IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

LAIBA KASHIF
15/DEC/2023

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

- 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

- Present your data collection with SpaceX REST calls using key phrases and flowcharts

- https://github.com/Laiba-gif/IBM-Data-Science-Capston-SpaceX-/blob/816fcb9572157315d57cbe62122430c6693fd3e4/jupyter-labs-spacex-data-collection-api.ipynb

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

In [13]:
```
static_json_url='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/da
```

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

In [14]:
```
response.status_code
```

Out[14]: 200

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

In [15]:
```
# 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

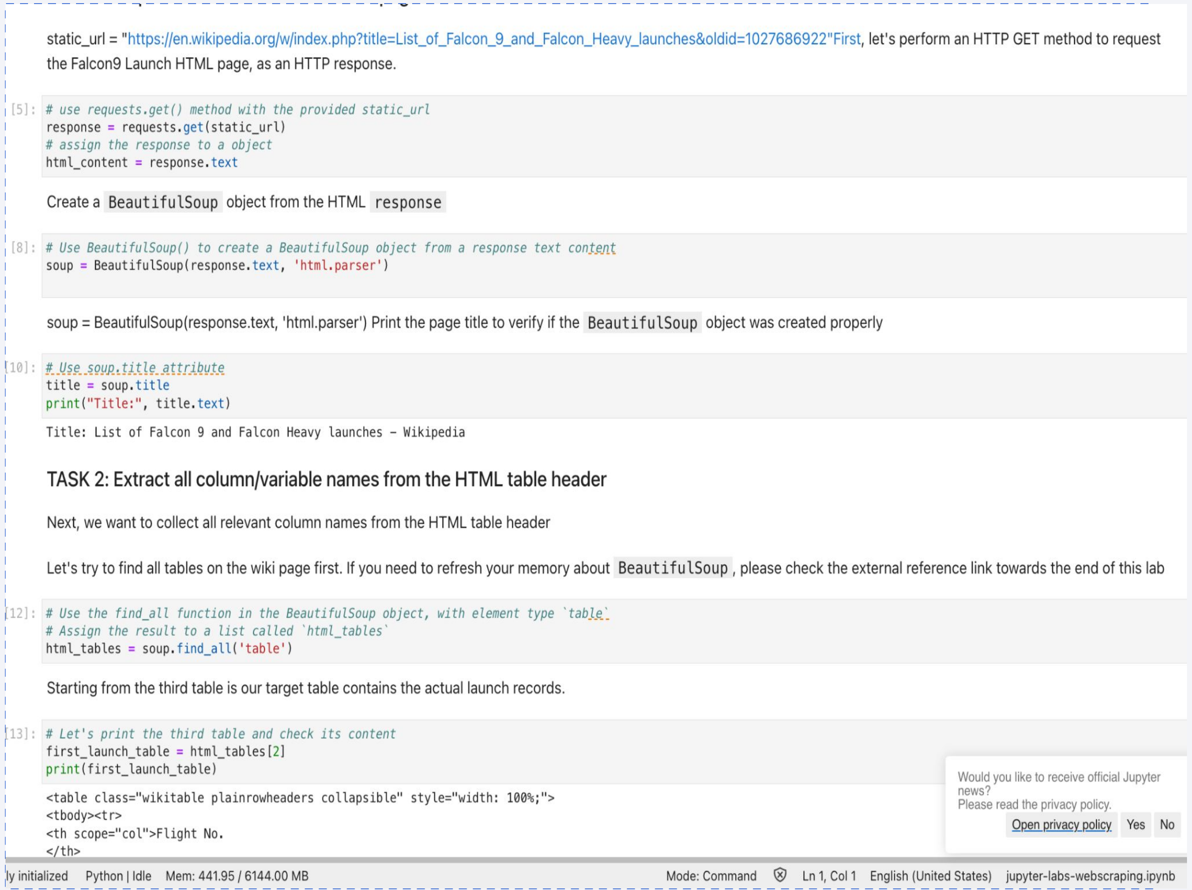
In [16]:
```
# Get the head of the dataframe
data.head()
```

Out[16]:

	static_fire_date_utc	static_fire_date_unix	net	window	rocket	success	failures	details	crew
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.
- https://github.com/Laiba-gif/IBM-Data-Science-Capston-SpaceX-/blob/83468425abad0bfaeafd9ee944f4a4af3aa93066/jupyter-labs-webscraping.ipynb



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.

https://github.com/Laiba-gif/IBM-Data-Science-Capston-SpaceX-/blob/db691fe0fb0d5d35a9e52b9524b1c32b97465ed2/labs-jupyter-spacex-Data%20wrangling.ipynb

TASK 2: Calculate the number and occurrence of each orbit

Use the method `.value_counts()` to determine the number and occurrence of each orbit in the column `Orbit`

In [8]:
```
# Apply value_counts on Orbit column
Orbit_counts = df['Orbit'].value_counts()
print(Orbit_counts)
```

```
GTO      27
ISS      21
VLEO     14
PO        9
LEO       7
SSO       5
MEO       3
ES-L1     1
HEO       1
SO        1
GEO       1
Name: Orbit, dtype: int64
```

TASK 3: Calculate the number and occurence of mission outcome of the orbits

Use the method `.value_counts()` on the column `Outcome` to determine the number of `landing_outcomes`.Then assign it to a variable landing_outcomes.

In [11]:
```
# landing_outcomes = values on Outcome column
landing_outcomes =df['Outcome'].value_counts()
print(landing_outcomes)
```

```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

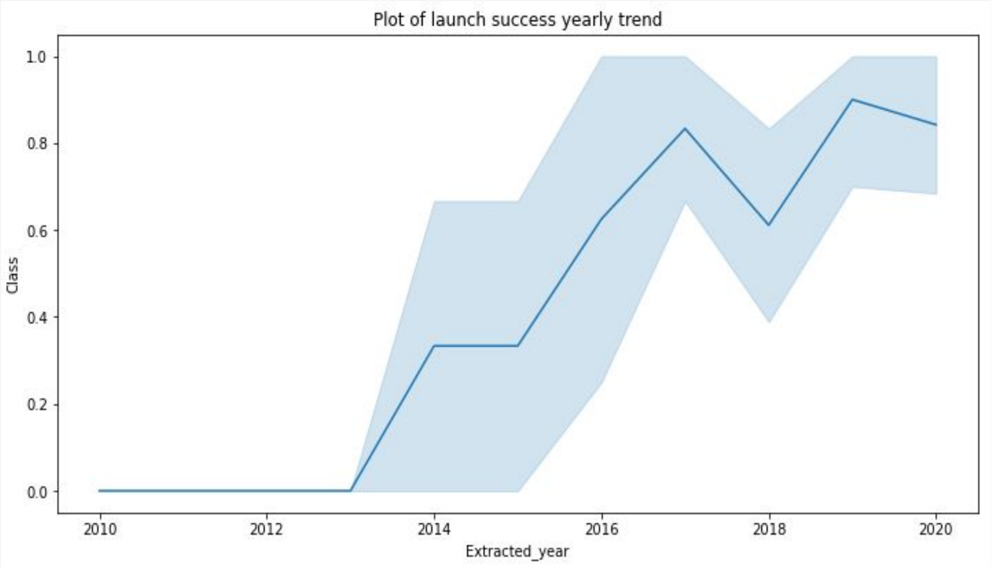
`True Ocean` means the mission outcome was successfully landed to a specific region of the ocean while `False Ocean` means the mission outcome was unsuccessfully landed to a specific region of the ocean. `True RTLS` means the mission outcome was successfully landed to a ground pad `False RTLS` means the mission outcome was unsuccessfully landed to a ground pad. `True ASDS` means the mission outcome was successfully landed to a drone ship `False ASDS` means the mission outcome was unsuccessfully landed to a drone ship. `None ASDS` and `None None` these represent a failure to land.

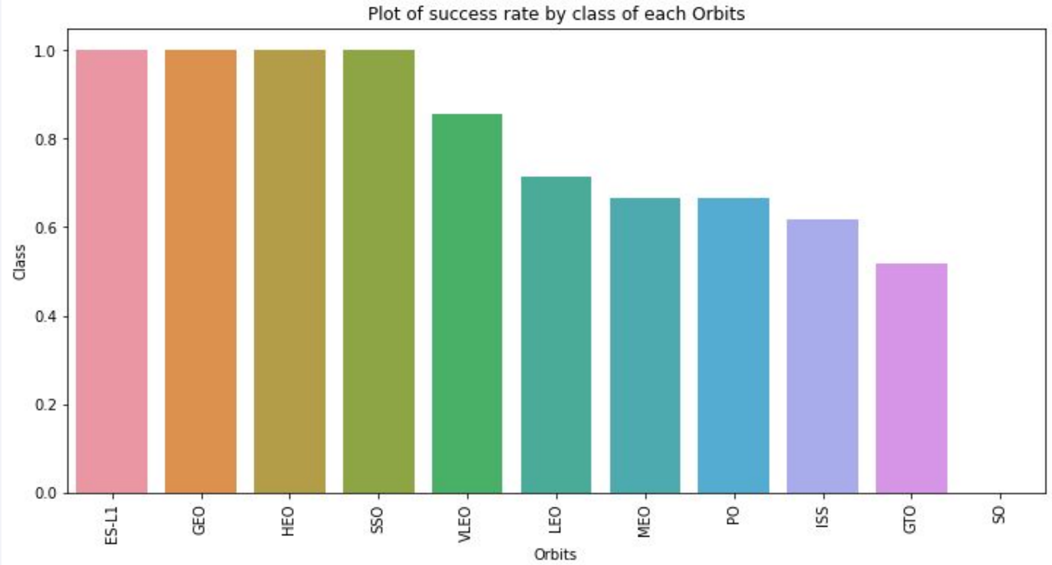
In [12]:
```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
```

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.
- https://github.com/Laiba-gif/IBM-Data-Science-Capston-SpaceX-/blob/acd0a977a80213918cf819e3112ddf907e88de38/EDA%20with%20Data%20Visualization.ipynb





EDA with SQL

Unique Launch Sites:

Query: Obtained the names of distinct launch sites involved in the space missions.

Total Payload Mass by NASA (CRS):

Query: Calculated the cumulative payload mass carried by boosters launched by NASA under the CRS (Commercial Resupply Services) program.

Average Payload Mass by Booster Version F9 v1.1:

Query: Computed the average payload mass carried by boosters of the specific version F9 v1.1.

Total Number of Mission Outcomes:

Query: Determined the overall count of both successful and failed mission outcomes.

Failed Landing Outcomes on Drone Ship:

Query: Identified instances of failed landing outcomes on drone ships, providing details such as the associated booster version and launch site names

Notebook LInk:

https://github.com/Laiba-gif/IBM-Data-Science-Capston-SpaceX-/blob/acd0a977a80213918cf819e3112ddf907e88de38/EDA%20with%20SQL.ipynb .

Build an Interactive Map with Folium

In our mapping visualization project, we annotated all launch sites and incorporated map elements such as markers, circles, and lines to visually represent the success or failure of launches on a Folium map. To categorize launch outcomes into binary classes, we assigned the label 0 for failure and 1 for success. This allowed us to create color-coded marker clusters, enabling us to easily identify launch sites with higher success rates.

As part of our analysis, we calculated the distances between each launch site and its surrounding areas. We addressed specific questions such as:

Proximity to Transportation Networks:

Explored whether launch sites are located near railways and highways.

Coastline Proximity:

Investigated the distance of launch sites from coastlines.

Urban Distances:

Determined whether launch sites maintain a certain distance from cities.

By integrating these spatial considerations into our mapping visualization, we gained valuable insights into the geographical aspects of launch sites and their surroundings. This approach provided a clear and intuitive way to assess the success rates of launches across different locations.

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.
- https://github.com/Laiba-gif/IBM-Data-Science-Capston-SpaceX-/blob/acd0a977a80213918cf819e3112ddf907e88de38/app.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.
- https://github.com/Laiba-gif/IBM-Data-Science-Capston-SpaceX-/blob/acd0a977a80213918cf819e3112ddf907e88de38/Machine%20Learning%20Prediction.ipynb

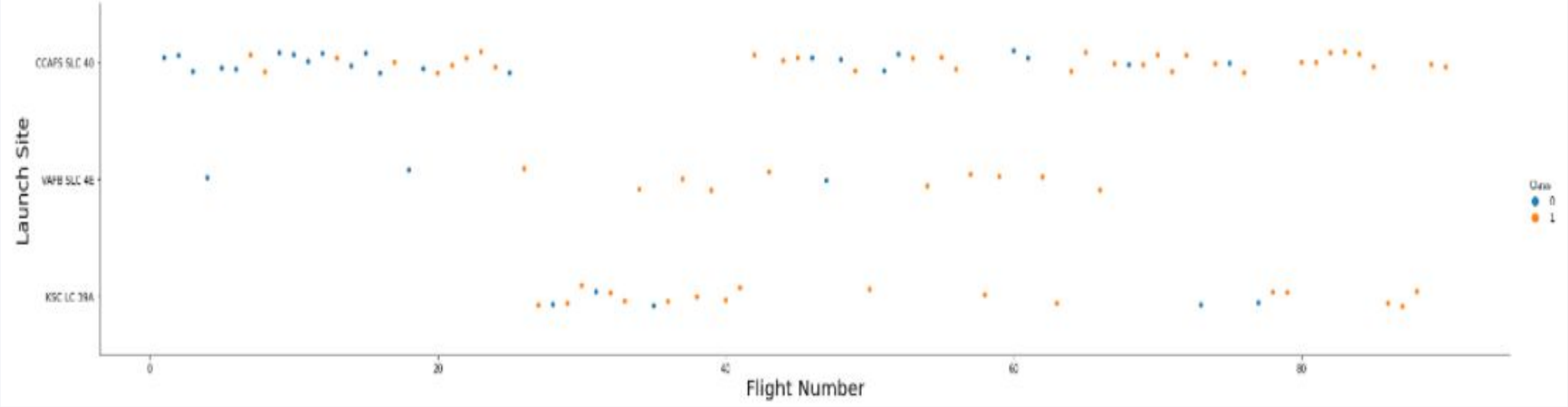
Results

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

Section 2

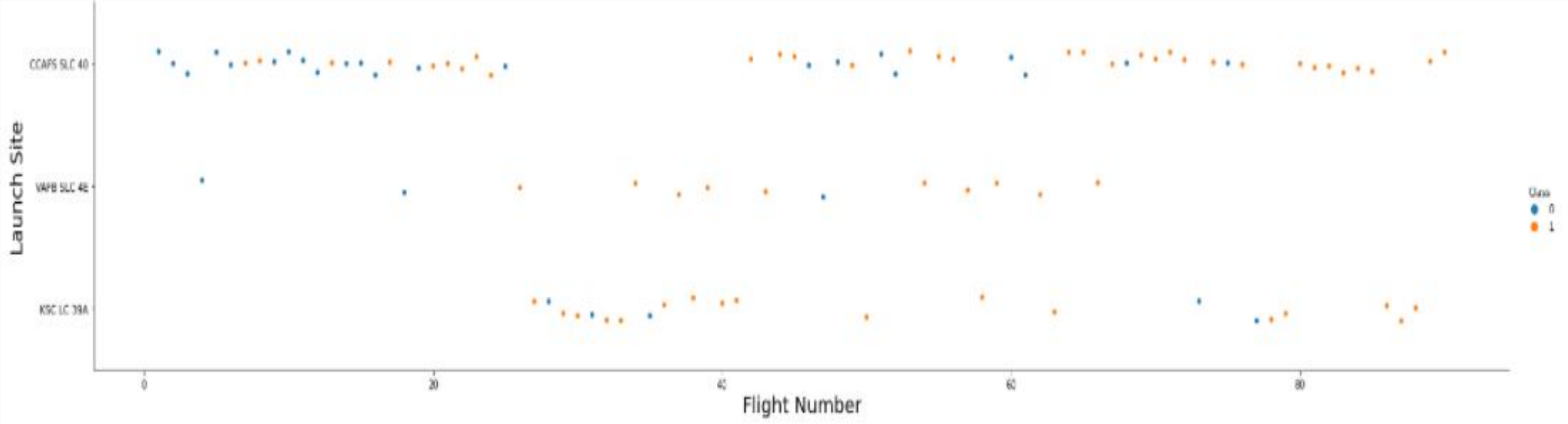
Insights drawn from EDA

Flight Number vs. Launch Site

- The analysis of the plot revealed a positive correlation between the number of flights conducted at a launch site and its success rate. Specifically, launch sites with higher flight volumes tended to exhibit greater success rates in their space missions.
- 

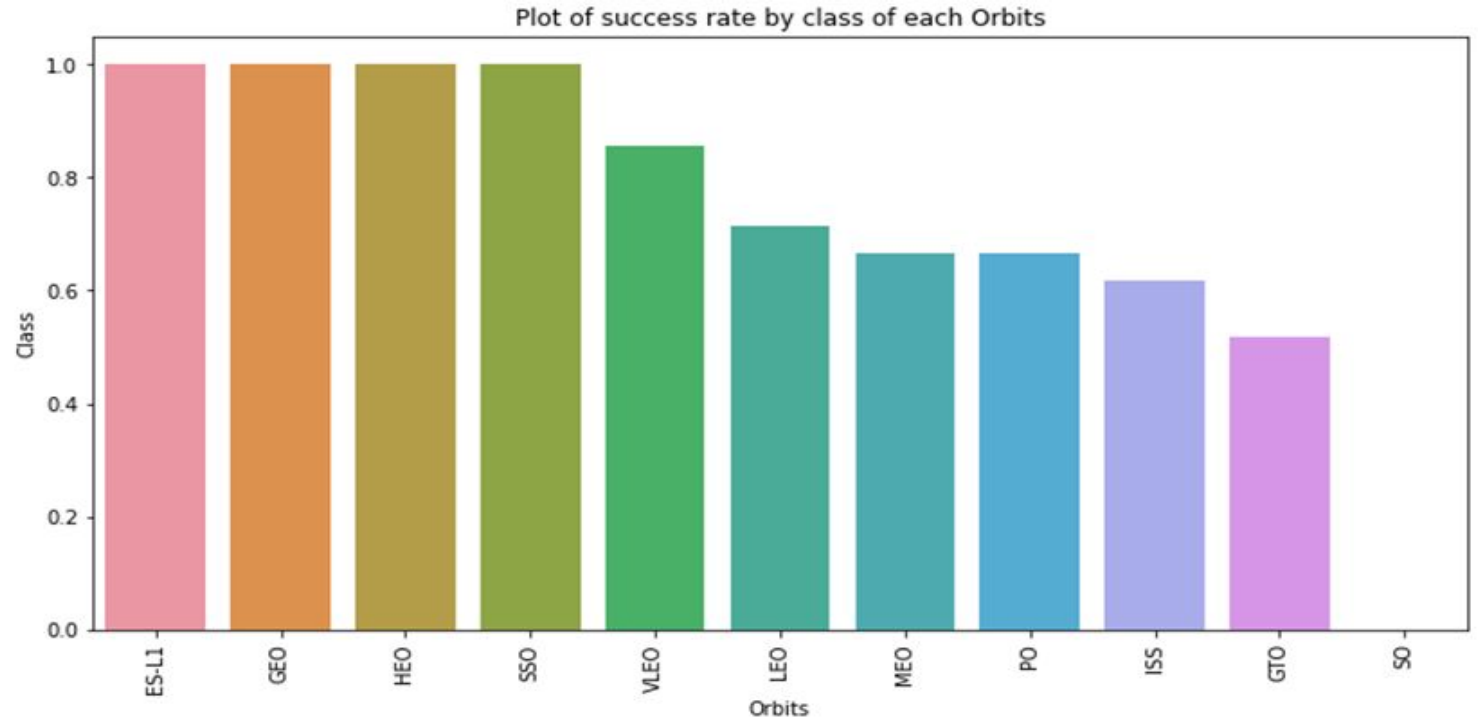
Payload vs. Launch Site

- greeter the payload mass higher the launch site success



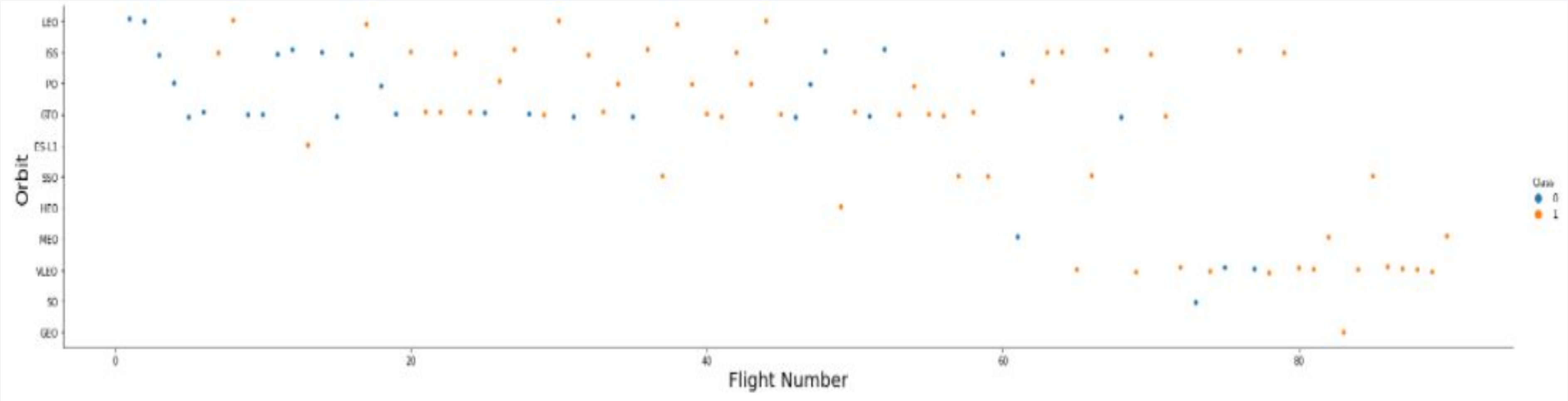
Success Rate vs. Orbit Type

- From the plot, we can see that ES-L1, GEO, HEO, SSO, VLEO had the most success rate.

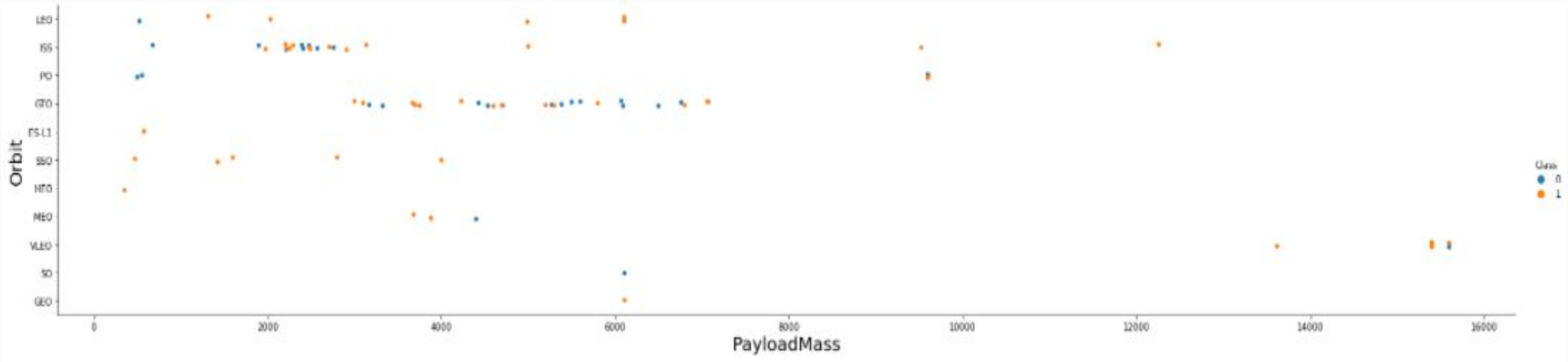


Flight Number vs. Orbit Type

- 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.

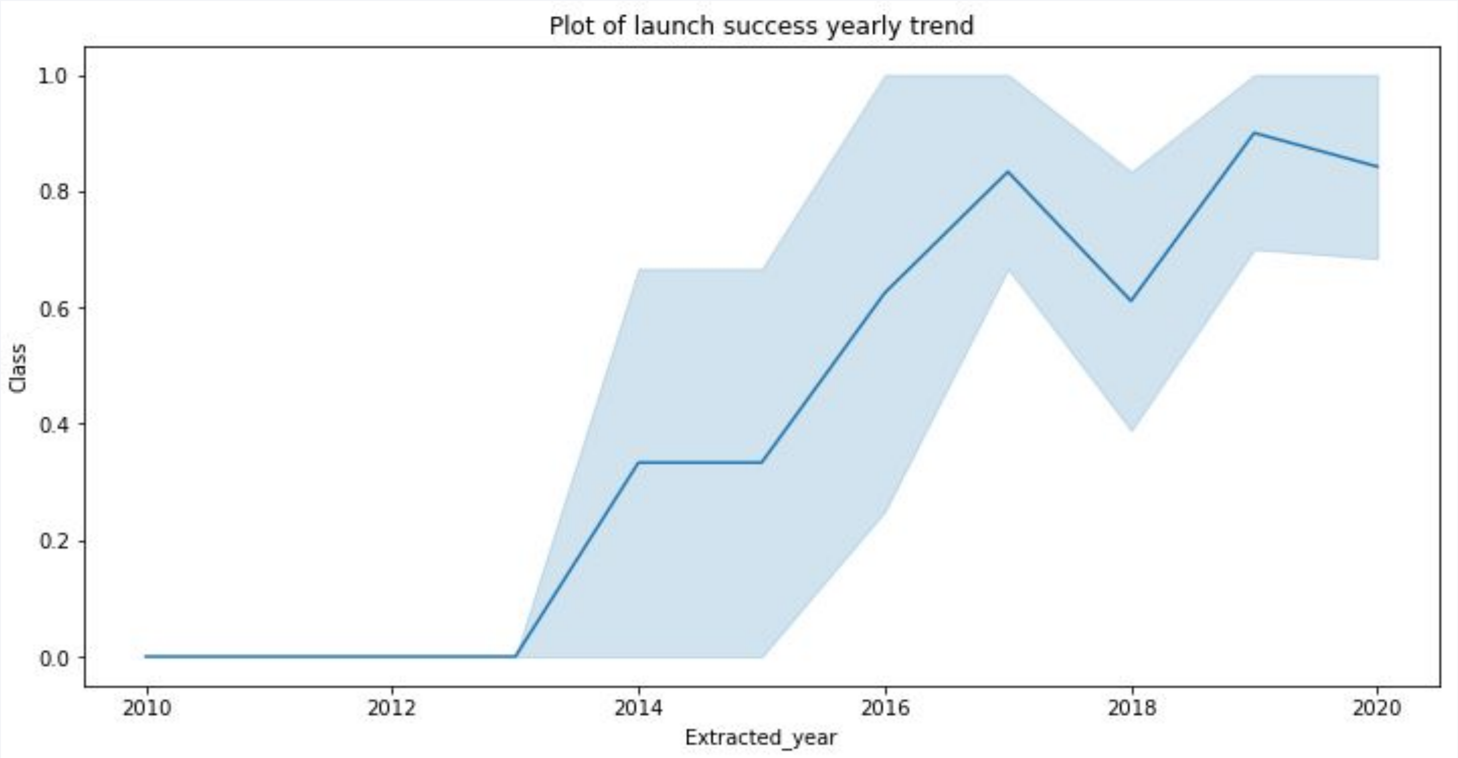


Payload vs. Orbit Type

- 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 success rate since 2013 kept on increasing till 2020.



All Launch Site Names

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

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

In [10]:
```
task_1 = '''
        SELECT DISTINCT LaunchSite
        FROM SpaceX
'''
create_pandas_df(task_1, database=conn)
```

Out[10]:

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

Launch Site Names Begin with 'CCA'

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

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

In [11]:
```
task_2 = '''
        SELECT *
        FROM SpaceX
        WHERE LaunchSite LIKE 'CCA%'
        LIMIT 5
        '''
create_pandas_df(task_2, database=conn)
```

Out[11]:

	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

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)

In [12]:
```
task_3 = '''
        SELECT SUM(PayloadMassKG) AS Total_PayloadMass
        FROM SpaceX
        WHERE Customer LIKE 'NASA (CRS)'
        '''
create_pandas_df(task_3, database=conn)
```

Out[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

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

In [13]:

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

Out[13]:

	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

-

In [14]:
```
task_5 = '''
        SELECT MIN(Date) AS FirstSuccessfull_landing_date
        FROM SpaceX
        WHERE LandingOutcome LIKE 'Success (ground pad)'
        '''
create_pandas_df(task_5, database=conn)
```

Out[14]:

	firstsuccessfull_landing_date
0	2015-12-22

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
-

```
In [15]: task_6 = '''
                 SELECT BoosterVersion
                 FROM SpaceX
                 WHERE LandingOutcome = 'Success (drone ship)'
                     AND PayloadMassKG > 4000
                     AND PayloadMassKG < 6000
                 '''
         create_pandas_df(task_6, database=conn)
```

Out[15]:

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

Total Number of Successful and Failure Mission Outcomes

- We used wildcard like ‘%’ to filter for **WHERE** MissionOutcome was a success or a failure.

List the total number of successful and failure mission outcomes

In [16]:

```
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)
```

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

	successoutcome
0	100

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

Out[16]:

	failureoutcome
0	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.
-

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

In [17]:
```
task_8 = '''
        SELECT BoosterVersion, PayloadMassKG
        FROM SpaceX
        WHERE PayloadMassKG = (
                                SELECT MAX(PayloadMassKG)
                                FROM SpaceX
                                )
        ORDER BY BoosterVersion
        '''
create_pandas_df(task_8, database=conn)
```

Out[17]:

	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 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

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

In [18]:
```
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)
```

Out[18]:

	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

- 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.

Rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad))

In [19]:
```
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)
```

Out[19]:

	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

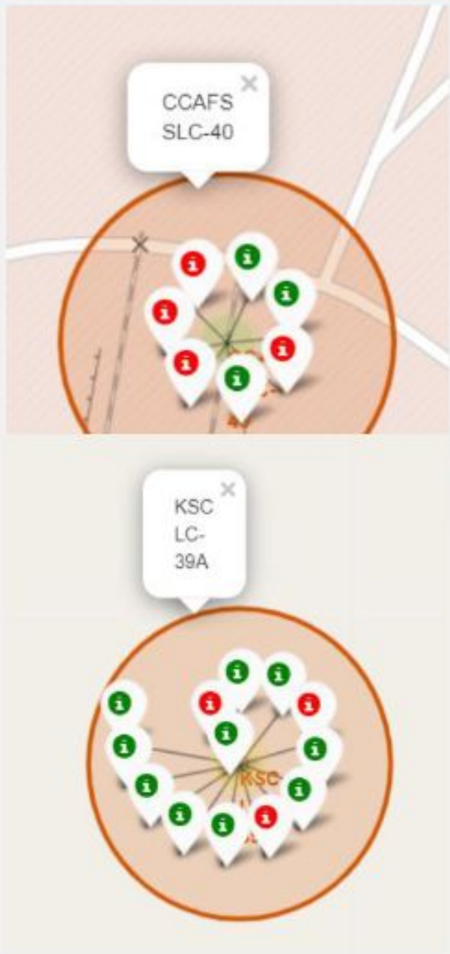
Section 3

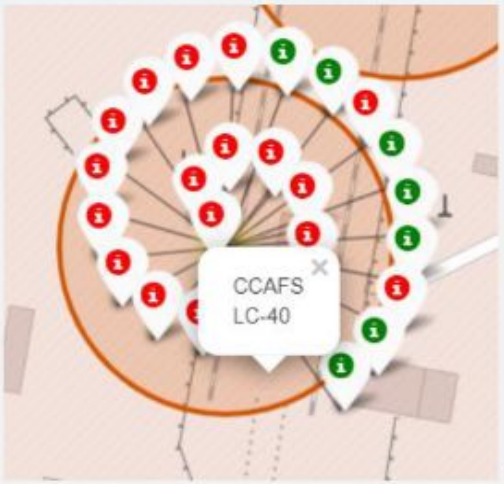
Launch Sites Proximities Analysis

GLOBAL MAP


VAFB
SLC-
4E
We can see that the SpaceX launch sites are in the United States of America coasts.
Florida and California

Markers showing launch sites with color labels



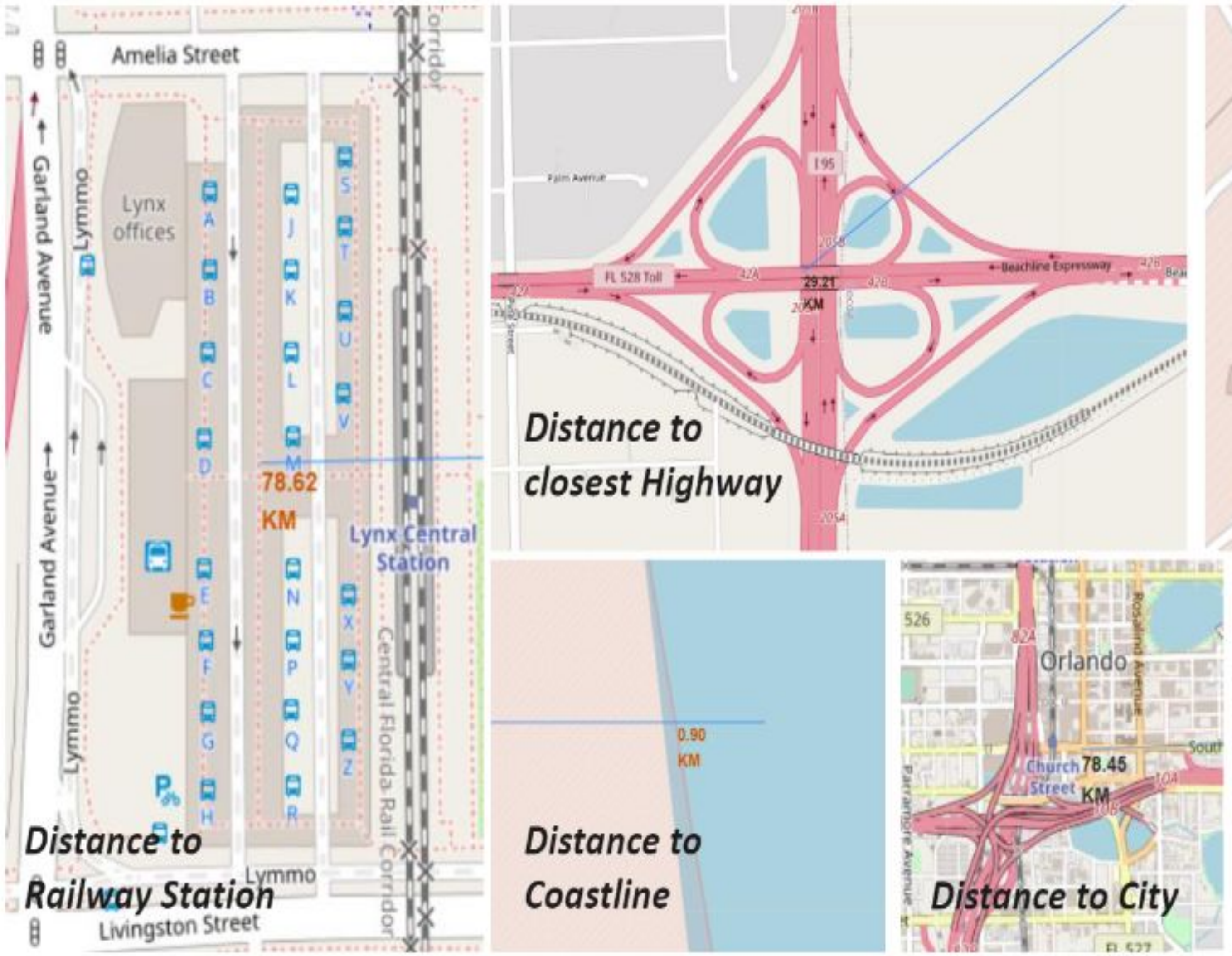


Florida Launch Sites

Green Marker shows successful Launches and Red Marker shows Failures

California Launch Site

Launch Site distance to landmarks

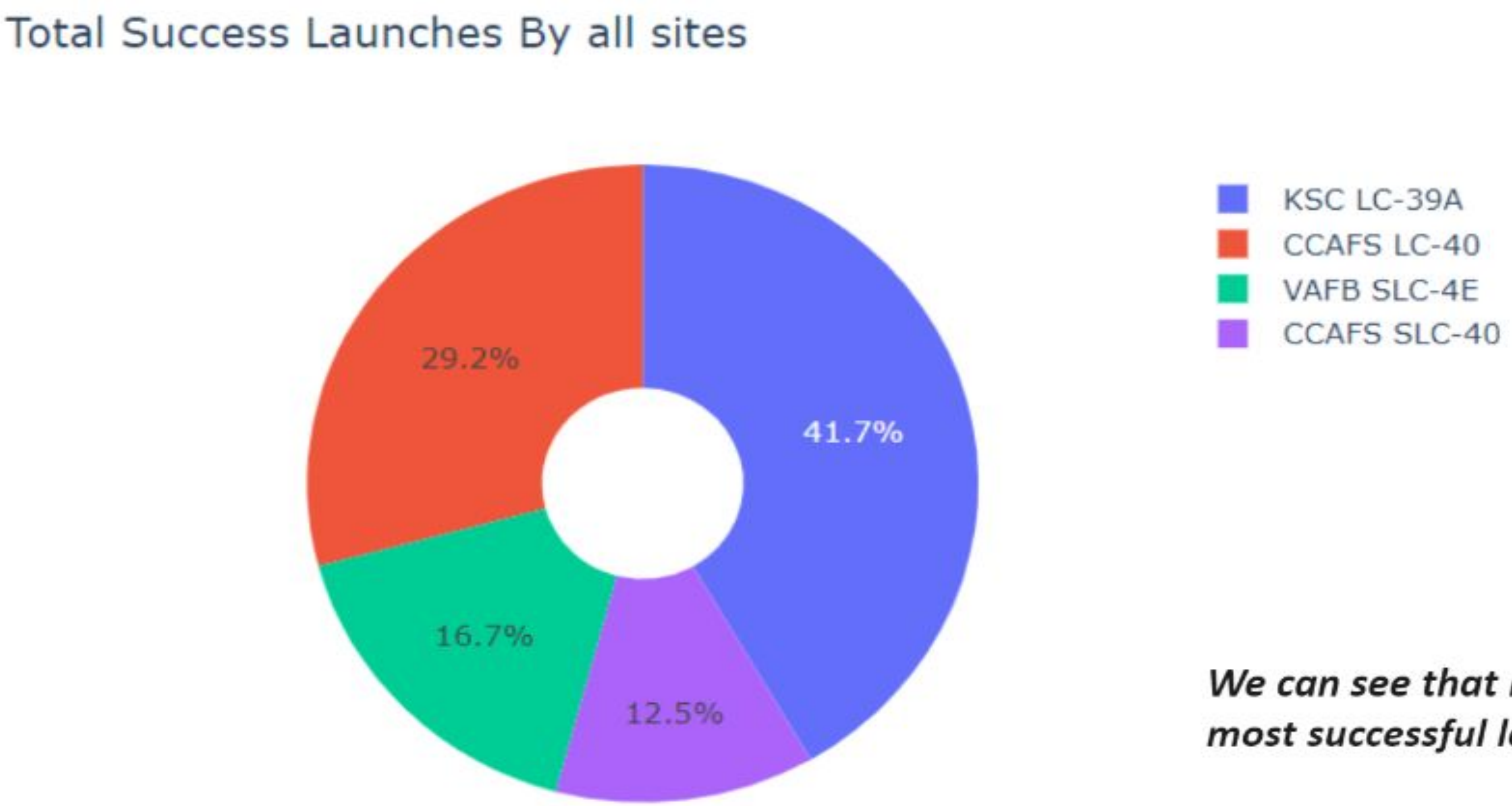


- Are launch sites in close proximity to railways? No
- Are launch sites in close proximity to highways? No
- Are launch sites in close proximity to coastline? Yes
- Do launch sites keep certain distance away from cities? Yes

Section 4

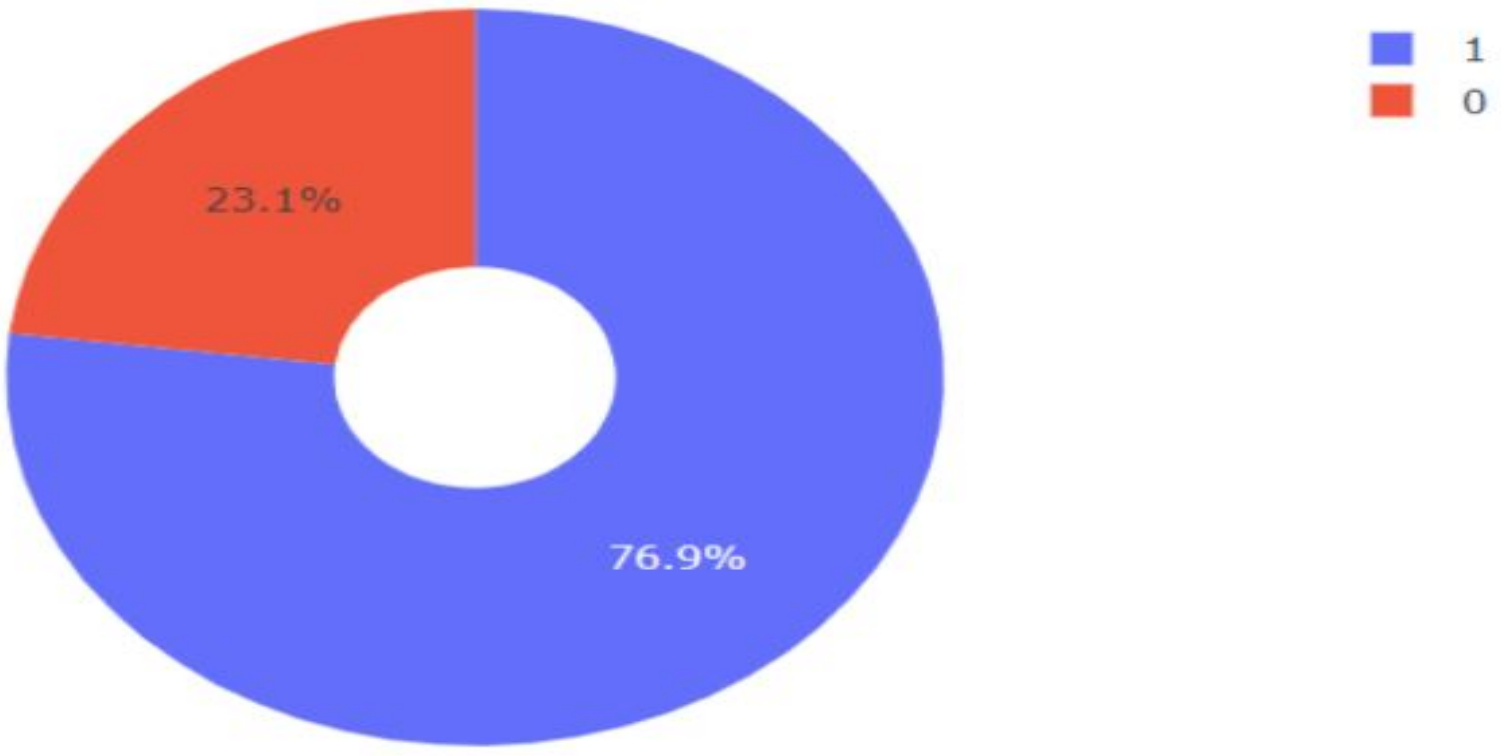
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



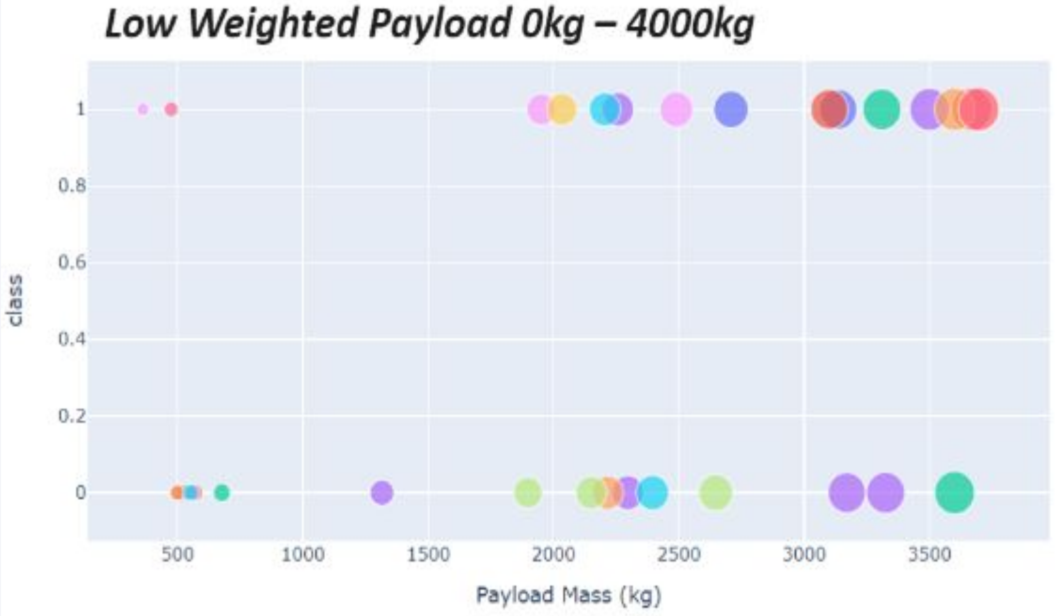
We can see that KSC LC-39A had the most successful launches from all the sites

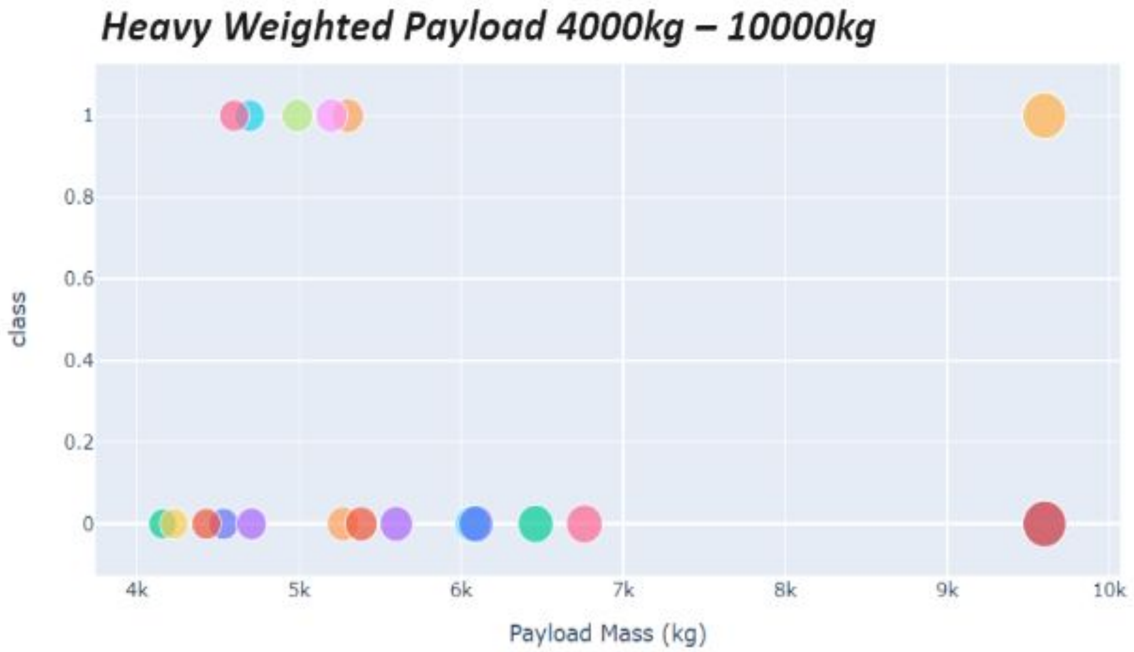
Pie chart showing the Launch site with the highest launch success ratio



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





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

Section 5

Predictive Analysis (Classification)

Classification Accuracy

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

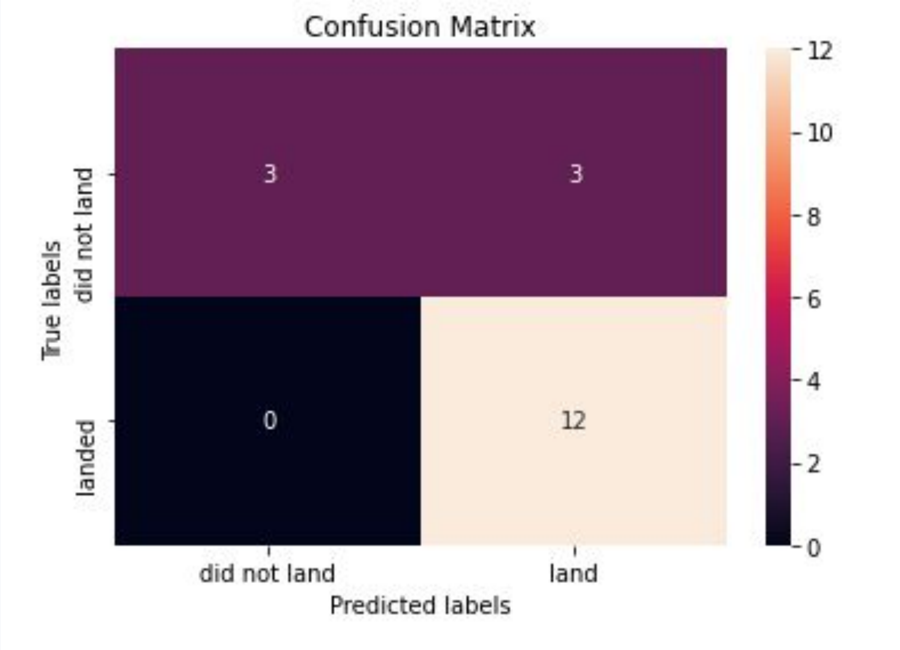
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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

```
Best model is DecisionTree with a score of 0.8732142857142856
Best params is : {'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', '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!