Winning Space Race with Data Science

Jesus Pulido
01/31/2024

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
- - Machine Learning Prediction
- · Summary of all results
- - Exploratory Data Analysis result
- - Interactive analytics in screenshots
- - Predictive Analytics 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 land successfully?
- The interaction amongst various features that determine the success rate of a successful landing.
- What operating conditions needs to be in place to ensure a successful landing program.

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Describe how data was collected
- Perform data wrangling
  - Describe how data was processed
- 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

- **1. SpaceX API:**
- Retrieved data via GET requests, decoding JSON responses, and structuring into a Pandas dataframe.
- Rigorous cleaning addressed missing values for data integrity.
- **2. Web Scraping Wikipedia:**
- Employed BeautifulSoup for web scraping Falcon 9 launch records from Wikipedia.
- Extracted data from HTML tables, seamlessly integrating it into the dataset.
- This meticulous process ensured a comprehensive dataset, combining API and web-sourced data, paving the way for detailed Falcon 9 launch record analysis.

Data Collection – SpaceX API

- We use get request to extract the data from SpaceX, clean the requested data, clean it and do wrangling and formatting the data for data manipulation.

- Here is the link of the Git-hub repository: https://github.com/ChuyPL23/Space-B/blob/main/SpaceX-data-collection-API.ipynb.

From the `rocket` column we would like to learn the booster name.

In [2]:
```
# Takes the dataset and uses the rocket column to call the API and append the data to the list
def getBoosterVersion(data):
    for x in data['rocket']:
       if x:
        response = requests.get("https://api.spacexdata.com/v4/rockets/"+str(x)).json()
        BoosterVersion.append(response['name'])
```

From the `launchpad` we would like to know the name of the launch site being used, the logitude, and the latitude.

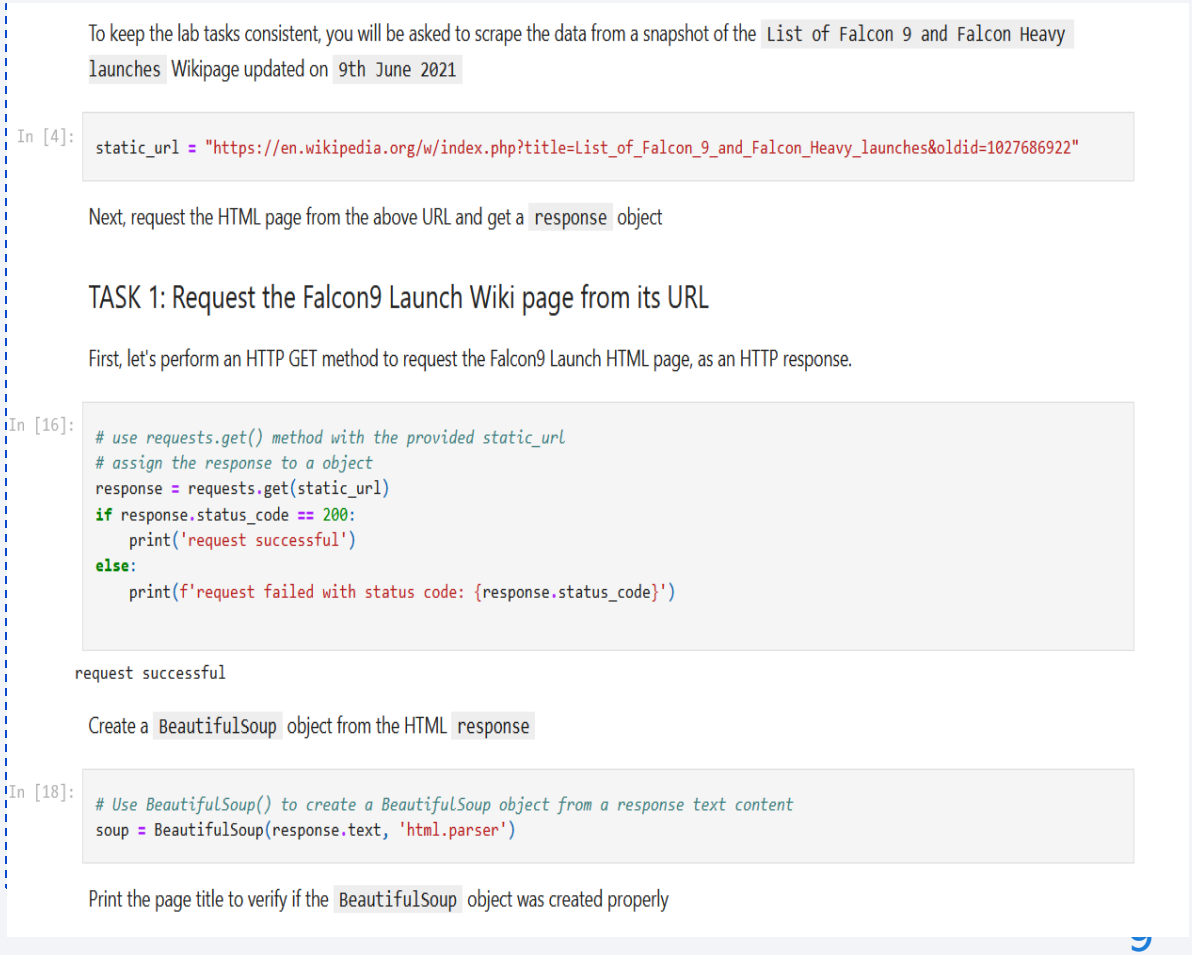
In [3]:
```
# Takes the dataset and uses the launchpad column to call the API and append the data to the list
def getLaunchSite(data):
    for x in data['launchpad']:
       if x:
         response = requests.get("https://api.spacexdata.com/v4/launchpads/"+str(x)).json()
         Longitude.append(response['longitude'])
         Latitude.append(response['latitude'])
         LaunchSite.append(response['name'])
```

From the `payload` we would like to learn the mass of the payload and the orbit that it is going to.

In [4]:
```
# Takes the dataset and uses the payloads column to call the API and append the data to the lists
def getPayloadData(data):
    for load in data['payloads']:
       if load:
        response = requests.get("https://api.spacexdata.com/v4/payloads/"+load).json()
        PayloadMass.append(response['mass_kg'])
        Orbit.append(response['orbit'])
```

Data Collection - Scraping

- We used web scrapping to an specific rocket in this case our reference was Falcon9 records launch and to facilitate we used BeautifulSoup.
- We converted the data into a Pandas DataFrame.
- Here is the link in Git-Hub: https://github.com/ChuyPL23/Space-B/blob/main/SpaceX-webscraping.ipynb.



To keep the lab tasks consistent, you will be asked to scrape the data from a snapshot of the `List of Falcon 9 and Falcon Heavy launches` Wikipage updated on `9th June 2021`

In [4]:
```
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 [16]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url)
if response.status_code == 200:
    print('request successful')
else:
    print(f'request failed with status code: {response.status_code}')
```

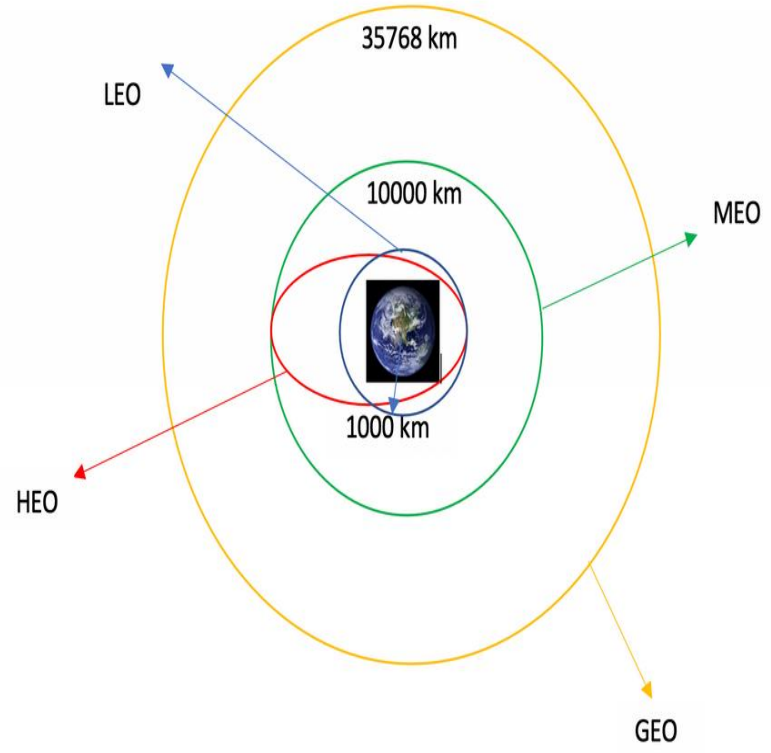
```
request successful
```

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

In [18]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(response.text, 'html.parser')
```

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

Data Wrangling



We performed exploratory data analysis and determined training and test labels for each site.

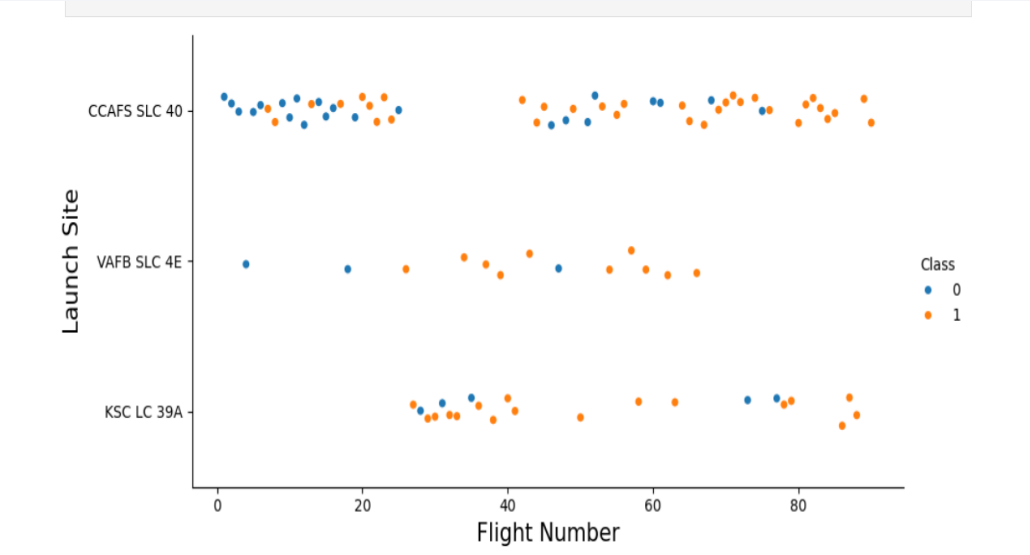
We calculated the number of launches and if it was successful or non-successful for each site.

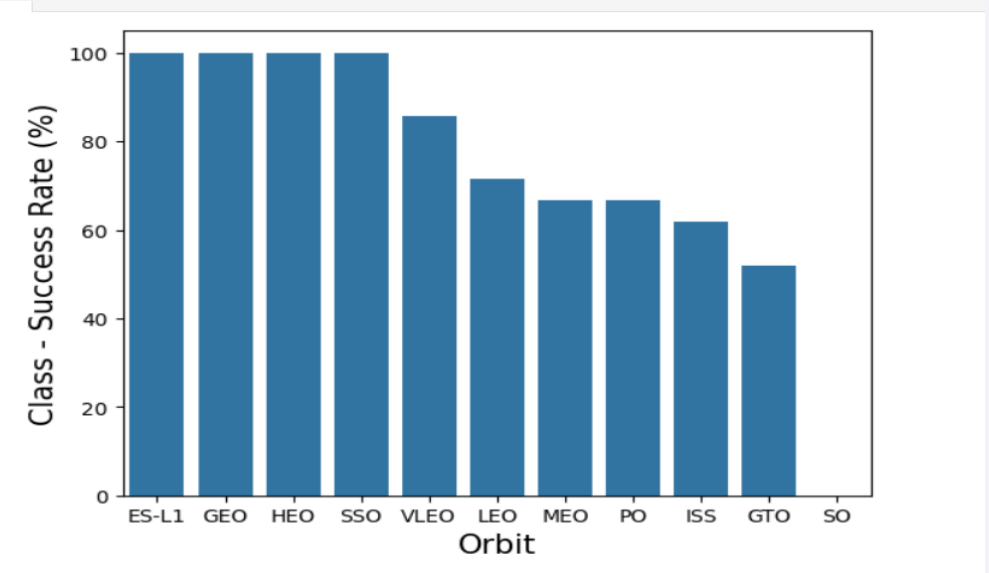
The final results that we had as an outcome we converted to csv to export that info for the next analysis.

Git-Hub
link: https://github.com/ChuyPL23/Space-B/blob/main/SpaceX-Data%20wrangling.ipynb

EDA with Data Visualization

- We analyze the data by visualizing the relationship between flight number and launch site, payload mass, orbit type and date to compare what independent variable are affecting the output and analyze how variables has fluctuations against each other. URL: https://github.com/ChuyPL23/Space-B/blob/main/EDA-data-visualization.ipynb.





EDA with SQL

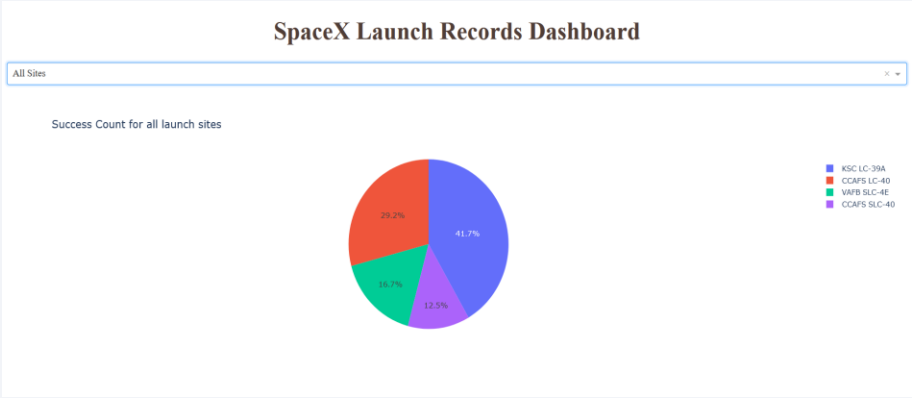
- We loaded our dataset into SQL to make some queries and we use the magic function to use SQL language over jupyter notebooks.
- We applied EDA over this queries to find insights in our data.
  - Displaying the unique names of launch sites
  - Total mass carried by boosters launches by NASA
  - Booster version
  - Date when the first successful landing was achieved
  - Total number of successful and failure mission outcomes
  - Link: https://github.com/ChuyPL23/Space-B/blob/main/SQL-SpaceX_sqllite.ipynb

Build an Interactive Map with Folium

- We build a map with all the markers, circles, lines, locations over the success and failure launches for each site using folium map and given by our dataset it was provided.
- We converted our success rate into a 2 variables: 1 (success) 0 (failure) giving this information we were capable to determine what was the result of each launch on every site.
- Using a color labeled marker-cluster to identify which one of launch sites had been succeeded.
- We answered some questions related to the nearest proximities as railways, highways and coastlines, also very important to identify if this location has proximity to the city or small towns due to dangerous situations could possibly happen.
- Link: https://github.com/ChuyPL23/Space-B/blob/main/SPACEX-Launch-site_location.ipynb

Build a Dashboard with Plotly Dash

- We developed a versatile Dashboard using Plotly Dash, equipped with multiple functionalities for plotting extensive graphs. This platform serves various purposes due to its diverse capabilities
- We created pie charts depicting the distribution of launch sites for each year, both individually and collectively, to observe their respective behaviors.
- We plotted in a scatter plot of every booster version category against payload range (KM).
- Here is an image of one of the dashboarding plots:



Predictive Analysis (Classification)

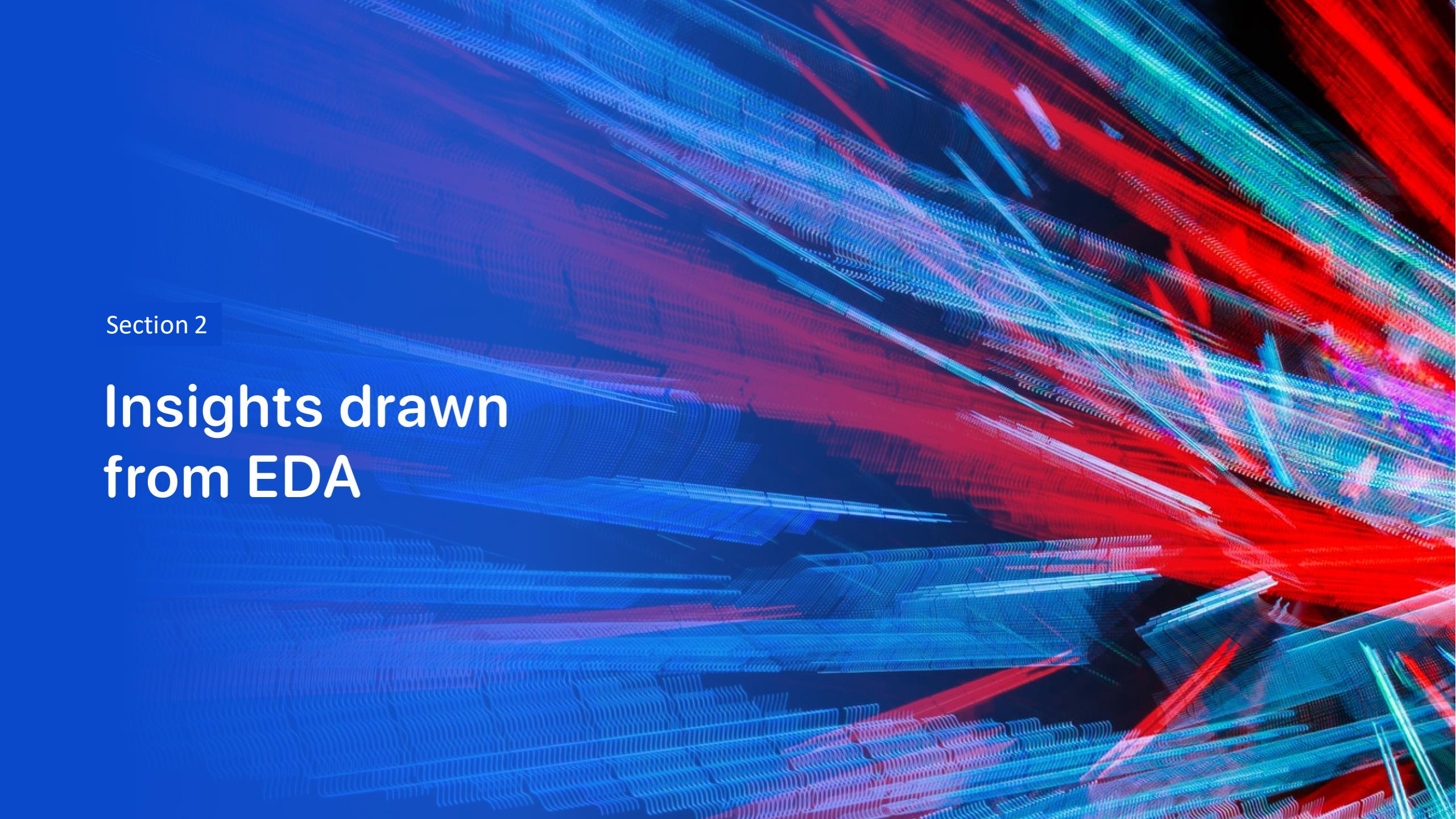
- We loaded the data into DataFrame from Pandas as csv format, clarify the depedent variable to split the data into a training and testing in 2 diferent variables in this case X and Y.
- We built a different models to determine which has the best fit for our comparison and which model can predict the best output accuracy.
- We use multiple of models and functions to improve our output such as logistics regression and observe all the details in every output improvement if it was the case.
- We found the best classification model.
- Link: https://github.com/ChuyPL23/Space-B/blob/main/SpaceX_Machine-Learning-Prediction.ipynb

Results

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

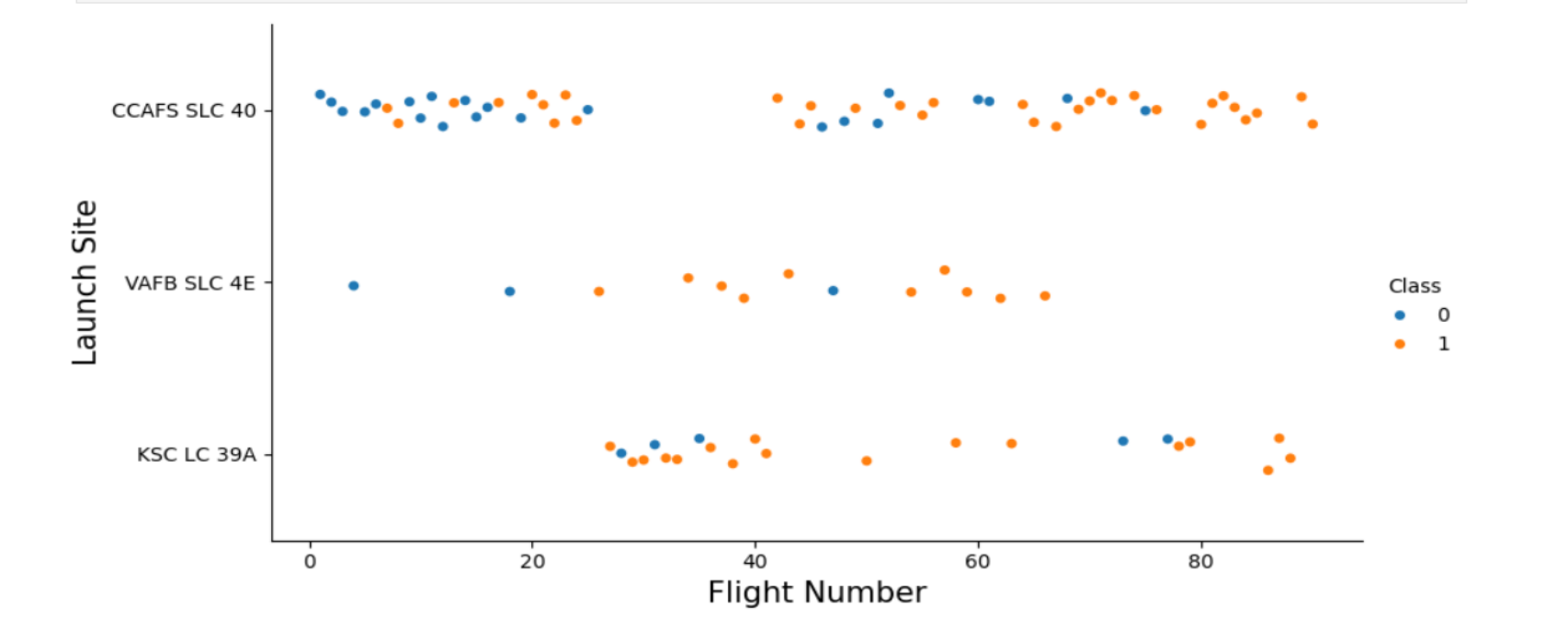
Section 2

Insights drawn from EDA

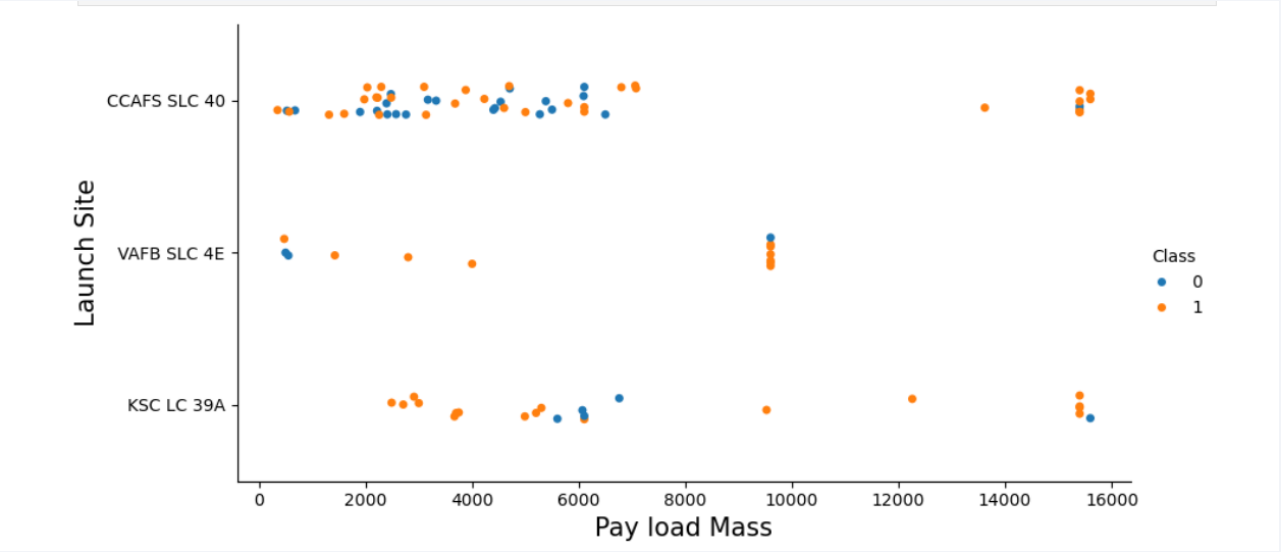


Flight Number vs. Launch Site

- We observed a direct correlation between the increasing number of flights and the consistent success rate. Notably, out of 80 flights, there were no instances of non-successful launches."

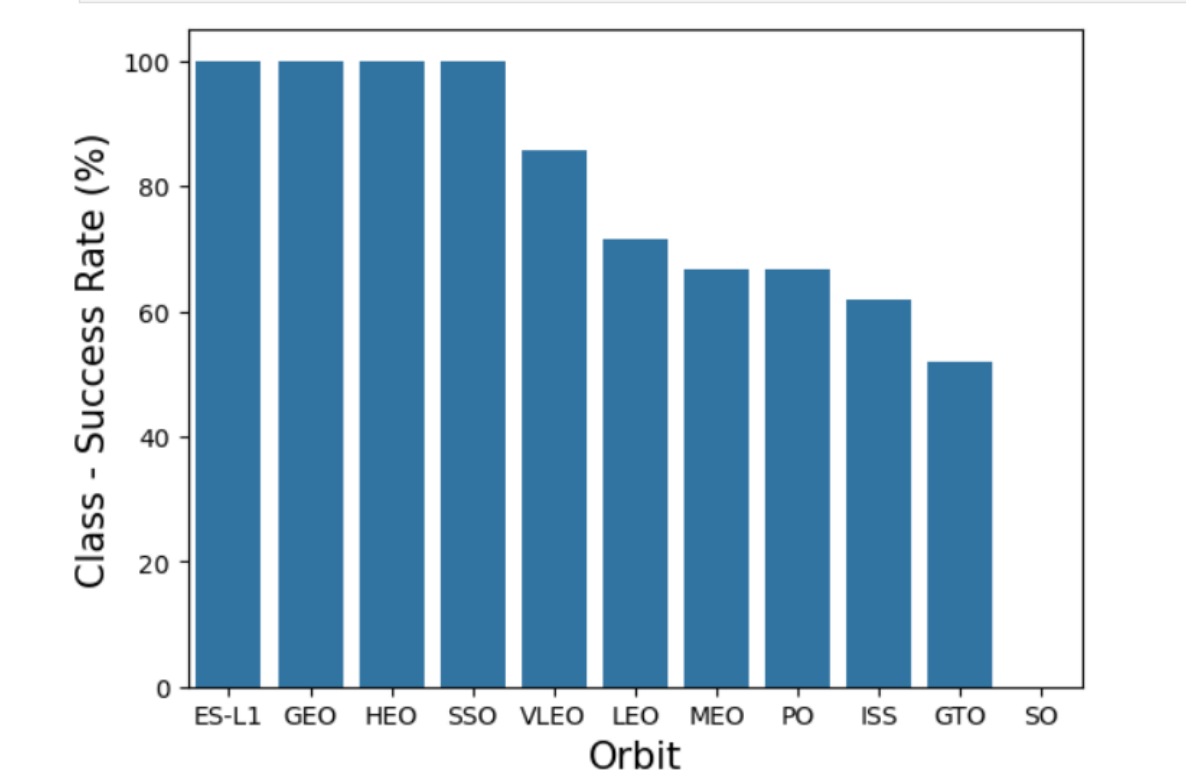


Payload vs. Launch Site



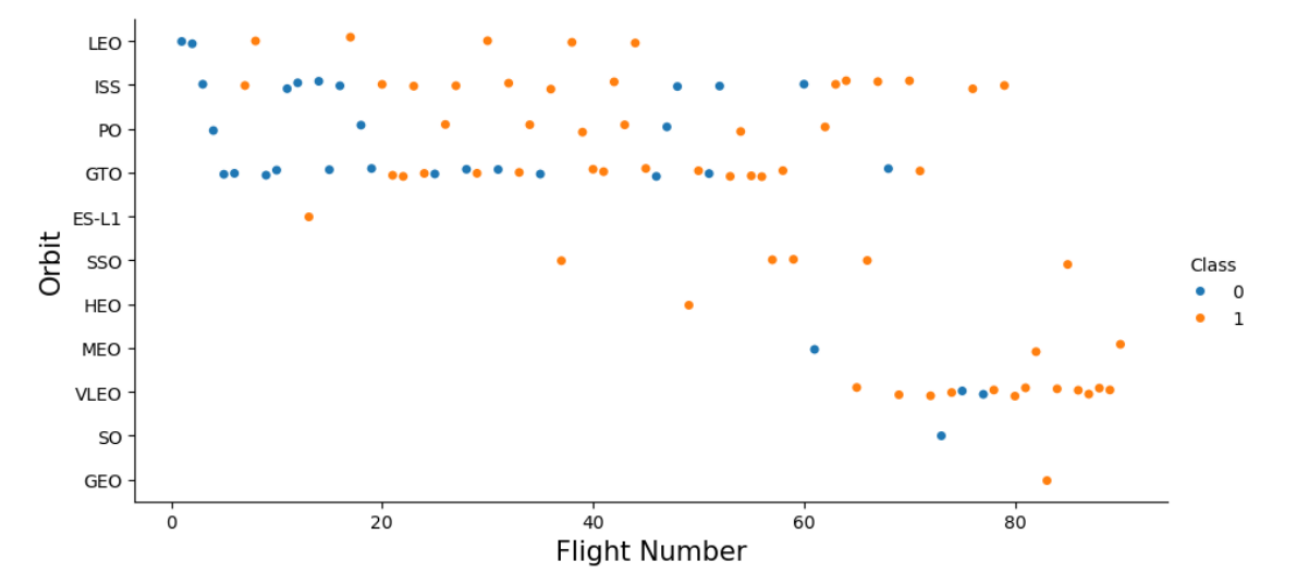
The greater of the payload mass for CCAPS SLC 40 site the higher success rate to the rocket.

Success Rate vs. Orbit Type



- As we can observe into this barplot, there are four highest success rate which are: ES-L1, GEO, HEO and SSO.

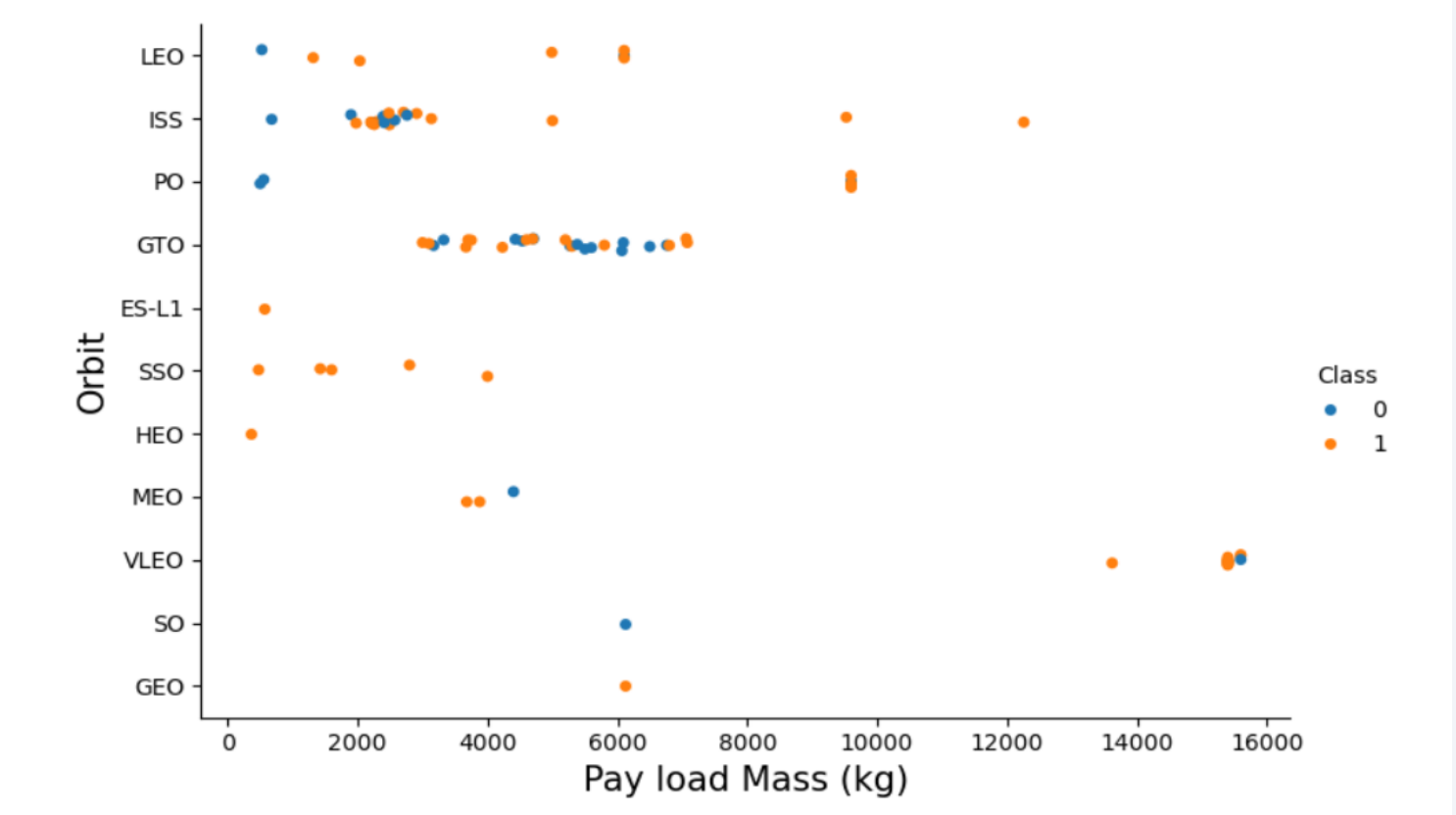
Flight Number vs. Orbit Type



- From this plot we can analyze that LEO has the highest success rate considering number of flights which is lower than other sites but has higher rate in percentage.

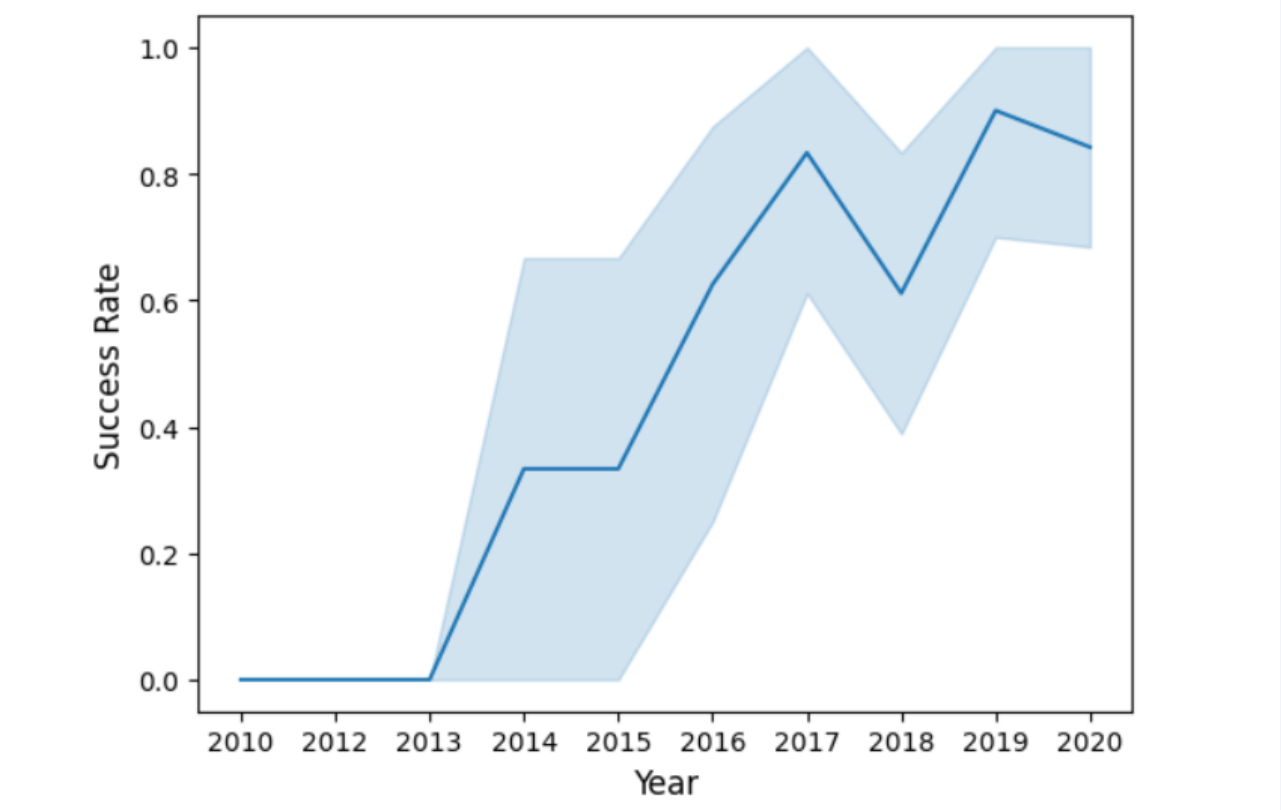
Payload vs. Orbit Type

- We can see that LEO, ISS, PO orbits has the highest success rate in payload mass.



Launch Success Yearly Trend

- We can observe that the success rate since 2013 kept increasing until 2020.



All Launch Site Names

- We harnessed the power of SQL to manipulate information, utilizing the DISTINCT keyword to selectively choose unique names without any duplicates.

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

Launch Site Names Begin with 'CCA'

- We used the query below to select only the launch sites starting with CCA.

In [92]:
```
%sql SELECT * FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT 5;
```

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

Out[92]:

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_	Orbit	Customer	Mission_Outcome	Landing_Outc
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (paracl
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 (paracl
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No att
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No att
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No att

Total Payload Mass

- We calculated the total payload mass as TotalPayloadMass as a label for only NASA (CRS) with the query below.

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

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS TotalPayloadMass FROM SPACEXTBL WHERE Customer = 'NASA (CRS)';
```

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

TotalPayloadMass
45596

Average Payload Mass by F9 v1.1

- We calculated the average of F9v1.1 as Average_F9v11 with the querie below.

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

```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS Average_F9v11 FROM SPACEXTBL WHERE Booster_Version  = 'F9 v1.1';
```

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

Average_F9v11
2928.4

First Successful Ground Landing Date

- We selected the MIN value of Date to retrieve the first date where Date equals to `Success (ground pad)`.

```
%sql SELECT MIN(Date) as FirstLandingSuccess FROM SPACEXTBL WHERE Landing_Outcome = 'Success (ground pad)';
```

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

FirstLandingSuccess
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We selected the booster version where success (drone ship) with one condition as equals or greater than 4000 and equals or less than 6000 with the query below.

```
%%sql SELECT Booster_Version FROM SPACEXTBL
WHERE Landing_Outcome = 'Success (drone ship)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 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

- We counted all the mission outcomes and grouped it as Total_Mission_Outcomes with the query below.

```
%%sql SELECT Mission_Outcome, COUNT(*) AS Total_Mission_Outcomes
FROM SPACEXTBL GROUP BY Mission_Outcome;
```

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

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

Boosters Carried Maximum Payload

- We used a subquery to determine 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

- We used SUBSTR to extract the month of Date and retrieve only Landing Outcome, Booster Version, Launch Site happened in 2015 with Landing Outcome equals to Failure (drone ship).

```
%%sql SELECT SUBSTR(Date, 6,2) AS Month, Landing_Outcome, Booster_Version, Launch_Site
FROM SPACEXTBL WHERE SUBSTR(Date, 0,5) = '2015'
AND Landing_Outcome ='Failure (drone ship)';
```

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

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

. 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.

```
%%sql SELECT Landing_Outcome, COUNT(*) AS Outcome_Count
FROM SPACEXTBL
WHERE Date BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY Landing_Outcome
ORDER BY Outcome_Count DESC;
```

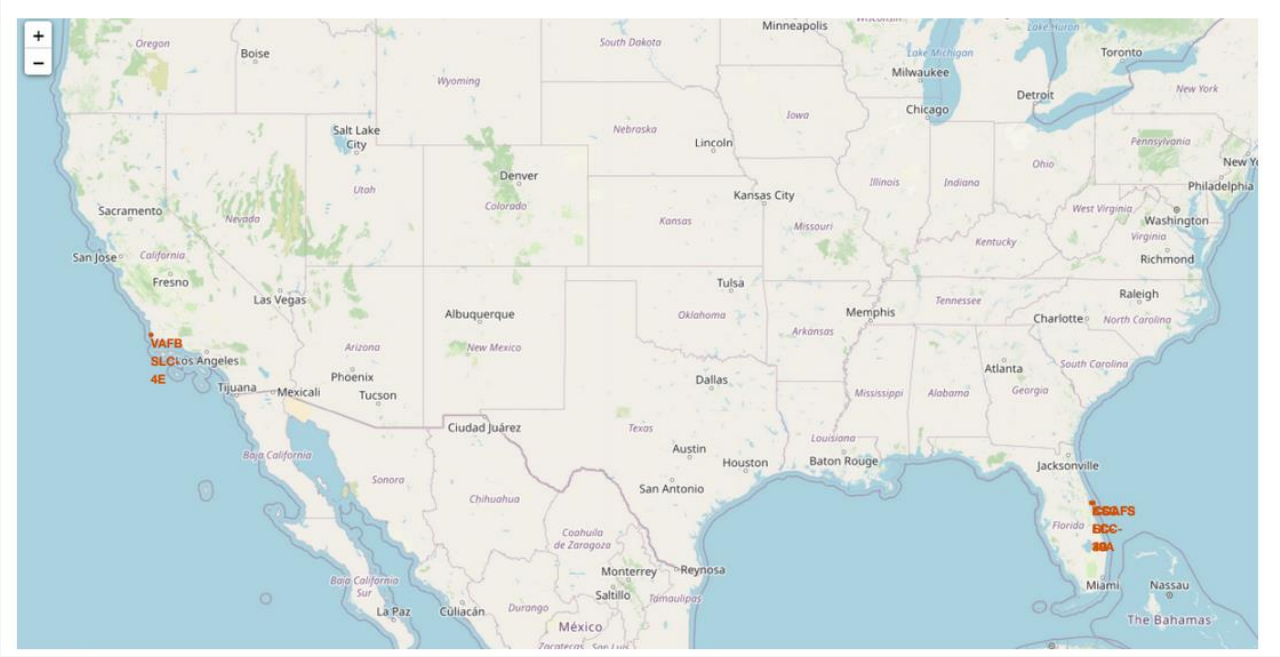
```
 * sqlite:///my_data1.db
Done.
```

Landing_Outcome	Outcome_Count
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1


Section 3
Launch Sites
Proximities Analysis

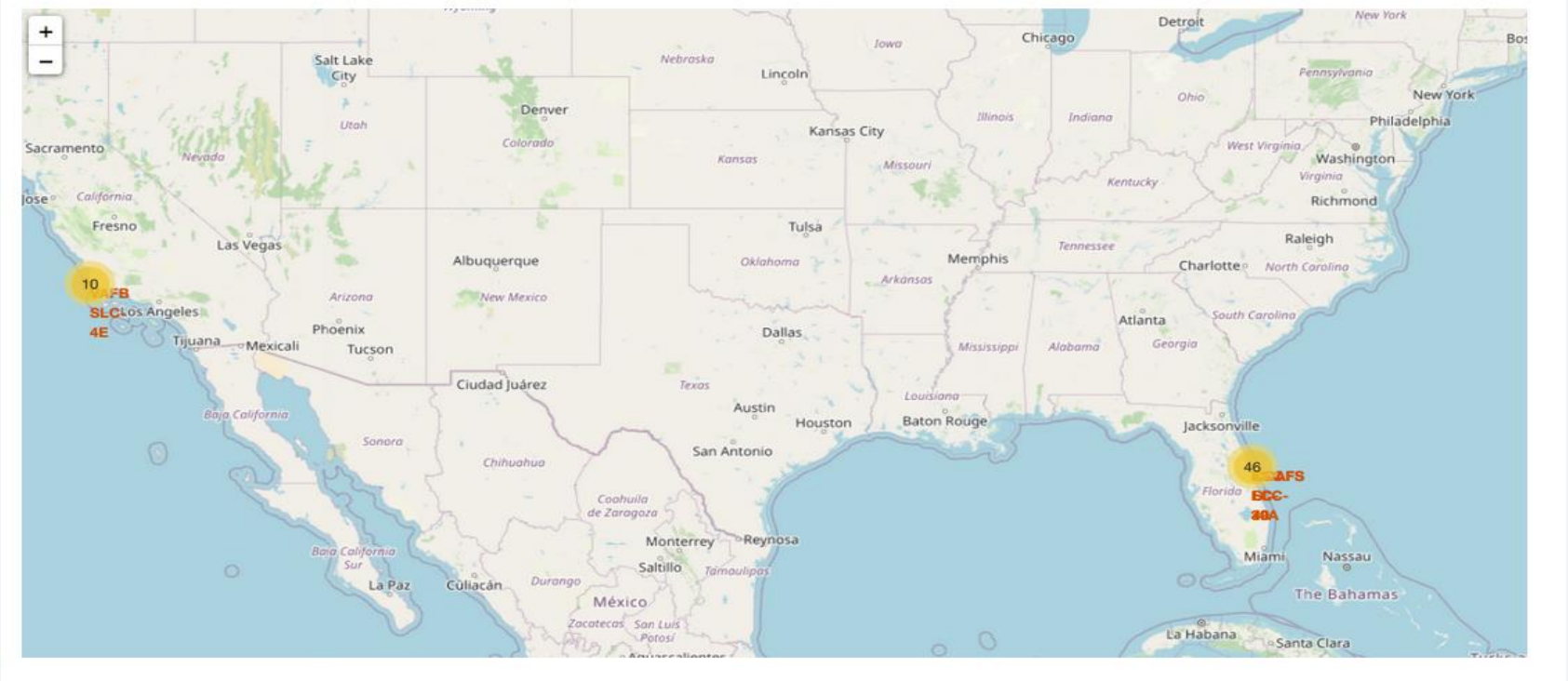
<Folium Map Screenshot 1>

- **LAUNCH SITES WORLDWIDE OF SPACE X.**
- They are close to the coast as specifically manner, to prevent risk situations from diverse factors.



<Folium Map Screenshot 2>

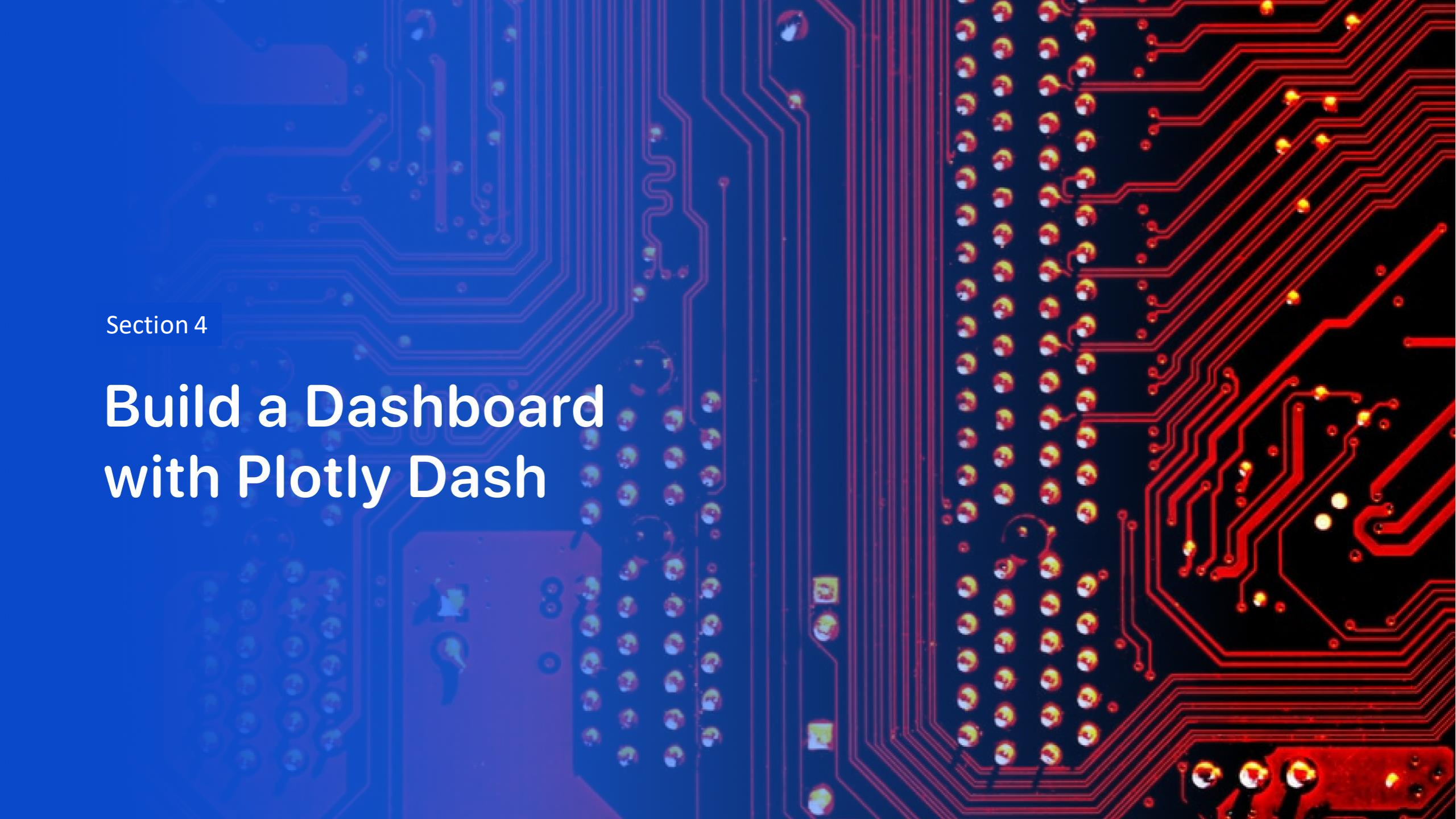
- **Every location in United States from the states of California and Florida.**
- We can noticed that they are to close to each other due to easy transportation



<Folium Map Screenshot 3>

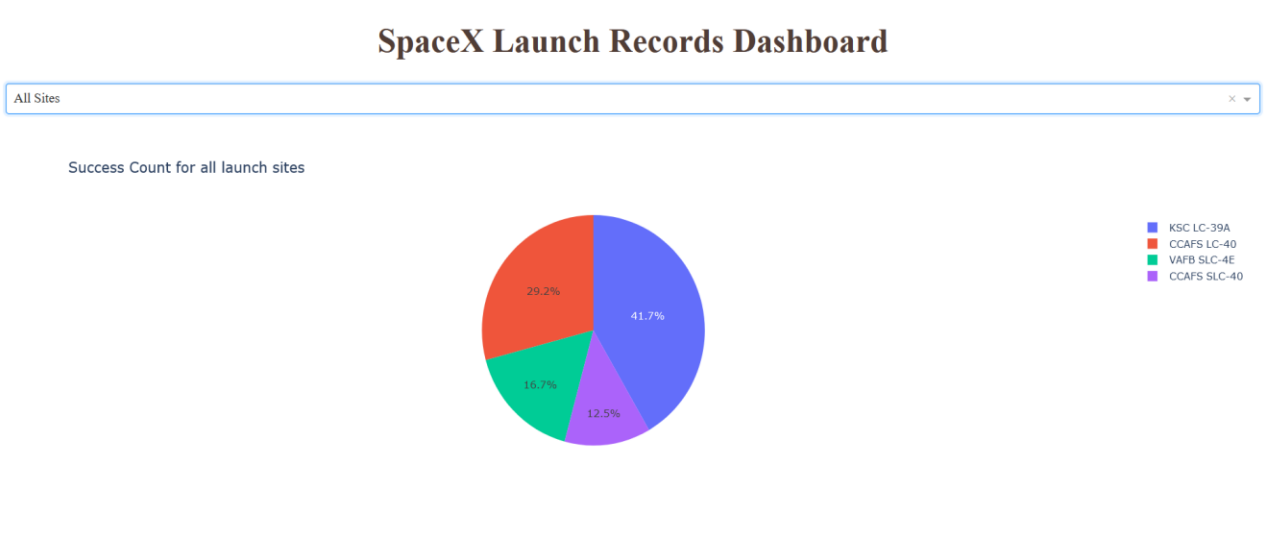
PROXIMITIES AS RAILROADS, HIGHWAYS, NEAREST CITIES

We can see that the proximities plays very important role due to mobility in certain cases as to transport heavy cargo.


Section 4
Build a Dashboard with Plotly Dash

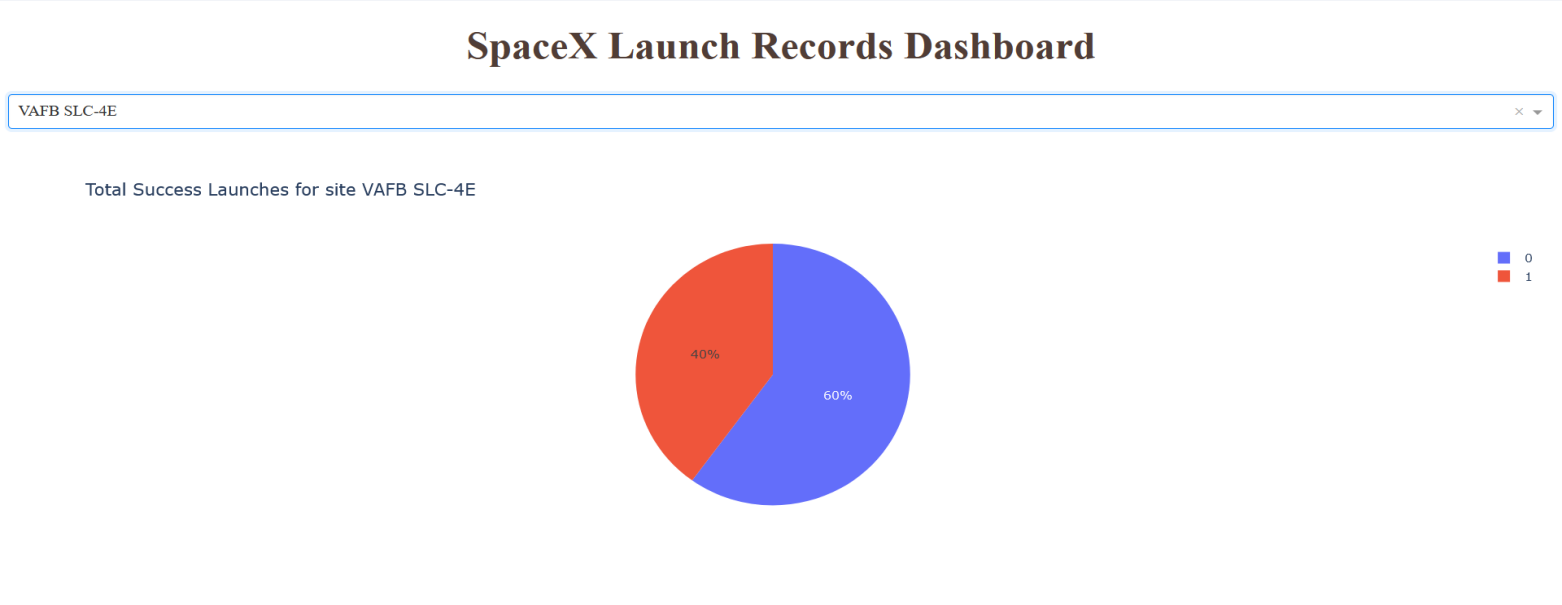
<Dashboard Screenshot 1>

- **ALL SITES SELECTED PIE CHART FOR THE LAUNCH SITE WITH THE HIGHEST LAUNCH SUCCESS RATIO.**
- We can see that KSC LC 39A has the highest launch ratio with 41.7% of succeeded.



<Dashboard Screenshot 2>

- **SUCCESS RATE FOR VAFB SLC-4E**
- We can analyze the percentage of success rate for VAFB SLC-4E with 40% of success rate that concludes this is not the best launch site.



<Dashboard Screenshot 3>

- SUCCESS RATE COUNT ON PAYLOAD MASS OF EACH BOOSTER CATEGORY

Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree was the best classifier model for this project giving the best result due its accuracy.

```
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': [None, 'log2', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

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

```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': [None, 'log2', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

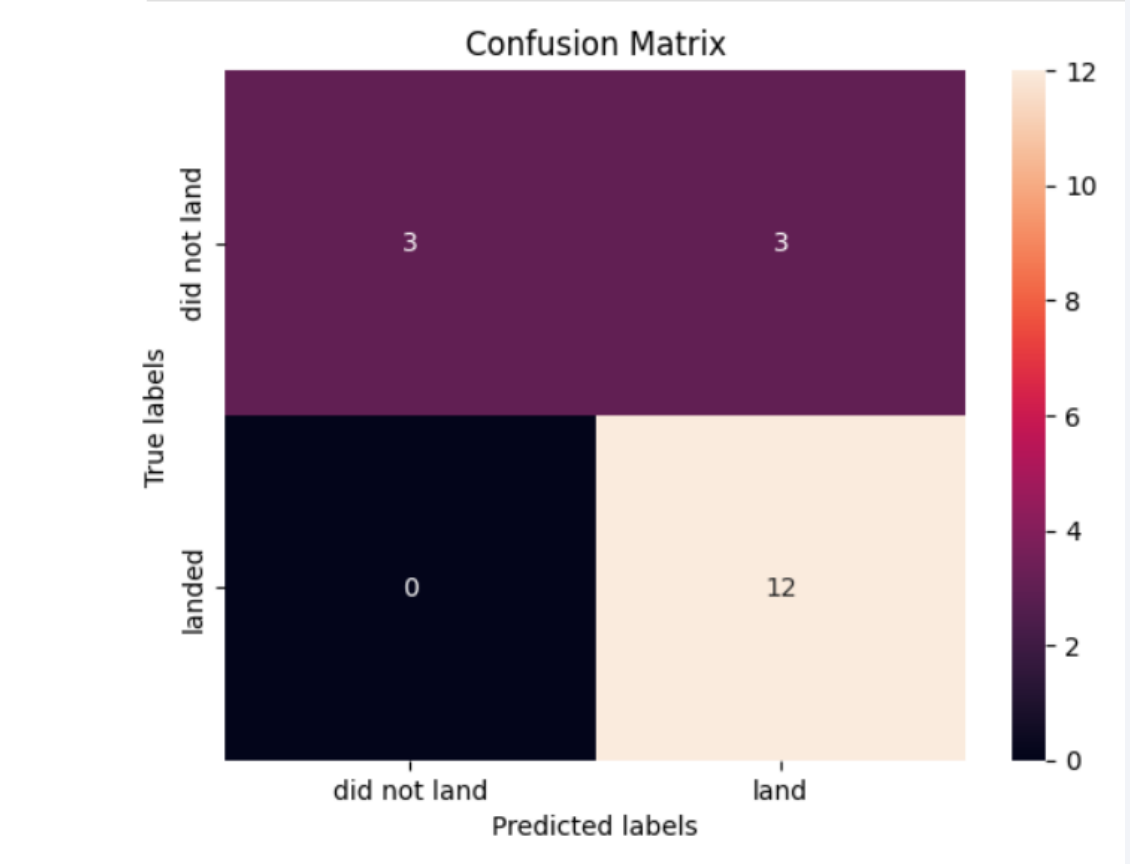
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

```
tuned hpyerparameters :(best parameters)  {'criterion': 'entropy', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf':
1, 'min_samples_split': 10, 'splitter': 'random'}
accuracy : 0.875
```

Confusion Matrix

- The confusion matrix for the decision tree classifier reveals its ability to distinguish between various classes. However, a notable issue arises with false positives, signifying instances where the classifier incorrectly identifies an unsuccessful landing as a successful one.



Conclusions

- A positive correlation exists between the number of flights at a launch site and its success rate.
- The overall launch success rate demonstrated an upward trend from 2013 to 2020.
- Orbits such as ES-L1, GEO, HEO, SSO, and VLEO exhibited notably high success rates.
- KSC LC-39A emerged as the launch site with the highest number of successful launches.
- The Decision Tree classifier proved to be the most effective machine learning algorithm for this analysis.

Thank you!