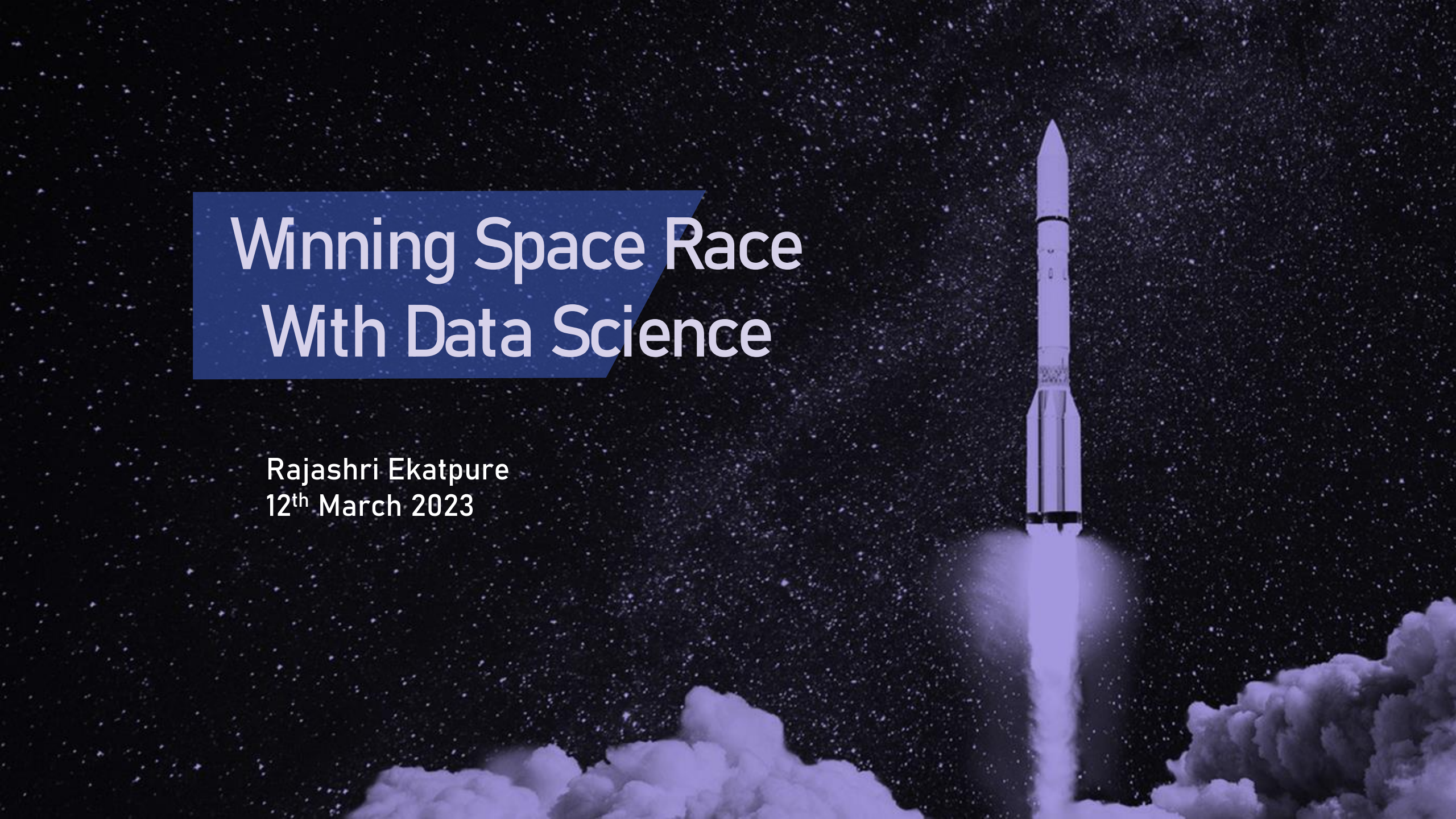


Winning Space Race With Data Science

Rajashri Ekatpure

12th March 2023

Outline

1. Executive Summary
2. Introduction
3. Methodology
4. Results
5. Conclusion
6. Appendix

Executive Summary

1. Summary of methodologies

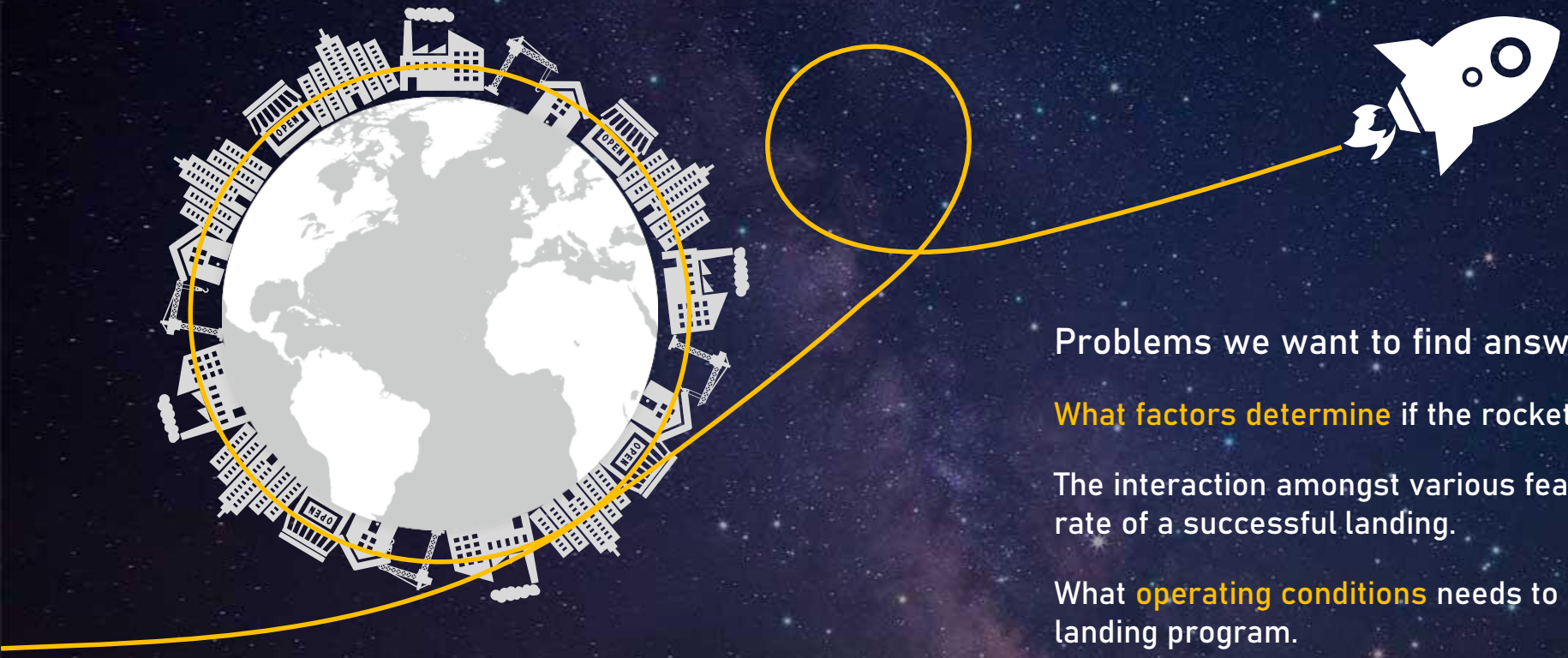
- Data Collection through API
- Data Collection with Web Scraping
- Data Wrangling
- Exploratory Data Analysis with SQL
- Exploratory Data Analysis with Data Visualization
- Interactive Visual Analytics with Folium
- Machine Learning Prediction

2. Summary of all Results

- Exploratory Data Analysis result
- Interactive analytics in screenshots
- Predictive Analytics result

Introduction Summary

Space X 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 Space X can reuse the first stage. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. This information can be used if an alternate company wants to bid against space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.



Problems we want to find answers

What factors determine if the rocket will land successfully?

The interaction amongst various features that determine the success rate of a successful landing.

What operating conditions needs to be in place to ensure a successful landing program.

Methodology

01 Data collection methodology

- SpaceX Rest API
- (Web Scrapping) from Wikipedia

02 Performed data wrangling (Transforming data for Machine Learning)

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

03 Performed exploratory data analysis (EDA) using visualization and SQL.

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

04 Performed interactive visual analytics.

- Using Folium and Plotly Dash

05 Performed predictive analysis using classification models.

- How to build, tune, evaluate classification models.

Data collection

Data collection – SpaceX API

1 .Getting Response from API

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

```
print(response.content)
```

```
b'[{"fairings":{"reused":false,"recovery_attempt":false,"rec
```

2. Converting Response to a .json

```
# Use json_normalize meethod to conv
json_data = response.json()
data = pd.json_normalize(json_data)
```

3. Apply custom functions to clean 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}
```

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

```
# save and export new dataset
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data collection – Web Scrapping

1 .Getting Response from HTML

```
# assigning the response to a object
response = requests.get(static_url)
response
```

2. Creating BeautifulSoup Object

```
soup = BeautifulSoup(response.text)
soup.title
```

```
<title>List of Falcon 9 and Falcon Heavy
```

3. Finding Tables

```
# Assign the result to a list
html_tables = soup.find_all('table')
```

4. Getting column names

```
column_names = []
th_elements = first_launch_table.find_all('th')
for th_element in first_launch_table.find_all('th'):
    column_name = extract_column_from_header(th_element)
    if column_name is not None and len(column_name) > 0:
        column_names.append(column_name)
```

5. Creation of Dictionary

```
launch_dict= dict.fromkeys(column_names)

del launch_dict['Date and time ( )']
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
```

6. Appending data to keys

```
extracted_row = 0
#Extract each table
for table_number,table in enumerate(soup.find_all('ta
   # get table row
    for rows in table.find_all("tr"):
        #check to see if first table heading is as nu
        if rows.th:
            if rows.th.string:
                flight_number=rows.th.string.strip()
                flag=flight_number.isdigit()
        else:
            flag=False
        #get table element
        row=rows.find_all('td')
        #if it is number save cells in a dictonary
        if flag:
            extracted_row += 1
            # Flight Number value
            # TODO: Append the flight_number into lau
            launch_dict['Flight No.'] = flight_number
```

7. Converting Dictionary to dataframe

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

8. DataFrame to CSV

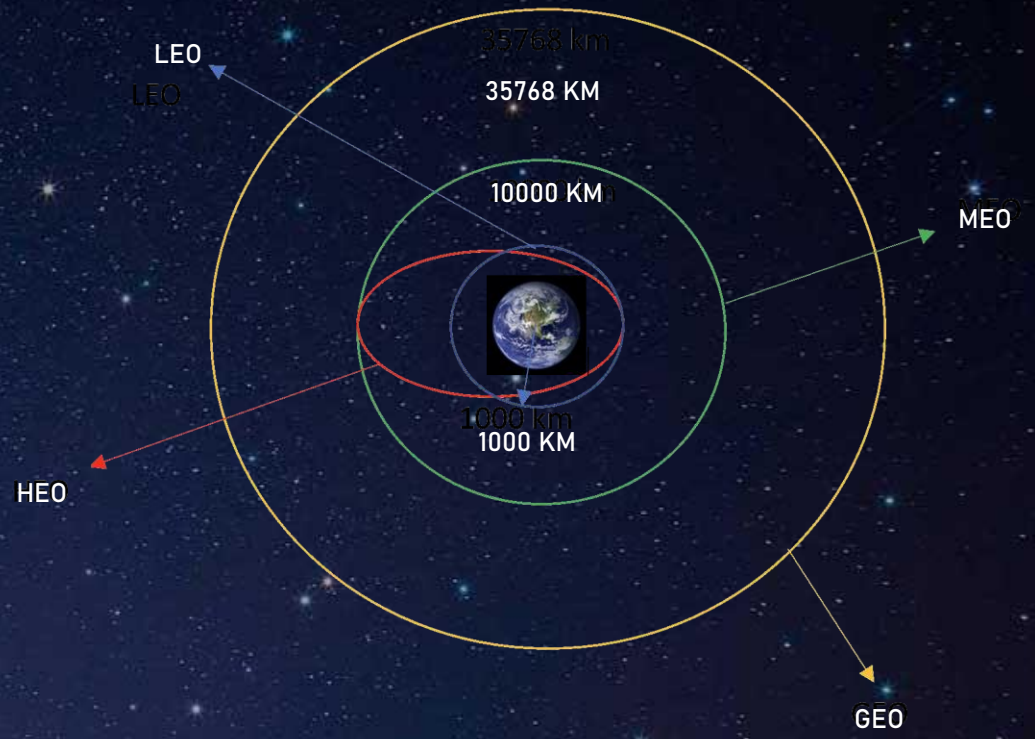
```
df=pd.DataFrame(launch_dict)
```

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.



Performance of 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 → Create a landing outcome label from Outcome column → Work out success rate for every landing in dataset → Export dataset as .CSV

EDA With Data Visualization

Scatter Graphs being drawn:

Flight Number VS. Payload Mass

Flight Number VS. Launch Site

Payload VS. Launch Site

Orbit VS. Flight Number

Payload VS. Orbit Type

Orbit VS. Payload Mass

Scatter plots show how much one variable is affected by another. The relationship between two variables is called their correlation . Scatter plots usually consist of a large body of data

Bar Graph being drawn:

Mean VS. Orbit

Bar Graph being drawn: Mean VS. Orbit Line Graph being drawn: Success Rate VS. Year Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

Line Graph being drawn:

Success Rate VS. Year

Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded

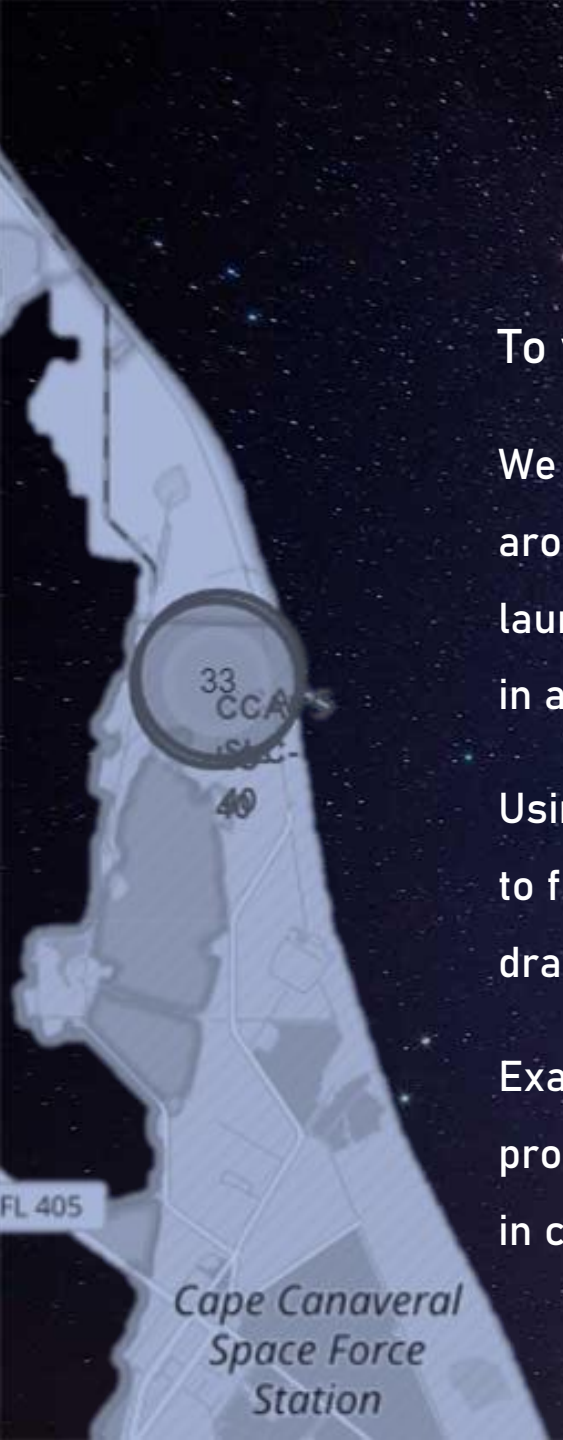
EDA With SQL

Performed SQL queries to gather information about the dataset.

For example of some questions we were asked about the data we needed information about. Which we are using SQL queries to get the answers in the dataset :

- Displaying the names of the unique launch sites in the space mission
- Displaying 5 records where launch sites begin with the string 'KSC'
- Displaying the total payload mass carried by boosters launched by NASA (CRS)
- Displaying average payload mass carried by booster version F9 v1.1
- Listing the date where the successful landing outcome in drone ship was achieved.
- Listing the names of the boosters which have success in ground pad 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 records which will display the month names, successful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017
- Ranking the count of successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order

Build an Interactive Map on 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? 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

Built an interactive dashboard with Flask and Dash

Used Python Anywhere to host the website live 24/7 so your can play around with the data and view the data

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.

Built an interactive dashboard with Flask and Dash

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 hyper parameters for each type of algorithms
- Plot Confusion Matrix IMPROVING MODEL
- Feature Engineering
- Algorithm Tuning

FINDING THE BEST PERFORMING CLASSIFICATION MODEL

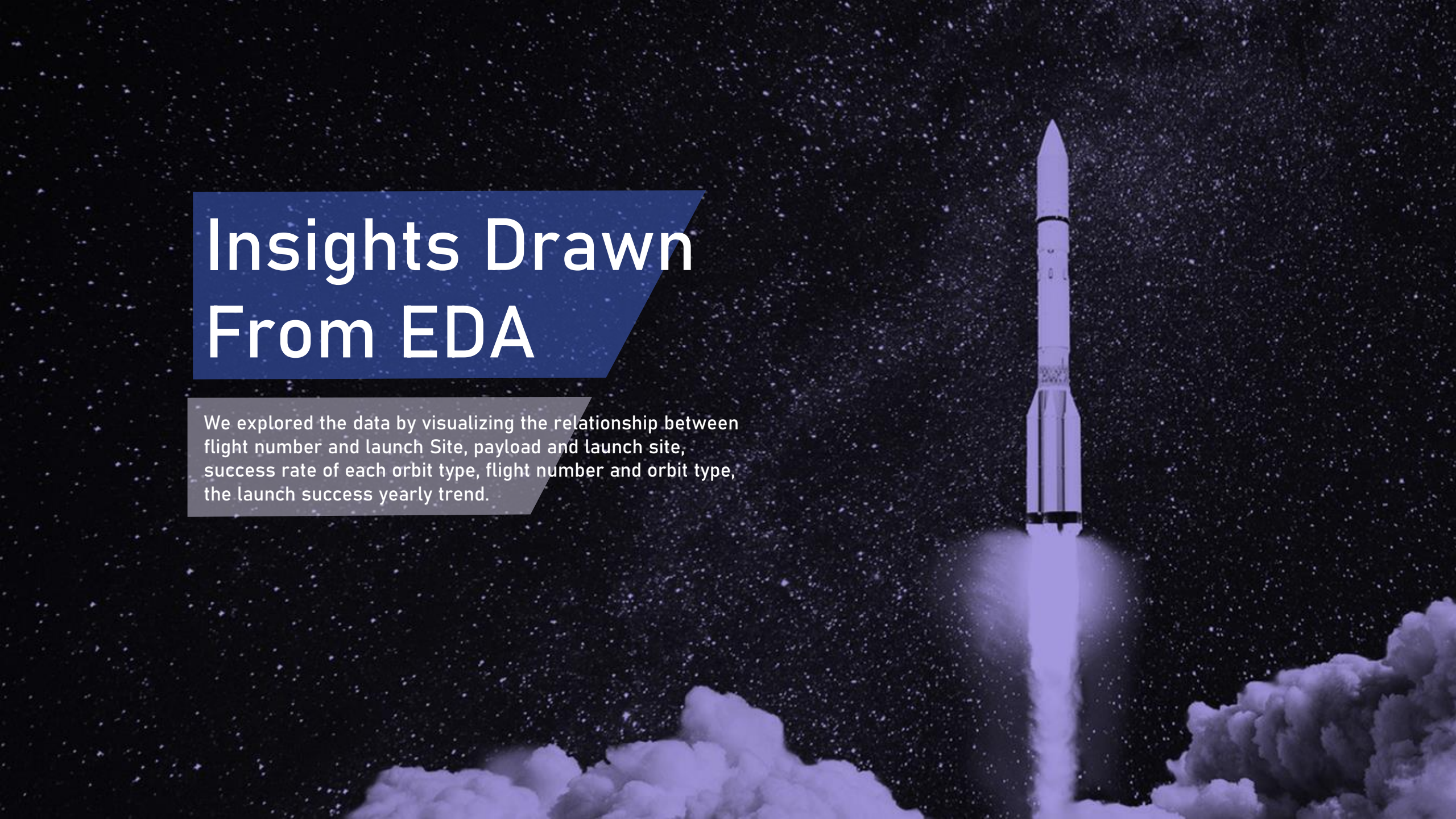
- The model with the best accuracy score wins the best performing model
- In the notebook there is a dictionary of algorithms with scores at the bottom of the notebook

Results

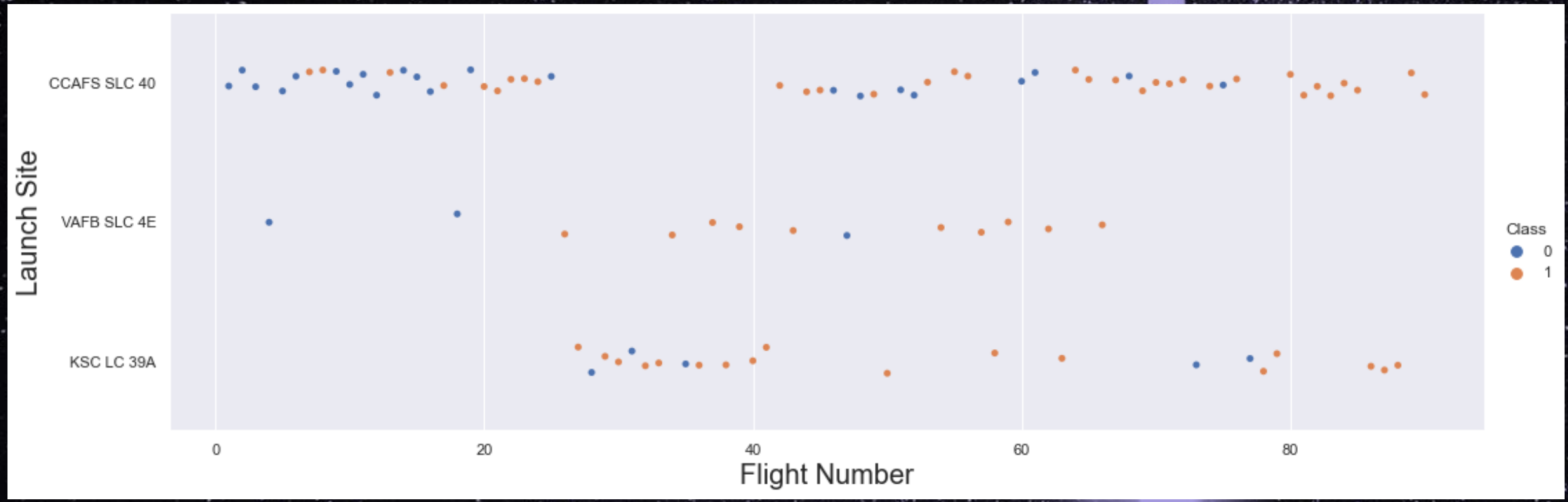
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Insights Drawn From EDA

We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.

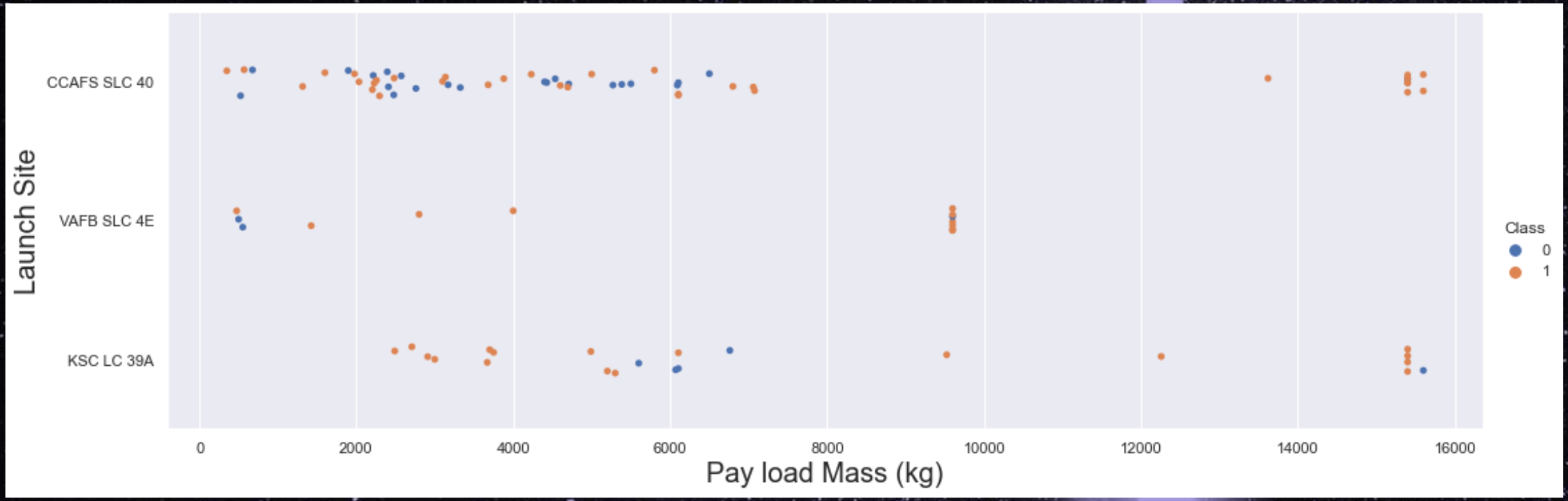


Flight Number Vs. Flight Site



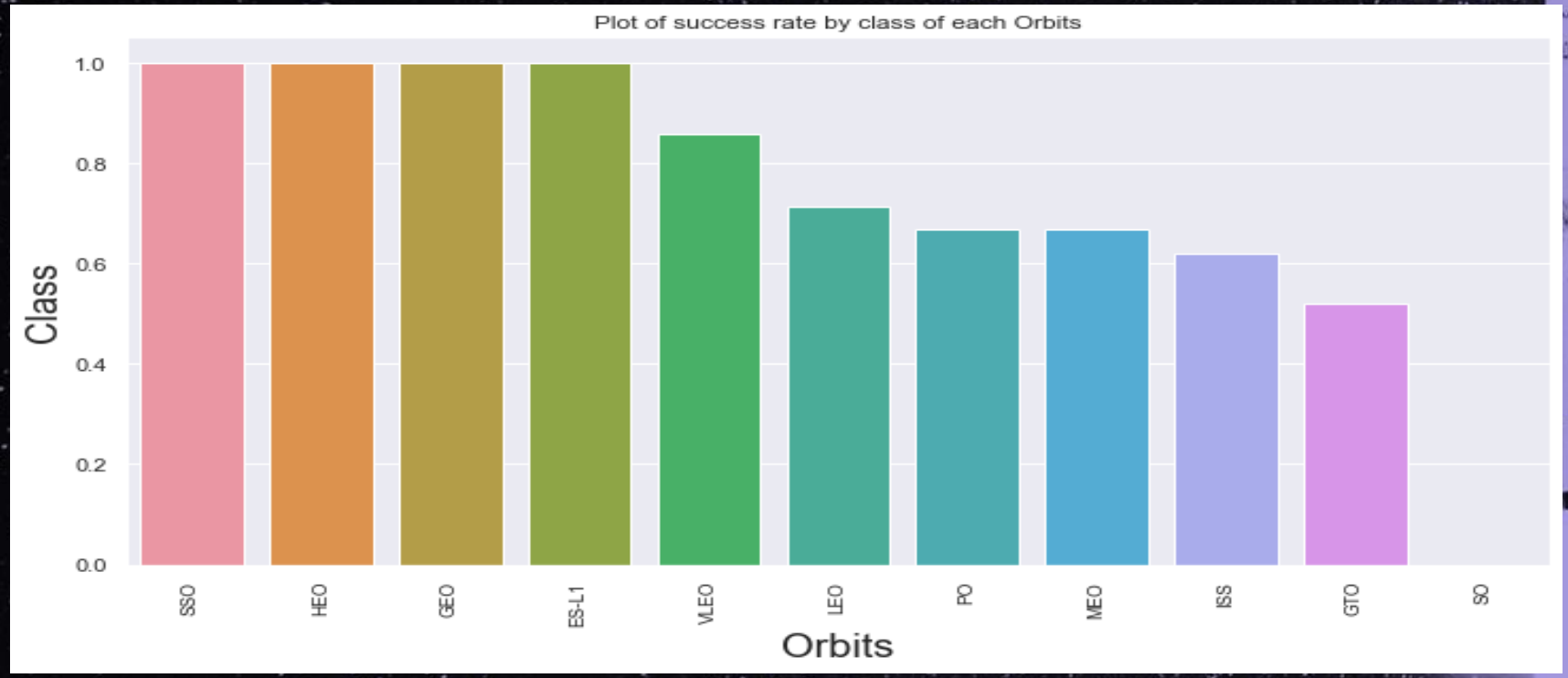
The more amount of flights at a launch site the greater the success rate at a launch site

Payload Mass vs. Launch Site



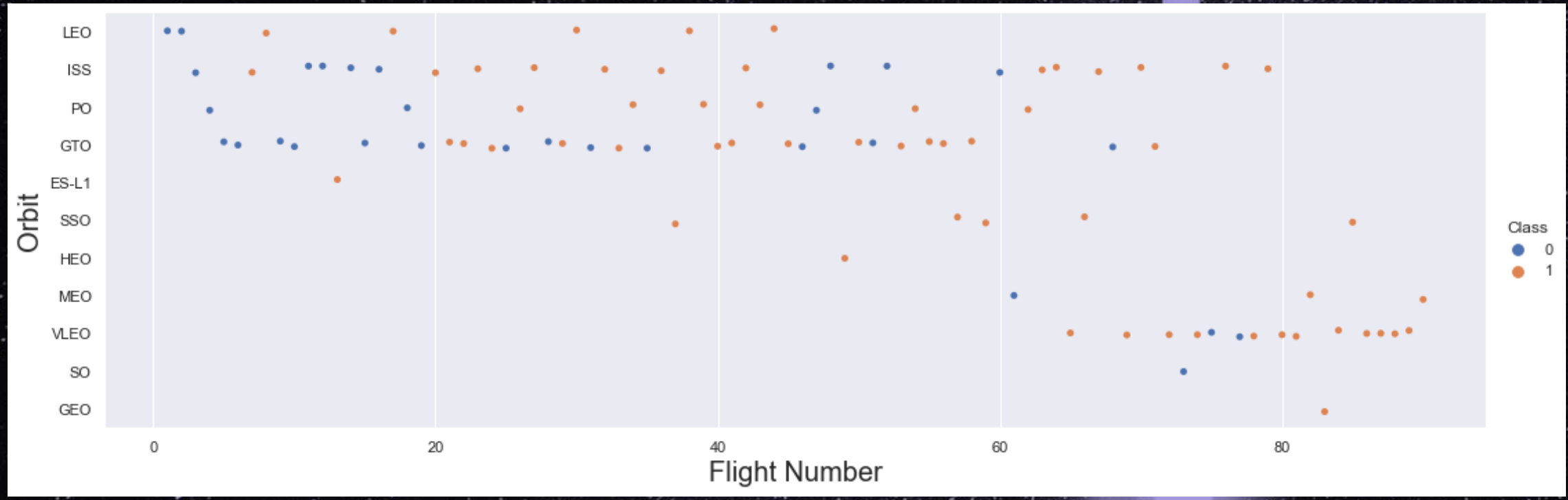
The greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. There is not quite a clear pattern to be found using this visualization to make a decision if the Launch Site is dependant on Pay Load Mass for a success launch

Success rate vs. Orbit type



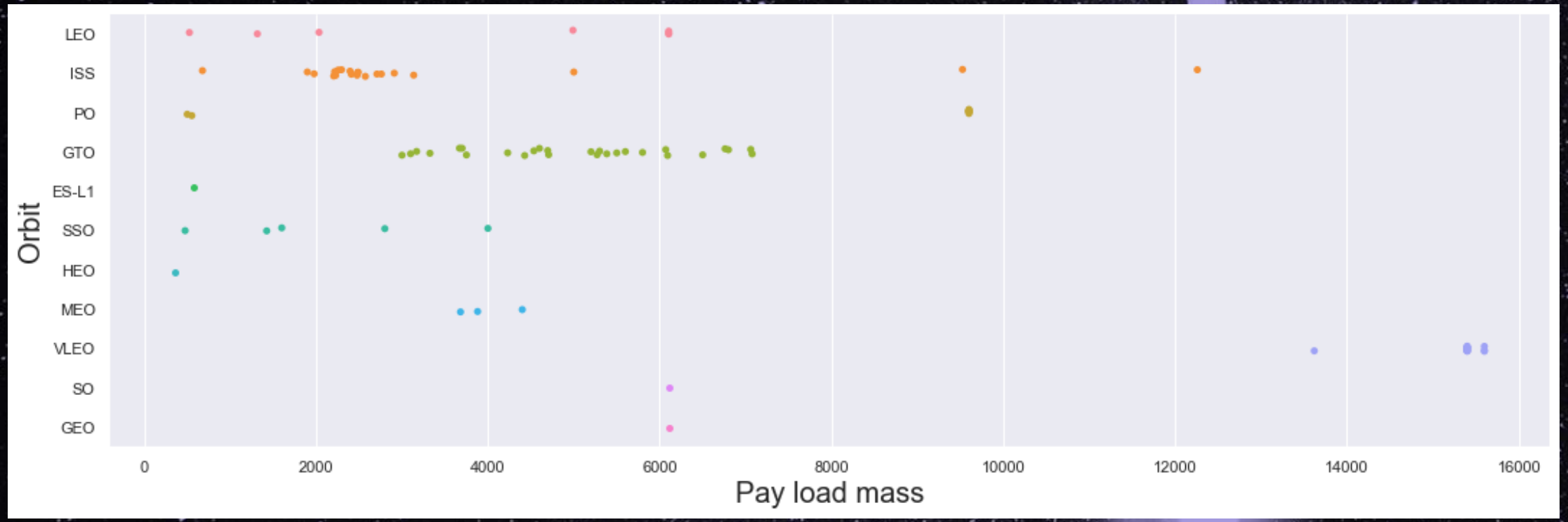
Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate

Flight Number vs. Orbit type



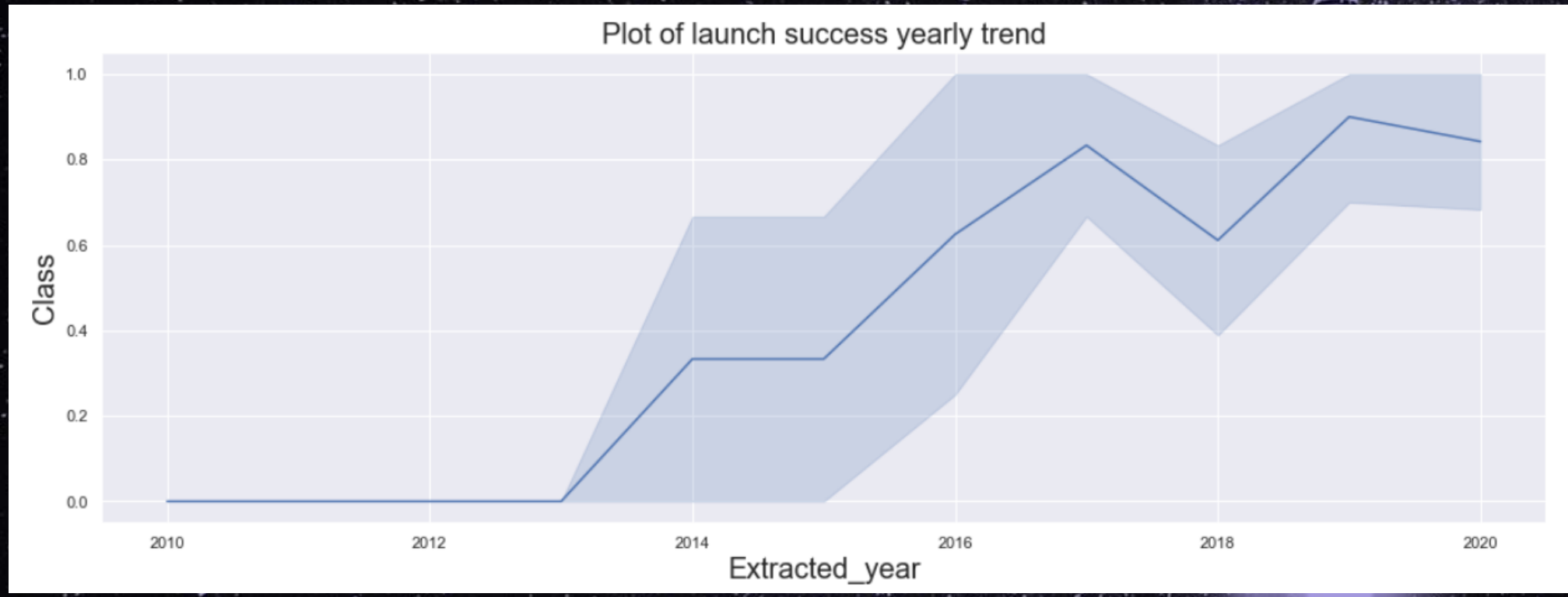
You should see that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit

Payload vs. Orbit type



You should observe that Heavy payloads have a negative influence on GTO orbits and positive on GTO and Polar LEO (ISS) orbits.

Launch success yearly trend



We can observe that the success rate since 2013 kept increasing till 2020

Section 2

EDA With SQL

We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.

Unique Launch Sites

```
%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

Here we performed SQL query to find unique launch site from the SpaceX data, that will help us for further analysis.

Unique Launch Sites

```
%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

Using the word TOP 5 in the query means that it will only show 5 records from tblSpaceX and LIKE keyword has a wild card with the words ‘KSC%’ the percentage in the end suggests that the Launch_Site name must start with KSC

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

Using the word TOP 5 in the query means that it will only show 5 records from tblSpaceX and LIKE keyword has a wild card with the words 'KSC%' the percentage in the end suggests that the Launch_Site name must start with KSC

Total Payload Mass by Customer NASA (CRS)

```
%sql SELECT SUM("PAYLOAD_MASS__KG_") from SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

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

SUM(PAYLOAD_MASS__KG_)
45596

Using the function SUM summates the total in the column PAYLOAD_MASS_KG_ The WHERE clause filters the dataset to only perform calculations on Customer NASA (CRS)

Average Payload Mass carried by booster version F9 v1.1

```
%sql SELECT AVG("PAYLOAD_MASS__KG_") from SPACEXTBL WHERE Booster_Version = 'F9 v1.1';
```

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

AVG(PAYLOAD_MASS__KG_)
2928.4

Using the function SUM summates the total in the column PAYLOAD_MASS_KG_ The WHERE clause filters the dataset to only perform calculations on Customer NASA (CRS)

The date where the successful landing outcome on ground pad was achieved

```
%sql SELECT Date from SPACEXTBL WHERE "Landing _Outcome" ='Success (ground pad)' limit 1;
```

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

Date
22-12-2015

Using the function MIN works out the minimum date in the column Date The WHERE clause filters the dataset to only perform calculations on Landing_Outcome Success (Ground pad)

Successful drone ship landing with payload between 4000 and 6000

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

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

Booster_Version
F9 FT B1021.1
F9 FT B1022
F9 FT B1023.1
F9 FT B1026
F9 FT B1029.1
F9 FT B1021.2
F9 FT B1029.2
F9 FT B1036.1
F9 FT B1038.1
F9 B4 B1041.1
F9 FT B1031.2
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

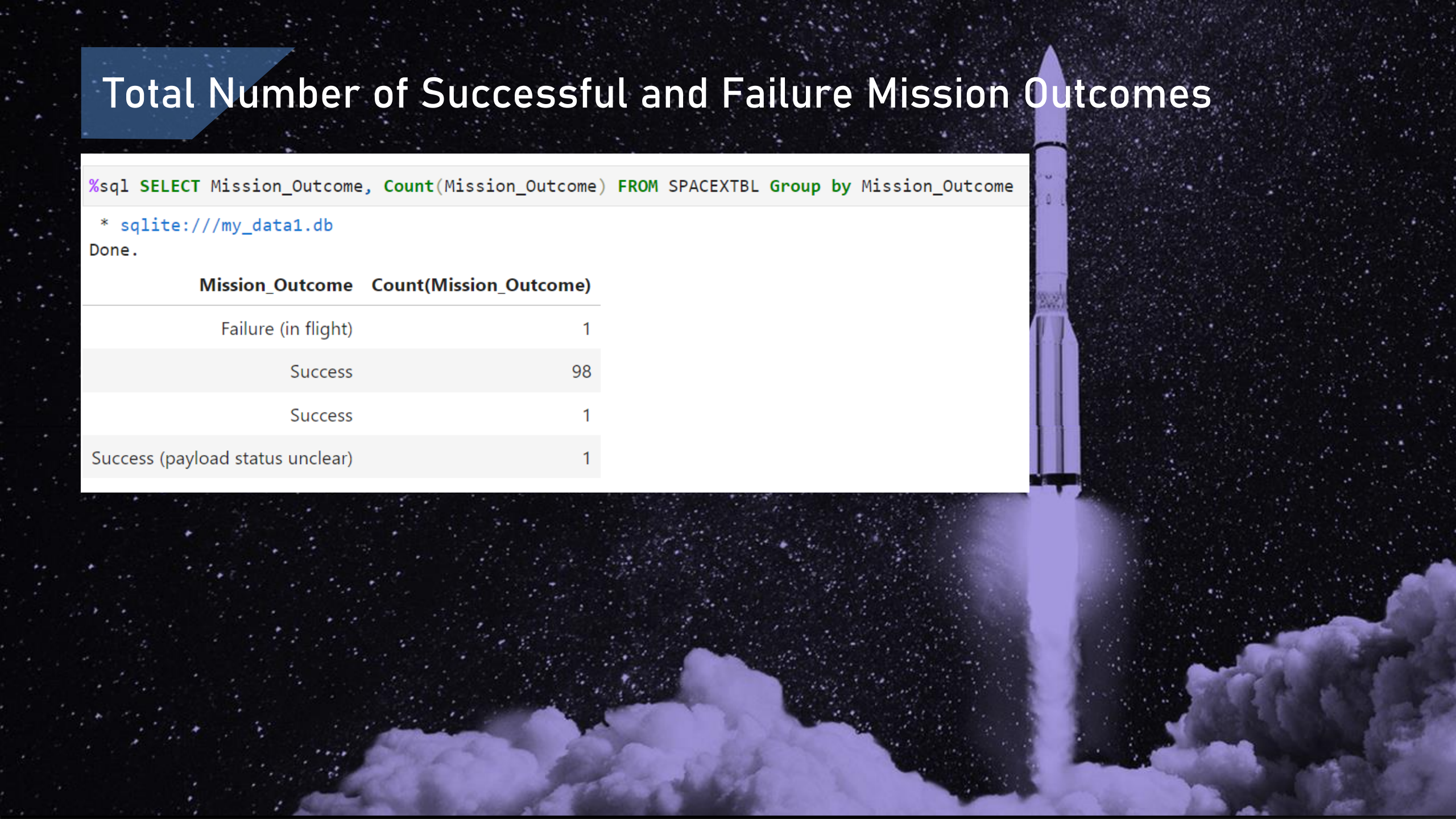
Selecting only Booster_Version The WHERE clause filters the dataset to Landing_Outcome = Success (drone ship) The AND clause specifies additional filter conditions Payload_MASS_KG_ > 4000 AND Payload_MASS_KG_ < 6000

Total Number of Successful and Failure Mission Outcomes

```
%sql SELECT Mission_Outcome, Count(Mission_Outcome) FROM SPACEXTBL Group by Mission_Outcome
```

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

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

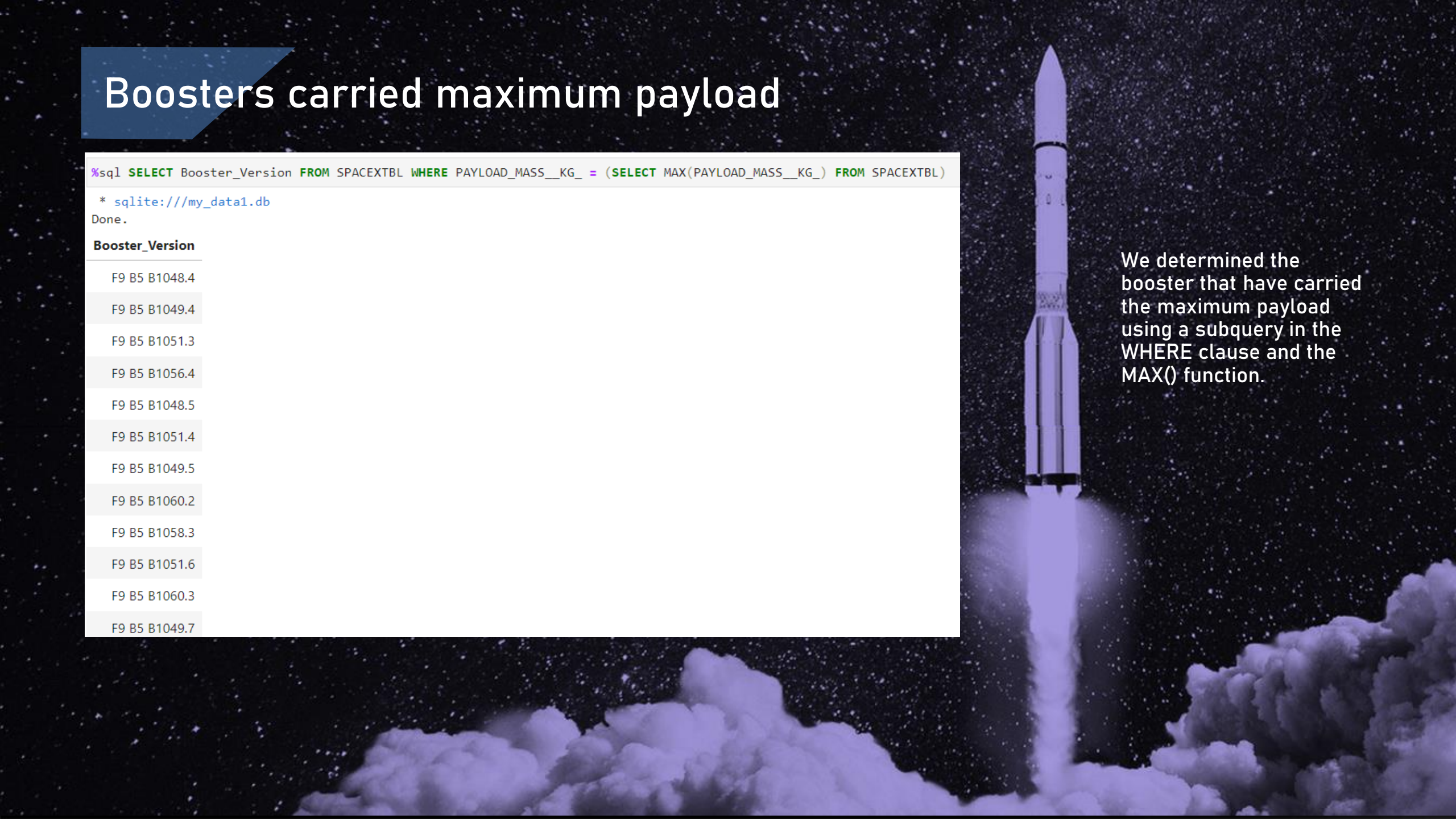


Boosters carried maximum payload

```
%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
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7



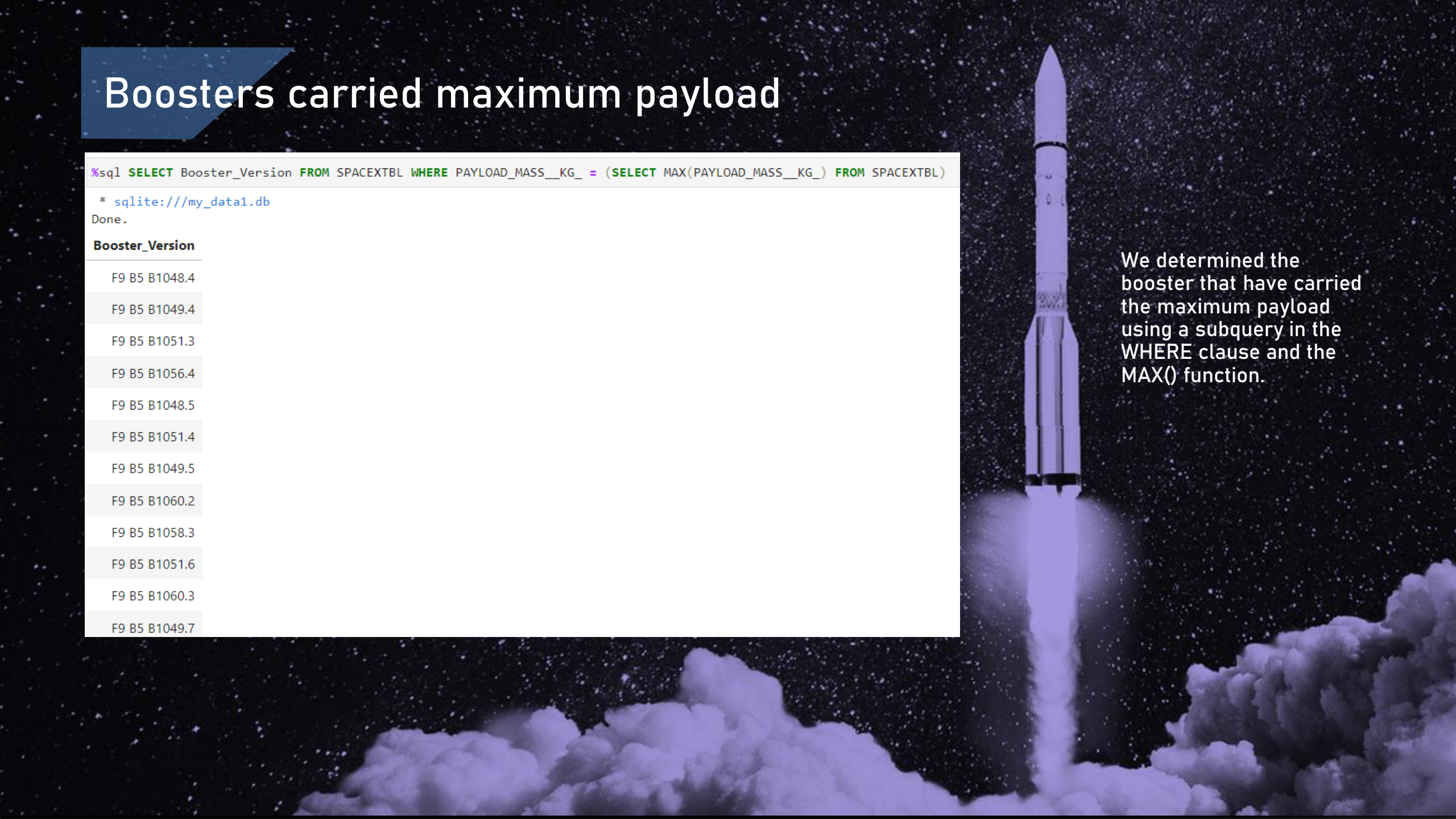
We determined the booster that have carried the maximum payload using a subquery in the WHERE clause and the MAX() function.

Boosters carried maximum payload

```
%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
F9 B5 B1058.3
F9 B5 B1051.6
F9 B5 B1060.3
F9 B5 B1049.7



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

The date where the successful landing outcome on ground pad was achieved

```
%sql SELECT CASE substr(Date, 4, 2) WHEN '01' THEN 'January' WHEN '02' THEN 'February' WHEN '03' THEN 'March' WHEN '04' THEN 'April' WHEN '05'
THEN 'May' WHEN '06' THEN 'June' WHEN '07' THEN 'July' WHEN '08' THEN 'August' WHEN '09' THEN 'September' WHEN '10' THEN 'October' WHEN '11' T
HEN 'November' WHEN '12' THEN 'December' END AS month_name, "Landing _Outcome", Booster_Version, Launch_Site from SPACEXTBL
WHERE substr(Date,7,4) = '2015' AND "Landing _Outcome" ='Failure (drone ship)';
```

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

month_name	Landing _Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

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

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 num_successfull_landing from SPACEXTBL WHERE Date> '04-06-2010' and Date <'20-03-2017'
Group by "Landing _Outcome" order by "num_successfull_landing" DESC;
```

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

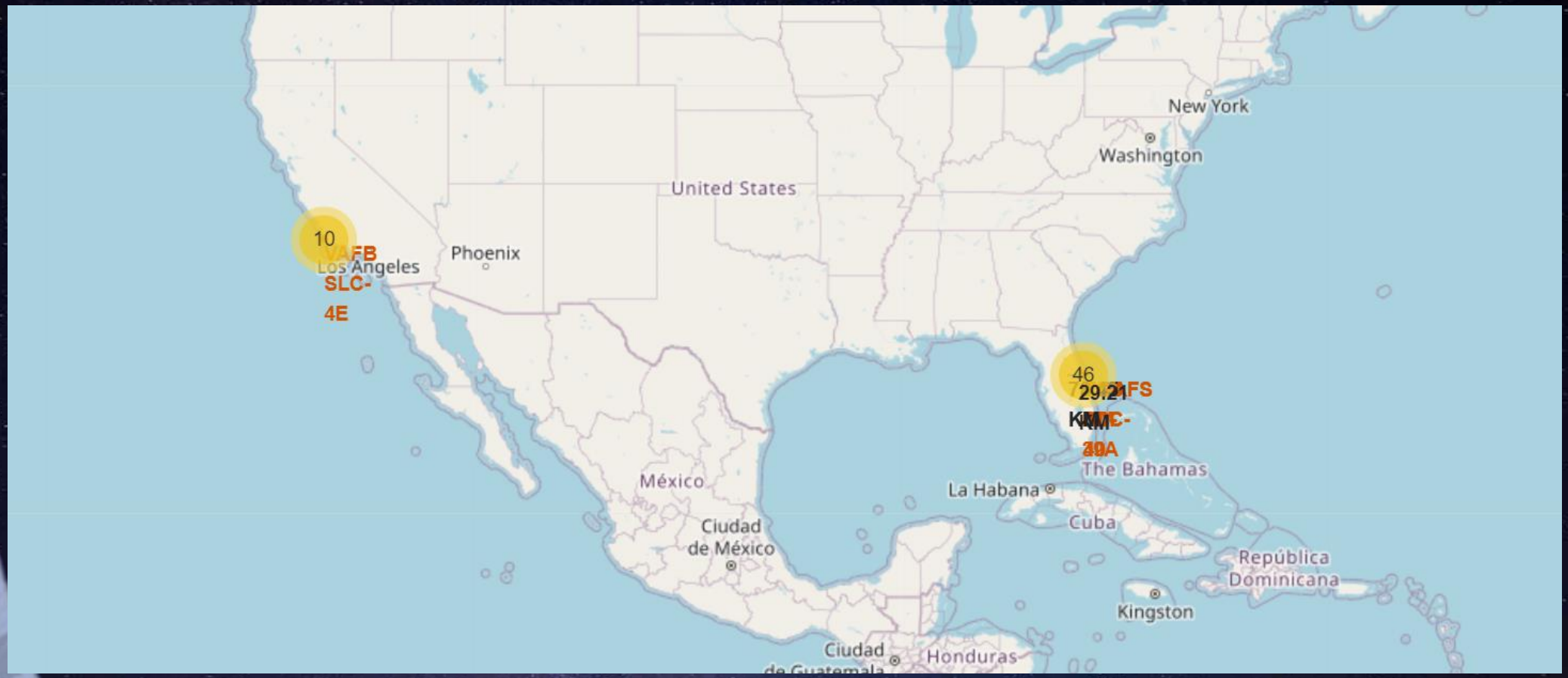
Landing _Outcome	num_successfull_landing
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	6
Failure (drone ship)	4
Failure	3
Controlled (ocean)	3
No attempt	1
Failure (parachute)	1

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.

Interactive map with Folium

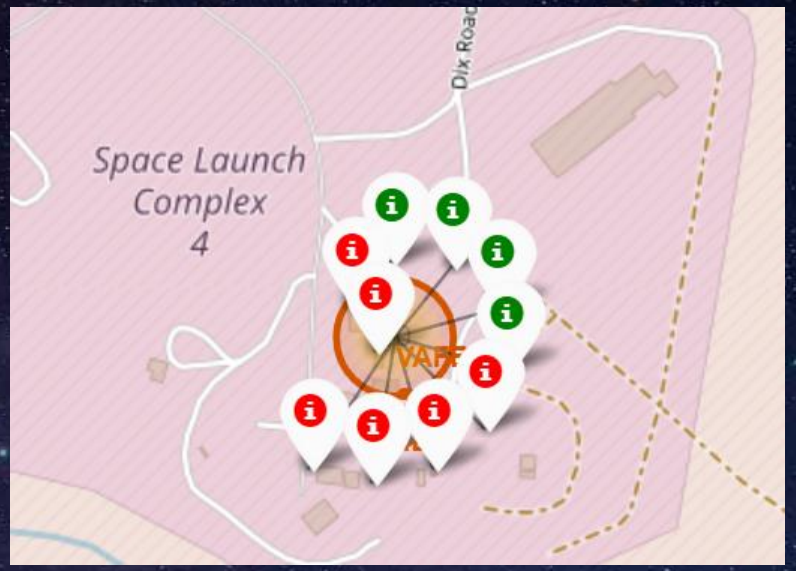
All launch sites global map markers



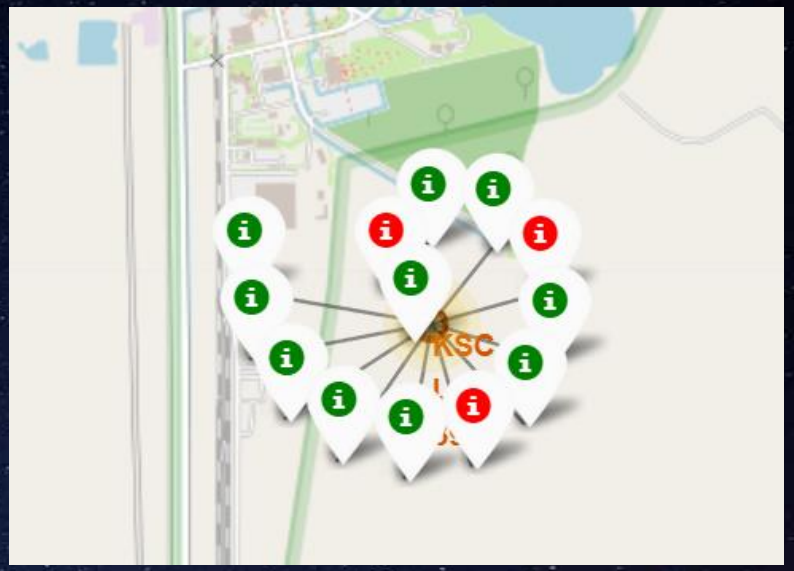
We can see that the SpaceX launch sites are in the United States of America coasts. Florida and California

Markers showing launch sites with color labels

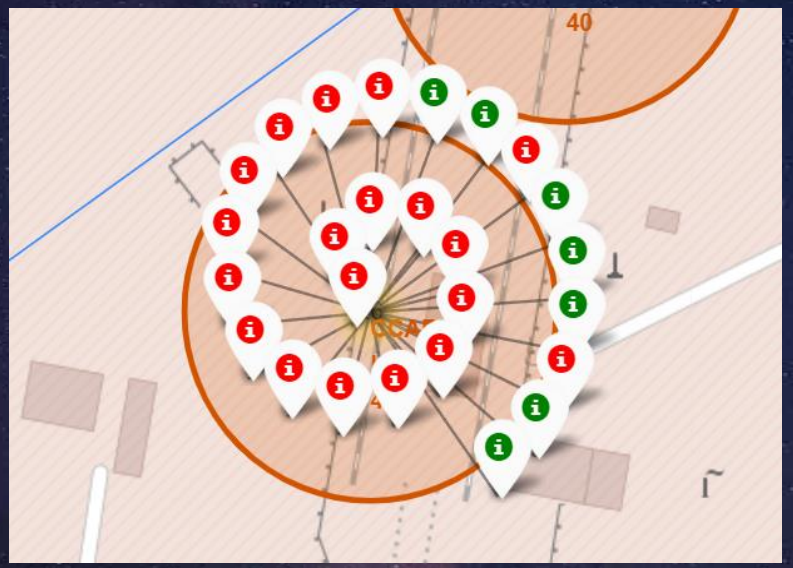
Florida Launch Sites
Green Marker shows successful Launches and
Red Marker shows Failures



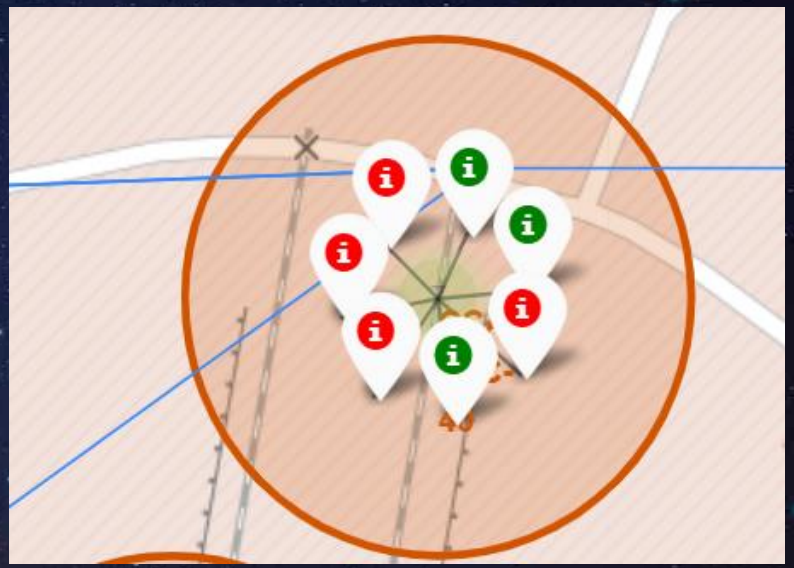
VAFB SLC-4E



KSC LC-39A



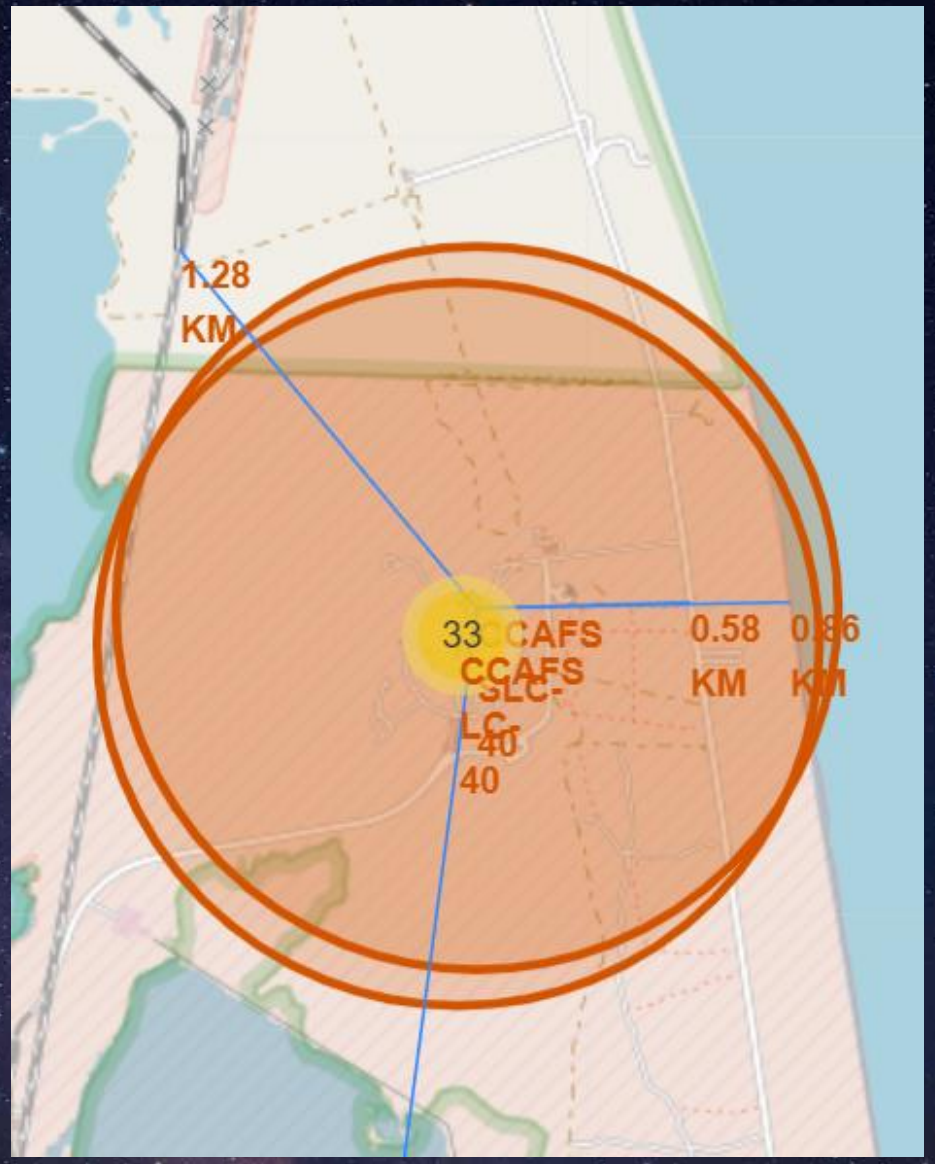
CCAFS LC-40



CCAFS SLC-40

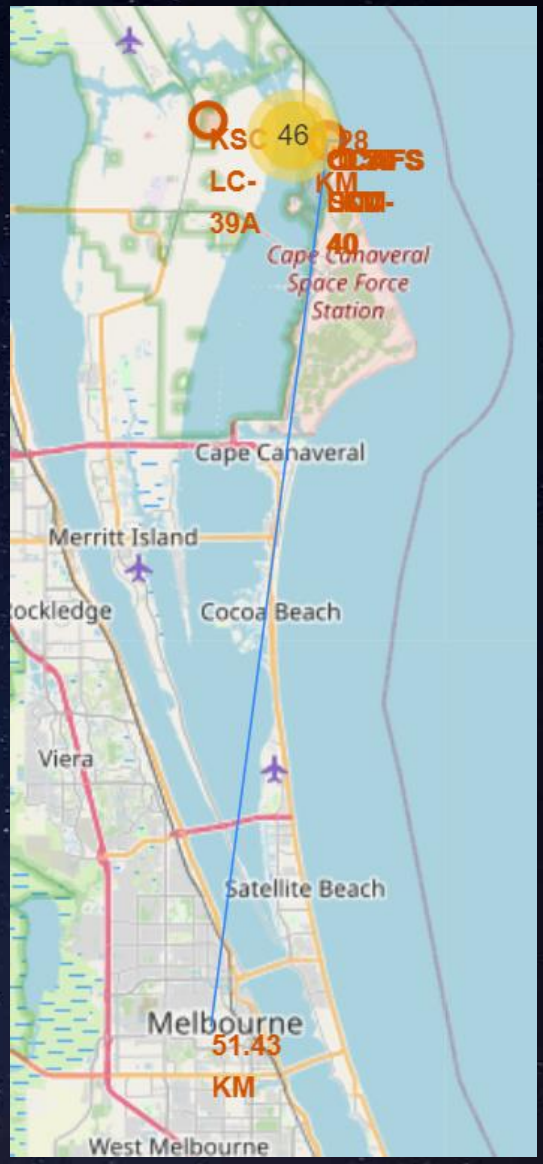
Launch Sites distance to landmarks to find trends with Haversine formula using CCAFS-SLC-40 as a reference

- Launch sites are in close proximity to equator to minimize fuel consumption by using Earth's ~ 30km/sec eastward spin to help spaceships get into orbit.
- Launch sites are in close proximity to coastline so they can fly over the ocean during launch, for at least two safety reasons—
- (1) crew has option to abort launch and attempt water landing (
- 2) minimize people and property at risk fro m falling debris.
- Launch sites are in close proximity to highways, which allows for easily transport required people and property.
- Launch sites are in close proximity to railways, which allows transport for heavy cargo.
- Launch sites are not in close proximity to cities, which minimizes danger to population dense areas.



Railway Distance

Costal Distance

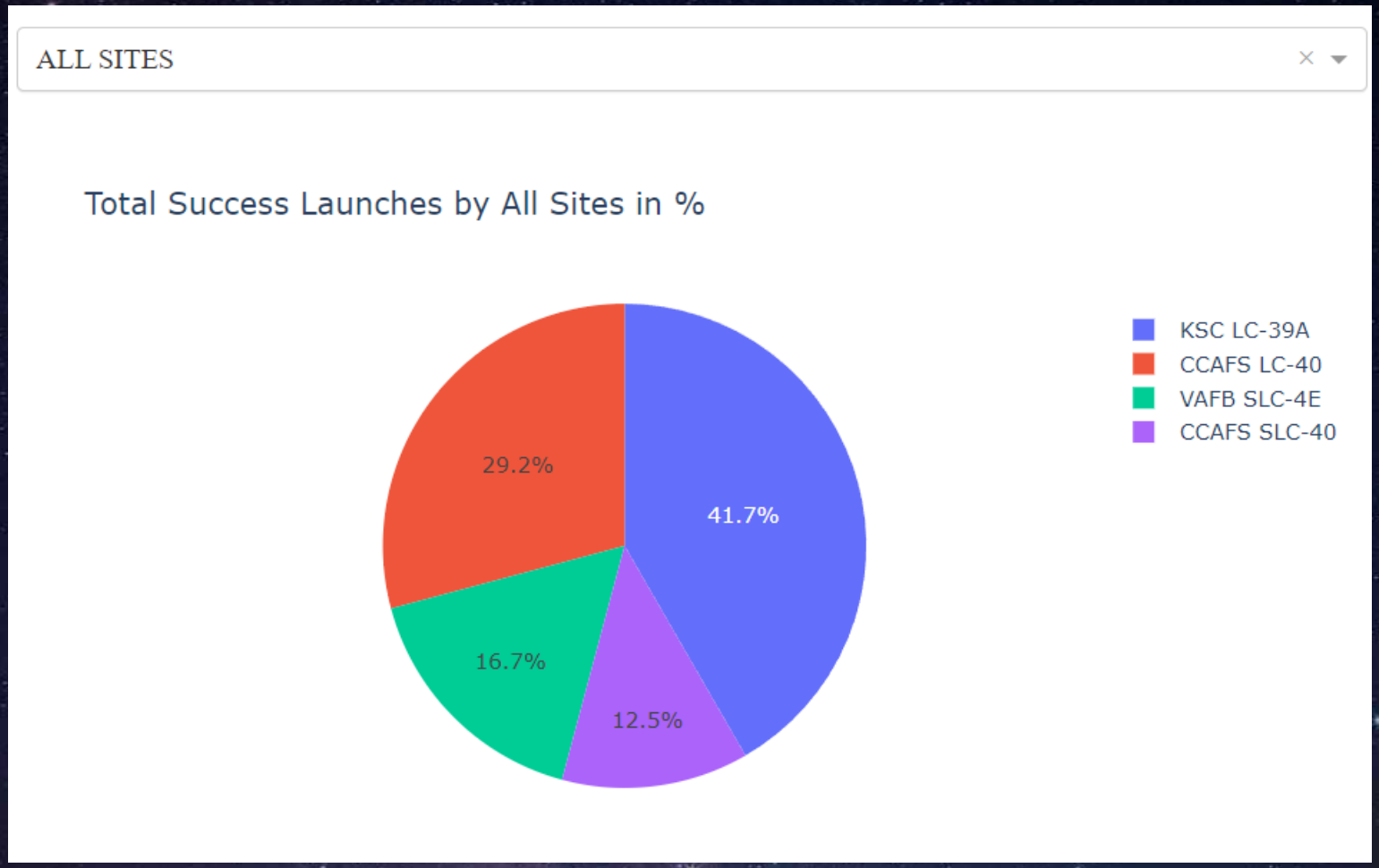


Nearest City Distance

plotly

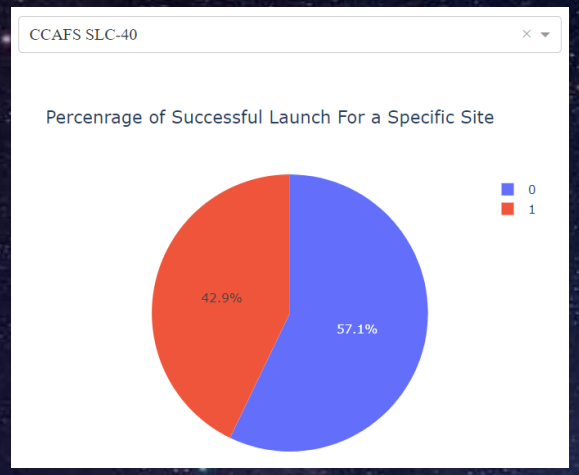
Dashboard with Plotly Dash

All launch sites global map markers

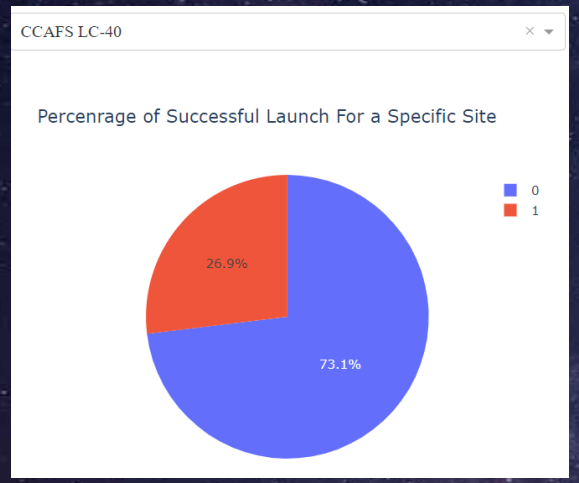


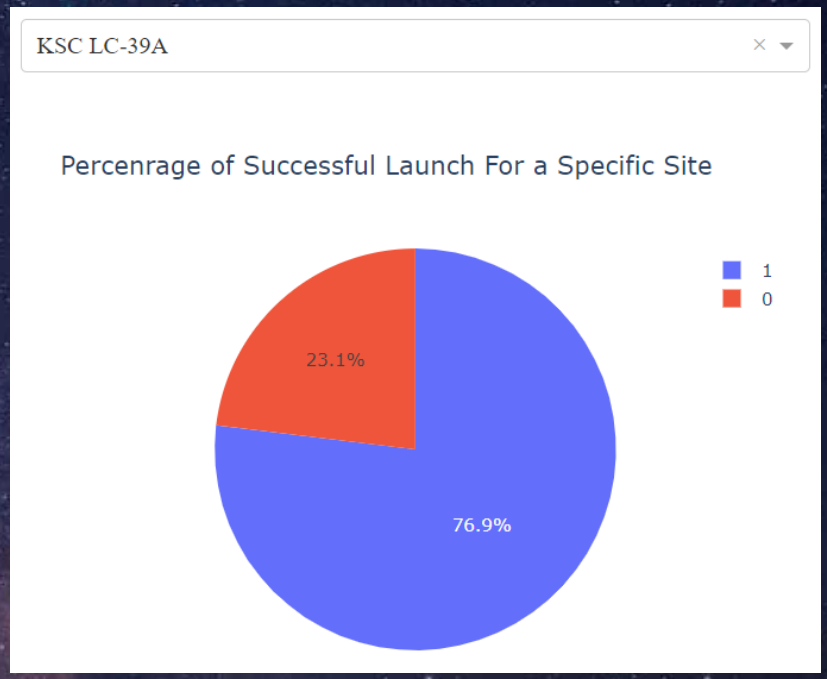
We can see that KSC LC-39A had the most successful launches from all the sites

DASHBOARD – Pie chart for the launch site with highest launch success ratio

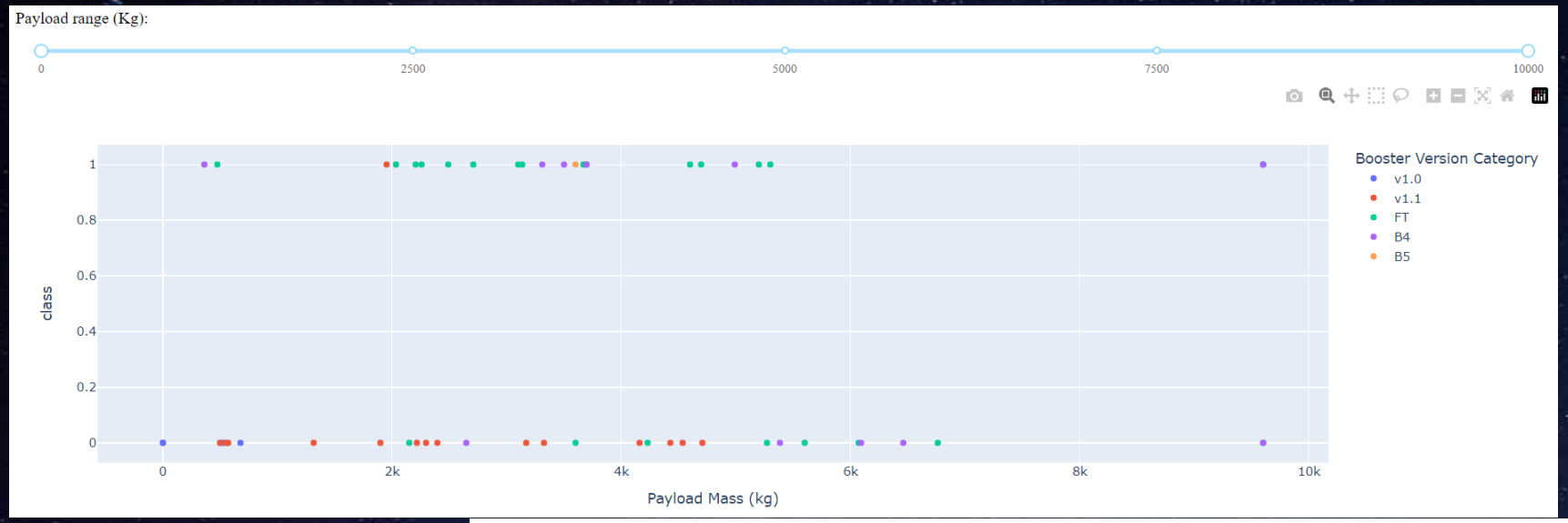


We can see that KSC LC-39A had the most successful launches from all the sites



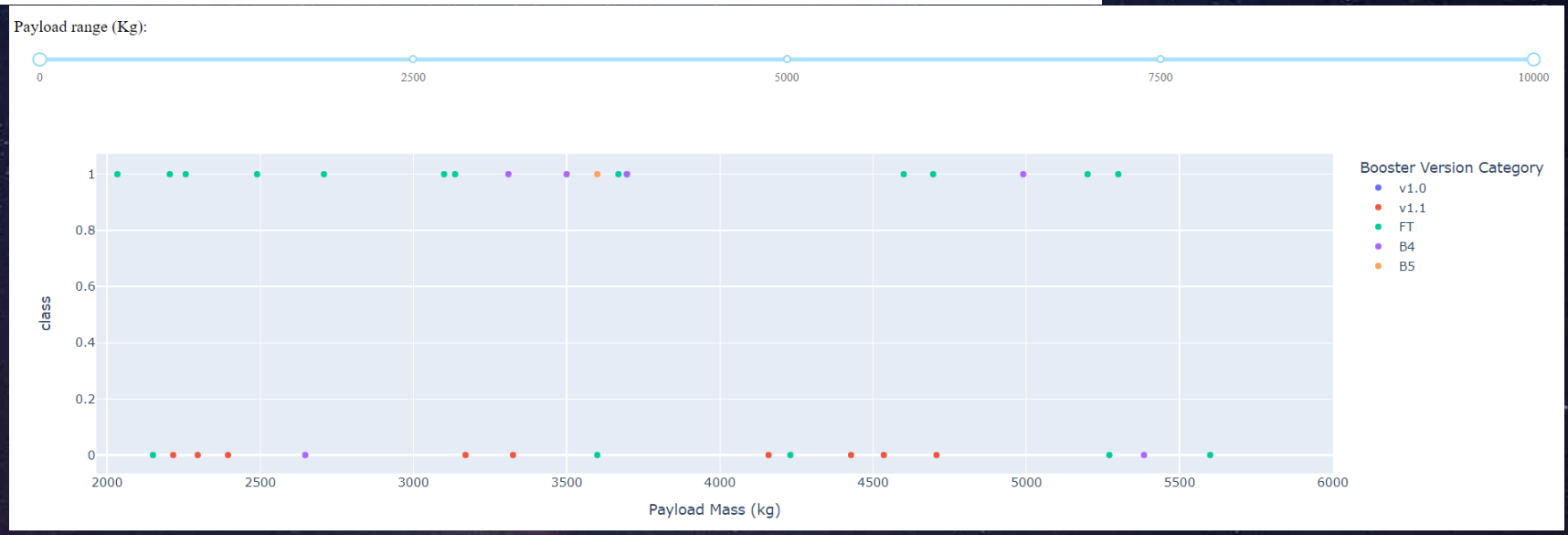


DASHBOARD – Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slide



We can see the success rates for 2000-4000 kg payloads is higher than the heavy weighted payloads.

Also in that payload category, FT Booster Category Version has higher success rate



Section 6

Classification Accuracy using training data

As you can see our accuracy is extremely close but we do have a winner its down to decimal places! using this function

Here The tree algorithm wins!!

After selecting the best hyperparameters for the decision tree classifier using the validation data, we achieved 87.32% accuracy on the test data.

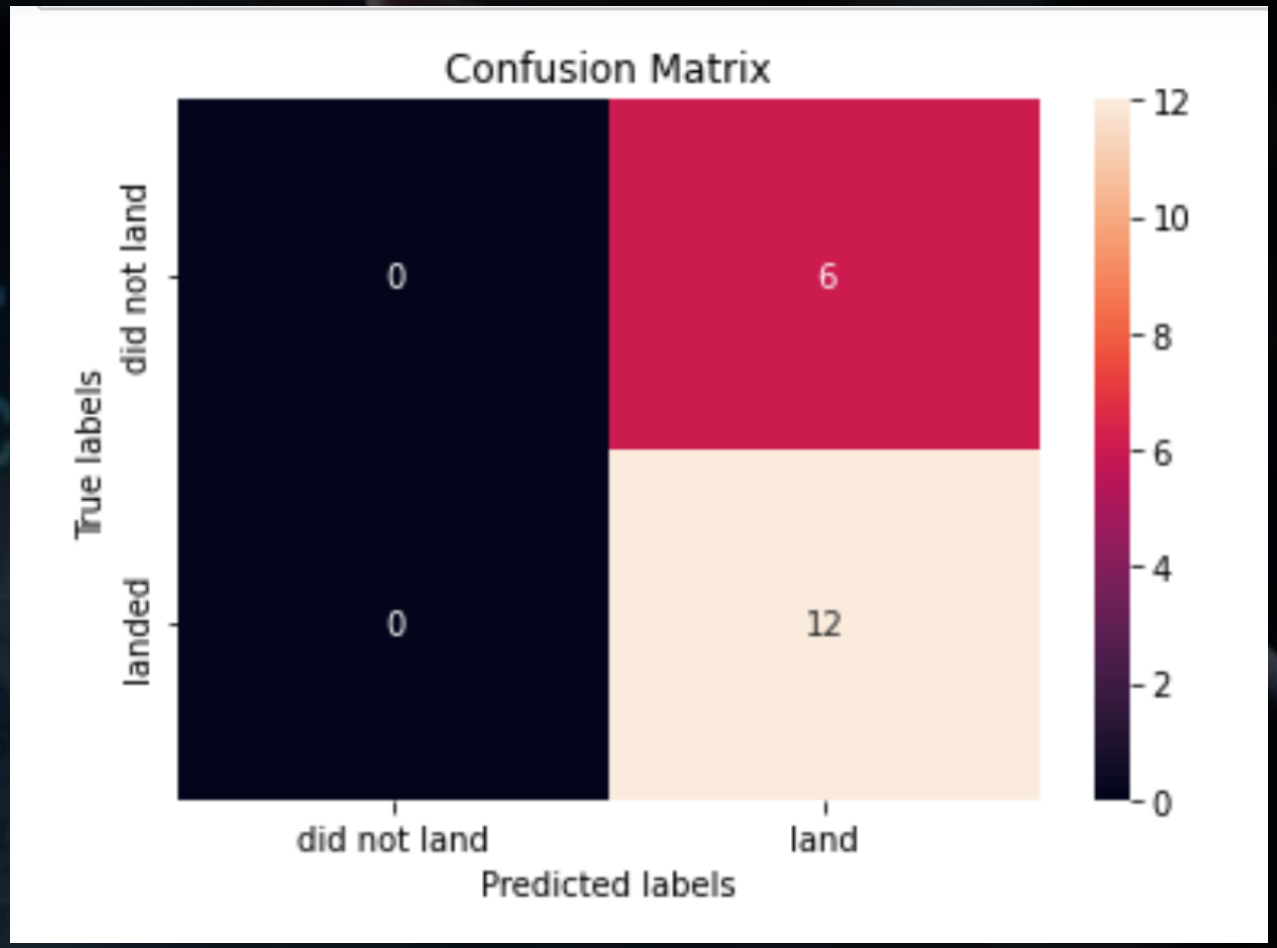
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto',
'splitter': 'random'}
```

Classification Accuracy using training data

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.



Conclusion

- 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
- KSC LC-39A had the most successful launches of any sites.
- Orbit GEO,HEO,SSO,ES-L1 has the best Success Rate
- The Tree Classifier Algorithm is the best for Machine Learning for this dataset

THANK YOU