IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Nguyen The Nhut
17/05/2023

Outline

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

Executive Summary

Summary of methodologies

- Data Collection through API
- Data Collection with Web Scraping
- Data Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

Summary of all results

- Exploratory Data Analysis result
- Interactive analytics in screenshots
- Predictive Analytics result from Machine Learning Lab

Introduction

- **Project background and context**

SpaceX advertises Falcon 9 rocket launches on its website, with a cost of 62 million dollars; other providers cost upward of 165 million dollars each, much of the savings is because SpaceX 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 SpaceX for a rocket launch.

- **Problems you want to find answers**
  - If Falcon 9 will land successfully in the first launch?
  - How to predict using available data from SpaceX API and Wikipedia?
  - Which dependent variables best predict the outcome of landing?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Use GET REQUEST to obtain data from SPACEX API
  - Web scrape history of Falcon 9 Launch from Wikipedia
- Perform data wrangling
  - Convert outcomes into training label
- 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 API

Import helper functions → GET Request → Create DataFrame → Filter

Import Helper functions that will help us use the API to extract information using identification numbers in the launch data.

Request and parse the SpaceX launch data using the GET request

Extract data from API, create a dictionary with respective keys and convert it into a dataframe

Filter the dataframe to only include Falcon 9 launches

WEB SCRAPING

Import helper functions → GET Request → Extract columns name → Create DataFrame

Import helper functions to help process web scrape HTML

Request Data from URL and create a “soup” object

Collect all relevant column names from the HTML table header using FIND ALL method

Create a data frame by parsing the launch HTML tables

Data Collection – SpaceX API

Get Request from SPACEX API → Create Dictionary using helper functions → Create Dictionary → Transform dictionary into dataframe → Filter Falcon 9 only

CODE SNIPPET

In [13]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API
```

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

In [34]:
```
response.status_code
```

Out[34]: 200

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

In [35]:
```
# Use json_normalize meethod to convert the json result into a dataframe
data = response.json()
data = pd.json_normalize(data)
```

Using the dataframe `data` print the first 5 rows

In [36]:
```
# Get the head of the dataframe
data.head(5)
```

In [43]:
```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-packages/pandas/core/indexing.py:1773: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

Out[43]:

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	La
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	...
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6b

Github: REF

Data Collection - Scraping

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

response = requests.get(static_url)
```

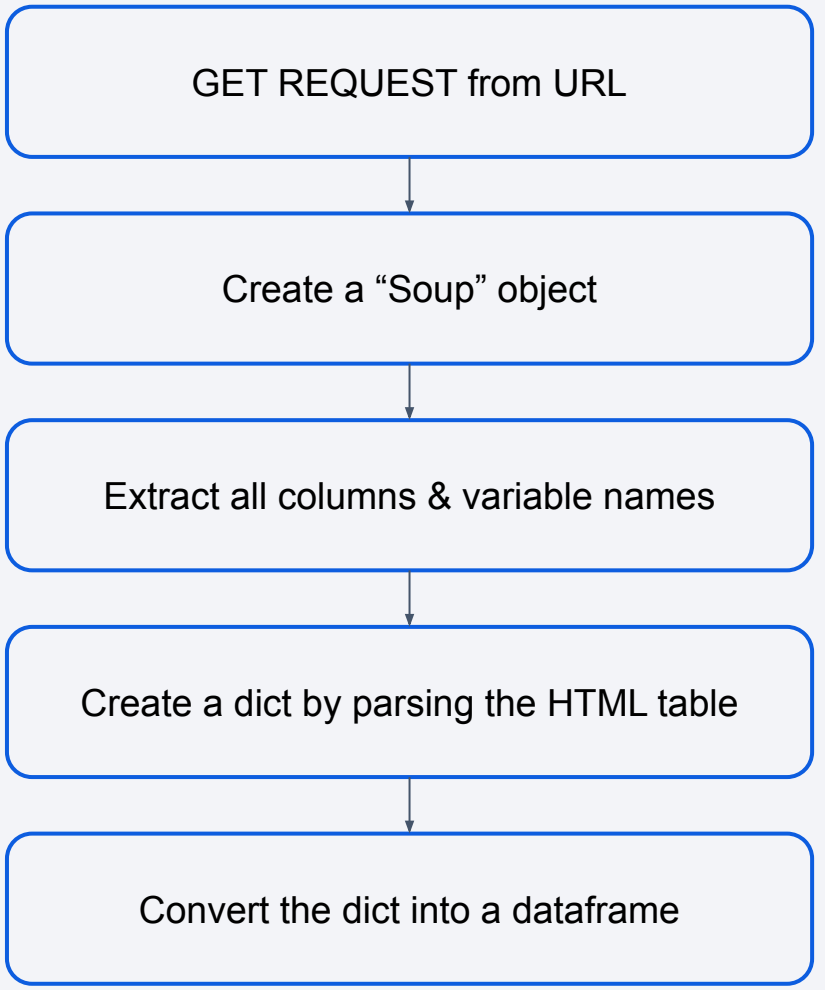
Create a `BeautifulSoup` object from the HTML `response`

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

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

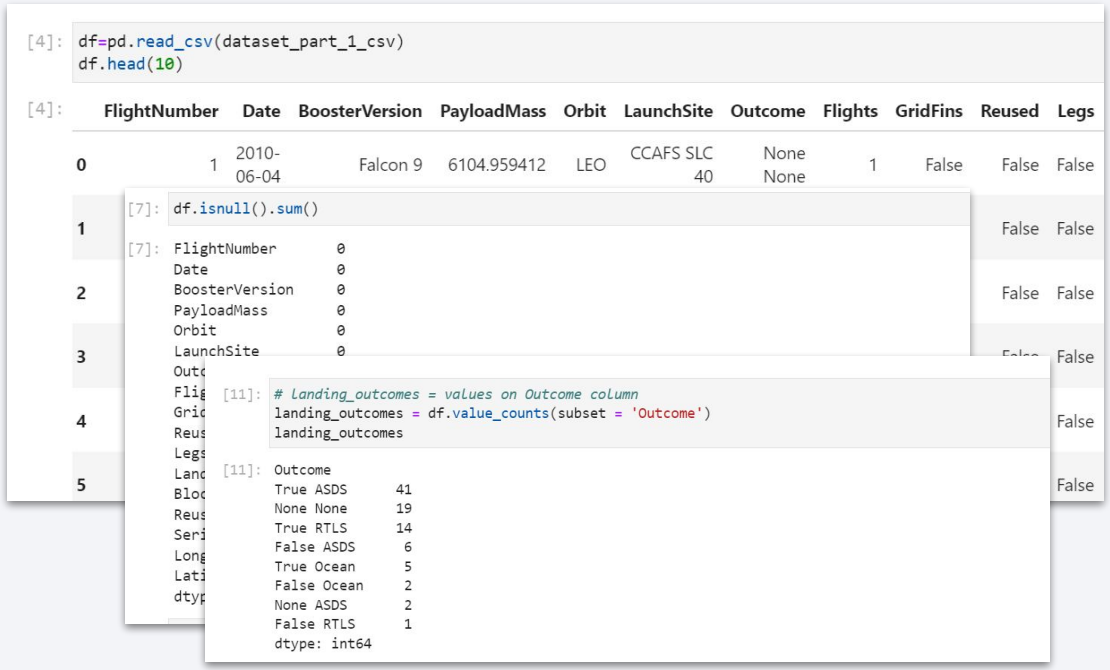
In [7]:
```
# Use soup.title attribute
soup.title
```

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



Github: REF

Data Wrangling



```
[4]: df=pd.read_csv(dataset_part_1_csv)
     df.head(10)
```

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1	False	False	False
1											False
2											False
3											False
4											False
5											False

```
[7]: df.isnull().sum()

[7]: FlightNumber      0
     Date              0
     BoosterVersion    0
     PayloadMass       0
     Orbit             0
     LaunchSite        0
```

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

[11]: 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
      dtype: int64
```

Read Data

Explore DataFrame (df) using methods: value_counts, dtypes and isnull,..

Filter “Outcome” from the dataframe and convert it into a List called **bad_outcome**

Create a **landing_class** where 1= success and 0=failed based on **bad_outcome**

Insert a new column into the df from **landing_class**

Github: REF

EDA with Data Visualization

01	**Scatter Point (Outcome vs Payload & Flight Number)**	• Understand the relationship between the launching outcome vs Payload and Flight Number.
02	**Scatter Point (Outcome vs LaunchSite & Flight Number)**	• Understand the relationship between the launching outcome vs LaunchSite and Flight Number.
03	**Scatter Point (Outcome vs Payload & LaunchSite**	• Understand the relationship between the launching outcome vs Payload and LaunchSite.
04	**Bar Chart (Success Rate vs Orbit Type)**	• Compare the Success Rate among orbit types.
05	**Scatter Point (Outcome vs Flight Number & Orbit Type**	• Understand the relationship between the launching outcome vs Flight Number and Orbit type.
06	**Line Chart (AVG Success Rate over years)**	• See the trend of the average Success rate over years

Github: REF

EDA with SQL

SQL 1
Display the names of the unique launch sites in the space mission

```
%%sql select distinct("Launch_Site")
from SPACEXTBL;
```

SQL 2
Display 5 records where launch sites begin with the string 'CCA'

```
%%sql select *
from SPACEXTBL
where "Launch_Site" like 'CCA%'
limit 5;
```

SQL 3
Display the total payload mass carried by boosters launched by NASA (CRS)

```
%%sql select SUM("PAYLOAD_MASS__KG_") AS "TOTAL PAYLOAD"
from SPACEXTBL
where "Customer" = "NASA (CRS)";
```

SQL 4
Display average payload mass carried by booster version F9 v1.1

```
%%sql SELECT ROUND(AVG("PAYLOAD_MASS__KG_"))
FROM SPACEXTBL
WHERE "Booster_Version" like 'F9 v1.1%';
```

SQL 5
List the date when the first successful landing outcome in ground pad was acheived.

```
%%sql SELECT MIN("Date") AS "The first date" FROM SPACEXTBL
WHERE "Landing_Outcome" = 'Success (ground pad)';
```

SQL 6
List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000¶

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


Github: REF

EDA with SQL

SQL 7
List the total number of successful and failure mission outcomes

```
%%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome")
FROM SPACEXTBL
GROUP BY "Mission_Outcome";
```

SQL 8
List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

```
%%sql SELECT "Booster_Version"
FROM SPACEXTBL
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
```

SQL 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.

```
%%sql SELECT substr("Date",4,2) AS "Month",
"Landing_Outcome",
"Booster_Versions",
"Launch_Site"
FROM SPACEXTBL
WHERE substr("Date",7,4) = '2015'
AND "Landing_Outcome" = 'Failure (drone ship)';
```

SQL 10
Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

```
%%sql SELECT "Landing_Outcome",
COUNT("Landing_Outcome") AS "COUNT OF LANDING_OUTCOME"
FROM SPACEXTBL
GROUP BY "Landing_Outcome"
HAVING "Landing_Outcome" like 'Success%'
OR "Landing_Outcome" = 'Success'
AND "Date" BETWEEN '04-06-2010' AND '20-03-2017'
ORDER BY "COUNT OF LANDING_OUTCOME" DESC ;
```


Github: REF

Build an Interactive Map with Folium

01	**Circle Object**	• Add a highlighted circle area with a text label on a specific coordinate.
02	**Popup**	• Show the name of a specific coordinate when click the circle.
03	**Marker**	• Name the circle object and all the launch sites. Also use colors to show the success/fail of each site.
04	**Mouse Position**	• Get coordinate for a mouse over a point on the map
05	**PolyLine**	• Draw a line between two places on the map

Github: REF

Build a Dashboard with Plotly Dash

01	**Dropdown**	• Add a drop down list of launch sites as an input for the pie chart below.
02	**Pie chart**	• An interactive pie chart with the input from dropdown list to show and compare the success rate of launch sites.
03	**Range Slider**	• Add a range slider of payload as an input for the scatter plot
04	**Scatter Plot**	• An interactive scatter plot to see the distribution of success rate with the input of launch site and payload for each booster version.

Github: REF

Predictive Analysis (Classification)

```
URL2 = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datas
resp2 = await fetch(URL2)
text2 = io.BytesIO((await resp2.arrayBuffer()).to_py())
X = pd.read_csv(text2)
```

```
X.head(100)
```

```
# students get this
transform = preprocessing.StandardScaler().fit(X)
X = transform.transform(X)
X
```

```
array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.67441914e+00, -1.19523159e+00, -6.53912840e-01, ...,
        -8.35531692e-01,  1.93309133e+00, -1.93309133e+00],
       [-1.63592675e+00, -1.16267307e+00, -6.53912840e-01, ...,
```

```
parameters ={'C':[0.01,0.1,1],
             'penalty':['l2'],
             'solver':['lbfgs']}
```

```
parameters ={"C":[0.01,0.1,1],'penalty':['l2'], 'solver':['lbfgs']}# l1 lasso l2 ridge
lr=LogisticRegression()
logreg_cv = GridSearchCV(lr, parameters, cv=10)
logreg_cv.fit(X_train, Y_train)
```

```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.01, 0.1, 1], 'penalty': ['l2'],
                         'solver': ['lbfgs']})
```

Load Dataframe

↓

Standardize data and convert to numpy arrays

↓

Split data into train and test sample, using train_test_split()

↓

Create model objects find the best param using **GridSearchCV**

↓

Compare the accuracy of each model using test sample with **score()** method

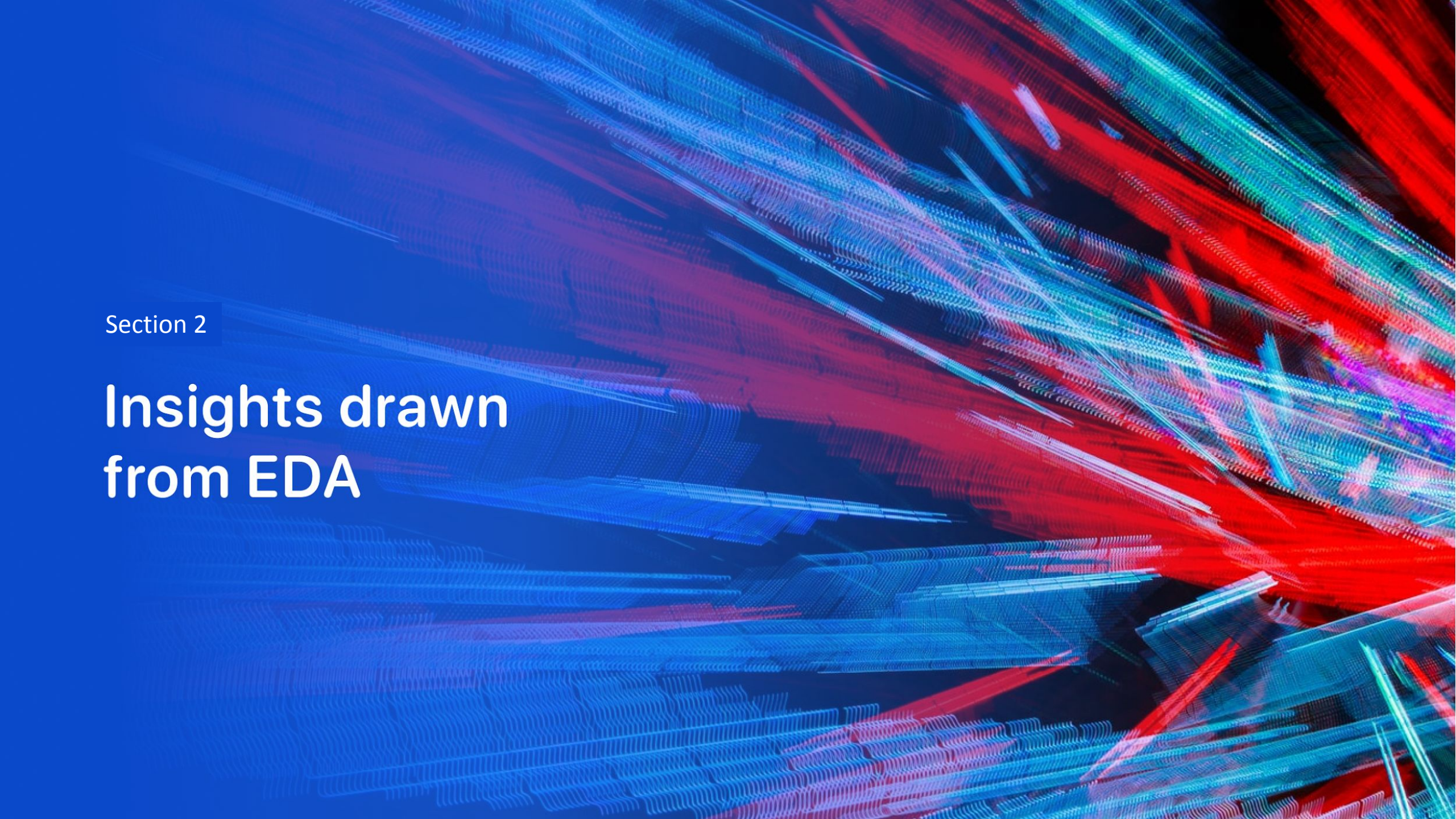
Github: REF

Results

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

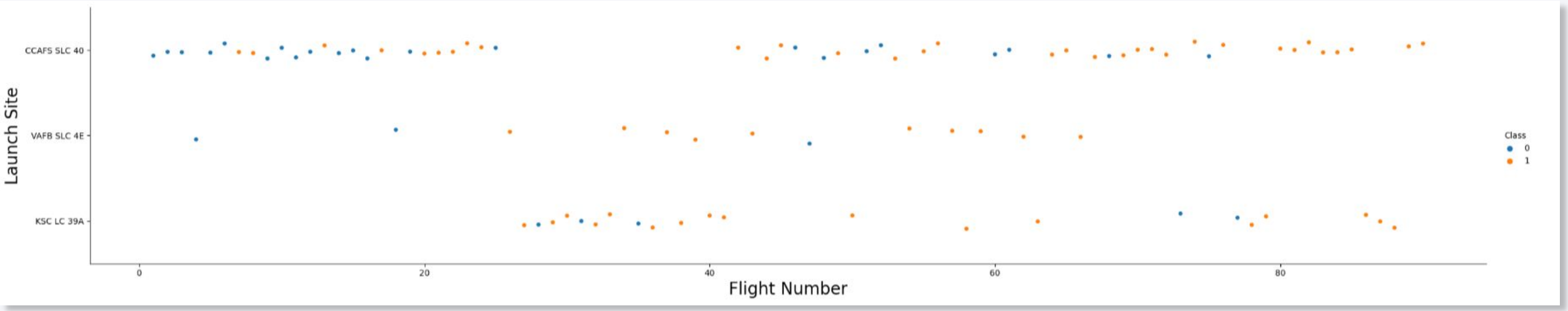
Section 2

Insights drawn from EDA



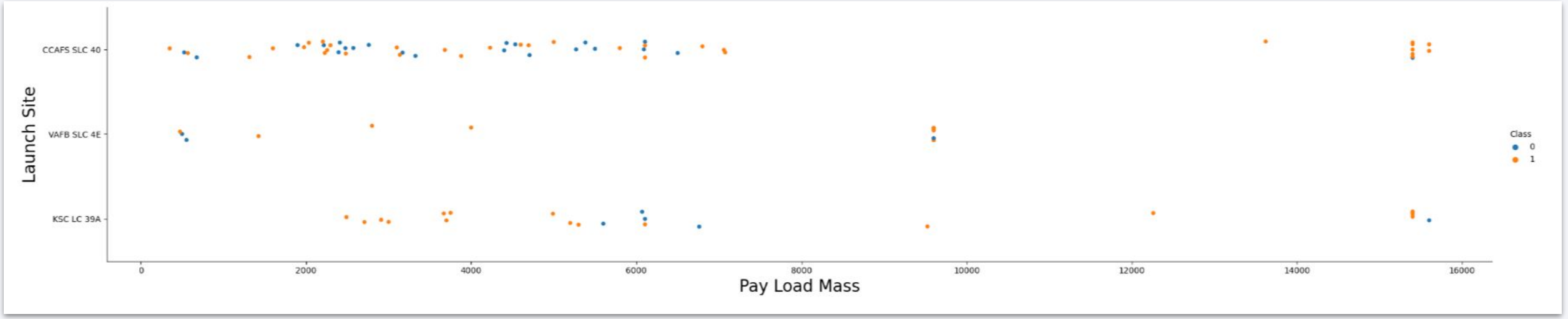
Flight Number vs. Launch Site

The success rate of Launch Sites have a positive correlation with Flight Number while the others show no sign of correlation with Flight Number.



Payload vs. Launch Site

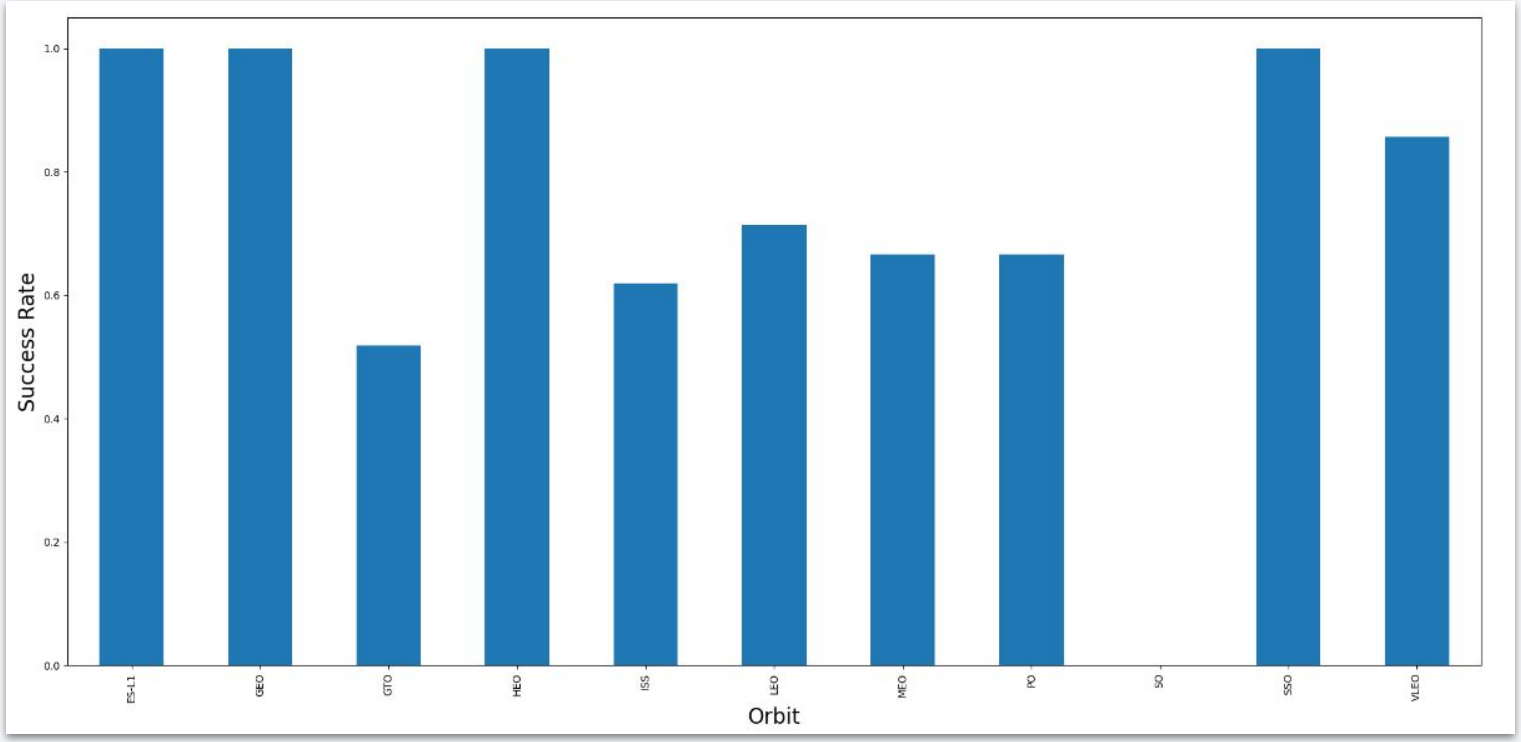
There is no rockets launched on VAFB-SLC have over 10000 heavy payload mass. We can see there is no clear correlation between LaunchSite and PayloadMass in terms of Success Rate.



Success Rate vs. Orbit Type

The bar chart shows the success rate of each orbit type, some of the orbit types, such as ES-L1, GEO, HEO, and SSO, have 100% success rate while SO have 0% success rate.

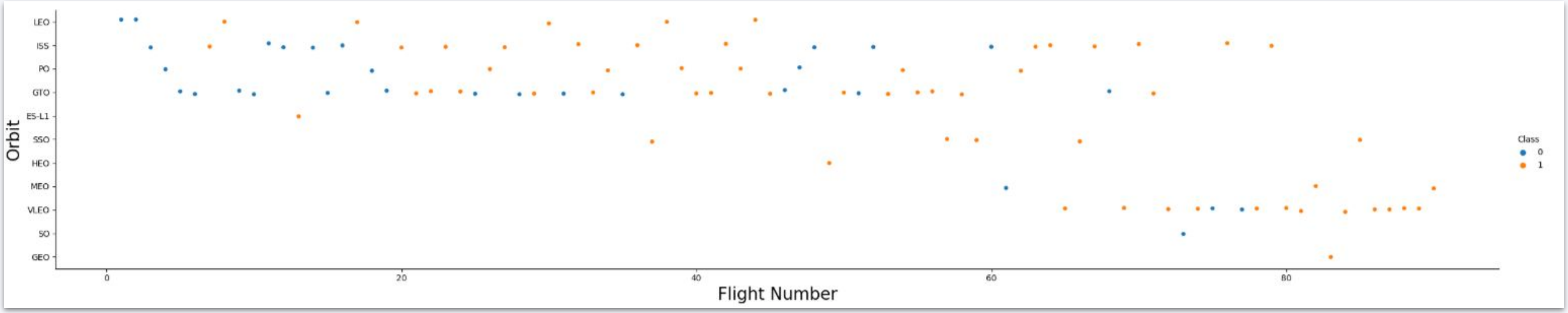
Moreover, there some orbit types have only one occurrences in the dataset, so we might need more data to clearly see the pattern of relationship of orbit type and success rate.



Flight Number vs. Orbit Type

For the orbit types such as LEO, ISS, PO, GTO and VLEO, the scatter plot show that the more rockets were launched, the higher success rate is while there is no clear sign of correlation for the other orbit types.

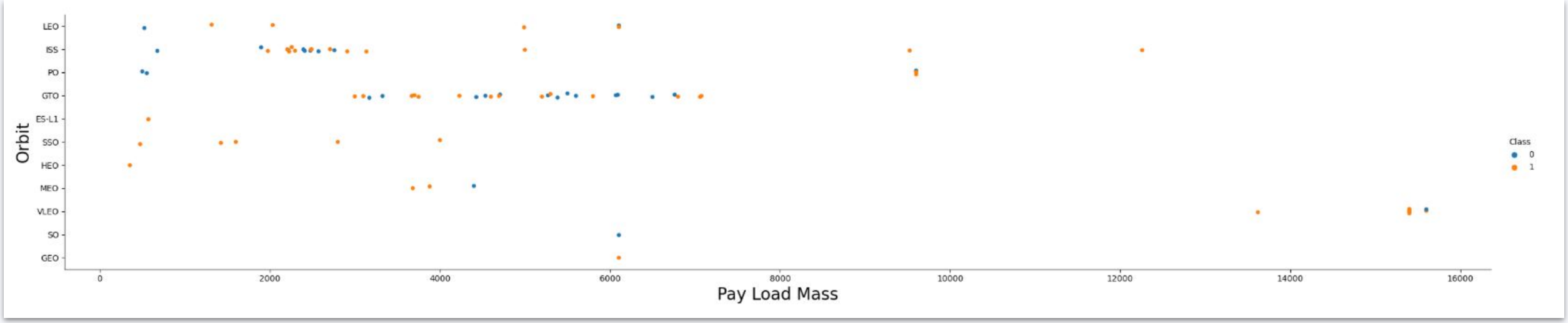
There are some orbit types that only have one occurrence which should be excluded from the dataset.



Payload vs. Orbit Type

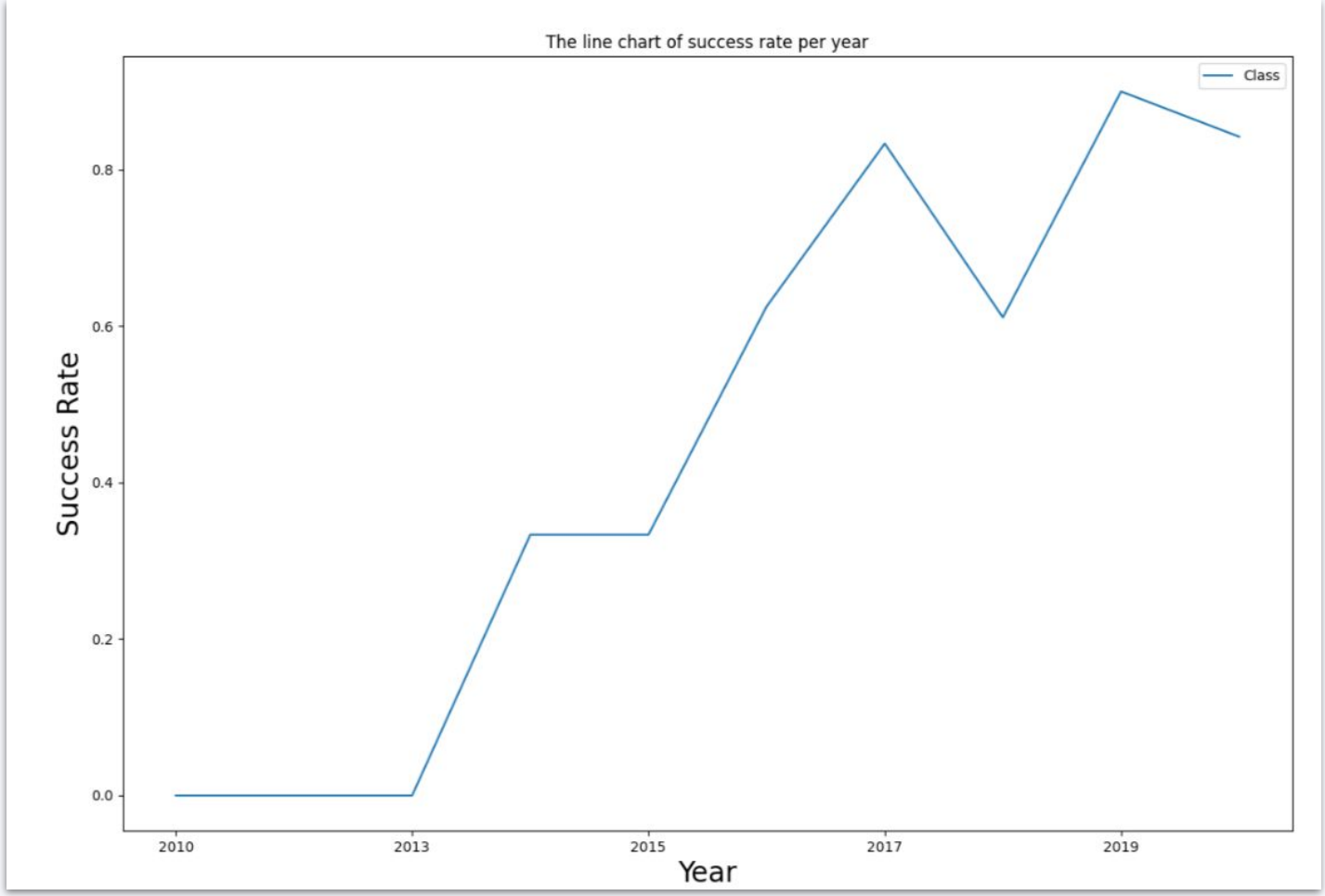
With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS.

However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



Launch Success Yearly Trend

The line chart shows that the success rate kept increasing till 2020.



All Launch Site Names

Use **DISTINCT** to retrieve the unique Launch Site Name.

```
#%sql select distinct(launch_site) from SPACEXTBL;
%sql select distinct(launch_site) from SPACEXTBL;
```

```
 * 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'

Use string pattern **LIKE** in the WHERE clause with LIMIT 5 to retrieve 5 records that have Launch Site Names begin with ‘CCA’

```
%sql select * from SPACEXTBL where launch_site like 'CCA%' limit 5;
```

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing _Outcome
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
08-12-2010	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)
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

- Calculate the total payload carried by boosters from NASA
- Retrieve payload mass using WHERE clause with **SUM()** method.

```
%sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where Customer='NASA (CRS)';
```

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

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

- Calculate the average payload mass carried by booster version F9 v1.1
- Retrieve Payload mass using string pattern **LIKE** and apply **AVG()**.

```
%sql select AVG(PAYLOAD_MASS__KG_) from SPACEXTBL where Booster_Version like '%v1.1%';
```

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

AVG(PAYLOAD_MASS__KG_)
2534.6666666666665

First Successful Ground Landing Date

- Find the dates of the first successful landing outcome on ground pad
- Use **sub query** and apply MIN() function in WHERE clause.

```
%sql select Date from SPACEXTBL where Date = (select min(Date) from SPACEXTBL);
```

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

Date
01-03-2013

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.
- Add two predicates in WHERE clause: “Landing_Outcome” = “Success (drone ship)” and using **BETWEEN..AND…**

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

```
 * 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

- Calculate the total number of successful and failure mission outcomes
- Use **COUNT()** to count the total outcome of success and failure and use GROUP BY per Mission Outcome.

```
%%sql SELECT "Mission_Outcome", COUNT("Mission_Outcome")
FROM SPACEXTBL
GROUP BY "Mission_Outcome";
```

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

Mission_Outcome	COUNT("Mission_Outcome")
None	0
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- List the names of the booster which have carried the maximum payload mass
- Use a subquery with **MAX()** function in WHERE clause to retrieve the boosters with maximum payload mass.

```
%%sql SELECT "Booster_Version"
FROM SPACEXTBL
WHERE "PAYLOAD_MASS__KG_" = (
    SELECT MAX("PAYLOAD_MASS__KG_") FROM SPACEXTBL);
```

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

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

2015 Launch Records

- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Use **substr()** function to extract the year in Date and apply it as a predicate in WHERE clause. Also add another predicate “Landing_Outcome = ‘Failure (drone ship)’.

```
%%sql SELECT substr("Date",4,2) AS "Month",
"Landing_Outcome",
"Booster_Versions",
"Launch_Site"
FROM SPACEXTBL
WHERE substr("Date",7,4) = '2015'
AND "Landing_Outcome" = 'Failure (drone ship)';
```

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

Month	Landing_Outcome	"Booster_Versions"	Launch_Site
10	Failure (drone ship)	Booster_Versions	CCAFS LC-40
04	Failure (drone ship)	Booster_Versions	CCAFS LC-40

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

- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.
- Use COUNT to count the landing_outcome.
- Use GROUP BY to group landing outcome, add a predicate in HAVING clause to retrieve only successful outcome.
- Set another condition using BETWEEN… AND …
- ORDER BY the count of laundding outcome in descending order.

```
%%sql SELECT "Landing_Outcome",
COUNT("Landing_Outcome") AS "COUNT OF LANDING_OUTCOME"
FROM SPACEXTBL
GROUP BY "Landing_Outcome"
HAVING "Landing_Outcome" like 'Success%'
OR "Landing_Outcome" = 'Success'
AND "Date" BETWEEN '04-06-2010' AND '20-03-2017'
ORDER BY "COUNT OF LANDING_OUTCOME" DESC ;
```

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

Landing_Outcome	COUNT OF LANDING_OUTCOME
Success	38
Success (drone ship)	14
Success (ground pad)	9

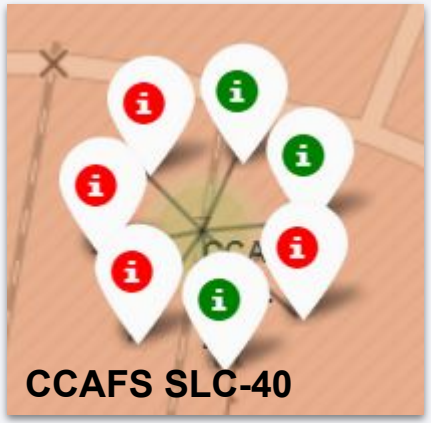

Section 3
Launch Sites
Proximities Analysis

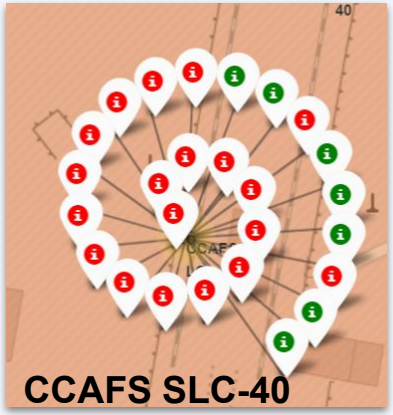
Mark all launch sites on a map

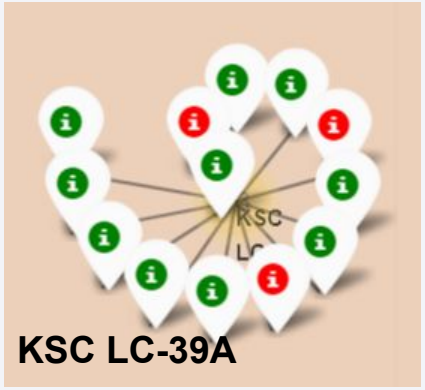
- Add **Circle** to highlight launch sites.
- Add **Marker** to name launch site on the map.
- **Popup** element to show the name when click on the Launchsite on the map.
- Except for **VAFB SLC 4E** which locates on West of the US, all the other launch sites are very close together.

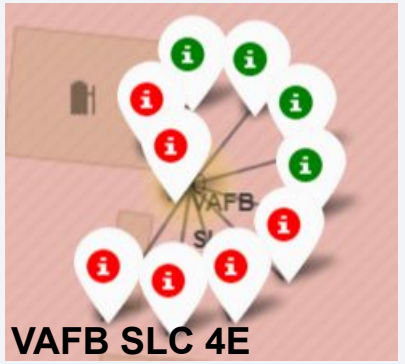
Mark the success/failed launches for each site on the map

- Add a column “marker color” to display success/fail with green/red respectively.
- Add **Marker** with green/red Icon for each launches in launch sites on the map
- From the map we can see the Launch Site CCAFS SLC-40 has the highest success rate because it has more green icons.



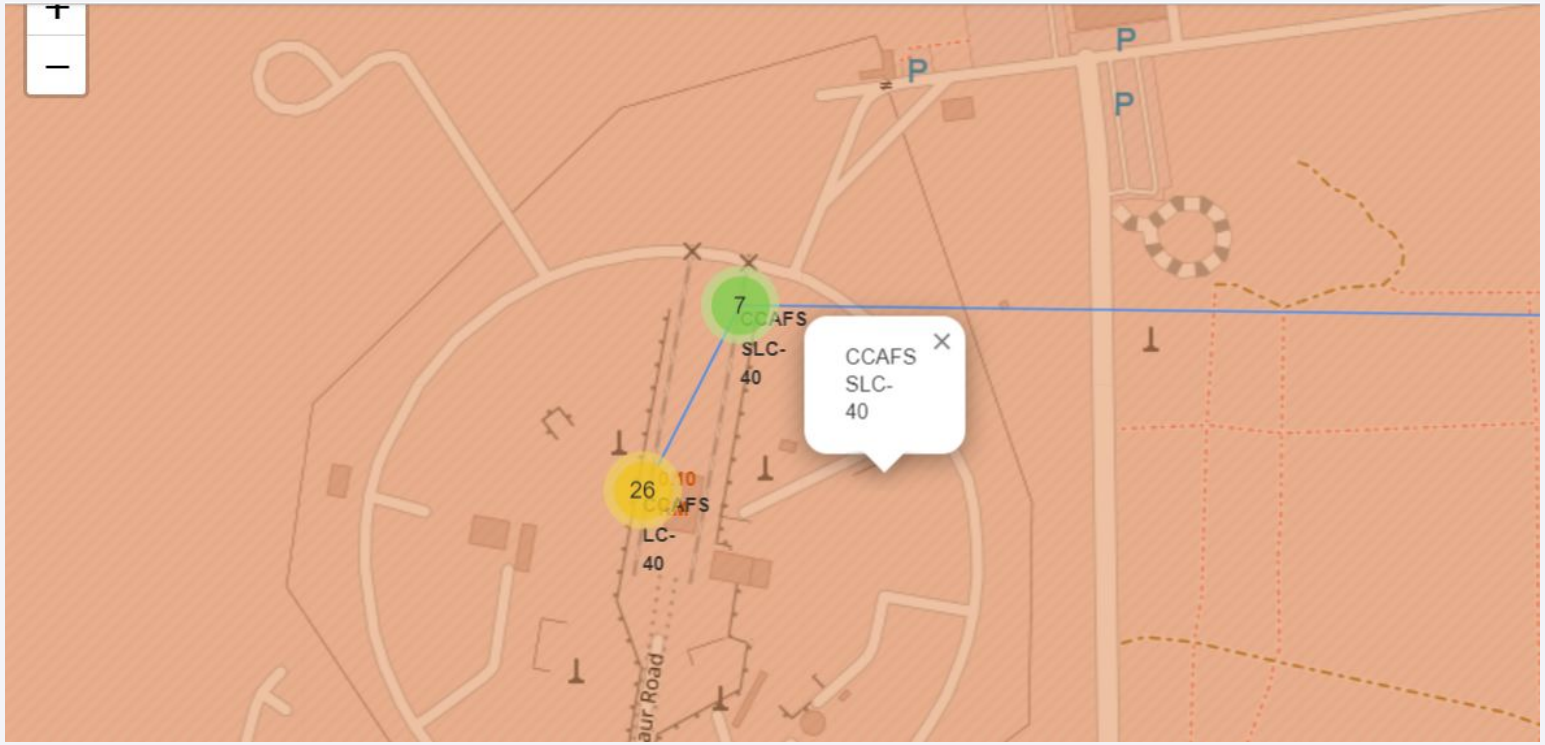


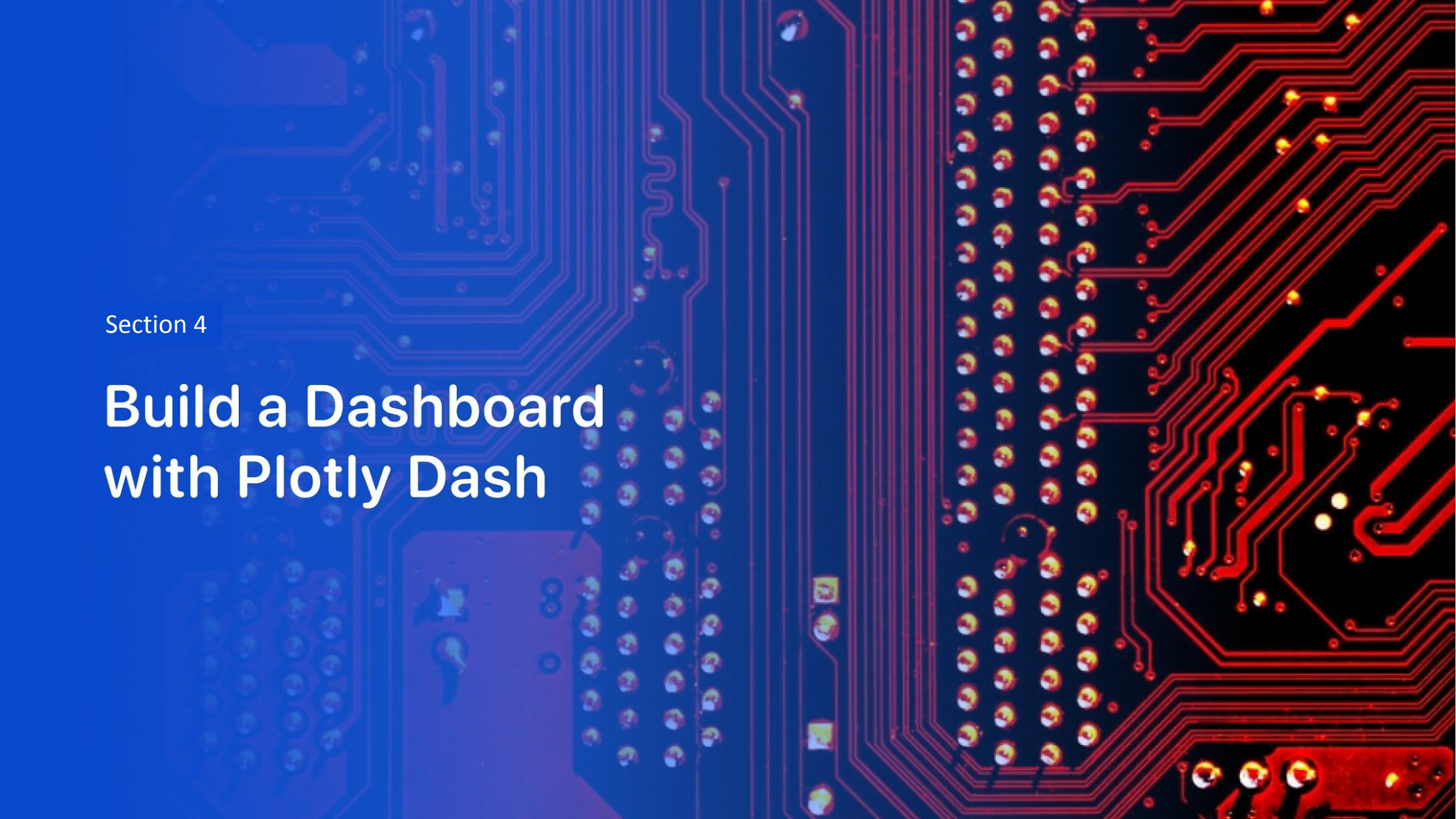




Calculate the distances between a launch site to its proximities

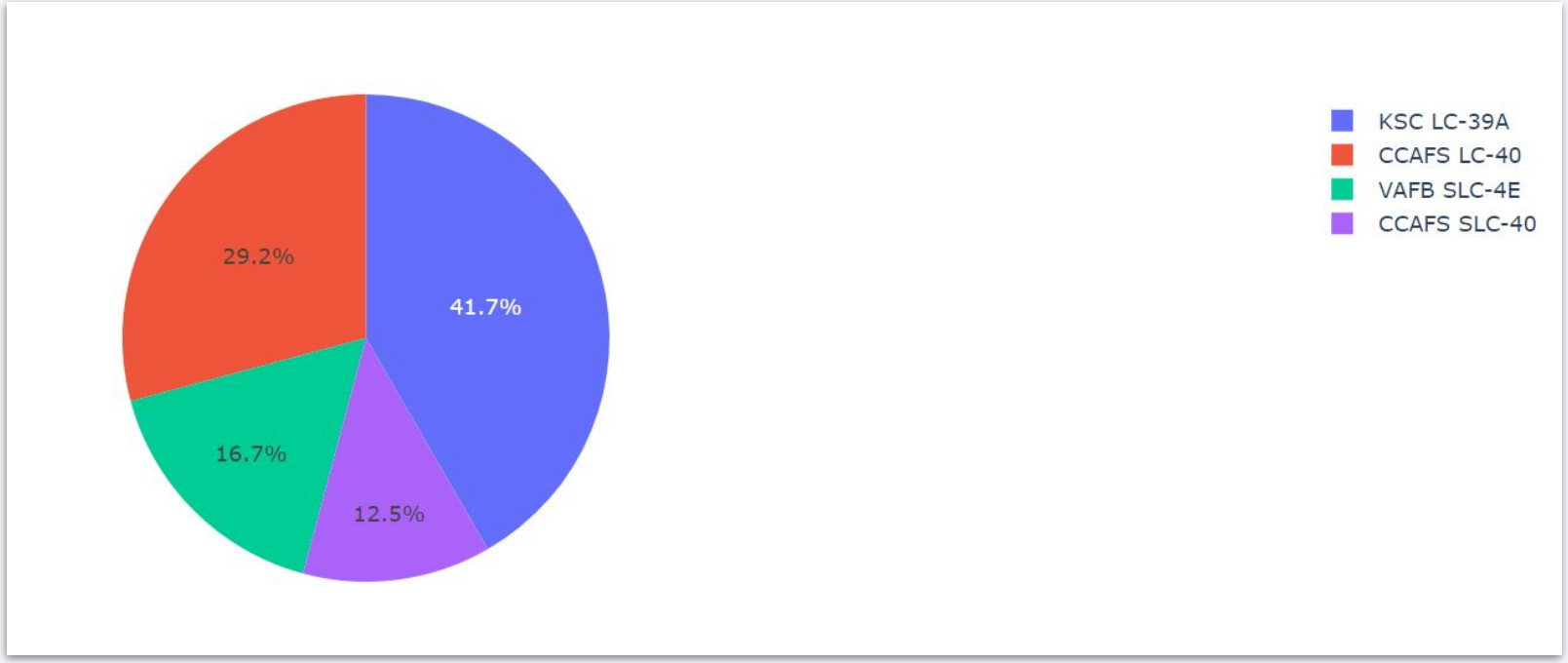
- Add **Marker** to note the distance on the map
- Add a **PolyLine** to draw a line from the launch site to Samuel Strêt




Section 4
Build a Dashboard with Plotly Dash

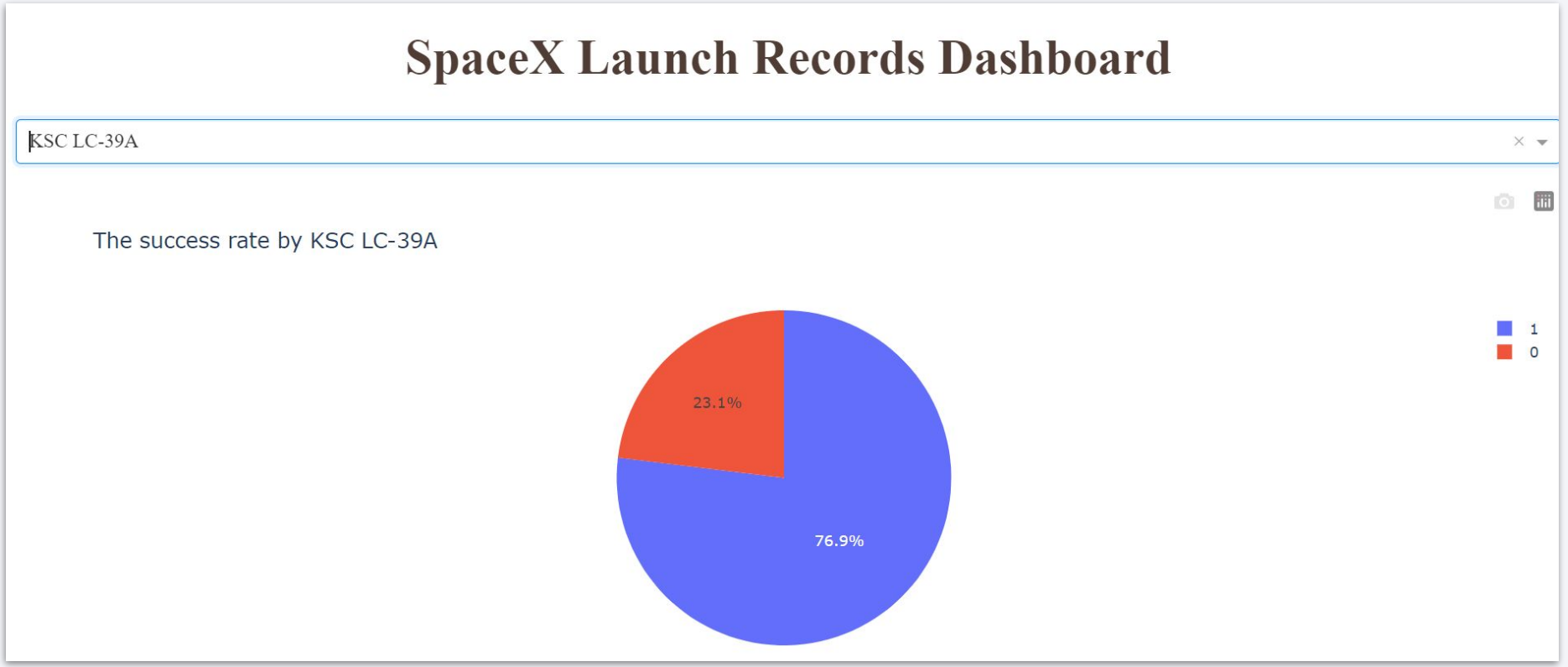
Success Rate by Launch Sites

KSC LC-39A Launch Site has the largest successful launches



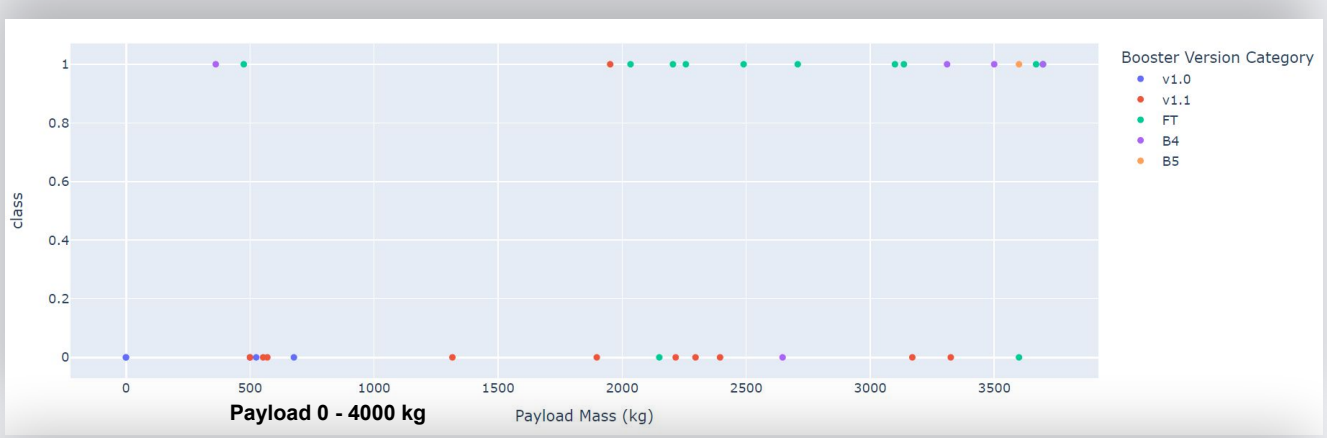
Success ratio by KSC LC-39A

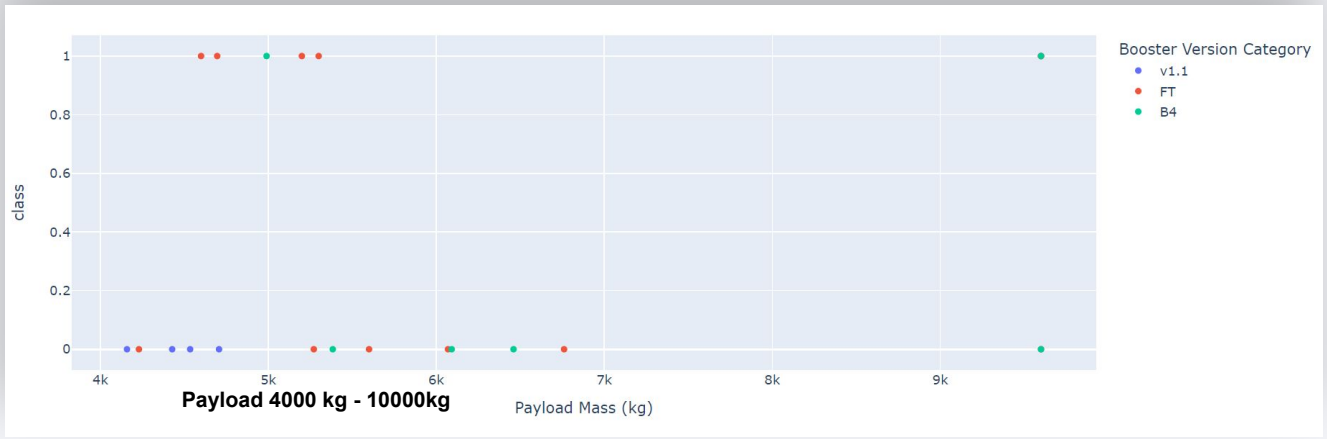
KSC LC-39A Launch Site has the largest success ratio



Payload vs Launch Outcome scatter plot

We can see the success rate of low weighted payload is higher than the high weighted payload



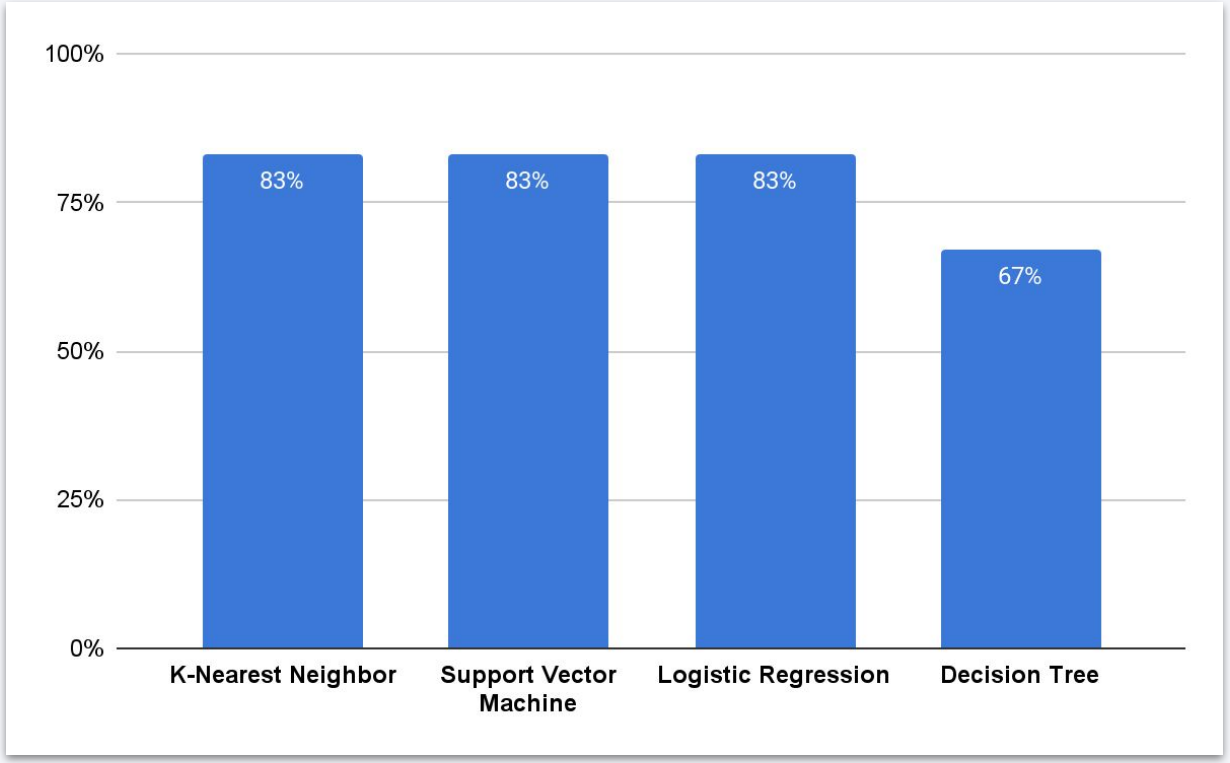


Section 5

Predictive Analysis (Classification)

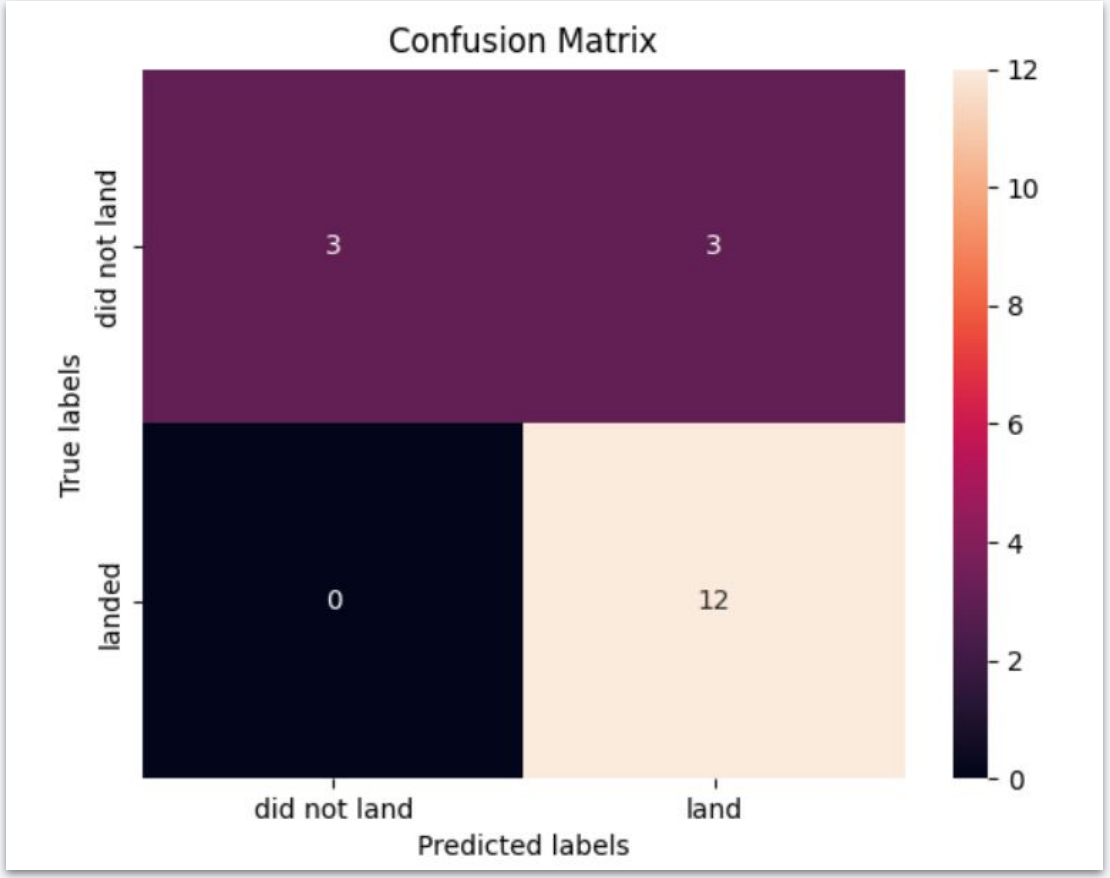
Classification Accuracy

- K-Nearest Neighbor, SVM and Logistic Regression have the highest accuracy compared to Decision Tree Model.



Confusion Matrix

- The confusion matrix of the best three models can distinguish the true positive. The main problem is the false positive.



Conclusions

We can conclude that:

- We can use K-nearest neighbor, Support vector machine and logistics regression to predict the landing outcome.
- The low weighted payloads (under 4000 kg) have the better performance than the high weighted payloads
- Starting from the year 2013, the success rate for SpaceX launches is increased, directly proportional time in years to 2020, which it will eventually perfect the launches in the future.
- KSC LC-39A have the most successful launches of any sites; 76.9%.
- SSO orbit have the most success rate; 100% and more than 1 occurrence.

Thank you!