IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

CRISTIAN JOSUÉ MARTÍNEZ OBANDO
19 MAYO 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data collection trough API REST
  - Data collection with Web Scraping
  - Data Wrangling process
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization (Matplotlib and Seaborn)
  - Interactive Visual Analytics with Folium
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive Analytics
  - Predictive Analytics result from Machine Learning Lab

Introduction

SpaceX is a game-changing company that has disrupted the space industry by offering Falcon 9 rocket launches at significantly lower costs compared to other providers. Their innovative idea of reusing the first stage of the rocket has enabled them to achieve substantial cost savings. As a data scientist working for a startup competing with SpaceX, the objective of this project is to develop a machine learning pipeline to predict the landing outcome of the first stage in future missions. This prediction will help determine the optimal bidding price for rocket launches, competing effectively against SpaceX.

The problems included:

- Identify which factors affect the landing outcome
- the relationship between all the variables and see how they affect the landing outcome
- finding the optimal conditions for a successful landing result

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - The data was collected using SpaceX API REST and web scrapping from Wikipedia
- Perform data wrangling
  - Data was processed using one-hot encoding for categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build, tune, and evaluate classification models using GridSearchCV and 10-fold cross-validation for optimal performance.

Data Collection

- I used the SpaceX REST API as the main data source for this project. To collect the data, I set the necessary parameters and made the corresponding requests to the API. Then, I normalized the obtained data and stored it in a JSON file. Next, I loaded the JSON file into a data frame and performed a selection of the relevant columns for our analysis. Subsequently, I performed a cleanup of the data, treating null values and applying necessary transformations. Finally, I stored the resulting data frame in a CSV file for further use in data analysis and modeling.

Github:
https://github.com/CJ7MO/Applied-Data-Science-Capstone/tree/main/week_1

Data Collection – SpaceX API

The code makes a request to the SpaceX API to get data from past launches. It then converts the JSON response into a Data Frame using '**json_normalize**'. It then selects the relevant columns and filters out rows containing multiple cores and multiple payloads. It also extracts the unique values from the lists in the '**cores**' and '**payloads**' columns. It then converts the '**date_utc**' column to a date data type and extracts the date part. Finally, it filters out rows with a date before or equal to November 13, 2020. In summary, the code obtains, transforms and filters past SpaceX launch data according to certain characteristics and date constraints.

Github:
https://github.com/CJ7MO/Applied-Data-Science-Capstone/blob/main/week_1/Collecting_the_Data/jupyter-labs-spacex-data-collection-api.ipynb

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```
Executed in 31ms, 19 May at 19:58:10

```
response = requests.get(spacex_url)
```
Executed in 961ms, 19 May at 19:58:11

```
# Use json_normalize meethod to convert the json result into a dataframe
response = response.json()
data = pd.json_normalize(response)
```
Executed in 74ms, 19 May at 19:58:12

```
# 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 that have multiple payloads in a single rocket.
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 feature.
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)]
```
Executed in 299ms, 19 May at 19:58:12

Data Collection - Scraping

The code performs web scraping on a web page to extract detailed information about rocket launches. It uses the requests and '**BeautifulSoup**' libraries to make an HTTP request, parse the page content and extract data from various tables. The extracted information includes details such as flight number, date, time, rocket version, launch site, payload, payload mass, orbit, client, launch result and rocket landing. The data is compiled into dictionaries and stored in a list called '**launch_data**'.

Github:
https://github.com/CJ7MO/Applied-Data-Science-Capstone/blob/main/week_1/Collecting_the_Data/jupyter-labs-webscraping.ipynb

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
Executed in 1s, 13 May at 18:45:31
```

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
Executed in 860ms, 13 May at 18:45:32
```

```
extracted_row = 0
launch_data = []
#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()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
```

Data Wrangling

Data cleaning and organization, known as Data Wrangling, is a process that seeks to simplify complex data sets for easy access and exploratory analysis. In this case, we will perform several calculations to obtain relevant information. First, we will determine the number of launches at each launch site. Then, we will calculate the number and frequency of mission results by orbit type. In addition, we will create a label to classify the landing results, which will facilitate their analysis, visualization and use in Machine Learning. Finally, we will export the results to a CSV file. In summary, this process involves cleaning and organizing the data, calculating important statistics, and creating labels for more effective analysis.

Github:
https://github.com/CJ7MO/Applied-Data-Science-Capstone/blob/main/week_1/Data_Wrangling/labs-jupyter-spacex-data_wrangling_jupyterlite.jupyterlite.ipynb

EDA with Data Visualization

- Swarmplots to compare the success of launches according to a class (1: success, 0: failure).
- Swarmplots to analyze the number of flights per launch site and the payload to launch site ratio.
- Barplots to show the success rate by orbit type.
- Catplots to analyze the number of flights per orbit and their success or failure.
- Line graph to display the success rate on a yearly basis.

Github:
https://github.com/CJ7MO/Applied-Data-Science-Capstone/blob/main/week_2/Exploratory_Analysis_Using_Pandas_and_Matplotlib/jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- query of the distinct launch sites
- query of the average payload
- query of the average payload where the **booster_version** 'F9 v1.1' is set to 'F9 v1.1'.
- query of the type of **booster_version** with payload between 4000 and 6000
- query of the different results of the tests
- query of the count of successful **landing_outcomes** between the date 04-06-2010 and 20-03-2017 in descending order.

Github:
https://github.com/CJ7MO/Applied-Data-Science-Capstone/blob/main/week_2/Exploratory_Analysis_Using_SQL/jupyter-labs-eda-sql-coursera_sqllite.ipynb

Build an Interactive Map with Folium

- To visualize the launch sites on an interactive map we used the coordinates information available in the dataset, and added a circle around the site and added a label with the name of the corresponding launch site.
- We did cluster of the data, using marker cluster to visualize the number of launches, for each launch site, and the result of the launch, being green for successful and red for failed.
- We have also added a mouse position function that shows us the coordinates where the pointer is, useful for calculating the distance between strategic points.

Github:
https://github.com/CJ7MO/Applied-Data-Science-Capstone/blob/main/week_3/site_map/lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- We build a dashboard that receives as input the launch site and in a pie chart shows the ratio of successful vs. unsuccessful results, and if no launch site is chosen, it shows the success rate of each launch site.Explain why you added those plots and interactions
- A scatter plot was also made showing the amount of payload and the result of the launch, being 1 successful and 0 unsuccessful.

Github:
https://github.com/CJ7MO/Applied-Data-Science-Capstone/blob/main/week_3/site_map/lab_jupyter_launch_site_location.jupyterlite.ipynb

Predictive Analysis (Classification)

- To develop, evaluate, improve and select the best classification model, the following steps were performed: the data set was divided into training and test sets, one-hot coding was applied, the parameters of each model were configured, **'Logistic Regression'**, **'SVM'**, **'Decision Tree'** and **'KNN'** models were built, they were evaluated and improving using **'GridSearchCV'** with cross-validation equal to ten, iterated to improve the models by adjusting the parameters and, finally, the model with the best result obtained during the evaluation was selected.

Github:
https://github.com/CJ7MO/Applied-Data-Science-Capstone/blob/main/week_4/SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

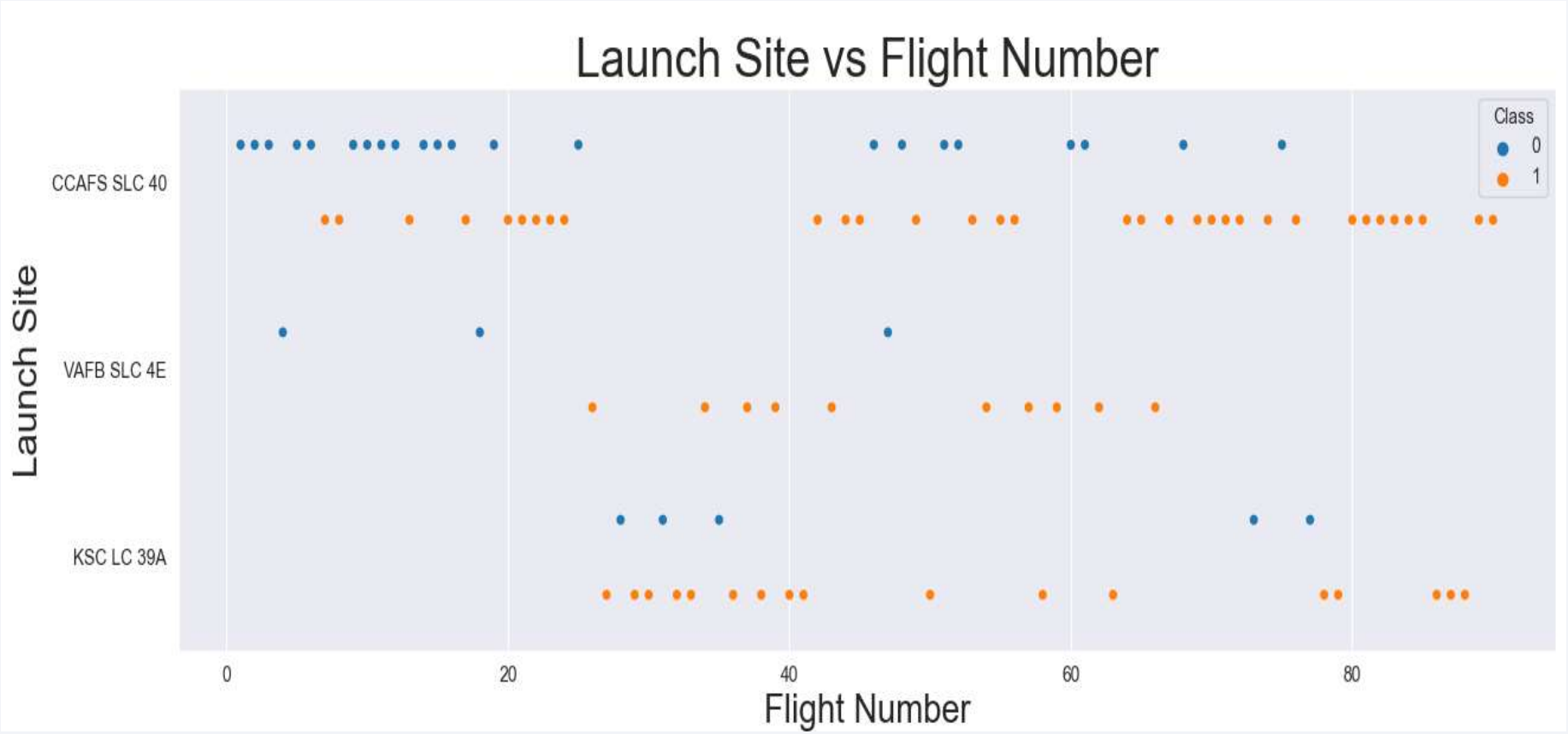
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

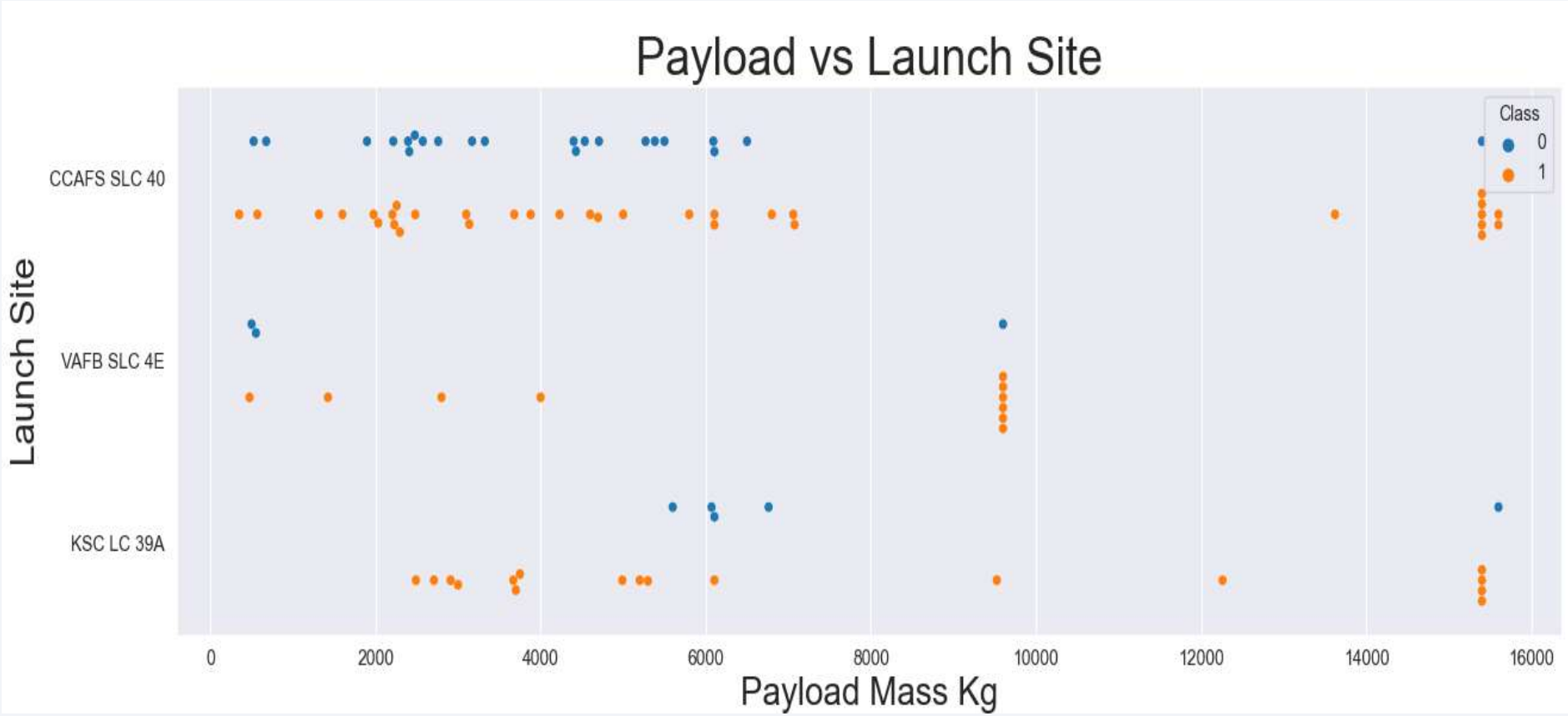
Insights drawn from EDA

Flight Number vs. Launch Site

This chart shows the number of flight and the launch site, and separates them according to the result, being the orange color for successful result Class = 1, and blue for failed Class = 0, also a tendency is seen that to greater number of flights greater success, demonstrating this that from the flight number 78 there was great percentage of success.

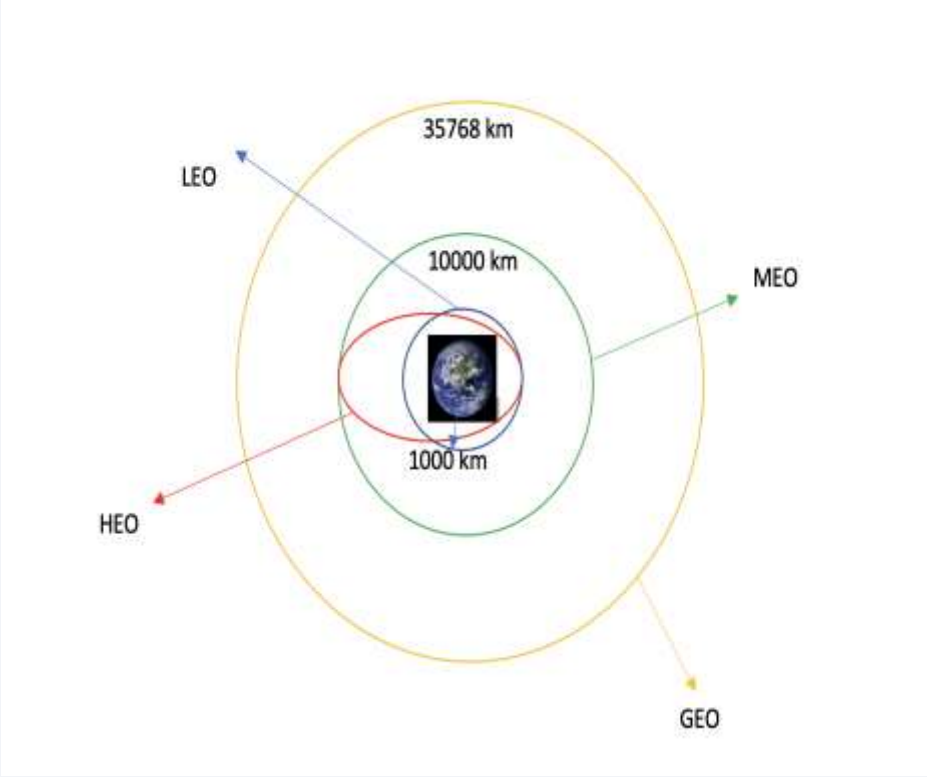


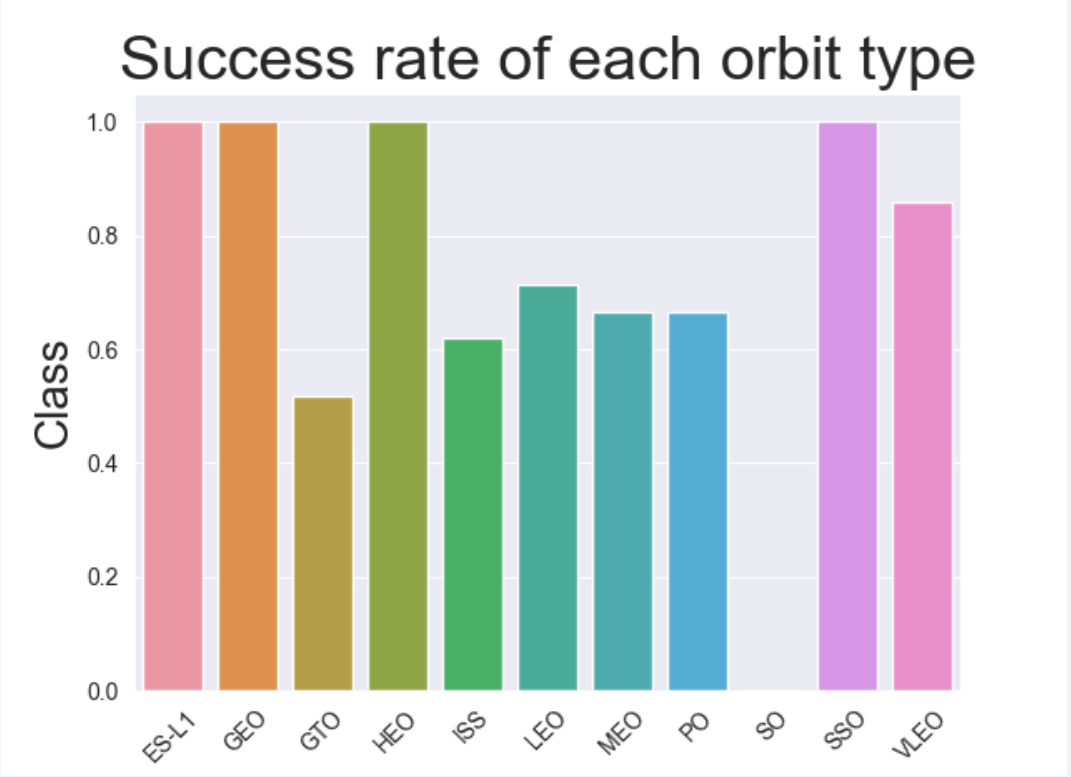
Payload vs. Launch Site



This chart shows the payload mass used per launch and launch site, and separates them according to the result, being the orange color for successful result Class = 1, and blue for failed Class = 0, the chart shows how above 8000 kilograms payload mass the success rate increases considerably in all launch sites.

Success Rate vs. Orbit Type





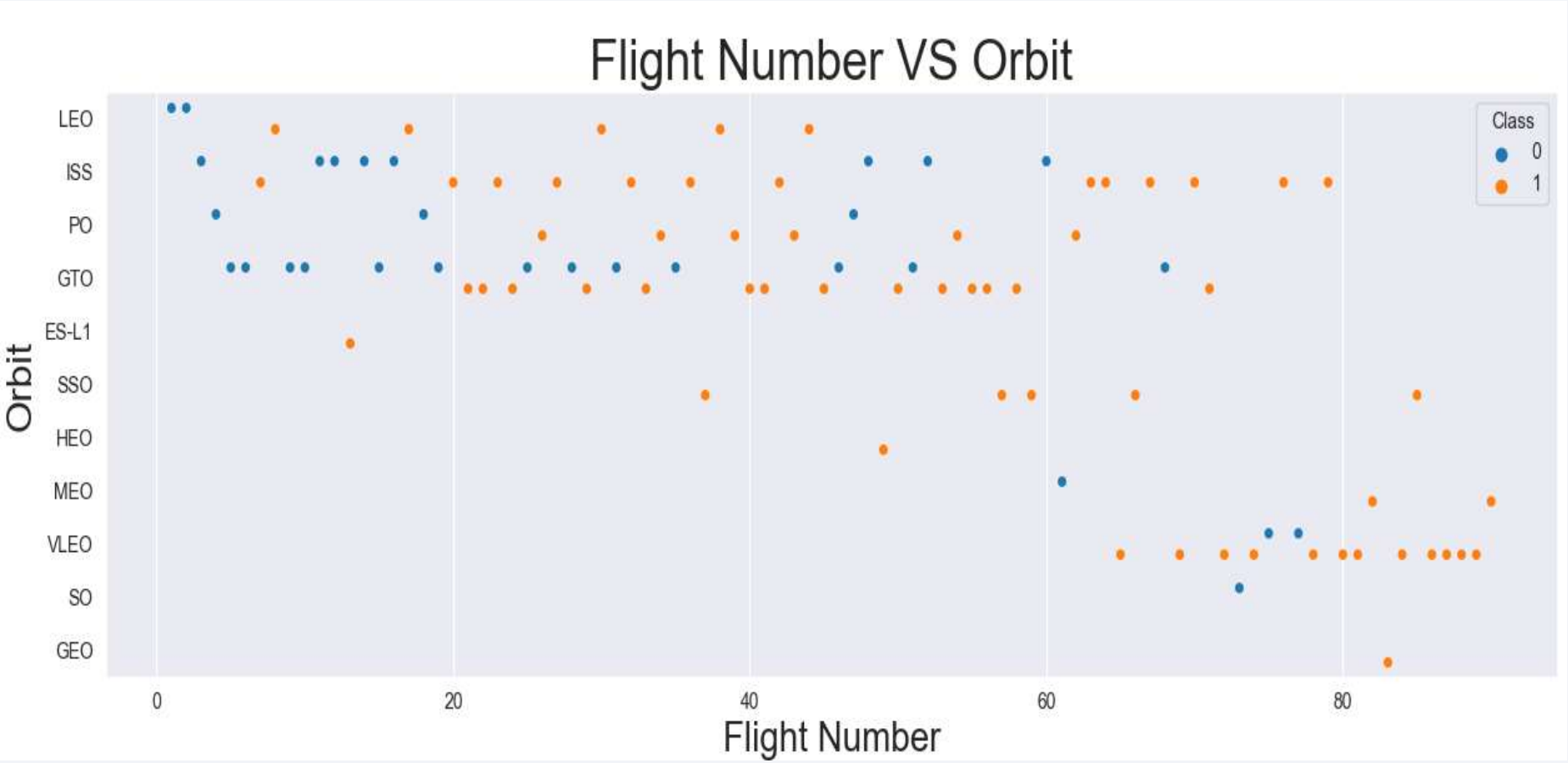
This bar chart shows the success rate for each type of orbit, with the following 4 orbits having the highest success rate:

- ES-L1
- GEO
- HEO
- SSO

Flight Number vs. Orbit Type

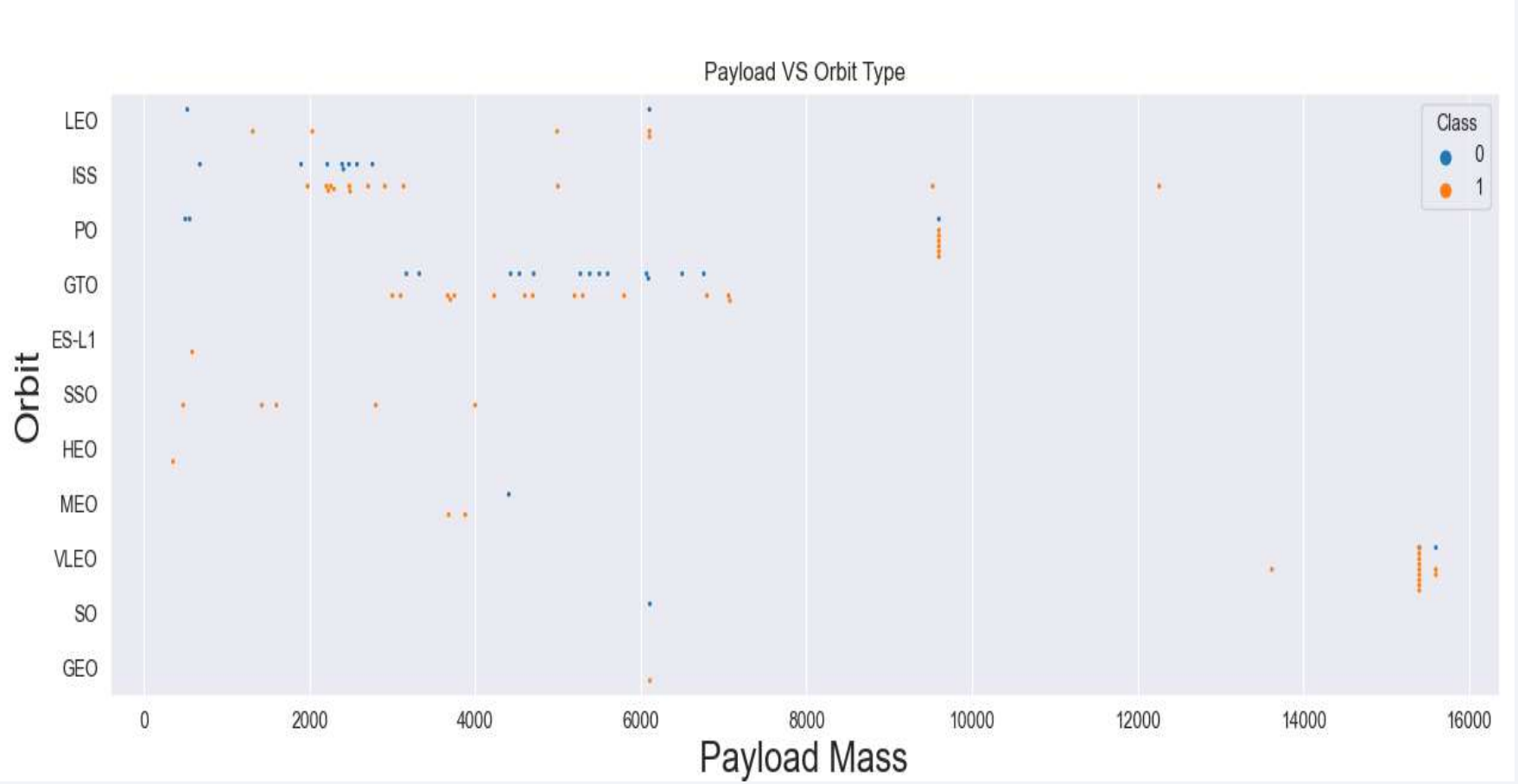
This chart shows the number of flight and the type of orbit, and separates them according to the result, being the orange color for successful result Class = 1, and blue for failed Class = 0, as in the previous chart we can see those orbits with higher success rate

- ES-L1
- GEO
- HEO
- SSO

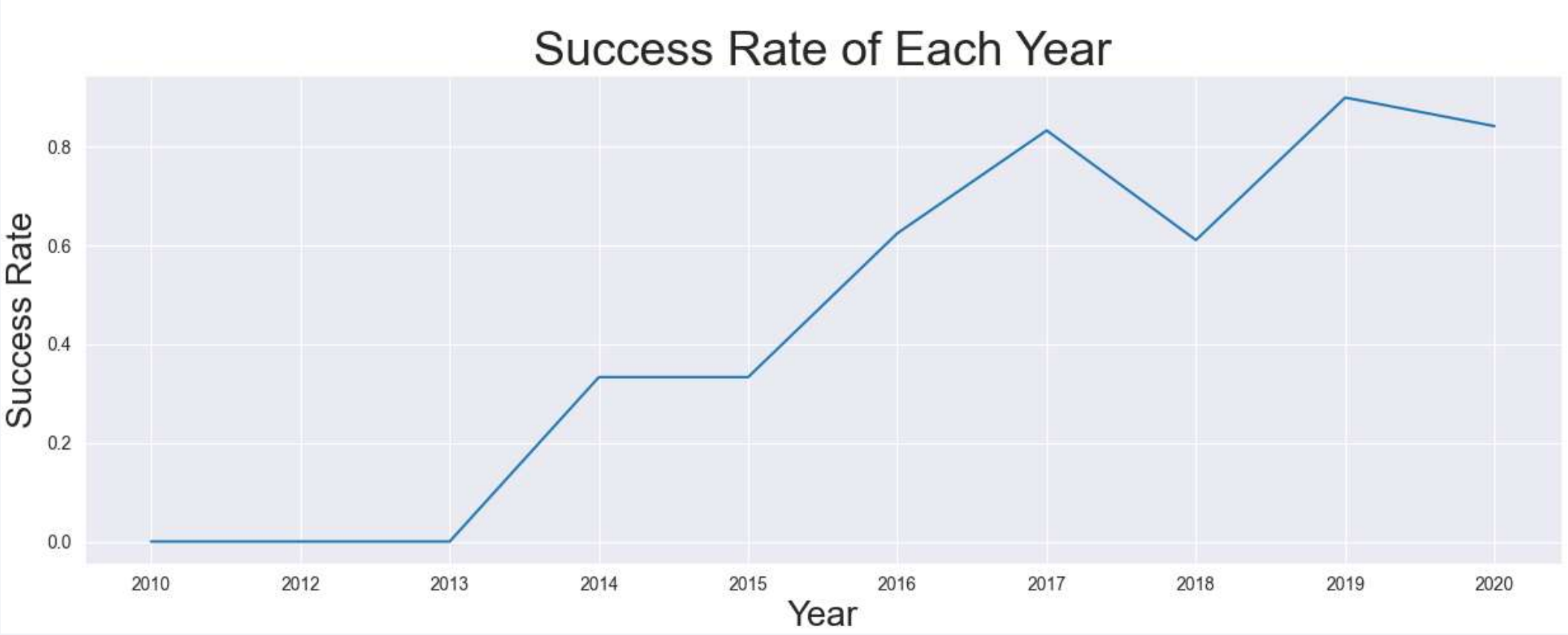


Payload vs. Orbit Type

This chart shows the payload mass used per launch and orbit type, and separates them according to the result, being the orange color for successful result Class = 1, and blue for failed Class = 0, the chart shows how above 8000 kilograms payload mass the success rate increases considerably in all orbit types



Launch Success Yearly Trend



This line chart shows how the annual trend of successful Launches has performed, which since 2013 has been increasing and peaked above 80% in 2019.

All Launch Site Names

We use this query with **SELECT DISTINCT** to display the unique values in the '**Launch_Site**' column of the **SPACEXTBL** table.

```
%sql SELECT DISTINCT Launch_Site FROM SPACEXTBL
Executed in 29ms, 19 May at 10:24:43
 * sqlite:///my_data1.db
Done.
```

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

Launch Site Names Begin with 'CCA'

This query Selects the first 5 records beginning with 'CCA'.

```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5
```

Executed in 38ms, 19 May at 10:24:43

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (parachute)
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No attempt
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No attempt
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

This query shows the sum of the total payload mass where the client is '**Nasa (CRS)**'.

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS 'Total Payload Mass by NASA (CRS)' FROM SPACEXTBL WHERE Customer = 'NASA (CRS)'
```

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

Total Payload Mass by NASA (CRS)
45596.0

Average Payload Mass by F9 v1.1

This query shows the Average Payload Mass where the booster version is '**F9 v1.1**'.

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS 'AVG Payload Mass by F9 v1.1' FROM SPACEXTBL WHERE Booster_Version = 'F9 v1.1'
Executed in 30ms, 22 May at 13:04:53

 * sqlite:///my_data1.db
Done.

AVG Payload Mass by F9 v1.1

2928.4
```

First Successful Ground Landing Date

This query shows the date of the first successful landing outcome.

Using the included function min() we get the result of 01 / 08 / 2018.

```
%sql SELECT min(Date)  as 'First Succes Landing Outcome' FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)'
Executed in 73ms, 22 May at 14:42:17

 * sqlite:///my_data1.db
Done.

First Succes Landing Outcome
01/08/2018
```

Successful Drone Ship Landing with Payload between 4000 and 6000

In this query we selected the '**Booster_Version**' whose '**Landing_Outcome**' has been '**Succes (drone ship)**' and the '**Payload_Mass__KG_**' is between 4000 kg and 6000kg, which gave us the different types of '**Booster_Version**' that meet the query statement

```
%%sql
SELECT booster_version
FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (drone ship)' AND
PAYLOAD_MASS__KG_ BETWEEN 4000 AND 6000
```

Executed in 61ms, 22 May at 14:42:17

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

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

Total Number of Successful and Failure Mission Outcomes

- In this query we use the built-in COUNT function to count the number of '**Mission_Outcome**' had a result starting with '**Success**' or '**Failure**' using The **"%"** symbol in the query acts as a wildcard and can represent any number of characters in that position. Therefore, this query will look for any value in a column that begins with the string '**Success**' or '**Failure**' followed by any other character or characters.

```
%sql SELECT COUNT(Mission_Outcome) as Successful_Mission FROM SPACEXTBL WHERE Mission_Outcome LIKE 'Success%'
```

Executed in 231ms, 22 May at 14:42:17

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

Successful_Mission
100

```
%sql SELECT COUNT(Mission_Outcome) as Failure_Mission FROM SPACEXTBL WHERE Mission_Outcome LIKE 'Failure%'
```

Executed in 217ms, 22 May at 14:42:17

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

Failure_Mission
1

Boosters Carried Maximum Payload

```
%%sql
SELECT booster_version, payload_mass__kg_ FROM SPACEXTBL
WHERE payload_mass__kg_ = (SELECT max(payload_mass__kg_) FROM SPACEXTBL)
```

Executed in 197ms, 22 May at 14:42:17

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600.0
F9 B5 B1049.4	15600.0
F9 B5 B1051.3	15600.0
F9 B5 B1056.4	15600.0
F9 B5 B1048.5	15600.0
F9 B5 B1051.4	15600.0
F9 B5 B1049.5	15600.0
F9 B5 B1060.2	15600.0
F9 B5 B1058.3	15600.0
F9 B5 B1051.6	15600.0
F9 B5 B1060.3	15600.0
F9 B5 B1049.7	15600.0

This query lists the '**Booster_Version**' that have carried the maximum payload mass in kg using a subquery and the built-in max() function.

2015 Launch Records

This query seeks to select the columns '**Booster_Version**' and '**Launch_Site**' from the table '**SPACEXTBL**' where the landing result is '**Failure (drone ship)**', and the date year is "2015".

```
%%sql
SELECT booster_version, launch_site
FROM SPACEXTBL
WHERE landing_outcome = 'Failure (drone ship)' and substr(Date, 7,4) = '2015'
```

Executed in 37ms, 22 May at 15:11:56

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

Booster_Version	Launch_Site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

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

This query counts the number of occurrences of each landing result in the '**SPACEXTBL**' table between the dates '**04-06-2010**' and '**20-03-2017**'. It then groups the results by the '**Landing_Outcome**' field and sorts them in descending order according to the count.

```
%%sql
SELECT COUNT(landing_outcome), landing_outcome FROM SPACEXTBL
WHERE Date BETWEEN '04-06-2010' and '20-03-2017' GROUP BY landing_outcome
ORDER BY COUNT(Landing_outcome) DESC
```

Executed in 20ms, 22 May at 15:12:05

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

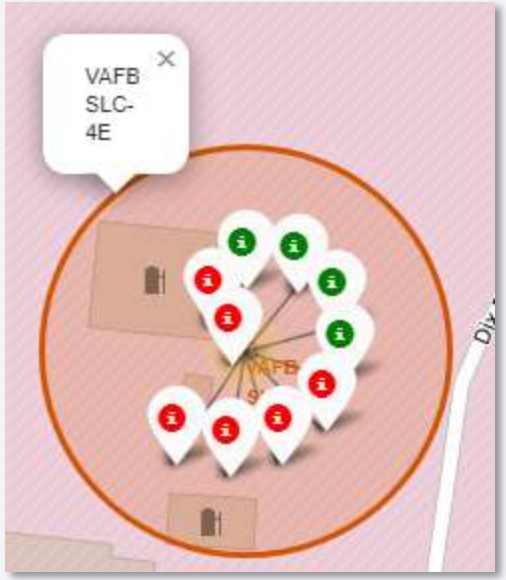
COUNT(landing_outcome)	Landing_Outcome
20	Success
10	No attempt
8	Success (drone ship)
7	Success (ground pad)
3	Failure (drone ship)
3	Failure
2	Failure (parachute)
2	Controlled (ocean)
1	No attempt


Section 3
Launch Sites
Proximities Analysis

Location of all Launch Sites

In this map created with folium and using folium and folium plugins you can see all the SpaceX launch sites.

Landing Outcomes Markers by each Launch Site



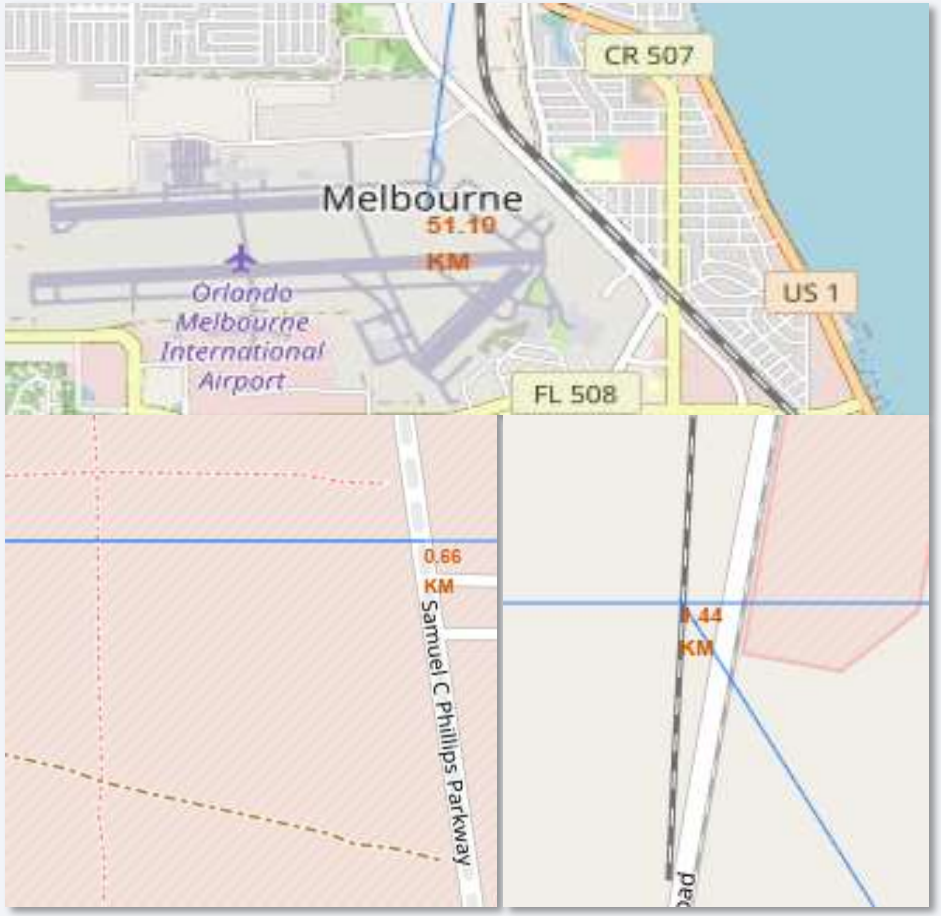
California Launch Site

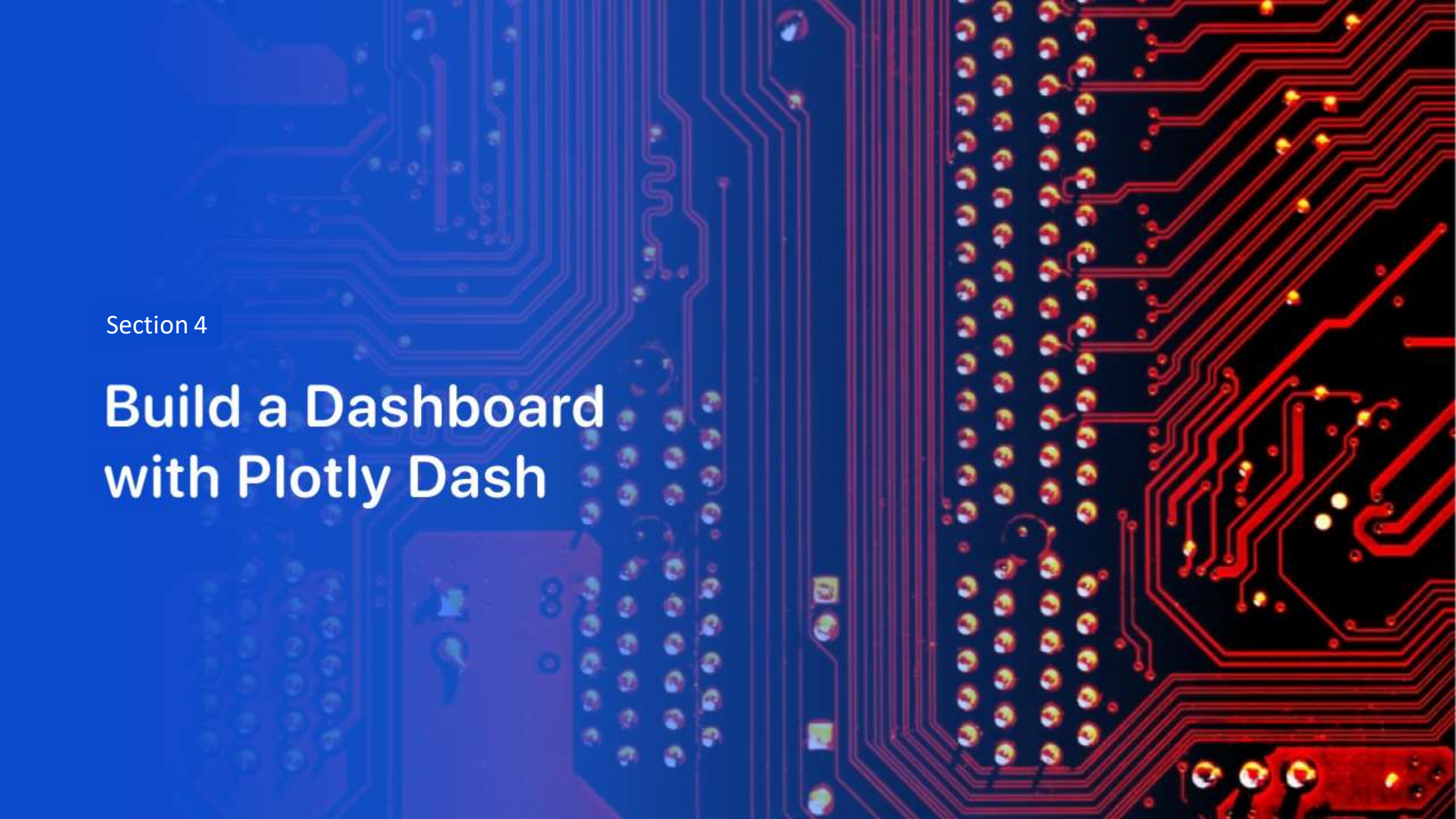
Florida Launch Sites

Green markers indicate a successful landing and Red Markers indicate a failed landing.

Distance between critical points of interest

These images are taken from the map generated with folium in which we used a formula to calculate the distance between different points of interest, the distance is the one shown in the captures, and we took as a reference point the launch site with more failed landings, which was CCAFS LC-40, being quite far from the surrounding cities, but not so far from the Nasa Railroad or the Samuel C Phillips Parkway.



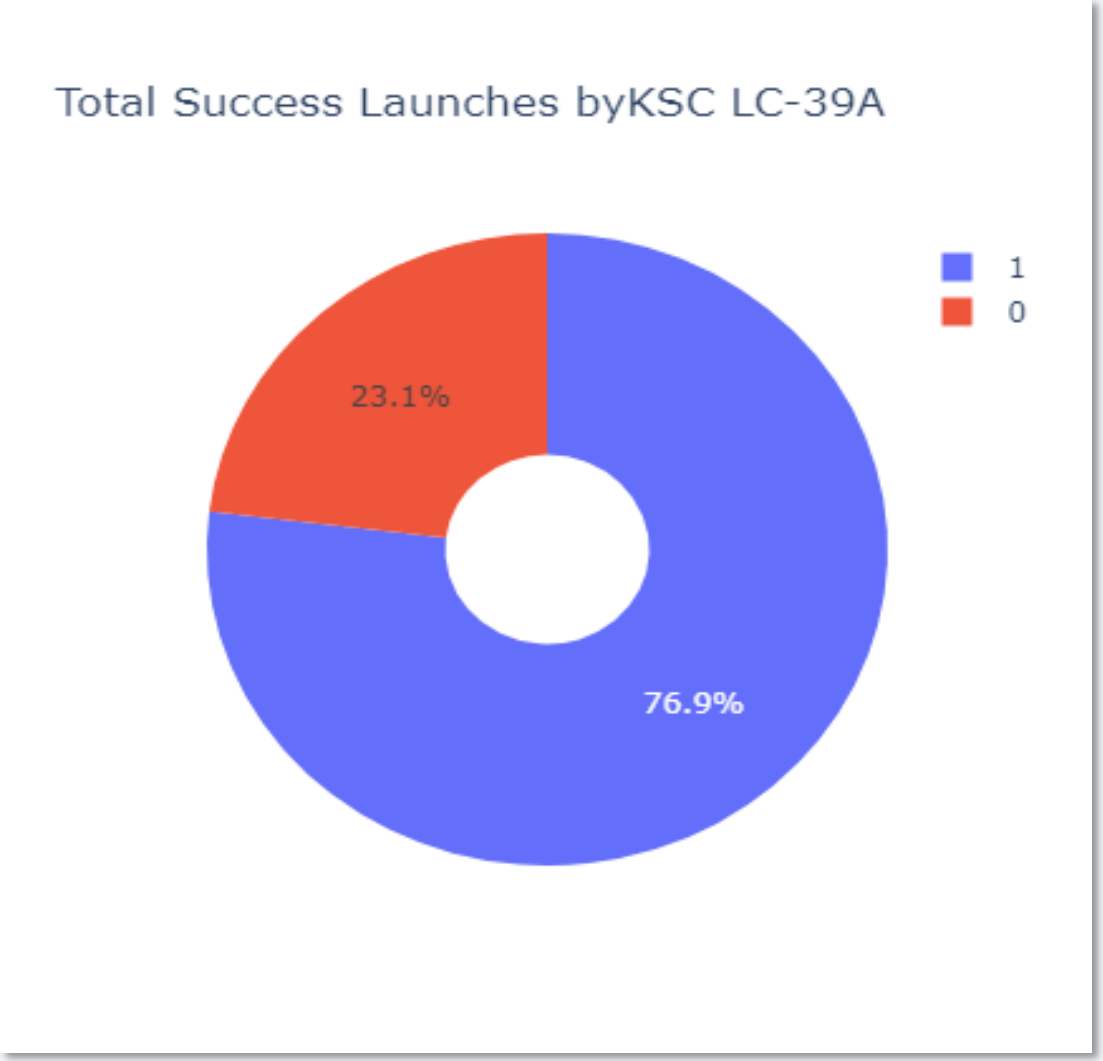

Section 4
Build a Dashboard with Plotly Dash

Pie chart of percentage of successful flights by launch site.

In this pie chart which stands out the launch site with the highest percentage of success which is **'KSC LC-39A'** exceeding 40% of total successful launches.

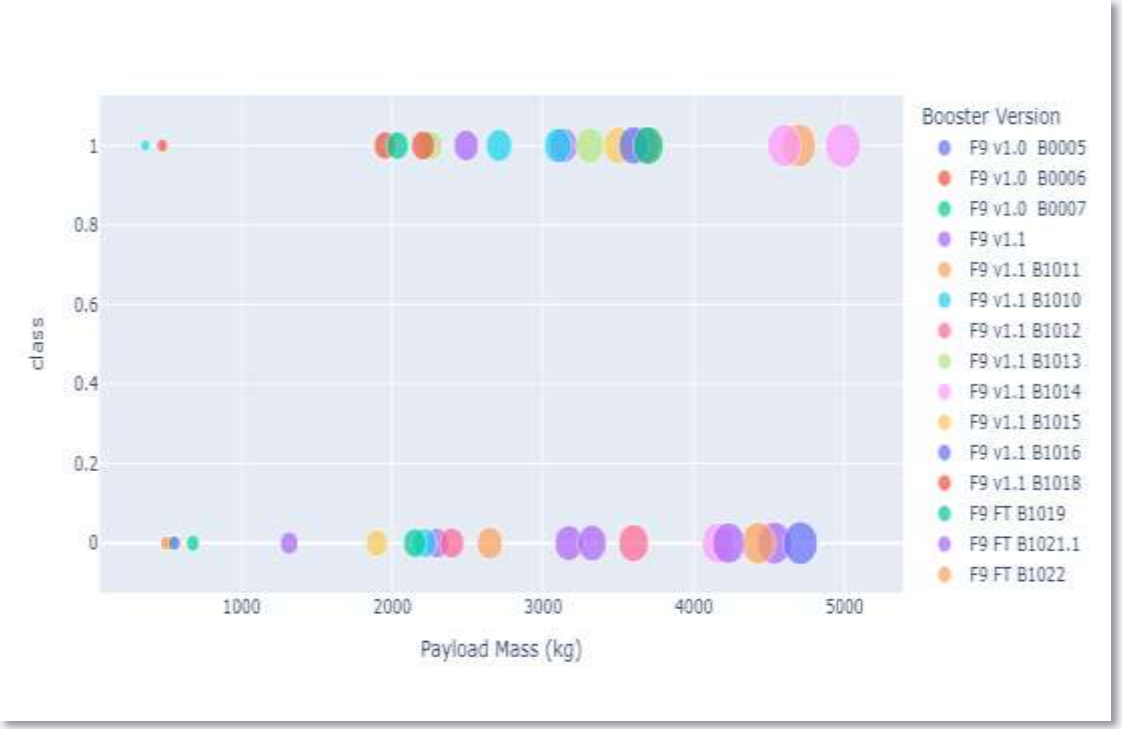
The Launch Site with the Highest Success Rate

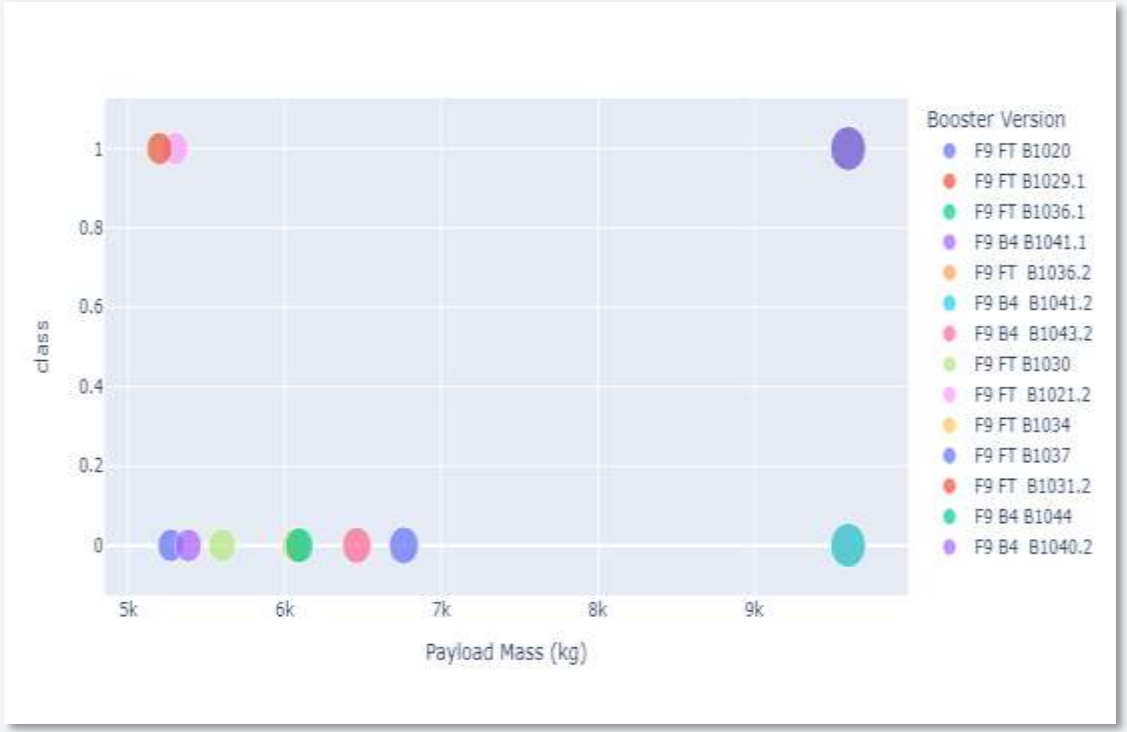
'KSC LC-39A' achieved a higher percentage than the other launch sites, with a **76.9%** success rate and only **23.1%** of failed launches, which is a great achievement.



Payload Mass VS Launch Outcome

These charts show the behavior of the launch outcome depending on the amount of payload mass used for the launch.





Section 5

Predictive Analysis (Classification)

Classification Accuracy

Using the code below, we performed a comparison which gave us the model with the best accuracy, and its parameters, which was the Decision Tree Classifier.

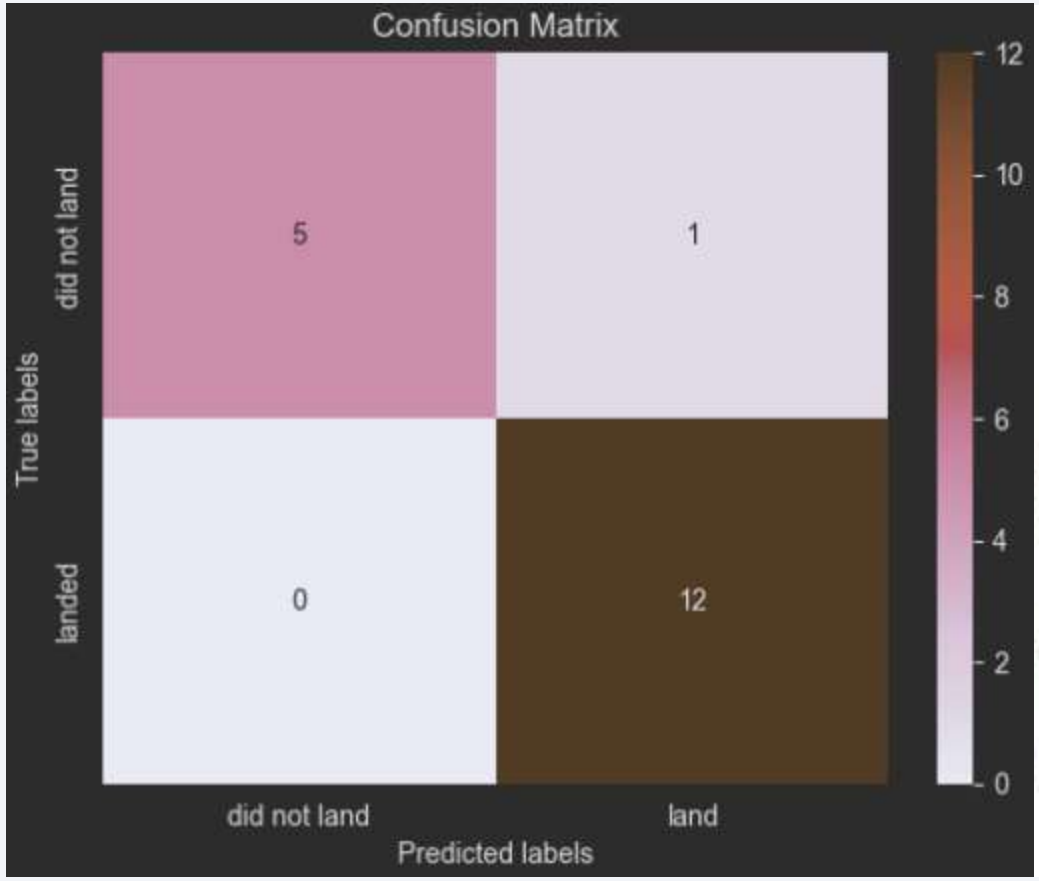
```
models = {'Logistic Regression': {'score': logreg_cv.best_score_, 'params': logreg_cv.best_params_},
    'Support Vector Machine': {'score': svm_cv.best_score_, 'params': svm_cv.best_params_},
    'Decision Tree Classifier': {'score': tree_cv.best_score_, 'params': tree_cv.best_params_},
    'KNN': {'score': knn_cv.best_score_, 'params': knn_cv.best_params_}}

best_model = max(models, key=lambda x: models[x]['score'])
best_score = models[best_model]['score']
best_params = models[best_model]['params']

print('The best model is:', best_model, 'with a score of', best_score)
print('Best parameters of the model are:', best_params)
```

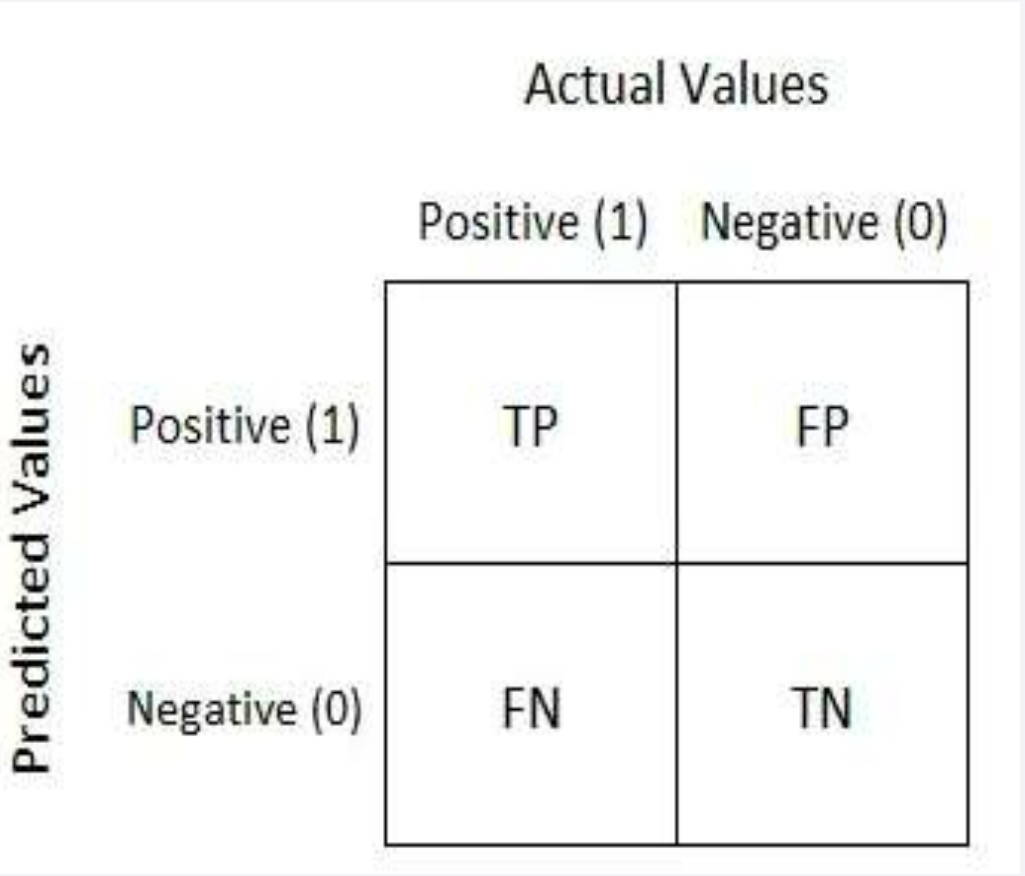
Executed in 93ms, 22 May at 16:41:40

```
The best model is: Decision Tree Classifier with a score of 0.875
Best parameters of the model are: {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 2,
'min_samples_split': 5, 'splitter': 'random'}
```

Confusion Matrix

The confusion matrix shows how the Decision Tree Classifier classification model predicts satisfactorily and with a very good fit on test and validation data.





Conclusions

- the distance between the launch sites is quite far from the surrounding cities, but not so far from railroads or highways.
- KSC LC-39A is the launch site with the most successful launches of any other site, with an 76.9% of successful and and 23.1% of failure.
- ES-L1, GEO, HEO, SSO orbits; are orbits with a high percentage of successful launches.
- The best model that best fits the data set and obtains the best accuracy score is Decision Tree Classifier.
- Since 2013 the success rate of each year's launches has increased significantly, and has peaked in 2019

Thank you!