

Winning Space Race with Data Science

Bandhavi Bendi
11-13-2022



Outline

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

Executive Summary

- Data collection
- Data wrangling
- Exploring data with analysis with data visualization
- Exploring Data analysis with SQL
- Building maps with Folium
- Building an interactive dashboard with Plotly Dash
- Predictive Analysis(Classification)

Summary of results

- Exploratory Data Analysis result
- Interactive analysis
- Predictive analysis result from Machine Learning Lab

Introduction

SpaceX is a revolutionary company who has disrupted the space industry by offering a rocket launch specifically Falcon 9 as low as 62 million dollars; while other providers cost upward of 165 million dollars each. Most of this saving thanks to SpaceX's astounding idea to reuse the first stage of the launch by re-landing the rocket to be used on the next mission. Repeating this process will make the price even further. As a data scientist of a startup rivaling SpaceX, the goal of this project is to create the machine learning pipeline to predict the landing outcome of the first stage in the future. This project is crucial in identifying the right price to bid against SpaceX for a rocket launch.

The problems included:

- Identifying all factors that influence the landing outcome.
- The relationship between each variables and how it is affecting the outcome.
- The best condition needed to increase the probability of successful landing.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
 - Data was collected using SpaceX REST API 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
 - 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 relevant questions and evaluate outcomes. As mentioned, the dataset was collected by REST API and Web Scrapping from Wikipedia

For REST API, its started by using the get request. Then, we decoded the response content as Json and turn it into a pandas dataframe using `json_normalize()`. We then cleaned the data, checked for missing values and fill with whatever needed.

For web Scrapping, we will use the BeautifulSoup to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for further analysis

Data Collection – SpaceX API

```
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)]
```

Get request for Rocket
launch data using API

Use json_normalize
method to convert json
result to dataframe

Performed data cleaning
and filling the missing
value

Data Collection - Scraping

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

```
data = requests.get(static_url).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')
```

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

```
# Use soup.title attribute  
soup.title
```

```
# 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('tr')
```

Starting from the third table is our target table contains the actual launch records.

```
# Let's print the third table and check its content  
first_launch_table = html_tables[2]
```

[https://github.com/bbandhu/Applied-Data-Science-Capstone/blob/master/jupyter-labs-web-scraping-2%20\(1\).ipynb](https://github.com/bbandhu/Applied-Data-Science-Capstone/blob/master/jupyter-labs-web-scraping-2%20(1).ipynb)

Request the Falcon9 Launch Wiki page from its URL

Create a BeautifulSoup object from the HTML response

Extract all column/variable names from the HTML table header

Create a data frame by parsing the launch HTML tables

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 will first calculate the number of launches on each site, then calculate the number and occurrence of mission outcome per orbit type.

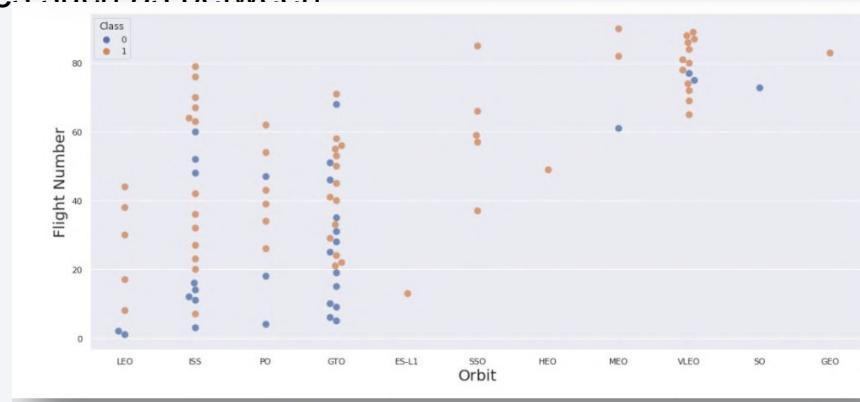
We then create a landing outcome label from the outcome column. This will make it easier for further analysis, visualization, and ML. Lastly, we will export the result to a CSV

<https://github.com/bbandhu/Applied-Data-Science-Capstone/blob/master/labs-jupyter-spacex-Data%20wrangling.ipynb>

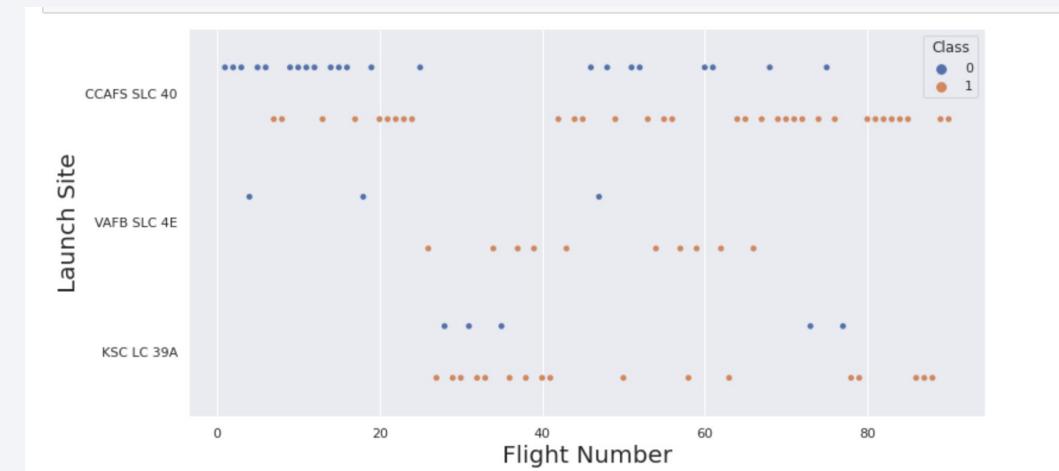
EDA with Data Visualization

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.



[GithubURL](#)

EDA with SQL

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

- 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/bbandhu/Applied-Data-Science-Capstone/blob/master/notebook:Exploratory_Data_Analysis_with_SQL_eqznon1EA.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/bbandhu/Applied-Data-Science-Capstone/blob/master/notebook:Interactive_Visual_Analytics_with_Folium_M8uUhCmHY.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/bbandhu/Applied-Data-Science-Capstone/blob/master/spacex_dash_app.py

Predictive Analysis (Classification)

Building the Model

1. Load the dataset into NumPy and Pandas
2. Transform the data and then split into training and test datasets
3. Decide which type of ML to use
4. 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

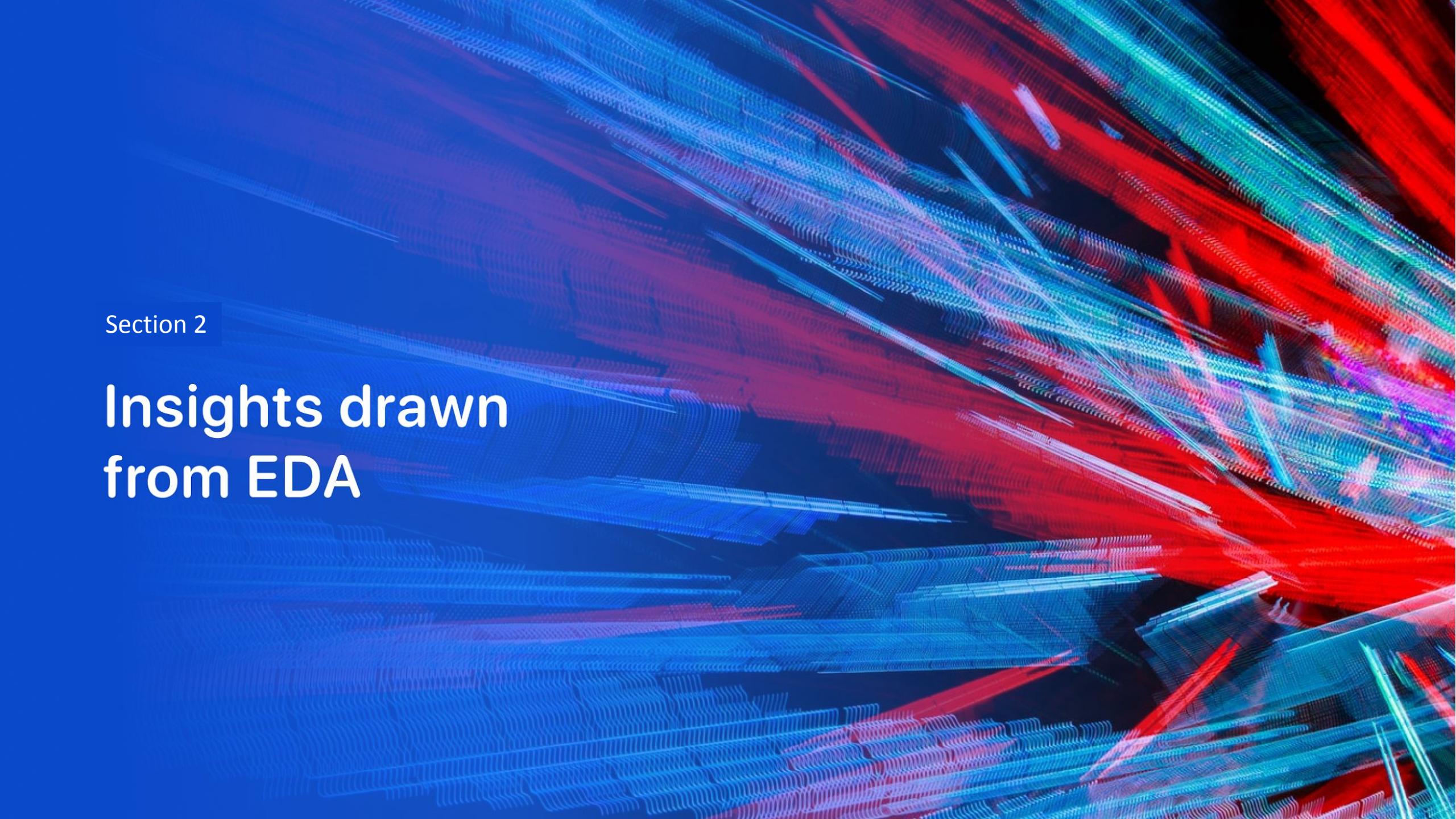
Find the Best Model

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

Results

The results will be categorized to 3 main results which is:

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

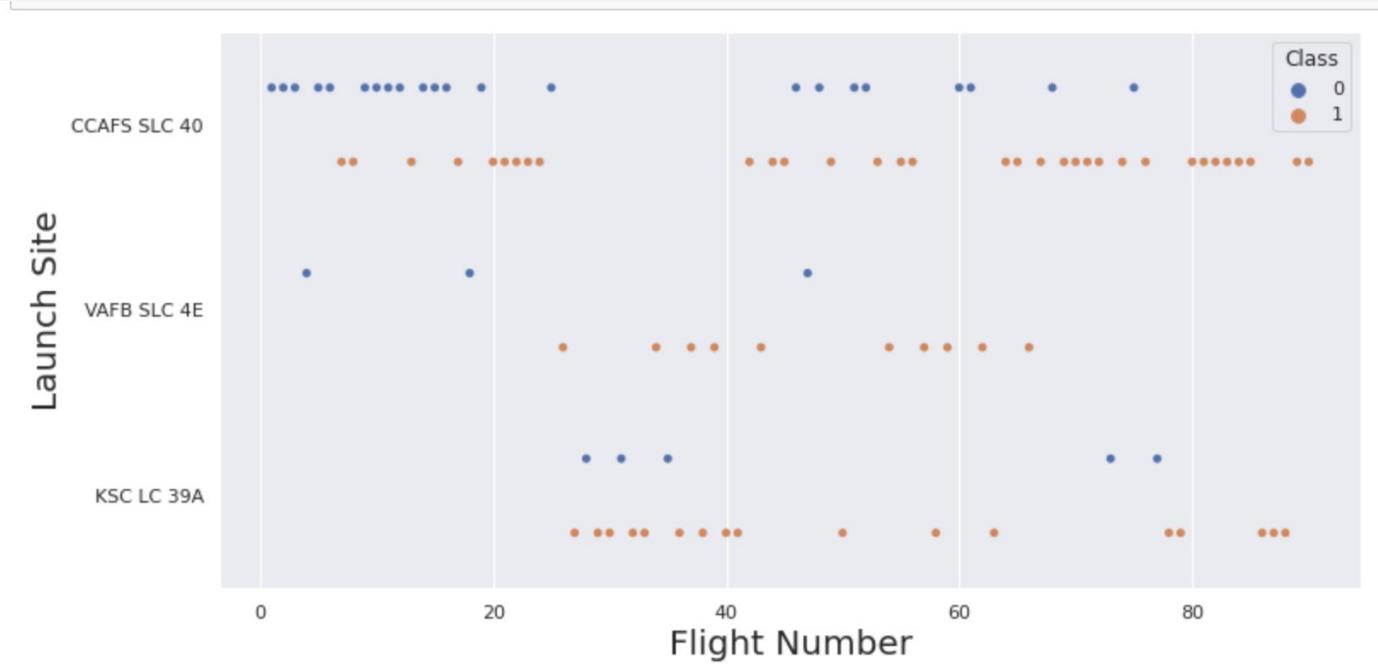
The background of the slide features a complex, abstract pattern of glowing lines. These lines are primarily blue and red, creating a sense of depth and motion. They appear to be composed of numerous small, glowing particles or dots, giving them a textured, almost liquid-like appearance. The lines converge and diverge, forming various shapes and directions across the dark, solid-colored background.

Section 2

Insights drawn from EDA

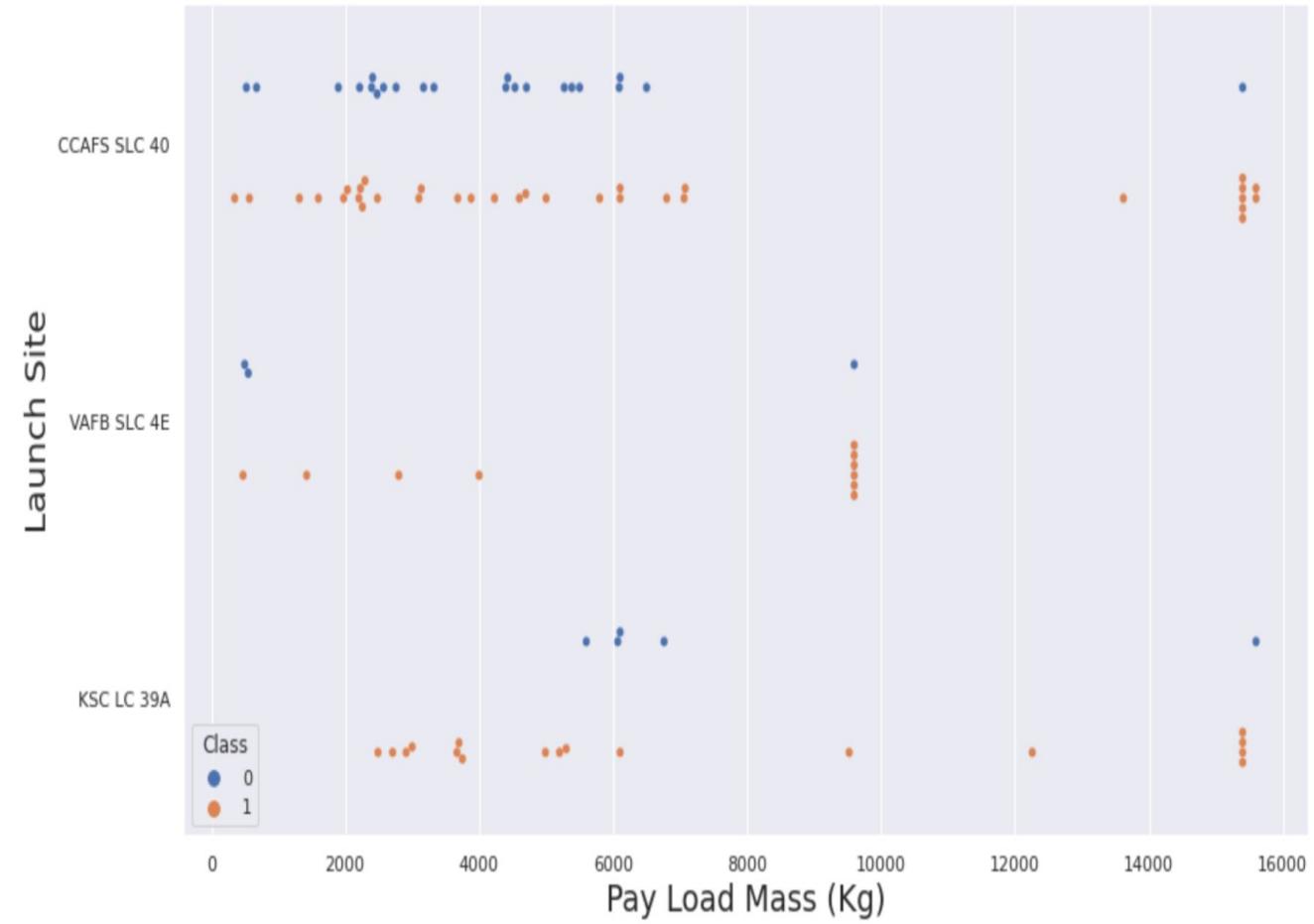
Flight Number vs. Launch Site

This scatter plot shows that the larger the flights amount of the launch site, the greater the success rate will be. However, site CCAFS SLC40 shows the least pattern of this.



Payload vs. Launch Site

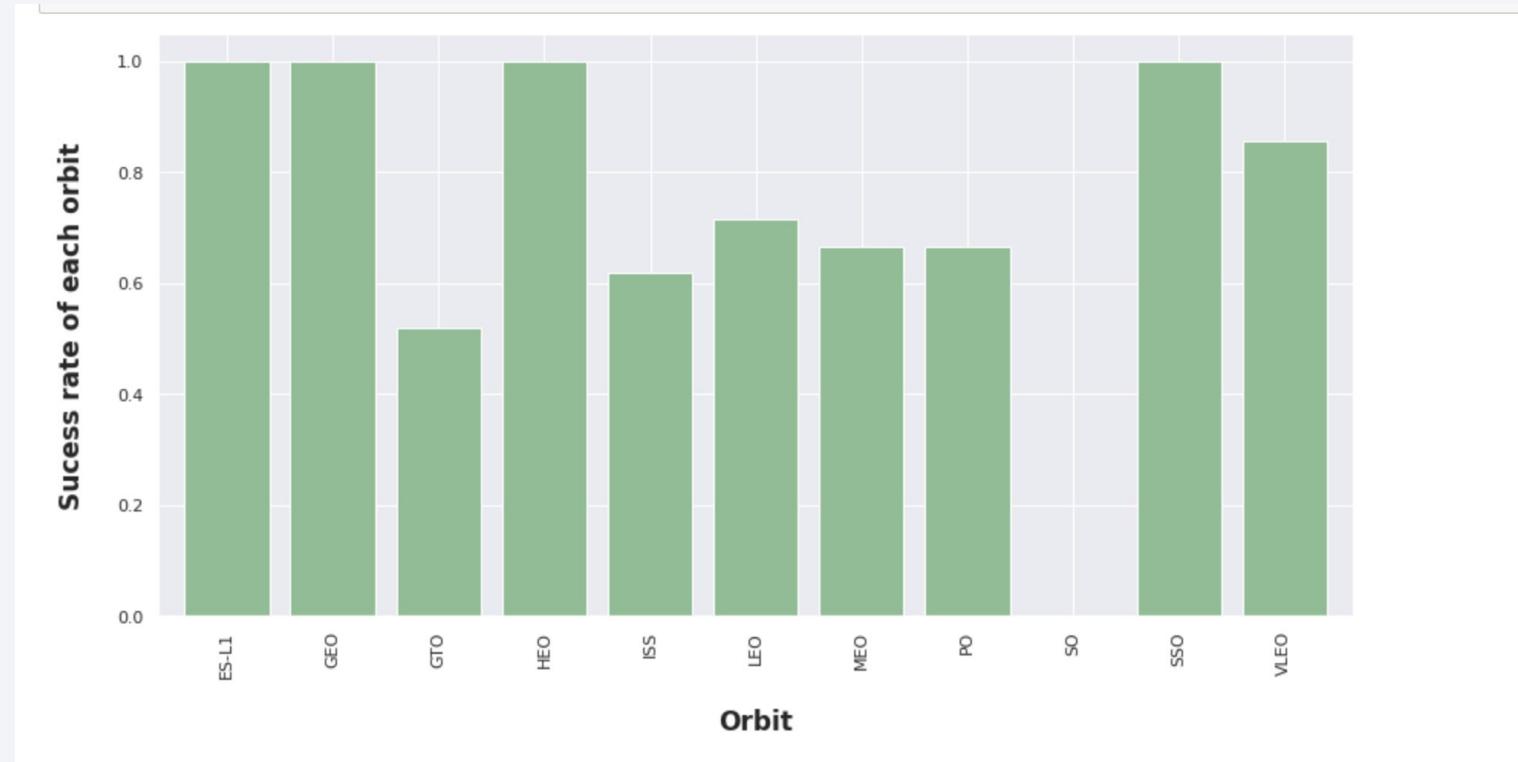
This scatter plot shows once the pay load mass is greater than 7000kg, the probability of the success rate will be highly increased.
However, there is no clear pattern to say the launch site is dependent to the pay load mass for the success rate.



Success Rate vs. Orbit Type

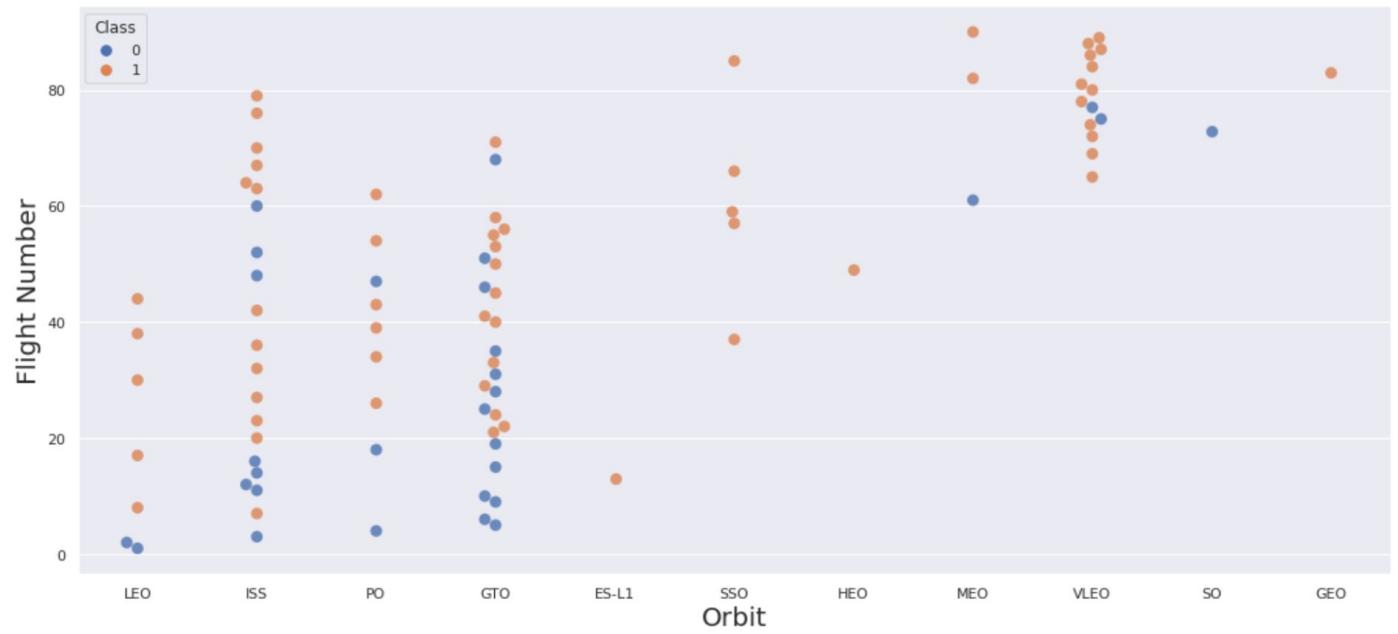
This figure depicted the possibility of the orbits to influences the landing outcomes as some orbits has 100% success rate such as SSO, HEO, GEO AND ES-L1 while SO orbit produced 0% rate of success.

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.



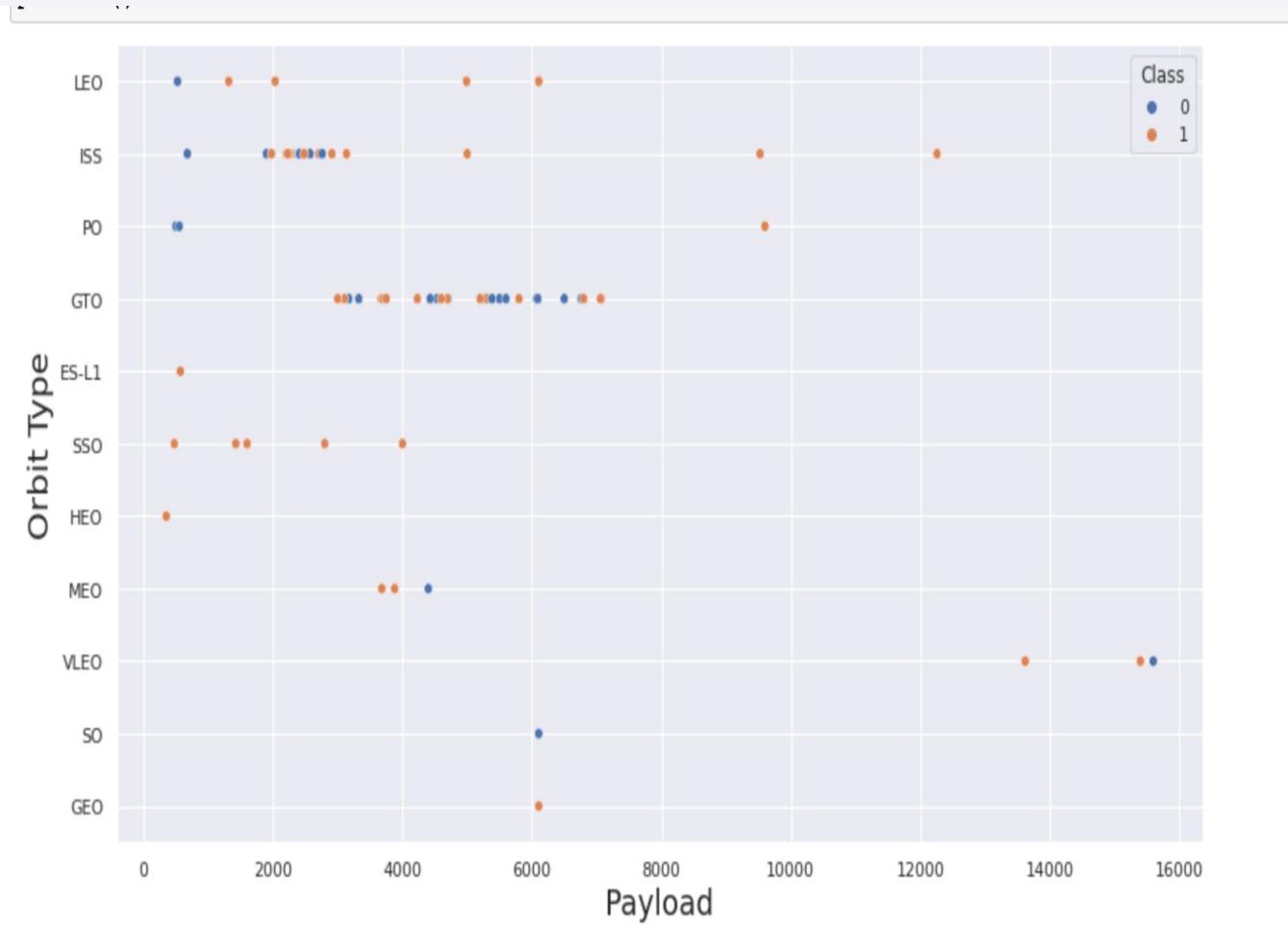
Flight Number vs. Orbit Type

You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



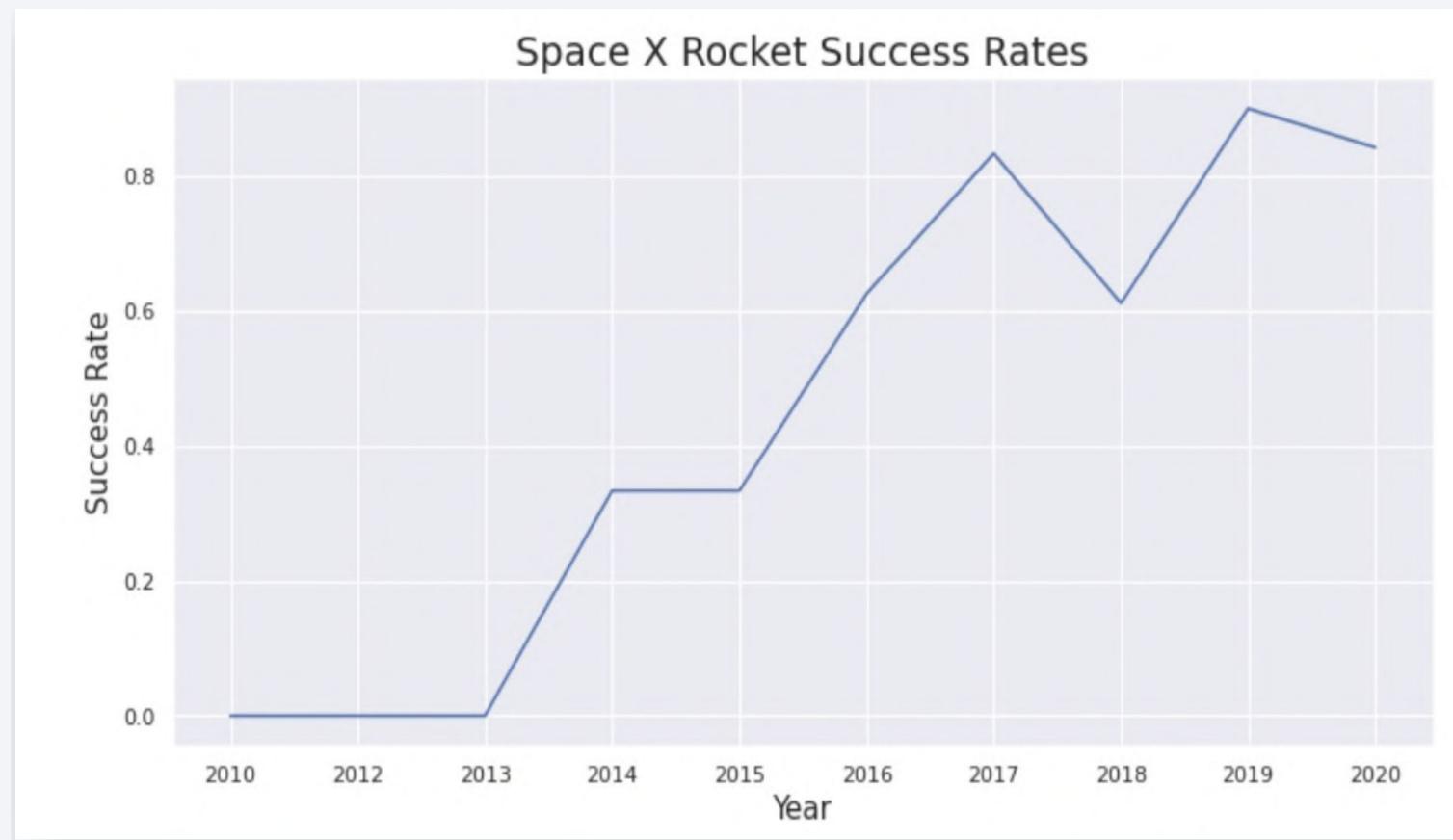
Payload vs. Orbit Type

With 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

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 as "Launch_Sites" FROM SPACEX;  
* ibm_db_sa://zpw86771:***@fbdb88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3  
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb  
Done.  
Out[5]: Launch_Sites  
CCAFS LC-40  
CCAFS SLC-40  
KSC LC-39A  
VAFB SLC-4E
```

Launch Site Names Begin with 'CCA'

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

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

In [11]: task_2 = """
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        """
create_pandas_df(task_2, database=conn)
```

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

Total Payload Mass

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

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)
```

```
* ibm_db_sa://zpw86771:***@fbdb88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Total Payload Mass by NASA (CRS)

45596

Average Payload Mass by F9 v1.1

We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)
```

```
* ibm_db_sa://zpw86771:***@fbdb88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb
Done.
```

Total Payload Mass by NASA (CRS)

45596

First Successful Ground Landing Date

Used 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) AS "First Sucessful Landing Outcome in Ground Pad  
WHERE LANDING_OUTCOME = 'Success (ground pad)';
```

```
* ibm_db_sa://zpw86771:***@fbdb88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3  
sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb  
Done.
```

First Sucessful Landing Outcome in Ground Pad

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 SPACEX WHERE LANDING_OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS_KG_ > 4000 AND PAYLOAD_MASS_KG_ < 6000;

* ibm_db_sa://zpw86771:***@fbdb88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.datab
ases.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

We used wildcard like '%' to filter for WHERE MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Success%';  
* ibm_db_sa://zpw86771:****@fdb88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb  
Done.  
Successful Mission  
100
```

```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Failure%';
```

```
* ibm_db_sa://zpw86771:****@fdb88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.databases.appdomain.cloud:32731/bludb  
Done.  
Failure Mission
```

Boosters Carried Maximum Payload

```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEX  
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);  
  
* ibm_db_sa://zpw86771:***@fdb88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lgde00.databases.appdomain.cloud:32731/bludb  
Done.  
  
Booster Versions which carried the Maximum Payload Mass  
F9 B5 B1048.4  
F9 B5 B1048.5  
F9 B5 B1049.4  
F9 B5 B1049.5  
F9 B5 B1049.7  
F9 B5 B1051.3  
F9 B5 B1051.4  
F9 B5 B1051.6  
F9 B5 B1056.4  
F9 B5 B1058.3  
F9 B5 B1060.2  
F9 B5 B1060.3
```

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

2015 Launch Records

We used a combinations of the WHERE clause, LIKE, AND, and BETWEEN conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015

```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
* ibm_db_sa://zpw86771:***@fbdb88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lgde00.
databases.appdomain.cloud:32731/bludb
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

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 BY clause to group the landing outcomes and the ORDER BY clause to order the grouped landing outcome in descending order.

```
%sql SELECT LANDING_OUTCOME as "Landing Outcome", COUNT(LANDING_OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY LANDING_OUTCOME \
ORDER BY COUNT(LANDING_OUTCOME) DESC ;
```

```
* ibm_db_sa://zpw86771:***@fbdb88901-ebdb-4a4f-a32e-9822b9fb237b.clogj3sd0tgtu0lqde00.databases.appdomain.c
loud:32731/bludb
Done.
```

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

The background of the slide is a photograph taken from space at night. It shows the curvature of the Earth's horizon against a dark blue sky. City lights are visible as small white dots, with larger clusters of lights indicating major urban areas. In the upper right corner, there is a faint, greenish glow of the aurora borealis or a similar atmospheric phenomenon.

Section 3

Launch Sites Proximities Analysis

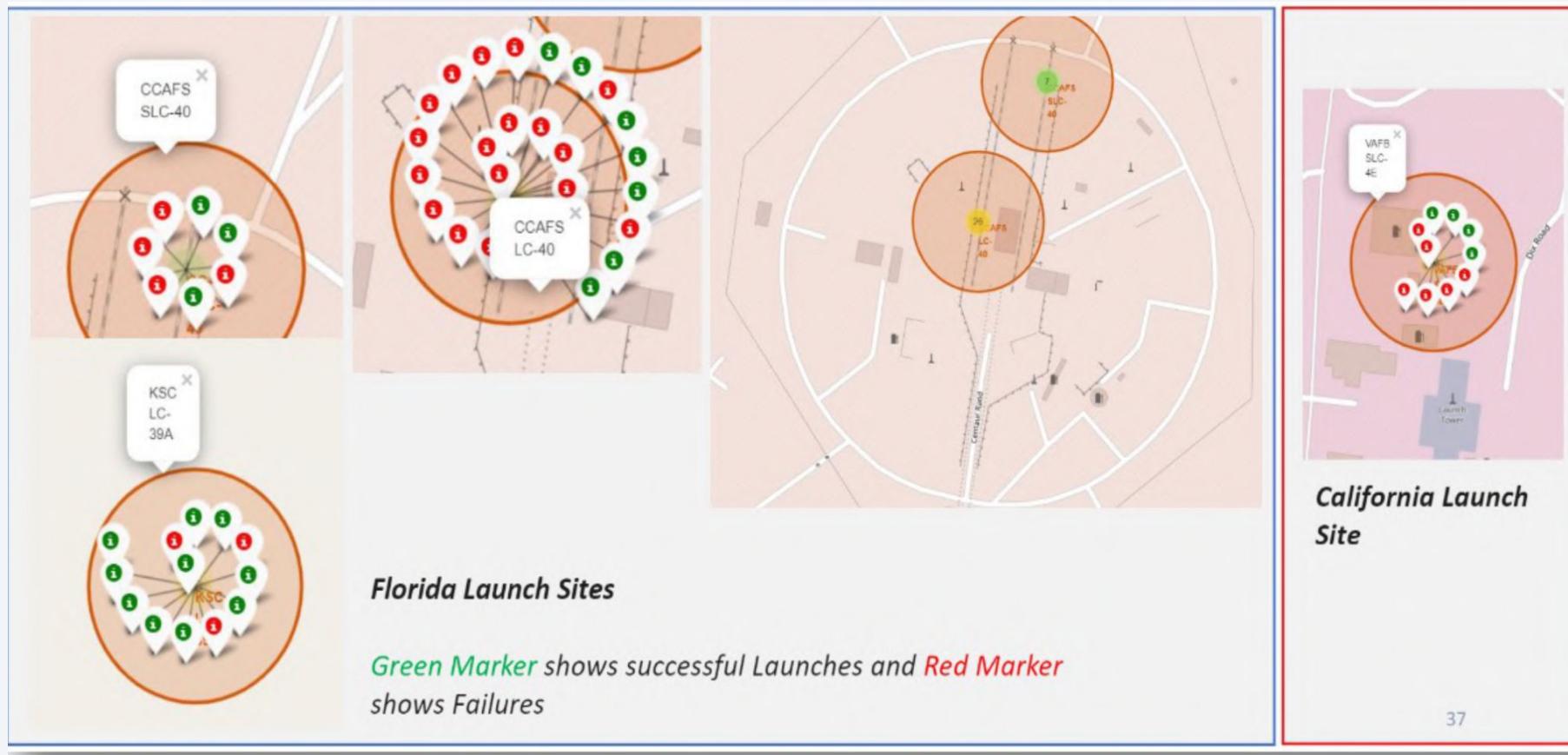
Location of all LaunchSites

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

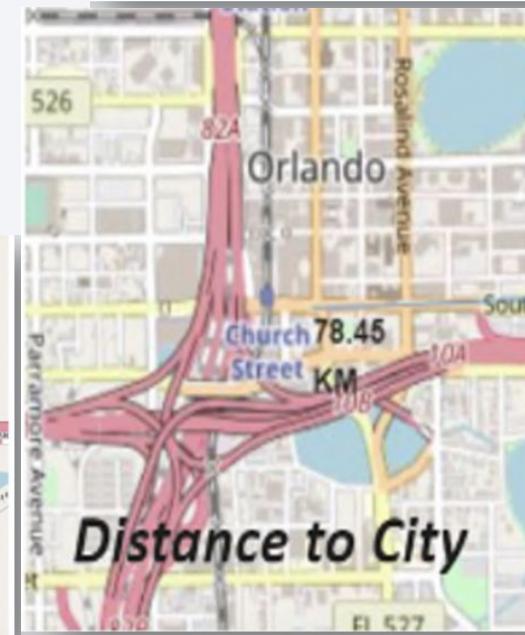
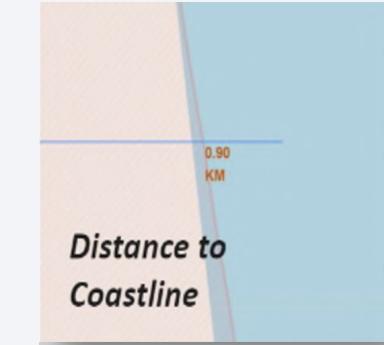


Markers showing launch sites

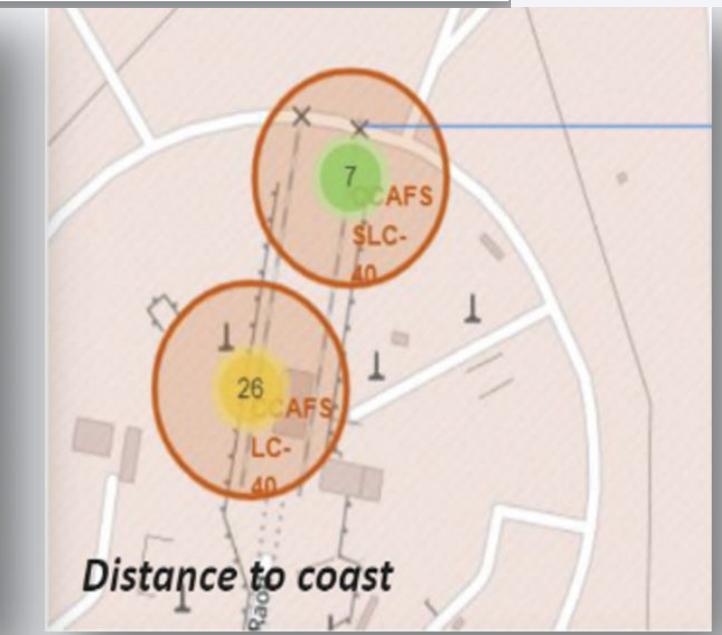
Markers showing launch sites with colors



Launch sites distance to landmarks

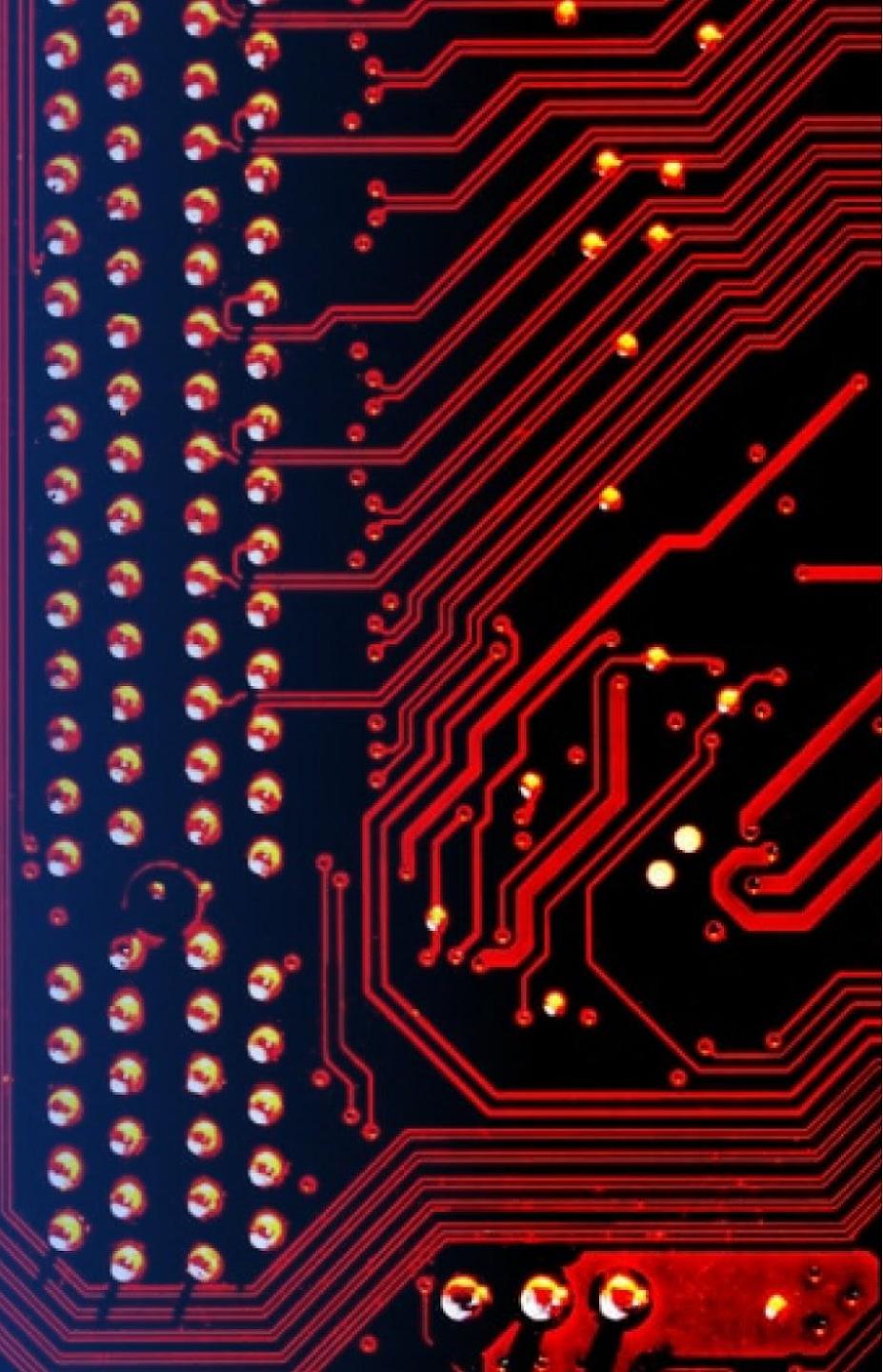


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

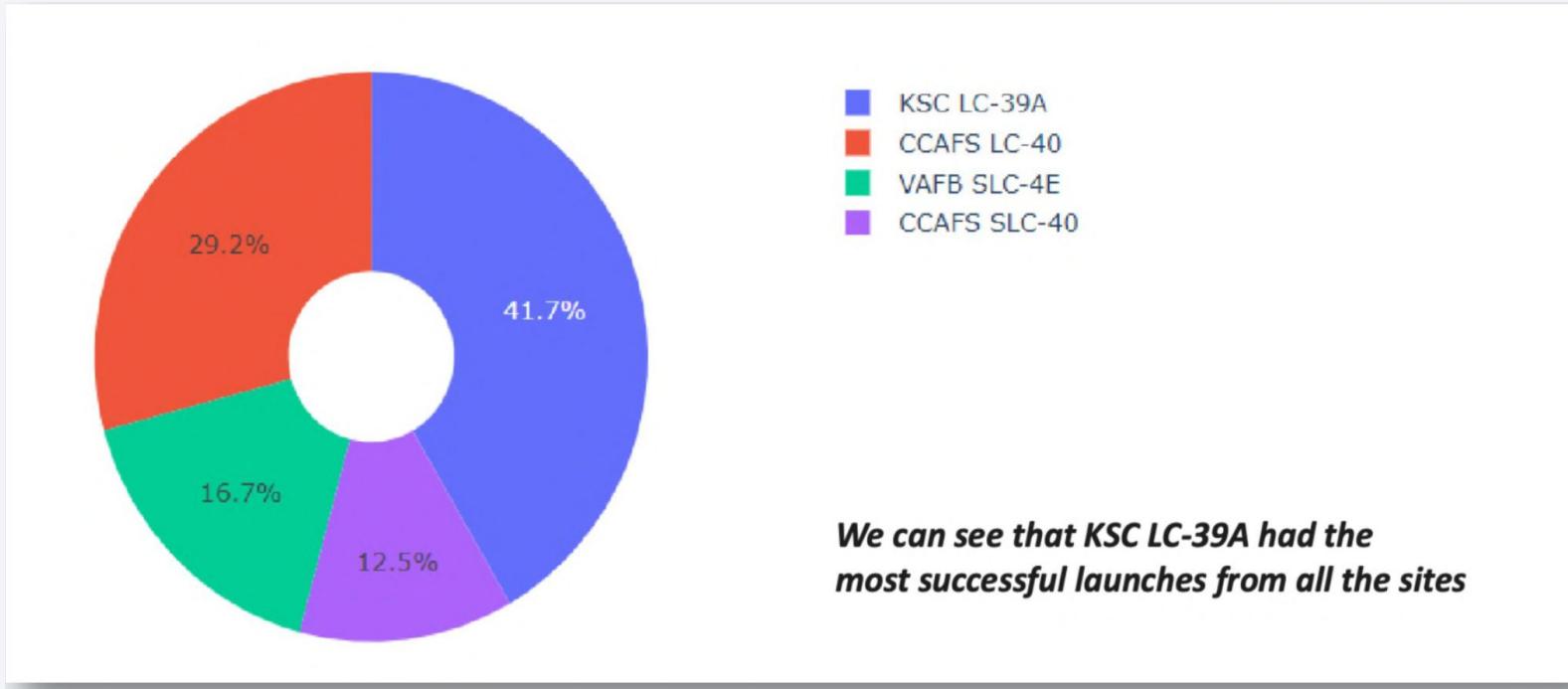


Section 4

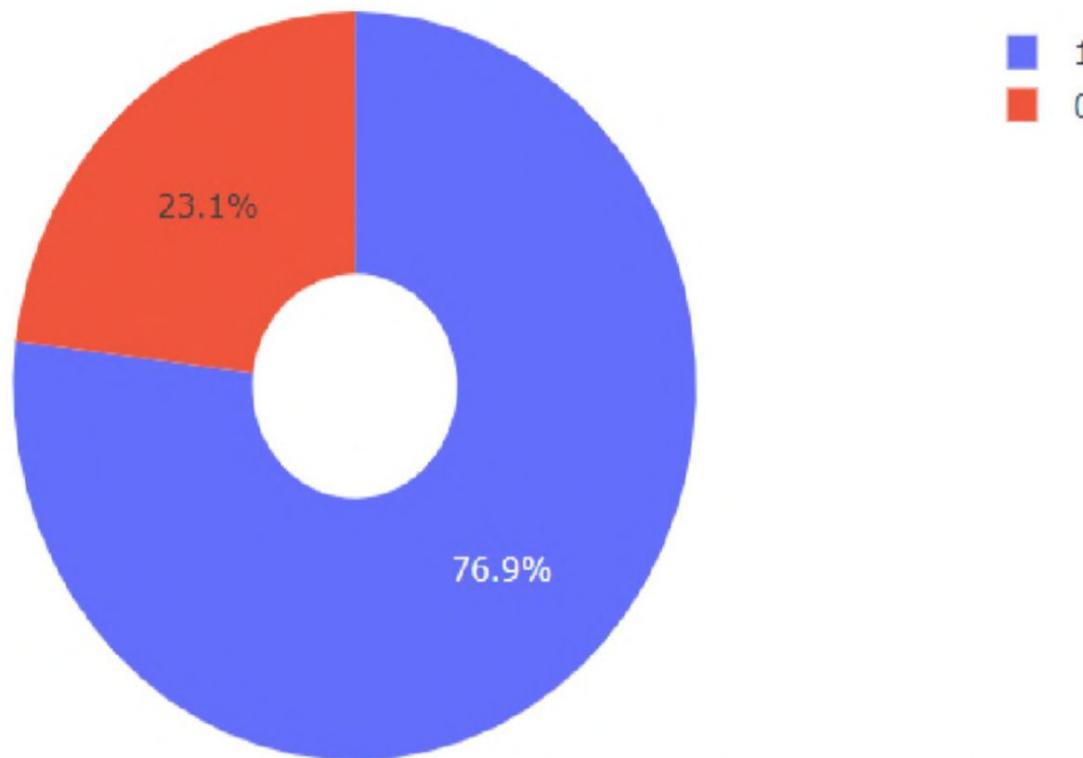
Build a Dashboard with Plotly Dash



The Success percentage by each sites

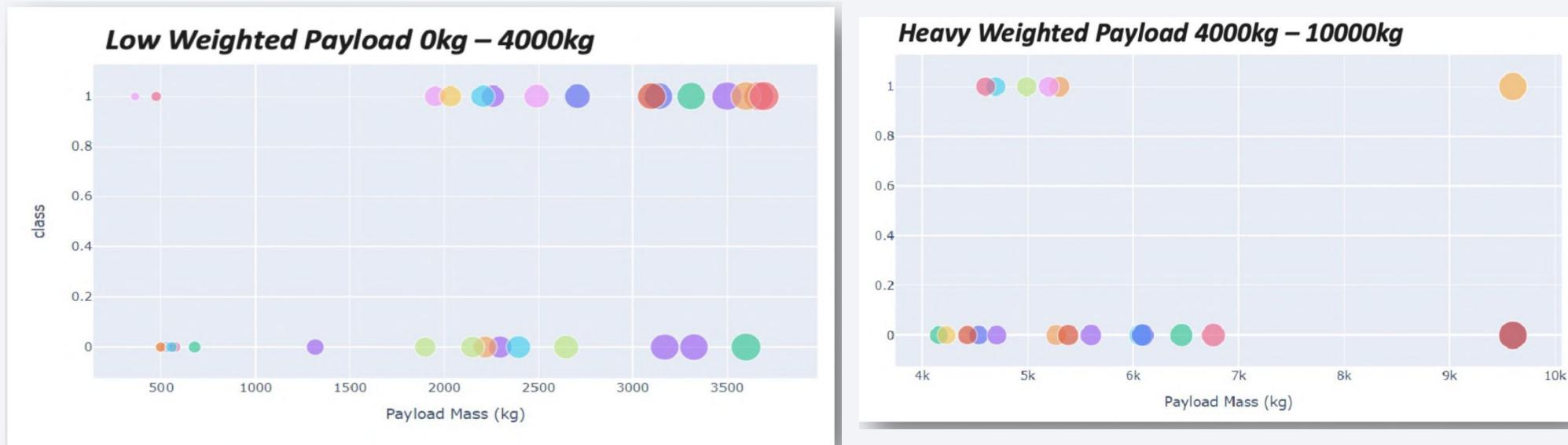


The highest launch-success ratio: KSC LC-39A



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

Payload Vs Launch Outcome Scatter Plot



Section 5

Predictive Analysis (Classification)

Classification Accuracy

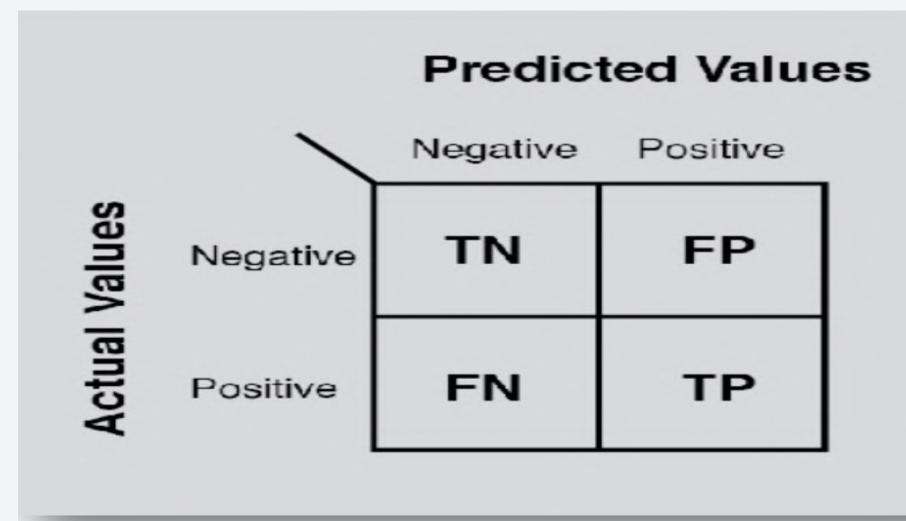
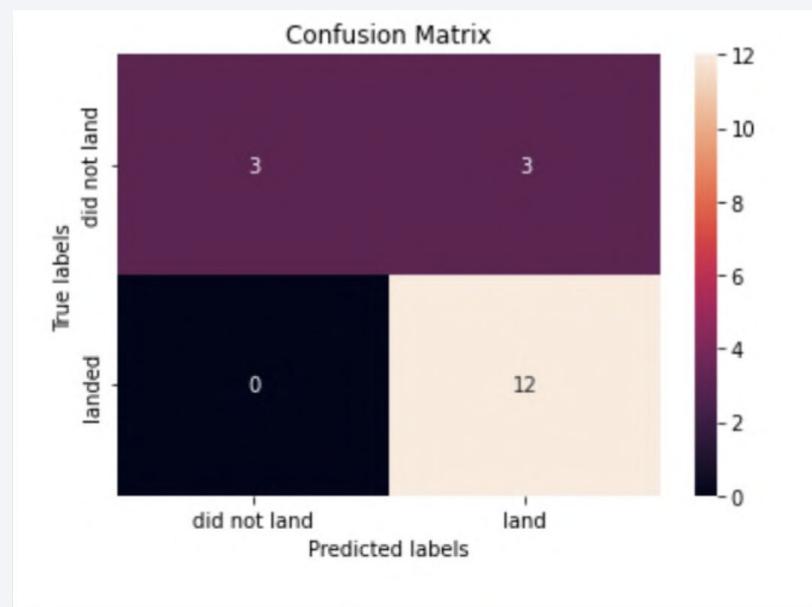
As we can see, by using the code as below: we could identify that the best algorithm to be the Tree Algorithm which have the highest classification accuracy.

```
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])
if bestalgorithm == 'Tree':
    print('Best Params is :',tree_cv.best_params_)
if bestalgorithm == 'KNN':
    print('Best Params is :',knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best Params is :',logreg_cv.best_params_)

Best Algorithm is Tree with a score of 0.9017857142857142
Best Params is : {'criterion': 'entropy', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
```

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!

