IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Jure Banovšek
7th January 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
  - 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 need 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 SpaceX API
  - Response was than decoded as json using .json() function and than turned into a pandas dataframe using .json_normalize()
  - Cleaning the data: checking for missing values and filling in missing values
  - Web scraping from Wikipedia for Falcon 9 launch records with BeautifulSoup
  - Objective was to extract the launch records as HTML table, parse it and convert it to pandas dataframe 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/baulee56/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-spacex-data-collection-api.ipynb

1. Get request for rocket launch data using API

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

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

2. Use json_normalize method to convert json result to dataframe

In [12]:
```
# Use json_normalize method to convert the json result into a dataframe

# decode response content as json
static_json_df = res.json()
```

In [13]:
```
# apply json_normalize
data = pd.json_normalize(static_json_df)
```

3. We then performed data cleaning and filling in the missing values

In [30]:
```
rows = data_falcon9['PayloadMass'].values.tolist()[0]

df_rows = pd.DataFrame(rows)
df_rows = df_rows.replace(np.nan, PayloadMass)

data_falcon9['PayloadMass'][0] = df_rows.values
data_falcon9
```

Data Collection - Scraping

- We applied web scraping to get Falcon 9 launch records with BeautifulSoup
- We parsed table and converted it to a pandas dataframe.
- The link to the notebook is:

https://github.com/baulee56/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-webscraping.ipynb.ipynb

1. Apply HTTP Get method to request the Falcon 9 rocket launch page

In [4]:
```
static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

In [5]:
```
# 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[5]:
```
200
```

2. Create a BeautifulSoup object from the HTML response

In [6]:
```
# 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 [7]:
```
# Use soup.title attribute
soup.title
```

Out[7]:
```
<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

3. Extract all column names from the HTML table header

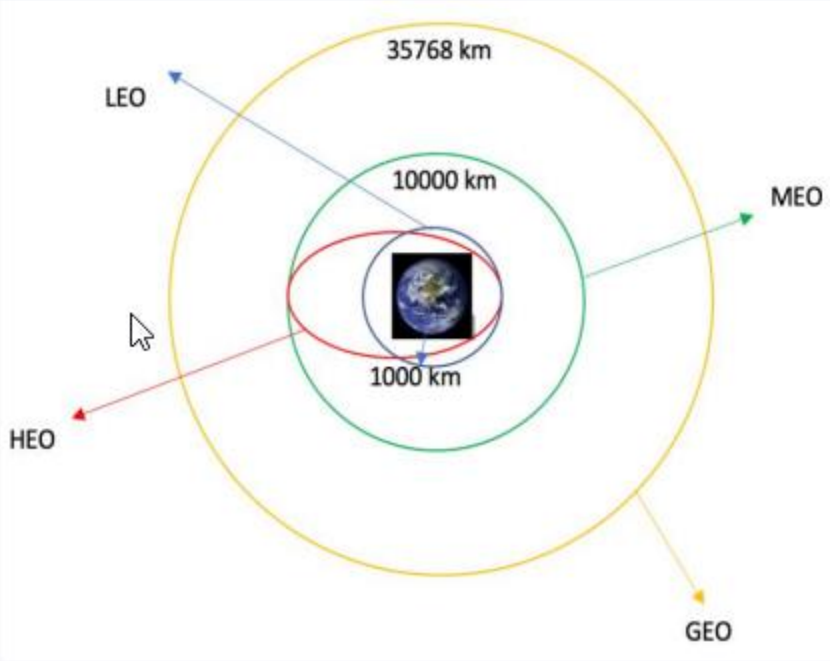
In [10]:
```
column_names = []

# Apply find_all() function with `th` element on first_launch_table
# Iterate each th element and apply the provided extract_column_from_header() to get a column name
# Append the Non-empty column name (`if name is not None and len(name) > 0`) into a list called column_names

element = soup.find_all('th')
for row in range(len(element)):
    try:
        name = extract_column_from_header(element[row])
        if (name is not None and len(name) > 0):
            column_names.append(name)
    except:
        pass
```

4. Create a dataframe by parsing the launch HTML tables
5. Export data to csv

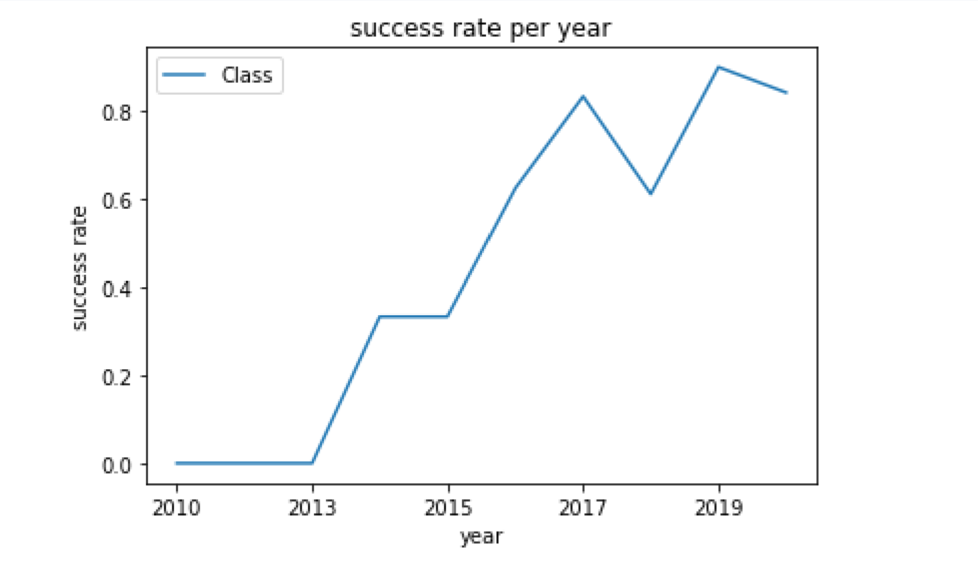
Data Wrangling



- We performed exporatory data analysis and determined the training labels
- We calculated the number of launches at each site and the number of occurrence at each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the notebook is:

https://github.com/baulee56/IBM-Applied-Data-Science-Capstone/blob/main/labs-jupyter-spacex-Data%20wrangling.ipynb.ipynb

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 and the launch success yearly trend.
- The link to the notebook is:

https://github.com/baulee56/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-dataviz.ipynb.ipynb

EDA with SQL

- We 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:
  - 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 failure 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/baulee56/IBM-Applied-Data-Science-Capstone/blob/main/jupyter-labs-eda-sql-coursera.ipynb.ipynb

Build an Interactive Map with Folium

- On the folium map 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.
- We assigned the feature launch outcomes (failure or success) to class 0 and 1
- Using the color-labeled marker clusters we identified which launch sites have relatively high success rate.
- We calculated the distance between a launch site to its proximities. We answered some questions:
    - Are launch sites near railways, highways and coastlines?
    - Do launch sites keep certain distance away from cities?

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 scatter graph showing the relationship between Outcome and Payload Mass (kg) for the different booster version.
- The link to the notebook is:

https://github.com/baulee56/IBM-Applied-Data-Science-Capstone/blob/main/spacex_dash_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.
- The link to the notebook is:

https://github.com/baulee56/IBM-Applied-Data-Science-Capstone/blob/master/SpaceX_Machine%20LearningPrediction_Part_5.ipynb.ipynb

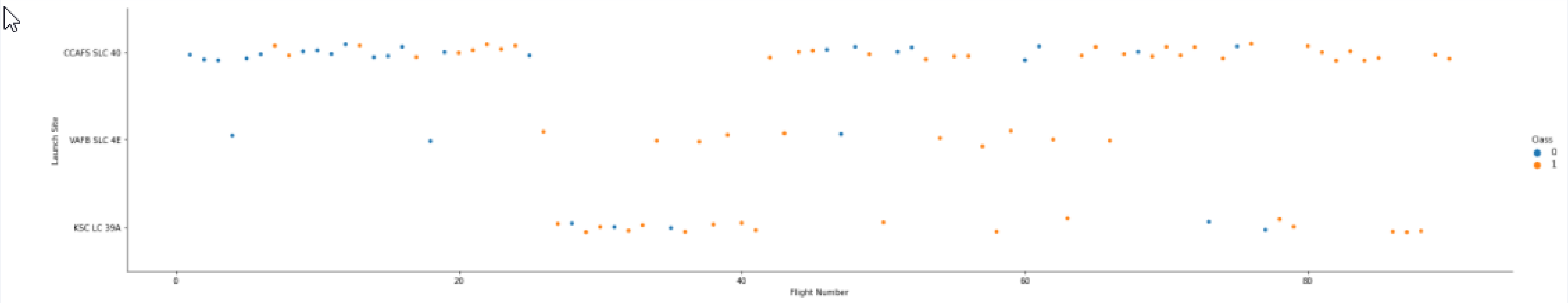
Results

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

Section 2

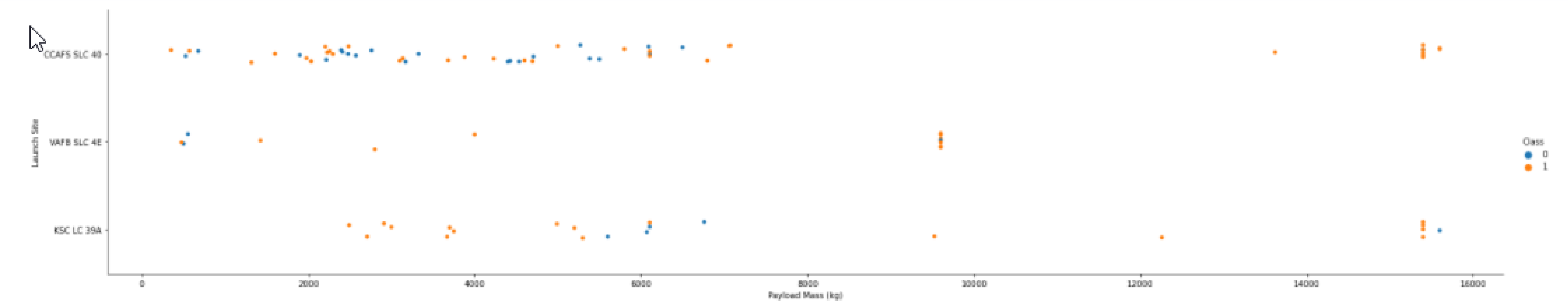
Insights drawn from EDA

Flight Number vs. Launch Site



- We found that the larger the flight number at a launch site, the greater the success rate at a launch site.

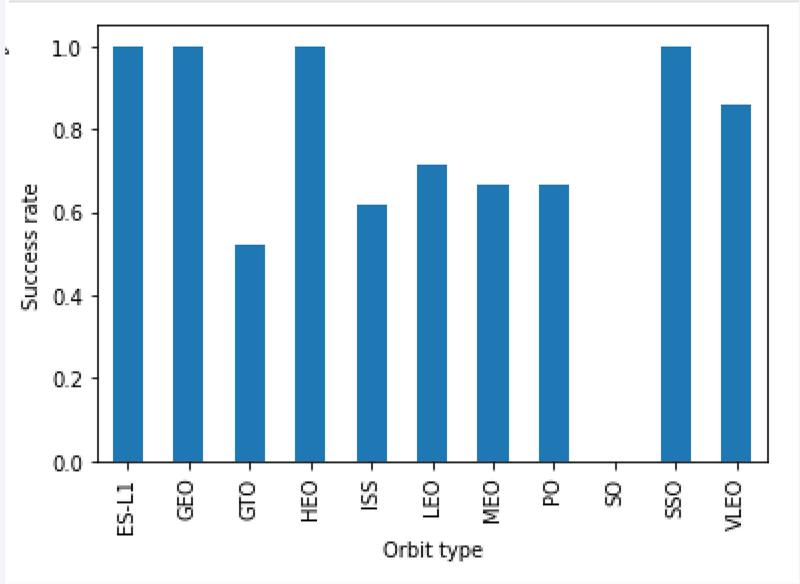
Payload vs. Launch Site



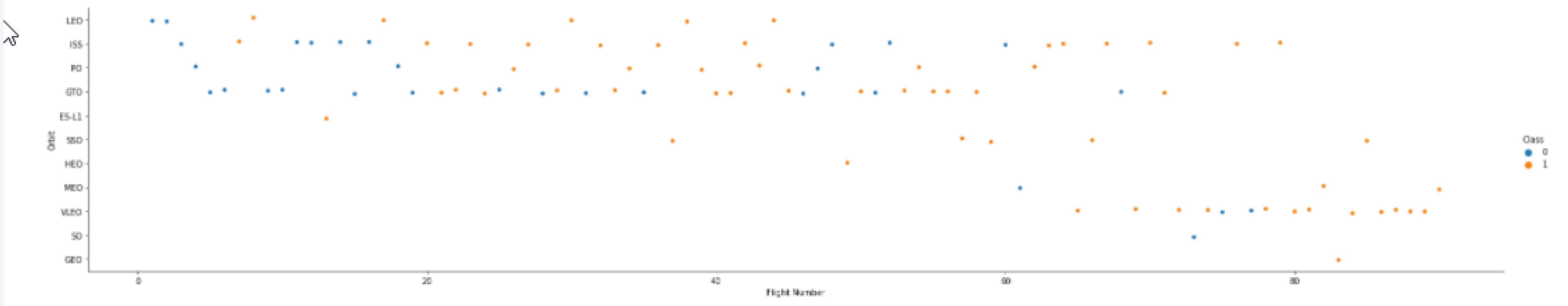
- We found out that the greater the payload mass for launch site CCAFS 40, the higher the success rate for the rocket

Success Rate vs. Orbit Type

- We can see from the plot that ES-L1, GEO, HEO, SSO nad VLEO had the most success rate.

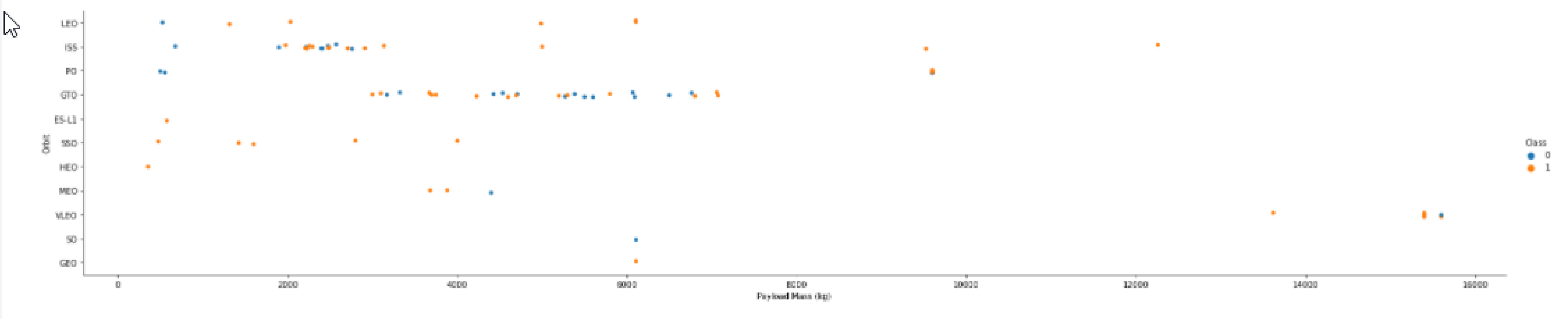


Flight Number vs. Orbit Type



- We can see 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.

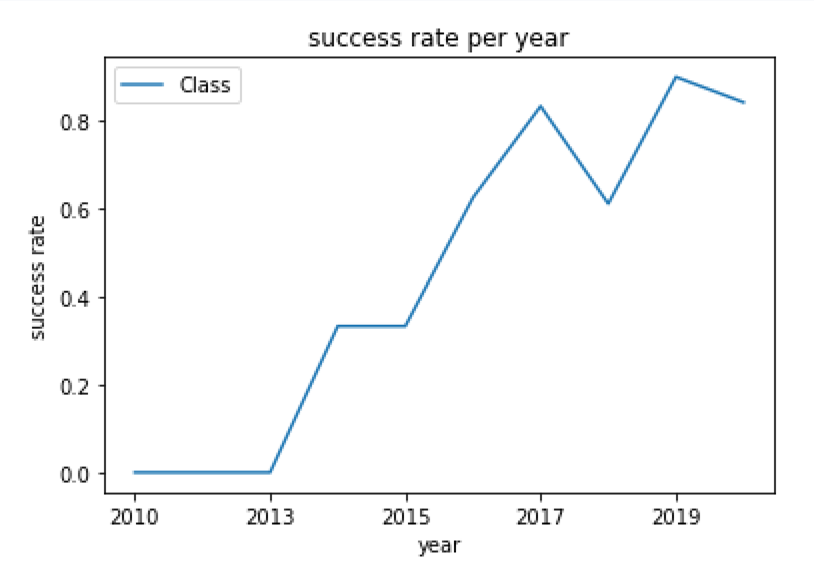
Payload vs. Orbit Type



- We can see that for heavy payloads, the success rate is in PO, LEO and ISS orbits.

Launch Success Yearly Trend

- We can see that success rate is increasing from 2013 on until 2020.



All Launch Site Names

- We used the key word DISTINCT to show only uniqe launch sites from SpaceX data.

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

In [8]:
```
%%sql
SELECT DISTINCT LAUNCH_SITE
FROM SPACEXTBL;
```

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb?authSou
replset
Done.
```

Out[8]:

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

Launch Site Names Begin with 'CCA'

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

In [9]:

```
%%sql
SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'CCA%' LIMIT 5
```

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb?authSource=admin&replicaSet=replset
Done.
```

Out[9]:

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

- 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 this query:

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

In [16]:

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

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb?authSource=a
replset
Done.
```

Out[16]:

1
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2534 using this query:

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

In [19]:
```
%%sql
SELECT AVG(PAYLOAD_MASS__KG_)
FROM SPACEXTBL
WHERE booster_version LIKE 'F9 v1.1%'
```

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/b
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/b
replset
Done.
```

Out[19]:

1
2534

First Successful Ground Landing Date

- We can see that the first successful landing outcome on ground pad was on 22nd December 2015.

In [23]:
```
%%sql
SELECT MIN(DATE)
FROM SPACEXTBL
WHERE landing__outcome = 'Success (ground pad)'
```

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb?authSource=admin&replicaSet=replset
Done.
```

Out[23]:

1
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used **WHERE** clause to filter boosters which have successfully landed on the drone ship and applied the **AND** condition to determine successful landing with payload mass between 4000 and 6000 kg.

List the names of the boosters which have success in drone

In [29]:
```
%%sql
SELECT booster_version
FROM SPACEXTBL
WHERE landing__outcome = 'Success (drone ship)'
AND 4000 < payload_mass__kg_ < 6000
```

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-
replset
Done.
```

Out[29]:

booster_version
F9 FT B1021.1
F9 FT B1023.1
F9 FT B1029.2
F9 FT B1038.1
F9 B4 B1042.1
F9 B4 B1045.1
F9 B5 B1046.1

Total Number of Successful and Failure Mission Outcomes

- We used COUNT(MissionOutcome) to count all outcomes and than grouped them by outcom using GROUP BY. We can see that 99 flights were successful.

List the total number of successful and failure mission outcomes

In [32]:

```
%%sql
SELECT MISSION_OUTCOME, COUNT(MISSION_OUTCOME) as Total
FROM SPACEXTBL
GROUP BY MISSION_OUTCOME;
```

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31929/bludb?authSource=
replset
Done.
```

Out[32]:

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

Boosters Carried Maximum Payload

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

In [38]:
```
%%sql
SELECT DISTINCT booster_version
FROM SPACEXTBL
WHERE payload_mass__kg_ = (SELECT MAX(payload_mass__kg_) FROM SPACEXTBL)
```

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8l
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8l
replset
Done.
```

Out[38]:

booster_version
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

- We used MAX(payolad_mass__kg_) as subquery to determine which boosters have carried the maximum payload.

2015 Launch Records

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

In [42]:

```
%%sql
SELECT landing__outcome, booster_version, launch_site
FROM SPACEXTBL
WHERE landing__outcome = 'Failure (drone ship)'
AND DATE LIKE '2015%'
```

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.
replset
Done.
```

Out[42]:

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

- With query **WHERE landing__outcome = „Failure (drone ship)“** we found all the failed landing outcomes in drone ship, and than also limit it for year 2015 using **DATE LIKE „2015%“**

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

- We used COUNT and GROUP BY to get all types od landing outcomes and than filter it using BETWEEN.
- We than ordered the result in descending order using **ORDER BY** Total **DESC**.

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

In [46]:

```
%%sql
SELECT landing__outcome, COUNT(landing__outcome) AS Total
FROM SPACEXTBL
WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20'
GROUP BY landing__outcome
ORDER BY Total DESC
```

```
 * ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od
   ibm_db_sa://byt63339:***@55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od
replset
Done.
```

Out[46]:

landing__outcome	total
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

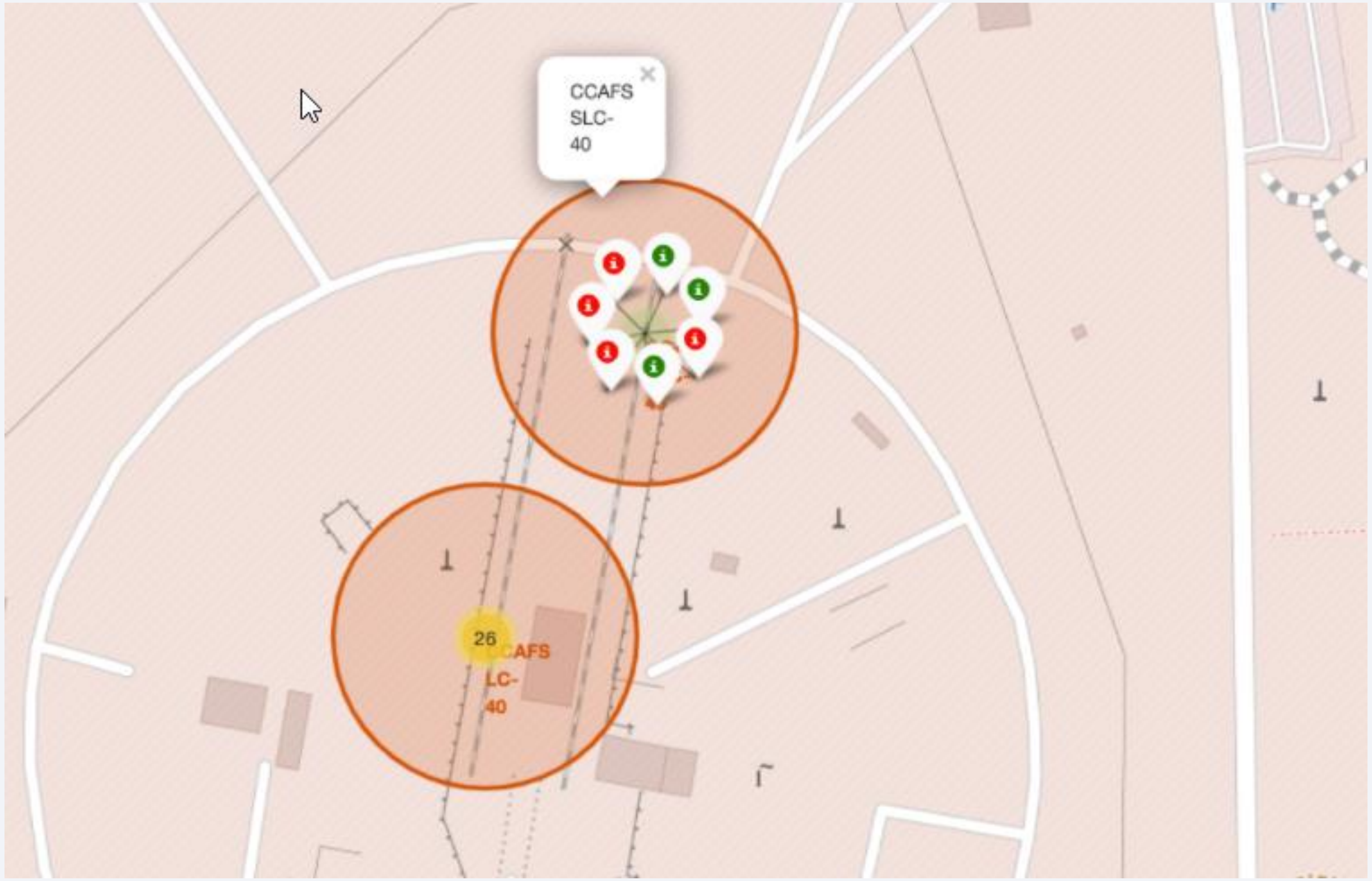
Section 4

Launch Sites Proximities Analysis

All launch sites global map markers

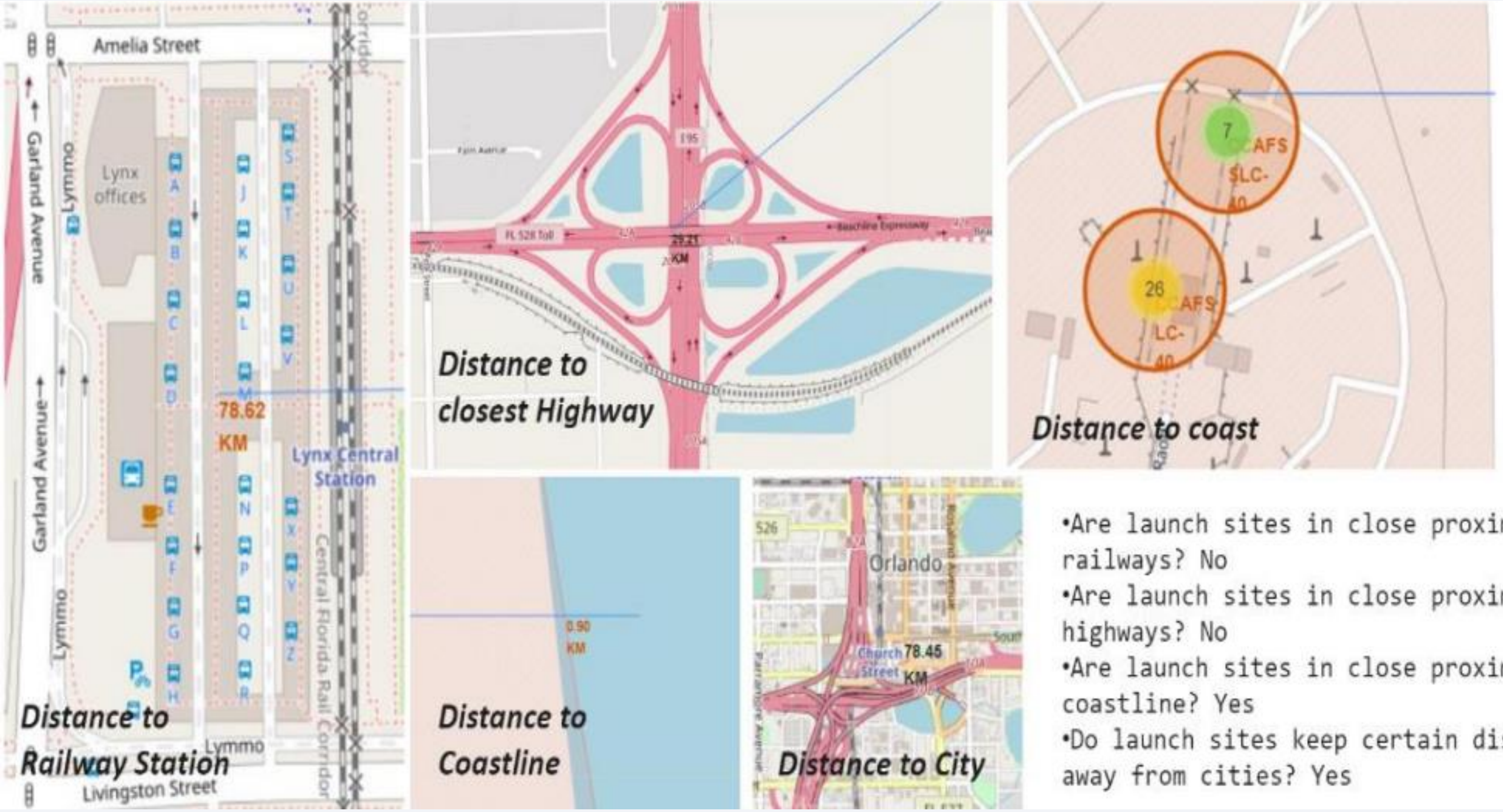
- We can see that the SpaceX launch sites are in the USA coastal area Florida and California.
- We can also see that all lauch sites are near equator.

Markers showing launch sites with color labels



- Green marker shows successful launches and red marker shows failures

Launch Site distance to landmarks

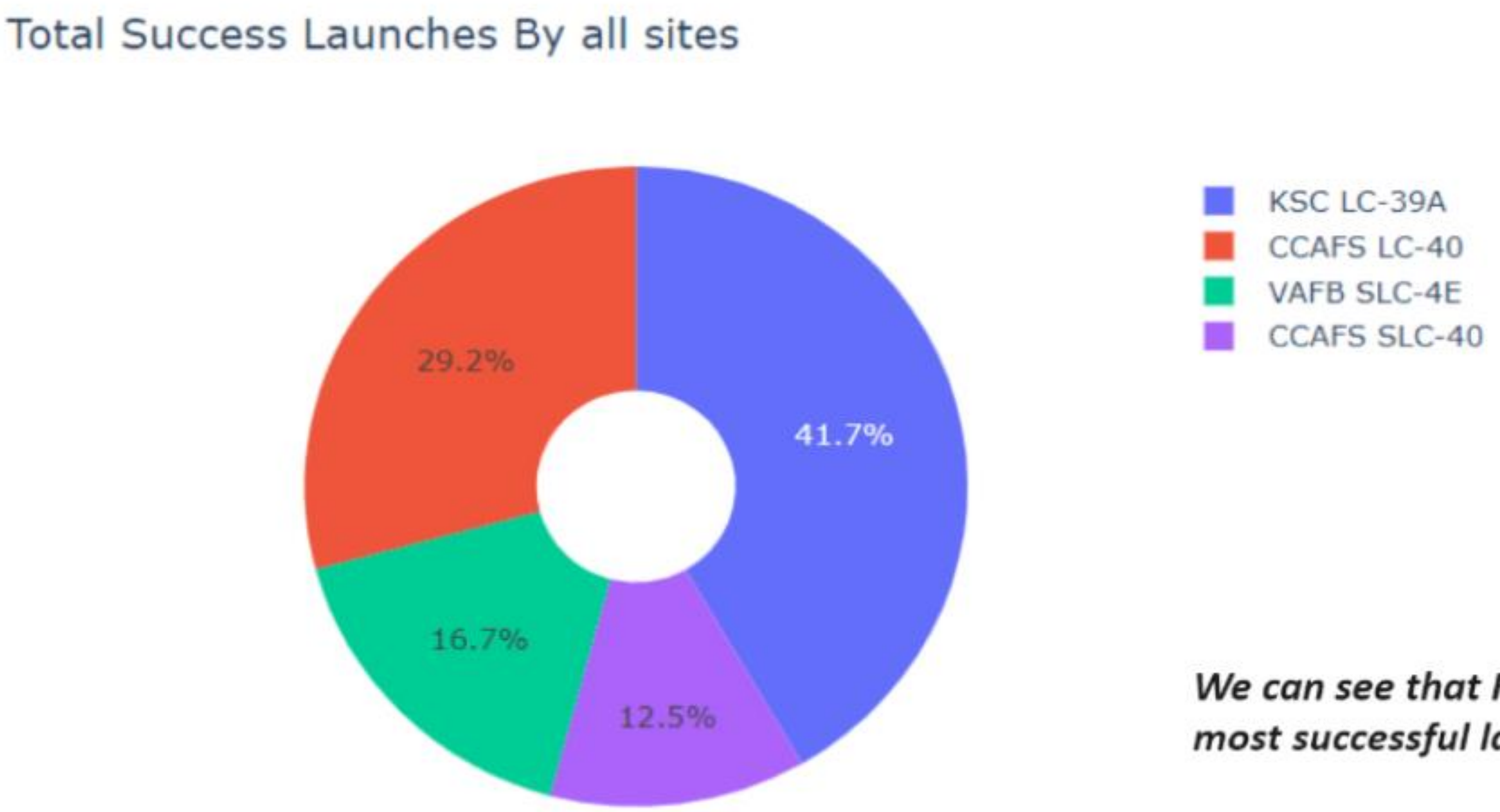


•Are launch sites in close proximity to railways? No
•Are launch sites in close proximity to highways? No
•Are launch sites in close proximity to coastline? Yes
•Do launch sites keep certain distance away from cities? Yes

Section 5

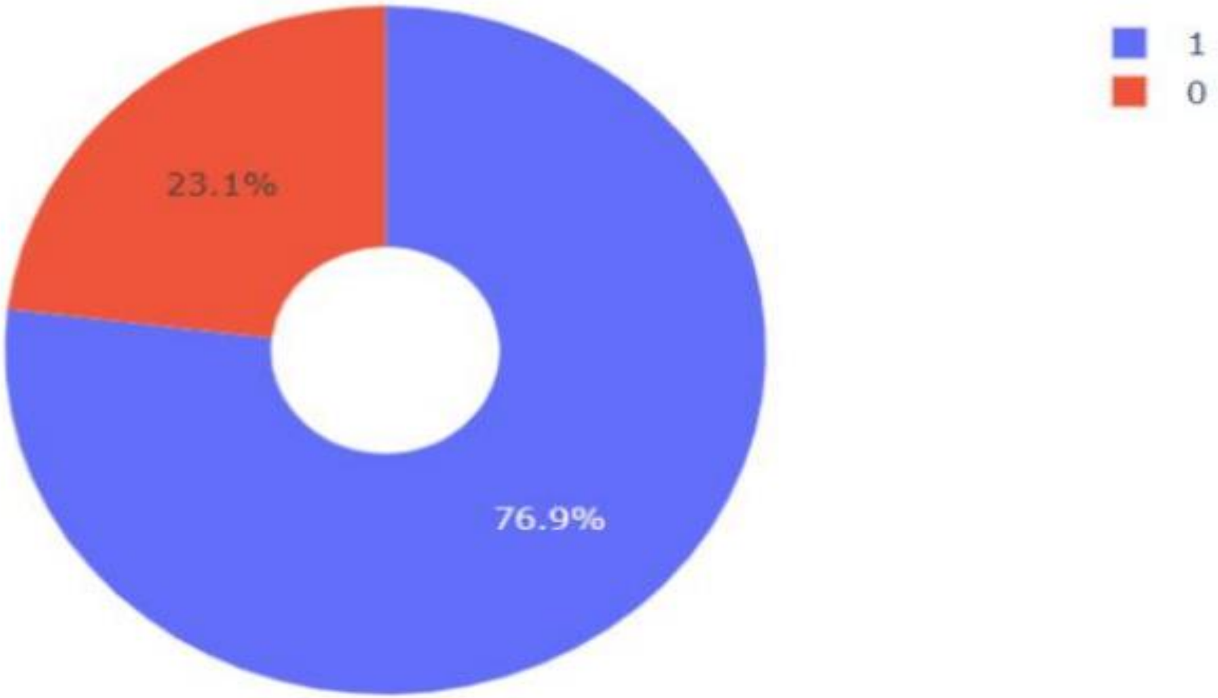
Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site



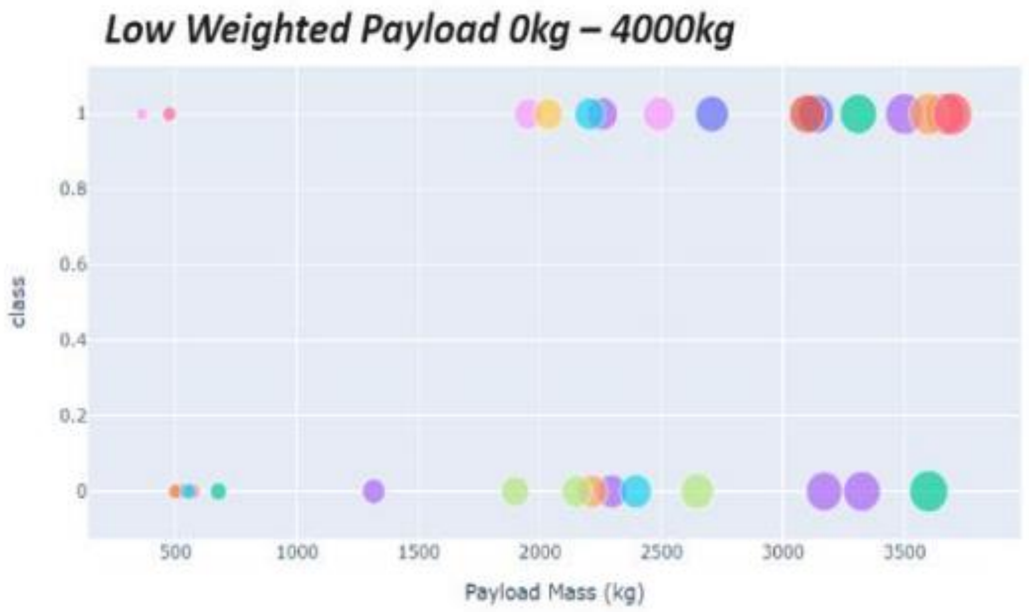
We can see that KSC LC-39A had the most successful launches from all the sites

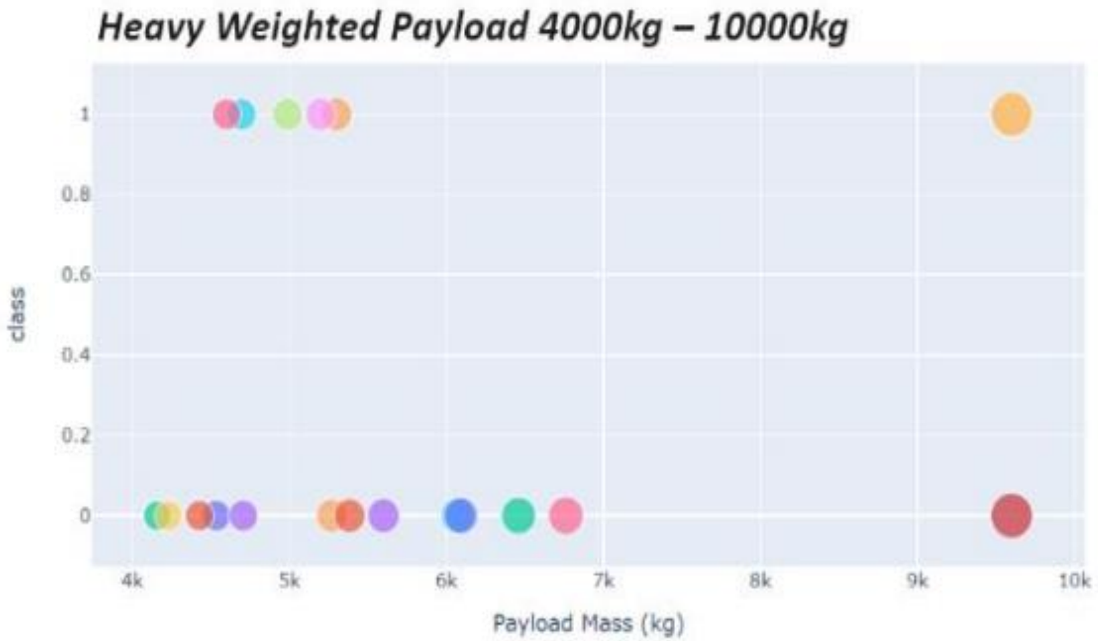
Pie chart showin the Launch site with the highest launch success rate



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 6

Predictive Analysis (Classification)

Classification Accuracy

```
In [20]:
parameters = {'criterion': ['gini', 'entropy'],
     'splitter': ['best', 'random'],
     'max_depth': [2*n for n in range(1,10)],
     'max_features': ['auto', 'sqrt'],
     'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10]}

tree = DecisionTreeClassifier()
```

```
In [21]:
tree_cv = GridSearchCV(tree, parameters, cv=10)
tree_cv.fit(X_train,Y_train)
```

```
Out[21]: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'splitter': ['best', 'random']})
```

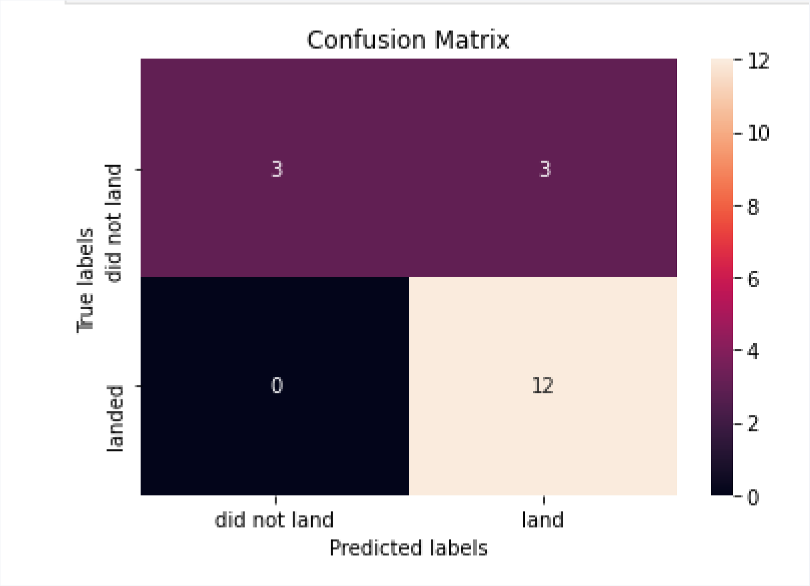
```
In [22]:
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
print("accuracy :",tree_cv.best_score_)
```

```
tuned hpyerparameters :(best parameters)  {'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2,
'splitter': 'best'}
accuracy : 0.8892857142857145
```

- The decicion three classifier is the model with the highest classification accuracy

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

- The larger the flight amount at launch site, the greater the success rate at launch site
- Launch success rate started to increase in 2013
- Orbits ES-L1, GEO, HEO, SSO, VLEO had the most success rate.
- KSC LC-39A had the most successful launches of all sites
- The Decision tree classifier is the best machine learning algorithm for this task.

Thank you!