IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

LOVISH GARLANI

June 17, 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
  - Screenshots: Interactive analytics
  - 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. **The project aims at creating a machine-learning model 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 are required to ensure a successful landing program?

Section 1

Methodology

Methodology

Executive Summary

- Data collection methodology:
  - Data was collected **using SpaceX API** followed by **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 the following 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 converted it to a pandas data frame using .json_normalize().
  - We then cleaned the data, checked for missing values, and filled 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 an HTML table, parse the table, and converted it to pandas data frame 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 link to the notebook is https://github.com/lovishgarlani/IBM-Applied-Data-Science-Capstone-SpaceX-Lovish-Garlani/blob/main/02%20Spacex-data-collection-api.ipynb

SpaceX API Calls: FlowChart

Step 1: Requesting rocket launch data from SpaceX API with the following URL:

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

Step 2: To make the requested JSON results more consistent, we will use the following static response object for this project:

```python
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/
```

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

```python
# Use json_normalize meethod to convert the json result into a dataframe
data = pd.json_normalize(response.json())
# Get the head of the dataframe
data.head()
```

Spaces: 4 CRLF Cell 8 of 8

The output of the SpaceX API calls representing the data extracted

```python
# Get the head of the dataframe
data.head()
```

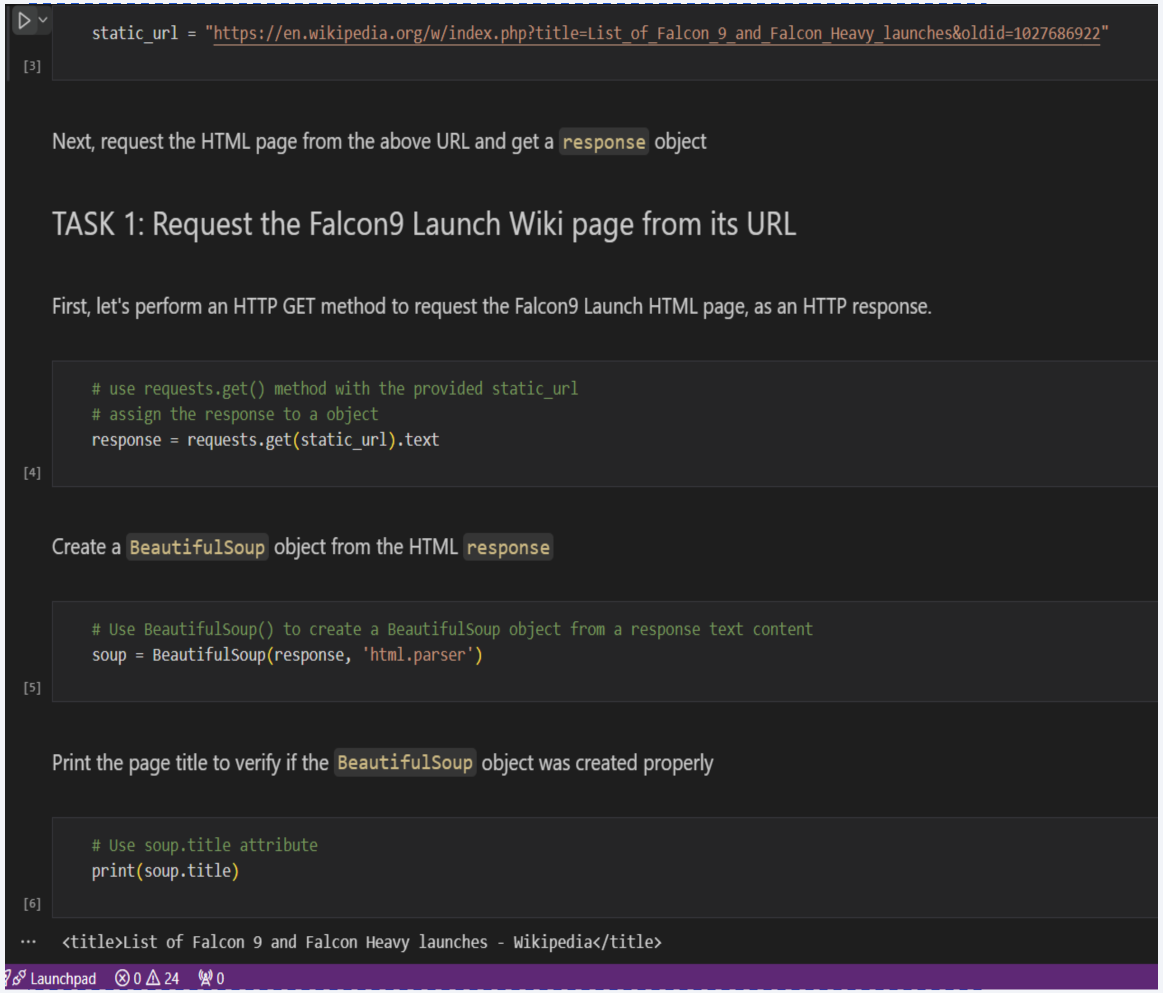
[12] Python

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew	ships	capsules	payloads	launchpad
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	[]	[]	[]	[5eb0e4b5b6c3bb0006eeb1e1]	5e9e4502f5090995de566f86
1	None	NaN	False	0.0	5e9d0d95eda69955f709d1eb	False	[{'time': 301, 'altitude': 289, 'reason': 'harmonic oscillation leading to premature engine shutdown'}]	Successful first stage burn and transition to second stage, maximum altitude 289 km, Premature engine shutdown at T+7 min 30 s, Failed to reach orbit, Failed to recover first stage	[]	[]	[]	[5eb0e4b6b6c3bb0006eeb1e2]	5e9e4502f5090995de566f86
							[{'time': 140, 'altitude': 35,	Residual stage 1					

Launchpad ⊗ 0 ⚠ 21 0 Spaces: 4 Cell 38 of 82

Data Collection - Scraping

- We applied web scrapping to scrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is https://github.com/lovishgarlani/IBM-Applied-Data-Science-Capstone-SpaceX-Lovish-Garlani/blob/main/03%20Data%20Collection%20with%20Web%20Scraping%20lab.ipynb



```
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.

```
# use requests.get() method with the provided static_url
# assign the response to a object
response = requests.get(static_url).text
```

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

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

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

```
# Use soup.title attribute
print(soup.title)
```

```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

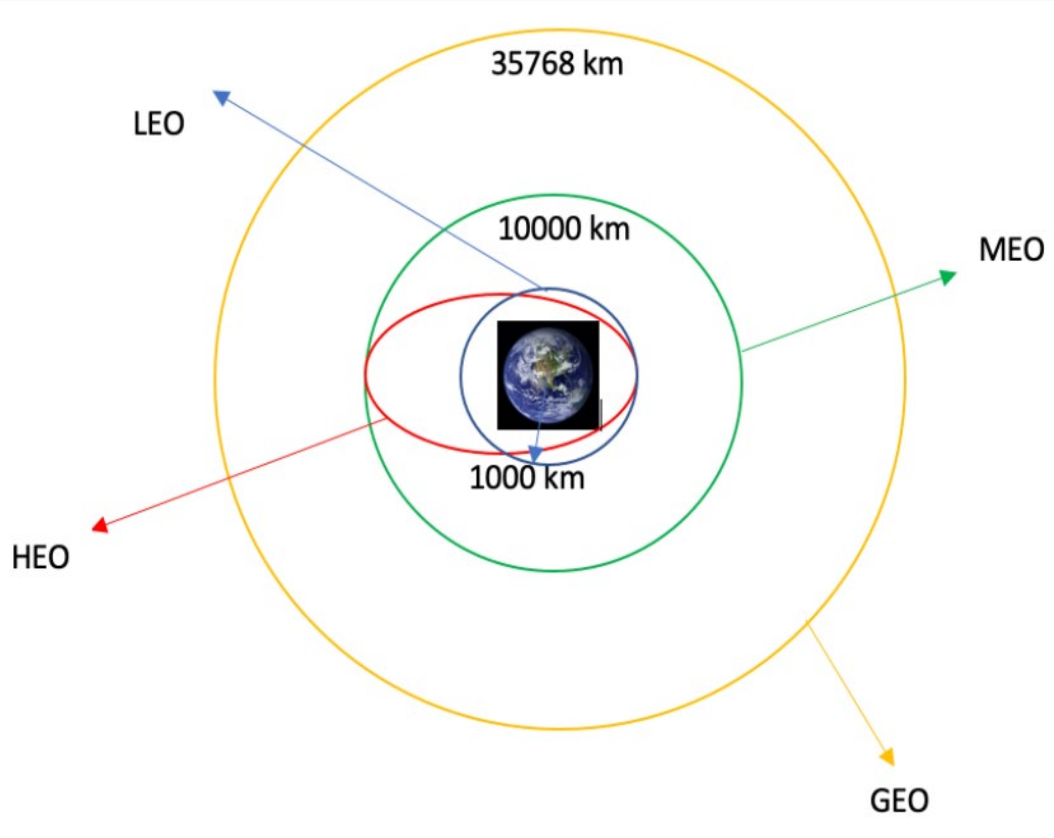
- Next, we Extracted all column/variable names from the HTML table header.

- Then we created a data frame by parsing the launch HTML tables.

- In the end, we removed the NaN values from the columns and extracted the file as csv using the df.to_csv method.

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 a landing outcome label from the outcome column and exported the results to CSV.
- The link to the notebook is https://github.com/lovishgarlani/IBM-Applied-Data-Science-Capstone-SpaceX-Lovish-Garlani/blob/main/04%20SpaceX%20Data%20Wrangling.ipynb



Data Wrangling FLOWCHART

```python
# TASK 1: Calculate the number of launches on each site
# Apply value_counts() on column LaunchSite
df['LaunchSite'].value_counts()

# TASK 2: Calculate the number and occurrence of each orbit
# Apply value_counts on Orbit column
df['Orbit'].value_counts()

# TASK 3: Calculate the number and occurence of mission outcome of the orbits
landing_outcomes = df['Outcome'].value_counts()

# TASK 4: Create a landing outcome label from Outcome column
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []

for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)

# This variable will represent the classification variable that represents the
# outcome of each launch. If the value is zero, the  first stage did not land
# successfully; one means  the first stage landed Successfully

df['Class']=landing_class
df.head()
```

[ ] Pyth

Spaces: 4 CRLF Cell 10 of 11

Data Wrangling Outcome

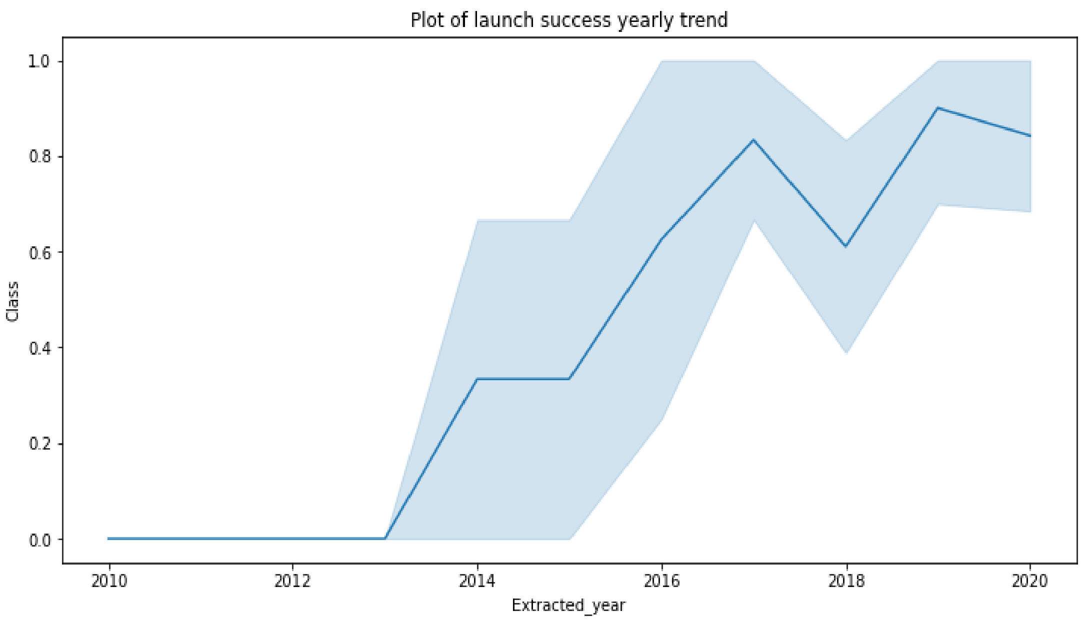
```python
df.head(10)
```

[23] Python

	FlightNumber	Date	BoosterVersion	PayloadMass	Orbit	LaunchSite	Outcome	Flights
0	1	2010-06-04	Falcon 9	6104.959412	LEO	CCAFS SLC 40	None None	1
1	2	2012-05-22	Falcon 9	525.000000	LEO	CCAFS SLC 40	None None	1
2	3	2013-03-01	Falcon 9	677.000000	ISS	CCAFS SLC 40	None None	1
3	4	2013-09-29	Falcon 9	500.000000	PO	VAFB SLC 4E	False Ocean	1
4	5	2013-12-03	Falcon 9	3170.000000	GTO	CCAFS SLC 40	None None	1
5	6	2014-01-06	Falcon 9	3325.000000	GTO	CCAFS SLC 40	None None	1
6	7	2014-04-18	Falcon 9	2296.000000	ISS	CCAFS SLC 40	True Ocean	1
7	8	2014-07-14	Falcon 9	1316.000000	LEO	CCAFS SLC 40	True Ocean	1
8	9	2014-08-05	Falcon 9	4535.000000	GTO	CCAFS SLC 40	None None	1
9	10	2014-09-07	Falcon 9	4428.000000	GTO	CCAFS SLC 40	None None	1

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 link to the notebook is https://github.com/lovishgarlani/IBM-Applied-Data-Science-Capstone-SpaceX-Lovish-Garlani/blob/main/06%20EDA%20with%20Visualization%20Lab.ipynb

EDA with SQL

- We loaded the SpaceX dataset into a PostgreSQL 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 failed mission outcomes
  - The failed landing outcomes in drone ship, their booster version, and launch site names.
- The link to the notebook is https://github.com/lovishgarlani/IBM-Applied-Data-Science-Capstone-SpaceX-Lovish-Garlani/blob/main/05%20EDA%20with%20SQL.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 link of the notebook is https://github.com/lovishgarlani/IBM-Applied-Data-Science-Capstone-SpaceX-Lovish-Garlani/blob/main/07%20Interactive%20Visual%20Analytics%20with%20Folium%20Lab.ipynb

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pie charts showing the total launches by certain sites
- We plotted a scatter graph showing the relationship between Outcome and Payload Mass (Kg) for the different booster versions.
- The link to the notebook is: https://github.com/lovishgarlani/IBM-Applied-Data-Science-Capstone-SpaceX-Lovish-Garlani/blob/main/08%20SpaceXDashboard%20App.ipynb

Predictive Analysis (Classification)

- We loaded the data using Numpy and pandas, transformed the data, and split our data into training and testing.
- We built different machine-learning models and tuned different hyperparameters using GridSearchCV.
- We used accuracy as the metric for our model, and improved the model using feature engineering and algorithm tuning.
- We found the best-performing classification model.
- The link to the notebook is https://github.com/lovishgarlani/IBM-Applied-Data-Science-Capstone-SpaceX-Lovish-Garlani/blob/main/09%20SpaceX%20Machine%20Learning%20Prediction.ipynb

Results

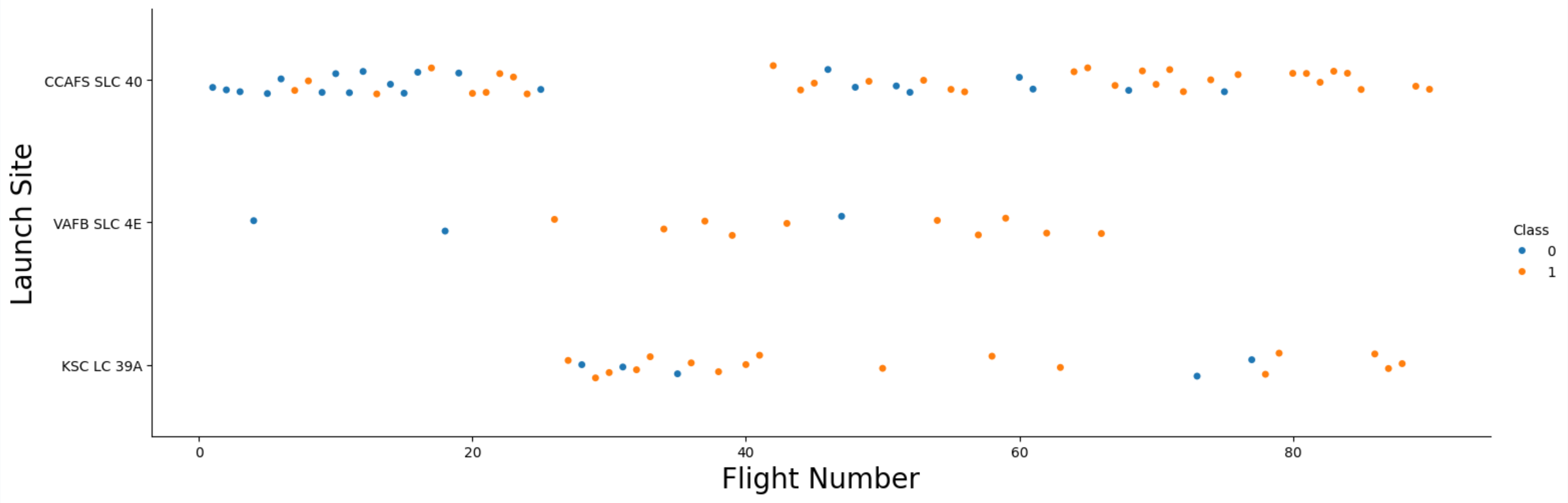
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

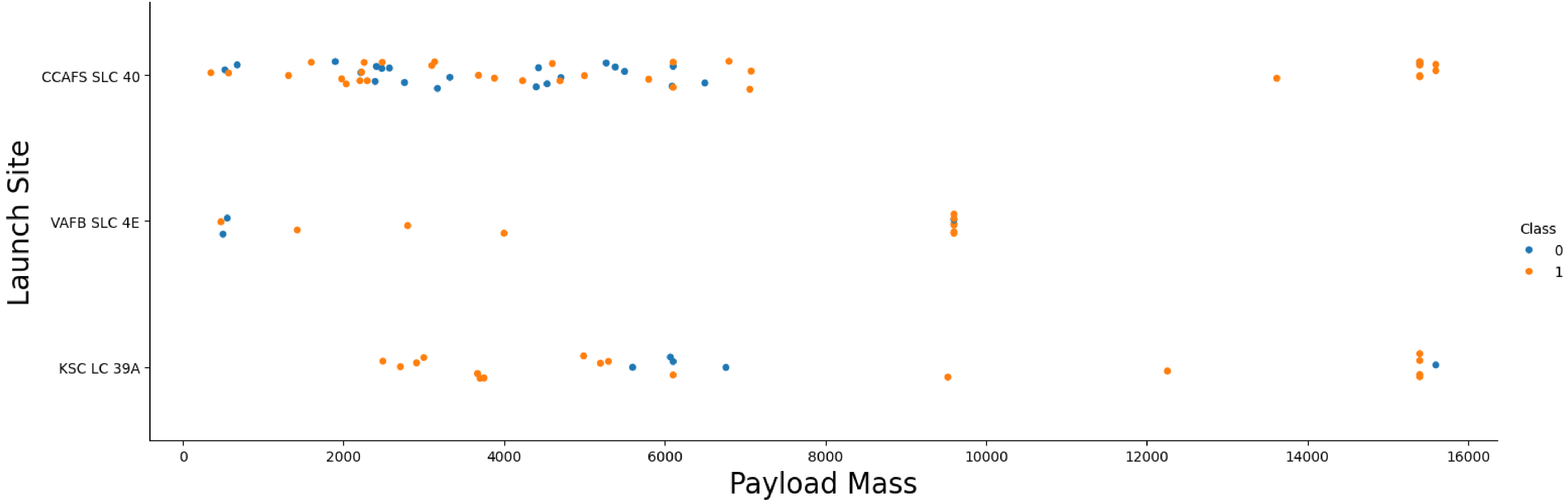
Flight Number vs. Launch Site

- 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

- We found that, The greater the payload mass for the launch site CCAFS SLC 40, the higher the success rate of the rocket.

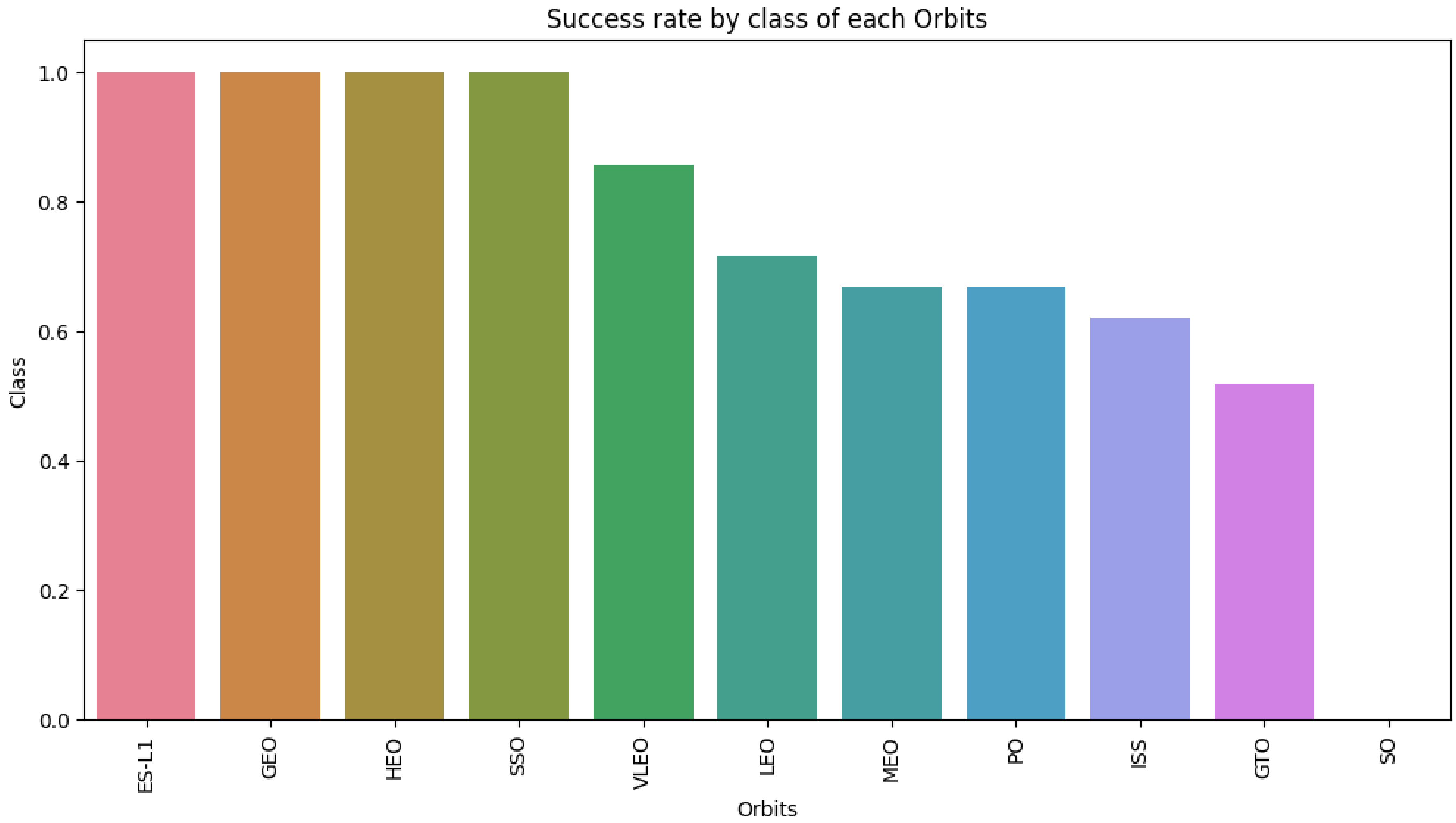


Success Rate vs. Orbit Type

From the plot, we can see that

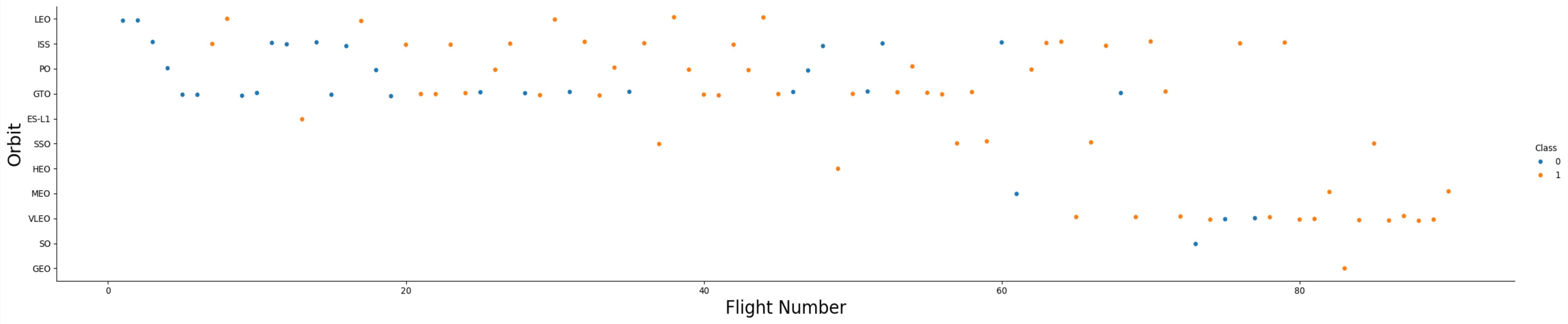
ES-L1, GEO, HEO, SSO, VLEO

had the most success rate.



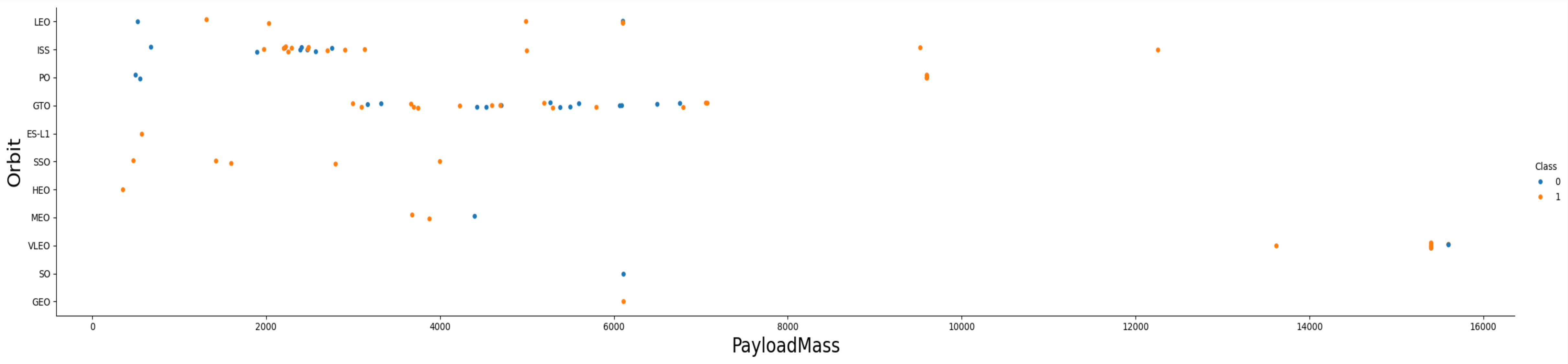
Flight Number vs. Orbit Type

- 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.



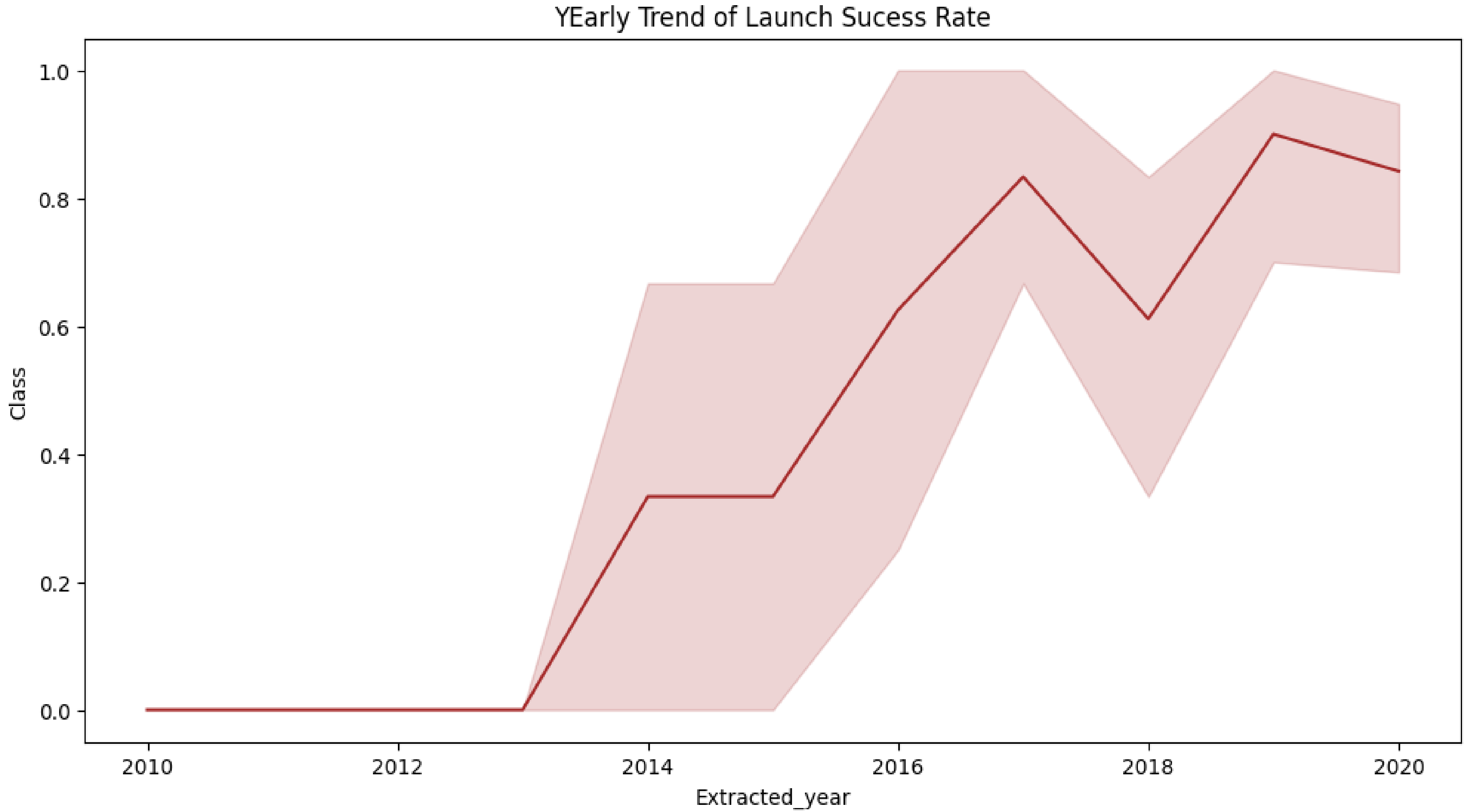
Payload vs. Orbit Type

- We can observe that with heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- From the plot, we can observe that the success rate kept on increasing from 2013 till 2017, then dropped between 2017-18, and again increased till 2020.



All Launch Site Names

- We used the keyword **DISTINCT** to show unique launch sites from the SpaceX data.

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

+ Code + Markdown

```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Python

	launchsite
0	KSC LC-39A
1	CCAFS LC-40
2	CCAFS SLC-40
3	VAFB SLC-4E

Launch Site Names Begin with 'CCA'

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

+ Code + Markdown

```python
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

[11] Python

...

	date	time	boosterversion	launchsite	payload	payloadmasskg	orbit	customer	missionoutcome	landingoutcome
0	2010-04-06	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	Success	Failure (parachute)
1	2010-08-12	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of...	0	LEO (ISS)	NASA (COTS) NRO	Success	Failure (parachute)
2	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
3	2012-08-10	00:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500	LEO (ISS)	NASA (CRS)	Success	No attempt
4	2013-01-03	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677	LEO (ISS)	NASA (CRS)	Success	No attempt

- 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

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

```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

[12]

...

	total_payloadmass
0	45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.4

Task 4

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

```python
task_4 = '''
        SELECT AVG(PayloadMassKG) AS Avg_PayloadMass
        FROM SpaceX
        WHERE BoosterVersion = 'F9 v1.1'
        '''
create_pandas_df(task_4, database=conn)
```

[13] Python

	avg_payloadmass
0	2928.4

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 successful landing outcome in ground pad was acheived.

Hint:Use min function

```python
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

[14] Python

	firstsuccessfull_landing_date
0	2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 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

```
task_6 = '''
        SELECT BoosterVersion
        FROM SpaceX
        WHERE LandingOutcome = 'Success (drone ship)'
            AND PayloadMassKG > 4000
            AND PayloadMassKG < 6000
        '''
create_pandas_df(task_6, database=conn)
```

[15] Python

	boosterversion
0	F9 FT B1022
1	F9 FT B1026
2	F9 FT B1021.2
3	F9 FT B1031.2

- We used the **WHERE** clause to filter for boosters that have successfully landed on a 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

```python
task_7a = '''
        SELECT COUNT(MissionOutcome) AS SuccessOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Success%'
        '''

task_7b = '''
        SELECT COUNT(MissionOutcome) AS FailureOutcome
        FROM SpaceX
        WHERE MissionOutcome LIKE 'Failure%'
        '''
print('The total number of successful mission outcome is:')
display(create_pandas_df(task_7a, database=conn))
print()
print('The total number of failed mission outcome is:')
create_pandas_df(task_7b, database=conn)
```

[16] Python

```
The total number of successful mission outcome is:
```

	successoutcome
0	100

```
The total number of failed mission outcome is:
```

	failureoutcome
0	1

- We used wildcards like '%' to filter for **WHERE** Mission Outcome was a success or a failure.

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.

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

```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

[17] Python

	boosterversion	payloadmasskg
0	F9 B5 B1048.4	15600
1	F9 B5 B1048.5	15600
2	F9 B5 B1049.4	15600
3	F9 B5 B1049.5	15600
4	F9 B5 B1049.7	15600
5	F9 B5 B1051.3	15600
6	F9 B5 B1051.4	15600
7	F9 B5 B1051.6	15600
8	F9 B5 B1056.4	15600
9	F9 B5 B1058.3	15600
10	F9 B5 B1060.2	15600
11	F9 B5 B1060.3	15600

2015 Launch Records

- We used a combination of the **WHERE** clause, **LIKE**, **AND**, and **BETWEEN** conditions to filter for failed landing outcomes in drone ships, their booster versions, and launch site names for the year 2015.

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

```python
task_9 = '''
        SELECT BoosterVersion, LaunchSite, LandingOutcome
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Failure (drone ship)'
            AND Date BETWEEN '2015-01-01' AND '2015-12-31'
        '''
create_pandas_df(task_9, database=conn)
```

[18] Python

	boosterversion	launchsite	landingoutcome
0	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
1	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

```python
task_10 = '''
        SELECT LandingOutcome, COUNT(LandingOutcome)
        FROM SpaceX
        WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
        GROUP BY LandingOutcome
        ORDER BY COUNT(LandingOutcome) DESC
        '''
create_pandas_df(task_10, database=conn)
```

[19] Python

...

	landingoutcome	count
0	No attempt	10
1	Success (drone ship)	6
2	Failure (drone ship)	5
3	Success (ground pad)	5
4	Controlled (ocean)	3
5	Uncontrolled (ocean)	2
6	Precluded (drone ship)	1
7	Failure (parachute)	1

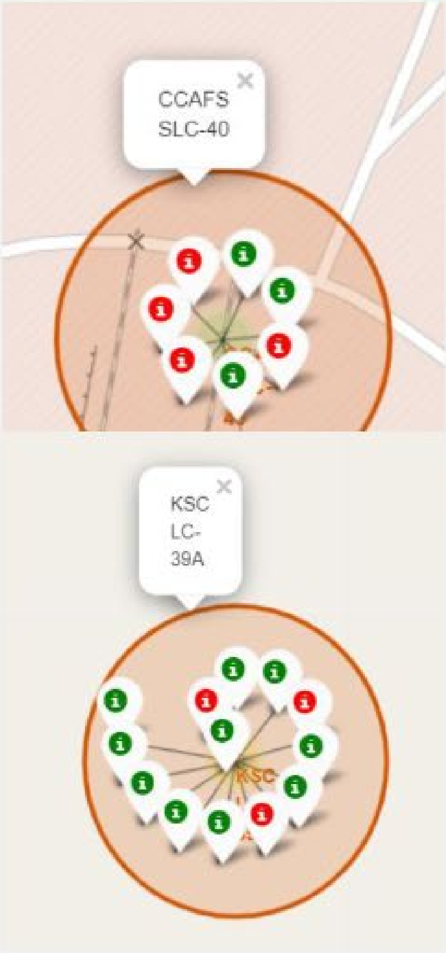
- We selected Landing outcomes and the **COUNT** of landing outcomes from the data and used the **WHERE** clause to filter for landing outcomes **BETWEEN** 2010-06-04 to 2010-03-20.
- We applied the **GROUP BY** clause to group the landing outcomes and the **ORDER BY** clause to order the grouped landing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

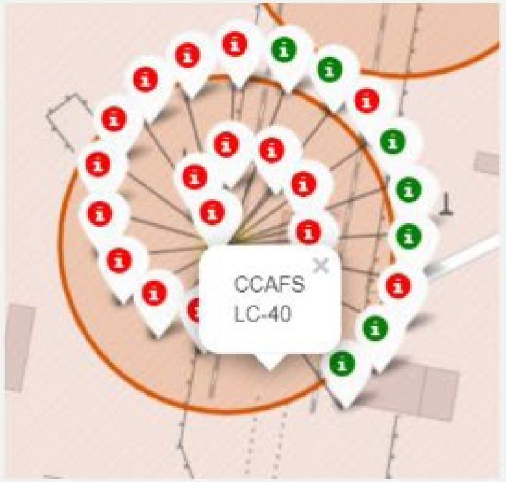
Section 4

Launch Sites Proximities Analysis

All launch sites global map markers

Markers showing launch sites with color labels





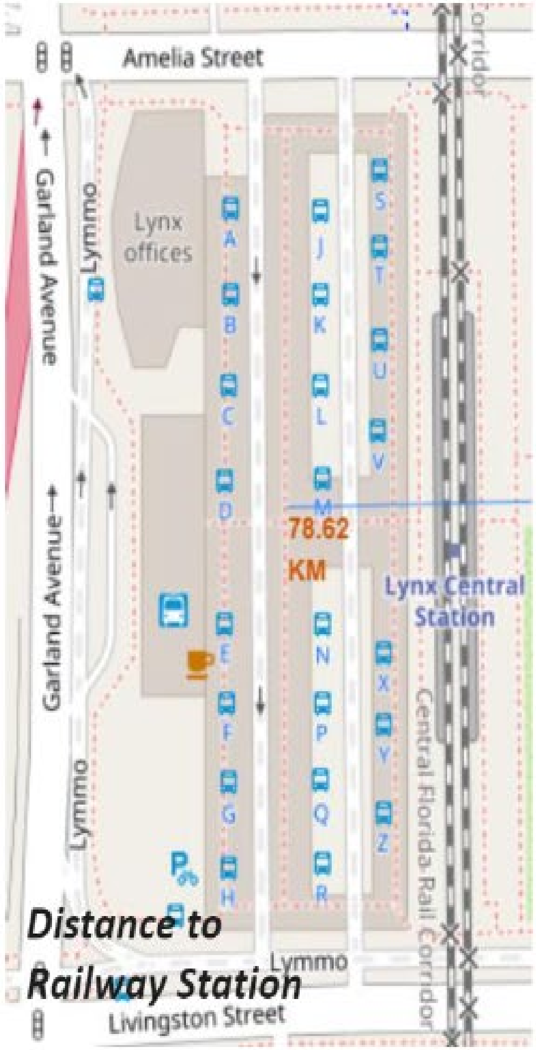
Florida Launch Sites

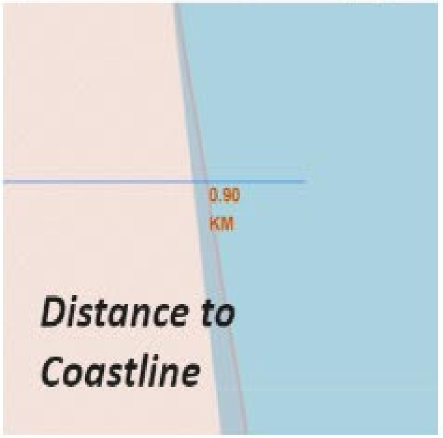
Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

37

Launch Site distance to landmarks





•Are launch sites in close proximity to railways? No

•Are launch sites in close proximity to highways? No

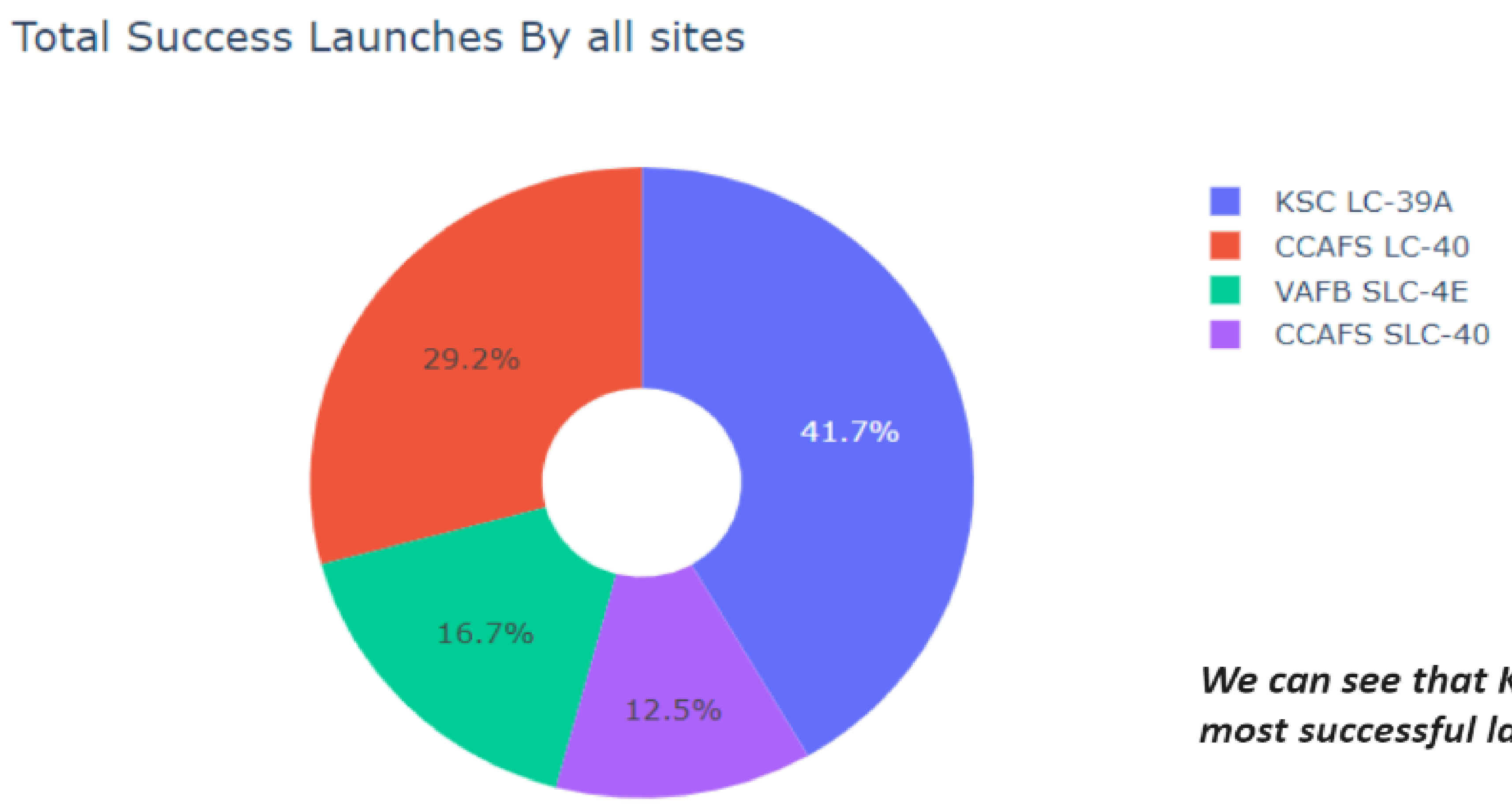
•Are launch sites in close proximity to coastline? Yes

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

Section 5

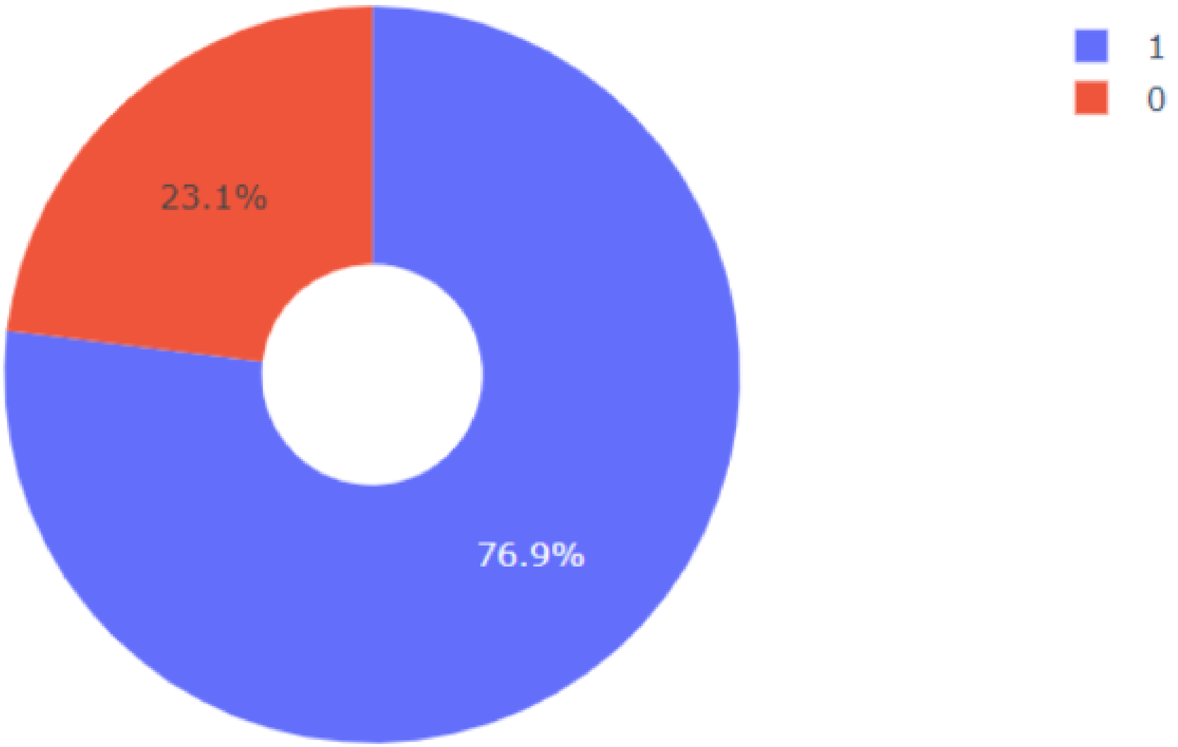
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



We can see that KSC LC-39A had the most successful launches from all the sites

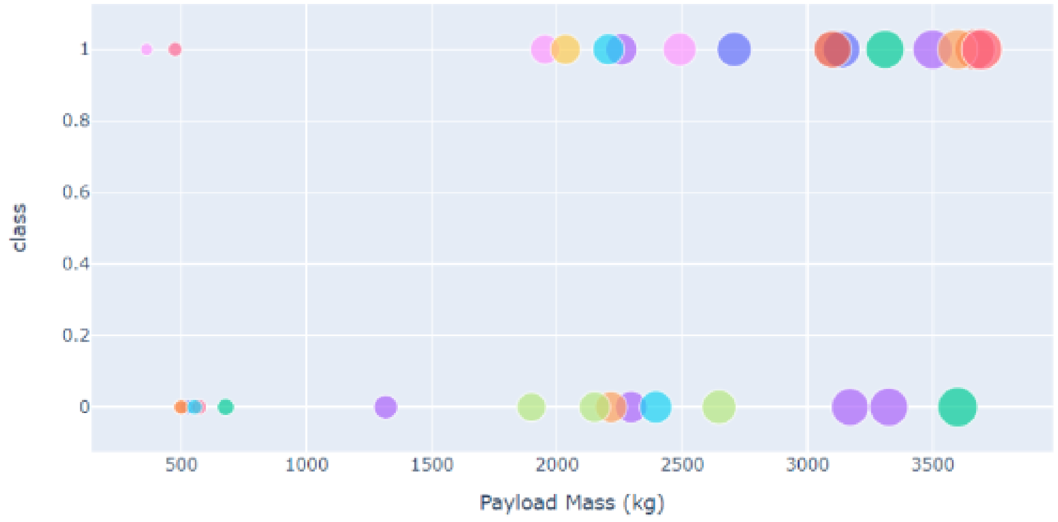
Pie chart showing the Launch site with the highest launch success ratio



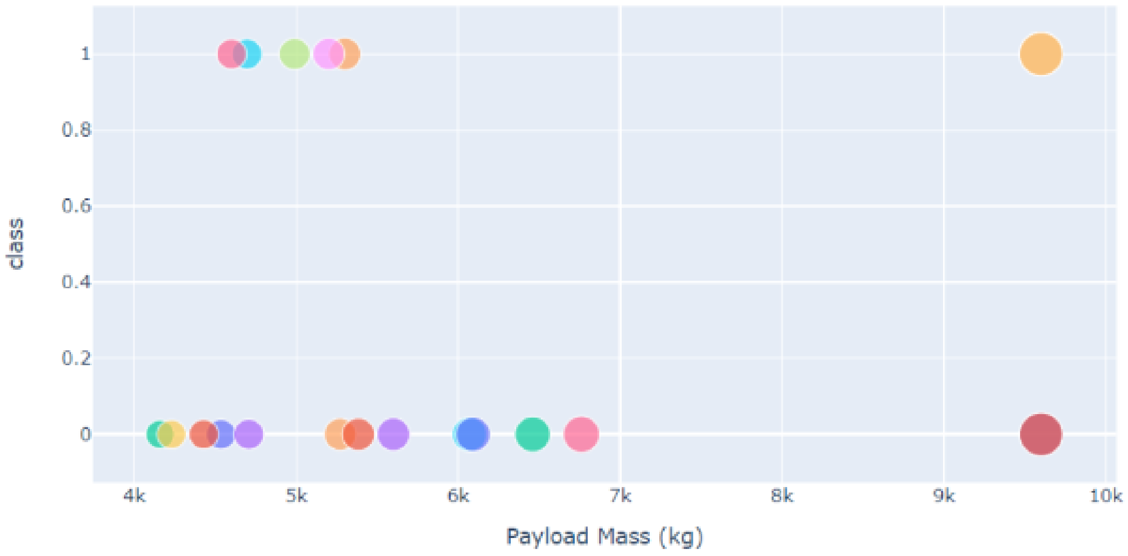
KSC LC-39A achieved a 76.9% success rate while getting a 23.1% failure rate

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

Low Weighted Payload 0kg – 4000kg



Heavy Weighted Payload 4000kg – 10000kg



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

Section 6

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier is the model with the highest classification accuracy.

Find the method performs best:

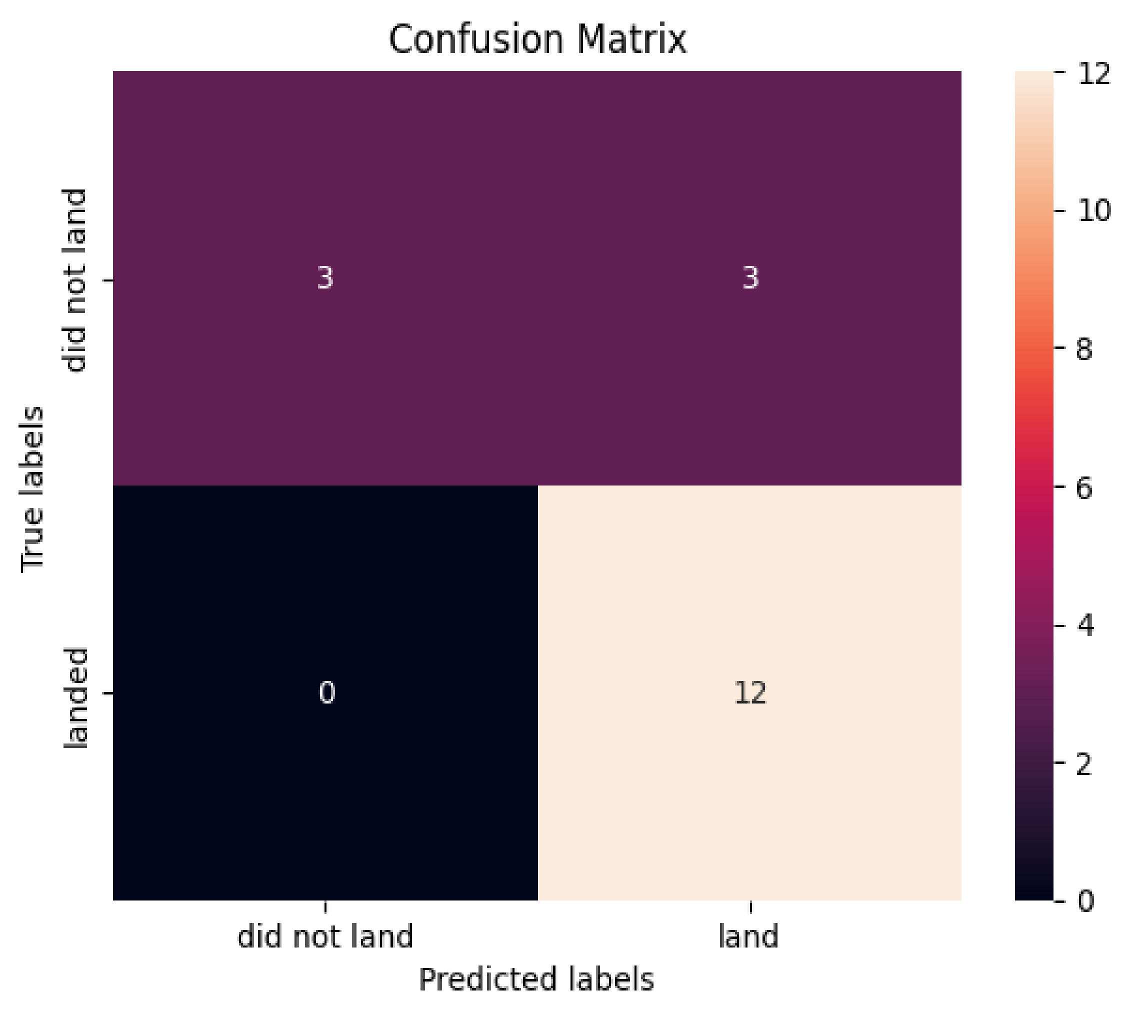
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

bestalgorithm = max(models, key=models.get)
print('Best model is', bestalgorithm,'with a score of', models[bestalgorithm])
if bestalgorithm == 'DecisionTree':
    print('Best parameter is :', tree_cv.best_params_)
if bestalgorithm == 'KNeighbors':
    print('Best parameter is :', knn_cv.best_params_)
if bestalgorithm == 'LogisticRegression':
    print('Best parameter is :', logreg_cv.best_params_)
if bestalgorithm == 'SupportVector':
    print('Best parameter is :', svm_cv.best_params_)
```

29]

```
Best model is DecisionTree with a score of 0.8732142857142856
Best parameter is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'splitter': 'random'}
```

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 this task.

Thank you!