IBM Data Science Capstone Project
Space X

Lurvïsh Polodoo
18/11/2021

WINNING SPACE RACE WITH DATA SCIENCE

Outline

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

Executive Summary

- **Summary of methodologies**

- Data Collection
- Data Wrangling
- EDA with Visualization and with SQL
- Building an interactive map with Folium
- Building a Dashboard with Plotly Dash
- Predictive analysis (Classification)

- **Summary of all results**

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

Introduction

- **Project Background and Context**

We predicted if the Falcon 9 first stage will land successfully.
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.

- **Problems we want to find solutions to**

- What influences if the rocket will land successfully?
- The effect each relationship with certain rocket variables will impact in determining the success rate of a successful landing.
- What conditions does SpaceX have to achieve to get the best results and ensure the best rocket success landing rate.

Methodology

- **Data collection methodology:**

  - Web Scraping from Wikipedia

  - SpaceX Rest API

- **Perform data wrangling**

  - One Hot Encoding data fields for Machine Learning and dropping irrelevant columns

- **Perform exploratory data analysis (EDA) using visualization and SQL**

  - Plotting of Scatter Graphs, Bar Graphs to show relationships between variables to show patterns of data.

- **Perform interactive visual analytics using Folium and Plotly Dash**

- **Perform predictive analysis using classification models**

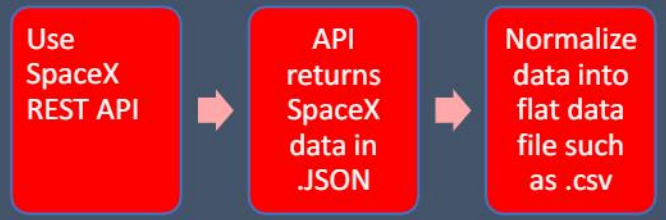
  - Building different models such as Linear Regression, Decision Tree and evaluating the models using F1-score, Jaccard score etc.

Methodology

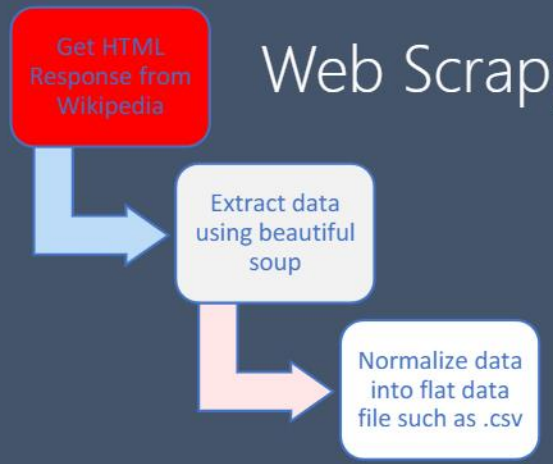
- The following datasets was collected by
  - We worked with SpaceX launch data that is gathered from the SpaceX REST API.
  - This API will give us data about launches, including information about the rocket used,

  payload delivered, launch specifications, landing specifications, and landing outcome.
  - Our goal is to use this data to predict whether SpaceX will attempt to land a rocket or not.
  - The SpaceX REST API endpoints, or URL, starts with api.spacexdata.com/v4/.
  - Another popular data source for obtaining Falcon 9 Launch data is web scraping Wikipedia using BeautifulSoup.

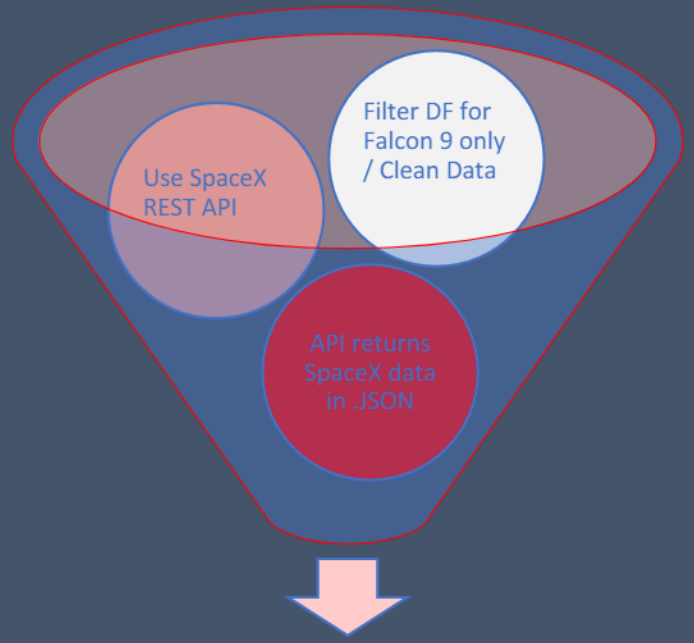
SpaceX API



Web Scrapping



Data collection – SpaceX API



Normalize data into flat data file such as .csv

[GitHub link to Notebook]

1 .Getting Response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url)
```

2. Converting Response to a .json file

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

3. Apply custom functions to clean data

```
getBoosterVersion(data), getLaunchSite(data), getPayloadData(data), getCoreData(data)
```

4. Assign list to dictionary then dataframe

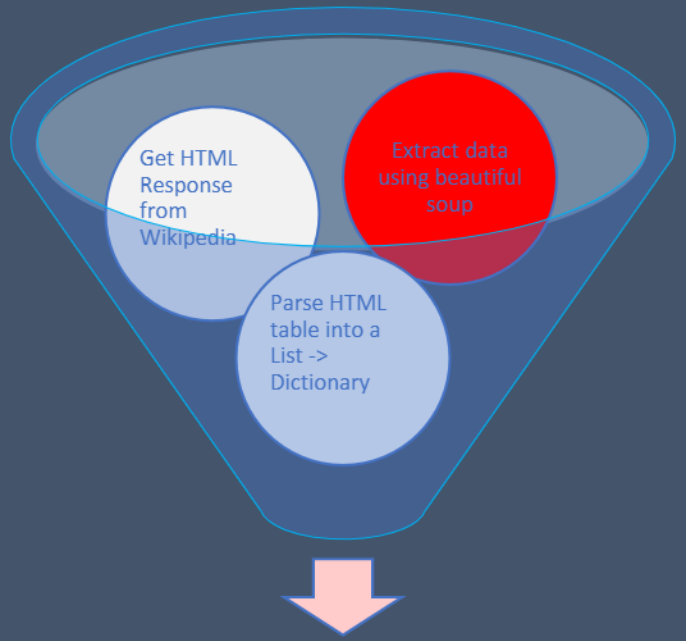
```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}

# Create a data from launch_dict
df = pd.DataFrame(launch_dict)
```

5. Filter dataframe and export to flat file (.csv)

```
data_falcon9 = df[df['BoosterVersion'] != 'Falcon 1']
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data collection – Web Scrapping



Normalize data into flat data file such as .csv

[GitHub link to Notebook]

1 .Getting Response from HTML

```
# use requests.get() method with the provided static_url
page = requests.get(static_url)
```

2. Creating BeautifulSoup Object

```
soup = BeautifulSoup(page.text, 'lxml')
```

3. Finding tables

```
html_tables = soup.find_all('table')
```

4. Getting column names

```
column_names = []

for item in first_launch_table.find_all('th'):
    col_name = extract_column_from_header(item)

    if col_name != None and len(col_name) > 0:
        column_names.append(col_name)
```

5. Creation of dictionary

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

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']=[]
```

6. Appending data to keys (Refer to notebook block 15)

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('t
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as n
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
```

7. Converting dictionary to dataframe

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

8. Dataframe to .CSV

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

Data Wrangling

Introduction

In the data set, there are several different cases where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident; for example, True Ocean means the mission outcome was successfully landed to a specific region of the ocean while False Ocean means the mission outcome was unsuccessfully landed to a specific region of the ocean. True RTLS means the mission outcome was successfully landed to a ground pad False RTLS means the mission outcome was unsuccessfully landed to a ground pad. True ASDS means the mission outcome was successfully landed on a drone ship False ASDS means the mission outcome was unsuccessfully landed on a drone ship.
We mainly convert those outcomes into Training Labels with 1 means the booster successfully landed 0 means it was unsuccessful.

Process

Perform Exploratory Data Analysis EDA on dataset

Calculate the number of launches at each site

Calculate the number and occurrence of each orbit

Calculate the number and occurrence of mission outcome per orbit type

Export dataset as .CSV

Create a landing outcome label from Outcome column

Work out success rate for every landing in dataset

[GitHub link to Notebook]

Each launch aims to an dedicated orbit, and here are some common orbit types:

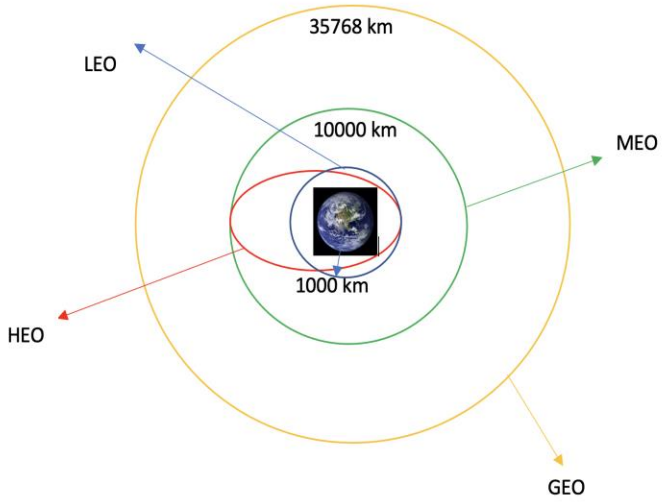


Diagram showing common orbit types SpaceX uses

EDA with Data Visualization

Scatter Graphs being drawn:

Flight Number vs Launch Site

Payload vs Launch Site

Flight Number vs Orbit Type

Payload vs Orbit Type

Scatter plots are used to show how much one variable is affected by another.
The relationship between the two variables are called a correlation.

Bar Chart being drawn:

Success Rate vs Orbit Type

A bar graph makes it easy to compare sets of data between different groups. The graph represents categories on one axis and discrete values on the other axis. The goal is to represent the relationship between the two axes.

Line Graph being drawn:

Launch Success Yearly Trend (Success Rate vs Year)

A line graph shows us clearly how two variables vary. We can use line graphs to predict results of data not yet seen.

GitHub link to Notebook

EDA with SQL

Perfomed SQL queries to gather information about the dataset.

The questions which were answered using SQL queries are:

- ***Display the names of the unique launch sites in the space mission***
- ***Display 5 records where launch sites begin with the string 'CCA'.***
- ***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 when the first successful landing outcome in ground pad was achieved.***
- ***List the names of the boosters which have success in drone ship 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. Use a subquery***
- ***List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015***
- ***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***

GitHub link to EDA with SQL

Building 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 assigned the dataframe launch_outcomes(failures, successes) to *classes 0 and 1* with Green and Red markers on the map in a MarkerCluster()

Using Haversine's formula we calculated the distance from the Launch Site to various landmarks to find various trends about what is around the Launch Site to measure patterns. **Lines** are drawn on the map to measure distance to landmarks

Example of some trends in which the Launch Site is situated in.

- Are launch sites in close proximity to railways? Yes
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

GitHub link to Notebook

Built an interactive dashboard with Flask and Dash

GitHub link to Notebook

- **The dashboard is built with Flask and Dash web framework.**

Graphs

- **Pie Chart showing the total launches by a certain site/all sites**
  - *display relative proportions of multiple classes of data.*
  - *size of the circle can be made proportional to the total quantity it represents.*

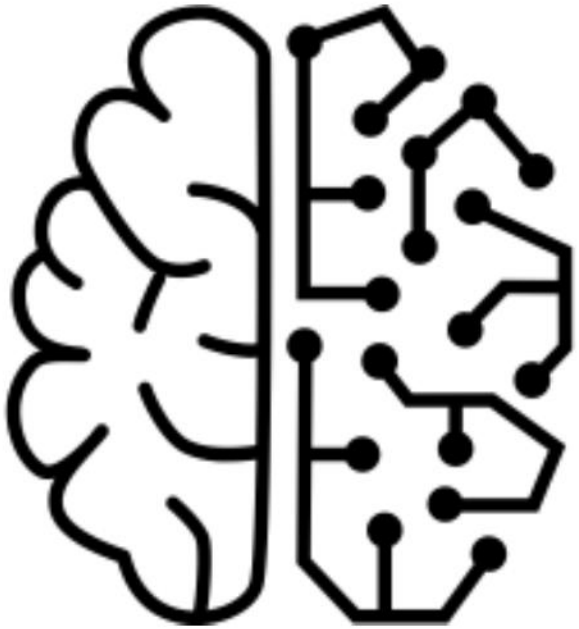
Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions

- It shows the relationship between two variables.
- It is the best method to show you a non-linear pattern.
- The range of data flow, i.e. maximum and minimum value, can be determined.
- Observation and reading are straightforward.

Predictive analysis (Classification)

BUILDING MODEL

- Load our dataset into NumPy and Pandas
- Transform Data
- Split our data into training and test data sets
- Check how many test samples we have
- Decide which type of machine learning algorithms we want to use
- Set our parameters and algorithms to GridSearchCV
- Fit our datasets into the GridSearchCV objects and train our dataset.



EVALUATING MODEL

- Check accuracy for each model
- Get tuned hyperparameters for each type of algorithms
- Plot Confusion Matrix

IMPROVING MODEL

- Feature Engineering
- Algorithm Tuning

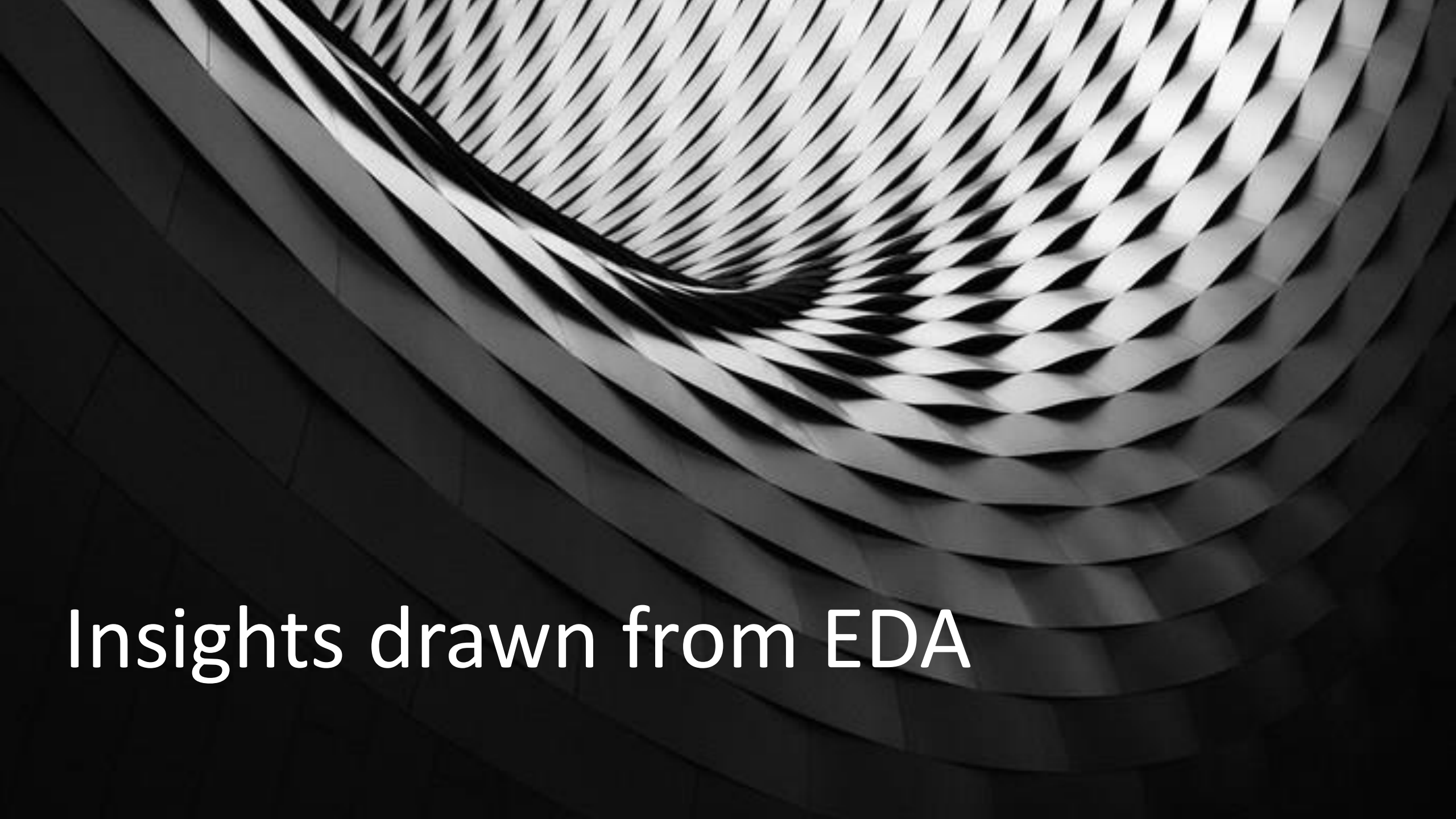
FINDING THE BEST PERFORMING CLASSIFICATION MODEL

- The model with the best accuracy score wins the best performing model
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook.

GitHub link to Notebook

Results

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

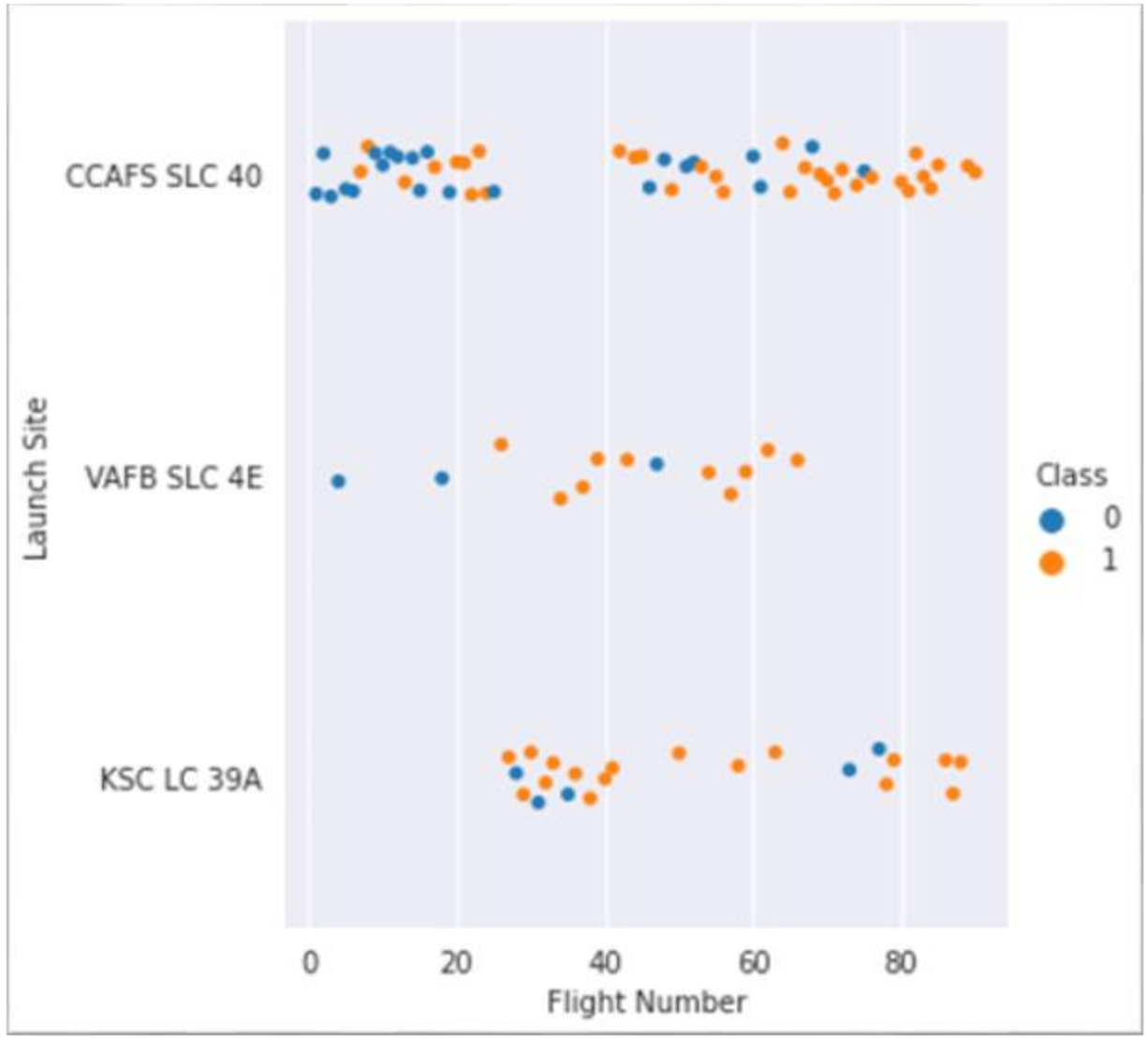


Insights drawn from EDA

EDA with Visualization

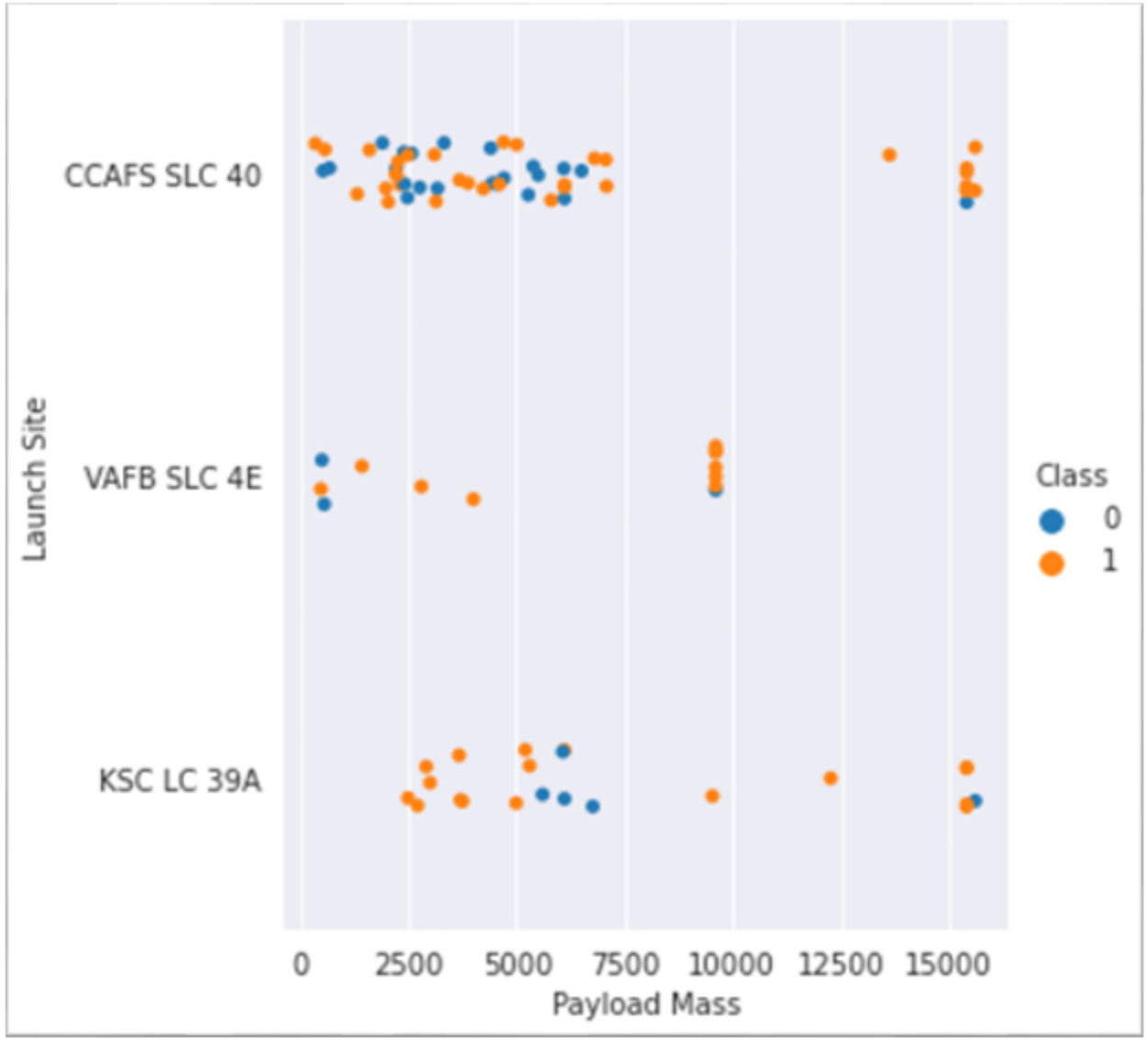
Flight Number vs. Launch Site

- We can see that there has been more flights at the launch site CCAFS SLC 40.
- There is no strong relationship between number of flights and success rate.



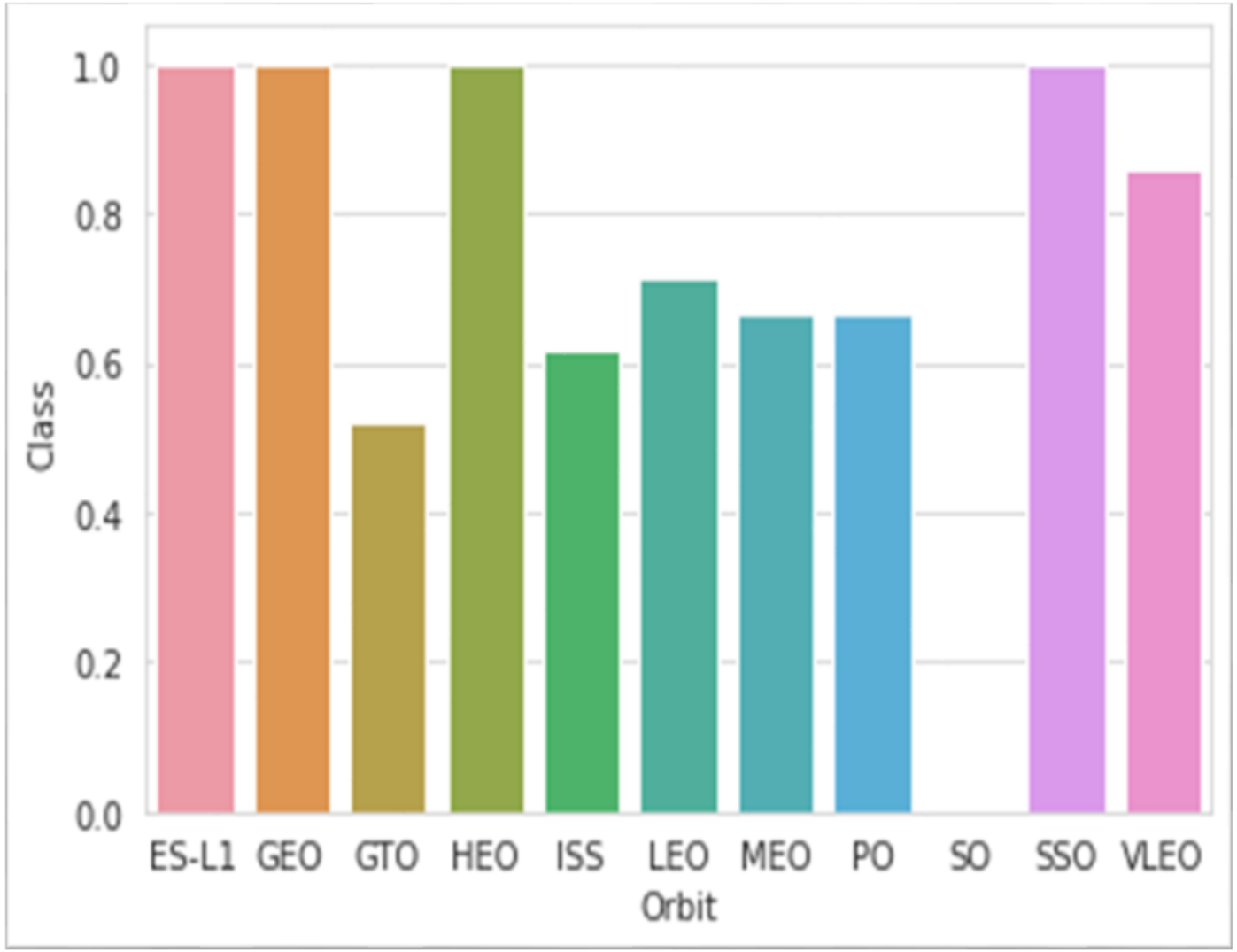
Payload vs. Launch Site

- We can denote that higher payload masses have higher success rates for most cases.
- KSC LC 39A launch site tends to have 100% success rate for lower payload masses (less than 5000kg)



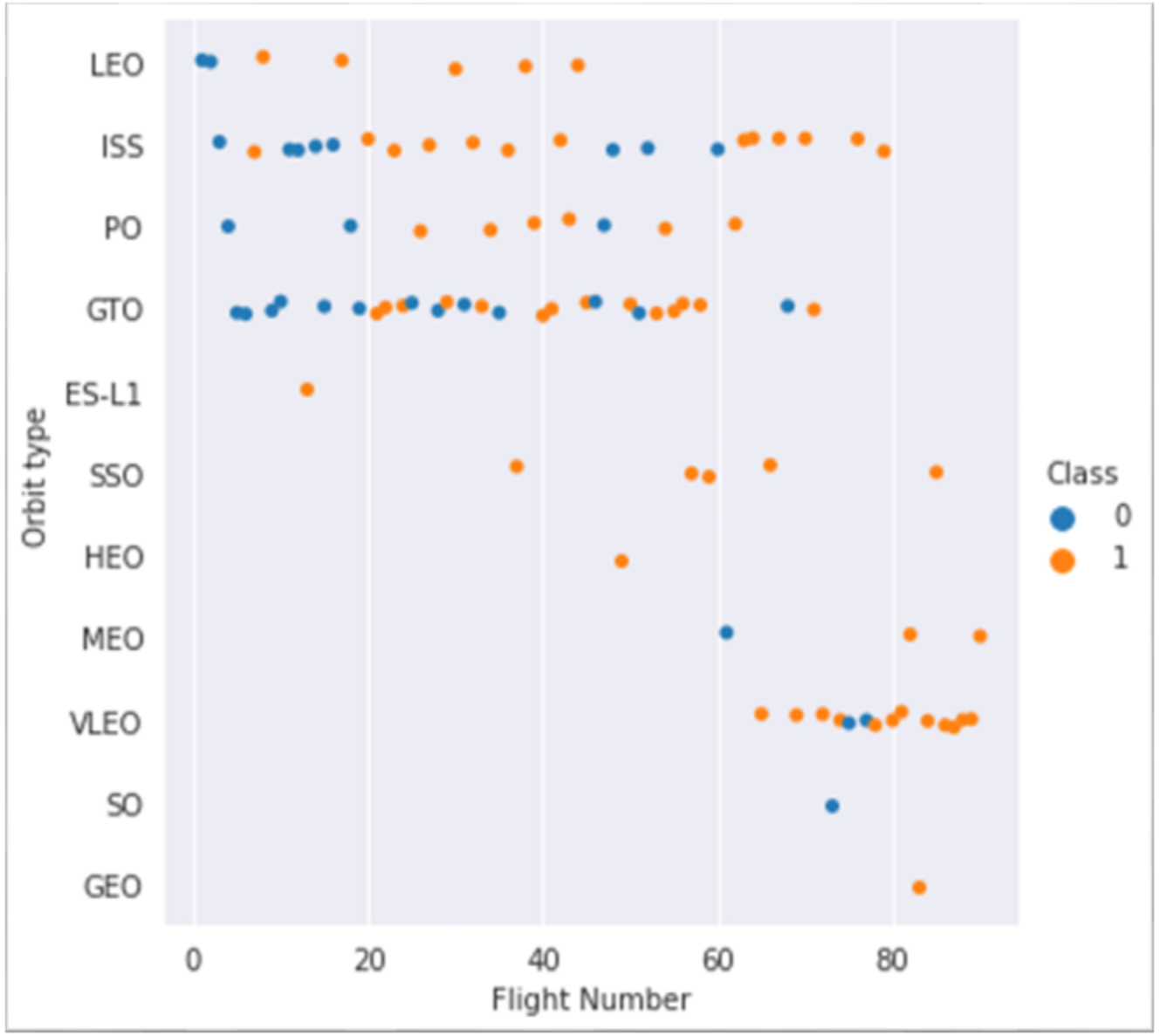
Success Rate vs. Orbit Type

- It is clear that orbits ES-L1, GEO, HEO and SSO have the highest success rate.
- Orbit SO has zero success or no attempt.



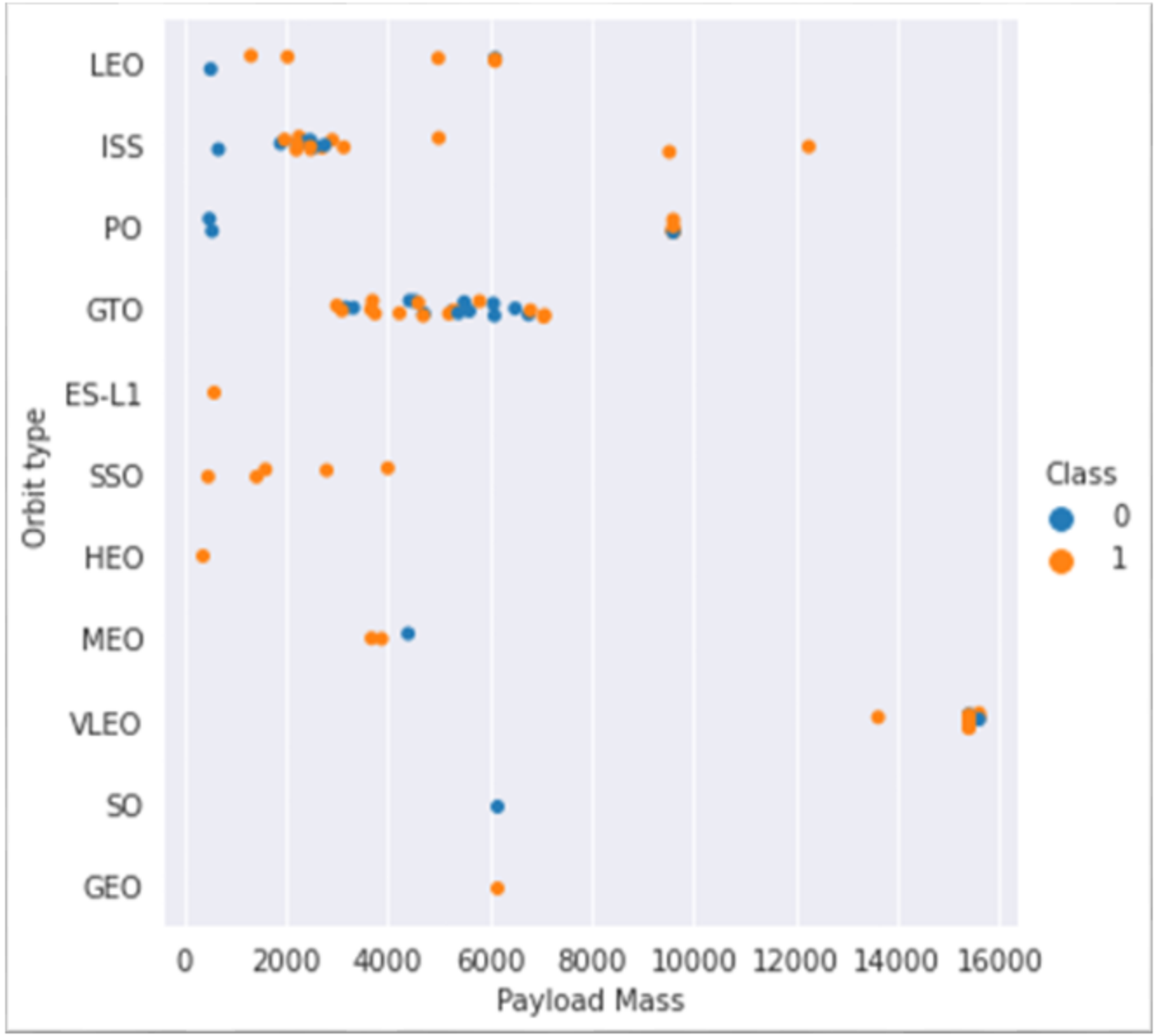
Flight Number vs. Orbit Type

- LEO orbit appears to be linearly correlated to the number of flights.
- Most orbits seem to have a positive success rate except for GTO orbit where there seems to have no relationship with number of flights.



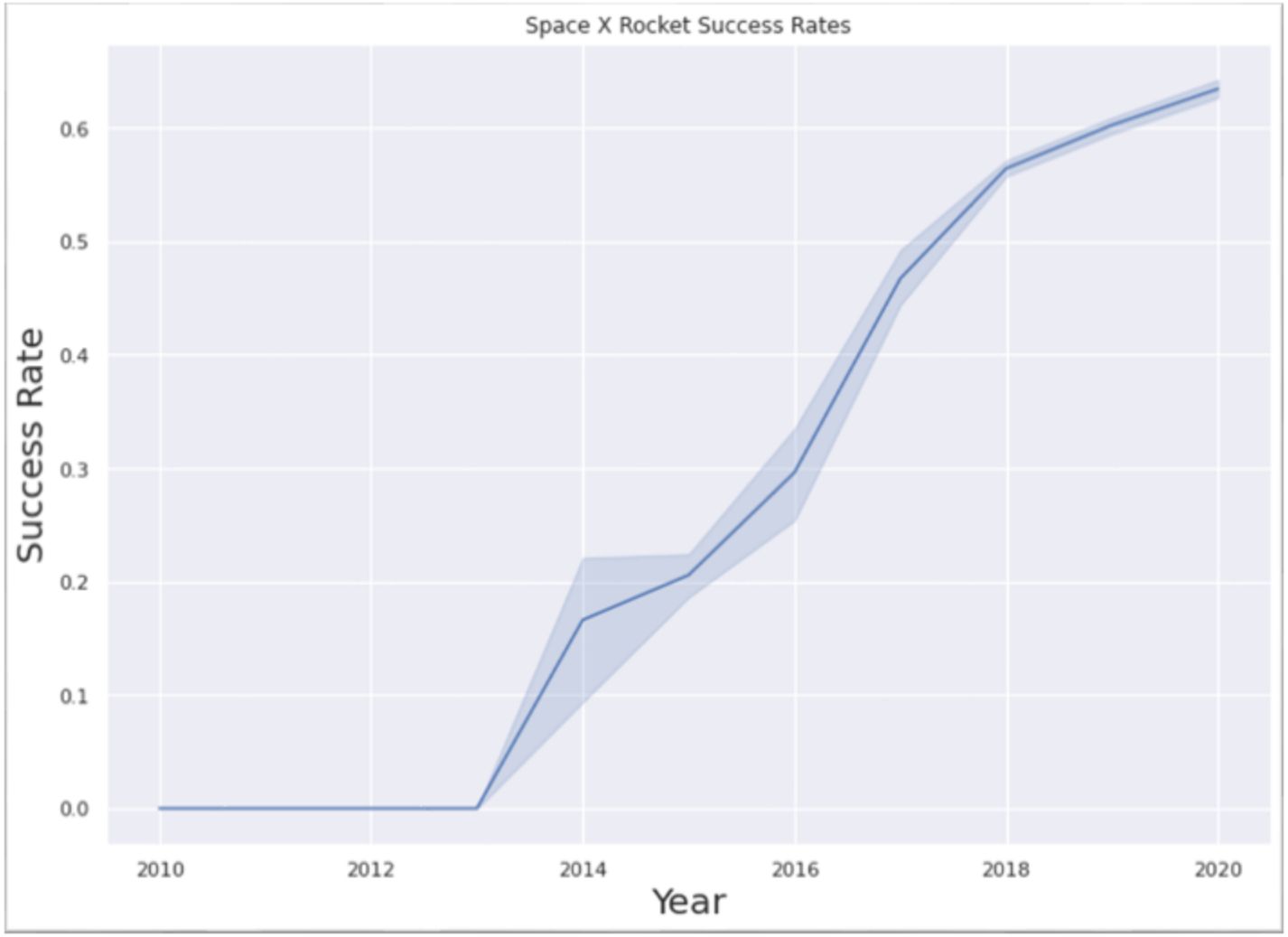
Payload vs. Orbit Type

We can observe that higher payload affects GTO orbit negatively.



Launch Success Yearly Trend

We can see that the success rate kept increasing from 2013 to 2020




EDA WITH
.SQL

All Unique Launch Site Names

SQL Query

```
select unique("LAUNCH_SITE")
from SPACEXDATASET
```

SQL Query result:

Unique Launch Sites
CCAFS LC-40
CCAFS SLC-40
KSC LC-39A
VAFB SLC-4E

SQL Query explanation

Using the word **unique** means that we will get only the unique/distinct **launch site names** from the table **SPACEXDATASET**

First 5 Launch Site Names Beginning with 'CCA'

SQL Query

```
select * from SPACEXDATASET
where "LAUNCH_SITE" like 'CCA%'
limit 5;
```

SQL Query explanation

Using the * symbol to select all the columns from the table **SPACEXDATASET**
Further using the **where** keyword to find the launch sites which start with **'CCA'** by making use of a wildcard('**%**')
Using **limit 5** to get the first 5 results from the result set

SQL Query result

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 from Customer NASA (CRS)

SQL Query

```
select sum("PAYLOAD_MASS__KG_")
from SPACEXDATASET
where "CUSTOMER"='NASA (CRS)';
```

SQL Query result

Total Payload Mass
45596

SQL Query explanation

Using the function **sum** to find the total of the column **PAYLOAD_MASS__KG_** from the table **SPACEXDATASET**

The **where** clause filters the dataset to only the **'NASA (CRS)'** customers

Average Payload Mass by F9 v1.1

SQL Query

```
select avg("PAYLOAD_MASS__KG_")
from SPACEXDATASET
where "BOOSTER_VERSION"='F9 v1.1'
```

SQL Query result

Average Payload Mass by F9 v1.1
2928

SQL Query explanation

Using the function **avg** to find the average of the column **PAYLOAD_MASS__KG_** from the table **SPACEXDATASET**

The **where** clause filters the dataset to only the **'F9 v1.1'** booster version

First Successful Ground Pad Landing Date

SQL Query

```
select min("DATE") from SPACEXDATASET
where "LANDING__OUTCOME"='Success (ground pad)';
```

SQL Query result

Date at which first successful outcome in ground pad was achieved
2015-12-22

SQL Query explanation

Using the function **min** to find the minimum date of the column **DATE** from the table **SPACEXDATASET**

The **where** clause filters the dataset to only the landing outcome which were **'Successful with ground pad'**

Successful Drone Ship Landing with Payload between 4000kg and 6000kg

SQL Query

```
select "BOOSTER_VERSION" from SPACEXDATASET
where "LANDING__OUTCOME"='Success (drone ship)'
and "PAYLOAD_MASS__KG_" between 4000 and 6000;
```

SQL Query explanation

Selecting the **BOOOSTER_VERSION** from the table **SPACEXDATASET**

The **where** clause filters the dataset to only the landing outcome which were **'Successful with drone ship'** and the payload mass found between 4000kg and 6000kg

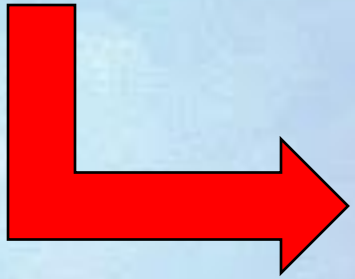
SQL Query result

Names of boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

Total Number of Successful and Failure Mission Outcomes

SQL Query

```
select (select COUNT("MISSION_OUTCOME") from SPACEXDATASET
        where "MISSION_OUTCOME" like '%Success%'),
        (select COUNT("MISSION_OUTCOME") from SPACEXDATASET
         where "MISSION_OUTCOME" like '%Failure%')
from SPACEXDATASET
limit 1;
```



SQL Query result

Number of Successful missions	Number of Failed missions
100	1

SQL Query explanation

Selecting the **Number of Successful missions** and the **Number of Failed missions** from the table **SPACEXDATASET**

Using **subqueries** and **wildcards** to get the count of the respective mission outcomes

Boosters which carried the Maximum Payload

SQL Query

```
select "BOOSTER_VERSION" from SPACEXDATASET
where "PAYLOAD_MASS__KG_" = (select max("PAYLOAD_MASS__KG_") from SPACEXDATASET)
order by "BOOSTER_VERSION";
```

SQL Query result

Names of boosters which have 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

SQL Query explanation

Selecting the **BOOOSTER_VERSION** from the table **SPACEXDATASET**

Using a subquery with the function **max** to get the **maximum payload mass**

The **where** clause filters the dataset to only the boosters which carried the **maximum payload mass**

Ordering the result by **BOOSTER_VERSION**

Launch records which failed with drone ship in 2015

SQL Query

```
select "DATE", "BOOSTER_VERSION", "LAUNCH_SITE", "LANDING__OUTCOME"
from SPACEXDATASET
where "LANDING__OUTCOME"='Failure (drone ship)'
and year("DATE") = 2015;
```

SQL Query result

DATE	booster_version	launch_site	landing__outcome
2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

SQL Query explanation

Selecting **DATE**, **BOOSTER_VERSION**, **LAUNCH_SITE**, **LANDING__OUTCOME** from the table **SPACEXDATASET**

The **where** clause filters the dataset to only the Landing Outcomes which **Failed with drone ship**
and
Using the **year** function to get the year of the **DATE** which is set to **2015**

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

SQL Query

```
select "LANDING__OUTCOME", count("LANDING__OUTCOME")
from SPACEXDATASET
where "DATE" between '2010-06-04' and '2017-03-20'
group by "LANDING__OUTCOME"
order by count("LANDING__OUTCOME") desc;
```

SQL Query result

Landing outcome	Between 2010-06-04 and 2017-03-20
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

SQL Query explanation

Selecting the **LANDING__OUTCOME** and **COUNT** of **LANDING__OUTCOME** from the table **SPACEXDATASET**

The **where** clause filters the dataset to only the Landing Outcomes Between **2010-06-04** and **2017-03-20**

Grouping by the **LANDING__OUTCOME**
Ordering the result by **COUNT** of **LANDING__OUTCOME**

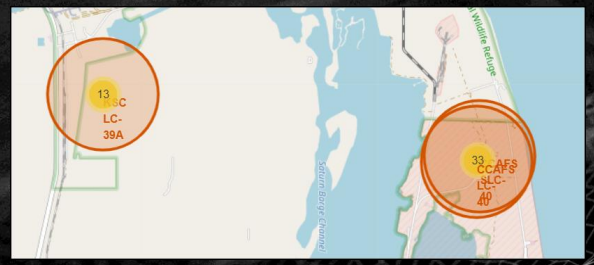

Interactive maps with Folium

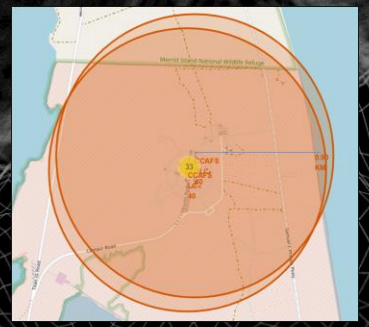
All launch sites indicated with markers on global map

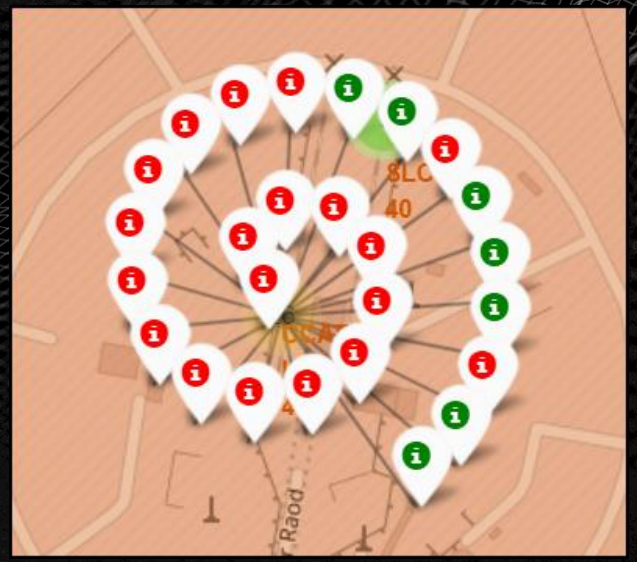
It can be seen that the SpaceX launch sites are found on the coasts of California and Florida in the United States of America

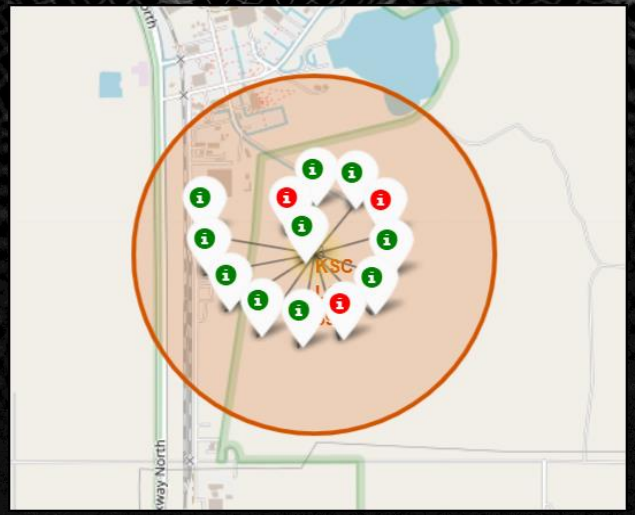
Colour-labelled markers at different launch sites

Florida

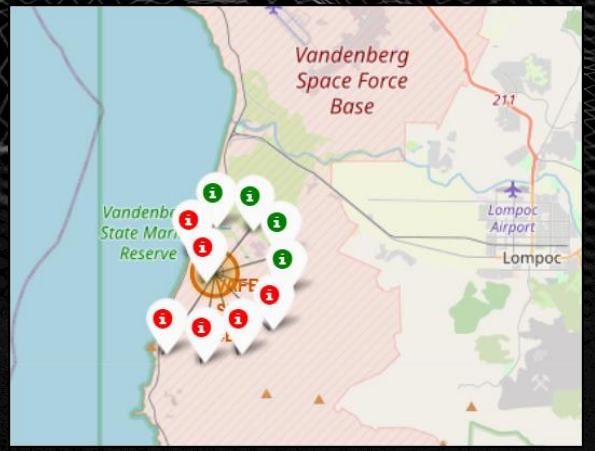






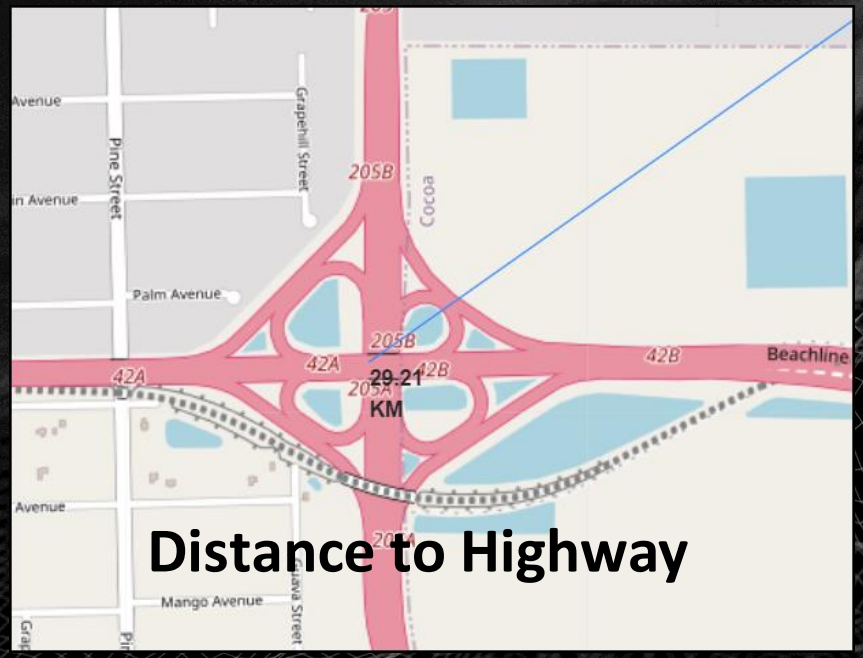


California

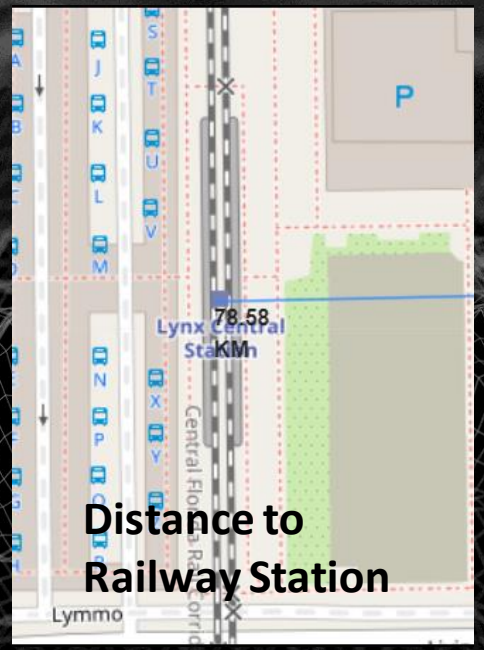


Green markers show successful launches. Red markers show failed launches

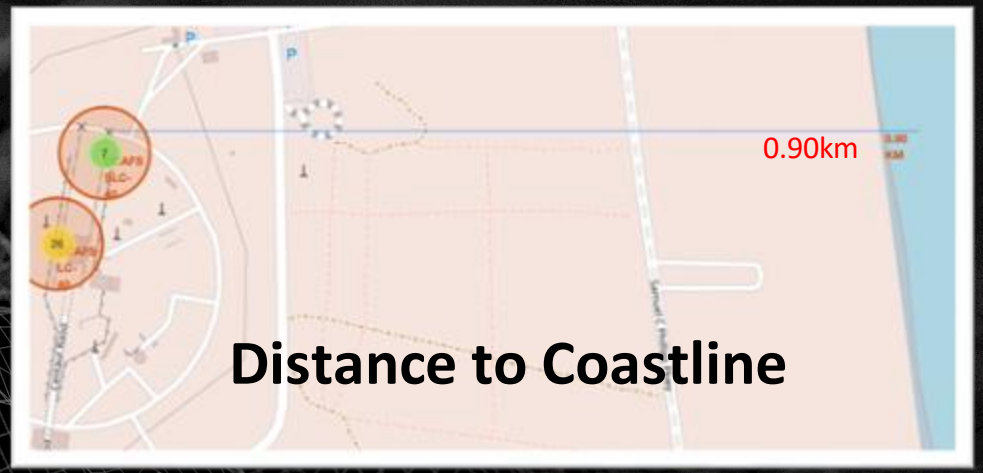
Distance to landmarks using CCAFS SLC-40 launch site as a reference



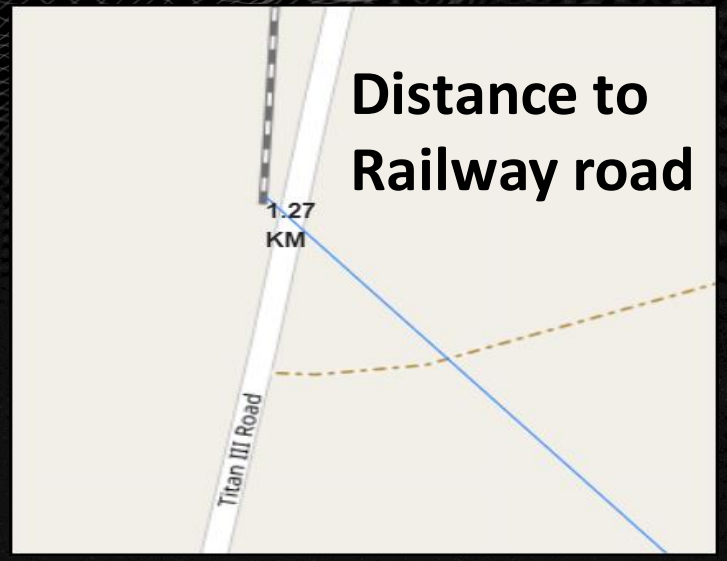
Distance to Highway



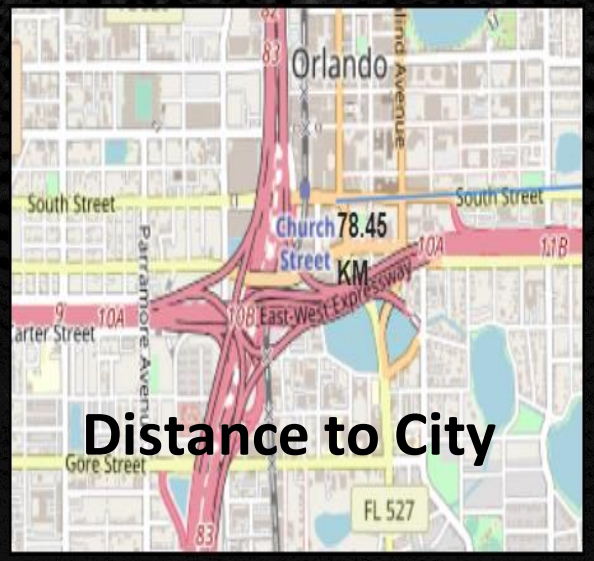
Distance to Railway Station



Distance to Coastline



Distance to Railway road

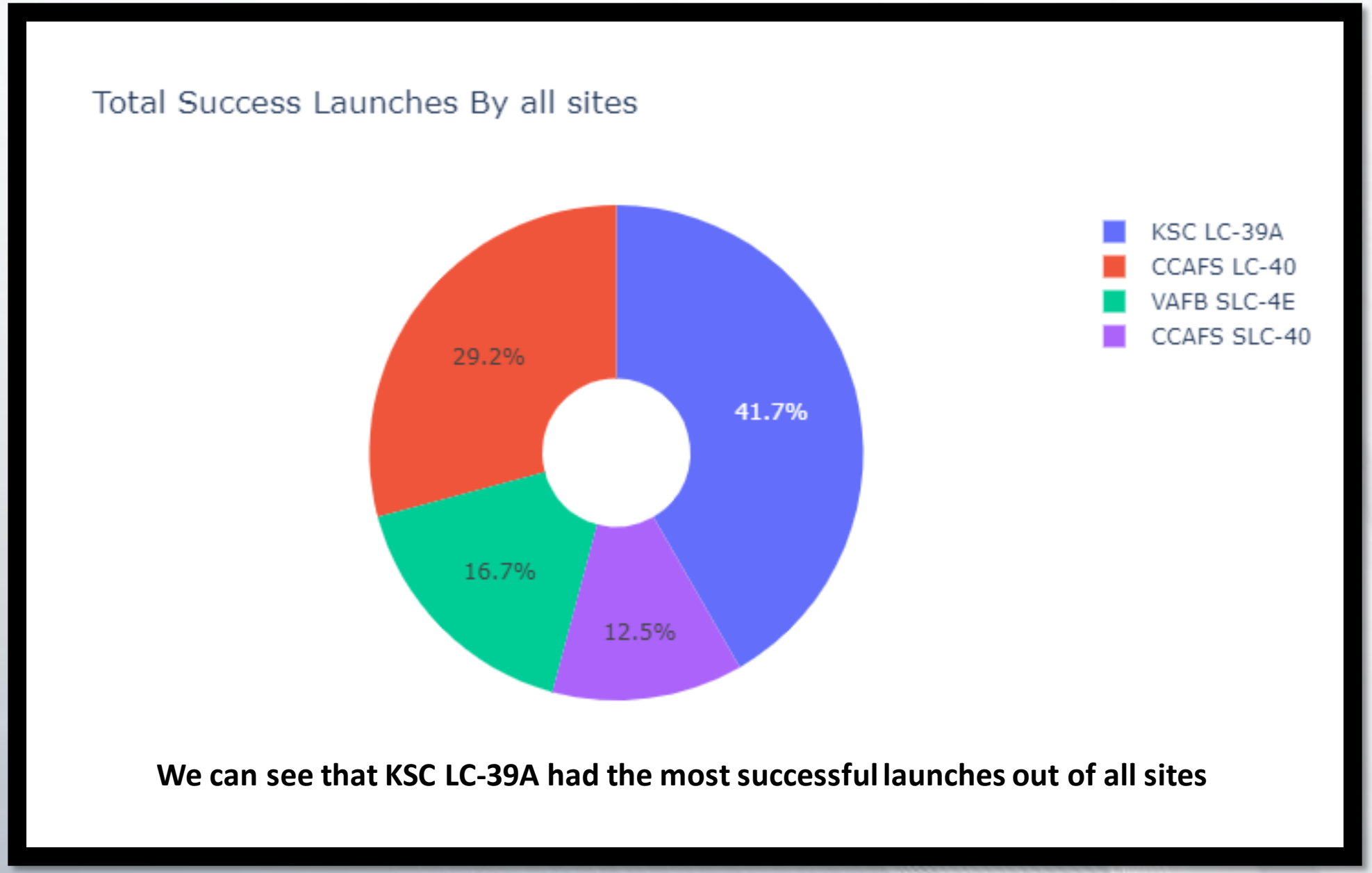


Distance to City

- Are launch sites in close proximity to railways? Yes (1.27 km)
- Are launch sites in close proximity to railway stations? No (78.58 km)
- Are launch sites in close proximity to highways? No (29.21 km)
- Are launch sites in close proximity to coastline? Yes (0.90 km)
- Do launch sites keep certain distance away from cities? Yes (78.45 km)

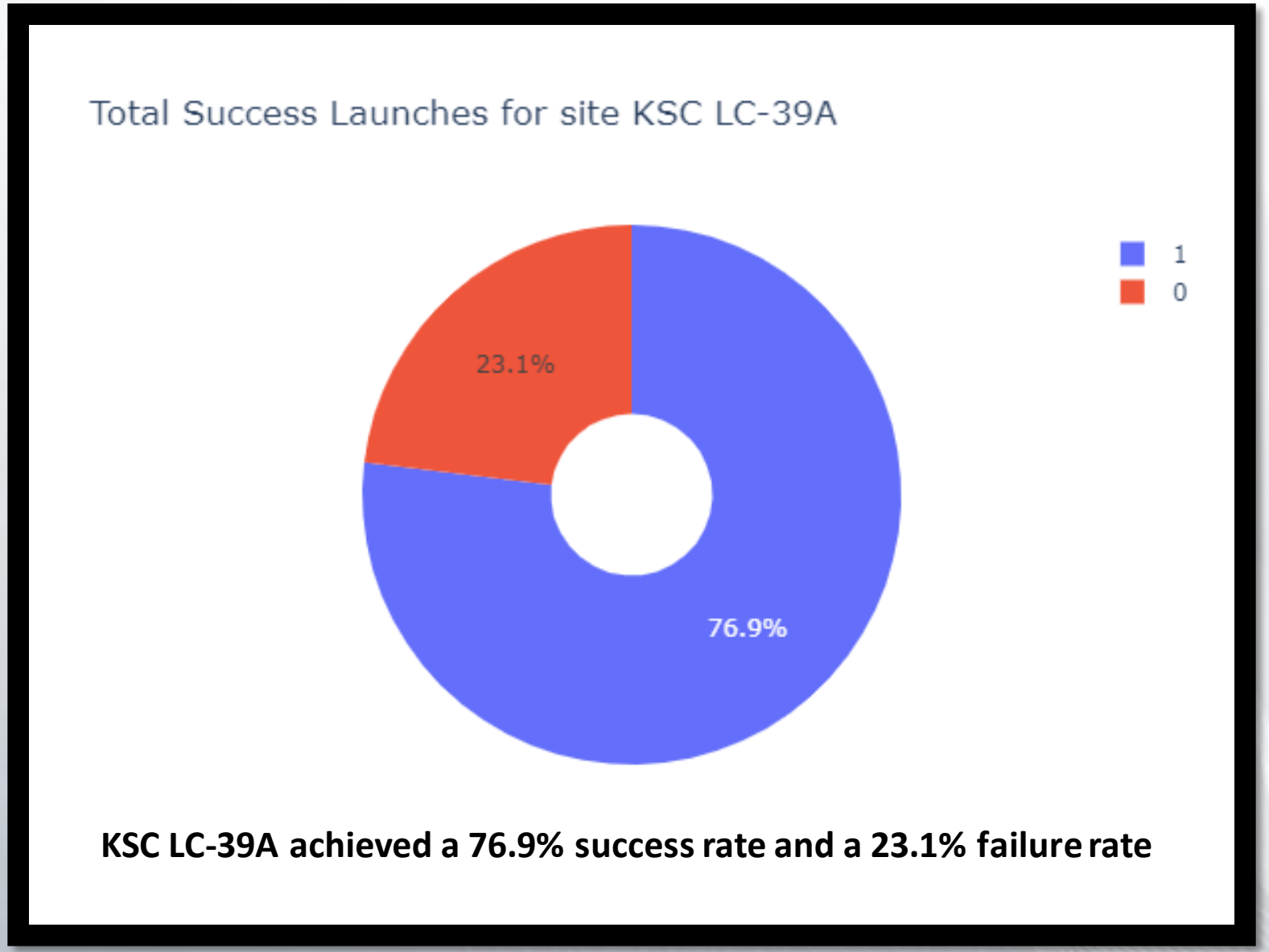

Dashboard with Plotly Dash

Piechart for the launch success count for all sites



We can see that KSC LC-39A had the most successful launches out of all sites

Piechart for the launch site with highest launch success ratio

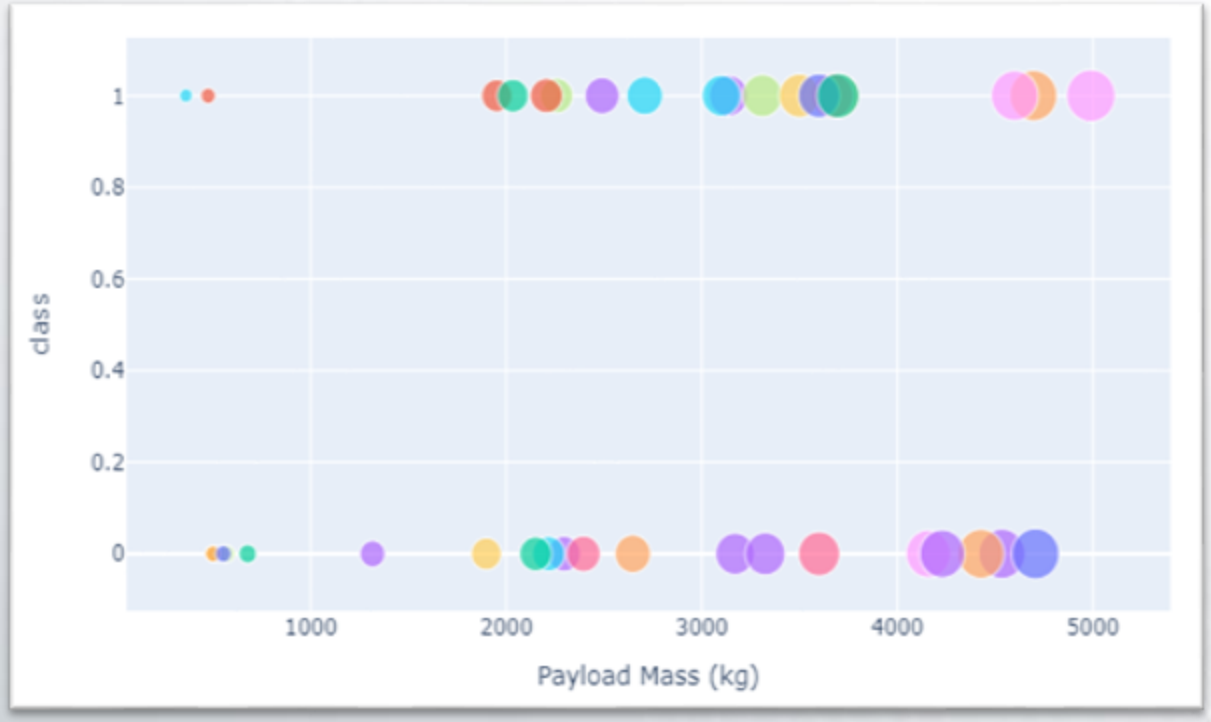


KSC LC-39A achieved a 76.9% success rate and a 23.1% failure rate

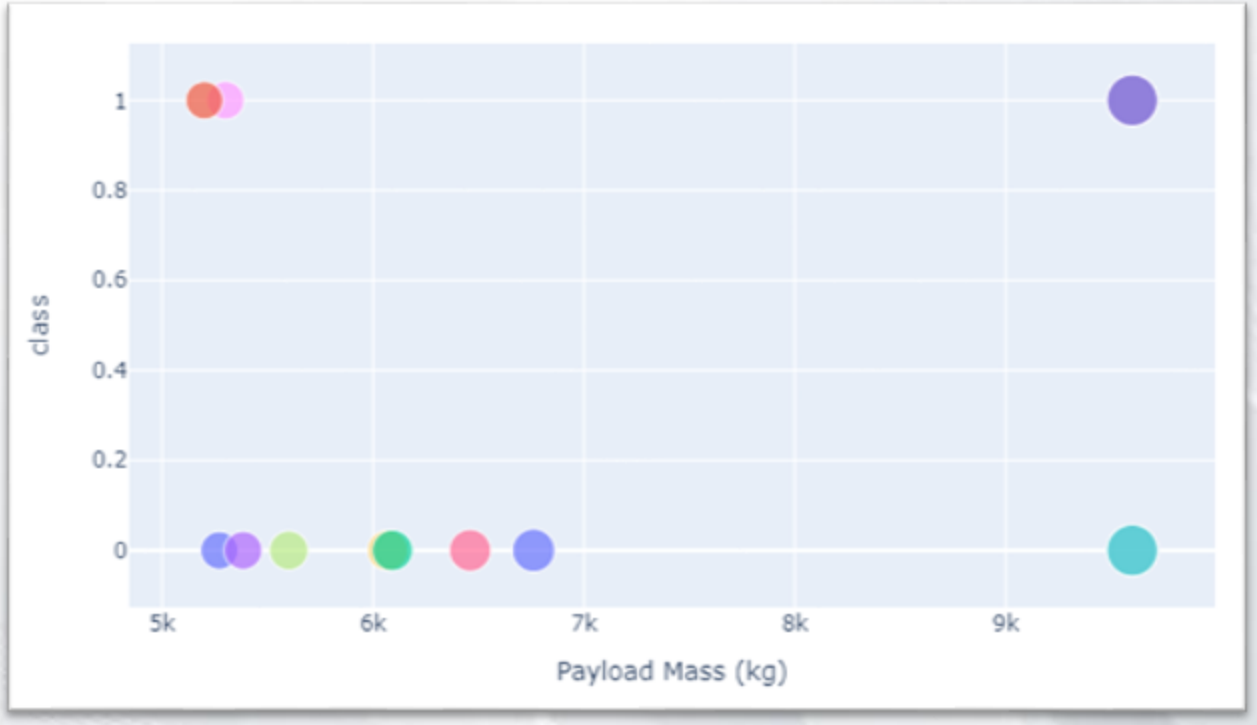
1 means Success
0 means Failure

Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider

Light-weighted payload mass (0-5000 kg)



Heavy-weighted payload mass (5000-10000 kg)



We can see that the Light-weighted payloads have a higher success rate than the heavy-weighted payloads


Predictive analysis (Classification)

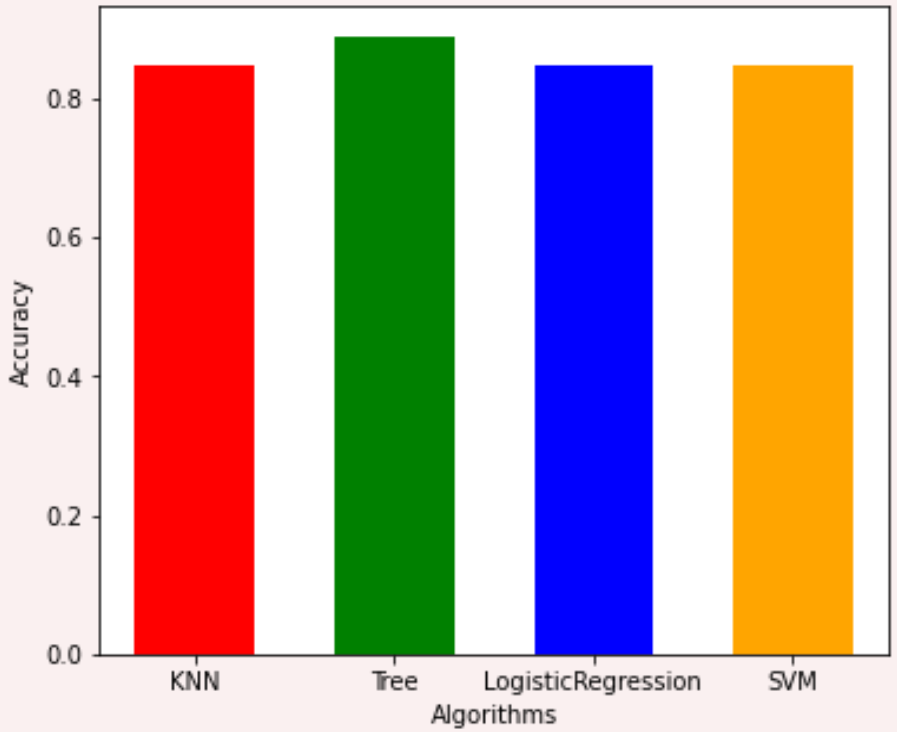
Classification Accuracy using training data

As we can see, the algorithms perform very closely. But we can deduce that the Decision Tree algorithm came out as the better classifier.

```
bestalgorithm = max(algorithms, key=algorithms.get)
```

	Algorithm	Accuracy
0	KNN	0.848214
1	Tree	0.889286
2	LogisticRegression	0.846429
3	SVM	0.848214

Bar graph showing accuracy for each algorithm

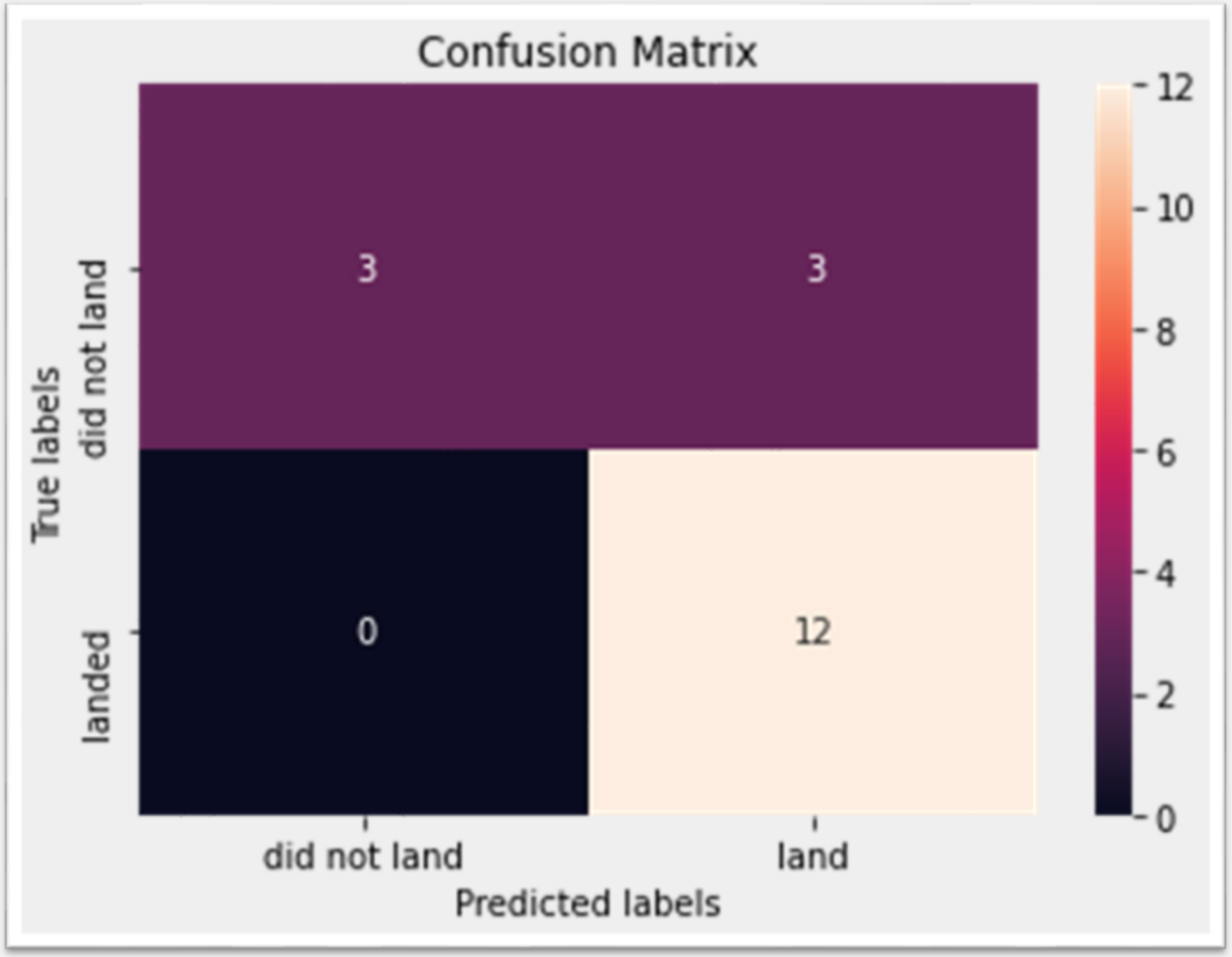



```
Best Algorithm is Tree with a score of 0.8892857142857142
Best Params is : {'criterion': 'gini', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'random'}
```

After selecting the proper hyperparameters for the decision tree classifier using the validation data, we managed to achieve a 88.9% accuracy.

Confusion Matrix

Examining the confusion matrix, we can see that Decision Tree can distinguish between the different classes. We see that the major problem is false negatives.



Conclusion

- The Tree Classifier Algorithm is the best for Machine Learning for this dataset
- Low weighted payloads perform better than the heavier payloads
- The success rates for SpaceX launches is directly proportional time in years they will eventually perfect the launches
- We can see that KSC LC-39A had the most successful launches from all the sites
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Thank you!