IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Bhoomi Baraiya
03/07/2022

Outline

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:
  - The data was collected by the information of all the unsuccessful landings. Most unsuccessful landing were planned and controlled by the Space X.
- 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

Data Collection

- The Data collection was done using get request to the spaceX API.
- Next, We decoded the response content as Json using .Jason() function call and turn it into a pandas dataframe.
- Then the data was cleaned, checked for missing values and fill in the missing values.
- The objective was to extract the launch records as HTML table, parse the table and convert it to a pandas datafram.

Data Collection – SpaceX API

- Get request method is used to collect data, and for cleaning and did some basic wrangling and formatting.
- The link to notebook is Data Collection With API.ipynb

In [5]:
```python
# Takes the dataset and uses the cores column to call the API and append the data to the lists
def getCoreData(data):
    for core in data['cores']:
            if core['core'] != None:
                response = requests.get("https://api.spacexdata.com/v4/cores/"+core['core']).json()
                Block.append(response['block'])
                ReusedCount.append(response['reuse_count'])
                Serial.append(response['serial'])
            else:
                Block.append(None)
                ReusedCount.append(None)
                Serial.append(None)
            Outcome.append(str(core['landing_success'])+' '+str(core['landing_type']))
            Flights.append(core['flight'])
            GridFins.append(core['gridfins'])
            Reused.append(core['reused'])
            Legs.append(core['legs'])
            LandingPad.append(core['landpad'])
```

Now let's start requesting rocket launch data from SpaceX API with the following URL:

In [6]:
```python
spacex_url="https://api.spacexdata.com/v4/launches/past"
```

In [7]:
```python
response = requests.get(spacex_url)
```

Check the content of the response

In [8]:
```python
print(response.content)
```

Data Collection - Scraping

- Applying web scarping, we recorded web scrap Falcon 9 launch with BeautifulSoup.
- We converted the table into dataframe.
- The link to the notebook [Data Collection Web Scraping]()

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 [6]:
```
# use requests.get() method with the provided static_url
# assign the response to a object
html_data = requests.get(static_url)
html_data.status_code
```

Out[6]:
```
200
```

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

In [7]:
```
# Use BeautifulSoup() to create a BeautifulSoup object from a response text content
soup = BeautifulSoup(html_data.text, 'html.parser')
```

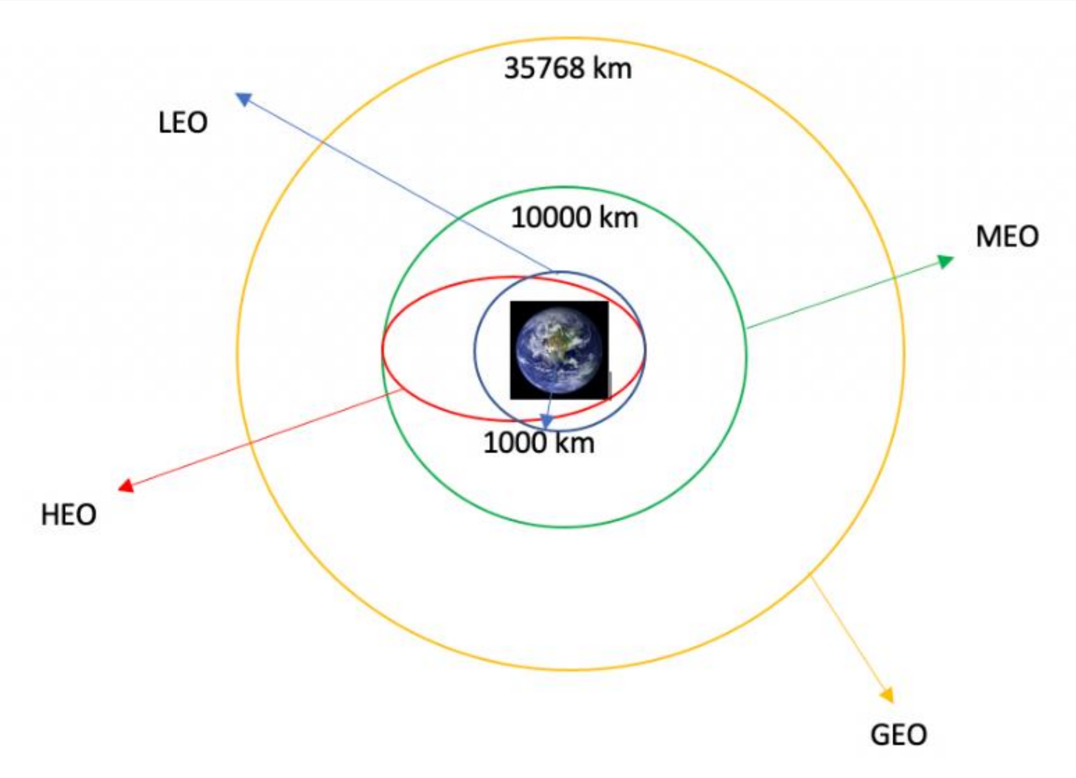
Print the page title to verify if the `BeautifulSoup` object was created properly

In [8]:
```
# Use soup.title attribute
soup.title
```

Out[8]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

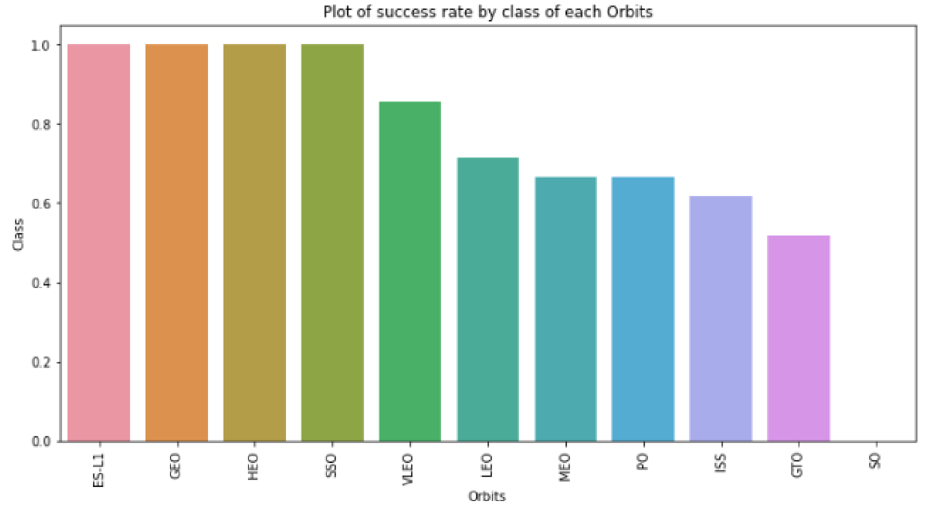
Data Wrangling

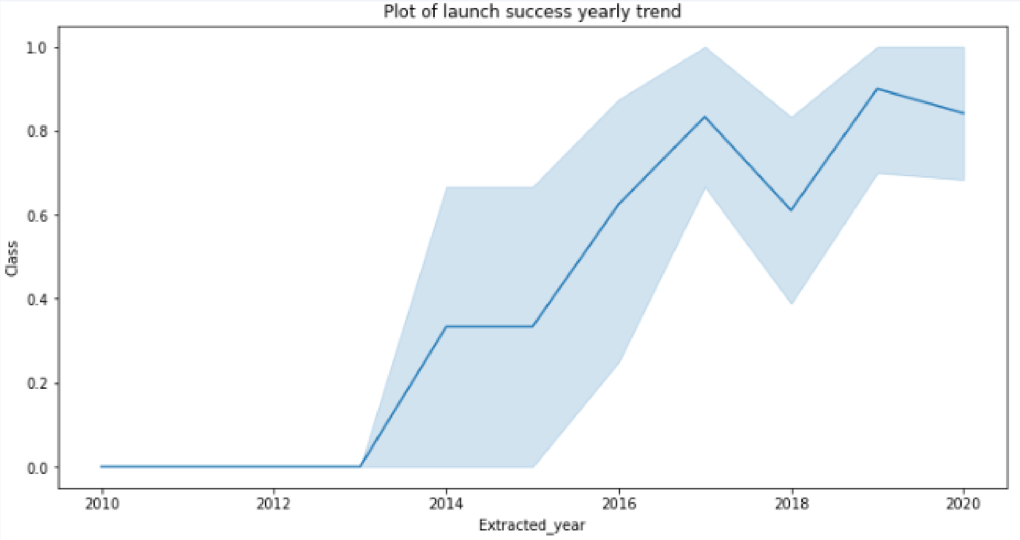
- Performing exploratory data analysis, we determined the training labels.
- Calculated the number of each slid and occurrence of each orbit.
- The link to the notebook [Data Wrangling]



EDA with Data Visualization

- Using data visualization, we explored the relationship between flight number and launch site, payload and launch site, success rate of each orbit type.





- The link to the notebook [EDA With Data Visualization]

EDA with SQL

- Loaded the SpaceX dataset into 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
  - The average payload mass carried by booster version F9 v1.1
- The link to the notebook [EDA with SQL]

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 to class 0 and 1.
- Using the color-labeled marker clucters, we identified which launch sites have relatively high success rate.
- The link to the notebook Interactive Map with Folium

Build a Dashboard with Plotly Dash

- We built an interactive dashboard with Plotly dash
- We plotted pi charts showing total launched by a certain sites
- We plotted scatter graph showing the relationship with outcomes and payload mass for the different booster version
- The link to the notebook Dashboard with Plotly Dash

Predictive Analysis (Classification)

- We loaded data using numpy and pandas, transformed the data, split our data into training and testing.
- We built different machine leanring models and tune different hyperparameters using GridSearchCV.
- The link to the notebook Predictive Analysis

Results

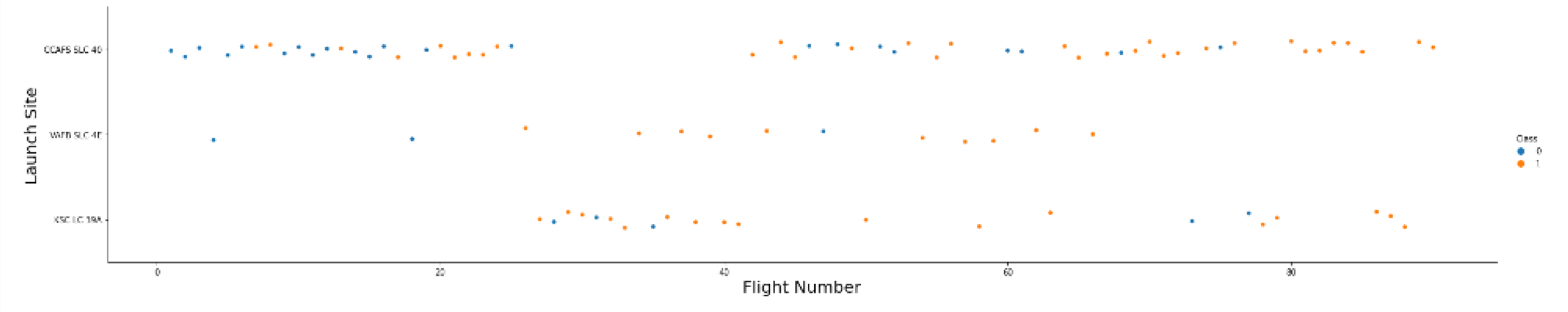
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

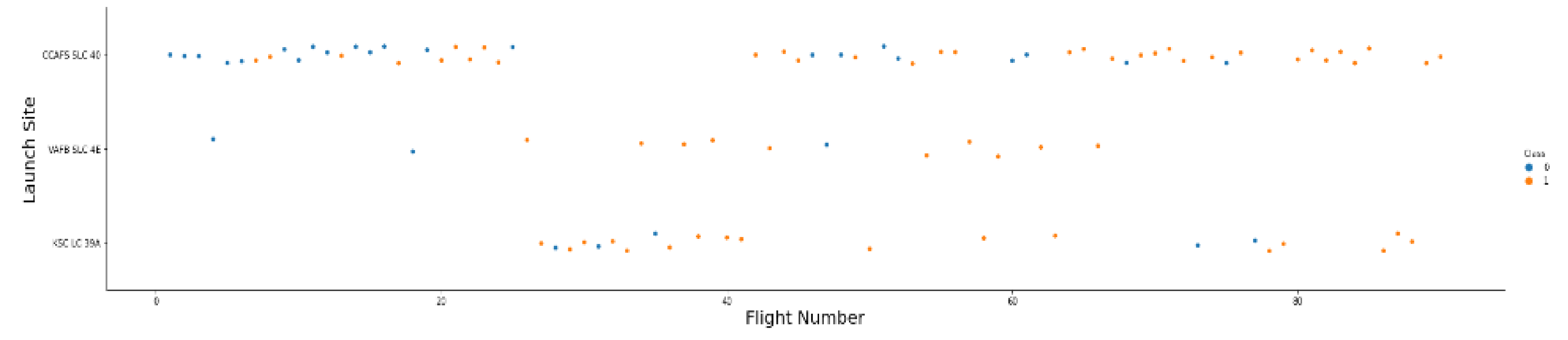
Flight Number vs. Launch Site

- The larger the flight amount at launch site, the greater the success rate at launch site.



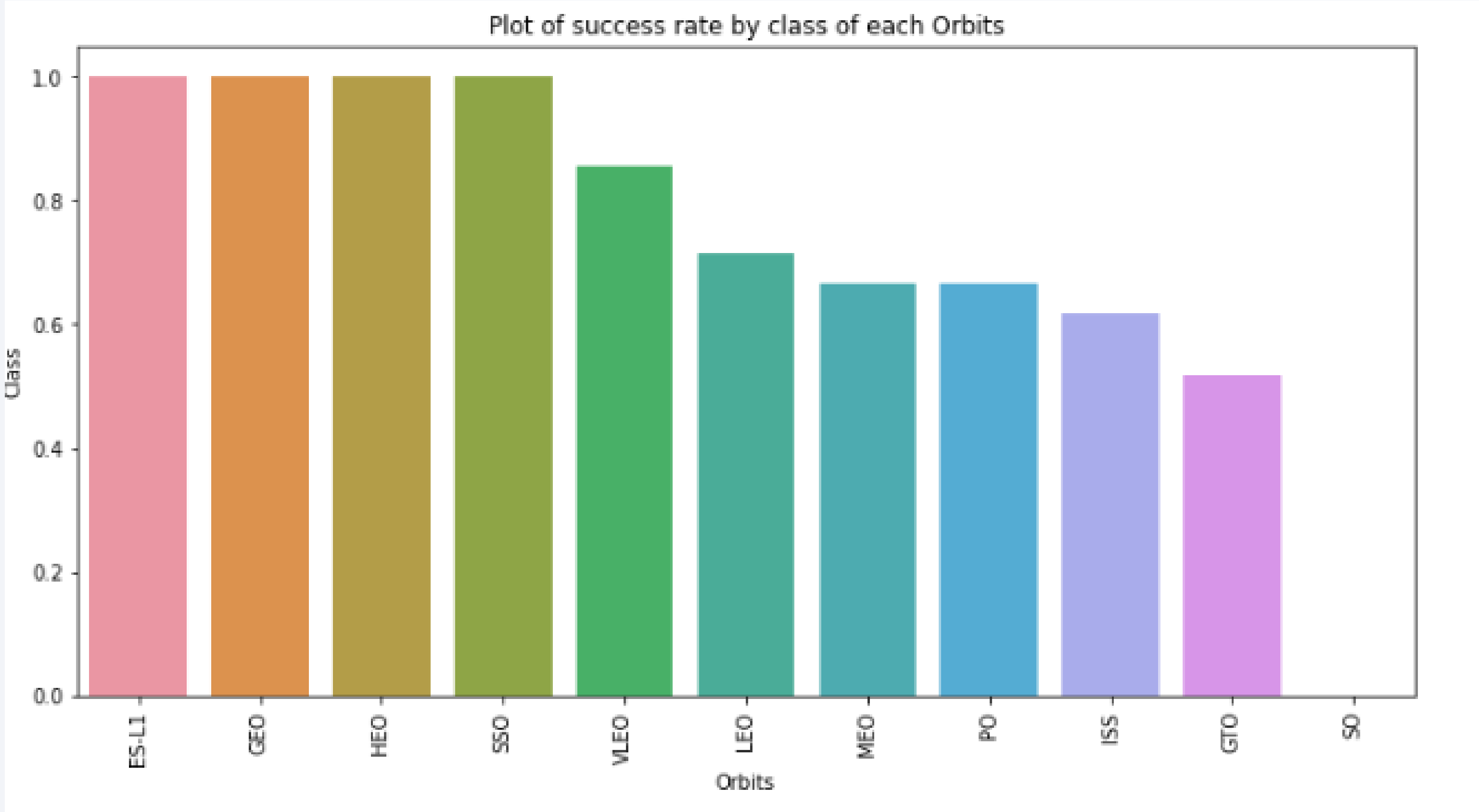
Payload vs. Launch Site

- The greater the payload mass, the higher the success rate



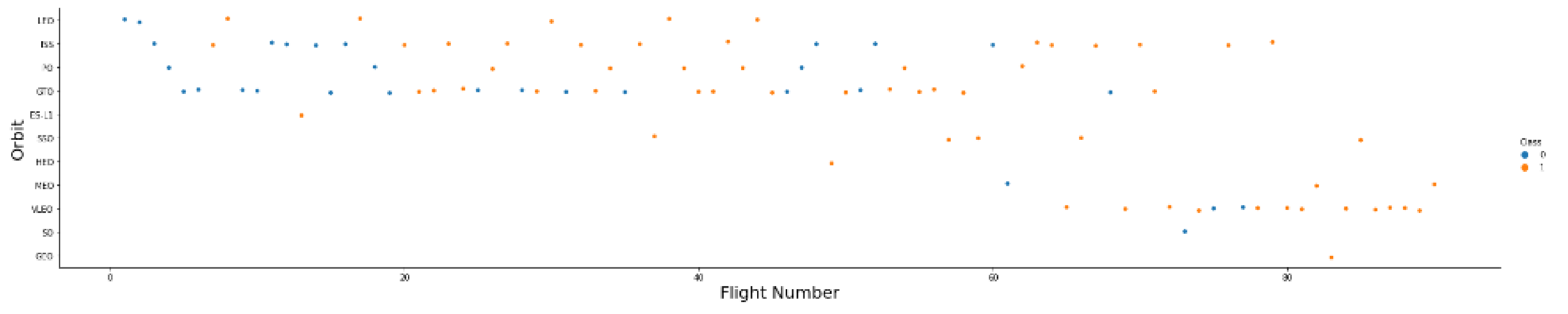
Success Rate vs. Orbit Type

- ES-L1, GEO, HEO, SSO, VLEO had the most success rate.



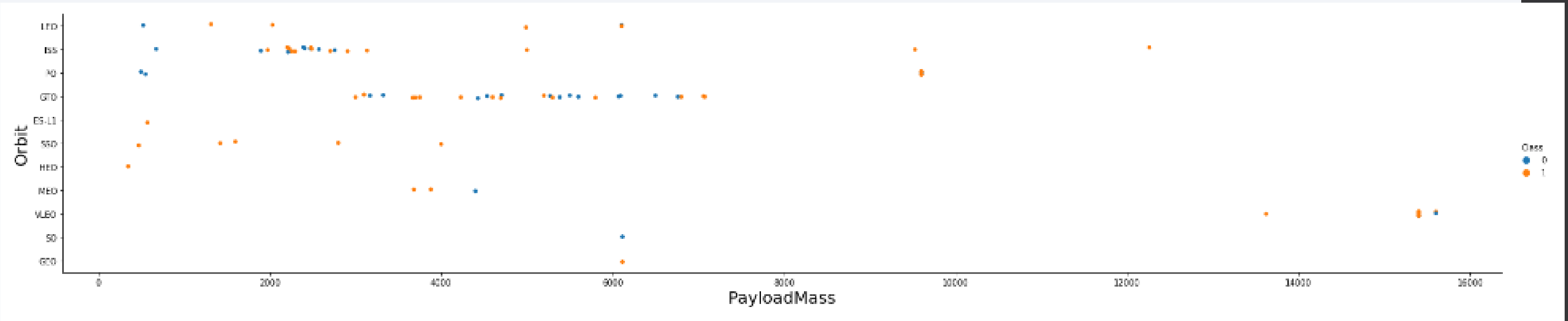
Flight Number vs. Orbit Type

- In the LEO orbit, success is related to the number of flights whereas in the GTO orbit, there is no relationship between flight number and the orbit.



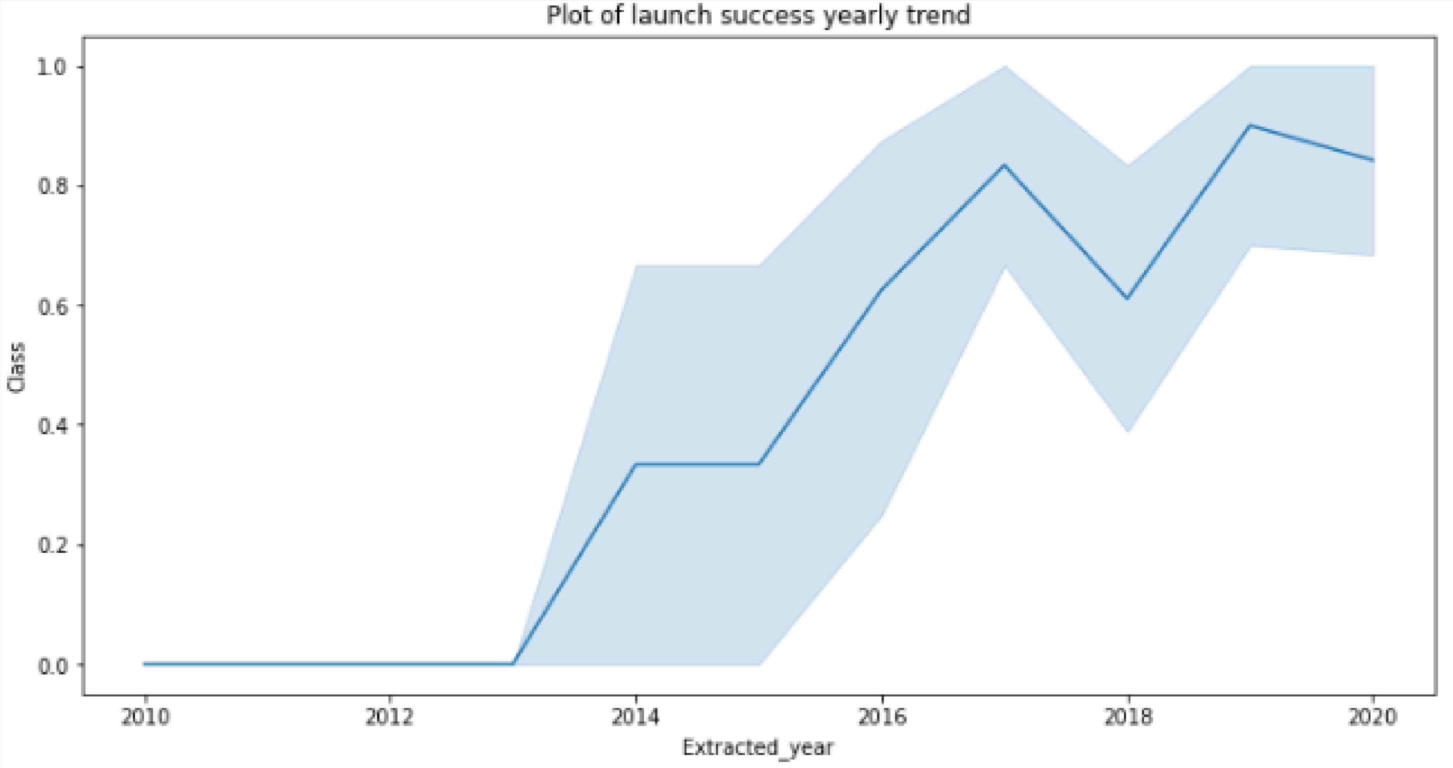
Payload vs. Orbit Type

- With heavy payloads, the successful landing are more for PO, LEO and ISS orbits.



Launch Success Yearly Trend

- The success rate since 2013 kept on increasing till 2020.



All Launch Site Names

- We used the DISTINCT keyword to get the unique launch sites

Task 1

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

In [6]:

```
%sql SELECT DISTINCT LAUNCH_SITE as "Launch_Sites" FROM SPACEX;
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[6]:

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

Launch Site Names Begin with 'CCA'

We used the below query to get the launch sites name begin with CCA

Task 2

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

In [7]:
```
%sql SELECT * FROM SPACEX WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5;
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[7]:

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

Total Payload Mass

We used where clause to get the total payload mass carried by the booster launches

Task 3

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

In [8]:

```
%sql SELECT SUM(PAYLOAD_MASS__KG_) AS "Total Payload Mass by NASA (CRS)" FROM SPACEX WHERE CUSTOMER = 'NASA (CRS)';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[8]:

Total Payload Mass by NASA (CRS)
45596

Average Payload Mass by F9 v1.1

We used where clause in BOOSTER_VERSION to get the payload mass by F9 v1.1

Task 4

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

In [9]:
```
%sql SELECT AVG(PAYLOAD_MASS__KG_) AS "Average Payload Mass by Booster Version F9 v1.1" FROM SPACEX \
WHERE BOOSTER_VERSION = 'F9 v1.1';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[9]:

Average Payload Mass by Booster Version F9 v1.1
2928

First Successful Ground Landing Date

We used where clause with landing_outcome to get the first successful landing outcome in ground pad

Task 5

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

Hint:Use min function

In [10]:
```
%sql SELECT MIN(DATE) AS "First Succesful Landing Outcome in Ground Pad" FROM SPACEX \
WHERE LANDING__OUTCOME = 'Success (ground pad)';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90108kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[10]:

First Succesful Landing Outcome in Ground Pad
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

We used AND condition in PAYLOAD_MASS_KG to get he 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

In [11]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEX WHERE LANDING__OUTCOME = 'Success (drone ship)' \
AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000;
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[11]:

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

Total Number of Successful and Failure Mission Outcomes

We used where clause with MISSION_OUTCOME to get the number of successful and failure missions

Task 7

List the total number of successful and failure mission outcomes

In [12]:
```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Successful Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Success%';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[12]:

Successful Mission
100

In [13]:
```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Failure Mission" FROM SPACEX WHERE MISSION_OUTCOME LIKE 'Failure%';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[13]:

Failure Mission
1

In [15]:
```
%sql SELECT COUNT(MISSION_OUTCOME) AS "Total Number of Successful and Failure Mission" FROM SPACEX \
WHERE MISSION_OUTCOME LIKE 'Success%' OR MISSION_OUTCOME LIKE 'Failure%';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[15]:

Total Number of Successful and Failure Mission
101

In [14]:
```
%sql SELECT sum(case when MISSION_OUTCOME LIKE '%Success%' then 1 else 0 end) AS "Successful Mission", \
    sum(case when MISSION_OUTCOME LIKE '%Failure%' then 1 else 0 end) AS "Failure Mission" \
FROM SPACEX;
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[14]:

Successful Mission	Failure Mission
100	1

Boosters Carried Maximum Payload

We used WHERE clause with PAYLOAD_MASS_KG and inline nested select function to get the maximum payload.

Task 8

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

In [16]:
```
%sql SELECT DISTINCT BOOSTER_VERSION AS "Booster Versions which carried the Maximum Payload Mass" FROM SPACEX \
WHERE PAYLOAD_MASS__KG_ =(SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEX);
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90108kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[16]:

Booster Versions which carried the Maximum Payload Mass
F9 B5 B1048.4
F9 B5 B1048.5
F9 B5 B1049.4
F9 B5 B1049.5
F9 B5 B1049.7
F9 B5 B1051.3
F9 B5 B1051.4
F9 B5 B1051.6
F9 B5 B1056.4
F9 B5 B1058.3
F9 B5 B1060.2
F9 B5 B1060.3

2015 Launch Records

We used WHERE and LIKE clause in LAUNCH_OUTCOME to get the launch records.

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

In [17]:
```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE DATE LIKE '2015-%' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[17]:

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

In [18]:
```
%sql SELECT BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE year(DATE) = '2015' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[18]:

booster_version	launch_site
F9 v1.1 B1012	CCAFS LC-40
F9 v1.1 B1015	CCAFS LC-40

In [19]:
```
%sql SELECT month(DATE) as Month, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE year(DATE) = '2015' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[19]:

MONTH	booster_version	launch_site
1	F9 v1.1 B1012	CCAFS LC-40
4	F9 v1.1 B1015	CCAFS LC-40

In [20]:
```
%sql SELECT {fn MONTHNAME(DATE)} as "Month", BOOSTER_VERSION, LAUNCH_SITE FROM SPACEX WHERE year(DATE) = '2015' AND \
LANDING__OUTCOME = 'Failure (drone ship)';
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[20]:

Month	booster_version	launch_site
January	F9 v1.1 B1012	CCAFS LC-40
April	F9 v1.1 B1015	CCAFS LC-40

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

We used WHERE clause with DATE and GROUP BY with LANDING_OUTCOME and ORDER BY to get the landing outcome between 2010-06-04 and 2017-03-20.

Task 10

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 des

In [21]:

```
%sql SELECT LANDING__OUTCOME as "Landing Outcome", COUNT(LANDING__OUTCOME) AS "Total Count" FROM SPACEX \
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' \
GROUP BY  LANDING__OUTCOME \
ORDER BY COUNT(LANDING__OUTCOME) DESC ;
```

```
 * ibm_db_sa://ddr13828:***@ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31505/bludb
Done.
```

Out[21]:

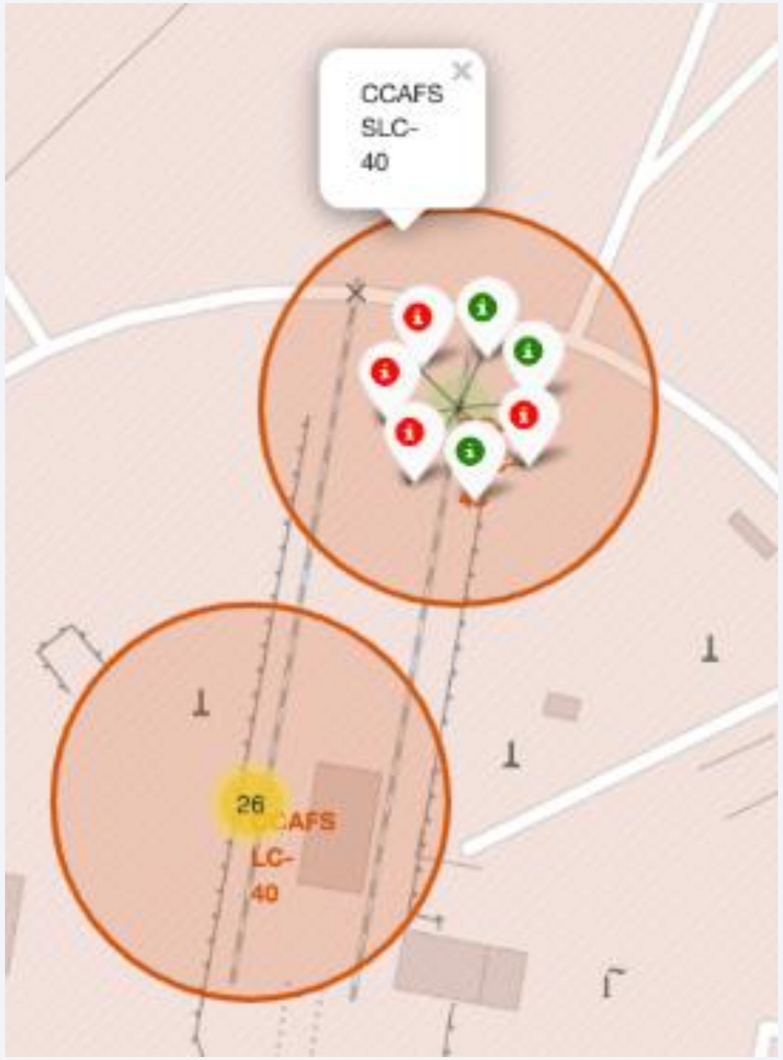
Landing Outcome	Total Count
No attempt	10
Failure (drone ship)	5
Success (drone ship)	5
Controlled (ocean)	3
Success (ground pad)	3
Failure (parachute)	2
Uncontrolled (ocean)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

All Launch Sites Global map marker

Markers showing launch sites with color labels



Map symbols

A railway map symbol may look like this:

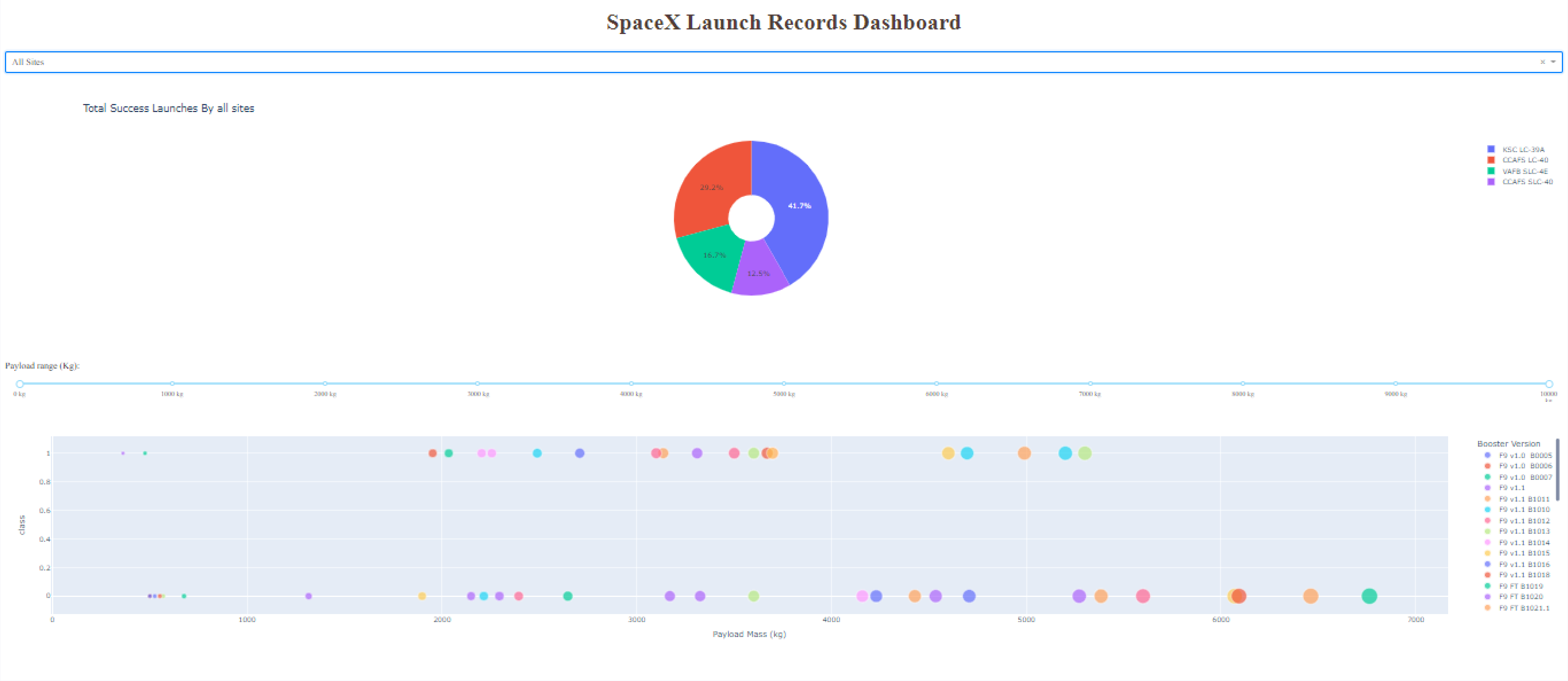
A highway map symbol may look like this:

A city map symbol may look like this:

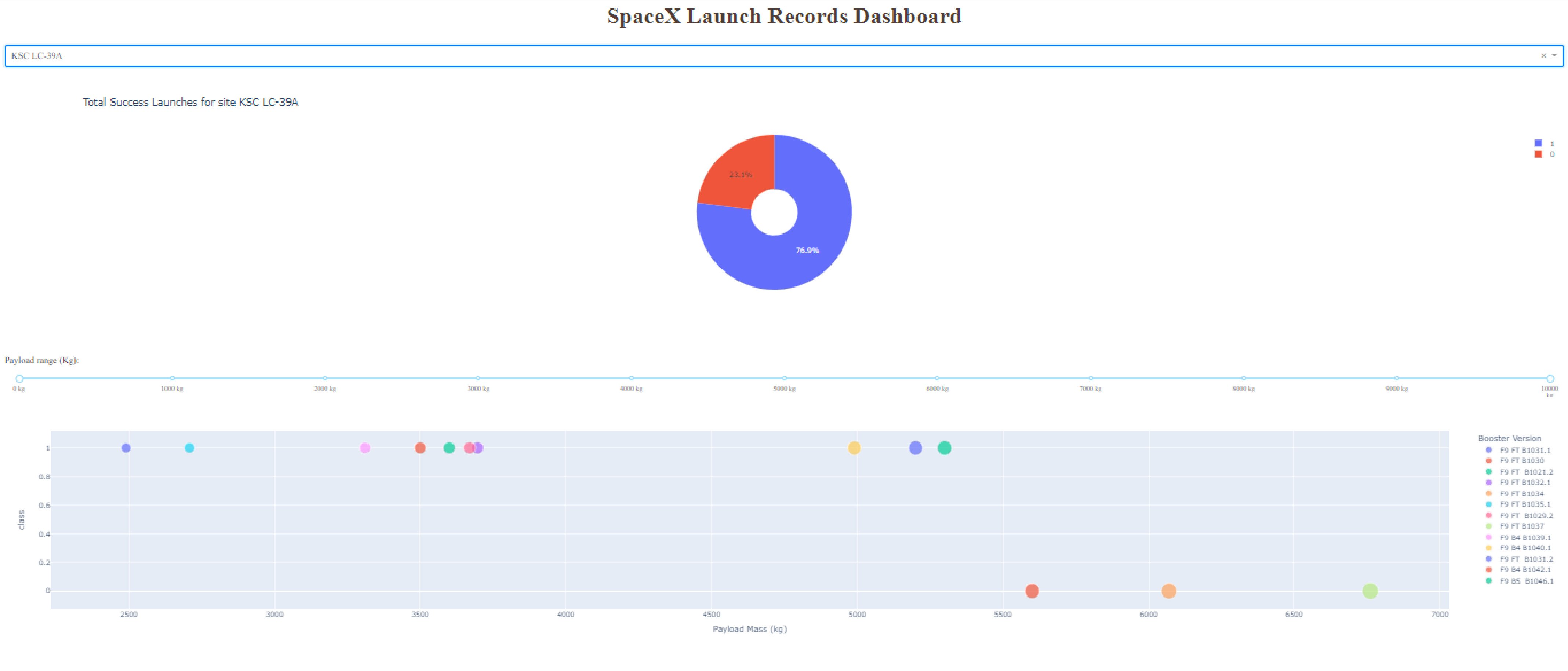
Section 4

Build a Dashboard with Plotly Dash

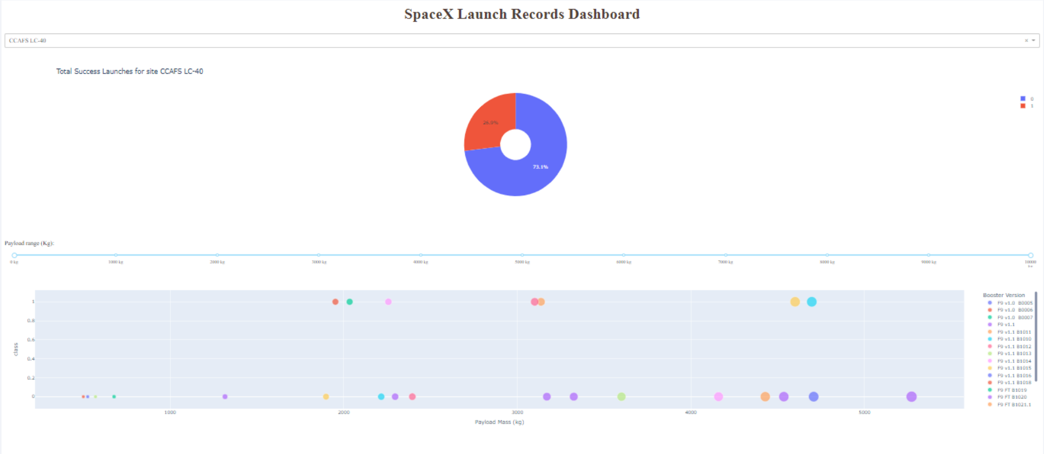
Pie chart showing the success percentage achieved by each launch site

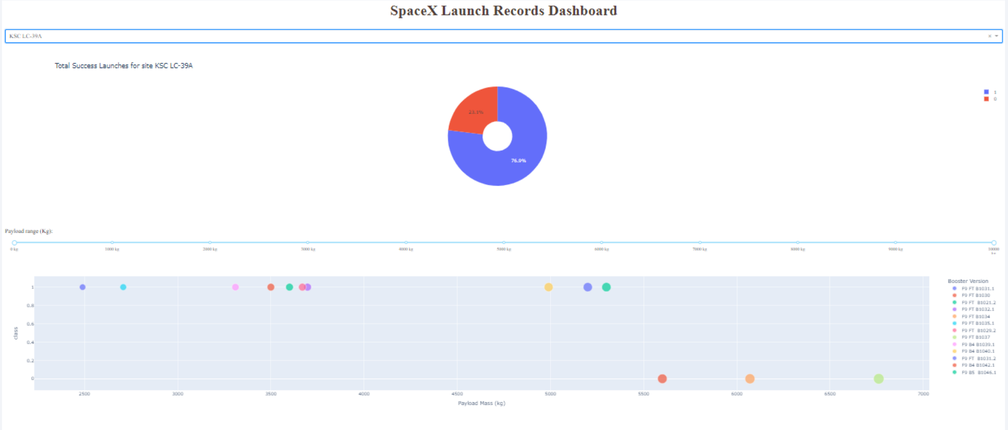


Pie chart showing the Launch site with the highest launch success ratio



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





Section 5

Predictive Analysis (Classification)

Classification Accuracy

- The decision tree classifier provides highest classification accuracy

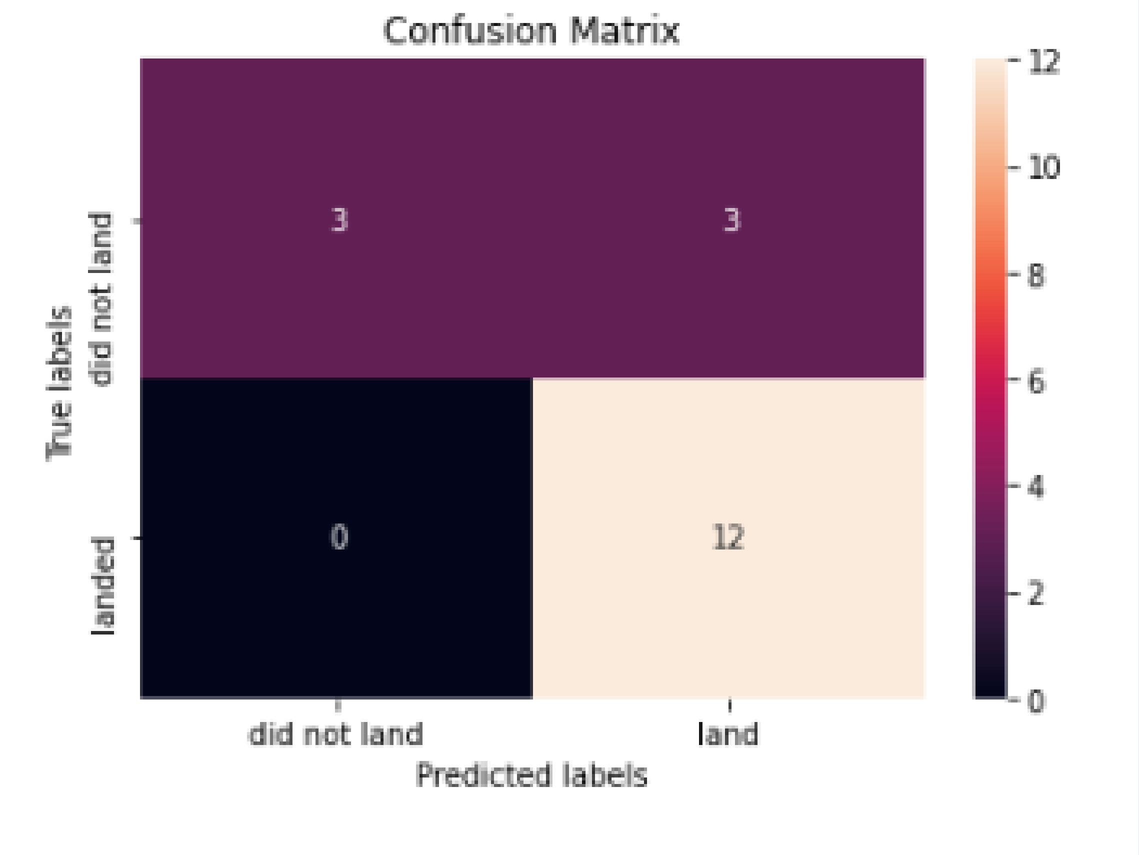
```
models = {'KNeighbors':knn_cv.best_score_,
              'DecisionTree':tree_cv.best_score_,
              'LogisticRegression':logreg_cv.best_score_,
              'SupportVector': svm_cv.best_score_}

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 0.875
Best params is : {'criterion': 'entropy', '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.



Conclusions

- We concluded that…
  - The success rate depends upon the flight amount at launch site.
  - Launch success rate started to increase in 2013 till 2020.
  - Orbits ES-L1, GEO, HEP, SSO, VLEO had the most success rate.
  - KCS LC-39A had the most successful launches of any sites.
  - The decision tree classifier is the best machine learning algorithm for the task.

Appendix

- Include any relevant assets like Python code snippets, SQL queries, charts, Notebook outputs, or data sets that you may have created during this project

Thank you!