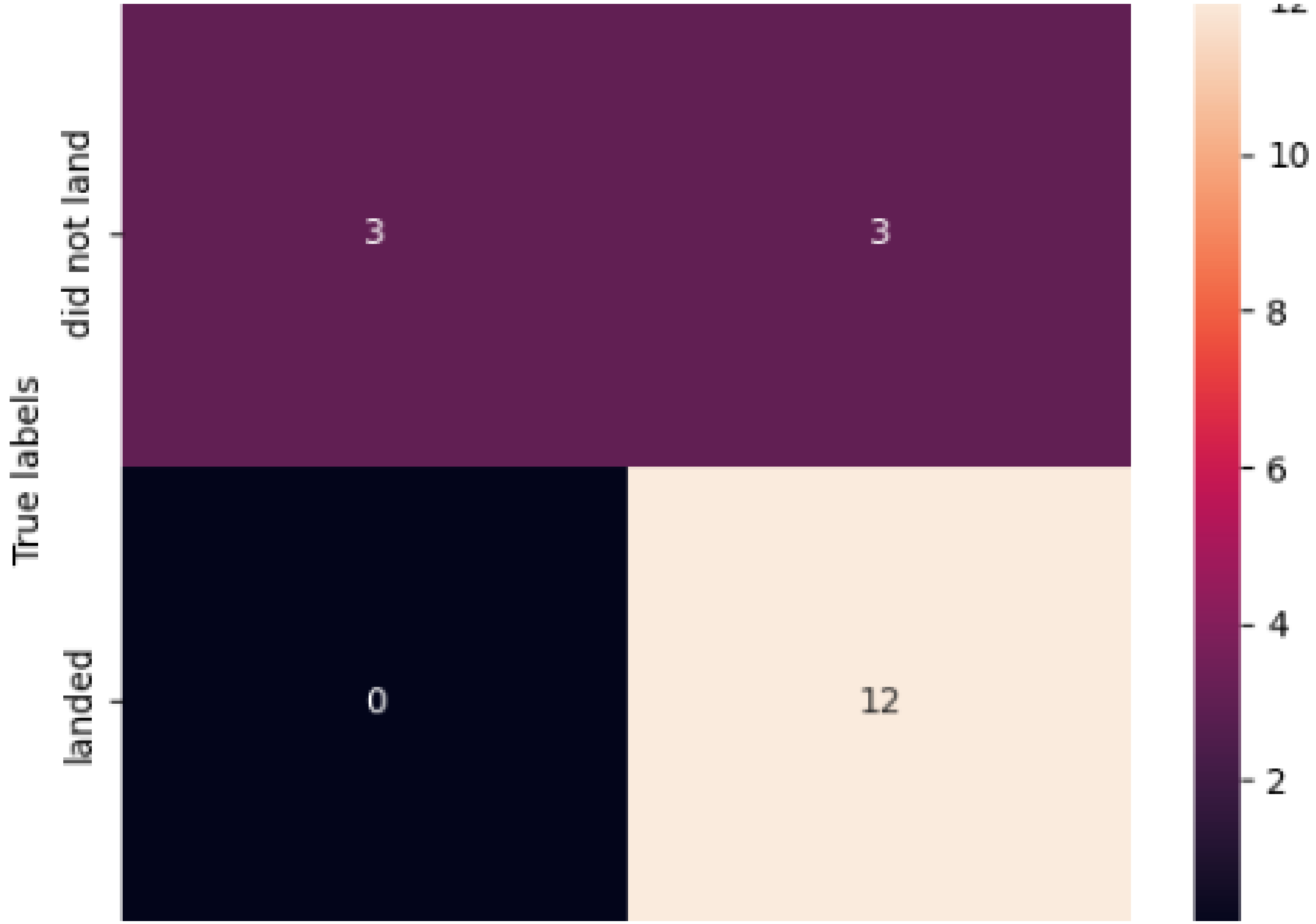
Predictive Analysis (Classification)

Classification Accuracy

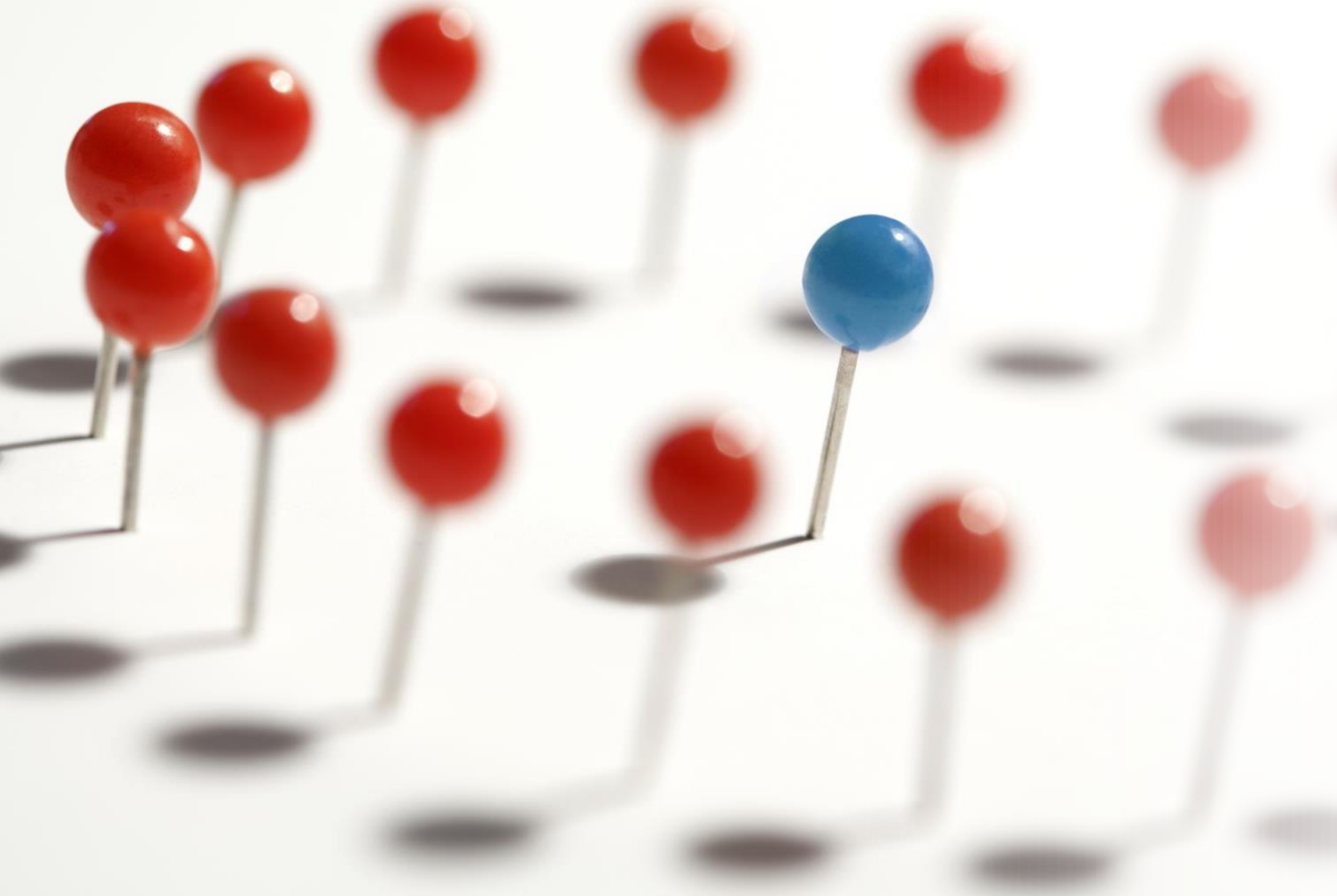


- For the hyperparameters that were picked, all the models had the same accuracy (0.83).
- The only that had to be changed a bit was the Decision Tree since there was a warning due to deprecation that only allowed us to run the 'sqrt' mode for max_features.

Confusion Matrix



- All the methods had the same Confusion Matrix.
- Seeing the Confusion Matrix we can see that we see the methods could distinguish between the different classes. We see that the major problem is false positives.



Conclusions

We can conclude that:

- The larger the Flight Number at a Launch Site the greater the success.
- Launch success rate started to increase in 2013 till 2020.
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- Launch Site KSC LC-39A had the most successful launches of any sites, while CCAFS SLC-40 had the lowest.
- Any of the Machine Learning methods could be used for this dataset.

Appendix

- Since SQL was the toughest part for us, we decided to add some new queries to further study the Dataset
- GitHub Link: https://github.com/Aegiel/ds_capstone/blob/main/extra%20queries.ipynb
- Some examples:
    1. Retrieve the launch site with the highest success rate for missions with payloads to a specific orbit
    2. Calculate the average payload mass for missions with a successful landing outcome and compare it to the average payload mass for missions with a failed landing outcome for each booster version

1)

```
%sql SELECT Launch_Site
    FROM SPACEXTABLE
    WHERE Orbit = 'Desired_Orbit' AND Mission_Outcome = 'Success'
    GROUP BY Launch_Site
    HAVING COUNT(*) = (
        SELECT MAX(Count_Mission)
        FROM (
            SELECT Launch_Site, COUNT(*) AS Count_Mission
            FROM SPACEXTABLE
            WHERE Orbit = 'Desired_Orbit'
            GROUP BY Launch_Site, Mission_Outcome
        )
    );
```

2)

```
%sql SELECT Booster_Version, AVG(PAYLOAD_MASS__KG_) AS Avg_Successful_Landing_Mass, Avg_Failed_Landing_Mass
    FROM (
        SELECT Booster_Version, AVG(CASE WHEN Landing_Outcome = 'Success' THEN PAYLOAD_MASS__KG_ ELSE NULL END) AS Avg_Succe
        FROM SPACEXTABLE
        GROUP BY Booster_Version
    ) AS SuccessfulLanding
    LEFT JOIN (
        SELECT Booster_Version, AVG(CASE WHEN Landing_Outcome = 'Failure' THEN PAYLOAD_MASS__KG_ ELSE NULL END) AS Avg_Fail
        FROM SPACEXTABLE
        GROUP BY Booster_Version
    ) AS FailedLanding ON SuccessfulLanding.Booster_Version = FailedLanding.Booster_Version;
```

Thank you!