Winning Space Race with Data Science

Neetu Dabas
2 September, 2022

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion

Executive Summary

- Summary of methodologies
  - Data collection through API
  - Data collection through Webscraping
  - Data Wrangling
  - Exploratory Data Analysis with SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium
  - Machine Learning Analysis for Prediction
- Summary of all results
  - Exploratory Data Analysis result
  - Interactive Analytics in screenshots
  - Predictive Analysis results from Machine Learning Lab

Introduction

Project background and context:

SapceX is a revolutionary company who has disrupt the space industry by offering a rocket launch specifically falcon 9 as low as 62 million dollar; while other providers cost upward of 165 million dollars each. Most of this saving is due to the idea of reusing the first stage of the launch by re-land the rocket to be used on the next mission. Repeating this process will make the price down further. As a data scientist of a startup SapceY rivaling SapceX, the goal of this project is to create a machine learning pipeline to predict the landing outcomes of the first stage in the future. This project is crucial in identifying the right price to bid against to SapceX for a rocket launch.

- Problems we want to find answers

- Identifying all factors that influence the landing outcomes

- Relatonship between each variables and how it is affecting the outcome

- Best condition needed to increase the probability of the successful landing

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collecting using SapceX REST API and Webscraping from Wikipedia
- Perform data wrangling
  - Data was processed by 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
  - How to build, tune, evaluate classification models

Data Collection

- Data collection is the process of gathering and measuring information on targeted variables in an established system, which then enables one to answer relevent questions and evaluate outcomes. As mentioned the dataset was coolected by REST API and Webscraping from Wikipedia.
- For REST API, it started by using get request. Then, we decoded the reponse content as JSON and turn it into a pandas dataframe using json_normalize(). We then clean the data, check for missing values and fill with whatever needed.
- For Webscraping, we will use the BeautifulSoup to extract the launch records as HTML table, parse the table and convert it to the dataframe for further analysis.

Data Collection – SpaceX API

Get request for rocket launch data using API

Use json_normalize() method to convert Json results in dataframe

Performed data cleaning and filling missing values

```
spacex_url="https://api.spacexdata.com/v4/launches/past"

response = requests.get(spacex_url)
```

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

```
# Lets take a subset of our dataframe keeping only the features we want a
nd 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 rocket
s 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 s
ingle 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 ex
tracting 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)]
```

https://github.com/Neetu-v/SpaceX_project/blob/1b5db698e460e5b3633804a7e984ae576b3545a5/Data%20collection%20through%20API.ipynb

Data Collection - Scraping

Request the Falcon9 launch Wiki page from url

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

Create a BeautifulSoup from the html response

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

Extarct all column names from html header

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('table',"wikitable plai
nrowheaders collapsible")):
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as number corresponding t
o launch a number
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
```

https://github.com/Neetu-v/SpaceX_project/blob/1b5db698e460e5b3633804a7e984ae576b3545a5/Data%20collection%20using%20web%20scraping%20(1).ipynb

Data Wrangling

Data Wrangling is the process of cleaning and unifying messy and complex data sets for easy access and Exploratory Data Analysis (EDA).

- We first calculate the number of launches on each site

```
# 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
```

- Secondly calculate the number and occurrence of each orbit

```
# 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
```


https://github.com/Neetu-v/SpaceX_project/blob/1b5db698e460e5b3633804a7e984ae576b3545a5/Data%20Wrangling.ipynb

Data Wrangling.....

- Then we calculate the number and occurence of mission outcome per orbit type

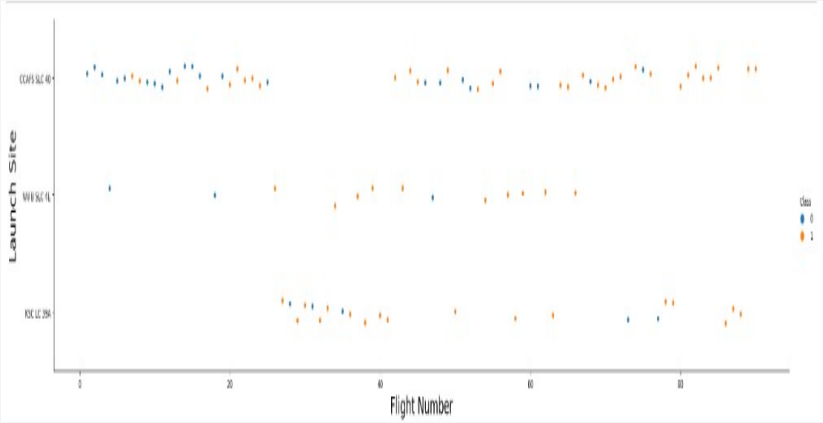
```
# landing_outcomes = values on Outcome column
landing_outcomes = df.Outcome.value_counts()
landing_outcomes
```

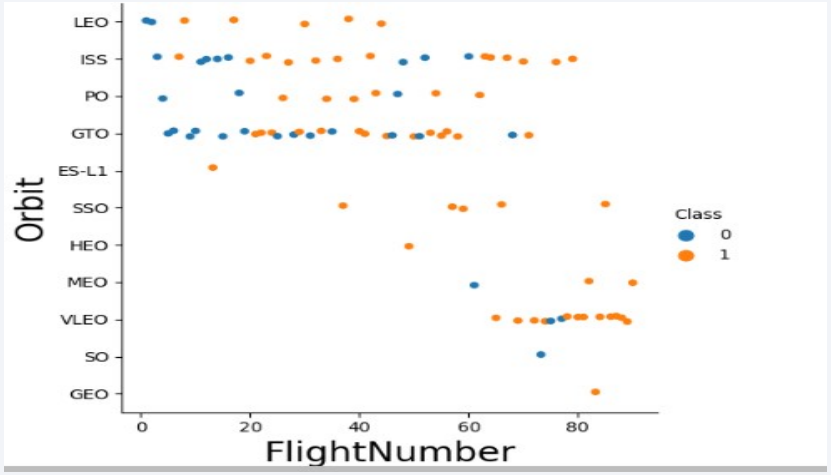
```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
```

- At last we create a landing outcome label from Outcome column

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for i in df['Outcome']:
    if i in set(bad_outcomes):
        landing_class.append(0)
    else:
        landing_class.append(1)
```

EDA with Data Visualization





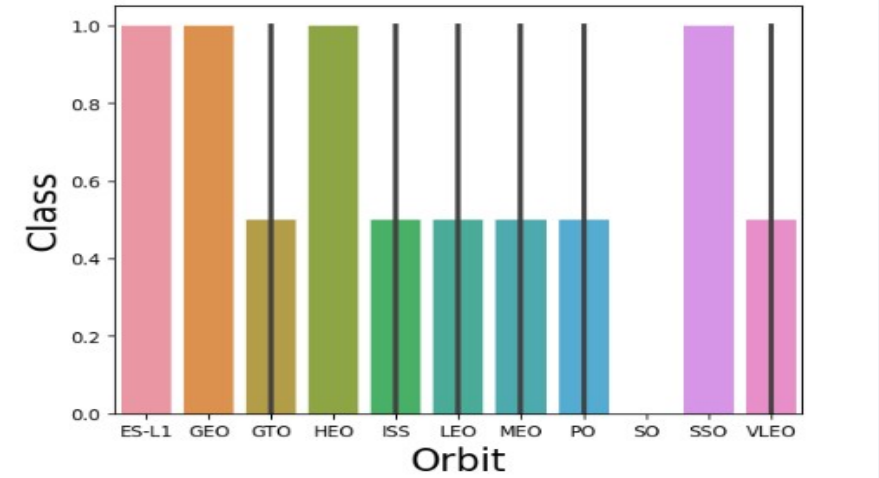
We first started by using scatter graph to find the relationship between the attributes such as between:

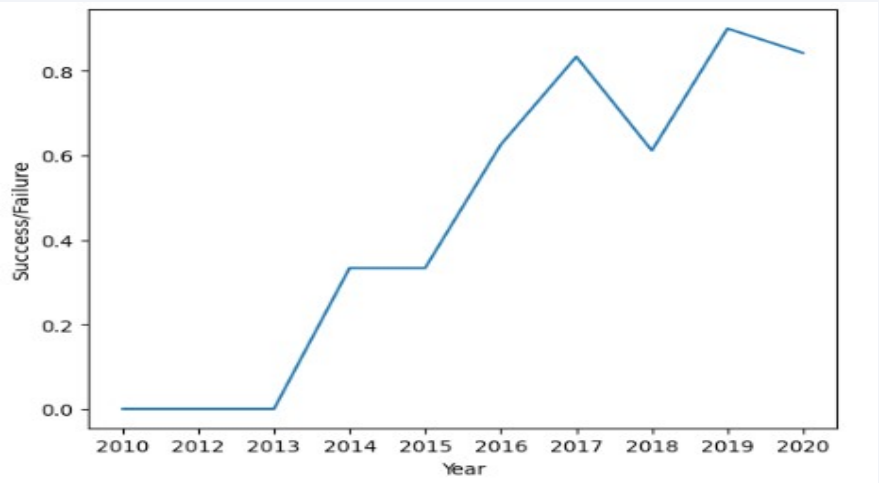
- Payload and Flight Number.
- Flight Number and Launch Site.
- Payload and Launch Site.
- Flight Number and Orbit Type.
- Payload and Orbit Type.

Scatter plots show dependency of attributes on each other. Once a pattern is determined from the graphs. It's very easy to see which factors affecting the most to the success of the landing outcomes.

https://github.com/Neetu-v/SpaceX_project/blob/1b5db698e460e5b3633804a7e984ae576b3545a5/EDA%20with%20Visualization%20(2).ipynb

EDA with Data Visualization





Once we get a hint of the relationships using scatter plot. We will then use further visualization tools such as bar graph and line plots graph for further analysis.

Bar graphs is one of the easiest way to interpret the relationship between the attributes. In this case, we will use the bar graph to determine which orbits have the highest probability of success.

We then use the line graph to show a trends or pattern of the attribute over time which in this case, is used for see the launch success yearly trend.

We then use Feature Engineering to be used in success prediction in the future module by created the dummy variables to categorical columns.

EDA with SQL

Using SQL, we had performed many queries to get better understanding of the dataset.

- Displaying the names of the launch sites.
- Displaying 5 records where launch sites begin with the string 'CCA'.
- Displaying the total payload mass carried by booster launched by NASA (CRS).
- Displaying the average payload mass carried by booster version F9 v1.1.
- Listing the date when the first successful landing outcome in ground pad was achieved.
- Listing the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000.
- Listing the total number of successful and failure mission outcomes.
- Listing the names of the booster_versions which have carried the maximum payload mass.
- Listing the failed landing_outcomes in drone ship, their booster versions, and launch sites names for in year 2015.
- Rank the count of landing outcomes or success between the date 2010-06-04 and 2017-03-20, in descending order.

https://github.com/Neetu-v/SpaceX_project/blob/1b5db698e460e5b3633804a7e984ae576b3545a5/EDA%20with%20SQL%20(1).ipynb

Build an Interactive Map with Folium

To visualize the launch data into an interactive map. We took the latitude and longitude coordinates at each launch site and added a circle marker around each launch site with a label of the name of the launch site.

We then assigned the dataframe launch_outcomes(failure,success) to classes 0 and 1 with Red and Green markers on the map in MarkerCluster().

We then used the Haversine's formula to calculated the distance of the launch sites to various landmark to find answer to the questions of:

- How close the launch sites with railways, highways and coastlines?
- How close the launch sites with nearby cities?

https://github.com/Neetu-v/SpaceX_project/blob/1b5db698e460e5b3633804a7e984ae576b3545a5/lab_jupyter_launch_site_location.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash which allowing the user to play around with the data as they need.
- We plotted pie charts showing the total launches by a certain sites.
- We then plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

https://github.com/Neetu-v/SpaceX_project/blob/1b5db698e460e5b3633804a7e984ae576b3545a5/spacex%20launch%20record%20dashboard.pdf

Predictive Analysis (Classification)

Building the Model

- Load the dataset into NumPy and Pandas
- Transform the data and then split into training and test datasets
- Decide which type of ML to use
- set the parameters and algorithms to GridSearchCV and fit it to dataset.

Evaluating the Model

- Check the accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- plot the confusion matrix

Improving the Model

- Use Feature Engineering and Algorithm Tuning

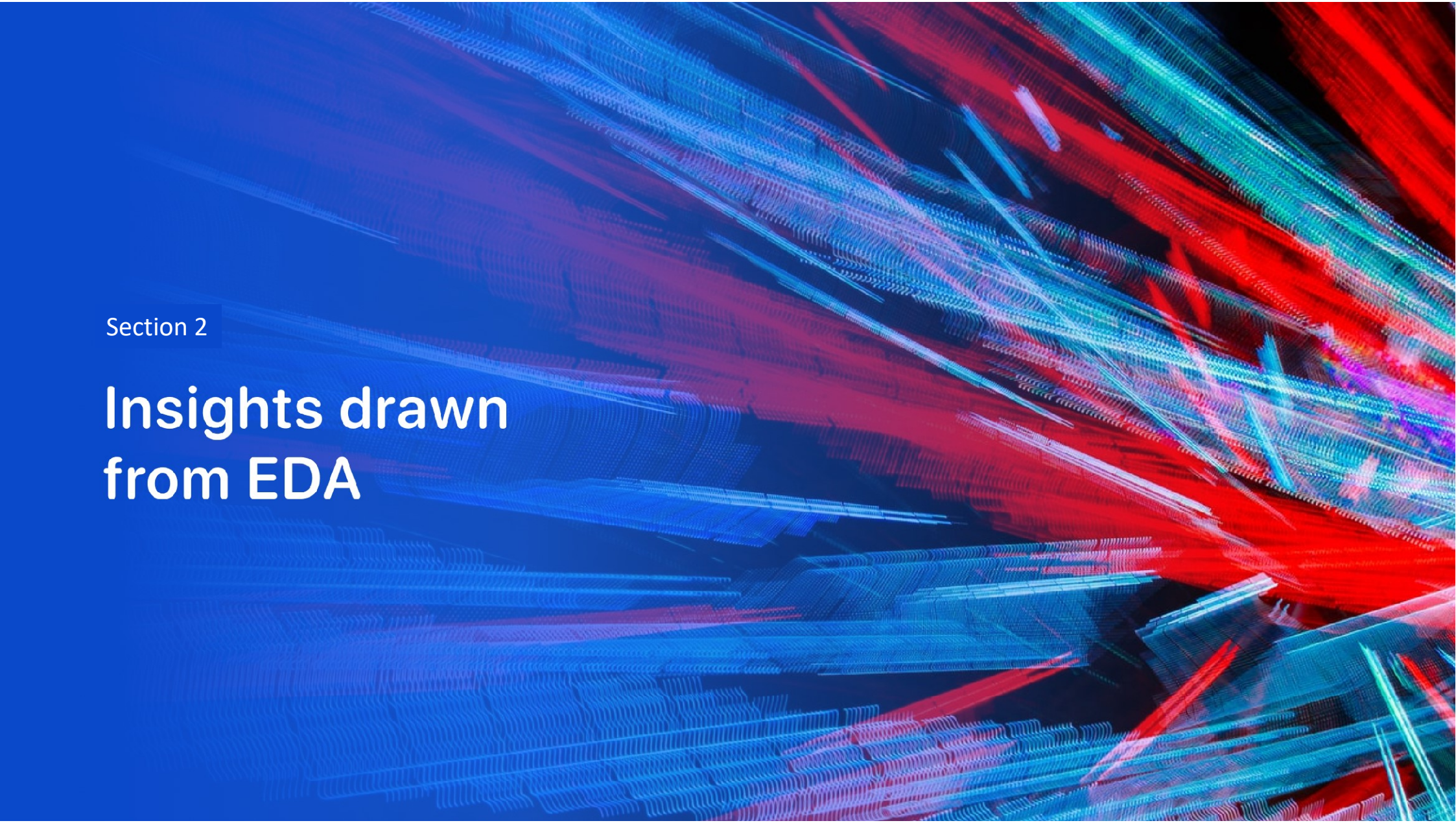
Find the Best Model

- The model with the best accuracy score will be the best performing model.

https://github.com/Neetu-v/SpaceX_project/blob/89895d922d9bf071a31e219087a91eafb08e187a/SpaceX_Machine%20Learning%20Prediction.ipynb

Results

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



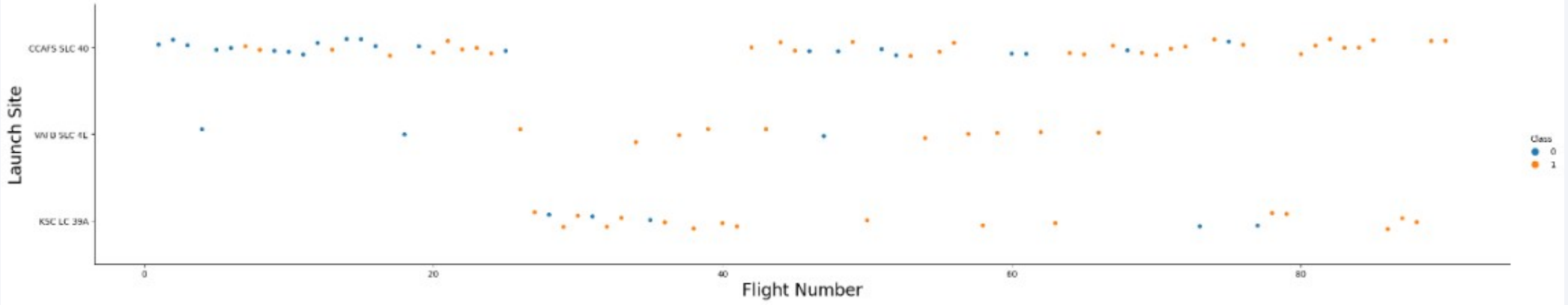
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

This scatter plot shows that the larger the flights amount of the launch site, the greater the the success rate will be.

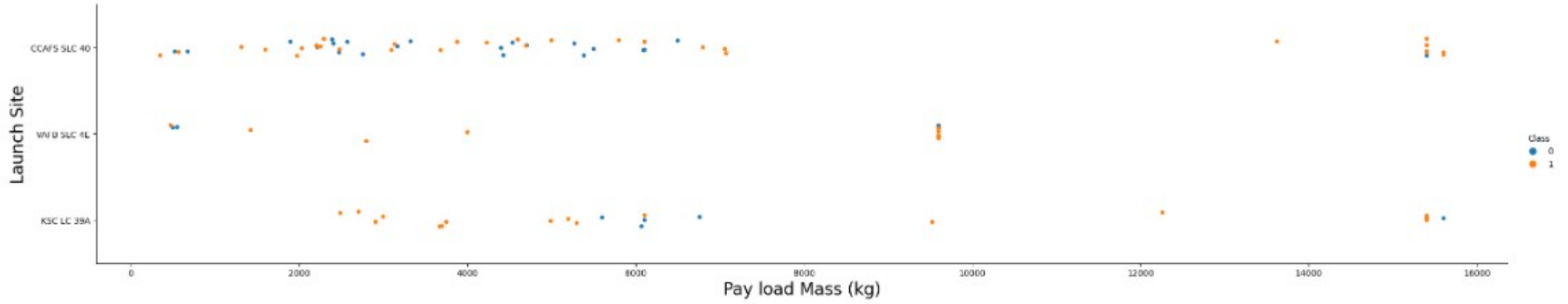
However, site CCAFS-SLC40 shows the least pattern of this.



Payload vs. Launch Site

This scatter plot shows once the pay load mass is greater than 7000kg, the probability of the success rate will be highly increased.

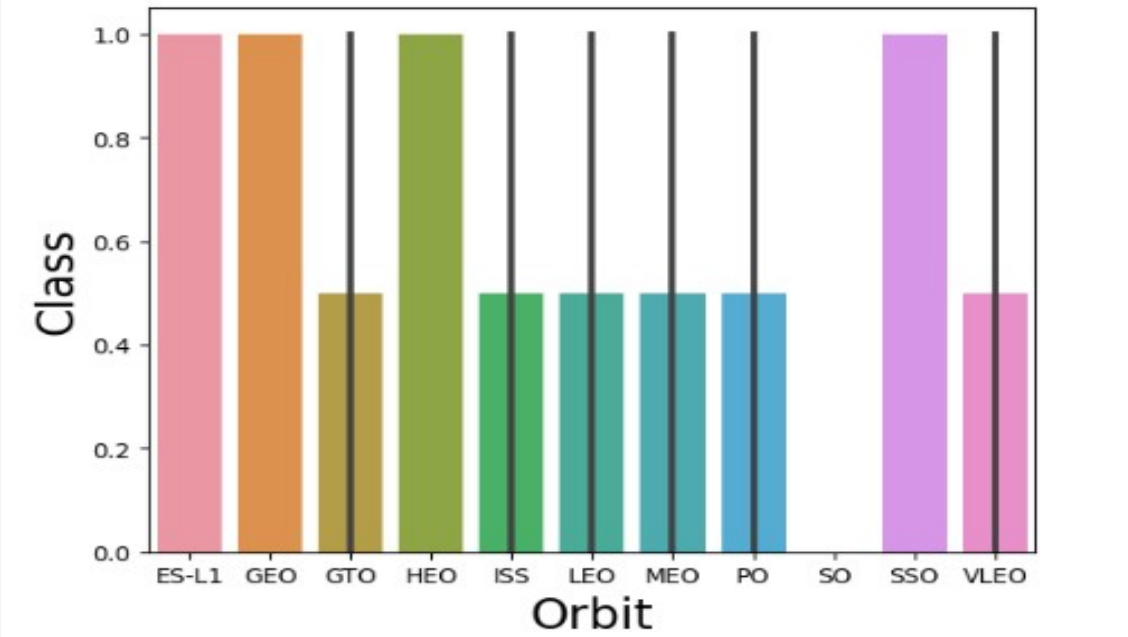
However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.



Success Rate vs. Orbit Type

This figure depicted the possibility of the orbits to influences the landing outcomes as some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.

However, deeper analysis show that some of this orbits has only 1 occurrence such as GEO, SO, HEO and ES-L1 which mean this data need more dataset to see pattern or trend before we draw any conclusion.

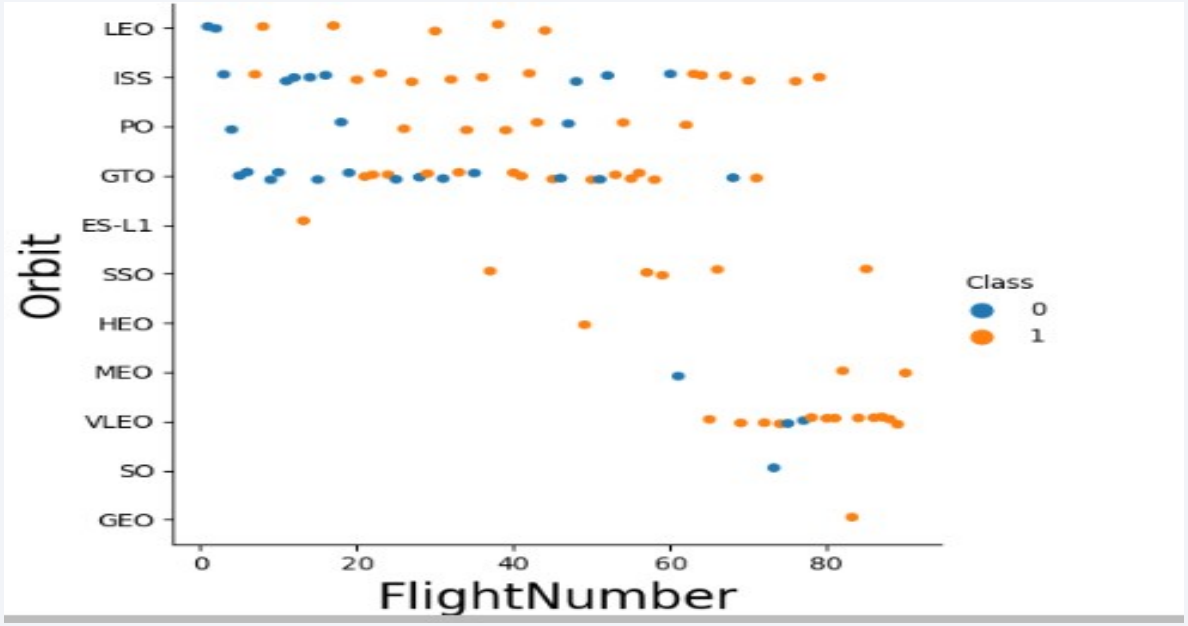


Flight Number vs. Orbit Type

This scatter plot shows that generally, the larger the flight number on each orbits, the greater the success rate (especially LEO orbit) except for GTO orbit which depicts no relationship between both attributes.

Orbit that only has 1 occurrence

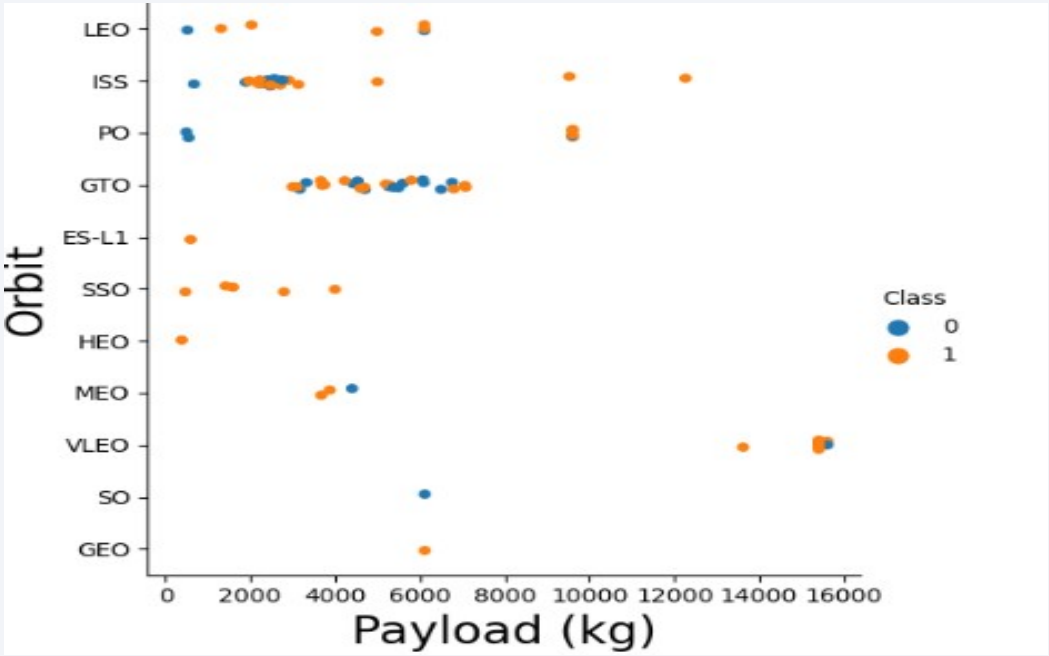
should also be excluded from above statement as it’s needed more dataset.



Payload vs. Orbit Type

Heavier payload has positive impact on LEO, ISS and PO orbit.

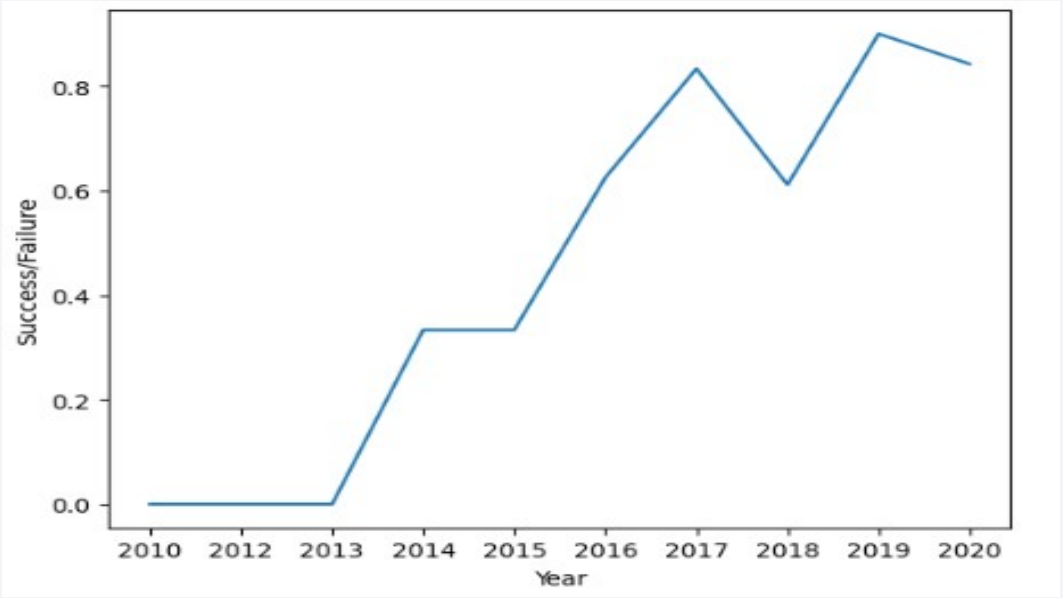
However, it has negative impact on MEO and VLEO orbit. GTO orbit seem to depict no relation between the attributes. Meanwhile, again, SO, GEO and HEO orbit need more dataset to see any pattern or trend.



Launch Success Yearly Trend

This figures clearly depicted and increasing trend from the year 2013 until 2020.

If this trend continue for the next year onward. The success rate will steadily increase until reaching 1/100% success rate.



All Launch Site Names

We used the key word DISTINCT to show only unique launch sites from the SpaceX data.

```
In [5]: %sql select distinct (LAUNCH_SITE) from SPACEXTBL
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Out[5]:

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

Launch Site Names Begin with 'CCA'

We used the query following to display 5 records where launch sites begin with `CCA`

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

```
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2012-05-22	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

We calculated the total payload carried by boosters from NASA as 45596 using the query below

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

```
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

1
45596

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 2928

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'
```

```
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

1
2928

First Successful Ground Landing Date

We use the min() function to find the result.

We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015.

```
%sql select min(DATE) from SPACEXTBL where Landing__Outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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.

```
%sql select BOOSTER_VERSION from SPACEXTBL where Landing__Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
```

```
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
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

Here we use count clause on mission outcome and where clause to find the mission outcome equal to success of equal to failure.

```
%sql select count(MISSION_OUTCOME) from SPACEXTBL where MISSION_OUTCOME = 'Success' or MISSION_OUTCOME = 'Failure (in flight)'
```

```
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

1
100

Boosters Carried Maximum Payload

We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

```
%sql select BOOSTER_VERSION from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

```
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
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
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7

2015 Launch Records

we use Extract clause to find 2015 launch records.

```
%sql SELECT MONTH(DATE),MISSION_OUTCOME,BOOSTER_VERSION,LAUNCH_SITE FROM SPACEXTBL where EXTRACT(YEAR FROM DATE)='2015'
```

```
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

1	mission_outcome	booster_version	launch_site
1	Success	F9 v1.1 B1012	CCAFS LC-40
2	Success	F9 v1.1 B1013	CCAFS LC-40
3	Success	F9 v1.1 B1014	CCAFS LC-40
4	Success	F9 v1.1 B1015	CCAFS LC-40
4	Success	F9 v1.1 B1016	CCAFS LC-40
6	Failure (in flight)	F9 v1.1 B1018	CCAFS LC-40
12	Success	F9 FT B1019	CCAFS LC-40

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

We selected Landing outcomes and the COUNT of landing outcomes from the data and used the WHERE clause to filter for landing outcomes BETWEEN 2010-06-04 to 2010-03-20.We applied the GROUP BYclause to group the landing outcomes.

```
%sql SELECT LANDING__OUTCOME, COUNT(*) AS COUNT_LAUNCHES FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP B
Y LANDING__OUTCOME
```

```
 * ibm_db_sa://vwh94811:***@fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

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

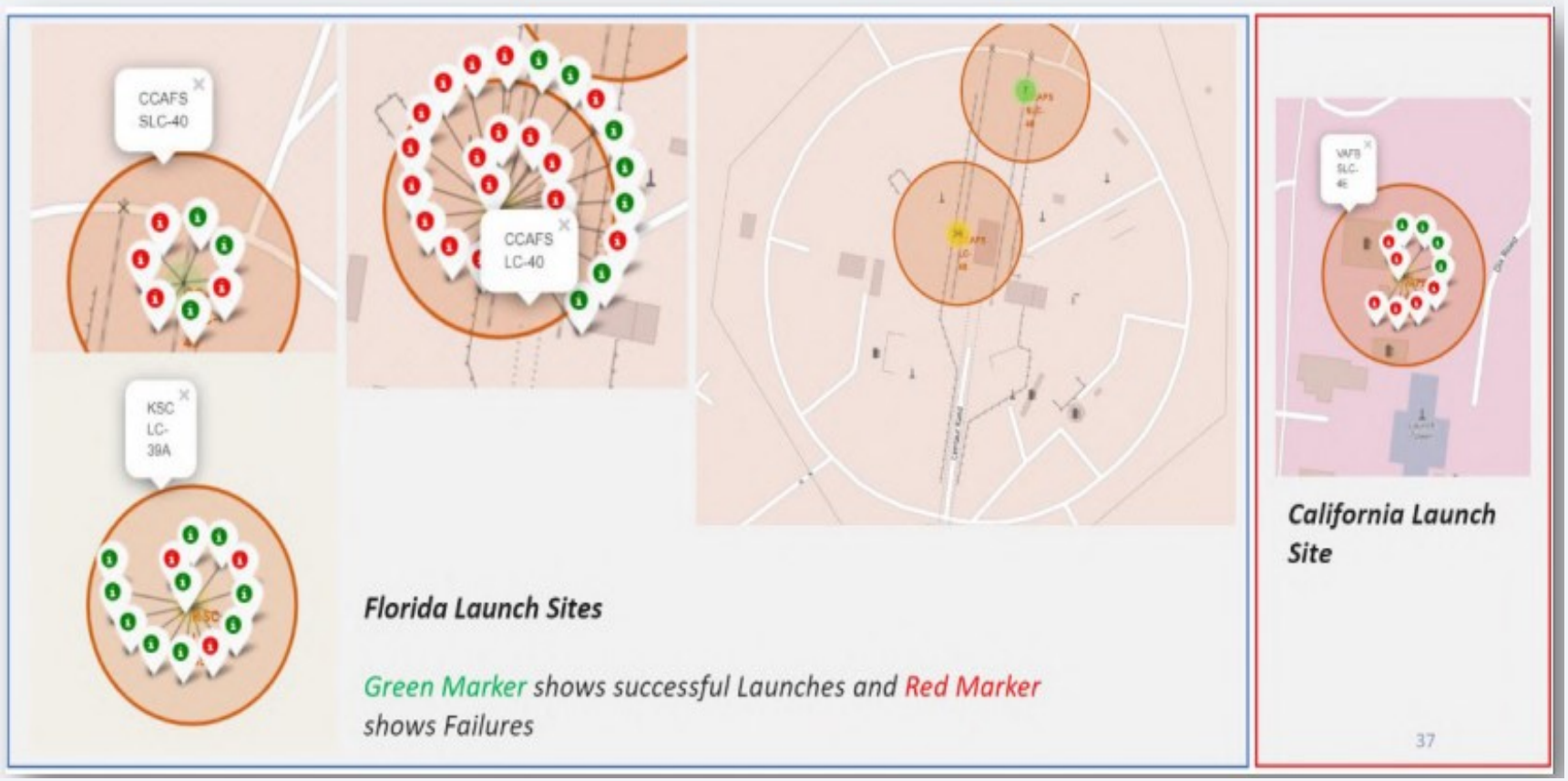
Section 3

Launch Sites Proximities Analysis

Location of all the Launch Sites

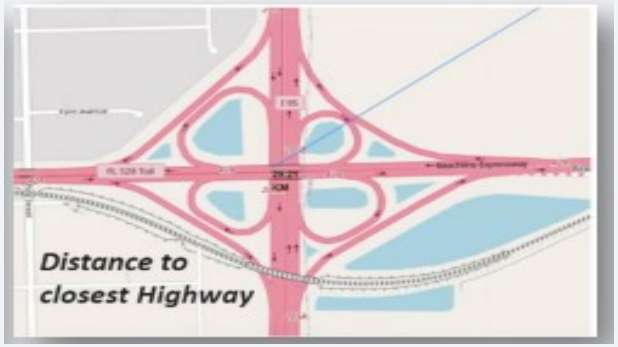
- We can see that all the SpaceX launch sites are located inside the United States

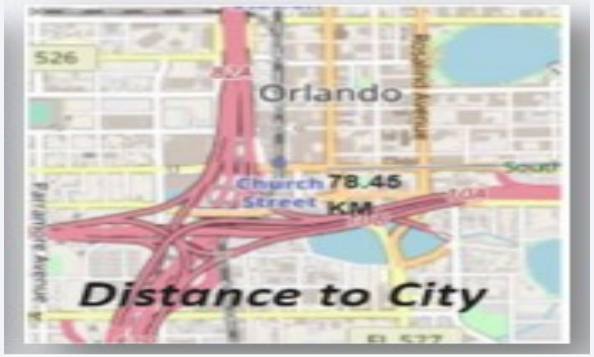
Markers showing launch sites with color labels

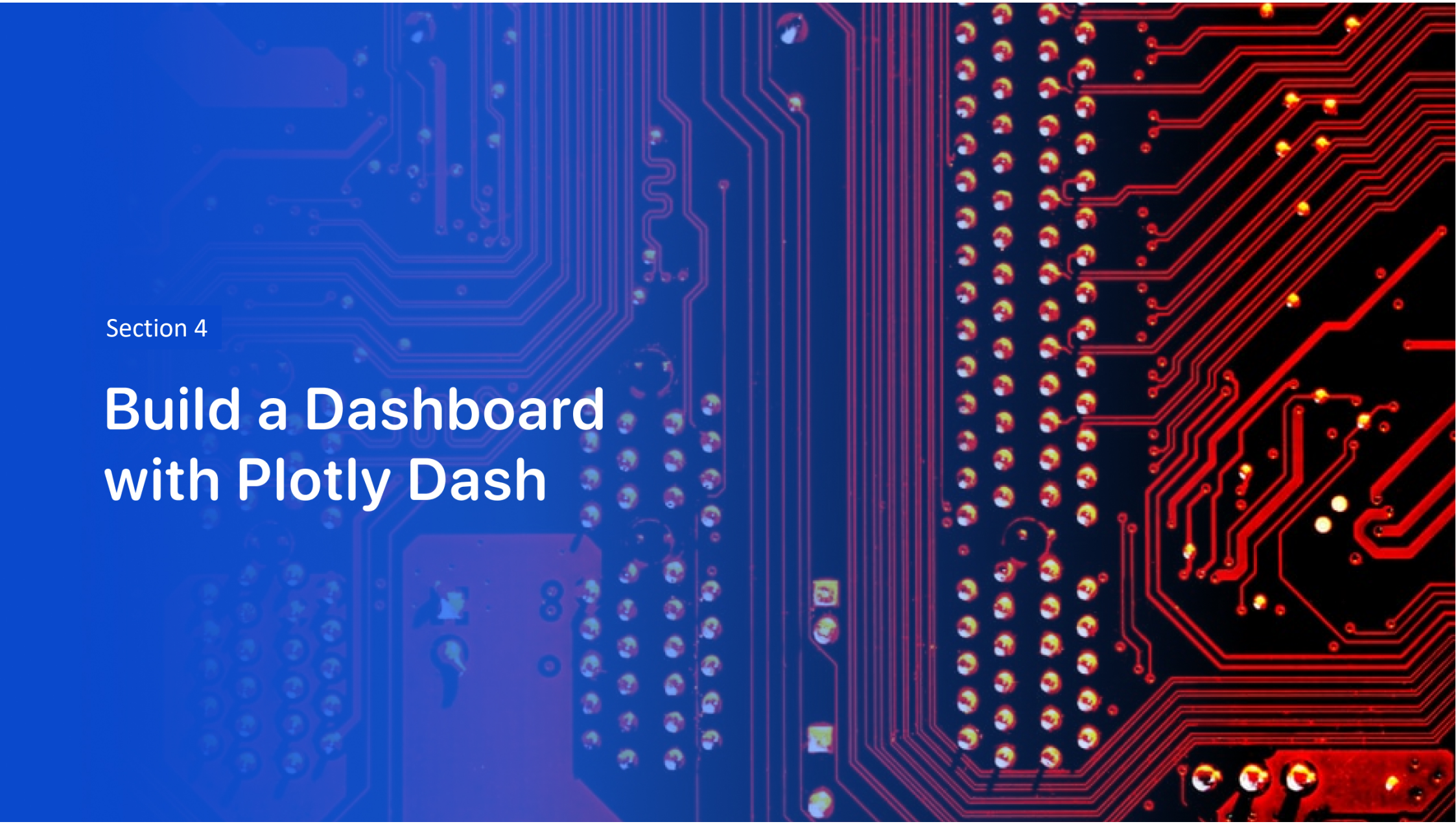


Launch Sites Distance to Landmarks

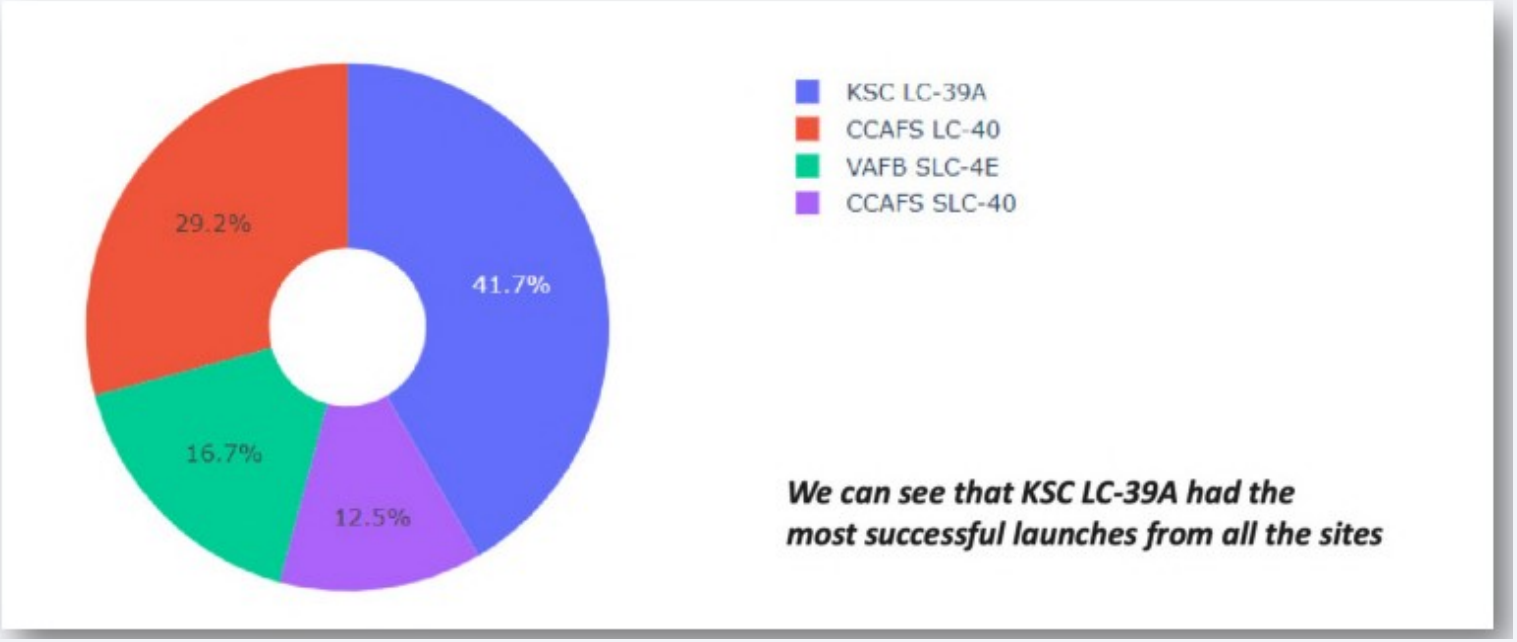
- Are launch sites in close proximity to railway station? No
- Are launch sites in close proximity to highways?No
- Are launch sites in close proximity to coastline?Yes
- Do launch sites keep certain distance away from cities?Yes



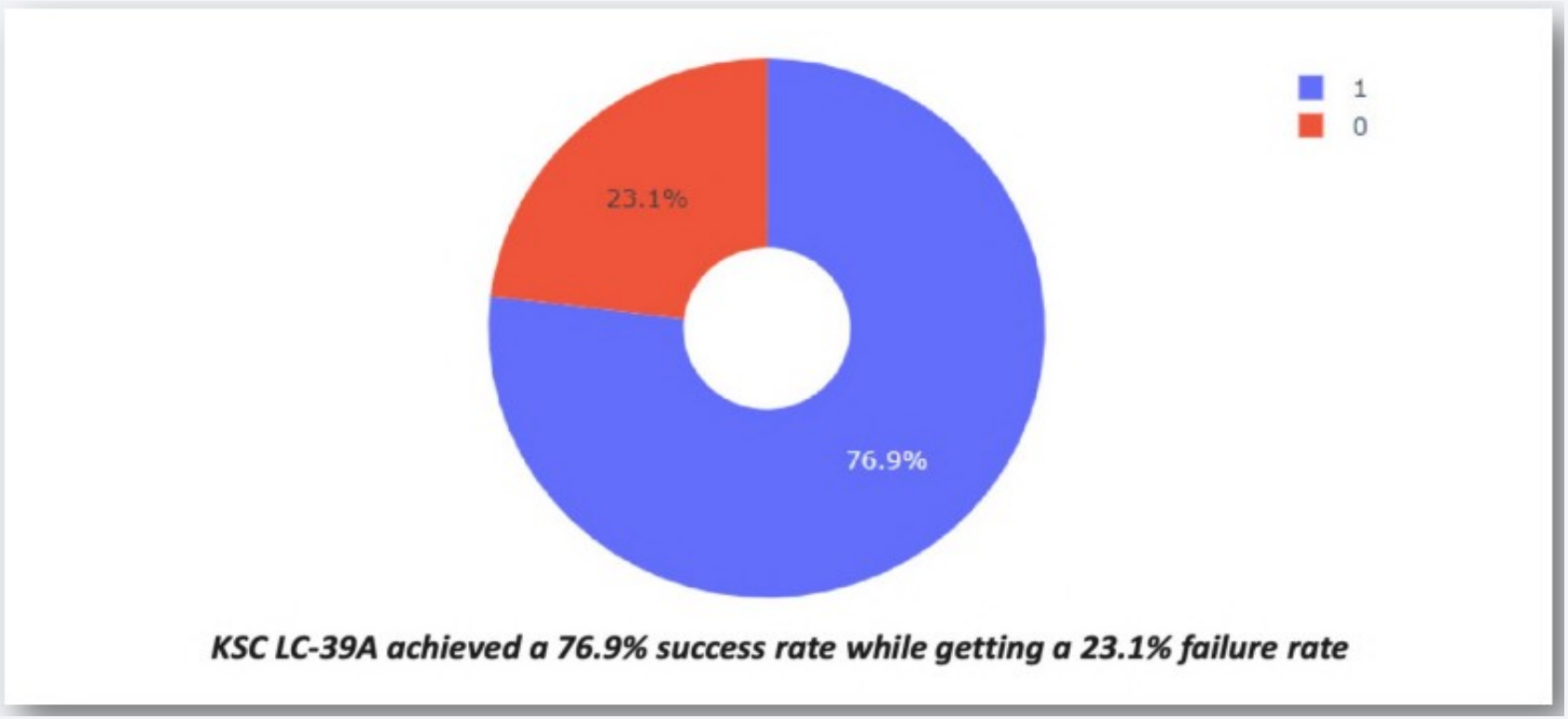



Section 4
Build a Dashboard with Plotly Dash

The success percentage by each sites.



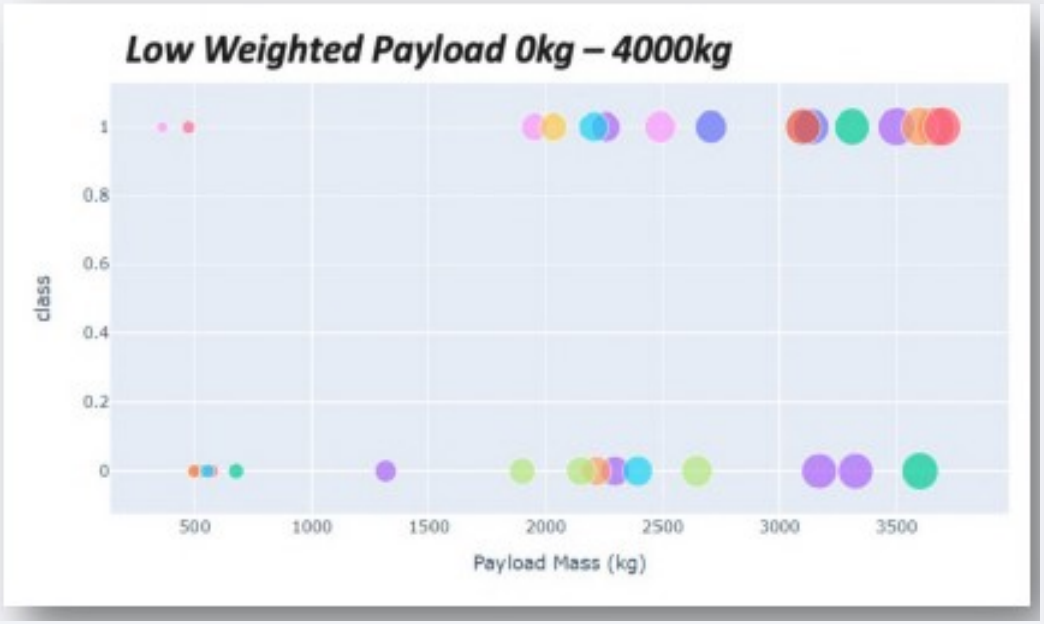
The highest launch-success ratio: KSC LC-39A

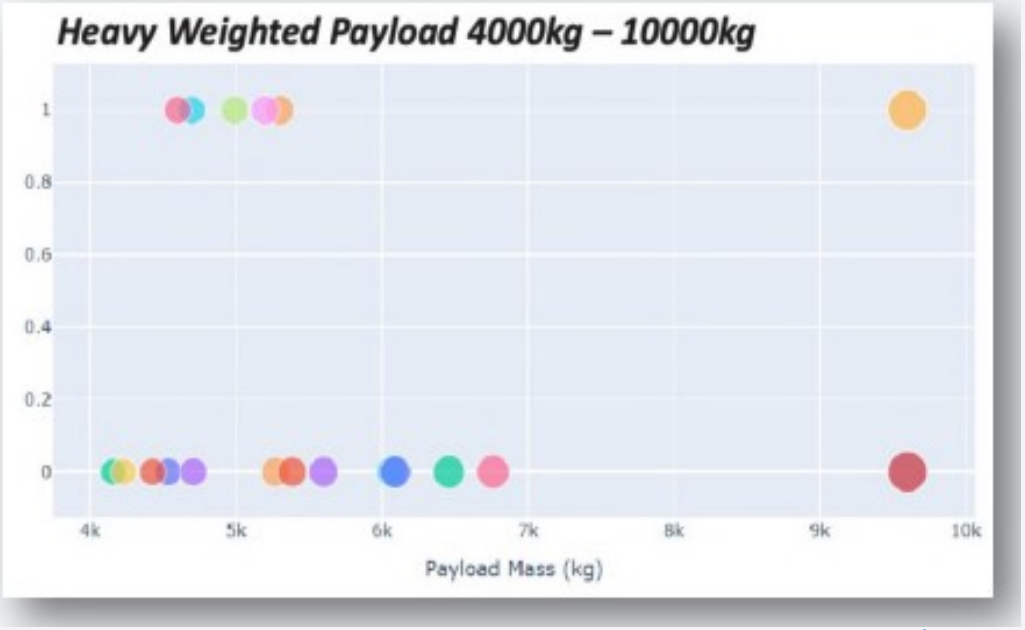


KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

Payload vs Launch Outcome Scatter Plot

We can see that all the success rate for low weighted payload is higher than heavy weighted payload





Section 5

Predictive Analysis (Classification)

Classification Accuracy

As we can see, by using the code as below: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

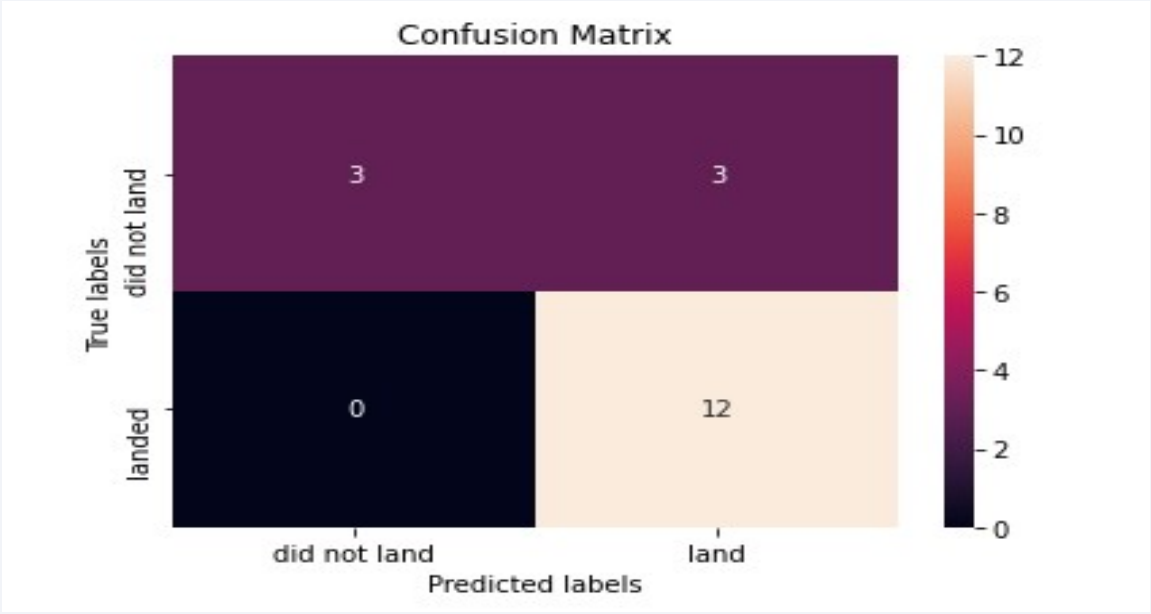
In [37]:

```
algorithms = {'KNN':knn_cv.best_score_,'Tree':tree_cv.best_score_,'LogisticRegression':logreg_cv.best_score_}
bestalgorithm = max(algorithms, key=algorithms.get)
print('Best Algorithm is', bestalgorithm,'with a score of',algorithms[bestalgorithm])
```

```
Best Algorithm is Tree with a score of 0.875
```

Confusion Matrix

The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes. The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.



Conclusions

We can conclude that:

- The Tree Classifier Algorithm is the best Machine Learning approach for this dataset.
- The low weighted payloads (which define as 4000kg and below) performed better than the heavy 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!