
# IBM Data Science Capstone Project - SpaceX

OLADIPUPO TAOFIK LATONA

11/04/2022

# EXECUTIVE SUMMARY

- PROJECT METHODS
  - Data Collection
  - Data Wrangling
  - Exploratory Data Analysis with Visualization
  - Exploratory Data Analysis with SQL
  - Interactive Map with Folium
  - Dashboard with Plotly Dash
  - Predictive Analysis (Classification)
- PROJECT RESULTS
  - Results of the Exploratory Data Analysis
  - Screenshot of Interactive Dashboard
  - Results of Predictive Analysis

# TABLE OF CONTENTS

# 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.

Questions for Analysis

What are the features/variables that affect the launch success rate of the rocket?

What are the effects of each variable on the launch outcome ?

What factors are involved in finding an optimal location for building a launch site?

# Methodology

# METHODOLOGY

- Data Collection Methodology
  - SpaceX API
  - Web scrapping from Wikipedia
- Data Wrangling Methodology
  - One-hot encoding was applied to categorical features
- EDA with Visualization & SQL Methodology
- Visual Analytics using Folium and Dash Methodology
- Predictive Analysis Methodology
  - Classification

# Data Collection – SpaceX API

- In this project, SpaceX launch data gathered specifically from the SpaceX REST API was used
- This API provided data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications and landing outcome
- The goal is to use this data to predict whether SpaceX will attempt to land a rocket or not
- The API endpoint/URL used for this is: api.spacexdata.com/v4/launches/past

How it works:

Link to Notebook

# Data Collection – SpaceX API

1. Requesting launch data from SpaceX API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
response = requests.get(spacex_url)
```

2. Decoding the response content as a Json using .json() and turn it into a Pandas data frame using .json_normalize()

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

3. Creating lists for new dataframe

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

Link to Notebook

# Data Collection - SpaceX API

4. Assigning lists to a dictionary then creating pandas dataframe from said dictionary

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

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

Link to Notebook

5. Filter dataframe to only include Falcon9 launches then export to flat file (.csv)

```
data_falcon9 = launch_data[launch_data['BoosterVersion']!='Falcon 1']
data_falcon9
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

# Data Collection - WebScrapping

- In this project, Python's BeautifulSoup package is used to web scrape some HTML tables that valuable Falcon 9 launch records
- The Wikipedia page used can be found at: https://en.wikipedia.org/wiki/List_of_Falcon_9_and_Falcon_Heavy_launches
- The data will then be parsed from those tables and converted into a Pandas dataframe for further analysis and visualization

How it works:

Link to Notebook

# Data Collection - WebScrapping

1. Using the HTTP GET method to request the Falcon 9 launch HTML page as a HTTP response

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

```
r = requests.get(static_url).text
```

2. Creating a BeautifulSoup object from the HTML response

```
soup = BeautifulSoup(r,"html.parser")
```

3. Finding all tables from the Wiki page

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

4. Iterating through the <th> elements and applying a custom function to extract all column names

```
column_names = []

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

Link to Notebook

# Data Collection - WebScrapping

5. Creating an empty dictionary with keys from the extracted column names

```
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']=[]
```

6. Appending data to keys ( see block 82 of notebook)

7. Converting dictionary to dataframe then export it to a csv

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

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

Link to Notebook

# Data Wrangling

In this section, Exploratory Data Analysis was performed in order to find patterns in the data and determine what the label for training supervised models would be. Furthermore, there are several different cases in the dataset where the booster did not land successfully. Sometimes a landing was attempted but failed due to an accident. The different cases include:

- **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 on a drone ship
- **False ASDS** - the mission outcome was unsuccessfully landed on a drone ship

Therefore, these outcomes were converted into Training Labels with 1 meaning the booster successfully landed and 0 meaning it was unsuccessful.

# Data Wrangling – Process

1. Calculate the number of launches on each site

```
# 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
```

2. Calculate the number and occurrence of each orbit ( Each launch aims to a dedicated orbit)

```
# 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
```

Link to Notebook

# Data Wrangling - Process

3. Calculate the number and occurrence of mission outcome per orbit type

```
# 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
```

4. Create a landing outcome label from the Outcome column

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for outcome in df['Outcome']:
    if outcome in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

Link to Notebook

# EDA with Visualization

Link to Notebook

In this section, the data was explored by visualizing the relationship between variables. These relationships were visualized using Scatter Graphs, Bar Graphs and Line Graphs.

**Scatter Graphs:**

- Flight Number vs Payload Mass
- Flight Number vs Launch Site
- Payload Mass vs Launch Site
- Orbit Type vs Flight Number
- Payload Mass vs Orbit Type
- Orbit Type vs Payload Mass

A Scatter graph shows the effect of one variable on another. Their relationship is called a correlation. Scatter graphs typically consists of a large body of data.

**Bar Graph:**

- Mean Success Rate vs Orbit Type

A Bar Graph makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value on the other with the goal being to show the relationship between the two axes. Bar Charts also show big changes in data over time

**Line Graph:**

- Success Rate vs Year

The main use of a line graph is to show data variables and trends very clearly and also help to make predictions about results that are not yet recorded

# EDA with SQL

In this section, Exploratory Data Analysis was done using SQL in order to learn more about the data. The data was loaded into the IBM DB2 database and multiple queries were ran to perform the following tasks:

- Display the names of the unique launch sites in the space mission
- Display 5 records where launch sites begin with the string 'CCA'
- Display the total payload mass carried by boosters launched by NASA (CRS)
- Display average payload mass carried by booster version F9 v1.1
- List the date when the first successful landing outcome in ground pad was achieved
- List the names of the boosters which have success in drone ship and have payload mass greater than
- 4000 but less than 6000
- List the total number of successful and failure mission outcomes
- List the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- List the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015
- Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the
- date 2010-06-04 and 2017-03-20, in descending order

Link to Notebook

# Interactive Map with Folium

Launch success rate can also depend on the location and proximities of a launch site i.e. the initial position of rocket trajectories. Therefore, in order to find an optimal location for building a launch site, some factors were discovered through the analysis of existing launch site locations.

The process of achieving this include:

- Marking all launch sites on a map using the latitude and longitude coordinates. Circle markers were also added on the map to highlight launch site area
- Marking every successful and failed launches for each site on the map. Green colour markers indicate a successful launch while a red colour marker indicates an unsuccessful launch
- Calculating the distances between a launch site and its proximities. This was done using Haversine's formula and by drawing a polyline from a launch site to selected proximities.

After lines to proximities were plotted, some questions were asked. These include:

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

Link to Notebook

# Dashboard with Plotly Dash

- An interactive dashboard was built using Plotly Dash
- A Pie chart capable of showing total launches by all sites and also total launches by a certain site was created
- A scatter graph showing the relationship between the Outcome and Payload Mass (Kg) for different booster versions
- A range slider was also included to values within the minimum and maximum Payload Mass (kg)

Link to Notebook

# Predictive Analysis (Classification)

A Machine Learning pipeline was created to predict if the first stage will land given all the acquired data. In order to do this, the following process was followed:

- The data was first loaded into pandas and NumPy, standardized and then split into training and testing set in the ratio 80:20
- A machine learning model object will then be created alongside a GridSearchCV object in order to fit our training dataset
- The best parameters and accuracy on the validation data are displayed using the attributes " best_params_" and "best_score_"
- The accuracy of the test data for each model is also calculated using the "score" method and a confusion matrix is also plotted
- The model with the highest accuracy score will be classified as the best performing model

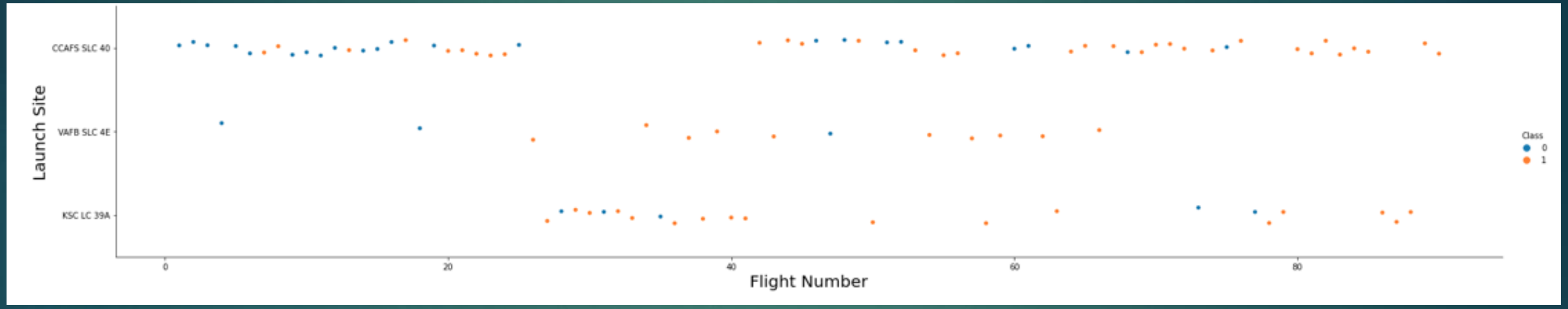
Link to Notebook

# Results

- Exploratory Data Analysis results
- Interactive Analysis Demo in screenshots
- Predictive Analysis results

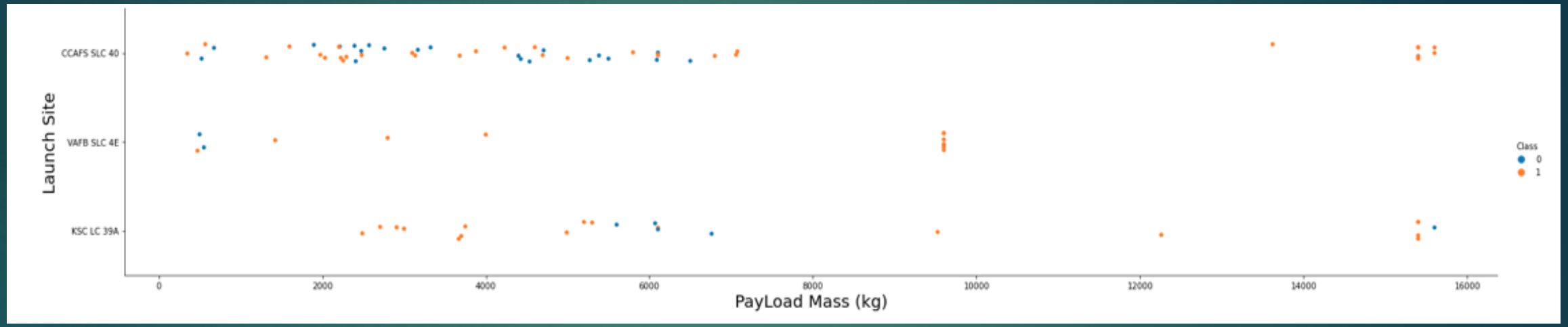
# Results from EDA with Visualization

# Flight Number vs. Launch Site



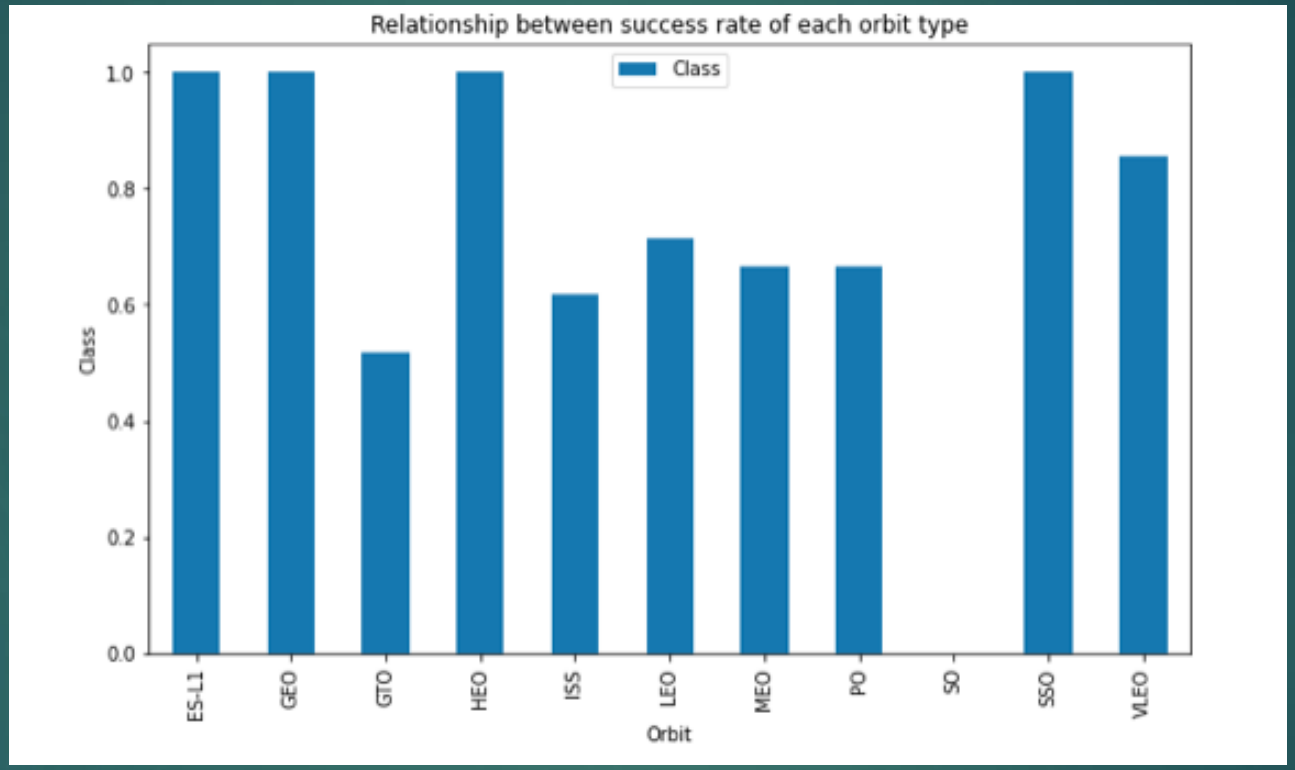

It was found that the higher the number of flights, the higher the success rate at a launch site.

# Payload Mass (Kg) vs. Launch Site



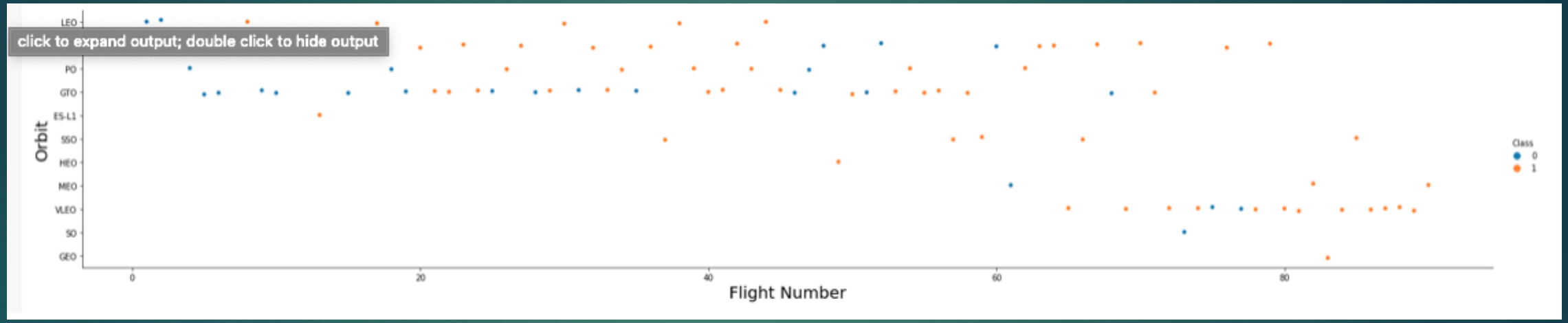

It can be observed that for the CCAFS SLC 40 launch site, the higher the mass of the payload, the higher the success rate.

# Success Rate vs. Orbit Type



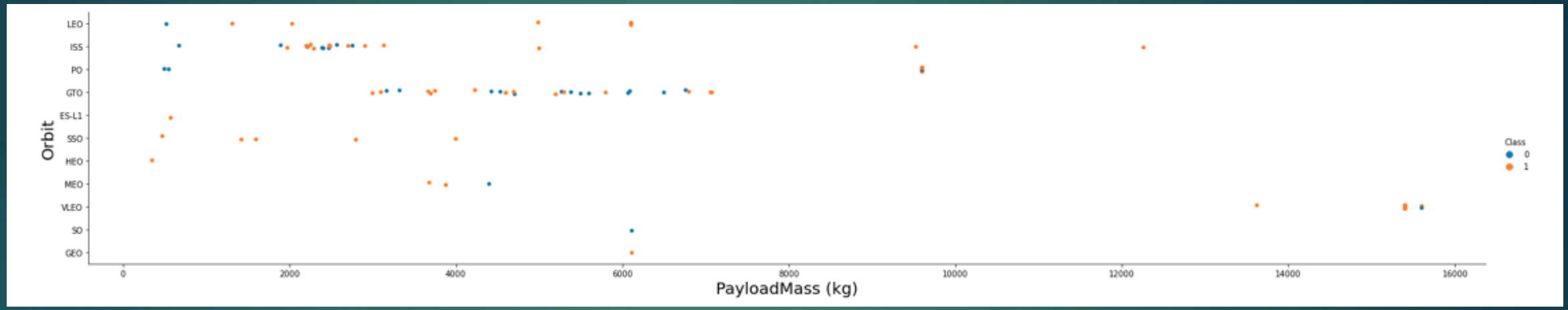

From the chart, we can see that orbit types ES-L1, GEO, HEO and SSO had the highest success rates.

# Flight Number vs. Orbit Type



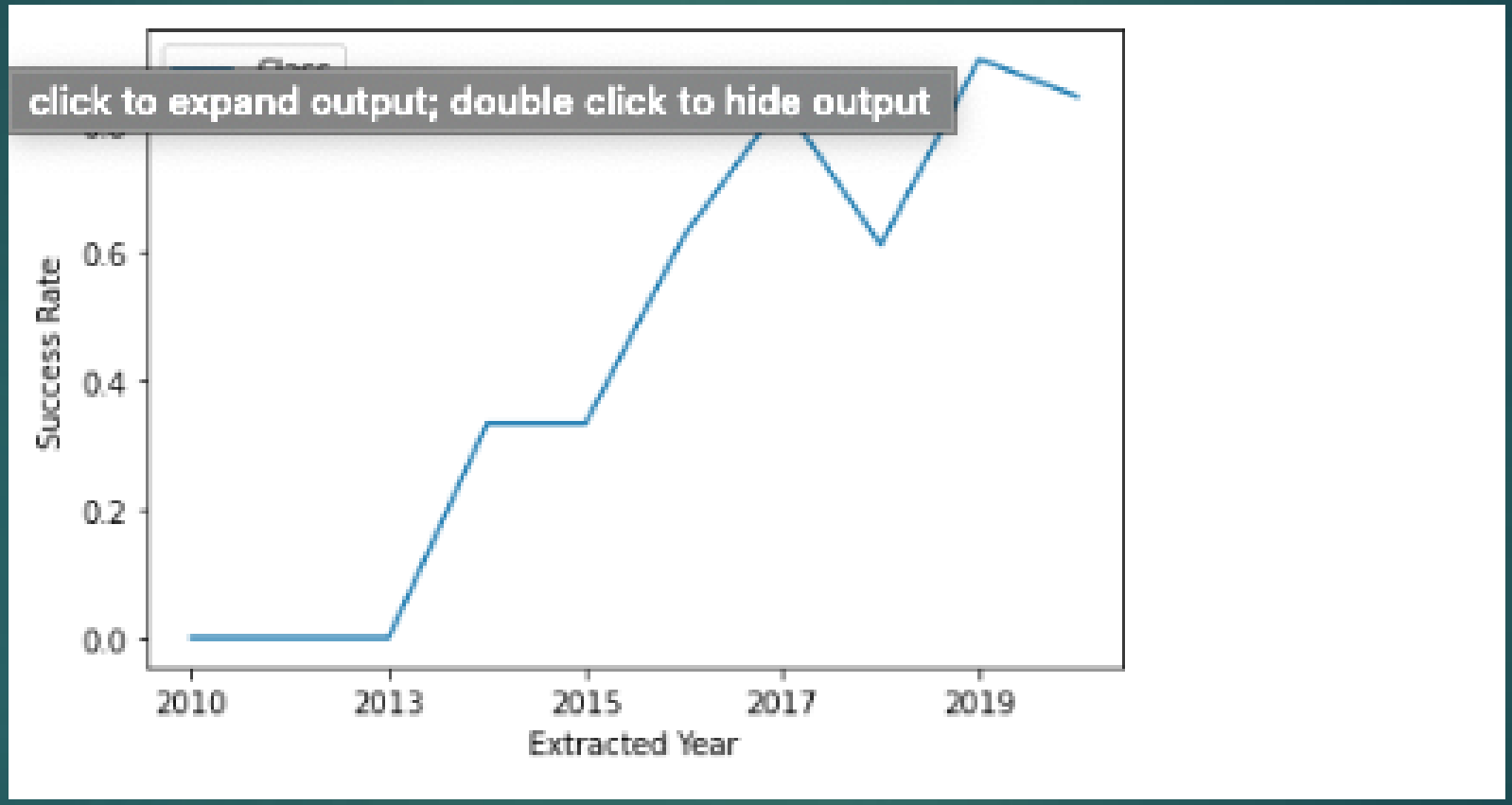

From the plot, it can be observed that for the LEO orbit , the success rate appears related to the number of flights whereas there seems to be no relationship between the GTO orbit and flight number

# Payload Mass (Kg) vs. Orbit Type




It can be observed that with heavy payloads, the successful landing rate is more for Polar, LEO and ISS. However this cannot be distinguished for GTO as both successful and unsuccessful landings are equally present

# Launch Success Yearly Trend




It can be observed that from 2013, the success rate kept increasing till 2020

# Results from EDA with SQL

# Unique Launch Sites

By using the word **DISTINCT**, the query returns unique launch sites from the SpaceX data

```
%sql select DISTINCT LAUNCH_SITE from SPACEXTBL
 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-
Done.
```

8]:

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

# Launch Sites that begin with 'CCA'

```
%sql select * from SPACEXTBL WHERE LAUNCH_SITE LIKE '%CCA%'limit 5
```

```
 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/BLUDB
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 | 07:44:00 | F9 v1.0 B0005 | CCAFS LC-40 | Dragon demo flight C2 | 525 | LEO (ISS) | NASA (COTS) | Success | No attempt |
| 2012-10-08 | 00:35:00 | F9 v1.0 B0006 | CCAFS LC-40 | SpaceX CRS-1 | 500 | LEO (ISS) | NASA (CRS) | Success | No attempt |
| 2013-03-01 | 15:10:00 | F9 v1.0 B0007 | CCAFS LC-40 | SpaceX CRS-2 | 677 | LEO (ISS) | NASA (CRS) | Success | No attempt |

The query above was used to select launch sites that have 'CCA' in their name. 'Limit 5' restricts the return to 5

# Total Payload Mass launched by NASA (CRS)

```
%sql select SUM(PAYLOAD_MASS__KG_) as Total_Payloadmass from SPACEXTBL where CUSTOMER = 'NASA (CRS)'
 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdo
Done.
```

4]:

| total_payloadmass |
|---|
| 45596 |

The query above calculates the total payload mass carried by boosters launched by NASA (CRS) as indicated in the WHERE clause

# Average Payload Mass carried by booster version F9 v1.1

```
%sql select AVG(PAYLOAD_MASS__KG_) as AVG from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'

 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databa
Done.
```

5]:

| AVG |
| --- |
| 2928 |

The query shown above calculates the average payload mass carried by booster version F9 v1.1 as indicated in the WHERE clause

# The date when the First Successful Landing Outcome in Ground Pad was Achieved

```
%sql select DATE from SPACEXTBL where LANDING__OUTCOME = 'Success (ground pad)' LIMIT 1
```

```
 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databa
Done.
```

5]:

| DATE |
| --- |
| 2015-12-22 |

The query returned '2015-12-22' as the date of the first successful landing in ground pad. The where clause filtered the landing outcome to only 'success (ground pad' while 'LIMIT 1' returns the earliest date

# Booster Names With Successful Drone Ship Landing With Payload Between 4000 And 6000

```
%sql select BOOSTER_VERSION from SPACEXTBL where LANDING__OUTCOME = 'Success (drone ship)' and PAYLOAD_MASS__KG_ between 4000 and 6000
```

```
 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/BLUDB
Done.
```

6]:

| booster_version |
|---|
| F9 FT B1022 |
| F9 FT B1026 |
| F9 FT B1021.2 |
| F9 FT B1031.2 |

The query above returns booter versions that have successful drone ship landing and have a payload mass greater than 4000 but less than 6000. The WHERE clause filtered the landing outcome to 'Success (drone ship) while the 'and' clause indicates an additional condition

# Total Number Of Successful And Failure Mission Outcomes

```
%sql select MISSION_OUTCOME, COUNT(MISSION_OUTCOME) as total from SPACEXTBL GROUP BY MISSION_OUTCOME
 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appd
Done.
```

| mission_outcome | total |
|---|---|
| Failure (in flight) | 1 |
| Success | 99 |
| Success (payload status unclear) | 1 |

The query above calculates the total number of successful and failure mission outcomes. The word 'COUNT' counts each mission outcome and the 'GROUP BY' clause groups each mission outcome according to its total

# Names Of Booster Versions Which Have Carried The Maximum Payload Mass

```
%sql select DISTINCT BOOSTER_VERSION from SPACEXTBL where PAYLOAD_MASS__KG_ = (select MAX(PAYLOAD_MASS__KG_) from SPACEXTBL)
 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/BLUDB
Done.
```

| booster_version |
|---|
| F9 B5 B1048.4 |
| F9 B5 B1048.5 |
| F9 B5 B1049.4 |
| F9 B5 B1049.5 |
| F9 B5 B1049.7 |
| F9 B5 B1051.3 |
| F9 B5 B1051.4 |
| F9 B5 B1051.6 |
| F9 B5 B1056.4 |
| F9 B5 B1058.3 |
| F9 B5 B1060.2 |
| F9 B5 B1060.3 |

The query above returns unique names of booster versions which have carried the maximum payload mass. A sub query was used here in the WHERE clause to calculate the maximum payload mass

# Failed Launch Records In 2015

```
%sql select BOOSTER_VERSION, LAUNCH_SITE, LANDING__OUTCOME from SPACEXTBL where LANDING__OUTCOME = 'Failure (drone ship)' and YEAR(DATE)='2015'
```

```
 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/BLUDB
Done.
```

9]:

| booster_version | launch_site | landing__outcome |
|---|---|---|
| F9 v1.1 B1012 | CCAFS LC-40 | Failure (drone ship) |
| F9 v1.1 B1015 | CCAFS LC-40 | Failure (drone ship) |

The query above is used to select the booster version and launch site of failed launch attempts on drone ship in the year 2015.

# Rank Landing Outcomes Between 2010-06-04 And 2017-03-20 In Descending Order

```
%sql select LANDING__OUTCOME, COUNT(*) as count from SPACEXTBL where DATE between '2010-06-04' and '2017-03-20' GROUP BY LANDING__OUTCOME ORDER BY count DESC
```

```
 * ibm_db_sa://xmf09166:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/BLUDB
Done.
```

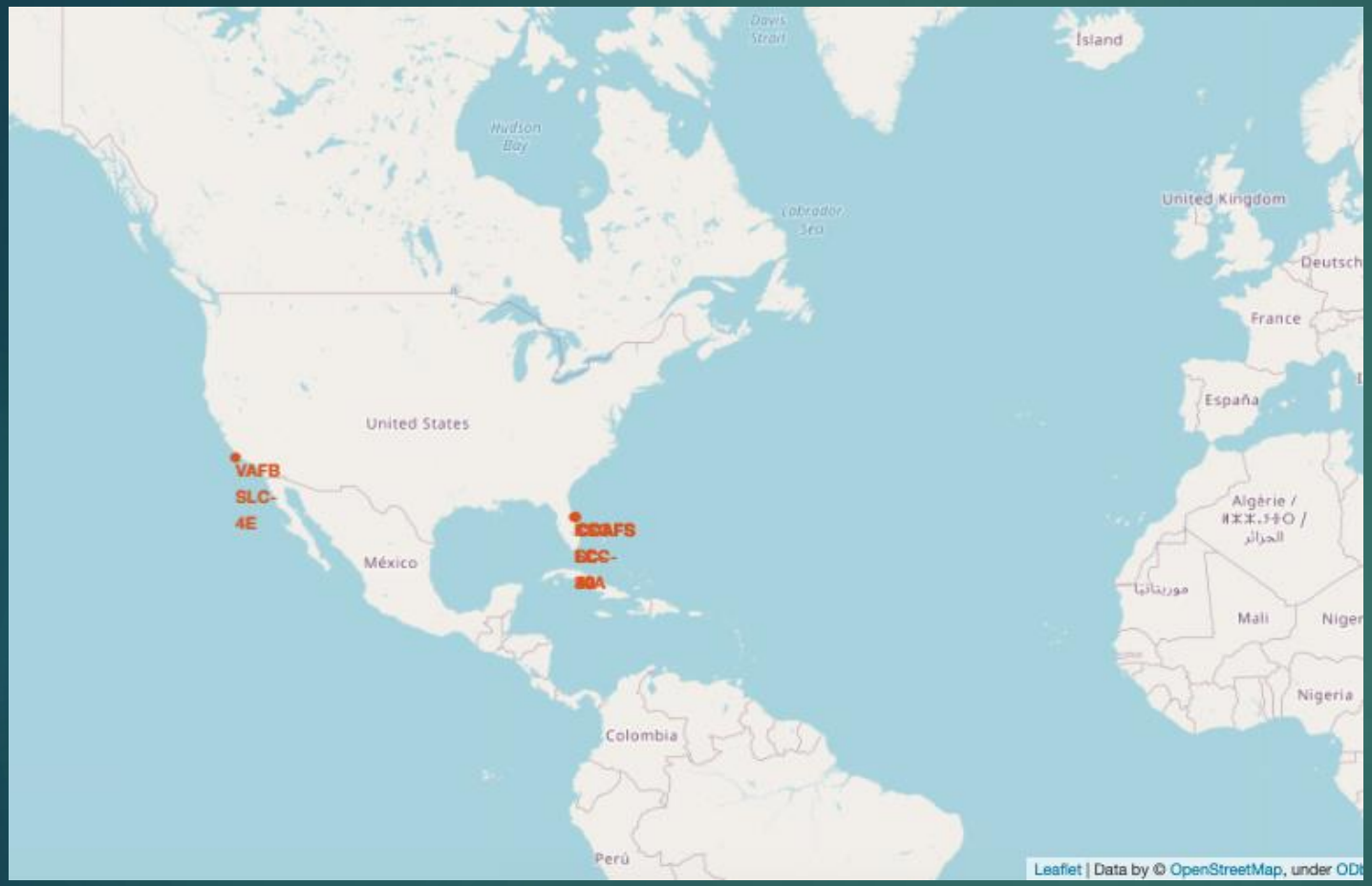
6]:

| landing__outcome | COUNT |
|---|---|
| No attempt | 10 |
| Failure (drone ship) | 5 |
| Success (drone ship) | 5 |
| Controlled (ocean) | 3 |
| Success (ground pad) | 3 |
| Failure (parachute) | 2 |
| Uncontrolled (ocean) | 2 |
| Precluded (drone ship) | 1 |

The query is used to count the different landing outcomes from 2010-06-04 to 2017-03-20 as shown in the WHERE clause. The 'GROUP BY' clause groups each landing outcome according to its count while the 'ORDER BY' clause orders it in descending order.

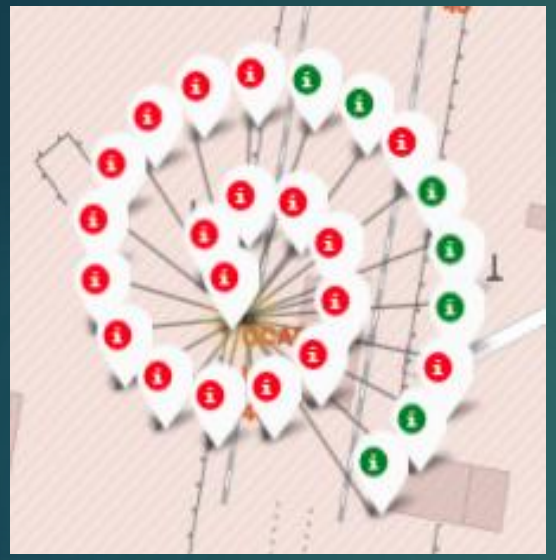
# Interactive Map with Folium

# All Launch Sites Global Markers

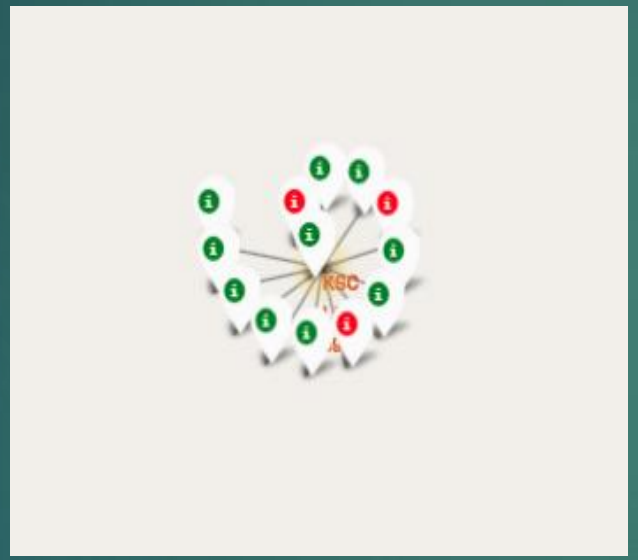



We can see that all the SpaceX launch sites are located in the United states of America

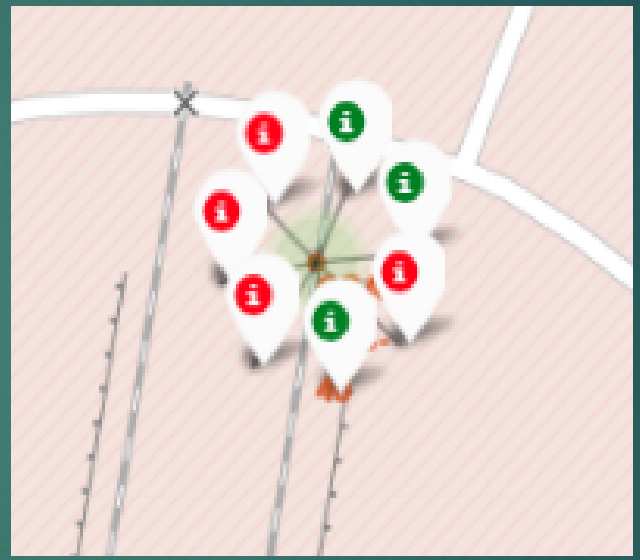
# Markers Showing The Successful/Failed For Each Site On The Map



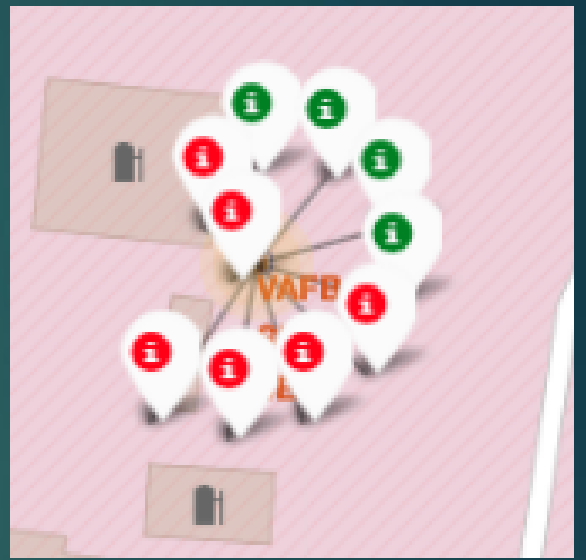
CCAFS LC-40




KSC LC-39A



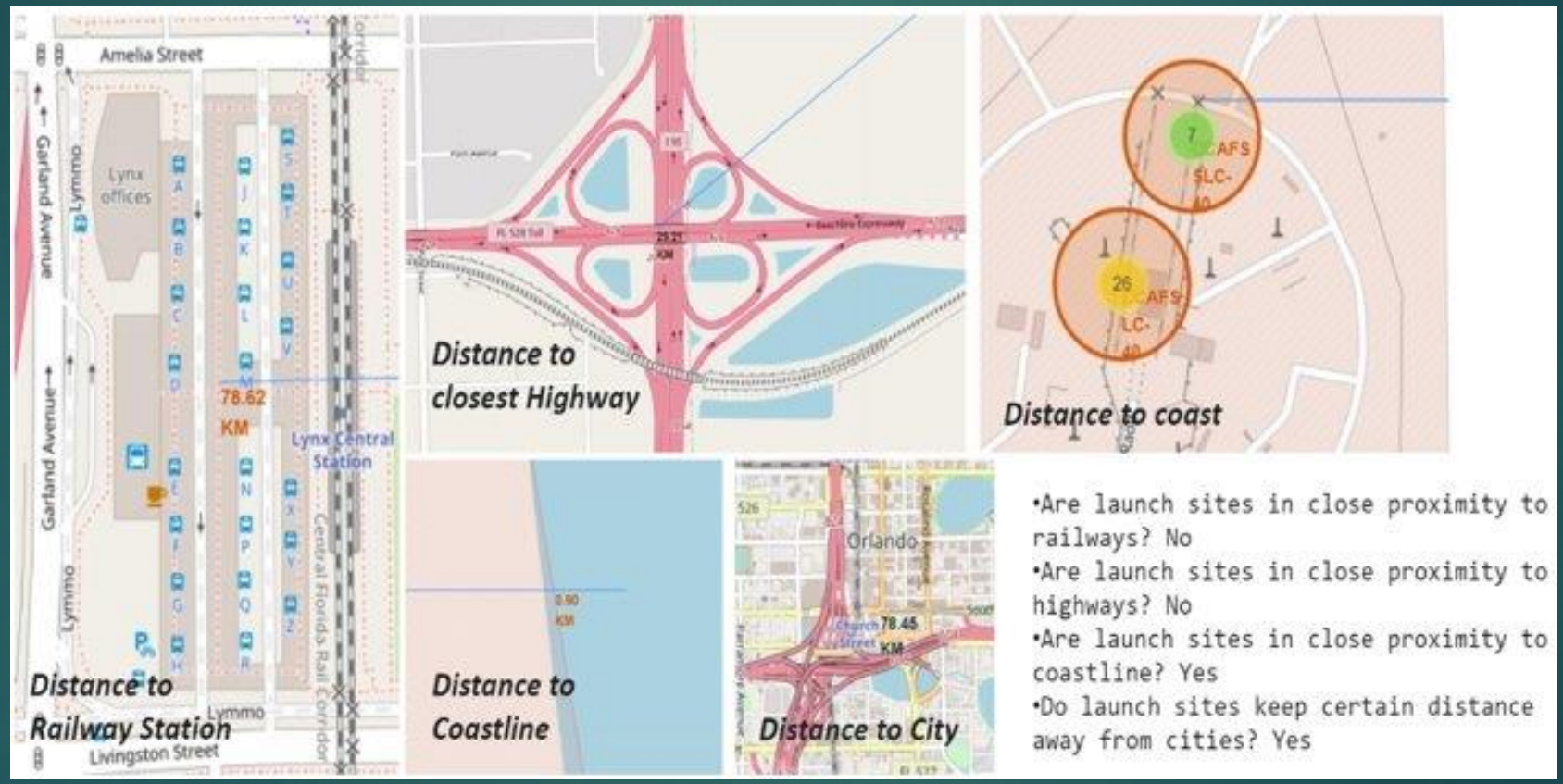
CCAFS SLC-40




VAFB SLC-4E

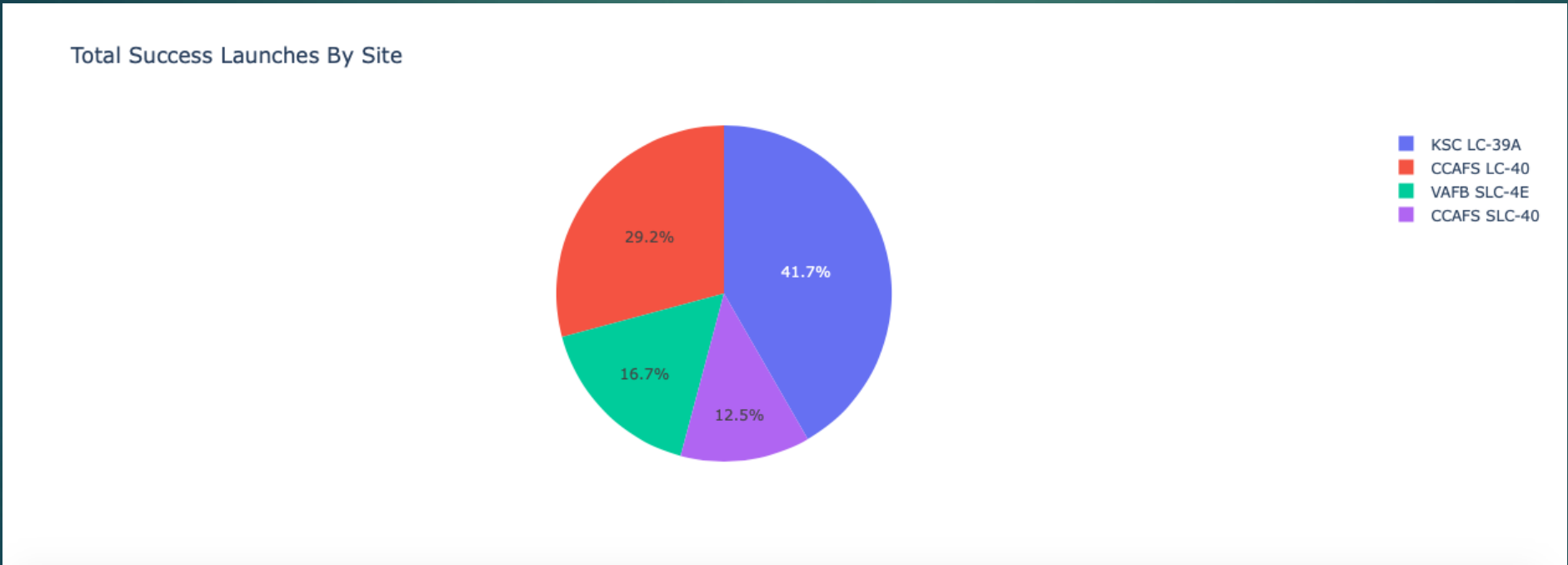
*Green Marker shows successful launches and Red marker shows failures*

# Launch Site Distance to Proximities



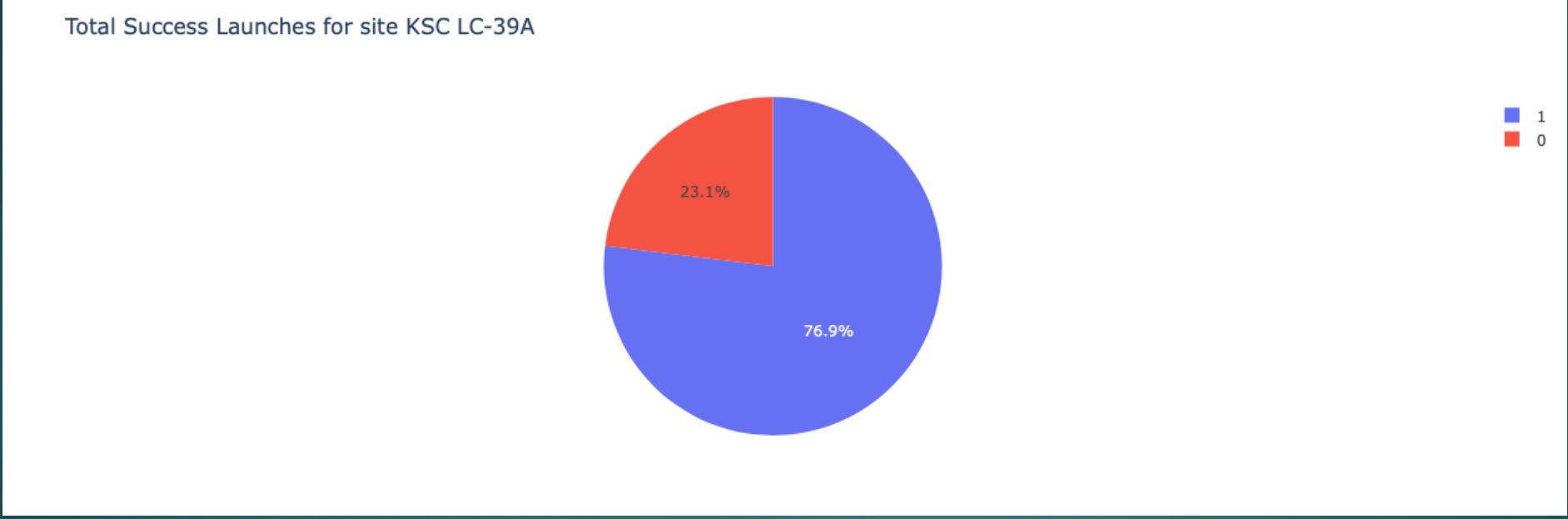
# Interactive Dashboard With Plotly Dash

# Pie Chart showing Success Rate by all Launch Sites




From the chart, we can see that the KSC LC-39A launch site had the highest success rate

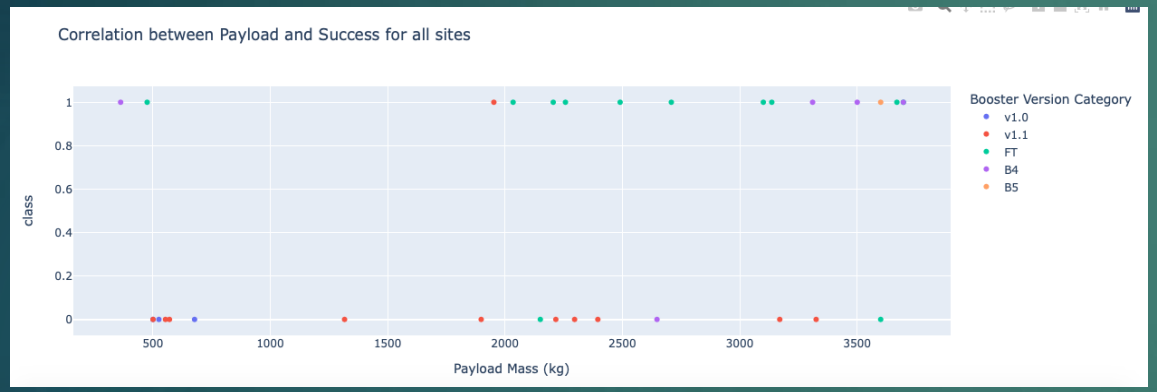
# Pie Chart Showing Launch Site With The Highest Success Rate



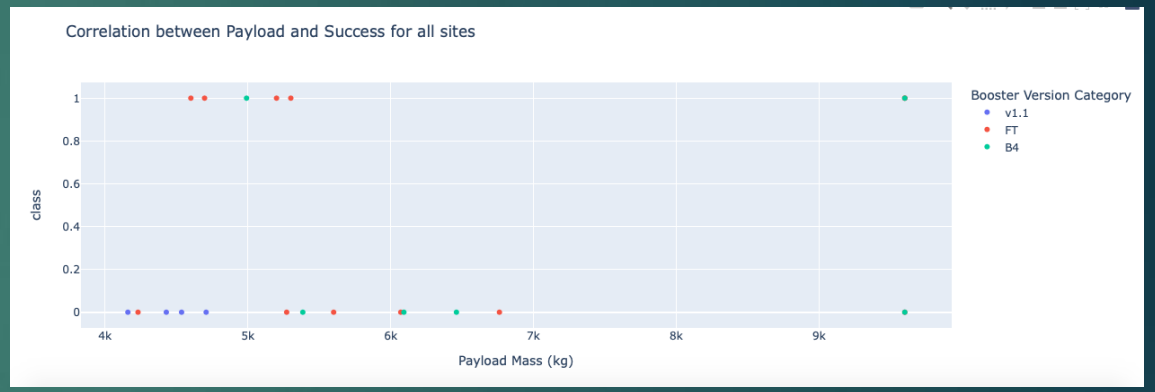

KSC LC- 39A achieved a success rate of 76.9% while only having a failure rate of just 23.1%

# Payload vs Launch Outcome Scatter plot with different masses selected from the range slider

Low Payload Mass (0kg-4000kg)




High Payload Mass (4000kg-10000kg)




From the scatter plots above, we can see that the success rate for launches with low payload masses are significantly higher than launches with high payload mass

# Classification Accuracy

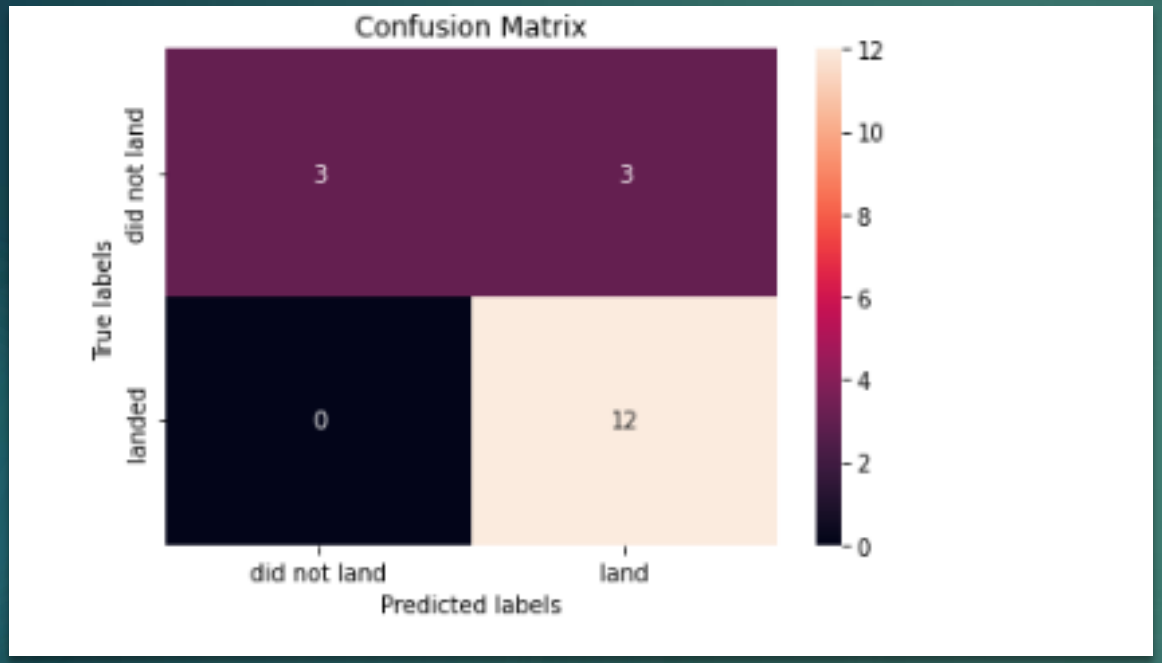
```
models = {'kneighbors': knn_cv.best_score_,
          'DecisionTree': tree_cv.best_score_,
          'SVM': svm_cv.best_score_,
          'LogisticRegression': logreg_cv.best_score_ }

best_model = max(models, key = models.get)
print('The best model is',best_model, 'with a score of',models[best_model])
```

```
The best model is DecisionTree with a score of 0.8767857142857143
```

According to the code above, the Decision Tree model is the best model to use for predictive analysis as it has the highest accuracy

# Confusion Matrix




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

# Conclusion

We can conclude that:

- The higher the amount of flights, the higher the success rate at a launch site
- Orbit types ES-L1, GEO, HEO, SSO and VLEO and the highest success rates
- Heavy payloads are much more successful in low orbit. Eg LEO, PO and ISS orbits
- The launch success rate had a constant increase from 2013 up until 2020. This may be as a result of the increase in the number of flights
- The KSC LC-39A launch site had the highest success rate out of all the launch sites
- On the average, launches with low payloads have a higher success rate than launches with heavy payloads
- For this project, the best model to use is the Decision Tree Model