

IBM DATA SCIENCE CAPSTONE PROJECT

A photograph of a Space X Falcon 9 rocket launching, viewed from the ground. The rocket is ascending vertically, leaving a large, bright orange and yellow plume of fire and white smoke behind it. The sky is a clear blue with scattered white clouds. The horizon is visible at the bottom of the frame.

Space X Falcon 9 Landing Analysis

Nguyen Thi Huong Giang – November 2024

OUTLINE



- 01 Executive Summary
- 02 Introduction
- 03 Methodology
- 04 Results
- 05 Conclusions

1. EXECUTIVE SUMMARY



Summary of Methodologies:

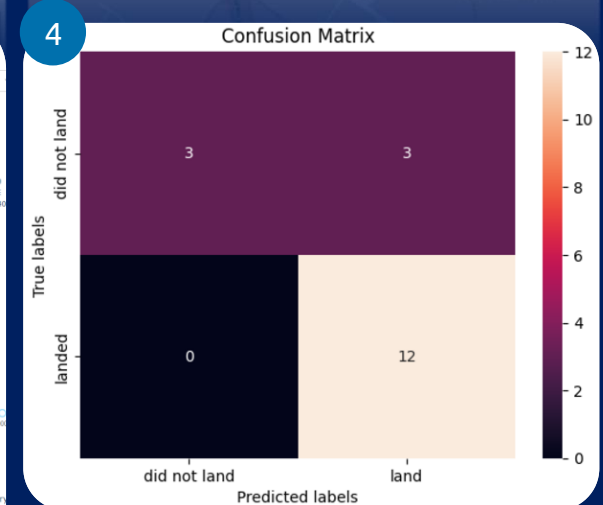
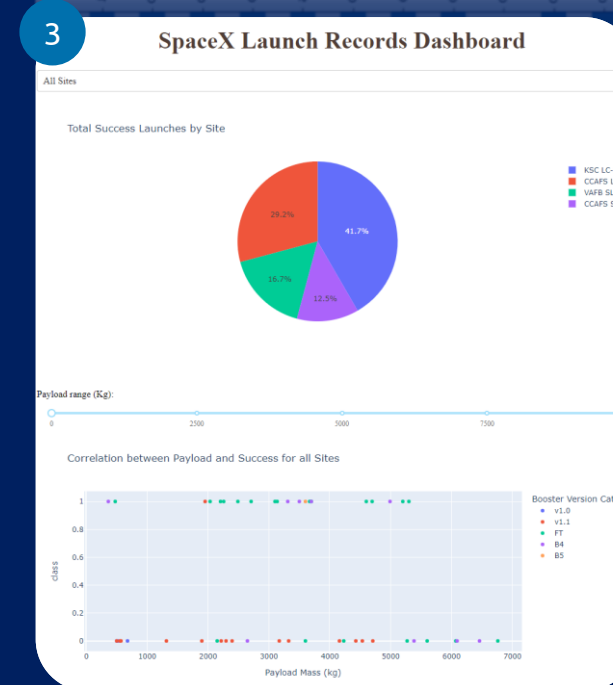
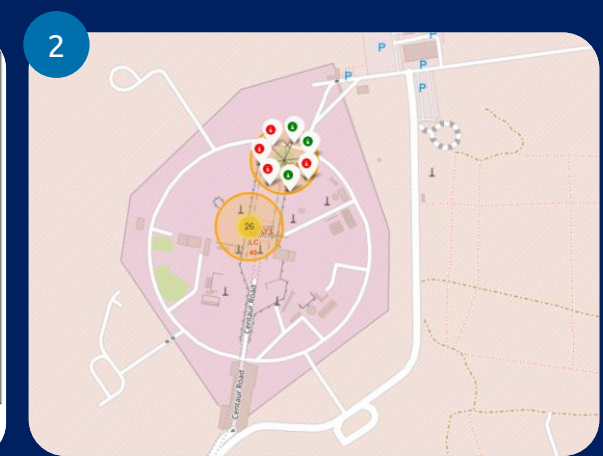
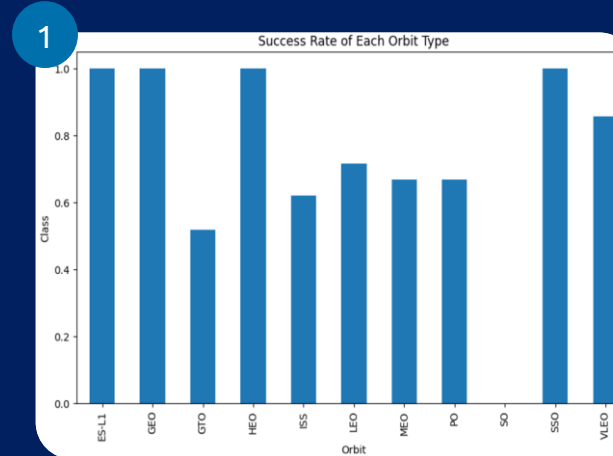
This project follows these steps:

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Analytics
- Predictive Analysis (Classification)

Summary of Results:

This project produced the following outputs and visualizations:

1. Exploratory Data Analysis (EDA) results
2. Geospatial analytics
3. Interactive dashboard
4. Predictive analysis of classification models



2. INTRODUCTION

- SpaceX launches Falcon 9 rockets at a cost of around \$62m. This is considerably cheaper than other providers (which usually cost upwards of \$165m), and much of the savings are because SpaceX can land, and then re-use the first stage of the rocket.
- If we can make predictions on whether the first stage will land, we can determine the cost of a launch, and use this information to assess whether or not an alternate company should bid and SpaceX for a rocket launch.
- This project will ultimately **predict if the Space X Falcon 9 first stage will land successfully.**



3. METHODOLOGY SUMMARY



1. Data Collection

- Making GET requests to the SpaceX REST API
- Web Scraping

2. Data Wrangling

- Using the `.fillna()` method to remove NaN values
- Using the `.value_counts()` method to determine the following:
 - Number of launches on each site
 - Number and occurrence of each orbit
 - Number and occurrence of mission outcome per orbit type
- Creating a landing outcome label that shows the following:
 - 0 when the booster did not land successfully
 - 1 when the booster did land successfully

3. Exploratory Data Analysis

- Using SQL queries to manipulate and evaluate the SpaceX dataset
- Using Pandas and Matplotlib to visualize relationships between variables, and determine patterns

4. Interactive Visual Analytics

- Geospatial analytics using Folium
- Creating an interactive dashboard using Plotly Dash

5. Data Modelling and Evaluation

- Using Scikit-Learn to:
 - Pre-process (standardize) the data
 - Split the data into training and testing data using `train_test_split`
 - Train different classification models
 - Find hyperparameters using `GridSearchCV`
- Plotting confusion matrices for each classification model
- Assessing the accuracy of each classification model

DATA COLLECTION – SPACE X REST API

Using the SpaceX API to retrieve data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.

1

- Make a GET response to the SpaceX REST API
- Convert the response to a .json file then to a Pandas DataFrame

2

- Use custom logic to clean the data (see Appendix)
- Define lists for data to be stored in
- Call custom functions (see Appendix) to retrieve data and fill the lists
- Use these lists as values in a dictionary and construct the dataset

3

- Create a Pandas DataFrame from the constructed dictionary dataset

4

- Filter the DataFrame to only include Falcon 9 launches
- Reset the FlightNumber column
- Replace missing values of PayloadMass with the mean PayloadMass value

1

```
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())
```

2

```
#Global variables  
BoosterVersion = []  
PayloadMass = []  
Orbit = []  
LaunchSite = []  
Outcome = []  
Flights = []  
GridFins = []  
Reused = []  
Legs = []  
LandingPad = []  
Block = []  
ReusedCount = []  
Serial = []  
Longitude = []  
Latitude = []
```

```
# Call getBoosterVersion  
getBoosterVersion(data)
```

```
# Call getLaunchSite  
getLaunchSite(data)
```

```
# Call getPayloadData  
getPayloadData(data)
```

```
# Call getCoreData  
getCoreData(data)
```

```
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}
```

3

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

4

```
data_falcon9 = launch_data[launch_data['BoosterVersion']!='Falcon 1']  
data_falcon9.loc[:, 'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))  
  
# Calculate the mean value of PayloadMass column  
mean=data_falcon9['PayloadMass'].mean()  
# Replace the np.nan values with its mean value  
data_falcon9['PayloadMass'].replace(np.nan,mean,inplace=True)
```



DATA COLLECTION – WEB SCRAPING

Web scraping to collect Falcon 9 historical launch records from a Wikipedia page titled List of Falcon 9 and Falcon Heavy launches.

1

- Request the HTML page from the static URL
- Assign the response to an object

2

- Create a BeautifulSoup object from the HTML response object
- Find all tables within the HTML page

3

- Collect all column header names from the tables found within the HTML page

4

- Use the column names as keys in a dictionary
- Use custom functions and logic to parse all launch tables (see [Appendix](#)) to fill the dictionary values

5

- Convert the dictionary to a Pandas DataFrame ready for export

1

```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"

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

2

```
soup = BeautifulSoup(html_data.text)
html_tables = soup.find_all('table')
```

3

```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (if name is not None and len(name) > 0) into a list called column_names
for element in first_launch_table.find_all('th'):
    name = extract_column_from_header(element)
    if name is not None and len(name) > 0:
        column_names.append(name)
```

4

```
launch_dict = dict.fromkeys(column_names)

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

# Let's initial the launch_dict with each value to be an empty list
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'] = []
```

5

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

DATA MANIPULATION/WRANGLING – PANDAS

Context:

- The SpaceX dataset contains several Space X launch facilities, and each location is in the `LaunchSite` column.
- Each launch aims to a dedicated orbit, and some of the common orbit types are shown in the figure below. The orbit type is in the `Orbit` column.



Initial Data Exploration:

- Using the `.value_counts()` method to determine the following:
 1. Number of launches on each site
 2. Number and occurrence of each orbit
 3. Number and occurrence of landing outcome per orbit type

1

```
# Apply value_counts() on column LaunchSite  
df['LaunchSite'].value_counts()
```

```
LaunchSite  
CCAFS SLC 40    55  
KSC LC 39A      22  
VAFB SLC 4E     13  
Name: count, dtype: int64
```

2

```
# Apply value_counts on Orbit column  
df['Orbit'].value_counts()
```

```
Orbit  
GTO      27  
ISS      21  
VLEO     14  
PO        9  
LEO        7  
SSO        5  
MEO        3  
HEO        1  
ES-L1     1  
SO         1  
GEO        1  
Name: count, dtype: int64
```

3

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

```
Outcome  
True ASDS      41  
None None      19  
True RTLS      14  
False ASDS      6  
True Ocean      5  
False Ocean      2  
None ASDS        2  
False RTLS        1  
Name: count, dtype: int64
```


DATA MANIPULATION/WRANGLING – PANDAS



Context:

- The landing outcome is shown in the `Outcome` column:
 - `True Ocean` – the mission outcome was successfully landed to a specific region of the ocean
 - `False Ocean` – the mission outcome was unsuccessfully landed to a specific region of the ocean.
 - `True RTLS` – the mission outcome was successfully landed to a ground pad
 - `False RTLS` – the mission outcome was unsuccessfully landed to a ground pad.
 - `True ASDS` – the mission outcome was successfully landed to a drone ship
 - `False ASDS` – the mission outcome was unsuccessfully landed to a drone ship.
 - `None ASDS` and `None None` – these represent a failure to land.

Data Wrangling:

- To determine whether a booster will successfully land, it is best to have a binary column, i.e., where the value is 1 or 0, representing the success of the landing.
- This is done by:
 - Defining a set of unsuccessful (bad) outcomes, `bad_outcome`
 - Creating a list, `landing_class`, where the element is 0 if the corresponding row in `Outcome` is in the set `bad_outcome`, otherwise, it's 1.
 - Create a `Class` column that contains the values from the list `landing_class`
 - Export the DataFrame as a .csv file.

1

```
bad_outcomes = set(landing_outcomes.keys()[[1,3,5,6,7]])
bad_outcomes

{'False ASDS', 'False Ocean', 'False RTLS', 'None ASDS', 'None None'}
```

2

```
# 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)
```

3

```
df['Class'] = landing_class
```

4

```
df.to_csv("dataset_part_2.csv", index=False)
```

EXPLORATORY DATA ANALYSIS (EDA) – VISUALIZATION



SCATTER CHARTS

Scatter charts were produced to visualize the relationships between:

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

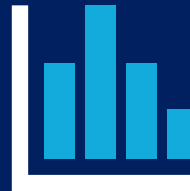


Scatter charts are useful to observe relationships, or correlations, between two numeric variables.

BAR CHART

A bar chart was produced to visualize the relationship between:

- Success Rate and Orbit Type



Bar charts are used to compare a numerical value to a categorical variable. Horizontal or vertical bar charts can be used, depending on the size of the data.

LINE CHARTS

Line charts were produced to visualize the relationships between:

- Success Rate and Year (i.e. the launch success yearly trend)



Line charts contain numerical values on both axes, and are generally used to show the change of a variable over time.



EXPLORATORY DATA ANALYSIS (EDA) – SQL

To gather some information about the dataset, some SQL queries were performed.

The SQL queries performed on the data set were used to:

1. Display the names of the unique launch sites in the space mission
2. Display 5 records where launch sites begin with the string 'CCA'
3. Display the total payload mass carried by boosters launched by NASA (CRS)
4. Display the average payload mass carried by booster version F9 v1.1
5. List the date when the first successful landing outcome on a ground pad was achieved
6. List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg
7. List the total number of successful and failed mission outcomes
8. List the names of the booster versions which have carried the maximum payload mass
9. List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015
10. 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

GEOSPATIAL ANALYSIS – FOLIUM



The following steps were taken to visualize the launch data on an interactive map:

1. Mark all launch sites on a map

- Initialise the map using a Folium `Map` object
- Add a `folium.Circle` and `folium.Marker` for each launch site on the launch map

2. Mark the success/failed launches for each site on a map

- As many launches have the same coordinates, it makes sense to cluster them together.
- Before clustering them, assign a marker colour of successful (class = 1) as green, and failed (class = 0) as red.
- To put the launches into clusters, for each launch, add a `folium.Marker` to the `MarkerCluster()` object.
- Create an icon as a text label, assigning the `icon_color` as the `marker_colour` determined previously.

3. Calculate the distances between a launch site to its proximities

- To explore the proximities of launch sites, calculations of distances between points can be made using the `Lat` and `Long` values.
- After marking a point using the `Lat` and `Long` values, create a `folium.Marker` object to show the distance.
- To display the distance line between two points, draw a `folium.PolyLine` and add this to the map.



INTERACTIVE DASHBOARD – PLOTLY DASH

The following plots were added to a Plotly Dash dashboard to have an interactive visualisation of the data:

1. Pie chart (`px.pie()`) showing the total successful launches per site
 - This makes it clear to see which sites are most successful
 - The chart could also be filtered (using a `dcc.Dropdown()` object) to see the success/failure ratio for an individual site
2. Scatter graph (`px.scatter()`) to show the correlation between outcome (success or not) and payload mass (kg)
 - This could be filtered (using a `RangeSlider()` object) by ranges of payload masses
 - It could also be filtered by booster version

PREDICTIVE ANALYSIS - CLASSIFICATION



The following steps were taking to develop, evaluate, and find the best performing classification model:

Model Development



Model Evaluation



Finding the Best Classification Model

- To prepare the dataset for model development:
 - Load dataset
 - Perform necessary data transformations (standardise and pre-process)
 - Split data into training and test data sets, using `train_test_split()`
 - Decide which type of machine learning algorithms are most appropriate
- For each chosen algorithm:
 - Create a `GridSearchCV` object and a dictionary of parameters
 - Fit the object to the parameters
 - Use the training data set to train the model

- For each chosen algorithm:
 - Using the output `GridSearchCV` object:
 - Check the tuned hyperparameters (`best_params_`)
 - Check the accuracy (`score` and `best_score_`)
 - Plot and examine the Confusion Matrix

- Review the accuracy scores for all chosen algorithms
- The model with the highest accuracy score is determined as the best performing model

4. RESULTS

Exploratory Data Analysis

Interactive Analytics

Predictive Analysis





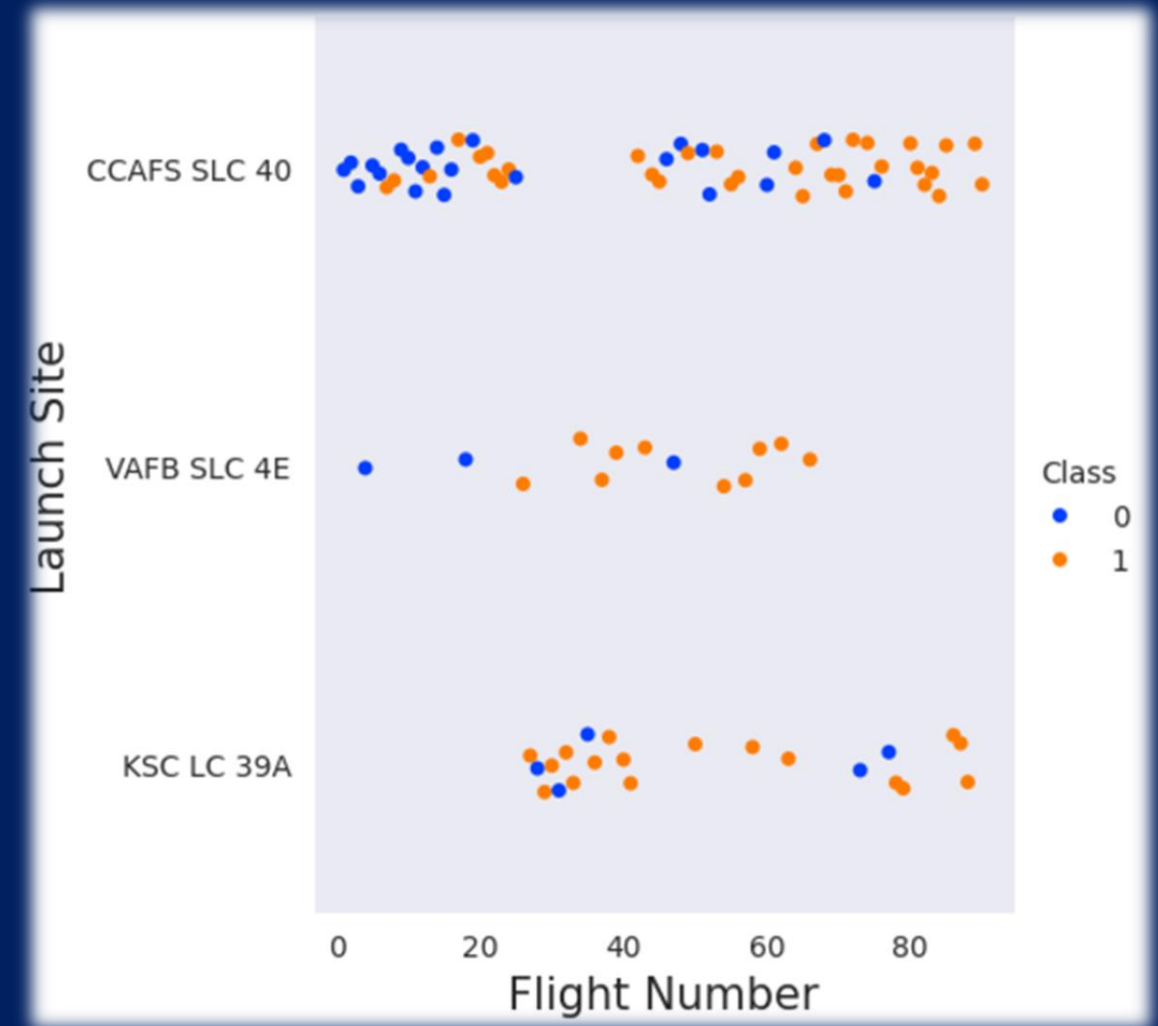
EDA - WITH VISUALIZATION

LAUNCH SITE VS. FLIGHT NUMBER



The scatter plot of Launch Site vs. Flight Number shows that:

- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight numbers < 30) were launched from CCAFS SLC 40, and were generally unsuccessful.
- The flights from VAFB SLC 4E also show this trend, that earlier flights were less successful.
- No early flights were launched from KSC LC 39A, so the launches from this site are more successful.
- Above a flight number of around 30, there are significantly more successful landings (Class = 1).

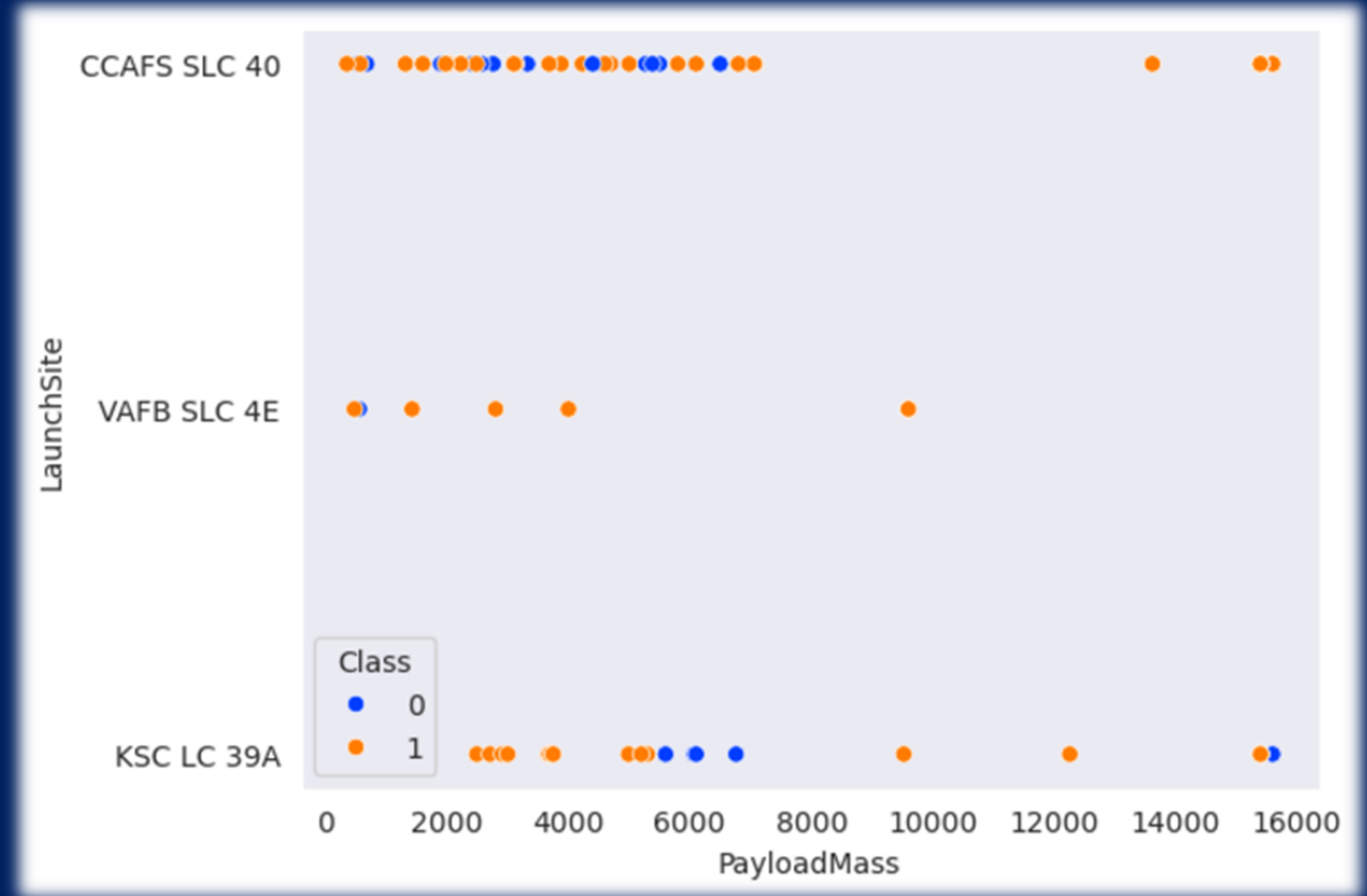


LAUNCH SITE VS. PAYLOAD MASS



The scatter plot of Launch Site vs. Payload Mass shows that:

- Above a payload mass of around 7000 kg, there are very few unsuccessful landings, but there is also far less data for these heavier launches.
- There is no clear correlation between payload mass and success rate for a given launch site.
- All sites launched a variety of payload masses, with most of the launches from CCAFS SLC 40 being comparatively lighter payloads (with some outliers).



SUCCESS RATE VS. ORBIT TYPE

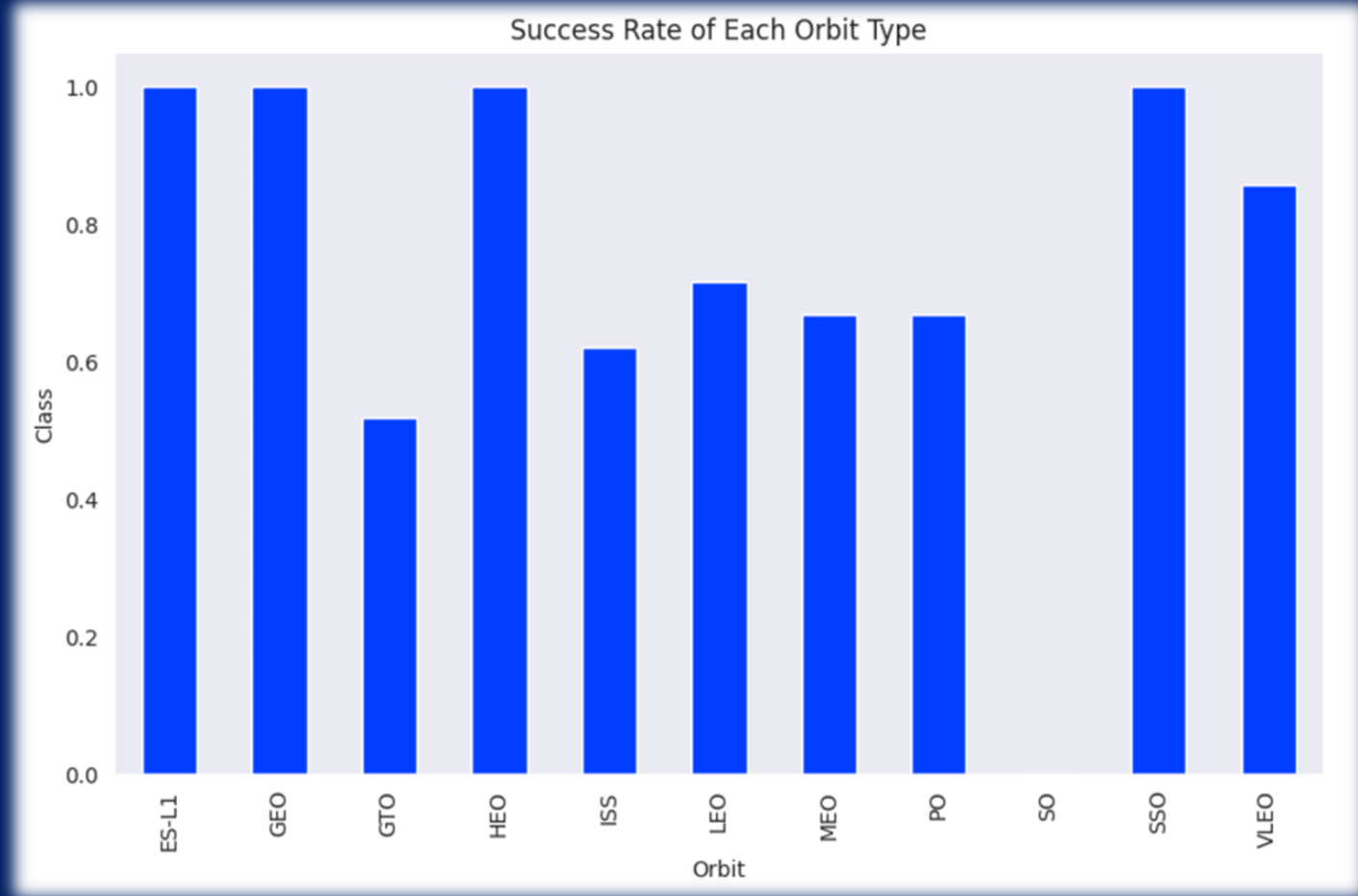


The bar chart of Success Rate vs. Orbit Type shows that the following orbits have the highest (100%) success rate:

- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)

The orbit with the lowest (0%) success rate is:

- SO (Heliocentric Orbit)

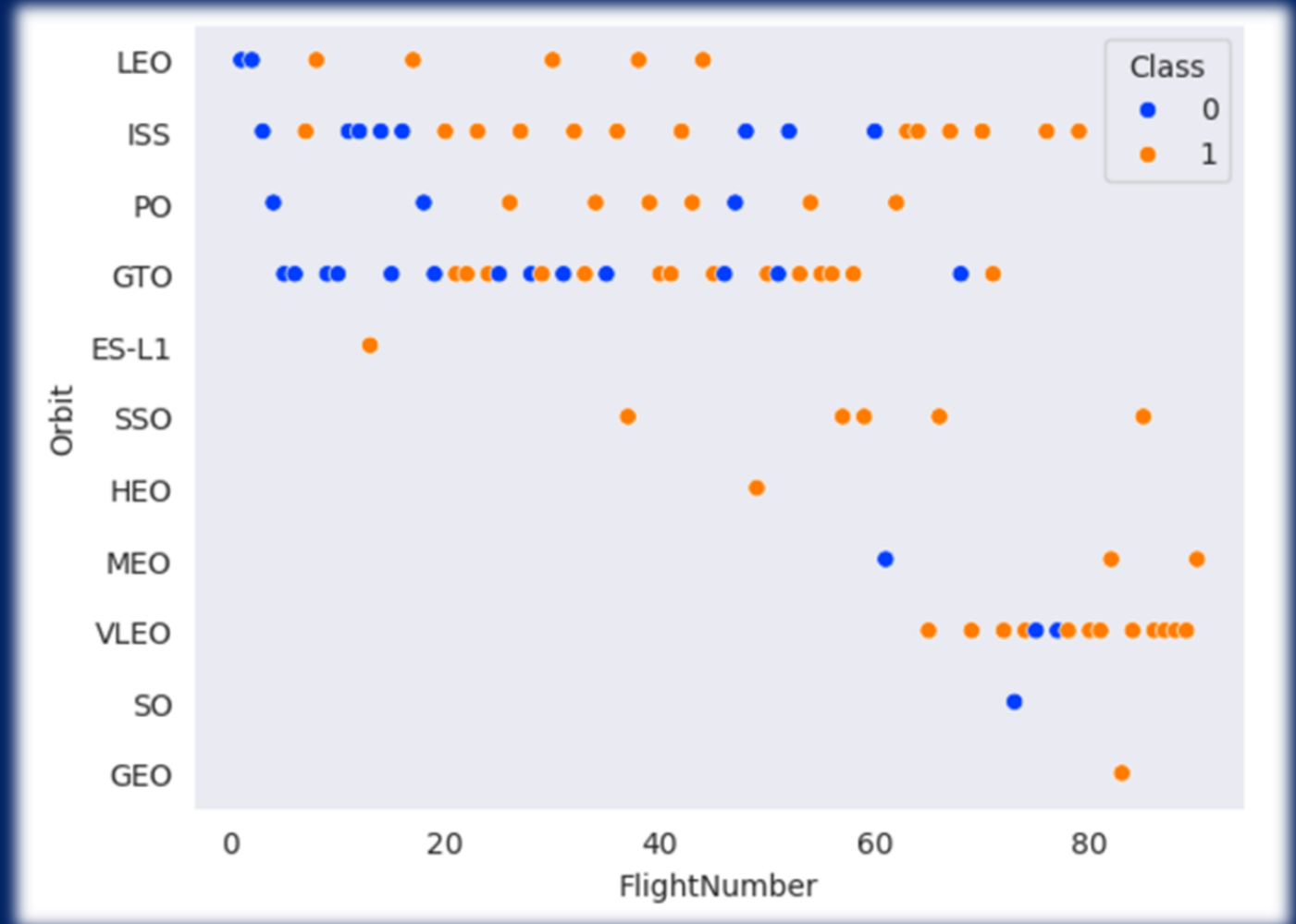




ORBIT TYPE VS. FLIGHT NUMBER

This scatter plot of Orbit Type vs. Flight number shows a few useful things that the previous plots did not, such as:

- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is little relationship between Flight Number and Success Rate for GTO.
- Generally, as Flight Number increases, the success rate increases. This is most extreme for LEO, where unsuccessful landings only occurred for the low flight numbers (early launches).

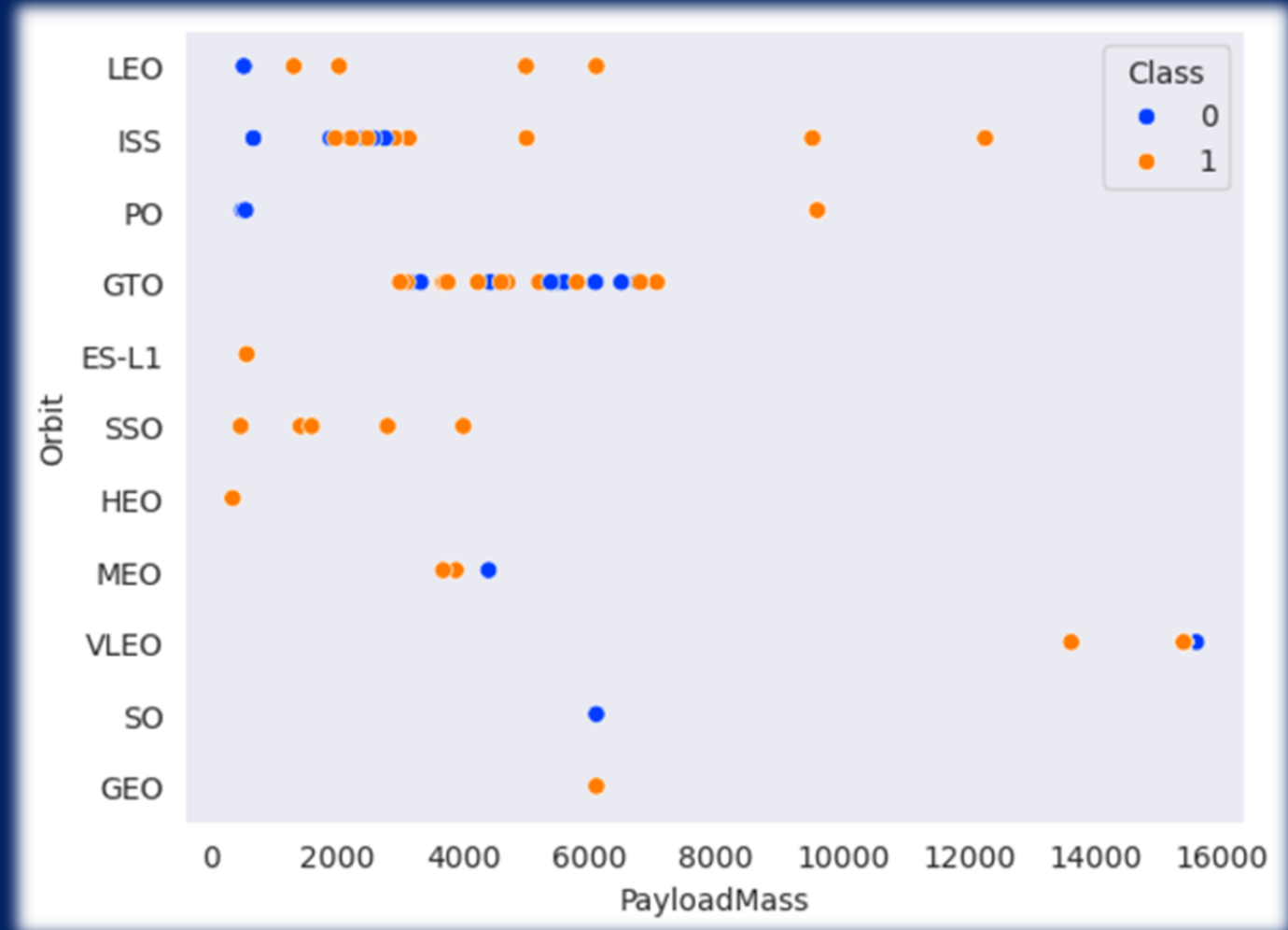


ORBIT TYPE VS. PAYLOAD MASS



This scatter plot of Orbit Type vs. Payload Mass shows that:

- The following orbit types have more success with heavy payloads:
 - PO (although the number of data points is small)
 - ISS
 - LEO
- For GTO, the relationship between payload mass and success rate is unclear.
- VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.

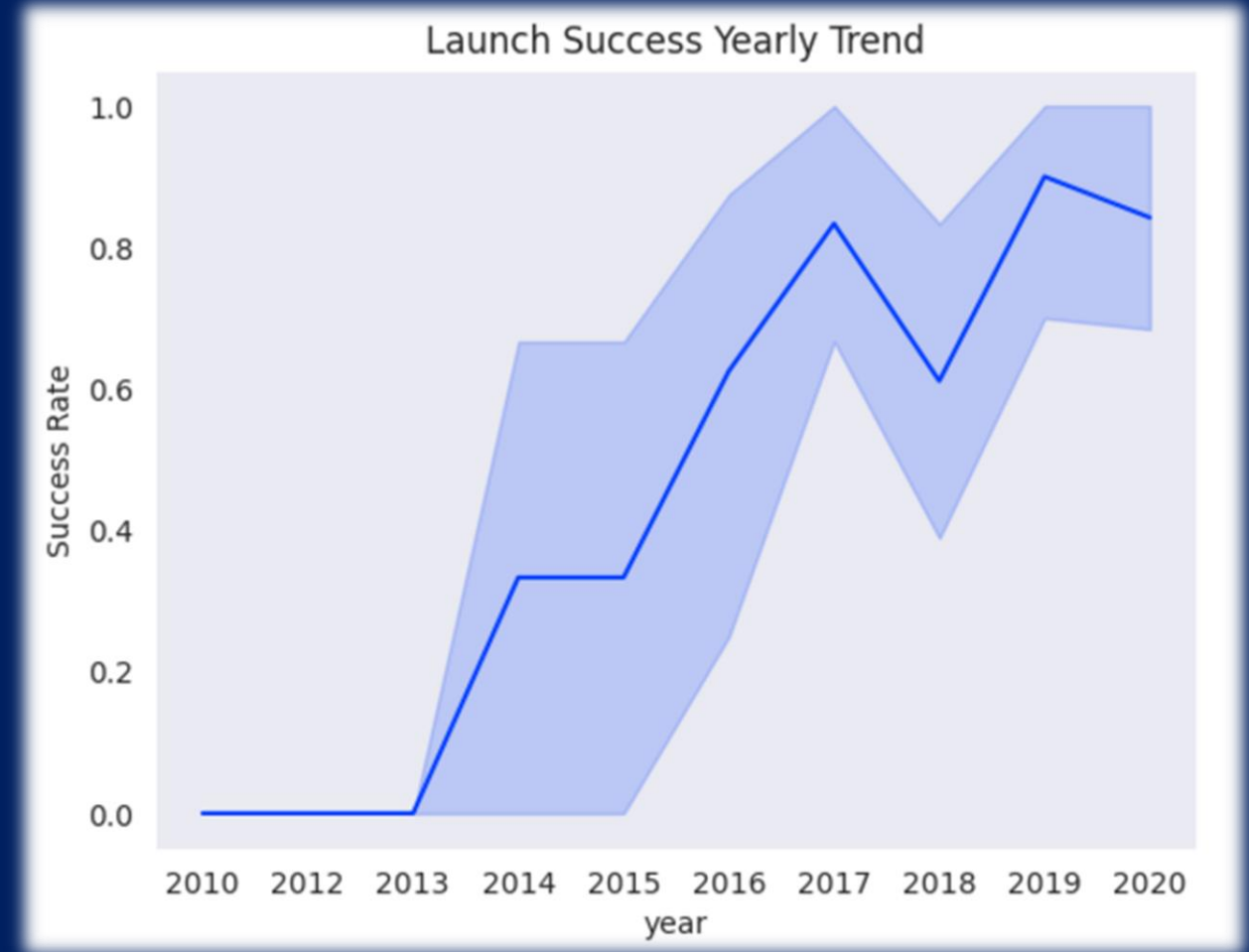


LAUNCH SUCCESS YEARLY TREND



The line chart of yearly average success rate shows that:

- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
- After 2016, there was always a greater than 50% chance of success.





EDA - WITH SQL



ALL LAUNCH SITE NAMES

Find the names of the unique launch sites.

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL;
```



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


The key word **DISTINCT** to show only unique **LAUNCH_SITE** from the SpaceX data.

LAUNCH SITE NAMES BEGIN WITH 'CCA'



Find 5 records where launch sites begin with 'CCA'.

```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```



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	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	0:35:00	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

LIMIT 5 fetches only 5 records, and the **LIKE** keyword is used with the wild card 'CCA%' to retrieve string values beginning with 'CCA'.



TOTAL PAYLOAD MASS

Calculate the total payload carried by boosters from NASA.

```
%sql SELECT SUM(PAYLOAD_MASS_KG_) FROM SPACEXTBL WHERE CUSTOMER = 'NASA (CRS)';
```



SUM(PAYLOAD_MASS_KG_)
45596

The **SUM** keyword is used to calculate the total of the **LAUNCH** column, and the **SUM** keyword (and the associated condition) filters the results to only boosters from NASA (CRS).



AVERAGE PAYLOAD MASS BY F9 V1.1

Calculate the average payload mass carried by booster version F9 v1.1.

```
%sql SELECT AVG(PAYLOAD_MASS_KG_) FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1'
```



AVG(PAYLOAD_MASS_KG_)

2928.4

The **AVG** keyword is used to calculate the average of the **PAYLOAD_MASS_KG_** column, and the **WHERE** keyword (and the associated condition) filters the results to only the F9 v1.1 booster version.



FIRST SUCCESSFUL GROUND LANDING DATE

Find the dates of the first successful landing outcome on ground pad.

```
%sql SELECT MIN(DATE) FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';
```



MIN(DATE)

2015-12-22

The **MIN** keyword is used to calculate the minimum of the **DATE** column, i.e. the first date, and the **WHERE** keyword (and the associated condition) filters the results to only the successful ground pad landings.

SUCCESSFUL DRONE SHIP LANDING WITH PAYLOAD BETWEEN 4000 AND 6000



List the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000.

```
%sql SELECT BOOSTER_VERSION from SPACEXTBL WHERE LANDING_OUTCOME = 'Success (drone ship)' and PAYLOAD_MASS__KG_ >4000 and PAYLOAD_MASS__KG_ <6000;
```



Booster_Version
F9 FT B1022
F9 FT B1026
F9 FT B1021.2
F9 FT B1031.2

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

TOTAL NUMBER OF SUCCESSFUL AND FAILURE MISSION OUTCOMES



Calculate the total number of successful and failure mission outcome.

```
%sql select Mission_Outcome, count(Mission_Outcome) as counts from spacextbl group by Mission_Outcome;
```



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

The **COUNT** keyword is used to calculate the total number of mission outcomes, and the **GROUPBY** keyword is also used to group these results by the type of mission outcome.



BOOSTERS CARRIED MAXIMUM PAYLOAD

List the names of the booster which have carried the maximum payload mass.

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



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

A subquery is used here. The **SELECT** statement within the brackets finds the maximum payload, and this value is used in the **WHERE** condition.



2015 LAUNCH RECORDS

List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015.

```
%sql SELECT substr(Date,6,2) as month, DATE, BOOSTER_VERSION, LAUNCH_SITE, LANDING_OUTCOME FROM SPACEXTBL \
WHERE LANDING_OUTCOME = 'Failure (drone ship)' and substr(Date,0,5)='2015'
```



month	Date	Booster_Version	Launch_Site	Landing_Outcome
01	2015-01-10	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	2015-04-14	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

The **WHERE** keyword is used to filter the results for only failed landing outcomes, **AND** only for the year of 2015.

RANK LANDING OUTCOMES BETWEEN 2010-06-04 AND 2017-03-20



Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order.

```
%sql SELECT [Landing_Outcome], count(*) as count_outcomes \
FROM SPACEXTBL \
WHERE DATE between '2010-06-04' and '2017-03-20' group by [Landing_Outcome] order by count_outcomes DESC;
```

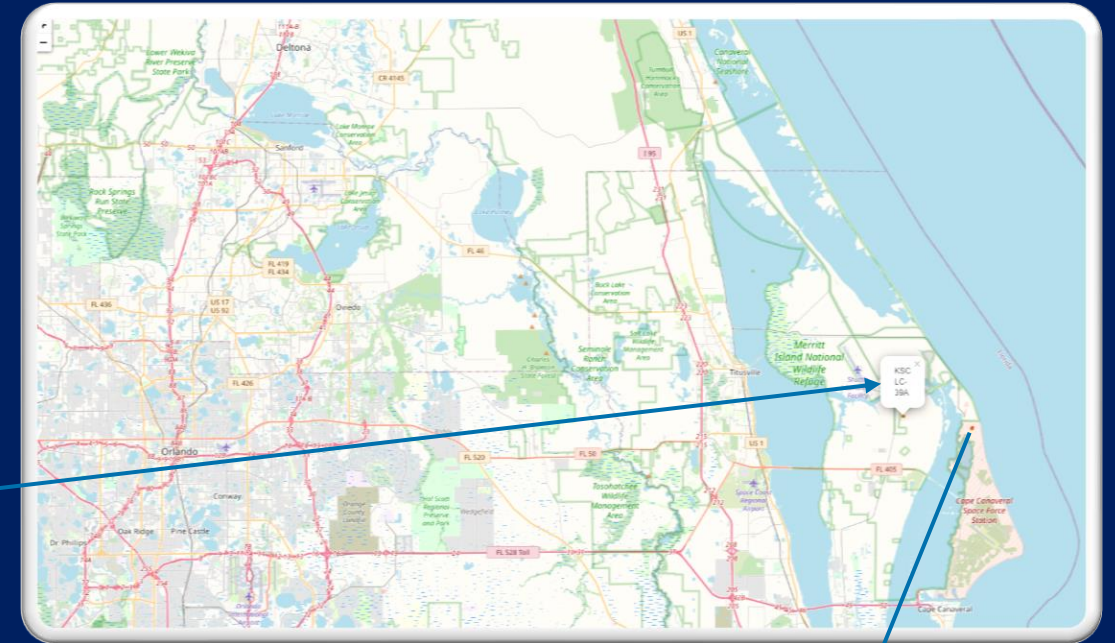
Landing_Outcome	count_outcomes
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

The **WHERE** keyword is used with the **BETWEEN** keyword to filter the results to dates only within those specified. The results are then grouped and ordered, using the keywords **GROUP BY** and **ORDER BY**, respectively, where **DESC** is used to specify the descending order.

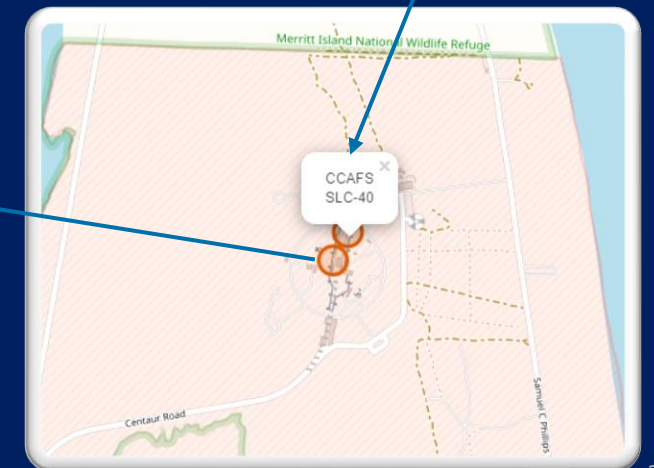
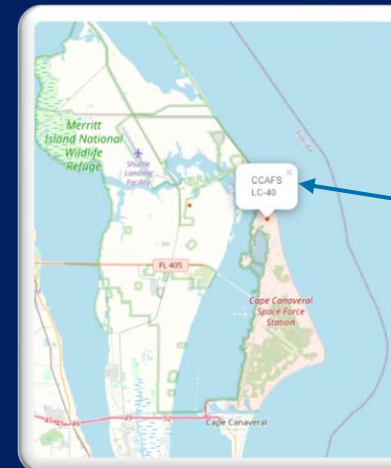


LAUNCH SITES PROXIMITY ANALYSIS – FOLIUM INTERACTIVE MAP

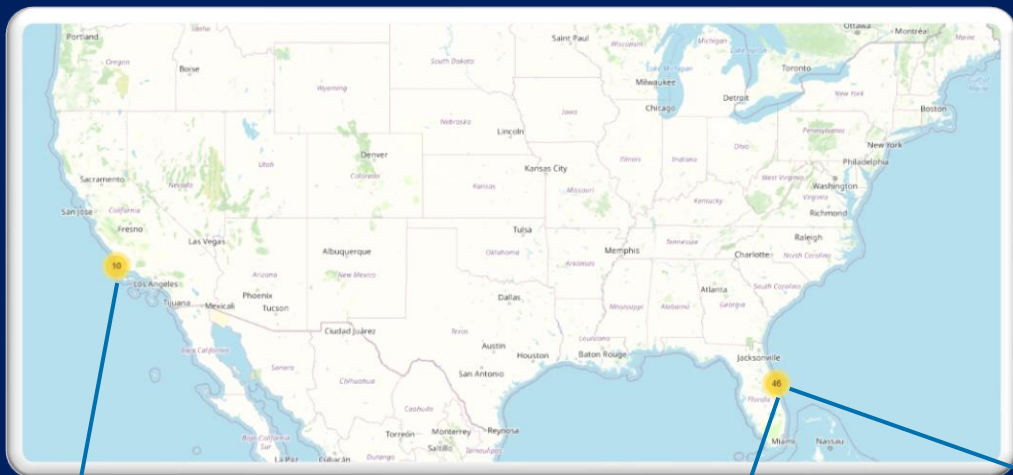
ALL LAUNCH SITES ON A MAP



All SpaceX launch sites are on coasts of the United States of America, specifically Florida and California.

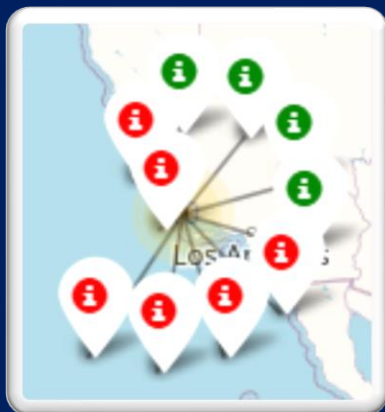


SUCCESS/FAILED LAUNCHES FOR EACH SITE

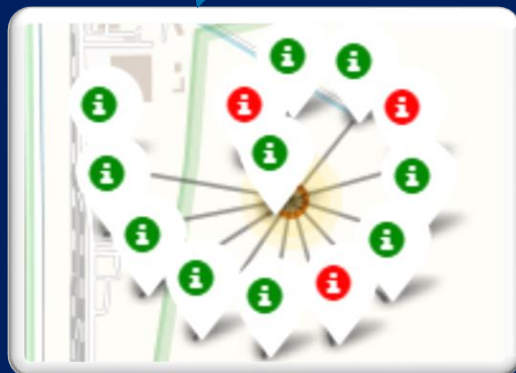


Launches have been grouped into clusters, and annotated with **green icons** for successful launches, and **red icons** for failed launches.

VAFB SLC-4E



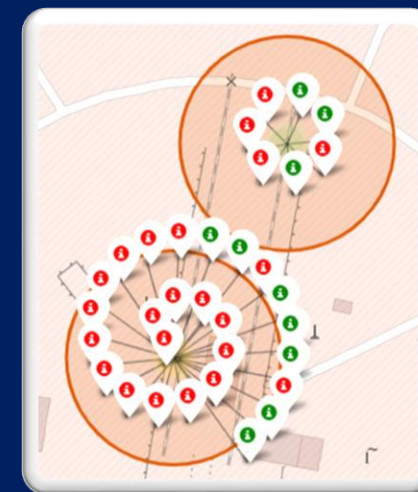
KSC LC-39A



CCAFS SLC-40 and CCAFS LC-40



=





PROXIMITY OF LAUNCH SITES TO OTHER POINTS OF INTEREST

Using the **CCAFS SLC-40** launch site as an example site, we can understand more about the placement of launch sites.

Are launch sites in close proximity to coastline?

- **YES.** The coastline is only 0.86 km due East.

Are launch sites in close proximity to highways?

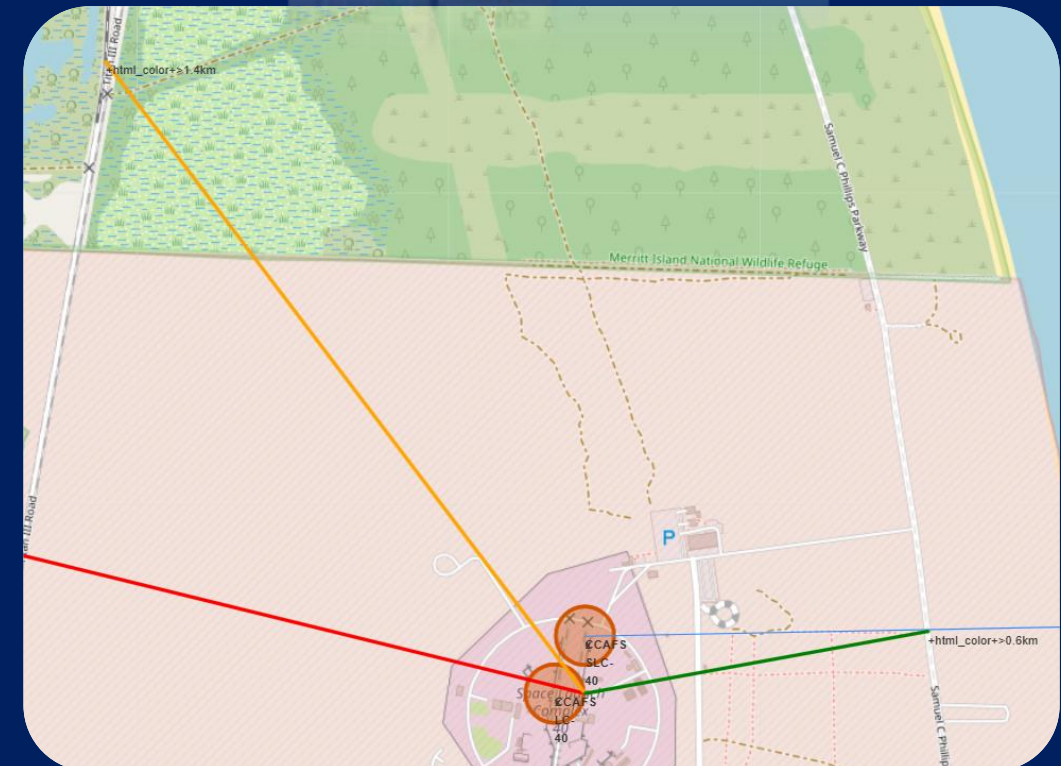
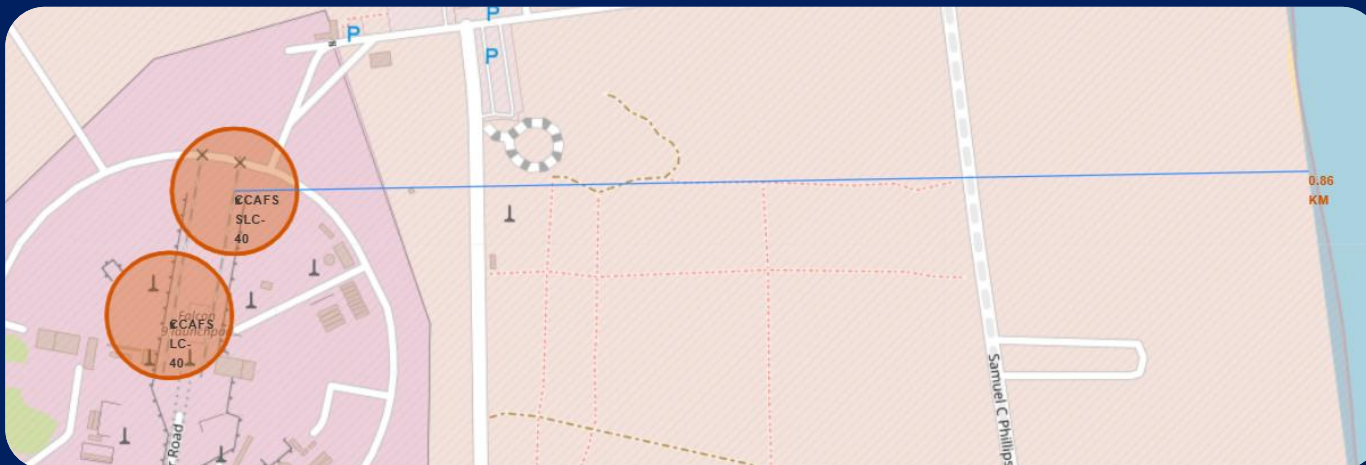
- **YES.** The nearest highway (green line) is only 0.6 km away.

Are launch sites in close proximity to railways?

- **YES.** The nearest railway (orange line) is only 1.4 km away.

Do launch sites keep certain distance away from cities?

- **YES.** The nearest city (Titusville – red line) is 23.2 km away.





INTERACTIVE DASHBOARD

- PLOTLY DASH

LAUNCH SUCCESS COUNT FOR ALL SITES

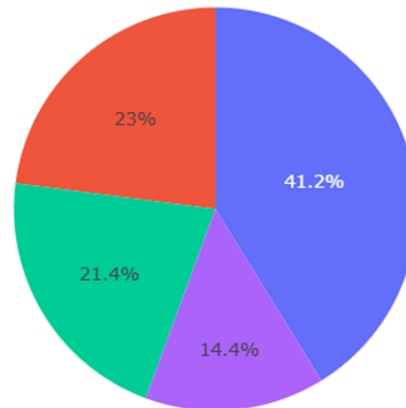


SpaceX Launch Records Dashboard

All Sites



Total Success Launches by Site



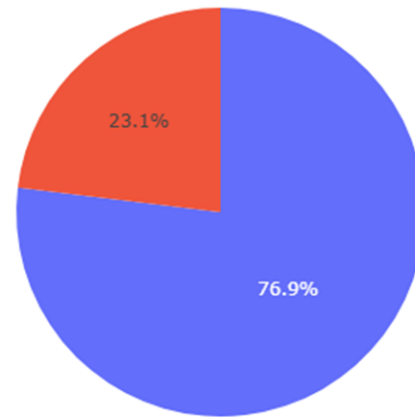
- KSC LC-39A
- CCAFS SLC-40
- VAFB SLC-4E
- CCAFS LC-40

The launch site **KSC LC-39 A** had the most successful launches, with 41.2% of the total successful launches.

PIE CHART FOR THE LAUNCH SITE WITH HIGHEST LAUNCH SUCCESS RATIO



Total Success Launches for Site KSC LC-39A



The launch site **KSC LC-39 A** also had the highest rate of successful launches, with a 76.9% success rate.

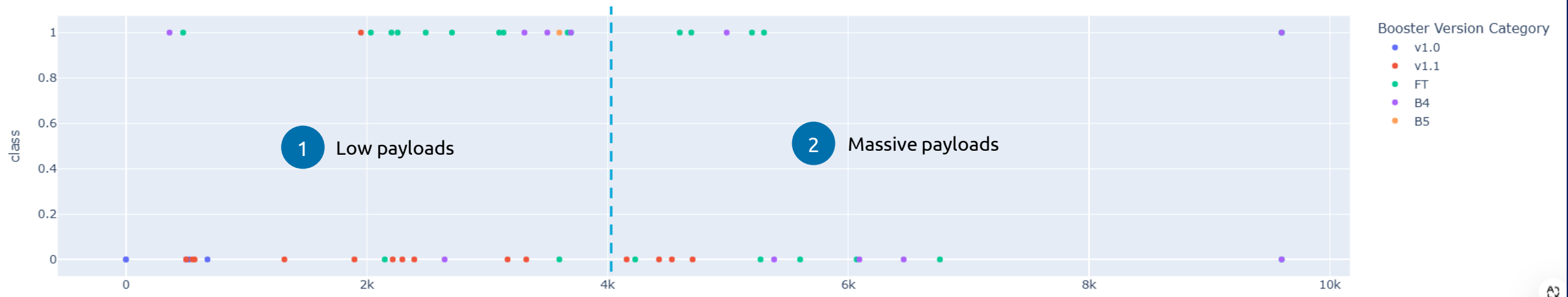
LAUNCH OUTCOME VS. PAYLOAD SCATTER PLOT FOR ALL SITES



Payload range (Kg):



Correlation Between Payload and Success for All Sites



- Plotting the launch outcome vs. payload for all sites shows a gap around 4000 kg, so it makes sense to split the data into 2 ranges:
 - 0 – 4000 kg (low payloads)
 - 4000 – 10000 kg (massive payloads)
- From these 2 plots, it can be shown that the success for massive payloads is lower than that for low payloads.
- It is also worth noting that some booster types (v1.0 and B5) have not been launched with massive payloads.



PREDICTIVE ANALYSIS - CLASSIFICATION

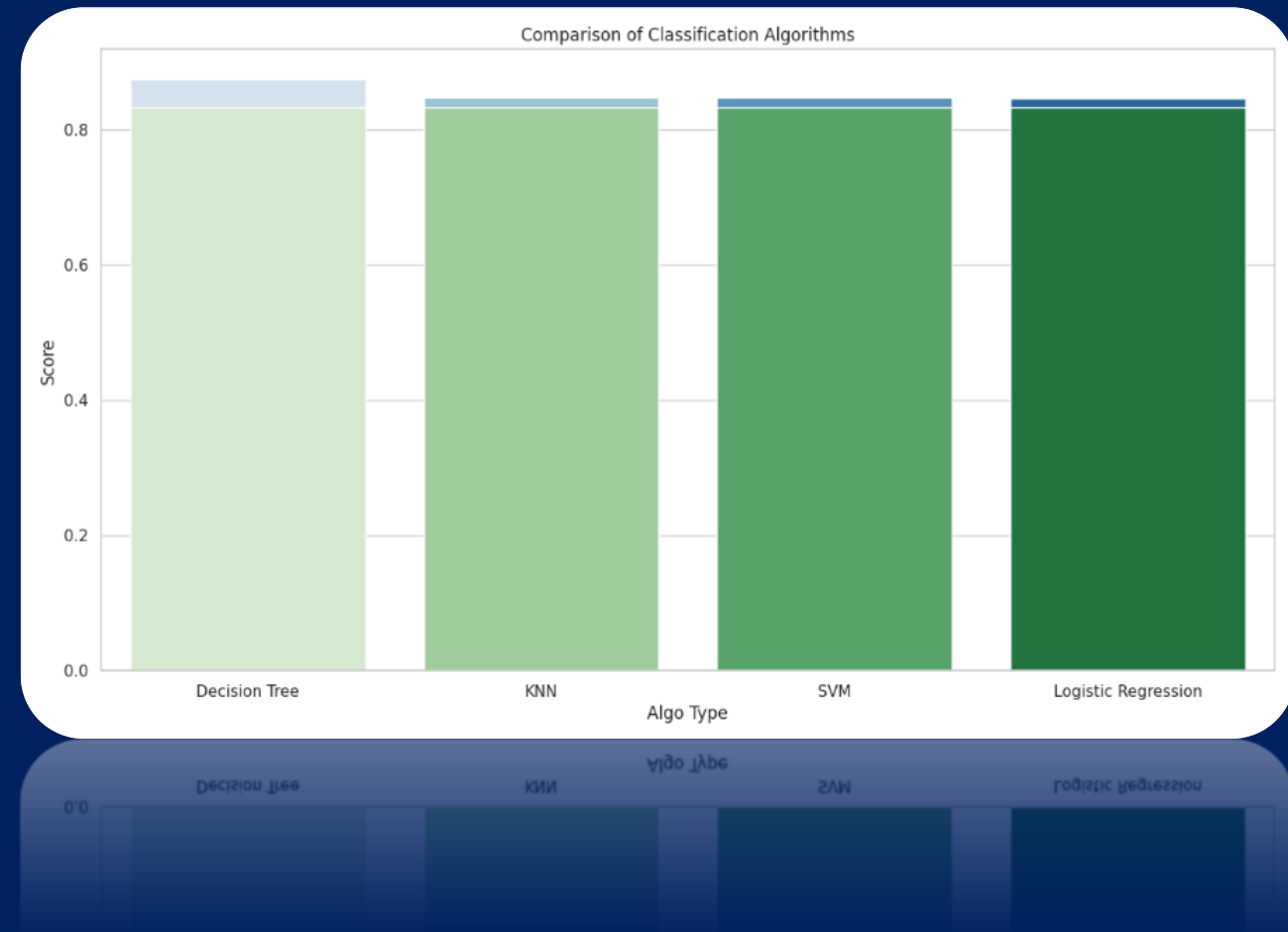


CLASSIFICATION ACCURACY

Plotting the Accuracy Score for each classification algorithm produces the following result:

- The **Decision Tree** model has the highest classification accuracy with the Accuracy Score is 87.5%

	Algo Type	Accuracy Score	Test Data Accuracy Score
2	Decision Tree	0.875000	0.833333
3	KNN	0.848214	0.833333
1	SVM	0.848214	0.833333
0	Logistic Regression	0.846429	0.833333





CLASSIFICATION ACCURACY

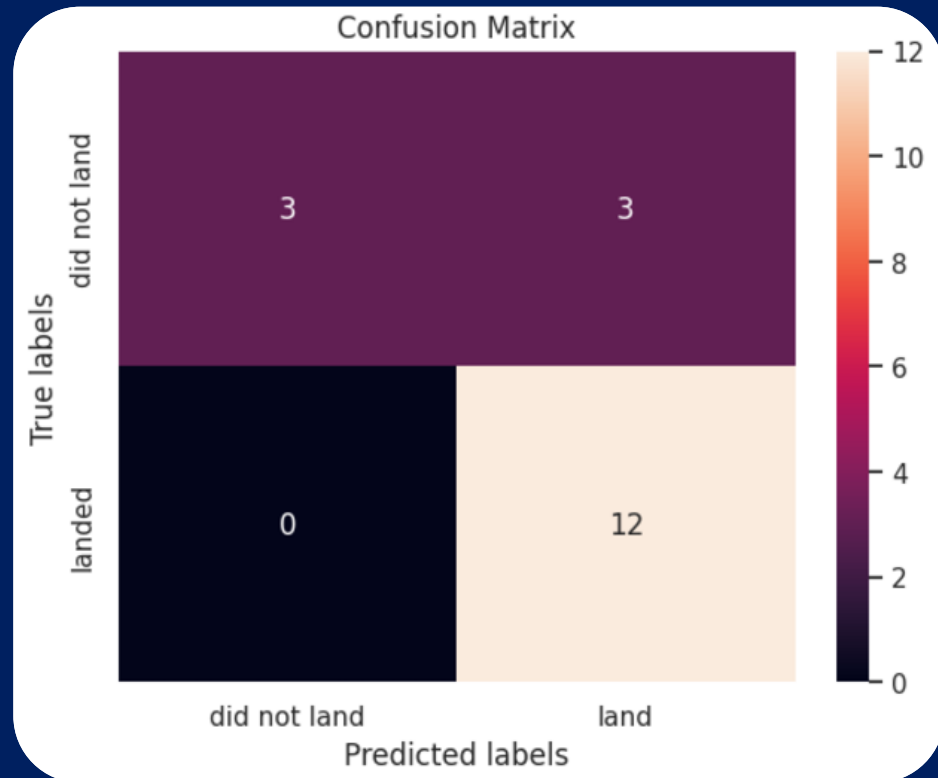
The decision tree classifier is the model with the highest classification accuracy

```
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 0.875
Best params is : {'criterion': 'entropy', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'splitter': 'random'}
```

CONFUSION MATRIX

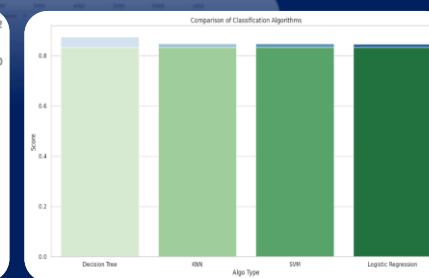
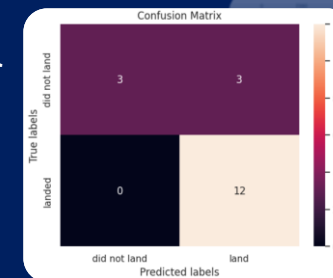
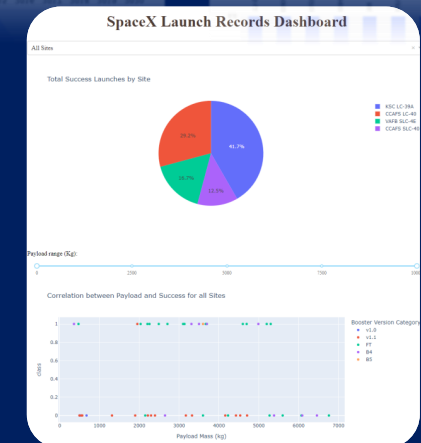
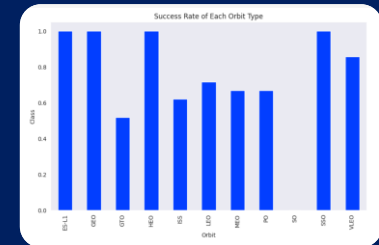


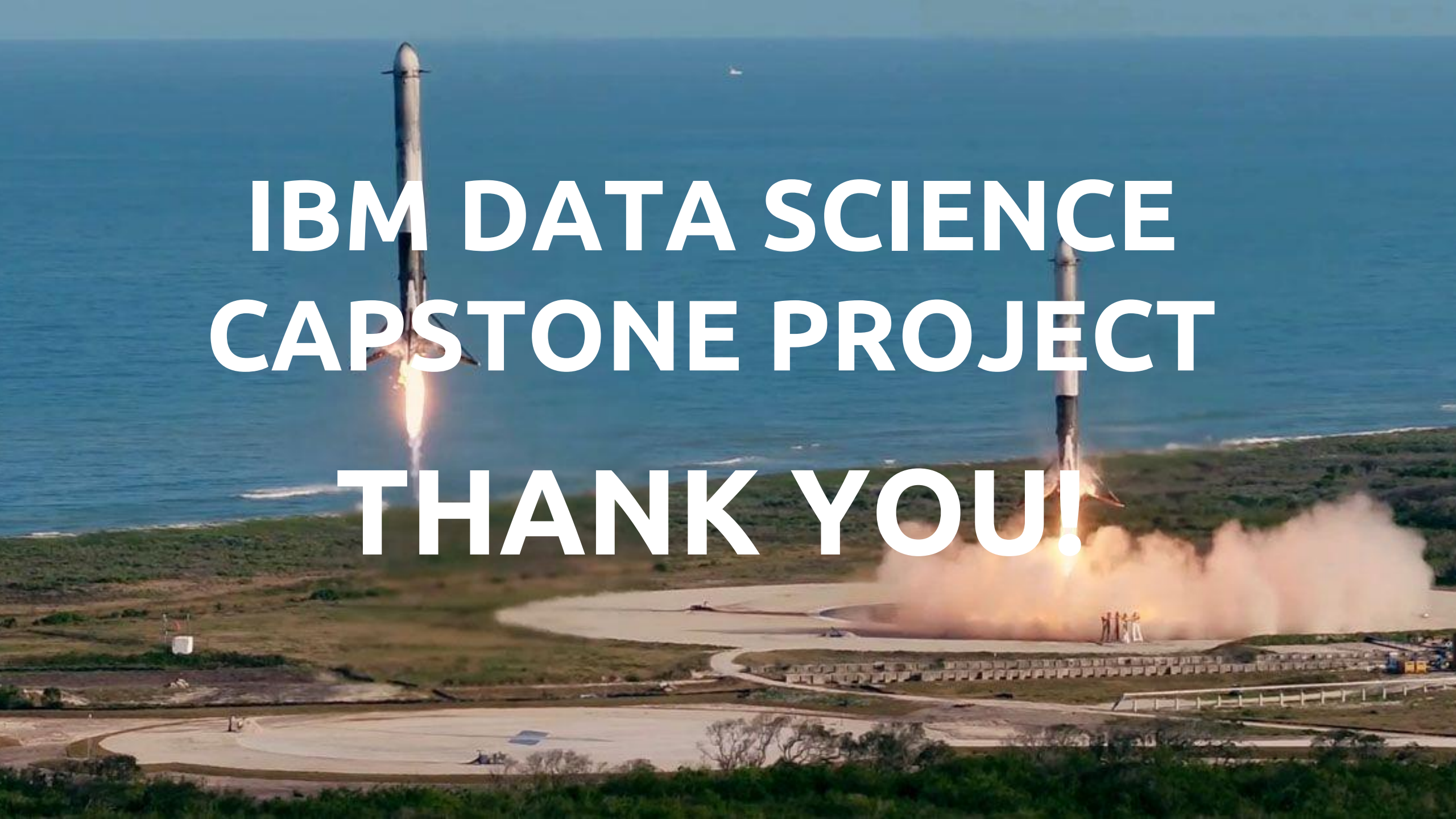
- As shown previously, best performing classification model is the **Decision Tree** model, with an accuracy of 87.5%.
- This is explained by the confusion matrix, which shows only 3 out of 18 total results classified incorrectly (a false positive, shown in the top-right corner).
- The other 15 results are correctly classified (3 did not land, 12 did land).

5. CONCLUSIONS



- As the number of flights increases, the rate of success at a launch site increases, with most early flights being unsuccessful. I.e. with more experience, the success rate increases.
 - Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
 - After 2013, the success rate generally increased, despite small dips in 2018 and 2020.
 - After 2016, there was always a greater than 50% chance of success.
- Orbit types ES-L1, GEO, HEO, and SSO, have the highest (100%) success rate.
 - The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
 - The 100% success rate in SSO is more impressive, with 5 successful flights.
 - The orbit types PO, ISS, and LEO, have more success with heavy payloads
 - VLEO (Very Low Earth Orbit) launches are associated with heavier payloads, which makes intuitive sense.
- The launch site **KSC LC-39 A** had the most successful launches, with 41.2% of the total successful launches, and also the highest rate of successful launches, with a 76.9% success rate.
- The success for massive payloads (over 4000kg) is lower than that for low payloads.
- The best performing classification model is the Decision Tree model, with an accuracy of 87.5%.



The background image shows two rockets launching from a coastal launch site. The rockets are white with black and orange accents. They are ascending vertically, leaving a trail of orange and white smoke and fire at their bases. The launch site is a flat, sandy area with some green vegetation in the foreground. In the background, there is a blue body of water under a clear sky. The text "IBM DATA SCIENCE CAPSTONE PROJECT" and "THANK YOU!" is overlaid in large, white, bold, sans-serif font across the center of the image.

**IBM DATA SCIENCE
CAPSTONE PROJECT
THANK YOU!**