IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Stuti Ghildiyal
15 May 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
- EDA with SQL
- EDA with Data Visualization
- Interactive Visualization with Folium
- Machine Learning prediction

Summary of all results

- Exploratory Data Analysis Results
- Interactive Analytics Results
- Predictive Analysis Results

INTRODUCTION

- Project background and context

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. Therefore, if we can determine if the first stage will land, we can determine the cost of a launch. Unlike other rocket providers, SpaceX's Falcon 9 Can recover the first stage. This information can be used by other companies that want to compete and bid against SpaceX.

- Problems you want to find answers

✓What factors determines if the rocket with land successfully ?

✓How is each factor affecting the success rate of the first stage?

✓What condition does SpaceX have to achieve to get ensure a successfully landing ?

Section 1

Methodology

Methodology

- Executive Summary
- Data collection methodology:

  Data was collected using SpaceX REST API and web scaping from Wikipedia.
- Perform data wrangling

  One-hot encoding was used to determine the label for training supervised models.
- 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 two method:

1. SpaceX Rest API : This API will give us data about Launches, outcome of the landing, the type of the landing, payload delivered ,rocket used and landing specific outcomes.

   The API URL starts with “api.spacexdata.com/v4/”

2. Web Scraping : Another way of obtaining Falcon 9 Launch data is through web scrapping from Wikipedia using BeautifulSoup.

Data Collection – SpaceX API

We used the get request to the SpaceX API to collect data, clean the requested data and after doing basic formatting, export the data to csv

GIT LINK:

https://github.com/stuti2907/IBM-data-Science/blob/master/Data%20collection%20with%20API.ipynb

1.Getting Response from API

```
spacex_url="https://api.spacexdata.com/v4/launches/past"
response = requests.get(spacex_url).json()
```

2.Converting Response to a .json File

```
response = requests.get(static_json_url).json()
data = pd.json_normalize(response)
```

3. Apply custom functions to clean the data

```
getLaunchSite(data)
getPayloadData(data)
getCoreData(data)
```

```
getBoosterVersion(data)
```

4.Assign list to dict then Data frame

```
launch_dict = {'FlightNumber': list(data['flight_number']),
'Date': list(data['date']),
'BoosterVersion':BoosterVersion,
'PayloadMass':PayloadMass,
'Orbit':Orbit,
'LaunchSite':LaunchSite,
'Outcome':Outcome,
'Flights':Flights,
'GridFins':GridFins,
'Reused':Reused,
'Legs':Legs,
'LandingPad':LandingPad,
'Block':Block,
'ReusedCount':ReusedCount,
'Serial':Serial,
'Longitude': Longitude,
'Latitude': Latitude}
```

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

5.Filter DF and export to CSV

```
data_falcon9 = df.loc[df['BoosterVersion']!="Falcon 1"]
```

```
data_falcon9.to_csv('dataset_part_1.csv', index=False)
```

Data Collection - Scraping

We applied web scrapping to HTML Falcon 9 launch records with BeautifulSoup

Parsed the table and converted it into a pandas dataframe.

GIT LINK:

https://github.com/stuti2907/IBM-data-Science/blob/master/Data%20collection%20with%20Web%20Scraping.ipynb

1.Getting Response from HTML

2.Create a BeautifulSoup object from HTML response

3. Extract all columns/variable from HTML table header

4.Create a dataframe by parsing the HTML table

5.Export the date to CSV

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

```
soup = BeautifulSoup(data, 'html5lib')
```

```
column_names = []
for row in first_launch_table.find_all('th'):
    name = extract_column_from_header(row)
    if (name != None and len(name) > 0):
        column_names.append(name)
```

```
launch_dict= dict.fromkeys(column_names)

# Remove an irrelvant column
del launch_dict['Date and time ( )']

# Let's initial the launch_dict with each value to be an empty list
launch_dict['Flight No.'] = []
launch_dict['Launch site'] = []
launch_dict['Payload'] = []
launch_dict['Payload mass'] = []
launch_dict['Orbit'] = []
launch_dict['Customer'] = []
launch_dict['Launch outcome'] = []
# Added some new columns
launch_dict['Version Booster']=[]
launch_dict['Booster landing']=[]
launch_dict['Date']=[]
launch_dict['Time']=[]
```

```
df.to_csv('spacex_web_scraped.csv', index=False)
```

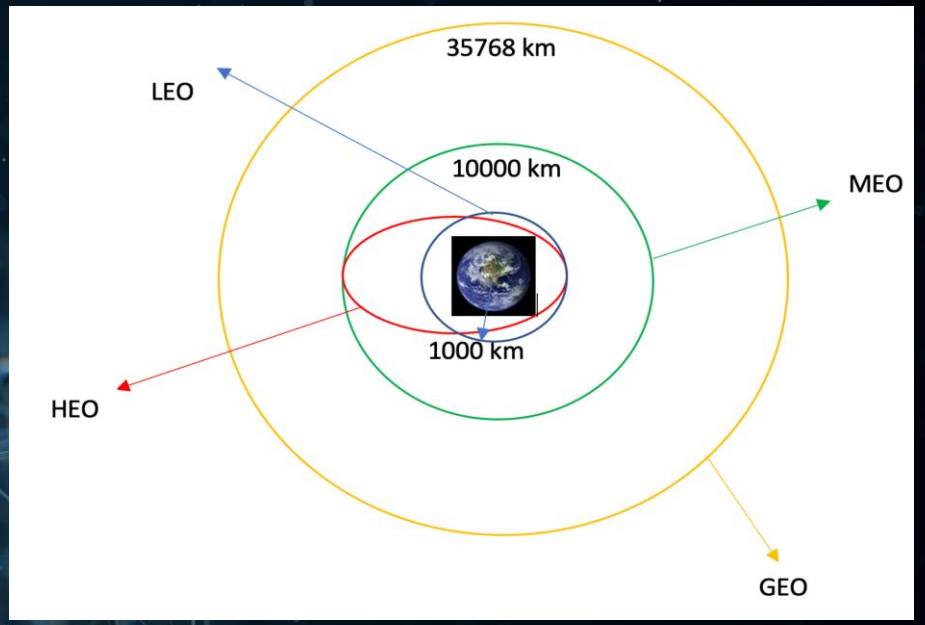
Data Wrangling

Here we performed some Exploratory Data Analysis (EDA) to find some patterns in the data and determine what would be the label for training supervised models.

Process:

Perform EDA on Data

- Calculate the number of Launches on each site.
- Calculate the number and occurrence of each orbit.
- Calculate the number and occurrence of mission outcome per orbit type
- Create a landing outcome label from outcome column



Each Launch aims for a particular orbit. Above diagram shows some orbit used by SpaceX Rockets

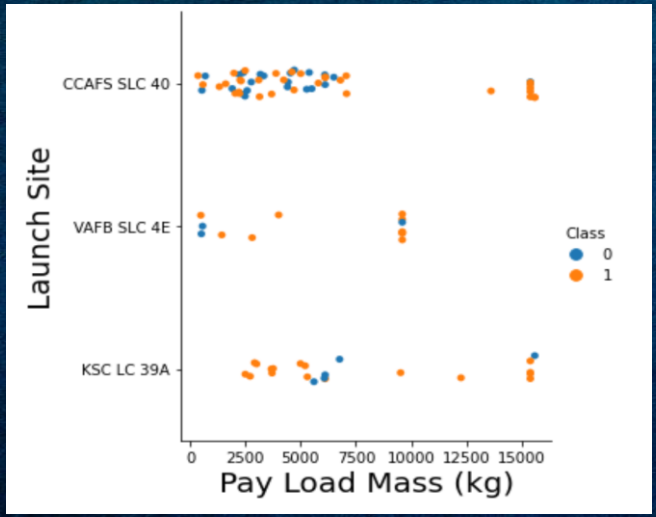
GIT LINK:

https://github.com/stuti2907/IBM-data-Science/blob/master/Data_wrangling.ipynb

EDA with Data Visualization

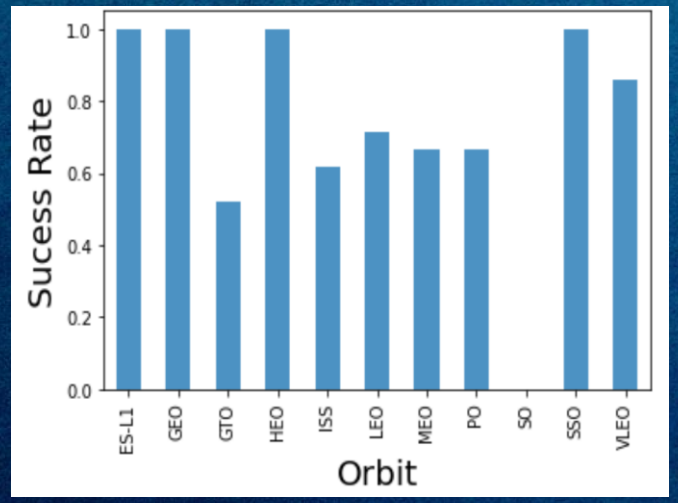
Scatter plot :We explored data by plotting scatter plot b/w payload Mass and Flight Number, Flight number and launch Site, payload and launch site, flight number and orbit type, payload and orbit type.

Scatter plots are used **to determine whether two variables have a relationship or correlation**.



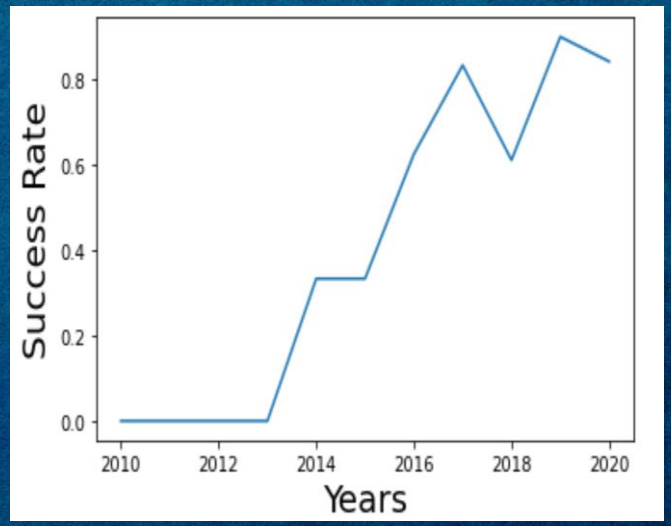
Bar Plot: Success rate of each orbit type.

A bar diagram makes it easy to compare sets of data between different groups at a glance. The graph represents categories on one axis and a discrete value in the other. The goal is to show the relationship between the two axes. Bar charts can also show big changes in data over time



Line Plot: Launch success yearly trend

Line graphs are useful in that they show data variables and trends very clearly and can help to make predictions about the results of data not yet recorded.



GIT LINK:

https://github.com/stuti2907/IBM-data-Science/blob/master/eda-dataviz.ipynb

EDA with SQL

We loaded the SpaceX data to Db2 table and performed EDA with SQL to get insights about the data. Below are the queries that we performed:

- Display the names of the unique launch sites in the space mission.
- Display 5 records where launch sites begin with the string 'CCA'.
- Display the total payload mass carried by boosters launched by NASA (CRS).
- Display 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 using a subquery.
- List the failed landing outcomes in drone ship, their booster versions, and launch site names for in year 2015.
- 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.

GIT LINK: https://github.com/stuti2907/IBM-data-Science/blob/master/EDA_with_SQL.ipynb

Build an Interactive Map with Folium

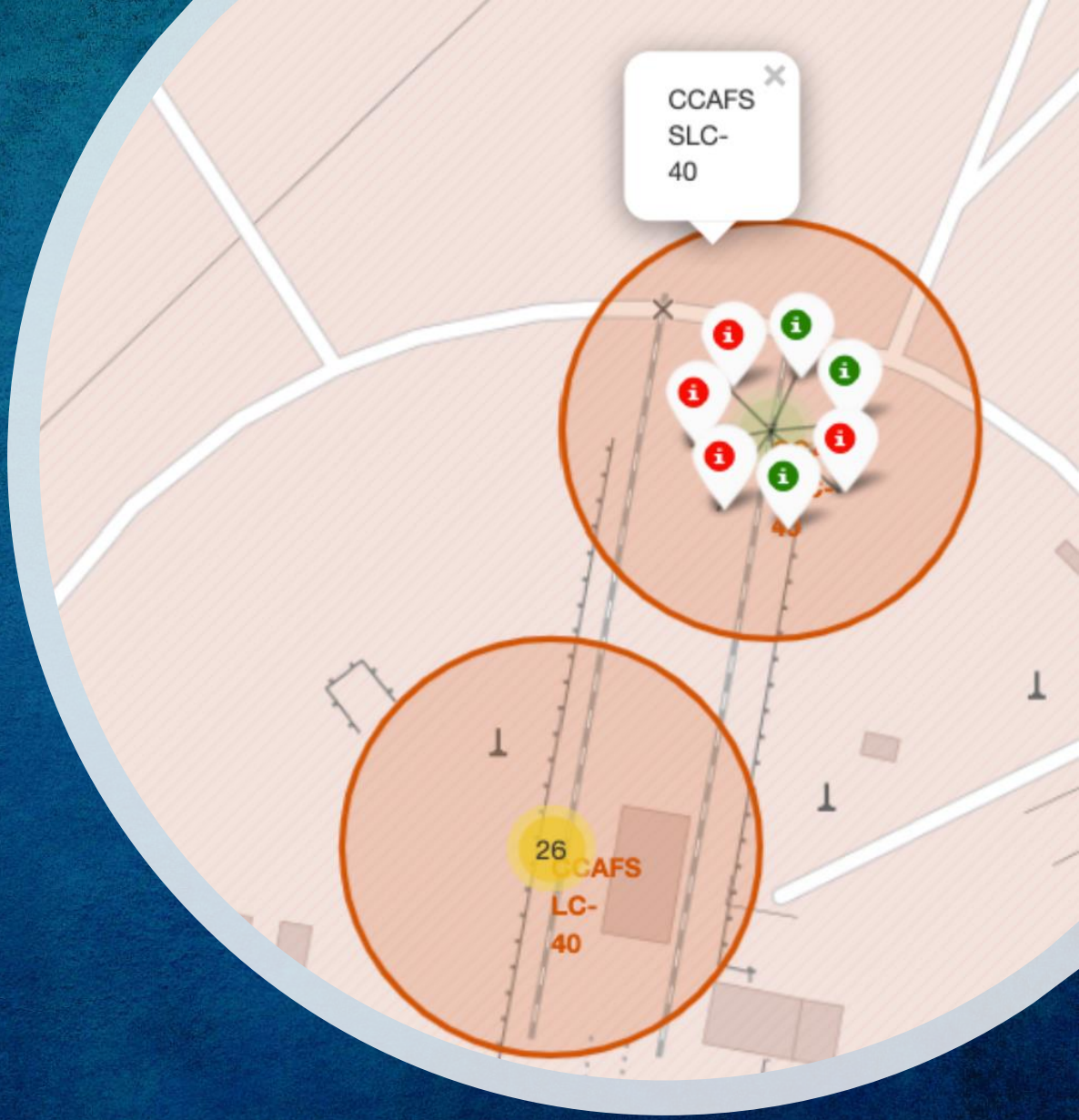
- Marked the Launch sites into an interactive map using longitude and latitude coordinates of the sites.
- Added circle marker with label name of the launch site.
- We assigned the feature launch_outcomes (failure or success) to class 0 and 1.i.e., 0 for failure, and 1 for success and makers ‘red’ and ‘green’ on the map respectively for the outcomes using MarkerCluster().
- Using the color-labeled marker clusters, we identified which launch sites have relatively high success rate.
- We calculated the distance from the Launch Site to its proximities. Lines are drawn on the map to measure distance to landmarks.
- Some insights on Launch Sites:

✓ Are launch sites in close proximity to railways? No

✓ Are launch sites in close proximity to highways? No

✓ Are launch sites in close proximity to coastline? Yes

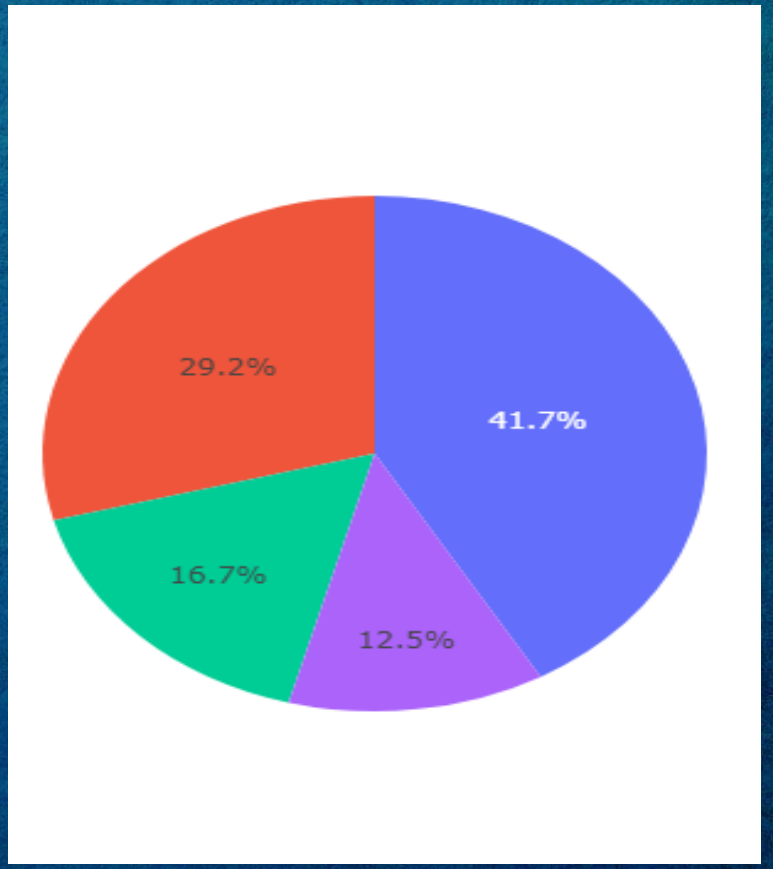
✓ Do launch sites keep certain distance away from cities? Yes



GIT LINK: https://github.com/stuti2907/IBM-data-Science/blob/master/Launch_site_analysis_with_folium.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash having pie chart and scatter plot.
- Dashboard is built to have an option to select a Launch site.
- Success Pie charts showing the total launches by a certain sites.
- Display relative proportions of success and failure of Launches from a Launch site.
- Scatter Graph showing the relationship with Outcome and Payload Mass (Kg) for the different Booster Versions.
- Scatter plots are used to show a non-linear pattern and we can determine the range of data flow i.e. maximum, minimum.



GIT LINK:

https://github.com/stuti2907/IBM-data-Science/blob/master/spacex_dash_app.py

Predictive Analysis (Classification)

- Load the data using NumPy and Pandas.
- Transformed the data and split it into Test and Train datasets.
- Check the number of samples in test datasets.

- Build different Machine Learning models.
- Perform hyperparameter tunning to find the pest parameters using GridSearchCV.

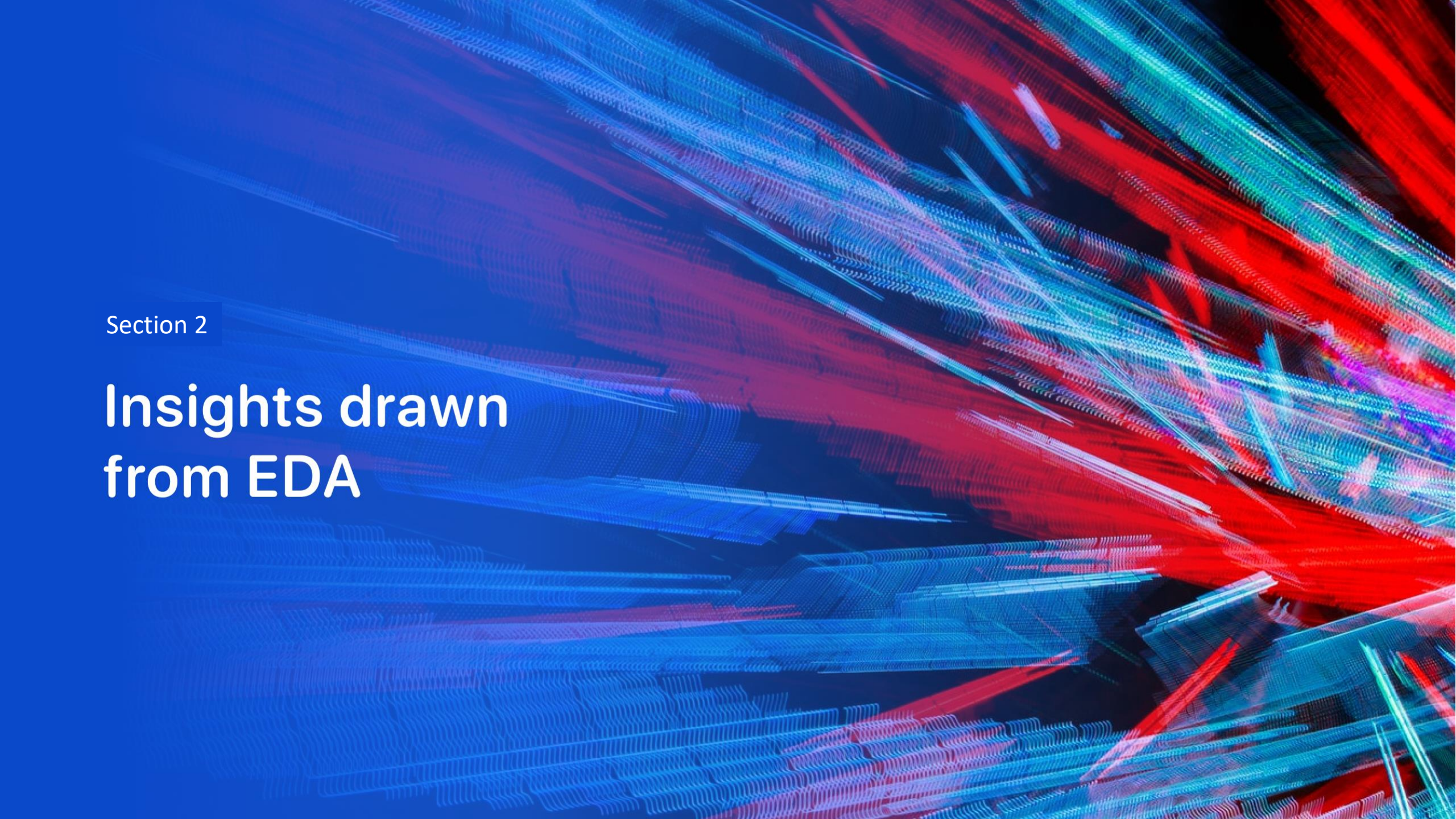
- Used Accuracy as the metric for the model.
- Improved model using feature engineering and algorithm tuning.

- The model with the best accuracy score wins the best performing model.

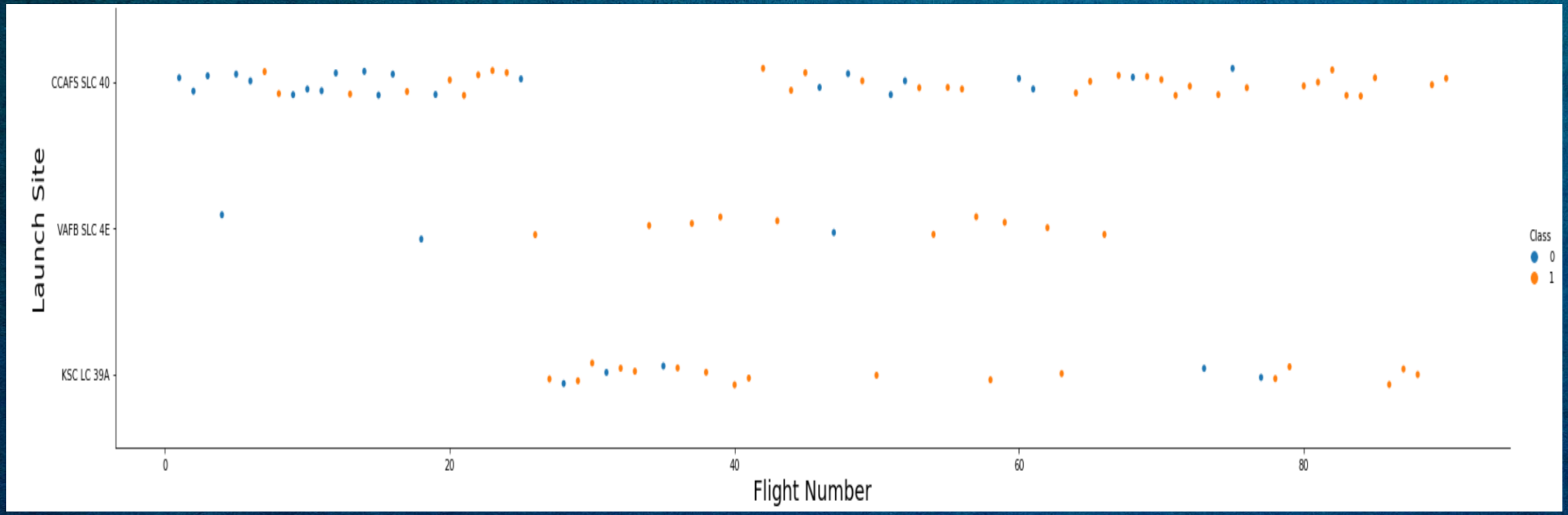
GIT LINK :
https://github.com/stuti2907/IBM-data-Science/blob/master/Machine_learing_prediction.ipynb

Results

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

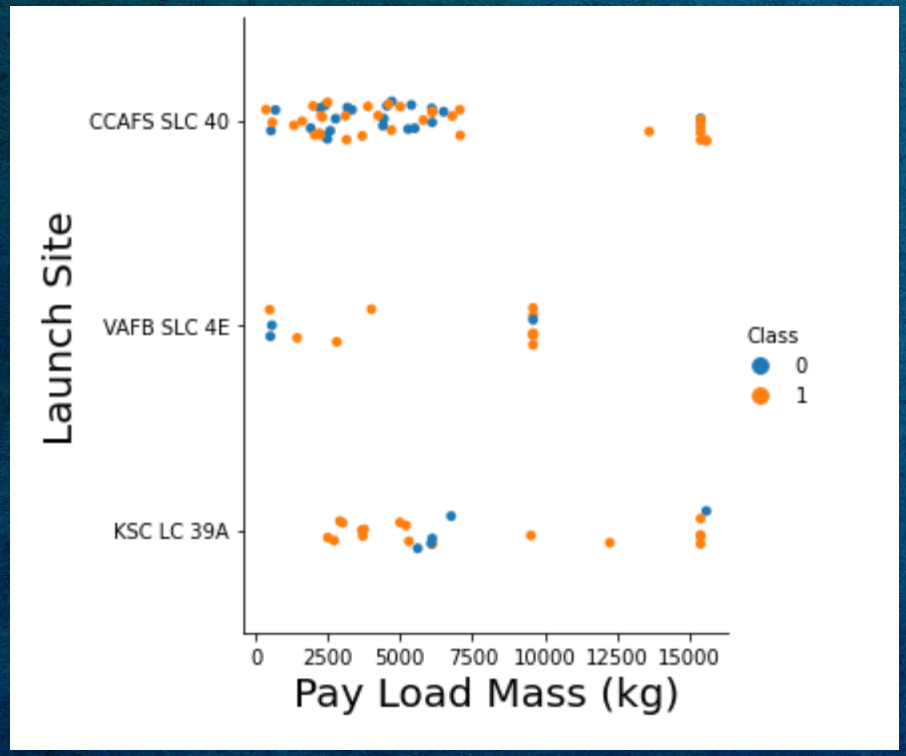

Section 2
Insights drawn from EDA

Flight Number vs. Launch Site



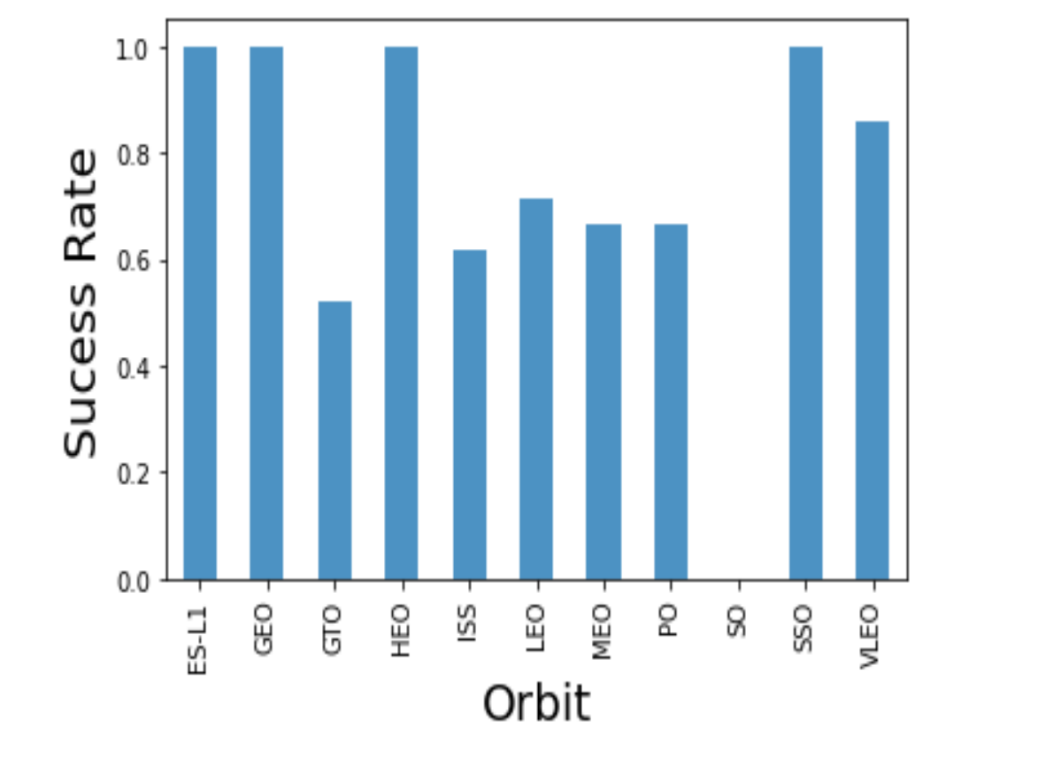
From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

Payload vs. Launch Site



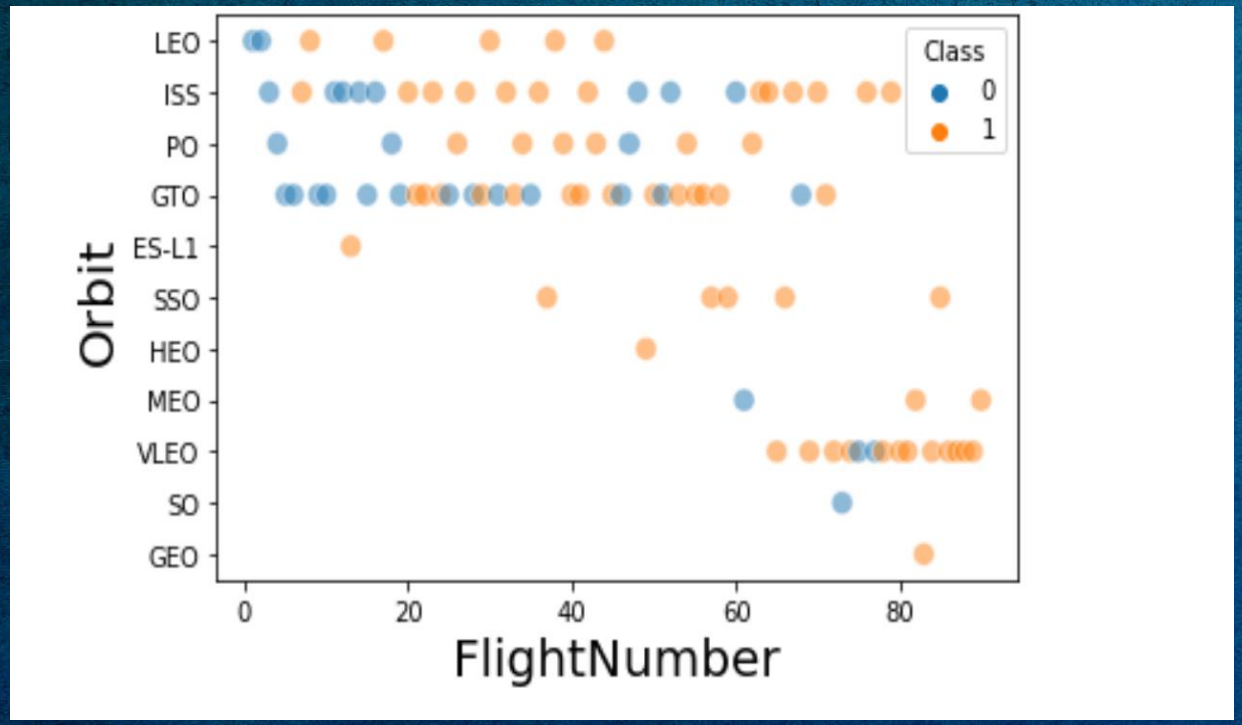
From the plot, we found that the greater the payload mass for Launch Site CCAFS SLC 40 the higher the success rate for the Rocket. However, for other two site we can see some failure mission even when the payload mass is more, but the success rate is more for all the 3 sites for higher payload mass.

Success Rate vs. Orbit Type



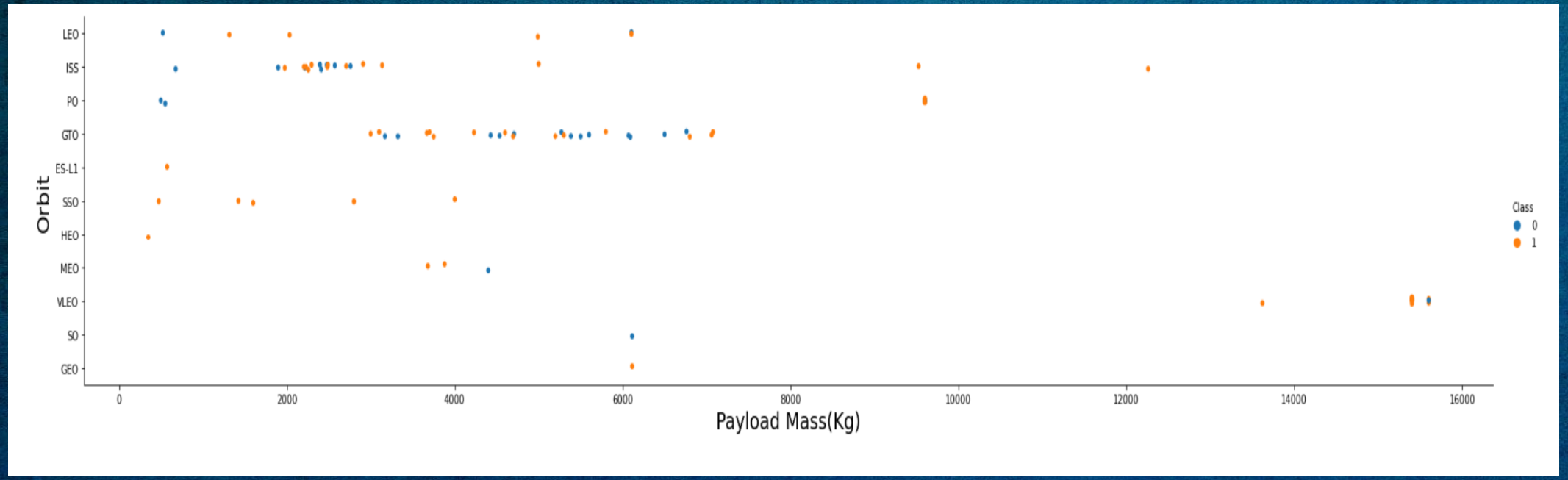
From the plot, we found that the Orbit ES-L1, GEO, HEO and SSO has the best Success Rate.

Flight Number vs. Orbit Type



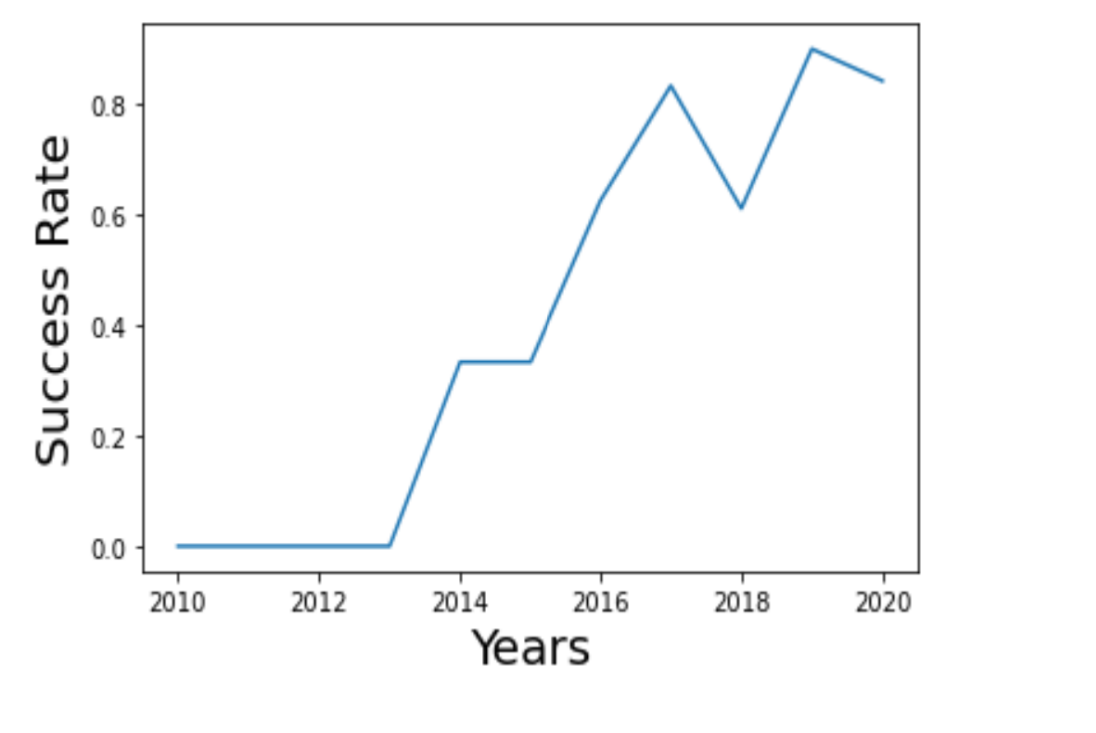
From the plot, we found that in LEO orbit, number of flights and successful landings are positively related. While other orbits like ISS, GTO are not showing any relationship between flight numbers and success of the landing.

Payload vs. Orbit Type



From the plot, we can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits. While for SSO, all the landings are successful irrespective of payload mass.

Launch Success Yearly Trend



From the plot, we can observe that the success rate started increasing from 2013 till 2017,with a slight decrease in 2017 and again increased from 2018 till 2020 with slight changes in success rate.

All Launch Site Names

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

```
%sql select distinct(LAUNCH_SITE) from SPACEXTBL;
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

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

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

Launch Site Names Begin with 'CCA'

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

```
%sql SELECT * from SPACEXTBL where LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

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	07:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525	LEO (ISS)	NASA (COTS)	Success	No attempt
2012-10-08	00:35:00	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

Using the word LIMIT 5 in the query means that it will only show 5 records from SPACEXTBL and LIKE keyword has a wild card with the words ‘CCA%’ the percentage in the end suggests that the LAUNCH_SITE name must start with CCA.

Total Payload Mass

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

```
%sql select sum(PAYLOAD_MASS__KG_) as total_payloadmass from SPACEXTBL where customer = 'NASA (CRS)';
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

total_payloadmass
45596

Using the function SUM summates the total in the column PAYLOAD_MASS_KG_

The WHERE clause filters the dataset to only perform calculations on Customer equal to NASA (CRS)

Average Payload Mass by F9 v1.1

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

```
%sql select avg(PAYLOAD_MASS__KG_) from SPACEXTBL where BOOSTER_VERSION = 'F9 v1.1'
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

1
2928

Using the function AVG calculates the average in the column PAYLOAD_MASS_KG_

The WHERE clause filters the dataset to only perform calculations on Booster_version = F9 v1.1

First Successful Ground Landing Date

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

Hint:Use min function

```
%sql select min(DATE) from SPACEXTBL where Landing__Outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

1
2015-12-22

Using the function MIN calculates the minimum date in the column DATE.

The WHERE clause filters the dataset to only perform calculations on Landing__Outcome = Success (ground pad)

Successful Drone Ship Landing with Payload between 4000 and 6000

```
%sql select BOOSTER_VERSION from SPACEXTBL where Landing__Outcome = 'Success (drone ship)' and PAYLOAD_MASS__KG_ > 4000 and PAYLOAD_MASS__KG_ < 6000
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

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

Using 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

Total Number of Successful and Failure Mission Outcomes

List the total number of successful and failure mission outcomes

```
%sql SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) AS OUTCOME FROM SPACEXTBL GROUP BY MISSION_OUTCOME
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

mission_outcome	outcome
Failure (in flight)	1
Success	99
Success (payload status unclear)	1

Using the COUNT function to calculate the number of mission and GROUP BY to group the mission outcomes so as to get the total number of mission per mission outcome.

Boosters Carried Maximum Payload

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

```
%sql select BOOSTER_VERSION from SPACEXTBL where PAYLOAD_MASS__KG_ = (select max(PAYLOAD_MASS__KG_) from SPACEXTBL)
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

25]:

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

2015 Launch Records

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

```
%sql select landing__outcome,booster_version,launch_site from SPACEXTBL where extract(year from DATE) = 2015 and landing__outcome = 'Failure (drone ship)'
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

39]:

landing__outcome	booster_version	launch_site
Failure (drone ship)	F9 v1.1 B1012	CCAFS LC-40
Failure (drone ship)	F9 v1.1 B1015	CCAFS LC-40

We used WHERE clause to get landing__outcomes = ‘Failure (drone ship)’ and EXTRACT function to extract year from DATE

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 landing__outcome,count(landing__outcome) as count from SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' group by landing__outcome order by count desc
```

```
 * ibm_db_sa://fgg10313:***@19af6446-6171-4641-8aba-9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud:30699/bludb
Done.
```

2]:

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

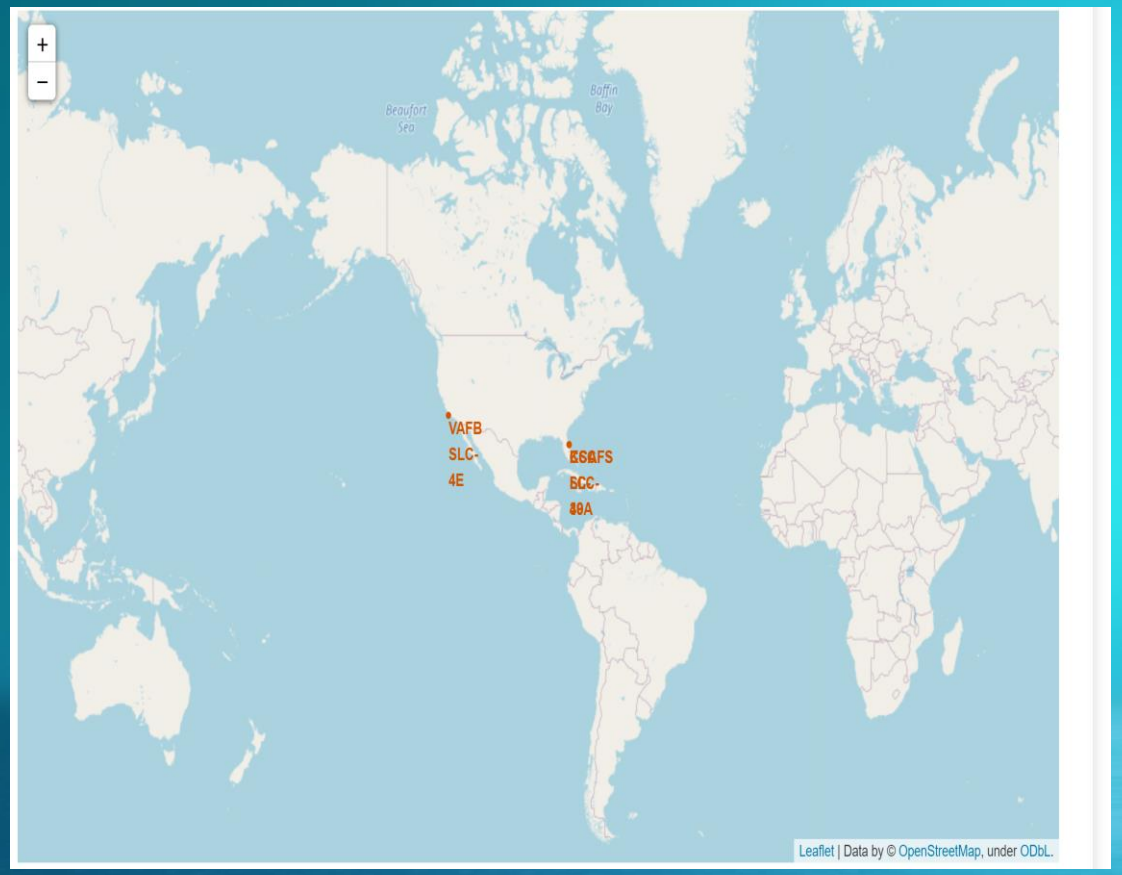
We selected Landing outcomes and the **COUNT** of landing outcomes and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.

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.

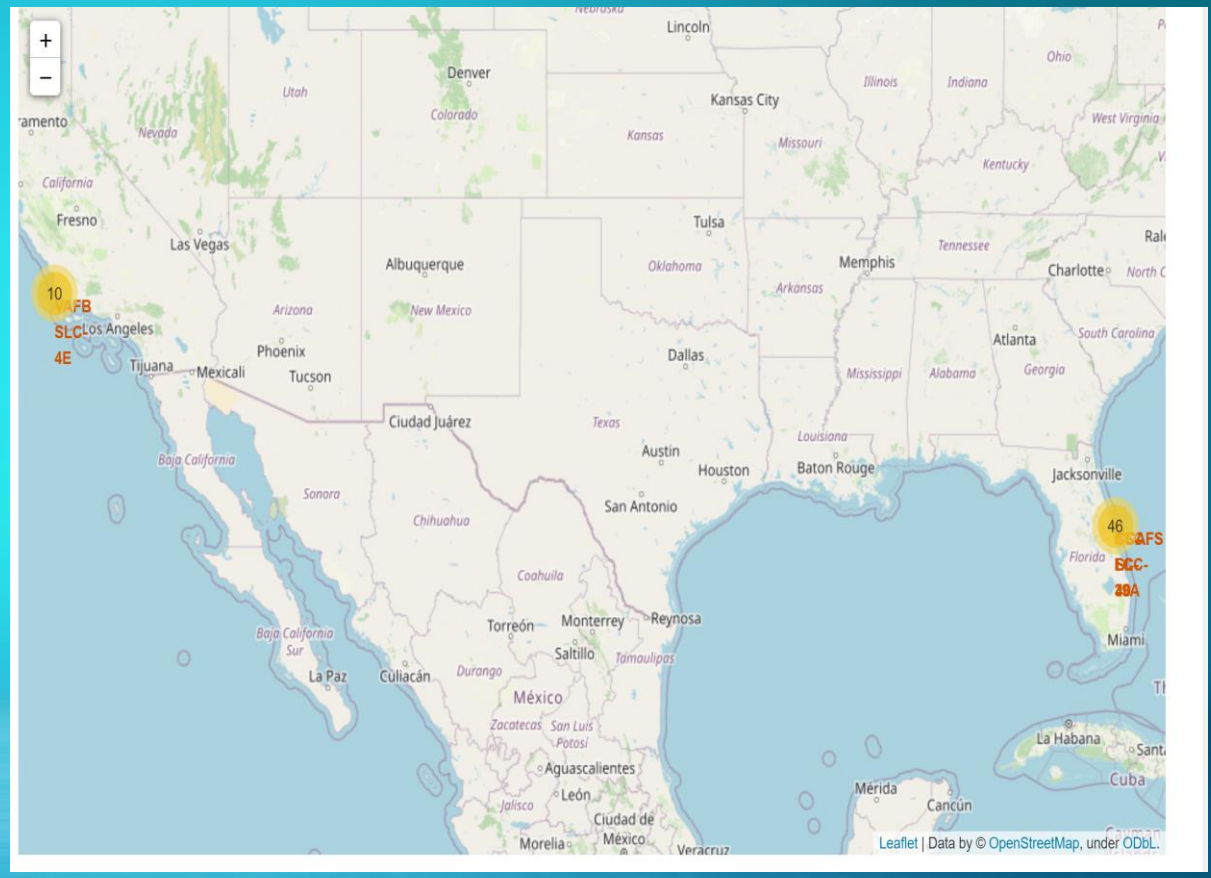
Section 3

Launch Sites Proximities Analysis

Launch site marking on Global Map



Zoom -Out

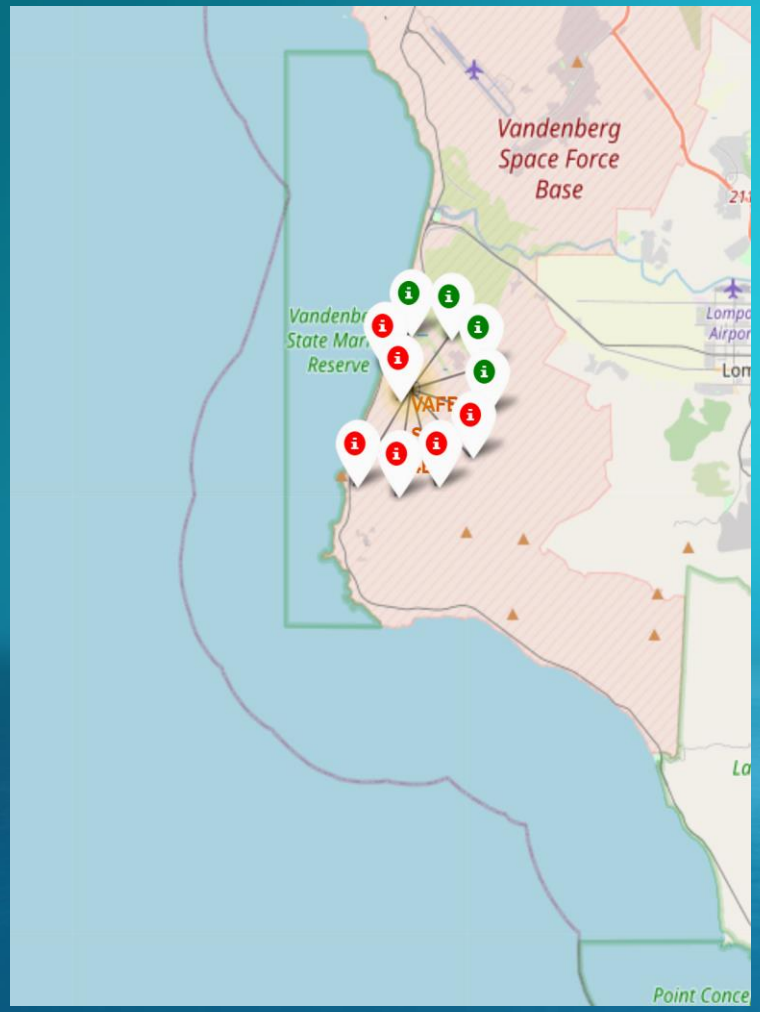


Zoom -In

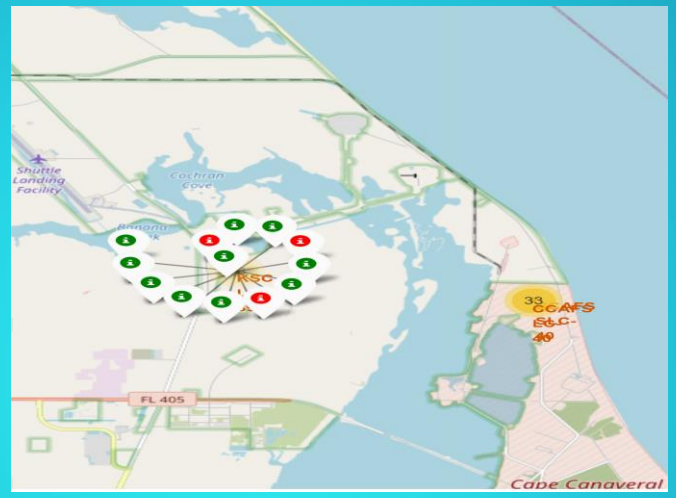
SPACEX launch sites are in USA coastline near California and Florida

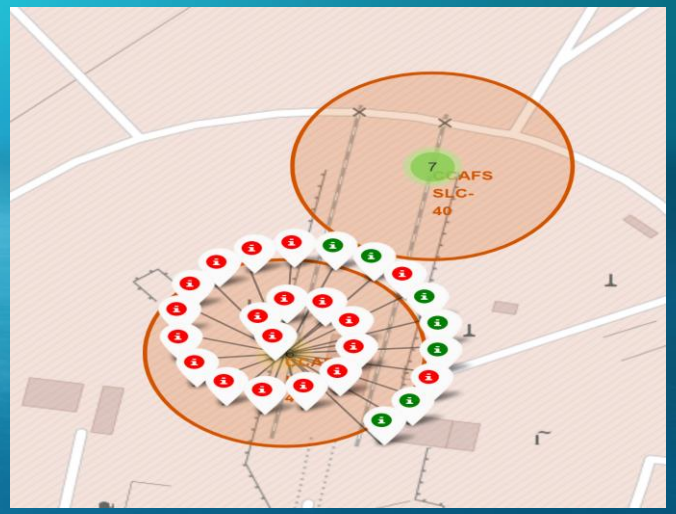
Successful and Failed Launches in each site

Green Marker for successful Launches and Red Markers for Failed



California Launch Site

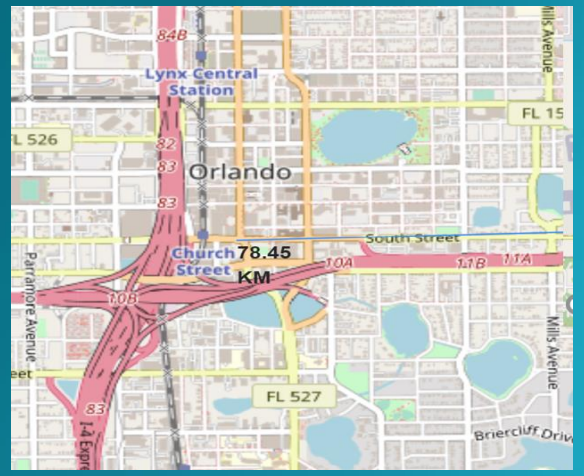




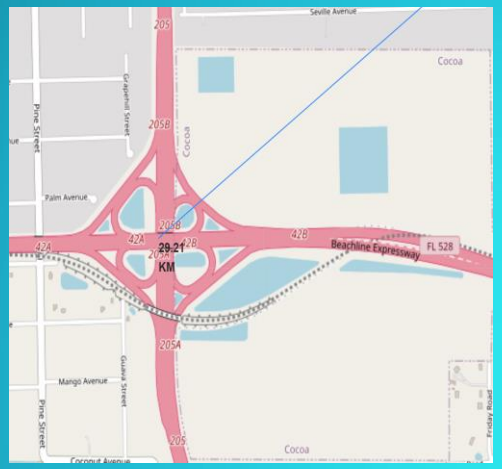
Florida Launch Site

Distance between Launch Site and its proximities

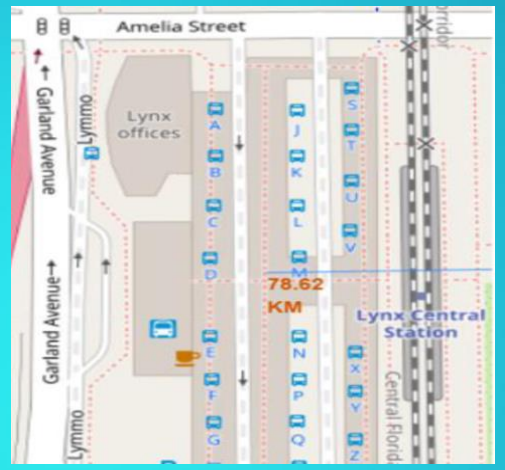
Using CCAFS-SLC-40 as a reference Launch Site



Distance to nearest city



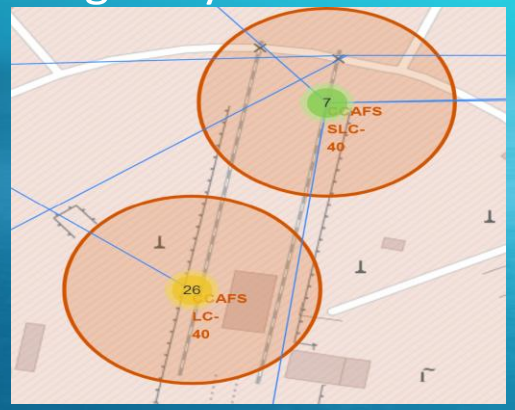
Distance to nearest highway



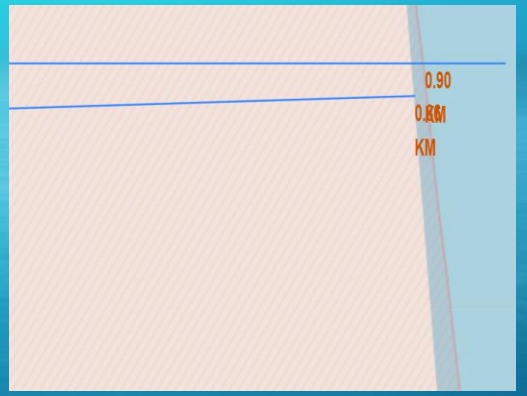
Distance to railway station

- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

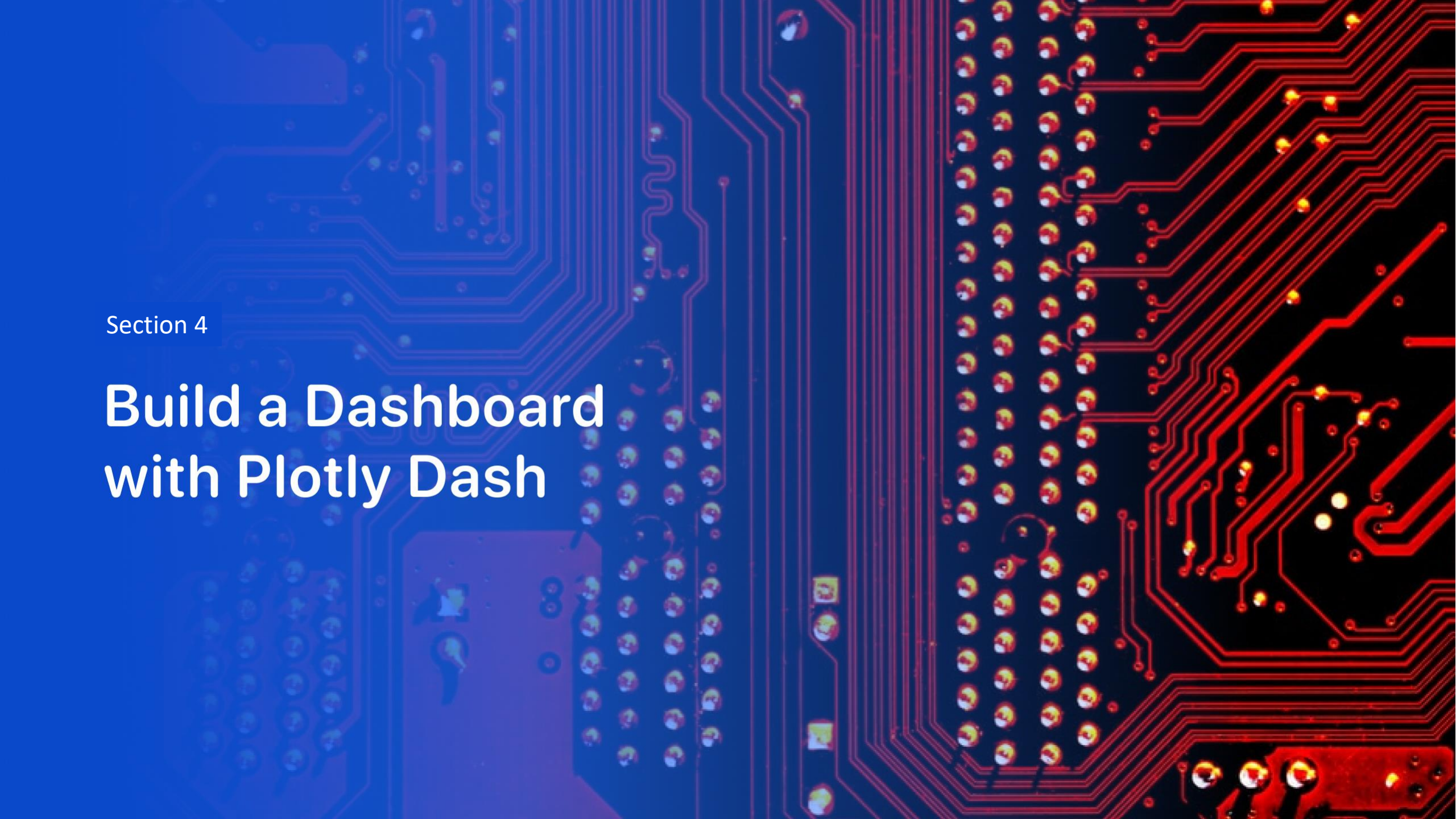
Distance to nearest airport



Distance to nearest coast

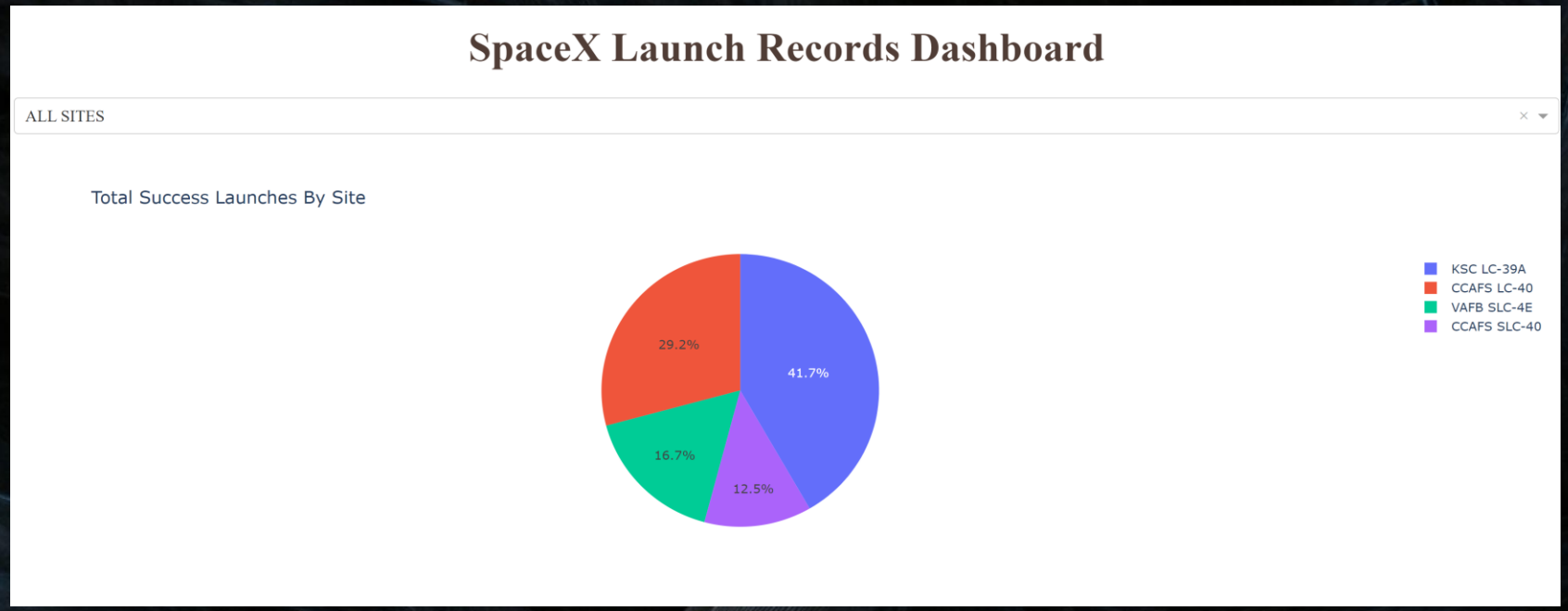


Distance to nearest coastline


Section 4
Build a Dashboard with Plotly Dash

Success Pie Chart – Showing data for all Launch sites

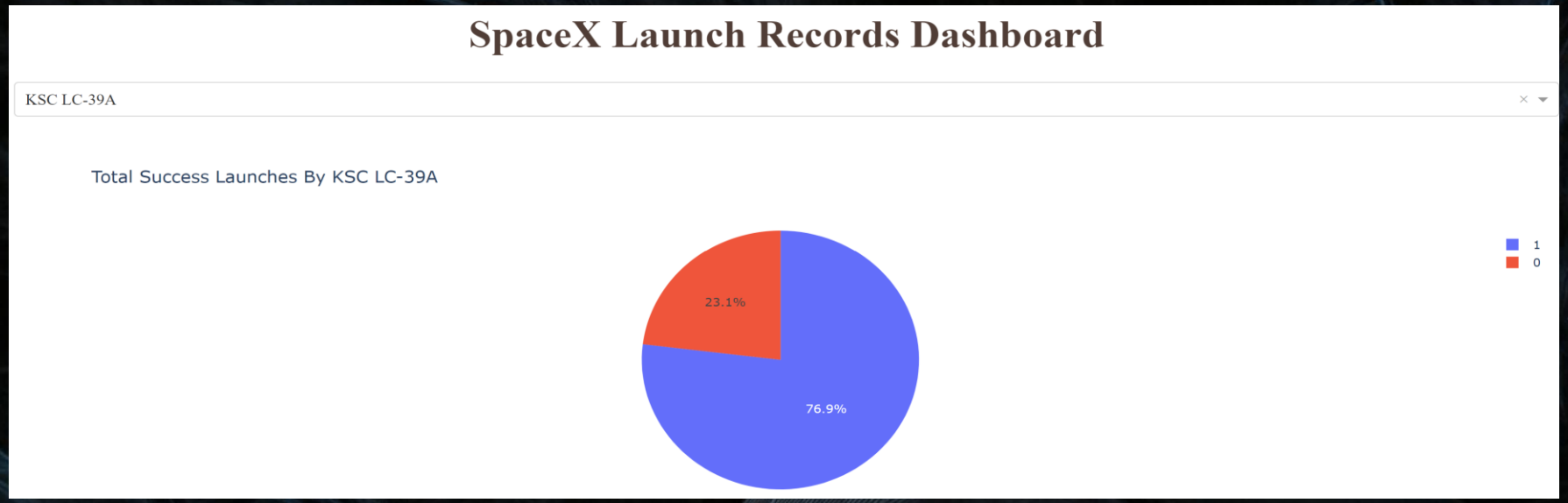
Pie chart showing the success percentage achieved by each launch site.



From the chart, we can find that KSC LC-39A had the highest number of successful Launches.

Success Pie Chart – Showing data for a particular launch site

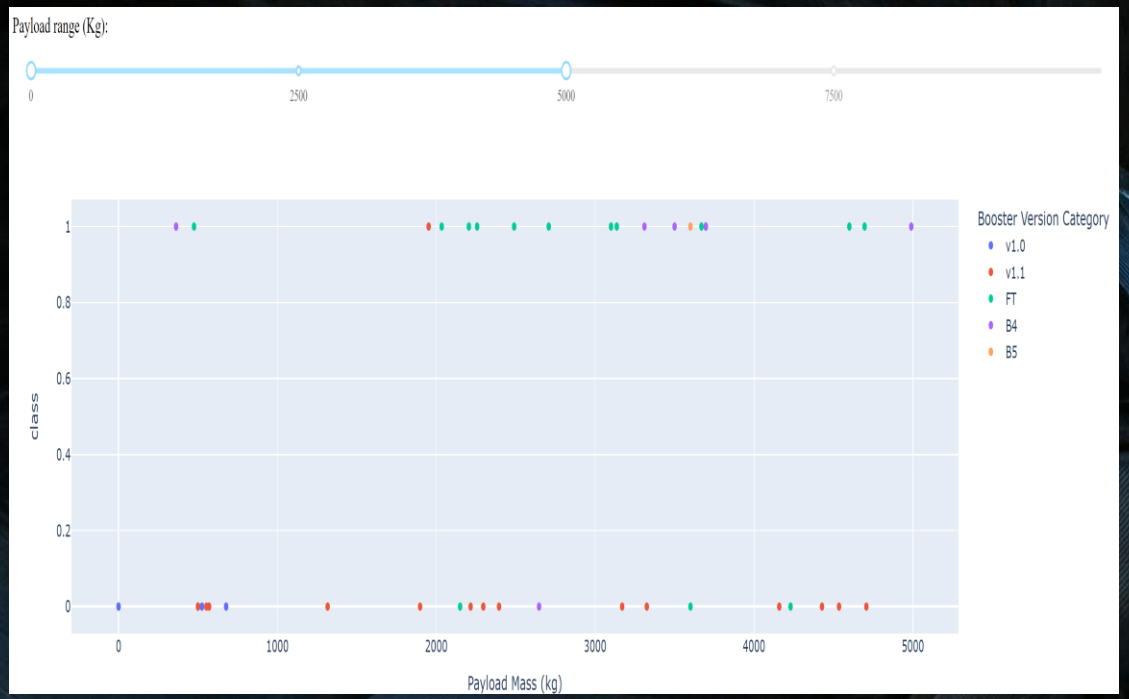
Pie chart showing the Launch site with the highest launch success ratio



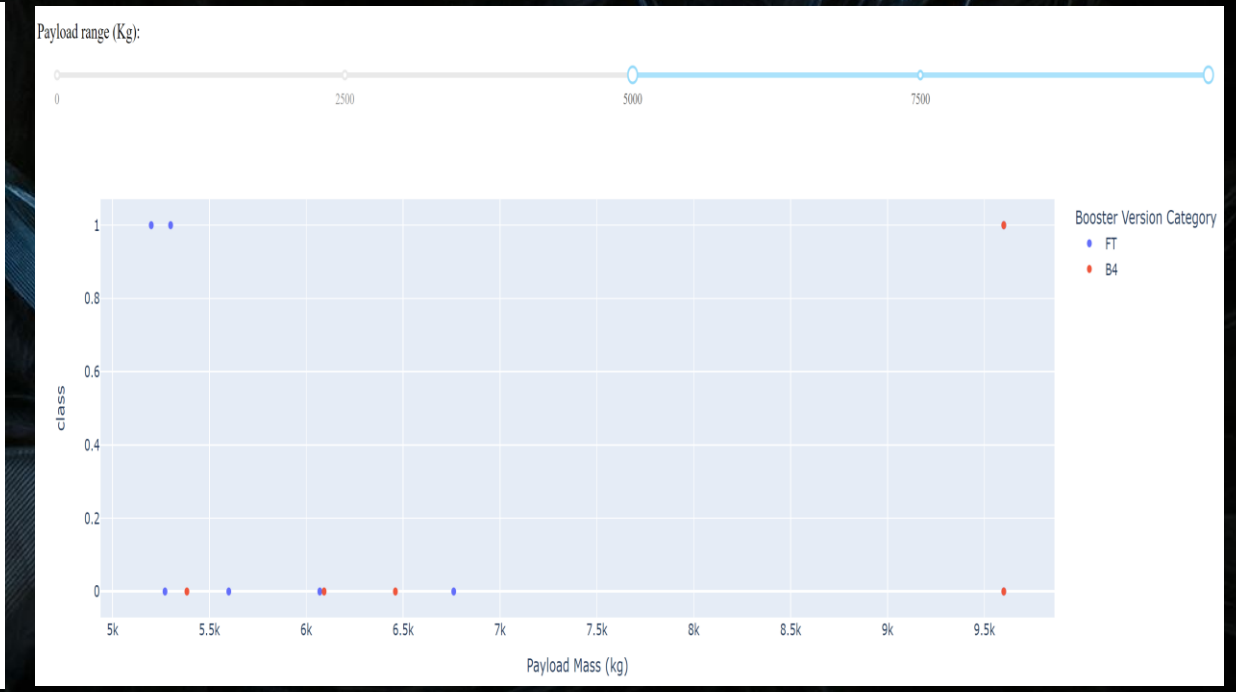
From the chart, we can see that site KSC LC-39A had a success rate of 76.9% with a failure rate of 23.1%.

Scatter Plot – Payload vs Launch Outcome

Payload vs. Launch Outcome scatter plot for all sites, with different payload selected in the range slider



Payload Mass Range – 0-5000 Kg



Payload Mass Range – 5000 - 10000 Kg

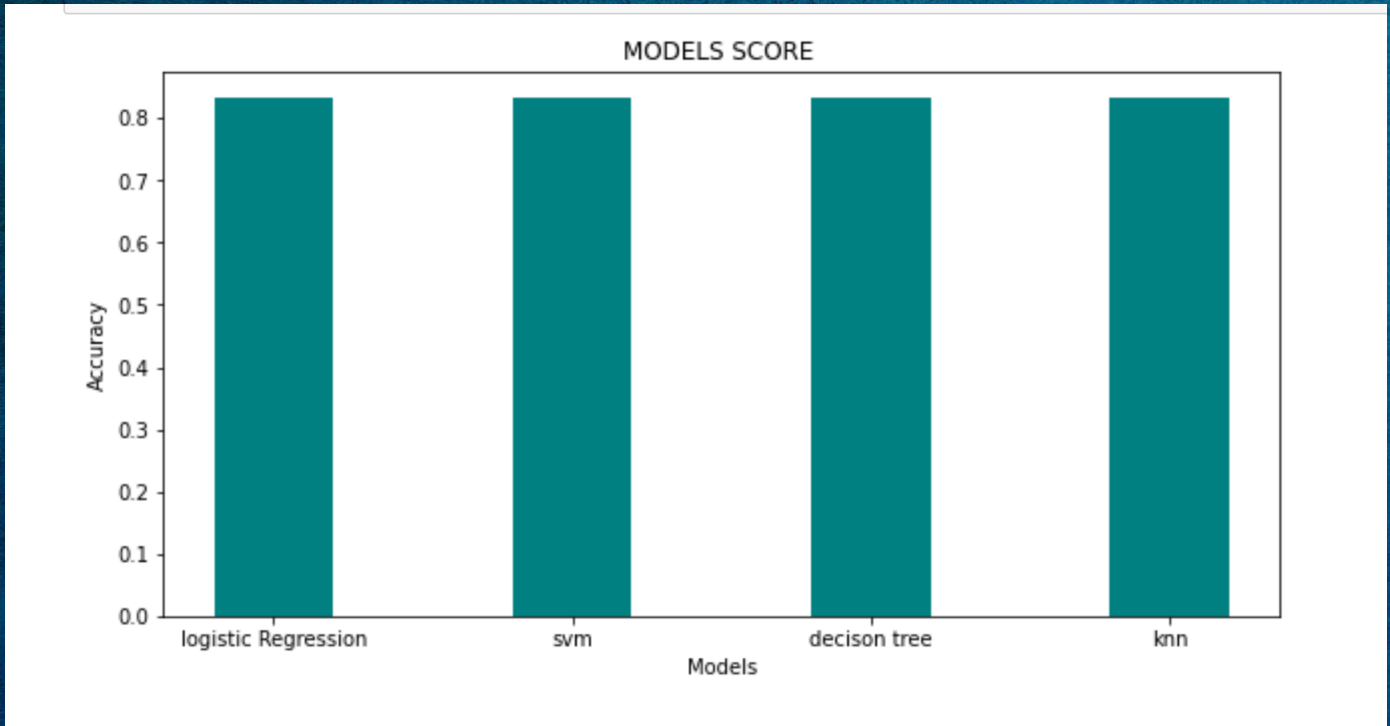
We can see that lower payload mass has more success rate. Also, the booster version FT has the highest number of successful launches.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

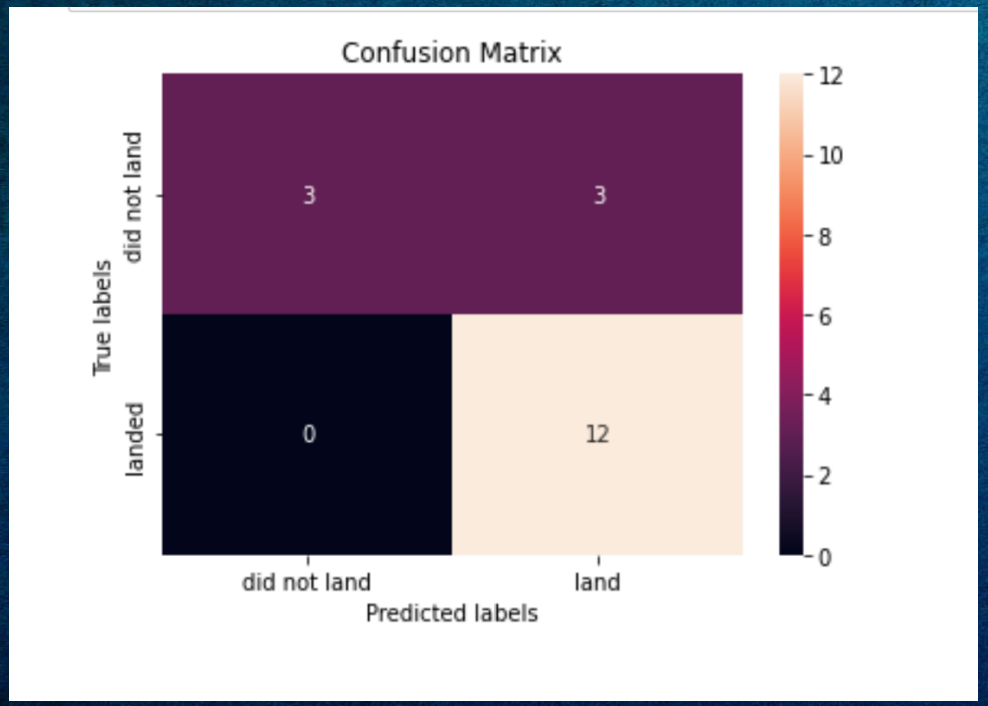
From the bar chart, although the accuracy for all models looks almost same but decision tree had slightly higher accuracy in decimal points.



The accuracy score of best models was around 83.34

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 had the most success rate.
- KSC LC-39A had the most successful launches of any sites.
- Low weighted payloads perform better than the heavier payloads.
- Most of the launch sites are near Equator and closer to the coastline.
- The Decision tree classifier is the best machine learning algorithm for this task.

Appendix

✓ Built advanced ML model using Light GBM

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

✓ Performed hyperparameter tunning on Light GBM using hyperopt module.

Hyperopt uses a form of Bayesian optimization for parameter tuning that allows you to get the best parameters for a given model. It can optimize a model with hundreds of parameters on a large scale.

Appendix

```
space = {
            'n_estimators': hp.quniform('n_estimators', 500,1200,50),
            'num_leaves': hp.quniform('num_leaves', 3,50,1),
            'colsample_bytree': hp.quniform('colsample_bytree', 0.19, 0.35, 0.01),
            'min_child_samples': hp.quniform('min_child_samples', 1, 100,1),
            'boosting_type':hp.choice('boosting_type',['gbdt', 'dart', 'goss']),
             'subsample': hp.quniform('subsample', 0.1, 0.95, 0.05),

            }

def objective(params):
    params = {
        'n_estimators':int(params[ 'n_estimators']),
        'num_leaves': int(params[ 'num_leaves']),
        'colsample_bytree': params[ 'colsample_bytree'],
        'min_child_samples':int(params[ 'min_child_samples']) ,
        'subsample':params['subsample'],
        'boosting_type': params['boosting_type']
    }

    clf = LGBMClassifier(objective  = 'binary',
                         class_weight ='balanced',
                         tree_method ='hist',
                         max_depth = -1,
                         n_jobs = -1,
                         min_split_gain=0.2,max_bin=250,
                         random_state = 27,**params)
```

Appendix

```
    clf.fit(X_train, Y_train)
    score_lgbm_best = clf.best_score_


    print("score_lgbm_best:", score_lgbm_best)

    return{'loss':1-score_lgbm_best, 'status': STATUS_OK }


 trials = Trials()
best = fmin(fn=objective,
            space=space,
            algo=tpe.suggest,
            max_evals=10,trials=trials)

print(best)
```

Thank you!