



IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Chukwuemeka Okoli
2nd December 2021



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 publicizes Falcon 9 rocket dispatches on its site with an expense of 62 million bucks; different suppliers cost vertical of 165 million bucks every, a significant part of the investment funds is on the grounds that Space X can reuse the primary stage. In this manner, in the event that we can decide whether the primary stage will land, we can decide the expense of a send off. This data can be utilized to offer against space X for a rocket send off. This objective of the undertaking is to make an AI pipeline to foresee on the off chance that the primary stage will land effectively.

- Problems you want to find answers

- What factors determine if the rocket will land successfully?
- The success rate of a successful landing is determined based on the interaction among various features.
- 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 API and web scraping from Wikipedia.
- Perform data wrangling
 - One-hot encoding was applied to categorical features
- Perform exploratory data analysis (EDA) using visualization and SQL
- Perform interactive visual analytics using Folium and Plotly Dash
- Perform predictive analysis using classification models
 - How to build, tune, evaluate classification models

Data Collection

- The information was gathered utilizing different strategies
- Get demand was utilized to gather information of the SpaceX Programming interface.
- Then, we decoded the reaction content as a Json utilizing `.json()` capability call and transform it into a pandas information outline utilizing `.json standardize()`.
- We then cleaned the information, checked for missing qualities and fill in missing qualities where essential.
- What's more, we performed web scratching from Wikipedia for Hawk 9 send off records with BeautifulSoup.
- The goal was to extricate the send off records as HTML table, parse the table and convert it to a pandas information outline for future examination.

Data Collection – SpaceX API

- We utilized the get solicitation to the SpaceX Programming interface to gather information, clean the mentioned information and did a few essential information fighting and designing.
- The link to the notebook is <https://github.com/neelvanani3/Neel-DS-Capstone/blob/master/jupyter-labs-webscraping.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 – Scrapping

- We applied web scrapping to webscrap Falcon 9 launch records with BeautifulSoup
- We parsed the table and converted it into a pandas dataframe.
- The link to the notebook is <https://github.com/neelvanani3/Neel-DS-Capstone/blob/master/jupyter-labs-webscraping.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
```

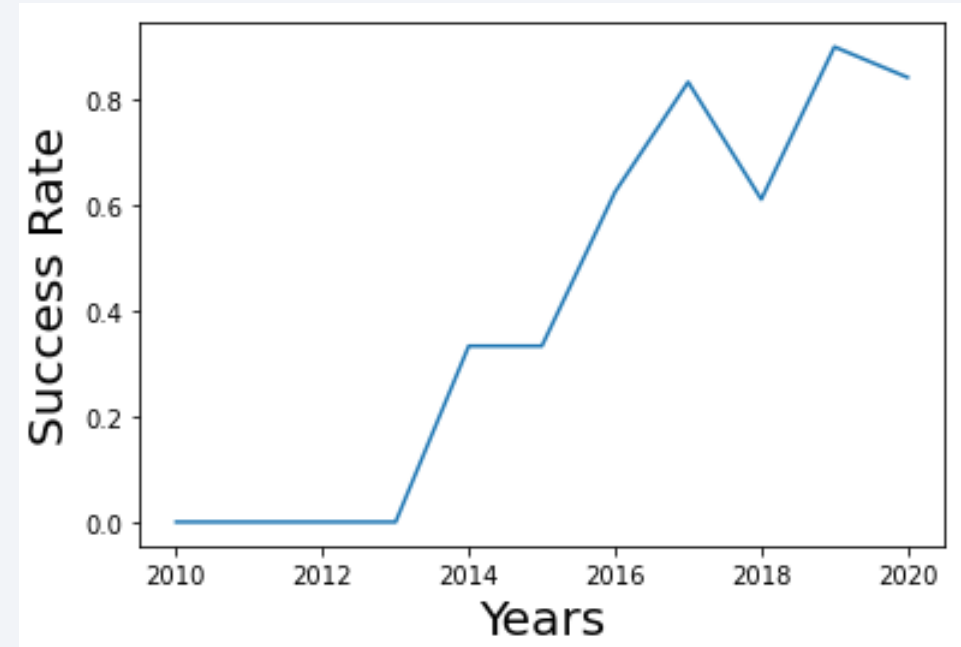
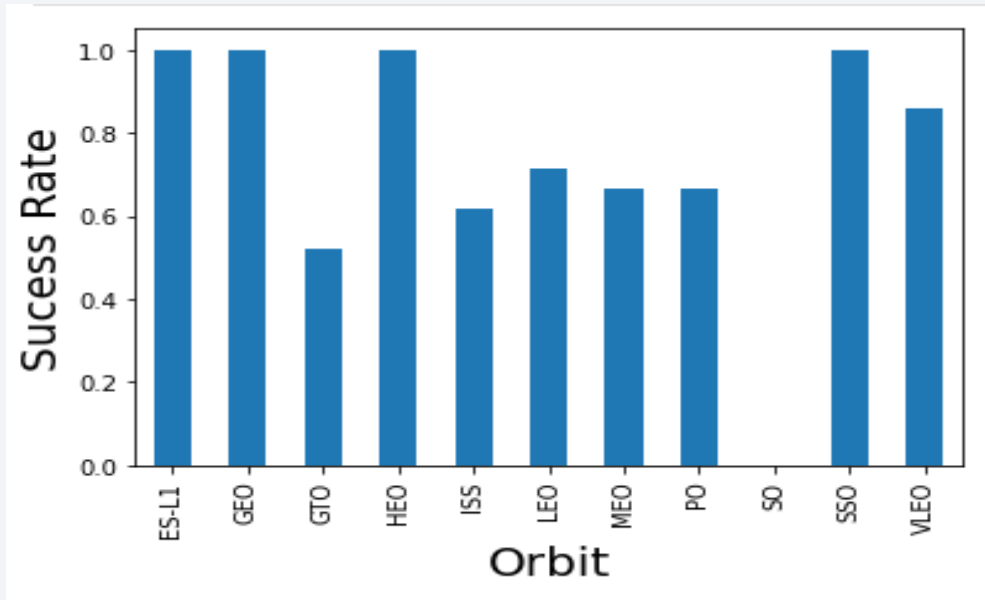
Data Wrangling



- We performed exploratory information examination and decided the preparation marks.
- We determined the quantity of dispatches at each site, and the number and event of each circles
- We made landing result name from result segment and sent out the outcomes to csv.
- The link to the notebook is <https://github.com/neelvanani3/Neel-DS-Capstone/blob/master/labs-jupyter-spacex-Data%20wrangling.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, the launch success yearly trend.



- The link to the notebook is <https://github.com/neelvanani3/Neel-DS-Capstone/blob/master/jupyter-labs-eda-dataviz.ipynb>

EDA with SQL

- We load the SpaceX dataset into a PostgreSQL data set without leaving the jupyter journal.
- We applied EDA with SQL to get knowledge from the information. We composed questions to find out for example:
 - Special send off destinations names in the space mission.
 - The complete payload mass conveyed by supporters sent off by NASA (CRS)
 - The typical payload mass conveyed by sponsor rendition F9 v1.1
 - The all out number of effective and disappointment mission results
 - The bombed landing results in drone transport, their sponsor rendition and send off site names.
- The link to the notebook is <https://github.com/neelvanani3/Neel-DS-Capstone/blob/master/jupyter-labs-eda-sql-coursera.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?

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 notebook is <https://github.com/neelvanani3/Neel-DS-Capstone/blob/master/APP.py>

Predictive Analysis (Classification)

- We stacked the information utilizing numpy and pandas, changed the information, split our information into preparing and testing.
- We fabricated different AI models and tune different hyperparameters utilizing GridSearchCV.
- We involved exactness as the measurement for our model, further developed the model utilizing highlight designing and calculation tuning.
- We found the best performing classification model.
- The link to the notebook is [https://github.com/neelvanani3/Neel-DS-Capstone/blob/master/SpaceX_Machine_Learning_Prediction_Part_5_\(2\).ipynb](https://github.com/neelvanani3/Neel-DS-Capstone/blob/master/SpaceX_Machine_Learning_Prediction_Part_5_(2).ipynb)

Results

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

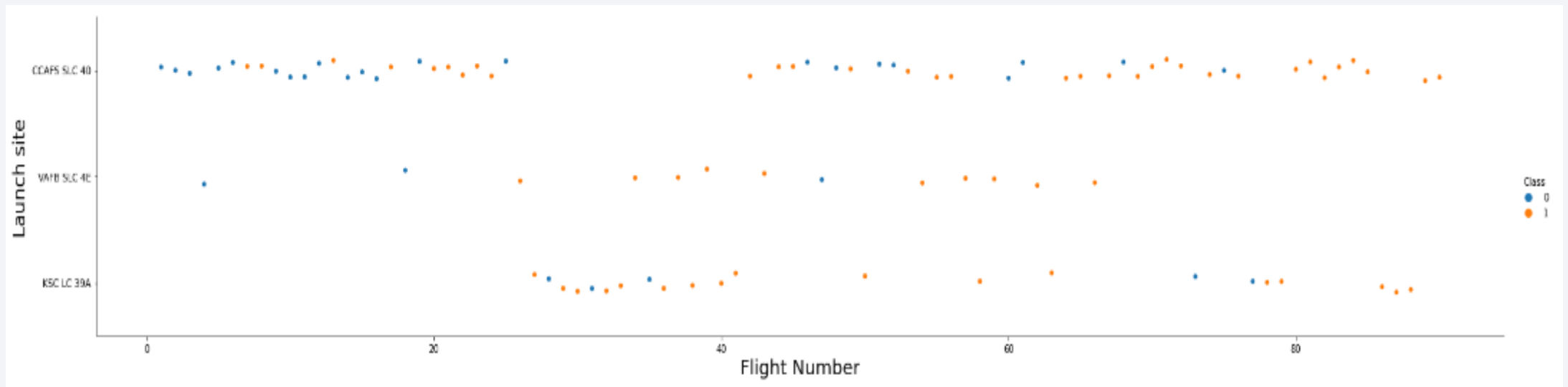
The background of the slide is an abstract composition. It features a solid blue area on the left side, which transitions into a dynamic pattern of diagonal streaks in shades of blue, red, and cyan on the right. These streaks are layered over a faint, dark grid pattern, creating a sense of depth and movement, reminiscent of a digital or data visualization theme.

Section 2

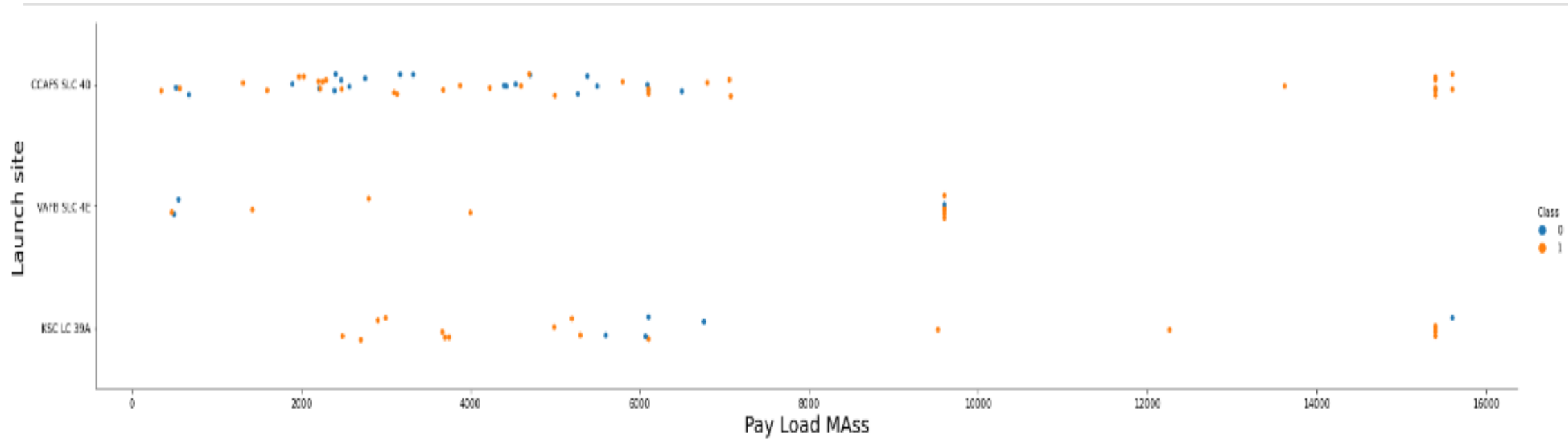
Insights drawn from EDA

Flight Number vs. Launch Site

- From the plot, we found that the larger the flight amount at a launch site, the greater the success rate at a launch site.

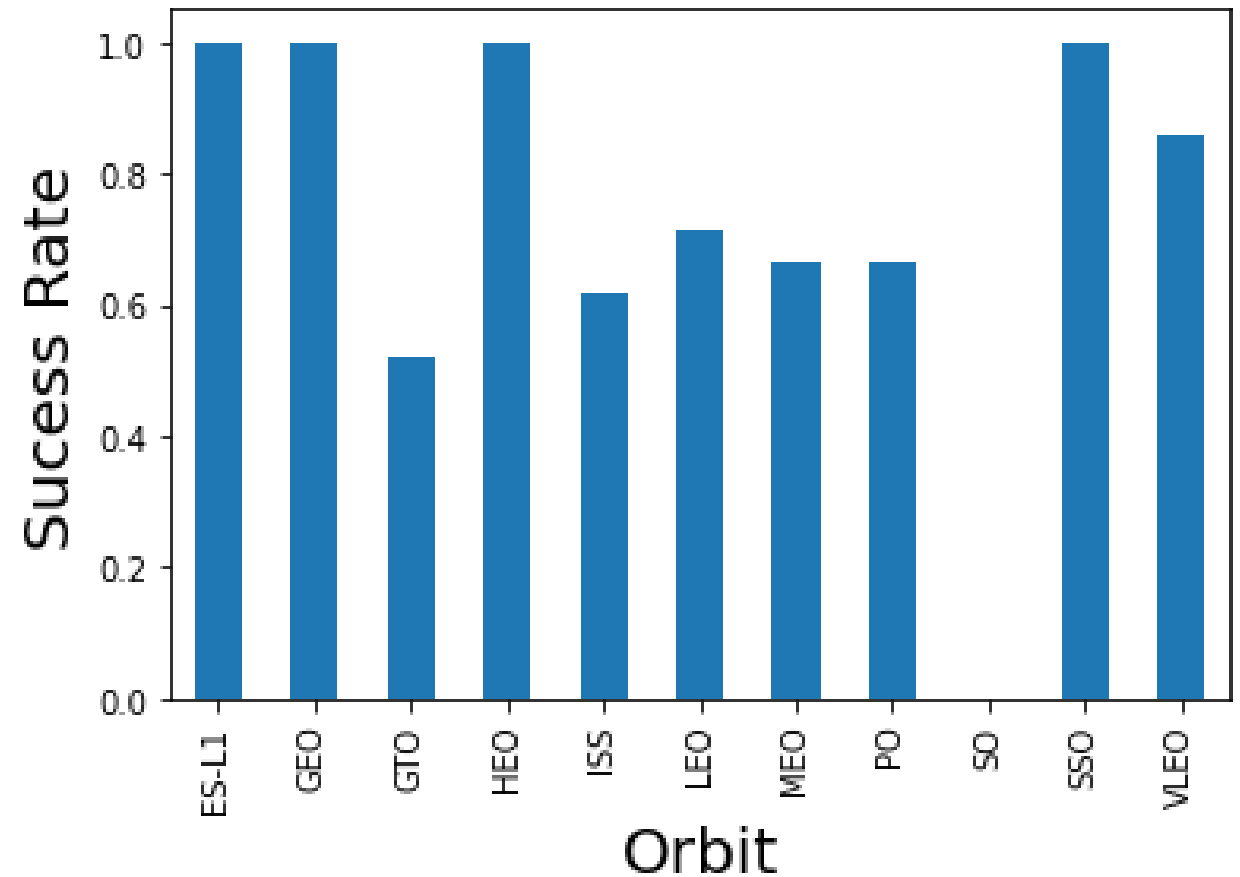


Payload vs. Launch Site



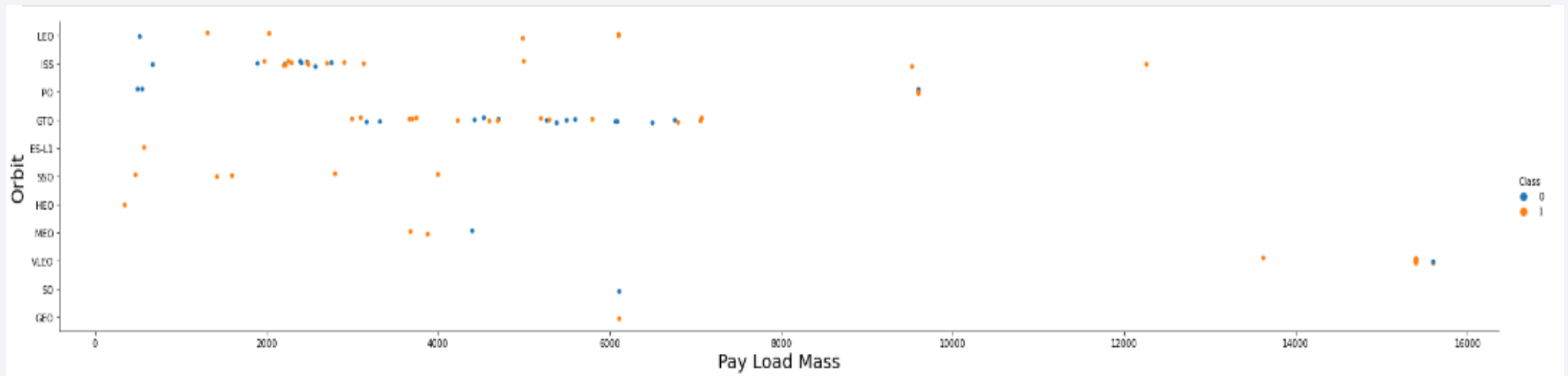
Success Rate vs. Orbit Type

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



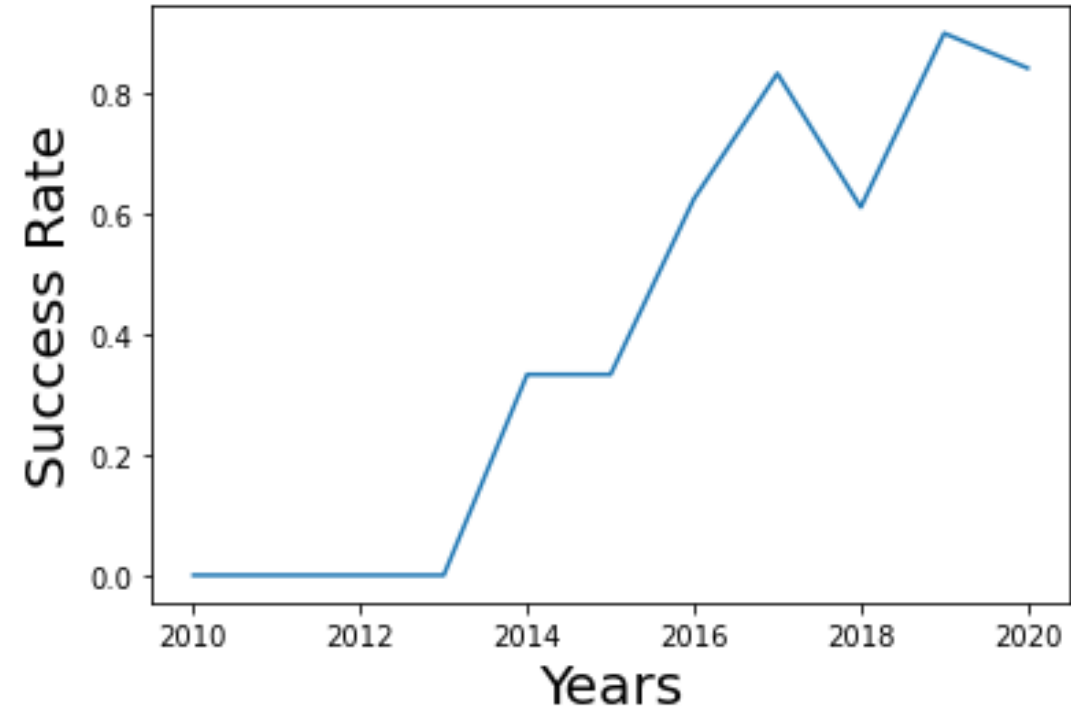
Payload vs. Orbit Type

- 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 key word **DISTINCT** to show only unique launch sites from the SpaceX data.

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

In [4]:

```
%sql SELECT DISTINCT LAUNCH_SITE FROM SPACEX
```

```
* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4  
Done.
```

Out[4]:

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

```
* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

```
Out[10]:
```

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 the query below

Task 3

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

```
In [23]: %sql SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEX WHERE CUSTOMER LIKE 'NASA (CRS)'
```

* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqn timer 39u98g
Done.

```
Out[23]: 1
```

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 [26]: %sql SELECT AVG(PAYLOAD_MASS_KG_) FROM SPACEX WHERE BOOSTER_VERSION='F9 v1.1'

* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39t
Done.

Out[26]: 1
          2928
```

First Successful Ground Landing Date

- We observed that the dates of the first successful landing outcome on ground pad was 22nd December 2015

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

Hint: Use min function

```
In [33]: %sql SELECT MIN(DATE) FROM SPACEX WHERE LANDING__OUTCOME='Success (ground pad)'
```

```
* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g  
Done.
```

```
Out[33]:      1  
  
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

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 [35]: %sql SELECT BOOSTER_VERSION FROM SPACEX WHERE PAYLOAD_MASS_KG_ BETWEEN 4000 AND 6000 AND LANDING__OUTCOME = 'Success (drone ship)'
```

```
* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

```
Out[35]: 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 wildcard like '%' to filter for **WHERE** Mission Outcome was a success or a failure.

List the total number of successful and failure mission outcomes

```
In [38]: %sql select count(*) from SPACEX where mission_outcome like '%Success%' or mission_outcome like '%Faliure%'
* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud
Done.
```

```
Out[38]: 1
100
```

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 [40]: %sql select booster_version from SPACEX where payload_mass_kg_ = (select max(payload_mass_kg_) from SPACEX)
```

* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqn timer 39u98g.databases.appdomain.cloud:32
Done.

Out[40]: **booster_version**

F9 B5 B1048.4

F9 B5 B1049.4

F9 B5 B1051.3

F9 B5 B1056.4

F9 B5 B1048.5

F9 B5 B1051.4

F9 B5 B1049.5

F9 B5 B1060.2

F9 B5 B1058.3

F9 B5 B1051.6

F9 B5 B1060.3

F9 B5 B1049.7

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 [46]: %sql select landing__outcome,booster_version,launch_site from SPACEX where DATE like '%2015%' and landing__outcome='Failure (drone ship)'
```

```
* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

```
Out[46]: 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
```

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 landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

```
In [57]: %sql select DATE, count(landing__outcome) as count from SPACEX where DATE between '2010-06-04' and '2017-03-20' AND landing__outcome like 'Success (gr
```

```
* ibm_db_sa://hkj11927:***@1bbf73c5-d84a-4bb0-85b9-ab1a4348f4a4.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud:32286/bludb
Done.
```

```
Out[57]:
```

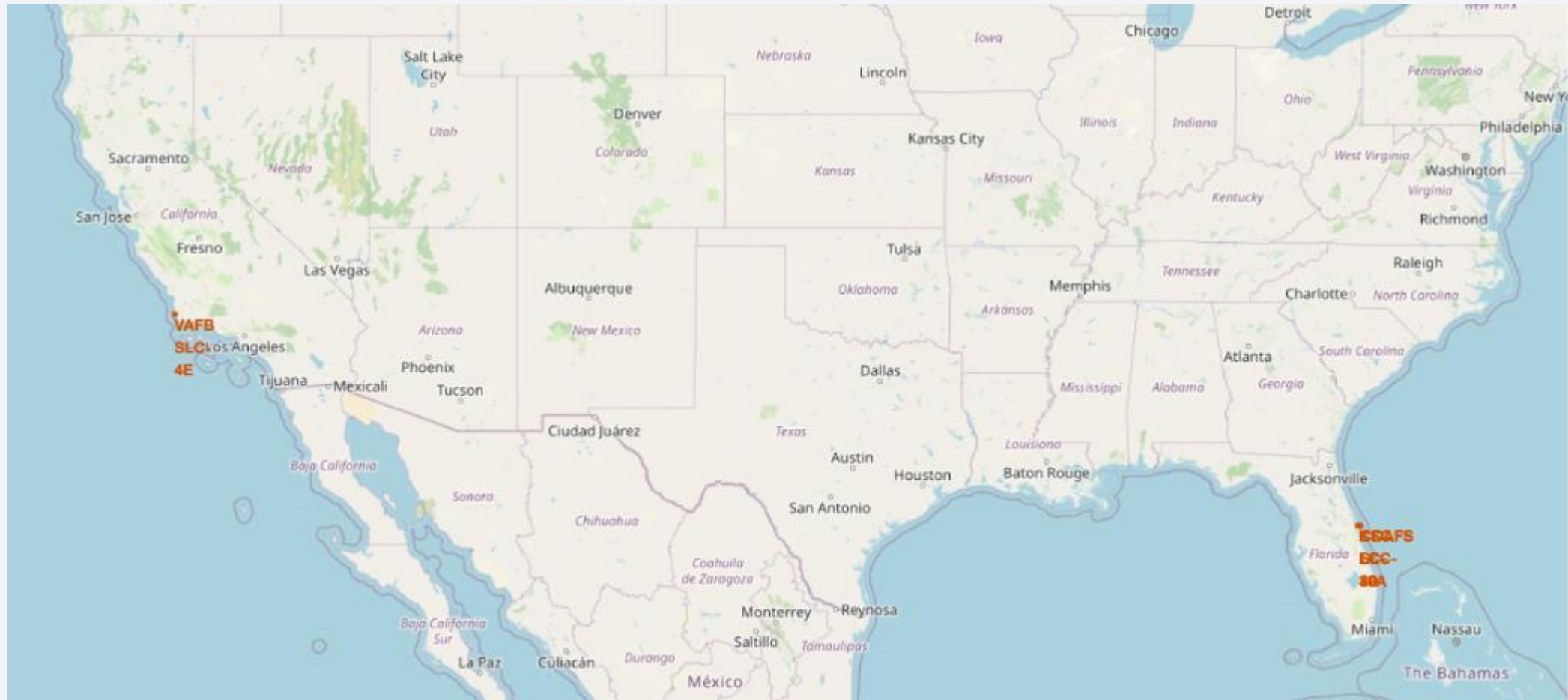
DATE	COUNT
2015-01-10	1
2015-04-14	1
2015-12-22	1
2016-01-17	1
2016-03-04	1
2016-06-15	1
2016-07-18	1
2017-02-19	1

Section 4

Launch Sites Proximities Analysis



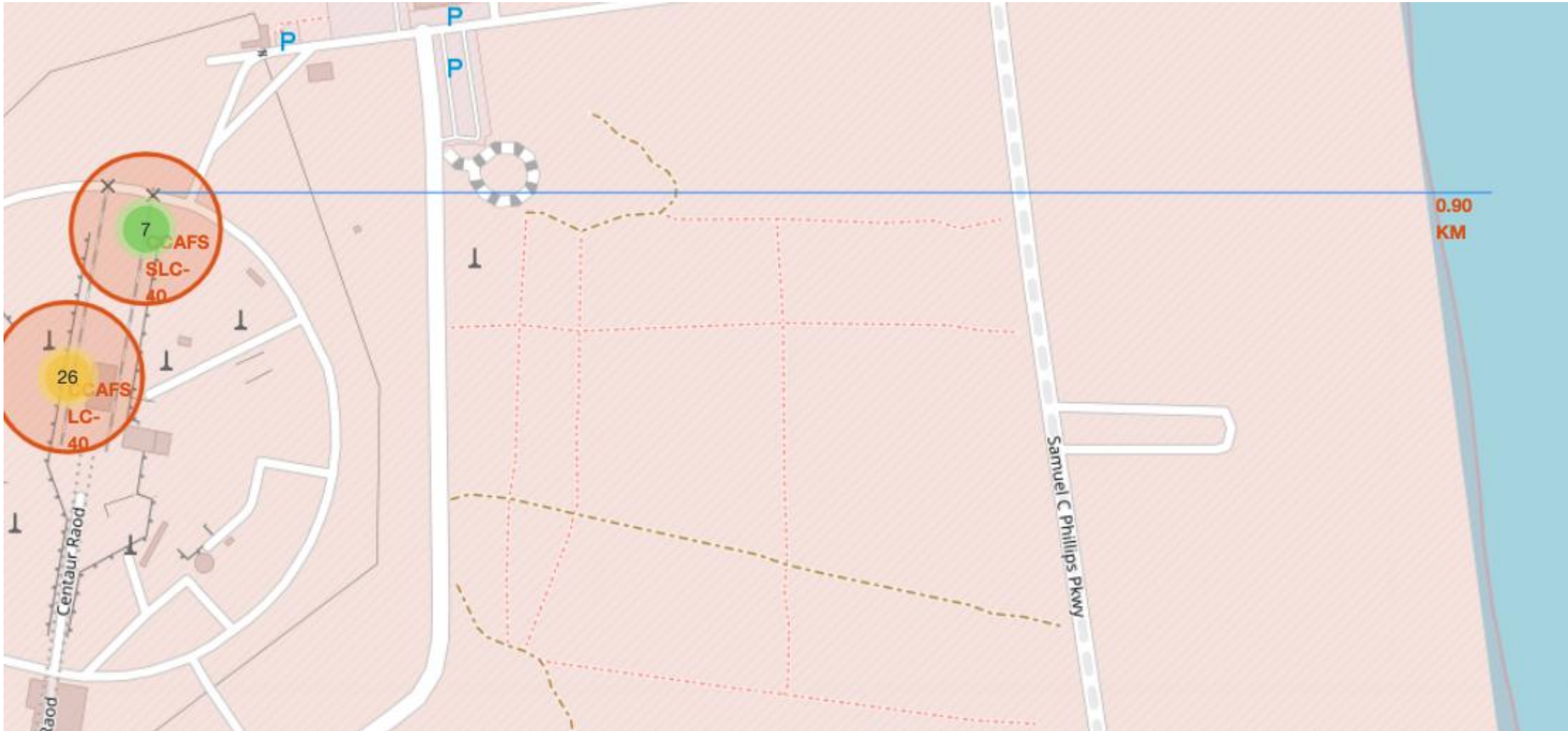
All launch sites global map markers



Markers showing launch sites with color labels



Launch Site distance from coast





Section 5

Build a Dashboard with Plotly Dash

Pie chart showing the success percentage achieved by each launch site

Total Success Launches By all sites

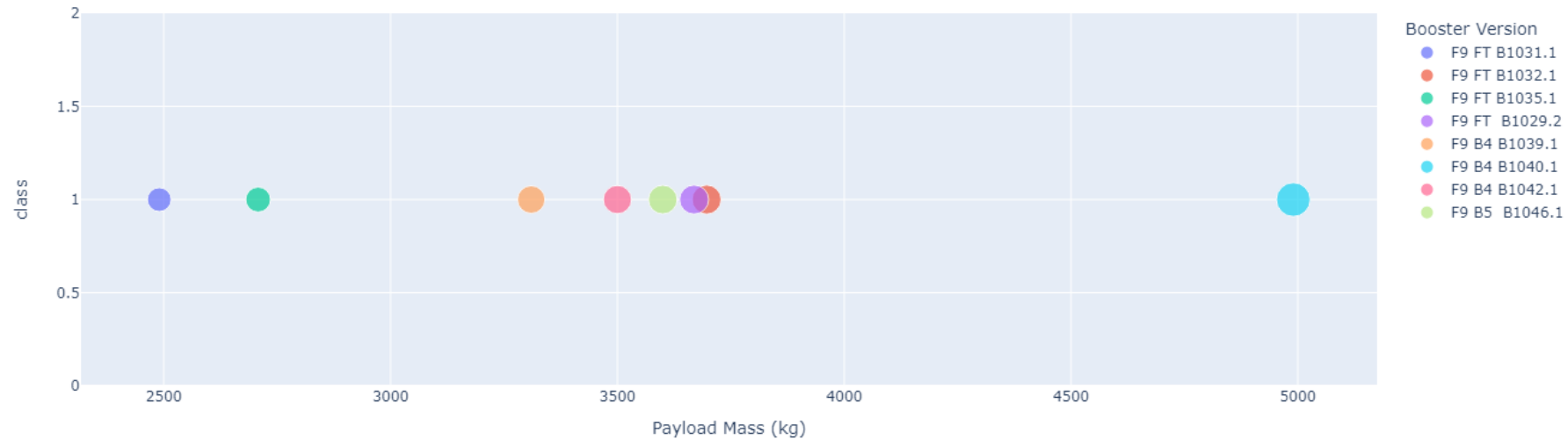


Pie chart showing the Launch site with the highest launch success ratio

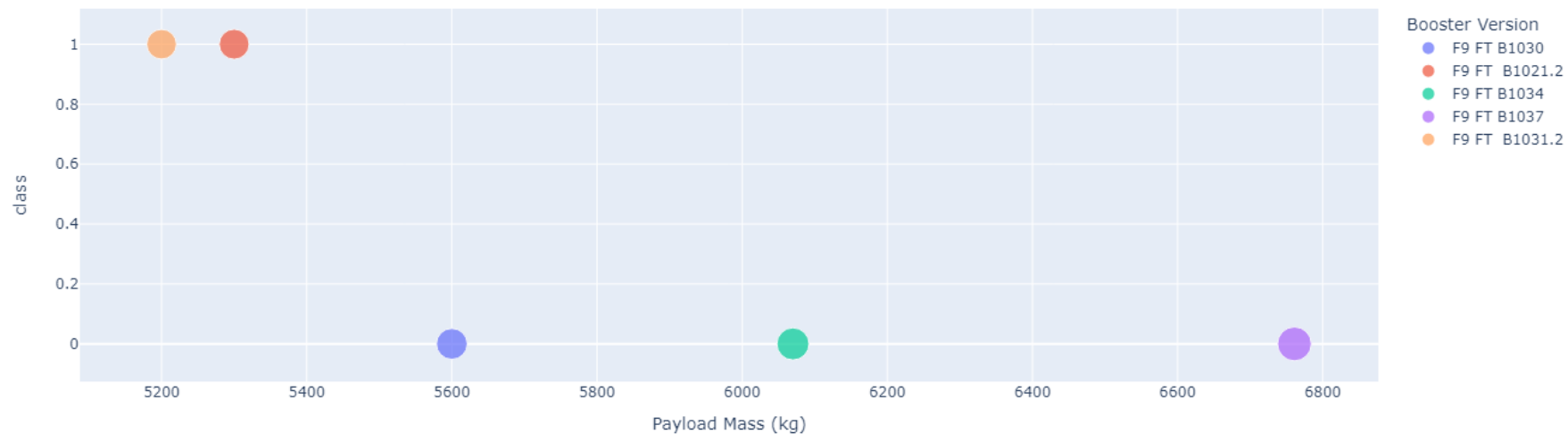
Total Success Launches for site KSC LC-39A



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



Low payload
mass 0-5000 kg



Low payload
mass 5000-
10000 kg

Section 6

Predictive Analysis (Classification)

Classification Model Accuracy

The decision tree classifier gives the highest 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)
```

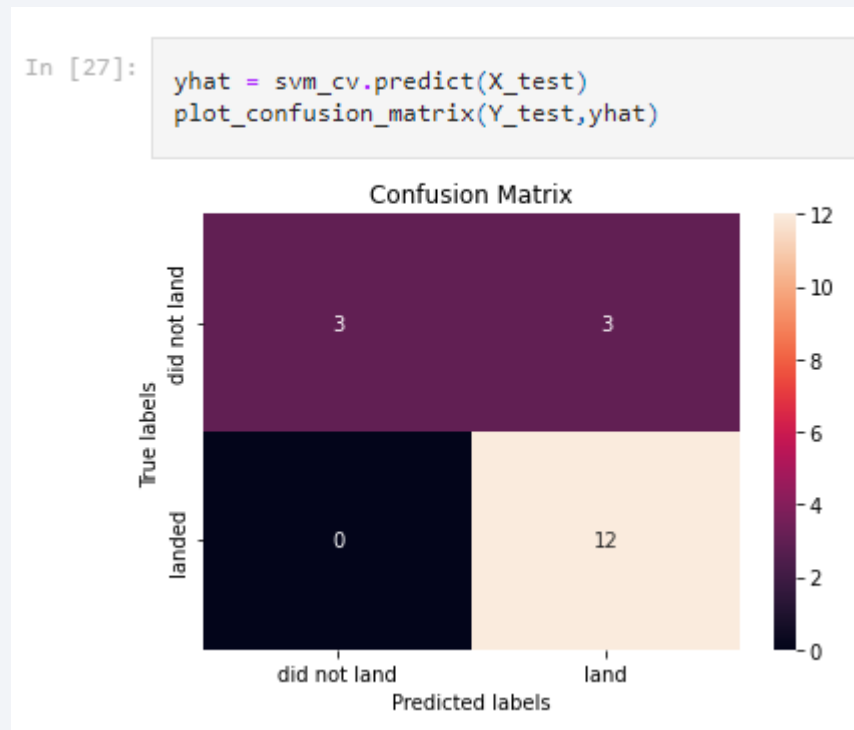
```
Out[21]: GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
                    param_grid={'criterion': ['gini', 'entropy'],
                                'max_depth': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                                'max_features': ['auto', 'sqrt'],
                                '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': 10,
'splitter': 'random'}
accuracy : 0.8892857142857142
```

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:

- Larger the amount of flight at a launch site, the greater the success rate.
- 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.
- The Decision tree classifier model turns out to be the best machine learning model for this problem statement.

Thank you!

