IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

SpaceX Falcon-9 first stage Landing Prediction

Hardikkumar Patel
17 June 2025

Outline

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

Executive Summary

- In this project, we will predict if the Falcon 9 first stage will land successfully or not through various machine learning models.
- Summary of methodologies:
  - Data collection and wrangling
  - Exploratory data analysis
  - Data visualization
  - Creating model of machine learning
- Our graph shows that some features of the rocket launches have a correlation with the outcome of the launches.
- In conclusion, Decision Tree machine learning model would be the go-to model as it is giving higher accuracy.

Introduction

- In this capstone, we will predict if the Falcon 9 first stage will land successfully. 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. In this lab, you will collect and make sure the data is in the correct format from an API. The following is an example of a successful and launch.
- We are trying to find the answer for success or failure landing first stage of rocket which features of the given data affects the most, and how it is impacting the outcome.

Section 1

Methodology

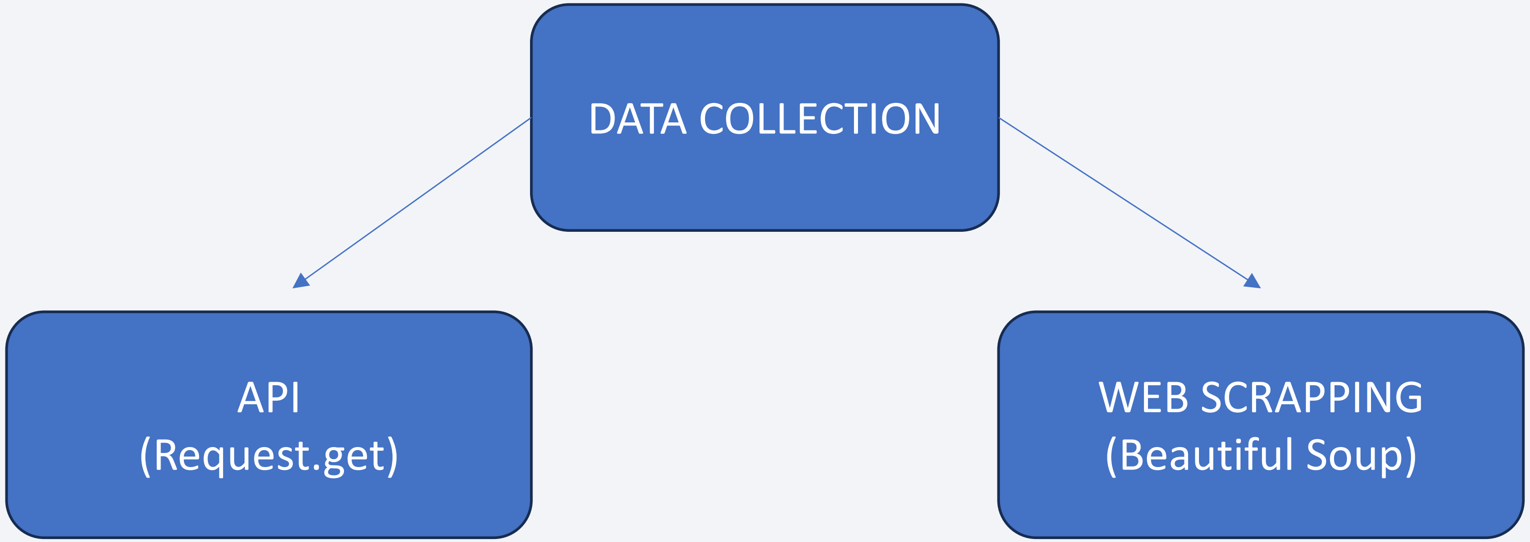
Methodology

Executive Summary

- Data collection methodology:
  - Data collected via API and web scrapping
- Perform data wrangling
  - We used Numpy, Pandas for the data wrangling
- Perform exploratory data analysis (EDA) using visualization and SQL
  - We used Matplotlib, seaborn for the exploratory data analysis
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - To build model we used Logistics regression, SVM, KNN and Decision Tree algorithms, tuned with GridsearchCV, evaluated through accuracy score.

Data Collection

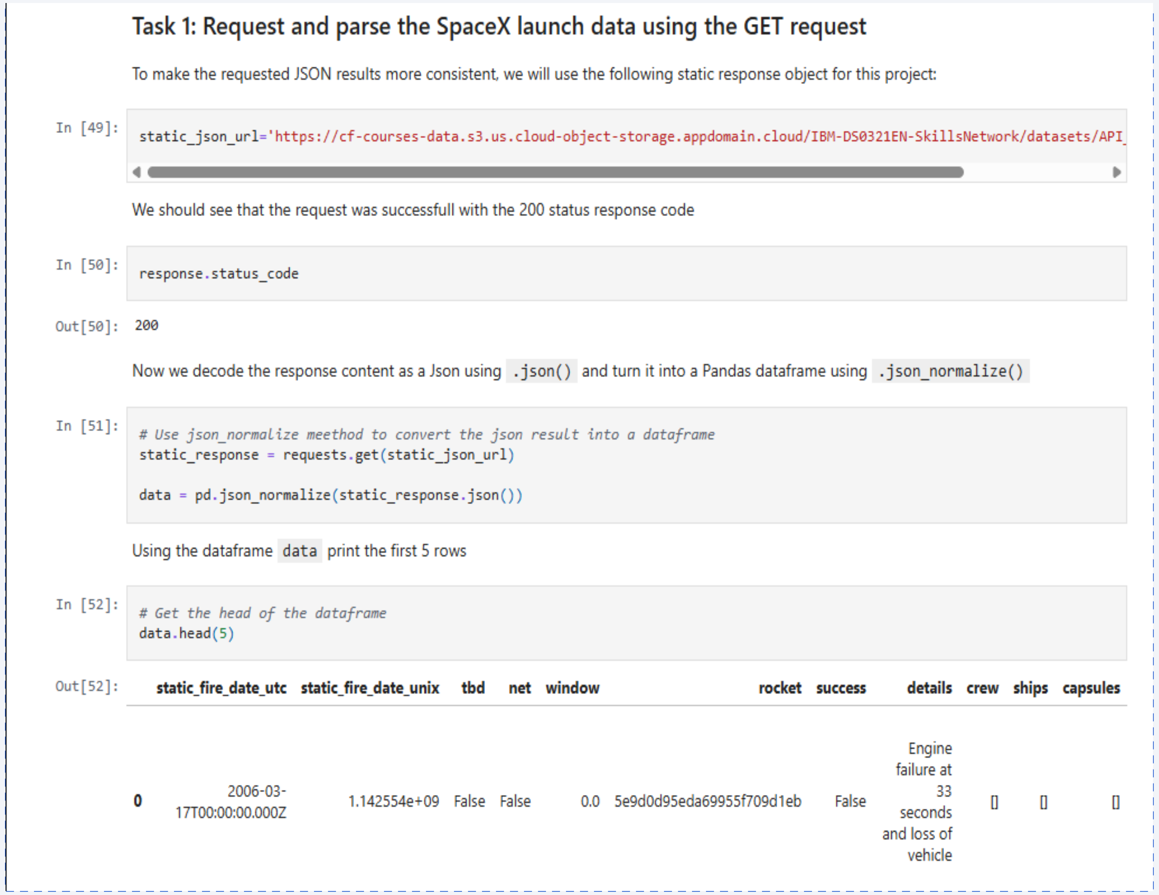
- Data sets were collected via API, web scrapping from Wikipedia for this project.



Data Collection – SpaceX API

API
(Request.get)

- GitHub URL of the completed SpaceX API calls notebook is https://github.com/Hardikrpatel243/Applied-DS-Capstone-Project/blob/main/1%20Spacex-data-collection-API.ipynb



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 [49]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/API_
```

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

In [50]:
```
response.status_code
```

Out[50]: 200

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

In [51]:
```
# Use json_normalize meethod to convert the json result into a dataframe
static_response = requests.get(static_json_url)

data = pd.json_normalize(static_response.json())
```

Using the dataframe `data` print the first 5 rows

In [52]:
```
# Get the head of the dataframe
data.head(5)
```

Out[52]:

	static_fire_date_utc	static_fire_date_unix	tbd	net	window	rocket	success	details	crew	ships	capsules
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	False	0.0	5e9d0d95eda69955f709d1eb	False	Engine failure at 33 seconds and loss of vehicle	[]	[]	[]

Data Collection - Scraping

WEB SCRAPPING (Beautiful Soup)

- GitHub URL of the completed web scraping notebook is https://github.com/Hardikrpatel243/Applied-DS-Capstone-Project/blob/main/2%20Spacex-data-collection-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.

In [22]:
```python
# use requests.get() method with the provided static_url
# assign the response to a object

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

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

In [23]:
```python
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response, 'lxml')
```

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

In [24]:
```python
# Use soup.title attribute
print(soup.title.string)
```

```
List of Falcon 9 and Falcon Heavy launches - Wikipedia
```

TASK 2: Extract all column/variable names from the HTML table header

Next, we want to collect all relevant column names from the HTML table header

Let's try to find all tables on the wiki page first. If you need to refresh your memory about `BeautifulSoup`, please check the external reference link towards the end of this lab

In [ ]:
```python
# Use the find_all function in the BeautifulSoup object, with element type `table`
# Assign the result to a list called `html_tables`

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

Data Wrangling

- In this, we will perform some EDA to find patterns in the data such as calculating the number of launches on each site.

- GitHub URL of your completed data wrangling related notebooks is https://github.com/Hardikrpatel243/Applied-DS-Capstone-Project/blob/main/3%20Spacex-Data%20wrangling.ipynb

In [4]:
```
df.dtypes
```

Out[4]:
```
FlightNumber        int64
Date               object
BoosterVersion     object
PayloadMass       float64
Orbit              object
LaunchSite         object
Outcome            object
Flights             int64
GridFins             bool
Reused               bool
Legs                 bool
LandingPad         object
Block             float64
ReusedCount         int64
Serial             object
Longitude         float64
Latitude          float64
dtype: object
```

TASK 1: Calculate the number of launches on each site

The data contains several Space X launch facilities: Cape Canaveral Space Launch Complex 40 **VAFB SLC 4E** , Vandenberg Air Force Base Space Launch Complex 4E **(SLC-4E)**, Kennedy Space Center Launch Complex 39A **KSC LC 39A** .The location of each Launch Is placed in the column `LaunchSite`

Next, let's see the number of launches for each site.

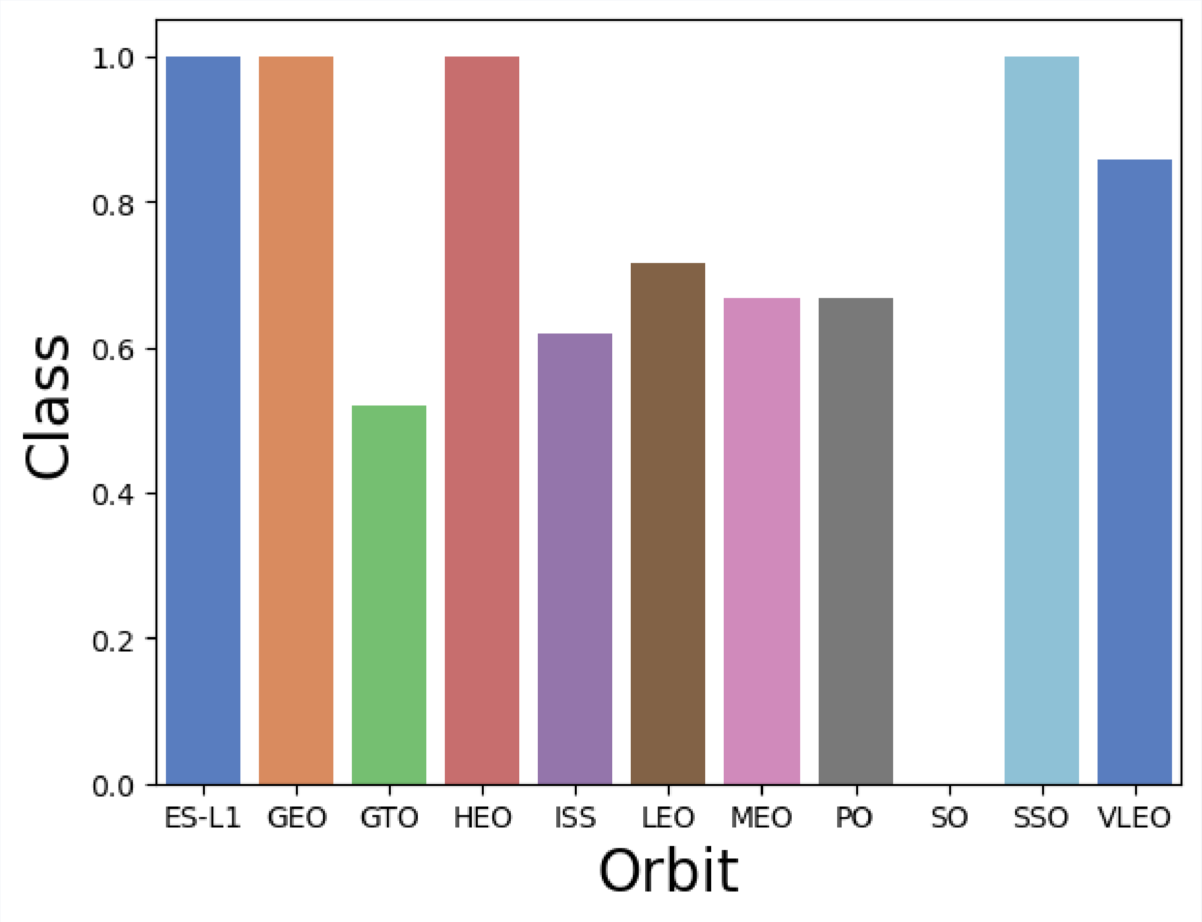
Use the method `value_counts()` on the column `LaunchSite` to determine the number of launches on each site:

In [5]:
```
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()
```

Out[5]:
```
LaunchSite
CCAFS SLC 40    55
KSC LC 39A      22
VAFB SLC 4E     13
Name: count, dtype: int64
```

EDA with Data Visualization

- We have explored the relationship between success rate and orbit type.
- GitHub URL of your completed EDA with data visualization notebook is https://github.com/Hardikrpatel243/Applied-DS-Capstone-Project/blob/main/5%20EDA-data-visualization.ipynb



EDA with SQL

- Loaded dataset in a Db2 database.
- Executed SQL queries to dip dive into dataset.
  - Find unique launch sites of the space mission.
  - How much total payload mass carried by boosters launched by NASA.
  - How much average payload mass carried by booster version F9 v1.1
  - When the first successful landing outcome in ground pad was achieved.
- GitHub URL of your completed EDA with SQL notebook is https://github.com/Hardikrpatel243/Applied-DS-Capstone-Project/blob/main/4%20EDA-sql-sqllite.ipynb

Build an Interactive Map with Folium

- We created map objects such as markers, circles, lines and added to a folium map.
- For the differentiation of success and failure of the launch outcome for each site we marked 1 and 0 respectively along with the green and red marker color so one can visualize the outcome by each site.
- GitHub URL of your completed interactive map with Folium map is https://github.com/Hardikrpatel243/Applied-DS-Capstone-Project/blob/main/6%20Launch-site-location%20with%20Folium.ipynb

Build a Dashboard with Plotly Dash

- We created dashboard application with Plotly Dash module.
- In the application we plotted pie chart to show total successful launches count as per the dropdown selection of sites.

- GitHub URL of your completed Plotly Dash aap is https://github.com/Hardikrpatel243/Applied-DS-Capstone-Project/blob/main/7%20spacex_dash_app.py

Predictive Analysis (Classification)

- We loaded data into pandas dataframe and numpy, did transformation with standard scaler and split data into training and testing.
- Created different machine learning model and tuned different hyperparameter using GridSearchCV.
- As per the accuracy score, we concluded Decision Tree is the suitable model for this dataset.
- GitHub URL of your completed predictive analysis is https://github.com/Hardikrpatel243/Applied-DS-Capstone-Project/blob/main/8%20SpaceX-Machine-Learning-Prediction.ipynb

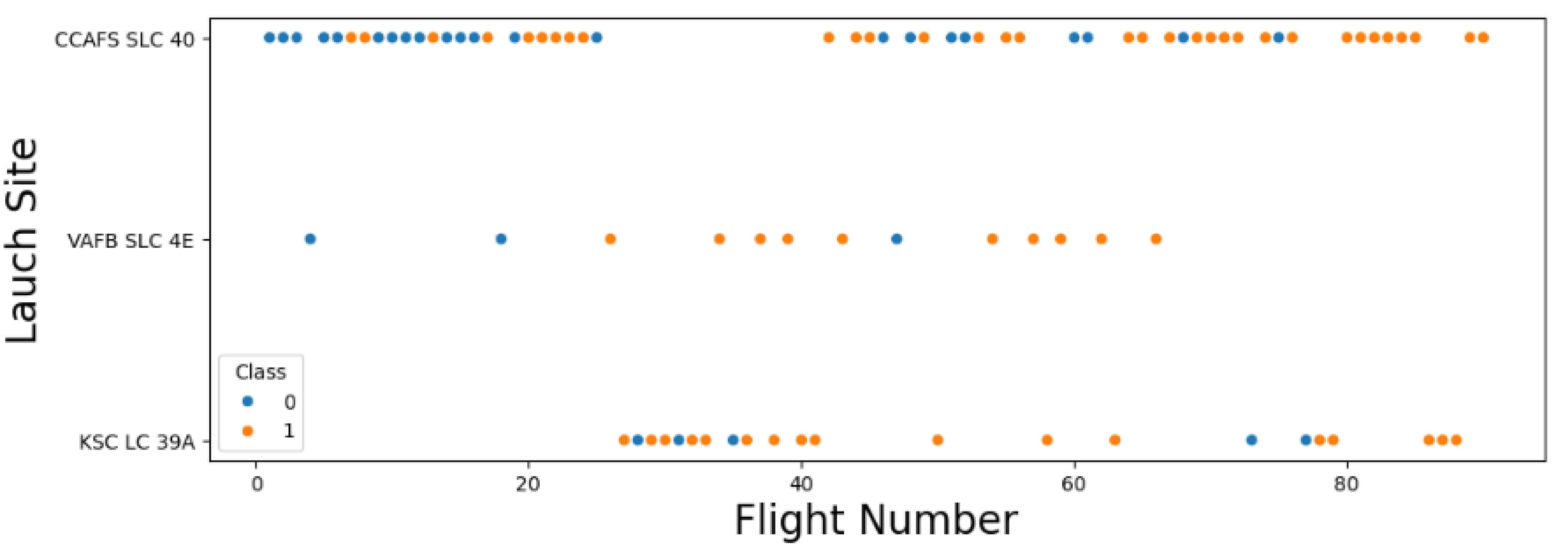
Results

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

Section 2

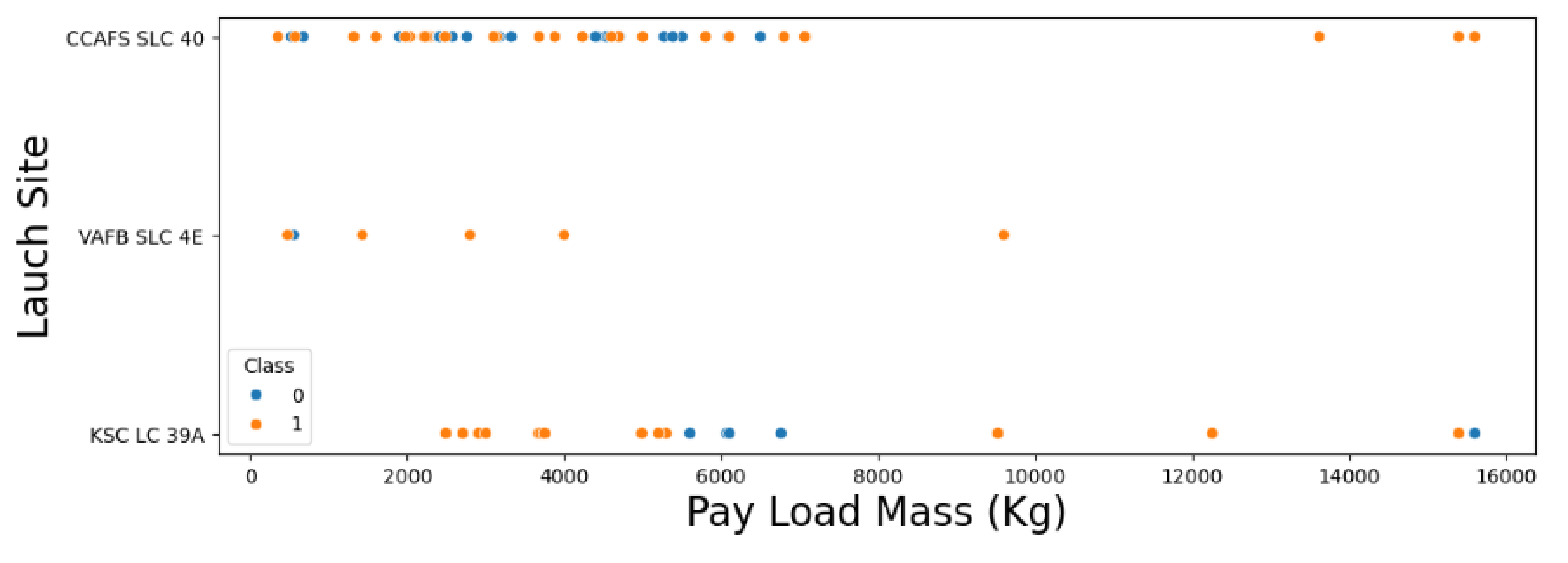
Insights drawn from EDA

Flight Number vs. Launch Site



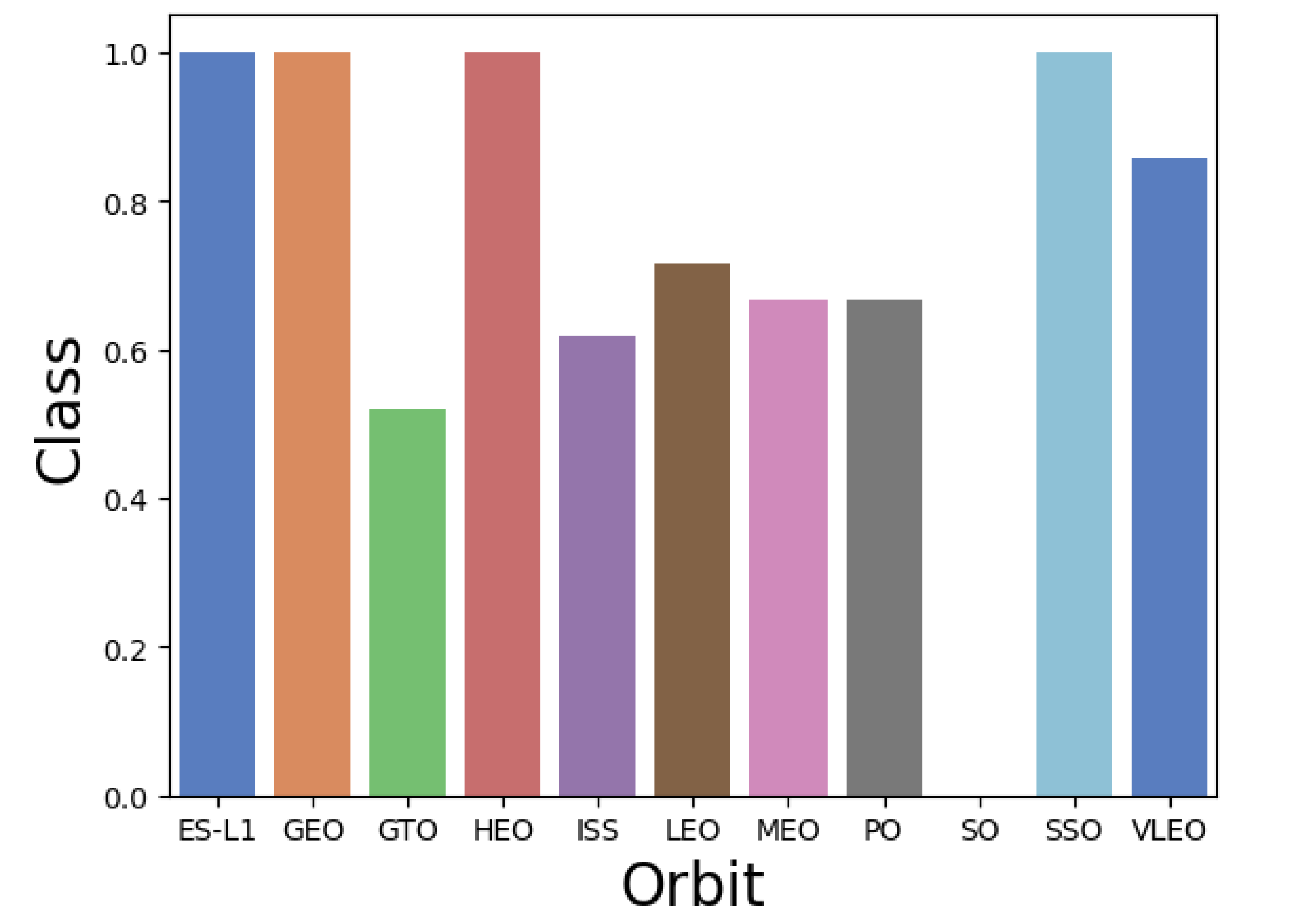
- As per the plot, as number of flights grew success of the launch also grew with it. It is showing positive correlation between the variables.

Payload vs. Launch Site



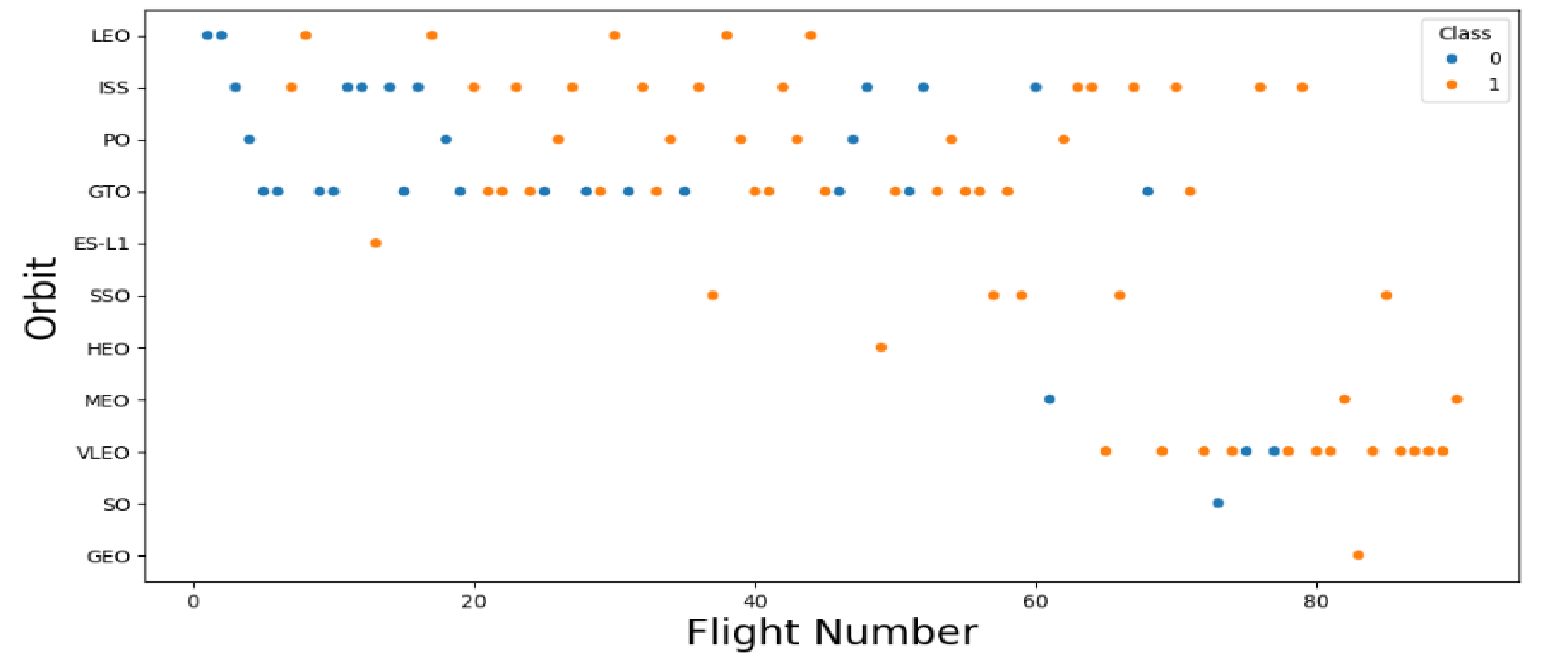
- As the payload mass increases for the site named CCAFS SLC40 the chances of success is also increasing.

Success Rate vs. Orbit Type



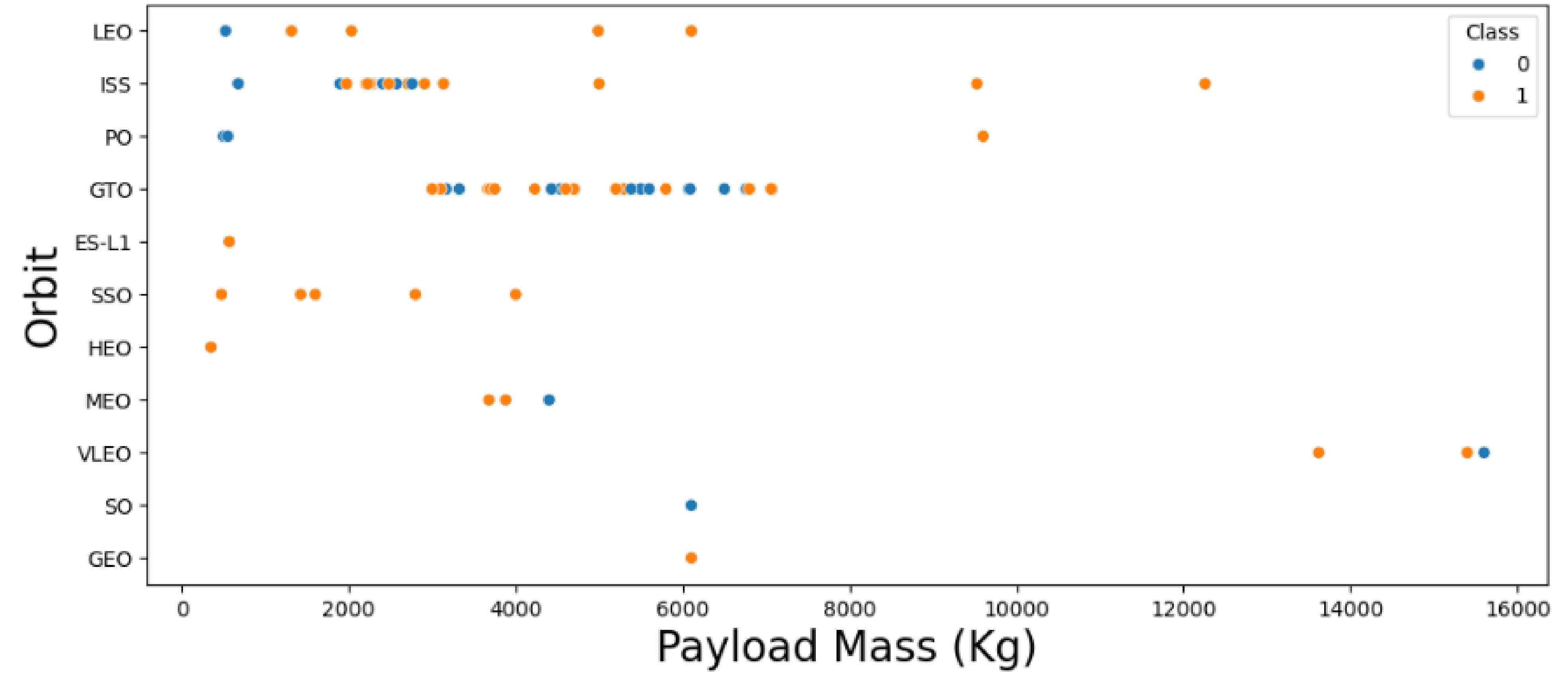
- From the bar plot we can derive that ES-L1, GEO, HEO and SSO orbits are the most successful.

Flight Number vs. Orbit Type



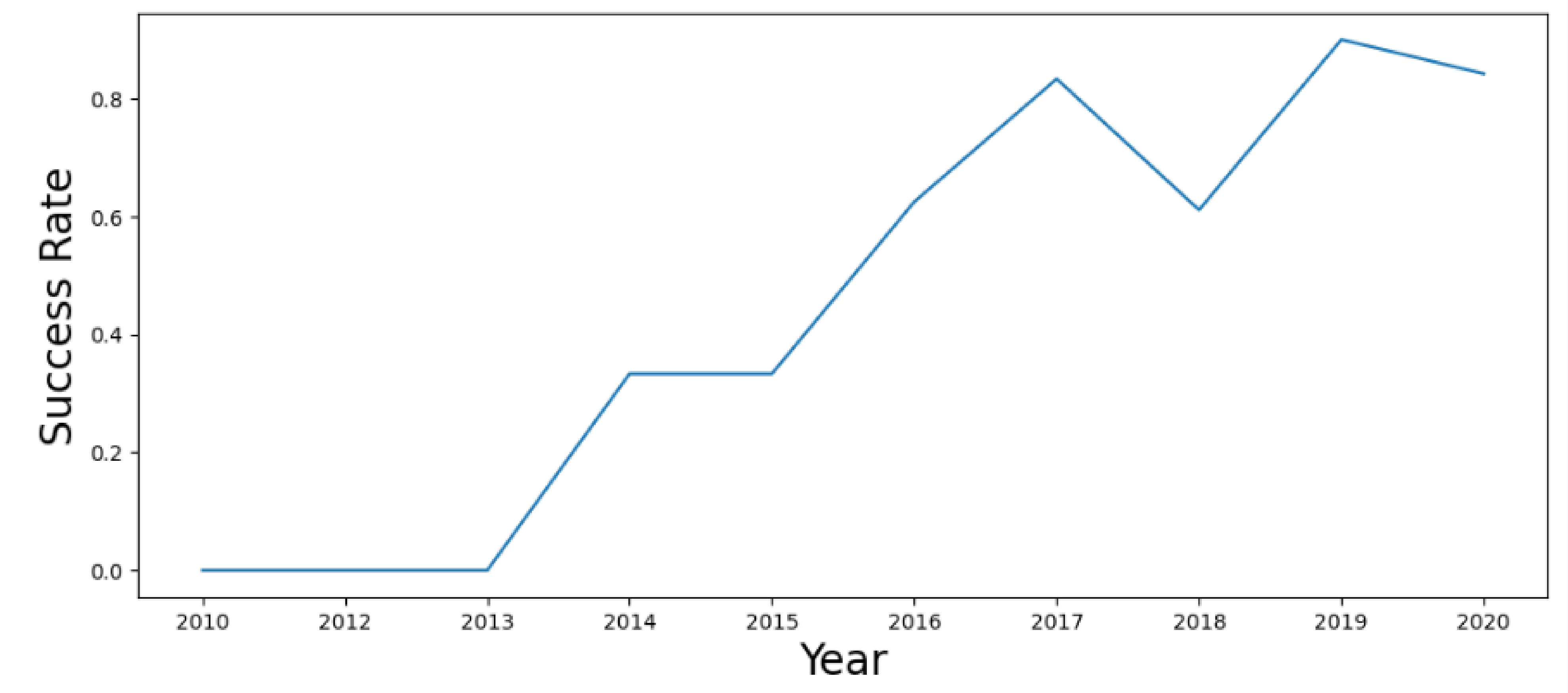
- It can be observed that LEO orbit has positive correlation with flight numbers in getting successful landing.

Payload vs. Orbit Type



- From the scatter plot we can observe that with the heavy payloads, the successful landing are more for PO, LEO, and ISS.

Launch Success Yearly Trend



- From the line plot, we can derive that between the year 2013 and 2017 success rate was steep.

All Launch Site Names

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

In [23]:
```
%sql SELECT distinct Launch_Site FROM SPACEXTABLE;
```

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

Out[23]:

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

- Used distinct function of SQL to query the unique launch sites.

Launch Site Names Begin with 'CCA'

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

In [34]:

```
%sql SELECT * FROM SPACEXTABLE WHERE Launch_Site like 'CCA%' LIMIT 5;
```

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

Out[34]:

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

- To filter data, we used where clause and like 'CCA%' expression to drill down the output.

Total Payload Mass

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

In [44]:
```
%sql select sum(PAYLOAD_MASS__KG_) as total_payload_mass from SPACEXTABLE where Customer LIKE '%NASA (CRS)%';
```

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

Out[44]:

total_payload_mass
48213

- We calculated the total payload mass by NASA (CRS) is 48231.

Average Payload Mass by F9 v1.1

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

```
%sql SELECT avg(PAYLOAD_MASS__KG_) as average_payload_mass FROM SPACEXTABLE WHERE Booster_Version LIKE 'F9 v1.1';
```

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

average_payload_mass
2928.4

- Calculated the average payload mass carried by booster version F9 v1.1 is 2928.4

First Successful Ground Landing Date

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

Hint:Use min function

In [54]:
```
%sql select min(Date) as First_Successful_landing from (select * from SPACEXTABLE WHERE Landing_outcome = 'Success')
```

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

Out[54]:

First_Successful_landing
2018-07-22

- The dates of the first successful landing outcome on ground pad is 2018-07-22

Successful Drone Ship Landing with Payload between 4000 and 6000

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

In [61]:
```
%sql select Booster_Version from SPACEXTABLE where Landing_Outcome like 'success%drone%' and PAYLOAD_MASS__KG_ between 4000
```

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

Out[61]:

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

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

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

In [77]:
```
Success = %sql select count(Mission_Outcome) as success_count from SPACEXTABLE where Mission_Outcome like '%success%'
Unsuccess = %sql select count(Mission_Outcome) as unsuccess_count from SPACEXTABLE where Mission_Outcome like '%failure%'

print(Success)
print(Unsuccess)
```

```
 * sqlite:///my_data1.db
Done.
 * sqlite:///my_data1.db
Done.
+---------------+
| success_count |
+---------------+
|      100      |
+---------------+
+-----------------+
| unsuccess_count |
+-----------------+
|        1        |
+-----------------+
```

- The total 100 successful and 1 failure mission outcomes.

Boosters Carried Maximum Payload

List all the booster_versions that have carried the maximum payload mass. Use a subquery.

In [86]:

```
%sql select Booster_Version FROM SPACEXTABLE WHERE PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTABLE);
```

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

Out[86]:

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

- We sub queried to find the maximum payload by booster version.

2015 Launch Records

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, 6,2) as month to get the months and substr(Date,0,5)='2015' for year.

In [100…

```
%sql SELECT CASE substr(Date, 6, 2) WHEN '01' THEN 'January' WHEN '02' THEN 'February' WHEN '03' THEN 'March' WHEN '04' THEN

# %sql select * from spacextable where landing_outcome like '%drone%ship%'
```

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

Out[100…

Month	Landing_Outcome	Booster_Version	Launch_Site
January	Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
April	Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40
June	Precluded (drone ship)	F9 v1.1 B1018	CCAFS LC-40

- I am unable to fit entire query on this slide because it is too long but from the link you can view. here

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

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

```
%sql SELECT * FROM SPACEXTABLE WHERE (Landing_Outcome like '%Failure%Drone%Ship%' or Landing_Outcome like '%Success%ground%pad%') and (Date between '2010-06-04' and '2017-03-20') order by Landing_Outcome DESC;
```

[106] Python

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

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outcome
2015-12-22	1:29:00	F9 FT B1019	CCAFS LC-40	OG2 Mission 2 11 Orbcomm-OG2 satellites	2034	LEO	Orbcomm	Success	Success (ground pad)
2016-07-18	4:45:00	F9 FT B1025.1	CCAFS LC-40	SpaceX CRS-9	2257	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2015-01-10	9:47:00	F9 v1.1 B1012	CCAFS LC-40	SpaceX CRS-5	2395	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2015-04-14	20:10:00	F9 v1.1 B1015	CCAFS LC-40	SpaceX CRS-6	1898	LEO (ISS)	NASA (CRS)	Success	Failure (drone ship)
2016-01-17	18:42:00	F9 v1.1 B1017	VAFB SLC-4E	Jason-3	553	LEO	NASA (LSP) NOAA CNES	Success	Failure (drone ship)
2016-03-04	23:35:00	F9 FT B1020	CCAFS LC-40	SES-9	5271	GTO	SES	Success	Failure (drone ship)
2016-06-15	14:29:00	F9 FT B1024	CCAFS LC-40	ABS-2A Eutelsat 117 West B	3600	GTO	ABS Eutelsat	Success	Failure (drone ship)

Generate + Code + Markdown

- Ranked 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
- In the query we applied GROUPBY and ORDER BY functions of SQL.

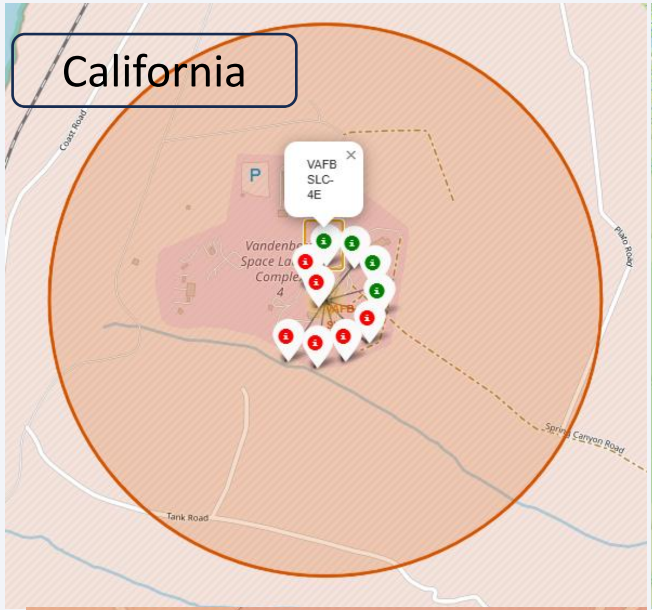
Section 3

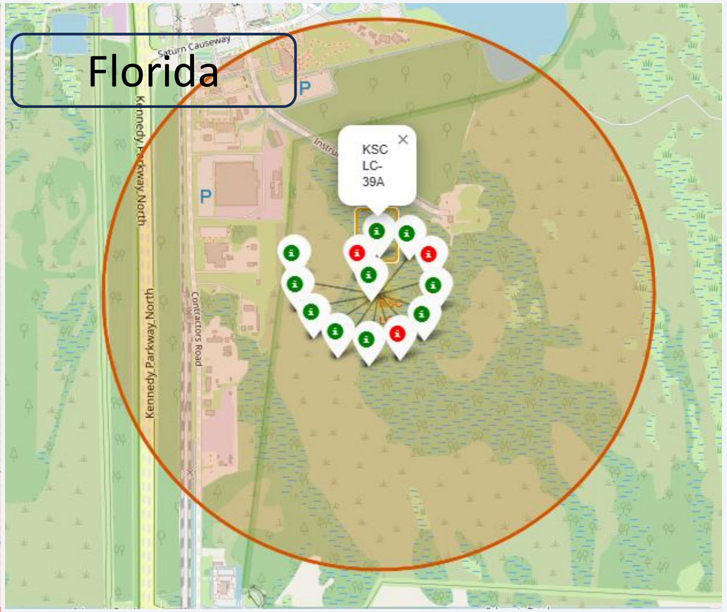
Launch Sites Proximities Analysis

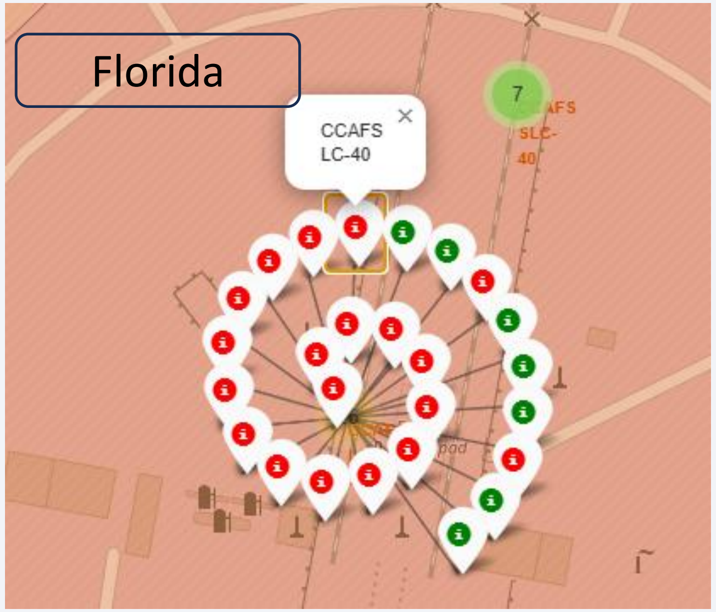
Global map with launch site markers

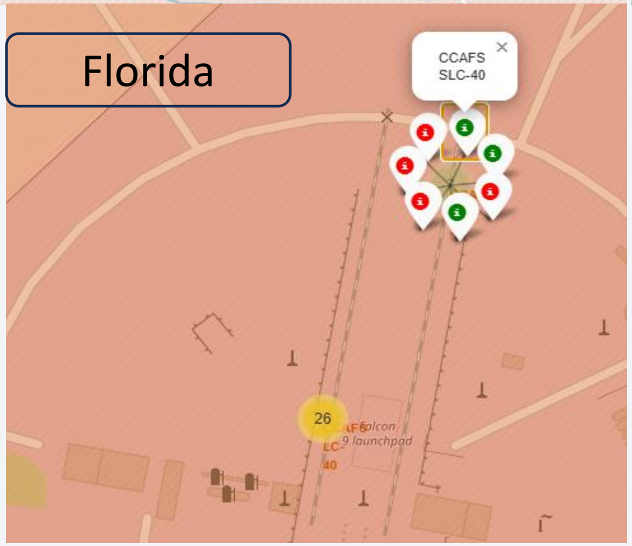
- Launch sites are located in USA precisely on west and east cost.

Markers on launch sites with color label









- Green marker shows successful launches and red one shows unsuccessful launches.

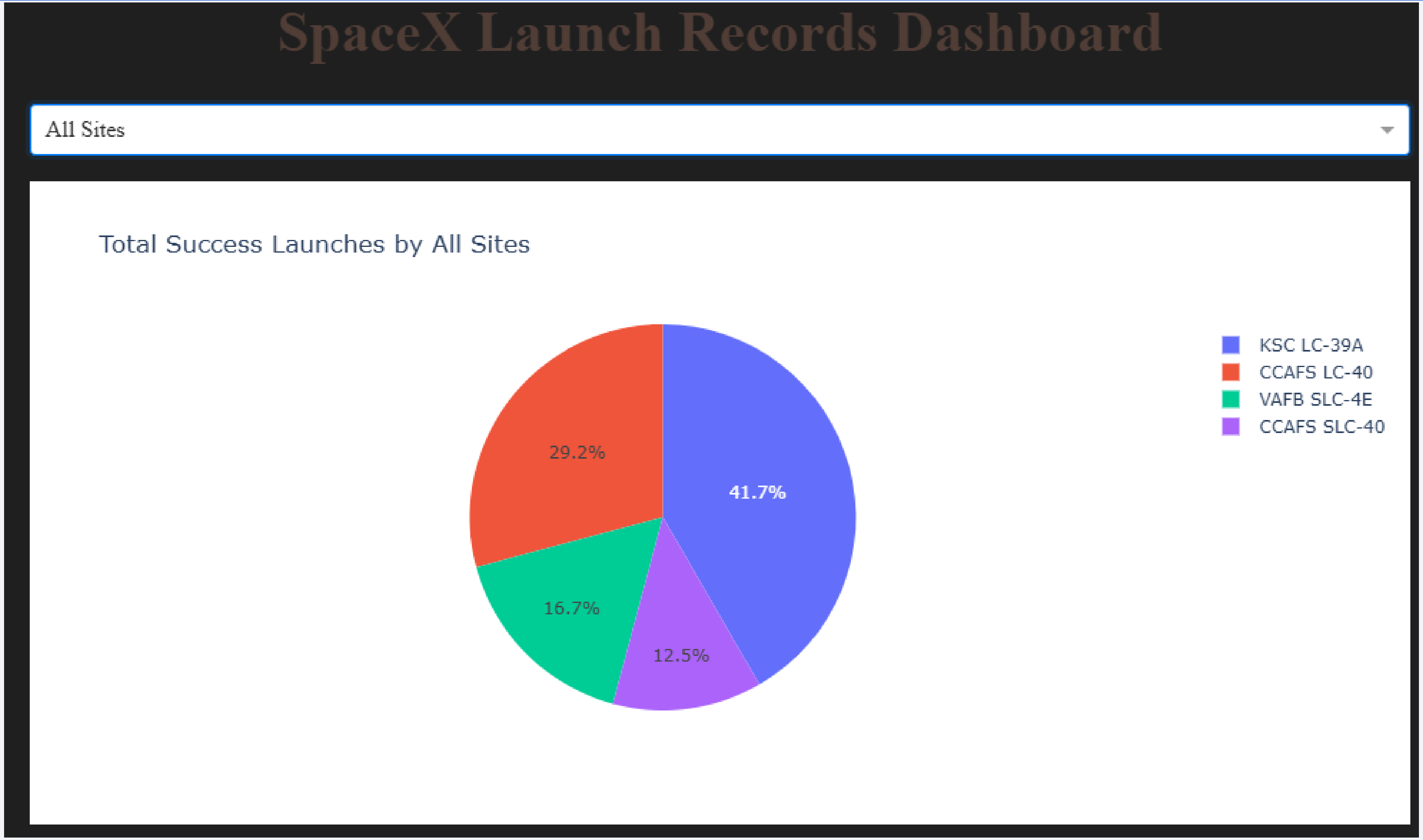
Launch site distance from landmarks

- The distance of ocean from the launch pad is 0.92 Km.

Section 4

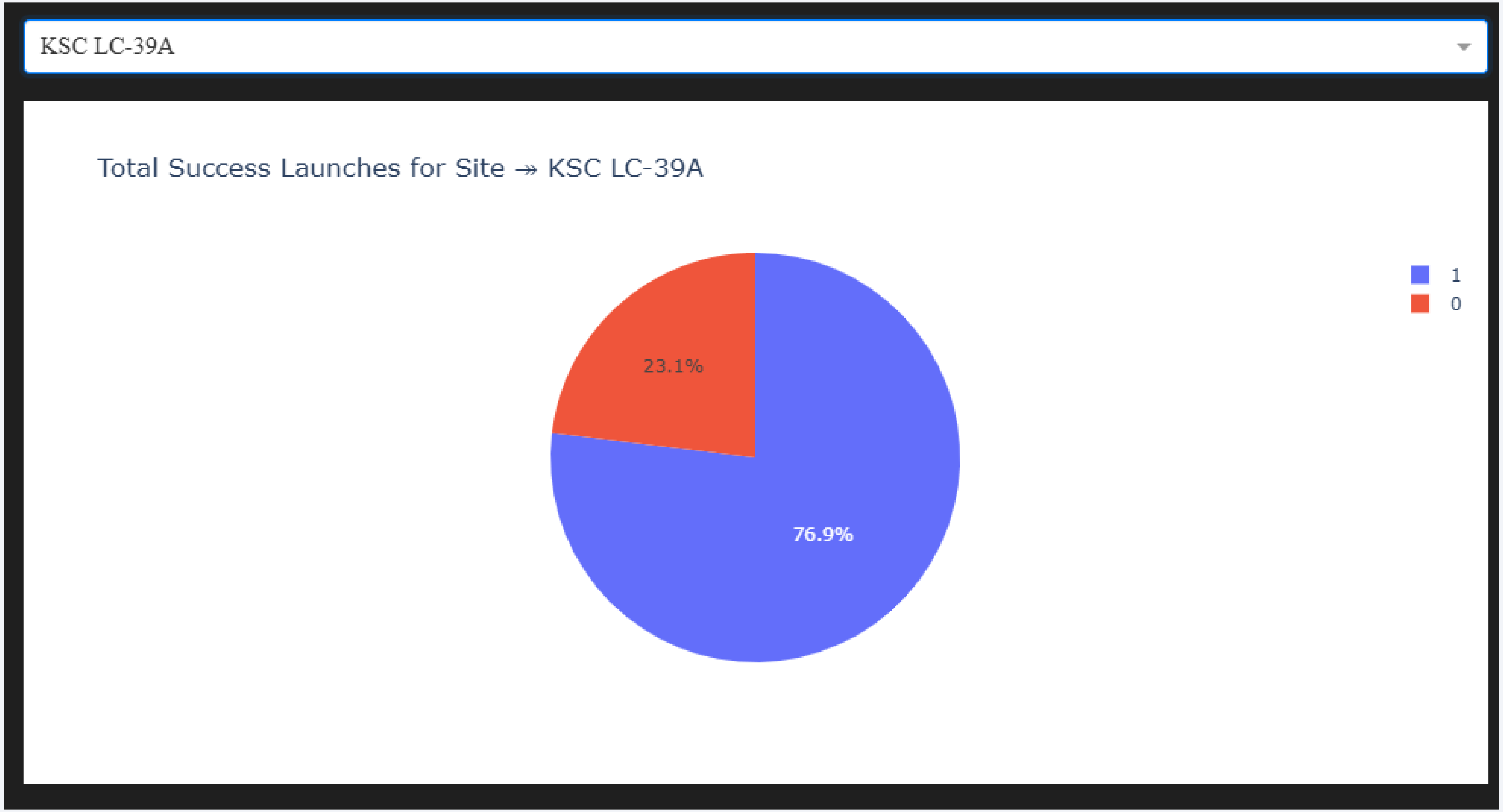
Build a Dashboard with Plotly Dash

Launch success count for all sites



- Highest launch success achieved from site KSC LC-39A

Launch site with highest launch success ratio



- Site KSC LC-39A achieved 76.9% success rate.

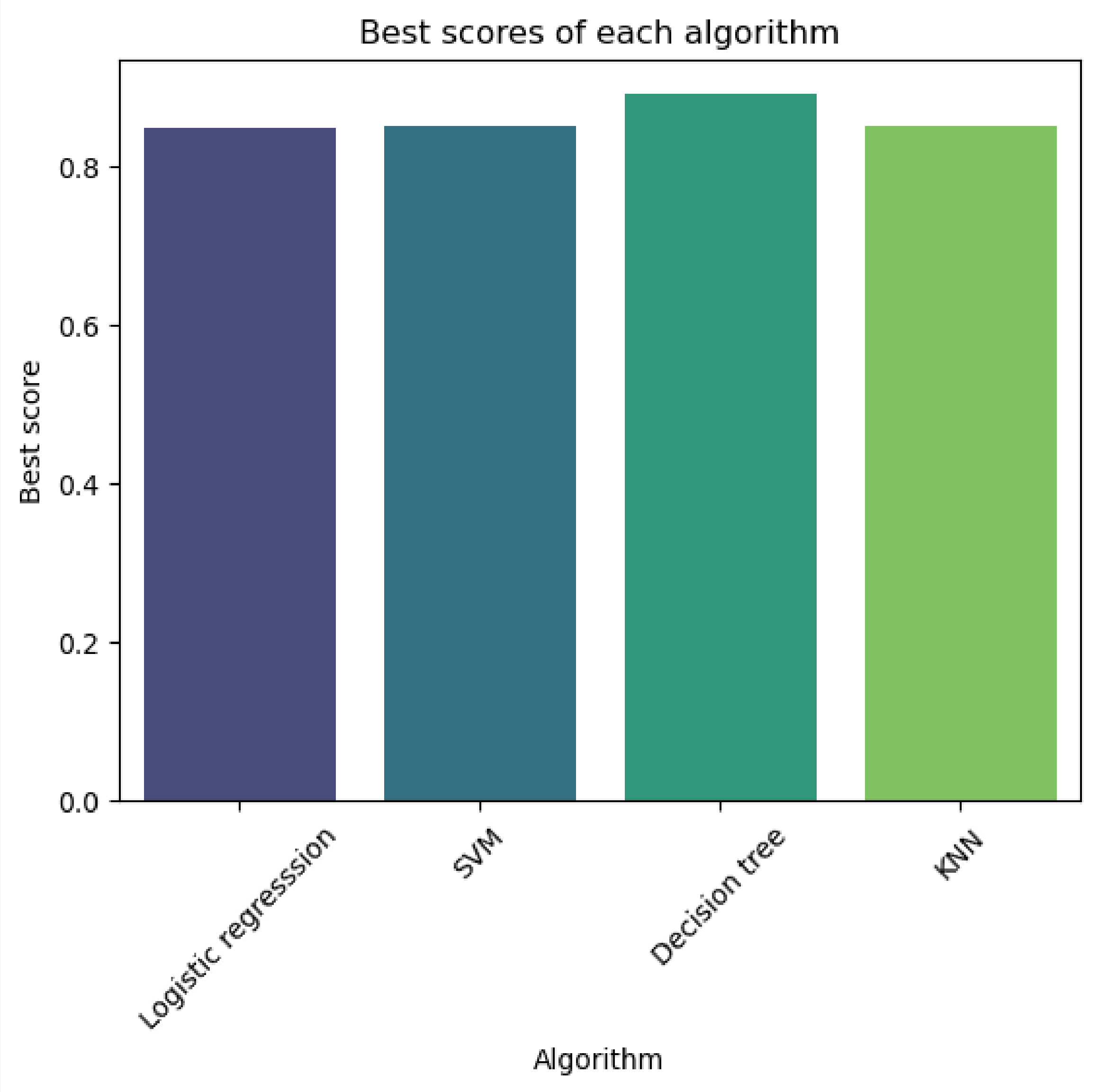
scatter plot for all sites, with different payload selected in the range slider

- It can be depicted from the above charts; low weighed payloads are more prone to success than heavy ones.

Section 5

Predictive Analysis (Classification)

Classification Accuracy



- The best accuracy score has been achieved by Decision tree classification model which is 0.88 in the chart.

Confusion Matrix

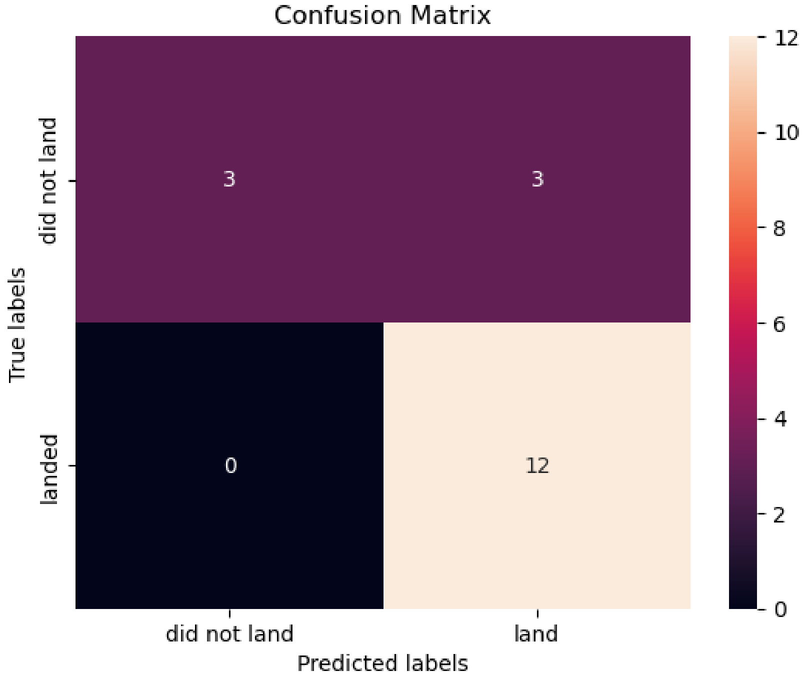
Calculate the accuracy of tree_cv on the test data using the method `score`:

In [59]:
```
tree_accuracy = tree_cv.score(X_test, Y_test)
tree_accuracy
```

Out[59]: 0.8333333333333334

We can plot the confusion matrix

In [60]:
```
yhat = tree_cv.predict(X_test)
plot_confusion_matrix(Y_test,yhat)
```



- Confusion matrix for the decision tree classifier shows that it can able to understand the difference between the classes.
- Though it predicted 3 instances as landed but factually it did not land.

Conclusions

- Decision tree classifier is the most suitable machine learning model for this dataset.
- Most successful orbits are ES-L1, GEO, HEO and SSO.
- There is drastic improvement after 2013 which resulted in steep success rate.
- The most successful launch site was KSC LC-39A.

Thank you!