IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Sara Gifford
July 2023

Outline

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

Executive Summary

- Summary of methodologies
  - Data Collection through API
  - Data Collection with Web Scraping
  - Data Wrangling
  - Exploratory Data Analysis with Data Visualization
  - Exploratory Data Analysis with SQL
  - Interactive Visual Analytics with Folium
  - Build a Dashboard with Plotly Dash
  - Machine Learning Prediction
- Summary of all results
  - Exploratory Data Analysis Results
  - Interactive Analytics Screenshots
  - Predictive Analysis Results

Introduction

The Falcon 9 rocket is a reusable two-stage rocket developed by SpaceX. The reusable first stage, the most expensive part of the rocket, allows SpaceX to offer space access for people and payloads for 62 million dollars. Other launch providers offer the same service for more than 165 million dollars. If the probability of successful first stage landings can be predicted, then the cost of the a launch can be determined. SpaceX competitors can use this information to bid for rocket launches. The goal of this project is to create a machine learning pipeline to predict if the first stage of Falcon 9 will successfully land.

This project seeks to answer:

- What factors influence the landing outcome of the Falcon 9 first stage?
- How do the various features interact and contribute to the landing outcome?
- What is the probability of successful first stage landing of the Space X Falcon 9 rocket?
- What conditions best contribute to a successful landing outcome?

Section 1

Methodology

Methodology

- Data collection methodology
  - Collect data using SpaceX API and web scraping from Wikipedia
- Perform data wrangling
  - Apply One-hot encoding to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Logistic Regression, SVM, Decision Tree Classifier, and K Nearest Neighbors Models

Data Collection

- Data set collection methods:
  - Data collected from the SpaceX open source REST API using a GET request
    - Decode response content as a JSON using .json() function and create pandas dataframe using .jsonj_normalize() method
    - Extract launch information using identification numbers from rocket, payload, launchpad, and cores columns
    - Clean data and check for missing values
      - Filter data for only Falcon 9 launches
      - Replace None values with mean value for the Payload column
  - Data collected from Wikipedia HTML table using Beautiful Soup
    - Extract Falcon 9 launch records from Wikipedia HTML table
    - Parse the table and convert it to a Pandas data frame

Data Collection – SpaceX API

1. GET request from SpaceX API

2. Convert response content as a JSON to Pandas dataframe

3. Clean and fill missing data

4. GitHub URL

https://github.com/SaraG51/IBM-Applied-Data-Science-Capstone-Project/blob/main/1_1%20Data%20Collection%20API.ipynb

1

Now let's start requesting rocket launch data from SpaceX API with the following URL:

```
In [6]: spacex_url="https://api.spacexdata.com/v4/launches/past"
```

```
In [7]: response = requests.get(spacex_url)
```

2

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

Using the dataframe `data` print the first 5 rows

```
# Get the head of the dataframe
data.head()
```

```
# Hint data['BoosterVersion']!='Falcon 1'
data_falcon9=df[df['BoosterVersion']!='Falcon 1']
```

Now that we have removed some values we should reset the FlgihtNumber column

```
data_falcon9.loc[:,'FlightNumber'] = list(range(1, data_falcon9.shape[0]+1))
data_falcon9
```

3

```
# Calculate the mean value of PayloadMass column
PayloadMass_mean=data_falcon9['PayloadMass'].mean()
Pa
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(value=
```

```
6123.547647058824
```

Data Collection - Scraping

1. Request Falcon 9 launch records from Wikipedia page

2. Create a Beautiful Soup Object from HTML response

3. Parse the table

4. Create a pandas dataframe from dictionary of parsed launch record values

GitHub link

https://github.com/SaraG51/IBM-Applied-Data-Science-Capstone-Project/blob/main/1_2%20Data%20Collection%20Webscraping.ipynb

1

First, let's perform an HTTP GET method to request the Falcon9 Launch HTML page, as an HTTP response.

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

2

Create a `BeautifulSoup` object from the HTML `response`

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

3

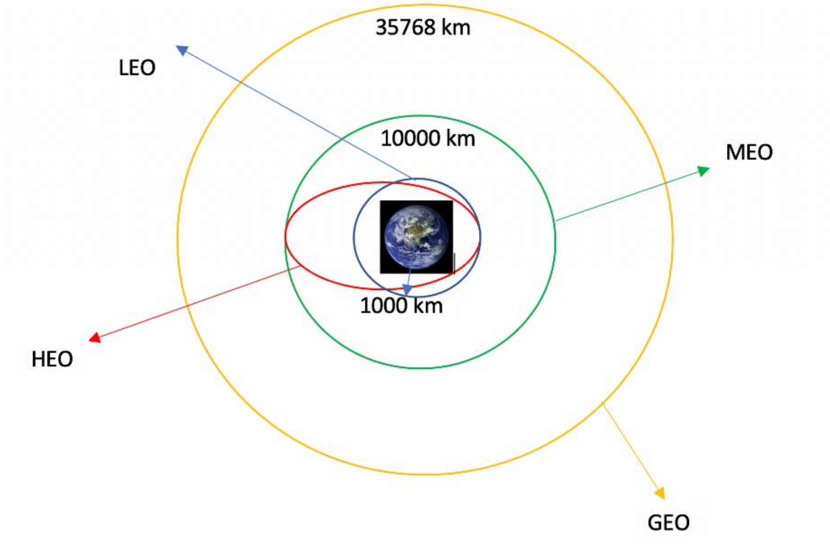
```
if flag:
    extracted_row += 1
    # Flight Number value
    # TODO: Append the flight_number into launch_dict with key `Flight No.`
    launch_dict['Flight No.'].append('flight_number')
    #print(flight_number)
    datatimelist=date_time(row[0])

    # Date value
    # TODO: Append the date into launch_dict with key `Date`
    date = datatimelist[0].strip(',')
    launch_dict['Date'].append(date)
    #print(date)

    # Time value
    # TODO: Append the time into launch_dict with key `Time`
    time = datatimelist[1]
    launch_dict['Time'].append(time)
    #print(time)
```

4

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

Data Wrangling

- Exploratory Data Analysis (EDA) based on launch site and orbit
  - Calculate the number of launches at each site
  - Calculate the number and occurrence of each orbit
  - Calculate the number and occurrence of each mission outcome per orbit type
- Determine training labels
  - For use in further analysis, visualization, and predictive models
- Export results to CSV
- GitHub URL
  - [Data Wrangling]()



Common orbit types

EDA with SQL

- SQL queries were used to better understand the data set:
  - Unique launch site names
  - 5 records where launch sites begin with 'CCA' string
  - Total payload mass carried by boosters launched by NASA (CRS)
  - Average payload mass carried by booster version F9 v1.1
  - Date of first successful landing outcome at ground pad
  - Names of boosters which have had success in drone ship and have payload mass greater than 4000 but less than 6000
  - Total number of successful and failure mission outcomes
  - Booster versions which have carried the maximum payload mass (using a subquery)
  - List of records by month with failure landing outcomes in 2015 including drone ship, booster version, and launch site
  - Rank the count of landing outcomes between 2010-06-04 and 2017-03-20 in descending order
- GitHub URL: [EDA with SQL]

EDA with Data Visualization

Scatter plots were created to examine the relationship between different features. The scatter plots shown on the right show the relationship between flight number with orbits and launch sites on landing outcomes.

Flight Number v. Launch Site

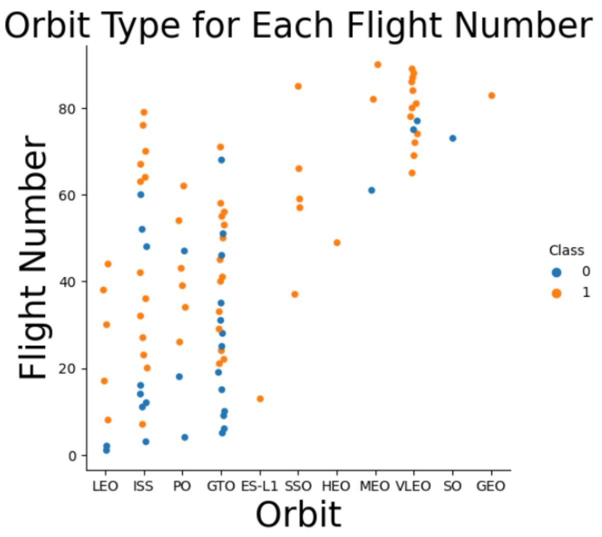
Payload Mass v. Launch Site

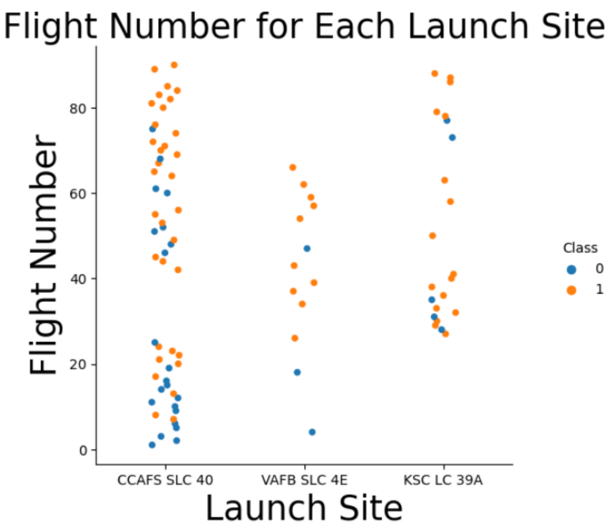
Flight Number v. Orbit Type

Payload Mass v. Orbit Type

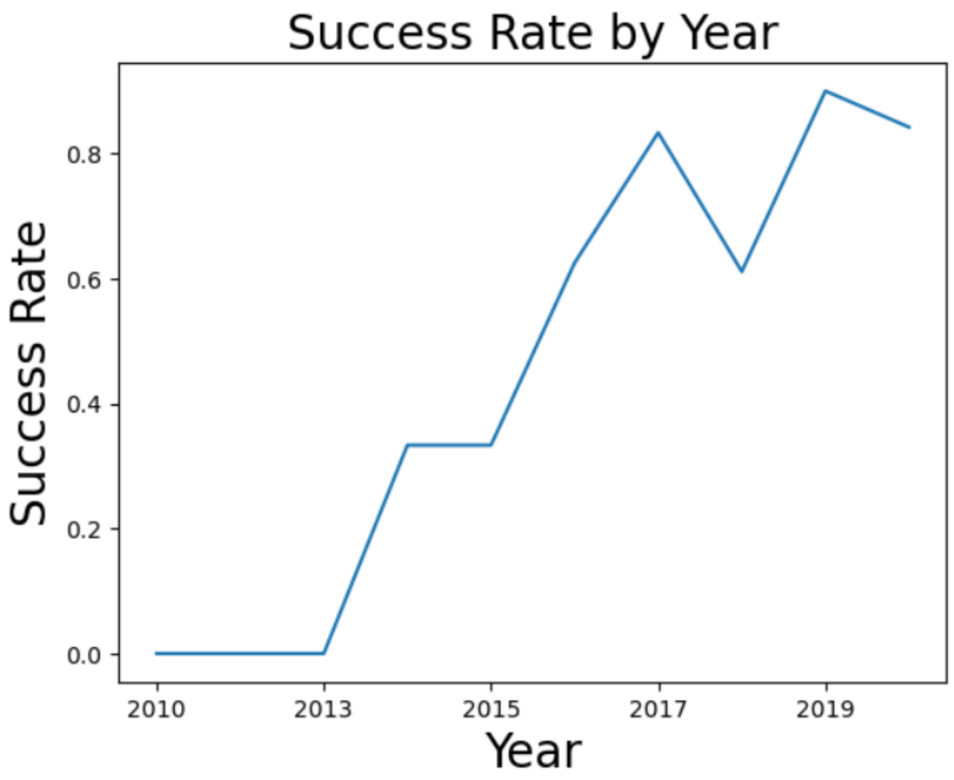
Launch Success Yearly Trend

GitHub URL: [EDA with Data Visualization]





EDA with Data Visualization



- Line plots shows the relationship between 2 variables which allows trends to be identified. The line plot on the left suggests there is a positive relationship between year of launch and success rate.
- GitHub URL https://github.com/SaraG51/IBM-Applied-Data-Science-Capstone-Project/blob/main/2_2%20EDA%20Data%20Visualization.ipynb

Build an Interactive Map with Folium

- Launch outcomes are visualized on an interactive map. Map objects were created for each launch site. Circle markers are placed at the launch site's longitude and latitude coordinates. Text labels for each launch site were included to aid in launch site identification.
- Launch outcomes for each launch site are included with Marker Clusters to reduce map clutter. Green markers indicate successful launches. Red markers indicate failed launches.
- Lines with distance labels are drawn on the map to understand the proximity of launch sites to coastlines, railways, highways, and cities.
- GitHub URL: Launch Sites Interactive Map with Folium

Build a Dashboard with Plotly Dash

- The Dashboard dropdown menu allows users to examine data for all launch sites or individual launch sites. The relationship between launch outcomes, payload mass (kg), and booster version can be explored with the dashboard.
- Pie charts show the successful outcomes for launch sites.
- Scatter plots with a slider range allows users to examine launch outcomes based on payload mass and booster version for launch sites.
- GitHub URL: Plotly Interactive Dashboard

Predictive Analysis (Classification)

Build ML Models

- Load data into NumPy and Pandas
- Standardize data
- Split data into training and testing data sets
- Set parameters and fit models to the data set

Evaluate ML Models

- Check the accuracy of each ML model
- Get tuned hyperparameters for each ML model
- Plot the confusion matrix

Improve ML Models

- Use Feature Engineering and Algorithm Tuning

Select the Best Model

- Use accuracy score

GitHub URL : Machine Learning Prediction

Results

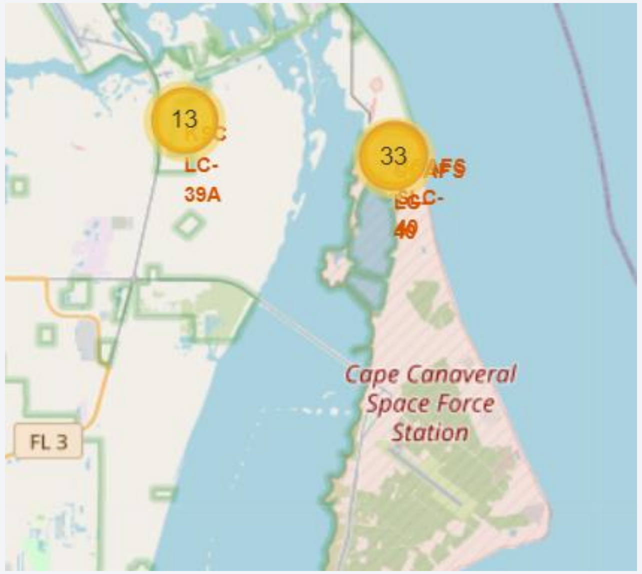
Exploratory data analysis results

- SpaceX uses 4 launch sites located in CA and FL
- The average F9 v1.1 booster payload is 2,928 kg
- 12 booster versions carried maximum payload
- Almost 100% of missions were successful
- Higher flight numbers had a higher success rate for each launch site and a variety of orbits
- Higher payload mass had a higher success rate for each launch site and a variety of orbits
- Success rates generally increased yearly with noticeable drops in 2018 and 2020

Results

Interactive analytics show that launch sites characteristics

- Away from population centers
- Near the coast
- Railway access available
- Most launches occur on east coast



Results

Predictive analysis results

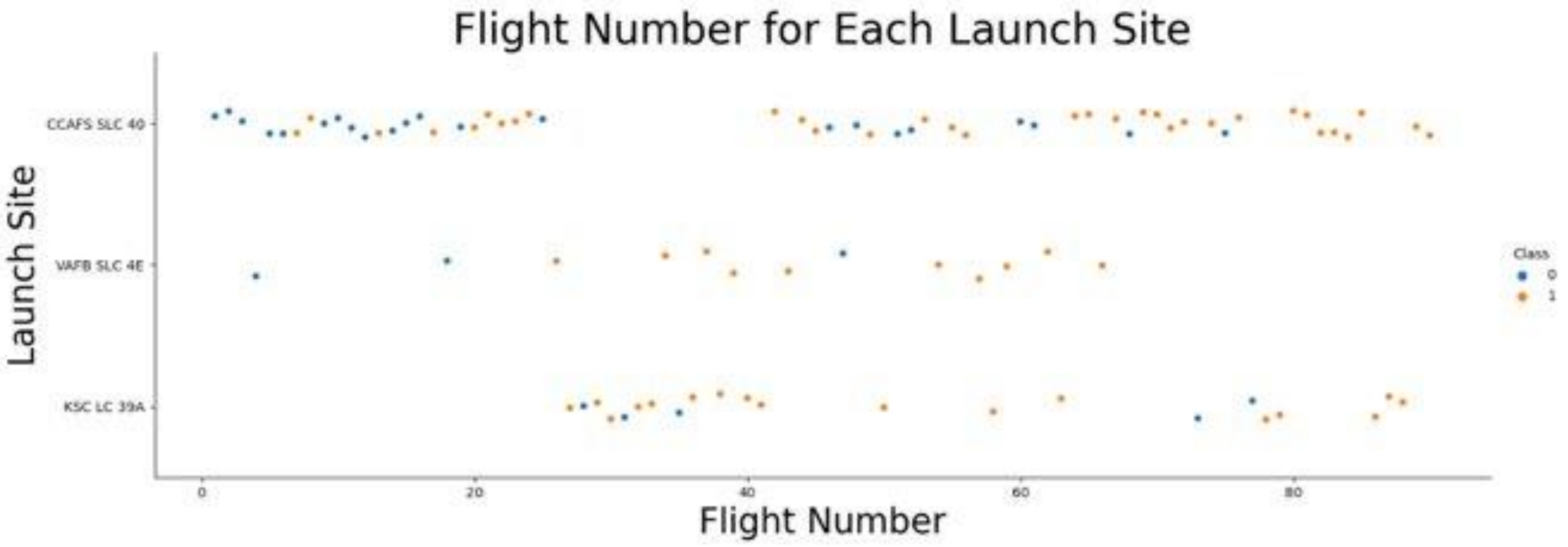
- Decision Tree Classifer is the model with the highest accuracy
- The Decision Tree Classifier model can distinguish between different classes.
  - The Confusion Matrix reveals that unsuccessful landings may be predicted as successful landings by the model.

Section 2

Insights drawn from EDA

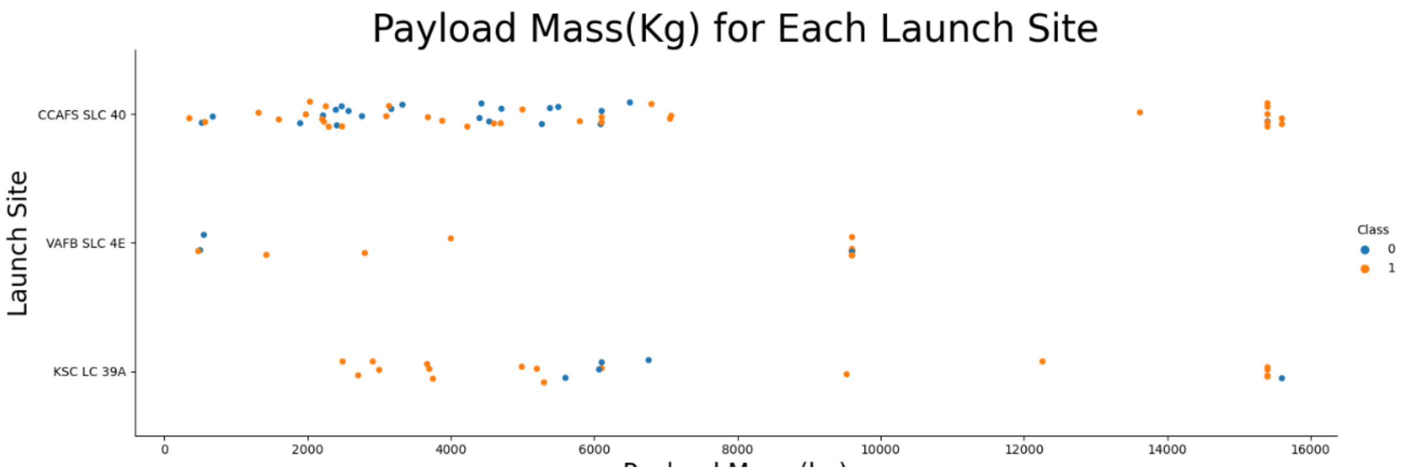
Flight Number vs. Launch Site

As the flight number increases at each launch site, the success rate increases.



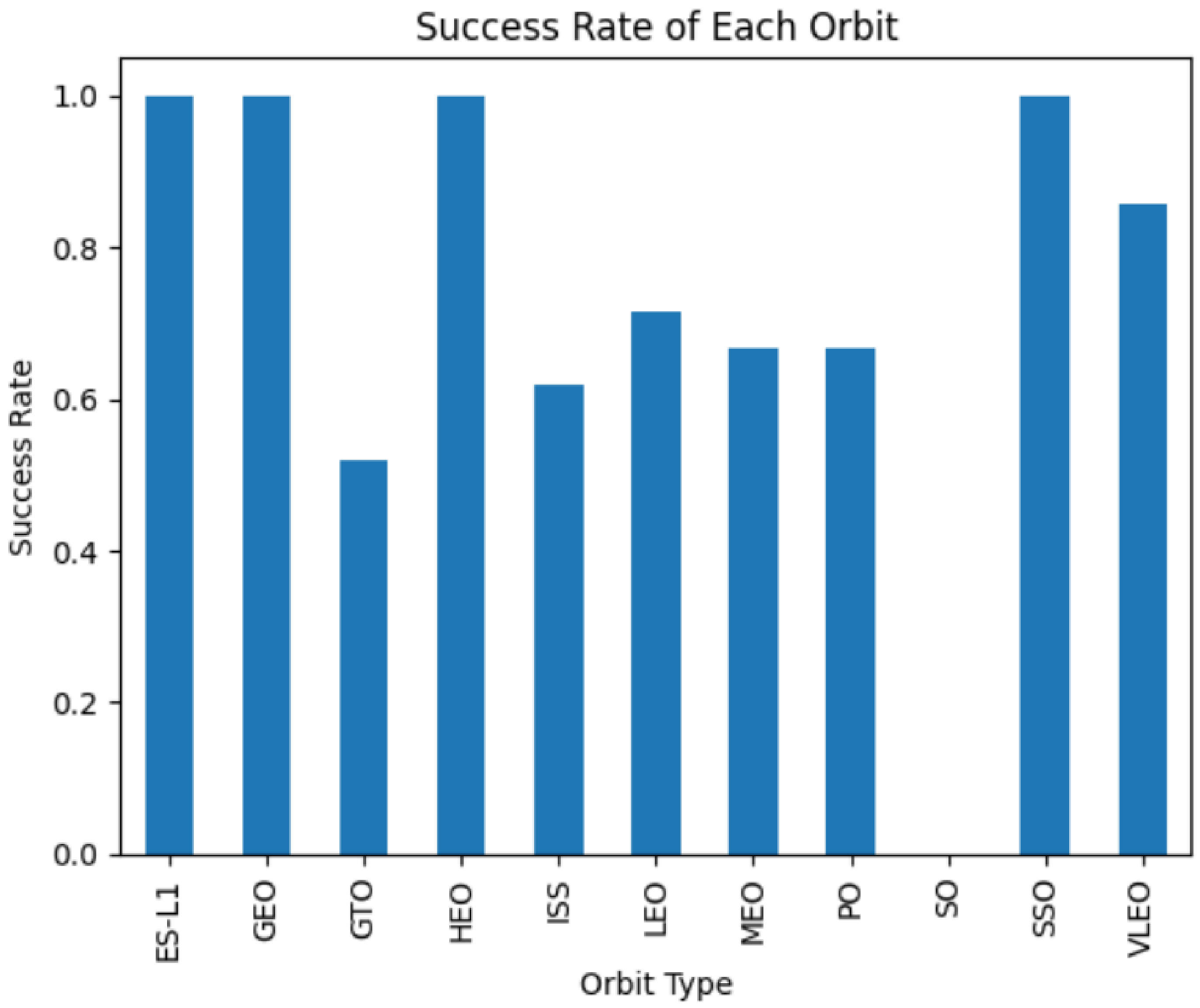
Payload vs. Launch Site

- Payloads greater than 7000 kg have a highly increased success rate. VAFB SLC 4E and KSC LC 39A show high success rates for payloads less than 5000 kg. The data does not show a clear pattern for the relationship between payload and launch site.



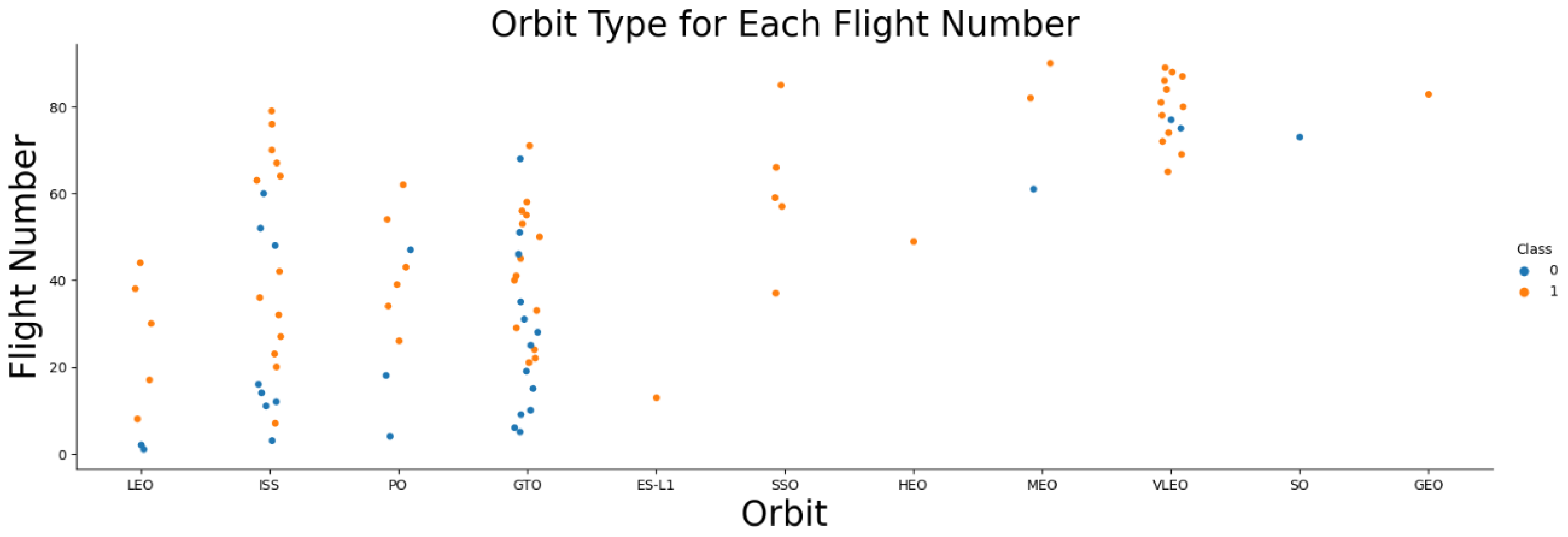
Success Rate vs. Orbit Type

ES-L1, GEO, HEO, and SSO orbits had 100% success rates. VLEO also had a high success rate. GTO, ISS, LEO, MEO, PO orbits had success rates between 50% and 70%. The SO had 1 unsuccessful launch resulting in a 0% success rate. The 100% GEO success rate is based on 1 successful launch. Upon closer examination, no orbit had a significant number of launches. More data for each orbit is required before the orbit type can be used to draw a meaningful conclusion about the success rate.



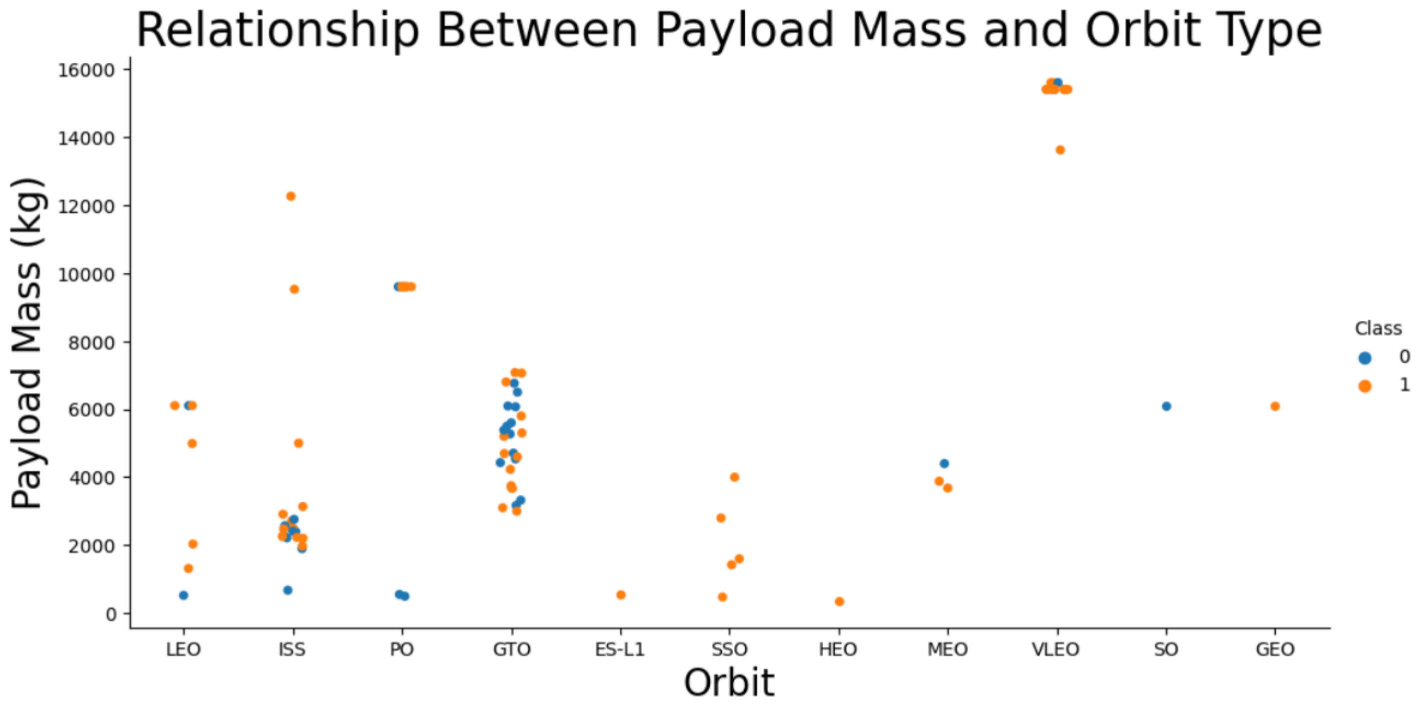
Flight Number vs. Orbit Type

Generally, as the flight number increases the success rate increases as seen for the LEO, PO, SSO, MEO, VLEO, and GEO orbits. The ISS and particularly the GTO orbits show mixed outcomes based on flight numbers. The ES-L1, HEO, SO, and GEO orbits had 1 flight each suggesting that more data is needed.



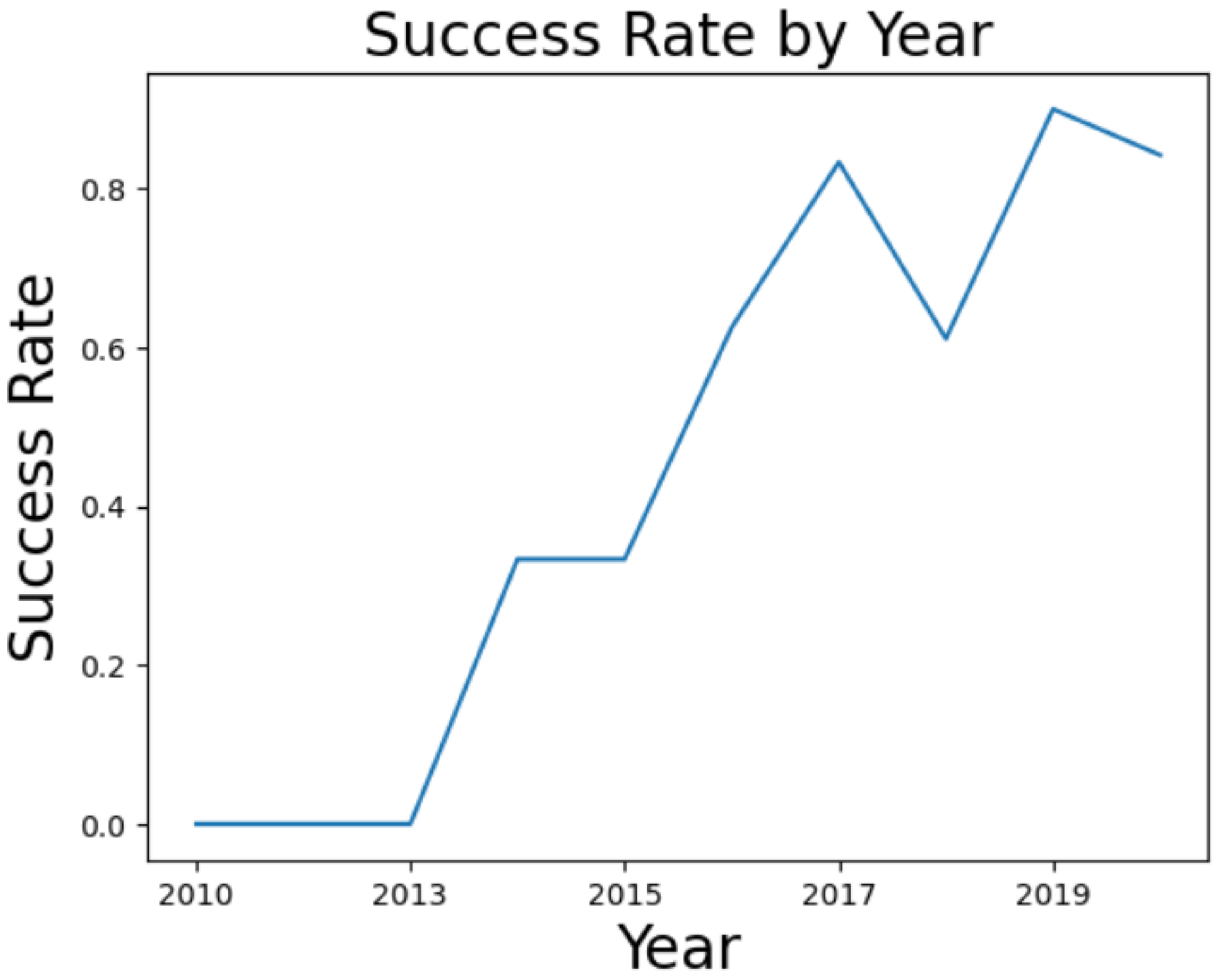
Payload vs. Orbit Type

As Payload Mass increases the success rate increases for the LEO, ISS, and PO orbits. The GTO orbit did not show a relationship between Payload Mass and success rate. More data is needed to draw a conclusion for the ES-L1, HEO, SO, and GEO orbits. The SSO orbit showed a high success rate a low Payload Mass.



Launch Success Yearly Trend

The line plot shows the trend of a success rate increasing each year from 2013 to 2020 with noticeable dips in 2018 and 2020.



All Launch Site Names

The key word **DISTINCT** is used to show only unique launch sites from the SpaceX data.

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

```
%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
None

Launch Site Names Begin with 'CCA'

5 records where launch sites begin with `CCA` are displayed.

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
)one.
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
06/04/2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0.0	LEO	SpaceX	Success	Failure (paracl
12/08/2010	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0.0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (paracl
22/05/2012	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525.0	LEO (ISS)	NASA (COTS)	Success	No att
10/08/2012	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500.0	LEO (ISS)	NASA (CRS)	Success	No att
03/01/2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677.0	LEO (ISS)	NASA (CRS)	Success	No att

Total Payload Mass

Total payload carried by boosters from NASA (CRS) is 45,596.0 kg.

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

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

SUM(PAYLOAD_MASS__KG_)
45596.0

Average Payload Mass by F9 v1.1

The average payload mass carried by booster version F9 v1.1 is 2,928.4 kg.

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

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

AVG(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

The first successful landing outcome on ground pad occurred on January 8, 2018.

```
]: %sql select min(Date) from SPACEXTBL where Landing_Outcome = "Success (ground pad)";

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

min(Date)
01/08/2018

Successful Drone Ship Landing with Payload between 4000 and 6000

Boosters F9 FT B1022, F9 FT B1026, F9 FT B1021.2 have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000

```
%sql select distinct(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

Total Number of Successful and Failure Mission Outcomes

The total number of successful missions was 100. The total number of failed missions was 1.

List the total number of successful and failure mission outcomes

```
%sql select count(Mission_Outcome) as Successful_Missions from SPACEXTBL where Mission_Outcome Like "Success%";
```

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

Successful_Missions
100

```
%sql select count(Mission_Outcome) as Failed_Missions from SPACEXTBL where Mission_Outcome like "Fail%";
```

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

Failed_Missions
1

Boosters Carried Maximum Payload

These booster versions have carried the maximum payload mass.

```
%sql select Booster_Version from SPACEXTBL\
    where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL);
```

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

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

2015 Launch Records

There were 2 failed drone ship landing outcomes in 2015. They occurred in April and October.

```
%sql select substr(Date,4,2) as month, Date, Landing_Outcome, Booster_Version, Launch_Site from SPACEXTBL\
    where [Landing_Outcome] = "Failure (drone ship)" and substr(Date,7,4)='2015';
```

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

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

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

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

Rank the count of landing outcomes (such as Failure (drone ship) or Success (grou

```
%sql SELECT Landing_Outcome, count(Landing_Outcome) from SPACEXTBL \
    where Date Between '04-06-2010' and '20-03-2017' \
    group by Landing_Outcome \
    order by count(Landing_Outcome) DESC;
```

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

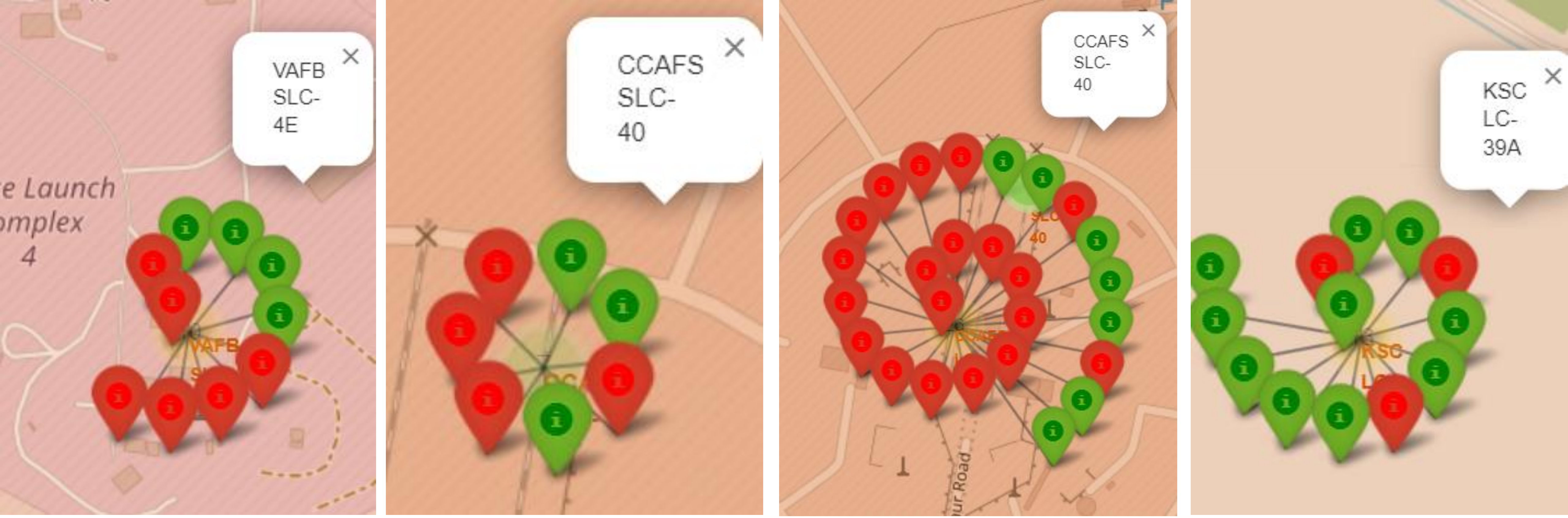
Landing_Outcome	count(Landing_Outcome)
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

Section 3

Launch Sites Proximities Analysis

Launch Site Locations

All SpaceX Falcon 9 launch locations are located in the United States of America.



Launch Outcomes for Each Launch Site

Launch outcomes for each launch site are shown.

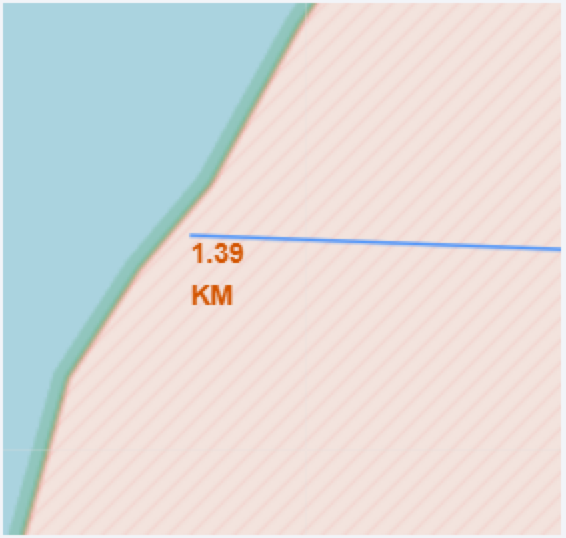
Green markers indicate a successful launch

Red markers indicate a failed launch

VAFB SLC 4E Launch Site Proximity to Landmarks

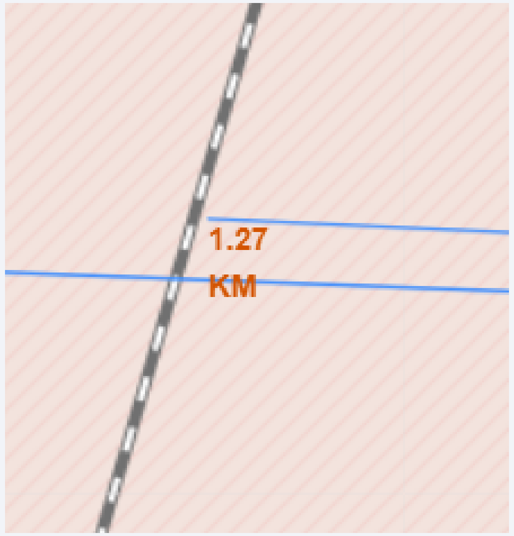
- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to population centers? No
- Are launch sites in close proximity to coastlines? Yes

Proximity to coastline = 1.39 km



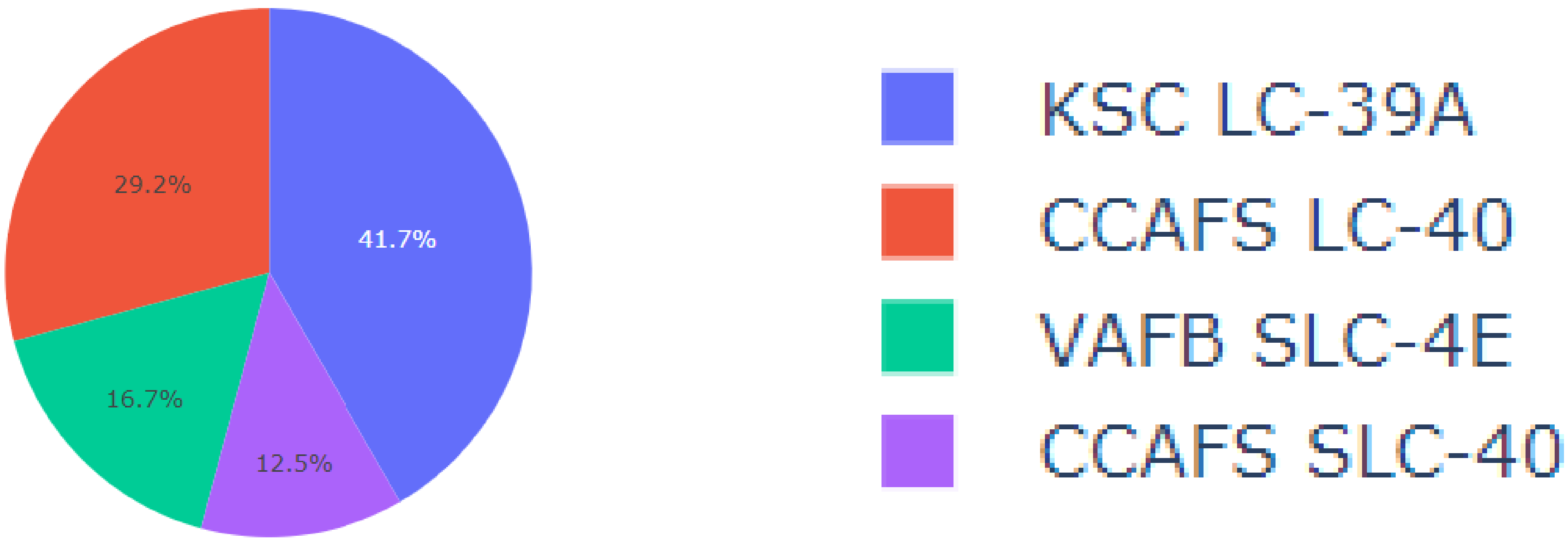
Proximity to town = 12.34 km
Proximity to highway = 13.97 km

Proximity to railroad = 1.27 km



Section 4

Build a Dashboard with Plotly Dash

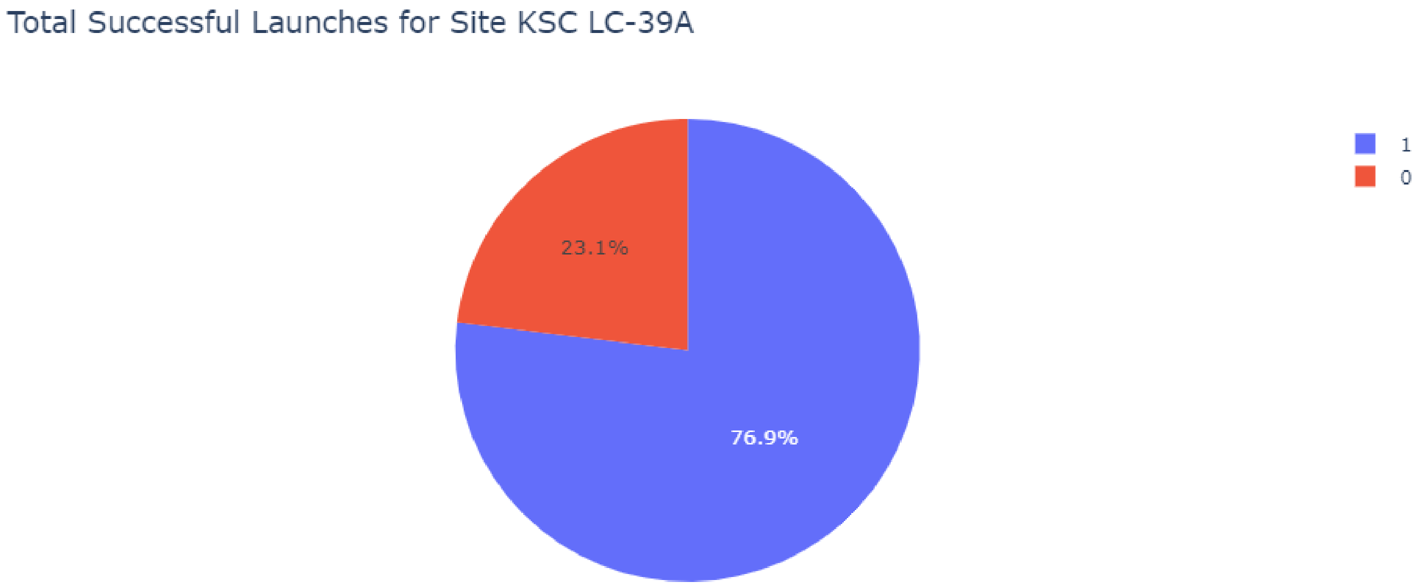


Success Percentage for All Launch Sites

- KSC LC-39A had the highest success percentage.
- VAFB SLC-4E and CCAFS SLC-40 have the lowest success percentages.

Launch Site with Highest Success Ratio

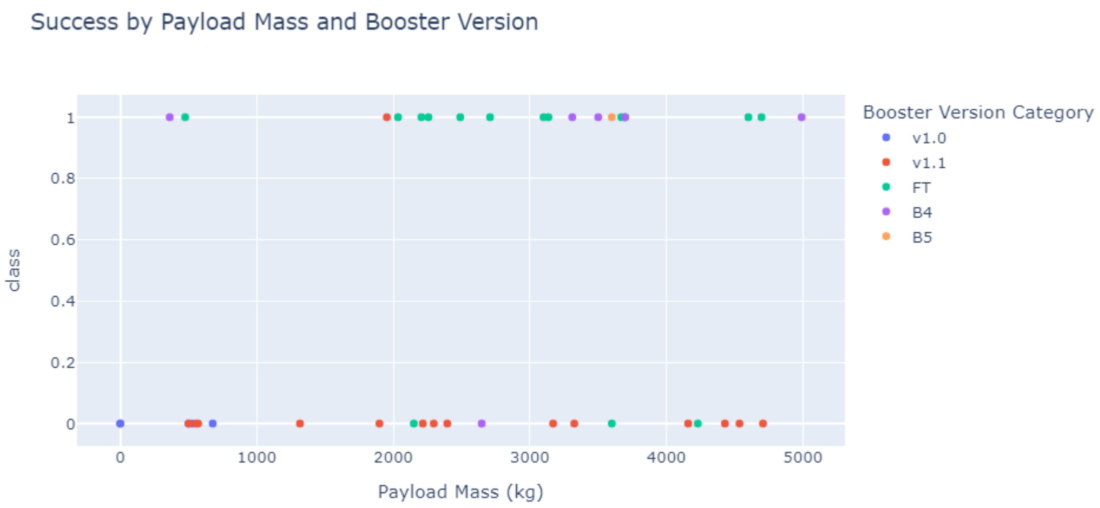
- KSC LC-39A had the highest successful launch rate
- 76.9% of launches were successful at KSC LC-39



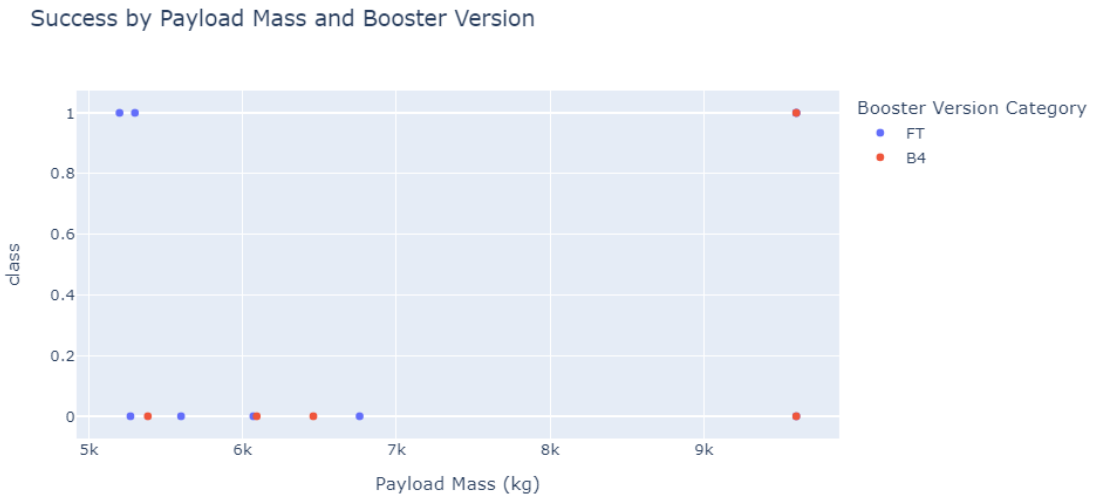
Payload vs. Launch Outcome for All Sites

- The success rate for payloads less than 5,000 kg is higher than the success rate for payloads above 5,000 kg.

Success Rate for Payloads Less than 5,000 kg



Success Rate for Payloads Greater than 5,000 kg

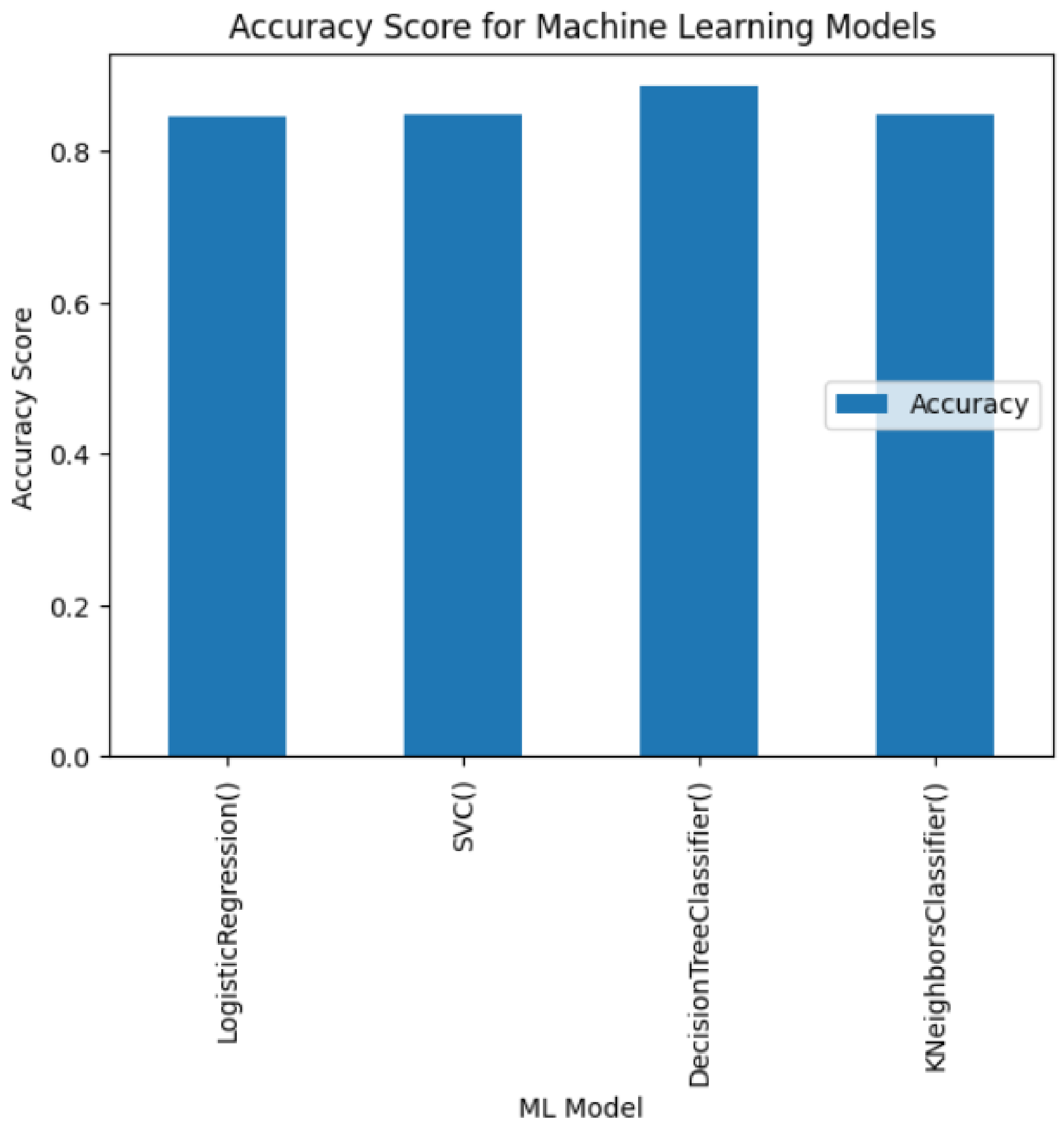


Section 5

Predictive Analysis (Classification)

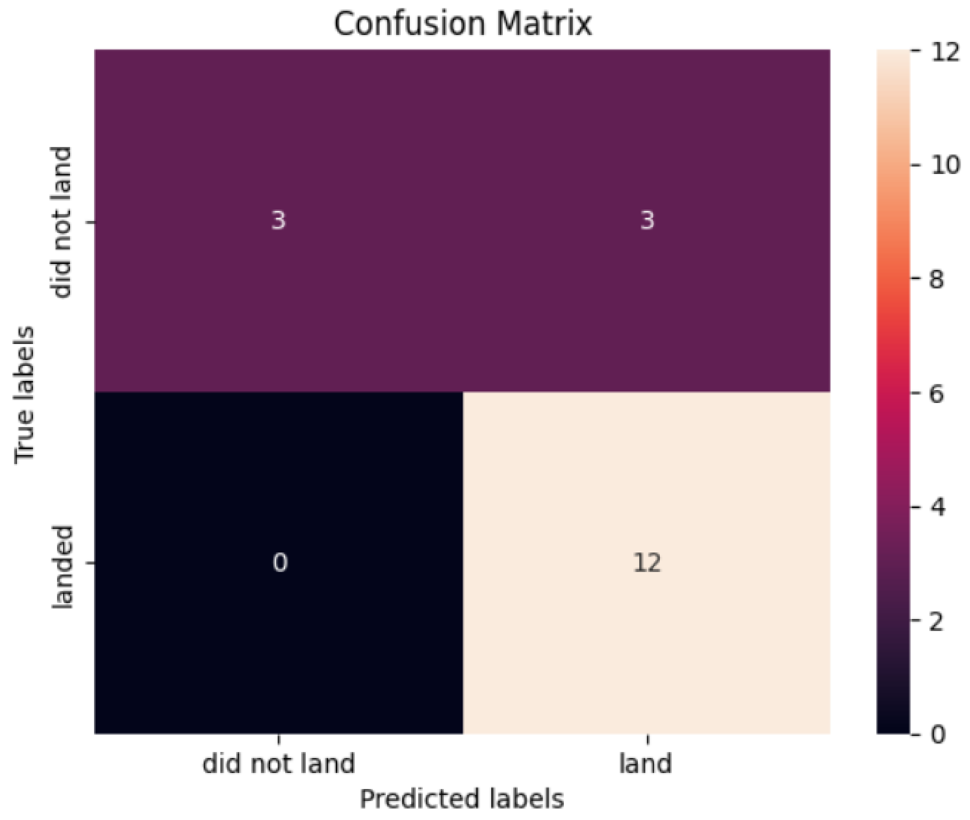
Classification Accuracy

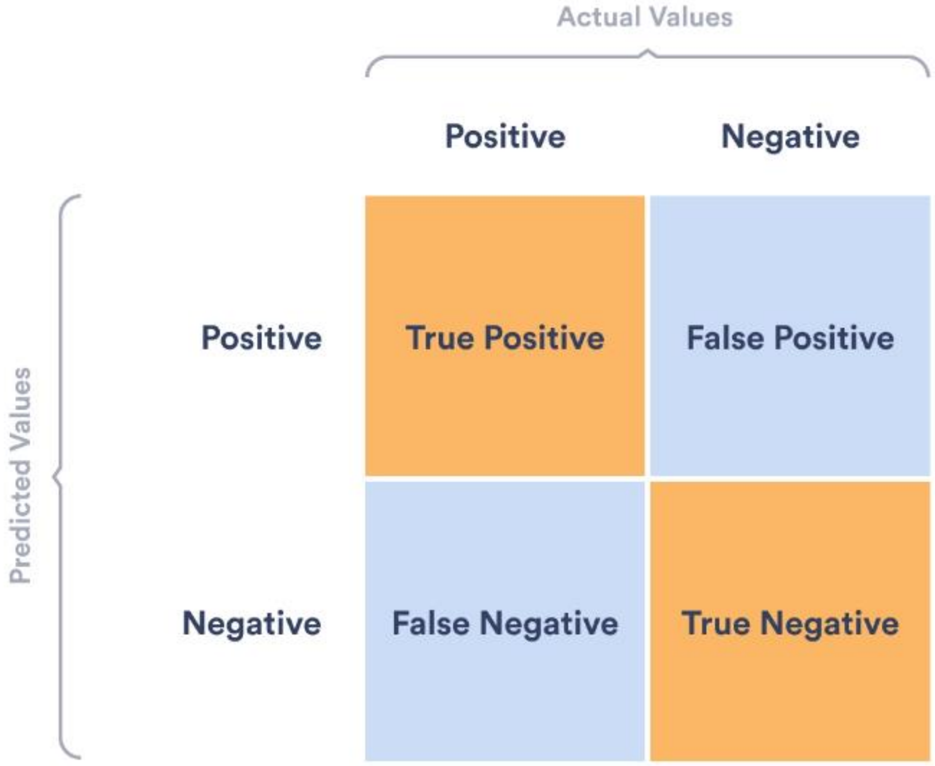
Decision Tree Classifier has the highest accuracy score.



Confusion Matrix for Decision Tree Classifier

The confusion matrix for the Decision Tree Classifier Model shows that the model can distinguish between the different classes. The false positive category indicates that some of the failed landings are classified as successful.





Conclusions

- Higher flight numbers have higher success rates at each launch site
- Launch rate success increased from 2013 to 2020
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the highest success rates
- KSC LC-39A had the most successful launches
- The Decision Tree Classifier is the most accurate machine learning algorithm to predict successful launches

Appendix

Notebooks, data sets, and scripts are found in this GitHub repository

[GitHub Repository Link]

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Data Collection

- Select and parse the Spacex data (shown as the head of a pandas data frame)

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude	Latitude
0	1	2006-03-24	Falcon 1	20.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin1A	167.743129	9.047721
1	2	2007-03-21	Falcon 1	NaN	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2A	167.743129	9.047721
2	4	2008-09-28	Falcon 1	165.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin2C	167.743129	9.047721
3	5	2009-07-13	Falcon 1	200.0	LEO	Kwajalein Atoll	None None	1	False	False	False	None	NaN	0	Merlin3C	167.743129	9.047721
4	6	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366	28.561857

- Filter data to include only Falcon 9 launches

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial	Longitude
4	1	2010-06-04	Falcon 9	NaN	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0003	-80.577366
5	2	2012-05-22	Falcon 9	525.0	LEO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0005	-80.577366
6	3	2013-03-01	Falcon 9	677.0	ISS	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B0007	-80.577366
7	4	2013-09-29	Falcon 9	500.0	PO	VAFB SLC 4E	False Ocean	1	False	False	False	None	1.0	0	B1003	-120.610829
8	5	2013-12-03	Falcon 9	3170.0	GTO	CCSFS SLC 40	None None	1	False	False	False	None	1.0	0	B1004	-80.577366
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
89	86	2020-09-03	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	2	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1060	-80.603956
90	87	2020-10-06	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	3	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	13	B1058	-80.603956
91	88	2020-10-18	Falcon 9	15600.0	VLEO	KSC LC 39A	True ASDS	6	True	True	True	5e9e3032383ecb6bb234e7ca	5.0	12	B1051	-80.603956
92	89	2020-10-24	Falcon 9	15600.0	VLEO	CCSFS SLC 40	True ASDS	3	True	True	True	5e9e3033383ecbb9e534e7cc	5.0	12	B1060	-80.577366
93	90	2020-11-05	Falcon 9	3681.0	MEO	CCSFS SLC 40	True ASDS	1	True	False	True	5e9e3032383ecb6bb234e7ca	5.0	8	B1062	-80.577366

90 rows × 17 columns

Data Wrangling

- Check for missing values in the data set

- Replace missing values with the mean of the payload mass

```
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
PayloadMass_mean=data_falcon9['PayloadMass'].mean()
Pa
# Replace the np.nan values with its mean value
data_falcon9['PayloadMass'].fillna(value=
```

```
6123.547647058824
```

Web Scraping

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Web Scraping

- Check extracted column names
- Create a dictionary by parsing the launch HTML tables
- Create a dataframe from the dictionary

```
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)
```

Check the extracted column names

```
print(column_names)
```

```
['Flight No.', 'Date and time ( )', 'Launch site', 'Payload', 'Payload mass', 'Orbit', 'Customer', 'Launch outcome']
```

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

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

Data Analysis

- Use .value_counts() to determine the number of launches at each Launch Site, number and occurrence of each orbit, number and occurrence of mission outcome for each orbit type
- Create set of outcomes where the second stage did not land successfully
- Create a landing outcome label from the Outcomes column to represent successful and unsuccessful landings

```
# 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
```

```
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 = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class=[]

landing_class = np.where(df['Outcome'].isin(bad_outcomes), 0, 1)
landing_class
```

```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1])
```

EDA with Data Visualization with SQL

- Select distinct launch site names
- Select the first successful landing outcome in ground pad
- Rank the count of landing outcomes between 2010-06-04 and 2017-03-20 in descending order

```
%sql select min(Date) as first_successful_ground_pad_landing from SPACEXTBL where Landing_Outcome = "Success (ground pad)";
```

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

first_successful_ground_pad_landing
01/08/2018

```
%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
None

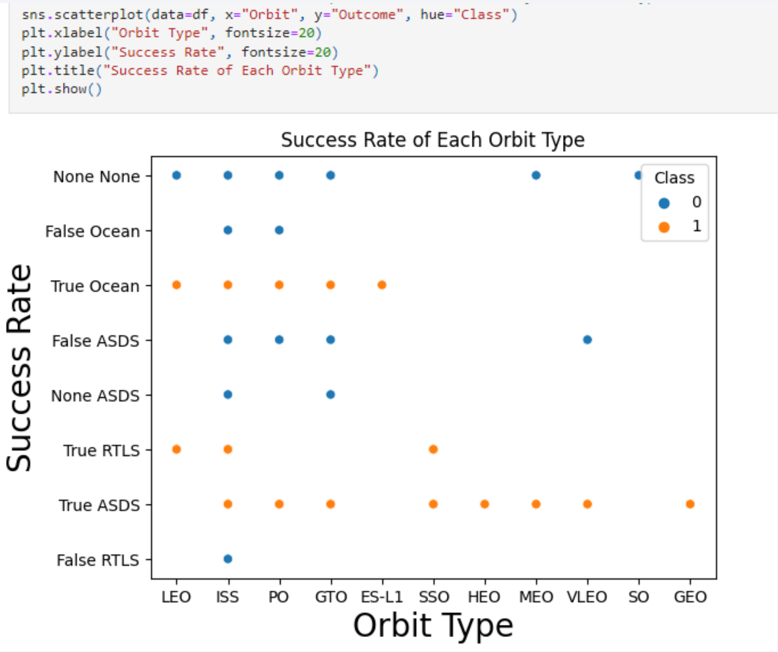
```
%sql SELECT Landing_Outcome, count(Landing_Outcome) from SPACEXTBL \
    where Date Between '04-06-2010' and '20-03-2017' \
    group by Landing_Outcome \
    order by count(Landing_Outcome) DESC;
```

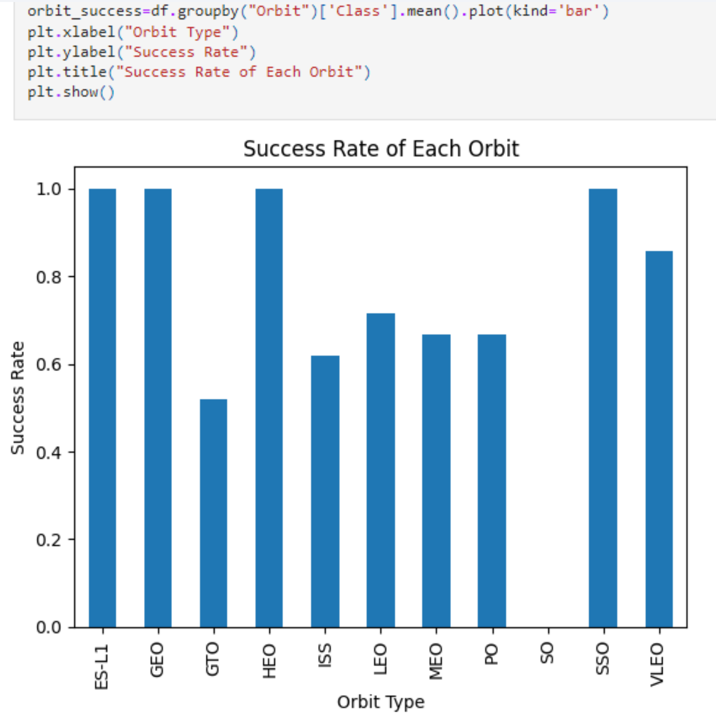
```
* sqlite:///my_data1.db
Done.
```

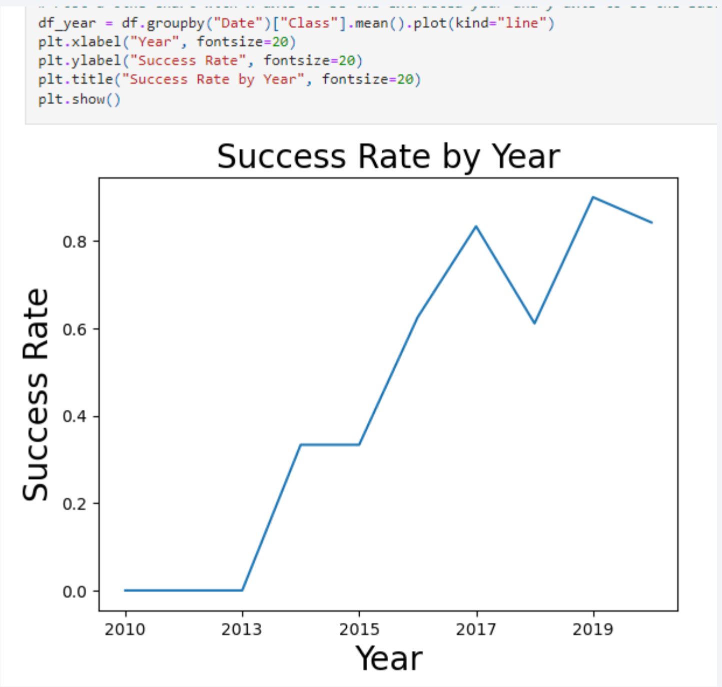
Landing_Outcome	count(Landing_Outcome)
Success	20
No attempt	10
Success (drone ship)	8
Success (ground pad)	7
Failure (drone ship)	3
Failure	3
Failure (parachute)	2
Controlled (ocean)	2
No attempt	1

EDA Data Visualization

- Scatter plot of success rate for each orbit type
- Bar chart of success rate for each orbit type
- Line chart of success rate by year







Site Location Analysis with Folium

- Mark all launch sites on a map using folium.Circle
- Mark the success/failed launches for each site
- Calculate the distances between a launch site and its landmarks

```
# Create a blue circle at NASA Johnson Space Center's coordinate with a popup label showing its name
circle = folium.Circle(nasa_coordinate, radius=1000, color='#d35400', fill=True).add_child(folium.Popup('NASA Johnson Spa
# Create a blue circle at NASA Johnson Space Center's coordinate with a icon showing its name
marker = folium.map.Marker(
    nasa_coordinate,
    # Create an icon as a text label
    icon=DivIcon(
        icon_size=(20,20),
        icon_anchor=(0,0),
        html='<div style="font-size: 12; color:#d35400;"><b>%s</b></div>' % 'NASA JSC',
        )
    )
site_map.add_child(circle)
site_map.add_child(marker)
```

```
# Add marker_cluster to current site_map
site_map.add_child(marker_cluster)

# for each row in spacex_df data frame
# create a Marker object with its coordinate
# and customize the Marker's icon property to indicate if this launch was successed or failed,
# e.g., icon=folium.Icon(color='white', icon_color=row['marker_color']
    # TODO: Create and add a Marker cluster to the site map
    # marker = folium.Marker(...)
for index, record in spacex_df.iterrows():
    folium.Marker(
        location=[record['Lat'], record['Long']],
        icon=folium.Icon(color=record['marker_color'], icon_color=record['marker_color'],
                         popup=record['class'])
    ).add_to(marker_cluster)
site_map
```

```
# find coordinate of the closet coastline
# e.g.,: Lat: 28.56367  Lon: -80.57163
# distance_coastline = calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon
launch_site_lat=34.63308
launch_site_lon=-120.61084
coastline_lat=34.63355
coastline_lon=-120.62599

distance_coastline=calculate_distance(launch_site_lat, launch_site_lon, coastline_lat, coastline_lon)
distance_coastline
```

```
1.3875219825106
```

Machine Learning Prediction

- Standardize the data
- Split into training data and test data
- Create a prediction model object using GridSearchCV and fit the model
- Calculate accuracy score using the score method

```
transform = preprocessing.StandardScaler()
X=transform.fit_transform(X)
X[0:5]
```

```
array([[-1.71291154e+00, -1.94814463e-16, -6.53912840e-01,
        -1.57589457e+00, -9.73440458e-01, -1.05999788e-01,
        -1.05999788e-01, -6.54653671e-01, -1.05999788e-01,
        -5.51677284e-01,  3.44342023e+00, -1.85695338e-01,
        -3.33333333e-01, -1.05999788e-01, -2.42535625e-01,
        -4.29197538e-01,  7.97724035e-01, -5.68796459e-01,
        -4.10890702e-01, -4.10890702e-01, -1.50755672e-01,
        -7.97724035e-01, -1.50755672e-01, -3.92232270e-01,
         9.43398113e+00, -1.05999788e-01, -1.05999788e-01,
        -1.05999788e-01, -1.05999788e-01, -1.05999788e-01,
        -1.05999788e-01, -1.05999788e-01, -1.05999788e-01,
        -1.05999788e-01, -1.05999788e-01, -1.05999788e-01,
        -1.05999788e-01, -1.05999788e-01, -1.05999788e-01,
        -1.05999788e-01, -1.05999788e-01, -1.05999788e-01,
```

```
X_train, X_test, Y_train, Y_test = train_test_split( X, Y, test_size=0.2, random_state=2)
```

we can see we only have 18 test samples.

```
Y_test.shape
```

```
(18,)
```

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
grid_search=GridSearchCV(estimator=tree, param_grid=parameters, cv=10)
tree_cv=grid_search.fit(X_train, Y_train)
```

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_lea
f': 1, 'min_samples_split': 5, 'splitter': 'best'}
accuracy : 0.9017857142857144
```

```
tree_cv.score(X_test, Y_test)
```

```
0.8888888888888888
```

We can plot the confusion matrix

```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```

Thank you!