IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

<Vael Kokach>
<25/12/2023>

GitHub Repository

Outline

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

Executive Summary

- **Methodologies**
  The Project is aimed to accurately predict the successful landing of a SpaceX rocket (Falcon 9), using the features affecting the landing outcome as our target.
  Used Methodologies:
  - **Data Collection**: Data was collected using both web scraping and SpaceX API.
  - **Data wrangling**: Removed, encoded and created new features to enhance model training.
  - **Data analysis**: Analyzed the relation between total payload, launch site, etc.. with the success rate of the landing outcome.
  - **Data Exploration**: Learned that some launch sites had more success rate but lower payload.
  - **Visualization**: Visualized various launch sites with most success rate.
  - **Model building**: Utilized multiple ML algorithms to decide what is best for our case, algorithms such as Logistic Regression, SVM, Decision trees and KNNs.

Executive Summary

- **Results**
  - **Exploratory Data Analysis:**
    - Some Orbits showed better success rates than others, such as GEO, HEO and SSO.
      Launch success increased over time with different outcomes across all launch sites.
  - **Predictive Analytics:**
    - All models performed relatively the same at an average accuracy of 83%, However; the Decision tree model slightly outperformed with a 4% increase.

Introduction

- SpaceX launches Falcon 9 rockets at a cost of around $62m. This is considered cheap in comparison to other companies that doesn't re-use the first stage of their rocket and have fees of more than $165m.
- Since the successful landing of the first stage greatly affects the price, other companies can use this aspect to try and predict the outcome of the stage landing and bid against SpaceX for the rocket launch.

Section 1

Methodology

Methodology

- Data collection

  Using SpaceX API using GET library and web scraping using BeautifulSoup library.

- Data wrangling

  All columns and rows with NaN values were either removed or replaced with the mean of the same category.

  Creating a landing outcome feature where 1 is success and 0 is failure.

- Exploratory Data Analysis (EDA)

  SQL, Pandas and matplotlib were used to evaluate the dataset and visualize the relationships between variables.

- Interactive visual analytics

  Geospatial analytics were built using folium.

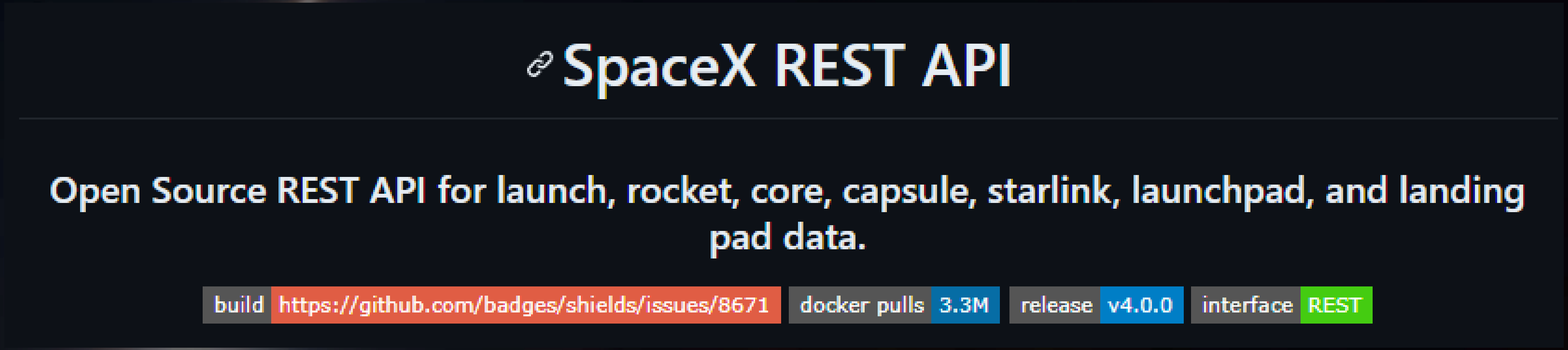
  Interactive dashboards were created using Dash.

- Predictive analysis using classification models

  Different classification models were trained and hyperparameters were tuned using GridSearchCV

Data Collection

- Using the SpaceX API to retrieve data about launches, including information about the rocket used, payload delivered, launch specifications, landing specifications, and landing outcome.



Data Collection – SpaceX API

Using the v4/launches/past endpoint of the SpaceX API to return all response dictionaries.

SpaceX API Github Notebook

```python
import requests
import json

def collect_spacex_launches():
  """
  Collects information about SpaceX launches from the API "api.spacexdata.com/v4/launches/past".

  Returns:
    A list of dictionaries containing information about each launch.
  """

  # Make a request to the API.
  response = requests.get("https://api.spacexdata.com/v4/launches/past")

  # Check if the request was successful.
  if response.status_code == 200:
    # Parse the JSON response.
    launches = json.loads(response.content)

    # Return the list of launches.
    return launches
  else:
    # Raise an exception if the request failed.
    raise Exception("Request to API failed.")
```

Data Collection – Scraping

Using the static URL from SpaceX Wikipedia page we pass the response object to the BeautifulSoup Method to get a soup Dataframe object.

[SpaceX Web Scraping Github Notebook]()

```
In [6]: static_url = "https://en.wikipedia.org/w/index.php?title=List_of_Falcon_9_and_Falcon_Heavy_launches&oldid=1027686922"
```

Next, request the HTML page from the above URL and get a `response` object

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 [9]: response = requests.get(static_url)
```

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

```
In [10]: soup = BeautifulSoup(response.content, 'html.parser')
```

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

```
In [12]: print(soup.title)

<title>List of Falcon 9 and Falcon Heavy launches - Wikipedia</title>
```

Out[58]:

	Flight No.	Launch site	Payload	Payload mass	Orbit	Customer	Launch outcome	Version Booster	Booster landing	Date	Time
0	1	F9 v1.080003.1	Dragon Spacecraft Qualification Unit	0	LEO	SpaceX	1	F9 v1.080003.1	Success\n	4 June 2010	18:45
1	1	F9 v1.080004.1	Dragon	0	LEO	NASA	1	F9 v1.080003.1	Success	4 June 2010	18:45
2	1	F9 v1.080005.1	Dragon	525 kg	LEO	NASA	1	F9 v1.080003.1	Success	4 June 2010	18:45
3	1	F9 v1.080006.1	SpaceX CRS-1	4,700 kg	LEO	NASA	Success\n	F9 v1.080003.1	Success\n	4 June 2010	18:45
4	0	F9 v1.080007.1	SpaceX CRS-2	4,877 kg	LEO	NASA	2	F9 v1.080003.1	Success\n	4 June 2010	18:45

Data Wrangling

Using Data Wrangling methods to separate different categories of the landing outcome feature and making them into 2 categories bad_outcome or good_outcome.

SpaceX Data Wrangling

```
for i,outcome in enumerate(landing_outcomes.keys()):
    print(i,outcome)
```

```
0 True ASDS
1 None None
2 True RTLS
3 False ASDS
4 True Ocean
5 False Ocean
6 None ASDS
7 False RTLS
```

```
# landing_class = 0 if bad_outcome
# landing_class = 1 otherwise
landing_class = []
for key, value in df['Outcome'].items():
    if value in bad_outcomes:
        landing_class.append(0)
    else:
        landing_class.append(1)
```

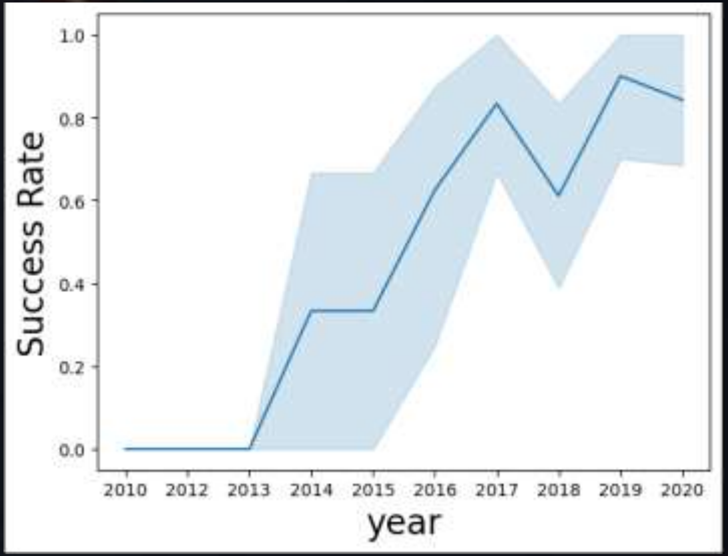
```
# landing_outcomes = values on Outcome column
landing_outcomes = df['Outcome'].value_counts()
landing_outcomes
```

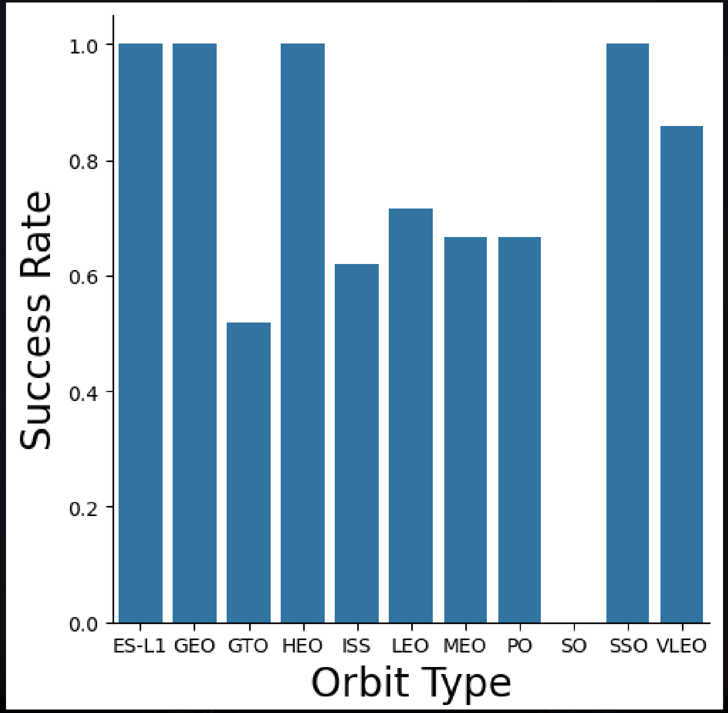
```
True ASDS      41
None None      19
True RTLS      14
False ASDS      6
True Ocean      5
False Ocean     2
None ASDS       2
False RTLS      1
Name: Outcome, dtype: int64
```

EDA with Data Visualization

Scatter plots to show relationship between numerical variables such as payload and launch site, Bar charts were used to analyze categorical variables such as Success Rate and Orbit Type and line charts were used to show the change of variables over time such as Success Rate and Year.

[SpaceX EDA & Data Visualization Github Notebook]()





EDA with SQL

[EDA With SQL GitHub Notebook]

To gather more information about the dataset, the following SQL queries were performed.

- Display the names of unique launch sites involved in space missions.
- Show five records where launch sites start with the prefix 'CCA.'
- Provide the total payload mass carried by NASA (CRS) boosters.
- Present the average payload mass carried by boosters of version F9 v1.1.
- List the date of the first successful landing on a ground pad.
- Identify boosters with successful drone ship landings and a payload mass between 4000 and 6000 kg.
- Display the total number of successful and failed mission outcomes.
- List booster versions that have carried the maximum payload mass.
- Provide details of failed landing outcomes on drone ships, including booster versions and launch site names for the year 2015.
- Rank the count of landing outcomes (e.g., Failure (drone ship) or Success (ground pad)) between June 4, 2010, and March 20, 2017, in descending order.

Interactive Map with Folium

EDA With Folium GitHub Notebook

The following steps were taken to visualize the launch data on an interactive map:

- Display all launch sites on a map by initializing a Folium Map object.
- Utilize folium.Circle and folium.Marker to add markers for each launch site on the map.
- Represent success or failure for each launch site on the map. Assign green for success (class = 1) and red for failure (class = 0) before clustering launches with identical coordinates. Employ folium.Marker within a MarkerCluster() object for clustering and use an icon with the specified marker color.
- Calculate distances between launch sites and their proximities. Use the Lat and Long values to mark points and create folium.Marker objects to indicate distances. For visualizing the distance line between two points, employ folium.PolyLine and add it to the map.

Dashboard with Plotly Dash

Dashboards GitHub

The following plots were added to a Plotly Dash dashboard to have an interactive visualization of the data:

- Generate a pie chart (px.pie()) to visualize the total number of successful launches per site. This chart provides a clear overview of the most successful launch sites. Additionally, implement a dcc.Dropdown() object for filtering to examine the success/failure ratio for individual sites.
- Create a scatter graph (px.scatter()) to depict the correlation between launch outcomes (success or failure) and payload mass (kg). Implement a RangeSlider() object to facilitate filtering based on payload mass ranges. Furthermore, provide the option to filter the scatter graph by booster version.

Predictive Analysis (Classification)

Predictive Analysis GitHub

The following steps were taken to develop, evaluate, and find the best performing classification model:

- Decide which type of machine learning algorithms are most appropriate
- For each chosen algorithm:
  - Create a GridSearchCV object and a dictionary of parameters
  - Fit the object to the parameters
  - Use the training data set to train the model

- For each chosen algorithm:
  - Using the output GridSearchCV object:
  - Check the tuned hyperparameters (best_params_)
  - Check the accuracy (score and best_score_)
  - Plot and examine the Confusion Matrix

Results

EXPLORATORY DATA ANALYSIS RESULTS

INTERACTIVE ANALYTICS DEMO IN SCREENSHOTS

PREDICTIVE ANALYSIS RESULTS

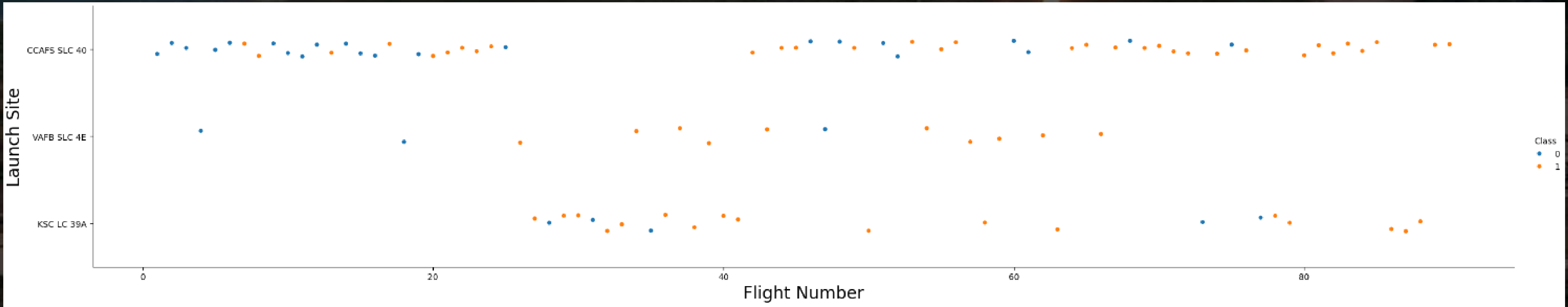
Section 2

Insights drawn from EDA

Flight Number vs. Launch Site

The correlation depicted in the scatter plot between Launch Site and Flight Number indicates the following trends:

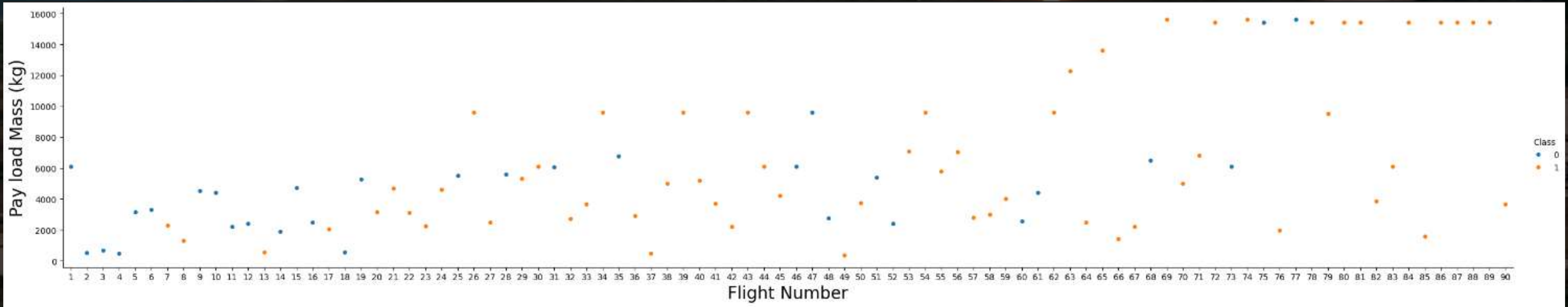
- As the flight number increases, there is a corresponding increase in the success rate at a given launch site.
- CCAFS SLC 40 was the primary launch site for early flights (flight numbers < 30), which generally experienced lower success rates.
- A similar pattern is observed at VAFB SLC 4E, where earlier flights were less successful.
- KSC LC 39A did not host any early flights, contributing to a higher overall success rate for launches from this site.



Payload vs. Launch Site

- The scatter plot depicting Launch Site vs. Payload Mass indicates the following observations:

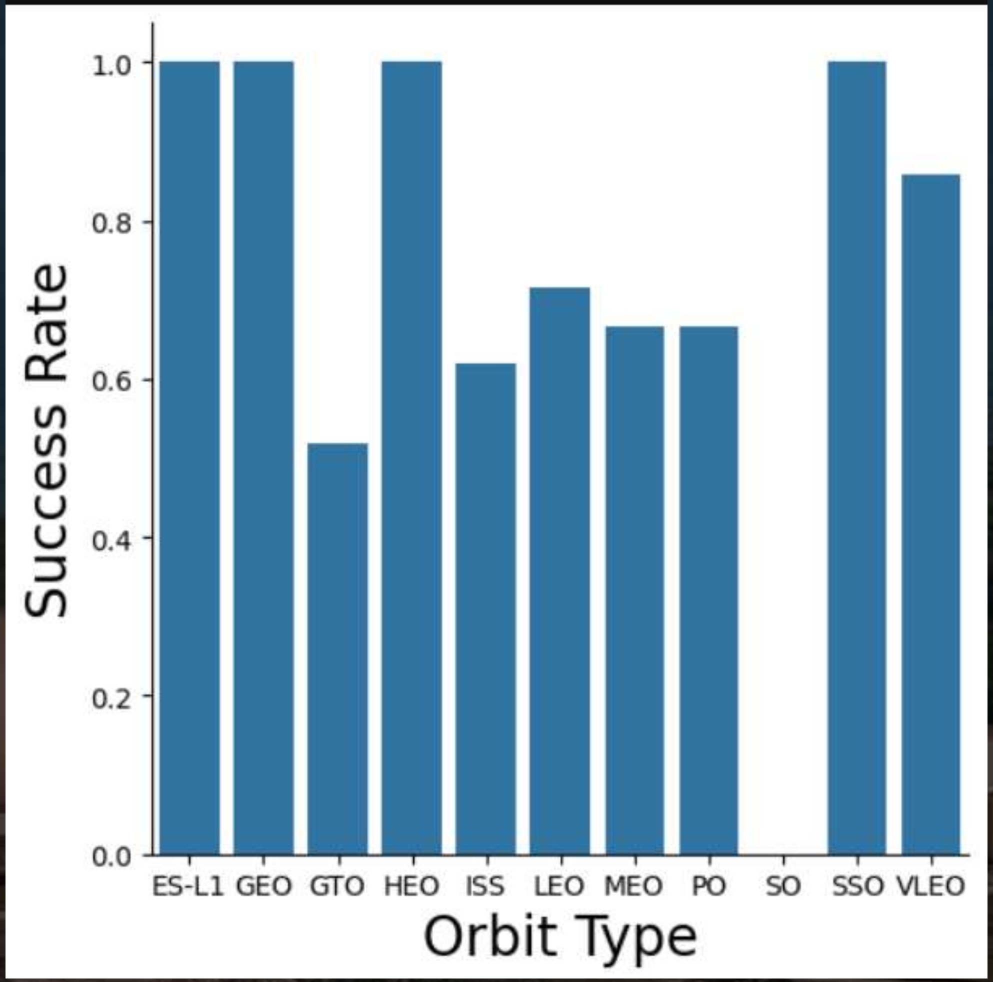
- Beyond a payload mass of approximately 7000 kg, there is a scarcity of unsuccessful landings; however, it's important to note that there is also a limited amount of data available for these heavier launches.
- No distinct correlation is evident between payload mass and the success rate for a specific launch site.



Success Rate vs. Orbit Type

- According to the bar chart illustrating Success Rate vs. Orbit Type, the orbits with the highest success rates, each achieving 100%, are as follows:

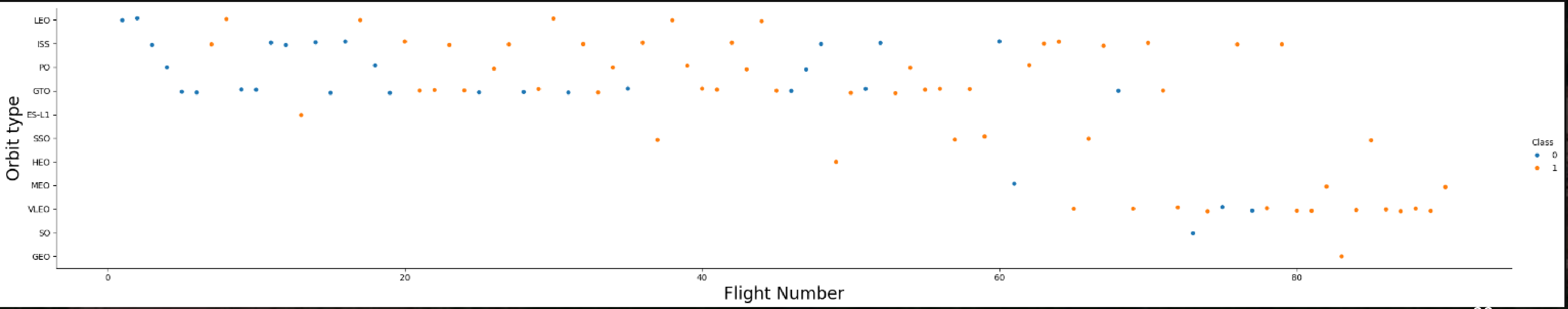
- ES-L1 (Earth-Sun First Lagrangian Point)
- GEO (Geostationary Orbit)
- HEO (High Earth Orbit)
- SSO (Sun-synchronous Orbit)



Flight Number vs. Orbit Type

- The scatter plot detailing Orbit Type vs. Flight Number provides additional insights that were not apparent in previous plots:

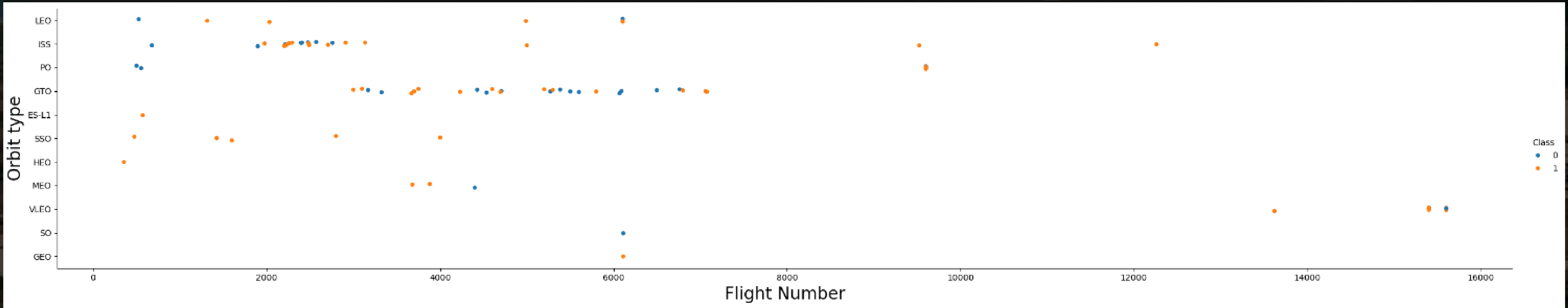
- The 100% success rates observed for GEO, HEO, and ES-L1 orbits can be attributed to the fact that there was only one flight into each of these respective orbits.
- The noteworthy 100% success rate for SSO is particularly impressive, given that there were five successful flights into this orbit.
- In general, there is an observable trend where, as Flight Number increases, the success rate also tends to increase.



Payload vs. Orbit Type

- The scatter plot depicting Orbit Type vs. Payload Mass reveals the following patterns:

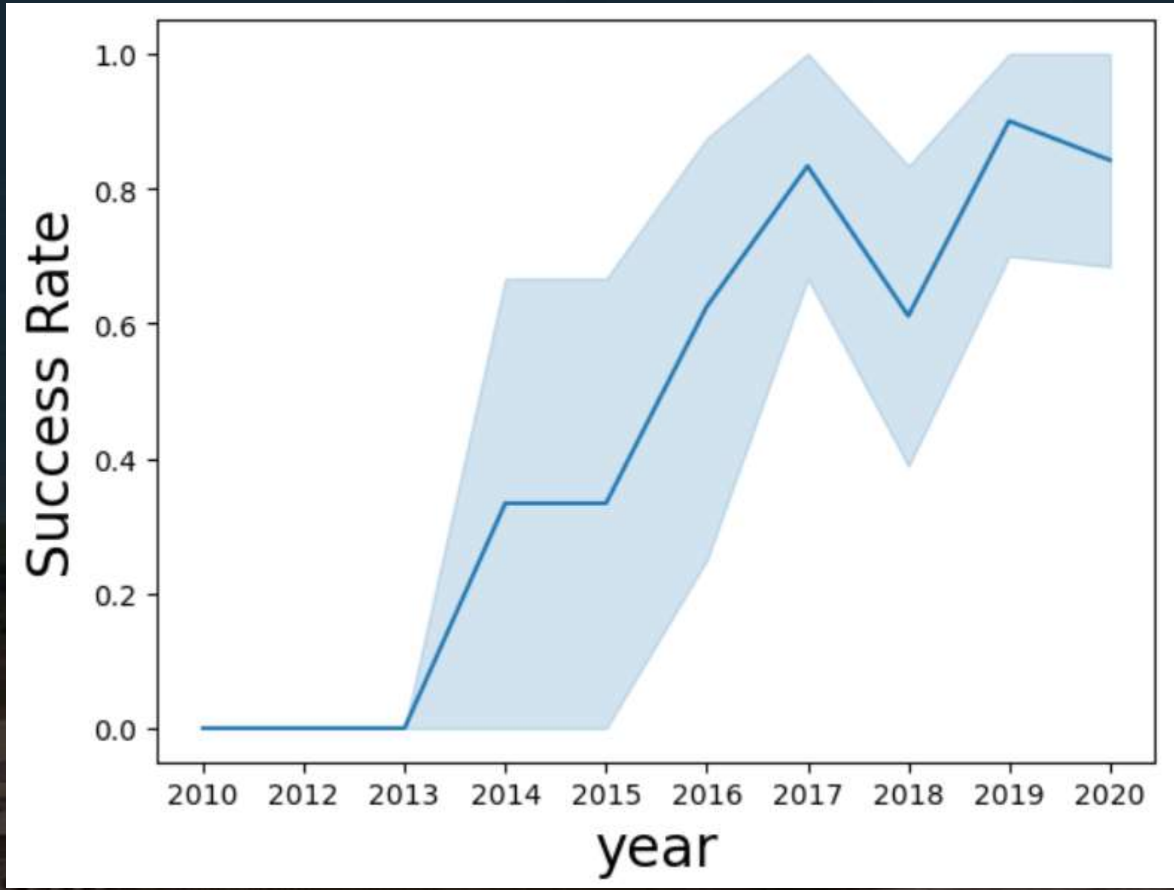
- Orbit types PO, ISS, and LEO exhibit a tendency to achieve higher success rates with heavier payloads, even though the data points for PO are limited.
- The relationship between payload mass and success rate for GTO remains unclear from the plot.



Launch Success Yearly Trend

- The line chart illustrating the yearly average success rate indicates the following trends:

- From 2010 to 2013, there were no successful landings, resulting in a 0% success rate during this period.
- Subsequently, post-2013, there was a general upward trend in success rates, with minor declines in 2018 and 2020.
- After 2016, the success rate consistently exceeded 50%, signifying a more favorable probability of success in later years.



All Launch Site Names

Returns only unique values from the LAUNCH_SITE column of the SPACEXTBL table.

In [16]:

```
%sql select distinct launch_site from SPACEXTBL;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[16]:

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

Launch Site Names Begin with 'CCA'

`LIMIT 5` fetches only 5 records, and the `LIKE` keyword is used with the wild card `‘CCA%’` to retrieve string values beginning with ‘CCA’.

```
In [20]: %sql select * from SPACEXTBL WHERE launch_site LIKE "CCA%" limit 5;

 * sqlite:///my_data1.db
Done.

Out[20]:
```

Date	Time (UTC)	Booster_Version	Launch_Site	Payload	PAYLOAD_MASS__KG_
2010-06-04	18:45:00	F9 v1.0 B0003	CCAFS LC-40	Dragon Spacecraft Qualification Unit	0
2010-12-08	15:43:00	F9 v1.0 B0004	CCAFS LC-40	Dragon demo flight C1, two CubeSats, barrel of Brouere cheese	0
2012-05-22	7:44:00	F9 v1.0 B0005	CCAFS LC-40	Dragon demo flight C2	525
2012-10-08	0:35:00	F9 v1.0 B0006	CCAFS LC-40	SpaceX CRS-1	500
2013-03-01	15:10:00	F9 v1.0 B0007	CCAFS LC-40	SpaceX CRS-2	677

Total Payload Mass

The SUM keyword is used to calculate the total of the LAUNCH column, and the SUM keyword (and the associated condition) filters the results to only boosters from NASA (CRS).

```
In [27]: %sql SELECT sum(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Customer LIKE "NASA %";

 * sqlite:///my_data1.db
Done.

Out[27]:
```

sum(PAYLOAD_MASS__KG_)
99980

Average Payload Mass by F9 v1.1

The AVG keyword is used to calculate the average of the PAYLOAD_MASS__KG_ column, and the WHERE keyword (and the associated condition) filters the results to only the F9 v1.1 booster version.

```
In [28]: %sql SELECT avg(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE Booster_Version = "F9 v1.1";

 * sqlite:///my_data1.db
Done.

Out[28]:
```

avg(PAYLOAD_MASS__KG_)
2928.4

First Successful Ground Landing Date

The MIN keyword is used to calculate the minimum of the DATE column, i.e. the first date, and the WHERE keyword (and the associated condition) filters the results to only the successful ground pad landings.

In [31]:

```
%sql SELECT min(Date) as First_Succesful_Landing FROM SPACEXTBL WHERE Landing_Outcome = "Success (ground pad)"
```

```
 * sqlite:///my_data1.db
Done.
```

Out[31]:

First_Succesful_Landing
2015-12-22

Successful Drone Ship Landing with Payload between 4000 and 6000

The WHERE keyword is used to filter the results to include only those that satisfy both conditions in the brackets (as the AND keyword is also used). The BETWEEN keyword allows for 4000 < x < 6000 values to be selected.

```
In [33]: %sql SELECT Booster_Version FROM SPACEXTBL WHERE Landing_Outcome = "Success (drone ship)" AND PAYLOAD_MASS__KG_ > 4000 AND

 * sqlite:///my_data1.db
Done.
```

Out[33]:

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

Total Number of Successful and Failure Mission Outcomes

The COUNT keyword is used to calculate the total number of mission outcomes, and the GROUPBY keyword is also used to group these results by the type of mission outcome.

```
In [35]: %sql SELECT Mission_Outcome, COUNT(*) FROM SPACEXTBL GROUP BY Mission_Outcome;

 * sqlite:///my_data1.db
Done.
```

Out[35]:

Mission_Outcome	COUNT(*)
Failure (in flight)	1
Success	98
Success	1
Success (payload status unclear)	1

Boosters Carried Maximum Payload

A subquery is used here. The SELECT statement within the brackets finds the maximum payload, and this value is used in the WHERE condition. The DISTINCT keyword is then used to retrieve only distinct /unique booster versions.

```
In [39]: %sql select booster_version from SPACEXTBL where payload_mass__kg_ = (select max(payload_mass__kg_) from SPACEXTBL);

 * sqlite:///my_data1.db
Done.
Out[39]:
```

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

The WHERE keyword is used to filter the results for only failed landing outcomes, AND only for the year of 2015.

In [50]:

```
%%sql SELECT substr(Date, 6,2) as month, Booster_Version, Launch_Site, Landing_Outcome from SPACEXTBL
    WHERE Landing_Outcome = 'Failure (drone ship)' and  substr(Date,0,5)='2015' ;
```

```
 * sqlite:///my_data1.db
Done.
```

Out[50]:

month	Booster_Version	Launch_Site	Landing_Outcome
01	F9 v1.1 B1012	CCAFS LC-40	Failure (drone ship)
04	F9 v1.1 B1015	CCAFS LC-40	Failure (drone ship)

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

The WHERE keyword is used with the BETWEEN keyword to filter the results to dates only within those specified. The results are then grouped and ordered, using the keywords GROUP BY and ORDER BY, respectively, where DESC is used to specify the descending order.

```
In [55]: %%sql SELECT Landing_Outcome, COUNT(*) as OUTCOME_COUNT FROM SPACEXTBL
            WHERE Date between '2010-06-04' and '2017-03-20' GROUP BY Landing_Outcome ORDER BY COUNT(Landing_Outcome) DESC;

 * sqlite:///my_data1.db
Done.
Out[55]:
```

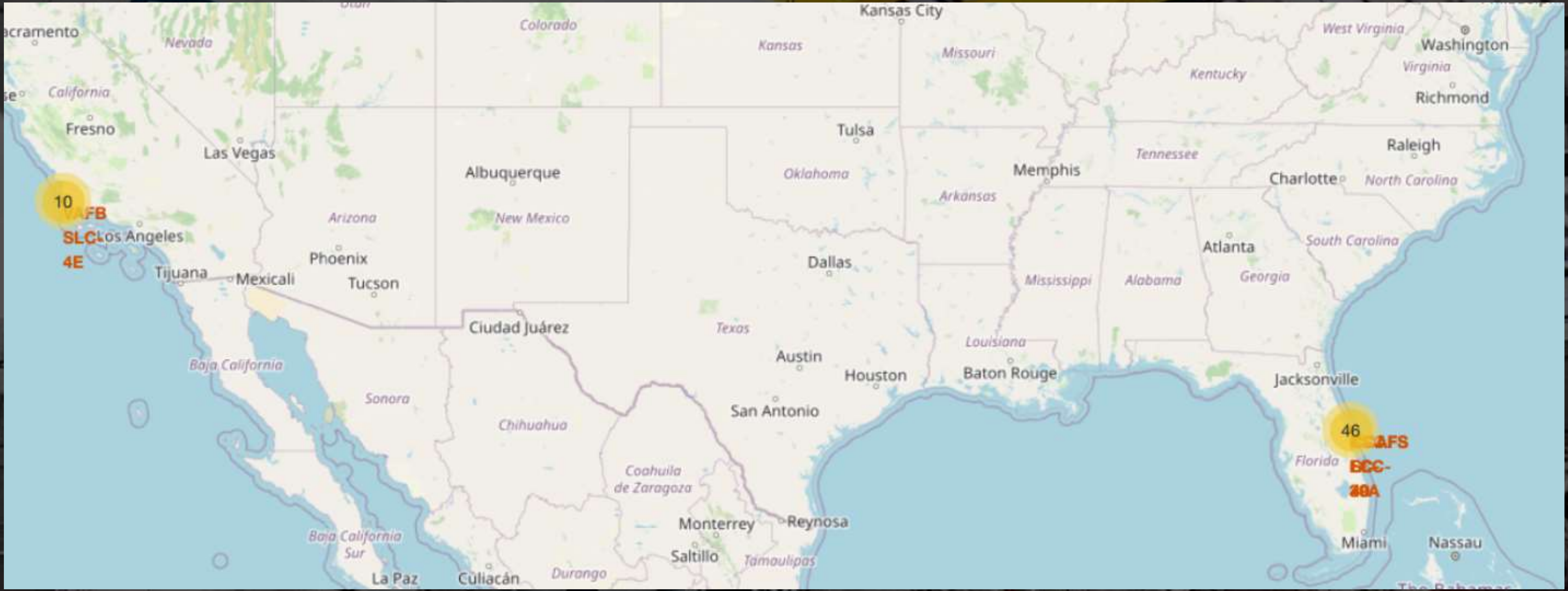
Landing_Outcome	OUTCOME_COUNT
No attempt	10
Success (drone ship)	5
Failure (drone ship)	5
Success (ground pad)	3
Controlled (ocean)	3
Uncontrolled (ocean)	2
Failure (parachute)	2
Precluded (drone ship)	1

Section 3

Launch Sites Proximities Analysis

ALL LAUNCH SITES ON A MAP

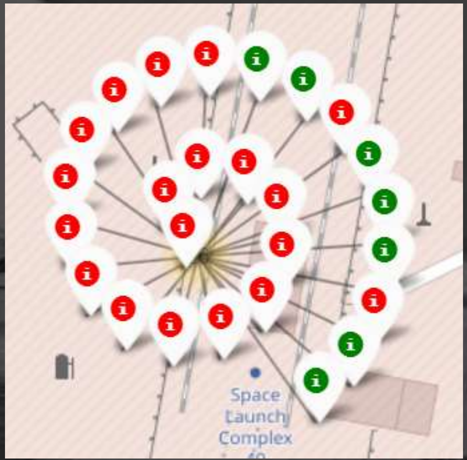
- All SpaceX launch sites are on coasts of the United States of America, specifically Florida and California.



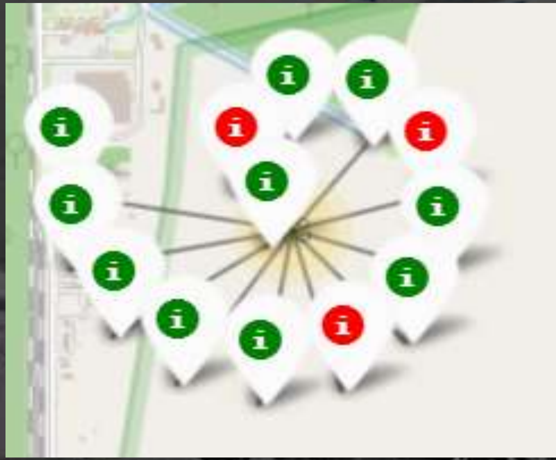
SUCCESS/FAILED LAUNCHES FOR EACH SITE

- Launches have been grouped into clusters, and annotated with green icons for successful launches, and red icons for failed launches.

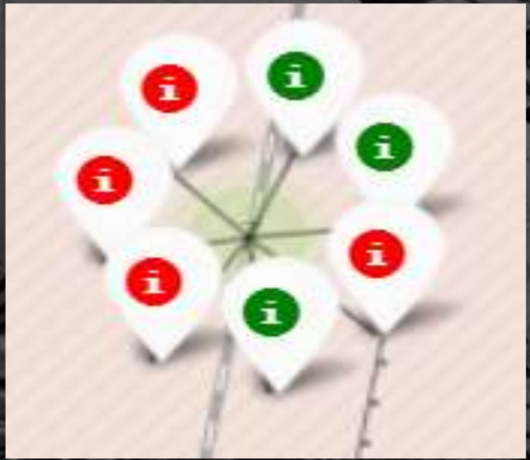
CCAFS SLC-40



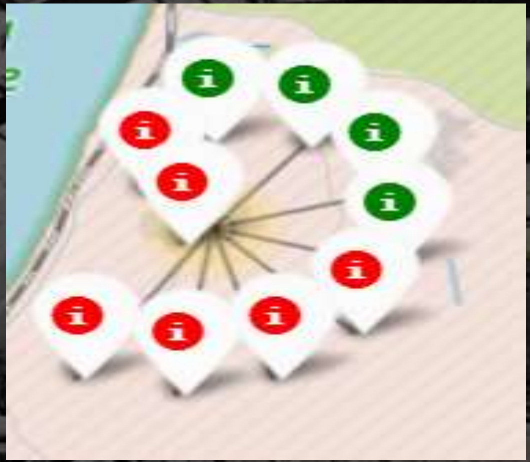
KSC LC-39A



CCAFS LC-40



VAFB SLC-4E



PROXIMITY OF LAUNCH SITES TO OTHER POINTS OF INTEREST

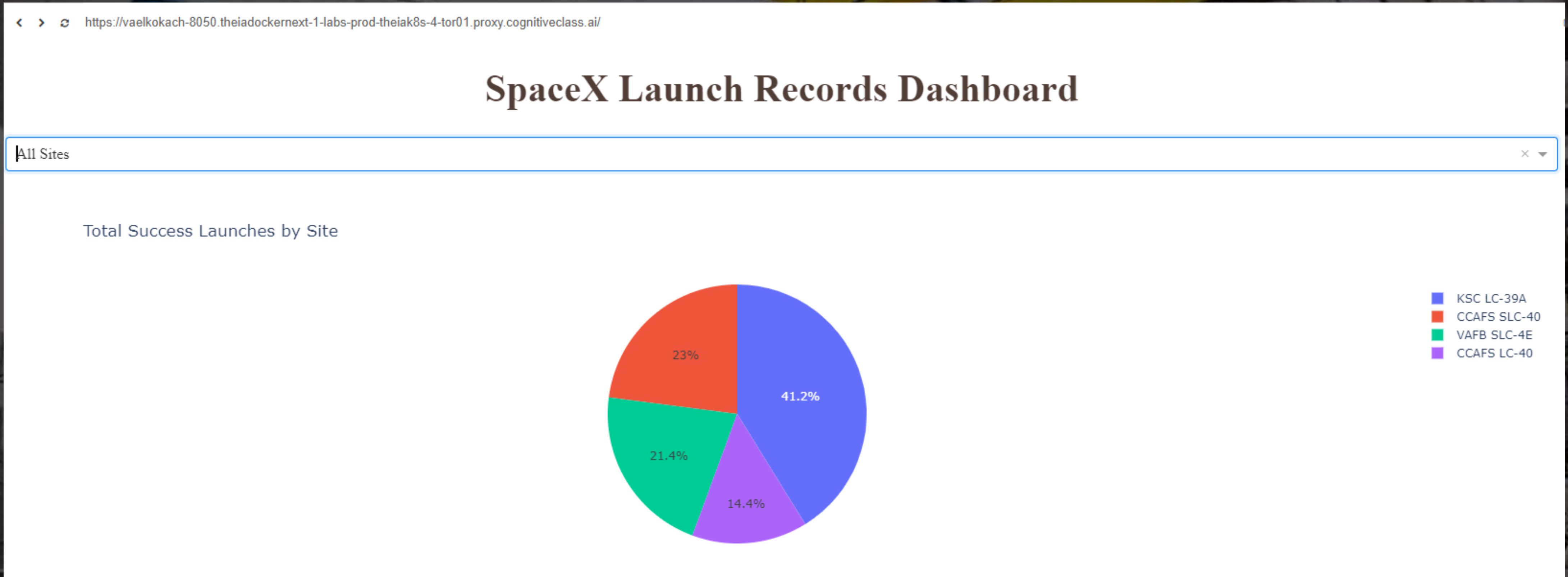
- Using the CCAFS SLC-40 launch site as an example site, we can see that the nearest coastline is 0.86 km due east, the nearest highway is 26.88 km away, the nearest railway is 21.96 km away and the nearest city is 23.23 km away.

Section 4

Build a Dashboard with Plotly Dash

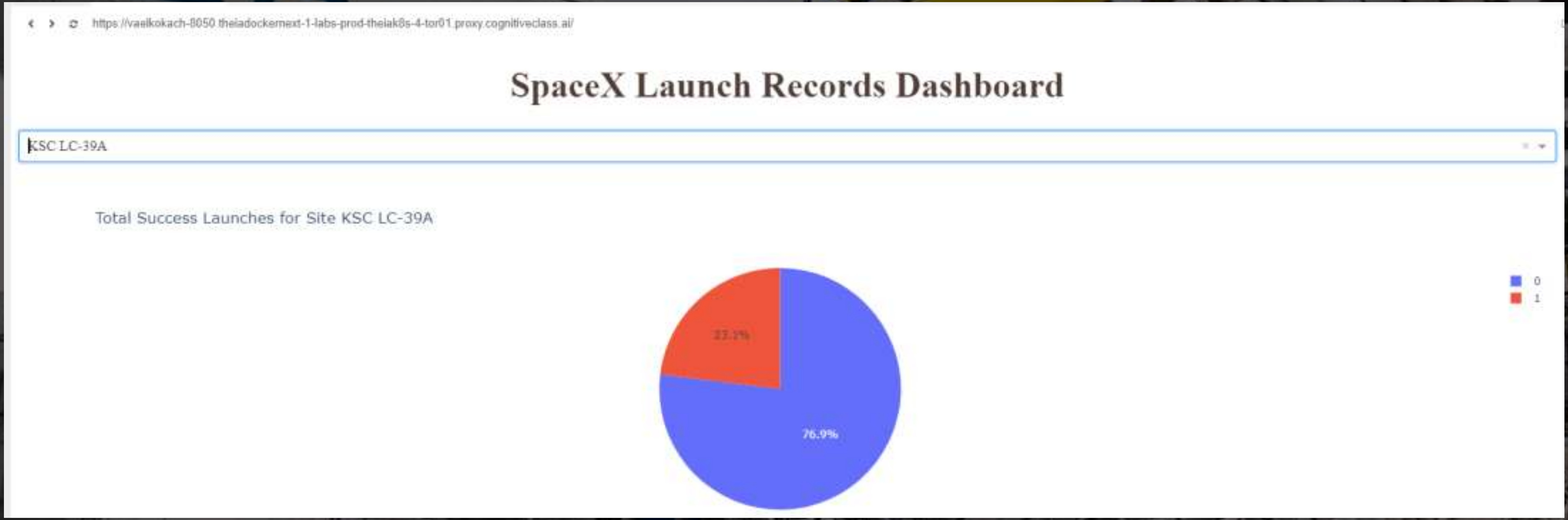
Launch Success Count For All Sites

- The launch site KSC LC-39 A had the most successful launches, with 41.7% of the total successful launches.



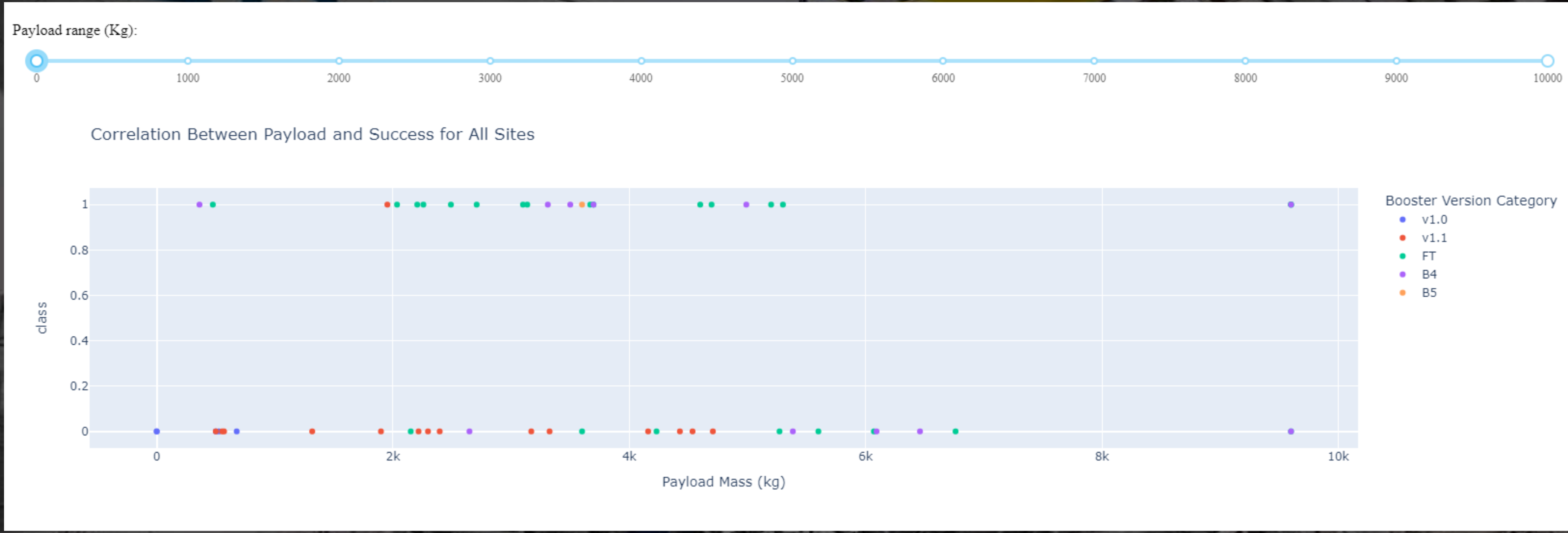
Pie chart for the launch site with highest launch success ratio

- The launch site KSC LC-39 A also had the highest rate of successful launches, with a 76.9% success rate.



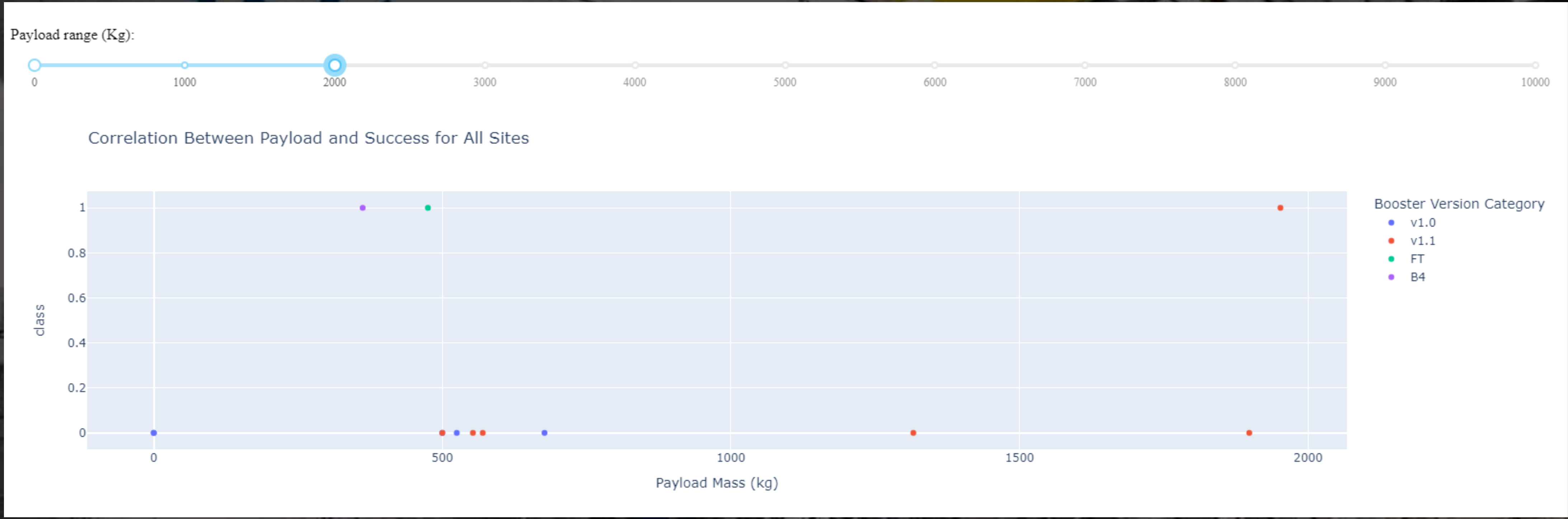
Launch Outcome VS. Payload scatter plot for all sites

- Plotting the launch outcome vs. payload for all sites shows almost three sections the first being between 0 – 2000, the second 2000 – 4000 and the third 4000 – 10000



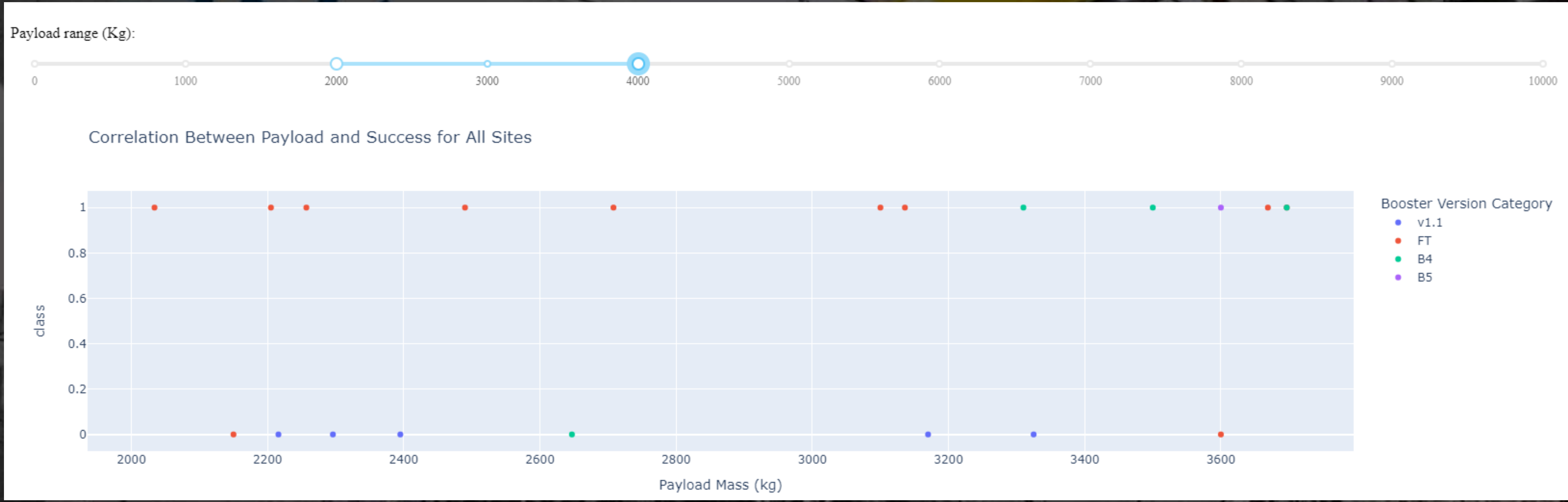
Launch Outcome VS. Payload scatter plot for all sites

- 0 – 2000 Payload mass scatter plot shows 5 failed launches in the range of 500 kg in comparison to 2 successful launches.



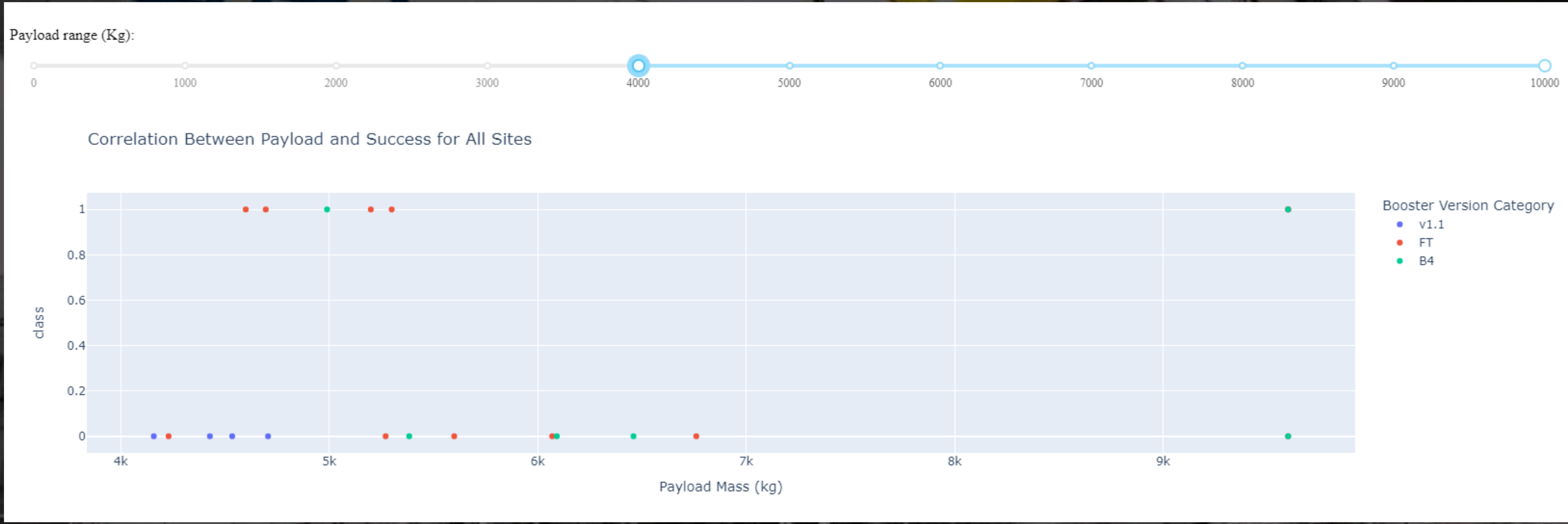
Launch Outcome VS. Payload scatter plot for all sites

- 2000 – 4000 Payload mass scatter plot shows a balance between success/fail launches across most booster categories but shows a significant success rate for FT boosters in this payload range.



Launch Outcome VS. Payload scatter plot for all sites

- 4000 – 10000 Payload mass scatter plot shows a lower success rate in the range of 5000 kg for most boosters except FT booster which performs well, and we notice 1 successful and 1 failed launch with a 10000 kg payload using the B4 booster.



Section 5

Predictive Analysis (Classification)

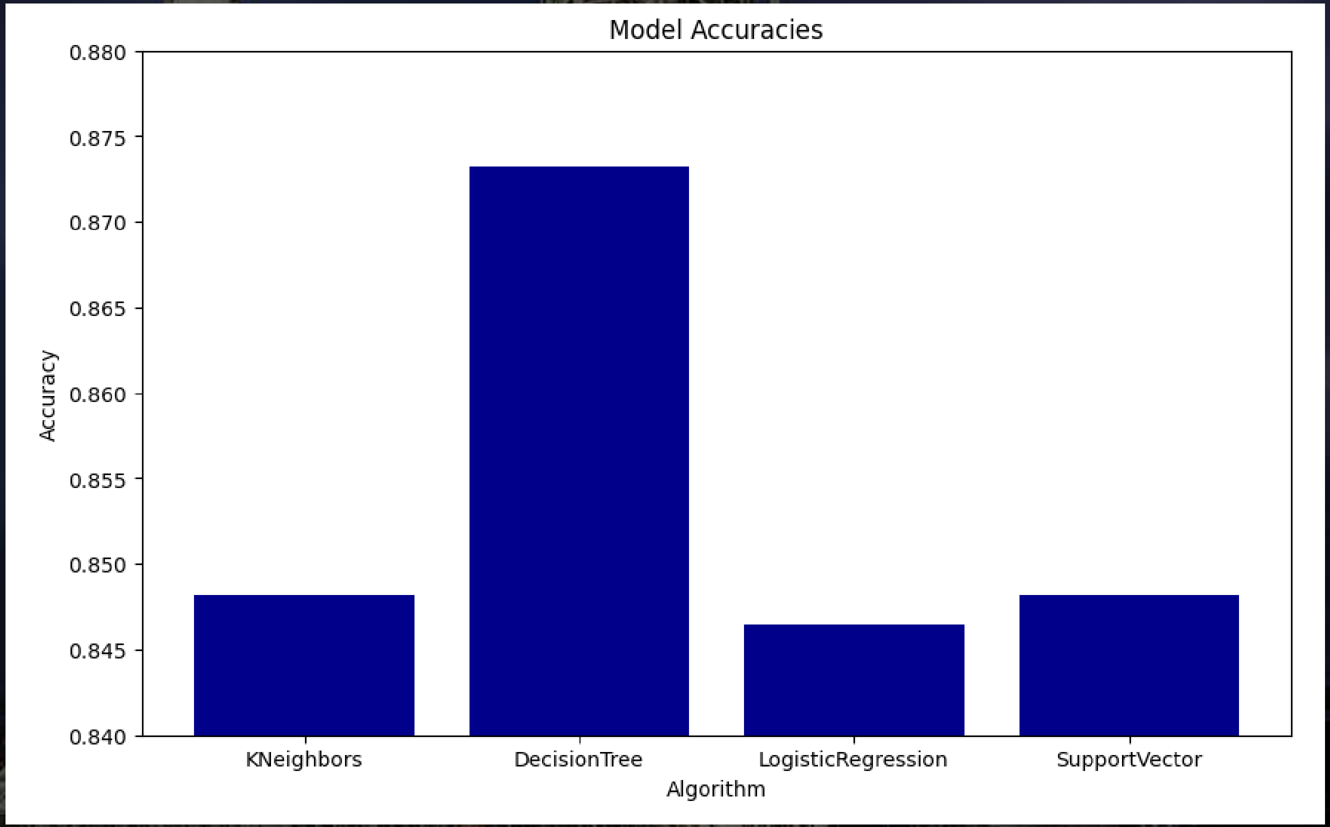
Classification Accuracy

- Plotting the Accuracy Score and Best Score for each classification algorithm produces the following result:
- The Decision Tree model has the highest classification accuracy
  - The Accuracy Score is 83.333%
  - The Best Score is 87.321%

	ML Method	Accuracy Score (%)
0	Support Vector Machine	83.333333
1	Logistic Regression	83.333333
2	K Nearest Neighbour	83.333333
3	Decision Tree	83.333333

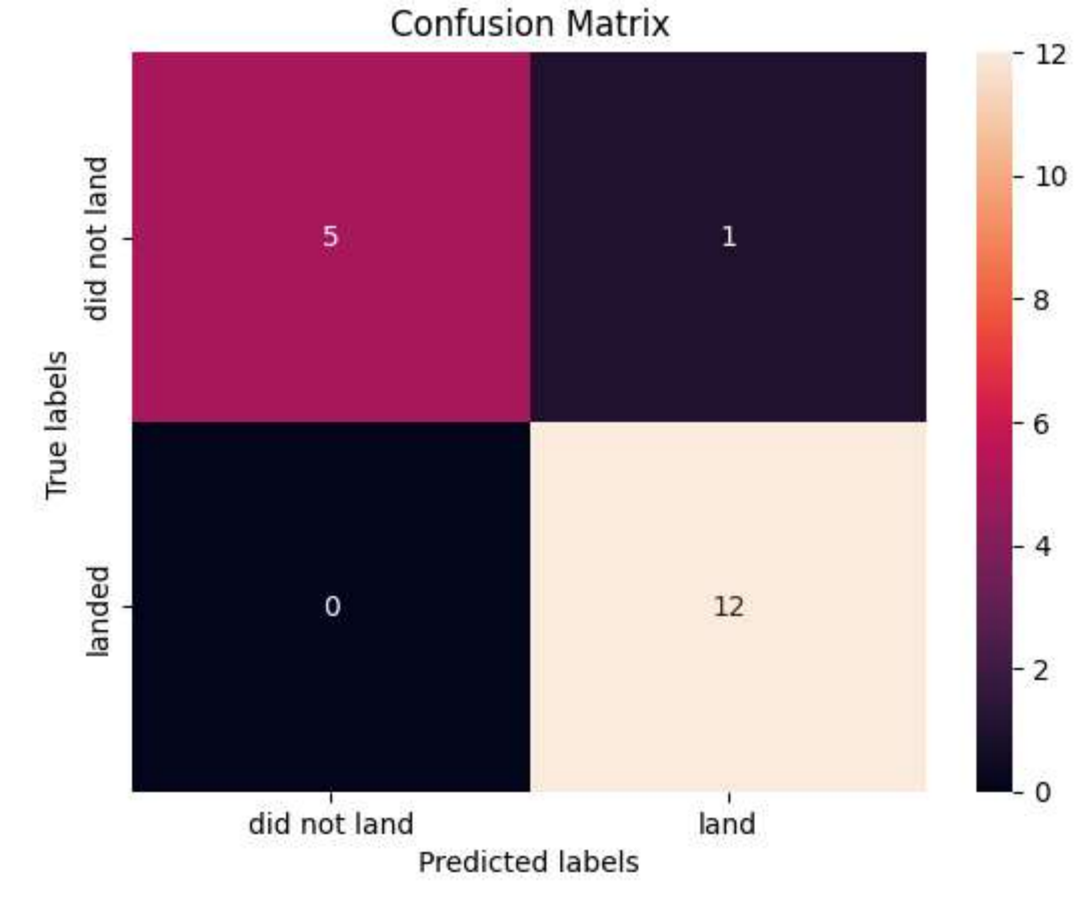
```
models
```

```
{'KNeighbors': 0.8482142857142858,
 'DecisionTree': 0.8732142857142857,
 'LogisticRegression': 0.8464285714285713,
 'SupportVector': 0.8482142857142856}
```



Best Model Confusion Matrix

- As shown previously, best performing classification model is the Decision Tree model, with an accuracy of 87.321%.
- This is explained by the confusion matrix, which shows only 1 out of 18 total results classified incorrectly (a false positive, shown in the top-right corner).
- The other 17 results are correctly classified (5 did not land, 12 did land).



Conclusion

- The likelihood of a successful space launch improves as more flights are conducted from a given site, indicating that increased experience leads to better outcomes. Initially, most launches were not successful. However, starting from 2010 through 2013, the success rate was zero, indicating complete failure in all attempts during that period. Post-2013 witnessed a general uptrend in launch success rates.

- Specific orbit categories, namely ES-L1, GEO, HEO, and SSO, boast perfect success records of 100%. This flawless performance for GEO, HEO, and ES-L1 is less remarkable due to it being based on a single launch each. In contrast, the SSO's 100% record is notable due to it being maintained across five launches. Orbits classified as PO, ISS, and LEO are more likely to achieve success when carrying heavier payloads. It is logical to find that VLEO missions, which operate in a very low Earth orbit, commonly carry heavier payloads.

- KSC LC-39 A emerges as the premier launch site, contributing to 41.7% of all successful space missions and achieving the highest individual success rate of 76.9%.

- Payloads that exceed 4000 kilograms have a reduced chance of launch success compared to their lighter counterparts.

- In the realm of classification models, the Decision Tree stands out as the top performer, boasting an impressive accuracy of 87.321%.

Appendix

Extracting html Headings into pandas Dataframe

```
headings = []
for key,values in dict(launch_dict).items():
    if key not in headings:
        headings.append(key)
    if values is None:
        del launch_dict[key]

def pad_dict_list(dict_list, padel):
    lmax = 0
    for lname in dict_list.keys():
        lmax = max(lmax, len(dict_list[lname]))
    for lname in dict_list.keys():
        ll = len(dict_list[lname])
        if  ll < lmax:
            dict_list[lname] += [padel] * (lmax - ll)
    return dict_list

pad_dict_list(launch_dict,0)

df = pd.DataFrame(launch_dict)
df.head()
```

Best Algorithm Parameters

```
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_)
```

Thank you!