Winning Space Race with Data Science

Indranil Chakraborty
20th July, 2022

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 SQL
  - Exploratory Data Analysis with Data Visualization
  - Interactive Visual Analytics with Folium and Dashboard
  - Predictive Analysis using Machine learning

- Summary of all results
  - Exploratory Data Analysis result
  - Interactive Analytics with screenshots
  - Predictive Analysis result

Introduction

- Project background and context

  Space X 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 Space X 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 space X for a rocket launch. This goal of the project is to create a machine learning pipeline to predict if the first stage will land successfully.

- Problems you want to find answers

  - What Factors determine if the rocket will launch successfully?
  - The interaction among various features that determine the success rate of a successful landing
  - What operating conditions need to be in place to ensure a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected using SpaceX API and web scraping from Wikipedia
- Perform data wrangling
  - One hot encoding was applied 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
  - How to build, tune, evaluate classification models

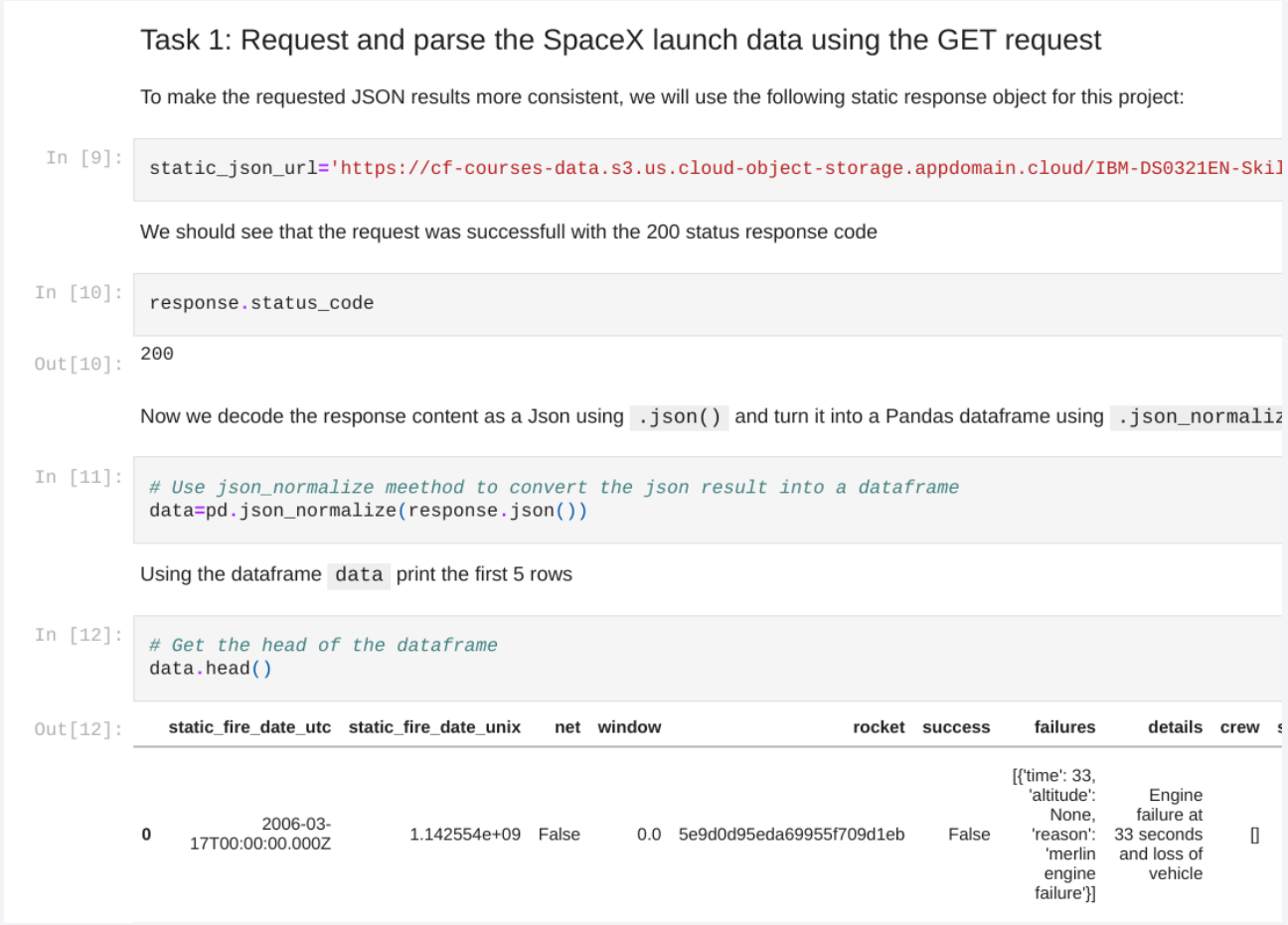
Data Collection

- **The data was collected using various methods**
  - Data collection was done using get request to the SpaceX API.
  - Next, we decoded the response content as a Json using .json() function call and turn it into a pandas dataframe using .json_normalize().
  - We then cleaned the data, checked for missing values and fill in missing values where necessary.
  - In addition, we performed web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup.
  - The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas dataframe for future analysis.

Data Collection – SpaceX API

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.
- The GitHub URL of the completed SpaceX API calls notebook is:

https://github.com/indranilch2014/Capstone-project-SpaceX/blob/main/jupyter-labs-spacex-data-collection-api11.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 [9]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-Skil
```

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

In [10]:
```
response.status_code
```

Out[10]:
```
200
```

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

In [11]:
```
# 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 [12]:
```
# Get the head of the dataframe
data.head()
```

Out[12]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	s
0	2006-03-17T00:00:00.000Z	1.142554e+09	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 33, 'altitude': None, 'reason': 'merlin engine failure'}]	Engine failure at 33 seconds and loss of vehicle	[]	

Data Collection - Scraping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The GitHub URL of the completed web scraping notebook:

https://github.com/indranilch2014/Capstone-project-SpaceX/blob/main/jupyter-labs-webscraping11.ipynb

In [47]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

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 [48]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
page=requests.get(static_url)
```

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

In [49]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup=BeautifulSoup(page.text, 'html.parser')
```

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

In [50]:
```
# Use soup.title attribute
soup.title
```

Out[50]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

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

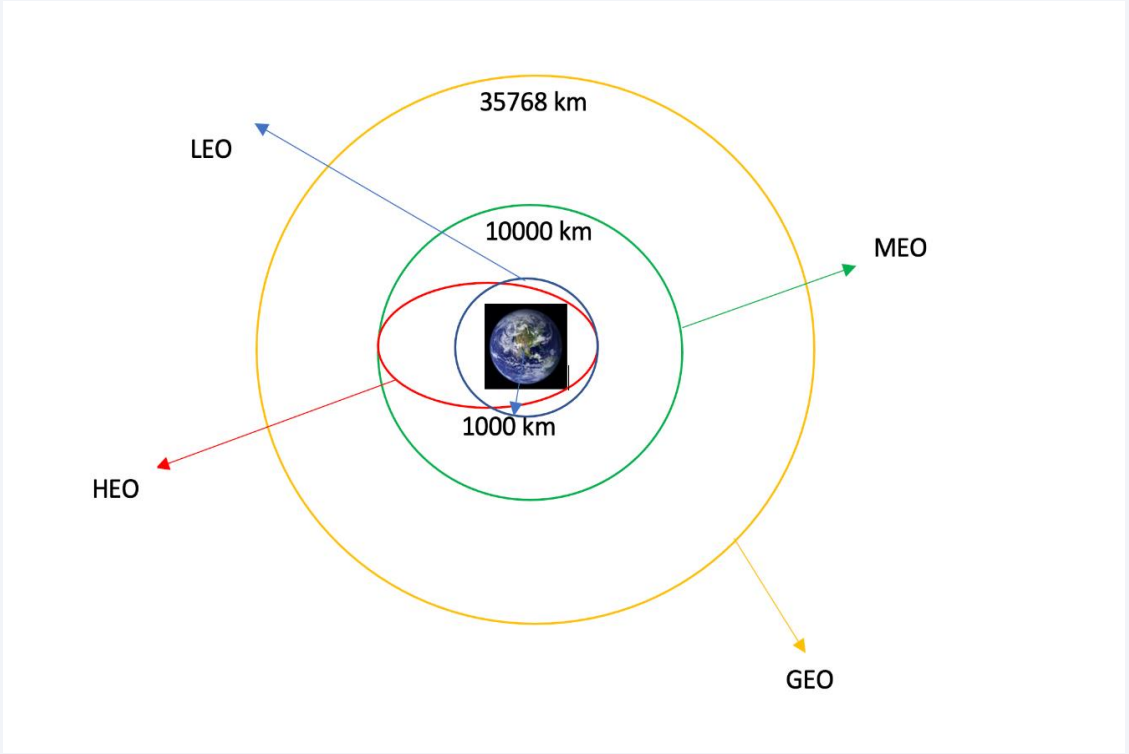
Next, we want to collect all relevant column names from the HTML table header

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 this lab

In [51]:
```
# 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

- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.

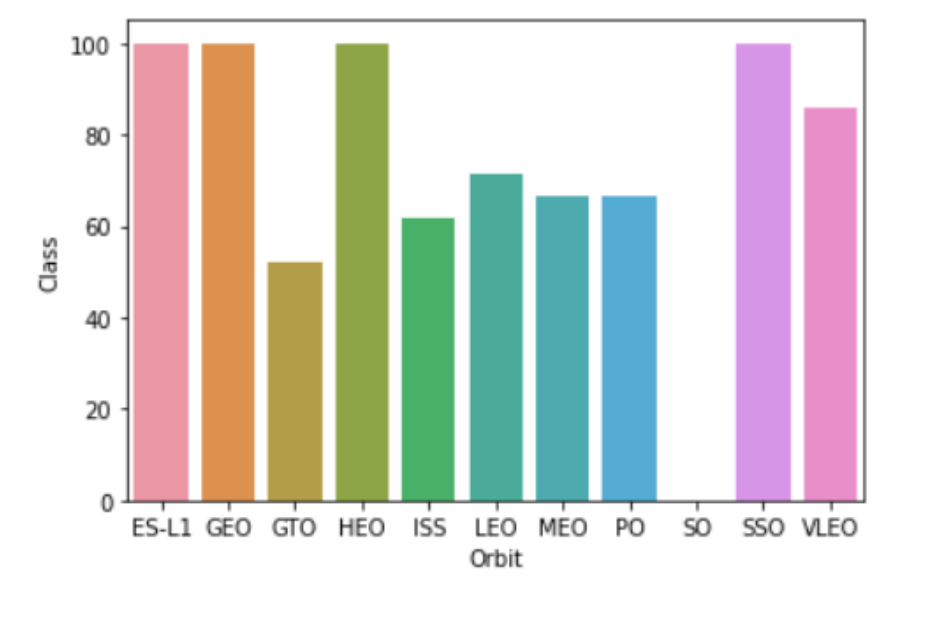


- The GitHub URL of my completed data wrangling related notebook:

https://github.com/indranilch2014/Capstone-project-SpaceX/blob/main/labs-jupyter-spacex-Data%20wrangling11.ipynb

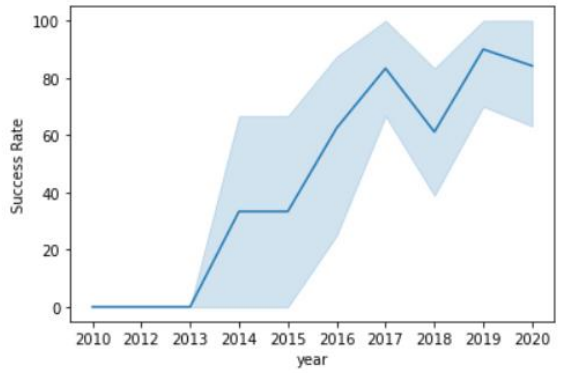
EDA with Data Visualization

- We explored the data by visualizing the relationship between flight number and launch Site, payload and launch site, success rate of each orbit type, flight number and orbit type, the launch success yearly trend.
- The plot below shows the success rate of each Orbit type.




```
In [13]:
# Plot a line chart with x axis to be the extracted year and y axis to be the success rate
year = []
df["year"] = Extract_year(year)
df["Success Rate"] = df["Class"] * 100
sns.lineplot(data = df, x = "year", y = "Success Rate")
```

```
Out[13]: <AxesSubplot:xlabel='year', ylabel='Success Rate'>
```



you can observe that the sucess rate since 2013 kept increasing till 2020

The GitHub URL of my completed EDA with data visualization notebook:

https://github.com/indranilch2014/Capstone-project-SpaceX/blob/main/jupyter-labs-eda-dataviz11.ipynb

EDA with SQL

- We loaded the SpaceX dataset into an SQL database without leaving the jupyter notebook.
- We applied EDA with SQL to get insight from the data. We wrote queries to find out for instance:
  - The names of unique launch sites in the space mission.
  - The total payload mass carried by boosters launched by NASA (CRS)
  - The average payload mass carried by booster version F9 v1.1
  - The total number of successful and failure mission outcomes
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The GitHub URL of my completed EDA with SQL notebook:

https://github.com/indranilch2014/Capstone-project-SpaceX/blob/main/jupyter-labs-eda-sql-coursera_sqllite11.ipynb

Build an Interactive Map with Folium

- We marked all launch sites, and added map objects such as markers, circles, lines to mark the success or failure of launches for each site on the folium map.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success.
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distances between a launch site to its proximities. We answered some question for instance:
  - Are launch sites near railways, highways and coastlines?
  - Do launch sites keep certain distance away from cities?
- The GitHub URL of my completed interactive map with Folium map:

https://github.com/indranilch2014/Capstone-project-SpaceX/blob/main/lab_jupyter_launch_site_location11.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by a certain sites
- We plotted scatter graph showing the relationship with Outcome and Payload Mass (Kg) for the different booster version.

- The GitHub URL of my completed Plotly Dash lab:

https://github.com/indranilch2014/Capstone-project-SpaceX/blob/main/plotly_dash.py

Predictive Analysis (Classification)

- We loaded the data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine learning models and tune different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, improved the model using feature engineering and algorithm tuning.
- We found the best performing classification model and found the best hyperparameters.

- The GitHub URL of my completed predictive analysis lab:

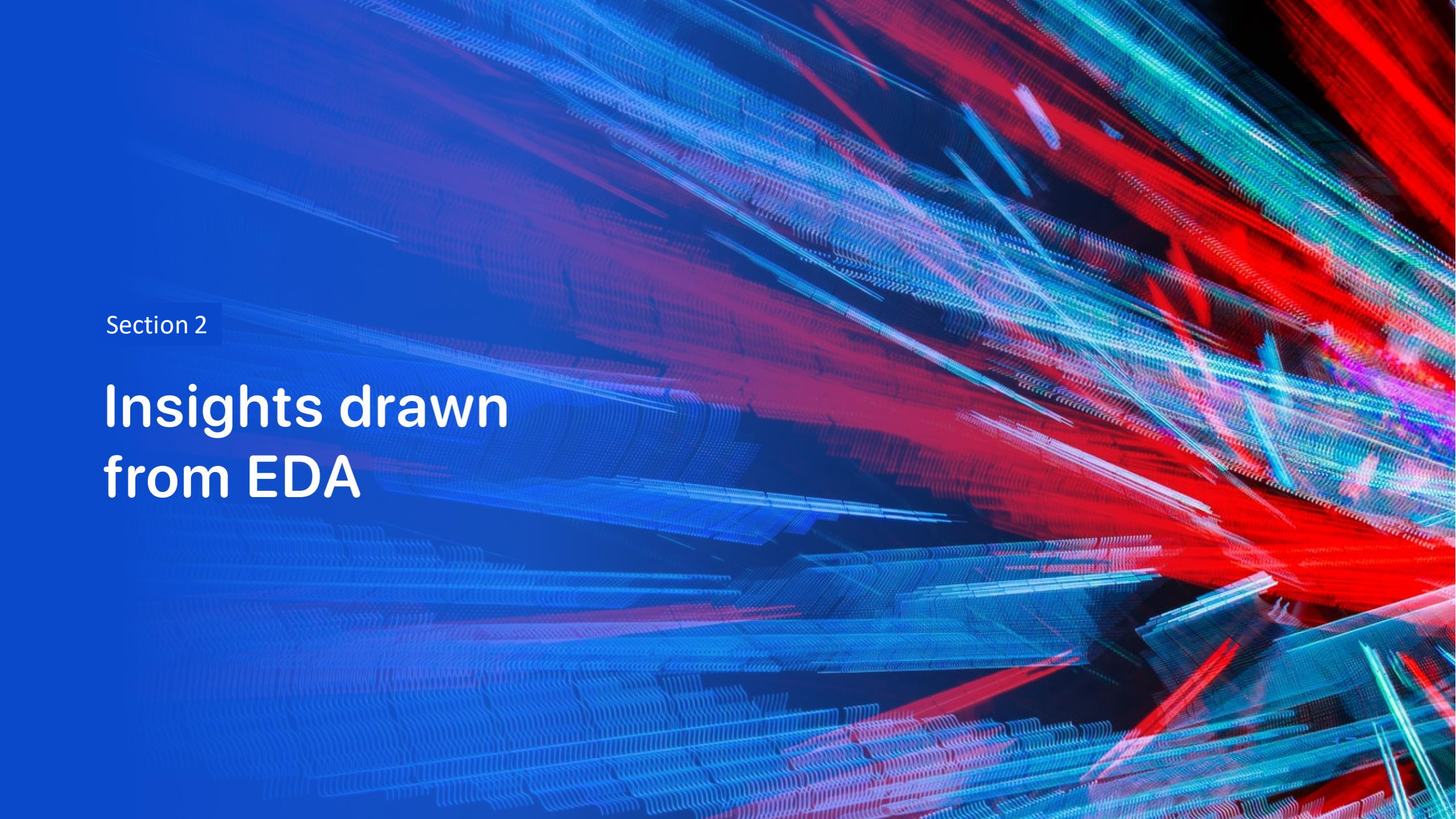
https://github.com/indranilch2014/Capstone-project-SpaceX/blob/main/Machine-learning-prediction.ipynb

Results

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

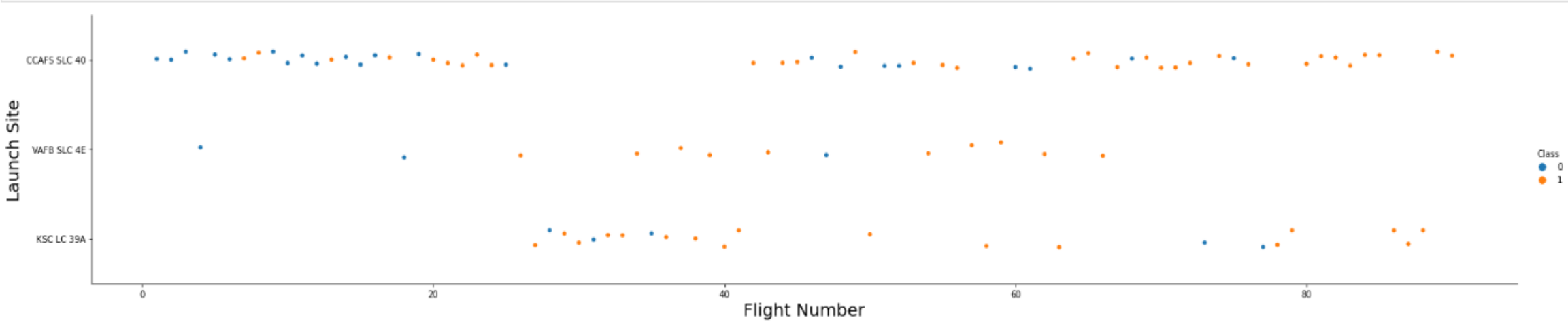
Section 2

Insights drawn from EDA



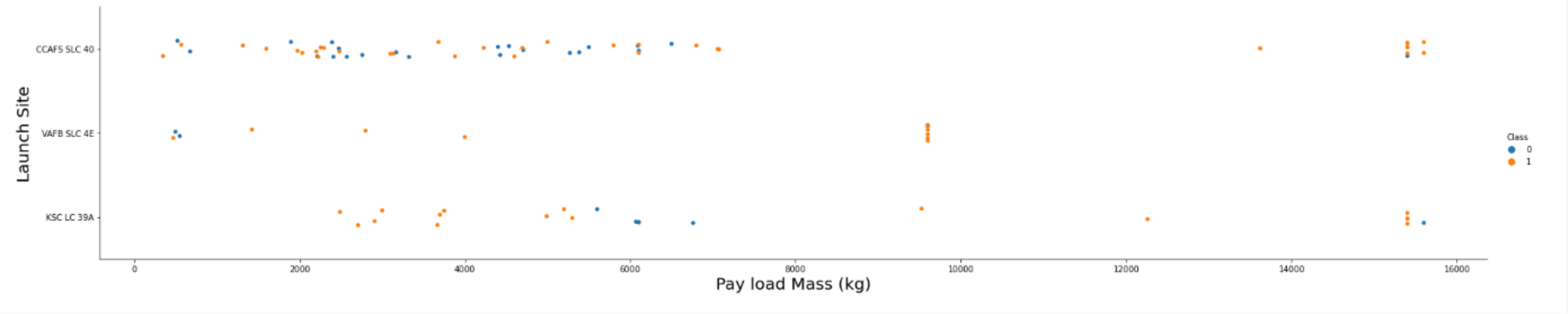
Flight Number vs. Launch Site

- From the plot, we found that the larger the flight number at a launch site, the greater the success rate at a launch site.



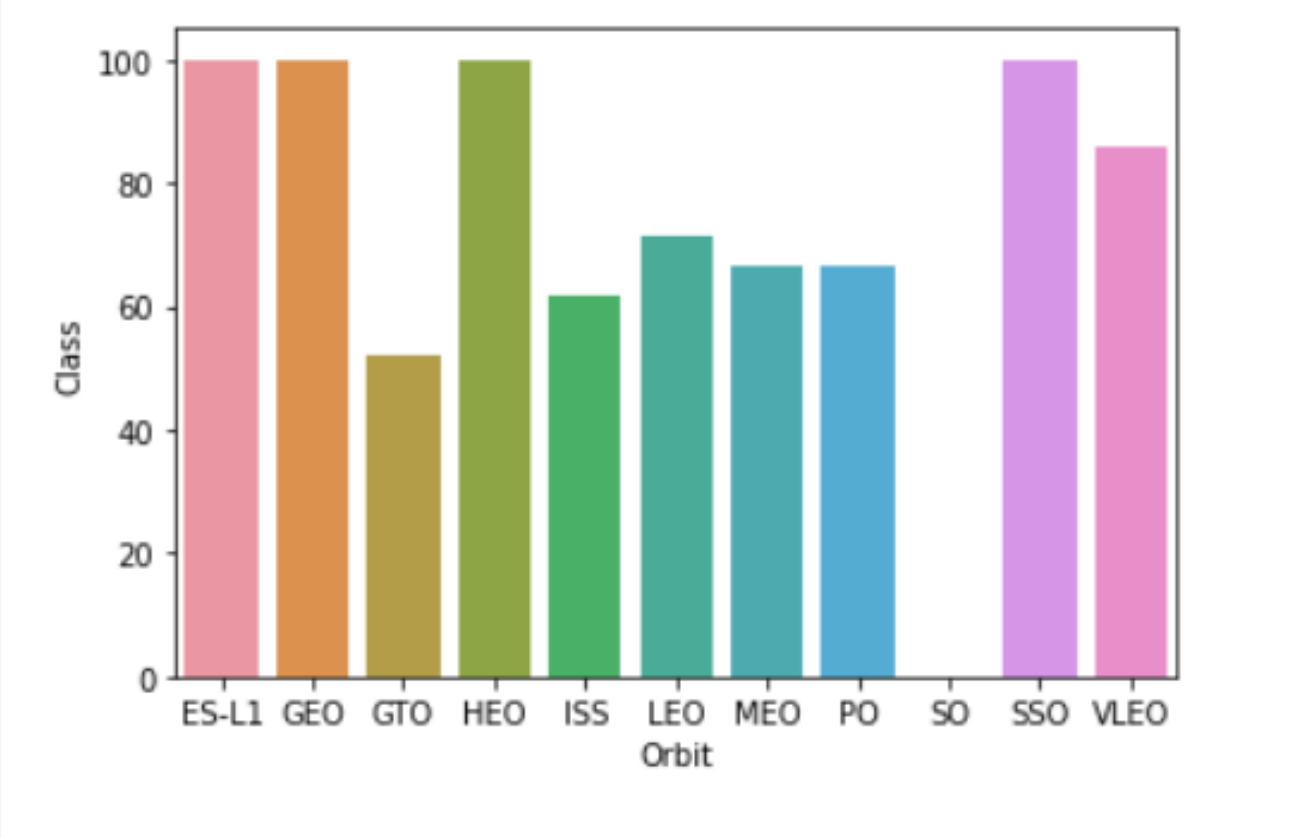
Payload vs. Launch Site

- From the plot, we found that if the payload mass is lesser than 5000kg or greater than 9000kg at the launch site named KSC LC 39A, there is higher success rate and for the launch site CCAFS SLC 40, there is higher success rate for payload mass above 13000kg



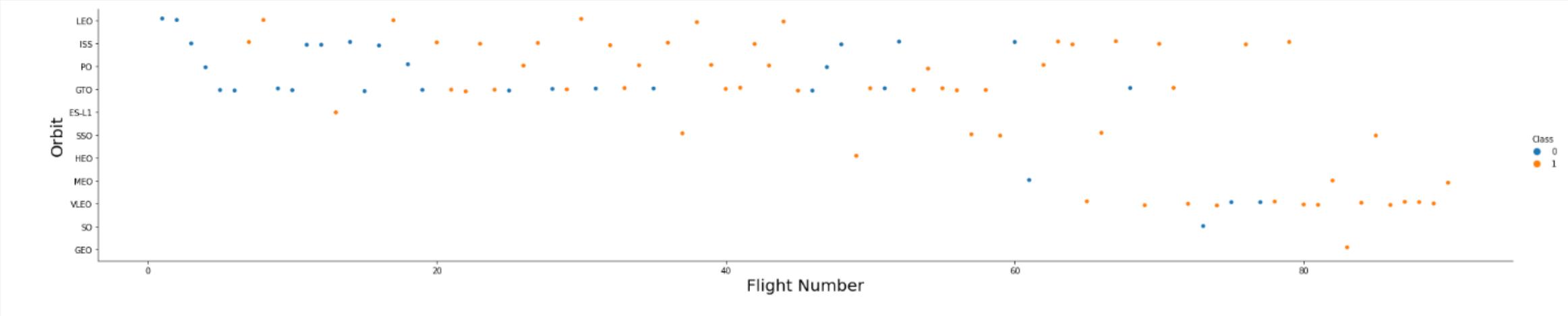
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO orbit types had the most success rate.
- The VLEO orbit type had the second highest success rate.
- The GTO orbit type had the lowest success rate.



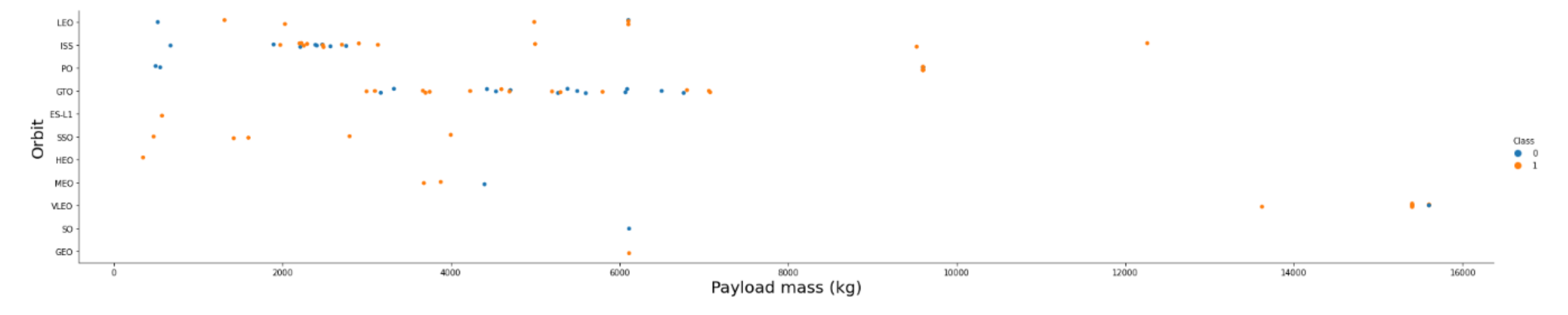
Flight Number vs. Orbit Type

- The plot below shows the Flight Number vs. Orbit type. We observe that in the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



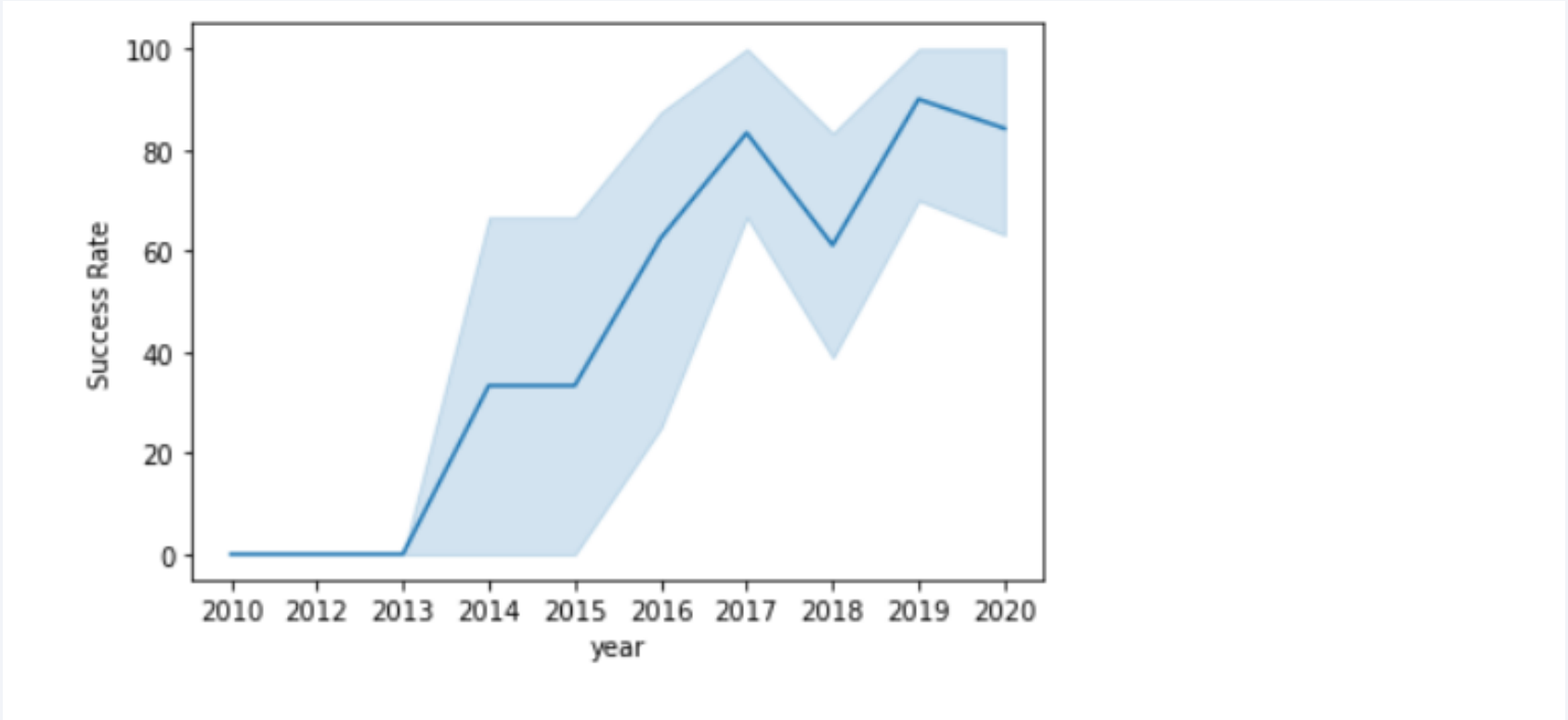
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar, LEO and ISS orbit types.
- However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



Launch Success Yearly Trend

- From the plot, we can observe that success rate since 2013 kept on increasing till 2020.



All Launch Site Names

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

Task 1

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

In [13]:
```
%sql select DISTINCT LAUNCH_SITE from SPACEXDATASET
```

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

Out[13]:

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

Launch Site Names Begin with 'CCA'

Task 2

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

In [14]:
```
%sql select * from SPACEXDATASET where Launch_Site like 'CCA%' limit 5
```

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

Out[14]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	landing_outcome	date_time
04-06-2010	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)	2010-06-04 18:45:00
08-12-2010	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)	2010-12-08 15:43:00
22-05-2012	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt	2012-05-22 07:44:00
08-10-2012	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt	2012-10-08 00:35:00
01-03-2013	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt	2013-03-01 15:10:00

- We used the query above to display 5 records where launch sites begin with `CCA`

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

Task 3

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

In [15]:
```
%sql select sum(payload_mass__kg_) as sum from SPACEXDATASET where customer like 'NASA (CRS)'
```

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

Out[15]:

sum
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2534.67 Kg approx.

Task 4

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

In [16]:
```
%sql select avg(payload_mass__kg_) as Average from SPACEXDATASET where Booster_Version like 'F9 v1.1%'
```

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

Out[16]:

Average
2534.6666666666665

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

Task 5

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

Hint:Use min function

```
In [18]: %sql select Date as FIRST_SUCCESS_GP from SPACEXDATASET where date_time in (select min(date_time) from SPACEXDATASET where landing_outcome
```

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

Out[18]:

FIRST_SUCCESS_GP
22-12-2015

- We used the command on the jupyter notebook:

%sql select Date as FIRST_SUCCESS_GP from SPACEXDATASET where date_time in (select min(date_time) from SPACEXDATASET where landing_outcome like 'Success (ground pad)')

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on drone ship and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000

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 [28]:
```
%sql select booster_version from SPACEXDATASET where payload_mass__kg_ between 4000 and 6000 and landing_outcome like 'Success (drone shi
```

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

Out[28]:

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

Total Number of Successful and Failure Mission Outcomes

- We used count(*) and the GROUP BY clause to find the total number of failure and successful mission outcomes. We can see below that there were 100 successful outcomes and 1 failure outcome.

Task 7

List the total number of successful and failure mission outcomes

In [30]:
```
%sql SELECT mission_outcome, count(*) as Number FROM SPACEXDATASET GROUP by mission_outcome
```

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

Out[30]:

Mission_Outcome	Number
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

- We determined the booster that have carried the maximum payload using a subquery in the **WHERE** clause and the **MAX()** function.

Task 8

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

In [34]:
```
%sql select booster_version from SPACEXDATASET where payload_mass__kg_=(select max(payload_mass__kg_) from SPACEXDATASET)
```

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

Out[34]:

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

- We used a combinations of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ship, their booster versions, and launch site names for year 2015 and also used the SUBSTR() function to obtain the month and year from the date column

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 [61]:
```
%sql select substr(Date,4,2) as Month, landing_outcome, booster_version, launch_site from SPACEXDATASET where substr(date,7,4)='2015' AND
```

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

Out[61]:

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

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

- The following command was used :

%sql select landing_outcome, count(*) as count from SPACEXDATASET where date_time between '2010-06-04' AND '2017-03-20' and landing_outcome like 'Success%' GROUP by landing_outcome ORDER BY count Desc

- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing outcome in descending order.

Task 10

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

In [62]:
```
%sql select landing_outcome, count(*) as count from SPACEXDATASET where date_time between '2010-06-04' AND '2017-03-20' and landing_outcor
```

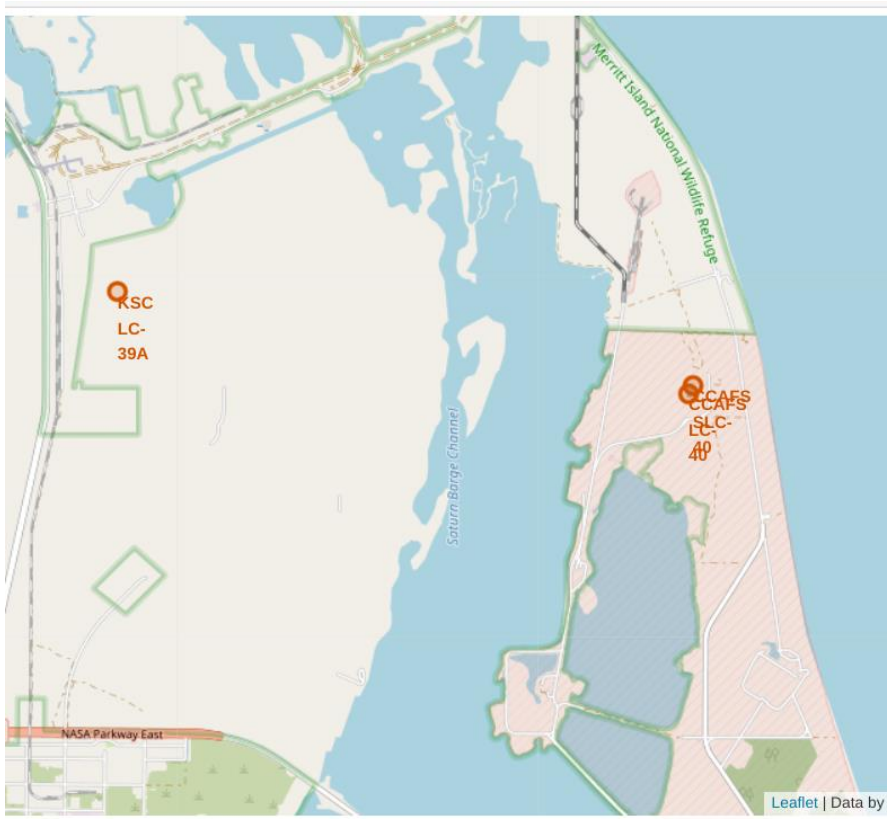
```
 * sqlite:///my_data1.db
Done.
```

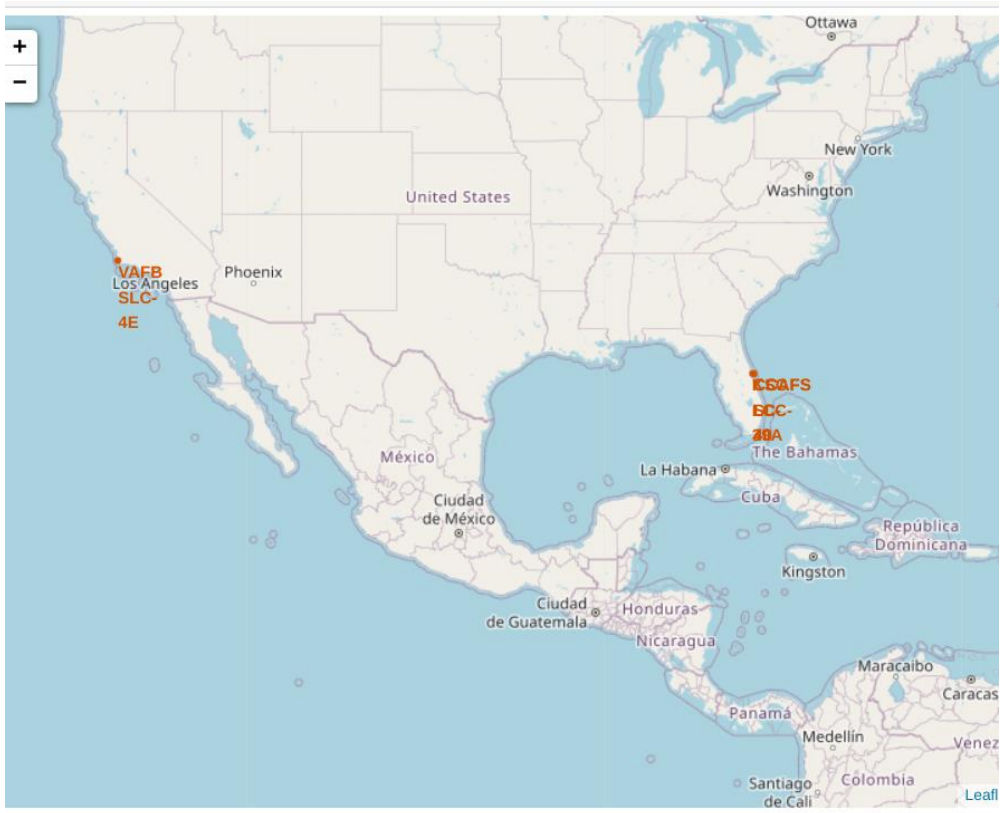
Out[62]:

landing_outcome	count
Success (drone ship)	5
Success (ground pad)	3

Section 3

Launch Sites Proximities Analysis

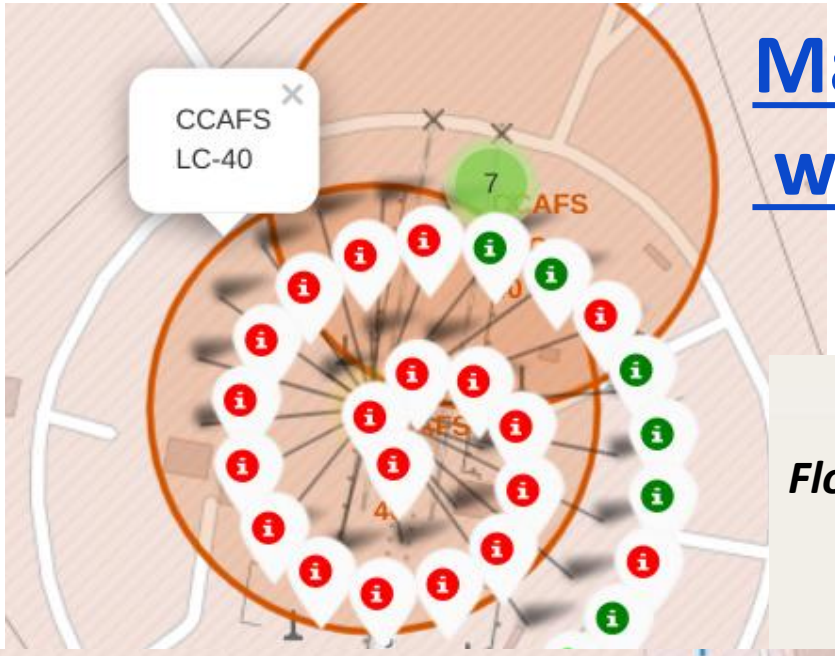


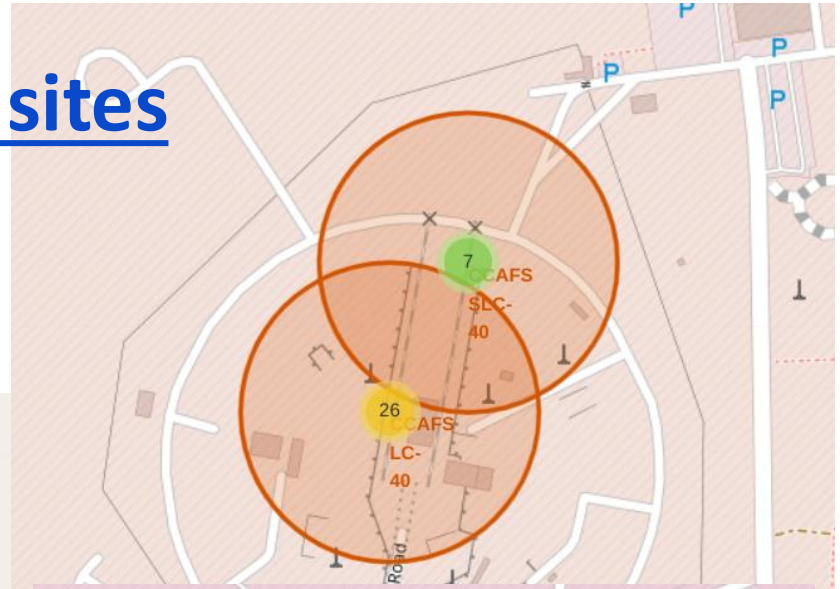


All Launch Sites' location markers

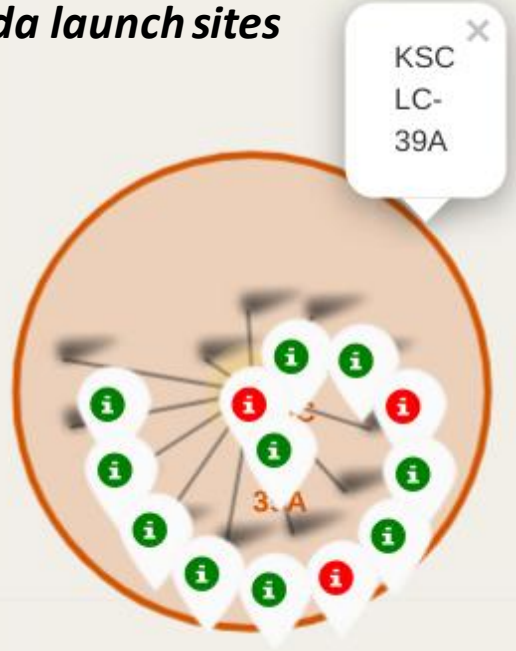
We can see that the Launch sites of SpaceX are located in the USA coastal states of Florida and California

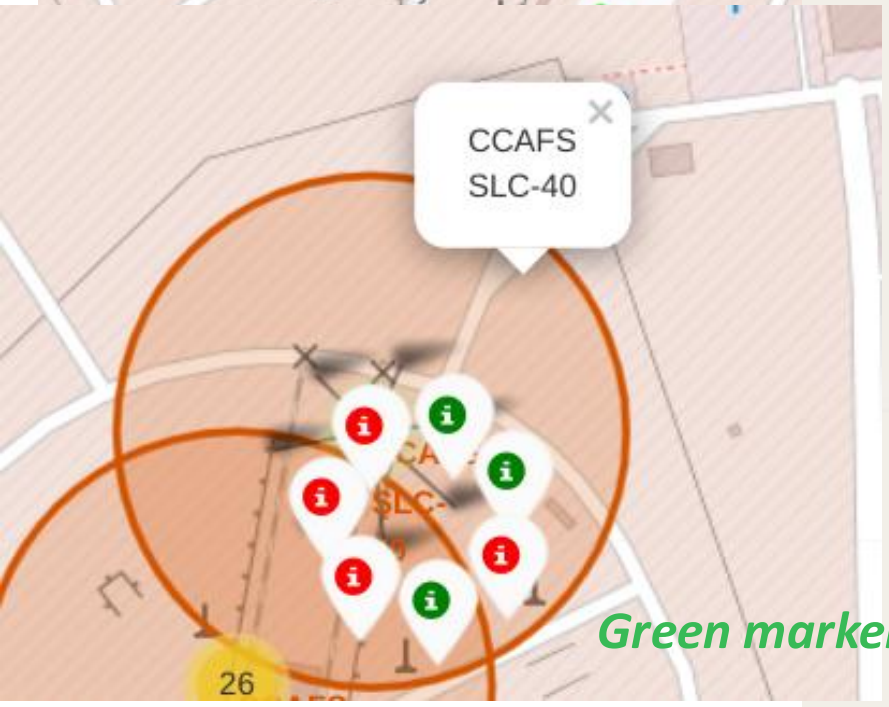
Markers showing launch sites with color labels



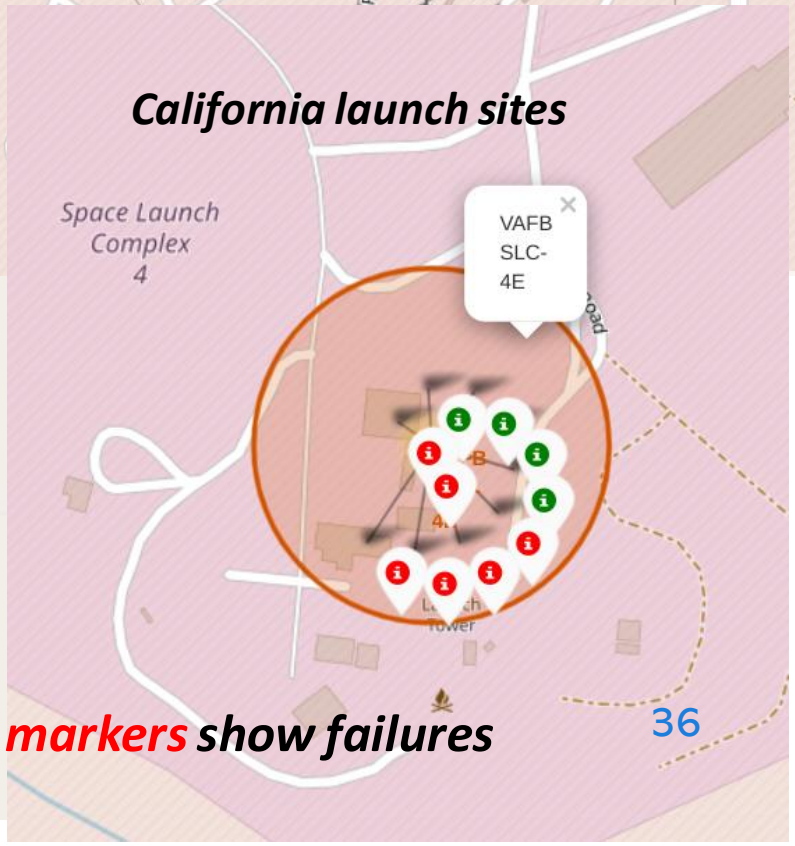


Florida launch sites



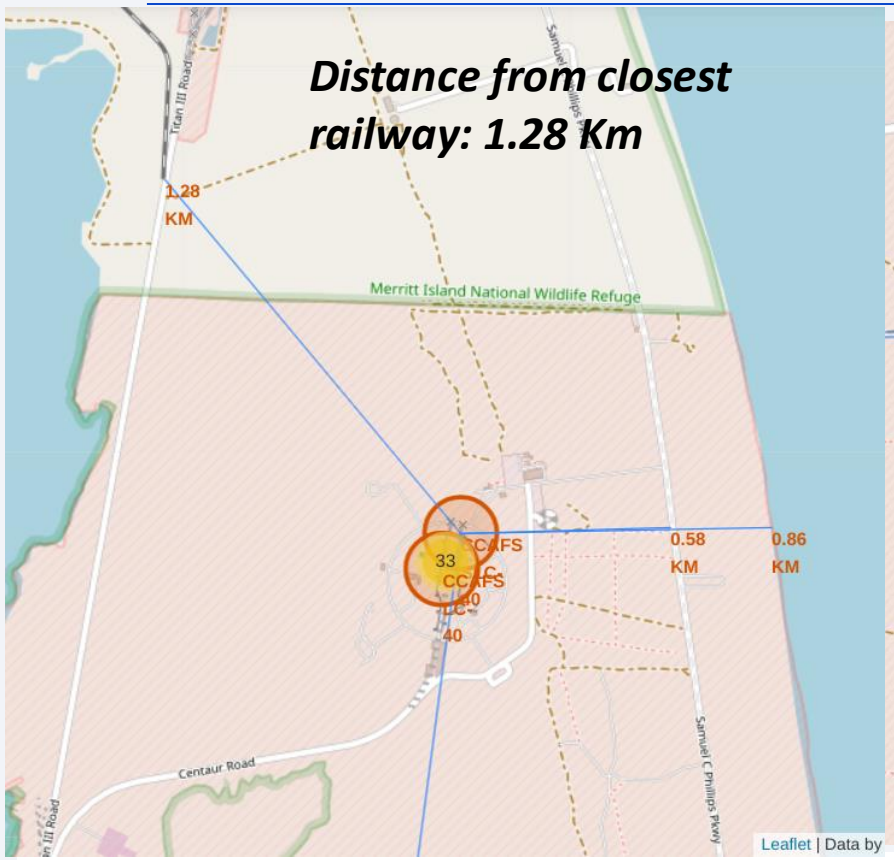


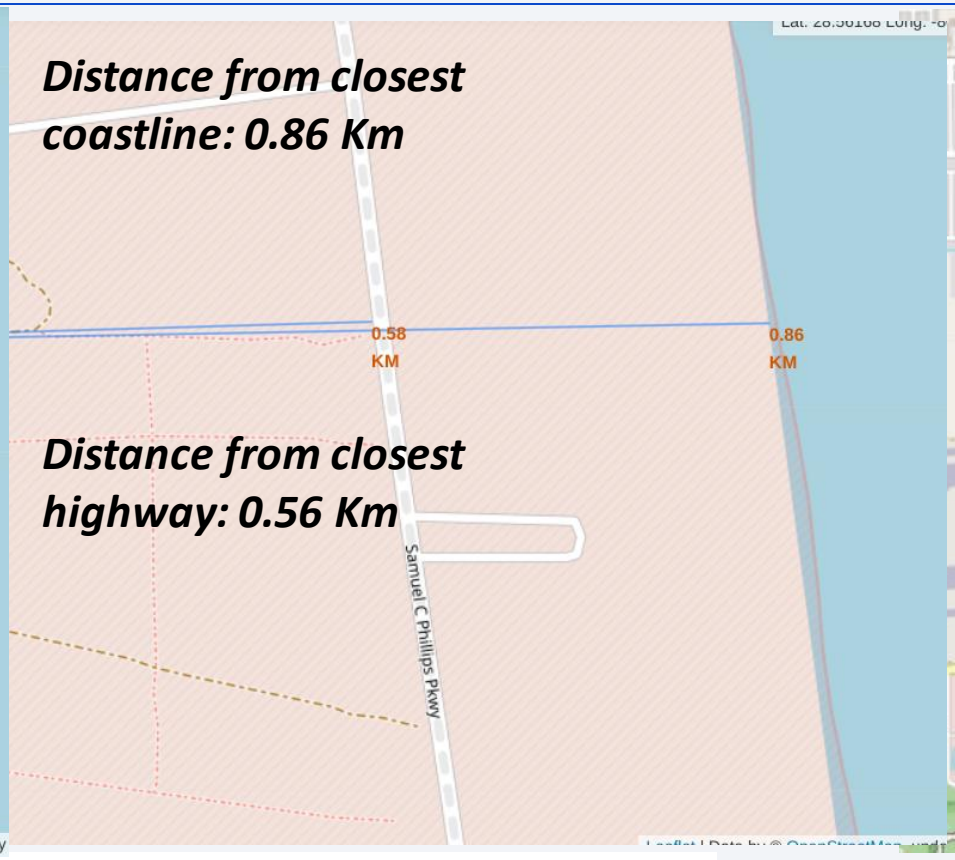
California launch sites



Green markers ***show successful launches and*** ***Red markers*** ***show failures***

Launch site CCAFS SLC-40 and its proximities such as railway, highway, coastline, with distance are calculated and displayed





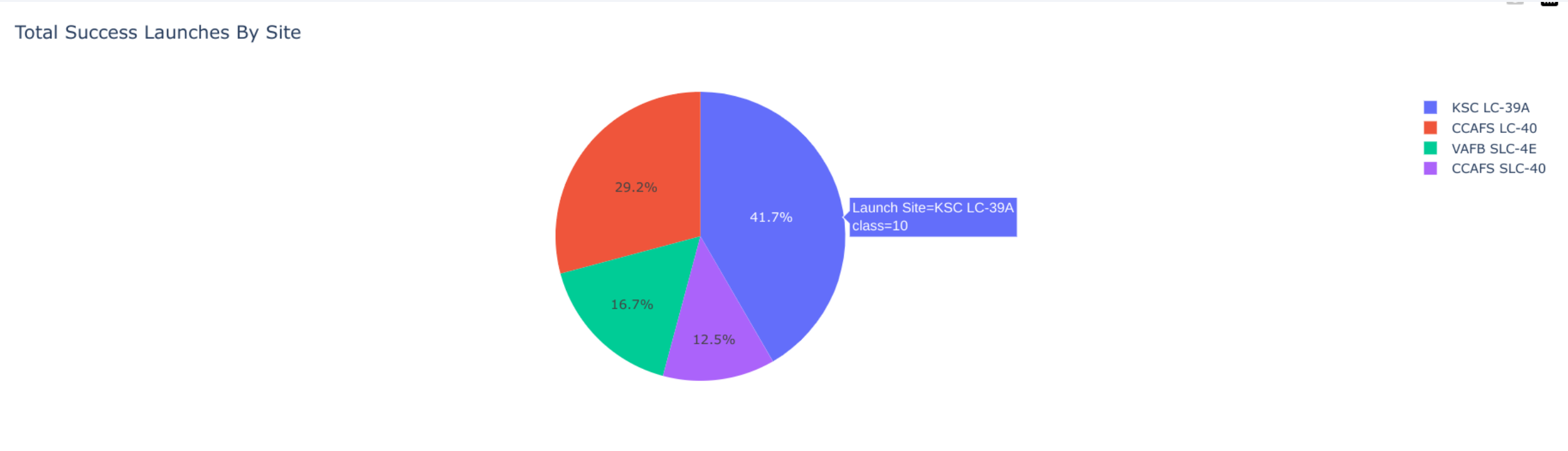
My Findings

- As mentioned before, launch sites are in close proximity to equator to minimize fuel consumption by using Earth's ~ 30km/sec eastward rotational speed to help spaceships get into orbit.
- Launch sites are in close proximity to coastline so they can fly over the ocean during launch.
- Launch sites are in close proximity to highways and railways.
- Launch sites are not in close proximity to cities, which minimizes danger to population dense areas.

Section 4

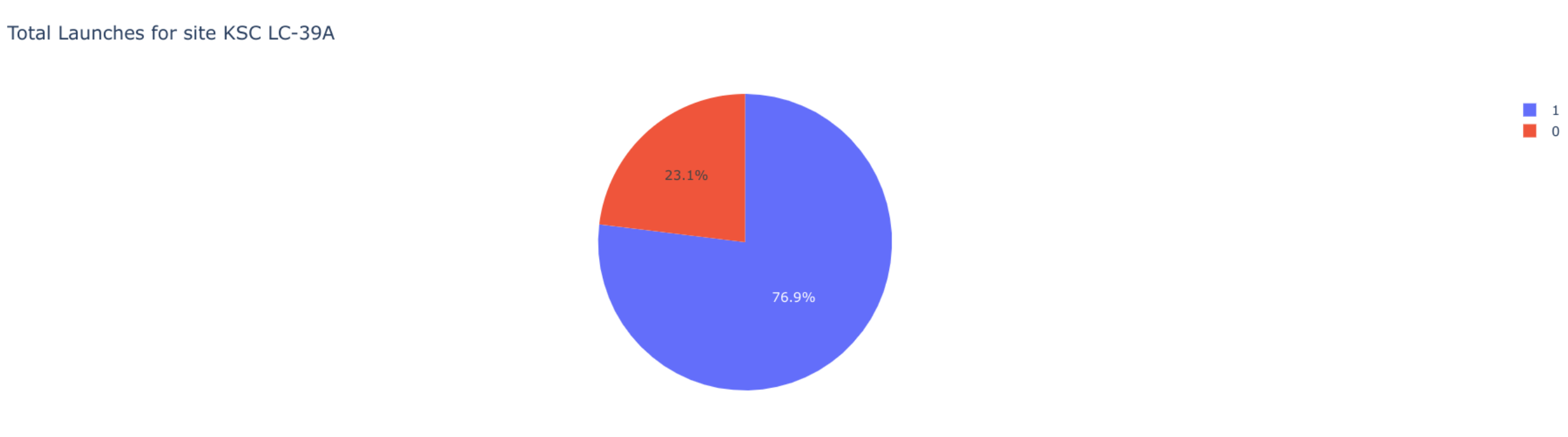
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



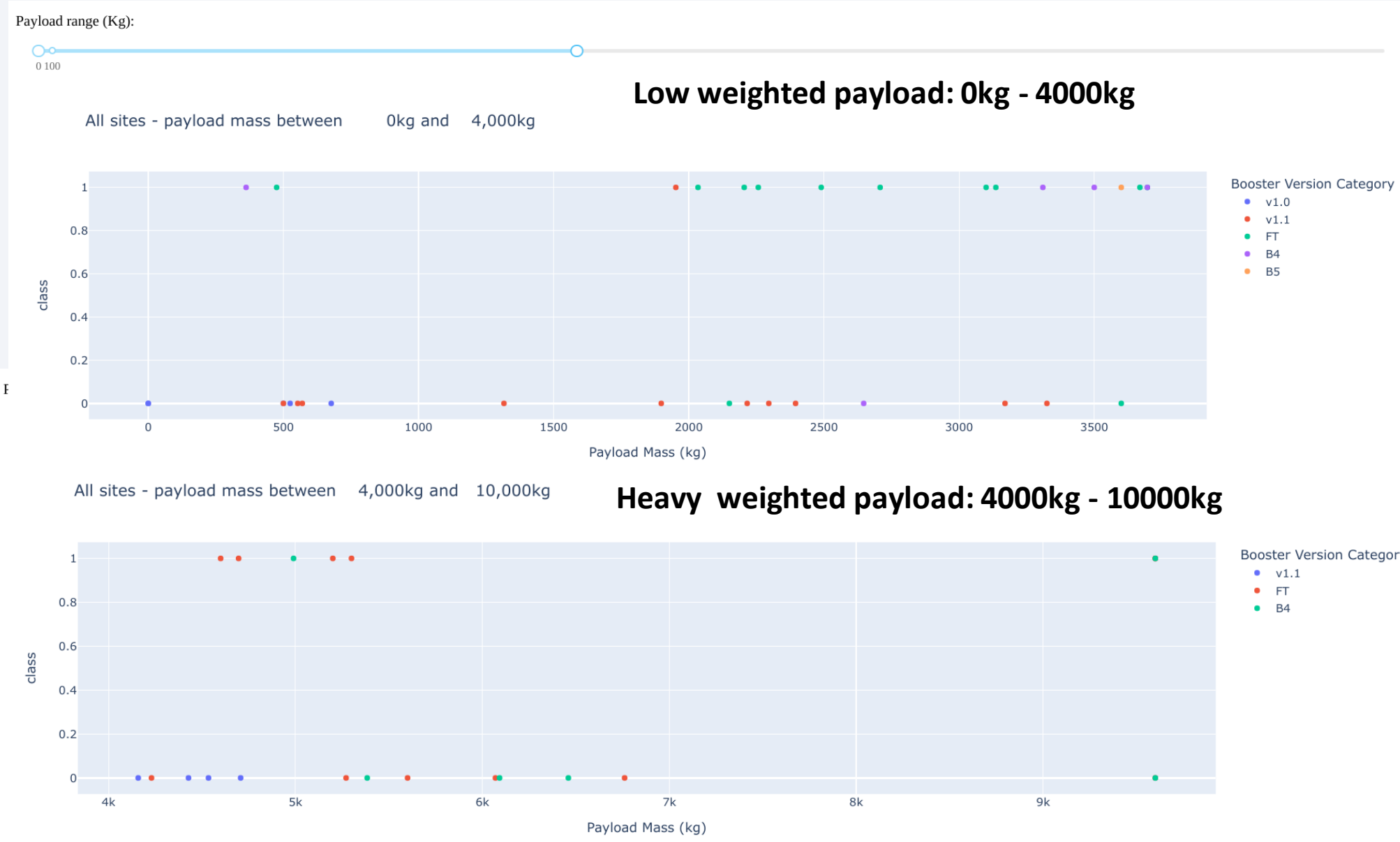
From the pie chart shown above we can see that the launch site KSC LC-39A had the most successful launches among all the launch sites

Pie chart showing the Launch site with the highest launch success ratio



From the above pie chart we can see that the launch site KSC LC-39A has a success rate of 76.9% and a failure rate of 23.1%

Scatter plots of Payload vs Launch Outcome for all sites, with different payload selected in the range slider



We can see that the success rates for low weighted payloads is higher than the heavy weighted payloads

Section 5

Predictive Analysis (Classification)

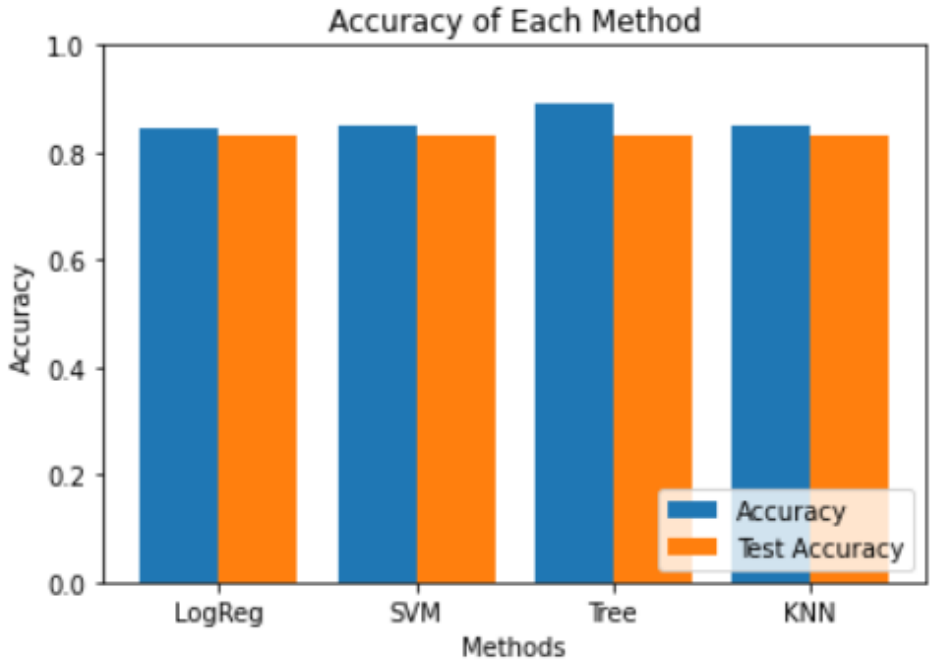
Classification Accuracy

Find the method performs best:

In [30]:

```
print("Model\t\tAccuracy\tTestAccuracy")#,logreg_cv.best_score_)
print("LogReg\t\t{}\t\t{}".format((logreg_cv.best_score_).round(5), logreg_cv.score(X_test, Y_test).round(5)))
print("SVM\t\t{}\t\t{}".format((svm_cv.best_score_).round(5), svm_cv.score(X_test, Y_test).round(5)))
print("Tree\t\t{}\t\t{}".format((tree_cv.best_score_).round(5), tree_cv.score(X_test, Y_test).round(5)))
print("KNN\t\t{}\t\t{}".format((knn_cv.best_score_).round(5), knn_cv.score(X_test, Y_test).round(5)))
```

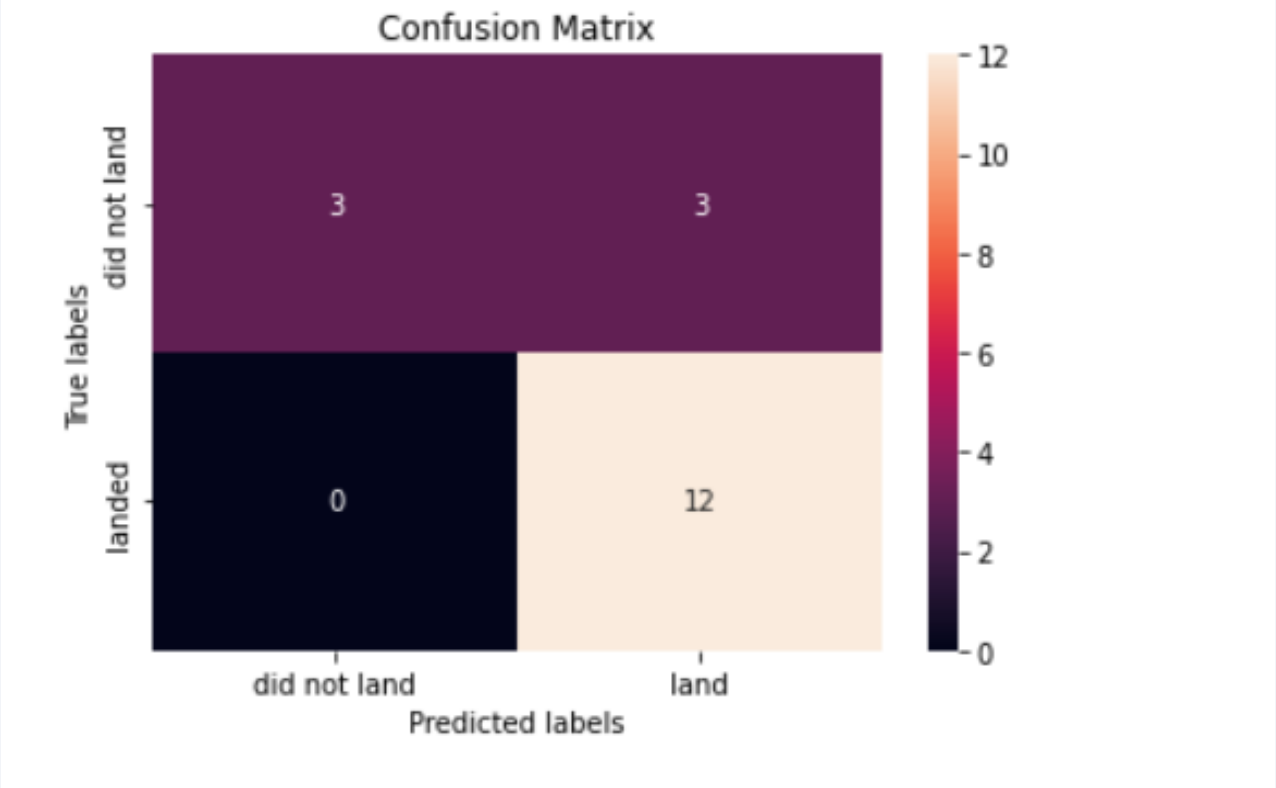
```
Model           Accuracy        TestAccuracy
LogReg          0.84643         0.83333
SVM             0.84821         0.83333
Tree            0.88929         0.83333
KNN             0.84821         0.83333
```



- ***We can see that the decision tree classifier is the classification model with the highest classification accuracy***

Confusion Matrix

- *The confusion matrix for the decision tree classifier shows that the classifier can distinguish between the different classes.*
- *The major problem is the false positives .i.e., unsuccessful landing marked as successful landing by the classifier.*



Conclusions

We can conclude that:

- *The larger the flight amount at a launch site, the greater the success rate at a launch site.*
- ***Launch success rate started to increase in 2013 till 2020.***
- *Orbits* ***ES-L1, GEO, HEO, SSO, VLEO had the most success rate.***
- ***KSC LC-39A had the most successful launches of any sites.***
- ***The Decision tree classifier is the best machine learning algorithm for predictive analysis (classification).***

Acknowledgements

- GitHub repository URL:

https://github.com/indranilch2014/Capstone-project-SpaceX

- Instructors:

Rav Ahuja, Alex Aklson, Aije Egwaikhide, Lakshmi Holla, Svetlana Levitan, Romeo Kienzler, Polong Lin, Joseph Santarcangelo,

Nayef Abo Tayoun, Azim Hirjani, Hima Vasudevan,Saishruthi Swaminathan,

Saeed Aghabozorgi, Yan Luo

- *Special thanks to the instructors:*

https://www.coursera.org/professional-certificates/ibm-data-science?#instructors

Appendix-1

The 'features' dataframe is formed which uses various features from the SpaceX dataset to predict the success through various classification algorithms like K-nearest neighbor, Decision tree classifier, etc.

Features Engineering

By now, you should obtain some preliminary insights about how each important variable would affect the success rate, we will select the features that will be used in success prediction in the future module.

[14]:
```
features = df[['FlightNumber', 'PayloadMass', 'Orbit', 'LaunchSite', 'Flights', 'GridFins', 'Reused', 'Legs', 'LandingPad', 'Block', 'Reus
features.head()
```

[14]:

	FlightNumber	PayloadMass	Orbit	LaunchSite	Flights	GridFins	Reused	Legs	LandingPad	Block	ReusedCount	Serial
0	1	6104.959412	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0003
1	2	525.000000	LEO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0005
2	3	677.000000	ISS	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B0007
3	4	500.000000	PO	VAFB SLC 4E	1	False	False	False	NaN	1.0	0	B1003
4	5	3170.000000	GTO	CCAFS SLC 40	1	False	False	False	NaN	1.0	0	B1004

TASK 7: Create dummy variables to categorical columns

Use the function `get_dummies` and `features` dataframe to apply OneHotEncoder to the column `Orbits`, `LaunchSite`, `LandingPad`, and `Serial`. Assign the value to the variable `features_one_hot`, display the results using the method head. Your result dataframe must include all features including the encoded ones.

[15]:
```
# HINT: Use get_dummies() function on the categorical columns
oh_orbit = pd.get_dummies(features["Orbit"])
oh_launch = pd.get_dummies(features["LaunchSite"])
oh_landing = pd.get_dummies(features["LandingPad"])
oh_serial = pd.get_dummies(features["Serial"])
remainder = features[["FlightNumber","PayloadMass", "Flights", "GridFins", "Reused", "Legs", "Block","ReusedCount"]]
features_one_hot = pd.concat([oh_launch, oh_landing, oh_serial, oh_orbit], axis=1)
features_one_hot.head()
```

[15]:

	CCAFS SLC 40	KSC LC 39A	VAFB SLC 4E	5e9e3032383ecb267a34e7c7	5e9e3032383ecb554034e7c9	5e9e3032383ecb6bb234e7ca	5e9e3032383ecb761634e7cb	5e9e3033383ecbb9e534e7cc	B0003	B(
0	1	0	0	0	0	0	0	0	1	
1	1	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	

Appendix-2

We can see that we have obtained a list of the names of all the booster versions and their corresponding payload mass in kg where there is success in drone ship and in this way we can verify if the output of the previous SQL query is correct or not

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

[28]:
```
%sql select booster_version from SPACEXDATASET where payload_mass__kg_ between 4000 and 6000 and landing_outcome like 'Success (drone shi
```

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

t[28]:

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

[27]:
```
%sql select booster_version,payload_mass__kg_ from SPACEXDATASET where landing_outcome like 'Success (drone ship)'
```

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

t[27]:

Booster_Version	PAYLOAD_MASS__KG_
F9 FT B1021.1	3136
F9 FT B1022	4696
F9 FT B1023.1	3100
F9 FT B1026	4600
F9 FT B1029.1	9600
F9 FT B1021.2	5300
F9 FT B1029.2	3669
F9 FT B1036.1	9600
F9 FT B1038.1	475
F9 B4 B1041.1	9600
F9 FT B1031.2	5200
F9 B4 B1042.1	3500
F9 B4 B1045.1	362
F9 B5 B1046.1	3600

Appendix-3

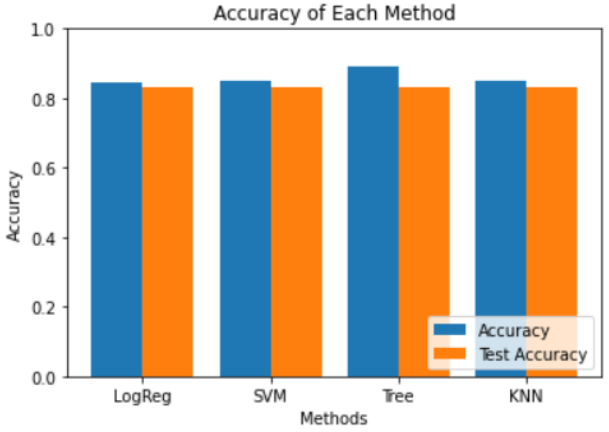
The Python code that was used to plot a Bar graph showing the accuracy on the training data and accuracy on the test data for various machine learning classification algorithms.

We can see that for all the four types of classifiers, the accuracy on the test data is the same that is 83.33%,

but on the training data, the Logistic Regression, Support Vector machine and KNN classifiers have same accuracy about 84.8% and Decision Tree has the highest accuracy which is 88.9%

In [31]:

```
comparison = {}

comparison['LogReg'] = {'Accuracy': logreg_cv.best_score_.round(5), 'TestAccuracy': logreg_cv.score(X_test, Y_test).round(5)}
comparison['SVM'] = {'Accuracy': svm_cv.best_score_.round(5), 'TestAccuracy': svm_cv.score(X_test, Y_test).round(5)}
comparison['Tree'] = {'Accuracy': tree_cv.best_score_.round(5), 'TestAccuracy': tree_cv.score(X_test, Y_test).round(5)}
comparison['KNN'] = {'Accuracy': knn_cv.best_score_.round(5), 'TestAccuracy': knn_cv.score(X_test, Y_test).round(5)}

x = []
y1 = []
y2 = []
for meth in comparison.keys():
    x.append(meth)
    y1.append(comparison[meth]['Accuracy'])
    y2.append(comparison[meth]['TestAccuracy'])


x_axis = np.arange(len(x))

plt.bar(x_axis - 0.2, y1, 0.4, label = 'Accuracy')
plt.bar(x_axis + 0.2, y2, 0.4, label = 'Test Accuracy')

plt.ylim([0,1])
plt.xticks(x_axis, x)

plt.xlabel("Methods")
plt.ylabel("Accuracy")
plt.title("Accuracy of Each Method")
plt.legend(loc='lower right')
plt.show()
```



Appendix-4

We have obtained the column names and datatypes of the columns in the SpaceX dataset

And we have also used the value_counts() method on the pandas dataframe to determine the total number of launches from each Launch site

Identify which columns are numerical and categorical:

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

Thank you!