IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Kadir Baver Kerimoğlu
16th April 2022

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Summary of methodologies
  - Data Collection with 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 Dash Plotly
  - Predictive Analysis 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:
  - Data was collected using SpaceX Rest API and Web Scraping from Wikipedia.
- Perform data wrangling
  - One hot encoding was applied to categorical columns for machine learning prediction and dropping irrelevant columns.
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
  - Build, tune, evaluate classification models

Data Collection

- Data sets was collected as follows;

    - Performed data collection by using get request the SpaceX Rest API.

    - Next step, we decoded the response content as a Json by calling .json() function and turn it into a pandas dataframe by using .json_normalize().

    - We then cleaned the data, checked for missing values and fill in missing values where necessary.

    - Moreover, 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

- We used the get request to the SpaceX API to collect data, clean the requested data and did some basic data wrangling and formatting.

- GitHub URL is as follow; https://github.com/kbkerimoglu/kbkerimoglu.github.io/blob/master/IBM_Data_Science_Capstone_Project_SpaceX/Data%20Collection%20with%20API.ipynb

1. Get request for rocket launches data from SpaceX API with the following URL.

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

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

2. Decoded the response content as a Json using .json() and turn it into a Pandas dataframe using .json_normalize().

```
In [14]: # Use json_normalize meethod to convert the json result into a dataframe
         # Decode response content as a Json
         static_json_df = res_url.json()

In [15]: # Apply json_normalize to turn into df
         data = pd.json_normalize(static_json_df)
```

3. Then performed data cleaning and filling the missing values with mean.

```
In [33]: # Replace the np.nan values with its mean value
         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 scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- https://github.com/kbkerimoglu/kbkerimoglu.github.io/blob/master/IBM_Data_Science_Capstone_Project_SpaceX/Data%20Collection%20with%20Web%20Scraping.ipynb

1. Scraped data from snapshot of the Falcon 9 rocket launch page from Wikipedia.

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

2. Performed an HTTP GET method to request the Falcon 9 launch HTML page.

```
In [5]: # use requests.get() method with the provided static_url
        # assign the response to a object
        data_html = requests.get(static_url)
        data_html.status_code

Out[5]: 200
```

3. Create BeatifulSoup object from the HTML response.

```
In [6]: # Use BeautifulSoup() to create a BeautifulSoup object from a response text content
        soup = BeautifulSoup(data_html.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>
```

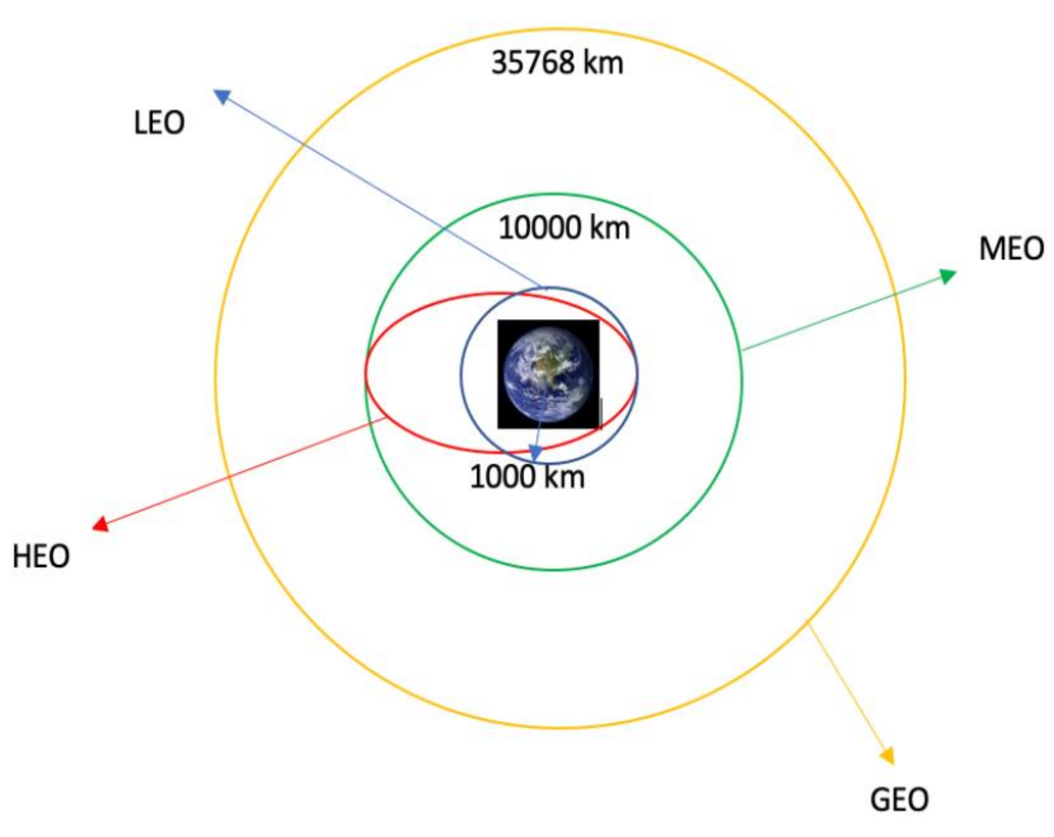
4. Extracted all column names one by one from HTML table reader.

```
In [11]: 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
```

5. Created a dataframe by parsing the launch HTML tables.

6. Exported data to csv.

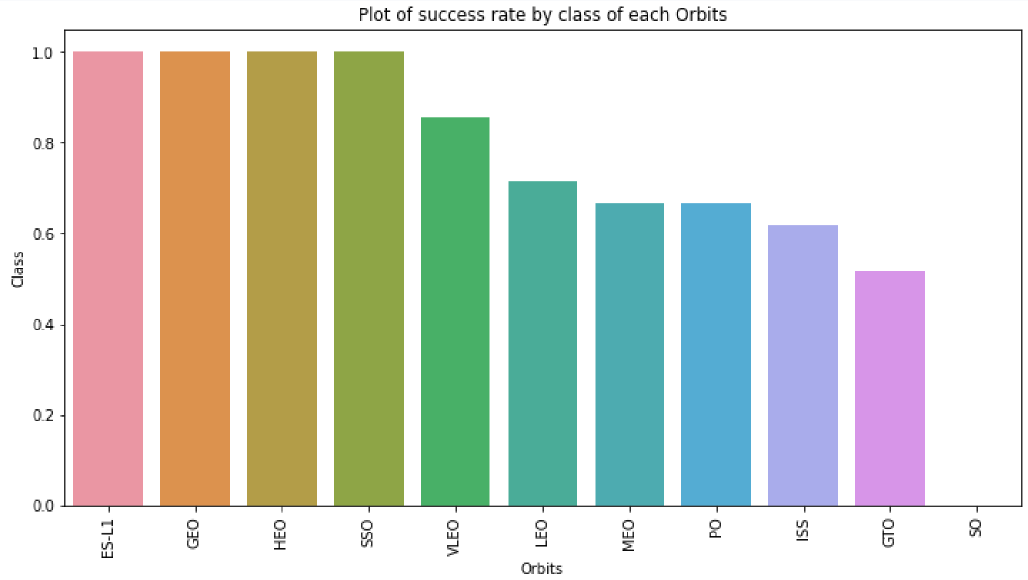
Data Wrangling



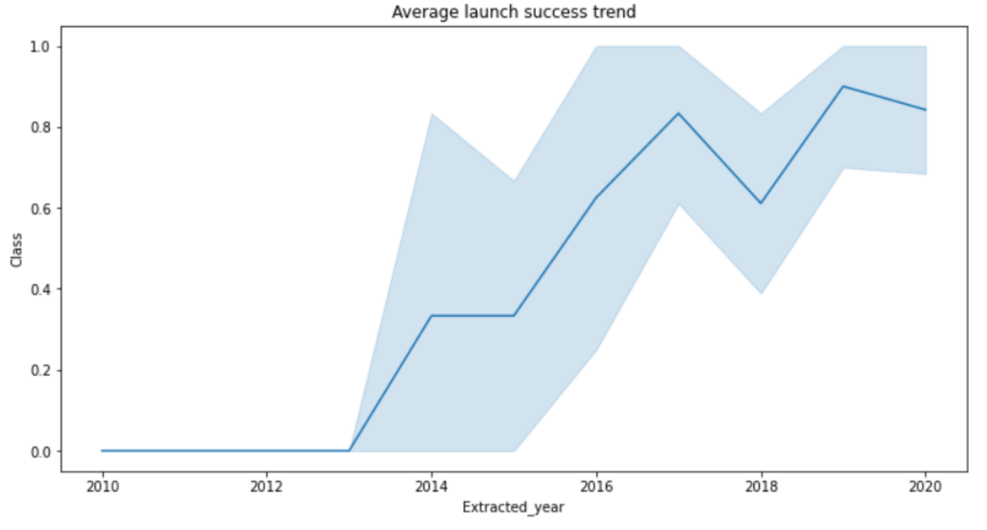
- We performed exploratory data analysis and determined the training labels.
- We calculated the number of launches at each site, and the number and occurrence of each orbits
- We created landing outcome label from outcome column and exported the results to csv.
- The link to the GitHub URL is; https://github.com/kbkerimoglu/kbkerimoglu.github.io/blob/master/IBM_Data_Science_Capstone_Project_SpaceX/Data%20Wrangling.ipynb

EDA with Data Visualization

- Analyze the ploted bar chart try to find which orbits have high sucess rate.
- We can see from bar chart, while ESL-1, GEO, HEO and SSO orbits that have highest success rate with 100%, SO orbit has the least success rate with 0%.



- Plotted line chart to get yearly average launch success trend. We can observe that the sucess rate since 2013 kept increasing till 2020.



- The link to the GitHub URL is; https://github.com/kbkerimoglu/kbkerimoglu.github.io/blob/master/IBM_Data_Science_Capstone_Project_SpaceX/EDA%20with%20Visualization.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 date where the first succesful landing outcome in drone ship was acheived.
  - The names of the booster versions which have carried the maximum payload mass.
  - The failed landing outcomes in drone ship, their booster version and launch site names.
- The link to the GitHub URL is; https://github.com/kbkerimoglu/kbkerimoglu.github.io/blob/master/IBM_Data_Science_Capstone_Project_SpaceX/EDA%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 to the GitHub URL is; https://github.com/kbkerimoglu/kbkerimoglu.github.io/blob/master/IBM_Data_Science_Capstone_Project_SpaceX/Interactive%20Visual%20Analytics%20with%20Folium.ipynb

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.
- The link to the GitHub URL is; https://github.com/kbkerimoglu/kbkerimoglu.github.io/blob/master/IBM_Data_Science_Capstone_Project_SpaceX/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 Github URL is; https://github.com/kbkerimoglu/kbkerimoglu.github.io/blob/master/IBM_Data_Science_Capstone_Project_SpaceX/Machine%20Learning%20Prediction.ipynb

Results

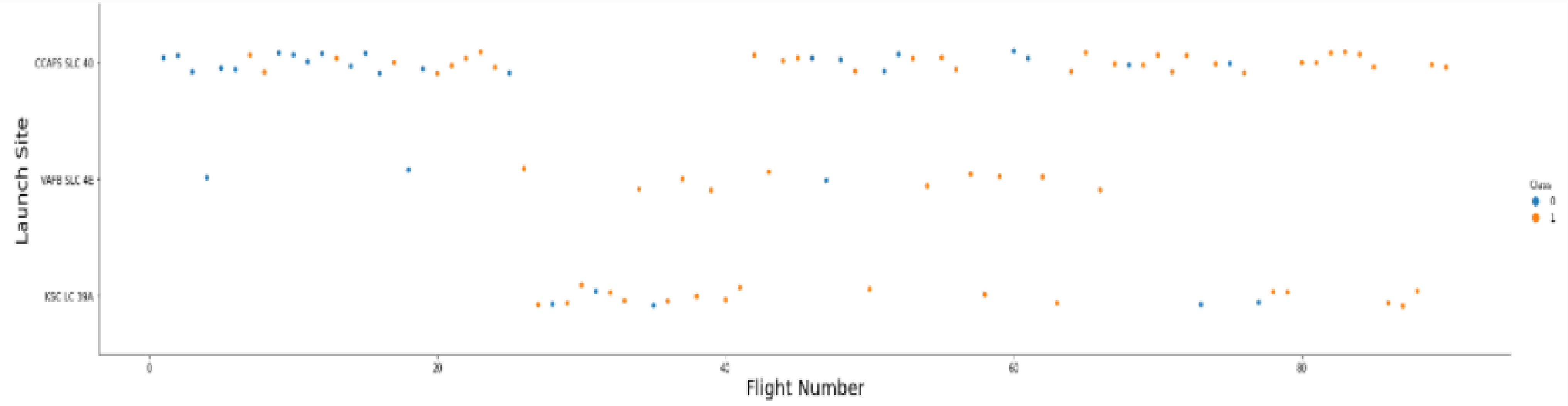
- Exploratory data analysis results
- Interactive analytics demo in screenshots
- Predictive analysis results

Section 2

Insights drawn from EDA

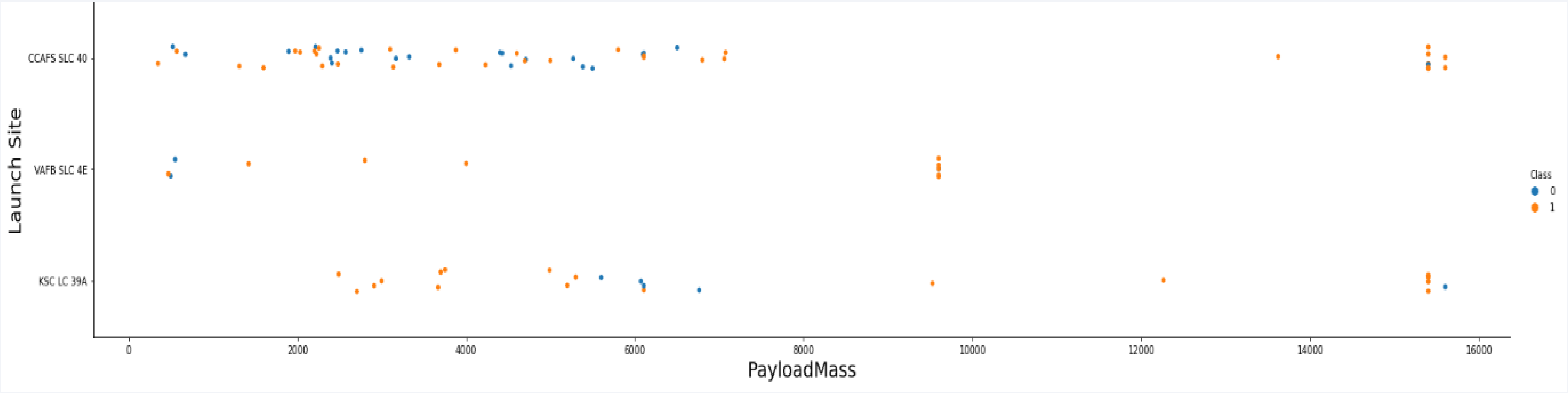
Flight Number vs. Launch Site

- As we can see on the line chart, the success rate increases as the flight number increases.



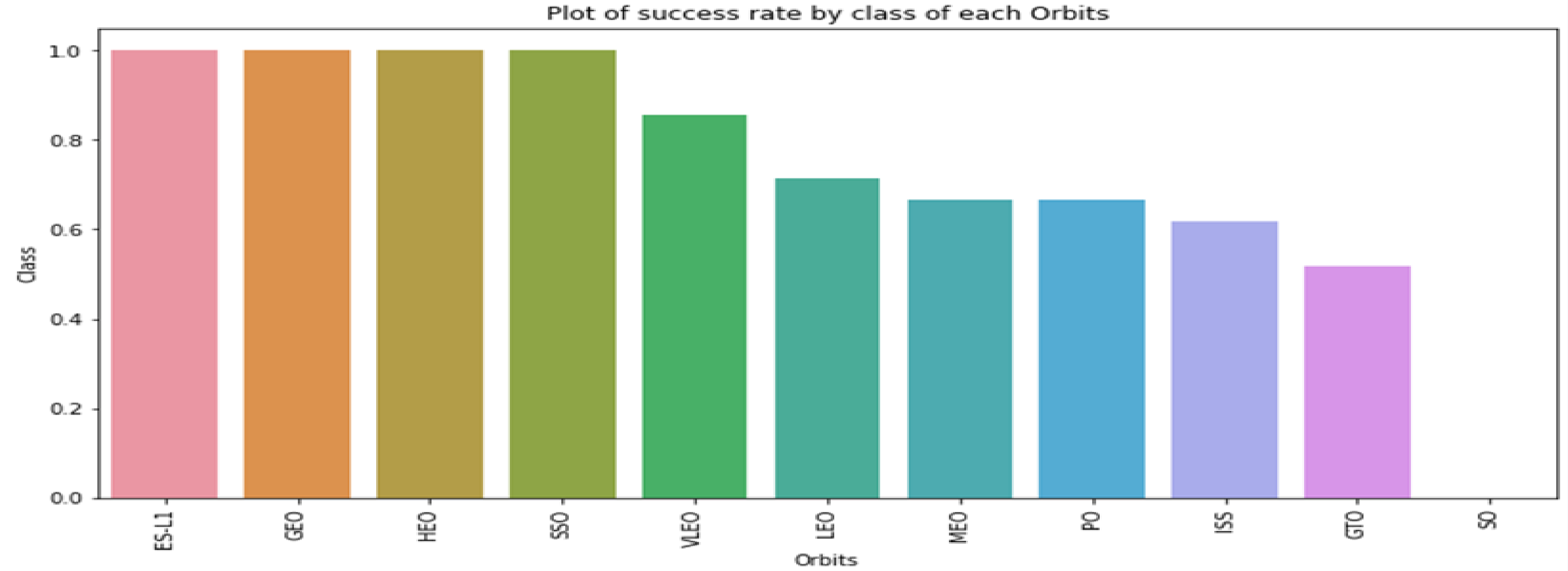
Payload vs. Launch Site

- We can observe Payload Vs. Launch Site scatter point chart that for the VAFB-SLC launchsite there are no rockets launched for heavypayload mass(greater than 10000).



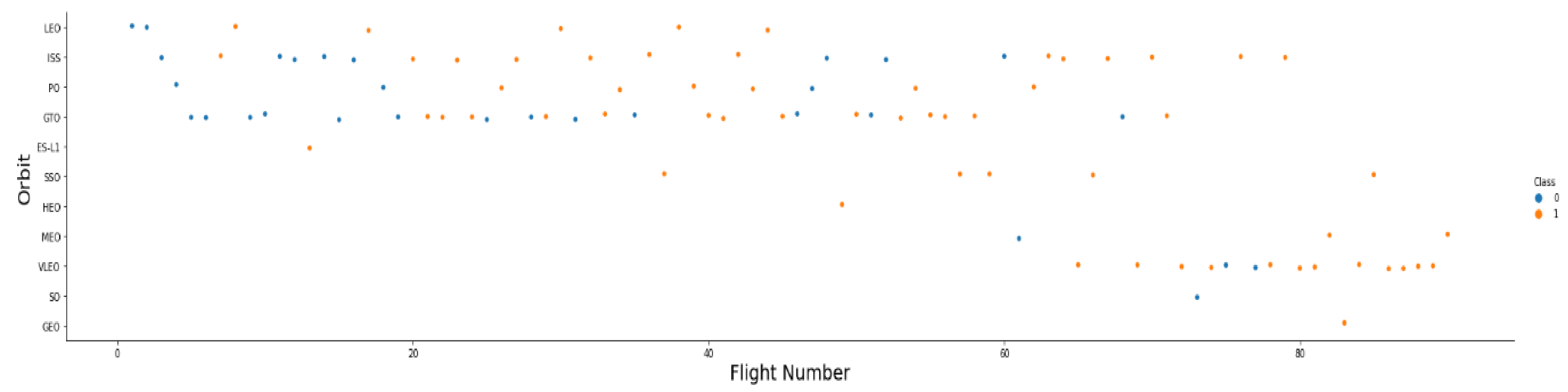
Success Rate vs. Orbit Type

- We can see from bar chart, while ESL-1, GEO, HEO and SSO orbits that have highest success rate with 100%, SO orbit has the least success rate with 0%.



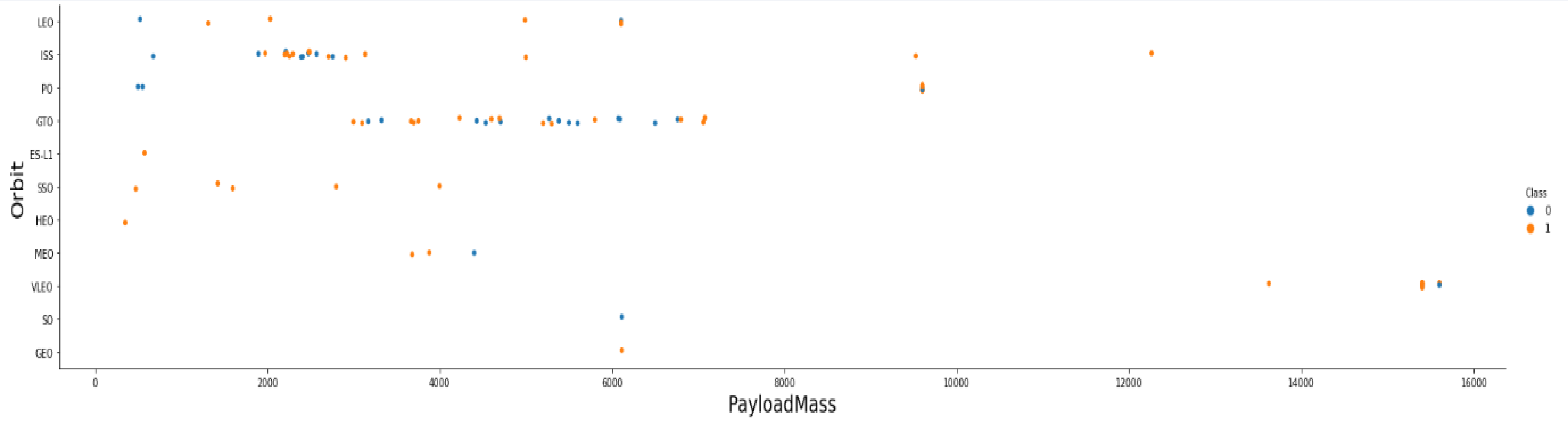
Flight Number vs. Orbit Type

- The line chart below illustrates that in the LEO orbit the Success appears related to the number of flights; on the other hand, there seems to be no relationship between flight number when in GTO orbit.



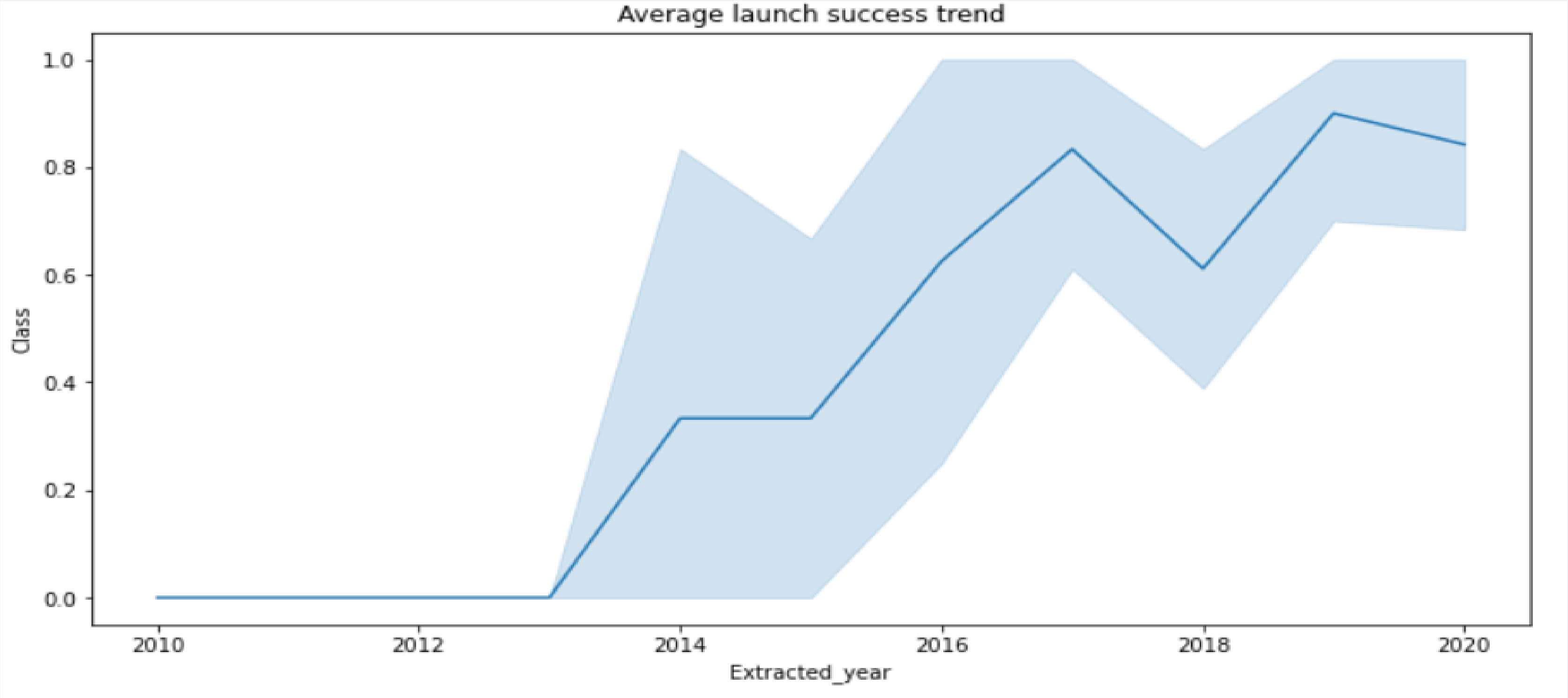
Payload vs. Orbit Type

- With heavy payloads the successful landing or positive landing rate are more for Polar,LEO and ISS. However for GTO we cannot distinguish this well as both positive landing rate and negative landing(unsuccessful mission) are both there here.



Launch Success Yearly Trend

- The line chart illustrates that the sucess rate since 2013 kept increasing till 2020.



All Launch Site Names

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

In [5]:
```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEXTBL
```

```
 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```

Out[5]:

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

Launch Site Names Begin with 'KSC'

- We used the query above to display 5 records where launch sites begin with `KSC`.

In [6]:
```
%sql SELECT * FROM SPACEXTBL WHERE LAUNCH_SITE LIKE 'KSC%' LIMIT 5
```

```
 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```

Out[6]:

DATE	time__utc_	booster_version	launch_site	payload	payload_mass__kg_	orbit	customer	mission_outcome	landing__outcome
2017-02-19	14:39:00	F9 FT B1031.1	KSC LC-39A	SpaceX CRS-10	2490	LEO (ISS)	NASA (CRS)	Success	Success (ground pad)
2017-03-16	06:00:00	F9 FT B1030	KSC LC-39A	EchoStar 23	5600	GTO	EchoStar	Success	No attempt
2017-03-30	22:27:00	F9 FT B1021.2	KSC LC-39A	SES-10	5300	GTO	SES	Success	Success (drone ship)
2017-05-01	11:15:00	F9 FT B1032.1	KSC LC-39A	NROL-76	5300	LEO	NRO	Success	Success (ground pad)
2017-05-15	23:21:00	F9 FT B1034	KSC LC-39A	Inmarsat-5 F4	6070	GTO	Inmarsat	Success	No attempt

Total Payload Mass

- We calculated the total payload carried by boosters from NASA as 45596 using the query below

```
In [15]:  %sql SELECT SUM(PAYLOAD_MASS__KG_) AS Total_PayloadMass FROM SPACEXTBL WHERE CUSTOMER LIKE 'NASA (CRS)'

 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```

Out[15]:

total_payloadmass
45596

Average Payload Mass by F9 v1.1

- We calculated the average payload mass carried by booster version F9 v1.1 as 2928.

```
In [21]: %sql SELECT AVG(PAYLOAD_MASS__KG_) AS Avg_PayloadMass FROM SPACEXTBL WHERE BOOSTER_VERSION = 'F9 v1.1'

 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.

Out[21]:
```

avg_payloadmass
2928

First Successful Drone Ship Landing Date

- We observed that the dates of the first successful landing outcome on drone ship was 8th April 2016.

List the date where the first succesful landing outcome in drone ship was acheived.

Hint:Use min function

In [36]:
```
%sql SELECT MIN(DATE) AS First_Succesful_landing_outcome FROM SPACEXTBL WHERE landing__outcome LIKE 'Success (drone ship)'
```

```
 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```

Out[36]:

first_succesful_landing_outcome
2016-04-08

Successful Drone Ship Landing with Payload between 4000 and 6000

- We used the **WHERE** clause to filter for boosters which have successfully landed on ground pad and applied the **AND** condition to determine successful landing with payload mass greater than 4000 but less than 6000.

List the names of the boosters which have success in ground pad and have payload mass greater than 4000 but less than 6000

In [37]:
```
%sql SELECT BOOSTER_VERSION FROM SPACEXTBL WHERE landing__outcome = 'Success (ground pad)' AND PAYLOAD_MASS__KG_ > 4000 AND PAYLOAD_MASS__KG_ < 6000
```

```
 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```

Out[37]:

booster_version
F9 FT B1032.1
F9 B4 B1040.1
F9 B4 B1043.1

Total Number of Successful and Failure Mission Outcomes

- We used like ‘%’ function to filter for **WHERE** Mission_Outcome was a success or a failure. We can see from results, total number of successful and failure missions are 101.

List the total number of successful and failure mission outcomes

In [48]:

```
%sql SELECT COUNT(MISSION_OUTCOME) AS SuccessOutcome FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Success%'
```

```
 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```

Out[48]:

successoutcome
100

In [49]:

```
%sql SELECT COUNT(MISSION_OUTCOME) AS FailureOutcome FROM SPACEXTBL WHERE MISSION_OUTCOME LIKE 'Failure%'
```

```
 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```

Out[49]:

failureoutcome
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 [51]:
```
%sql SELECT BOOSTER_VERSION, PAYLOAD_MASS__KG_ FROM SPACEXTBL WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL) ORDER BY BOOSTER
```
```
 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```
Out[51]:

booster_version	payload_mass__kg_
F9 B5 B1048.4	15600
F9 B5 B1048.5	15600
F9 B5 B1049.4	15600
F9 B5 B1049.5	15600
F9 B5 B1049.7	15600
F9 B5 B1051.3	15600
F9 B5 B1051.4	15600
F9 B5 B1051.6	15600
F9 B5 B1056.4	15600
F9 B5 B1058.3	15600
F9 B5 B1060.2	15600
F9 B5 B1060.3	15600

2017 Launch Records

- We used a combinations of the **WHERE** clause, **LIKE** and **AND** conditions to filter for successful landing outcomes in ground pad, their booster versions, and launch site names for year 2017.

List the records which will display the month names, succesful landing_outcomes in ground pad ,booster versions, launch_site for the months in year 2017

In [58]:
```
%sql SELECT MONTH(DATE), LANDING__OUTCOME, BOOSTER_VERSION, LAUNCH_SITE FROM SPACEXTBL WHERE YEAR(DATE) = 2017 AND LANDING__OUTCOME LIKE 'Success (grou
```

```
 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```

Out[58]:

1	landing__outcome	booster_version	launch_site
2	Success (ground pad)	F9 FT B1031.1	KSC LC-39A
5	Success (ground pad)	F9 FT B1032.1	KSC LC-39A
6	Success (ground pad)	F9 FT B1035.1	KSC LC-39A
8	Success (ground pad)	F9 B4 B1039.1	KSC LC-39A
9	Success (ground pad)	F9 B4 B1040.1	KSC LC-39A
12	Success (ground pad)	F9 FT B1035.2	CCAFS SLC-40

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 2017-03-20.
- 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 successful landing_outcomes between the date 2010-06-04 and 2017-03-20 in descending order.

```
In [64]: %sql SELECT LANDING__OUTCOME, COUNT(LANDING__OUTCOME) AS Count FROM SPACEXTBL WHERE DATE BETWEEN '2010-06-04' AND '2017-03-20' GROUP BY LANDING__OUTCOM
```

```
 * ibm_db_sa://gzt70091:***@21fecfd8-47b7-4937-840d-d791d0218660.bs2io90l08kqb1od8lcg.databases.appdomain.cloud:31864/bludb
Done.
```

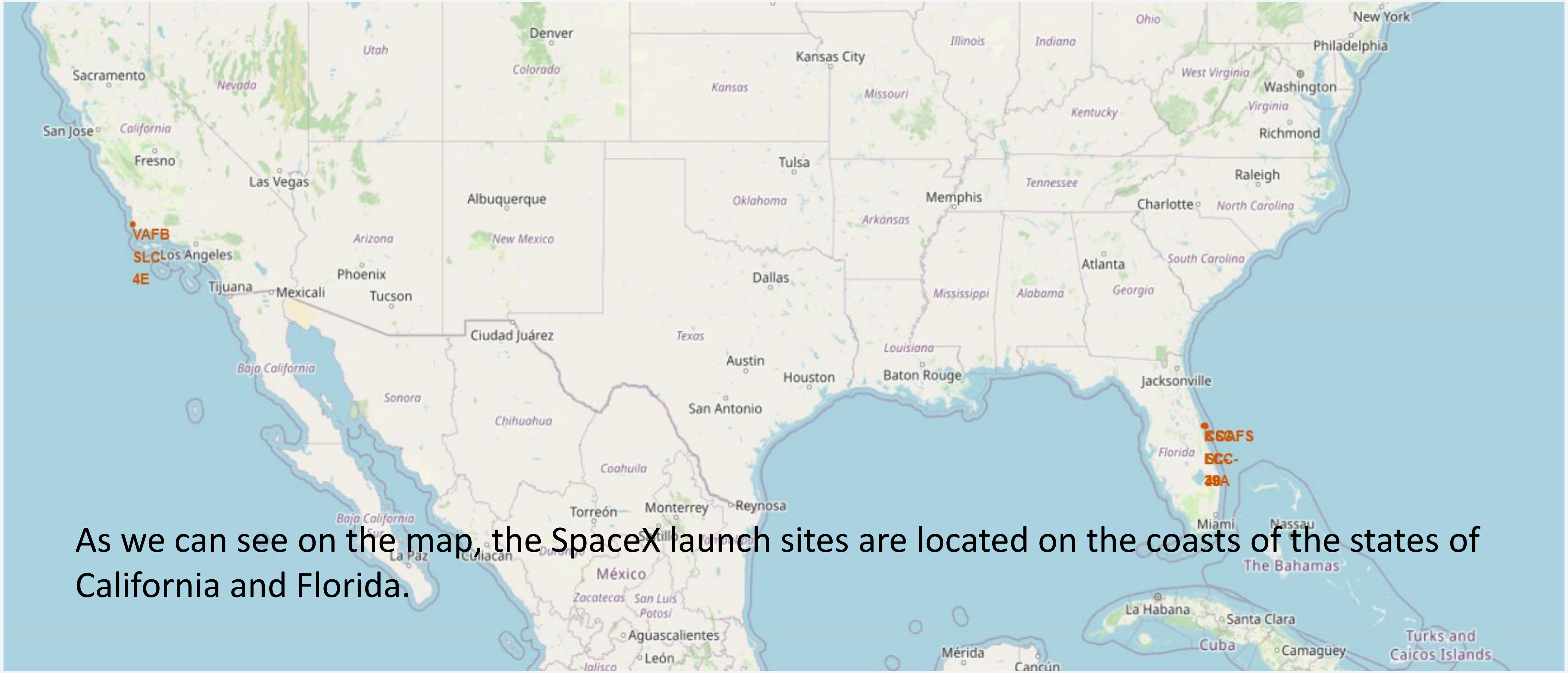
Out[64]:

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

Section 3

Launch Sites Proximities Analysis

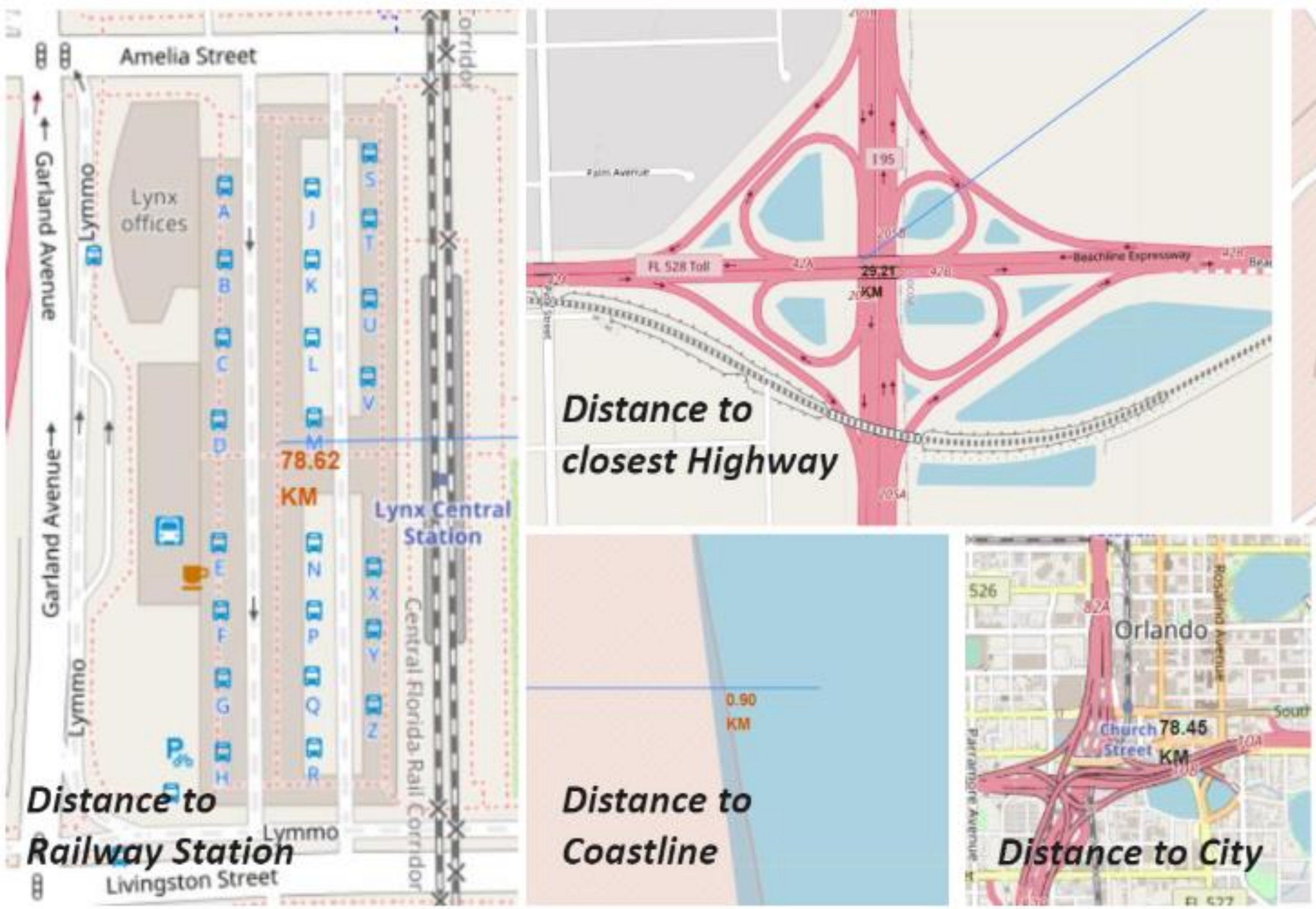
Mark All Launches Sites on a Global Map



As we can see on the map, the SpaceX launch sites are located on the coasts of the states of California and Florida.

Mark the Success/Failed Launches for Each Site on the Map

The Distances Between Launch Sites to Its Proximities



- 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

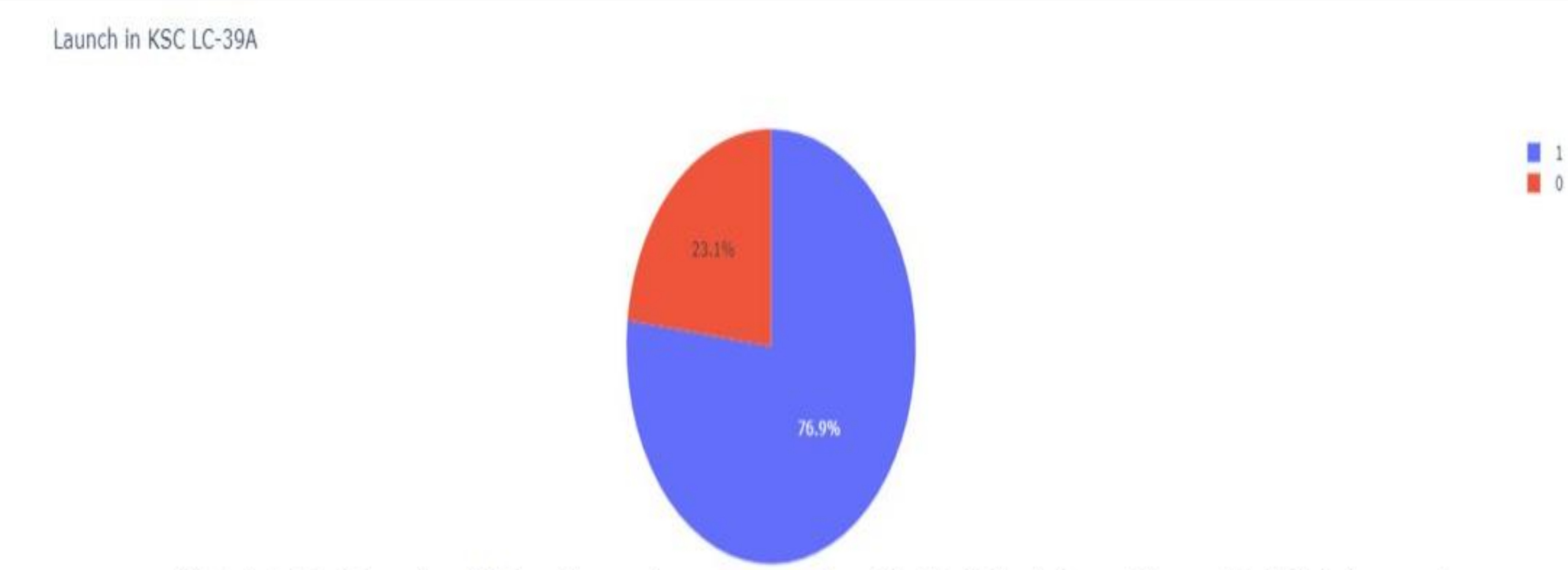
Section 4

Build a Dashboard with Plotly Dash

All Launch Sites Success Rate with Pie Chart

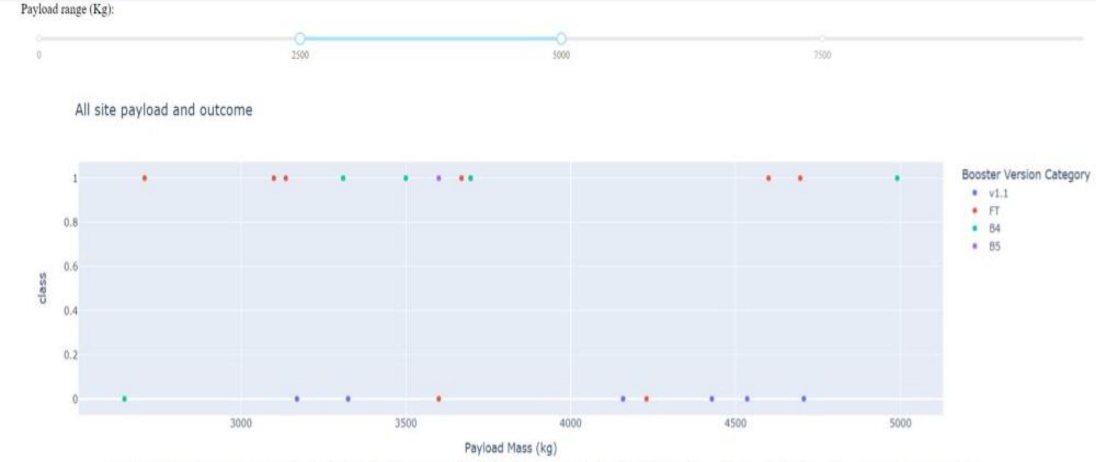
We can see that KSC LC-39A has the most successful launches from all the sites.

Highest Launch Site Success Ratio with Pie Chart

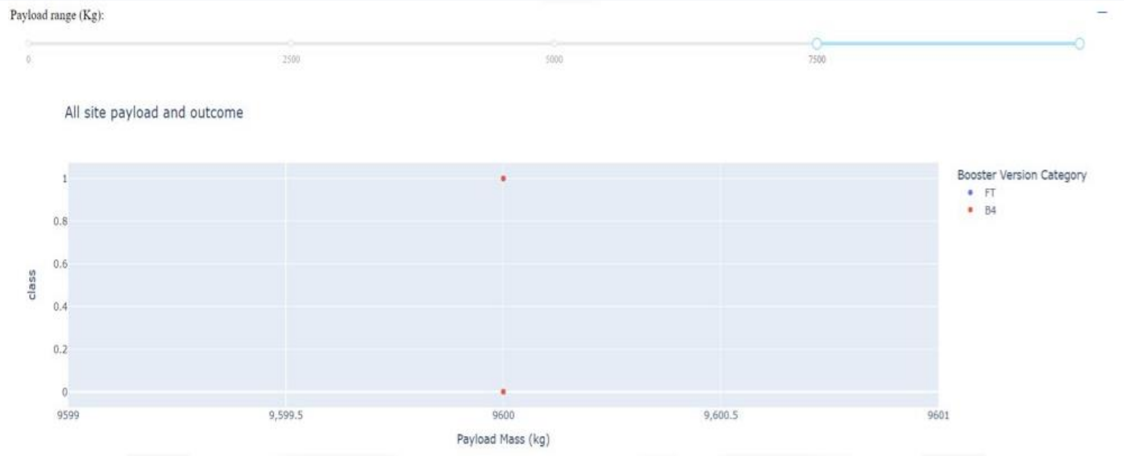


KSC LC-39A has the highest launch success rate with 76.9% while getting a 23.1% failure rate.

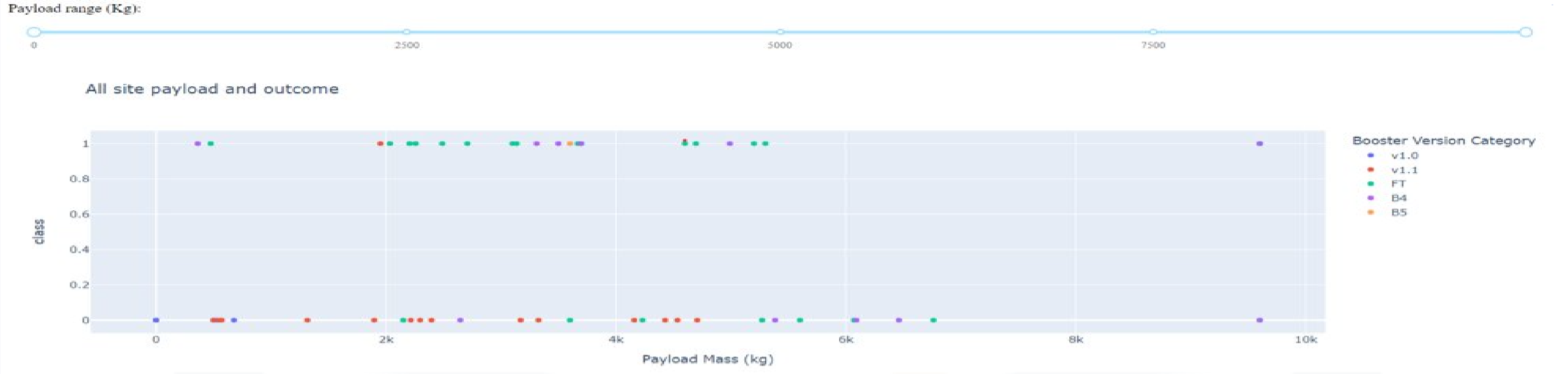
Scatter Plot of Payload vs Launch Outcome for all Category, with Different Payload Range



Payload range which is between 2500 kg and 5000 kg has the highest success rate.



Payload range which is between 7500 kg and 10000 kg has the lowest success rate.



We can see from scatter plot, FT category has the highest success rate.

Section 5

Predictive Analysis (Classification)

Classification Accuracy

The best model with the highest classification accuracy (87.5%) is the decision tree classifier.

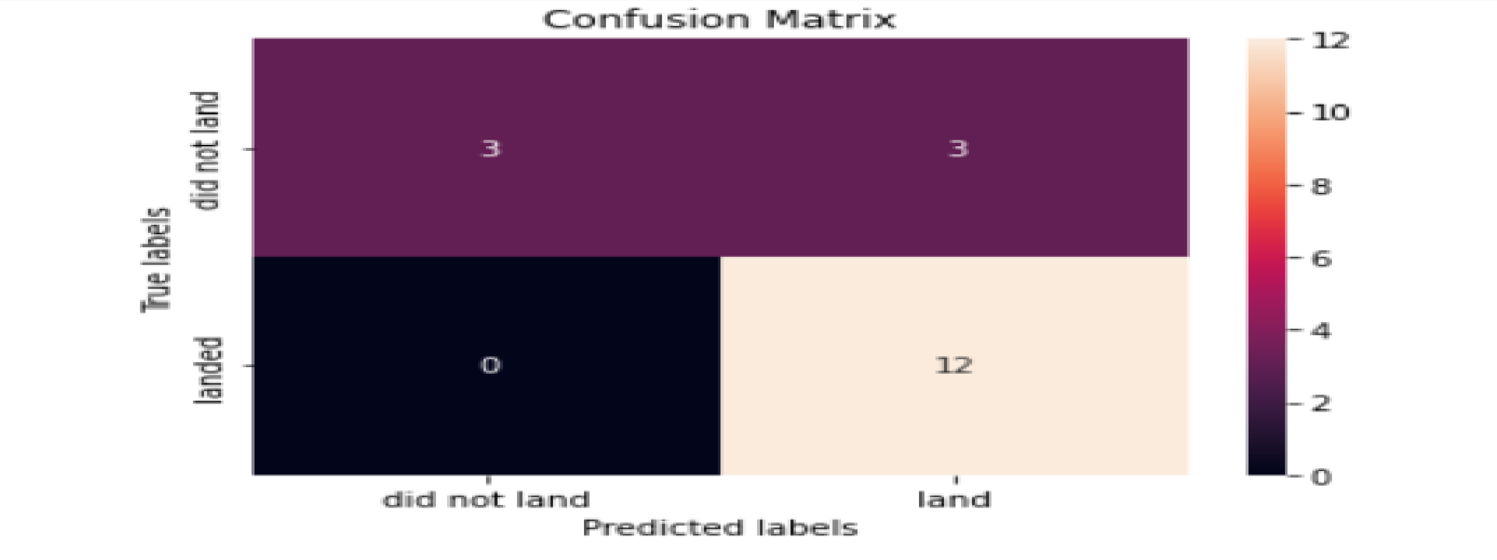
In [33]:

```
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': 'gini', 'max_depth': 18, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'random'}
```

Confusion Matrix

The Confusion Matrix for decision tree classifier illustrates that the classifier can distinguish between the different classes. We see that the major problem is false positives.



Conclusions

We can conclude that:

- The success rate increases as the flight number increases.
- While ESL-1, GEO, HEO and SSO orbits that have highest success rate with 100%, SO orbit has the least success rate with 0%.
- The sucess rate since 2013 kept increasing till 2020.
- The total payload carried by boosters from NASA as 45596.
- The dates of the first successful landing outcome on drone ship was 8th April 2016.
- The SpaceX launch sites are located on the coasts of the states of California and Florida.
- KSC LC-39A had the most successful launches from all sites.
- The best model with the highest classification accuracy (87.5%) is the decision tree classifier.

Thank you!