IBM DATA SCIENCE CAPSTONE PROJECT

SpaceX Falcon9 Landing Predictions

Artur Azimov

CONTENTS:

EXECUTIVE SUMMARY:

SUMMARY OF METHODOLOGIES

This project consists of following stages:

- Data Collection
- Data Wrangling
- Exploratory Data Analysis
- Interactive Visual Representation of Data
- Predictive Analysis Using Classification Models

SUMMARY OF RESULTS:

The outputs of this project are:

- Exploratory Data Analysis Results
- Data Visualizations
- Geospatial Analysis
- Interactive Dashboard
- Predictive Analysis

INTRODUCTION:

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. This information can be used if an alternate company wants to bid against SpaceX for a rocket launch.

Thus, in this project we will concentrate on Falcon 9 rockets and train our model to predict if the first stage of the rocket will land successfully.

METHODOLOGY:

1. Data Collection
   - The data was collected by making GET requests to the SpaceX REST API
   - Further data was collected from open sources like Wikipedia using web scraping library in Python programming language
2. Data Wrangling
   - As we are dealing with raw data, before proceeding with any analysis we had to clean our dataset and make sure it's consistent and free of any missing and redundant data
   - To achieve this, we utilized fill_na() method to remove NaN values
   - Further we utilized value_counts() methods in order to get understanding of number of launches at each site, number and occurrence of each orbit and number and occurrence of mission outcome
   - Abovementioned techniques allowed us to analyze the results and create a landing outcome value, which was taken as 1 for successful booster landing and 0 for unsuccessful one.
3. Exploratory Data Analysis
   - During this stage we used SQL queries, to manipulate and evaluate our dataset.
   - Using Pandas and Matplotlib we visualized relationships between variables and determined patterns
4. Interactive Visual Representation of Data
   - We conducted Geospatial analysis using Folium
   - After that we utilized power of Plotly and Dash to create interactive dashboard
5. Predictive Analysis Using Classification Models
   - Using scikit-learn library we applied standardizing our data to keep all features in line
   - The data has been split into training and testing sets in order to train our models
   - We used following classification algorithms and assessed accuracy of their predictions:
     - Logistic Regression
     - Support Vector Machine
     - Decision Trees
     - K-Nearest Neighbours

DATA COLLECTION: SPACEX REST API

SEE GITHUB

Using the SpaceX API we retrieve data about launches, including information on the rockets used, payload delivered, launch specifications and landing specifications, and their outcomes.

```
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())
```

- We made a GET response to the SpaceX REST API
- Converted the response to a .json file then to a Pandas DataFrame

```
#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}
```

- Used logic to clean the data
- Defined lists for data to be stored in
- Called functions to retrieve data and fill the lists
- Used these lists as values in a dictionary and constructed the dataset

```
data_falcon9.isnull().sum()
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       5
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad       26
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

```
# Calculate the mean value of PayloadMass column
plm_mean = df["PayloadMass"].mean()

# Replace the np.nan values with its mean value
data_falcon9["PayloadMass"].replace(np.nan, plm_mean, inplace=True)
data_falcon9.isnull().sum()
```

```
FlightNumber      0
Date              0
BoosterVersion    0
PayloadMass       0
Orbit             0
LaunchSite        0
Outcome           0
Flights           0
GridFins          0
Reused            0
Legs              0
LandingPad       26
Block             0
ReusedCount       0
Serial            0
Longitude         0
Latitude          0
dtype: int64
```

- We realized that our data contains null values
- In order to solve it – we calculated the mean value and replaced the null values with mean
- We also filtered our dataset to contain only Falcon9 type rockets

DATA COLLECTION: WEB SCRAPING

SEE GITHUB

Web scraping to collect Falcon 9 historical launch records from a Wikipedia

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

- We requested the HTML page from the static URL
- Assigned the response to an object

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

- Created a BeautifulSoup object from the HTML response object
- Found all tables within the HTML page
- Collected all column header names from the tables found within the HTML page

```
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 row in first_launch_table.find_all("th"):
    name = extract_column_from_header(row)
    if name != None and len(name)>0:
        column_names.append(name)
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant 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']=[]
```

- Used the column names as keys in a dictionary
- Used functions to parse all launch tables to fill the dictionary values

```
df= pd.DataFrame({ key:pd.Series(value) for key, value in launch_dict.items() })
```

- Converted the dictionary to a Pandas DataFrame ready for export

DATA WRANGLING: PANDAS

SEE GITHUB

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

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

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

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

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

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

- The SpaceX dataset contains several Space X launch facilities, and each location is in the LaunchSite column. We used value_counts() method to see the number of launches from each site
- Each launch aims to a dedicated orbit. We applied value_counts() method to the Orbit column to see the breakdown by orbits.

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

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

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

```
landing_class = []

# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise

for outcome in df["Outcome"]:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

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

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.

To achieve this we:
Defined a set of unsuccessful (bad) outcomes
Created a list, landing_class, where the element is 0 if the outcome is in the set bad_outcome, otherwise, it's 1.
Created a Class column that contains the values from the list landing_class

EXPLORATORY DATA ANALYSIS: SQL

SEE GITHUB

We used SQL queries to reply following questions:

```
%%sql
SELECT Landing_Outcome, COUNT(Landing_Outcome) AS TOTAL_NUMBER
FROM SPACEXTBL
WHERE Date BETWEEN "2010-06-04" AND "2017-03-20"
GROUP BY Landing_Outcome
ORDER BY TOTAL_NUMBER DESC
```

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

Landing_Outcome	TOTAL_NUMBER
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

- 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

- Display 5 records where launch sites begin with the string ‘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
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

```
%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

- Display the names of the unique launch sites in the space mission

```
%%sql
SELECT MIN(Date) AS FIRST_SUCCESSFUL_LANDING_DATE
FROM SPACEXTBL
WHERE Landing_Outcome = "Success (ground pad)"
```

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

FIRST_SUCCESSFUL_LANDING_DATE
2015-12-22

- List the date when the first successful landing outcome on a ground pad was achieved

```
%%sql
SELECT SUM(PAYLOAD_MASS__KG_) AS TOTAL_PAYLOAD_MASS_NASA_CRS
FROM SPACEXTBL
WHERE Customer = "NASA (CRS)"
```

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

TOTAL_PAYLOAD_MASS_NASA_CRS
45596

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

EXPLORATORY DATA ANALYSIS: SQL

SEE GITHUB

```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_) AS AVERAGE_PAYLOAD_MASS
FROM SPACEXTBL
WHERE Booster_Version = "F9 v1.1"
```

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

AVERAGE_PAYLOAD_MASS
2928.4

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

```
%%sql
SELECT Booster_Version, PAYLOAD_MASS__KG_
FROM SPACEXTBL
WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)
```

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

Booster_Version	PAYLOAD_MASS__KG_
F9 B5 B1048.4	15600
F9 B5 B1049.4	15600
F9 B5 B1051.3	15600
F9 B5 B1056.4	15600
F9 B5 B1048.5	15600
F9 B5 B1051.4	15600
F9 B5 B1049.5	15600
F9 B5 B1060.2	15600
F9 B5 B1058.3	15600
F9 B5 B1051.6	15600
F9 B5 B1060.3	15600
F9 B5 B1049.7	15600

- List the names of the booster versions which have carried the maximum payload mass

```
%%sql
SELECT TRIM(Mission_Outcome), COUNT(Mission_Outcome) AS TOTAL_NUMBER
FROM SPACEXTBL
GROUP BY TRIM(Mission_Outcome)
```

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

TRIM(Mission_Outcome)	TOTAL_NUMBER
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

- List the total number of successful and failed mission outcomes

```
%%sql
SELECT Booster_Version
FROM SPACEXTBL
WHERE (Landing_Outcome = "Success (drone ship)") AND (PAYLOAD_MASS__KG_ BETWEEN 4000 and 6000)
```

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

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

- List the names of the boosters which had success on a drone ship and a payload mass between 4000 and 6000 kg

```
%%sql
SELECT SUBSTR(DATE,6,2) AS MONTH, Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTBL
WHERE SUBSTR(Date,0,5) = "2015" AND Landing_Outcome = "Failure (drone ship)"
```

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

MONTH	Landing_Outcome	Booster_Version	Launch_Site
01	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
04	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

- List the failed landing outcomes on drone ships, their booster versions, and launch site names for 2015

EXPLORATORY DATA ANALYSIS: VISUALIZATION

SEE GITHUB

In this section we have built following vizualizations:

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

  Thus, we were able to make conclusions regarding correlation between numeric variables
- Bar Chart:
  - Success Rate and Orbit Type

  Thus, we were able to see the success rate based on a categorical variable
- Line Chart:
  - Success Rate and Year (i.e. the launch success yearly trend)

  Thus, we were able to assess change in success rate over time

GEOSPATIAL ANALYSIS: FOLIUM

SEE GITHUB

Following steps have been performed in order to make an interactive visualization:

1. We marked all launch sites on a map by:
   - Initialising the map using a Folium Map object
   - Adding a folium.Circle and folium.Marker for each launch site on the launch map
2. We marked the successful/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, we assigned a marker colour of successful (class = 1) as green, and failed (class = 0) as red
   - To put the launches into clusters, for each launch, we added a folium.Marker to the MarkerCluster() object
   - Finally, we created an icon as a text label, assigning the icon_color as the marker_colour
3. We calculated the distances between a launch site and its proximities:
   - To explore the proximities of launch sites, we have made calculations of distances between points using the Lat and Long values.
   - After marking a point using the Lat and Long values, we created a folium.Marker object to show the distance.
   - To display the distance line between two points, we drew a folium.PolyLine and added it to the map.

SEE GITHUB

INTERACTIVE VISUAL REPRESENTATION OF DATA: DASH

We have added the following plots to a Plotly Dash dashboard to have an interactive visualization of the data:

- 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

- Scatter graph (px.scatter()) to show the correlation between outcome (success or fail) 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

SEE GITHUB

We utilized the power of Scikit-Learn Python library in the following steps:

- We prepared the dataset for model development:
  - Loaded the dataset
  - Performed necessary data standardizing to keep all features in line
  - Performed split of the data into training and test data sets, using train_test_split()
  - Tested several machine learning algorithms and estimated their accuracy:
    - Logistic Regression
    - Support Vector Machine
    - Decision Trees
    - K-Nearest Neighbours

For each chosen algorithm:

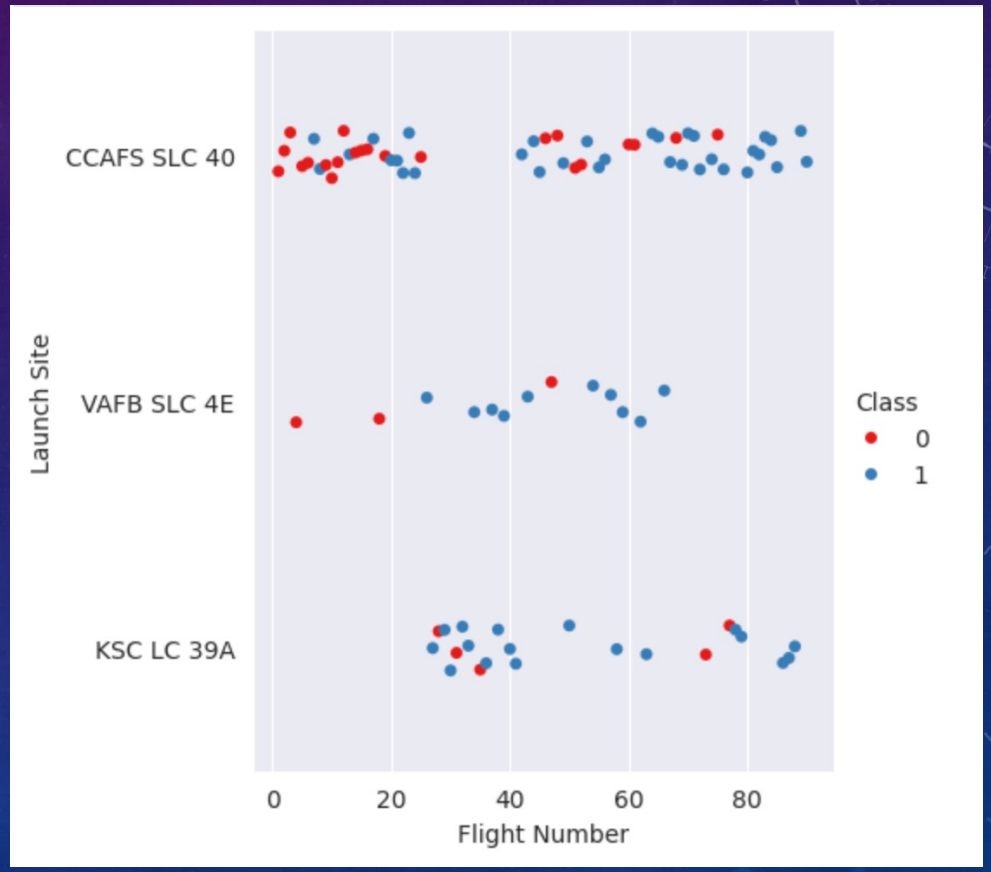
- ✓ We created a GridSearchCV object and a dictionary of parameters
- ✓ Fitted the object to the parameters
- ✓ Finally, we used the training data set to train the model and test data to check it's performance

RESULTS:
EXPLORATORY DATA ANALYSIS

FLIGHT NUMBER VS LAUNCH SITE

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

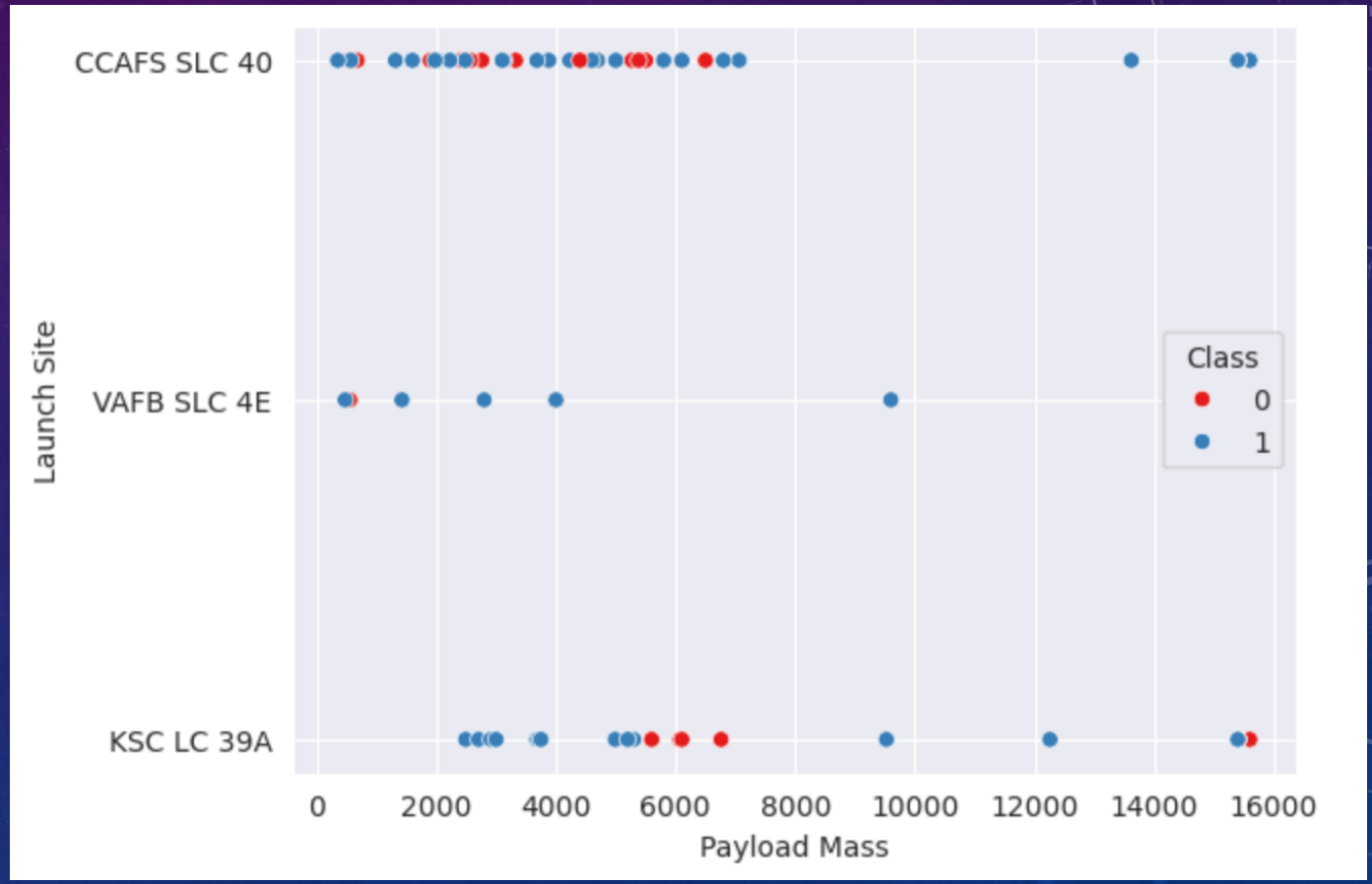
- As the number of flights increases, the rate of success at a launch site increases.
- Most of the early flights (flight numbers < 30) were launched from CCAFS SLC 40, and were mostly unsuccessful.
- No early flights were launched from KSC LC 39A, so the launches from this site are more successful.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



PAYLOAD MASS VS LAUNCH SITE

The scatter plot of Payload Mass vs Launch Site 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.
- It seems like there is no clear correlation between payload mass and success rate.
- All sites launched a variety of payload masses, with most of the launches from CCAFS SLC 40 being lighter payloads.



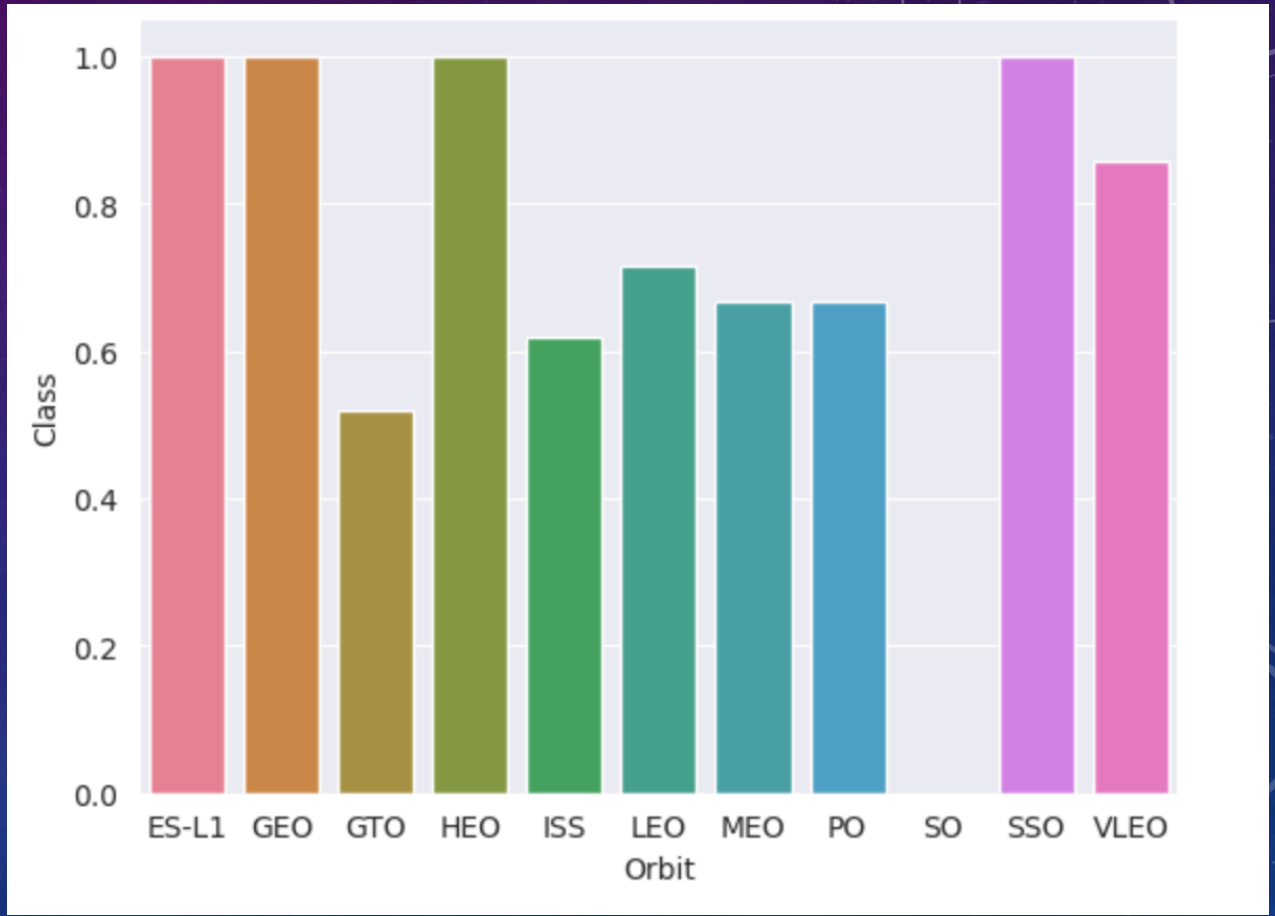
ORBIT TYPE VS SUCCESS RATE

This bar chart 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:

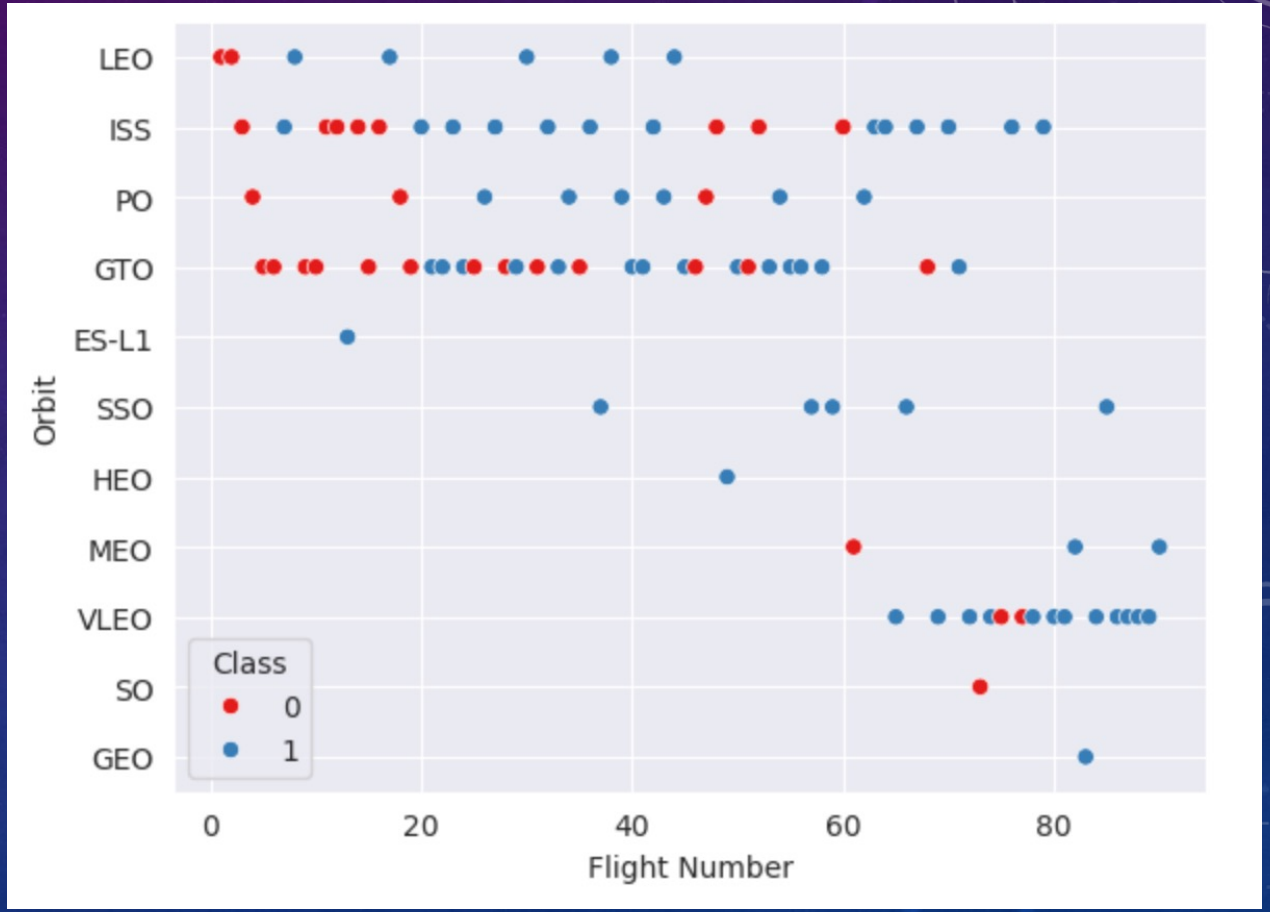
- SO (Heliocentric Orbit)



FLIGHT NUMBER VS ORBIT TYPE

This scatter plot shows a few useful things that the previous plots did not, such as:

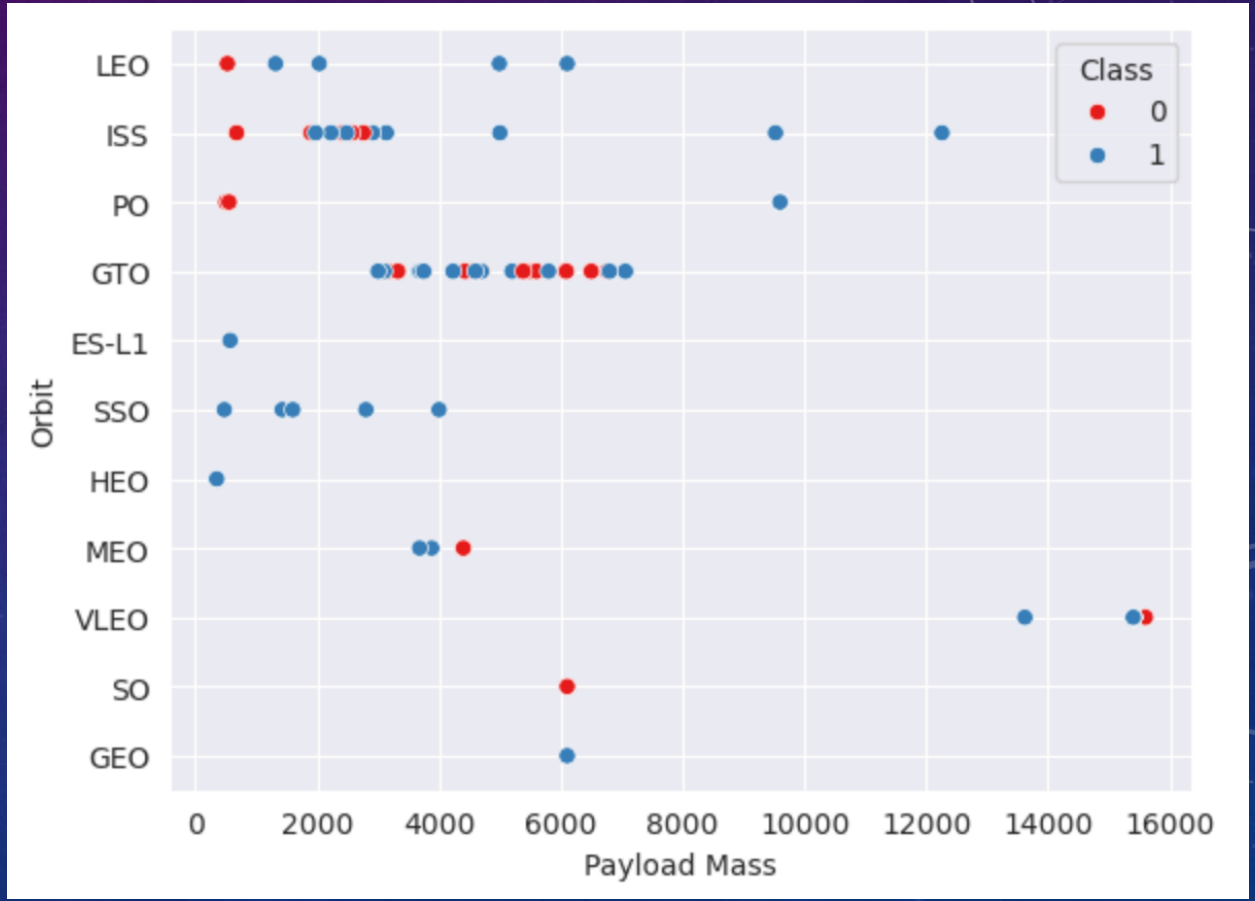
- The 100% success rate of GEO, HEO, and ES-L1 orbits can be explained by only having 1 flight into the respective orbits.
- The 100% success rate in SSO is more impressive, with 5 successful flights.
- There is weak relationship between Flight Number and Success Rate for GTO.
- Generally, as Flight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



PAYLOAD MASS VS ORBIT TYPE

This scatter plot shows that:

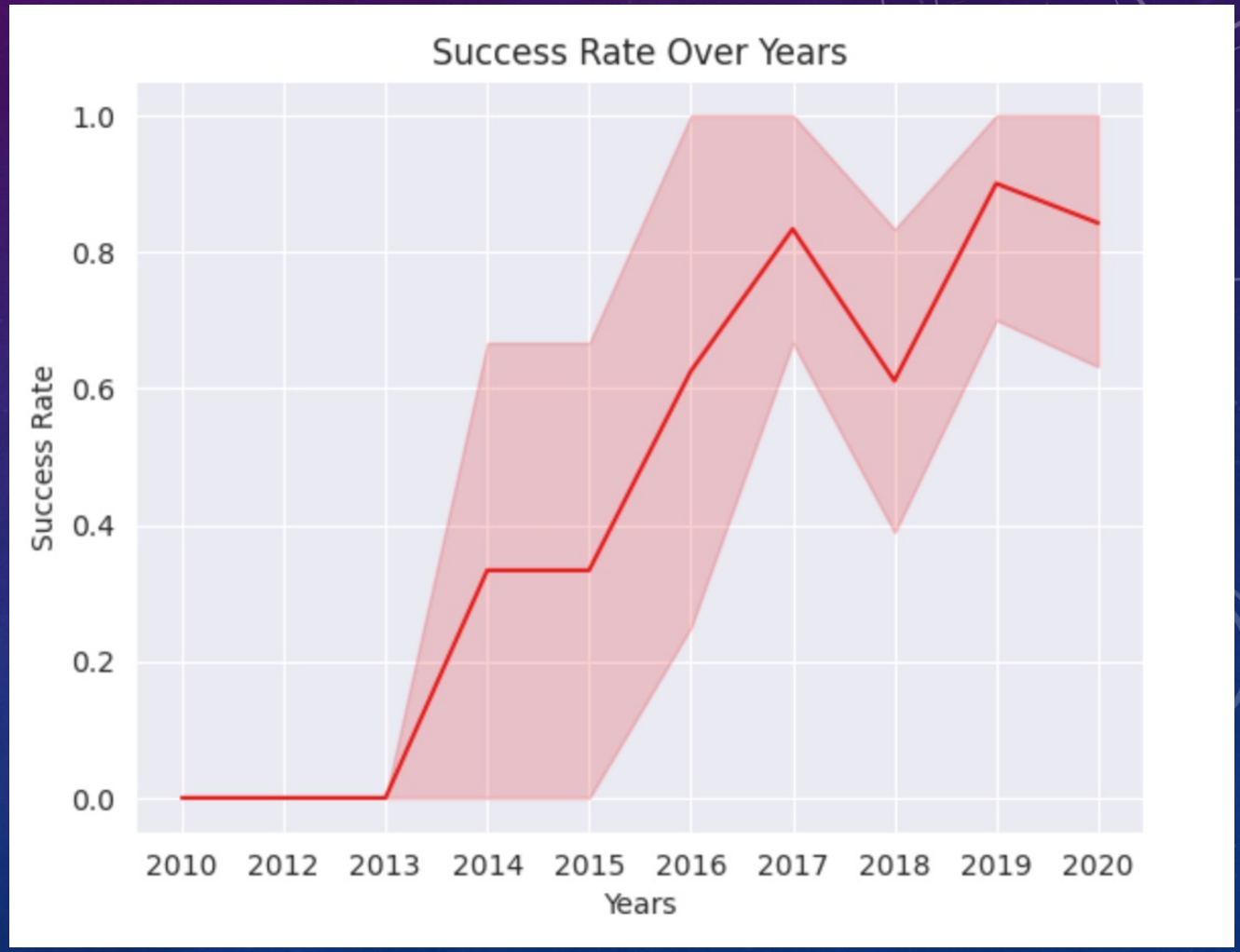
- The following orbit types have more success with heavy payloads:
  - PO (although the number of data points is small)
  - ISS
  - LEO
- For GTO, the relationship between payload mass and success rate is unclear.



LAUNCH SUCCESS YEARLY TREND

The line chart of average success rate by year shows that:

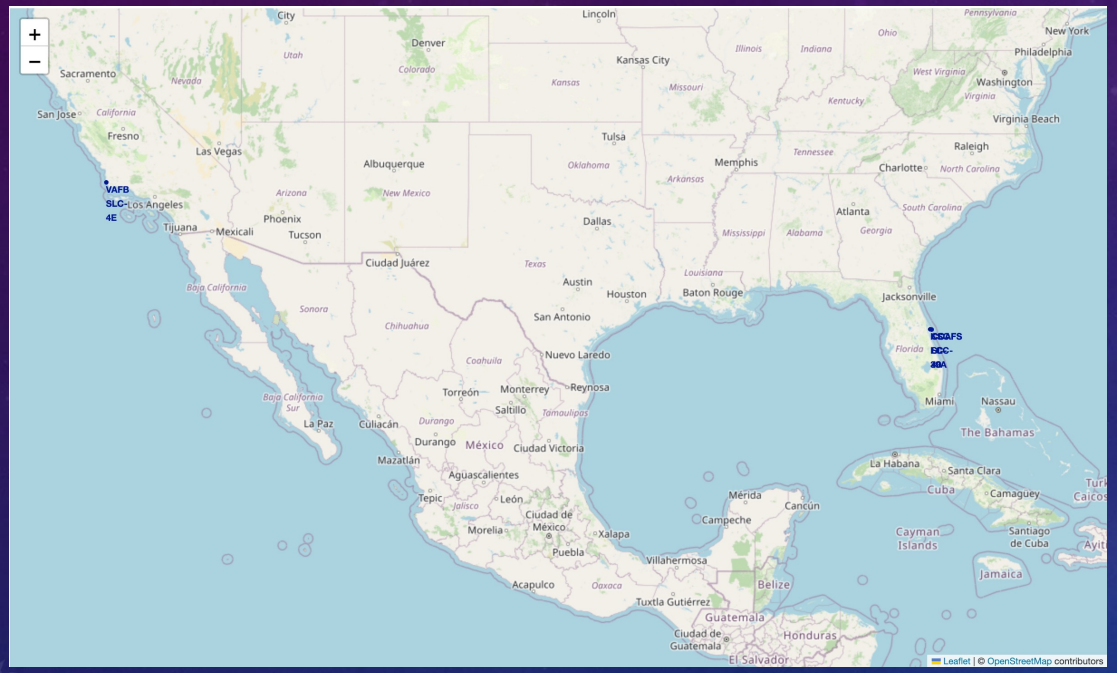
- Between 2010 and 2013, all landings were unsuccessful (as the success rate is 0).
- After 2013, the success rate increased, despite small dips in 2018 and 2020.
- In general, after 2016 there was always a greater than 50% chance of success.

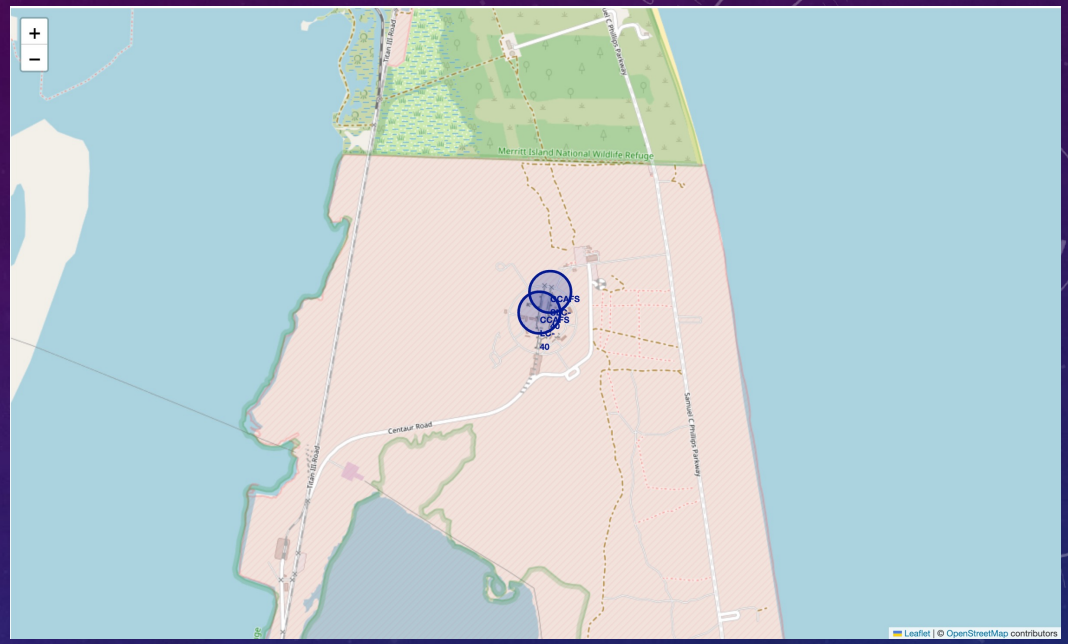


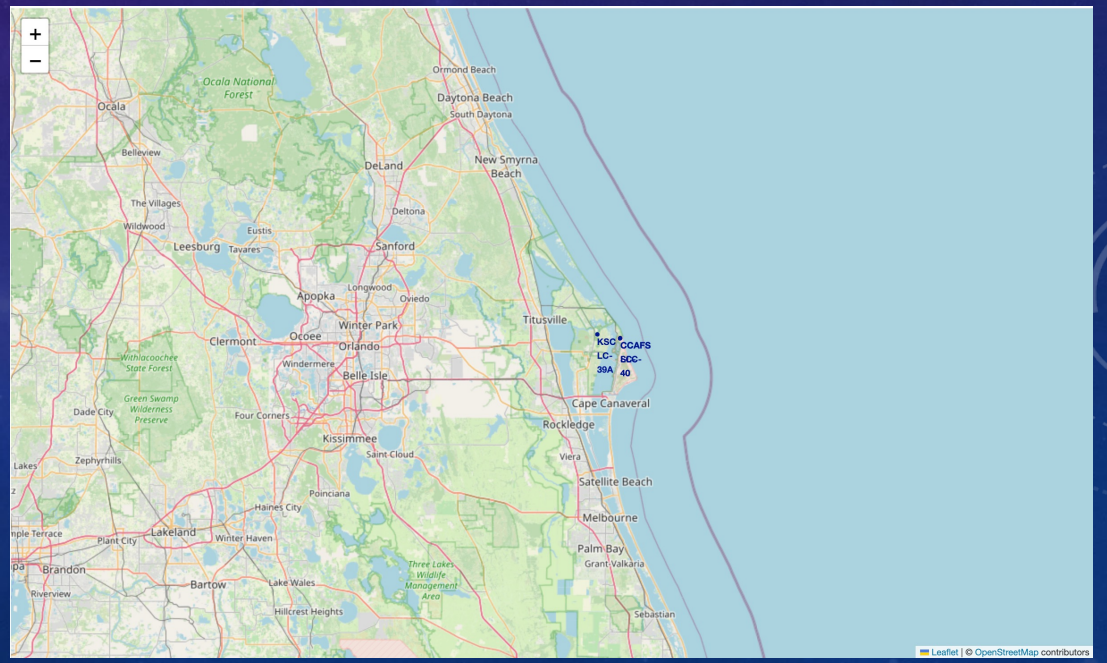
LAUNCH SITES PROXIMITY ANALYSIS FOLIUM INTERACTIVE MAP

LAUNCH SITES LOCATIONS

SpaceX launch sites are located on coasts of the USA, namely Florida and California





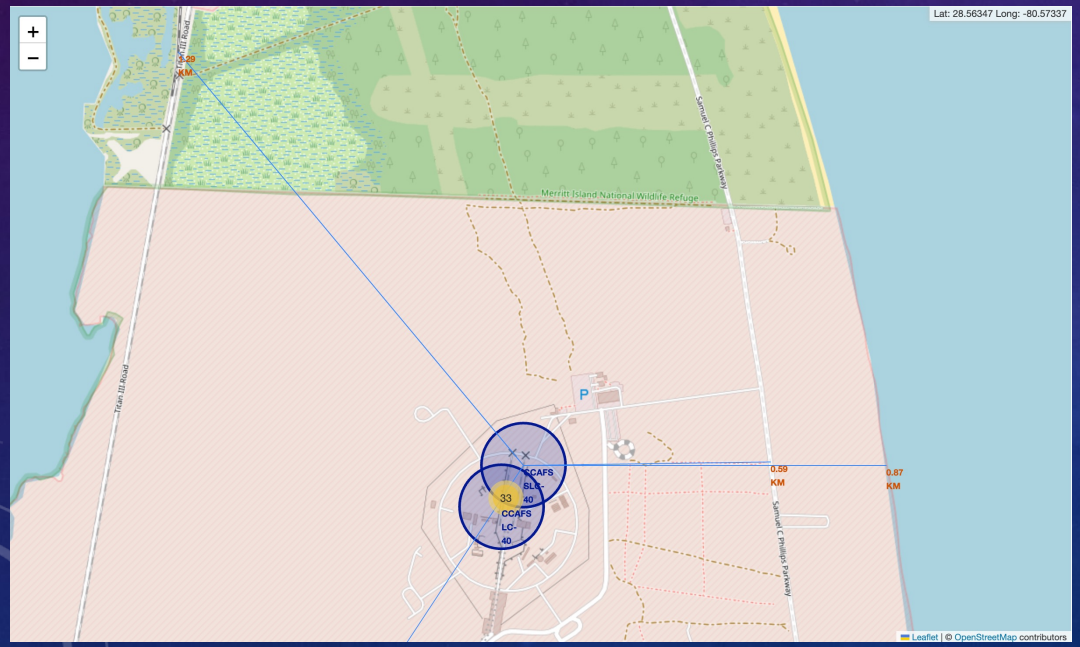


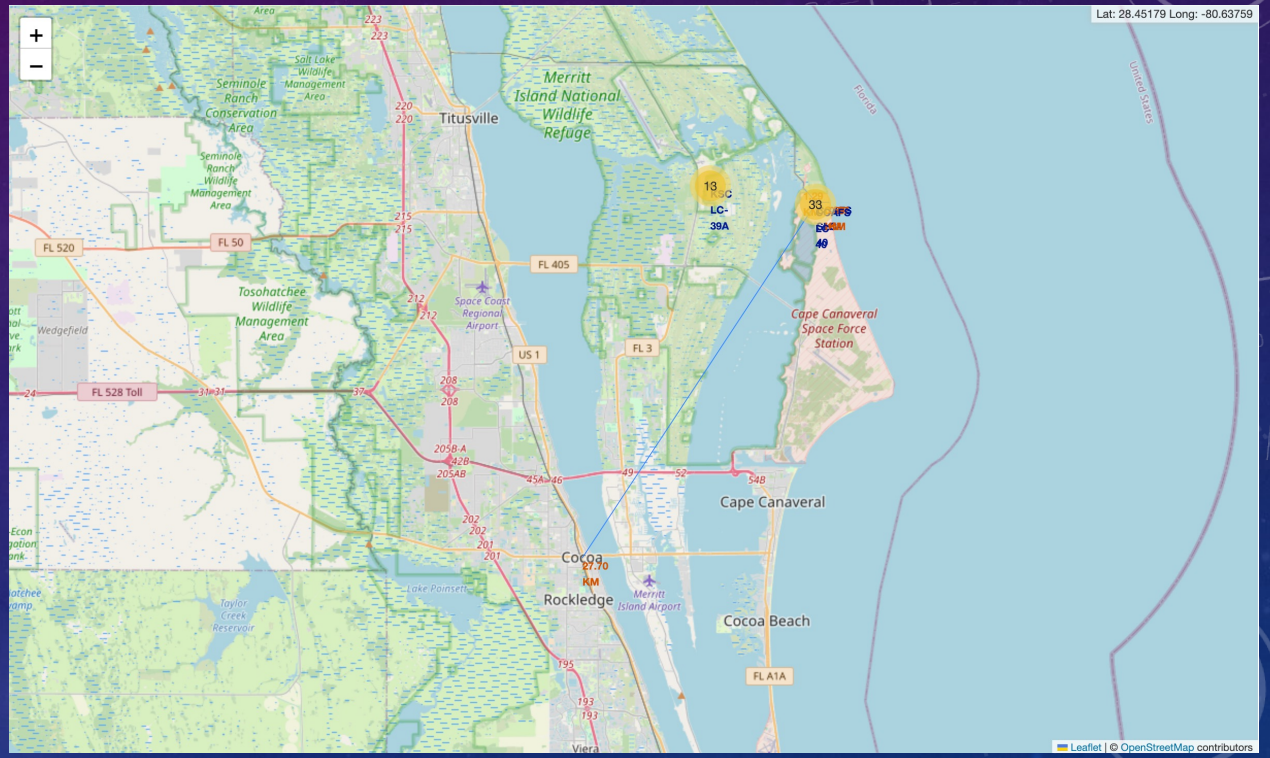
SUCCESSFUL/FAILED LAUNCHES PER SITE

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

PROXIMITY OF LAUNCH SITES TO CITIES, RAILWAYS, HIGHWAYS

As we can see, the launch sites are in a quite close proximity to highways (0.59 km), railways (1.29 km), cities (27.7 km)

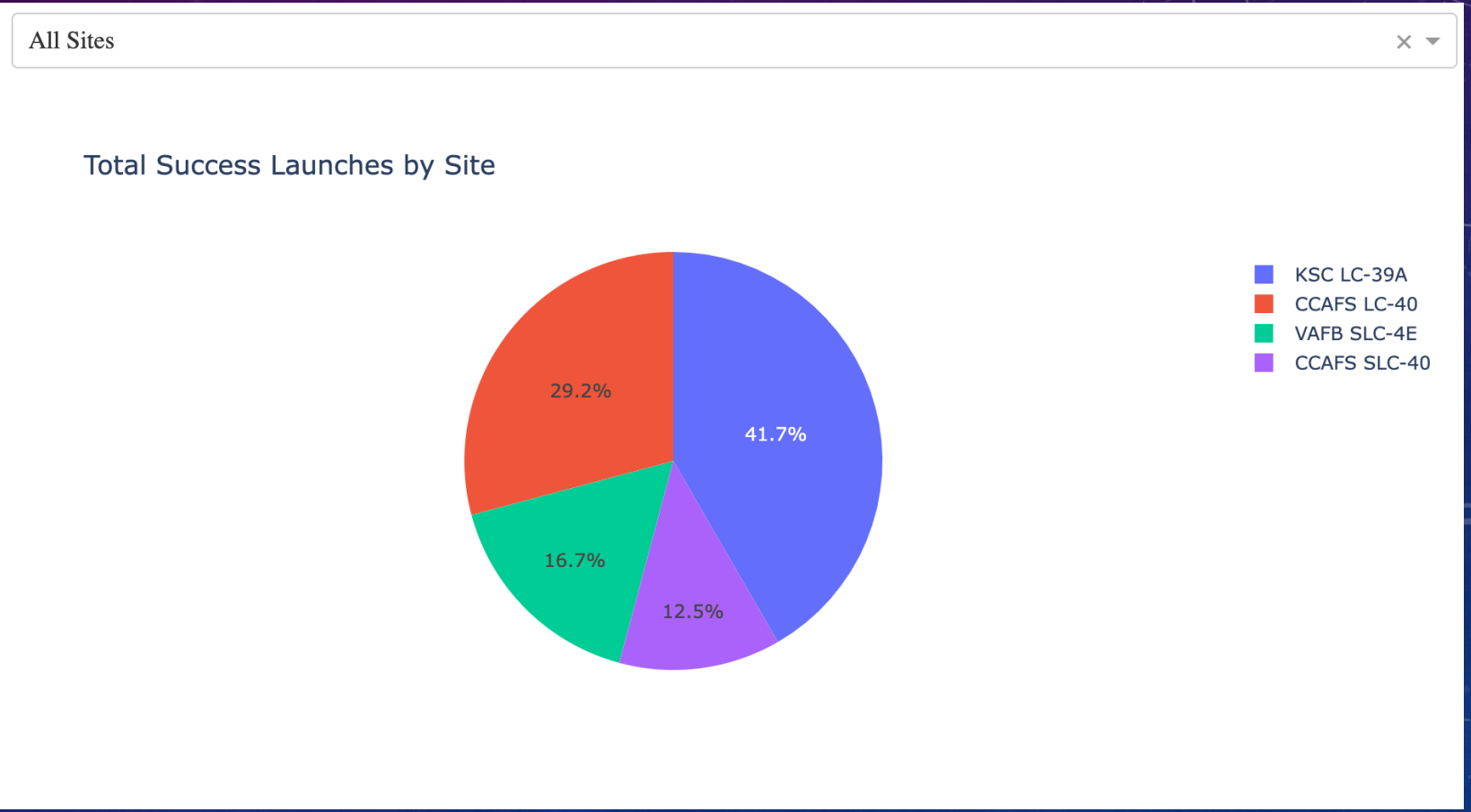




INTERACTIVE DASHBOARD PLOTLY DASH

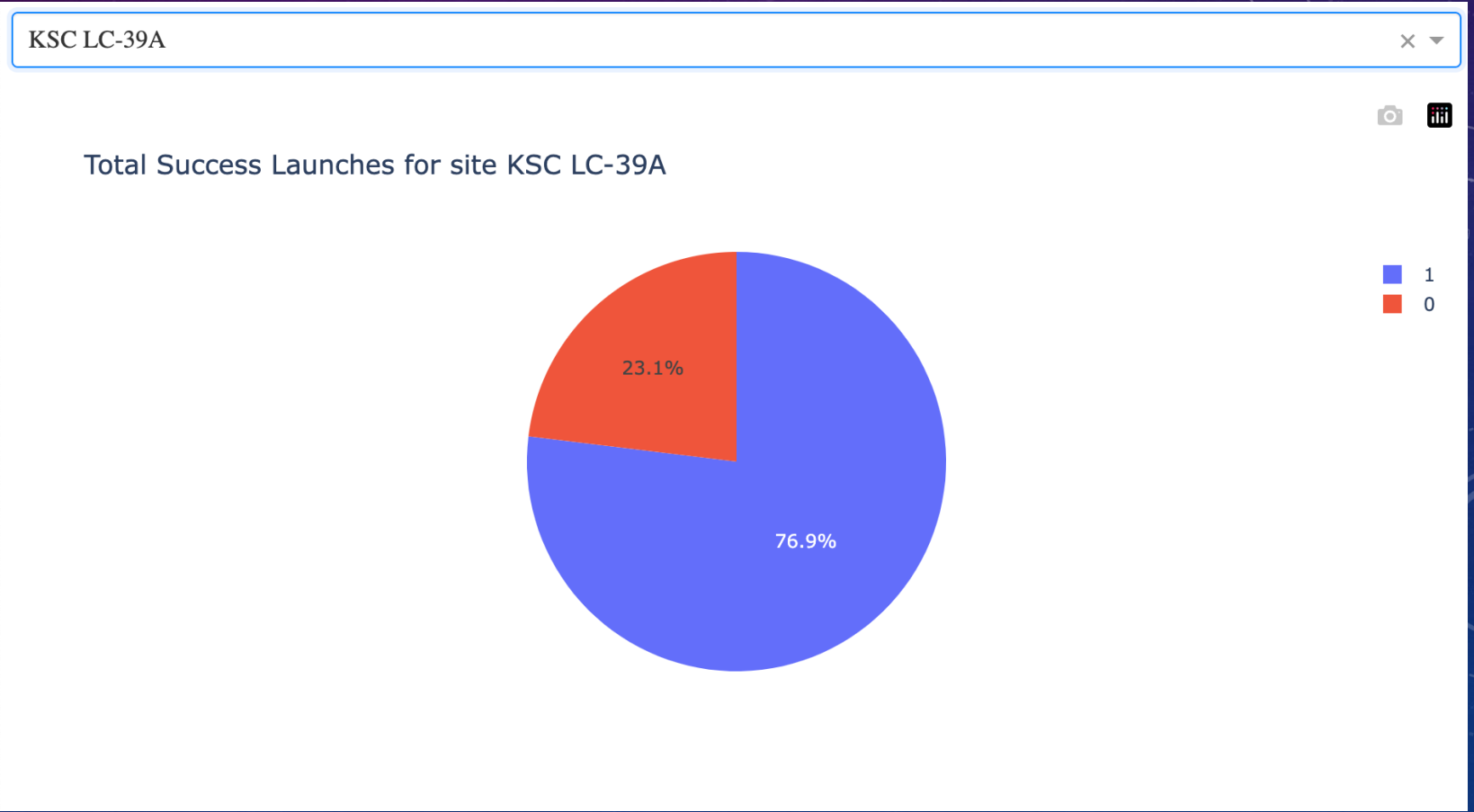
LAUNCH SUCCESS FOR ALL SITES:

Here we see that the launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches.



PIE CHART FOR THE SITE WITH THE HIGHEST LAUNCH SUCCESS RATE

Here we see that the launch site KSC LC-39A also has the highest rate of successful launches, with a 76.9% success rate.

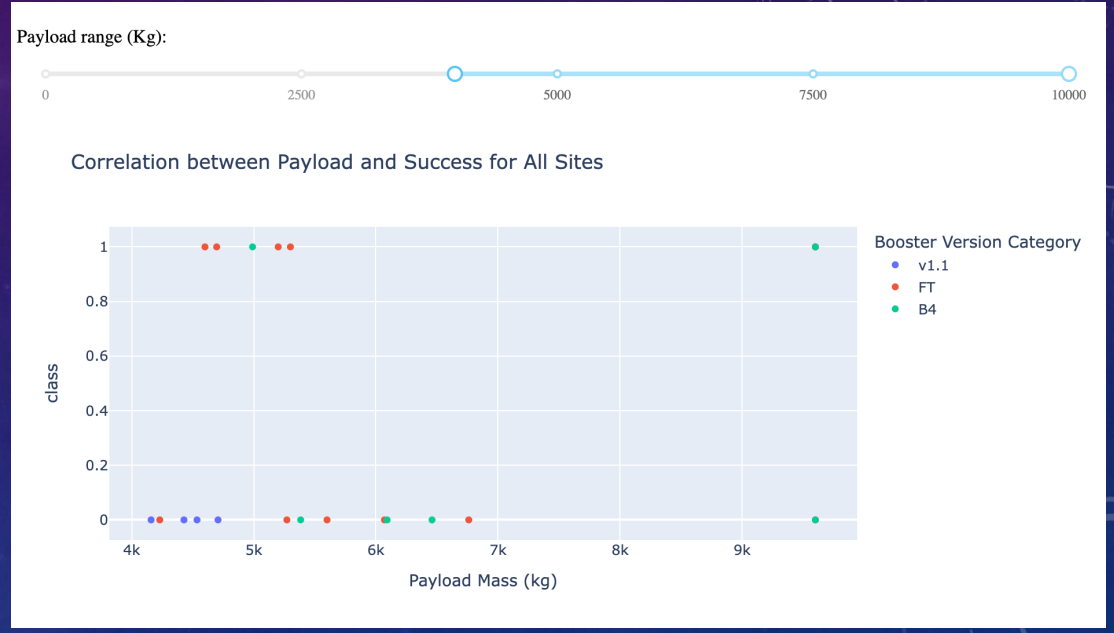


PAYLOAD MASS VS LAUNCH OUTCOME SCATTER PLOT FOR ALL SITES

Plotting the payload mass vs. launch outcome for all sites shows a gap around 4000 kg, so let's to split the data into 2 ranges:

- 0 – 4000 kg (low payloads)
- 4000 – 10000 kg (massive payloads)

PAYLOAD MASS VS LAUNCH OUTCOME SCATTER PLOT FOR ALL SITES



- From these 2 plots, it's seen 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 USING CLASSIFICATION MODELS

ALGORITHMS ACCURACY

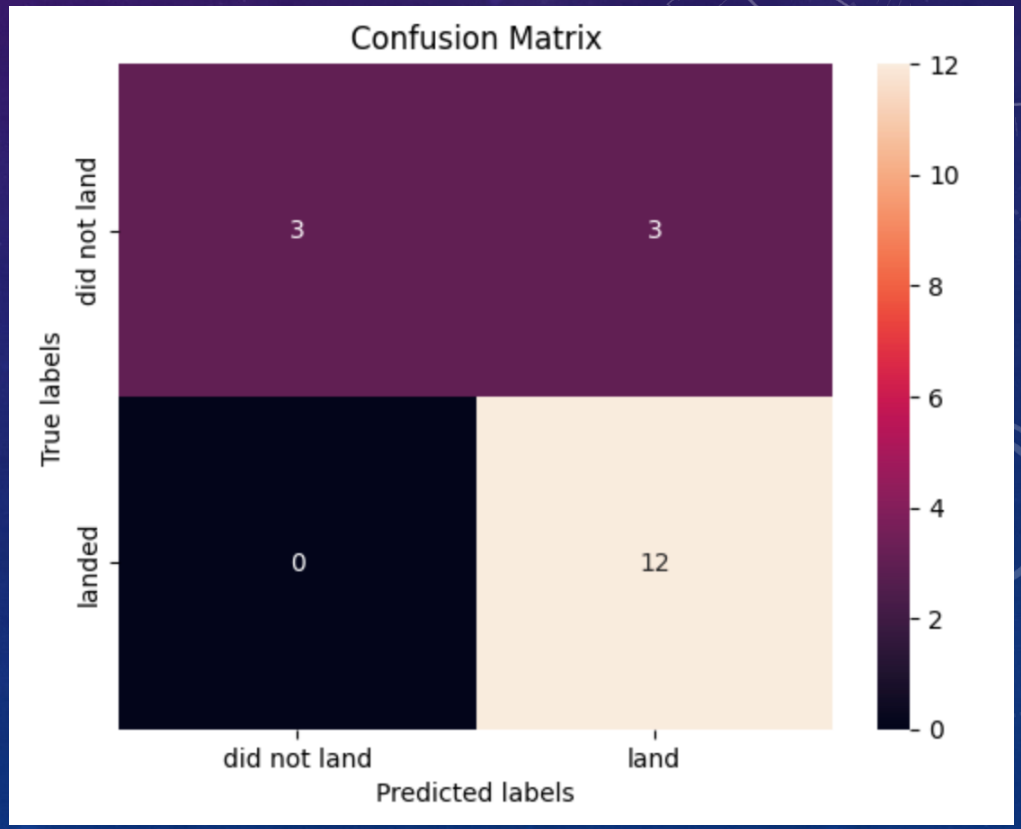
In this project we used following algorithms:

- Logistic Regression
- Support Vector Machine
- Decision Trees
- K-Nearest Neighbours

The resulting accuracy is following:

	Algorithm	Accuracy Score	Best Score
0	Logistic Regression	0.833333	0.846429
1	Support Vector Machines	0.833333	0.848214
2	Decision Trees	0.833333	0.875000
3	K-Nearest Neighboors	0.833333	0.848214

The Confusion Matrix can be seen below:



CONCLUSION

CONCLUSION:

- 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 launches are associated with heavier payloads, which makes intuitive sense.
- The launch site KSC LC-39 A had the most successful launches, with 41.7% 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 smaller payloads.
- All models showed pretty much the same accuracy, thus there is no distinctive difference.

APPENDIX

You can find the full project at: GITHUB