IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Daniel Rodríguez Borreguero
16-08-2023

Table of contents

Executive Summary

- Summary of methodologies
  - Data collection with SpaceX API
  - Data collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis using SQL
  - Exploratory Data Analysis using Data Visualization
  - Interactive Visual Analytics with Folium and Dashboard
  - Predictive Analysis with Machine Learning
- Summary of all results
  - Exploratory Data analysis results
  - Interactive analytics in screenshots
  - Predictive analysis results

Introduction

- **Background and context**
  - This project conducts a study about Space X, which advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars, with the aim to make space travel affordable for everyone. Other providers cost upward of 165 million dollars each, and much of the savings is because Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against Space X for a rocket launch. This main objective of the project is to use public data and machine learning models to predict if the first stage will land successfully.
- Problems you want to find answers
  - What are the main factors or characteristics for a successful landing?
  - What are the rate of successful landings in the next decade?

Section 1

Methodology

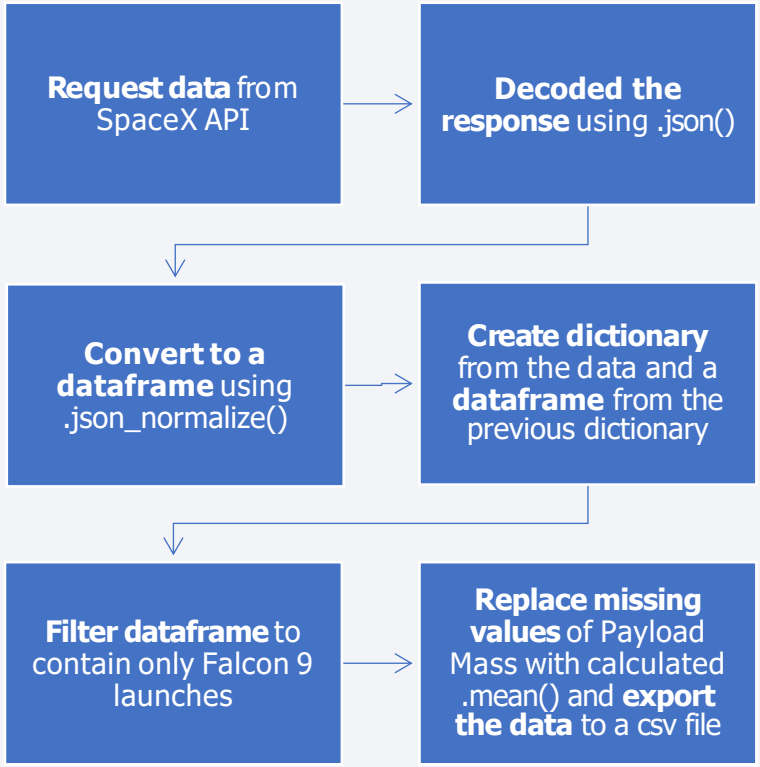
Methodology

Executive Summary

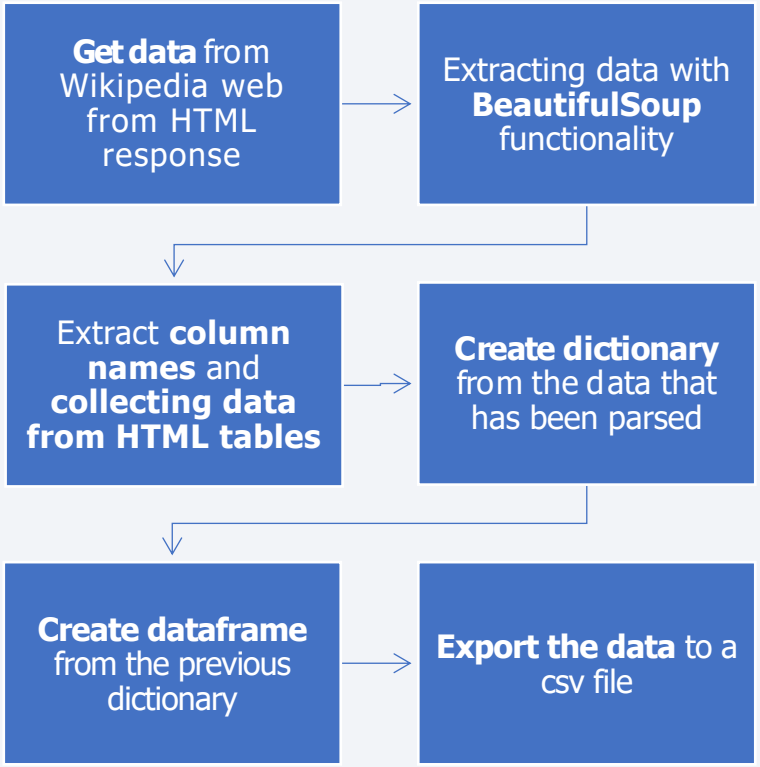
- Data collection methodology:
  - Collect data - using SpaceX REST API and web scraping techniques from Wikipedia.
- Perform data wrangling
  - Wrangle data – by filtering the data, handling missing values and applying one hot encoding – with the goal to prepare the data for analysis and modeling.
- Perform exploratory data analysis (EDA) using visualization and SQL
  - Explore data via EDA with SQL and data visualization techniques
- Perform interactive visual analytics using Folium and Plotly Dash
  - Visualize the data using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build Models to predict landing outcomes using classification models. Tune and evaluate models to find best model and parameters

Data Collection

- The main steps of the API REST data collection process are the following:



- The main steps of the web scrapping data collection process are the following:



Data Collection – SpaceX API

- It was used some basic data wrangling and formatting techniques and finally the missing values were solved.
- The link of the notebook is the following: https://github.com/danielrodriguez23-sudo/DR_DataScience_Capstone_Project/blob/main/1_DR_SPACEX_Data_Collection-API.ipynb

Task 1: Request and parse the SpaceX launch data using the GET request

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

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

We should see that the request was successfull with the 200 status response code

```
response.status_code
```

```
200
```

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

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

Task 2: Filter the dataframe to only include `Falcon 9` launches

Finally we will remove the Falcon 1 launches keeping only the Falcon 9 launches. Filter the data dataframe using the `BoosterVersion` column to only keep the Falcon 9 launches. Save the filtered data to a new dataframe called `data_falcon9`.

```
# Hint data['BoosterVersion']!='Falcon 1'
print(data_falcon9['BoosterVersion'].value_counts())
data_falcon9 = data_falcon9[data_falcon9['BoosterVersion']=='Falcon 9']
print(data_falcon9['BoosterVersion'].value_counts())
```

```
Falcon 9    90
Falcon 1     4
Name: BoosterVersion, dtype: int64
Falcon 9    90
Name: BoosterVersion, dtype: int64
```

Task 3: Dealing with Missing Values

Calculate below the mean for the `PayloadMass` using the `.mean()`. Then use the mean and the `.replace()` function to replace `np.nan` values in the data with the mean you calculated.

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

```
FlightNumber        0
Date                0
BoosterVersion      0
PayloadMass         0
Orbit               0
LaunchSite          0
Outcome             0
Flights             0
GridFins            0
Reused              0
Legs                0
LandingPad         26
Block               0
ReusedCount         0
Serial              0
Longitude           0
Latitude            0
```

Data Collection – Scraping

- It was used BeautifulSoup functionality for web scrapping to parse Falcon 9 launch records. Then we create a dataframe parsing the HTML tables.
- The link of the notebook is the following: https://github.com/danielrodriguez23-sudo/DR_DataScience_Capstone_Project/blob/main/2_DR_SPACEX_Web_Scraping.ipynb

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

```
# use requests.get() method with the provided static_url
# assign the response to a object
body = requests.get(static_url)
data = body.text
```

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

```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(data, 'html.parser')
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

```
# 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')
```

```
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
table_headers = first_launch_table.find_all('th')
# print(table_headers)
for j, table_header in enumerate(table_headers):
    name = extract_column_from_header(table_header)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

TASK 3: Create a data frame by parsing the launch HTML tables

We will create an empty dictionary with keys from the extracted column names in the previous task. Later, this dictionary will be converted into a Pandas dataframe

```
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']=[]
```

```
            # Launch outcome
            # TODO: Append the launch_outcome into launch_dict with key `Launch outcome`
            launch_outcome = list(row[7].strings)[0]
            launch_dict['Launch outcome'].append(launch_outcome)
            # print(launch_outcome)

            # Booster landing
            # TODO: Append the launch_outcome into launch_dict with key `Booster landing`
            booster_landing = landing_status(row[8])
            launch_dict['Booster landing'].append(booster_landing)
            # print(booster_landing)
```

After you have fill in the parsed launch record values into `launch_dict`, you can create a dataframe from it.

```
df=pd.DataFrame(launch_dict)
df.head()
```

Data Wrangling

- It was performed some Exploratory Data Analysis to find some patterns in the data and determine what would be the label for training supervised models
- It was calculated the number of launches at each site, and the number and occurrence of each orbits
- It was created the landing outcome label from outcome column
- The link to the notebook is the following: https://github.com/danielrodriguez23-sudo/DR_DataScience_Capstone_Project/blob/main/3_DR_SPACEX_Data_Wrangling.ipynb

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

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

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

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

TASK 4: Create a landing outcome label from Outcome column

Using the `Outcome`, create a list where the element is zero if the corresponding row in `Outcome` is in the set `bad_outcome`; otherwise, it's one. Then assign it to the variable `landing_class`:

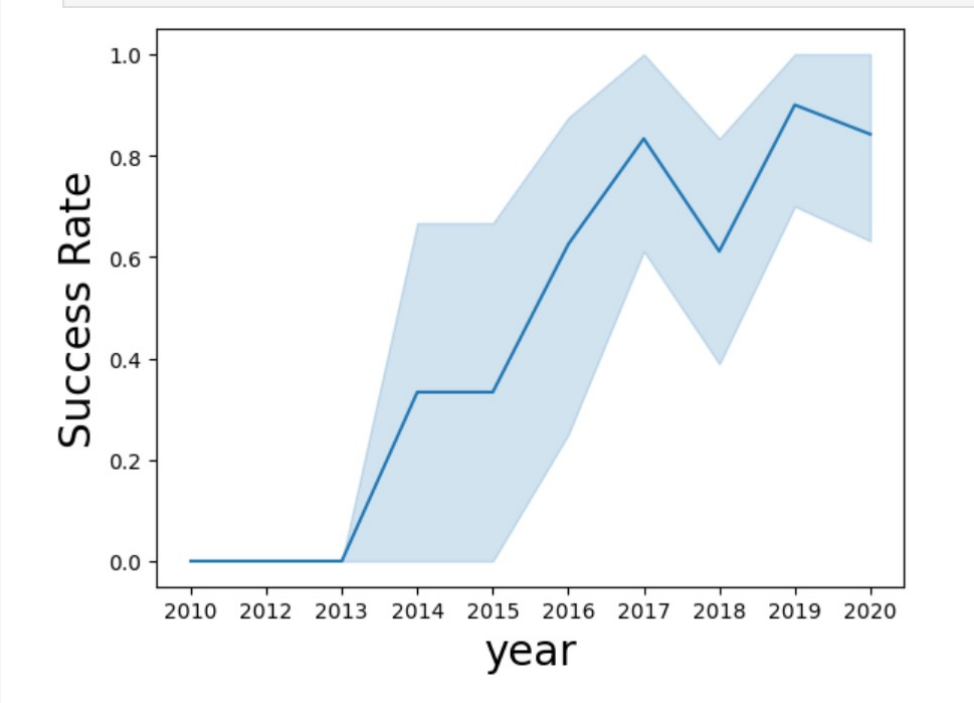
```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = [0 if x in bad_outcomes else 1 for x in df['Outcome']]
```

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

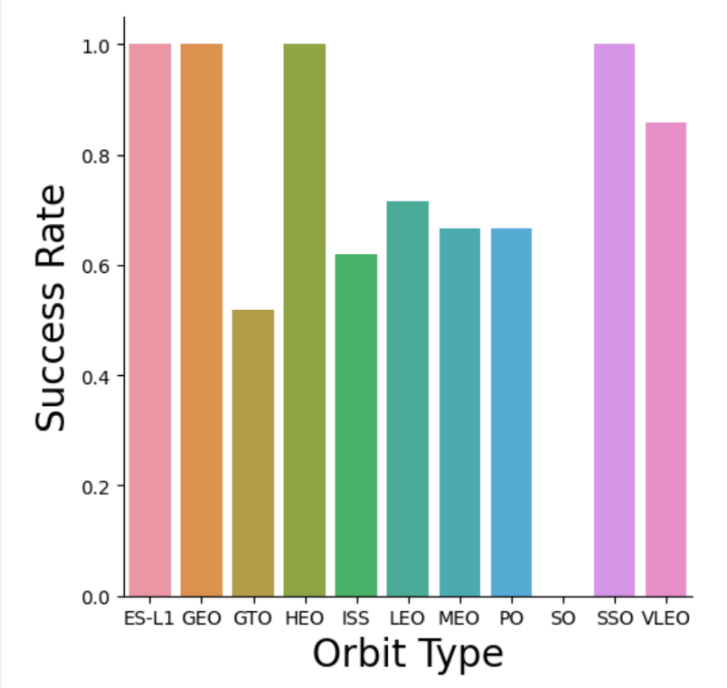
```
df['Class']=landing_class
```

EDA with Data Visualization

- It was checked the data visualizing several relationships such us flight number and launch site, payload and launch site, flight number and orbit type and the launch success yearly trend like the chart below



- The link of the notebook is the following: https://github.com/danielrodriguez23-sudo/DR_DataScience_Capstone_Project/blob/main/5_DR_SPACEX_EDA_Data_Visualization.ipynb
- The chart below represents the success rate of each orbit type.



EDA with SQL

- We performed SQL queries to get insight from the data:
  - Display the names of the unique launch sites in the space mission.
  - Display 5 records where launch sites begin with the string 'KSC'
  - 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 where the succesful landing outcome in drone ship was acheived.
  - List the names of the boosters which have success in ground pad 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, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
  - 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.

The link of the notebook is the following: https://github.com/danielrodriguez23-sudo/DR_DataScience_Capstone_Project/blob/main/4_DR_SPACEX_EDA_SQL.ipynb

Build an Interactive Map with Folium

- Regarding the task 1, it was marked all launch sites on a map.
- Which regards to task 2, it was added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- During that tasks, it was assigned the feature launch outcomes (failure or success) to class 0 and 1, being 0 for failure, and 1 for success. Using color-labeled marker clusters, it was identified which launch sites have relatively high success rate.
- Finally, as task 3 it was calculated the distances between a launch site to its proximities.
- The link of the notebook is the following: https://github.com/danielrodriguez23-sudo/DR_DataScience_Capstone_Project/blob/main/6_DR_SPACEX_Interactive_Visual_Analytics_Folium.ipynb

Build a Dashboard with Plotly Dash

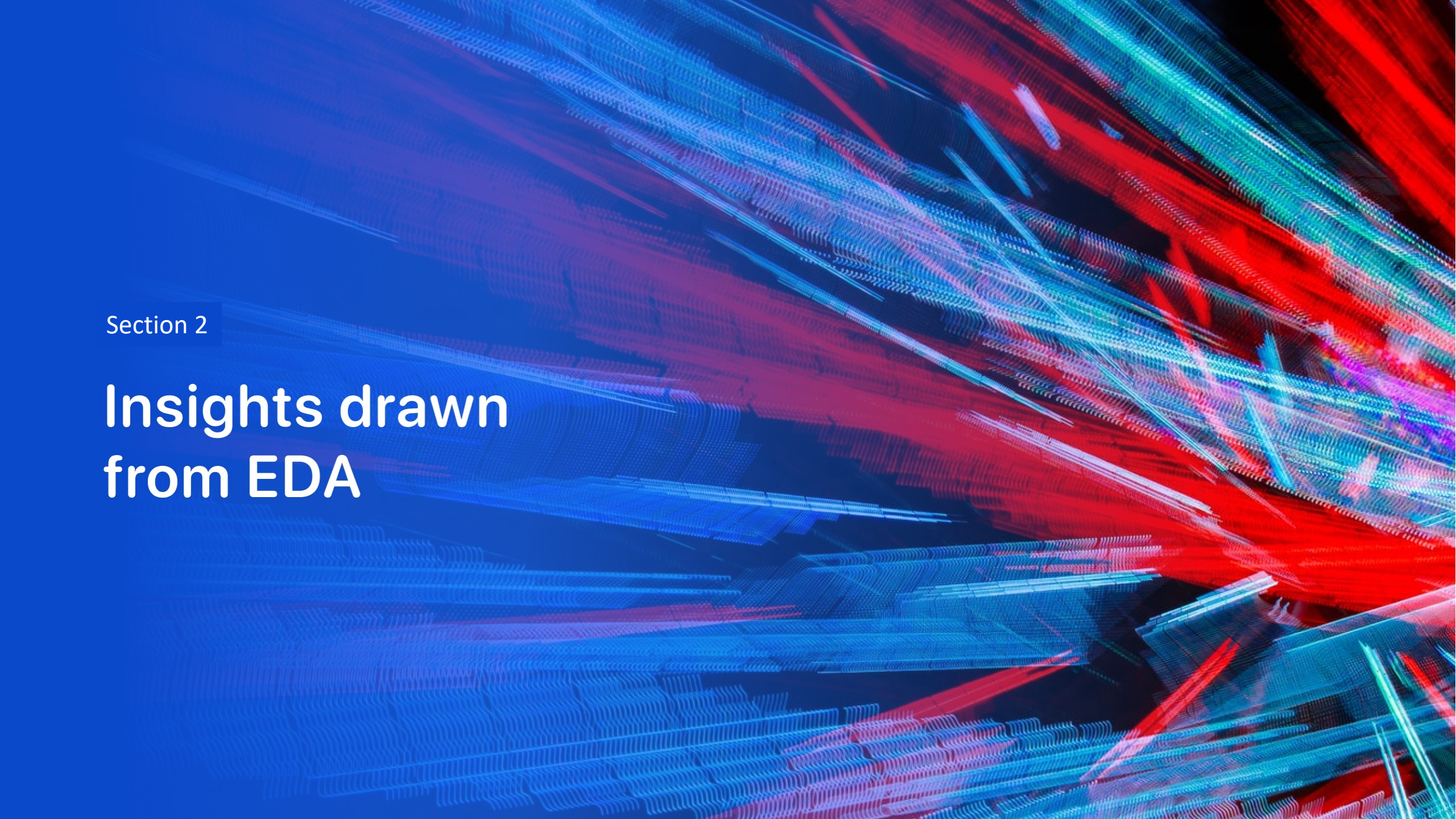
- It was built an interactive dashboard using Plotly dash with the following tasks:
  - It was plotted pie charts showing the total launches by a certain sites
  - Moreover, it was plotted too, scatter graph showing the correlation between Payload Mass (Kg) and Success for all sites for the different booster version.
- The link to the notebook is https://github.com/danielrodriguez23-sudo/DR_DataScience_Capstone_Project/blob/main/7_DR_SPACEX_dash_app.py

Predictive Analysis (Classification)

- It was loaded the data using the libraries numpy and pandas, then the data has been transformed and split into training and testing.
- It was set hyperparameters for each algorithm to GridSearchCV
- It was analyzed and taken into account the accuracy as the metric for our model.
- Compute accuracy for each model with the test dataset and it was built a Confusion Matrix
- It was chosen the best performing classification model.
- The link to the notebook is the following: https://github.com/danielrodriguez23-sudo/DR_DataScience_Capstone_Project/blob/main/8_DR_SPACEX_Machine_Learning_Prediction.ipynb

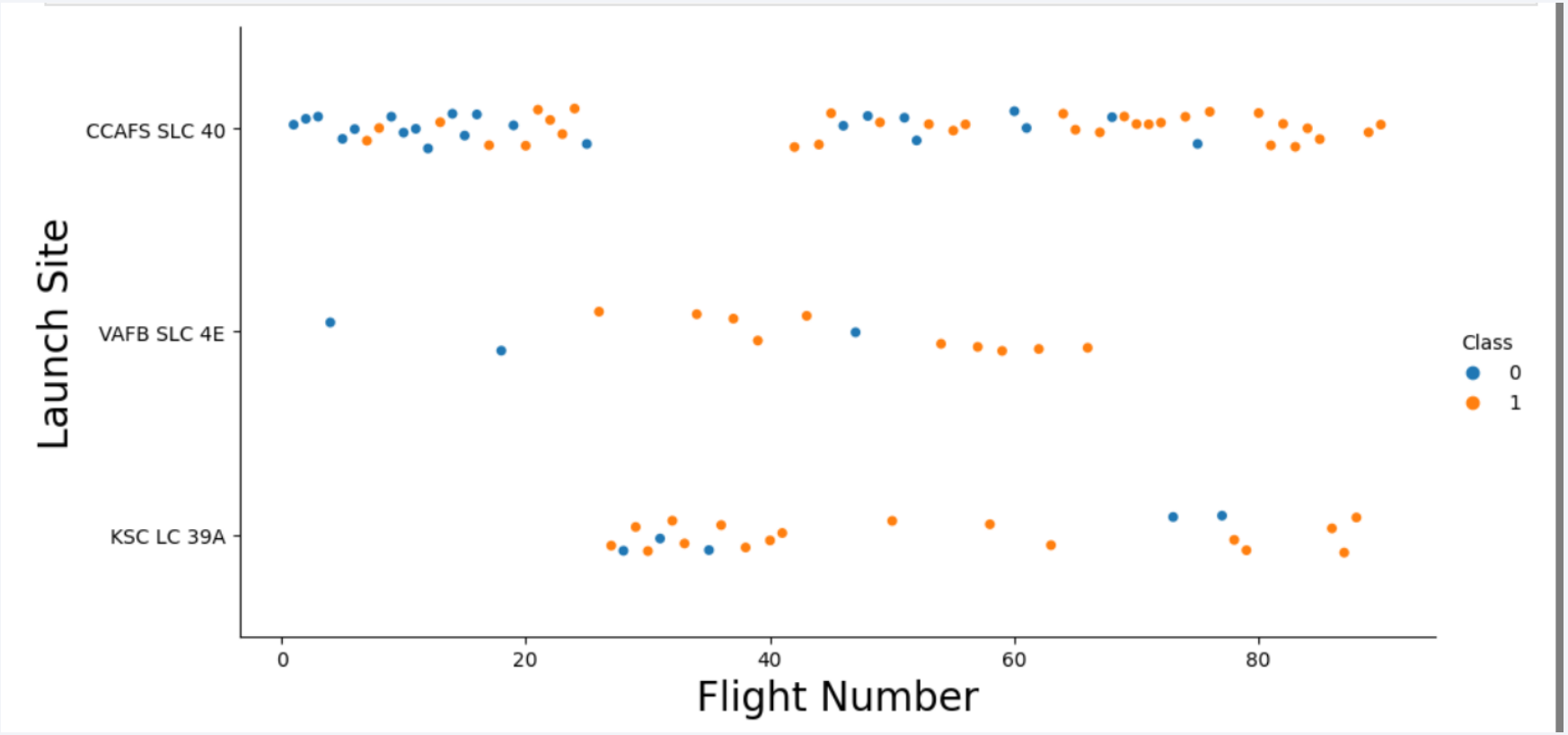
Results

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


Section 2
Insights drawn from EDA

Flight Number vs. Launch Site

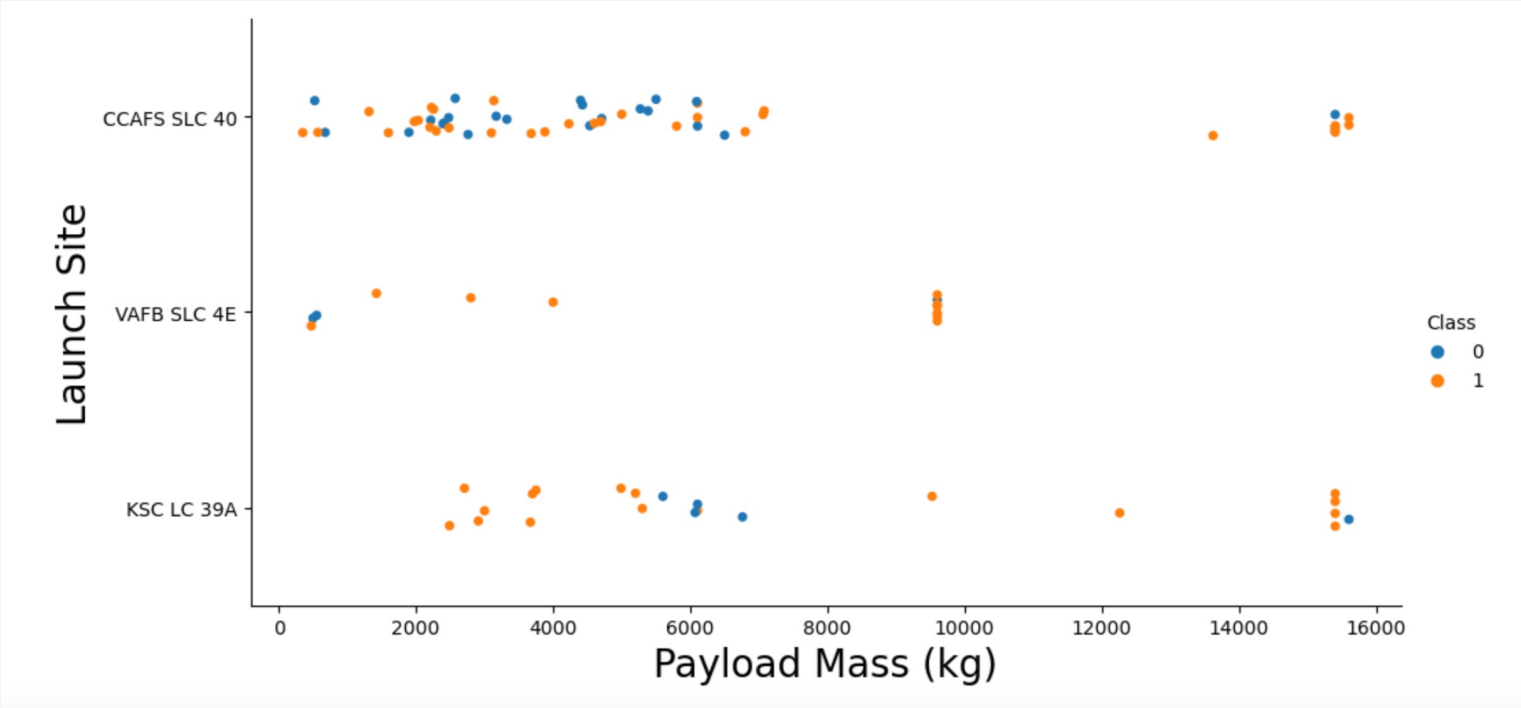
- Scatter plot of Flight Number vs. Launch Site



- It should be noted that, for each launch site, the larger the flight amount is the higher the success rate become.

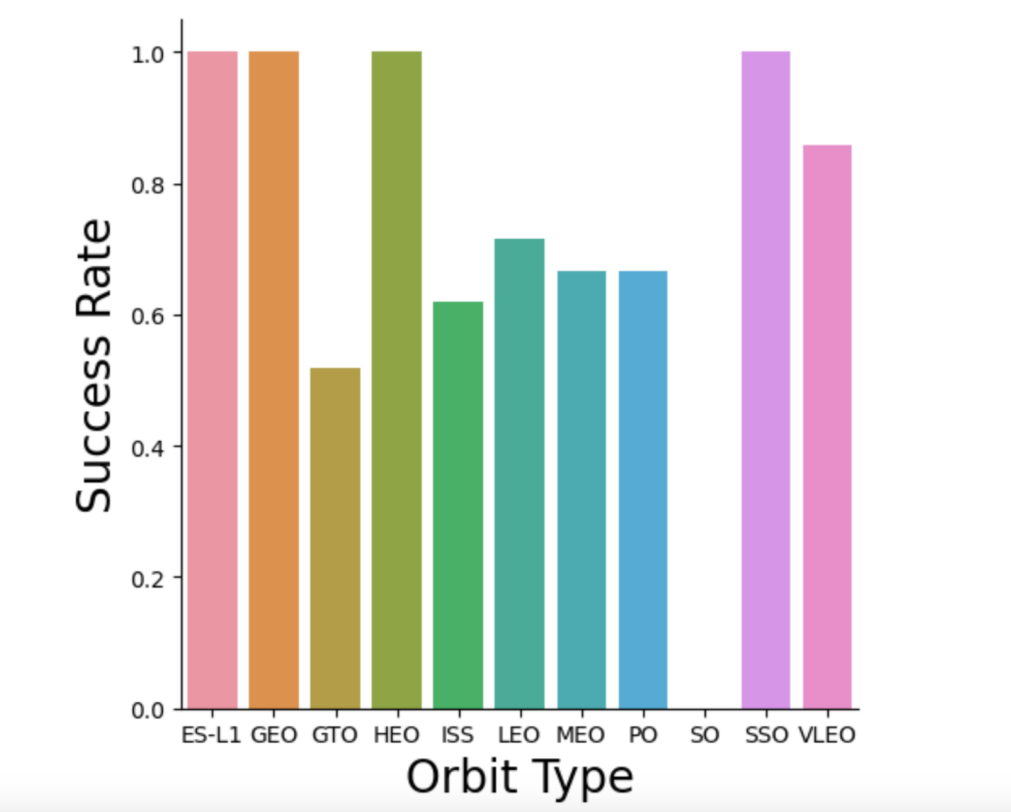
Payload vs. Launch Site

- Scatter plot of Payload vs. Launch Site



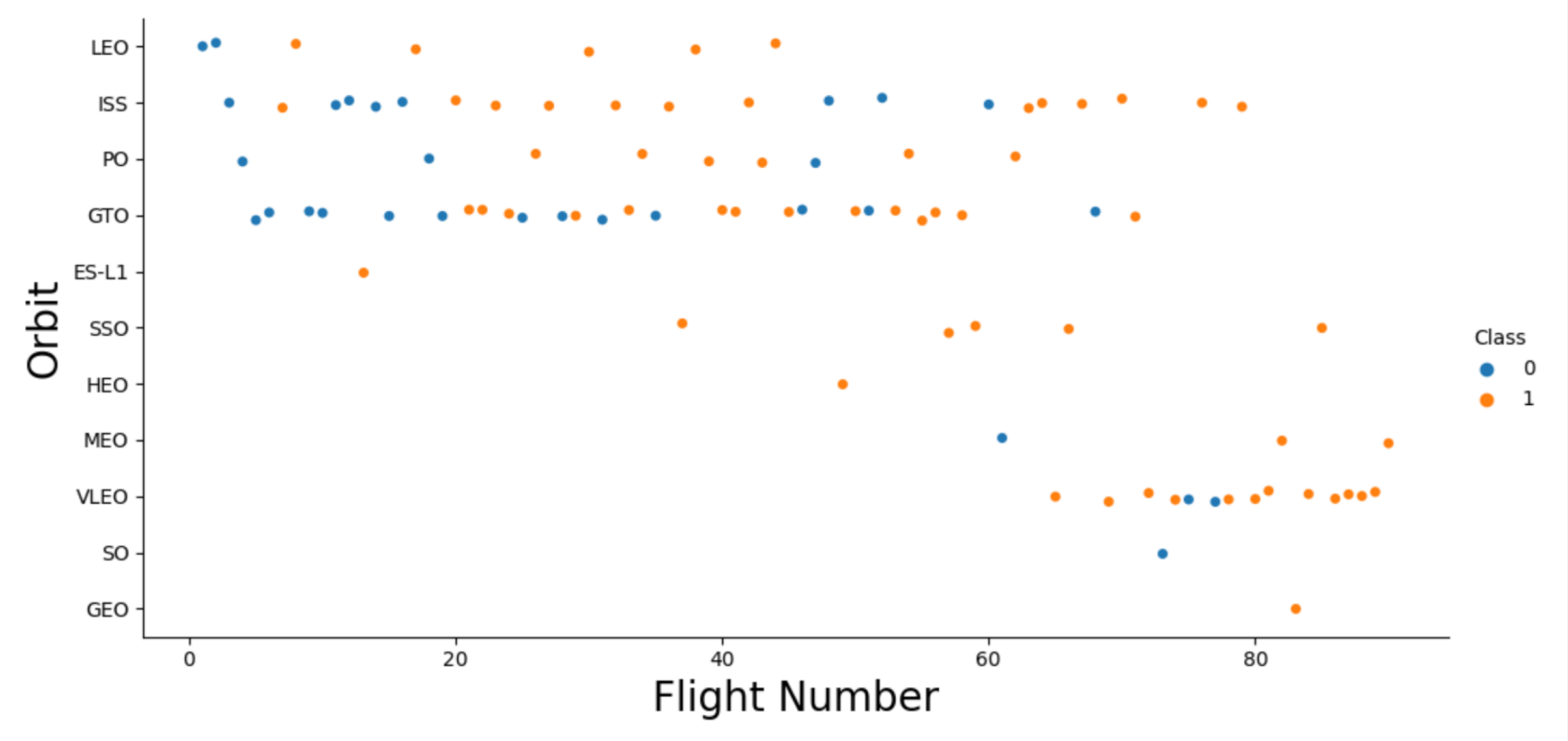
- It should be highlighted that on the one hand, a heavier payload could be a consideration for a successful landing. But, on the other hand, a too heavy payload could become a potential landing fail.

Success Rate vs. Orbit Type



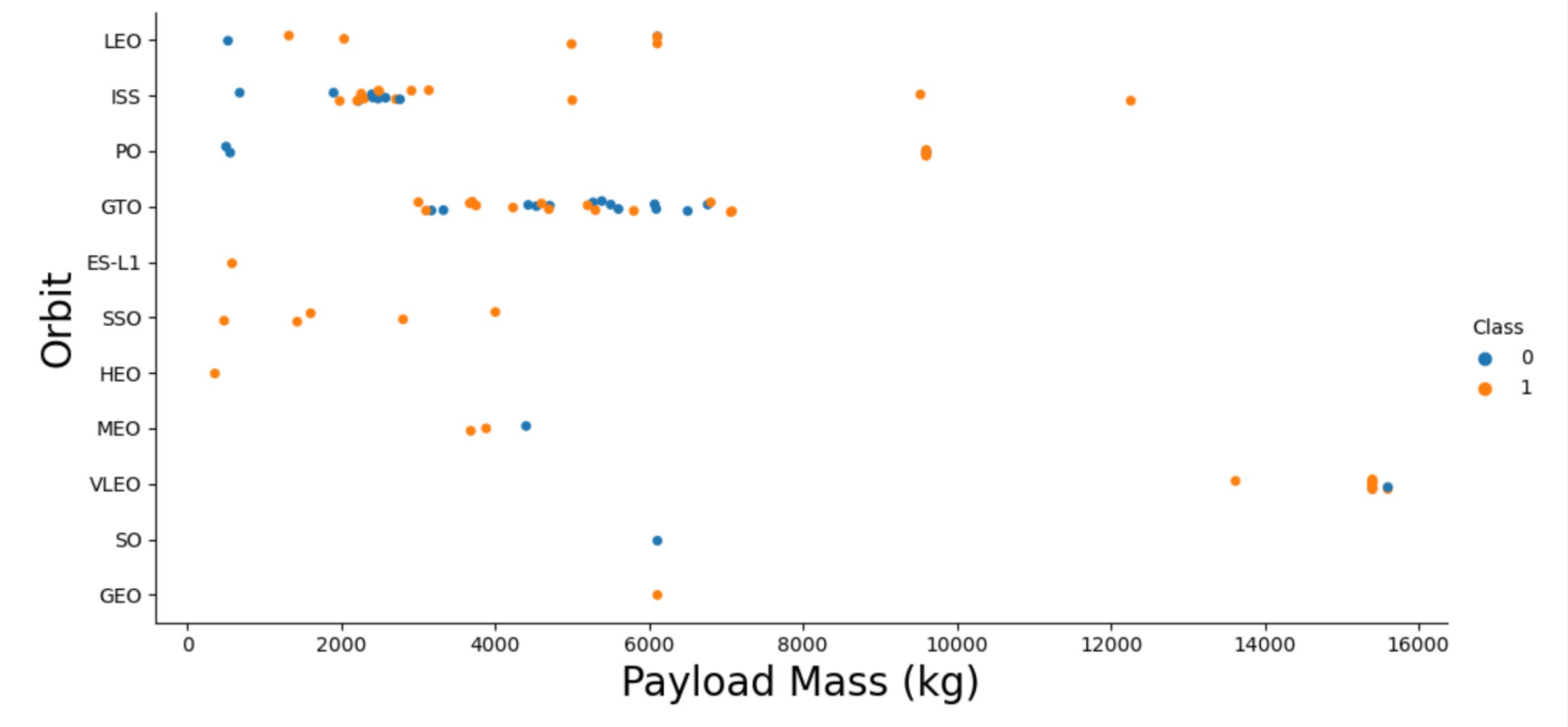
- In this bar chart it can be observed success rate for the different orbit types. In conclusión it should be noted that that ES-L1, GEO, HEO, SSO have the best success rate.

Flight Number vs. Orbit Type



- The plot above it can be observed that regarding the LEO orbit, success is directly related to the number of flights. On the other hand, in the GTO orbit, there isn't exist relationship between flight number and the orbit.

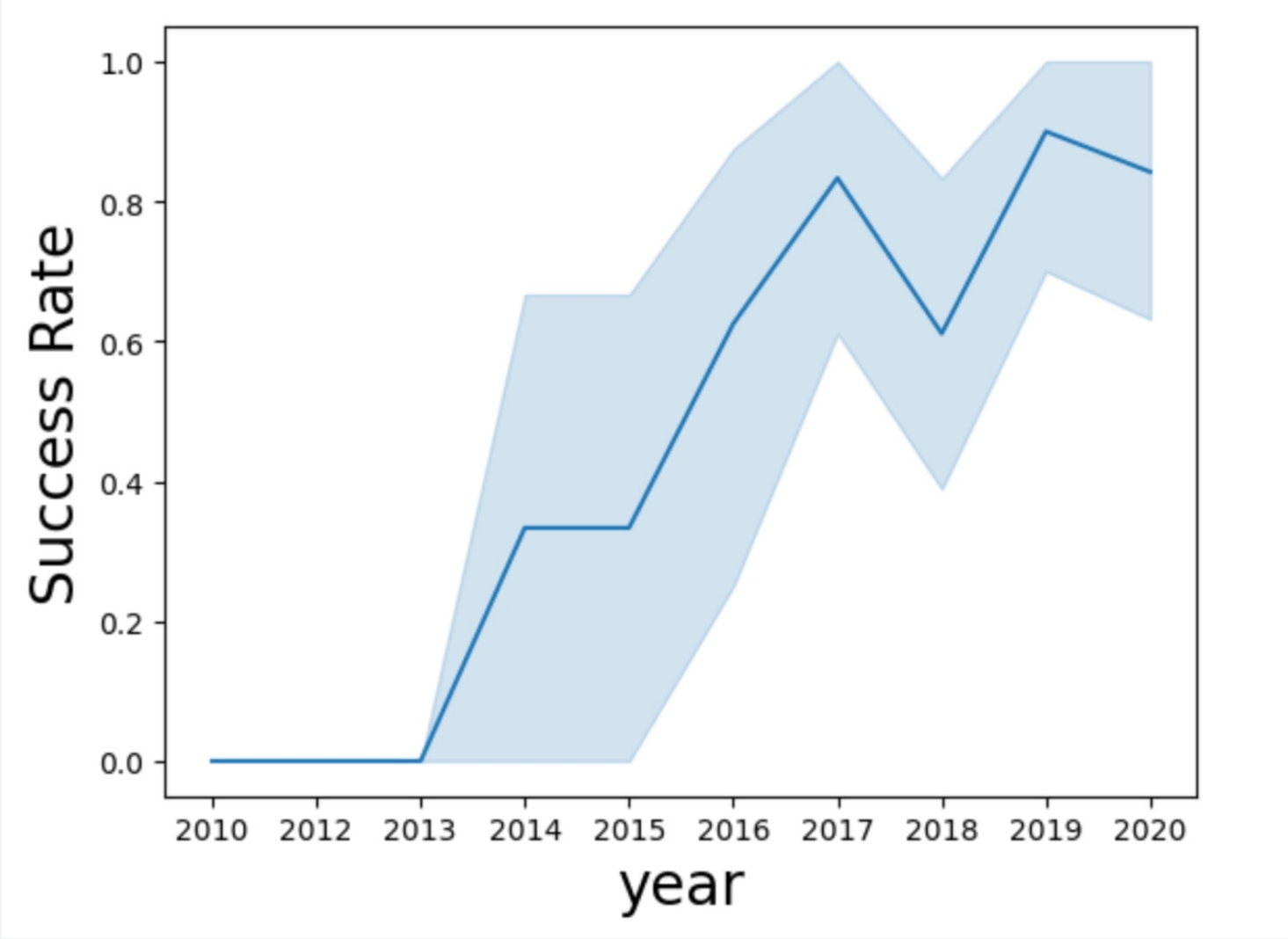
Payload vs. Orbit Type



- It should be highlighted the relevance of the payload weight for the success rate of the launches in some orbits. To conclude, heavier payloads improve the success rate for the LEO orbit. However, when decreasing the payload weight in a GTO orbit results that the success of a launch increases.

Launch Success Yearly Trend

- The success rate since 2013 kept increasing till 2020



All Launch Site Names

- It was used the "DISTINCT" in the query with the aim to remove duplicates Launch_Site and find the uniques values

Task 1

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

```
[8]: %sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL ORDER BY 1;
 * sqlite:///my_data1.db
Done.
```

[8]:

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

Launch Site Names Begin with 'KSC'

- It was selected all attributes from SPACEX table and WHERE clause followed by LIKE clause in order to filter launch sites that contain the substring KSC.
- Finally it was showed 5 records using LIMIT 5

Task 2

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

```
%sql SELECT * \
    FROM SPACEXTBL \
    WHERE LAUNCH_SITE LIKE'KSC%' LIMIT 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success
2017-01-05	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success

Total Payload Mass

- It was calculated the total payload mass with the operation SUM and then it was return the ones where the customer is NASA (CRS).

Task 3

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

```
[12]: %sql SELECT SUM(PAYLOAD_MASS__KG_) \
          FROM SPACEXTBL \
          WHERE CUSTOMER = 'NASA (CRS)';
```

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

[12]:

SUM(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- It was calculated the average of all payload masses using AVG and selected where the booster version contains the string ‘F9 v1.1’

Task 4 ¶

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

[13]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) \
    FROM SPACEXTBL \
    WHERE BOOSTER_VERSION = 'F9 v1.1';
```

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

[13]:

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

- Following the hint, using the MIN function it was showed that the first successful landing outcome on drone ship was 27th May 2016

Task 5

List the date where the succesful landing outcome in drone ship was acheived.

Hint:Use min function

[14]:
```
%sql SELECT MIN(DATE) \
FROM SPACEXTBL \
WHERE LANDING_OUTCOME = 'Success (drone ship)'
```

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

[14]:

MIN(DATE)
2016-05-27

Successful Ground pad Landing with Payload between 4000 and 6000

- In my notebook as it can be seen was not drone ship but ground pad. For that purpose, it was selected the booster version from the main table, where the landing_outcome was ‘Success ground pad’ and set the payload mass between 4000 & 6000.

Task 6

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

[11]:
```
%%sql select BOOSTER_VERSION from SPACEXTBL where LANDING_OUTCOME='Success (ground pad)'
and PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000;
```

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

[11]:

Booster_Version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

- It was used “Count” to calculate the total of mission outcomes and then with the clause Group by we can note the different outcomes. At the end 100 missions were successful and 1 was a failure.

Task 7

List the total number of successful and failure mission outcomes

16]:
```
%sql SELECT MISSION_OUTCOME, COUNT(*) as total_number \
FROM SPACEXTBL \
GROUP BY MISSION_OUTCOME;
```

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

16]:

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

Boosters Carried Maximum Payload

- First, it was selected the booster versión form the main table and then, was used a subquery to filter data by returning the heaviest payload mass with the MAX function. At the end the query returns the booter versions with the heaviest payload mass.

Task 8

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

[17]:
```
%sql SELECT BOOSTER_VERSION \
FROM SPACEXTBL \
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL);
```

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

[17]:

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

- Honestly, I followed the hints and instructions as can be seen in the image but I couldn't showed the correct answer.

Task 9

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2017' for year.

[24]:
```
%%sql
SELECT substr("Date",4,2) AS Month, "BOOSTER_VERSION","LAUNCH_SITE","Landing_Outcome"
FROM SPACEXTBL where "Landing_Outcome" = 'Success (ground pad)' and substr("Date",7,4)='2017';
```

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

[24]:

Month	Booster_Version	Launch_Site	Landing_Outcome

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

- It was selected to show the colums landing_outcome and a new column called count_launches filtering the date range required.

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

19]:
```
%%sql SELECT LANDING_OUTCOME, COUNT(*) AS COUNT_LAUNCHES FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING_OUTCOME ORDER BY COUNT_LAUNCHES DESC;
```

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

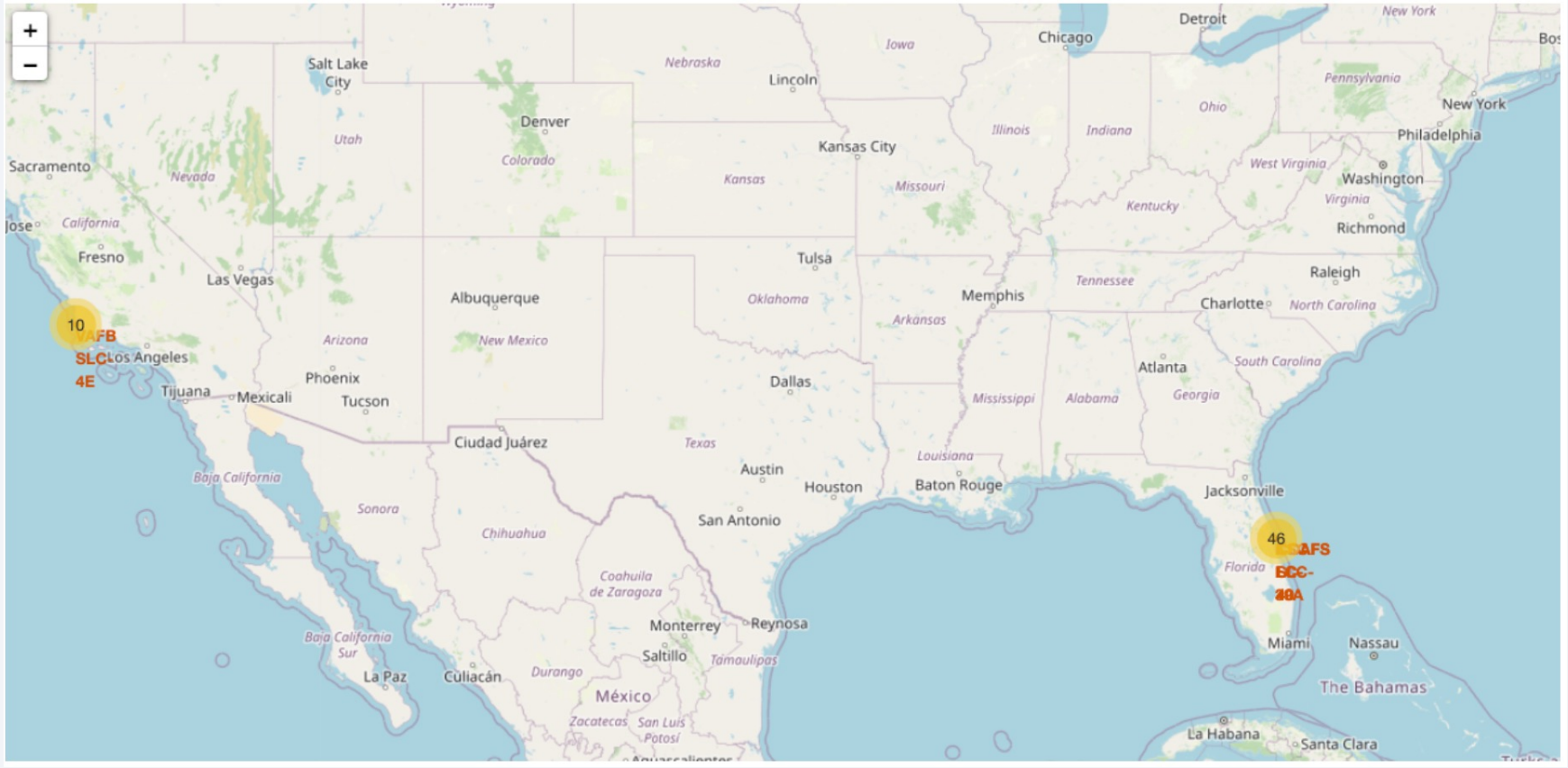
19]:

Landing_Outcome	COUNT_LAUNCHES
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

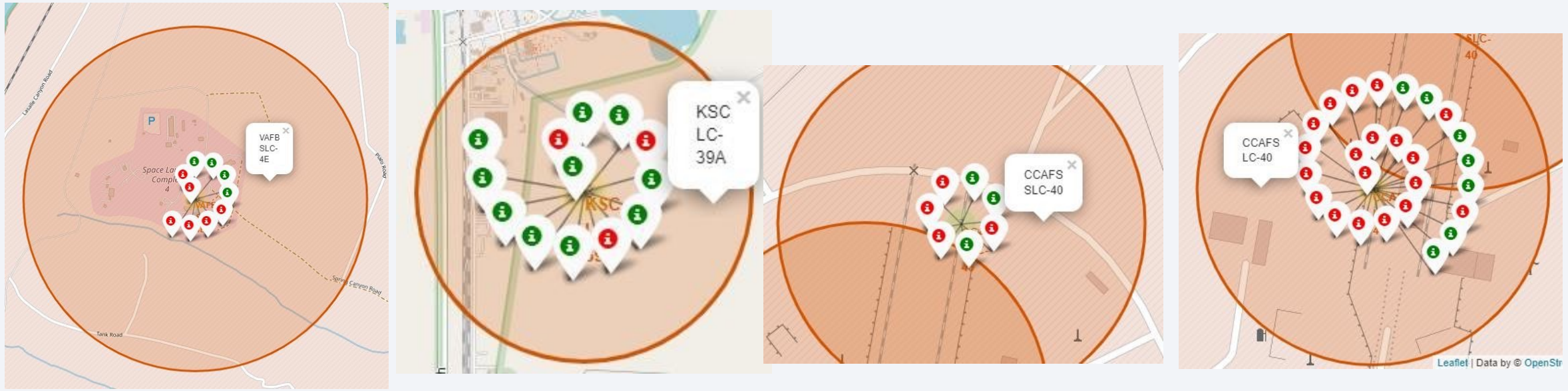
All launch sites location markers on a global map

- It was showed that see that Space X launch sites are located on the coast of the United States as it can be seen with the markers

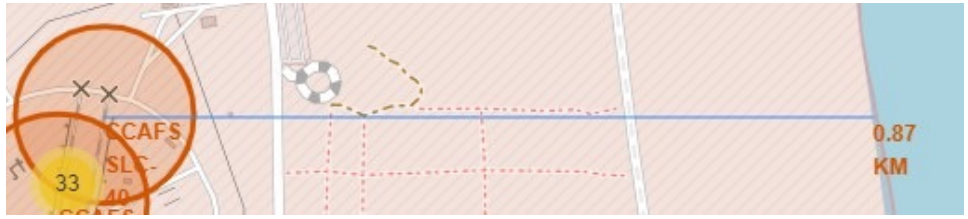


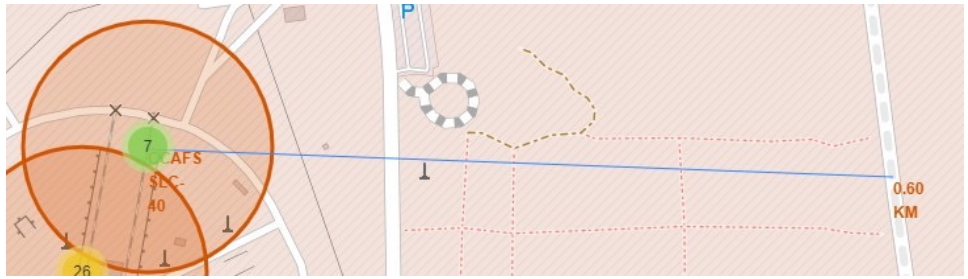
Color-labeled launch outcomes with markers

- It should be noted that green points represents successful launches and red points represents failure launches.



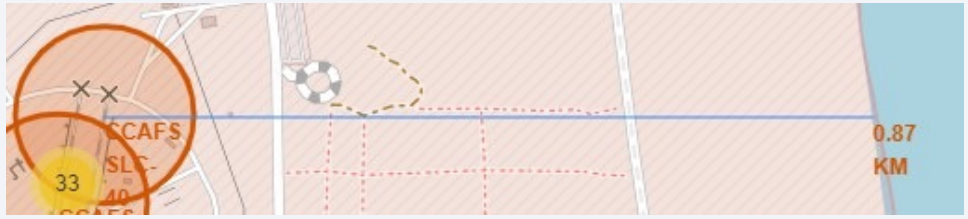
Folium Map – Distances between CCAFS SLC-40 and its proximities

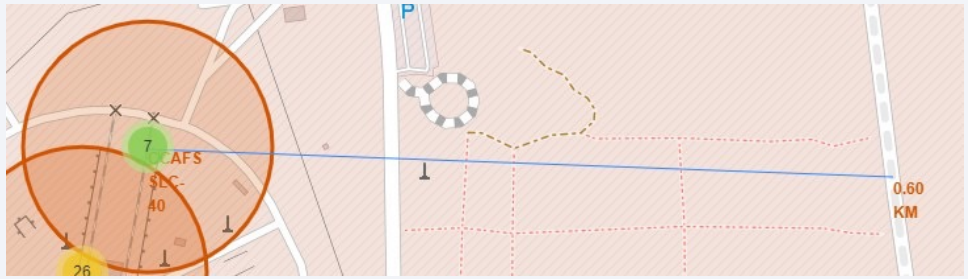




Is CCAFS SLC-40 in close proximity to railways ? Yes
Is CCAFS SLC-40 in close proximity to highways ? Yes
Is CCAFS SLC-40 in close proximity to coastline ? Yes
Do CCAFS SLC-40 keeps certain distance away from cities ? No

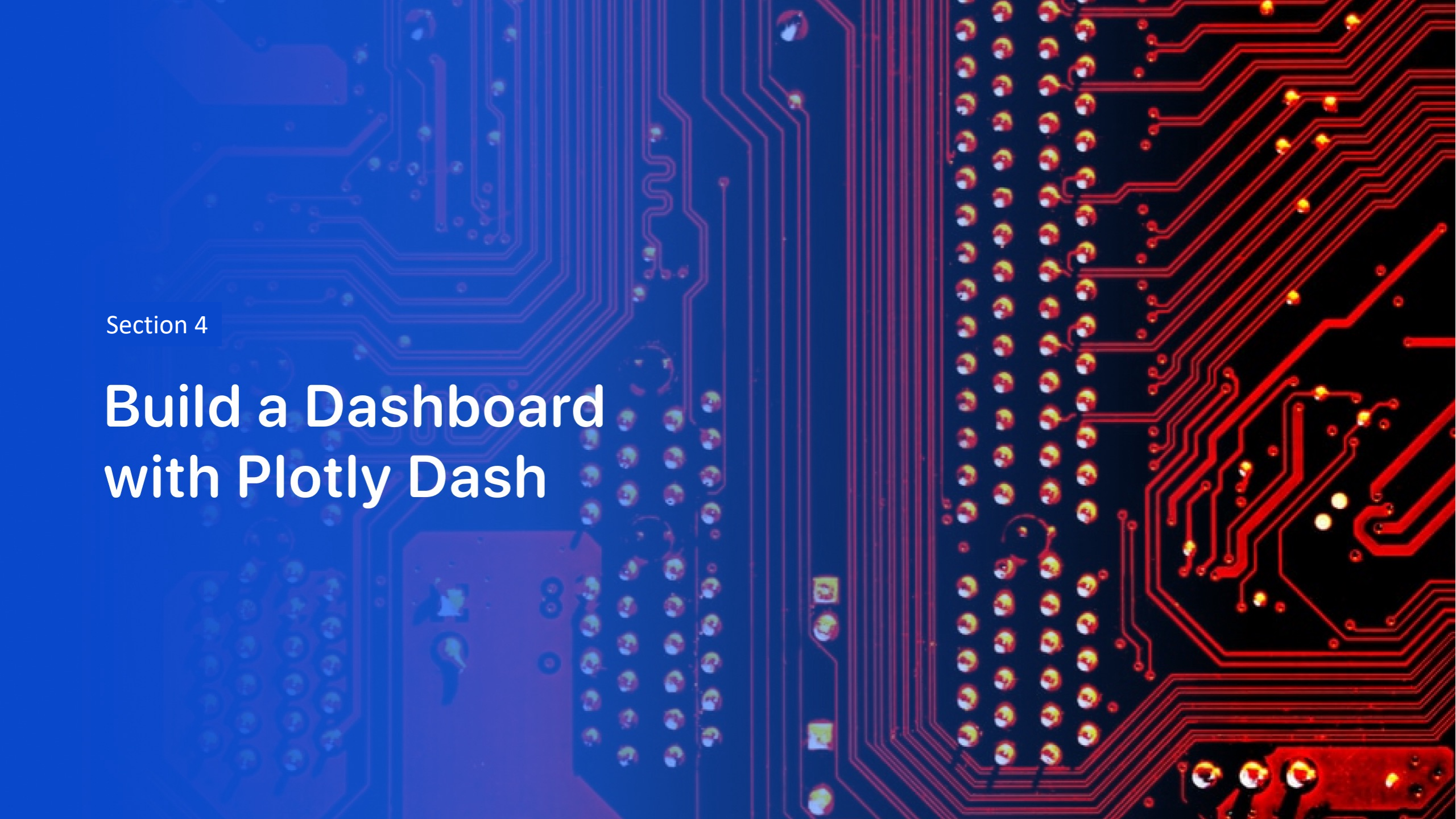
Distances between CCAFS SLC-40 and its proximities>



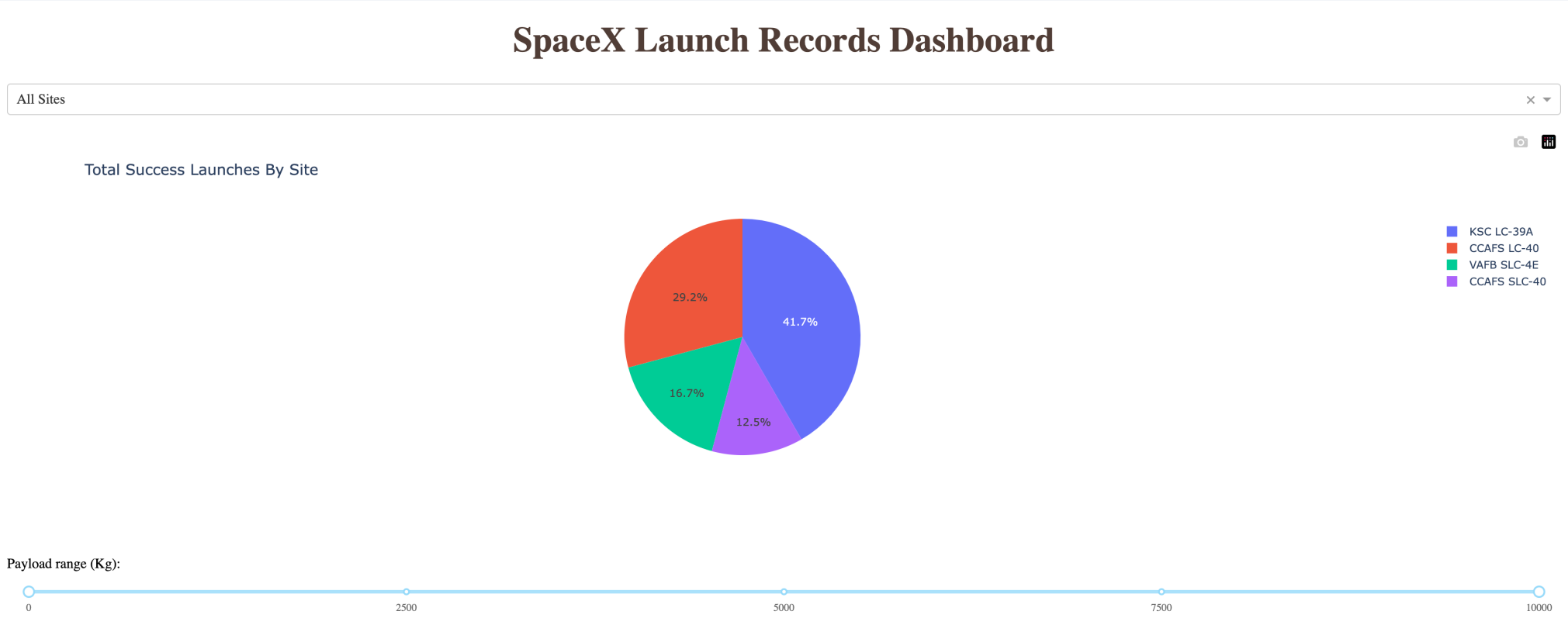


MAIN FINDINGS:

1) Is CCAFS SLC-40 in close proximity to railways ? Yes
2) Is CCAFS SLC-40 in close proximity to highways ? Yes
3) Is CCAFS SLC-40 in close proximity to coastline ? Yes
4) Do CCAFS SLC-40 keeps certain distance away from cities ? No

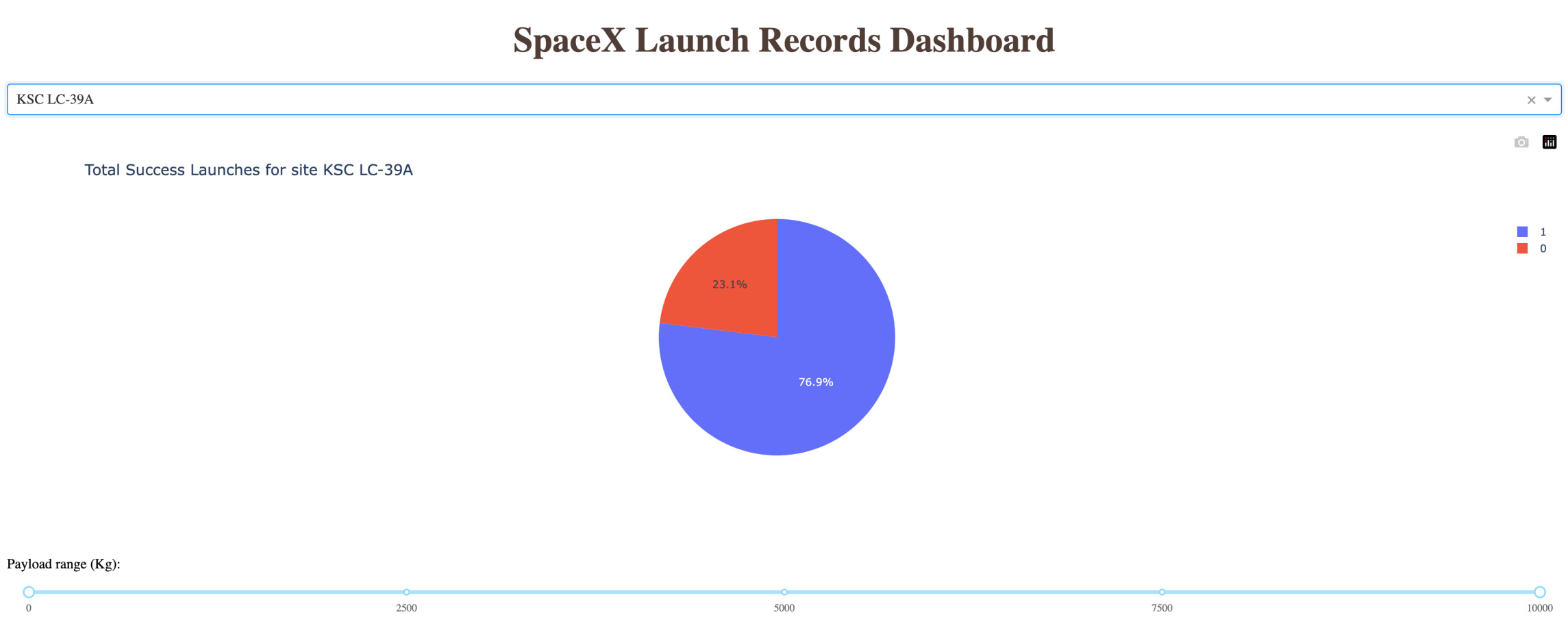

Section 4
Build a Dashboard with Plotly Dash

Launch success count for all sites



- It can be seen in the piechart that KSC LC-39A has the best success rate of launches.

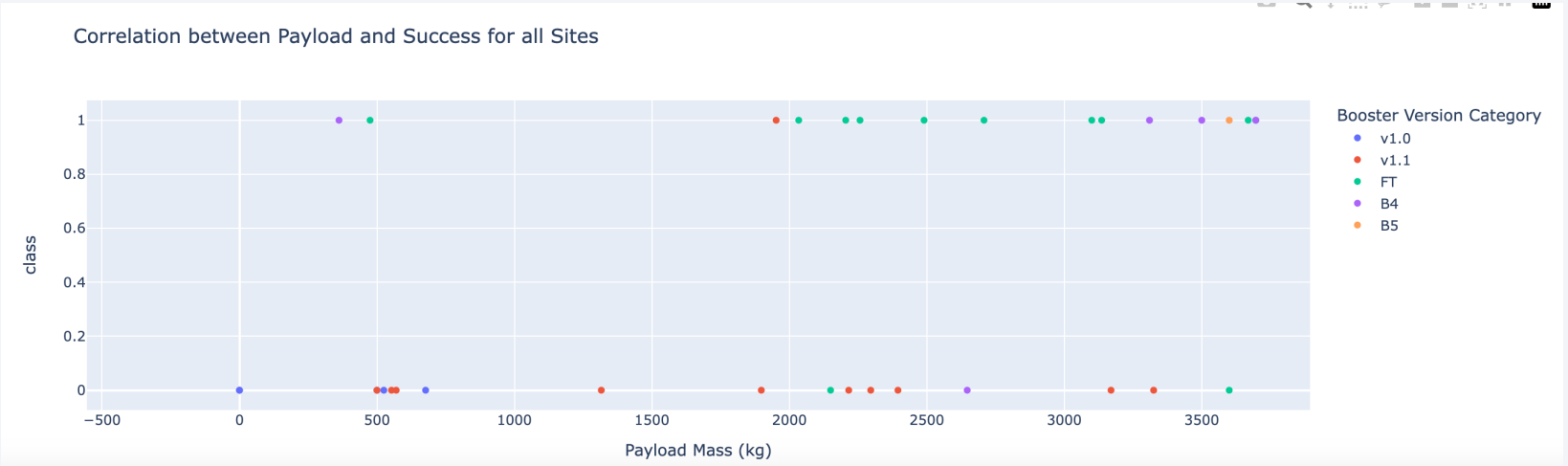
Launch site with highest launch success ratio



- It can be seen that KSC LC-39A has 76.9% success rate and 23.1% failure rate.

Dashboard- Different Payload mass selected & Outcome for all sites

Low weighted payload (0 – 3500 kg)



Heavy weighted payload (4000 – 10000 kg)

- It can be concluded that low weighted payloads have a better success rate than the heavy weighted payloads.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- It should be noted that all the models performed very similar in acuracy and score. If the .best_score is taken into account then it was found that Decision Tree is our best model.

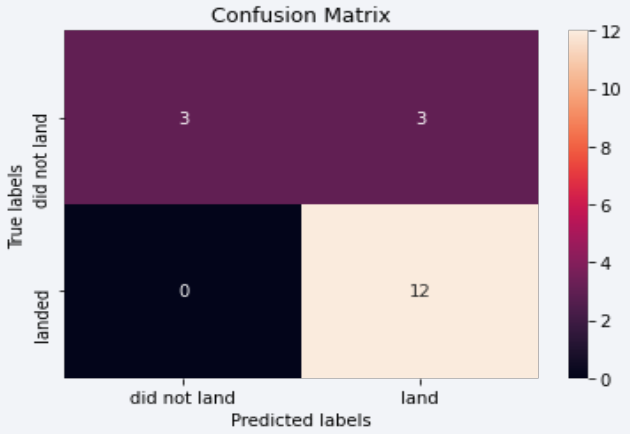
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best params is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best params is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best params is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best params is :', svm_cv.best_params_)
```

```
Best model is DecisionTree with a score of 0.9017857142857144
Best params is : {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_sa
mples_split': 10, 'splitter': 'random'}
```

Confusion Matrix

Logistic regression



Decision Tree

kNN

SVM

It should be highlighted that as the test accuracy are all equal, the confusion matrices are also identical. .

Conclusions

- 1) The orbits with the best success rates are GEO, HEO, SSO, ES-L1. On the other hand, KSC LC-39A had the most successful launches of any sites.
- 2) As shown in the chart, launch success rate started to increase in 2013 till 2020.
- 3) Payload mass can be a criteria to take into account for the success of a misión if the orbits are taken into account. It can be resumen that low weighted payloads perform better than the heavy weighted payloads.
- 4) For this dataset, the Decision Tree Algorithm is the best model even whether the test accuracy between all the models used is identical.

Thank you!