IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Purnima H
01/04/2023

Outline

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

Executive Summary

- Summary of methodologies

Data Collection Using Space X API (REST API)

Data Collection using Web Scraping (Beautiful Soup)

Data Wrangling

Exploratory analysis using SQL and Data Visualization using Matplotlib and seaborn library

Launch Site Analysis by creating Ploty Dashboard and using Folium Maps

Result Prediction using Machine Learning

- Summary of all results

Data Visualization

Machine Learning Prediction accuracy

Introduction

Background:

The commercial space age is here, companies are making space travel affordable for everyone. SpaceX advertises Falcon 9 rocket launches on its website with a cost of 62 million dollars; other providers cost upwards of 165 million dollars each, much of the savings is because SpaceX can reuse the first stage.

In this project, we are trying to predict the successful landing of the first stage of the Falcon 9, thus determining the cost of the launch.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data collected using the SpaceX REST API
- Perform data wrangling
  - In the dataset, the successful landing is TRUE and unsuccessful landing is FALSE converted TRUE to 1 and FALSE to 0by creating a new label
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Classification models: Linear Regression, SVM, Decision Trees and KNN used for analysis

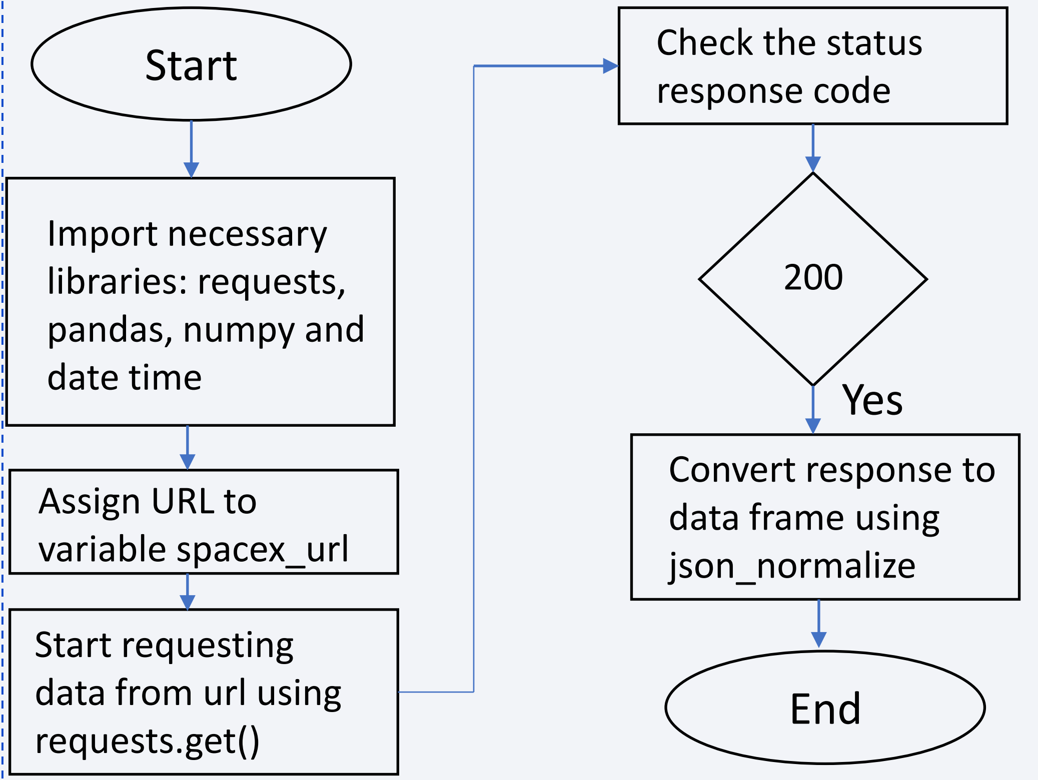
Data Collection

- Describe how data sets were collected.
- You need to present your data collection process use key phrases and flowcharts

Data Collection – SpaceX API

- Import the necessary libraries and request rocket launch data from SpaceX API using URL and *requests.get()*. Request and Parse the Space X launch data using GET request and using *json_normalize* method convert it into a dataframe for better understanding
- GitHub URL: https://github.com/purnimah/Data-Science-Capstone/blob/main/Final_Assignment%20(1).ipynb

Flow Chart



Data Collection – SpaceX API

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

In [89]:
```
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [90]:
```
response = requests.get(spacex_url)
```

Check the content of the response

In [91]:
```
print(response.content)
```

Task 1: Request and parse the SpaceX launch data using the GET request

To make the requested JSON results more consistent, we will use the following static response object for this project:

In [92]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_call_spacex_api.json'
```

We should see that the request was successfull with the 200 status response code

In [93]:
```
response.status_code
```

Out[93]: 200

Now we decode the response content as a Json using `.json()` and turn it into a Pandas dataframe using `.json_normalize()`

In [94]:
```
# 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

In [95]:
```
# Get the head of the dataframe
data.head()
```

Data Collection - Scraping

- Assign URL to variable and get a response using *requests.get()* function
- Create a Beautiful Soup object and parse html
- Retrieve title using soup objetc
- GitHub URL: https://github.com/purnimah/Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb

TASK 1: Request the Falcon9 Launch Wiki page from its URL

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

[7]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
```

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

[9]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.content, "html.parser")
```

Print the page title to verify if the `BeautifulSoup` object was created properly

[10]:
```
# Use soup.title attribute
soup.title
```

[10]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Data Wrangling

- Calculating Number of launches at each site using value_counts()
- Calculating number of re occurrence of launches from each orbit using value_counts()
- Calculating the number and occurence of mission outcome per orbit type using value_counts()
- Assign 0 and 1 to Failure and Successful missions respectively and create a new column class in the dataframe

GitHub: https://github.com/purnimah/Data-Science-Capstone/blob/main/Final_Assignment_2.ipynb

EDA with Data Visualization

- We try to find out the relationship between the dependent variable in the case the success and the failure of launch and the independent variables:

1. Relationship between: Flight Number and Payload Mass
2. Relationship between: Flight Number and Launch Site
3. Relationship between: Payload Mass and Launch Site
4. Relationship between: Success Rate and Orbit type
5. Relationship between: Payload Mass and Orbit type
6. Relationship between: Flight Number and Orbit type
7. Relationship between: Success Rate and Year

GitHub URL: https://github.com/purnimah/Data-Science-Capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_2_jupyter-labs-eda-dataviz.ipynb.jupyterlite.ipynb

EDA with SQL

- Connect to the data base
- Find out unique launch sites
- Finding out total payload mass by launches carried out by NASA
- 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
- Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.
- GitHub URL: https://github.com/purnimah/Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera_sqllite.ipynb

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

sum(PAYLOAD_MASS__KG_)
45596

avg(PAYLOAD_MASS__KG_)
2928.4

min("date")
01-05-2017

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

count("Mission_Outcome")
100

count("Mission_Outcome")
1

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

Build an Interactive Map with Folium

- Add the map of the site using the latitude and longitude co-ordinates.
- Using Folium Circle and Folium marker highlight the launching sites with text labels. So the launching sites are visible properly on the map *site_map.add_child(circle) site_map.add_child(marker)*
- Assigning green and red markers as per values in class 1 and 0 which indicates failure and successful mission. Making a marking cluster so that all the markers can be mapped at the same site location (having same latitude and longitude) using *MarkerCluster()*
- Next we find out the proximity of city, highway, costal area, railroad from the launching site.

GitHub URL: https://github.com/purnimah/Data-Science-Capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_3_lab_jupyter_launch_site_location.jupyterlite.ipynb

Build a Dashboard with Plotly Dash

- Building an interactive dash board, where the site of launch can be selected and the pie chard will display the success and failure rate of the mission
- Along with that a Bar to select the pay load mass is provided. For which the relationship between the Payload mass and success rate scatter plot is displayed which describes the relationship between the payload mass and the landing success.

GitHub URL: https://github.com/purnimah/Data-Science-Capstone/blob/main/spacex_dash_app.py

Predictive Analysis (Classification)

- Create an array using to_numpy for the class and assign to Y and the data frame to X
- Standardize the data by using fit_transform() to transform it into the data that is more suitable.
- Split the data to training at test set for the X and Y using train_test_split, the modelsare trained and hyperparameters are obtained using GridsearchCV
- Applied various classification technique: Logistic Regression, Decision Trees, SVM, and KNN fount the accuracy using .score method

For all the classification methods the accuracy obtained is 83.33%

Create array and Standardize data → Split the data to training and test set → Train the model using training data → Find the accuracy for test data using score

- GitHub URL: https://github.com/purnimah/Data-Science-Capstone/blob/main/IBM-DS0321EN-SkillsNetwork_labs_module_4_SpaceX_Machine_Learning_Prediction_Part_5.jupyterlite.ipynb

Results

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

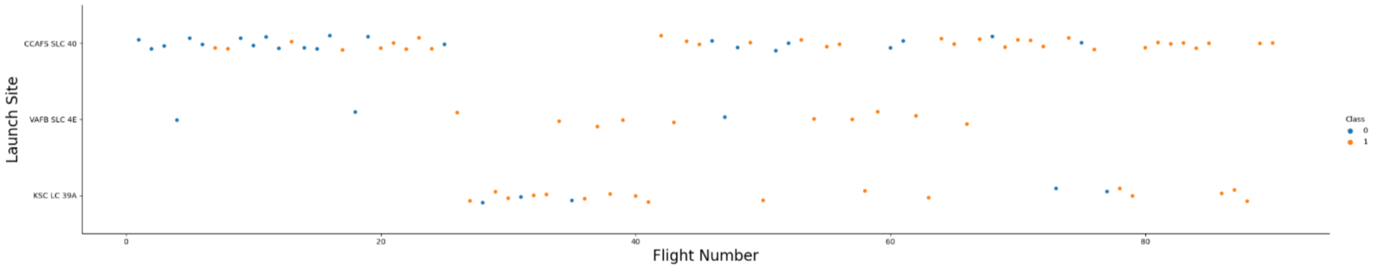
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

Relationship between: Flight Number and Launch Site

```
n [27]:
# Plot a scatter point chart with x axis to be Flight Number and y axis to be the launch site, and hue to be the class value
sns.catplot(y="LaunchSite", x="FlightNumber",hue='Class',data=df, aspect=5)
plt.xlabel("Flight Number",fontsize=20)
plt.ylabel("Launch Site",fontsize=20)
plt.show()
```



Class – 0 - Unsuccessful (Blue) | Class – 1 - Successful (Yellow)

From the scatter plot we can infer that more successful landings were done from site CCAFS SLC 40 and least were from VAFB SLC 4E. Initial landings were successful from CCAFS and later ON more failure results can be seen

And the bubbles in the scatter represent success and failure in the linear/non linear relationship

Payload vs. Launch Site

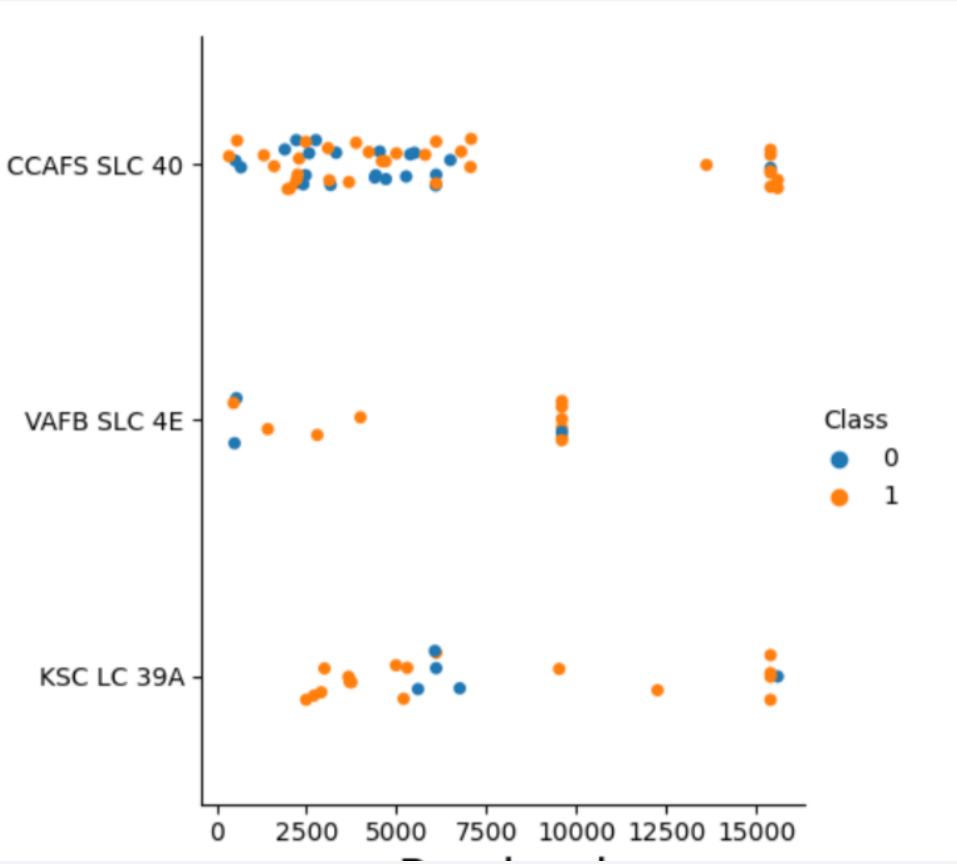
Relationship between: Payload Mass and Launch Site

Class – 0 - Unsuccessful (Yellow) | Class – 1 - Successful (Blue)

From the scatter plot we can infer less the pay load mass more successful landings, and more successful landings are from site CCAFS SLC 40 and less launches were from VAFB SLC 4E. No launches were done from VAFB SLC 4E with payload mass > 10000

Initial landings were successful from CCAFS and later ON more failure results can be seen on higher payload mass

And the bubbles in the scatter represent success and failure in the linear/non linear relationship



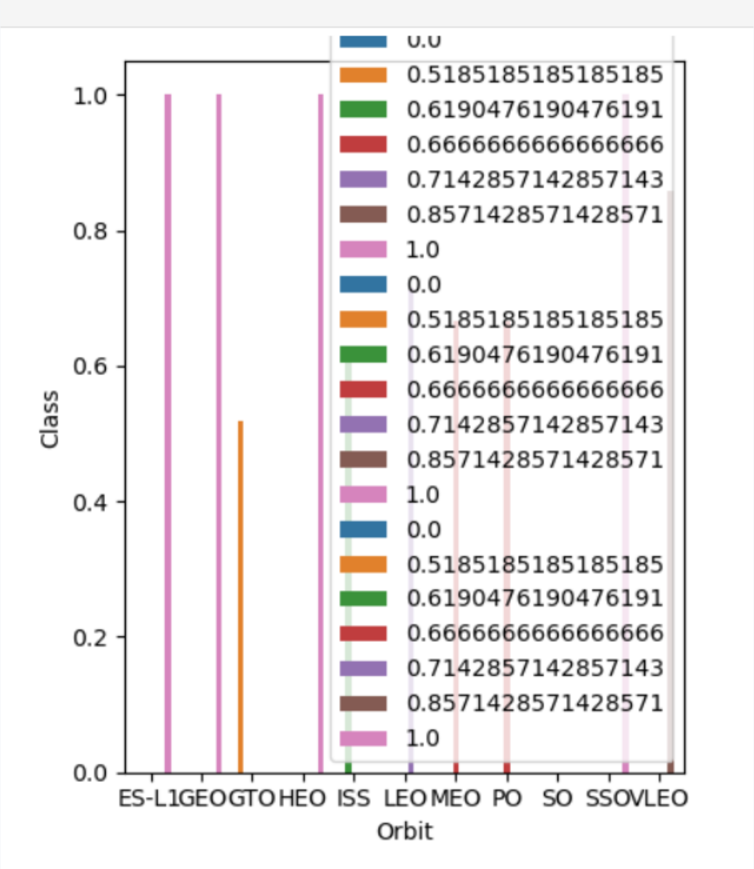
Success Rate vs. Orbit Type

Relationship between: Success Rate and Orbit type

From the bar plot we can infer that ES-L1,GEO, GTO, ISS, LEO, VELO have a higher successful landing rate. HEO has 50% success rate

Initial landings were successful from CCAFS and later ON more failure results can be seen on higher payload mass

Bar Plots are excellent for categorical representation as orbits is a categorical variable



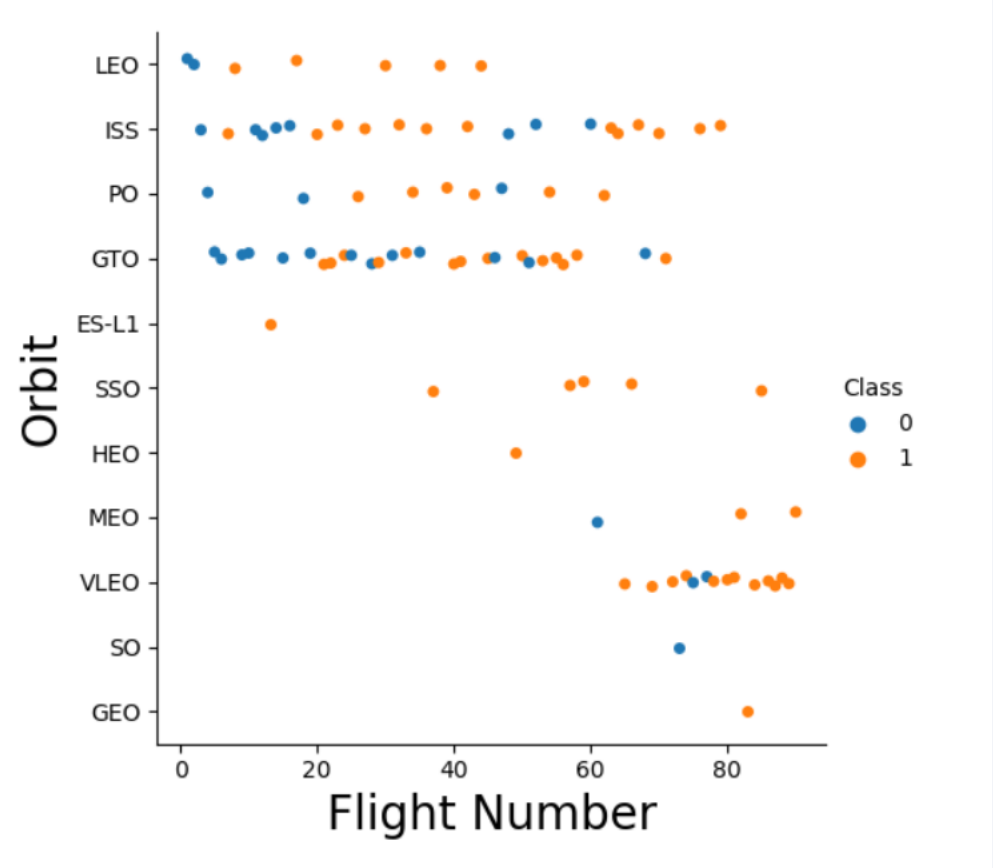
Flight Number vs. Orbit Type

Relationship between: Flight Number and Orbit type

From the scatter plot we can infer that LEO, ISS, PO, GTO had successful landings hence number of flights were increased.

Later on we can see that flights from VLEO were increased and successful landings were obtained hence there is a sudden increase in number of flights from VELO orbit.

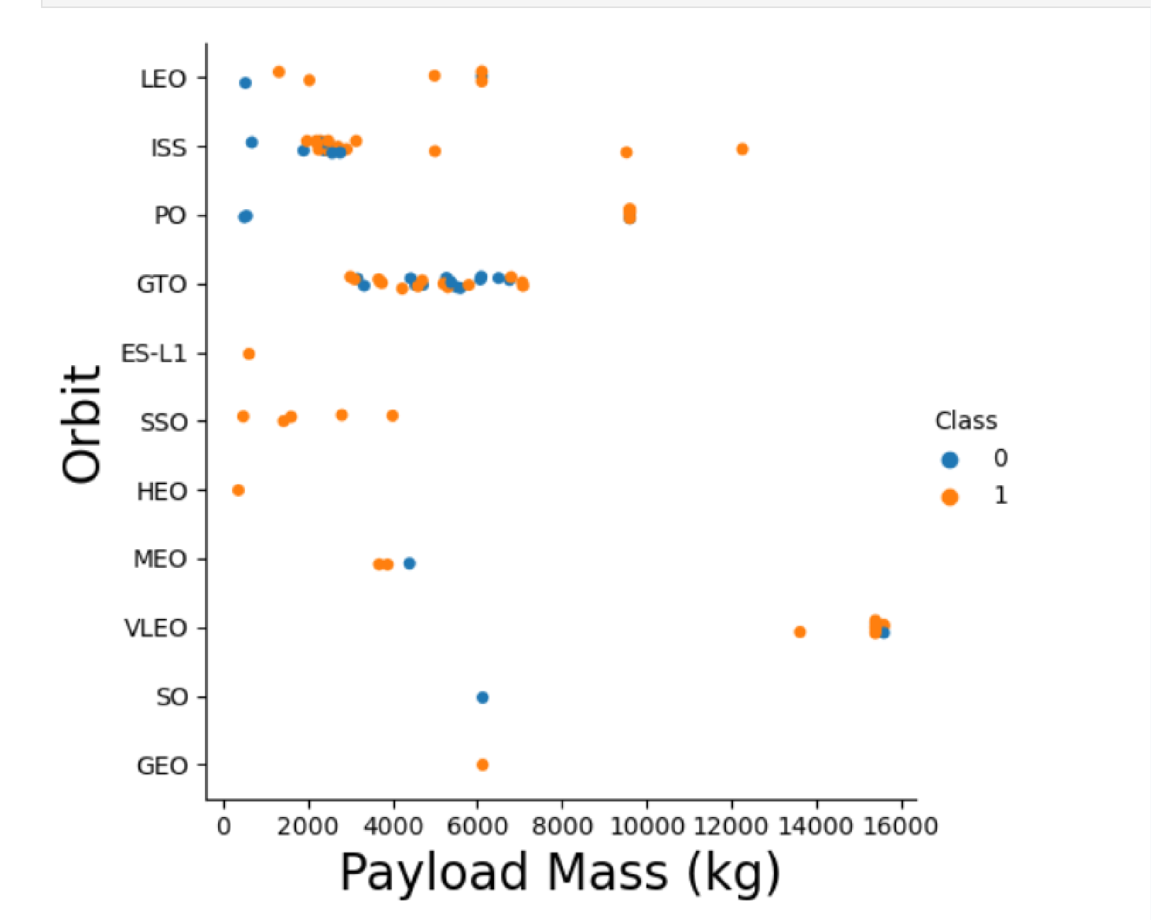
No landing failure in case of GEO, ES-L1, SSO, HEO can be seen



Payload vs. Orbit Type

Relationship between: Payload Mass and Orbit type

From the scatter plot we can infer that SSEO has successful landing in case of lower pay load mass. VELO has higher landing success in case of play load mass > 14000

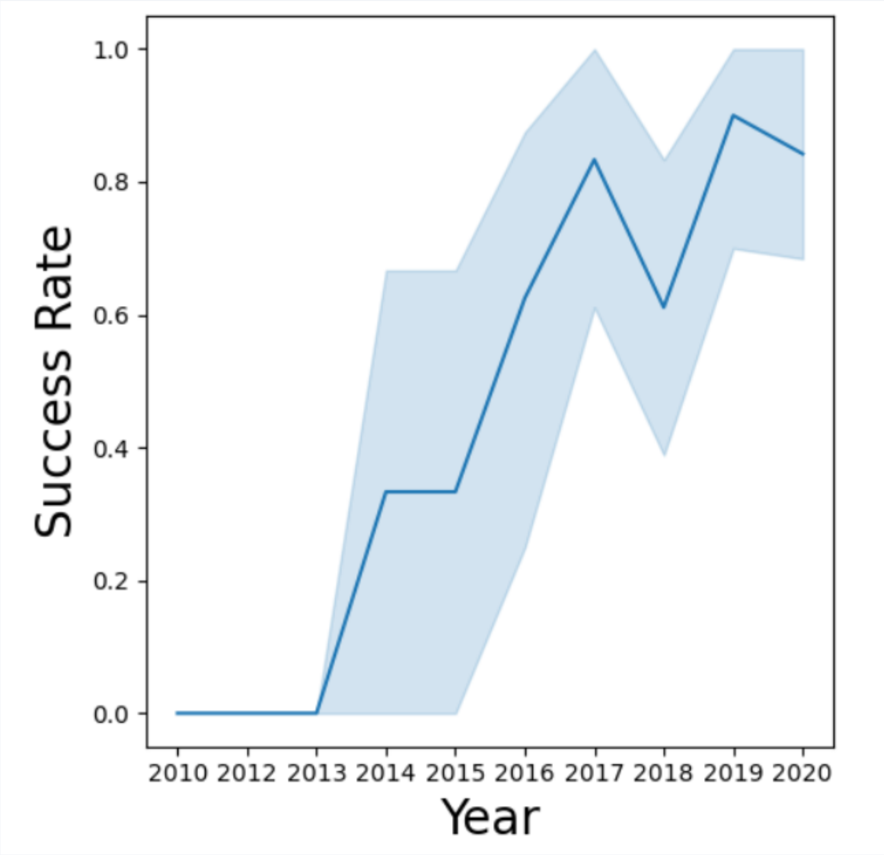


Launch Success Yearly Trend

Relationship between: Success Rate and Year

From the line plot we can infer that the success rate over the period of time has been linear. Due to advancement in science and technology higher landing success can be seen since 2010

Line Plot is the best representation as you can see the line rise and fall, to make out the relationship.



All Launch Site Names

- Find the names of the unique launch sites

Using DISTINCT unique names of the launch sites obtained from the table SPACETBL

Task 1

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

In [54]:
```
%sql select distinct "Launch_Site" from SPACEXTBL;
```

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

Out[54]:

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

Launch Site Names Begin with 'CCA'

- Find 5 records where launch sites begin with `CCA`

5 Launch sites with string CAA displayed using “like” and “Limit”

Task 2

Display 5 records where launch sites begin with the string 'CCA'

In [55]:

```
%sql select distinct "Launch_Site" from SPACEXTBL where "Launch_site" like "%CCA%" limit 5;
```

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

Out[55]:

Launch_Site
CCAFS LC-40
CCAFS SLC-40

Total Payload Mass

- Calculate the total payload carried by boosters from NASA

Total payload data of NASA displayed using sum()

Task 3

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

```
In [56]: %sql select sum(PAYLOAD_MASS__KG_) from SPACEXTBL where "Customer" = "NASA (CRS)";
```

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

Out[56]:

sum(PAYLOAD_MASS__KG_)
45596

Average Payload Mass by F9 v1.1

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

Average payload mass for booster version F9 v1.1 displayed using avg()

Task 4

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

In [57]:
```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where "Booster_Version" = "F9 v1.1"
```

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

Out[57]:

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

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

Date of first successful landing outcome displayed using min()

Task 5

List the date when the first succesful landing outcome in ground pad was acheived.

Hint:Use min function

In [65]:
```
%sql select min("date") from SPACEXTBL where "Landing _outcome" = "Success (ground pad)";
```

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

Out[65]:

min("date")
01-05-2017

Successful Drone Ship Landing with Payload between 4000 and 6000

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

List of successfully landed booster with pay load mass between 4000 and 6000 displayed using “between”

Task 6

List the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000

In [66]:
```
%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.
```

Out[66]:

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

Total Number of Successful and Failure Mission Outcomes

- Calculate the total number of successful and failure mission outcomes

Total successful mission = 100 and total failure mission = 1

Task 7

List the total number of successful and failure mission outcomes

In [60]:
```
%sql select count("Mission_Outcome") from SPACEXTBL where "Mission_Outcome" like "%Success%";
```

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

Out[60]:

count("Mission_Outcome")
100

In [61]:
```
%sql select count("Mission_Outcome") from SPACEXTBL where "Mission_Outcome" like "%Failure%";
```

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

Out[61]:

count("Mission_Outcome")
1

Boosters Carried Maximum Payload

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

Using a subquery and max () a list of booster with maximum payload displayed

Task 8

List the names of the booster_versions which have carried the maximum payload mass. Use a subquery

In [62]:

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

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

Out[62]:

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

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

Task 9

List the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

Note: SQLLite does not support monthnames. So you need to use substr(Date, 4, 2) as month to get the months and substr(Date,7,4)='2015' for year.

In [67]:
```
%sql select substr(Date,4,2) , "Booster_Version","Launch_Site" where "Landing _Outcome" = "Failure (drone ship)" and substr(Date,7,4) = "2015";
```

```
 * sqlite:///my_data1.db
(sqlite3.OperationalError) no such column: Date
[SQL: select substr(Date,4,2) , "Booster_Version","Launch_Site" where "Landing _Outcome" = "Failure (drone ship)" and substr(Date,7,4) = "2015";]
(Background on this error at: http://sqlalche.me/e/e3q8)
```

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

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

- List of landing outcome failure using “order by” and “between”

Task 10

Rank the count of successful landing_outcomes between the date 04-06-2010 and 20-03-2017 in descending order.

In [64]:
```
%sql select "Landing _Outcome" from SPACEXTBL where "Date" BETWEEN "04-06-2010" and "20-02-2017" order by "Date" desc;
```

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

Out[64]:

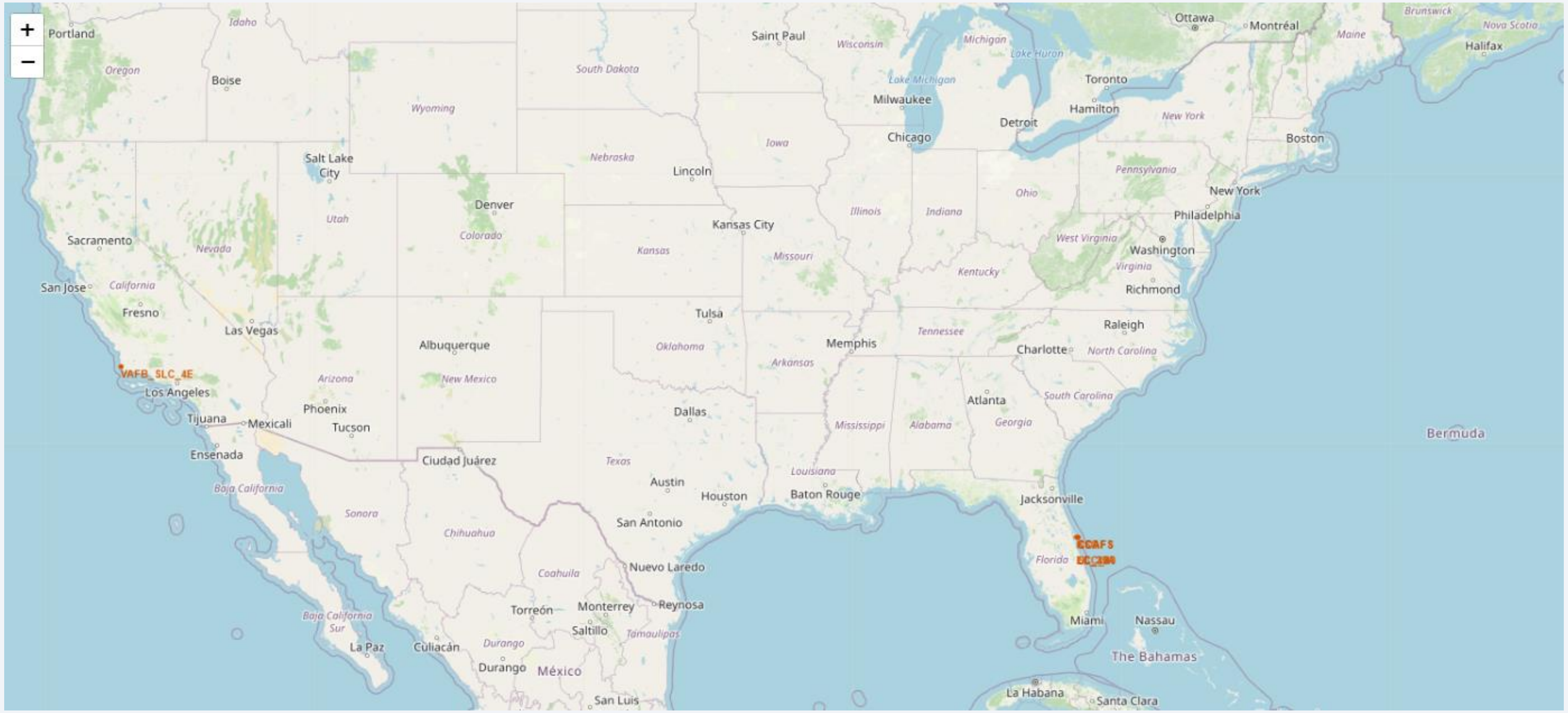
Landing _Outcome
Success (ground pad)
No attempt
Success
Success
Success (ground pad)
Success (drone ship)
Controlled (ocean)
Failure
Success
Failure
Failure (drone ship)
Success
No attempt
Success (ground pad)
Success
Failure (drone ship)
No attempt
Success (ground pad)
Success (drone ship)
Controlled (ocean)
Failure (drone ship)
Success (drone ship)
Success
Success
Success
Success (drone ship)

Section 3

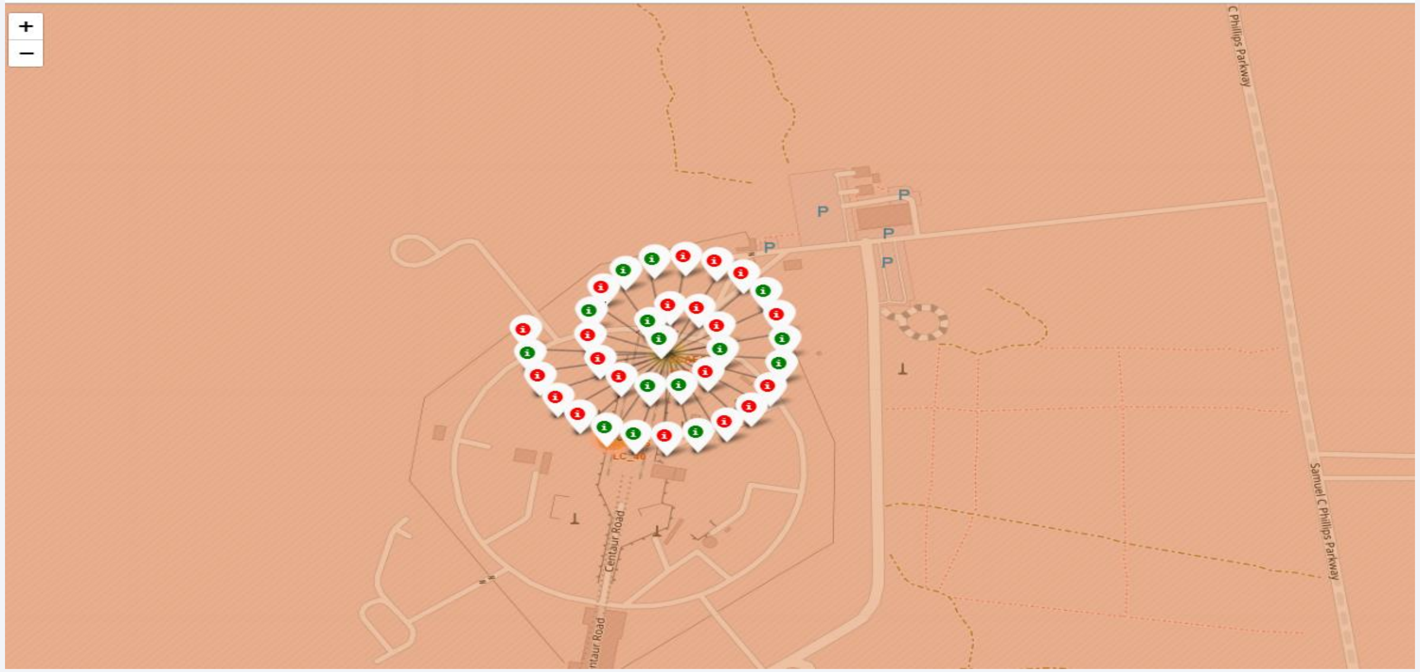
Launch Sites Proximities Analysis

Launch Sites Marking

- Marking of the 4 Launch sites on the map using folium using .add_child()
- As seen from the map the launching sites are near the coastal lines

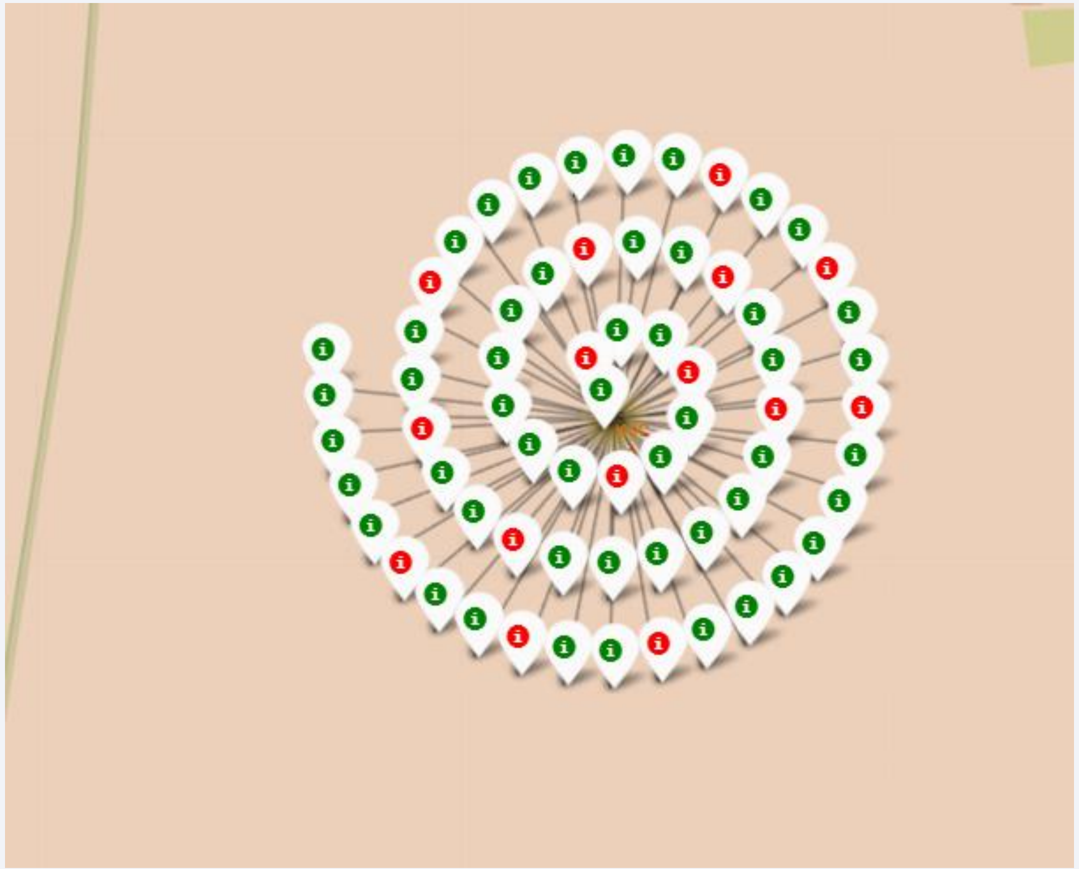


Marking Cluster on Map of Site CCAF SLC 40



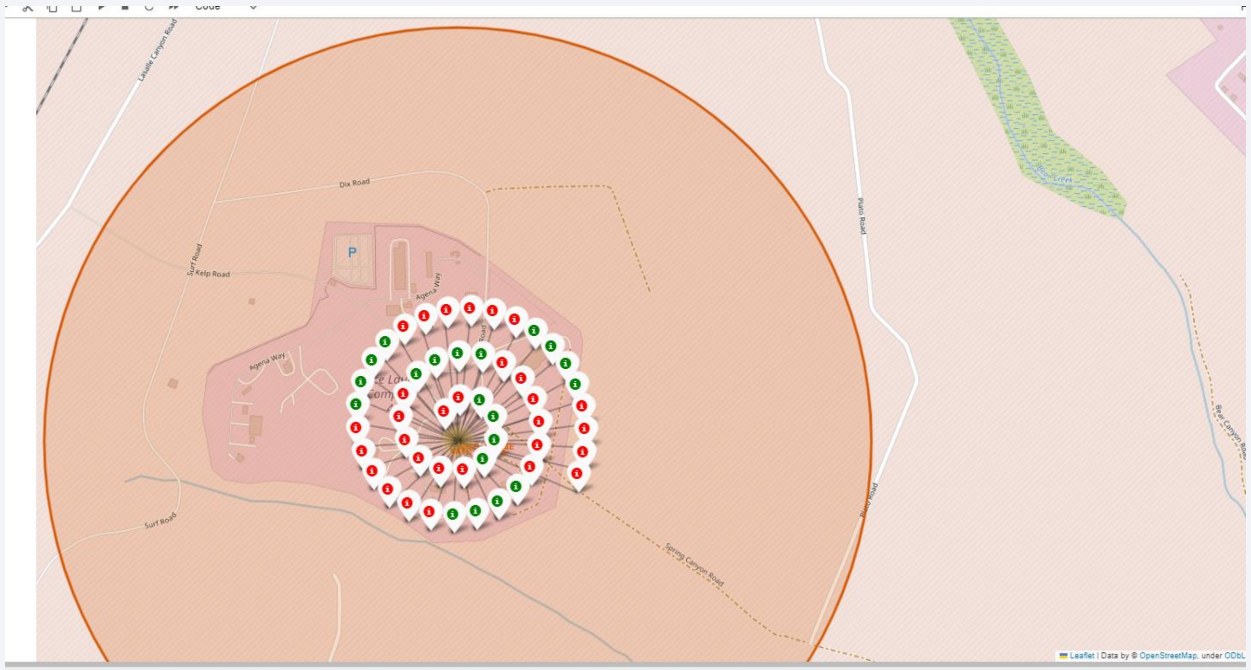
Green marker indicate successful landing.
Red marker indicate failure

Marking Cluster on Map of Site KSC LC 39A



As you can see from the markings it has more successful landing rate than other sites

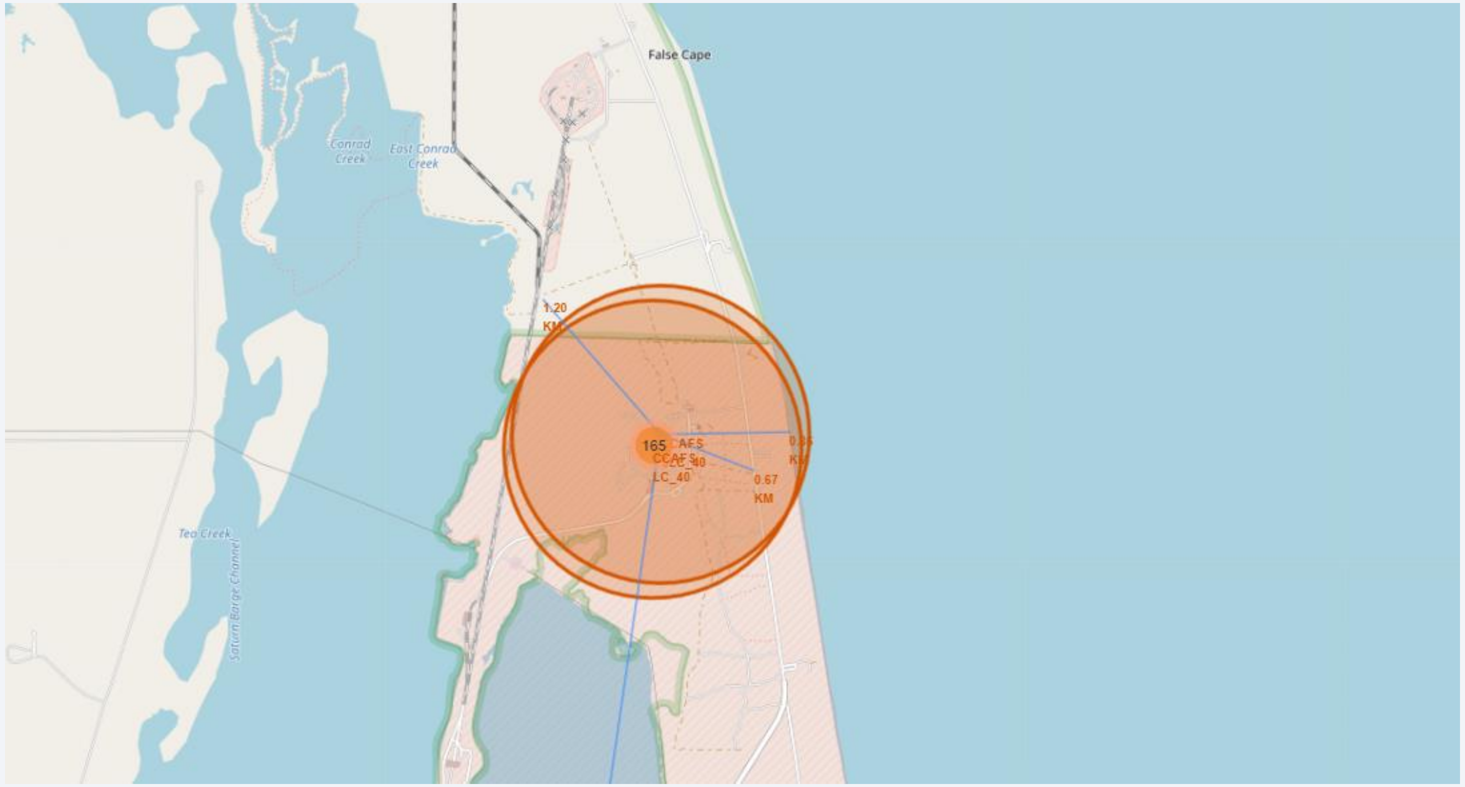
Marking Cluster on Map of Site VAFB SLC 4E



Green marker indicate successful landing.
Red marker indicate failure

Proximities from Site CCAF SLC 40

Marking of nearest Railway road, Highway road, City and coastal Line. We can observe that the launch site is away from the city for the safety and near to railroad and highway for easier access to transport facility. The launch site is near coastal area so that in case of failure the parts should not fall on built land

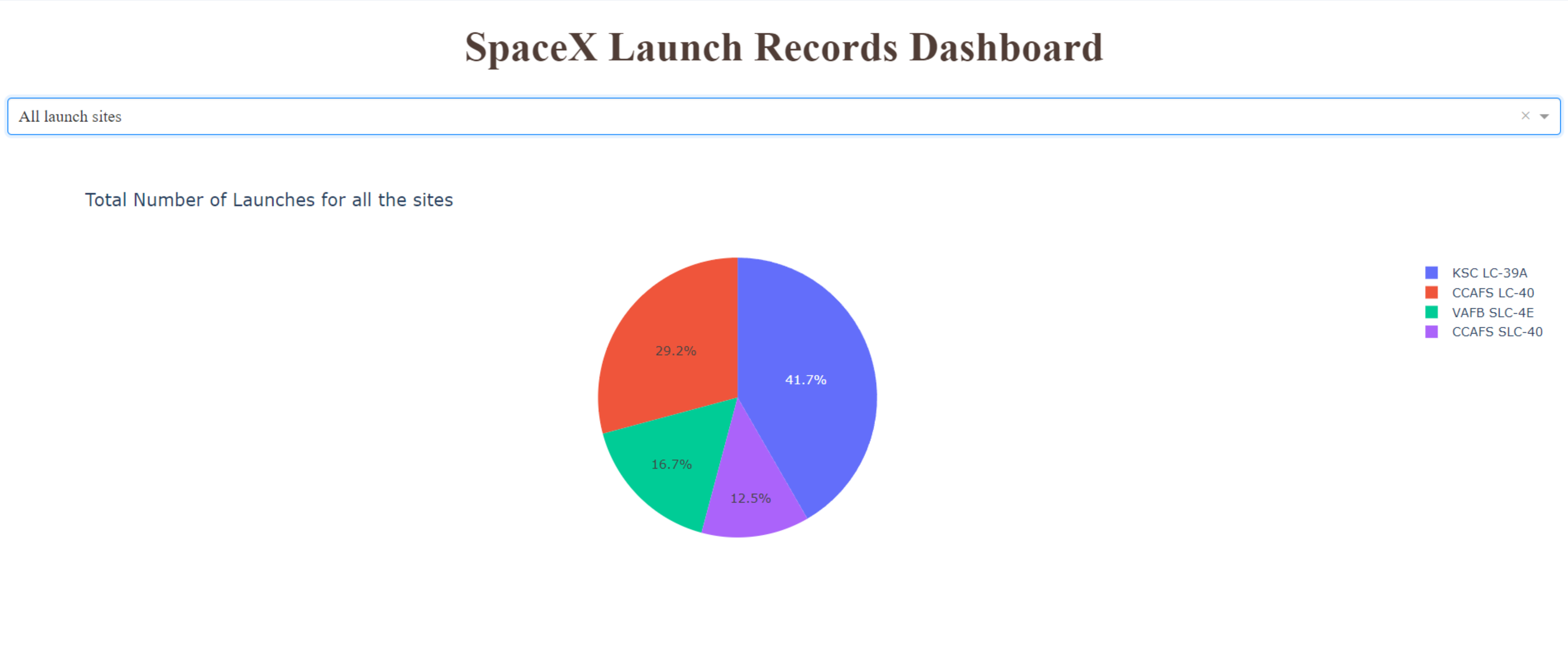


Section 4

Build a Dashboard with Plotly Dash

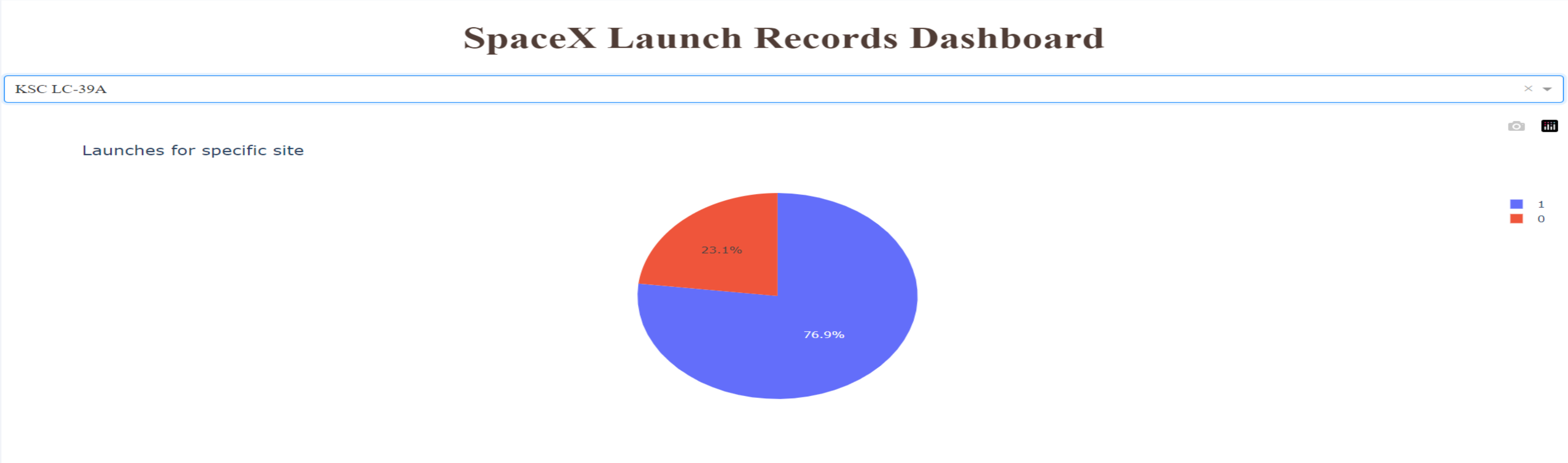
Pie Chart for all Launching sites

From the dashboard you can infer that the site KSC LC-39A has most successful landing rate. There is an option on the top slider to select and view data for each launching site



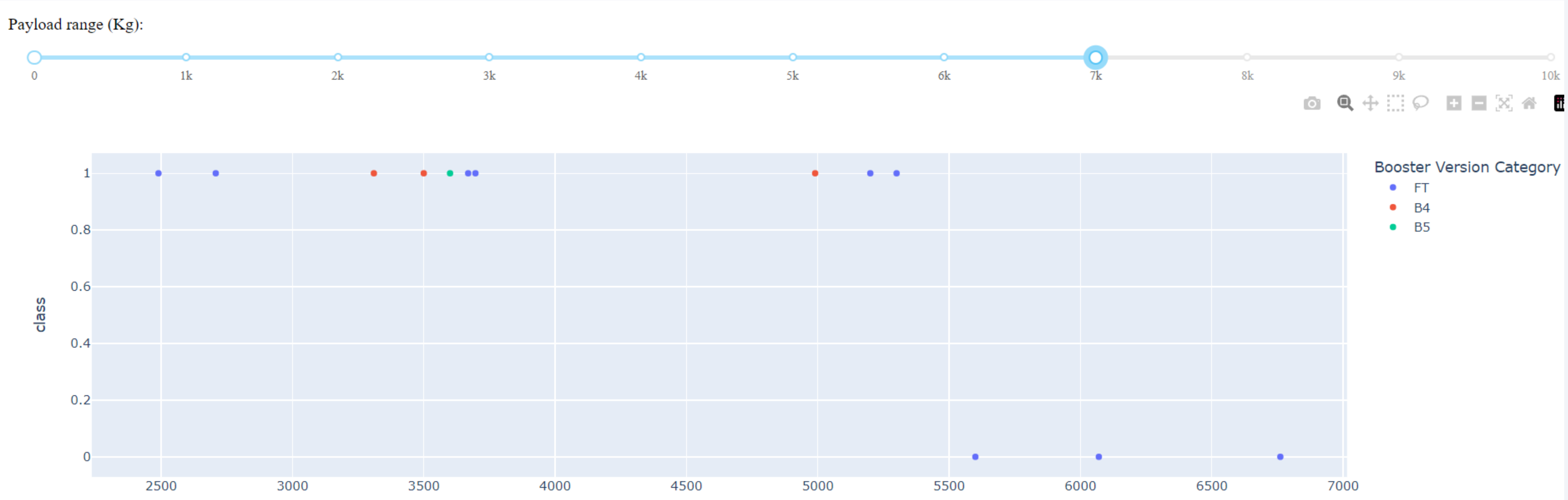
Pie Chart for KSC LC-39A

KSC LC 39A has successful landing of 76.9%



Payload vs Launch Outcome

From the graph we can infer that for pay load > 5500 there is no successful landing



Section 5

Predictive Analysis (Classification)

Classification Accuracy

- Accuracy for the Logistic Regression:

Calculate the accuracy on the test data using the method `score` :

```
[129]: logreg_cv.score(X_test,Y_test)
[129]: 0.8333333333333334
```

- Accuracy for SVM:

Calculate the accuracy on the test data using the method `score` :

```
[134]: svm_cv.score(X_test,Y_test)
[134]: 0.8333333333333334
```

- Accuracy for Decision Trees:

Calculate the accuracy of tree_cv on the test data using the method `score` :

```
[139]: tree_cv.score(X_test, Y_test)
[139]: 0.8333333333333334
```

- Accuracy for the KNN:

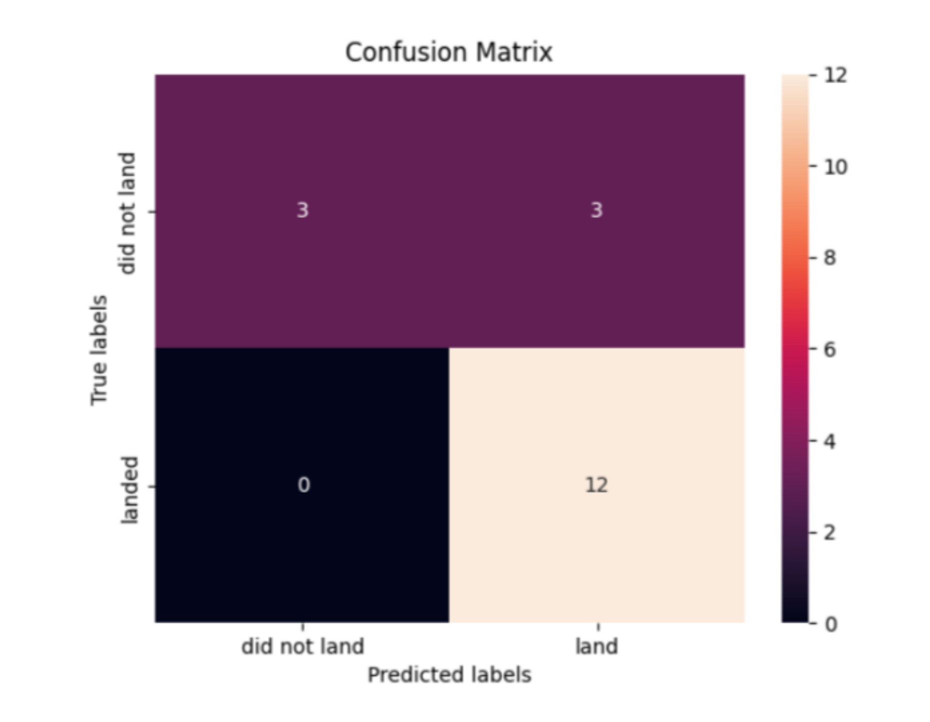
Calculate the accuracy of knn_cv on the test data using the method `score` :

```
[145]: knn_cv.score(X_test,Y_test)
[145]: 0.8333333333333334
```

The accuracy for all the classification methods is the same for the test data that is 83.33%

Confusion Matrix

The confusion matrix for all the classification models is same. We can see that the major issue is the false positives that is obtained



Conclusions

In conclusion, after all the analysis, we can infer that the success of the launch depends on various factors

All launching sites had different success rate. The most KSC LC 39A has the most success landing rate, followed by CCAF LC 40A

We can also infer that the successful landing rate for less payload is more can be seen from the Pay Load Vs Launch Site Plot

The orbits ES-L1, GEO, HEO, SSO has the highest success rate of 100%

The success rate of launching is increasing linearly

From the maps we can observe that the launch sites are located near the coastal areas so that in case of any failure the parts fallen from the rocket should not damage the constructed surface. Also they are away from the cities for public safety and near to the road and rail for faster transport.

Appendix

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

Thank you!